INTERNET OF THINGS

Architecture and Design Principles

About the Author

Dr. Raj Kamal is presently, Professor of Computer Science and Engineering at Medi-Caps Institute of Science and Technology, Indore. He has worked with reputed institutions like Devi Ahilya Vishwavidyalaya, Indore; Punjabi University, Patiala; Kalasalingam University, Krishnankoil; and Guru Nanak Engineering College, Hyderabad. He completed his M.Sc. at the age of 17 and published his first research paper in Physics Letters Volume journal at the age of 18. He was awarded a Ph.D. degree by the prestigious Indian Institute of Technology, Delhi, at the age of 22. He completed his post-doctoral studies Uppsala University in Sweden.

Professor Raj Kamal is immensely popular among students and academicians. He is lovingly addressed by his colleagues as a 'learning machine' or 'human dynamo'. He is known in the country for his constant drive for understanding emerging technologies and passion for acquiring latest knowledge and its dissemination. His areas of interest include Mobile Computing, Embedded Systems, and Cloud Computing and Analytics.

Professor Raj Kamal has over 49 years of teaching and research experience. He has successfully guided 17 Ph.D. students and has published around 150 research papers in various national and international conferences and journals that include IEEE Transactions on Industrial Electronics. He has authored 10 textbooks in areas such as Computer Science, Electronics and Communication, and Information Technology.

Professor Raj Kamal has also authored the following titles:

- *Embedded Systems—Architecture, Programming and Design*, 3/e
- *Computer Architecture*, 2/e
- *Internet and Web Technologies*

INTERNET OF THINGS

Architecture and Design Principles

Raj Kamal
Professor, Computer Science and Engineering
Medi-Caps University
Rau, Indore, Madhya Pradesh, India

McGraw Hill Education (India) Private Limited
CHENNAI

McGraw Hill Education Offices
Chennai New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto

McGraw Hill Education (India) Private Limited

Published by McGraw Hill Education (India) Private Limited
444/1, Sri Ekambara Naicker Industrial Estate, Alapakkam, Porur, Chennai - 600 116

Internet of Things: Architecture and Design Principles

Print Edition
ISBN-13: 978-93-5260-522-4
ISBN-10: 93-5260-522-5

E-Book
ISBN-13: 978-93-5260-523-1
ISBN-10: 93-5260-523-3

Managing Director: *Kaushik Bellani*

Director—Science & Engineering Portfolio: *Vibha Mahajan*
Lead—Science & Engineering Portfolio: *Hemant Jha*
Specialist—Product Development: *Mohammad Salman Khurshid*

Content Development Lead: *Shalini Jha*
Specialist—Custom Products: *Ranjana Chaube*

Production Head: *Satinder S Baveja*
Assistant Manager—Production: *Jagriti Kundu*

General Manager—Production: *Rajender P Ghansela*
Manager—Production: *Reji Kumar*

Typeset at The Composers, 260, C.A. Apt., Paschim Vihar, New Delhi 110 063 and printed at

Cover Printer:

Preface

Overview

What if an umbrella could sense the local weather and remind the user if it is needed to be carried along that day; or if some kind of wearable device could monitor a patient's health and when it is about to deteriorate, communicate the nature of emergency to the doctor directly and promptly; or if a car could have some computation and predictive analytics system which could apprise the user about the upcoming servicing schedules to avoid any sudden component failures beforehand?

Internet of Things (IoT) and Internet-connected Cloud Platform-as-a-Service (PaaS) can make the above cases come true as real-life situations. In this book, the author takes the readers through all the important design and implementation details of various functions possible with IoT. For instance, the first chapter of the book introduces the concept of a smart umbrella which can, using specific sensors and computational devices and connectivity to the Internet, share weather updates with its users. A chapter on smart cars or connected cars elucidates how such cars can perform automatic detection of service requirements by direct transfer of service-relevant data to the car maintenance and service center or to the driver/car user server, and also send automatic reminders of servicing appointments, thereby lessening service visits.

IoT is now widely researched and being rapidly implemented as well. A TCS study, 'Internet of Things: The Complete Re-Imaginative Force', 2015, quotes technology researcher Gartner Projection as,

> *'There will be 4.9 billion 'connected things' (or 'smart-connected products', as Harvard Business School Professor Michael Porter refers to them) this year, and we haven't seen anything yet. The number is predicted to grow five times by the end of the decade, to 25 billion*

connected things, including a quarter billion vehicles. Gartner says, by 2020, a quarter billion connected vehicles will enable new in-vehicle services and automated driving capabilities.'[1]

Target Audience

Internet of Things design and IoT products, services and applications development need a team of hardware as well as software professionals. These professionals in turn need a reserve to make them aware of latest in this field. Thus, this book would suffice the needs of these individuals and also of undergraduate and postgraduate students of engineering, computer science and information technology.

About the Book

The book has been written in an easy-to-understand and student-friendly manner, and includes several illustrative figures and examples, sample codes and project case studies. The text explains architecture, design principles, hardware and software designs, while keeping multidisciplinary under-graduates and post-graduates in mind as primary readers. Each chapter begins by defining the learning objectives for each section, recall from previous chapters, introduction and the important terms covered in that chapter. End of each section enlists the points understood and provides self-assessment questions, classified as per three difficulty levels. Additionally, key concepts, learning outcomes, objective questions, short-answer questions, review questions and practice exercises are provided to revise and apply concepts learned from the chapter.

Learning and Assessment Tools

Learning Objectives

Learning Objectives (LO) have been drafted for content in this book. This is an educational process that emphasizes on developing required skills amongst students and tests the outcomes of the study of a course, as opposed to routine learning. This approach creates an ability to acquire knowledge and apply fundamental principles to analytical problems and applications.

Self-assessment Exercises

Each learning objective is followed by a set of questions for self-assessment of students. This offers great retention through looping mechanism.

Pedagogical Classification

The pedagogy is arranged as per levels of difficulty. All checkpoint problems are linked with Learning Objectives (LOs) and marked with Levels of Difficulty (LOD), to help assess

[1]http://www.gartner.com/newsroom/id/2970017 (January 2015).

the student's learning. This assessment of levels of difficulty is derived as per Bloom's Taxonomy.

- ★ Indicates Level 1 and Level 2, i.e. knowledge and comprehension based easy-to-solve problems.
- ★★ Indicates Level 3 and Level 4, i.e. application and analysis based medium-to-difficult problems.
- ★★★ Indicates Level 5 and Level 6, i.e. synthesis and evaluation based very difficult problems.

Learning Outcomes

Summary points specific to each LO are provided at the end of each chapter. It helps in recapitulating the ideas initiated with the outcomes achieved.

Chapter-end Exercises

More than 500 carefully designed chapter-end exercises are arranged as per levels of difficulty and are constructed to enhance knowledge and test technical skills. These include objective questions, short-answer questions, review problems and practice problems.

Salient Features

- New IoT/M2M architecture, reference models, standards and protocols covered in detail
- Latest topics such as *Industrial IoT*, *Industry 4.0*, *Participative Sensing*, *Connected Car* form a part of this edition
- Explains architecture, design principles, hardware and software designs and applications/services/processes
- Additional online material for students and instructors shared via web resources.
- Rich pedagogy
  - Illustrations: 101
  - Solved examples: 88
  - Case studies: 4
  - Self-assessment questions: 422
  - Objective questions: 133
  - Short-answer questions: 172
  - Review questions: 129
  - Practice questions: 118

Chapter Organisation

Chapter 1 gives an overview of 'Internet of Things'. It first gives vision and definitions of IoT, and meaning of smart hyperconnected devices which enable the IoT applications/services. Next, the chapter describes IoT conceptual framework and architectural views, technology behind IoTs, communication modules and protocols such as MQTT. Then the chapter describes the sources of IoTs, such as RFIDs and wireless sensor networks.

Machine-to-machine communication technologies now enhanced to IoTs. The chapter also introduces the concept of wearbale watches, smart home and smart cities.

Chapter 2 describes the design principles for connected devices. It describes IETF six-layered design for IoT applications, ITU-T reference model and ETSI M2M domains and high-level capabilities. It then describes first architectural layer/device and gateway domain wireless and wired communication protocols and technologies. It then describes second architectural layer/device and gateway domain functionalities which are data enrichment, transcoding, consolidation, privacy issues, and device configuring, management, and ID management. The chapter also explains the need of ease of designing and affordability.

Chapter 3 describes the design principles for web connectivity, the data format standards JSON, TLV and MIME for communication, and devices CoAP, CoAP-SMS, CoAP-MQ, MQTT and XMPP protocols for the devices connectivity to the web. The chapter also describes SOAP, REST, HTTP RESTful and WebSockets methods, which the communication gateway deploys.

Chapter 4 describes Internet connectivity principles. Concepts such as IPv4, IPv6, 6LowPAN, TCP/IP suite of protocols; IP addressing of the IoT devices and MAC address of communication circuits are covered. HTTP, HTTPS, FTP, Telnet and other protocols used at IETF layer 6, application layer by IoT applications/services/processes have also been discussed.

Chapter 5 defines the methods of data acquiring, organising and analytics in IoT/M2M applications/services/business processes. The chapter covers data generation, acquisition, validation, data and events assembly and data store processes.

Data centre and server management functions have also been described along with organisation of data as database, spatial and time series database, SQL and NOSQL methods, and concepts of queries processing, transactions and events processing, OLTP, business processes, business intelligence and Internet-enabled embedded devices and their network protocols.

Most important conceptual aspects such as distributed business processes, complex applications integration and service-oriented architecture, integration and enterprise systems, IoT/M2M descriptive, events, real-time, prescriptive and predictive analytics using databases and big data form the highlights of this chapter. The chapter towards the end describes concepts of knowledge acquiring, managing and storing.

Chapter 6 describes the concepts of data collection, storage and computing using a cloud platform for IoT/M2M applications/services. The chapter describes cloud computing paradigm, cloud deployment SaaS, IaaS, Paas, DaaS models, Everything-as-a-Service and cloud service models. The chapter further explains the methods of cloud platforms for device collection, device data store and computing at the cloud. The usage of two cloud-based services, viz., Xively (Pachube/COSM) and Nimbits have been considered as examples.

Chapter 7 describes sensors, their examples, working principles and usage technologies. A highlight of this chapter is description of latest topics of participatory sensing

and Industrial IoT. It describes automobile IoT and includes how IoT Innovations are reshaping future automobiles and usages of Internet connectivity of cars, and Vehicle-to-Infrastructure (V2I) technology. The chapter also describes usages of actuators, RFID and wireless sensor network technology in detail.

Chapter 8 describes the prototyping of the embedded devices for IoTs. The chapter explains the embedded computing basics and embedded devices hardware and software for embedded computing. It also describes the features of Arduino, Intel Galileo, Edison, Raspberry Pi, BeagleBone and mBed Boards which are the popular embedded platforms for prototyping. Usages of mobiles and tablets in IoT Apps and things always connected to the Internet/cloud have also been covered.

Chapter 9 describes prototyping of devices embedded device software, gateways, Internet and web/cloud services and software components. The chapter describes prototyping of programs for the embedded devices using popular Arduino platform IDE. The chapter describes programming the embedded Galileo, Raspberry Pi, BeagleBone and mBed device platforms for IoT. The chapter also explains how to prototype online-component APIs and web APIs required in apps, web apps and web services, developed using IoT devices data store, databases and analytics.

Chapter 10 covers the most important issues in uses of IoT applications and connected devices need data privacy, security and vulnerabilities solutions. It describes vulnerabilities, security requirements and threat analysis and importance of use cases as well as misuse cases for provisioning of right security for IoT. The chapter then describes IoT security tomography and layered attacker models and connected sources identity management, identity establishment and access control.

Chapter 11 provides insight with business models and business model innovation. The chapter explains concept of value creation using IoTs. The chapter introduces Industry 4.0 model and familiarises with business model scenarios for usages IoT devices, mobile APIs and customer data.

Chapter 12 first explains the designing levels (stages) during an IoT prototype and product development and introduces six complexity levels in designing with examples of IoT systems of each complexity level. The chapter covers the uses of connected platform-as-a-Service (PaaS) cloud with examples of AWS IoT platform and TCS Connected Universe Platform that accelerates the design, development and deployment of the M2M, IoT, IIoT applications and services. The chapter further outlines the applications of PaaS platforms for IoT/IIoT in globally trending usages in four core business categories/areas. A new viz. concept, connected car, with the example of Tesla and its usages in the automobiles of future has also been covered in this chapter. The chapter familiarises the reader with approaches to IoT applications for smart homes, smart cities, smart environment monitoring, smart agriculture and smart production using illustrative examples. The chapter then explains a new IoT project design case study, 'Smart city streetlights control and monitoring'.

Web Resources

The web has become the best companion of teachers and students, and is needed much like food and water. The web supplements for the book are available at http://www.mhhe.com/rajkamal/iot. The web link has the following material which will be periodically updated:

For Instructors

- Solution guide to Short answer questions, Review Questions and Practice Exercises
- Chapter-wise PowerPoint slides with diagrams and notes for effective presentation

For Students

- New case studies
- Answers to queries raised by students through author's website: http://www.rajkamal.org

Acknowledgements

The author is grateful to Shri R.C. Mittal, Chancellor; Shri Gopal Agrawal, Pro-Chancellor; Dr. Sunil K Somani, Vice Chancellor; and Dr. Shamsher Singh, ex-Chief Executive Director, Medi-Caps University for providing a great platform to learn and continue research and teaching activities in new exciting areas of computer science and engineering.

The author is also grateful to Prof. Sanjay K. Tanwani, Prof. Maya Ingle, Dr. Preeti Saxena, Dr. Shraddha Masih, Prof. Abhay Kumar, Dr. Manju Chattopadhaya and other colleagues at Devi Ahilya Vishwavidyalaya, Indore, for continuous encouragement.

Thanks are due to Sushil Mittal (wife) for love and affection and being a constant source of support during this endeavour, and family members Dr. Shilpi Kondaskar (daughter), Dr. Atul Kondaskar (son-in-law), Shalin Mittal (son), Needhi Mittal (daughter-in-law) and Arushi Kondaskar, Atharv Raj Mittal, Shruti Shreya Mittal and Ishita Kondaskar (grandchildren) for their love and affection.

The author would also like to thank the team at McGraw Hill Education for bringing out this publication.

Invitation for Feedback

The author expects that students and professionals will love this book and it will help the students hone their design skills and key concepts in the domain of Internet of Things—architecture, designing principles and applications.

Although much care has been taken to ensure an error-free text, some errors may have crept in. The author shall be grateful to the readers for pointing out these errors to him. The feedback on content of the book as well as supplementary web material will be highly appreciated. The author can be contacted at professor@rajkamal.org or dr_rajkamal@hotmail.com.

Raj Kamal

Publisher's Note

McGraw Hill Education (India) invites suggestions and comments, all of which can be sent to info.india@mheducation.com (kindly mention the title and author name in the subject line). Piracy-related issues may also be reported here.

Contents

List of Abbreviations and Acronyms

3GPP	3G Partnership Project Protocols
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ACID	Atomicity, Consistency, Isolation and Durability of transactions
ADAS	Advance Driver Assistance Systems
ADC	Analog-to-Digital Converter
ADFG	Acrylic Data Flow Graph
ADSL	Asymmetric Digital Subscriber Line
AES	Advanced Encryption 128- or 192- or 256- bit key length Algorithm
AES-CCM	AES with CCM
API	Application Programming Interface
AQI	Air Quality Index
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
ATM	Automated Teller Machine
AWS	Amazon Web Services
BB	BeagleBone
BI	Business Intelligence
BJT	Bipolar Junction Transistor

Bootpc	Bootstrap Protocol Client
Bootps	Bootstrap Protocol Server
BP	Business Process
BT BR	Bluetooth Basic data rate in 1.0, 2.0, 3.0 or 4.0 device
BT EDR	Bluetooth Enhanced Data Rate
BT LE	Bluetooth Low Energy
CA	Certification Authority
CAN	Controller Area Network bus
CAP	Consistency, Availability and Partitions
CBC	Cryptographic Block Cipher for block ciphers with a block length of 128 bits
CCD	Charge Coupled Device
CCM	Counter with CBC-MAC
CEP	Complex Event Processing
CGA	Cryptographically Generated Addresses
CIDR	Classless Inter-Domain Routing
CIMD	Computer Interface to Message Distribution
CoAP	Constrained Application Protocol
CORE	Constrained RESTful Environment
CRC	Cyclic Redundancy Check
CRM	Customer Relations Management
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CUP	Connected Universe Platform
CVS	Concurrent Versions System
CWI	Cloud Web Interface
DAG	Directed Acrylic Graph
DB	Database
DBMS	Database Management System
DBP	Distributed Business Process
DFG	Data Flow Graph
DHCP	Dynamic Host Control Protocol
DLL	Dynamically Linked Library
DM	Device Management
DNS	Domain Name System
DODAG	Destination Oriented Directed Acrylic Graph

DoS	Denial-of-Service
DSL	Digital Subscriber Line
DSP	Digital Signal Processor
DSSS	Direct Sequence Spread Spectrum
DTLS	Datagram Transport Layer Security
DWI	Device Web Interface
ECU	Electronic Control Unit
EPC	Electronic Product Code
ESP	Event Stream Processing
ETL	Extract, Transform and Load
FC	Functional Component
FG	Functional Group
FHSS	Frequency Hopping Spread Spectrum
FOTA	Firmware Over-The-Air
FPT	Phototransistor
FTP	File Transfer Protocol
GPIO	General Purpose Input-Output
GPRS	General Packet Radio Service
GSM	Global System for Mobiles
HAB	Home Automation Bus
HAN	Home Area Network
HDFS	Hadoop File System
HLR	Home Location Register
HSPA	High Speed Packet Access
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over TLS/SSL
I/O	Input-Output
I2C	Inter-Integrated Circuit
IaaS	Infrastructure-as-a-Service
IANA	Internet Assigned Number Authority
ICCM	Internet Connected Car Maintenance
ICMP	Internet Control Message Protocol
ICSP	In-Circuit Serial Programming

ICT	Information and Communications Technology
IDE	Integrated Development Environment
IdM	Identity Management
IEC	International Electrotechnical Commission for Standards
IETF	Internet Engineering Task Force
IFTTT	If This Then That service
IIC	Industrial Internet Consortium
IIoT	Industrial IoT
IM	Instant Messaging
IO	Input-Output
IoT	Internet of Things
IP	Internet Protocol
IPSec	IP Security Protocol
IPSP	Internet Protocol Support Profile
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
iq	Information/Query
IR-LED	Infrared Light Emitting Diode
ISDN	Integrated Services Data Network
ISM	Industrial, Scientific and Medical
ISO	International Organization for Standardization
JAR	Java Archive
JID	Jabber ID
JMS	Java Message Service
JSON	Java Script Object Notation
KPI	Key Performance Indicators
LAN	Local Area Network (of Computers)
LED	Light Emitting Diode
LIDAR	Light + Radar, also Laser Imaging, Detection and Ranging
LIN	Local Interconnect Network Bus
LLN	Low Power Lossy Networks
LoRaWAN	Low-power and Range WAN
LPWAN	Low-power WAN
LTE	Long Term Evolution

LWM2M	Lightweight M2M Protocol
LWT	Last Will and Testament on failure of a session, for example, between a client and broker or server
M2M	Machine-to-Machine
MAC	Media Access Control
mDNS	Multicast Domain Name System
MEMS	Micro-Electro-Mechanical Sensor
MFLOPS	Million Floating Point Operations Per Second
MIME	Multipurpose Internet Mail Extension
MINA	Multi-Hop Infrastructure Network Architecture
MIPS	Million Instructions Per Second
MMC	Multimedia Card
MO	Mobile Origin
MOSFET	Metal-oxide Field Effect Transistor
MOST	Media Oriented System Transport
Mote	Mobile Terminal
MPLS	Multiprotocol Label Switching
MPP	Massively Parallel Processing
MQ	Message Queue
MQTT	Message Queue Telemetry Transport protocol
MS	Mobile Station
MSISDN	Mobile Station ISDN Number
MT	Mobile Terminal
MTC	Machine Type Communication
MTU	Maximum Transmission Unit
MUT	Multi-User Chat
NAN	Neighbourhood Area Network
ND	Neighbour Discovery
NDP	Network Discovery Protocol
NFC	Near Field Communication
NFV	Network Function Virtualization
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
ODBC	Open Database Connectivity

OID	Object Identifier
OLAP	On-Line Analytical Processing
OLTP	On-Line Transactions Processing
OMA	Open Mobile Alliance
ONS	Object Name Service
ORCHID	Overlay Routable Cryptographic Hash Identifier
OS	Operating System
OSGi	Open Services Gateway initiative
OWASP	Open Web Application Security Project
P2P	Point-to-Point
PaaS	platform-as-a-service
PAN	Personal Area Network
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PDU	Protocol Data Unit for a layer
PII	Personally Identifiable Information
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PS	Participatory Sensing
PSK	Pre-Shared Key
pubsub	Publication by a service and subscription by end point or client or server
PWM	Pulse Width Modulator
QoS	Quality of Service
QR code	Quick Response code
RAM	Random Access Memory
RARP	Reverse Address Resolution Protocol
RDMS	Relational Database Management System
REST	Representational State Transfer
RF	Radio Frequency of MHz
RFC	IETF Request for Comments standardisation document
RFD	Reduced Function Device
RFIC	RF Integrated Circuits
RFID	Radio Frequency Identification
ROLL	Routes Over the Low power and Lossy network

ROM	Read Only Memory
RPC	Remote Procedure Call
RPi	Raspberry Pi
RPK	Random Pair-wise Keys, Raw-Public-key
RPL	IPv6 Routing Protocol for LLNs (Low Power Lossy Networks)
RPM	Revolution per Minute
RPMP	Re-Planning Manufacturing Process
RTC	Real Time Clock
RTOS	Real Time Operating System
Rx	Receiver
RxD	Receiver Data line
SaaS	Software as a Service
SASL	Simple Authentication and Security Layer
SCADA	Supervisory Control and Data Acquisition
SCL	Serial Clock
SCOVARS	Supply Chain Order Verification, Automated Reordering and Shipping
SD	Service Discovery
SDA	Serial Data
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SIM	Subscriber Identity Module (generally a card) in mobile
SLA	Service Level Agreement
SMPP	Short Message Peer to Peer
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SPI	Serial Peripheral Interface Bus
SPINS	Security Protocols in Network of Sensors
SQL	Structured Query Language
SS7	Signaling Service Protocol
SSID	Service Set Identifier
SSL	Secure Scoket Layer
STP	Spanning Tree Protocol
SWE	Sensor Web Enablement services

TCCICDD	Tracking of Customer Carrying Internet Connected Digital Devices
TCP	Transmission Control Protocol
TCUP	TCS Connected Universe Platform
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TLS	Transport Layer Security
TLV	Tag Length Value
TSDB	Time Series Database
TTP	Trusted Third Party
Tx	Transmitter
TxD	Transmitter Data line
UART	Universal Asynchronous Receiver and Transmitter
UCP/UMI	Universal Computer Interface Protocol/Machine Interface
UDP	User Datagram Protocol
UI	User Interface
UPC	Universal Product Code
URI	Universal Resource Identifier
URL	Universal Resource Locator
USB	Universal Serial Bus
V2I	Vehicle to Infrastructure Communication
VLAN	Virtual Local Area Network
W3C	World Wide Web Consortium
WAN	Wide Area Network
WEP	Wired Equivalent Privacy
WIDL	Web Interface Definition Language
Wi-Fi	Wireless Fidelity
WLAN	Wireless 802.11 Local Area Network
WPA	Wireless Protected Access
WSAPI	WebSocket Application Programming Interface
WSN	Wireless Sensor Node/Network
WWAN	Wireless Wide Area Network
XAAS	Everything-as-a-Service
xep	XMPP Extension Protocol
XHTML	EXtensible HyperText Markup **L**anguage

XML	EXtensible Markup Language
XMPP	EXtensible Messaging and Presence Protocol
XMPP-IoT	XMPP xeps for the IoT/M2M
XSF	XMPP Standards Foundation

1

Internet of Things: An Overview

Learning Objectives

LO 1.1 Describe the basics, definition and vision of Internet of Things (IoT)
LO 1.2 Analyse IoT in terms of a suggested IoT conceptual framework
LO 1.3 Explain the suggested architectural views for IoTs amid diverse technologies
LO 1.4 Describe enabler technologies which are used in designing of: (i) IoT devices, (ii) communication methods between devices and remote server, cloud and applications
LO 1.5 Categorise the resources which enable the development of IoT prototype and product
LO 1.6 Outline the functions of M2M architectural domains and relationships of an M2M system with an IoT system
LO 1.7 Summarise IoT examples of usages in wearable devices, smart homes and smart cities, and understand the architectural frameworks for smart homes and smart cities

1.1 INTERNET OF THINGS

LO 1.1 Describe the basics, definition and vision of Internet of Things (IoT)

Internet of Things (IoT) is a concept which enables communication between internetworking devices and applications, whereby physical objects or 'things' communicate through the Internet. The concept of IoT began

with things classified as identity communication devices. Radio Frequency Identification Device (RFID) is an example of an identity communication device. Things are tagged to these devices for their identification in future and can be tracked, controlled and monitored using remote computers connected through the Internet.

The concept of IoT enables, for example, GPS-based tracking, controlling and monitoring of devices; machine-to-machine (M2M) communication; connected cars; communication between wearable and personal devices and Industry 4.0.

The IoT concept has made smart cities a reality and is also expected to make self-driving cars functional very soon.[1]

The following subsections describe IoT basics, definition, vision, conceptual frameworks and architectures.

1.1.1 IoT Definition

Meaning of the words Internet, Things and Internet of Things

The *Internet* is a vast global network of connected servers, computers, tablets and mobiles that is governed by standard protocols for connected systems. It enables sending, receiving, or communication of information, connectivity with remote servers, cloud and analytics platforms.

Thing in English has number of uses and meanings. In a dictionary, *thing* is a word used to refer to a physical object, an action or idea, a situation or activity, in case when one does not wish to be precise. Example of *reference to an object* is—an umbrella is a useful thing in rainy days. Streetlight is also referred to as a thing. Example of *reference to an action* is—such a thing was not expected from him. Example of *reference to a situation* is—such things were in plenty in that regime.

Thus, combining both the terms, the definition of IoT can be explained as follows:

> *Internet of Things means a network of physical things (objects) sending, receiving, or communicating information using the Internet or other communication technologies and network just as the computers, tablets and mobiles do, and thus enabling the monitoring, coordinating or controlling process across the Internet or another data network.*

Another source, defines the term IoT as follows:

> *Internet of Things is the network of physical objects or 'things' embedded with electronics, software, sensors and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices. Each thing is uniquely identifiable through its embedded computing system but is able to interoperate within the existing Internet infrastructure.*

[1]For more information refer http://en.wikipedia.org/wiki/Autonomous_car#cite_note-99 and http://www.dailycamera.com/the-bottom-line/ci_24021170/self-driving-cars-could-be-decade-away

1.1.2 IoT Vision

Vision of IoT—things becoming intelligent, smart and behaving alive

Internet of Things is a vision where things (wearable watches, alarm clocks, home devices, surrounding objects) become 'smart' and function like living entities by sensing, computing and communicating through embedded devices which interact with remote objects (servers, clouds, applications, services and processes) or persons through the Internet or Near-Field Communication (NFC) etc. The vision of IoT can be understood through Examples 1.1 and 1.2.

Example 1.1

Through computing, an umbrella can be made to function like a living entity. By installing a tiny embedded device, which interacts with a web based weather service and the devices owner through the Internet the following communication can take place. The umbrella, embedded with a circuit for the purpose of computing and communication connects to the Internet. A website regularly publishes the weather report. The umbrella receives these reports each morning, analyses the data and issues reminders to the owner at intermittent intervals around his/her office-going time. The reminders can be distinguished using differently coloured LED flashes such as red LED flashes for hot and sunny days, yellow flashes for rainy days.

A reminder can be sent to the owner's mobile at a pre-set time before leaving for office using NFC, Bluetooth or SMS technologies. The message can be—(i) *Protect yourself from rain. It is going to rain. Don't forget to carry the umbrella; (ii) Protect yourself from the sun. It is going to be hot and sunny. Don't forget to carry the umbrella*. The owner can decide to carry or not to carry the umbrella using the Internet connected umbrella.

Example 1.2

Streetlights in a city can be made to function like living entities through sensing and computing using tiny embedded devices that communicate and interact with a central control-and-command station through the Internet. Assume that each light in a group of 32 streetlights comprises a sensing, computing and communication circuit. Each group connects to a group-controller (or coordinator) through Bluetooth or ZigBee. Each controller further connects to the central command-and-control station through the Internet.

The station receives information about each streetlight in each group in the city at periodic intervals. The information received is related to the functioning of the 32 lights, the faulty lights, about the presence or absence of traffic in group vicinity, and about the ambient conditions, whether cloudy, dark or normal daylight.

The station remotely programs the group controllers, which automatically take an appropriate action as per the conditions of traffic and light levels. It also directs remedial actions in case a fault develops in a light at a specific location. Thus, each group in the city is controlled by the 'Internet of streetlights'. Figure 1.1 shows the use of the IoT concept for streetlights in a city.

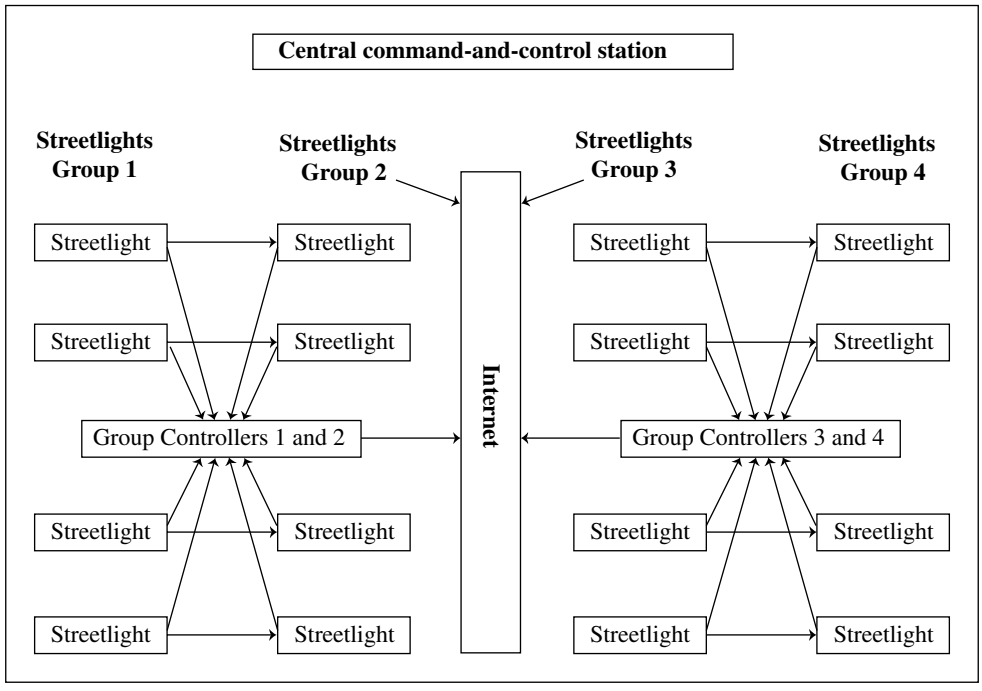


Figure 1.1 Use of Internet of Things concept for streetlights in a city

1.1.3 Smart and Hyperconnected Devices

Hyperconnectivity means 'constant connectivity between devices, network, server and multiple systems'.

As per Collins Dictionary, hyperconnectivity means *use of multiple systems and devices to remain constantly connected to social networks and streams of information*. Smart devices are devices with computing and communication capabilities that can constantly connect to networks. For example, a city network of streetlights which constantly connects to the controlling station as shown in Figure 1.1 for its services. Another example is hyperconnected RFIDs. An RFID or a smart label is tagged to all consignments. This way many consignments sent from a place can be constantly tracked. Their movement through remote places, inventories at remote locations, sales and supply chain are controlled using a hyper-connected framework for Internet of RFIDs.

Figure 1.2 shows a general framework for IoT using smart and hyperconnected devices, edge computing and applications. A device is considered at the edge of Internet infrastructure. Edge computing implies computations at the device level before the computed data communicates over the internet. Several new terms have been used in the figure. Chapters ahead will define and explain these terms in more detail.

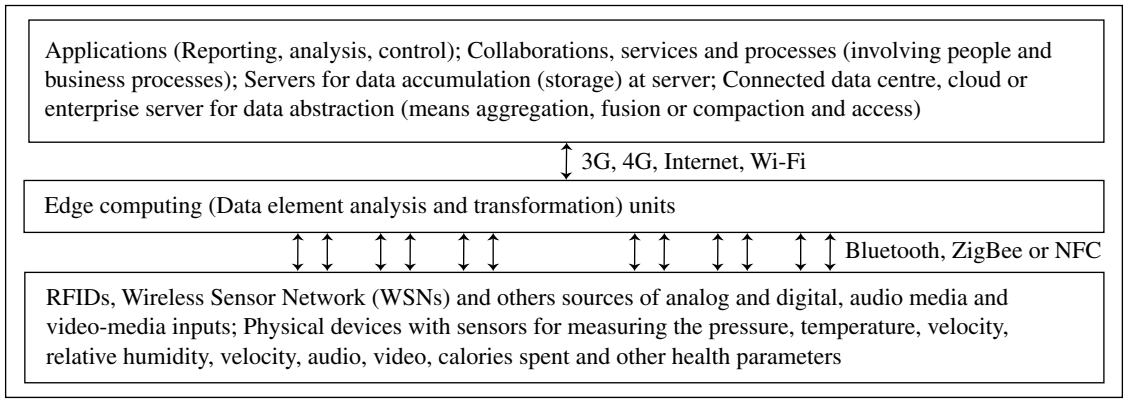


Figure 1.2 A general framework for IoT using smart and hyperconnected devices, edge computing and applications

Reconfirm Your Understanding

- Definition of Internet of Things
- Vision of Internet of Things
- Examples of communicating devices and the internet
- A general framework for smart and hyperconnected devices, edge computing and applications, collaborations, services and processes

Self-Assessment Exercise

1. How can a tracking service track an object communicating its identity whenever it comes close to an Internet Access Point (IAP) in a chain of IAPs at many locations? ★
2. Recall Example 1.1 of smart and alive umbrella. Draw a diagram showing communication of messages between the umbrella, Internet, web-based weather service and a mobile phone. Assuming that the service deploys publish/subscribe (pub/sub) communication mode, define the pub/sub mode of message communication between the two entities. ★★
3. Recall Example 1.2 of smart and alive groups of streetlights. What is the advantage obtained when using a group controller programmed by a central station through the Internet as compared to direct connectivity of each streetlight with the station? ★★
4. Write the definition for Internet of Things. ★
5. Why is hyperconnectivity required for controlling, monitoring, collaborating, rendering services, gathering and analysing information and extracting knowledge? Explain the entities required for each. ★★★

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

1.2 IoT CONCEPTUAL FRAMEWORK

LO 1.2 Analyse IoT in terms of a suggested IoT conceptual framework

Example 1.1 showed a single object (umbrella) communicating with a central server for acquiring data. The following equation describes a simple conceptual framework of IoT[2]:

Physical Object + Controller, Sensor and Actuators + Internet = Internet of Things … 1.1

Equation 1.1 conceptually describes the Internet of umbrellas as consisting of an umbrella, a controller, sensor and actuators, and the Internet for connectivity to a web service and a mobile service provider.

Generally, IoT consists of an internetwork of devices and physical objects wherein a number of objects can gather the data at remote locations and communicate to units managing, acquiring, organising and analysing the data in the processes and services. Example 1.2 showed the number of streetlights communicating data to the group controller which connects to the central server using the Internet. A general framework consists of the number of devices communicating data to a data centre or an enterprise or a cloud server. The IoT framework of IoT used in number of applications as well as in enterprise and business processes is therefore, in general, more complex than the one represented by Equation 1.1. The equation below conceptually represents the actions and communication of data at successive levels in IoT consisting of internetworked devices and objects.

Gather + Enrich + Stream + Manage + Acquire + Organise and Analyse = Internet of Things with connectivity to data centre, enterprise or cloud server … 1.2

Equation 1.2 is an IoT conceptual framework for the enterprise processes and services, based on a suggested IoT architecture given by Oracle (Figure 1.5 in Section 1.3). The steps are as as follows:

1. At level 1 data of the devices (things) using sensors or the things gather the pre data from the internet.
2. A sensor connected to a gateway, functions as a smart sensor (smart sensor refers to a sensor with computing and communication capacity). The data then enriches at level 2, for example, by transcoding at the gateway. Transcoding means coding or decoding before data transfer between two entities.
3. A communication management subsystem sends or receives data streams at level 3.
4. Device management, identity management and access management subsystems receive the device's data at level 4.
5. A data store or database acquires the data at level 5.
6. Data routed from the devices and things organises and analyses at level 6. For example, data is analysed for collecting business intelligence in business processes.

[2] McEwen, Adrian and Cassimally, Hakim, *Designing Internet of Things*, Wiley, 2014.

The equation below is an alternative conceptual representation for a complex system. It is based on IBM IoT conceptual framework. The equation shows the actions and communication of data at successive levels in IoT. The framework manages the IoT services using data from internetwork of the devices and objects, internet and cloud services, and represents the flow of data from the IoT devices for managing the IoT services using the cloud server.

Gather + Consolidate + Connect + Collect + Assemble + Manage and Analyse
= Internet of Things with connectivity to cloud services … 1.3

Equation 1.3 represents a complex conceptual framework for IoT using cloud-platform-based processes and services. The steps are as follows:

1. Levels 1 and 2 consist of a sensor network to gather and consolidate the data. First level gathers the data of the things (devices) using sensors circuits. The sensor connects to a gateway. Data then consolidates at the second level, for example, transformation at the gateway at level 2.
2. The gateway at level 2 communicates the data streams between levels 2 and 3. The system uses a communication-management subsystem at level 3.
3. An information service consists of connect, collect, assemble and manage subsystems at levels 3 and 4. The services render from level 4.
4. Real time series analysis, data analytics and intelligence subsystems are also at levels 4 and 5. A cloud infrastructure, a data store or database acquires the data at level 5.

Figure 1.3 shows blocks and subsystems for IoT in the IBM conceptual framework. New terms in the figure will be explained in the subsequent chapters.

Various conceptual frameworks of IoT find number of applications including the ones in M2M communication networks, wearable devices, city lighting, security and surveillance and home automation. Smart systems use the things (nodes) which consist of smart devices, smart objects and smart services. Smart systems use the user interfaces (UIs), application programming interfaces (APIs), identification data, sensor data and communication ports.

Reconfirm Your Understanding

- *Adrian McEwen* and *Hakim Cassimally* equation is a simple conceptualisation of a framework for IoT with connectivity to a web service:
  Physical Object + Controller, Sensor and Actuators + Internet = Internet of Things
- An equation to conceptualise a general framework for IoT with connectivity to a data centre, application or enterprise server for data storage, services and business processes is:
  Gather + Enrich + Stream + Manage + Acquire + Organise and Analyse
  = Internet of Things
  Orcale suggested IoT architecture is the basis for this equation.
- Another equation which conceptualises the general framework for IoT using the cloud based services is:
  Gather + Consolidate + Connect + Collect + Assemble + Manage and Analyse
  = Internet of Things

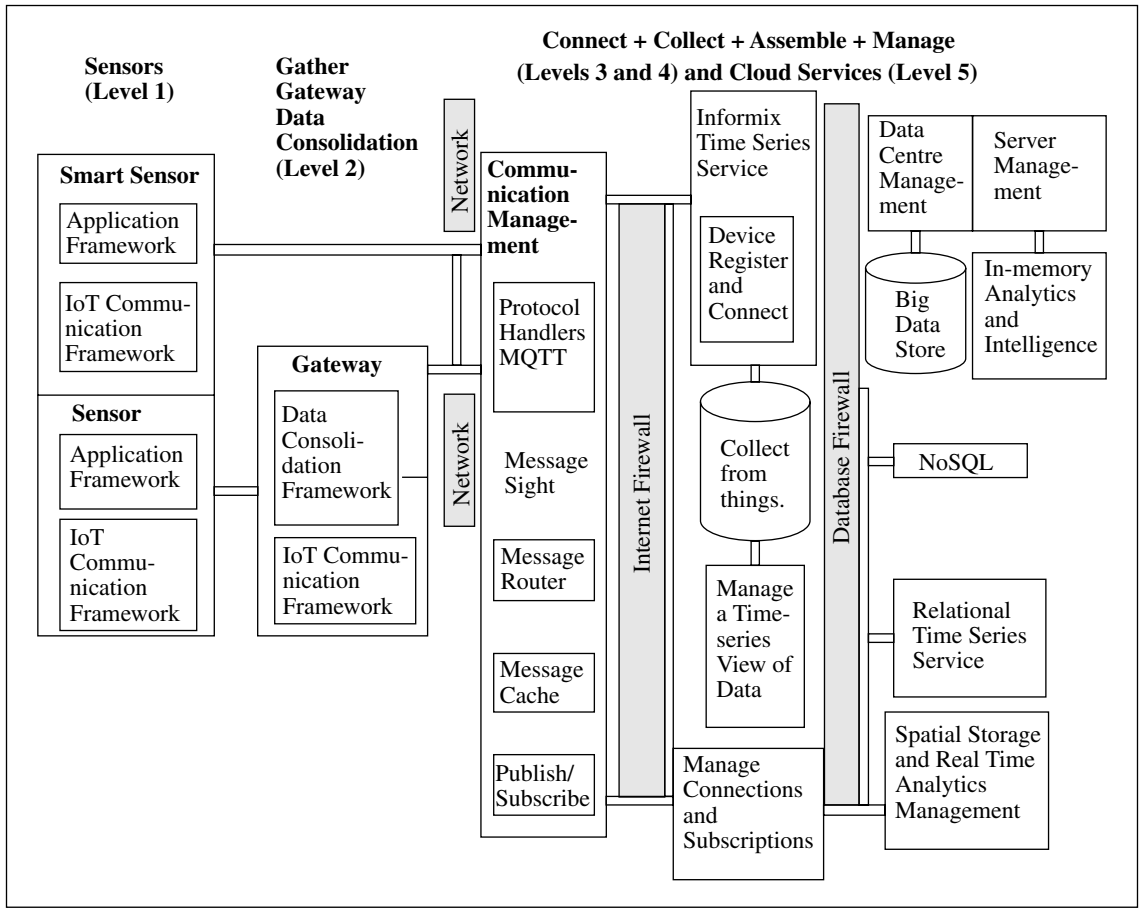


Figure 1.3 IBM IoT conceptual framework

- IBM IoT conceptual framework blocks and components are the basis of this equation.
- In general, things refer to an internetwork of devices and physical objects. This framework consists of a number of subsystems. The data is acquired at remote locations in a database or data store. The services and processes need data managing, acquiring, organising and analysing.

Self-Assessment Exercise

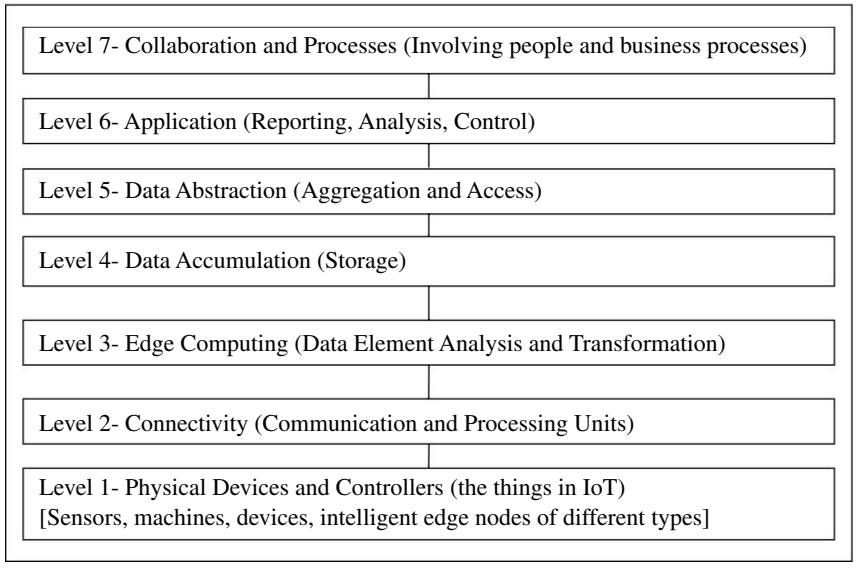
1. How does Equation 1.1 relate to the smart umbrella in Example 1.1. ★
2. Recall Example 1.1 of the smart umbrella. Draw a conceptual framework showing communication of messages between the umbrella, Internet, web-based weather service and mobile phone. ★

3. Draw the conceptual framework for a tracking service to track an object which communicates its identity whenever it comes close to an Internet Access Point (IAP) in a chain of IAPs at many locations. ★★
4. How does Equation 1.2 relate to smart and alive groups of streetlights in Example 1.2. Make a table relating the terms in the equation with the functionalities of units in Figure 1.1. ★★
5. How does Equation 1.3 relate to the IBM IoT framework in Figure 1.3 for Internet-connected cloud-based services? Explain. ★★★

1.3 IoT ARCHITECTURAL VIEW

LO 1.3 Explain the suggested architectural views for IoTs amid diverse technologies

An IoT system has multiple levels (Equations 1.1 to 1.3). These levels are also known as tiers. A model enables conceptualisation of a framework. A reference model can be used to depict building blocks, successive interactions and integration. An example is CISCO's presentation of a reference model comprising seven levels (Figure 1.4). New terms in the figure will be explained in the subsequent chapters.



CISCO seven leveled reference model

Figure 1.4 An IoT reference model suggested by CISCO that gives a conceptual framework for a general IoT system

A reference model could be identified to specify reference architecture. Several reference architectures are expected to co-exist in the IoT domain. Figure 1.5 shows an Oracle suggested IoT architecture. New terms in the figure will be explained in the subsequent chapters.

IoT architecture by Oracle

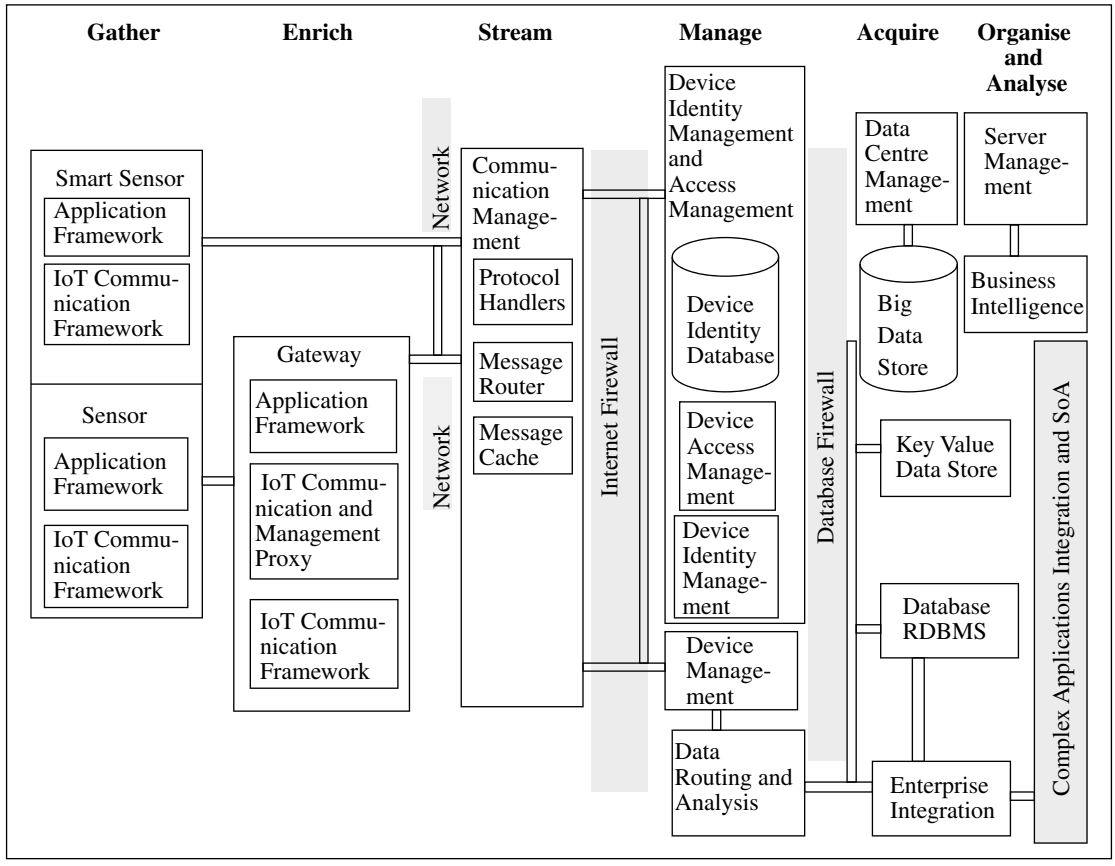


Figure 1.5 Oracle's IoT architecture (Device identity management means identifying a device, registering a device for actions after identifying, de-registering the device, assigning unique identity to the device. Device access management means enabling, disabling the device access, authenticating a device for access, authorizing a device for access to a subsystem. Chapter 2 will explain these in greater detail.)

An architecture has the following features:

- The architecture serves as a reference in applications of IoT in services and business processes.
- A set of sensors which are smart, capture the data, perform necessary data element analysis and transformation as per device application framework and connect directly to a communication manager.

- A set of sensor circuits is connected to a gateway possessing separate data capturing, gathering, computing and communication capabilities. The gateway receives the data in one form at one end and sends it in another form to the other end.
- The communication-management subsystem consists of protocol handlers, message routers and message cache.
- This management subsystem has functionalities for device identity database, device identity management and access management.
- Data routes from the gateway through the Internet and data centre to the application server or enterprise server which acquires that data.
- Organisation and analysis subsystems enable the services, business processes, enterprise integration and complex processes (These terms are explained in Chapter 5).

A number of models (CISCO, Purdue and other models) have been proposed at SWG (Sub Working Group) Teleconference of December 2014. Standards for an architectural framework for the IoT have been developed under IEEE project P2413. IEEE working group is working on a set of guidelines for the standards.

IEEE suggested P2413[3] standard for architecture of IoT. It is a reference architecture which builds upon the reference model(s). The reference architecture covers the definition of basic architectural building blocks and their integration capability into multi-tiered systems.

IEEE P2413 standard provides reference architecture for various IoT domains

P2413 architectural framework[4] is a reference model that defines relationships among various IoT verticals, for example, transportation and healthcare. P2413 provides for the following:

- Follows top-down approach (consider top layer design first and then move to the lowest)
- Does not define new architecture but reinvent existing architectures congruent with it
- Gives a blueprint for data abstraction
- Specifies abstract IoT domain for various IoT domains
- Recommends quality 'quadruple' trust that includes protection, security, privacy and safety
- Addresses how to document
- Strives for mitigating architecture divergence(s)

Scope of IEEE P2413 standard defines an architectural framework for the IoT. It includes descriptions of various IoT domains, definitions of IoT domain abstractions and identification of commonalities between different IoT domains. Smart manufacturing, smart grid, smart buildings, intelligent transport, smart cities and e-health are different IoT domains. P2413 leverages existing applicable standards. It identifies planned or ongoing projects with a similar or overlapping scope.[5]

[3] http://grouper.ieee.org/groups/2413/Sept14_meeting_report-final.pdf
[4] Jan Höller et al., *From Machine-to-Machine to the Internet of Things*, Academic Press, 2014.
[5] http://grouper.ieee.org/groups/2413/April15_meeting_report-final.pdf.

Reconfirm Your Understanding

- A reference model for architectural view consists of seven levels from physical devices to data storage, abstraction, application and collaboration processing (CISCO).
- IoT architecture builds to serve as a reference architecture in applications of IoTs in services and business processes (Oracle).
- IEEE P2413 standard provides reference architecture for the IoT that builds upon the reference models which cover the definition of basic architectural building blocks and their integration capability into multi-tiered systems.
- Several reference architectures are expected to co-exist in IoT domains.
- Specification of abstract IoT domain for various IoT domains have been discussed.
- Quality 'quadruple' trust that includes protection, security, privacy and safety has also been discussed.

Self-Assessment Exercise

1. List the features of CISCO reference model. ★
2. List the features of Oracle reference architecture. ★
3. Describe the meaning of reference architecture and reference model. ★
4. What should reference architecture cover in IoT general conceptual framework? ★
5. How do the blocks and components in IoT IBM conceptual framework and Oracle reference architecture correlate? ★★
6. How do the CISCO reference model and Oracle reference architecture correlate in an IoT architecture? ★★★
7. Why does a smart sensor connect directly while a regular sensor connects through a gateway to the communication management subsystem (Fig. 1.5)? ★
8. Why does a network of sensors and smart sensors need the functionalities of device identity database, device identity management, and device access and device management? ★★
9. How does the IoT device data organise? ★★
10. What are the uses of data after analysis of acquired data? ★★
11. What are the provisions in the architectural framework in IEEE P2413? ★★★

1.4 TECHNOLOGY BEHIND IoT

LO 1.4 Describe enabler technologies which are used in designing of: (i) IoT devices, (ii) communication methods between devices and remote server, cloud and applications

The following entities provide a diverse technology-environment and are examples of technologies, which are involved in IoT.

- Hardware (Arduino Raspberry Pi, Intel Galileo, Intel Edison, ARM mBed, Bosch XDK110, Beagle Bone Black and Wireless SoC)
- Integrated Development Environment (IDE) for developing device software, firmware and APIs
- Protocols [RPL, CoAP, RESTful HTTP, MQTT, XMPP (Extensible Messaging and Presence Protocol)]
- Communication (Powerline Ethernet, RFID, NFC, 6LowPAN, UWB, ZigBee, Bluetooth, WiFi, WiMax, 2G/3G/4G)
- Network backbone (IPv4, IPv6, UDP and 6LowPAN)
- Software (RIOT OS, Contiki OS, Thingsquare Mist firmware, Eclipse IoT)
- Internetwork Cloud Platforms/Data Centre (Sense, ThingWorx, Nimbits, Xively, openHAB, AWS IoT, IBM BlueMix, CISCO IoT, IOx and Fog, EvryThng, Azure, TCS CUP)
- Machine learning algorithms and software. An example of machine-learning software is GROK from Numenta Inc. that uses machine intelligence to analyse the streaming data from clouds and uncover anomalies, has the ability to learn continuously from data and ability to drive action from the output of GROK's data models and perform high level of automation for analysing streaming data.[6]

The following five entities can be considered for the five levels behind an IoT system (Figure 1.3):

1. Device platform consisting of device hardware and software using a microcontroller (or SoC or custom chip), and software for the device APIs and web applications
2. Connecting and networking (connectivity protocols and circuits) enabling internetworking of devices and physical objects called things and enabling the internet connectivity to remote servers
3. Server and web programming enabling web applications and web services
4. Cloud platform enabling storage, computing prototype and product development platforms
5. Online transactions processing, online analytics processing, data analytics, predictive analytics and knowledge discovery enabling wider applications of an IoT system

1.4.1 Server-end Technology

IoT servers are application servers, enterprise servers, cloud servers, data centres and databases. Servers offer the following software components:

- Online platforms
- Devices identification, identity management and their access management
- Data accruing, aggregation, integration, organising and analysing
- Use of web applications, services and business processes

[6] http://numenta.com/grok/

1.4.2 Major Components of IoT System

Major components of IoT devices are:

1. **Physical object** with embedded software into a hardware.
2. **Hardware** consisting of a microcontroller, firmware, sensors, control unit, actuators and communication module.
3. **Communication module:** Software consisting of device APIs and device interface for communication over the network and communication circuit/port(s), and middleware for creating communication stacks using 6LowPAN, CoAP, LWM2M, IPv4, IPv6 and other protocols.
4. **Software** for actions on messages, information and commands which the devices receive and then output to the actuators, which enable actions such as glowing LEDs, robotic hand movement etc.

Sensors and Control Units

Sensors

Sensor types—analog and digital output sensors

Sensors are electronic devices that sense the physical environments. An industrial automation system or robotic system has multiple smart sensors embedded in it. Sensor-actuator pairs are used in control systems. A smart sensor includes computing and communication circuits.

Recall Example 1.2 of Internet of streetlights. Each light has sensors for measuring surrounding light-intensity and surrounding traffic-proximity for sensing and transmitting the data after aggregation over a period. Sensors are used for measuring temperature, pressure, humidity, light intensity, traffic proximity, acceleration in an accelerometer, signals in a GPS, proximity sensor, magnetic fields in a compass, and magnetic intensity in a magnetometer.

Sensors are of two types. The first type gives analog inputs to the control unit. Examples are thermistor, photoconductor, pressure gauge and Hall sensor. The second type gives digital inputs to the control unit. Examples are touch sensor, proximity sensor, metal sensor, traffic presence sensor, rotator encoder for measuring angles and linear encoders for measuring linear displacements. Sensors and circuits are explained in detail in Chapter 7.

Control Units

Most commonly used control unit in IoT consists of a Microcontroller Unit (MCU) or a custom chip. A microcontroller is an integrated chip or core in a VLSI or SoC. Popular microcontrollers are ATmega 328, ATMega 32u4, ARM Cortex and ARM LPC.

An MCU comprises a processor, memory and several other hardware units which are interfaced together. It also has firmware, timers, interrupt controllers and functional IO units. Additionally, an MCU has application-specific functional circuits designed as per the specific version of a given microcontroller family. For example, it may possess Analog to Digital Converters (ADC) and Pulse Width Modulators (PWM). Figure 1.6 shows various functional units in an MCU that are embedded in an IoT device or a physical object. New terms in the figure will be discussed in detail in Chapter 8.

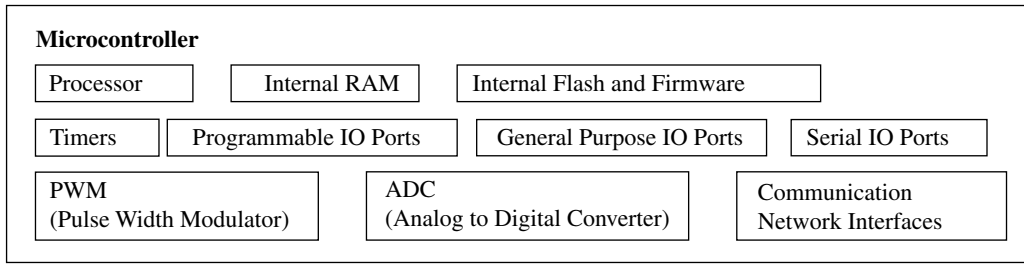


Figure 1.6 Various functional units in an MCU that are embedded in an IoT device or a physical object

Communication Module

A communication module consists of protocol handlers, message queue and message cache. A device message-queue inserts the messages in the queue and deletes the messages from the queue in a first-in first-out manner. A device message-cache stores the received messages.

Representational State Transfer (REST) architectural style can be used for HTTP access by GET, POST, PUT and DELETE methods for resources and building web services. Communication protocols and REST style are explained in detail Chapter 3 and 4.

Software

IoT software consists of two components—software at the IoT device and software at the IoT server. Figure 1.7 shows the software components for the IoT device hardware and server. Embedded software and the components are explained in Chapter 8. Software APIs, online component APIs and web APIs are explained in Section 9.4.

Middleware

OpenIoT is an open source middleware. It enables communication with sensor clouds as well as cloud-based 'sensing as a service'. IoTSyS is a middleware which enables provisioning of communication stack for smart devices using IPv6, oBIX, 6LoWPAN, CoAP and multiple standards and protocols. The oBIX is standard XML and web services protocol oBIX (Open Building Information Xchange).

Operating Systems (OS)

Examples of OSs are RIOT, Raspbian, AllJoyn, Spark and Contiki.

RIOT, Raspbian, AllJoyn, Spark and Contiki OS for IoT

RIOT is an operating system for IoT devices. RIOT supports both developer and multiple architectures, including ARM7, Cortex-M0, Cortex-M3, Cortex-M4, standard x86 PCs and TI MSP430.

Raspbian is a popular Raspberry Pi operating system that is based on the Debian distribution of Linux.

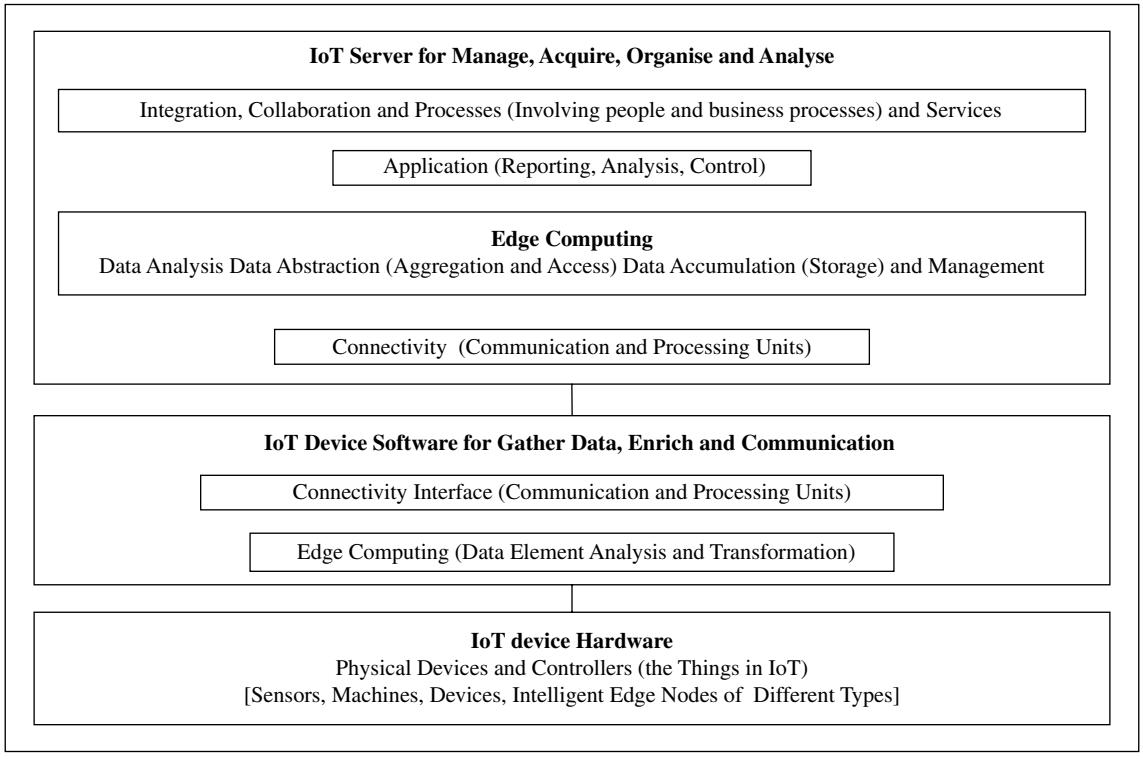


Figure 1.7 IoT software components for device hardware

AllJoyn is an open-source OS created by Qualcomm. It is a cross platform OS with APIs available for Android, iOS, OS X, Linux and Windows OSs. It includes a framework and a set of services. It enables the manufacturers to create compatible devices.

Spark is a distributed, cloud-based IoT operating system and web-based IDE. It includes a command-line interface, support for multiple languages and libraries for working with several different IoT devices.

Contiki OS[7] is an open-source multitasking OS. It includes 6LowPAN, RPL, UDP, DTLS and TCP/IP protocols which are required in low-power wireless IoT devices. Example of applications are street lighting in smart cities, which requires just 30 kB ROM and 10 kB RAM.

Firmware

Thingsquare Mist is an open-source firmware (software embedded in hardware) for true Internet-connectivity to the IoT. It enables resilient wireless mesh networking. Several microcontrollers with a range of wireless radios support Things MIST.

[7] https://en.wikipedia.org/wiki/Contiki and http://anrg.usc.edu/contiki/index.php/Contiki_tutorials

1.4.3 Development Tools and Open-source Framework for IoT Implementation

Eclipse IoT (www.iot.eclipse.org) provides open-source implementation of standards such as MQTT CoAP, OMA-DM and OMA LWM2M, and tools for working with Lua, services and frameworks that enable an Open Internet of Things. Eclipse developed the IoT programming language—Lua. Eclipse website provides sandbox environments for experimenting with the tools and a live demo. Eclipse-related popular projects are Paho, Koneki and Mihini.

Arduino development tools provide a set of software that includes an IDE and the Arduino programming language for a hardware specification for interactive electronics that can sense and control more of the physical world.[8]

Kinoma software platform—Kinoma Create (kit for prototyping), Kinoma Studio development environment and Kinoma Platform Runtime are three different open-source projects. Kinoma Connect is a free app for iOS and Android smartphones and tablets with IoT devices.

1.4.4 APIs and Device Interfacing Components

Connectivity interface consists of communication APIs, device interfaces and processing units. Figure 1.8 shows the mbed™ API and device interfacing components.

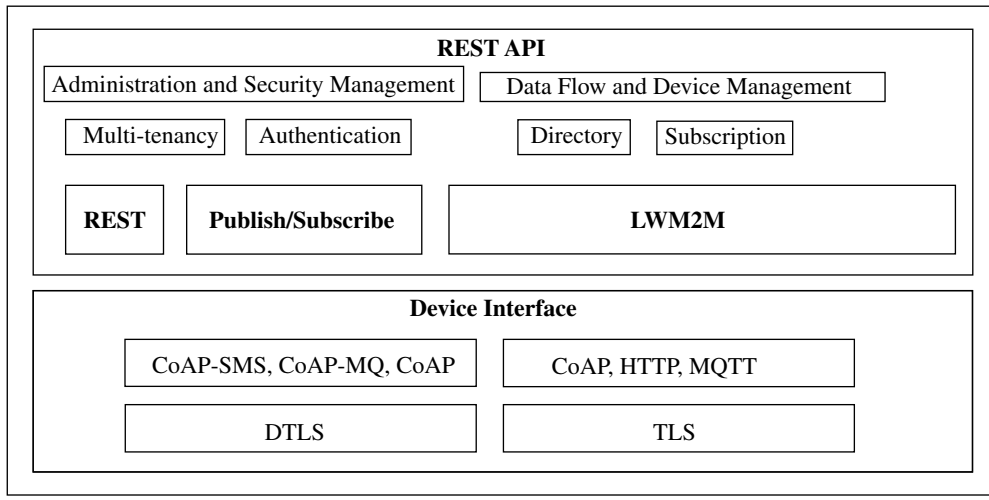


Figure 1.8 mBed™ API and device interfacing components

Example 1.3

List the development platform components for an ARM cortex microcontroller using mBed™ application and IoT prototype and product development platform.

[8] http://www.datamation.com/

A development platform consists of the OS (real-time operating system) and prototype development hardware and software for the IoT. An mbed™ Software Development Kit (SDK) enables development of a firmware for smart devices. SDK has the following software components:

- mbed C/C++ software platform
- IDE consisting of tools for creating microcontroller firmware
- CoRE libraries, microcontroller peripheral drivers, RTOS, runtime environment, build tools and test and debug scripts
- Networking module

An online IDE (freeware) consists of an editor and compiler. The code compiles within the web browser using ARMCC (ARM C/C++ Compiler). The application developer uses mbed IDE or Eclipse with GCC ARM embedded tools.

1.4.5 Platforms and Integration Tools

ThingSpeak is an open data platform with an open API. It consists of APIs that enable real-time data collection, geolocation data, data processing and visualisations. It enables device status messages and plugins. It can process HTTP requests and store and process data. It can integrate multiple hardware and software platforms. It supports Arduino, Raspberry Pi,[9] ioBridge/RealTime.io, and Electric Imp. An important feature of ThingSpeak is the support to MATLAB data analytics, mobile, web applications and social networks.

Nimbits is a cloud platform which supports multiple programming languages, including Arduino, JavaScript, HTML or the Nimbits.io Java library.[10] The software deploys on Google App Engine, any J2EE server on Amazon EC2 or Raspberry Pi. It processes a specific type of data and can also store the data. The data can be time- or geo-stamped.

IoT Toolkit offers Smart Object API, HTTP-to-CoAP Semantic mapping and a variety of tools for integrating multiple IoT-related sensor networks and protocols.[11]

SiteWhere provisions a complete platform for managing IoT devices. It enables gathering of data and integrating it with external systems. SiteWhere can be used on Amazon's cloud or downloaded. It also integrates MongoDB, ApacheHBase and multiple big data tools.[12]

Other platforms are Microsoft Azure,[13] TCS connected universe platform (TCS CUP),[14] Xively,[15] smartliving[16], thethings.io[17] and exosite.[18] Usage of Xively and Nimbits cloud are described in detail in Chapter 6. Details of TCS CUP are described in Chapter 12.

[9] https://thingspeak.com/
[10] http://www.nimbit.com/and http://bsautner.github.io/com.nimbits/
[11] http:// www. iot-toolkit.com//
[12] http://www.sitewhere.org/
[13] https://azure.microsoft.com/en-in/develop/iot/get-started/
[14] http://www.tcs.com/research/Pages/TCS-Connected-Universe-Platform.aspx
[15] https://www.xively.com/
[16] http://www.smartliving.io
[17] https://thethings.io/
[18] https://exosite.com/

Reconfirm Your Understanding

- IoT design involves number of technology areas for IoT applications and services.
- IoT design spans over number of technology areas: hardware, interfaces, firmware, communication protocols, internet connections, data storage, analytics and machine learning tools.
- IoT hardware needs sensors, actuators and device platform, IDE, development tools.
- IoT software needs device platform communication protocols, number of network communication protocols and network backbone protocols.
- IoT design needs platforms for prototyping and product development and integration tools.
- Several open-source software and OSs for system development are available.
- Several cloud platforms and data centres for system development are available.

Self-Assessment Exercise

1. List the microcontrollers and device platforms which IoTs can use. ★
2. What are the functions of various functional units in a microcontroller that embeds in an IoT device? ★★
3. List the device platform communication protocols, network communication protocols and network backbone protocols which IoTs can use. ★★
4. List the available open-source software which can be used. Specify the features of RIOT and Eclipse IoT. ★★
5. List the available cloud platform and data centre open sources. ★
6. What does platform and integration tool mean? What are the features of ThingSpeak? ★★
7. Explain mBed™ application and IoT product development platform. ★★★

1.5 SOURCES OF IoT

LO 1.5 Categorise the resources which enable the development of IoT prototype and product

Examples of hardware sources for IoT prototype development are Arduino Yún, Microduino, Beagle Board and RasWIK. Hardware prototype needs an IDE for developing device software, firmware and APIs.

1.5.1 Popular IoT Development Boards

Arduino Yún

Arduino Yún board uses microcontroller ATmega32u4 that supports Arduino and includes Wi-Fi, Ethernet, USB port, micro-SD card slot and three reset buttons. The board also combines with Atheros AR9331 that runs Linux.

Microduino

Microduino is a small board compatible with Arduino that can be stacked with the other boards. All the hardware designs are open source.

Intel Galileo

Intel Galileo is a line of Arduino-certified development boards. Galileo is based on Intel x86 architecture. It is open-source hardware that features the Intel SOC X1000 Quark based Soc.

Galileo is pin-compatible with Arduino. It has 20 digital I/O (12 GPIOs fully native), 12-bit PWM for more precise control, six analog inputs and supports power over Ethernet (PoE).

Intel Edison

Intel Edison[19] is a compute module. It enables creation of prototypes and fast development of prototyping projects and rapidly produces IoT and wearable computing devices. It enables seamless device internetworking and device-to-cloud communication. It includes foundational tools. The tools collect, store and process data in the cloud, and process rules on the data stream. It generates triggers and alerts based on advanced analytics.

Beagle Board

Beagle Bone based board has very low power requirement. It is a card-like computer which can run Android and Linux. Both the hardware designs and the software for the IoT devices are open source.

Raspberry Pi Wireless Inventors Kit (RasWIK)

RasWIK enables Raspberry Pi Wi-Fi connected devices. It includes documentation for 29 different projects or you can come up with one of your own. There is a fee for the devices but all of the included code is open source, and you can use it to build commercial products as well.

Prototype development boards are described in detail in Chapter 8.

1.5.2 Role of RFID and IoT Applications

Earlier IoT systems were internet-connected RFID based systems. RFID enables tracking and inventory control, identification in supply chain systems, access to buildings and road tolls or secured store centre entries, and devices such as RFID-based temperature sensors. RFID networks have new applications in factory design, 3PL-management, brand protection, and anti-counterfeiting in new business processes for payment, leasing, insurance and quality management. RFID systems are described in detail in Chapter 7.

[19] http://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison_pb_331179002.pdf

1.5.3 Wireless Sensor Networks (WSNs)

Sensors can be networked using wireless technology and can cooperatively monitor physical or environmental conditions. Sensors acquire data from remote locations, which may not be easily accessible. Each wireless sensor also has communication abilities for which it uses a radio-frequency transceiver. Each node either has an analog sensor with signal conditioner circuit or a digital sensor. Sensing can be done to monitor temperature, light intensity, presence of darkness, metal proximity, traffic, physical, chemical and biological data etc.

WSN Definition

Wireless Sensor Network (WSN) is defined as a network in which each sensor node connects wirelessly and has capabilities of computations for data compaction, aggregation and analysis plus communication and networking. WSN node is autonomous. Autonomous refers to independent computing power and capability to send requests and receive responses, and data forward and routing capabilities.

A web source defines the WSN as "a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations."

WSN Node

A WSN node has limited computing power. It may change topology rapidly. The WSN network in the topology-changing environment functions as an ad-hoc network. A WSN network in that environment is generally self-configuring, self-organising, self-healing and self-discovering. WSNs are described in detail in Chapter 7.

Reconfirm Your Understanding

- IoT sources are microcontroller-based boards—Arduino, Intel Galileo, Intel Edison, Beagle Board and Raspberry Pi. The sources have open source IDEs and development tools.
- Arduino uses a microcontroller, for example, ATmega 328 or ATmega 32u4.
- Raspberry Pi uses ARM Cortex and ARM LPC microcontroller-based boards.
- RFIDs are identifying devices which communicate their identity and have number of IoT applications, such as inventory control, tracking and supply chain.
- Wireless Sensor Nodes are wireless sensors which form a self-configuring and self-discovering network.

Self-Assessment Exercise

1. List the sources of IoT development board which can be used for prototype development. ★
2. What are the features in Arduino Yún? ★
3. When will Intel Galileo and Intel Edison be preferred in an IoT application? ★★
4. What are RFIDs? Draw the architecture for their inventory control application. ★★★
5. Define Wireless Sensor Network and list its applications. ★

1.6 M2M COMMUNICATION

LO 1.6 Outline the functions of M2M architectural domains and relationships of an M2M system with an IoT system

Machine-to-machine (M2M) refers to the process of communication of a physical object or device at machine with others of the same type, mostly for monitoring but also for control purposes. Each machine in an M2M system embeds a smart device. The device senses the data or status of the machine, and performs the computation and communication functions. A device communicates via wired or wireless systems. The communication protocols are 6LowPAN, LWM2M, MQTT, and XMPP. Each communication device is assigned 48-bits Ipv6 address.

1.6.1 M2M to IoT

IoT technology in industry involves the integration of complex physical machinery M2M communication with the networks of sensors, and uses analytics, machine learning, and knowledge discovery software.

M2M technology closely relates to IoT when the smart devices or machines collect data which is transmitted via the Internet to other devices or machines located remotely. The close difference between M2M and IoT is that M2M must deploy device to device, and carry out the coordination, monitoring, controlling of the devices and communicate without the usage of Internet whereas IoT deploys the internet, server, internet protocols and server or cloud end applications, services or processes.

M2M has many applications in fields such as industrial automation, logistics, smart grid, smart cities, health and defence. Initial applications of M2M were found in automation and instrumentation only, but now these include telemetric applications and Industrial Internet of Things (IIoT) as well.

Example 1.4

Problem
Give examples of usages of M2M and IIoT.

Solution
1. Examples of M2M usages are coordinated movement of tools, robots, drones, refinery operations and sequential control at each stage during manufacturing. These include manufacturing of food packets, assembly in assembly lines, tracking of failures along the railway tracks etc.
2. IIoT finds applications in the fields of manufacturing at multiple locations, railways, mining, agriculture, oil and gas, utilities, transportation, logistics and healthcare services along with usages of the internet, and usages of software for analytics, machine learning, and knowledge discovery in these areas.

1.6.2 M2M Architecture

M2M architecture consists of three domains (Figure 1.9):
1. M2M device domain
2. M2M network domain
3. M2M application domain

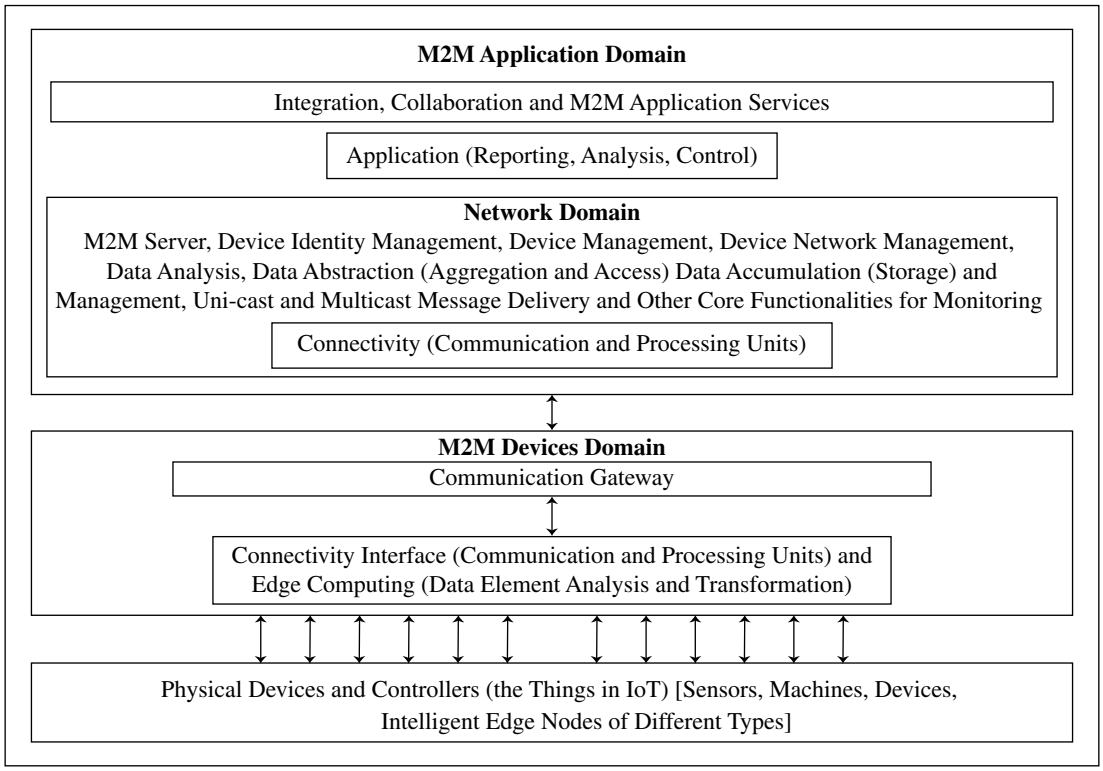


Figure 1.9 Three domains of M2M architecture

M2M device communication domain consists of three entities: physical devices, communication interface and gateway. Communication interface is a port or a subsystem, which receives the input from one end and sends the data received to another.

M2M network domain consists of M2M server, device identity management, data analytics and data and device management similar to IoT architecture (connect + collect + assemble + analyse) level.

M2M application domain consists of application for services, monitoring, analysis and controlling of devices networks.

1.6.3 Software and Development Tools

Examples of M2M software and development tools are as follows:

- **Mango** is an open source M2M web-based software. It supports multiple platforms, multiple protocols, databases, meta points, user-defined events and import/export.[20]
- **Mainspring** from M2MLabs is a development tool, and source framework for developing M2M applications.[21] It enables:
  - Flexible modelling of devices and their configurations
  - Communication between devices and applications
  - Validation and normalisation of data
  - Long-term data storage and data retrieval functions
  - Programming in Java and Apache Cassandra
  - Usages of no SQL database.
- **DeviceHive** is an M2M communication framework. It is an M2M platform and integration tool. It enables connecting devices to the IoT. It includes web-based management software that creates security-rules-based e-networks and monitoring devices. The web software enables prototype projects built with DeviceHub and online tests to find out how it works.
- **Open M2M Protocols, Tools and Frameworks:** Following are the open protocols, tools and frameworks for M2M:
  - XMPP, MQTT-OASIS standards group and OMA LWM2M-OMA standard group for protocol
  - Various projects of Eclipse M2M industry working groups' are Koneki, Eclipse SCADA for open standards for communication protocols, tools and frameworks
  - ITU-T Focus Group M2M global standardisation initiative for a common M2M service layer
  - 3GPPP study group for security aspects of M2M equipment and automatic SIM activation covering remote provisioning and change of subscription.
  - Weightless (wireless communications) group for standards and using wireless spaces for M2M

[20] https://lx-group.com.au/mango-worlds-popular-open-source-m2m-platform/
[21] www.m2mlabs.com/

Following is an example of M2M/IoT.

Example 1.5

An open-source software for designing connected cars is available with Local Motors (a company) that collaborates with IBM for software of IoT connected vehicles. Most of the open-source software and design specifications for the prototype can be downloaded from the Local Motors website[22].

Reconfirm Your Understanding

- M2M refers to communication of devices (machines) with others of same type mostly for monitoring and control purposes.
- M2M technology closely relates to IoT.
- Close differences exist in M2M and IoT. M2M uses device-to-device communication for coordinated monitoring plus control purposes but IoT uses the Internet and server or cloud end protocols and applications.
- M2M device communication domain consists of three entities: physical devices, communication interface and gateway.
- M2M network domain consists of M2M server, device identity management, data analytics, data management and device management similar to IoT architecture (Connect + Collect + Assemble + Analyse) level.
- M2M application domain consists of application for services, monitoring, analysis and control of the devices networks.
- Open protocols, tools and frameworks are available for M2M.
- Various projects of Eclipse M2M industry working groups' are Koneki, Eclipse SCADA. These are open standard communication protocols, tools and frameworks.

Self-Assessment Exercise

1. What does M2M mean? ★
2. How does M2M relate to IoT? What are the differences between the two? ★★
3. Give examples of M2M applications. ★
4. What are the three architectural domain functionalities in M2M architecture? ★★
5. What are the levels in M2M having close similarity with IoT? ★★
6. Correlate M2M architectural domains with IoT architecture levels. ★★★
7. What are the features of DeviceHive (M2M) communication framework? ★
8. What are the open protocols, tools and frameworks generally used in M2M? ★★★

[22] https://localmotors.com/

1.7 EXAMPLES OF IoT

LO 1.7 Summarise IoT examples of usages in wearable devices, smart homes and smart cities, and understand the architectural frameworks for smart homes and smart cities

Examples of IoT usages are wearable devices such as watches, fitness trackers, sleep monitors and heart monitors etc. Fitbit (for example, Fitbit Alta fitness tracker), Garmin and other companies manufacture many such devices. Microsoft (Microsoft band might soon be discontinued), Xiaomi and other manufacturers make tracking bands. A fitness tracker wearable band has the following functions:

- Track steps, distance, calories burned and active minutes
- See stats and time with a bright OLED tap display
- Automatically track how long and how well you sleep and set a silent, vibrating alarm
- Personalize with interchangeable metal, leather and classic bands
- Get calls, texts and calendar notifications at a glance when the phone is in a defined range.

Clothing and accessories nowadays incorporate computer and advanced electronic technologies. The design of watches, rings and bands often includes practical functions and features.[23] The next section provides examples of wearable watches and their features.

1.7.1 Wearable Smart Watch

Following are the examples of wearable watches based on IoT concept:

Table 1.1 Features of hyperconnected wearable smart watches

Samsung Galaxy Gear S Smartwatch Features	Apple Watch	Microsoft Wrist Band 2
• Two-inch curved display • Ability to make a phone call (completely independent of an actual smartphone) or send a text • Wi-Fi and Bluetooth connectivity options • GPS enabled • S Health App measures heart rate and UV monitors and informs the wearer of a good time to eat, when he/she has had enough exercise and a good time to take rest • Has navigational features to assist walking	• Apple iSmartwatch has Apps like Nike + Running to track morning or evening runs and health and fitness. It can: ○ track walks ○ measure heart rate ○ make payment using a payment wallet ○ enable listening to songs while exploring parks without the phone ○ enable chat with family ○ update email ○ find a taxi ○ update news ○ navigate for long car trips ○ control Apple TV ○ set reminders for baseball games to be watched	• Fitness tracking • Can help with productivity by displaying email, calendar and message notifications • Works with Windows phone, iOS devices and Android devices • Sensors: Optical heart rate, 3-axis accelerometer, gyrometer, GPS, ambient light, UV, skin temperature, capacitive sensor, galvanic skin response, Barometer

[23] https://en.wikipedia.org/wiki/Wearable_technology

Figure 1.10 shows a wearable smartwatch.

Figure 1.10 Apple iWatch

1.7.2 Smart Home

Sensors and actuators manage a smart home with an Internet connection. Wired and wireless sensors are incorporated into the security sensors, cameras, thermostats, smart plugs, lights and entertainment systems. Do-it-Yourself (DIY) sensors and actuators, include smart plug, motion detector, door/window detector, smoke detector, energy meter interface (electric, gas, water), remote control (built-in authentication), smart relay, surveillance camera, Wireless Hi-Fi speakers, HUE LED lights, electric utility meter etc.[24]

A connected home has the following applications deployed in a smart home:

- Mobile, tablets, IP-TV, VOIP telephony, video-conferencing, video-on-demand, video-surveillance, Wi-Fi and internet
- Home security: Access control and security alerts
- Lighting control
- Home healthcare
- Fire detection or Leak detection
- Energy efficiency
- Solar panel monitoring and control
- Temperature monitoring and HVAC control
- Refrigerator network with maintenance and service centres
- Automated meter reading

Figure 1.11 gives an architectural view of cloud-based IoT platform for a smart home.

[24] http://www.yogasystems.com

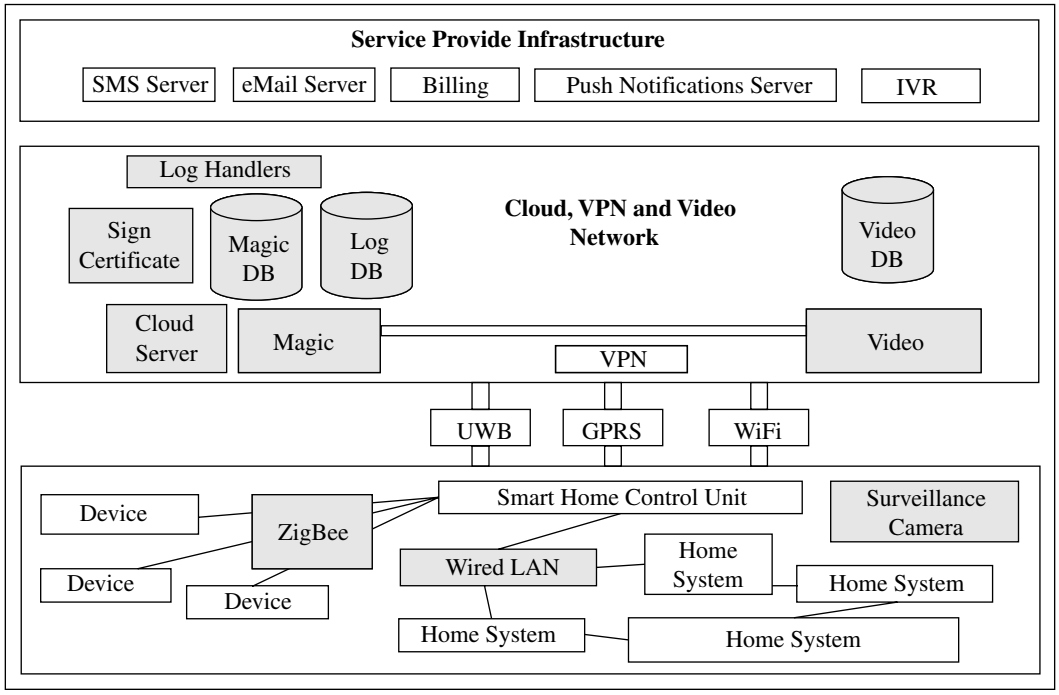


Figure 1.11 An architectural view of cloud (named Magic)-based IoT platform for smart home [VPN: Virtual Private Network, DB: Database, IVR: Interactive Voice Response System, UWB: Ultra Wideband]

Home Automation Software

Intel-based intelligent gateway enables creation of a home automation system offered by service providers for telephony, mobile, cable, broadband and security. **OpenHAB** (open Home Automation Bus) enables the smart home devices that communicate via home. It has a companion cloud computing service called my.openHAB. It runs on a Java-enabled system. **The Thing System** enables the smart home devices which communicate and are controlled via home. The language used is Node.js. It deploys a Raspberry Pi and consists of software components and network protocols for all Internet-connected things at home.

1.7.3 Smart Cities

Architectural four-layer framework of a smart city application

The IoT concept extends to Internet of Everything (IoE) for developing smart cities. A four-layer architectural framework developed at CISCO for a city is as follows (Figure 1.12):

1. Layer 1 consists of sensors, sensor networks and devices network in parking spaces, hospitals, streets, vehicles, banks, water supply, roads, bridges and railroads. Bluetooth, ZigBee, NFC, WiFi are the protocols used at this layer.

2. Layer 2 captures data at distributed computing points where data is processed, stored and analysed.
3. Layer 3 is meant for central collection services, connected data centres, cloud and enterprise servers for data analytics applications.

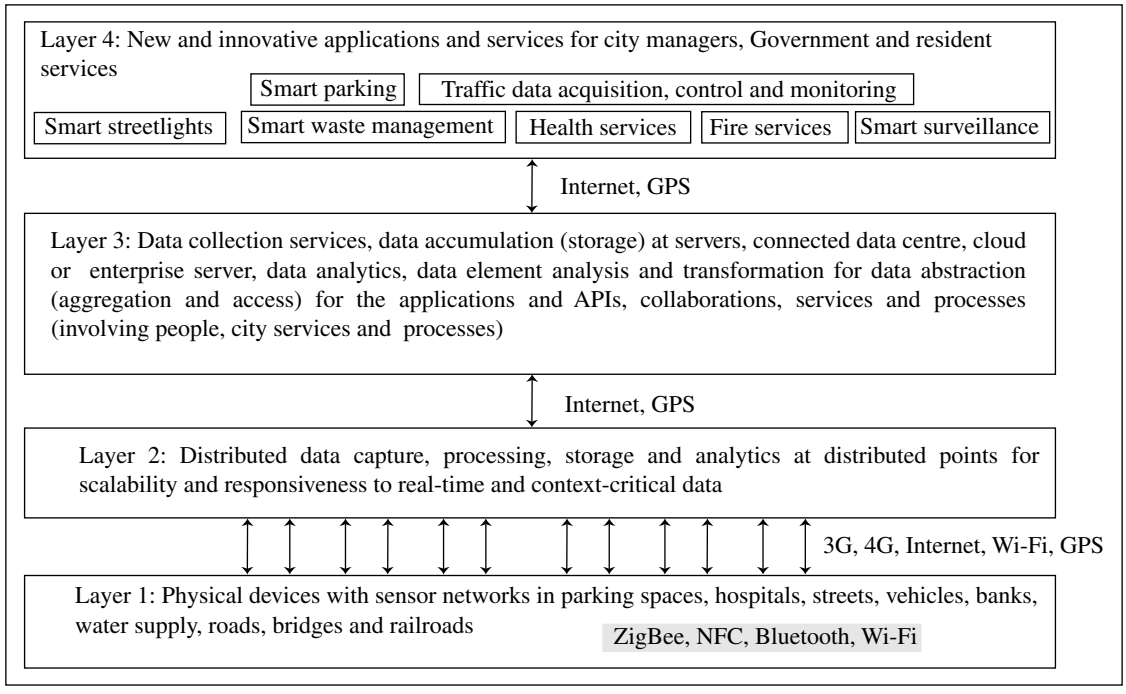


Figure 1.12 Four-layer architectural framework developed at CISCO for a city

4. Layer 4 consists of new innovative applications, such as waste containers' monitoring, WSNs for power loss monitoring, bike sharing management and smart parking. Smart parking refers to services for motorists that informs them about the nearby parking services with vacant spaces in advance.

Smart cities are becoming a reality with innovations being added each year.

Reconfirm Your Understanding

- Hyperconnected wearable smart watches and bands are widely used nowadays. These include wearable watches fitted with health and fitness devices.
- Sensors and actuators manage the smart home with the help of ZigBee, NFC, Wi-Fi and an internet connection. Smart homes deploy a number of wired and wireless sensors.
- An architectural view of cloud (named Magic) is based on IoT platform for a smart home.

- A smart home automation software comprises Intel-based intelligent gateway, OpenHAB and ThingSystem. They enable the smart home devices communicate and control at home.
- Smart cities deploy IoT conceptual frameworks. Four-layer architectural framework developed at CISCO for a city can be deployed.
- Examples of innovative applications for a smart city are smart streetlights control and monitoring, smart fire, health, surveillance, hospital services, waste containers monitoring, WSNs for power loss monitoring and smart parking.

Self-Assessment Exercise

1. What are the features of a wearable device watch? ★
2. Draw a data communication diagram for a smart watch. ★★
3. Give examples of IoTs used in a smart home with sensors, actuators and smart home automation software. ★★
4. What are the entities in network consisting of cloud, VPN and video and home automation software used in architectural view of a smart home? ★★
5. List the components available in Intel-based intelligent gateway for a smart home. ★
6. Explain the role of four-layers in a smart city architectural framework. ★★★
7. Correlate four-layer architecture view with seven level IoT architecture, both from CISCO. ★★★

Key Concepts

- Arduino
- CoAP
- Eclipse
- IoT applications
- IoT architecture
- IoT conceptual framework
- IoT definition
- IoT examples
- IoT hardware
- IoT reference model (CISCO)
- IoT software
- IoT sources
- IoT vision
- M2M communication
- MQTT
- Raspberry Pi
- RFID
- Smart city
- Smart home
- Wearable connected devices
- Wireless sensor network

Learning Outcomes

LO 1.1

- IoT definitions from different angles
- IoT extends the use of internet for physical things (objects)
- IoT vision is to make things (wearable watches, alarm clockes, home devices, surrounding objects) smart and function like living entities through sensing, computing and communicating

- Examples of internet-connected umbrella and RFIDs
- Internet of Things (IoT) devices mean smart hyperconnected devices

LO 1.2

- IoT conceptual framework can be given in terms of three equations:
  - Physical Object + Controller, Sensor and Actuators + Internet = Internet of Things (Adrian McEwen and Hakim Cassimally)
  - Gather + Enrich + Stream + Manage + Acquire + Organise and Analyse = Internet of Things with connectivity to Data Centre and Enterprise Server (Oracle IoT Architecture)
  - Physical Objects + Gather and Consolidate + Connect + Collect + Assemble + Manage = Internet of Things (IBM IoT Foundation)

LO 1.3

- A number of architectures have been suggested in the IoT domain. An IoT domain covers an extremely wide range of technologies. Not all possible suggested implementations can provide a blueprint for single reference architecture and single IoT domain.
- A reference model can probably be identified to specify reference architecture. Several reference architectures are expected to co-exist in the IoT domain. Figure 1.5 shows the Oracle IoT architecture to serve as a reference in applications of IoT in services and business processes.
- IEEE P2413 standard architecture framework is worked out for various IoT domains.

LO 1.4

- IoT deploys technologies of hardware design, sensors, actuators, microcontroller, software, OS, development platform, framework, network, cloud, data centre, enterprise server applications and services.
- IoT data communication gateway uses RPL, IPv4, IPv6, LWM2M, CoAP, MQTT and other protocols for communication.

LO 1.5

- IoT sources for design of prototypes and products are Arduino Uno, Arduino Yun, Raspberry Pi, Beagle Board, Intel Edison, mBed™ and Bosch XDK 110.
- Open source IDEs, development platforms and OSs (for example, RIOT) enable faster development of prototypes and products.

LO 1.6

- Machine-to-machine (M2M) communication refers to communication of machines with others of the same type, mostly for monitoring and also for control purposes.
- M2M technology closely relates to IoT.
- M2M communication protocols are similar to IoT, namely, LWM2M, CoAP, MQTT and XMPP.
- Each machine device is assigned 48-bits Ipv6 addresses.
- Three M2M domains are: device, network and application and their functionalities.

LO 1.7

- Examples of wearable connected devices.
- IoT applications in smart homes and smart cities.
- Understanding of architectural view of a cloud-based IoT platform for a smart home.
- Understanding of a four-layer architectural framework for a smart city.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. CISCO IoT reference models': ★
   (a) Highest level is collaboration and processes, middle level is sever or data centre and lowest level is physical devices, controllers or machines
   (b) Highest level is server and lowest level is communication and processing units
   (c) Highest level is server or data centre and lowest level is communication and processing units
   (d) Highest level is data organise and analyse and lowest level is physical devices or controllers or machines and communication and processing units
2. The equation representing the IBM IoT framework is: ★
   (a) Gateway + (Connect + Collect + Assemble + Manage) and Analyse = Internet of Things
   (b) Gather + Consolidate + (Connect + Collect + Assemble + Manage) and Analyse = Internet of Things
   (c) Physical object + Internet + Manage + Store and Analyse = Internet of Things
   (d) Gateway + (Connect + Database + Organise + Analyse) = Internet of Things
3. Consider the number of IoT applications. IoT is a network of physical things (objects) which consist of (i) sensors, (ii) actuators, (iii) identity communicating with (iv) computing and (v) communication circuit. The objects (vi) send (vii) receive (viii) communicate using the (ix) Internet or other built-in communication technologies. The objects (x) store their data (xi) collaborate among them. The server (xii) stores the data, (xiii) monitors the objects, (xiv) coordinates various processes based on the data, (xv) controls the objects across the Internet or a data network. ★★★
   (a) All are true
   (b) (i) or (ii) or (iii) or all always true, (iv), (v), (vi), (vii), (ix) and (x) always true, (x) and among (xi) to (xv) always true
   (c) (i) or (ii) and (iii) always true, (iv), (v), (ix) always true, (vi) or (vii) or both always true, one or more actions among (x) to (xv) always true
   (d) (i), (iii), (iv) (v), (vi), (vii) and (x) always true
4. The architectural framework for IoT in IEEE P2413 standard provides the following: (i) A reference model that defines relationships among various IoT verticals (ii) Common architecture elements (iii) A blueprint for data abstraction (iv) The quality "quadruple" trust that includes protection, security, privacy and safety." (v) Addressing how to document. ★★
   (a) (i) to (v)
   (b) (ii) to (v)
   (c) (i) to (iv) plus provisioning for limited architecture divergence
   (d) (i) to (v) plus provisioning the striving for mitigating the architecture divergence
5. ThingSpeak is an open data platform with an (i) open API. It consists of APIs. It enables (ii) real-time data collection, (iii) geolocation data, (iv) data processing and visualisations. (v) It enables device status messages and plugins. (vi) It ★★

supports multiple programming languages, including Arduino and JavaScript and (viii) Java library. (ix) It can process HTTP requests and store and process data. (x) It can integrate multiple hardware and software platforms.
(a) (vi) and (vii) not true
(b) All are true
(c) (x) not true
(d) (i), (viii) and (x) not true

6. A machine in M2M has a (i) microcontroller which consists of (ii) processor, (iii) memory (iv) firmware, (v) timers, (vi) interrupt controller (vii) communication module (viii) ADC and (ix) PWM ★
   (a) (i) and (ii) always present in M2M as well as IoT devices and remaining depends on the application
   (b) (i) to (vii) always present in M2M as well as IoT devices and (viii) and (ix) depend upon the application
   (c) (i) to (v) and (vii) always present in M2M devices and (vi), (viii) and (ix) are optional
   (d) All are required in an M2M or IoT device
7. Network domain in M2M consists of (i) M2M server, (ii) device identity management, (iii) machine databases, (iv) data analytics, (v) data management and (vi) device management. ★★
   (a) All are true
   (b) All except (ii) and (v) correct
   (c) Similar to IoT architecture (connect + collect + assemble + analyse) in IBM conceptual framework
   (d) All except (iii) and (vi) true

Short-Answer Questions

1. Define Internet of Things. What were the first uses of IoT? **[LO 1.1]** ★
2. How does Internet of Things differ from Internet-controlled devices? Explain by giving an example. **[LO 1.1]** ★★
3. What are the functions at each level in hyperconnected RFIDs for inventory control application? **[LO 1.1]** ★★★
4. What are the functions at each level in the IBM IoT framework for the processes and services using the cloud? Explain using the example of smart streetlights. **[LO 1.1]** ★★
5. Why is it not possible for single reference architecture to provide a blueprint for all suggested IoT implementations? **[LO 1.2]** ★
6. List the features of the Oracle IoT architecture. **[LO 1.3]** ★★
7. What are the server-end functions in IoT for business processes? **[LO 1.3]** ★★
8. Specify functions of CoAP, RESTful HTTP, MQTT and XMPP (Extensible Messaging and Presence Protocol) in IoT applications. **[LO 1.4]** ★
9. What are the software needs in a communication module in IoT applications? **[LO 1.4]** ★★
10. Why is an IDE required for developing a device platform for an IoT application? **[LO 1.4]** ★★★
11. What does device platform and integration tool mean? Explain using the example of ThingSpeak and Nimbits. **[LO 1.5]** ★★★

12. What are the functions of mBed™ SDK for an IoT application development? [**LO 1.5**] ★★★
13. Why does a WSN node need self-discovering, self-configuring and self-healing characteristics for networking? [**LO 1.5**] ★★
14. State the similarities and differences between the IoT and M2M. [**LO 1.6**] ★★
15. What is the difference between monitoring home surveillance using Internet of Things and a central server? [**LO 1.7**] ★
16. What are some innovative applications which a smart city can deploy? [**LO 1.7**] ★★★

Review Questions

1. What is vision of IoT? How does the vision reflect in use of IoT in smart street lighting? [**LO 1.1**] ★
2. Describe three conceptual frameworks using equations which give the steps at various levels or layers in IoT applications. [**LO 1.2**] ★
3. Describe the actions at different levels in the CISCO IoT reference model. [**LO 1.3**] ★★
4. Draw the Oracle reference architecture for IoT. [**LO 1.3**] ★
5. Give the provisions in IEEE P2413 standard specifications in proposed reference architecture of IoTs. **[LO 1.3]** ★★★
6. What are the technology areas behind IoTs? [**LO 1.4**] ★
7. How does device data communicate to create database in enterprise applications? What are the technologies behind the communication? [**LO 1.4**] ★★★
8. What are the open-source software components for developing an IoT application? [**LO 1.4**] ★★★
9. Describe sources for device platform development for developing an IoT application. [**LO 1.5**] ★
10. What is the composing unit in a circuit for development of a wireless sensor network? [**LO 1.5**] ★★
11. Draw the architectural view of the three domains in M2M applications. What are the software, development tools, open M2M protocols, tools and frameworks for the M2M applications development? [**LO 1.6**] ★★★
12. Describe the features of the Apple smart watch. [**LO 1.7**] ★
13. Draw and explain the architectural view of a cloud-based IoT platform for a smart home. [**LO 1.7**] ★★★
14. Explain the four levels in an architectural framework for a smart city. [**LO 1.7**] ★★

Practice Exercises

1. How will an alarm device be made smart and alive using the internet? Assume that publish, subscribe services for the flight Expected Arrival (ETA) and train ETA are available. [**LO 1.1**] ★
2. Draw an architectural view of an IoT application for a post parcel tracking service with each parcel marked with a bar code. What will be the conceptual equation for the application? [**LO 1.2**] ★
3. Show the similarities in Oracle, IBM and CISCO architectural views for IoT applications and services. [**LO 1.3**] ★★
4. What are the programs required for data analytics for library services and for monitoring issue, receipts and user demands? [**LO 1.4**] ★★

5. What can possibly be the device hardware and software platform for the smart alarm in Practice Exercise 1.1? ★★ **[LO 1.5]**
6. Draw the architectural view of an M2M application for a car for traffic reports, control and monitoring. ★★★ **[LO 1.6]**
7. Develop a conceptual design of a waste container management service in a smart city. ★★★ **[LO 1.7]**

2

Design Principles for Connected Devices

Learning Objectives

LO 2.1 Summarise recent initiatives of international organisations for design standardisation of IoT/M2M architectural layers and domains

LO 2.2 Explain wireless and wired communication technologies for physical cum data-link layer functions

LO 2.3 List functions of data-adaptation layer, and the device and gateway domain

LO 2.4 Discuss ease of designing and affordability of IoT devices

Recall from Previous Chapter

Things refer to physical objects, sensors, machines, devices, controllers and intelligent edge-nodes in the IoT (Equation 1.1). *Connected devices* refer to devices such as streetlights, RFIDs, ATMs and automobiles, which are connected to the internet for applications, services and processes.

Section 1.2 described reference Equations 1.1 to 1.3. The section also described conceptual main entities framework, reference model and reference architecture for IoT and M2M. *Framework* refers to a set of entities or software components which make provisions for generic functionalities. These functionalities can be selectively changed by user codes for developing an application. Conceptual framework means an abstraction of a framework. *Reference model* means main entities conceptualised and the relationships between them. *Reference architecture* means conceptualisation of main entities in a system, their functioning and deployment in the system and the processes rendered by the system.

Figures 1.3 to 1.5 showed that a connected device network communicates with the application server or data centre and cloud service. The applications, services and processes run using the server(s) data, notifications, alerts or messages. The data from the devices is gathered, enriched and streamed using communication and processing units.

Oracle reference architecture (Figure 1.5) based on Equation 1.2, showed that the things communicate with the managing, acquiring, organising and analysing units at the Internet. These enable running the applications and the processes.

The IBM conceptual framework, shown in Figure 1.3, consists of two sub-frameworks. The data from the devices gather, enrich and stream using communication and processing units.

Applications and processes in IoT need that the things communicate with the managing, acquiring, organising and analysing units at the Internet (Equation 1.2). Examples of things are the sensors, machines, devices, controllers and intelligent edge nodes of different types.

Figure 1.4 showed seven levels in IoT reference model. CISCO designed model has devices and other objects at the lowest level 1. The device data communication and processing units are at next level 2. The communication and processing enable the interconnection of devices to the Internet when a system manages, acquires, organises and analyses at higher layers at levels 3 to 7.

Figure 1.9 showed three domains—devices and communication, network, and applications in an M2M architecture. Figure 1.11 gave an architectural view of cloud-based IoT platform for a smart home. The figure showed three frameworks for a smart-home things system; cloud, VPN and video network; and service provider infrastructure. Figure 1.12 showed a four-layer architectural framework which has been developed at CISCO for a smart city.

2.1 INTRODUCTION

When a letter is written then it is written according to a *protocol* (etiquette). To send a letter, it is first put in an envelope, and then the envelope is marked with the receiver's address at the centre, sender's address at left hand bottom area, stamp(s) is/are affixed at right hand top corner and the type of *post* is mentioned on top line in the centre. All letters are then gathered (stacked) and the stack is sent to the target city. Each action takes place according to a specified protocol at each stage (layer). Similarly, when *data* is transferred from a sensor, then functional units create a stack for data communication to an application or service.

IoT or M2M device data refers to the data meant for communication to an application, service or process. Data also refers to data received by a device for its monitoring or for actions at actuator in it.

Data stack denotes the data received after the actions at various in-between layers (or levels or domains). Layers in Open Systems Interconnection (OSI) model are *Application, Presentation, Session, Transport, Network, Data-link* and *Physical*.

Actions at the data-adaptation or other layers can be related to data privacy, data security, data consolidation, aggregation, compaction and fusion. An action can be a *gateway* action—using one protocol for reception and another one for transmission.

The actions at a layer can be adding additional header after appropriate data formatting or transformation at functional units. For example, data from a device can be used by an application or in-between layers when a header (specifying *device ID* or address, destination addresses and other fields) adds and encrypts at the physical (device) layer for its security on the network.

Actions besides appending a header can be of appending the additional bits at the various in-between units. For example, appending additional *message(*s) or an *acknowledgement*.

Following are the key terms which need to be understood to learn the design principles of connected devices for IoTs:

Layer refers to a stage during a set of actions at which the action is taken as per a specific protocol or method, and then the result passes to the next layer until the set of actions complete. A layer may consist of various *sublayers*.

Physical layer refers to a layer at transmitting-node or at the receiving node for the data bits. The transfer uses physical systems and refers to wireless or wired transmission. This layer is the lowest layer.

Application layer refers to a layer for transmitting or receiving the data bits of an application. Data bits route across the network and transfer takes place as follows: application data from the application layer transfers after passing through several in-between layers to the physical layer, and from there it transmits to the receiving-end physical layer. Then, the data at the receiving node transfers from the physical layer to the application layer after passing through several in-between layers.

Level refers to a stage from the lowest to the highest. For example, acquiring device data and actions that may be considered at the lowest level and actions in business processes at the highest level.

Domain refers to a set of software, layers or levels having specific applications and capabilities. For example, CoRE network, access network, service capabilities and applications can be considered as one domain, say, network domain. A domain generally has limited interactions with other domains or outside the domain.

Gateway refers to software for connecting two application layers, one at the sender and the other at the receiver [application layer gateway (ALG)]. A gateway may be of different types. A communication gateway at device and gateway domain has capabilities as protocol-conversion during communication between two ends when each end uses distinct protocols. An Internet gateway may have capabilities besides protocol conversion, transcoding data, device management and data-enrichment before the data communicate over the Internet. Dictionary meaning of gateway is a place you go through because it leads to a much larger place (Section 2.4).

IP stands for Internet Protocol version 6 (IPv6) or Internet Protocol version 4 (IPv4) for the network layer (v6 means version 6, v4 version 4).

Header means a set of octets containing information about the data being sent. Header packs the data of a layer before transmission to the next layer during communication

between two end-points. The size of a header and its fields are according to the protocol used for creating data stack at a layer. For example, IPv4 header has fields as per IP network layer, Universal Datagram Protocol (UDP) header as per UDP at the transport layer and so on. Each header field has distinct meanings. The field size can be between 1 and 32-bit in a packet. A field helps in processing the packet when transferring it from one layer to the next one.

Packet means packaged data-stack which routes over the network. Packet size limit is according to the protocol. For example, IPv4 packet size limit is 2^{16} B (2^{14} words with 1 word = 4 octet).

Protocol Data Unit (PDU) is a unit of data which is specified in a protocol of a given layer which transfers from one layer to another. For example, PDU is *bit* which transfers from physical layer; *frame* from data-link layer; *packet* from network layer; *segment* from transport layer and *text* (plain, encrypted or compressed) from application and other layers.

Maximum Transmission Unit (MTU) is the largest size frame or packet or segment specified in octets (1 octet = 1 byte = 8 bits) that can be sent in a packet or frame-based network such as the Internet. For example, consider transfers of a segment from the transport layer using the Transmission Control Protocol (TCP) to the network layer. The MTU determines the maximum size of each data stack in any transfer to the network layer. The network layer determines the maximum size of each *frame* in any transfer to the data-link layer and then uses MTU of the data-link layer.

Star network denotes the number of nodes interacting with a coordinator or master node.

Mesh network denotes the number of nodes that may interconnect with each other.

End-point device or *node* denotes the one that provides connectivity to a coordinator or router.

Coordinator denotes the one that connects to a number of end-points as well as routers in a star topology and forwards the data stack from one attached end point/router to another.

Master refers to the one who initiates the pairing with the devices in a star topology network.

Slave means one that pairs with a master, uses the clock signals from master for synchronisation and uses address assigned by the master at the beginning.

Router refers to a device or node capable of storing paths to each destination to which it has logical links. The router sends the data stack according to the available path or paths at a receiving instance.

ISM band means Industrial, Scientific and Medical (ISM) radio frequency (RF) bands. 2.4 GHz and the frequencies are 915 MHz for North America, 868 MHz for Europe and 433 MHz band for Asia in ISM bands.

Application means software for specific tasks, such as streetlight monitoring or control.

Service means service software, for example, report generation or chart visualisation service.

Process means a software component, which processes the input and generates the output; for example after analysing the data or acquiring the data. An operating system controls a process, memory for the process and other parameters of the process.

Suggested models, conceptual frameworks, reference models and architectures for the IoT/M2M suggest the following:

- Need of designing a communication framework for connecting devices, for their local area networking and provisions for data gathering.
- Need of designing a data enrichment, data consolidation and data transformation framework.
- Need of designing gateway components for connecting the device's network with the web/Internet.
- Need of application and applications-support frameworks for services, applications and processes.

This chapter describes the physical cum data-link and data-adaptation layers, which communicate with the network layer. This chapter also describes communication frameworks for connecting devices, local area networks and applications. Subsequent chapters will describe the gateway components for connecting device networks with the web/Internet, application and application-support frameworks.

Following sections describe design principles for connected devices. They describe standardised layers in an IoT system, communication frameworks, and wireless and wired technologies for connectivity framework in the devices and network domain. Following sections also describe the frameworks for communication, data management and data enrichment.

2.2 IoT/M2M SYSTEMS, LAYERS AND DESIGNS STANDARDISATION

LO 2.1 Summarise recent initiatives of international organisations for design standardisation of IoT/M2M architectural layers and domains

A number of international organisations have taken action for IoT design standardisation. Following are the examples:

Internet Engineering Task Force (IETF), an international body initiated actions for addressing and working on the recommendations for the engineering specifications for the Internet of Things. IETF suggests the specifications for the layers, and the engineering aspects for the IoT communication, networks and applications.

International Telecommunication Union for Telecommunication (ITU-T) suggested a reference model for IoT domain, network and transport capabilities for the IoT services and the applications at the application and application-support layers.

European Telecommunication Standards Institute (ETSI) initiated the development of a set of standards for the *network*, and *devices and gateway* domains for the communication

between machines (M2M). ETSI proposed high-level architecture for applications and service capabilities.

Open Geospatial Consortium (OGC), an International Industry Consortium, has also suggested open standards for sensors' discovery, capabilities, quality and other aspects with support to geographical information web support.[1, 2]

Following subsections describe these standardisation efforts.

2.2.1 Modified OSI Model for the IoT/M2M Systems

OSI protocols mean a family of information exchange standards developed jointly by the ISO and the ITU-T. The seven-layer OSI model is a standard model. It gives the basic outline for designing a communication network. Various models for data interchanges consider the layers specified by the OSI model, and modify it for simplicity according to the requirement. Similarly, IETF suggests modifications in the OSI model for the IoT/M2M.

Figure 2.1 shows a classical seven-layer OSI model (on the left) and the modifications in that model proposed by IETF (in the middle). Data communicates from device end to application end. Each layer processes the received data and creates a new data stack which transfers it to the next layer. The processing takes place at the in-between layers, i.e. between the bottom functional-layer to the top layer. Device end also receives data from an application/service after processing at the in-between layers. Figure 2.1 also shows a similarity with the conceptual framework in Equation 1.2:

Gather + Enrich + Stream + (Manage + Acquire + Organise +Analyse) = IoT Applications and Services

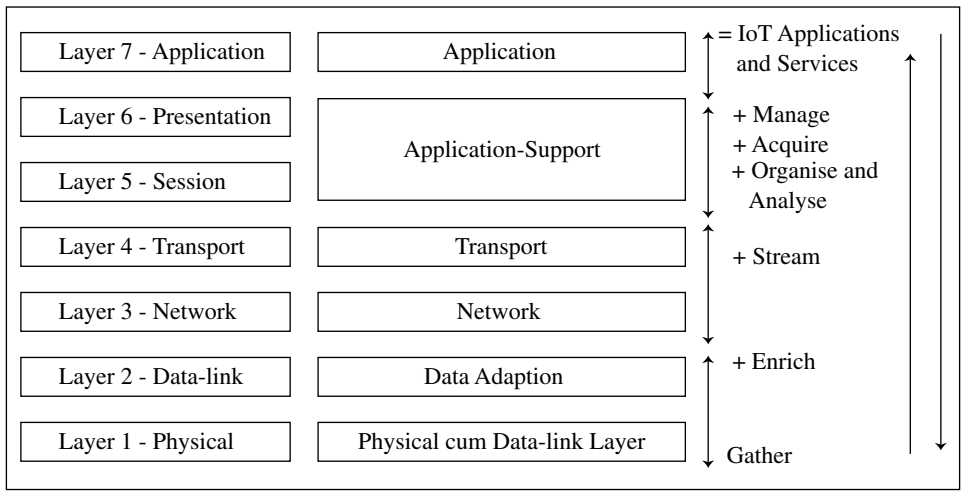


Figure 2.1 Seven-layer generalised OSI model (on left) and IETF six layer modified OSI model for IoT/M2M (in the middle), and similarity with the conceptual framework Equation 1.2 (on right) for IoT applications and services

[1] Höller Jan et al., *Machine-to-Machine to the Internet of Things*, pp. 163-165, Elsevier, 2014
[2] http://www.opengeospatial.org/projects/groups/sensorwebdwg.

New applications and services are present at the application layer 6. A modification to this is that the application-support layer 5 uses protocols, such as CoAP. (Section 3.1.1). IoT applications and services commonly use them for network communication. The CoAP protocol at the layer is used for the request/response interactions between the client and server at the network. Similarly, the application-support layer may include processes for data managing, acquiring, organising and analysing which are mostly used by applications and services.

Modifications are also at the data-link layer 2 (L2) and physical layer 1 (L1). The new layers are data-adaptation (new L2) and physical cum data-link (new L1). The data-adaptation layer includes a gateway. The gateway enables communication between the devices network and the web.

A physical IoT/M2M device hardware may integrate a wireless transceiver using a communication protocol as well as a data-link protocol for linking the data stacks of L1 and L2.

Example 2.1 explains the IETF six-layer OSI model for Internet of streetlights.

Example 2.1

Problem

What are the architectural layers in a modified OSI model for Internet of smart streetlights application in Example 1.1?

Solution

Consider a model for Internet of streetlights (Figure 1.1). Following are the layers for data interchange in the modified OSI model:

- L1: It consists of smart sensing and data-link circuits with each streetlight transferring the sensed data to L2.
- L2: It consists of a group-controller which receives data of each group through Bluetooth or ZigBee, aggregates and compacts the data for communication to the Internet, and controls the group streetlights as per the program commands from a central station.
- L3: It communicates a network stream on the Internet to the next layer.
- L4: The transport layer does device identity management, identity registry and data routing to the next layer
- L5: The application-support layer does data managing, acquiring, organising and analysing, and functionalities of standard protocols such as CoAP, UDP and IP.
- L6: The application layer enables remote programming and issue of central station directions to switch on-off and commands of services to the controllers along with monitoring each group of streetlights in the whole city.

2.2.2 ITU-T Reference Model

Figure 2.2 shows the ITU-T reference model RM1. It also shows correspondence of the model with the six-layers modified OSI model (Figure 2.1). The figure also shows a comparison with CISCO IoT reference model RM2 (Figure 1.4). RM1 considers four layers which are:

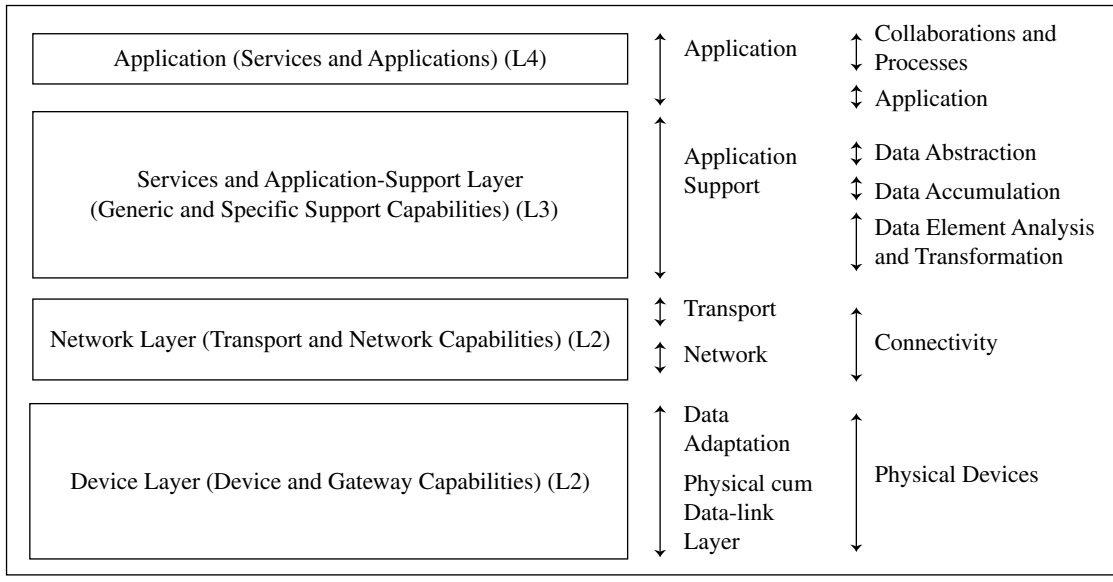


Figure 2.2 ITU-T reference model RM1, its correspondence with six layers of modified OSI and a comparison with seven levels suggested in CISCO IoT reference model RM2

- *Lowest layer*, L1, is the device layer and has device and gateway capabilities.
- *Next layer,* L2, has transport and network capabilities.
- *Next layer,* L3, is the services and application-support layer. The support layer has two types of capabilities—generic and specific service or application-support capabilities.
- *Top layer*, L4, is for applications and services.

ITU-T recommends four layers, each with different capabilities. A comparison of ITU-T RM1 with the six-layer OSI model can be made as follows:

- RM1 device layer capabilities are similar to data-adaptation and physical cum data-link layers.
- RM1 network layer capabilities are similar to transport and network layers.
- RM1 upper two layer capabilities are similar to top two layers.

A comparison with the CISCO IoT reference model (RM2) can be made as follows:

- RM1 L4 capabilities are similar to RM2 collaborations and processes, and application top two levels.
- RM1 L3 capabilities are similar to RM2 three middle-level functions of data abstraction, accumulation, analysis and transformation.
- RM1 L2 layer capabilities are similar to RM2 functions at connectivity level.
- RM1 L1 device layer capabilities are similar to RM2 functions at physical devices level.

Example 2.2 explains ITU-T reference model for Internet of RFIDs.

Example 2.2

Problem

What are the architectural layers in ITU-T reference model for Internet of RFIDs application?

Solution

Consider a model for Internet of RFIDs. Following are the capabilities of the layers and data interchange in the ITU-T reference model.

Layer 1: Device and gateway capabilities are present in the RFID physical device cum RFID reader which acquires the ID data, and communicates the enriched data according to a wireless protocol to an access point (AP).

Layer 2: Transport and network capabilities are present at access network consisting of APs and Internet connectivity to servers.

Layer 3: Services and application-support layer capabilities at server are RFID device's registry, ID management, RFIDs data routing to server or data centre, data analysis for the time-series device presence and device tracked positions.

Layer 4: Services and applications of RFIDs are tracking, inventory control of goods and business processes; for example, supply-chain management.

2.2.3 ETSI M2M Domains and High-level Capabilities

A domain specifies the functional areas. High-level architecture means architecture for functional and structural views. Figure 2.3 shows ETSI M2M domains and architecture, and the high-level capabilities of each domain. It also shows that the architecture correspondences with the six-layer modified OSI model as well as the four layers of the ITU-T reference model.

The ETSI network domain has six capabilities and functions:

1. M2M applications
2. M2M service capabilities
3. M2M management functions
4. Network management functions
5. CoRE network (for example, 3G and IP networks, network control functions, interconnections among networks)
6. Access network (for example, LPWAN (low power wide area network), WLAN (Wi-Fi) and WiMax networks)

The ETSI device and gateway domain has the following functional units:

- Gateway between M2M area network, and CoRE and access network, possessing M2M service capabilities and applications
- M2M area network (for example, Bluetooth, ZigBee NFC, PAN, LAN)
- M2M devices

Example 2.3 explains M2M ETSI domains and high-level architecture for applications and services ATMs-to-bank servers.

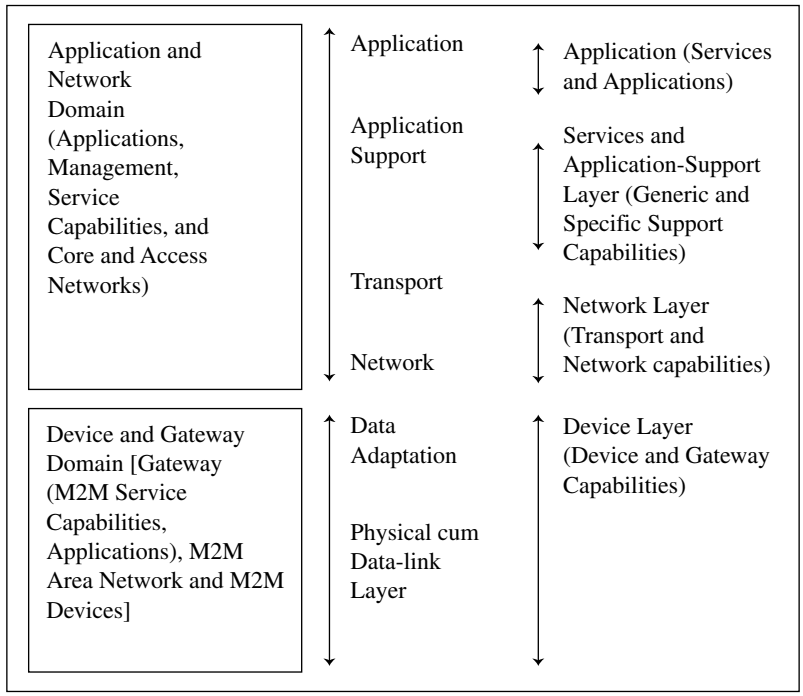


Figure 2.3 ETSI M2M domain architecture and its high-level capabilities, and its correspondences with six layers of modified OSI and four layers of ITU-T reference model

Example 2.3

Problem

What are domains and their service capabilities in ETSI high-level architecture for applications and services in Internet of ATM machines?

Solution

ETSI high-level architecture for applications and services in Internet of ATM machines has two domains:

- ***Device and gateway domain***: Device refer to cards and ATMs, while ATM service capabilities and ATM applications are present at the ATM gateway. The gateway has system for acquiring the card as well as banking data. Data interchange between an ATM machine and the bank server takes place through the gateway. The domain has cash dispensing and surveillance systems. All the ATM systems network through an access network. The gateway communicates the data after enriching and transcoding according to the network protocol between an AP and data for the machine. A domain subsystem monitors cash dispensing and other services.
- ***Applications and network domain:*** Application and network domain has two functional units—ATM management functions and network management functions. It has banking applications and service capabilities for the ATMs. It communicates with the bank CoRE network which connects all the access networks of ATM gateways.

Reconfirm Your Understanding

- International organisations IETF, ITU-T and ETSI have recently taken the initiative and recommendations for layers, capabilities, domains and architecture standardisation for IoT/M2M systems.
- Three proposed international standards, proposed conceptual frameworks, IoT architectures and reference models of IBM, Oracle and CISCO have similarities.
- Architectural layers and domains in IoT/M2M examples—Internet of streetlights, Internet of RFIDs and Internet of ATMs—are shown using modified OSI, ITU-T and ETSI standard specifications.
- The IETF six-layer model proposes an application-support layer in place of a session and presentation layers in the OSI model. CoAP is an application-support layer protocol. CoAP transactions use the request/response interactions model.
- The IETF six-layer model has a data-adaptation layer in place of a data-link layer. An adaptation layer protocol is 6LoWPAN [IPv6 over low power wireless personal area networks (WPAN)]. The model has a physical cum data-link layer in place of the physical layer in the OSI.
- The ITU-T reference model has a layer for applications and services, and three more layers for three other types of capabilities.
- The ETSI high-level M2M architecture has two domains. One is a network domain which has six capabilities and functions. The ETSI device and gateway domain has functional units consisting of gateway, CoRE and access network, M2M area network and M2M devices.

Self-Assessment Exercise

1. Draw the IETF six-layer model for Internet of RFIDs applications for tracking and inventory control. ★
2. Draw the ITU-T reference model for Internet of streetlights. ★
3. Draw the proposed ETSI high-level architecture for the ATMs-bank server applications and services. ★
4. How can the functional units in CISCO seven levels be associated with four layers in the ITU-T reference model? ★★
5. Why is a gateway necessary in a communication framework for IoT and M2M applications and services? ★★
6. Why is an additional layer for application-support required in IoT/M2M applications? ★★
7. How can the functional units in the Oracle IoT architecture be associated with each of the six layers in the IETF IoT model? ★★
8. Why does IETF consider it important that the modified OSI specifies physical/data-link and data-adaptation layers in place of separate physical and data-link layers in its model for computer networks? ★★★
9. Draw the ETSI M2M high-level architecture for a traffic management system for connected automobiles. Assume that application provisions for the real-time traffic density reports. The reports display in maps at the automobiles. ★★★

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

2.3 COMMUNICATION TECHNOLOGIES

LO 2.2 Explain wireless and wired communication technologies for physical cum data-link layer functions

Physical cum data-link layer in the model consists of a local area network/personal area network. A local network of IoT or M2M device deploys one of the two types of technologies—wireless or wired communication technologies. Figure 2.4 shows connected devices (1st to ith) connectivity using different technologies for communication of data from and to devices to the local network connectivity to a gateway.

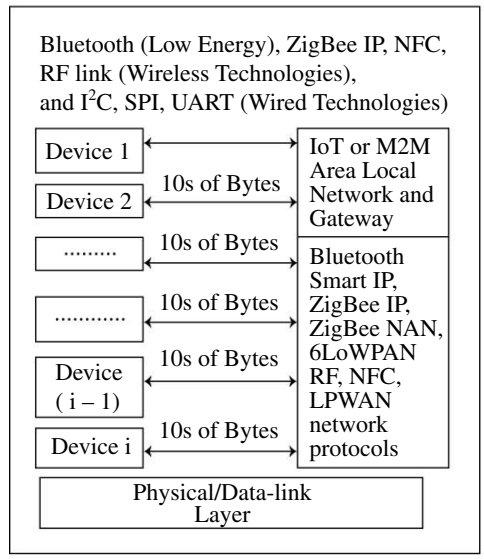


Figure 2.4 Connected devices 1st to ith connected to the local network and gateway using the WPAN or LPWAN network protocols

Wireless and wired technologies for communication of data from and to devices

Figure 2.4 shows number of devices present in an IoT or M2M devices network. The figure shows the local area network of devices. The connectivity between the devices (left-hand side) is by using RF, Bluetooth Smart Energy, ZigBee IP, ZigBee NAN (neighbourhood area network), NFC or 6LoWPAN or mobile. Tens of bytes communicate at an instance between the device and local devices network.

Following subsections describe the technologies and standards recommended for the communication.

2.3.1 Wireless Communication Technology

Physical cum data-link layer uses wired or wireless communication technologies. Examples of wireless communication technologies are NFC, RFID, ZigBee,

Bluetooth (BT), RF transceivers and RF modules. Following subsections describe these wireless communication technologies.

Near-Field Communication

Near-Field communication (NFC) is an enhancement of ISO/IEC[2]14443 standard for contact-less proximity-card.

NFC is a short distance (20 cm) wireless communication technology. It enables data exchange between cards in proximity and other devices. Examples of applications of NFC are proximity-card reader/RFID/IoT/M2M/mobile device, mobile payment wallet, electronic keys for car, house, office entry keys and biometric passport readers.

NFC devices transmit and receive data at the same instance and the setup time (time taken to start the communication) is 0.1 s. The device or its reader can generate RF fields for the nearby passive devices such as passive RFID. An NFC device can check RF field and detect collision of transmitted signals. The device can check collision when the received signal bits do not match with the transmitted signal bits. Features of an NFC device are:

Range of functioning is within 10 to 20 cm. The device can also communicate with Bluetooth and Wi-Fi devices in order to extend the distance from 10 cm to 30 m or higher. The device is able to receive and pass the data to a Bluetooth connection or standardised LAN or Wi-Fi using information handover functions. Device data transfer rates are 106 kbps, 212 kbps, 424 kbps and 848 kbps (bps stands for bit per second, kbps for kilo bit per second). Three modes of communication are:

1. ***Point-to-point (P2P) mode:*** Both devices use the active devices in which RF fields alternately generate when communicating.
2. ***Card-emulation mode:*** Communication without interruption for the read and write as required in a smart card and smart card reader. FeliCa™ and Mifare™ standards are protocols for reading and writing data on the card device and reader, and then the reader can transfer information to Bluetooth or LAN.
3. ***Reader mode:*** Using NFC the device reads passive RFID device. The RF field is generated by an active NFC device. This enables the passive device to communicate.

RFID

Radio Frequency Identification (RFID) is an automatic identification method. RFIDs use the Internet. RFID usage is, therefore, in remote storage and retrieval of data is done at the RFID tags. An RFID device functions as a tag or label, which may be placed on an object. The object can then be tracked for the movements. The object may be a parcel, person, bird or an animal. IoT applications of RFID are in business processes, such as parcels tracking and inventory control, sales log-ins and supply-chain management. Section 7.5.1 describes the details of the technology.

[2] ISO stands for International Organization for Standardization and IEC stands for International Electrotechnical Commission.

Bluetooth BR/EDR and Bluetooth Low Energy

Bluetooth devices follow IEEE 802.15.1 standard protocol for L1 (physical cum data-link layer). BT devices form a WPAN devices network. Two types of modes for the devices are Bluetooth BR/EDR (Basic Rate 1 Mbps/Enhanced Data Rate 2 Mbps and 3 Mbps) and Bluetooth low energy (BT LE 1Mbps). A latest version is Bluetooth v4.2. BT LE is also called *Bluetooth Smart*. Bluetooth v4.2 (December 2014) provides the LE data packet length extension, link layer privacy and secure connections, extended scanner and filter link layer policies and IPSP. BT LE range is 150 m at 10 mW power output, data transfer rate is 1 Mbps and setup time is less than 6 s.

Bluetooth v5, released in June 2016, has increased the broadcast capacity by 800%, quadrupled the range and doubled the speed.

A device may have provisions for single mode BT LE or dual mode BT BR/EDR (Mbps stands for Million Bits per second). Its features are:

- Auto-synchronisation between mobile and other devices when both use BT. BT network uses features of self-discovery, self-configuration and self-healing.
- Radio range depending on class of radio; Class 1 or 2 or radios: 100 m, 10 m or 1 m used in device BT implementation.
- Support to NFC pairing for low latency in pairing the BT devices.
- Two modes—dual or single mode devices are used for IoT/M2M devices local area network.
- IPv6 connection option for BT Smart with IPSP (Internet Protocol Support Profile).
- Smaller packets in LE mode.
- Operation in secured as well as unsecured modes (devices can opt for both link-level as well as service-level security or just service level or unsecured level).
- AES-CCM 128 authenticated encryption algorithm for confidentiality and authentication (Refer Example 2.4).
- Connection of IoT/M2M/mobile devices using BT EDR device to the Internet with 24 Mbps Wi-Fi 802.11 adaptation layer (AMP: Alternative MAC/PHY layer) or BT-enabled wire-bound connection ports or device. MAC stands for media access control sublayer at a data-link layer/sublayer.

Example 2.4

Problem

How does the Bluetooth layer provide confidentiality and authorisation?

Solution

A standard algorithm AES (Advanced Encryption Algorithm) based on symmetric 128-bit block data encryption and CCM mode (Counter with CBC-MAC) provides the confidentiality and authorisation. CBC stands for cryptographic block cipher with a block length of 128 bits. CCM is a method which provisions for the authenticated encryption algorithm for confidentiality and authentication.

ZigBee IP/ZigBee SE 2.0

ZigBee devices follow the IEEE 802.15.4 standard protocol L1 (physical cum data-link layer). ZigBee devices form a WPAN devices network.

ZigBee end-point devices form a WPAN of embedded sensors, actuators, appliances, controllers or medical data systems which connect to the Internet for IoT applications, services and business processes.

ZigBee Neighbourhood Area Network (NAN) is a version for a smart grid. ZigBee smart energy version 2.0 has energy management and energy efficiency capabilities using an IP network.

The features of ZigBee IP are:

- L1 layer PDU = 127 B
- Used for low-power, short-range WPAN
- The device can function in six modes—end point, ZigBee-ZigBee devices router, ZigBee network coordinator, ZigBee-IP coordinator, ZigBee-IP router and IP host.
- ZigBee IP enhancement provisions the IPv6 connectivity. A ZigBee IP device is a Reduced Function Device (RFD). RFD means one that functions for the 'sleepy'/ battery-operated device. Sleepy means one that wakes up infrequently, sends data and then goes back to sleep. ZigBee IP supports IPv6 network with 6LoWPAN header compression, connection for Internet communication and control of low power devices, TCP/UDP transport layer and TLSv1.2 public key (RSA and ECC) and PSK cipher suite for end-to-end security protocol, end-to-end means application layer to physical layer.
- The ZigBee router uses reactive and proactive protocols for routing mode, which enable applications in big-scale automation and remote controls.
- A self-configuring and self-healing dynamic pairing mesh network, supports both multicast and unicast options.
- Multicast forwarding to support multicast Domain Name System (mDNS) based service discovery (SD)
- Support to development of discovery mechanism with full application confirmation
- Support to pairing of coordinator with end-point devices and routers in star topology
- Provides bigger network using multiple star topology and inter-PAN communications
- Support to sensor nodes and sensor (or appliances) network integration, sensor and appliances devices configured as router or end-devices
- Low latency (< 10 ms) link layer connection
- Range is 10–200 m, data transfer rate is 250 kbps, low power operation
- ISM band frequencies direct sequence spread spectrum 16-channel radio, and provide link level security using AES-CCM-128 (Example 2.4)
- Includes RFD in ZigBee SE 2.0

ZigBee NAN is for devices which are used for smart-metering, distribution automation devices and smart grid communication profile. NAN enables a utility's last-mile at HAN (Home Area Network), outdoor access network that connects smart meters to WAN (wide-area network) gateways.

Figure 2.5 shows ZigBee End Point, Coordinator, Router, ZigBee IP Router modes forming star, mesh and IP networks of ZigBee sensors, end devices and ZigBee router device which interconnect to Internet IPv4, IPv6 and to cellular networks.

Figure 2.5 shows:

- Three end devices, two routers, one sensor node connected to coordinator ZigBee devices forming a star network.
- One end device, two routers and one coordinator forming a mesh network.
- Mesh network router connects to an AP/gateway, which in turn connects to a cellular network.
- Coordinator of mesh network connects to ZigBee IP border router, which enables local ZigBee networks' connectivity to the Internet.

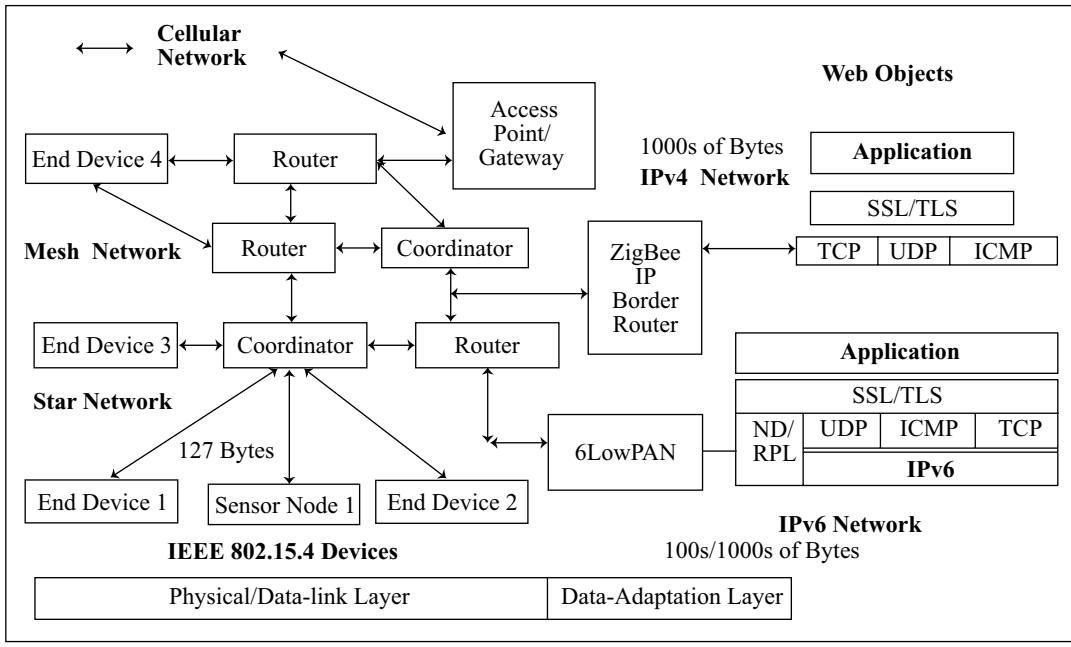


Figure 2.5 ZigBee end point, coordinator, router, ZigBee IP router nodes forming the star, mesh and IP networks of ZigBee sensors, end devices, and ZigBee router devices which interconnect to Internet IPv4, IPv6 and cellular networks

Features of a ZigBee network are:

- A router in star network connects to 6LoWPAN, which connects an IEEE 802.15.4 devices network to IPv6 network.
- 1000s of byte communicate between the network layer and IoT web objects.
- 127 B communication between the adaptation layer IEEE 802.15.4 devices at single data transfer.
- IETF ND (Neighbour Discovery), ROLL (Routing Over Low power Loss Network), RPL routing, IPv6/IPv4 network, TCP/UDP/ICMP transport, SSL/TLS security layer protocols for the communication between web object/application and ZigBee devices.

Wi-Fi

Wi-Fi is an interface technology that uses IEEE 802.11 protocol and enables the Wireless Local Area Networks (WLANs). Wi-Fi devices connect enterprises, universities and offices through home AP/public hotspots. Wi-Fi connects distributed WLAN networks using the Internet.

Automobiles, instruments, home networking, sensors, actuators, industrial device nodes, computers, tablets, mobiles, printers and many devices have Wi-Fi interface. They network using a Wi-Fi network.

Wi-Fi is very popular. The issues of Wi-Fi interfaces, APs and routers are higher power consumption, interference and performance degradation.

Wi-Fi interfaces connect within themselves or to an AP or wireless router using Wi-Fi PCMCIA or PCI card or built-in circuit cards and through the following:

- Base station (BS) or AP
- A WLAN transceiver or BS can connect one or many wireless devices simultaneously to the Internet.
- Peer-to-peer nodes without access point: Client devices within Independent Basic Service Set (IBSS) network can communicate directly with each other. It enables fast and easy setting of an 802.11 network.
- Peer to multipoint nodes with Basic Service Sets (BSSs) using one in-between AP point or distributed BSSs connect through multiple APs.
- Connectivity range of each BSS depends on the range of wireless bridges and antennae used and environmental conditions.
- Each BSS is a Service Set Identifier (SSID).

Figure 2.6 shows three WLAN networks (BSSs) for sensor device nodes, mobiles, tablets, laptops, computers and Internet connectivity of WLAN networks with the IP4 networks (here dashed lines represent wireless connectivity and solid lines represent wired connectivity).

Figure 2.6 shows the following:

- Sensor nodes connected to BT with Wi-Fi adaptation, 802.11 interfaces in a WLAN network 1 (WLAN1).
- Tablets, Wi-Fi, computers also connect in WLAN 1 though an AP.
- AP1 connects to a broadband router 1 and to the IP4 network 2.
- WLAN1 and WLAN2 function as BSS.
- WLAN 2 also consists of AP2, Wi-Fi router and other Wi-Fi enabled interfaces.
- Wi-Fi router connects to multiple Wi-Fi nodes as well as to a broadband router 2.
- Broadband routers 1 and 2 connect using wires, to IP4 networks and web objects for IoT apps, services and processes.

The Wi-Fi interfaces, access points, routers features are as follows:

- Generally used are the 2.4 GHz IEEE 802.11b adapters or 5 GHz (802.11a or 802.11g) or 802.11n or other 802.11 series protocols.
- Interfaces use 2.4 GHz or 5 GHz antenna
- Offers mobility and roaming

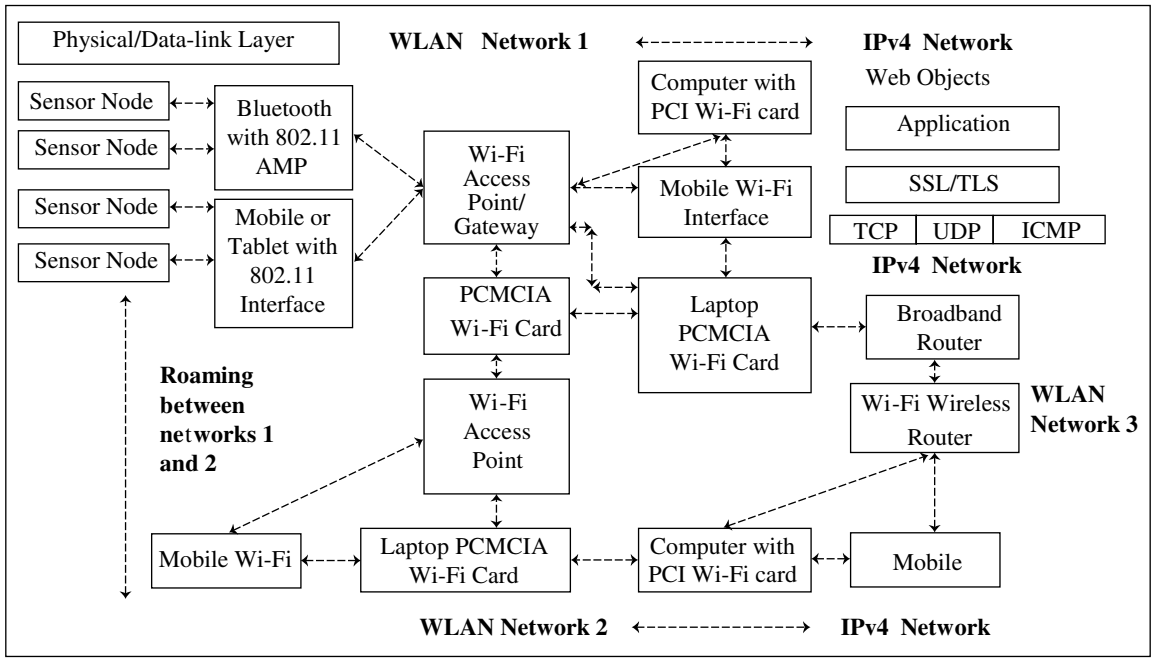


Figure 2.6 Three WLAN networks for sensor device nodes, mobiles, tablets, laptops, computers and Internet connectivity of WLAN networks with the IP4 networks (Dashed lines show wireless connectivity and solid lines show wired connectivity)

- Have easy installation simplicity and flexibility
- Coverage range is 30 m to 125 m
- Used in a room having the limited-coverage 802.11a which coexists with b, coexists with b and g
- Uses the 802.11b in wider coverage range because that is unaffected by walls and is meant for hotspots for public usage having range data rate 11 Mbps (802.11b) within 30 m
- Uses 802.11g for high data rates up to 54 Mbps, and 802.11n for very high 600 Mbps, using multiple antennas to increase data rates
- Interoperable with wireless as well as wired infrastructure which ensures compatibility and enables easier access and hides complexity when enabling the wireless access to data, media and streams, and applications and services.
- Provides a dynamic environment of network expendability and scalability. Scalability means a system can have a large number of smaller interfaces, routers and APs
- Provides security, integrity and reliability
- Uses Wireless Protected Access (WPA) and Wired Equivalent Privacy (WEP) security sublayers

RF Transceivers and RF Modules

RF transmitters, receivers, and transceivers are the simplest RF circuits. A transceiver transmits the RF from one end and receives the RF from the other end, but internally has

an additional circuit, which separates the signals from both ends. An oscillator generates RF pulses of required active duty cycle and connects to a transmitter. BT, ZigBee, and Wi-Fi radios deploy ISM band transceivers, which have comparatively complex circuits.

IoT/M2M applications deploy ISM band RF modules with transceivers or just transmitter or receiver. A number of systems use RF modules for applications needing wireless connectivity; for example, security, telemetry, telematics, fleet management, home automation, healthcare, automobiles wireless tire pressure monitors, back-up cameras and GPS navigation service, payment wallet, RFID and maintenance.

RF technology consists of the following elements:

- RF interface/physical layer, RF signals transmit between the nodes or endpoints, i.e. the sensors, actuators, controllers and a gateway where signals are received. Physical layer specifications consist of signal aspects and characteristics, including the frequencies, modulation format, power levels, the transmitting and receiving mode and signalling between end-point elements.
- RF network architecture includes the overall system architecture, backhaul, server and bidirectional end-devices with radio duty cycling in the applications. Radio duty cycling means managing the active intervals, transmission and receiving schedule, and time-intervals actions on an event during the active intervals and actions during the inactive (sleep) intervals using the RF Integrated Circuits (RFICs).

GPRS/GSM Cellular Networks-Mobile Internet

An IoT/M2M communication gateway can access a Wireless Wide Area Network (WWAN). The network access may use a GPRS cellular network or new generation cellular network for Internet access.

A mobile phone provisions for a USB wired port, BT and Wi-Fi connectivity. Wireless connectivity for Internet uses data connectivity using GSM, GPRS, UMTS/LTE and WiMax services of a mobile service provider or Wi-Fi using a modem. A phone, generally, provisions for number of sensors also; for example, acceleration, GPS and proximity.

Wireless USB

Wireless USB is a wireless extension of USB 2.0 and it operates at ultra-wide band (UWB) 5.1 GHz to 10.6 GHz frequencies. It is for short-range personal area network (high speed 480 Mbps 3 m or 110 Mbps 10 m channel). FCC recommends a host wire adapter (HWA) and a device wire adapter (DWA), which provides wireless USB solution. Wireless USB also supports dual-role devices (DRDs). A device can be a USB device as well as limited capability host.

2.3.2 Wired Communication Technology

Wired communication can be serial asynchronous communication (for example, UART interface) or synchronous serial communication (for example, SPI interface or parallel

input, output and input-output ports at the devices). Communication can be over a bus when a number of systems (chips, units, integrated circuits or ports or interfacing circuits) connect through a common set of interconnections.

Bus refers to a number of systems connected through a common set of control, address and data signals such that the data signals are accepted by the device at destination address only from a source at an instance. Bus signals may be sent in a serial or parallel manner. Communication is between a master at a given instance (sender) and the destined address computer (listener) at an instance.

Wired communication can be done using Ethernet IEEE 802.2 bus specifications. A MAC sublayer data frame may be according to Ethernet Protocol. Wired communication can also use a USB port, a microUSB or a USB 3.0 adapter.

UART/USART Serial Communication

A Universal Asynchronous Transmitter (UART) enables serial communication (transmission) of 8 bits serially with a start bit at the start of transmission of a byte on serial Transmitter Data (TxD) output line. Serial means present one after another at successive time intervals.

Asynchronous refers to all bytes in a frame transmit, which can result in variation in time interval spacing or phase differences between successive bytes and in-between wait interval. This is because clock information of transmitter does not transmit along with the data. The receiver clock also does not synchronise with the data. Further, successive set of bytes may wait after transmission till an acknowledgement is received from the receiving end.

The intervals of each transmit bit are controlled by a clock. When the clock period is $T = 0.01$ μs, then period for transmission of a byte on TxD = 10 T = 0.1 μs. A byte's transfer rate is 1 MBps. Reciprocal of T in UART is called Baud rate. When an additional bit appends between stop bit and last bit of the byte, then $T = 0.11$ μs. An additional bit can be used to identify the received byte on the serial line as the address or data. It can be used for error detection. UART receives the byte at RxD (transmitter data) input line.

An Universal Synchronous Asynchronous Transmitter (USART) enables serial communication (transmission) in synchronous as well as asynchronous modes.

Synchronous means all bytes in a frame transmit with equal time spacing or equal phase differences.

Serial Peripheral Interface

Serial Peripheral Interface (SPI) is one of the widely used serial synchronous communication methods. Source of serial synchronous output or input is called *master* when it also controls the synchronising clock information to the receiver. A receiver of serial synchronous input or output is called a *slave*, when along with the serial data it also receives the synchronising clock information from the master. Four sets of signals, viz., SCLK, MISO, MOSI, and SS (slave select) are used on four wires. When SS is active, then the device functions as a slave.

Master Input Slave Output (MISO) and Master Output Slave Input (MOSI) are synchronous serial bits I/Os at the master and slave and IOs are as per synchronising clock of the master SCLK.

MOSI is output from master and input at slave and SCLK (clock information or signal) is from the master to slave. Slave synchronises and receives the input bits at MOSI from the master as per the SCLK input at slave.

MISO is synchronous serial input at the master for the serial output from slave. Slave synchronises output as per SCLK of the master. Master synchronises the input as per SCLK of the master.

I2C Bus

A number of device integrated circuits for sensors, actuators, flash memory and touchscreens need data exchanges in a number of processes. ICs mutually network through a common synchronous serial bus, called inter-integrated circuit (I2C). Four potential modes of operation (viz. master transmit, master receive, slave transmit and slave receive) for I2C bus device and generally most devices have a single role and use two modes only.

The I2C was originally developed at Philips Semiconductors. There are three I2C bus standards: Industrial 100 kbps I2C, 100 kbps SM I2C and 400 kbps I2C.

I2C bus has two lines that carry the signals—one line is for the clock and one is for bidirectional data. I2C bus protocol has specific fields. Each field has a specific number of bits, sequences and time intervals between them.

Wired USB

Universal Serial Bus (USB) is for fast serial transmission and reception between the hosts, the embedded system and distributed serial devices; for example, like connecting a keyboard, printer or scanner. USB is a bus between the host system and a number of interconnected peripheral devices. Maximum 127 devices can connect with a host. USB standard provides a fast (up to 12 Mbps) as well as a low-speed (up to 1.5 Mbps) serial transmission and reception between the host and serial devices. Both the host and device can function in a system.

USB three standards are USB 1.1 (1.5 and 12 Mbps), 2.0 (mini size connector) 480 Mbps, 3.0 (micro size connector) 5 Gbps and 3.1 (super speed 10 Gbps).

Features of a USB are:

USB data format and transfer serial signals are Non Return to Zero (NRZI) and the clock is encoded by inserting synchronous code (SYNC) field before each packet. The receiver synchronises bit recovery clock continuously. The data transfer is of four types—controlled data transfer, bulk data transfer, interrupt driven data transfer and isosynchronous transfer.

USB is a polled bus. Polling mode functions as: A host controller regularly polls the presence of a device as scheduled by the software. It sends a token packet. The token consists of fields for type, direction, USB device address and device end-point number.

The device does handshaking through a handshake packet, indicating successful or unsuccessful transmission. A CRC field in a data packet permits error detection.

A USB supports three types of pipes—*Stream* with no USB-defined protocol is used when the connection is already established and the data flow starts. *Default control* is for providing access. *Message* is for control functions of the device. The host configures each pipe with the data bandwidth to be used, transfer service type and buffer sizes.

Ethernet

Ethernet standard is IEEE 802.2 (ISO 8802.2) protocol for local area network of computers, workstations and device LANs. Each frame at a LAN consists of header. Ethernet enables the services of local device nodes, computers, systems and local resources, such as printers, hard disk space, software and data.

Features of Ethernet network are:

- Uses passive broadcast medium and is wired connections based
- Formatting of frame (serially sent bits as PDU of MAC layer) is according to IEEE 802.2 standard
- Uses a 48-bit MAC address assigned distinctly to each computer on the LAN
- Address Resolution Protocol (ARP) resolves a 32 bit IP address at Internet device LANs. Each frame at a LAN destination host media address. Reverse Address Resolution Protocol (RARP) resolves 48 bit destination host media address into 32 bit IP addresses for Internet communication.
- Uses wired bus topology, and transmission speeds are 10 Mbps, 100 Mbps (unshielded and shielded wires), 1 Gbps (high-quality coaxial cable), 4 Gbps (in twisted pair wiring mode) and 10 Gbps (fiber-optic cables).
- Uses MAC-based on CSMA/CD (Carrier Sense Multiple Access with Collision Detection). The CSMA/CD mode is half-duplex (wired mode) which means transmit (Tx) and receive (Rx) signals can be sent on the same wire or data path. Each one connected to a common communication channel in the network listens and if the channel is idle then transmits. If not idle, it waits and tries again. A mode is full-duplex in case of optical-fibre-based Ethernet that transmits and receives signals separated on dedicated, one-way channels.
- Uses transmission data stack into frames at MAC layer, and each frame includes a header. The header's first eight bytes specify a preamble. The preamble is for indicating start of a frame and is used for synchronisation. Then the header has next six bytes' destination address and then six bytes of the source address. Then next six bytes are for the type field. These are meaningful only for the higher network layers and define the length of the data stack to the next layer. Next, minimum 72 bytes and maximum 1500 bytes of data follow the length definition. The last 4 bytes are for CRC (Cyclic Redundancy Check) for the frame sequence check.

2.3.3 Communication Technologies—A Comparison

Table 2.1 shows a comparison of communication technologies.

Table 2.1 Differences between NFC, BT LE, ZigBee and WLAN protocols

Property	NFC	BT LE	ZigBee IP	WLAN 802.11
IEEE Protocol		802.15.1	802.15.4	802.11z
Physical Layer	848, 424, 212, 106 kbps	2.4 GHz (LE-DSSS)	2.4 GHz or 915 MHz, 868 MHz and 433 MHz DSSS MAC layer CSMA/CA	2.4 GHz Two PHY layers MAC layer CSMA/CD
Data Transfer Rate	106 kbps	1 Mbps	250 kbps (2.4 GHz, 40 kbps 915 MHz, 20 kbps 868.3 MHz	11 Mbps/54 Mbps
Form Factor and Range	10–20 cm	Small	Small 10 m to 200 m	Bigger
Protocol Stack		Small in LE	127 B	Bigger than WPAN devices
Power Dissipation	Very low	Lower than ZigBee, much lower than WLAN 802.11	2 mW Router and 0.1 mW for end-device Much lower than WLAN 802.11	Much Higher than ZigBee
Set up/ Connection/ Disconnection Intervals	0.1 s	3s Connection time < 3 ms	20 ms Connection time < 10 ms	
Security	—	AES-CCM-128	AES-CCM-128	WEP
Applications	Payment wallet, short distance communi-cation	WPAN, IoT/M2M devices, widely present in mobiles and tablets and, need addition circuit in sensors, actuators, controllers and IoT devices	WPAN, wider presence in sensors, actuators, controllers, automobile and medical electronic and IoT devices connectivity using IPv6, 6loWPAN, ROLL, RPL and TLSv1.2	WLAN and WWAN network tablet, desktops, mobiles, devices with PCMCIA interface, home networking, Easy IPv4 connectivity
Network	Point to point between active and passive devices	Star topology, peer-to-peer piconet expended by inter-piconets data transactions and synchronisation	Low power, mesh or peer-to peer star networks using end devices, coordinator, router, ZigBee IP border router	LAN topology IBSS, BSS and distributed BSSs for WWLAN widely used for Internet connectivity of mobiles, tablets, desktops

Network Characteristics	P2P mode, card emulation mode and reader mode Passive neighbour activation	Self-configuring, self-healing, self-discovery	Self-configuring, self-healing, self-discovery	Scalable, interoperability, security, integrity and reliability
Broadcast/ Multicast/Unicast	Unicast	Unicast	Unicast/ multicast	Unicast

Reconfirm Your Understanding

- IoT/M2M devices, RFIDs, sensors, actuators and controllers send mostly 10s of bytes of data.
- NFC, BT low energy, ZigBee IP, ZigBee NAN, RF transceivers and modules and mobile GPRS/GSM are protocols which can be used for physical transmission of bits from the devices (sensors/actuators/RF devices/IEEE 802.15.4 devices/controllers/nodes) or reception at them. The devices communicate mostly 100s of bytes of data at an instance.
- NFC device range is 10 to 20 cm with data transfer rates as 106 kbps, 212 kbps, 424 kbps and 848 kbps
- NFC device is able to receive and pass data to Bluetooth connection and WLAN 802.11 for information handover functions
- RFID is a device which an access point identifies by charging the RFID circuit
- BT LE communicates using 802.15.1 protocol with low power dissipation and short frames in peer-to-peer star topology network and data transfer rates is 1 Mbps.
- ZigBee IP communicates using 802.15.4 protocol in low power dissipation and 127 B data frames, and communicates on Internet using 6LoWPAN, IPv6, RPL/ND and TLSv1.2. ZigBee data transfer rates are 250 kbps, 40 kbps and 20 kbps.
- Wi-Fi network uses WLAN IEEE 802.11 protocol. 802.11 a, b, g and n are popularly used.
- Wi-Fi network functions with high data transfer rates, mobility, flexible, reliability and scalable network. Interoperability with wireless as well as wired infrastructure ensures compatibility, enables easier access and hides complexity when enabling the wireless access to data, media and streams, applications and services.
- Wireless USB uses UWB (ultra wide band) 5.1 GHz to 10.6 GHz frequencies for short-range personal area network (high speed 480 Mbps 3 m or 110 Mbps 10 m channel USB protocol for wireless communication).
- UART, SPI, I2C, Ethernet and USB are bus-topology-based serial communication protocols and use wired communication.

Self-Assessment Exercise

1. List the characteristics, range, data transfer rates, three operational modes and two communication modes of NFC communication. ★
2. Draw a diagram of RFID devices' communication to access point. ★
3. List the characteristics of BT LE device communication: range, data transfer rates, dynamic pairing, IPv6 connectivity, latency, security modes, adaptation layer functions and Wi-Fi connectivity. ★★
4. List communication characteristics: PDU, range, data transfer rates, dynamic pairing and mesh network, IPv6 connectivity, latency, RFD, network integration and security modes of ZigBee IP device. ★★
5. How do the BT LE and ZigBee IP consume less energy compared to their classic versions? ★★
6. How do the WLAN IEE 802.11 a, b, g and n differ? ★
7. List Wi-Fi interface range, data transfer rates, security and flexibility. ★
8. Explain when each of the following will be used: NFC, BT LE, ZigBee IP and Wi-Fi. ★★★
9. What are the elements required for RF technology usage in devices? ★
10. Compare different USB versions. ★★
11. How does the functioning of UART, SPI and I2C differ? ★★★

2.4 DATA ENRICHMENT, DATA CONSOLIDATION AND DEVICE MANAGEMENT AT GATEWAY

LO 2.3 List functions data-adaptation layer, and the device and gateway domain

A gateway at a data-adaptation layer has several functions. These are *data privacy*, *data security*, *data enrichment*, *data consolidation, transformation and device management.* Figure 2.7 shows IoT or M2M gateway consisting of data enrichment, consolidation and device management, and communication frameworks (Figure 1.3).

Recall ITU-T reference model (Figure 2.2). The model's lowest layer is the device layer. This layer has *device and gateway capabilities.* Also, recall device and gateway domain (Figure 2.3) in ETSI IoT architecture. The domain consists of a gateway between M2M area network and CoRE and access network. A gateway consists of the data enrichment, consolidation and IoT communication frameworks. The communication gateway enables the devices to communicate and network with the web. The communication gateway (Section 3.3.1) uses message transport protocols and web communication protocols for the Internet. Chapter 3 describes these protocols.

The gateway includes two functions viz. data management and consolidation, and connected device management.

The following subsections describe the framework for data enrichment and consolidation.

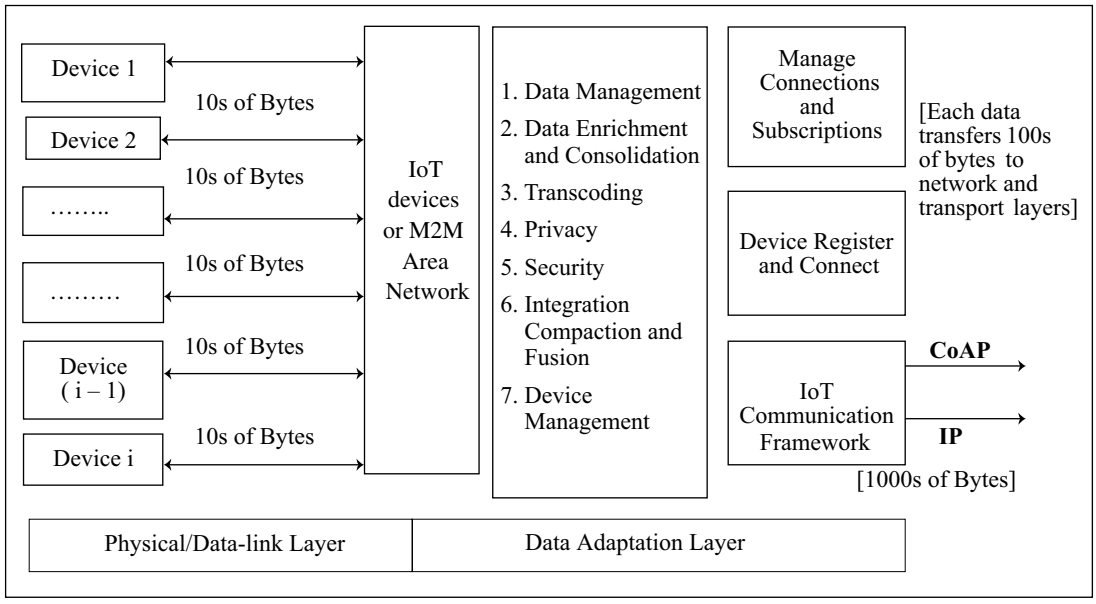


Figure 2.7 IoT or M2M gateway consisting of data enrichment and consolidation, device management and communication frameworks at the adaptation layer

2.4.1 Data Management and Consolidation Gateway

Gateway includes the provisions for one or more of the following functions: transcoding and data management. Following are data management and consolidation functions:

- Transcoding
- Privacy, security
- Integration
- Compaction and fusion

Transcoding does the data adaptation, conversion, and change of protocol, format or code

Transcoding

Transcoding means data adaptation, conversion and change of protocol, format or code using software. The gateway renders the web response and messages in formats and representations required and acceptable at an IoT device. Similarly, the IoT device requests are adapted, converted and changed into required formats acceptable at the server by the transcoding software.

For example, use of transcoding enables the message request characters to be in ASCII format at the device and in Unicode at the server. It also enables the use of XML format database at the device, while the server has a DB2, Oracle or any other database.

Transcoding involves formats, data and code conversion from one end to another when the multimedia data is transferred from a server to the mobile TV, Internet TV, VoIP

phone or smartphone as the client devices. Transcoding applications also involve filtering, compression or decompression.

A transcoding proxy can execute itself on the client system or the application server. A transcoding proxy has conversion, computational and analysing capabilities, while a gateway has conversion and computational capabilities only.

Privacy

Data such as patient medical data, data for supplying goods in a company from and to different locations, and changes in inventories, may need privacy and protection from conscious or unconscious transfer to untrustworthy destinations using the Internet.

Privacy is an aspect of data management and must be remembered while designing an application. The design should ensure privacy by ensuring that the data at the receiving end is considered anonymous from an individual or company.

Following are the components of the privacy model:

- Devices and applications identity-management
- Authentication
- Authorisation
- Trust
- Reputation

A suitable encryption of identification of data source enforces privacy. Device ID management provides for privacy. The analysed decrypted data is an input to application, service or process. IoT or M2M data have to be for the beneficiary individual person or company only.

When data is transfered from one point to another, it should be ensured that the stakeholder in future may not misuse the device end data or the application data. These static and dynamic relationships are components which depend on trust and reputation.

Secure Data Access

Access to data needs to be secure. The design ensures the authentication of a request for data and authorisation for accessing a response or service. It may also include auditing of requests and accesses of the responses for accountability in future. Example 2.4 described how a layer provides the confidentiality and authorisation using AES-128 and CCM.

End-to-end security is another aspect while implies using a security protocol at each layer, physical, logical link and transport layers during communication at both ends in a network.

Data Gathering and Enrichment

IoT/M2M applications involve actions such as data-gathering (acquisition), validation, storage, processing, reminiscence (retention) and analysis.

Data gathering refers to data acquisition from the devices/devices network. Four modes of gathering data are:

1. *Polling* refers to the data sought from a device by addressing the device; for example, waste container filling information in a waste management system
2. *Event-based gathering* refers to the data sought from the device on an event; for example, when the device reaches near an access point or a card reaches near the card reader or an initial data exchange for the setup of peer-to-peer or master-slave connection of BT device using NFC
3. *Scheduled interval* refers to the data sought from a device at select intervals; for example, data for ambient light condition in Internet of streetlights
4. *Continuous monitoring* refers to the data sought from a device continuously; for example, data for traffic presence in a particular street ambient light condition in Internet of streetlights

Data enrichment refers to adding value, security and usability of the data.

Data Dissemination

Consider the following three steps for data enrichment before the data disseminates to the network as aggregation, compaction and fusion.

Aggregation refers to the process of joining together present and previously received data frames after removing redundant or duplicate data.

Compaction means making information short without changing the meaning or context; for example, transmitting only the incremental data so that the information sent is short.

Fusion means formatting the information received in parts through various data frames and several types of data (or data from several sources), removing redundancy in the received data and presenting the formatted information created from the information parts. Data fusion is used in cases when the individual records are not required and/or are not retrievable later.

Energy Dissipation in Data Dissemination

Energy consumption for data dissemination is an important consideration in many devices in WPANs and in wireless sensor nodes (WSNs). This is due to limited battery life. Energy is consumed when performing computations and transmissions. Higher the data rate, the greater will be the energy consumed. Higher is RF used, the greater will be the energy consumed. Higher the gathering interval, the lower will be the energy consumed.

Energy efficient computations can be done by using concepts of data aggregation, compaction and fusion. Lesser the data bytes communication, greater the acquisition intervals, and lower the data rate for data transfer, lesser the energy dissipation.

Data Source and Data Destination

ID: Each device and each device resource is assigned an ID for specifying the data of source and a separate ID for data destination.

Address: Header fields add the destination address (for example, 48-bit MAC address at Link layer, 32-bit IPv4 address at IP network and 128-bit IPv6 address at IPv6 network) and may also add the port (for example, port 80 for HTTP application).

Data Characteristics, Formats and Structures

Data characteristics can be in terms of temporal data (dependent on the time), spatial data (dependent on location), real-time data (generated continuously and acquired continuously at the same pace), real-world data (from physical world for example, traffic or streetlight, ambient condition), proprietary data (copy right data reserved for distribution to authorised enterprises) and big data (unstructured voluminous data).

Data received from the devices, formats before transmission onto Internet. The format can be in XML, JSON and TLV (Section 3.1.3). A file can be MIME type for Internet (Section 3.1.5).

Structure implies the ways for arranging the data bytes in sequences with size limit = PDU for a layer.

2.4.2 Device-management Gateway

Device Management (DM) means provisioning for the device ID or address which is distinct from other resources, device activating, configuring (managing device parameters and settings), registering, deregistering, attaching and detaching.

Device management also means accepting subscription for its resources. Device fault management means course of actions and guidelines to be followed in case if a fault develops in the device.

Open Mobile Alliance (OMA)-DM and several standards are used for device management. OMA-DM model suggests the use of a DM server which interacts with devices through a gateway in case of IoT/M2M applications. A DM server is a server for assigning the device ID or address, activating, configuring (managing device parameters and settings), subscribing to device services or opting out of device services and configuring device modes. A device instead of a DM server, communicates to a gateway in case of low-power loss environment.

Gateway functions for device management are:

- Does forwarding function when the DM server and device can interact without reformatting or structuring
- Does protocol conversion when the device and DM server use distinct protocols
- Does proxy function in case an intermediate pre-fetch is required in a lossy environment or network environment needs.

OMA DM model for using a DM server

Reconfirm Your Understanding

- Data communication between personal/local area network of devices and a gateway for communicating via Internet.
- Gateway enables data enrichment and consolidation and device management.
- Data management functions at the gateway are transcoding, data privacy, data security, data enrichment, data consolidation, transformation and device management.
- Transcoding means adaptations, conversions, changes of protocol or format using software which renders the web response/messages in formats/representations as required and acceptable at the IoT device and rendering requests for messages in formats/representations as required and acceptable at the server.
- Data acquires and transfers to other end at scheduled intervals, on an event, or on polling.
- Data aggregation, compaction and fusion save energy during data dissemination.
- Data destinations may use the 48-bit MAC address, 32-bit IPv4 address, 48-bit IPv6 address or port number during communication at the data-link or network layers.
- Each device and application has an ID or address of communication source and each destination has an ID or address. Communication between the end points and between the layers is secure when using the authentication and authorisation processes.
- Device management functions are the device ID or address, activation, configuring (managing device parameters and settings), registering, deregistering, attaching, detaching and fault management.
- Gateway functions for device management are—forwarding function between DM server and device; protocol conversion when device and DM server use distinct protocols and proxy function.
- Communication gateway enables protocol conversion between two ends.

Self-Assessment Exercise

1. List the functions at the data-adaptation layer. ★
2. What are the transcoding functions required at data-adaptation layer gateway? ★★
3. What are the functions used in designing for data privacy of the connected devices? ★★
4. What are the four modes for gathering data? ★
5. Give examples of steps of aggregation, compaction and fusion of data before dissemination of data. ★★★
6. Where is the device ID, MAC address, IP address, IPv6 address and port number used for data destination? ★★
7. List six characteristics of data of devices, which communicate to the Internet for applications, services and business processes. ★
8. How is energy saved at a device when acquiring data? ★★
9. List the functions of device management. ★
10. What are the device management functions for end device, coordinator, router and IP router when using ZigBee IP based WPAN devices? ★★★

2.5 EASE OF DESIGNING AND AFFORDABILITY

LO 2.4 Discuss ease of designing and affordability of IoT devices

Design for connected devices for IoT applications, services and business processes considers the ease in designing the devices' physical, data-link, adaption and gateway layer.

It means availability of SDKs (software development kits), prototype development boards with smart sensors, actuators, controllers and IoT devices which are low in cost and hardware which embeds and are preferably open source software components and protocols. Hardware which includes the device should embed minimum number of components and use ready solutions for ease in designing local devices personal area network and secure connectivity with the Internet.

Designing also considers ease as well as affordances for example, RFID or card. The card has an embedded microcontroller, memory, OS, NFC peripheral interfaces, access point-based device activation, RF module and transceiver at low cost.

A wireless sensor uses, for example, a mobile terminal (Mote) which is a low cost device with an open-source OS (tiny OS) and software components. Usages of Motes provide ease and affordance in a WSN network.

Devices of smart homes and cities use ZigBee IP or BT LE 4.2 (dual mode or single mode) due to their affordability, ease of designing, usage and low cost.

A design may add to the complexity. For example, consider the umbrella in Example 1.1. How will the umbrella be programmed to schedule the SMSes for a user? The need to use an instruction manual adds to the complexity of designing Internet of umbrellas.

Connected devices may add complexity in the form ensuring data transfer to trusted destinations using encryption tools.

Reconfirm Your Understanding

- Ease of designing and affordability are important considerations for devices and applications, services and processes for IoTs.

Self-Assessment Exercise

1. What do the terms 'ease' and 'affordability' mean? ★
2. How does the use of connected devices add complexity as compared to unconnected devices? ★★

Key Concepts

- AES-CCM 128 bit
- Affordability of design
- Bluetooth low energy
- Data adaptation layer
- Data aggregation
- Data compaction
- Data dissemination
- Data formatting
- Data fusion
- Data integrity
- Data privacy
- Data processing
- Data security
- Data trust
- Device configuration
- Device fault management
- Device management
- Device, and application and network domains
- DM server
- Ease of designing
- ETSI high level reference architecture
- Gateway
- GPR/GSM/UMTS/LTE
- I2C
- IETF six layer model
- ITU-T four layer reference model
- Mobile internet
- NFC
- Physical/data-link layer
- RF module
- RFID
- SPI
- Transcoding
- UART
- USB
- Wireless personal area network
- Wireless USB
- WLAN 802.11
- ZigBee IP

Learning Outcomes

LO 2.1

- IETF standards organisation proposed for the IoT/M2M, a modified Open Systems Interconnection (OSI) model, which consists of six layers, introduces physical cum data-link layer, data-adaptation layer and application-support layer.
- ITU-T standards organisation proposed four layers for IoT/M2M reference mode: Device, network, services and application-support and application layers.
- ETSI standards organisation proposes two domains for M2M: Device and application and network domains. ETSI high-level architecture provides that device-domain communicates with M2M applications and services through the CoRE network and access network.

LO 2.2

- Wireless devices physical/data-link layer protocols for IoT/M2M designing are:
  - NFC (10–20 cm; data transfer rates 106 kbps, 212 kbps, 424 kbps and 848 kbps)
  - RFIDs
  - BT LE (low power dissipation, short frames, peer-to-peer star topology network and 1 Mbps data transfer rates)
  - ZigBee IP (low power dissipation, 127 B data frames, communicate on Internet using 6LoWPAN, IPv6, RPL/ND and TLSv1.2)
  - WLAN 802.11 (Wi-Fi) (high data transfer rates, mobility, flexible, reliability and scalable network. Interoperability with wireless as well as wired infrastructure)
  - Mobile GPRS/GSM/UMTS/LTE/WiMax
  - Wireless USB (UWB (ultra wide band) for short-range personal area network, high speed wireless communication
- IoT/M2M wired devices physical layer protocols are UART, SPI, I2C, Ethernet and USB bus topology serial communication protocols.

LO 2.3

- A gateway exists between personal/local area network of devices and the Internet. Gateway enables data enrichment and consolidation and device management.
- Data management functions at the gateway are data privacy, data security, data enrichment, data consolidation, transformation and device management.
- Privacy model components are:
  - Device and applications identities-management
  - Authentication
  - Authorisation
  - Trust
  - Reputation
- A communication gateway enables protocol conversion between two domains: Devices and gateway domain, and network and application domain.
- Device management functions are assigning each device ID or address, device activation, configuring (managing device parameters and settings), registering, deregistering, attaching, detaching, subscription and fault management.
- The OMA-DM model suggests the use of a DM server interacting with devices through a gateway.

LO 2.4

- IoT applications, services and business process design consider the ease and affordability in designing the devices physical, data-link, adaptation layer, gateway and Internet connectivity. A design should add minimum complexities.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. IETF OSI six-layer model for the IoT proposes the followings layers: (i) physical layer (ii) physical/data link layer (iii) logical link layer (iv) adaptation layer (v) gateway layer (vi) network layer (vii) UDP layer (viii) transport layer (ix) application layer as highest layer and then application-support layer (x) application-support layer as the highest layer and then application layer. ★★★
   (a) All layers except (i), (iii), (v), (vii) and (x) true
   (b) All layers except (ii), (v), (vii) and (ix) true
   (c) All layers except (ii), (iii), (iv), (viii) and (ix) true
   (d) All layers except (ii), (iv), (v), (vii) and (ix) true
2. The ITU-T reference model proposes (i) four layers (ii) lowest layer as the device layer has device and gateway capabilities, (iii) next layer to the device layer has transport and network capabilities, (iv) the next layer above is the services and application-support layer, and (v) generic and specific service capabilities layer. ★★
   (a) (i) is incorrect
   (b) Model layers are (ii) to (v) and (i) is true

(c) Layers are (ii), (iii) and , (iv) and applications and services layer, and (v) is not a separate layer
(d) (ii) to (v) incorrect

3. The ETSI high-level architecture proposes: ★
   (a) a device and gateway domain and the gateway has M2M service capabilities
   (b) a device and gateway domain, and the device domain has an M2M area network and M2M devices
   (c) a device, application and services domain which has M2M service capabilities, applications and M2M Devices
   (d) a device and gateway domain, and the gateway has M2M service capabilities, applications, M2M area network and M2M devices
4. NFC devices function as follows: (i) communicate within distances up to 10 m to 20 m, (ii) NFC device can also communicate with BT and Wi-Fi (iii) NFC device is able to receive and pass the data to a BT connection or a standardised LAN or Wi-Fi using information handover functions, (iv) communicate with data transfer rates 128 kbps, 512 kbps and 640 kbps. ★
   (a) (i) and (iv) true
   (b) (i) and (iv) not true
   (c) (i),(ii) and (iv) true
   (d) All are true
5. Bluetooth v4.2 LE mode is (i) IETE 802.15.4 protocol, (ii) supports 1 Mbps data rate, (iii) support to NFC pairing for low latency in pairing the BT devices, (iv) auto-synchronisation between mobile and other devices when both use BT (iv) range depends on radio class and can be 100 m, 10 m or 1 m used in device BT implementation, (v) forms mesh topology network (vi) frame size from device is 127 B (vi) inter-piconet communication between piconets to get bigger range network of 100-150 cm range (called scatter net) **[LO 2.2]** ★★
   (a) (iii) and (vi) not true
   (b) All are true
   (c) (ii), (v) and (vi) not true
   (d) (i), (v) and (vi) not true
6. ZigBee IP device communicate (i) using IETE 802.15.4 protocol, (ii) supports 2.4 GHz and different country specific ISM frequency bands, (iii) support to data routing and IP border routing, (iv) support IPv6 network with 6LoWPAN header, (v) forms mesh topology network using ZigBee router (vi) using end device and five other modes of networking (viii) provides link level security using AES-CCM-128. ★
   (a) All are true
   (b) All except (ii) and (v) correct
   (c) All except (iii)
   (d) All except (iii) true
7. Wi-Fi network (i) use generally 2.4 GHz or 915 GHz (ii) 802.11a or 802.11g or 802.11n or other 802.11 series protocols, (iii) mobility and roaming, (iv) has easy installation simplicity and flexibility, (v) coverage ranges 300 m to 425 m and data rate is 11 Mbps within 300 m (vi) scalability ★★
   (a) All are true
   (b) All except (ii) and (v) correct

(c) All except (i) and (v)
(d) All except (iii) and (vi) true

8. Asynchronous communication characteristics are each set of bytes that (i) can have variation in time interval spacing or phase differences between successive bytes and in-between wait interval, (ii) should have fixed time interval spacing or phase difference, (iii) Tx and Rx does not synchronise their internal clocks at transmitter and receiver, (iv) Rx clock frequency can be higher or lower than that of Tx during serial communication, (v) Tx end can wait for acknowledgement or response from Rx end before a new set of bytes transmit. All are true: ★★
   (a) except (ii) and (iv)
   (b) except (ii) and (v)
   (c) except (i) and (iii)
   (d) except (iii) and (vi)
9. SPI (i) at the device uses serial synchronous communication to another device, (ii) one of the device function as a master at an instant, (iii) master means one which synchronises its clock with another device, (iv) slave is a receiver of the serial synchronous output and synchronizing clock information from master, (v) two set of signals are MISO and MOSI. All are true: ★★★
   (a) except (iii)
   (b) (i) and (v)
   (c) except (iii) and (iv)
   (d) except (ii)
10. Ethernet (i) uses 48-bit MAC addresses assigned distinctly to each computer network card on the LAN, (ii) uses reverse Address Resolution Protocol (RARP) to MAC address into 32 bit IP addresses for Internet, (iii) uses wired bus topology (iv) data rates are 10/100 Mbps, 1 Gbps (high-quality coaxial cable), (v) uses MAC based on CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). All are true: ★★
   (a) none is false
   (b) all except (i) and (iv) correct
   (c) except (v)
   (d) except (ii) and (v) true
11. USB (i) host connects up to 256 devices, (ii) has no provisions for the polling mode functions, (iii) three standards are USB 1.1 (1.5 and 12 Mbps), 2.0 (mini size connector) 480 Mbps, 3.0 (micro size connector) 10 Gbps and 3.1 (wireless 10 Gbps), (iv) communication is asynchronous serial, (v) device is a plug and play device. All are incorrect: ★★★
   (a) except (ii)
   (b) except (ii) and (v)
   (c) except (v)
   (d) except (iii) and (iv)
12. Transcoding means one or more of the following functions: (i) adaptations, (ii) conversions, (iii) changes using software which render the web response and messages in formats and representations required and acceptable at an IoT device, (iv) adapting IoT device requests and convert and change into required formats acceptable at the server by transcoding software. All are true: ★★

(a) except (i) and (iv)
(b) none false
(c) except (iii) and (iv)
(d) except (ii)

13. Device management functions include (i) assigning a device ID or address, (ii) activation, (iii) proxy setting, (iv) gateway actions, (v) configuring, (vi) changing device parameters, (vii) adapting device parameters to permitted values, (viii) registering, (ix) deregistering, (x) attaching, (xi) detaching ★★
    (a) All are true except (vi)
    (b) (iii), (iv) and (vi) not true
    (c) All except (x) and (xi) true
    (d) All except (iv) and (vi) true
14. IoT data characteristics can be (i) temporal data, (ii) spatial data, (iii) real-time data, (iv) real-world data, (v) proprietary data, (vi) big data, (vii) random data ★★
    (a) All are true
    (b) All except (v) are true
    (c) All except (i) and (vii) are true
    (d) All except (vii) are true
15. Suggested privacy model depends on the following components: (i) trust, (ii) device and applications identities-management, (iii) authentication, (iv) authorisation, (vi) reputation ★
    (a) All except (vi) are true
    (b) All except (ii)are true
    (c) All except (i) and (vi) are true
    (d) All true
16. Data enrichment before the data disseminates to the network means (i) aggregation, (ii) compaction, (iii) fusion, (iv) adjusting data rate within the permitted limit, (v) communication as per protocol standards, (vi) formatting as per protocol specifications ★★
    (a) All are true
    (b) All except (iv) and (v) are true
    (c) All except (iii) and (v) are true
    (d) (i) to (iii) true

Short-Answer Questions

1. What are the modifications in OSI model for IoT? **[LO 2.1]** ★
2. Show a comparison between OSI layers and IUT-T reference model layers. **[LO 2.1]** ★★
3. What are the features of ETSI high-level architecture? **[LO 2.1]** ★★
4. Explain meaning of active, passive and card devices. **[LO 2.2]** ★★★
5. How does a passive device communicate on the Internet? **[LO 2.2]** ★★
6. What are the WPAN and network protocol functions which can be used for Internet of RFIDs? **[LO 2.2]** ★★★
7. Compare the features of BT LE device with BT BR/EDR device mode? **[LO 2.2]** ★★
8. Write the modes and data rates of ZigBee IP devices. **[LO 2.2]** ★
9. How is the ZigBee IP used in Internet of streetlights? **[LO 2.2]** ★★★
10. How does the local device M2M network connect to WiFi and the Internet? **[LO 2.2]** ★

11. What are the features of serial synchronous and serial asynchronous communication in a wired device network? **[LO 2.2]** ★★
12. Why is data adaptation layer must for connected devices for an IoT? **[LO 2.3]** ★★★
13. Why is a gateway needed for networking? **[LO 2.3]** ★
14. How does a chain of ATMs connect to a bank server? **[LO 2.3]** ★★★
15. Why are both data privacy components and provisioning of end-to-end security important in IoT? **[LO 2.3]** ★
16. How will you design a lamppost WSN with ease and affordability for Internet of streetlights services (Example 1.2)? **[LO 2.4]** ★★

Review Questions

1. Describe IETF proposed six-layer model for IoT. How does the IoT model relate to the OSI seven-layer model for computer networks? [**LO 2.1**] ★★
2. Describe four layers of reference model and the layer capabilities. [**LO 2.1**] ★
3. Draw ETSI M2M domains and high-level architecture. List the capabilities and functions of each domain. [**LO 2.1**] ★
4. What are the features of NFC? How does a device using NFC transfer information to Bluetooth or Wi-Fi interfaces and to the Internet? [**LO 2.2**] ★★★
5. Describe and list the protocol features in Bluetooth v 4.2 BR/EDR and low-energy modes. How does a BT device connect to Wi-Fi? **[LO 2.2]** ★★
6. How do ZigBee end point devices form a WPAN of embedded sensors, actuators, appliances, controllers or medical data systems, and how do they connect to application layer services, business processes and service? [**LO 2.2**] ★★★
7. How do Wi-Fi interfaces connect within themselves or to an access point or wireless router? How does Wi-Fi enable interconnecting through a modem and then to the Internet service provider? [**LO 2.2**] ★★★
8. Explain RF circuits, the actions using active duty cycle and modulation and transceiver actions. How does an RF circuit connect to Bluetooth, ZigBee or Wi-Fi radios using ISM band transceivers? [**LO 2.2**] ★★
9. Describe protocol layers of BT LE and ZigBee IP. [**LO 2.2**] ★
10. Explain RF circuit components. What are the actions during active duty cycle, and by modulator and transceiver? [**LO 2.2**] ★★
11. What are the features of Ethernet IEEE 802.2 that enable wired LAN communication between the computers? ★★★
12. Explain UART and I2C bus interfaces. When and where are these interfaces used? [**LO 2.2**] ★
13. Compare I2C, SPI bus topologies. When and where are these interfaces used? [**LO 2.2**] ★★★
14. Describe adaptation layer gateway, data enrichment, data consolidation and device management functions. **[LO 2.3]** ★★
15. Why are device management functions required? **[LO 2.3]** ★
16. Explain how to design with ease and affordability for local area network of M2M devices. **[LO 2.4]** ★★

Practice Exercises

1. Show three conceptual designs using IETF model layers, ITU-T reference model layers and ETSI domains high-level reference architecture for Internet of ATMs applications and services. **[LO 2.1]** ★★★
2. Show three conceptual designs using IETF model layers, ITU-T reference model layers and ETSI domains high-level reference architecture for a waste container management service in a smart city. **[LO 2.1]** ★★★
3. Draw and show how a Bluetooth network will connect for Internet applications and services. **[LO 2.2]** ★
4. Draw and show how a cellular network will connect to a ZigBee mesh network. **[LO 2.2]** ★★
5. How is mobile Internet used in IoT and M2M applications? **[LO 2.2]** ★★
6. Make charts to show connected devices using NFC, BL LE, ZigBee IP for IoT applications and services? **[LO 2.2]** ★
7. How will a network be used with features of privacy, security, data formatting, data destinations and source addresses, aggregation, data fusion, data compaction and minimum energy dissipation? **[LO 2.3]** ★★★
8. Find the ease and affordable hardware for streetlight lamppost connectivity with neighbouring lamppost. Hardware uses the sensors for ambient light, faulty lighting function and traffic, microcontroller, memory, BL LE connectivity and data adaption. When connecting to a group controller, what will be the addition hardware and software? **[LO 2.4]** ★★★

3

Design Principles for Web Connectivity

Learning Objectives

LO 3.1 Extend the understanding of web communication protocols used by connected IoT/M2M devices

LO 3.2 Illustrate the usage of messaging protocols between connected devices and the web

LO 3.3 Outline the usages of communication gateway protocols, such as SOAP, REST, RESTful HTTP and WebSocket by connected devices and the web

Recall from Previous Chapters

Section 1.1 described the suggested conceptual frameworks, reference models and architectures. Equation 1.1 of Adrian McEwen and Hakim Cassimally conceptualises the IoT framework and is given as "Physical Object + Controller, Sensor and Actuators + Internet = Internet of Things". Equation 1.2 conceptualises the Oracle IoT architecture and is given as "Gather + Enrich + Stream + Manage + Acquire + Organise and Analyse = Internet of Things with connectivity to data centre, enterprise or cloud server." Equation 1.1 tells that the Internet is an important element for IoT/M2M. Equation 2.2 tells that streaming of data is a must-required element. The Internet connects to the web. Streaming of data is to or from the web. IoT/M2M physical objects/devices therefore need web connectivity.

Section 2.1 described the network layer as one of the six layers in the IoT reference model. The ITU-T reference model has one layer for network capabilities. ETSI suggested one of the two domains in high-level architecture as a network domain. The domain includes CoRE and access

networks, management and service capabilities. Gateway connects to the network using which data streams to or from the web. An IoT/M2M gateway needs connectivity to the web that uses the Internet.

3.1 INTRODUCTION

An IoT/M2M device network gateway needs connectivity to web servers. A communication gateway enables web connectivity, while IoT/M2M specific protocols and methods enable web connectivity for a connected devices network. A server enables IoT device data accumulation (storage). Application (reporting, analysis and control), collaboration, service and processes (involving peoples and business processes) use this data.

Following are some key terms, which need to be understood for learning web connectivity and communication between the connected devices network and the web for IoT:

Application or App refers to a software for applications such as creating and sending an SMS, measuring and sending the measured data, receiving a message from a specific sender etc.

Application Programming Interface (API) refers to a software component, which receives messages from one end; for example, from an application or client or input. An API may consist of GUIs (button, check box, text box, dialog box). An API may get input to or from a server or a user. It then initiates actions and may send the messages, for example, to application software, server or a client at the other end etc.

For example, consider an API for a parcel tracking application. The API displays the fields for the user inputs required for tracking, accepts the user inputs, displays a 'wait message' to the user and sends the input parameters to an application at a server. The application in turn displays the response from the application or server.

An API also refers to software components, which enable easier development of an application. An API defines the functionalities for a programmer, which puts the building blocks together to develop an application. A building block consists of the operations, inputs, outputs and underlying types.

Web service refers to a servicing software which uses web protocols, web objects or WebSockets; for example, weather reports service, traffic density reports, streetlights monitoring and controlling service.

Object refers to a collection of resources; for example, collection of data and methods (or functions or procedures) to operate on that data. Take for instance, Time_Date object with second, minutes, hour, day, month and year fields and update methods. An *object instance* can be just one or more than one for an object. An example of an object instance is birth_date. Multiple object instances, abc_birth_date, pqr_birth_date, xyz_birth_date and many instances can be created from birth_date object in JavaScript. An example of object instance is weather report object for reporting the rains.

Object model is defined as the usage of objects for values, messages, data or resource transfer, and creation of one or more object-instances.

Class: Java uses concept of *class*, which creates one or more object instances.

Communication gateway is one that functions as communication protocol translator (convertor) for provisioning communication capabilities. For example, the gateway for communication between ZigBee and IP networks.

Client refers to a software object which makes request (or an API associated with it makes request) for data, messages, resources or objects. A client can have one or more object instances. A client may also have an API or many APIs for enabling the communication to the server. A client can be at a device or application on a network or Internet connected web, enterprise server or cloud.

Server is defined as a software which sends a response on a request. The server also sends messages, alerts or notifications. The server has access to resources, databases and objects. A server can be on a device or can be on a separate computer system, not necessarily on Internet connected web.

Web object is the one that retrieves a resource from the web object at other end using a web protocol.

Broker denotes an object, which arranges the communication between two ends; for example, between the message publisher and subscriber or for example taking the request from a source and sending the response received back for that source after arranging the response from another object, such as a server.

Proxy refers to an application which receives a response from the server for usage of a client or application and which also receives requests from the client for the responses retrieved or saved at proxy.

Communication protocol defines the rules and conventions for communication between networked devices and between systems. The protocol includes mechanisms for devices or systems to identify and make connections with each other. The protocol also includes formatting rules that specify how data is packaged into sent and received messages,[1] and header, its field and their meanings.

Web protocol is a protocol that defines the rules and conventions for communication between the web server and web clients. It is a protocol for web connectivity of web objects, clients, servers and intermediate servers or firewalls. It includes mechanisms for a web object to identify and make connections with other objects. The protocol also includes web object formatting rules that specify how that object packs into it the sent and received messages.

Firewall is one that protects the server from unauthentic resources.

A *header* consists of a set of words. The words contain the information and parameters about the processing at a communicating layer. The words are placed over a data stack

[1] http://compnetworking.about.com/

received from a previous layer and create a new data stack for transmission to next lower layer. Header words are placed at the sending layer lower in the hierarchy during transmission and removed at the receiving layer higher in hierarchy after using the information contained in the header.

A *state* refers to an aspect related to someone or something, or a form at a particular time (Collins English Language Dictionary). State, in reference to data interchange between web objects, refers to an aspect related to data or resource or object received at a particular instance at the server or client end.

Resource denotes something that can be read (used), written (created or changed) or executed (processed). A path specification is also a resource. The resource is atomic (not-further divisible) information, which is usable during computations. A resource may have multiple instances or just a single instance. Single instance example is a contact list. Examples of multiple instances of a contact are name, surname, city, address and email ID.

Each *resource instance* has a *resource ID*. A resource is accessible using a resource identifier. A resource identifier can be a path specification, such as a Universal Resource Locator (URL) or Universal Resource Identifier (URI).

A resource may have *resource registry*, which means it is usable after a resource instance is registered at the registry, and unusable when not-registered or when de-registered.

A resource can be updated. A resource can be used when an application uses *resource discovery*.

A resource may have a *resource repository*. Repository refers to subfolder, folder or directory, which contains the resources and their instances. A resource can have a resource directory, which directs to resource repository.

Resource structure or *resource directory* refers to a structure or collection of resources, applications, containers and groups, which may each have an attribute, access right, subscription and discovery.

Path denotes a navigation path between two ends when accessing a resource. Path specification can be of URI or URL type. The structure of URI is hierarchical. Entity after the sign '/' is a child of the parent before the sign '/'.

Universal Resource Identifier is generally used for saved resources, such as contacts or address book. Example of a URI is /Contacts/First_Character_R/ for a set of resource directory contacts having resource repository First_Character_R for contacts with first character *R* and resources giving information about a contact. Another example is URI sensorNetwork_J/sensorID_N/temperature for a temperature value. The value is at a resource directory sensorNetwork_J for a sensor network, which stores sensor data for a sensor of id sensorID_N.

Universal Resource Locator is generally used for retrieving a resource(s) by a client. The saved resources may be at a document or at a remote server accessed using Internet protocols. An example of a URL is http://www.mhhe.com/ for a set of resource directories, resource repositories and resources on the McGraw Hill Higher Education server.

Datagram refers to a limited size data (2^{16} byte). It is used for stateless connectionless transfer from a web object. Stateless means each single datagram transfer is independent of previous data interchanges and connectionless means there is no need of pre-establishing the connection for resource exchanges between the web objects and no connection closing after finishing the resource exchanges.

Representational State Transfer (REST) is a software architecture referring to ways of defining the identifiers for the resources, methods, access methods and data transfer during interactions. REST is a software architecture which also specifies the practices, constraints, characteristics and guidelines for creating scalable web services. Scalable means can be used as per the size. The architecture is used during the design of web software components, clients and web APIs .

REST also refers to usage of defined resource types when transferring the objects between two ends—URIs or URLs for representations of the resources. REST also refers to the usage of use verbs (commands), POST, GET, PUT and DELETE (Section 3.4.4) and files of MIME types.

Multipurpose Internet Mail Extensions or MIME refers to the type of files that are widely used on the Internet by web objects, applications and services.

RESTful refers to one which follows REST constraints and characteristics.

User Datagram Protocol (UDP) is a protocol at the transport layer for the web using Internet and Constrained RESTful Environment (CoRE). UDP specifies the way of enveloping the datagram by header words, which just specifies the ports and addresses of two ends between which the datagram transfers. UDP is described in detail in Chapter 4, Section 4.3.3.

Hypertext means text embedded with hyperlinks. *HyperText Transfer Protocol* (HTTP) means an application layer protocol for use of hypertext as app data transfer protocol. Clients and servers use the URLs http://… .

Hyperlink refers to a specification of the URL for a resource path, so that a link can be established between two objects. For example, hyperlink URL specifications for the author's earlier book's supplementary web-resources at McGraw-Hill Higher Education server are http://www.mhhe.com/kamal/emb3. Retrieval of a resource at a web object by the other object uses the click at a link shown on a displayed text on browser.

HyperText Markup Language (HTML) is a language for creating a hypertext which refers to text that embeds text, images, audio and video, image frames, forms, lists, tables, navigation links (reference to resources), APIs, Java Script and other codes for dynamic actions.

Extensible Markup Language (XML) is a language, which enables creation, sending and receiving documents, messages, commands, query responses, queries, and creation of forms. The form data, response or message uses a new tag (not defined earlier in HTML) with new data type definitions than the ones at HTML. *HTML5* is the latest version of HTML which includes features of XML and many new features as well. A form, for

example, has a number of fields such as, Name, Phone, Address, e-Mail ID. Form data means filled contents in the fields of a form. For example, one creates a new tag, 'name', and use in the XML line as follows: <name>Raj Kamal </name>, then name field of form has data 'Raj Kamal'. Sign '/'is for ending.

Browser is a client software which displays hypertext that enables navigation to the hypertext links shown on the user screen, and which displays GUIs of the apps, display form, display server responses, and so on.

A WebSocket denotes an API for bidirectional communication that has much less header size than HTTP communication and lower latency compared to HTTP approach of unidirectional communication at an instant (Section 3.4.5).

Framework refers to provisions for a number of software libraries, and a number of APIs including those that can be selectively changed by user codes in applications. An example of a framework is .NET framework. Advantages of frameworks are: easier to work with a set of discrete objects, software components, protocols, complex and different technologies; binds these together into something highly useful; forces the programmers to implement code in a way that promotes consistent coding, thus with fewer bugs, and provisioning more flexibility in the applications; and easy in testing and debugging of the code.

The following sections describe the design principles for web connectivity of connected devices. Later in the chapters how connected devices use the CoAP, LWM2M, XML, message queues, MQTT, XMPP, SOAP, RESTful HTTP, WebSockets and other protocols for the web connectivity to services, applications and processes has been discussed.

3.2 WEB COMMUNICATION PROTOCOLS FOR CONNECTED DEVICES

LO 3.1 Extend the understanding of web communication protocols used by connected IoT/M2M devices

Data of connected devices routes over the web in two types of communication environments. The environments are:

- ***Constrained RESTful Environment (CoRE):*** IoT devices or M2M devices communicate between themselves in a Local Area Network. A device typically sends or receives 10s of bytes. The data gathered after enriching and consolidating from a number of devices consists of 100s of bytes. A gateway in the communication framework enables the data of networked devices that communicate over the Internet using the REST software architecture.
  - Devices have the constraint in the sense that their data is limited in size compared to when data interchange between web clients and web servers takes place using HTTP, Transmission Control Protocol (TCP) and Internet Protocol (IP).
  - Another constraint is data-routing when Routing Over a network of Low power and (data) Loss (ROLL). ROLL network is a wireless network with low power transceiver. Another constraint is that the devices may sleep most of the time in

low power environment and awaken on an event or when required (on client initiative). Devices' connectivity may also break for long periods, have limited up intervals in loss environment, and have limited data size.

- ***Unconstrained Environment:*** Web applications use HTTP and RESTful HTTP for web client and web server communication. A web object consists of 1000s of bytes. Data routes over IP networks for the Internet. Web applications and services use the IP and TCP protocols for Internet network and transport layers (Section 4.3).

Figure 3.1 shows device's local area network connectivity, web connectivity in constrained and unconstrained RESTful environments and the protocols for communication.

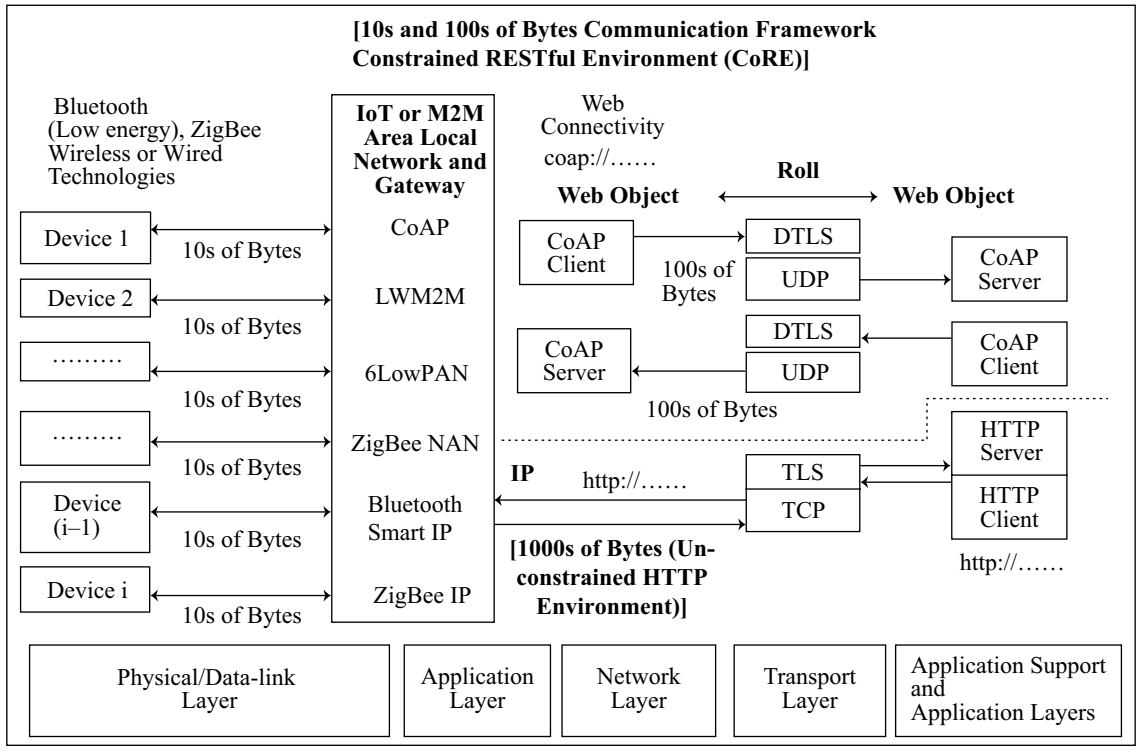


Figure 3.1 IoT or M2M devices local network connectivity and web connectivity in constrained (above thick dotted line) and unconstrained RESTful HTTP (below thick dotted line) environments using communication protocols

Assume that a web object refers to a web client or web server. The web object communicates a request of the client or a response of the server. Communication is over the ROLL network or Internet.

Web objects refer to client and server

Figure 3.1 shows the following:

- Assume i-devices (1, 2, … i^{th}) connected devices network, and local network having

connectivity between the devices at physical/data-link and adaptation layer (left-hand side)
- Communication between web objects (right-hand side)
- IETF CoRE specifications, which include CoAP and UDP
- Web objects' protocols for sending a request or response; for example, RESTful CoAP, CoAP client and CoAP server communication over the network and transport layers to other end CoAP client and server. Client/server use the URIs coap://… in place of http://… .
- Transport layer protocols used are Datagram Transport Layer Security (DTLS) and UDP. Data between web objects route using ROLL network specifications of IETF.
- 100s of bytes communicate between the IoT web objects
- Web objects HTTP client and HTTP server communication over the Internet using the IP and client/server use the URLs http://… .
- 1000s of bytes communicate between HTTP web objects using certain protocols for sending a request or response; for example, RESTful HTTP. IPv6 or IP is the network layer protocol used. The transport layer protocols used are TLS and TCP.

IoT device or machine applications need constrained environment protocols such as CoAP and LWM2M.

3.2.1 Constrained Application Protocol

IETF recommends Constrained Application Protocol (CoAP) which is for CoRE using ROLL data network. Features of CoAP are:
- An IETF defined application-support layer protocol
- CoAP web-objects communicate using request/response interaction model.
- A specialised web-transfer protocol which is used for CoRE using ROLL network.
- It uses object-model for the resources and each object can have single or multiple instances.
- Each resource can have single or multiple instances.
- An object or resource use CoAP, DTLS (security binding with PSK, RPK and Certificate) and UDP protocols for sending a request or response.
- Supports the resource directory and resource-discovery functions
- The resource identifiers use the URIs as follow coap://… .
- Has small message header, 4 bytes are for ver (version), T (message type) [Types are confirmable, non-confirmable, acknowledgement and reset], TKL (token length), code (request method or response code), message ID 16-bit identifier, token (optional response matching token). Confirmable means server must send an acknowledgement, while non-confirmable means no acknowledgement is required.
- CoRE communication is asynchronous communication over the ROLL.
- Integrates easily with the web using CoAP application cross-protocol proxies. This is facilitated because HTTP and CoAP, both share the REST model. A web client may not even be aware that it just accessed an IoT device sensor resource or sent data to an IoT device actuator resource.
- Use of REST: The access by a CoAP object or its resource is thus using:

- the URI
- a subset of MIME types
- a subset of response codes which are used for an HTTP object or resource.

Figure 3.2 (a) shows direct and indirect accesses of CoAP client objects to a CoAP server. Figure 3.2 (b) shows a CoAP client access for lookup of object or resource using a resource directory.

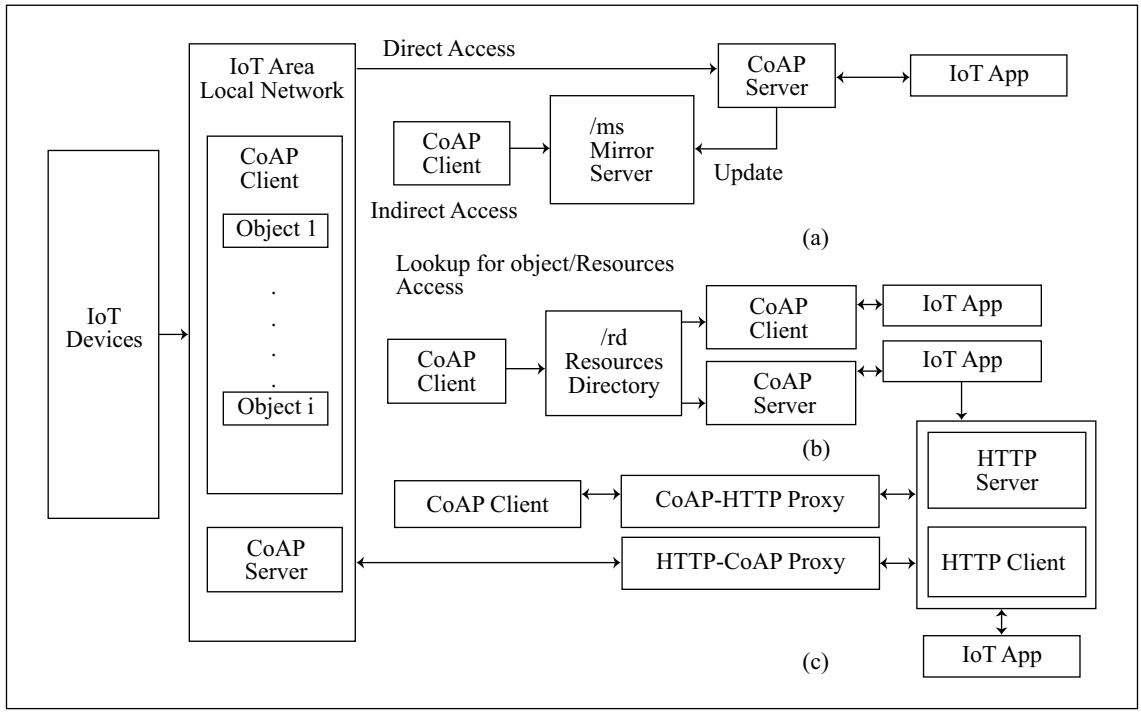


Figure 3.2 (a) Direct and indirect accesses of CoAP client objects to a CoAP server, (b) CoAP client access for lookup of object or resource using a resource directory, and (c) CoAP client and server access using proxies

CoAP Client Web Connectivity

A proxy is an intermediate server, which accepts a request from a client and sends the response to the client using a protocol. It also passes the request to the server and accepts a response from the server using the same or an other protocol. HTTP-CoAP proxy accepts requests from HTTP client using HTTP protocol and sends the request to the server using CoAP protocol. CoAP-HTTP proxy accepts requests from CoAP client using CoAP protocol and sends the request to the server using HTTP protocol.

Figure 3.2 (c) shows CoAP client and server access using proxies.

Transport Layer Security (TLS), earlier known as Secure Socket Layer (SSL) is the

protocol used for securing the TCP-based Internet data interchanges. DTLS is the TLS for datagram. The features of DTLS are:

- DTLS provisions for three types of security services—integrity, authentication and confidentiality.
- DTLS protocol derives from TLS protocol and binds UDP for secured datagram transport.
- DTLS is well suited for securing applications; for example, tunnelling applications (VPN), applications that tend to run out of file descriptors or socket buffers or applications which are delay sensitive (and thus use UDP).
- A part of DTLS is OpenSSL repository openssl-0.9.8 security based on PSK, RPK and certificate.

Secured Use of a Key for Client Authentication

PSK stands for Pre-Shared Key and is a method of securing using a key to authenticate a client. The key contains up to 133 characters in English. PSK method generates a unique encryption key for each client. A PSK is a symmetric key without forward secrecy (sender key not secret from receiver). Symmetric key means both ends 1 and 2 use the same key, K12 for encryption and decryption. Encryption uses an algorithm.

Private key refers to a key agreed for usage of data encryption (ciphering) between a pair of sender and receiver. The key is kept private between the two. Sender and receiver can be objects, applications, web services or processes.

RPK stands for Random Pair-wise Keys, which also stands for Raw Public Key meaning only the private/public key [means end 1 using RPK uses (K1 and Kp) and the other end using RPK uses (K2 and Kp)] are asymmetric.

Public key refers to a key (for example, a set of 128 or 256 bits), Kp which an intermediate server or trusted entity, such as a bank server (or object) provides to both the sender and receiver. The sender communicates using its key; for example, K1 and the receiver uses its key K2 when it communicates to the server; the server uses Kp. Messages exchange between sender and server using K1 and Kp. Messages exchange between receiver and server using K2 and Kp. The keys K1 and K2 are therefore kept secret between one another, sender and receiver. When message or data communication takes place each uses its own secret key and Kp of an intermediate server or entity. The advantage is that one communicating end cannot use the other end's key or even misuse that key later.

X.509 certificate is a certificate with a chain of trust based on an authorised Certification-Authority (CA) and Public Key Infrastructure (PKI). The sender submits a document once to the CA for digital signature, and the CA issues a certificate of verification of that document. Later, that document can be authenticated by the CA as same and from the same source. Public key infrastructure means provisioning a trusted service for assigning public keys.

3.2.2 Lightweight Machine-to-Machine Communication Protocol

Constrained environments communication with M2M apps and services using connectivity of LWM2M objects and resources

Lightweight Machine-to-Machine Communication (LWM2M) protocol is an application layer protocol specified by Open Mobile Alliance (OMA) for transfer of service data/messages. It finds applications in M2M. It enables functionalities for device management in cellular or sensor networks. Communication protocol 'light weight' means that it does not depend on call to the system resources during execution. Example of call to a system resource is calling an API for display menu or network function in the system software (OS). Lightweight presently means data transfer formats between client and server are binary and has Tag Length Value (TLV) or Java Script Object Notation (JSON) batches of object arrays or resource arrays and transfers up to 100s of bytes unlike the webpages of 1000s of bytes.

The protocol enables communication between LWM2M client at IoT device and an LWM2M server at the M2M application and service capability layer. The protocol is a compact one, meaning small header. It has an efficient data model. It is generally used in conjunction with CoAP.

Figure 3.3 shows M2M devices LAN connectivity. It shows constrained devices network connectivity with M2M applications and services using LWM2M OMA standard specifications of LWM2M.

Assume *i* number of M2M devices. Figure 3.3 shows the following:

- Local M2M constrained devices use, for example, the Bluetooth low energy, 6LowPAN (IPv6 over low power WPAN), CoRE, ROLL, NFC, ZigBee PAN, cellular, Wi-Fi or ZigBee IP (left-hand side) network technologies. M2M area network functions as PAN for connectivity between devices and the M2M gateway.
- 10s of bytes communicate between a device and the PAN.
- Communication between LWM2M objects (right-hand side). LWM2M client refers to object instances as per the OMA standard LWM2M protocol. A client object sends a request or receives a response of the LWM2M server over the access and CoRE networks.
- CoRE network, for example, 3GPP or other networks for the IP connectivity
- Communication from an object instance using interface functions. The functions are bootstrapping; registration, deregister or update a client and its objects; reporting the notifications with new resource values; and service and management access through the server.
- Use of the CoAP, DTLS, and UDP protocols by the object or resource.
- 100s of bytes communicate between objects at the client or server for plain text or JSON or binary TLV format data transfer.

LWM2M specifications and features are as follows:

- An object or resource use CoAP, DTLS, and UDP or SMS protocols for sending a request or response.

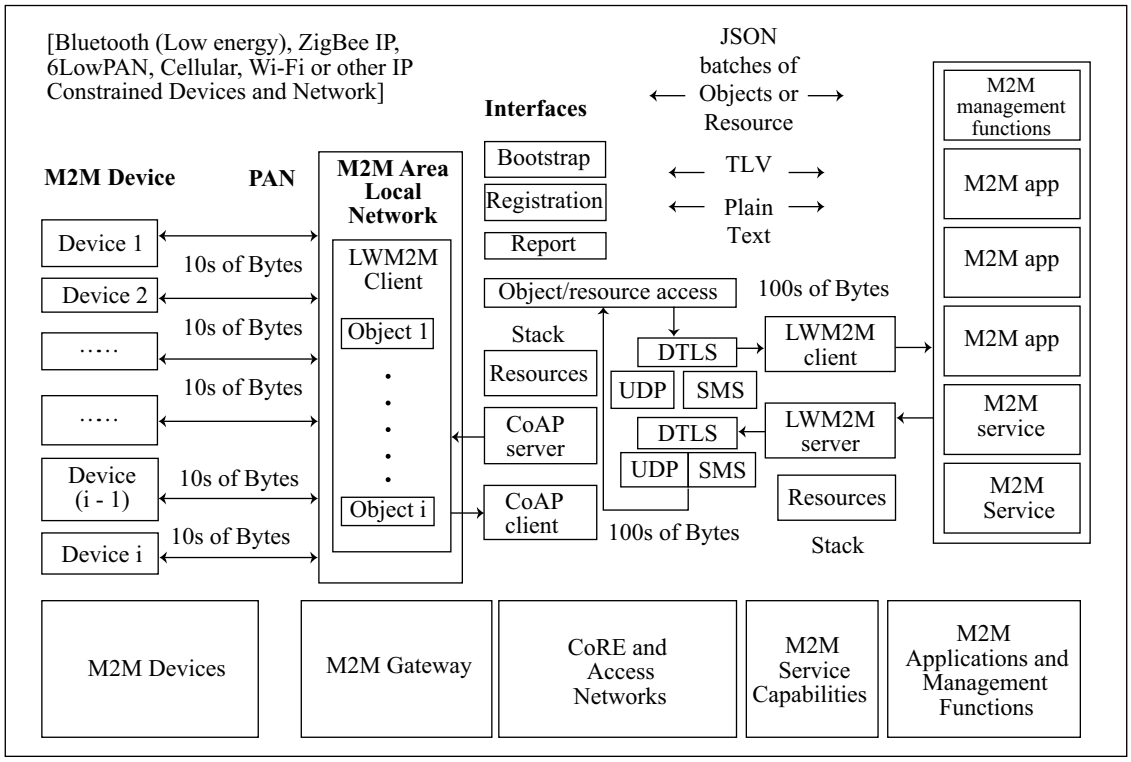


Figure 3.3 M2M devices local area network connectivity, and constrained devices network connectivity with M2M applications and services using LWM2M OMA standard specifications of LWM2M

- Use of plain text for a resource or use of JSON during a single data transfer or binary TLV format data transfer for a package for a batch of resource representations in a single data transfer.
- An object or its resource access using an URI.
- Interface functions are for—bootstrapping; registration, deregister or updating a client and its objects; reporting the notifications with new resource values; and service and management access through the server.
- Use of object model for resources and each object can have single or multiple instances. Each resource can have single or multiple instances.
- OMA or other standard specifying organisation defines the LWM2M objects for usages in M2M communication. An LWM2M client has object instances as per OMA standards. LWM2M client-server interactions are over the access and CoRE networks which generally use CoAP over ROLL. LWM2M device client and server, generally use CoAP client server interactions for data interchanges. Wi-Fi and WiMax are examples of access networks. Examples of CoRE network are GSM, GPRS, 3GPP and 4G LTE or other networks for IP connectivity as well as roaming.

- Organisations can register the other LWM2M objects and resources with OMA or other standards organisations.
- M2M management functions can be M2M Service Bootstrap Function (MSBF) for credentials of the devices and gateway, and M2M authentication server (MAS) for security, root key data store and device and data authentication.

Example 3.1 gives the OMA specifications[2] for management objects, query resources, location resources and resource attributes, and response codes for M2M.

Example 3.1

Problem

Give examples of LWM2M objects and resources.

Solution

Following are OMP-specifications-based objects and resources:

OMA Management Objects

Objects assignments are:

- Multiple instances case ID = 1 for LWM2M server, and 2 for access control (means rights granted yes or no).
- Single instance case ID = 3 for device object, 4 for connectivity monitoring, 5 for firmware (installation, update or actions after update) and 6 for location object.

Query Parameter Codes

The assignment of parameters in the queries is:

- h for end-point name, when registering in a domain
- rt for end-point type
- lt for life-time in second.

Locations Object Resources

Locations Object Resources have single instances. Each instance is specified by an ID as follows:

- 0 for latitude, 1 for longitude, 2 for altitude (in decimal numbers),
- 4 for velocity of non-static devices,
- 5 for time stamp
- 3 for uncertainty.

Resource Attributes

Attributes assignment is using an id, which is an integer for resource id. Attributes used in auto-observation are specified as:

- *if* (interface description at initiation at bootstrapping)
- *url* (for path)
- *rl* (for type)
- *ct* (for CoAP content type convert ASCII MIME type)
- *obs* (observable Boolean)
- *aobs* (auto-observable Boolean)

[2] https://github.com/OpenMobileAlliance/OMA-LWM2M-DevKit/blob/master/objects/lwm2m-object-definitions.json

Response Codes

Specified for creating successful registration and maximum lifetime or 4.00 for a bad CoAP request (Example 3.1).

3.2.3 JSON Format

Features of a JSON are:

- JSON is an open-standard format used primarily to transmit data between a server and web application, as an alternative to XML. The text is human readable. It transmits data objects as text to transmit. It consists of attribute–value pairs.
- Originally derived from the JavaScript scripting language, JSON, is now a language-independent data format, the coding of which can be in Java or C or another programming language used for parsing and generating JSON data.

Example 3.2 describes codes used for data transfer in JSON.

Example 3.2

Problem

Give a JSON object example in JavaScript using OMA object definitions.

Solution

Following are the codes as per OMA specifications for LWM2M objects[2]. The code is at:

```
{
"0":    {
          "id": 0,
          "name": "LWM2M Security",
           "instancetype": "multiple",
           "mandatory": true,
           "description": "This LWM2M Object provides the objects and resources of Client appropriate to access a specified LWM2M Server. The LWM2M Object Resources MUST only be changed by a LWM2M Bootstrap Server or Bootstrap from Smart Sensor and MUST NOT be accessible by any other LWM2M Server. One Object Instance SHOULD address a LWM2M Bootstrap Server.\n ",
           "resourcedefs": {
  "0": {
           "id": 0,
           "name": "LWM2M  Server URI",
           "operations": "-",
           "instancetype": "single",
           "mandatory": true,
           "type": "string",
```

```
        "range": "0-255 bytes",
        "units": "",
        "description": "Uniquely identifies an LWM2M Server or LWM2M Bootstrap Server. Its form is: \n\"coaps://host:port\", where port is the UDP port of the Server.host is an IP address or FQDN, and "
"1":    },
        {
        "id": 1,
        "name": "Bootstrap Server",
        "operations": "-",
        .
        .
        .
        }
"9": {
        "id": 9,
        "name": "LWM2M Server SMS Number",
        "operations": "-",
        "instancetype": "single",
        "mandatory": true,
        "type": "integer",
        "range": "",
        "units": "",
        "description": "MSISDN used by the LWM2M Client to send messages to the LWM2M Server via the SMS binding. \nThe LWM2M Client SHALL silently ignore any SMS not originated from unknown MSISDN"
},
```


Example 3.3 describes the codes used for data transfer in Java using JSON.

Example 3.3

Give a JSON object example in Java.

Solution

JSON object is subclass of java.util.HashMap. Assume sensor name is SensTemp1, temperature is integer, and number of sensors is a double. is_newSensorAdded ( ) is a method which returns Boolean data type.

Following is the code in Java:

```
//import org.json.simple.JSONObject;
  JSONObject obj=new JSONObject();
          obj.put ("name","SensTemp1");
           obj.put ("temperature",new Integer(100));
           obj.put ("numberOfSensors",new Double(1000.21));
          obj.put ("is_newSensorAdded",new Boolean(true));
  StringWriter out = new StringWriter();
          obj.writeJSONString(out);
  String jsonText = out.toString();
  System.out.print (jsonText);
```

Result: {"Name": "SensTemp1": "Temperature": 26" "numberOfSensors":1000000, "newSensorAdded": null, }

3.2.4 Tag Length Value Format

In a TLV format the first two bytes are used to identify the parameter. The first and fourth bytes indicate the length of the actual data which follows directly after these bytes.[3] Following is an example of codes used for data transfer in TLV format.

Example 3.4

Problem

Give examples of TLV format messages.

Solution

Assume that an application service uses CoAP for which code is "200" (standard number allotted by OMA.) Let engine parameter rpm is assigned ID = 125 and velocity assigned ID = 126. Format in TLV can be as follows:

</engine>;EngObj;id="20";ct="200",

</engine/rpm>; EngObj;id="125",

</engine/velocity>;EngObj;id="126"

3.2.5 MIME Type

Generic-type files are application/octet-stream; however, MIME-type files are used in web applications and services.[4] Features of MIME are:

- An Internet standard which is for description of the contents of various files.
- An Internet standard which extends the Simple Mail Transfer Protocol (SMTP) format of email to support the text in character set besides ASCII, and the non-text attachments such as audio, video, images and application programs.
- Supports messages with multiple parts and header information in non-ASCII character sets.

[3] https://en.wikipedia.org/wiki/Type-length-value
[4] https://en.wikipedia.org/wiki/Media_type

- Initially designed for mail, now it is used as Internet media type. Web client header specifies the content types using a MIME type.
- List of MIME type files is exhaustive. Few examples of MIME type files are—doc (application/msword), html (text/html), htm (text/htm), gif (image/gif), js (application/x-javascript), css (Application is text/css), mpeg (video/mpeg), pdf (application/pdf) and exe (application/octet-stream). HTML links, objects, and script and style tags, each has a type attribute the value of which can be set equal to a MIME type. The MIME type for HTML is text/html.

An associated file is also used which is called mime.types file. It associates filename extensions with MIME type. Such kinds of files allow one to use the extension in these cases. A mime.types file enables a web server to determine MIME type of a resource. This is because a file system may not store the MIME type information, but instead rely on the filename extension.

Reconfirm Your Understanding

- Meanings of important key terms—API, service, state, object, resource, directory, repository, identifier, discovery, representation, URI, URL, header, hypertext, hyper link, HTTP, HTML and XML have been discussed.
- REST is a software architecture. It refers to the practices and constraints to be followed. REST uses URIs/URLs for representations of the resources, uses access methods: POST, GET, PUT and DELETE. The architecture is used during design of software components.
- IETF recommended CoRE specifications for ROLL network. CoRE specifications include the CoAP and UDP.
- CoRE objects use standard protocols for constrained environment when sending a request or response. CoAP client and CoAP server objects communicate over the network, transport layers to other end CoAP client and server. The objects connect to web using HTTP-CoAP and CoAP-HTTP proxy. A client/server object use the URIs coap://… in place of http://… .
- CoAP client-server interactions use UDP and DTLS protocols. The HTTP client-server interactions use TCP/UDP and TLS protocols.
- LWM2M client has specifications of the object instances as per OMA specifications. LWM2M client-server interactions are over the access and CoRE networks. LWM2M client and server generally use the CoAP client server interactions over ROLL. Access network examples are Wi-Fi and WiMax. CoRE network examples are GSM, GPRS, 3GPP, 4G LTE and other networks which enable IP connectivity and roaming.
- An object instance communicates using the LWM2M Interface functions that are bootstrapping; registration, deregister or update a client and its objects; or reporting the notifications with new resource values; and service and management access through server.
- An application exchanges the objects using Internet protocols. 100s of bytes communicate between the objects at the client or server. The objects communicate plain text, XML, JSON, TLV or MIME type formats for communication on the Internet.

Self-Assessment Exercise

1. List the properties of constrained environments. Use examples of connected devices, such as streetlights, RFIDs and ATMs with the Internet. ★
2. Give the order of bytes which transmit or are received at the connected device, devices local area network cum gateway and at the HTTP web object during interchange of data between devices and network, and application and service domains. ★
3. List features of REST architectural style of designing software components. ★
4. Create a table of comparison between constrained devices and unconstrained system environment, protocols, client and server characteristics. ★
5. Consider Figure 3.1 and draw a data communication diagram for ZigBee IP connected streetlights local network gateway connectivity and secured web connectivity to streetlight monitoring and controlling server applications in the CoRE and unconstrained environments using the CoAP and HTTP protocols. ★★★
6. List the features of CoAP. ★★
7. What are the functions of DTLS? ★
8. Consider Figure 3.3 and draw a data communication diagram for RF links connected RFID local network gateway connectivity and inventory application and tracking service using LWM2M and IP network. ★★
9. List OMA specifications for management objects, query resources, location resources and resource attributes and response codes for M2M. ★★
10. Write code for JSON object in Java. Assume sensor names are StreetLightN_I and StreetLightN_J for Nth streetlight. Ambient-light sensor N_I output is Boolean and Street Traffic density sensor N_J output is a Byte. The is_SensorN_IAdded ( ) and is_SensorN_JAdded ( ) are methods which returns Boolean data type.
11. Write the format in TLV for data interchange in LWM2M. Assume that an application service uses CoAP for which code is "200" (Standard number allotted by OMA. Let traffic density parameter numPer100Meter is assigned ID = 4038 and street length, m is assigned ID = 126). ★★★
12. List 10 extensions in the list of MIME-type files. ★

3.3 MESSAGE COMMUNICATION PROTOCOLS FOR CONNECTED DEVICES

LO 3.2 Illustrate the usage of messaging protocols between connected devices and the web

A device/node/end-point/client/server sends and receives message(s). A communication module includes a protocol handler, message queue and message cache. The protocol-

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

handler functions during transmission and reception of messages according to the communication protocols for these actions. Message queue forms to keep a message until it is transmitted towards its destination. Message cache keeps an incoming message until it is saved into the module. A device message-queue inserts (writes) the message into the queue and deletes (reads) the message from the queue. Device message-cache stores the received messages. The following section explains the terms used by message-communication protocols.

3.3.1 Terminology

Request/Response (Client/Server)

A request/response message exchange refers to an object (client) requesting for resource(s) and an object (server) sending the response(s). The objects, for example, HTTP objects may use the REST functions. When sending a request, a protocol adds the header words. Each header has fields. Each field interprets the request at the receiver object.

Publish/Subscribe (pubsub)

Publish/subscribe or PubSub message exchanges differ from request/response. A service publishes the messages; for example, a weather information service publishes the messages of weather reports for the potential receivers. A group controller publishes the messages; for example, measured values of ambient light conditions, traffic density or traffic presence (Example 1.2).

A service can be availed by one or more clients or brokers. When a client subscribes to the service, it receives messages from that service. A *publish/subscribe messaging protocol* provisions for publication of messages and reception on subscription (PUT and GET methods) by the registered or authenticated devices.

Publication may be for measured values, for state information or resources of one or more types. *Subscription* is for a resource-type (or for a topic). *A separate subscription is required for each resource-type or topic*.

An example of resource type is measured values of ambient light condition in the smart streetlights example. Another resource type is traffic presence or absence on the street. Another resource type is lighting function report; functioning is proper or fault exists in the light.

Resource Directory

Resource directory (RD)[5] maintains information and values for each resource type. A resource of a resource type accesses from the RD using URI for the resource.

Resource Discovery

Resource discovery service may advertise (publish) at regular intervals, the availability of the resources or types of the resources available and their states. A client discovers the resource type and registers for the RD service.

[5] https://tools.ietf.org/html/draft-koster-core-coapmq-00

Registration/Registration Update

Registration means a receiver registers with a service, such as an a RD service. When one or more endpoints or devices or nodes registers, then that device gets the access to the resources and receives published messages. Security considerations may require authentication of both ends (service provider and receiver) before registration. A separate registration is required for each endpoint (client or server).

Registration updates means updating for one or more endpoints or devices or nodes. It also includes unregistering for one or more endpoints.

Pull (Subscribe/Notify) Data

Pull means pulling a resource, value, message or data of a resource-type by registering and subscribing. Pull may be using GET or on initiating OBSERVE. The server maintains state information for a resource and notifies on change of state. Client pulls again the resource on the change.

Polling or Observing

Polling means finding from where new messages would be available or whether new messages are available or updates are available or whether the information needs to be refreshed or finding the status if the state information has changed or not. When messages store at a database-server, then polling can be done by a client who uses the REST architecture GET method and server uses the POST.

A state may refer to a connection or disconnection, sleep, awake, created, alive (not deleted), old values persisting or updated with new values (GET + OBSERVE). *Observing* means looking for change, if any, of a state at periodic intervals (OBSERVE).

Push (Publish/Subscribe) Data

Push means a service that pushes the messages or information regularly. Interested device or endpoint or potential receiver receives these pushes. For example, a mobile service provider pushes the temperature and location information regularly for the potential receivers (registered mobile services subscribers) (PUT).

Push is efficient compared to polling, particularly when notifying or sending alerts. This is because there can be many instances when polling returns no data. Pull is efficient compared to polling or PUSH when certainty exists that within a reasonable time interval the server returns no data.

Message Cache

Cache means storing when available and can be used later on when required. Messages cache is useful in an environment of short or prolonged disconnections of a service. A message can be accessed once or more times from a cache.

Message Queue

Message queuing means storing the messages (data) in sequence from devices or endpoints so that when connection state changes then messages can be forwarded. Forwarding is done using the first-in first-out method for a resource type. A message forwards only once from a queue.

Separate queues are formed for each resource type. The messages are forwarded to the registered devices or endpoints and to the subscribed devices or endpoints. A separate registered device or endpoint list and a separate subscription list is maintained and used for each resource type. Forwarding takes place only after matching the subscription from a list.

Information/Query

The method is that an object (client) requests information using a query while another end-object (server) responds by replying to the query. The responding application processes the query using the query optimiser and retrieval plan. The query processing uses a database or resource directory resources.

3.3.2 Communication Protocols

Following are the protocols used in message communication.

CoAP-SMS and CoAP-MQ

M2M or IoT device uses SMS quite frequently. SMS is identified as the transport protocol for transmission of small data (up to 160 characters). It is used for communicating with a GSM/GPRS mobile device.

M2M or IoT device uses message queuing quite frequently due to ROLL environment and constrained devices (awake only when initiated) or connection-breaks for long periods. CoAP-SMS and CoAP-MQ are two protocols drafted and recommended by IETF.

CoAP-SMS

CoAP-SMS is a protocol when CoAP object uses IP as well as cellular networks and uses SMS. It is an alternative to UDP-DTLS over ROLL for CoAP object messages and when using cellular communication.

SMS is used instead of UDP + DTLS by a CoAP client or server. A CoAP client communicates to a mobile terminal (MT) endpoint over the General Packet Radio Service (GPRS), High Speed Packet Access (HSPA) or Long Term Evolution (LTE) networks using CoAP-SMS protocol.

Following is the IETF recommended terminology for use:

- SMS-C: SMS service centre
- SMS-SP SMS service provider

- CIMD: Computer interface to message distribution
- MS: Mobile station at a cellular network functioning as CoAP client or CoAP sever
- MO: Machine or IoT device or mobile origin functioning as CoAP client
- MT: Machine or IoT device or mobile terminal functioning as CoAP sever
- SMPP: Short message peer-to peer for CoAP-Data. Peer-to-peer means sending requests as well as receiving requests
- SS7: Signalling Service Protocol
- UCP/UMI: Universal computer interface protocol/machine interface for short message delivery contained in a CoAP request or CoAP response

The CoAP-SMS features are as follows:

- An URI used as coap+sms:// in place of coap://. For example, URI may be coap+sms://telNum/carLocatiobObject/latitude for latitude of location of a car after LocationObject measures location parameters using the GPS. URI is used when sending the SMS to the specified telephone number—telNum.
- A CoAP message encodes with alphabets for SMS communication. An SMS message consists of 160 characters in 7-bit encoding of a character. Maximum length for a CoAP message is thus 140 B (= 160 × 7 bits/8 bits) when an SMS-C supports 8-bit encoding. A CoAP message is thus 70 (= 160 × 7 bits/16 bits) when an SMS-C supports 16-bit encoding and multilingual alphabets. Concatenated short messages can be up to 255B, provided the SMS-C supports this.
- CoAP endpoints have to work with a Subscriber Identity Module (SIM) card for SMS in cellular networks. The points are addressed by using Mobile Station ISDN (MSISDN) number. A TP-DATA-Coding-Scheme inclusion enables the CoAP client to find that the short message contains a CoAP message.
- Does not support multi-casting
- Two additional options are Response-to-URI-Host (RUH) and Response-to-URI-Port (RUP) which make initiating CoAP client aware of the presence of the alternative interface CIMD and SMPP and UCP/UMI. RUH is string of size 0 to 255B. RUP is of 2B with default port number 5683. IANA (Internet Assigned Number Authority) registered TBD as CoAP Option Numbers for registry.
- Data interchange sequences are as follows: An MS/CoAP client sends a SMS request (SMS-SUBMIT) to SMS-C; SMS-C reports using SMS-SUBMIT-REPORT; SMS-C sends SMS (SMS-DELIVER) to MS/CoAP server; the server reports using SMS-DELIVER-REPORT; and SMS-C sends SMS-STATUS-REPORT to the client.
- Authentication of a client by the server provides the security. MSISDN of the MS and SIM based security is used during SMS data exchanges.

Figure 3.4(a) shows a CoAP request or response communication to a machine, IoT device or MT. A CoAP client sends request to SMS-C which transmits the request to an MT. A CoAP server sends response to SMS-C which transmits the response to the client.

Figure 3.4(b) shows a computer or machine interface using IP for sending request or receiving the CoAP data or HTTP request (REQ) to a mobile service provider using SMPP or CIMD for data interchange. The service provider communicates using SS7 or CIMD or

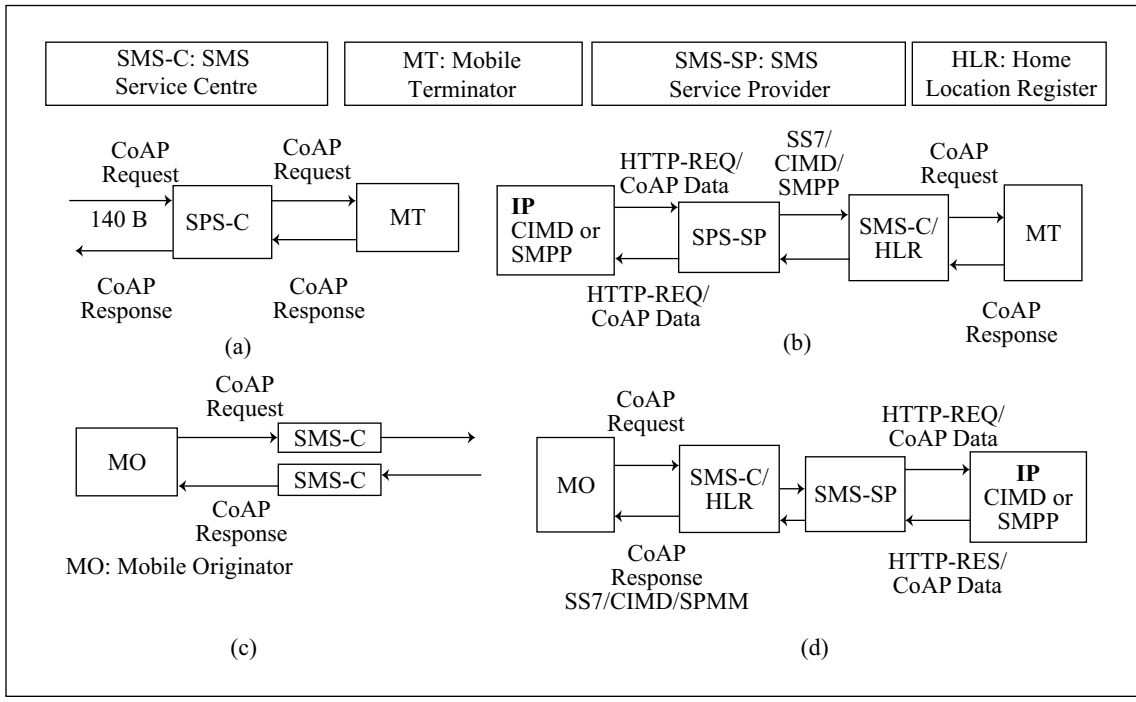


Figure 3.4 (a) A CoAP request or response communication to a machine, IoT device or mobile terminal (MT), (b) A computer or machine interface using IP communication to a mobile service provider for data interchange with terminal, (c) A machine or IoT device or mobile origin (MO) communication of CoAP request or response communication, and (d) An origin communication using SS7/CIMD/SMPP with a computer or machine interface using IP communication

SMPP with machine or IoT device or MT with an in-between node SMS-C. The terminal sends response to the origin using SMS-C and CoAP-MQ Broker.

Figure 3.4(c) shows a CoAP request or response communication from a machine, IoT device or MO. A CoAP client sends request to SMS-C which transmits the request. A CoAP server sends the response to SMS-C which transmits it to the client.

Figure 3.4(d) shows a computer or machine interface using IP for receiving request or sending response (RES) as CoAP data or HTTP REQ mobile service provider using SMPP or CIMD for data interchange. The SMS-C communicates that to CoAP-MQ BROKER using SS7 or CIMP or SMPP. SMS-SP receives request or sends response to machine or IoT device or mobile origin.

CoAP-MQ

CoAP-MQ is a message queue protocol using a broker and RD. Roles of CoAP endpoints have roles as a client and server.

Figure 3.5 shows data interchanges between CoAP-MQ endpoints, CoAP-MQ clients, CoAP-MQ servers through CoAP-MQ broker and its services.

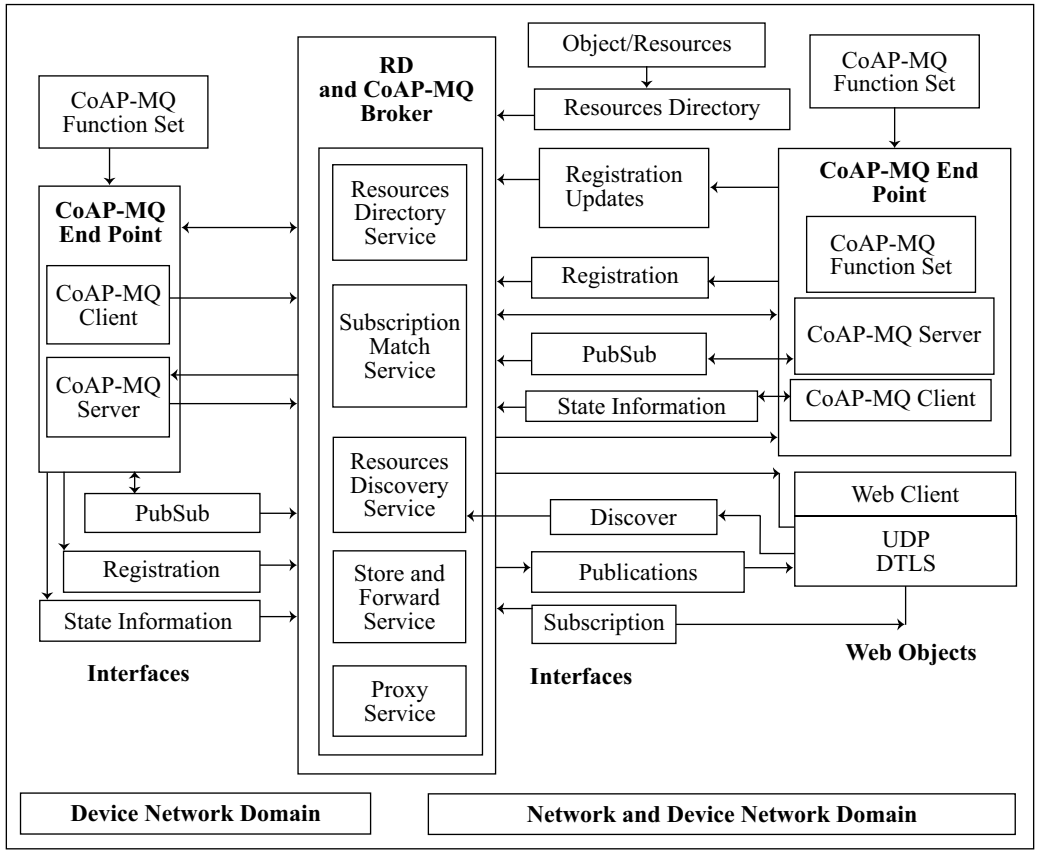


Figure 3.5 Data interchanges between CoAP-MQ endpoints, CoAP-MQ clients, CoAP-MQ servers through CoAP-MQ broker and its services [PubSub means publication to RD and subscription to MQ]

Figure 3.5 shows CoAP-MQ server provisioning for the resource-subscription, store from the publisher. The server also provisions for forwarding to the subscriber and proxy services. The figure also shows that RD services are resource discovery, directory and object registration services. The device objects communicate using the CoAP client and server protocols and CoAP web objects using DTLS as security protocol and UDP for CoAP APIs.

MQTT Protocol

Message Queuing Telemetry Transport (MQTT) is an open-source protocol for *machine-to-machine (M2M)/IoT connectivity*. Word 'telemetry', in English dictionary, means measuring and sending values or messages to far off places by radio or other mechanism.

IBM first created it and then donated it to M2M 'Paho' project of Eclipse. A version is MQTT v3.1.1. MQTT has been accepted (2014) as OASIS (Organization for the Advancement of Structured Information Standards) standard[6] MQTT protocol is used for connectivity in M2M/IoT communication.

A version is MQTT-SN v1.2. Sensor networks and non-TCP/IP networks, such as ZigBee can use the MQTT-SN. MQTT-SN is also a publish/subscribe messaging protocol. It enables extension of the MQTT protocol for WSNs, the sensor and actuator devices and their networks.

Recall from Figure 1.3, IBM conceptual framework for the IoT, which showed MQTT applications for one of the communication management functions. Figure 3.6 shows messages interchange between M2M/IoT device objects (publisher and subscriber) and web objects (publisher and subscriber) using an MQTT broker.

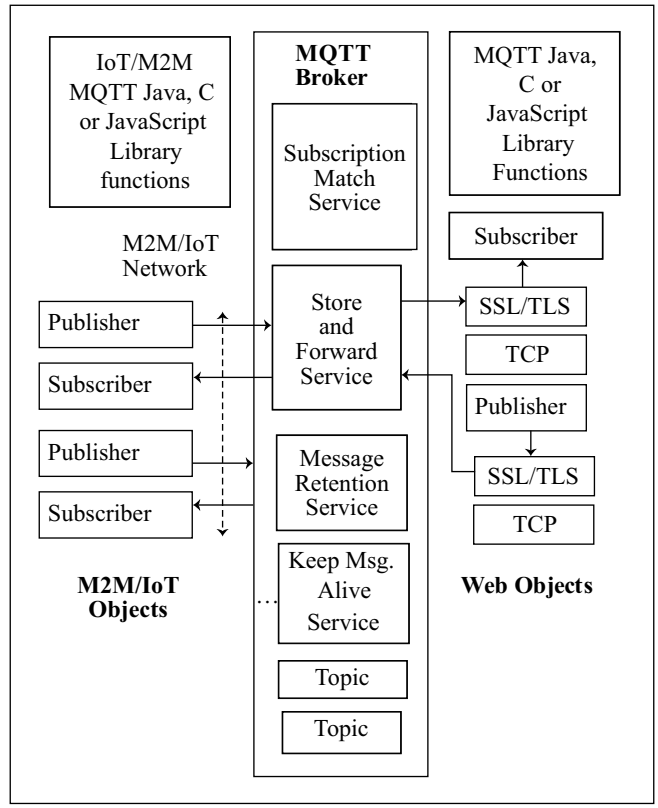


Figure 3.6 Messages interchange between M2M/IoT device objects (publisher and subscriber) and web objects (publisher and subscriber) using an MQTT Broker

[6] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.doc.

Figure 3.6 shows MQTT-broker subscription, subscription match, store and forward, last good message retention and keep message alive services. The figure also shows that device objects use MQTT Java, C or JavaScript library functions. The objects communicate using the connected devices network protocols such as ZigBee. Web objects also use MQTT library functions and communicate using IP network and SSL and TLS security protocols for subscribing and publishing web APIs.

MQTT Broker (also referred simply as MQ) does the following:

- Functions as a server node capable of storing messages from publishers and forwarding them to the subscribing clients.
- Receives topics from the publishers. Examples of topics are measured information of ambient light conditions, traffic density, nearby parking space availability and waste container status.
- Performs a store-and-forward function, stores topics from publishers and forwards to subscribers.
- Receives subscriptions from clients on the *topics*, matches subscriptions and publications in order to route messages to right endpoints.
- Recovers subscriptions on reconnect after a disconnection, unless the client explicitly disconnected.
- Acts as a broker between the publisher of the topics and their subscribers.
- Finds client disconnection until DISCONNET message receives, *keeps message alive* till explicit disconnection.
- Retains the last-received message from a publisher for a new connected subscriber on the same topic, when retain field in the header is set.
- Authentication by Username/Password in connect message and client security is through SSL/TLS. Security considerations are same as of CoAP, web-linking and CoRE resource directory.
- Support from Intelligent and business analyst server and other servers through a MQTT server with a gateway.

3.3.3 XMPP (Extensible Messaging and Presence Protocol)

XML

XML is an open-source IETF recommended language. XML is widely used for encoding messages and texts. A text element in an XML document can correspond to the data, message, alert, notification object, command, method or value of an entity. There is a way to tag in which text is featured between the tag and its associated end tag.

An XML tag specifies the type of encoded entity in an element. A tag may also associate an attribute(s) which is specified there itself. The interpretation and use of the text within a tag pair (a tag and its associated end tag) depends on the *parser* and associate *application*. The Parser uses XML file as input. The *application* uses the output from the parser. The *parser* and application can be in Java, C# or any other programming language. Following is an example of usage of XML tags and attributes in XML elements in a document.

Example 3.5

Problem

Give the usages of XML tags and attributes in XML messages using an XML document (text file), assuming that the XML messages are for a temperature sensor. Assume that more than one sensor measures the temperature and put time of measurement stamp also.

Solution

An XML message sent by a temperature sensor is as follows: <SensTemp> 22 degree C </SensTemp>. <SensTemp> can associate with the attributes as:

<SensorTemp ID = '250715' TimeDate = '19:28:33 Jul 17 2016'> 22 </SensTemp>.

A Java *parser* can read the file when it streams and in which the above text message is present. That parser creates a database. The database may be a table with the columns: *Sensor ID*, *Time*, *Date and Temperature*. An associated program (*application*) uses these results. For example, *application* sends a message for an actuator circuit to switch off Air Conditioner, again in XML. The message format from the Application with ID = 241206 to Air conditioner of ID = 2075 can be:

<ActuatorAirCond ID = '2075' AppId = '241206' > SwitchOff < /ActuatorAirCond>.

XMPP

XMPP is an XML-based specification for messaging and presence protocols. XMPP is also an open-source protocol recommended specification which is accepted by IETF. RFC is an international organisation and stands for 'Recommended for Comments'. RFC 6120 document specifies the XMPP for CoRE. Another recommendation, RFC 6121 XMPP specifies the instant messaging (IM) and presence, and RFC 6122 XMPP specifies the (message) address format.

Messages notify *presence* for the IMs to one or many at the same time. It enables chatting and Multi-User Chat (MUC) after creation of a chat room, where different users can do the IMs. XMPP enables interoperable communication: for example, Google Talk. XMPP enables IMs between many users as it uses presence-notifications and chat features.

Chat room is an application, in which all those who have subscribed (meaning persons and objects initiating chatting and messaging to one another at the same time) are provided a room-like view and use the IMs among themselves.

XMPP is extensible—XSF (XMPP standards foundation) develops and publishes the xeps (XMPP extension protocols). The xeps enable the addition of features and new applications. List of XMPP xeps for web objects is quite long. Examples of xeps are:

- xep-DataForms Format
- xep-XHTML-IM
- xep-Service Discovery
- xep-MUC
- xep-Publsih-Subscribe and Personal Eventing Protocol
- xep- File Transfer
- xep-Jingle for Voice and video

XMPP-IoT xeps extend the use of XMPP to IoT and machine-to-machine messaging.[7,8] List of extensions (xeps) is quite long. Examples of xeps are:

- xep-0322 efficient XML interchange (EXI) format
- xep-0323 Internet of Things-Sensor Data (http://xmpp.org/extensions/xep-0323.html)
- xep-0324 Internet of Things-Provisioning
- xep-0325 Internet of Things-Control (http://xmpp.org/extensions/xep-0325.html)
- xep-0326 Internet of Things-Concentrators.

The xeps are the base for IEEE/ISO/IEC recommended "Sensei/IoT" related standards, sensei/IoT ISO/IEC/IEEE P21451-1-4 and other standardizations.

XMPP-IoT enables communication machine to web application and M2M interoperable. Many connected-devices-based business applications, processes, smart home, smart city, smart energy saving and smart street lighting and traffic are possible using XMPP, XMPP-IoT and other servers communicating with XMPP servers.

Figure 3.7 shows use of the XMPP and XMPP extension protocols for connected devices and web objects. The protocols are for messaging, *presence* notification, response-on-demand and service discovery using XML streams.

Figure 3.7 shows that XMPP-IoT server consists of xeps and services. XMPP services are extensible to publisher/subscriber, MUC and other services, devices and connected devices area network which interact with XMPP-IoT services. The XMPP-IoT server, through a gateway between connected devices and IP networks, communicates with XMPP APIs in web objects.

Simple Authentication and Security Layer (SASL) and TLS are security protocols for APIs and web object messages using TCP/IP network. The XML streams in XMPP format communicate between the devices, devices to web objects and between the web objects.

Features of XMPP are:

- XMPP uses XML.
- XML elements are sent in the open-ended stream within the tag <stream> and corresponding end tag </stream>.
- Three basic types of XMPP stanzas (elements) are:
  - message
  - presence
  - iq (information/query, request/response)
- Extensibility to constrained environment messaging and presence protocols as well as IP network messaging.
- Extensibility of request-response (client-server) architecture to iq (information through querying), PubSub messaging, Chat room MUC messaging and other architecture (where group of people exchange information when present in a chat room), decentralised XMPP server.

[7] https://postscapes.com/internet-of-things-protocols/
[8] http://www.xmpp.org/extensions.html

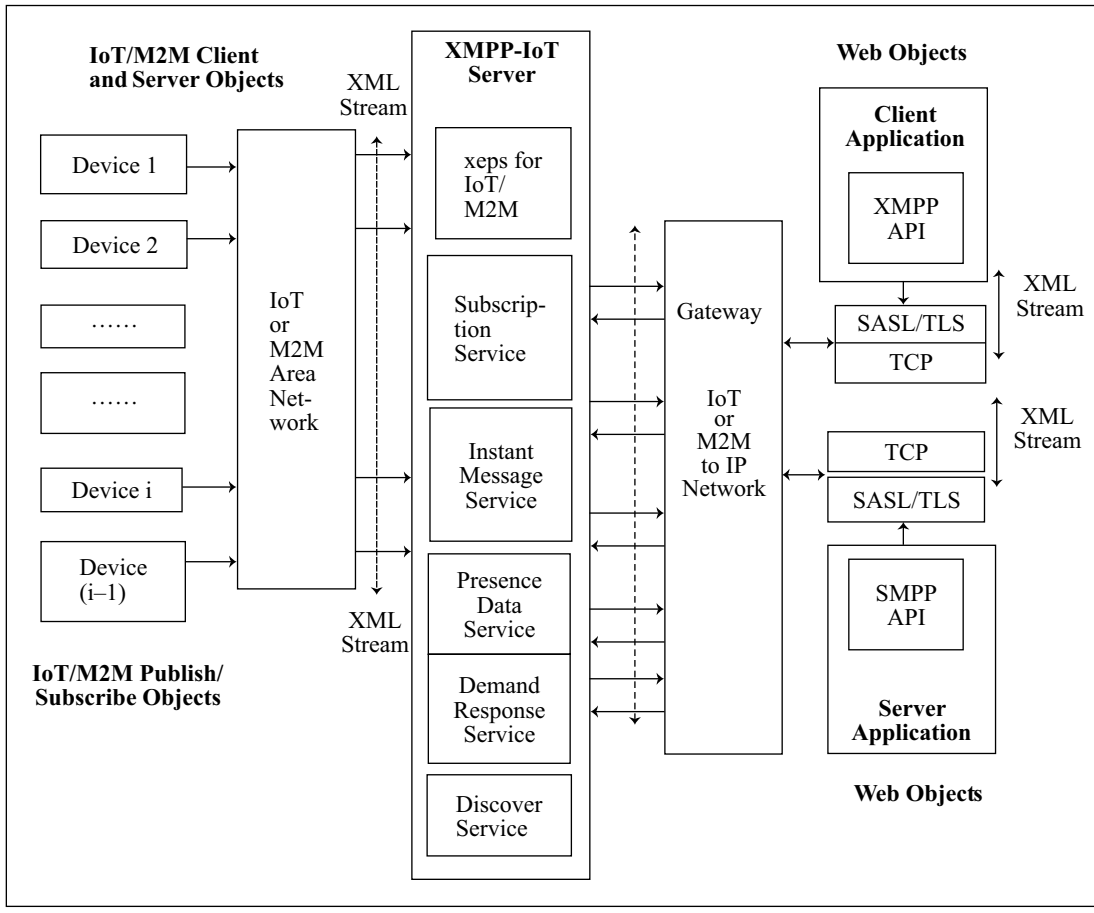


Figure 3.7 Use of XMPP and XMPP extension protocols for connected devices and web objects for the messaging, presence notifications, responses on demand and service discoveries using XML streams

- XMPP server set by anyone on the following standards recommended and using XSF xeps; for example, XMPP-IoT server, XMPP M2M server for messaging between the machines.
- Authentication by SASL/TLS, and support from intelligent and business analyst applications, and processing through XMPP server and gateway for connecting device network with IP network.

XMPP does the following:

- Binary data is first encoded using base 64 and then transmitted in-band. Therefore, the file first transmits out-of-band between nodes on messages from XMPP server but not directly like IMs.
- No end-to-end encryption
- Higher overhead being text based in place of binary implementations
- No support for QoS like MQTT does

Reconfirm Your Understanding

- CoAP-SMS is a protocol when CoAP object uses IP as well as cellular networks and SMSs.
- SMS transport through a service centre is used instead of UDP + DTLS by a CoAP client or server. A CoAP client communicates to a mobile terminal endpoint.
- Web objects use the Request/Response, Publish/Subscribe, Subscribe/Notify, Push or Pull and Polling, Message Queues, Instant Messaging, Presence notification for Multi-User Chat and instant messaging modes for transfer of messages, resources, representations, objects and states. APIs use messaging protocols, CoAP-SMS, CoAP-MQ, MQTT and XMPP.
- CoAP-MQ data interchanges between the CoAP-MQ endpoints, CoAP-MQ clients, CoAP-MQ servers through CoAP-MQ broker and its services.
- MQTT is a protocol for the constrained environment. MQTT architecture is PubSub messaging in place of request-response client-server architecture. *Publisher* (message sender at the device domain or web object at network and application domain) sends the messages on a topic. *Subscriber* (message receiver at the device domain or web object at network and application domain) receives the messages on a subscribed topic.
- XMPP CoRE, instant messaging, presence and address format are IETF recommendations for the XMPP messages server. The server notifies *presence* for the purpose of IMs (instant messaging) to one or many at the same time. The server enables chatting and multi-user chat (MUC) after creation of a chat room, where different users after subscriptions do IMs.

Self-Assessment Exercise

1. List IETF recommended terms and their usages in CoAP-SMS protocol. ★
2. Recapitulate Example 1.1 for an umbrella connected to a web-based weather service and to a mobile for messaging. Refer Figure 3.4 and draw data interchange using (a) a CoAP request or response communication to a weather data service and (b) an SMS from umbrella device using CoAP to the mobile device. ★★
3. List the functions of resource directory service by RD when using at the CoAP-MQ protocol. ★
4. List comparisons between CoAP-MQ and MQTT features. ★★
5. Refer Figure 3.5 and draw a diagram for data interchanges for connected temperature sensing devices using CoAP-MQ broker. Each published temperature data using Bluetooth Smart Energy to the broker and gateway publishes using CoAP-MQ client to CoAP server, which in turn connects through proxies to a web service. The weather service publishes data for subscribing web clients on the Internet and the web clients connect to subscribing actuators controlling room temperatures through the broker. ★★★
6. What are the functions of MQTT broker? ★
7. Consider Figure 3.6 and draw a diagram for data interchanges for connected RFIDs using MQTT broker. Each device sends the ID publishes at two-hour ★★

interval using RF links to the broker and gateway messages using MQTT Java, C or JavaScript library functions to inventory monitoring web application.

8. List five XMPP-IoT xeps which extend the use of XMPP to IoT and M2M messaging.
9. Refer Example 2.3 and Figure 3.7 and draw a data flow diagram showing the use of XMPP protocol for connected ATMs and bank server web objects for the messaging, presence notifications, responses on demand and service discoveries using XML streams. ★★★
10. Write code for XML message from two streetlights connected to a streetlight group. Assume sensor names are StreetLightN_I and StreetLightN_J. Assume sensor N_I output is 1 and Street Traffic density output of sensor N_J output is a 24 per 10 m. ★★

3.4 WEB CONNECTIVITY FOR CONNECTED-DEVICES NETWORK USING GATEWAY, SOAP, REST, HTTP RESTFUL AND WEBSOCKETS

LO 3.3 Outline the usage of communication gateway protocols, SOAP, REST, RESTful HTTP and WebSocket by connected devices and web

The following subsections describe the uses of gateway, SOAP, REST, RESTful HTTP and WebSockets for connecting web objects.

3.4.1 Communication Gateway

Figure 1.5 showed the gateway in the Oracle IoT architecture. The figure also showed gateway provisions for applications and IoT communication frameworks, management and proxy functions.

Communication gateway can connect a device using CoAP client and server using HTTP facilitates the communication

Communication gateway connects two application layers, one at sender and the other at receiver. The gateway also enables use of two different protocols, one at sender and the other at receiver ends. The gateway facilitates the communication between web server using the TCP/IP protocol conversion gateway and IoT devices. It also facilitates communication between the devices using CoAP client and server using HTTP.

The gateway provisions for one or more of the following functions:

- Connects the sender and receiver ends using two different protocols. For example, IoT devices network maybe ZigBee network for connecting the devices. The network then connects to the web server through a gateway. The server posts and gets the data using

HTTP. A gateway facilitates the communication between IoT devices and web server. For example, (i) ZigBee to SOAP and IP or (ii) CoAP protocol conversion gateway for RESTful HTTP (Section 3.4.4).

- Functions as *proxy* between the system and server.

3.4.2 HTTP Request and Response Method

An application uses a protocol. The application layer in TCP/IP suite of protocols for Internet uses HTTP, FTP, SMTP, POP3, TELNET and a number of other protocols. HTTP is most widely used application layer protocol for communication over TCP/IP. An HTTP client connects to an HTTP server using TCP and then the client sends a resource after establishing an HTTP connection.

Example 3.6

Problem

Give the request and response communication codes when (a) HTTP request message communicates to a server and (b) message communicates a response to a client.

Solution

(a) Following code sends the request:

```
POST /item HTTP/1.1
Host: ii.jj.kk.mm
Content-Type: text/plain
Content-Length: 200
```

(b) The server first processes the request and then sends an HTTP response back to the client. The response also contains the status code and content information for that.:

```
200 OK
Content-Type: text/plain
Content-Length: 200
```

200 is standard success code in an HTTP connection. 400 is standard failure code. The status returns as follows in that case.

```
400 Bad Request
Content-Length: 02
```

Data Exchanges Between HTTP Web-Objects

An HTTP connection enables a one-way communication at an instance from client API to a server or from server to the API.

HTTP polling is a method for receiving new messages or updates from an HTTP server. Polling means finding whether new messages or updates are available and receiving them in case they are.

An HTTP transfer is stateless which means each data transfer is an independent request. Therefore, header overhead information and meta-data of previous state need to be present with each HTTP request. Metadata is data which describes the data for interpretation in future. Therefore, each data interchange needs large headers over 100s of byte, and thus greater latency in request-response exchanges.

Ways of transferring both ways at the same instant are:

- Multiple TCP connection
- HTTP requests at short, regular intervals so that responses are nearly in real time
- Polling at successive intervals
- HTTP long polling means API sends a request to the server, which keeps the request open for a set period
- Stream hidden in iFrame

Polling methods have high latencies and header sizes of 100s of bytes. An alternative to this is using Java Applets, Silverlight or Flash plugins.

3.4.3 SOAP

Applications need to exchange objects on the Internet using protocols such as HTTP. Applications may be using different languages and platforms. Simple Object Access (SOAP) is a W3C approved open-source protocol.[9]

SOAP is a protocol for exchange of objects between applications using XML. It is also a protocol for access to a web service. SOAP specifies the formats and way of communicating the messages. Its usage is independent of the application language and platform (OS and hardware). It is extensible and is also used for APIs for the web services and Service-Oriented Architecture (SOA).

SOAP enables development of applications and APIs. SOAP functions connect the GUI applications to web servers using the standards of the Internet—HTTP and XML. Microsoft's .NET architecture supports SOAP for Internet application development.

W3C[10] recommended SOAP v1.2 second edition specifications—Part 0 is a primer, part 1 is a messaging framework and Part 2 is adjuncts. SOAP v1.2 specifications provision for assertions and test collection, XML-binary Optimized Packaging, SOAP message transmission optimisation mechanism and resource representation SOAP header block.

SOAP uses a body element after the specifications in an envelope. The body element contains the SOAP message intended for ultimate endpoint of the message.

A SOAP request could be an HTTP POST or an HTTP GET request. The HTTP POST request specifies at least two HTTP headers: content type and content length. A SOAP method uses HTTP request/response after the HTTP binding with the SOAP.

Ways of communication request using SOAP and obtaining SOAP response

[9] http://www.w3schools.com/soap/default.asp
[10] https://www.w3.org/TR/soap/

The request and response complies with the SOAP encoding rules. Let's take an example to explain the structure and usage of SOAP.

Example 3.7 describes the coding for SOAP request and response.

Example 3.7

Problem

Recall Example 1.1 and assume that a gateway sends requests for the weather service response and receives the weather message from a weatherMsgService daily. (a) How does a SOAP request or response communicate between two applications: one at gateway weatherMsgG and the other at weatherMsgService using two web objects www.weatherMsgG.com and www.weatherMsgService.com? (b) How is a SOAP message structured? (c) Write the SOAP message for request from End1 to End2. (d) What can be the SOAP message formats for the XML stream as response using HTPP 1.1 binding? (e) How will the error communicate?

Solution

(a) Figure 3.8 (a) shows a SOAP request or response communication between weatherMsgG and weatherMsgService. SOAP sends request from End1 (URL www.weatherMsgG/ weatherMsgRequest to End2 (URL www. weatherMsgService.com/ weatherMsgID250715) for the weatherMsgID250715. The weatherMsgID250715 value will be received at End1.

(b) Figure 3.8 (b) shows the SOAP message structure.

(c) Assume that HTTP POST method is use to post the SOAP request message. Assume that Get Temperature Response is a method for retrieving the weather message of the weatherMsgID250715 from resources at End2. SOAP sends request for the weather message. The weatherMsgID250715 value will be received at End1 from End2. Figure 3.8 (c) and (d) show SOAP message content and elements.

(d) The response will be from End2 http://www.weatherService.com/weatherMsgResponse to the End1. End2 response for weatherMsgID250715 will have first line for status of message receipt. Next lines will be HTTP binding and then the response message. Fig. 3.8(e) shows SOAP response from weatherMsgG to weatherMsgService using the Internet and HTTP binding

(e) An error message from a SOAP message is carried inside a Fault element. The element has child elements as follows: <faultcode>, <faultstring>, <faultactor> and <detail>. Error can be version mismatch or MustUnderstand or client or server.

3.4.4 REST and RESTful HTTP Web Applications

RESTful APIs

W3C Technical Architecture Group (TAG) developed the Representational State Transfer (REST) architectural style.[11] The group worked in parallel with HTTP 1.1. REST is a coordinated set of constraints which are used during design of software components in a distributed hypermedia, and the design depends on the characteristics of stateless, client-server, cacheable communication using a protocol. World Wide Web uses REST practices and constraints. REST is a simpler alternative for SOAP and Web Services Description Language (WSDL). REST style web resources and Resource-Oriented Architecture (ROA) have increasingly replacing SOAP.

[11] https://www.w3.org/2001/sw/wiki/REST

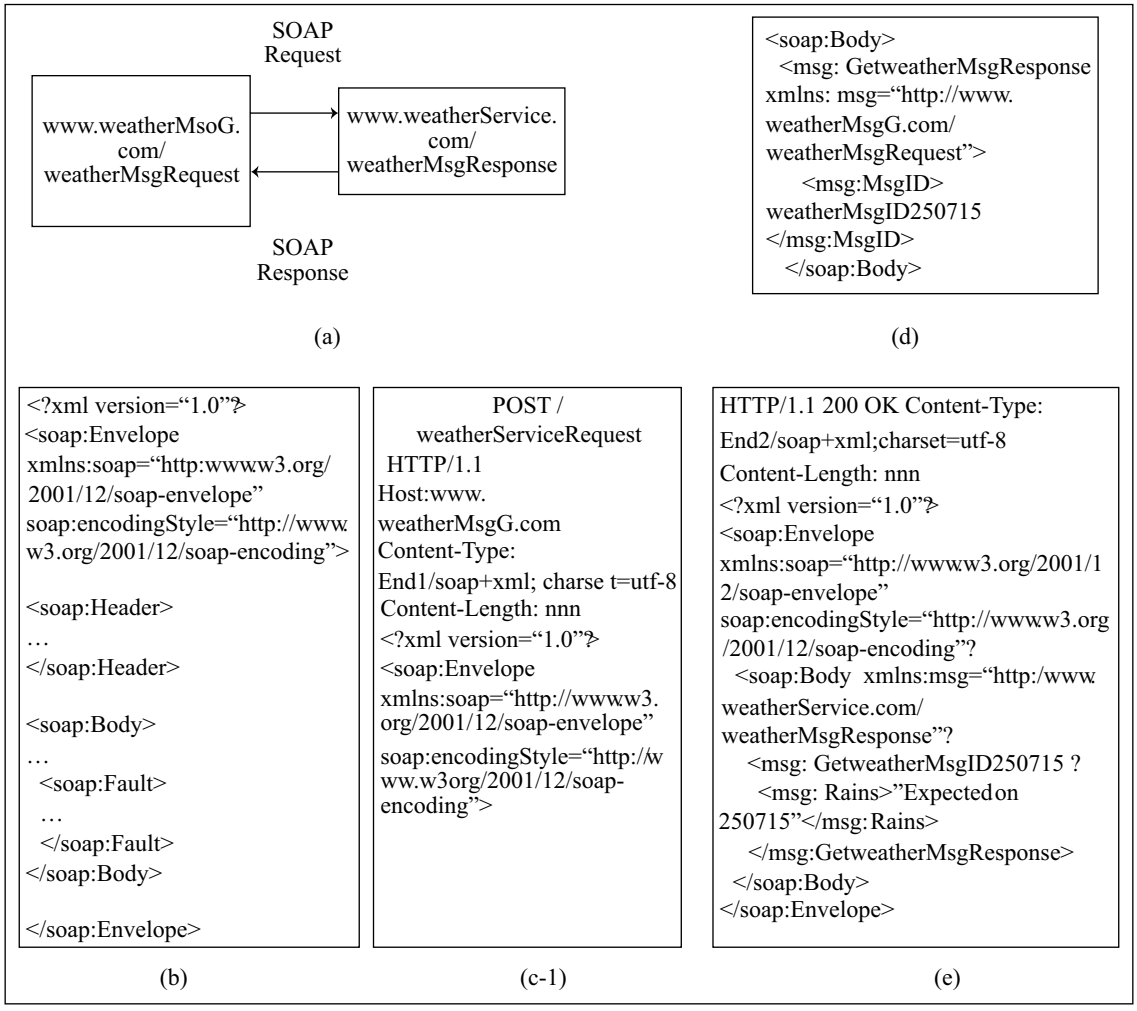


Figure 3.8 (a) A SOAP request or response communication between gateway weatherMsgG and weather service weatherMsgService, (b) SOAP message structure, (c) SOAP POST from weatherMsgG, (d) Request Get weatherMsgG, and (e) SOAP response weatherMsgID250715 at weatherMsgService to weatherMsgG using Internet and HTTP binding

The architectural properties of REST are realised by applying specific interaction constraints to data elements, components, connectors and objects. An architectural property of REST is separation of concern which means data storage at server is not a concern for the client, and client components can port on other objects.

REST is a simpler alternative for SOAP and Web Services Description Language (WSDL)

Also, user interface and user state are of no concern to the server. REST's client-server *separation of concerns* simplifies the component implementation, increases the scalability

of pure server components, reduces the complexity of connector semantics and improves the effectiveness of performance tuning.

The resources themselves are conceptually separate from the representations that are returned to the client. For example, the server may send data HTML, XML, TLV or JSON from its database which can have other distinct internal representation at a connected database server.

REST software architecture style provisions for the use of specific practices, such as client-server mode of communication and layered system architectural approach.

Client-server interactions have characteristics of performance and creation of scalable web objects and services. Scalability means ability to support greater number of interactions among components and greater number of components.

Layered system refers to a client which can connect through an intermediate layer (proxy, firewall, gateway, proxy or intermediary server). REST enables intermediate layer processing by constraining the messages such that the interactions at each layer are self-descriptive. A client may not ordinarily be made aware whether it is connected directly to an end-server. Intermediate layers may render assistance in transcoding and enabling usages of different protocols at two ends. Use of the intermediate system improves performance when using bigger scales and shared caches. Shared caches means caches shared between two layers; for example intermediate layer and server or proxy and server.

Representation of each application state contains links that may be used for a next-time interaction when the client selects those links and initiates a new state transition.

RESTful

When all interactions used in the applications conform fully to the REST constraints then these are called RESTful. RESTful APIs comply with these constraints and thus conform to the REST architectural style. Web services with RESTful APIs adhere to the REST architectural constraints.

REST architectural style can be used for HTTP access by GET, POST, PUT and DELETE methods for resources and building web services.

RESTful HTTP APIs

Standard HTTP methods are GET, PUT, POST and DELETE. HTTP based RESTful APIs use the following:

- URIs/URLs, such as http://weatherMsgService.com/weatherMsg/ and hypertext links to reference state and reference related resources, and JSON, TLV or an Internet media type (MIME type) hypertext links
- REST-based web objects communicate typically, but not always, over the HTTP. The World Wide Web itself represents the largest implementation of a system conforming to the REST architectural style. RESTful HTTP system feature is that communication is over the HTTP and use verbs (commands) same as in HTTP, namely GET, POST, PUT and DELETE.

RESTful HTTP Verbs

REST interfaces usually involve *resource repositories* with identifiers. For example, /deviceNetwork/device or /TemperatureApp, which can be operated upon using standard verbs as follows:

- GET command is to get a *list* of the URIs for *resource repository* of the resources and perhaps other details of the members in the *repository*. GET *retrieves* a representation of the resource item (means addressed member) of the *repository*. Representation is expressed in an appropriate Internet media type.
- POST command *creates* a new entry in the *resource repository* for the resources. The new entry's URI is assigned automatically and is usually returned by the operation. An option which is not generally used considers the resource item as a repository on its own right and *create* a new entry in it (not generally used).
- PUT command which *replaces* the entire *resource repository* with another *resource repository* or *replaces* the resource item of the *repository* (or if it does not exist, then *create* it).
- DELETE command from client *retrieves* web objects and sends data to remote servers.

3.4.5 WebSocket

RFC 6455 describes the specifications of the web protocol[12]. WebSocket is an IETF accepted protocol. WebSocket API (WSAPI) W3C standard is Web Interface Definition Language (Web IDL)[13].

Bidirectional communication request using WebSocket at client and obtaining response from WebSocket at server

Instant messaging and many applications need bidirectional data exchanges over the same connection. WebSocket enables bidirectional communication over a single TCP connection.

Figure 3.9 (a) shows Opcode and other fields at a WebSocket Frame. Figure 3.9 (b) shows WebSocket API provisions for *events*, *attributes* and *functions*. Event means occurrence of new condition, occurrence of which is listened by an event-listener function, and as soon as listening takes place, an event handling function executes. The handling function is also *callback action*. Figure 3.9 (c) shows data bidirectional communication using WebSocket APIs between the web objects and also the browsers and servers.

The WebSocket-protocol-based design usage now supersedes the existing bidirectional communication technologies that use HTTP at application layer. WSAPI attributes are *url*, *protocol*, *readystate*, *buffered amount* and are extendable. The API functions are send (frame) and close (socket). A frame contains header plus data. WebSocket frame contains 2 B header extended to 0, 2 or 4 B plus data bytes. Client side adds 4 B mark; for example, for identifying a frame. The WebSockets enable easier usage of existing infrastructure, authentication, filtering and proxies. Number of popular browsers support the protocol. HTTP case polling at periodic intervals are not required due to no-wait feature of WebSocket.

[12] www:/tools.ietf.org/html/rfc6455
[13] http://www.w3.org/TR/WebIDL/

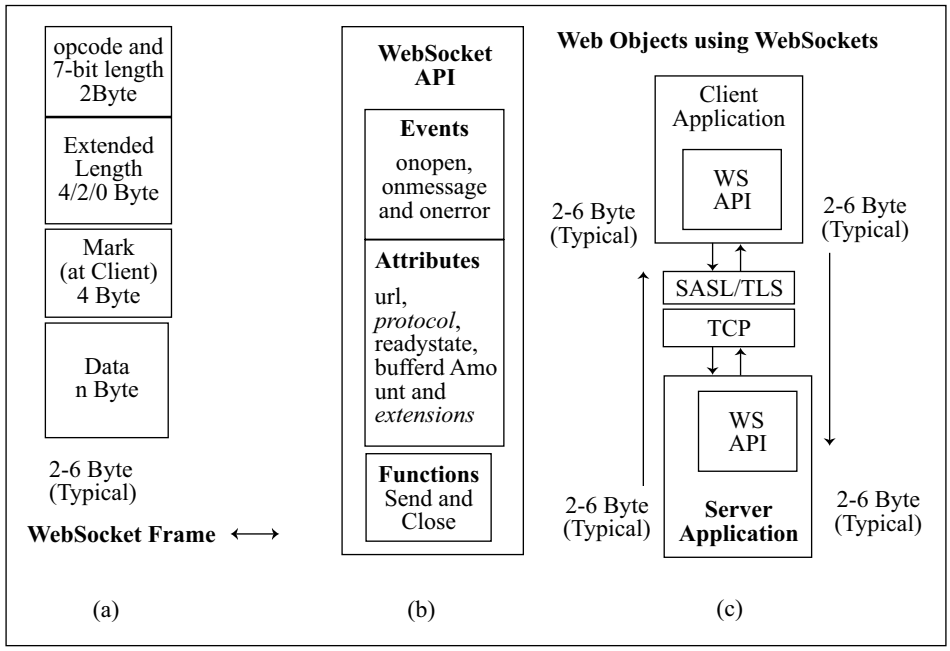


Figure 3.9 (a) Opcode and other fields at a WebSocket frame, (b) WebSocket API events, attributes and functions, and (c) Data bidirectional communication using WebSocket between web objects

Features of WebSocket are:

- Small header size (2B extended to 6 byte and above compared to 100s B and above for an HTTP header which results in high latency
- No new connection which will need a new header and thus no new latency period
- WSAPIs, because of very low connection latencies, facilitate live content and the creation of real-time games.
- Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an upgrade request.
- Protocol uses default port 80 for regular WebSocket connections using ws:// and port 443 when using wss:// for WebSocket connections tunnelled over Transport Layer Security (TLS).
- Protocol is intended to be compatible with HTTP-based server-side software and intermediaries, so that a single port can be used by both HTTP clients talking to that server and WebSocket clients talking to that server. Therefore, WebSocket client's handshake is an HTTP upgrade request. Response from the server is also as an HTTP upgrade (Example 3.7).
- Protocol specifies six frame types and leaves ten reserved for future use (version 13).
- Clients and servers exchange the 'messages' after a successful handshake. A WebSocket message does not necessarily correspond to a particular network layer framing. A frame has an associated type. Each frame belonging to the same message contains the

same type of data. Six types are (i) textual data (UTF-8) and (ii) binary data (whose interpretation is left up to the application).

- Extensibility of request-response (client-server) architecture to iq (information through and querying) chat and super chat extensibility to cloud services.
- Support from intelligent and business analyst applications and processing through web server or XMPP server and gateway for connecting the device network with the IP network.

Reconfirm Your Understanding

- A communication gateway connects the sender and receiver ends using two different protocols. The gateway can function as a proxy between two ends, such as HTTP-CoAP proxy.
- Web objects communicate using HTTP methods, such as POST, GET, PUT, DELETE, SOAP and WebSockets.
- REST is a software architecture specifying practices and constraints which are used during design of software components in a distributed hypermedia. Designing based on REST depends on the characteristics of stateless, client-server, cacheable communication using a protocol.
- RESTful means components designed with constraints specified in REST. For example, separation of concerns, use of layered system, defining cache ability or not between client, server and intermediate.
- Web objects communicate using WebSockets. Instant messaging and many applications need the bidirectional, very low latency, real time data exchanges over the same (without initiating new) connection. WebSocket enables bidirectional communication over a single TCP connection.

Self-Assessment Exercise

1. What are the functions of GET, POST, PUT and DELETE? ★
2. How will a weather service send the weather data probability of rain at a location using HTTP? Using HTTP post message, assume host is iii.jjj.kkk.com/, content type is text and length is 10 B. What can be the possible responses from the other end? ★★
3. How is a SOAP message structured? ★
4. Assume that an ATM gateway sends requests for the banking service response and receives the account information from a BankMsgService. How does a SOAP request or response communicate between two applications, viz. one at Gateway ATM_G and other at BankMsgService using two web objects, viz. one at www.ATMG.com and other www.BankService.com? ★★★

5. List the characteristics in interfaces designed using REST. ★
6. List WebSocket API events, attributes and functions, and the usage of each. ★★
7. Give five examples when you need bidirectional data exchanges over the same TCP connection? ★
8. List the features of a WebSocket. How do these differ from those of HTTP Request/Response and HTTP polling? ★★★

Key Concepts

- Client
- CoAP
- CoAP-MQ
- CoAP-SMS
- Communication gateway
- Content
- CoRE
- DTLS
- Header
- HTTP
- Hypertext
- Instant messaging
- Local area devices network
- LWM2M protocol
- Message broker
- Message queue
- MQTT
- Multi user chat room
- Object
- Proxy
- Publish/subscribe
- Push/pull
- Request/response
- REST
- RESTful HTTP
- Resource
- Resource directory
- Resource discovery
- Resource repository
- Registration
- Representation
- ROLL
- Server
- SMS
- SOAP
- State
- Stateless
- Subscribe/notify
- Subscription
- Topic
- TLS
- UDP
- URI/URL
- WebSocket
- XML
- XMPP

Learning Outcomes

LO 3.1

- Devices generally have a constraint that they sleep most of the time in low power environment and wake when required (client initiates). Devices may have constraints that their connectivity breaks for long periods, has limited up intervals in lossy environment and limited data in size.
- IoT/M2M devices connect using the CoAP and LWM2M protocols in the constrained RESTful environment (CoRE) and the data routes over a low power lossy network (ROLL). Client/server communication uses UDP and DTLS layers.
- CoRE objects use standard constrained environment protocols for sending a request or response. The CoAP-client and CoAP-server communication takes place over the network and transport layers to the other-end CoAP client and server. An HTTP-CoAP proxy connects an

HTTP client with the CoAP object. A CoAP-HTTP proxy connects a CoAP client with an HTTP object. A CoAP Client/server use the URIs coap://… HTTP Client/server use the URLs http://

- An LWM2M client has object instances as per the OMA standard. An LWM2M client–server interaction takes place over the access and CoRE networks.

LO 3.2

- IoT/M2M objects and applications use the Request/Response, Publish/Subscribe, Subscribe/Notify, Push or Pull, Polling, Message Queues, Instant Messaging, Presence notification for Multi-User Chat and instant messaging methods for transfer of messages, resources, representations, objects and states.
- The APIs for messaging use CoAP-SMS, CoAP-MQ, MQTT and XMPP as the messaging protocols in constrained environment and ROLL network.
- CoAP-SMS is a protocol when CoAP object uses IP as well as cellular networks and an SMS (Short Message Service).
- A CoAP client communicates to a mobile-terminal end-point (MT).
- MQTT is a protocol for the constrained environment PubSub messaging. A *Subscriber* receives the messages on a subscribed topic.
- XMPP CoRE, Presence and Address format are IETF recommendations for the XMPP Messages server. The server notify *presence* to the subscribers/registered clients (users) for the purpose of IMs (instant messaging) to one or many users at the same time. The server enables Instant Messaging (IM), chatting and multi-user chat (MUC) after the creation of a chat room, where different users after subscriptions do IMs.

LO 3.3

- APIs in connected-devices network communicate with the web using the communication gateway, SOAP, REST, RESTful HTTP and Web Sockets.
- A communication gateway connects the sender and receiver ends and can also function in proxy between the two ends, such as HTTP-CoAP proxy.
- Web objects communicate using HTTP methods such as POST, GET, PUT, DELETE, SOAP, and WebSockets are used for web communication.
- RESTful components provide simplicity and uniformity. The features are separation of concerns, use of layered systems, defining cache ability or not between client, server and an intermediate such as firewall or proxy.
- An HTTP connection enables one-way communication at an instance from client the API to a server or from the server to the API. HTTP transfer is stateless. A client polls the resources at the server and finds the changes in their states. When a state exhibits a change then the client retrieves that resource.
- SOAP is protocol for exchange of objects between applications using XML SOAP body elements. Specifications are in an envelope.
- WebSockets are used for bidirectional communication and instant messaging and chatting over a single TCP connection. WebSockets use the URIs ws://… and use the URIs wss://… when using the TLS.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. CoRE environment constraints are: (i) data is limited in size around 1000 B, (ii) web clients and web servers use HTTP (Hyper Text Transfer Protocol), TCP (Transmission Control Protocol) and IP (Internet Protocol), (iii) data routes as in an ROLL network, (iv) devices sleep most of the time, (v) wake when required (client initiates), (vi) connectivity for long periods, (vii) can have long up intervals. ★
   (a) (i) and (ii) true
   (b) All true except (vii)
   (c) (i) to (iv) true
   (d) (iii) true
2. CoAP features are: ★
   (a) W3C defined CoAP web-objects
   (b) A specialised web transfer protocol used for CoRE using ROLL network.
   (c) Use of object model for resources and each object can have single instance.
   (d) Object or resource use CoAP-TLS (security binding with RSA and Certificate), and TCP protocols for sending a request or response.
3. LWM2M specifications and features are as follows: (i) An object or resource use CoAP, DTLS, TLS, TCP and UDP or SMS protocols for sending a request or response, (ii) use of plain text for a resource or use of JSON (Java Script Object Notation) during a single data transfer or binary TLV format data transfer for a package for a batch of resource-representations in a single data transfer, (iii) objects or its resource access using URLs or file names, (iv) use of object model for resources and each object can have single instance, (v) LWM2M client-server interactions are over the access and CoRE networks which generally use CoAP over ROLL, and (vi) M2M management functions can be MSBF (M2M Service Bootstrap Function) for credentials of the devices and gateway, and MAS (M2M Authentication Server) for security, root key data store and device and data authentication. ★★
   (a) (i), (iii), (iv) not true
   (b) All true
   (c) Only (v) and (vi) true
   (d) All true except (ii)
4. CoAP-SMS protocol specifies as follows: ★★
   (i) Computer or machine interface using IP receiving request or sending response as CoAP data to a mobile service provider using SMPP for data interchange.
   (ii) The SMS-C communicates that to MO using SS7. SMS-SP receives request or sends response to machine or IoT device or mobile origin.
   (iii) Computer or machine interface using IP receiving request or sending response as HTTP request to a mobile service provider using CIMD for data interchange.

(iv) The SMS-C communicates that to Mo using CIMP or SMPP. SMS-SP receives request or sends response to machine or IoT device or mobile origin.

(a) (i) not true
(b) (ii) not true
(c) (iii) not true
(d) All true

5. CoAP-MQ Broker (MQ) does the following: (i) functions as a server node capable of storing messages to and from other nodes, (ii) performs a store, compact, re-format and forward function between web clients and the CoAP-MQ capable endpoints, (iii) matches subscriptions and publications in order to route messages to right endpoints, (iv) does not send state of an endpoint when endpoint or a web client subscribes to the state of the endpoint, (v) enables the web client publishing of updates to the endpoint state through the MQ, (vi) returns the last published value to web clients or other endpoints on behalf of endpoints that are sleeping, and (vii) acts as a firewall. ★★
   (a) (i) to (v) true
   (b) (ii), (iv) and (vii) not true
   (c) All true
   (d) Only (vii) not true
6. MQTT features are as follows: (i) M2Mqtt library set of functions for coding in visual C++ need just 100 kB and in Java just 30 kB, (ii) permits maximum number of exchanges, (iii) broker-based subscribe/notify messaging protocol, (iv) don't notify on an abnormal disconnection of a client, (v) does not have specifications for the final action to be taken on failure to send messages, (vi) UDP connectivity for the basic network, (vii) synchronous communication for real-time messaging at regular intervals. ★★
   (a) All true
   (b) All true except (iv) and (v)
   (c) None true
   (d) (i) and (vii) not true
7. XMPP-IoT server: (i) consists of xeps and services, (ii) extensible to publisher-subscriber, (iii) does not support multi-user chat, (iii) supports devices and connected devices area network, (iv) support through a gateway between connected devices and IP networks communicates with XMPP APIs in web objects, (v) support SASL and TLS security protocols for the APIs and web object messages using TCP, (vi) HTML request/response, (vii) XML streams in XMPP format communicates. ★★
   (a) All are true
   (b) (iii) and (vi) not true
   (c) (iii) and (v) not true
   (d) (iii) not true
8. Gateway does one or more actions as followings: (i) CoAP protocol conversion for RESTful HTTP client, (ii) HTTP-CoAP proxy, (iii) provisions RESTful interfaces between client and server, (iv) connectivity between ZigBee devices and IP ★★

network, (v) forwarding of HTTP requests/responses, (vi) functions as firewall, and (vii) connects WebSockets.
(a) (i) to (v) true
(b) (ii) and (vi) not true
(c) (iii), (v) to (vii) not true
(d) (i) to (v) and (vii) true

9. (i) SOAP message structure has the following tags: envelope, encoding style, header, body and fault (ii) dot net support SOAP for internet application development SOAP, (iii) object exchange could be through HTTP request/response when HTTP binding specified, and (iv) SOAP support the REST-style web resources. ★★
(a) All except (iv) true
(b) Encoding style and fault tags not present
(c) (ii) not true
(d) (iii) not true

10. Simplicity of REST interfaces is because of: (i) modifiability of components, (ii) visibility of communication between components by service agents, (iii) portability of components by moving the objects, (iv) reliability, (v) scalability, and (vi) separation of concerns between interacting components. ★
(a) Only (v) true
(b) All except (iii) true
(c) All except (i) and (ii) true
(d) All except (vi) true

11. HTTP objects can transfer both ways at the same instant using (i) polling, (ii) Multiple TCP connection, (iii) HTTP requests at short regular intervals so that responses are nearly in real time, (iv) polling at successive intervals, (v) HTTP long polling means API sends a request to the server, server keeps the request open for a set period, and (vi) stream hidden in iframe. ★★
(a) All except (i) true
(b) All except (i) and (iv) true
(c) cannot transfer simultaneously
(d) All except (i) and (vi), and (vi) true

12. WebSocket can include at a higher sublayer, the: (i) XMPP web object APIs, (ii) XMPP server and gateway for connecting device network with IP network, (iii) can extend messaging and presence protocol, (iv) can include JMS (Java Message Service) for messaging and PubSub messaging, (v) FTP (File Transfer Protocol) for file transfer, (vi) other protocols supported at the client and server, (vii) extensibility of request-response (client-server) architecture to iq (Information through querying), (viii) chat, (ix) super chat, (x) extensibility to cloud services, (xi) support from intelligent and business analyst applications, and (xii) support from business processing ★★★
(a) All true
(b) All except (v), (vi) and (x) true
(c) All except (x) to (xii) true
(d) All except (i), (ii), (x) to (xii) true

Short-Answer Questions

1. What does constrained environment mean for IoT/M2M? **[LO 3.1]** ★
2. What are the features of REST architectural style? **[LO 3.1]** ★★
3. How will a CoAP client send a request to an HTTP server using intermediate components? **[LO 3.1]** ★★★
4. What are the additional features in LWM2M over CoAP? **[LO 3.1]** ★★
5. Show diagrammatically how a device sends an SMS to a mobile terminal and how a mobile origin sends a message to an actuator device. **[LO 3.2]** ★★
6. List the features of MQTT. **[LO 3.2]** ★
7. How do the connected devices connect to server-end functions in IoT for business processes? **[LO 3.2]** ★★★
8. Specify functions of CoAP, RESTful HTTP, MQTT and XMPP (Extensible Messaging and Presence Protocol) in IoT apps. **[LO 3.2]** ★★
9. What are the protocols needed in web connectivity in IoT applications? **[LO 3.2]** ★
10. How do the actions of PSK, RPK and certification protocols compare? **[LO 3.2]** ★★★
11. How does XMPP-IoT apply in IoT applications? **[LO 3.2]** ★★
12. What are the functions of a communication gateway? **[LO 3.2]** ★
13. Compare the use of an HTTP request response and a SOAP object transfer. **[LO 3.2]** ★★
14. How can an HTTP request/response be used for bidirectional data transfers? **[LO 3.3]** ★
15. What are features in WebSocket that enable extensibility to cloud services, support from intelligent and business analyst applications, and support from business processing? **[LO 3.3]** ★★★

Review Questions

1. Why do constrained environment and data ROLL network need a distinct set of protocols? [**LO 3.1**] ★
2. Show diagrammatically and explain the communication gateway and proxies between CoAP objects and web applications. [**LO 3.1**] ★★
3. Show diagrammatically and explain the features of LWM2M. **[LO 3.1]** ★★
4. Show diagrammatically and explain the SMS communication between MO and CoAP object, CoAP object and MT, CoAP object and Mobile service provider. **[LO3.1]** ★★★
5. Show by a diagram the object exchanges between devices, CoAP-MQ Broker and web applications. [**LO 3.2**] ★★
6. Draw message exchanges between devices, MQTT broker and web applications. [**LO 3.2**] ★★
7. Explain REST architectural style of coding for client/server interactions. [**LO 3.3**] ★★★
8. What are the applications of SOAP? How does the SOAP message code and sent? **[LO 3.3]** ★
9. Describe the features of object exchanges using HTTP request/response and WebSockets. [**LO 3.3**] ★★

Practice Exercises

1. List the protocols behind M2M communication with applications and service capabilities. **[LO 3.1]** ★
2. Explain when CoAP-MQ and MQTT are used. **[LO 3.1]** ★★

3. Draw an architectural view of IDs data transfer for web applications in Internet of RFIDs (Example 2.2). **[LO 3.1]** ★★★
4. Recall Example 1.1. Draw an architectural view of data exchanges between the umbrella, web service for forecasting of rain and hot sunny days and mobile. **[LO 3.2]** ★★
5. How is XMPP used for IoT applications? **[LO 3.2]** ★★
6. How is HTTP POST method used for publishing weather service data? **[LO 3.3]** ★
7. Develop an architecture for bidirectional data exchanges between the CoAP client and WebSocket. **[LO 3.3]** ★★★

4

Internet Connectivity Principles

Learning Objectives

LO 4.1 Explain Internet-connectivity protocols—IP, IPv6, RPL, 6LoWPAN, TCP/IP suite, TCP and UDP

LO 4.2 Describe the functions of IP address, MAC address, DNS and DHCP

LO 4.3 Explain the functions of application layer protocols—HTTP, HTTPS, FTP, Telnet and Ports

Recall from Previous Chapters

We learnt about conceptual framework, reference model and reference architecture in Sections 1.1 and 1.2 in Chapter 1.

Equation 1.2 suggested that data gathers from the devices, enriches at the data adaptation stage and the enriched data-stream transfers to manage that at higher stages of data analysis the streaming process uses Internet connectivity.

Equation 1.3 suggested that data gathers from the devices and consolidates at the data adaptation stage. Then a gateway connects through Internet to the server and other stages where data of the stream collects, assembles, manages and analyses.

We learnt that architecture of IoT/M2M has network and transport layers for data streaming to server, application or other entities. We learnt about IoT architecture layered model, in which application layer is highest and physical cum data-link layer is the lowest (Figure 2.1). Thus, data receives at a $layer_i$ from previous $layer_j$ ($i > j$), when it receives at applications and services layer from the physical cum data-link layer (Section 2.1).

We learnt that connected devices data from gateway use the network and transport protocols for connecting to the subsystems where data of the stream collects, assembles, manages and analyses, and where the applications execute and provide the services. (Section 2.2). An important aspect is that Internet connectivity is the key architectural element between the applications and devices.

We learnt in Chapter 3 that Internet connectivity is a basic need from the device-end to applications or services-end. The connecting entities use the WPAN protocols for personnel area network, and Internet protocols for connectivity with the applications or services. Client-server, pub/sub and other communication modes use the IP protocol at network layer, and TCP or UDP at the transport layer, TLS or DTLS for secure connection, and HTTP, WebSocket and other application-layer protocols for applications support.

4.1 INTRODUCTION

Internet is a global network with a set of connectivity protocols for:

- Connected devices gateway for sending the data frames of the devices or to the devices. The data communicate over the network as packets, which communicate through a set of routers at the Internet. The processes manage, acquire, organise and analyse the data of the IoT devices for applications, services and business processes.
- The devices perform the controlling and monitoring functions using the messages, data-stacks and commands sent through the Internet by the applications, services or business processes.

Following are some of the key terms and their meanings, which are required for understanding Internet connectivity principles for communication between connected devices network and IoT applications.

Header refers to words, which are required for processing a received data stack at a layer and which envelopes the data stack of the preceding upper layer before transfer to the succeeding lower layer. Header consists of *header fields.* Each word has 32 bits. Each header word can have one or more fields. The fields in the words are as per the processing required at succeeding stages up to the destination.

IP header refers to header fields, which comprise parameters and their encodings as per the IP protocol. IP is Internet layer protocol at the source or destination.

TCP header denotes header fields containing parameters whose encoding is as per the TCP protocol. TCP is transport layer protocol at the source or destination.

Protocol Data Unit (PDU) is the unit of data stack maximum number of bytes, which can be processed at a layer as per the protocol at a layer or sublayer.

TCP stream is a sequence of bytes or words in the data stack created at the transport layer that transmits to the destination-end transport layer.

Maximum Transferable Unit (MTU) is the unit of data-stack maximum number of bytes, which can be transferred from a higher layer to lower layer or physical network.

Packet is a set of bytes with a fixed maximum specified size that transfers from network layer and communicates from one router to another, until it reaches at physical, data-link and network layer at the receiver's end. Internet technology provisions for packetising the received data stack at Internet layer for transmission to the next lower layer. Receiving-end Internet layer does de-packetising (unpacking) at the layer for sending it to the succeeding transport layer.

IP packet is a data stack, which includes IP header. It communicates from a source IP address through the routers to the destination IP address.

Data segment refers to data stack from application-support layer for transport. Application data is divided into the segments when its size is more than the transportable limit.

Network interface is a system software component or hardware for facilitating communication between two protocol layers/computers/nodes in a network. The interface software-component provides standard functions. For example, connection establishment, closing and message passing. Network interface examples are port (software or hardware component), network interface device and socket. An interface can be addressed by a unique port number/socket name/node id.

Port is an interface to the network using a protocol that sends an application layer data stack to the lower layer for transmission. The port receives the data stack at the receiver's end from the lower layer. Each port uses an assigned number according to a protocol, which is used for transmission or reception at the application layer. For example, Port 80 is assigned number to HTTP, an application layer protocol.

Socket is a software interface to the network that links to data stack using a port protocol and an IP address. Internet data can be considered as communicating between the sockets. Application data can be considered to flow between the sockets at sender and receiver.

Host is a device or node that connects to a network of computers. It provides information, resources, services and applications to the other nodes on the network. A network layer assigns a host address to each host.

IP host is the one that uses the Internet protocol suite. An IP host has one or more IP addresses for the network interfaces.

Subnet is a subnetwork, which is logical and visible subdivision of an IP network. Subdivision enables addressing a set of networked computers in the subnet using a common and identical IP address. For example, an organisation has 1024 computers but uses a common and identical IP address(es). An IP address consists of two groups: msbs and lsbs, total 32-bits; msbs means most significant bits in a word, while lsbs means least significant bits in a set of bits.

Routing Prefix: Thirty-two bits IP address can be divided into the msbs consisting of 8, 16 or 24 bits and remaining lsbs. The division results in the logical division of an IP address into two fields—a network address or routing prefix field and a rest field or *host identifier*. The *rest* field is an identifier for a specific host or network interface.[1]

Host Identifier: The *rest* field may also have two sub-fields—one for subnet id and other for the host identifier. When a network subdivides into subnets and subnet has a number of hosts.

Data Flow Graph (DFG) denotes a graphical representation using arrows from one stage to another. A circle represents a stage and arrow characterises the direction of flow of data. Inputs (for example, incoming data for routing) at each stage compute (process) and cause the outputs from the stage. Inputs at the beginning of a circle continue to flow to the next circle, until the outputs (for example, outgoing data for routing) reach at the end of the circle. Data can be said to flow from beginning to end after being processed in the multiple stages in-between.

Acyclic Data Flow Graph (ADFG) refers to a DFG where only one set of inputs generate only one set of outputs for the given input-set in the DFG model. All inputs are instantaneously available in APDFG, (no delay between various inputs) at each stage, except the processing interval at the stage. Examples of non-acyclic data input are an event input, a status-flag set (= 1) or reset (= 0) in a device, and input as per output condition of the previous process.

Directed Acyclic Graph (DAG) means an ADFG in which none of the output cycles back to a previous processing stage or level or rank as an input during data flow.

Following sections describe Internet-connectivity principles for connected devices. For details of the protocols, their usages and implementation, the reader can refer to any standard book on computer network or internet and web technology.

4.2 INTERNET CONNECTIVITY

LO 4.1 Explain Internet-connectivity protocols—IP, IPv6, RPL, 6LoWPAN, TCP/IP suite, TCP and UDP

Figure 4.1 shows a source-end network-layer connected to the destination through a set of IP routers. It also shows that a communication framework uses an IP address and communicates with the IoT/M2M IoT application and services layer using TCP/IP suite of application protocols to a destination IP address.

Internet connectivity is through a set of routers in a global network of routers which carry data packets as per IP protocol from a source end to another and vice versa. A source sends data packets to a destination using IETF standardised formats.

[1] https://www.ietf.org/rfc/rfc7608.txt.pdf

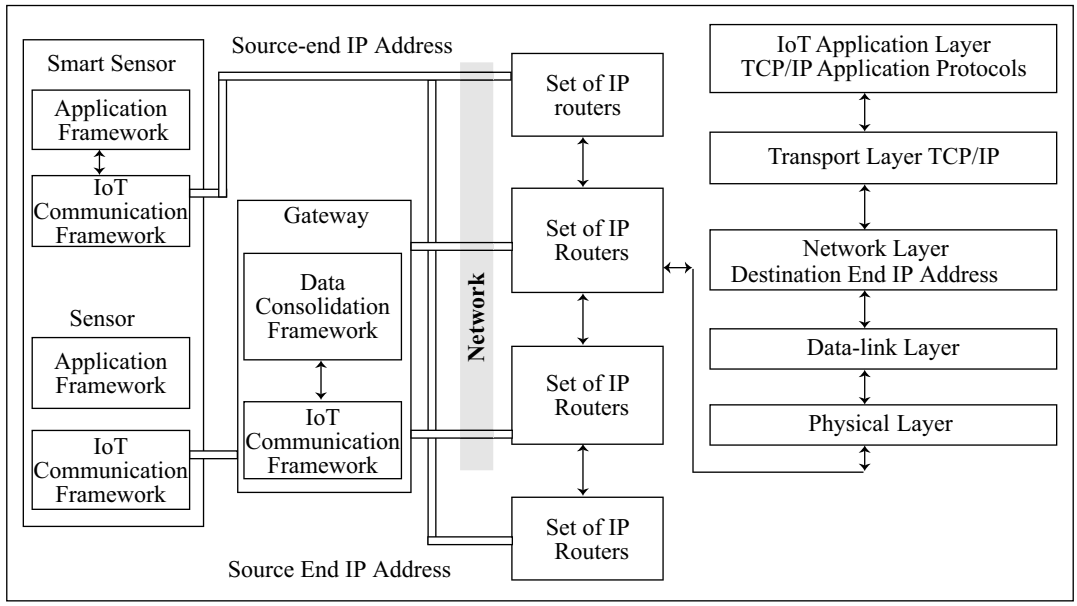


Figure 4.1 Source-end network-layer connected through a set of IP routers for data packets from an IP address and communicating with IoT/M2M IoT application and services layer using TCP/IP suite of application protocols

4.3 INTERNET-BASED COMMUNICATION

When data transmits from layer$_i$ to the next layer$_j$, following are the actions that occur:

- Each layer's processing of data is as per the protocol used for communication by that layer.
- Each layer sends the data stack received from the previous upper layer plus a new header, and thus creates a fresh stack after performing the actions specified at that layer.
- Layer$_j$ will specify new parameters as per protocol and create fresh stack for the subsequent lower layer.
- The process continues until data communicates over the complete network.

Remember that IoT application layer is the highest in the modified OSI model and the physical layer is the lowest (Figure 2.1). Thus, data transmits from layer$_i$ to the next layer$_j$, (i > j) when it transmits from the application layer to the physical layer.

When data is received at the next layer$_i$ from a layer$_j$, i.e. IoT device physical layer to the IoT application, the following actions are performed:

- Each layer performs the processing as per the header field bits, which are received according to the protocol to be used for decoding the fields for the required actions at that layer.

- Each layer receives the data stack from the previous lower layer and after the required actions, it subtracts the header words and creates a new stack specified for the next higher layer.
- The process continues until the data is received at the port on the highest application layer.

Upper layers use the header words alone. Lower layer, such as data-link layer protocol, such as Ethernet 802.3, provisions for the trailing bits also, in addition to the header words. Trailing-bits usage can be as error-control bits and end-of-the frame indicating bits.

Note: Ethernet frame communicates between the data-link layers of two computers on a local area network (LAN). 802.3 specifies the maximum frame size of 1518 B (later increased to 1522 B) and consists of 32 trailing bits. These are called Frame Sequence Check (FRC) bits or Cyclic Redundancy Check (CRC) bits. CRC bits append during transmission. The receiving-end recalculates the CRC from the received data bits. If both of these match, then the frame in the sequence is accepted, else an error is reported.

Only four OSI model layers, such as 7, 4, 3 and 2 are specified at the TCP/IP protocol suite for Internet communication. Layer 1 is per communication protocol for physical link to routers. Figure 4.2 shows the communication between the source and destination. Internet-based TCP/IP communication uses application layer L7, transport L4, Internet L3 and link L2 layers. Figure 4.2 shows the PDUs at the layers.

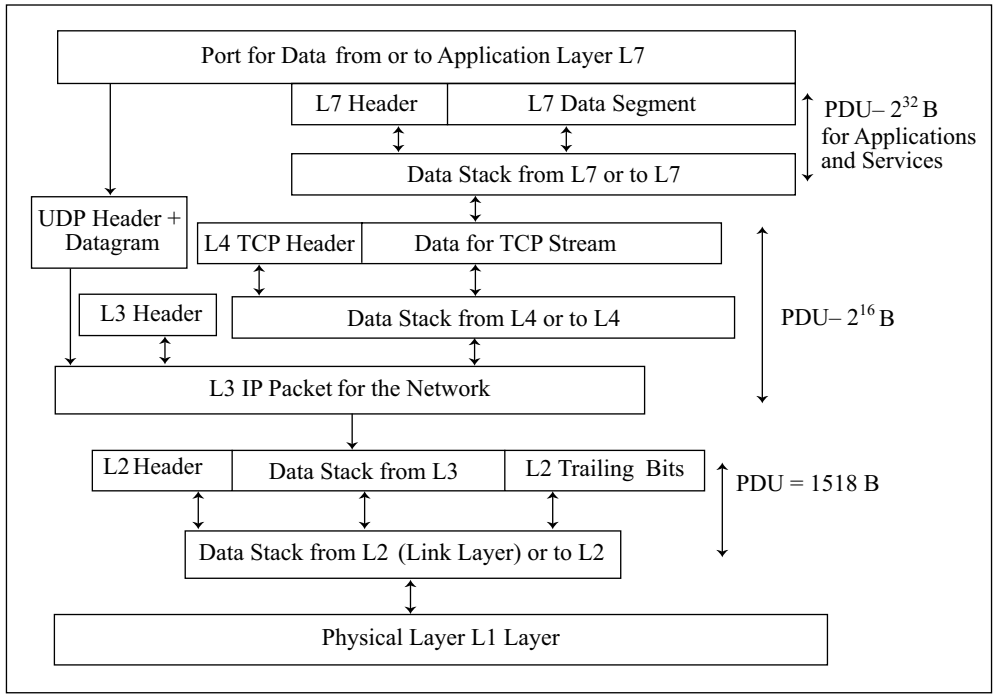


Figure 4.2 TCI/IP suite four layers generating data stack for the network and for physical layer during Internet communication

A data segment (maximum 2^{32} B per segment) is accepted from L7 layer for transport layer TCP. L4 then generates a TCP stream. The stream packetizes at the Internet layer L3. Alternatively, a datagram (maximum 2^{14} B) is accepted from L7 layer for transport layer UDP usage. L4 then generates a UDP datagram (maximum 2^{16}B). The stream packetises at the Internet layer L3 into packets. Packet sent from L3 has maximum size 2^{16}B including L3 header. The datagram sent from L3 also has maximum size 2^{16}B.

Internet layer uses the IP protocol (IPv4 or IPv6 or RPL). Packet routing occurs as: Each router has information about the path to the destination. When a number of paths are available, then the number of packets of the same source simultaneously follows different paths from a router. Destination-end transport layer reassembles the packet according to their sequences in the source transport layer data streams. Then the reassembled data segment transfers to IoT application layer at the destination end.

The data-link layer uses a protocol each in its sublayers. For example, Ethernet IEEE 802.3, MAC, PPP, ARP, RARP, NDP etc. Ethernet is protocol for logical-link layer on a LAN; MAC stands for Media Access Control; PPP stands for Point-to-Point Protocol; ARP stands for Address Resolution Protocol; RARP stands for Reverse Address Resolution Protocol. Resolution means using network data stack to find the MAC address. Reverse means MAC address used for finding IP address. NDP stands for Network Discovery Protocol.

The next subsections describe the features in IPv4, IPv6 and RPL.

4.3.1 Internet Protocols

Internet layer receives and forwards data to the next layer using IP version 4 (IPv4) or IP version 6 (IPv6) protocol.

Internet Protocol Version 4

Let n = total number of header words. Figure 4.3 shows data stack received or transmitted at or to network layer and IP packet consists of IP header fields of n words (= 160 bits and extended header option words). The extension is done when required actions and using data stack from or for the transport layer.

Internet layer protocol is abbreviated as IP and refers to the process when a packet transmits data. The transmission is unacknowledged data flow. IP packet segment consists of the data which the Internet layer receives on transfer from the transport layer to the receiver's end, when using the IP protocol. Protocol data unit, $PDU_{IP} = 1$ packet and has maximum 2^{16} B. PDU_{IP} is the maximum data unit which can transmit or receive at the layer when using IP packets.

Figure 4.3 shows an IP data stack which includes the IP header fields. The figure shows the start and end bit numbers in each word and data. The figure shows the data stack received or transmitted at or to the Internet layer. The IP packet consists of IP header fields 160 bits and the extended header consists of (n – 5) words when required, plus data stack of *len* words from or for the transport layer.

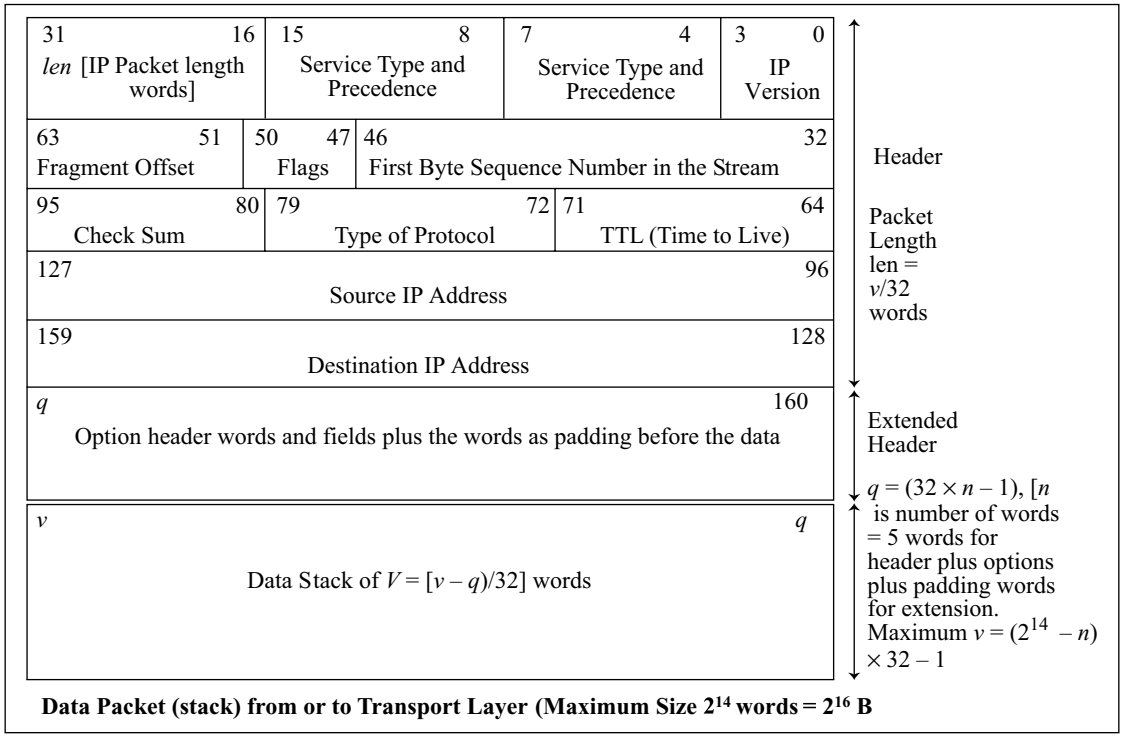


Figure 4.3 Data stack received or transmitted at or to network layer and IP packet consists of IP header field 160 bits and extended header up to bit q (extended when required) plus data stack from or for the transport layer

The features of IPv4 are:

- IP header consists of five words. The header can extend when using option and padding words. Data stack to the network layer has maximum $v = (n + len)$ words where $v <= (2^{14} – n)$.
- Header first, second and third word fields are as shown in the figure (meaning of header fields in the first three words and option words are not given here and the reader can refer to the author's Internet and Web Technology book).
- Header fourth and fifth words are source IP address and destination IP address.
- IP protocol transport is half duplex unacknowledged data flow from Internet layer at one end (End 1) to internet layer of other end (End 2).
- Each IP layer data stack is called IP packet and this packet is not guaranteed to reach the destination when the transport layer protocol is UDP, and guaranteed to reach the destination when the transport layer protocol is TCP.
- One packet communicates in one direction at an instance.

Internet Protocol Version 6

Internet Protocol Version 6 is a protocol with the following features:

- Provisions a larger addressing space
- Permits hierarchical address allocation, and thus route aggregation across the Internet, and limit the expansion of routing tables
- Provisions additional optimisation for the delivery of services using routers, subnets and interfaces
- Manages device mobility, security and configuration aspects
- Expanded and simple use of multicast addressing
- Provisions jumbo grams (big size datagram)
- Extensibility of options

While an IPv4 address is of size 32-bit, the IPv6 address has a size of 128 bits.[2] Therefore, IPv6 has a vastly enlarged address space compared to IPv4. The IPv6 address provides a numerical label. It identifies a network interface of a node or other network nodes and subnets participating in IPv6 Internet. A device is the *node* in the network when it communicates on a network.

RPL [IPv6 Routing Protocol for Low Power Lossy Networks (LLNs)]

Low Power Lossy Network refers to a constrained nodes network, which has low data transfer rate relative to IP, a low packet delivery-rate and unstable links. IETF has given specifications for RPL for the ROLL network. RPL is a non-storing routing mode.

IoT/M2M low power lossy environment uses RPL protocol. Internet layer IPv6 receives and transmits from/to data-adaptation layer (Figure 2.1) when using IEEE 802.15.4 WPAN devices (Figure 2.5).

Low power nodes need to confine communication to the nearest (or at most next) levels upwards or downwards. Lossy environment needs disjoint nodes, as in case of error or no acknowledgment, a repeat transmission can be made using the same previous instance. Data communicate with optimum data size at an instance.

Data flow between the nodes is as per Destination Oriented Directed Acyclic Graph (DODAG) model. Directed acyclic graph is a data flow model between the nodes. Destination orientation means either upwards (for transport) or downward directed (for device-layer end-node) in a tree-like structure of DODAGs. The following example clarifies the DODAG data flow approach for RPL.

Example 4.1

Problem

(a) Give an example of the DODAG data-flow approach for RPL assuming routing nodes at four ranks (levels) 0, 1, 2 and 3 consisting of 3, 6, 3 and 6 nodes, respectively. The node communicates in the RPL network environment. (b) How do the paths (called RPL instances) establish for DODAG data flow? (c) List the DODAG characteristics taking the nodes assumed in the example.

[2] http://www.webopedia.com/DidYouKnow/Internet/ipv6_ipv4_difference.html

Solution

(a) An access point can be root of a tree, which enables DODAG data flow for the RPL network. Assume that:

- A_0 is the root node at rank 0. A_1 and A2 are other nodes at rank 0.
- B_{01} and B_{02} are A_0's child nodes at rank 1 that have upward connectivity to rank 0 and downward connectivity to rank 2 nodes.
- C_{21}, C_{22} and C_{23} are three nodes at rank 2 that have upward connectivity to B's at rank 1 and downward connectivity to 3.
- D_{211} and D_{212} are C_{21}'s child nodes at rank 3. D_{221} and D_{222} are the C_{22}'s child nodes. D_{231} and D_{232} are the C_{23}'s child nodes.

Figure 4.4 shows RPL network nodes at four ranks, and RPL data-flow instances for upward and downward flow.

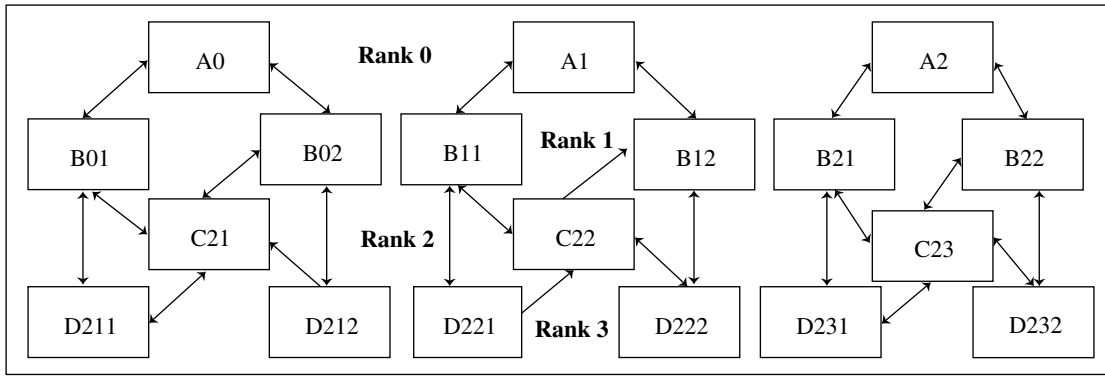


Figure 4.4 RPL network nodes at four ranks and RPL data-flow instances for upward and downward flow

Ranks 0, 1, 2 and 3 thus have 1, 2, 3 or 6 nodes communicating in the network. Consider that devices are at rank 3 and destination access point is at rank 0. DOADG in RPL specifies that all data flow is upward directed and will be from a rank 3 node to a rank 2 node, a rank 2 node to a rank 1 node, and a rank 1 node to the destination node rank 0. Data flow upwards can also be from a rank 3 node to a rank 1 node in case sufficient power is present at the node D211.

Similarly, all data flow downwards will be from a rank 0 node to a rank 1 node, a rank 1 node to a rank 2 node and a rank 2 node to the destination node, rank 3. Data flow downwards can also be from a rank i node to a rank $i + 2$ node in case sufficient power is present, where i is an integer greater or equal to 0, and i + 2 maximum value can be j, if leaf nodes (end nodes) are rank j.

(b) Paths (called RPL instances) establish using the Directed Acyclic Graph (DAG) Information Option (DIO) messages. The messages are broadcast to seek new connectivity instance, called RPL instance. Each node selects a parent. This is based on the received DIO messages. Each node calculates its rank in the tree.

Suppose rank 0 node, A0 broadcast the DIO to find a possible node with which it can connect and communicate. Assume two rank 1 nodes, B_{01} and B_{02} respond. Other nodes at rank 1 may not respond due to insufficient power. Other nodes at rank 2 may also not respond due to the need of higher power, which may not be there in the DIOs.

Assume rank 2 node, C_{011} also responds to B01, C_{012} also responds to B01, and then the downward paths A0-B01-C011, A0-B02-C012 establish, and upward path builds which forms the tree. DIOs includes a node's rank (its level) and enable the tree-like structure of RPL nodes and also establishes RPL instances at a given instance.

(c) DODAG characteristics are thus:
- DODAGS are disjoint (no shared nodes)
- Many-to-one communication: Upwards, where nodes at rank k can communicate to one node m at lower or higher in rank, where $k < m$
- One-to-many communication: Downwards, where nodes at rank m can communicate to one node m at lower or higher in rank, where $k > m$
- Point-to-point communication: Upwards or downwards where a node at rank m communicates with another node at rank k, where $m < k$ or $m > k$
- RPL instances create with objective of optimisation
- Multiple RPL instances with different optimisation objectives can coexist
- Root node or parent node need not use RPL instances to next to next ranks by keeping power level just sufficient to communicate with the nearest rank nodes. DODAG thus minimises the cost to the root per objective function.

The data flow directs downwards in an RPL instance from root at transport layer to child nodes and from child node to leaf node at physical layer device node. The data flow directs upwards in an RPL instance from a leaf/child node to another child node and then to the root. DODAGs are no share nodes (disjoint).

The features of RPL are:
- A routing protocol for the LLNs
- RPL-controlled messages are Destination Advertisement Object (DAO), DODAG Information Object (DIO) and DAG Information Object (DIO)
- Transfers data at an RPL instance data point to point (one device nodes to one receiver node) as well as point to multipoint (one to many device nodes) or multipoint to point (many device nodes to one receiver node)
- Sends or receives multiple DODAGs at each instance
- Sends data from DODAGs to DODAG root or DODAG leaves to support low data transfer rate, compared to IP low packet delivery rate
- Supports unstable links
- Has optimisation objective for an RPL instance
- Supports coexistence of different optimisation objectives using multiple RPL instances
- Supports nodes, inform parents of their presence and reach ability to descendants using DIOs
- Supports soliciting (seeking) information of a DODAG using DIS
- Supports destination, inform nodes for discovery of an RPL instance, know about configuration parameters and select DODAG parents using DIOs

4.3.2 6LoWPAN

Internet layer IPv6 receives and transmits from/to adaptation layer (Figure 4.1). The data stack uses 6LoWPAN (IPv6 Over Low Power Wireless Personal Area Network) protocol at adaptation layer before the data stack transmits to IPv6 Internet layer. An IEEE 802.15.4 WPAN device has a 6LowPAN interface serial port for connectivity.

6LoWPAN protocol is used at Adaptation layer before a data stack transmits to IPv6 Internet layer

6LoWPAN is an adaptation-layer protocol for the IEEE 802.15.4 network devices. The devices are the nodes having low speed and low power. They are the WPAN nodes of a multiple device mesh network.

Low-power devices need to limit data size per instance. Data compression reduces data size. Fragmentation of data also reduces data size per instance. Features of 6LoWPAN are header compression, fragmentation and reassembly. When data is fragmented before communication, the first fragment header has 27 bits which includes the datagram size (11 bits) and a datagram tag (16 bits). Subsequent fragments have header 8 bits which include the datagram size, datagram tag and the offset. Fragments reassembly time limit can be set equal to 60s.

Figure 4.5 (a) shows networked *i* devices physical layer in IEEE 802.15.4 WPAN. Figure 4.5 (b) shows data-link sublayer and adaptation layer 6LoWPAN protocol.

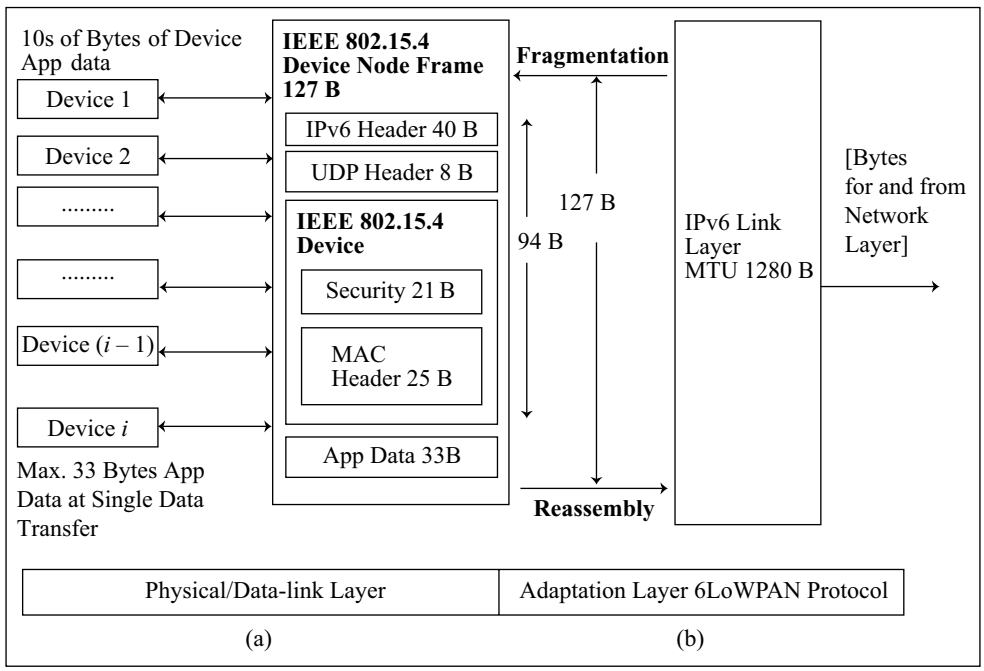


Figure 4.5 (a) Networked *i* devices at physical layer in IEEE 802.15.4 WPAN and (b) Adaptation layer 6LoWPAN protocol 127 B (maximum) fragmented frames reassembly into IPv6 maximum 1280 B or fragmentation of IPv6 MTU 1280 B into 127 B frames for transfer to a device.

Figure 4.5 shows that IPv6 over IEEE 802.15.4 standard network nodes uses the headers, security and application data as follows: IPv6 header = 40 B; UDP header = 8 B; device node MAC address = 25 B; AES-128 security = 21 B; Total device node frame size = 127 B (maximum). Therefore, maximum 33 Octets are left for the application data from the device. MAC is data communication protocol sublayer at the data-link layer (Refer Example 2.4 and Section 2.4.1 for meaning and usages of AES).

IPv6 Maximum Transmission Unit (MTU) at link layer = 1280 B. Therefore, link-layer frame fragmentation is needed in order to communicate frame of 127 B over IEEE 802.15.4 nodes (device). The frame MTU is 1280 B for transmission to network layer. Fragments from frames from the device of 127 B each reassemble into an IPv6 frame. Also, IPv6 MTU at data-link layer 1280 B fragments into frame of 127 B each for single transfer to a device node.

6LoWPAN has the following features:

- Specifies the IETF recommended methods for reassembly of fragments, and IPv6 and UDP (or ICMP) headers compression (6LoWPAN-hc adaptation layer), neighbour discovery (6LoWPAN-nd adaptation layer), and
- Supports mesh routing

ICMP stands for Internet Control Message Protocol. Routers or other devices on the network send error messages or relay the query messages.

6LoWPAN can be implemented using Berkley IP implementation with operating system TinyOS or 3BSD or other implementations for IoT nodes from Sensinode or Hitachi or others. IPv6 network layer has two options, viz. RH4 routing header and hop-to-hop header RPL option.

4.3.3 TCP/IP Suite

TCP/IP suite of protocols most used protocols for the global Internet networking

TCP/IP suite means a set of protocols with layers for the Internet. Application layers protocol examples are HTTPS, HTTP, MQTT, XMPP, SOAP, FTP, TFTP, Telnet, PoP3, SMTP, SSL/TLS and others for communication using TCP stream. DNS, TFTP, Bootpc, Bootps, SNMP, DHCP, CoAP, LWM2M and others are for datagram communication using UDP. Application layer security protocols are TLS and DTLS.

TCP is a transport layer, connection-oriented protocol that enables acknowledged data flow.

UDP is another transport layer protocol that is a connectionless protocol and is meant for datagram communication. Other protocols in the TCP/IP suite for the transport layer are RSVP and DCCP.

Internet layer protocol is IPv4/IPv6/RPL/ICMP/ICMPv6/IPSec or other. Data-link layer protocol is PPP/ARP/RARP/NDP, MAC or other. A MAC protocol is Ethernet or DSL or ISDN or other.

Figure 4.6 shows the protocol layers and representative protocols at each layer.

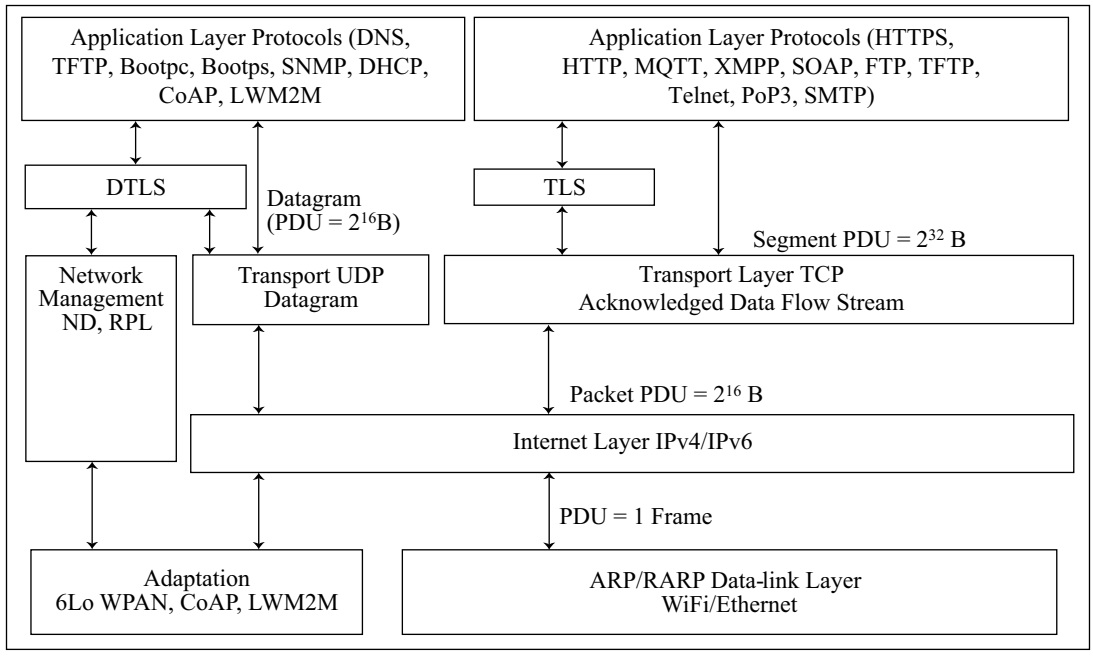


Figure 4.6 IoT TCP/IP suite of protocols for Internet

TCP/UDP Transport Layer for the Data Stack from or to Application Layer Port

Following subsections describe the TCP and UDP transport layer protocols. TCP/IP transport layer receives or transmits the data segment and datagram from or to a port at the application layer. Following subsections describe TCP and UDP protocols.

TCP

TCP protocol is used for the acknowledged data flow and almost confirmed delivery

Transport layer uses TCP when the receiver acknowledges the successful segment sequence number (within a specified interval) and remaining data segment sequences retransmit to the receiver. TCP also process data segment (maximum 2^{32} B from application-layer port but transmits the segment part only. The size of actual data transmission in a TCP stream instance depends on network traffic conditions. When congestion is high, less number of words in the packets are sent for the receiver-end Internet layer L3.

Protocol data unit, PDU_{TCP} is the maximum data unit which can transmit or receive at the layer when using the TCP stream. PDU_{TCP} = 1 segment and 1 segment maximum value = 2^{32} B. TCP stream means transmitted or received data from or to the layer in a sequence during one cycle of transmission and acknowledgement of data stack.

Figure 4.7 shows TCP data stack which includes TCP header fields. The figure shows start and end bit numbers in each word and data. The figure shows the data stack received

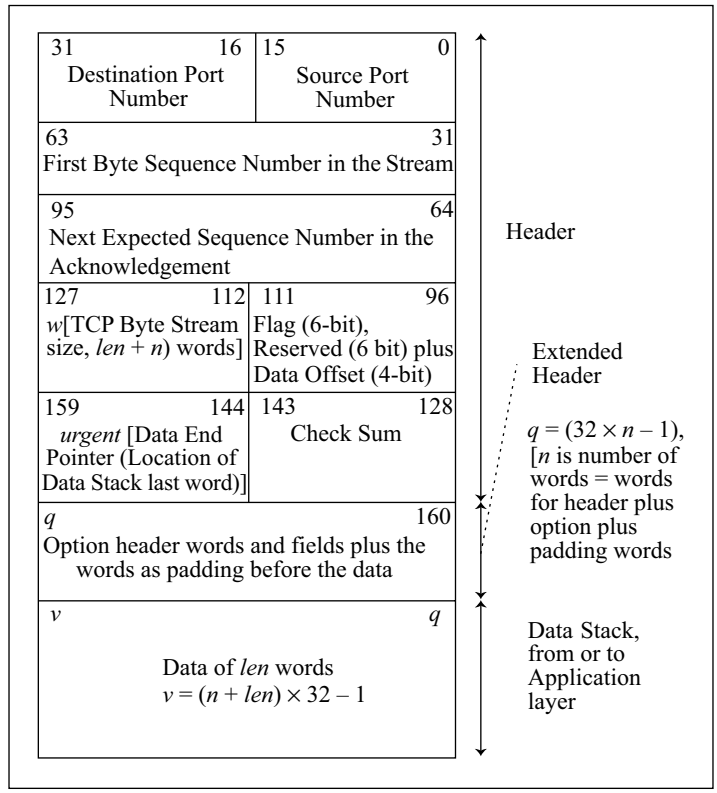


Figure 4.7 Data stack received or transmitted at or to transport layer stream consisting of TCP header field 160 bits (5 words) and extended-header words (= n – 5), extension is when required, plus data stack of *len* words from or for the application layer

or transmitted at or to transport layer stream consisting of TCP header field 160 bits and extended header words up to q^{th} bit when required, plus data stack of *len* words from or for the application layer.

The features of TCP protocol are as follows:

- TCP header consists of five words. The header can be extended by using option words and padding words. Data stack to next layer or data packet to router has maximum $v = (n + len)$ words where $v <= (2^{14} – n)$.
- Header first word fields are as follows: Upper 16 bits are for the source port number and lower 16 bits are for the destination port number. Second and third words are shown in Figure 4.7.
- Header fourth word upper 16 bits, lower 16 bits and header fourth word upper 16 bits, lower 16 bits are as shown in Figure 4.7.
- TCP protocol transport is a full-duplex acknowledged data flow from the transport layer at one end (End 1) to the transport layer of the other end (End 2).

- Each TCP layer data stack reaches the destination almost each time. This is because retransmission takes from the next of last acknowledged sequence number to another sequence number. Difference in numbers is called window size between two numbers.
- One TCP connection communicates in one direction at an instance.
- Acknowledged flow means that the request as well as response messages communicate in unicast mode. End 2 sends acknowledgement message and the header field of that conveys expected sequence number from transmitter by the receiver End 2.
- TCP is connection-oriented. The connection first establishes using a connection establishment procedure adopted when transmitting a TCP data stack for the first time. The connection closes using a connection closing procedure adopted when the last sequence completes transmission of the TCP segment stack.

Universal Datagram Protocol (UDP)

UDP protocol is used for the unacknowledged data flow when an application datagram transmits on the Internet

Application may need to transmit through the transport layer when no acknowledgment is needed from the receiver, and when data in a transporting instance limits to a datagram of size 2^{16} B including the header words enveloped at the layer. Then TCP/IP transport layer protocol is Universal Datagram Protocol (UDP). Datagram means data that it is just one message, which does not correlate with the next message or preceding message during an end-to-end transmission. UDP is also a protocol for the constrained environment and ROLL environment (Section 3.1).

UDP header fields are 2 words (1 word = 32 bit) and optionally of 4 words (when including source and destination IP addresses, and is for communication of application layer stack (For example, DNS, TFTP, Bootpc, Bootps, SNMP, DHCP) or an application-support layer protocol stack (for example, CoAP or LWM2M).

Figure 4.8 shows the UDP datagram format for transfer to the network layer during transmission or to the application layer during reception. The figure shows UDP header fields. The figure also shows a pseudo header of 2 words (64 bits) for source and destination IP addresses, which the next layer can use.

The features of UDP protocol are as follows:

- UDP header consists of two words. Data stack to network layer has maximum m words where $m <= (2^{14} – 2)$. Header first word fields are as follows: Upper 16 bits are for the source port number and lower 16 bits are for the destination port number. Header second word upper 16 bits are length and lower 16 bits are checksum.
- UDP protocol transport is a half-duplex unacknowledged data flow from the transport layer at one end (End 1) to the transport layer of the other end (End 2).
- Each UDP layer data stack may or may not reach the destination due to unacknowledged flow.
- One UDP connection communicates in one direction at an instance between two ends.

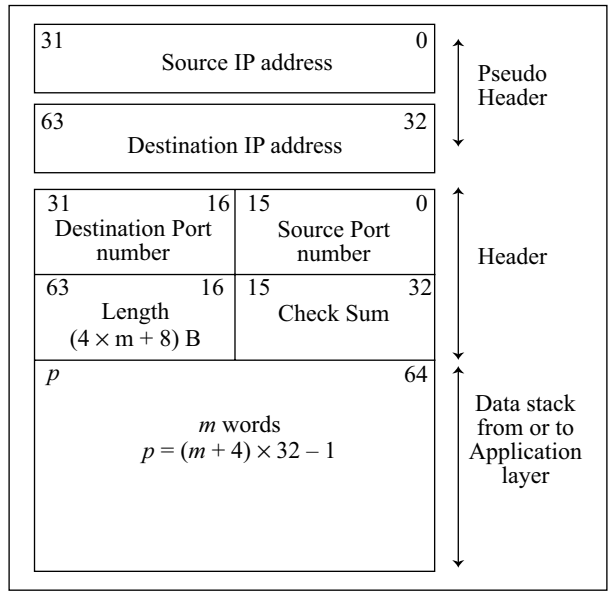


Figure 4.8 Transport layer UDP header fields with data stack from the application layer and pseudo header of 2 words (64 bits) for source and destination IP addresses

- Half duplex means that at a given instance only one direction, End 1 to End 2 or End 2 to End 1, is functional. Unacknowledged flow means that the request as well as response messages communicate in broadcast mode. End 2 does not send an acknowledgement message.
- UDP is connectionless. It thus permits multicasting (means to multiple destinations). Connection-less means no connection establishment procedure is adopted when transmitting a UDP data stack for the first time and no connection closure procedure is adopted.

Reconfirm Your Understanding

- Internet layer forwards data to the next layer using IP protocol. Internet connectivity is through a set of routers in a global network of routers. Routers carry data packets as per IP protocol from a source end to a destination and vice versa.
- Four layers for Internet communication through physical layer are application, transport, Internet and data-link layers. Internet communication layers use the TCP/IP suite of protocols.
- Application, transport and Internet layers add the header words in the data stack for transmission to the next layer as new data stack. Data-link layer adds the header words as well as the trailing bits at the end before transmitting using physical layer.
- Transport layer uses TCP or UDP. The Internet layer uses IP version 4 (IPv4) or IP version 6 (IPv6) or RPL.

- Application layer communicates a data segment or a datagram. It depends on the kind of protocol (or port) used by the layer.
- Application layer communicates a data segment (maximum 2^{32} B per segment) from L7 layer and generates a TCP stream at the transport layer using TCP. The stream packetises (maximum 2^{16} B per packet) at the Internet layer.
- Alternatively, the application layer communicates a datagram (maximum 2^{16} B) at L7 layer for sending a UDP datagram from the transport layer.
- The Internet layer uses an IP protocol for packet or message forwarding. IP protocol is IPv4 or IPv6 or RPL or other.
- RPL communication instance uses DODAG data flow.
- The Internet layer in IPv6 receives and transmits from/to data-adaptation layer when using IEEE 802.15.4 WPAN devices.
- The data stack uses 6LoWPAN protocol at the adaptation layer before data stack transmits to IPv6 Internet layer.
- UDP header has fields for source and destination port numbers.
- TCP header has fields for sequence number of the data segment from where the data stack communicates and for window size which specifies the number of transmitting bytes.
- IP packet header has fields for IP version and source and destination IP addresses.

Self-Assessment Exercise

1. Why does a header add when a data stack transfers from a higher layer to a lower layer? Why does a header subtract when data stack transfers from a lower layer to a higher layer? ★
2. How do four layers generate data stack for the network and physical layers during Internet communication? Show diagrammatically. ★★
3. Compare the features in IPv4 and IPv6. ★★
4. How do packets route over the Internet using a set of four routers between source and destination? Show diagrammatically. ★★★
5. What are the features of RPL? ★
6. List the features of 6LoWPAN. ★
7. Make a table for the application layer and corresponding protocol used at transport for the application/service/business process in TCP/IP suite of protocols. ★★
8. List the header fields in a UDP header and their functions. ★
9. List the functions of the header fields in an IP header when packets transmit between source and destination. ★★
10. List the functions of the header fields in a TCP header at the transmitting end. How are the fields used when sending an acknowledgement from the receiver end? ★★★

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

4.4 IP ADDRESSING IN THE IoT

LO 4.2 Describe the functions of IP address, MAC address, DNS and DHCP

An IP header consists of source and destination addresses, called IP addresses. The Internet generally uses IPv4 addresses. IoT/M2M use IPv6 addresses. Following subsections describe the addressing schemes.

4.4.1 IP Address

IP version 4 address consists of 32 bits. However, it can be considered as four decimal numbers separated by dots. For example, 198.136.56.2 for 32 bits—11000110 10001000 00111000 00000010. Each decimal number is decimal value of an Octet (= 8 bits). IP addresses can be between 0.0.0.0 to 255.255.255.255; total 2^{32} addresses due to 32-bit address. Three separate fields with a decimal number each for each set of 8 bits are easier to use. Let's see an analogy with postal network addressing method. Consider an address:

McGraw-Hill Education,
2 Pennsylvania Plaza,
New York City,
USA.

In the above example, three fields are separated by a comma and a new line each. A mail routes to the destination using the address. First the mail routes to USA, then to New York City, then to Pennsylvania Plaza and then the final destination. Similar are the actions, when a packets route to a destination IP address from a source IP address.

Each device or node which communicates on the Internet must have an IP address. The number of nodes or devices can be very large. Recall Figure 4.1 where a communication framework communicates with number of sensors. Each sensor or device is assigned the addresses used internally. The framework communicates with the Internet with one IP address externally to the applications. One solution for a large number of nodes is dividing the network into Internet address and subnet address.

For example, 2 Pennsylvania Plaza, New York City, USA may be a globally visible address. Internally, the mail is to be directed to which individual at McGraw-Hill Education is decided by an internal distribution network, invisible to the outside world.

Alternatively, New York City, USA may be a globally visible address. Internally, the mail to be directed to which individual at McGraw-Hill Education, 2 Pennsylvania Plaza is decided by an internal distribution network, invisible to outside world.

Internet address is visible to outside world means it is visible to the routers on the Internet. Subnet address is for use within the group internally and is invisible to the outside world. A subnet is a subnetwork consisting of number of hosts or nodes or devices or machines.

A subnet mask and network 32-bit IP address after ANDing operation gives individual subnet address. Logical AND operation with complement of subnet mask with the IP address gives the host identifier on the subnet. Example 4.3 will clarify the procedure.

IP address 198.136.56.2 is visible on the Internet. Number of servers, such as web server, mail server and FTP server are invisible and these have the same IP address globally and separate addresses internally on a subnet of server or nodes or devices.

An IP address serves the purpose of uniquely identifying an individual network interface of a host. The interface locates on the network using that address. The address enables routing of IP packets between the hosts. IP addresses are present in fields of the packet header for routing. The header indicates both source and destination of the packet.[3]

Static IP address

A static IP address is the one assigned by the Internet service provider. The service provider may provide an individual just one address. When a company has a number of hosts, a service provider may provide a class C network address consisting of a group of 254 (= $2^8 - 2$) IP addresses. The following example gives the classifications A, B, C and D and how the allocations of static addresses are made to a group of hosts.

Example 4.2

Problem

How are IP addresses allotted to a group of hosts on the Internet? What do class A, B, C and D network mean when using the subnets? Give an analogy with postal network for communication to McGraw-Hill Education, 2 Pennsylvania Plaza, New York City, USA.

Solution

A group of hosts on the Internet are allotted IP addresses as follows:

Class A network group address means address *n.x.x.x*, where x is between 0 to 255 and *n* is between 1 and 126 for the addresses between 1.0.0.0 and 126.x.x.x. This is because the IP address 32-bit has msb bit 31 = 0.

Three x.x.x specifies a network group of $2^{24} - 2$ hosts, just as USA specifies a large group of addresses in USA. Minus 2 is because all 0s and all 1s in .x.x.x are not allotted in subnet.

Class B network group address means address *n.m.x.x*, where x is between 0 to 255 and *n.m* is between 128.1 to 191.254 for the addresses between 128.1.0.0 and 191.254.x.x. This is because the IP address 32-bit has two msb bits 31–30 = 10.

Two x.x specifies a network group of ($2^{16} - 2$) hosts, just as New York City, USA specifies a group of addresses in New York City, USA. [Minus 2 is because all 0s and all 1s in .x.x are not allotted in subnet.]

Class C network group address means address *n.m.k.x*, where *x* is between 0 to 255 and *n.m.k* is between 192.0.1 and 223.255.254 for the addresses between 192.0.1.0 and 223.255.254.x. This is because the IP address 32-bit has three msb bits 31-30-29 = 110.

One *.x* specifies a smaller group of (2^8 - 2) hosts, just as 2 Pennsylvania Plaza, New York City, USA specifies a group of addresses at 2 Pennsylvania Plaza, New York City, USA and one of them is McGraw-Hill education. Minus 2 is because all 0s and all 1s in msbs 11 or 00 in 8 bits in .x not allotted in subnet.

Class D addresses are for multicasting and are from 224.0.0.0 and are less than 240.0.0.0.

Address 0.x.x.x is a reserved address. Address 127.x.x.x is also a reserved address.

[3] http://www.webopedia.com/DidYouKnow/Internet/ipv6_ipv4_difference.html

Example 4.3 gives the method of finding the subnet address and host on the subnet using the IP address and subnet mask of a subnet in case of classifications A, B, C and D for the allocations of static addresses to a group of hosts.

Example 4.3

Problem

A subnet and its hosts communicate with the Internet using a subnet mask. How does the IP address and subnet mask find the subnet address and host on the subnet? What is the procedure for evaluation of subnet address and host address?

Solution

An IP address consists of two parts: Upper bits (msbs) are for the network address and lower bits (lsbs) are for the subnet address and the individual host. Upper bits in a subnet mask allot for each node to address that node with the address of group of hosts on a subnet. Subnet mask bits depend on the class of the network group address. Mask bits are as follows:

Class A network group address node has a subnet mask and the mask consists of all eight msbs = 11111111 and remaining 24 bits are set as per the subnet and host address. Subnet id is invisible to the outside world of the Internet. These 24 bits identify the host on a subnet. Class A group subnet masks are $2^{24} - 2$ for $2^{24} - 2$ hosts.

Class B network group address node has a subnet mask and the mask consists of all sixteen msbs = 1s and remaining 16 bits are set as per the subnet id and host address at the subnet invisible to the outside world of the Internet. Last 16 bits identify subnet and the host on that subnet. Class B group subnet masks are $2^{16} - 2$ for $2^{16} - 2$ hosts.

Class C network group address node has a subnet mask and the mask consists of all 24 msbs = 1s and remaining 8 bits are set as per the subnet id and host address on the subnet. Subnet id is invisible to the outside world of the Internet. Last 8 bits identify a subnet id and the host on that subnet. Class C group subnet masks are $2^8 - 2$ for $2^8 - 2$ subnet plus host address bits.

Following is the procedure for evaluation of subnet address and host address:

Suppose 14 hosts function as subnets on a network. Assume that a Class C network group address when network communication is with 14 subnets. Each subnet thus identifies by 4 higher bits after the network address 24 bits. Therefore, subnet mask will 24 msbs as 1s plus 4 higher bits among 8 remaining bits in the subnet mask will be 1s. Subnet masks for an individual subnet will be (1111 1111 1111 1111 1111 1111 0001 0000), (1111 1111 1111 1111 1111 1111 0010 0000), …to (1111 1111 1111 1111 1111 1111 1110 0000).

ANDing of 32-bit IP group address and 32-bit subnet mask gives the internal subnet address. Higher 4 bits identify the individual subnet among the 14 subnets functioning as host. Subnet address is network address bits plus 4 subnet identifier bits and plus 4 bits 0000 in the example given in above paragraph. ANDing of complement of 32-bit subnet mask with 32-bit IP group address gives the host identifier the external IP address for the Internet.

If 30 hosts function as subnets, then subnet mask will 24 msbs as 1s plus 5 higher bits among 8 remaining bits in the subnet mask will be 1s. This is because 30 equals $2^5 - 2$. Then last 3 bits of subnet mask will be 000.

A new scheme, called Classless Inter-Domain Routing (CIDR) is used nowadays. For example, Class C 198.136.56.0/4 means Class C allotted four IP addresses for Internet routing by 4 public domain servers. The system administrator at a company allots servers for each of the 4 IP addresses, for the same domain name.

Dynamic IP address

Once a device connects to the Internet, it needs to be allotted an individual IP address. When the device connects to a router, the router and device use the DHCP (Dynamic Host Control Protocol) which assigns an IP address at an instance to the device. This address is called dynamic IP address. When a device disconnects or switches off or the router boots again, then the dynamic IP address is lost and a new allocation takes place when the device reconnects.

DNS

Consider an IP address, 198.136.56.2 (11000110 10001000 00111000 00000010). It is difficult to remember or use. The domain name for the address is rajkamal.org. Access to web-server hosted at the domain is made using the website name, http://www.rajkamal.org/. Access to mail-server hosted at the domain is made using the mail server name, http://mail.rajkamal.org/. Another example of domain name is mheducation.com/.

.com, .org, .in and .us are called Top Level Domain (TLD). A TLD can further be subdivided as .co.in or .gov.in. or .gov.uk.

A registrar provides domain names at a certain cost per year. The DNS server at the Registrar has a control panel (cPanel) and can be configured using cPanel. Domain Names System (DNS) is an application which provides an IP address for a corresponding service from the named domain service.

DHCP

When a sensor, actuator or IoT device or node needs to connect to the Internet, a server called DHCP server provides a dynamic IP address, subnet mask, and ARP and RARP caches. The server (subnet) has its own IP address to provide connectivity to the Internet.[4]

Dynamic Host Configuration Protocol (DHCP) is a protocol to dynamically provide new IP addresses and set subnet masks for the connected node so that it can use the subnet server and subnet router at the communication framework.

IP addresses are configured. An administrator or user does this process. DHCP enables the process of configuring IP addresses automatically at start-up.

A node has a software component for sending requests to the DHCP server and receiving responses. The component is called *DHCP client*. DHCP client protocol communicates with a server. Steps in the DHCP protocol for dynamically configuring the IP address and other networks are:

1. The DHCP client broadcasts a discover-request, known as DHCPDISCOVER.

[4] https://www.ietf.org/rfc/rfc2131.txt

2. A DHCP server listens to DHCPDISCOVER and finds the configuration, which can be offered to the client. Server(s) send(s) the configuration parameters, including an IP address not presently in use, at the subnet. The configuration parameters are in the DHCPOFFER for the offered configuration.
3. The selected DHCP server creates and manages bindings. DHCP server also sets a time interval during which the offered IP address will be valid for the DHCP client node.
4. The DHCP server confirms the binding through a message. It sends DHCPACK after creating the binding.
5. When the node with the DHCP client computer leaves the subnet, it sends a DHCPRELEASE message. If the client does not send DHCPRELEASE within a specified time interval, then the server frees the created binding.
6. The server and client also use authentication protocols before considering the DHCPDISCOVER from a client and before accepting a DHCPOFFER, respectively.

The DHCP protocol guarantees that any assigned network address, at a given instant, is in use by either one DHCP client or none.

4.4.2 IPv6 Address

Devices (nodes) for IoT need large number of addresses. IPv6 uses 128-bit address. A hexadecimal digit represents 4-bit, 0 hex = 0000 binary to f hex = 1111. Therefore, 128-bit address has 32 hexadecimal digits. Eight sets of 4 hex-digits are each separated by a colon or dot in an IPv6 address. Example is 16-hexadecimal digits, 40a0: 0acb:8a00:b372:0000: 0000:0000:0000. IANA manages the allocation process for IPv6 addresses; 64 bits in the last when all zeros then can be omitted.

Devices in mesh network may use 6LoWPAN protocol at the adaptation layer and Ipv6 when communication framework communicates on the Internet using IPv6. Last 64 bits are interface identifiers. An interface may have a distinct node.

IPv6 addresses are classified into three classes. Each class differs in primary addressing and routing methods. An interface may be at a distinct node. *Unicast address* is for a single network interface. 48 bit or more in unicast specify *routing prefix*. 16 bit or less specify a *subnet id*. 64 bit are interface identifiers.

Anycast address means address of a group of nodes or interfaces. A packet sent to an anycast address is delivered to just one of the member interfaces. One may be the *nearest* host. Nearest is measured according to the routing protocol's definition of distance.

Multicast address means an address used by multiple hosts, which acquires the multicast address destination by participating in the multicast distribution protocol among network routers. A packet with multicast address delivers to all interfaces that have joined the corresponding multicast group.

4.5 MEDIA ACCESS CONTROL

Each network connected node has an MAC address. A device node receives data stacks using its MAC address. *Media* means physical media, fibre or wire using which a device or node accesses the Internet. The nodes can use the same physical network and IP address. *Node* means an IoT device or sensor or actuator or controller or computer, the data-link layer of which communicates to the Internet. MAC address is 48 bit.

Each network card or Ethernet protocol using a communicating node has a unique MAC address for the source and destination node addresses. Ethernet frame communicates before the data stack source-node MAC address and destination-node MAC address. MAC address of each node is specified in the firmware of the network card or chip or core.

Usage of ARP enables each node to be addressed by the MAC address. The node receives the network data from a router having an IP address. Usage of RARP enables each node for sending on the network the data to a router having an IP address.

ARP and RARP could be used for address translation

Address Resolution Protocol (ARP) uses a lookup table. The network 32-bit address provides MAC address of the individual node using that table. RARP also uses that lookup table. The table stores the IP address in one column. MAC address is in another column in each row. Number of rows is equal to the number of nodes connected to the Internet. An individual node MAC address provides the network a 32-bit address for the Internet using the table. The lookup table is a table in which a value in a column of a row is used as a key to look for another value or a set of parameters from another column(s) of the row.

ARP cache serves as building up of the table for the address translation of the data stack received at the Internet layer to the node MAC address. Whenever the first time the network sends the receiver IP and MAC addresses, or a node sends the sender MAC and IP addresses, they save a cache called ARP cache. The cached addresses build a lookup table for the next communication between the node and the network. The process enables the node to transmit and receive the IP packets at the new IP address from the Internet by the address resolution.

Alternatively, an address can be stored locally in a memory card or flash memory of a board or microcontroller chip. Software, when the microcontroller starts, read this address. The board or chip connects to the Ethernet LAN uses the saved MAC address. The chip connects to the Internet using this address and RARP. The chip receives IP data stack using ARP.

Reconfirm Your Understanding

- IP version 4 address consists of 32 bits. However, it can be considered as four decimal numbers separated by dots. For example, 198.136.56.2 for 32 bits. Each decimal number's value is between 0 and 255.

- A subnet is a network of devices or nodes or machines or computers which connect using a common IP address. The Internet Service Provider allocates a common IP address to a group of subnets and their hosts. Each subnet in the group has a subnet mask.
- Class A network address corresponds to the biggest group of subnets and hosts [maximum (2^{24} – 2)] on a subnet (msb bit 31 = 0).
- Class B network address corresponds to a big group of subnets and hosts [maximum (2^{16} – 2)] on a subnet (Two msb bits 31-30 = 10).
- Class C network address corresponds to a small group of subnets and hosts (maximum [2^{16} – 2)] on a subnet (Three msb bits 31-30-29 = 110).
- Class D addresses corresponds to a multicasting address.
- An IPv6 address is a 128-bit address, consisting of 32 hexadecimal digits. Example is 16-hexadecimal digits, 40a0: 0acb:8a00:b372:0000: 0000:0000:0000. IANA manages the allocation process for IPv6 addresses. 64 bits in the last when all zeros, then they can be omitted.
- IPv6 addresses are classified into three classes: *unicast*, *anycast* and *multicast* addresses. 128 bits has 64 bits as interface identifier, 16 or fewer bits for subnet id and remaining msbs are the routing prefix.
- DNS is a domain name system to translate top-level domain and domain names into IP address and vice versa. The name can easily be remembered. The names are registered.
- An IP address can be static or dynamic. Dynamic IP is generated as follows: A DHCP server configures and provides an IP address for the Internet for a certain specific interval. A DHCP client seeks from a DHCP server and discovers the IP address.
- ARP finds the node (device) MAC address at data-link layer from the Internet layer IP address. The node (device) data-link layer with a MAC address connects to the Internet using an IP address. RARP finds the IP address using the MAC address.

Self-Assessment Exercise

1. List the uses of IP addresses of source and destination on the Internet. ★
2. A service provider provides a class C network address to a company. How many subnets and hosts can communicate with the Internet? ★
3. List the features of Class A, B, C and D network addresses. ★★
4. An enterprise has number of subnets and hosts. How does a subnet host connect to the Internet? ★★★
5. A subnet mask is 1111 1111 1111 1111 1001 0000 0000 0000. IP address is 198.136.56.2. How do these figures provide subnet and host addresses? ★★★
6. How do the fields in IPv6 unicast, anycast and multicast addresses differ? ★★
7. How does a domain name correlate with the URLs for mail and web servers? ★★
8. What are the steps in DHCP protocol for allocation of a dynamic IP address? ★★
9. Why is each node provided a fixed MAC address? When a new IoT device is made, then an MAC address is placed in the firmware. Why? ★
10. How is an ARP cache used? Show it diagrammatically. ★★

4.6 APPLICATION LAYER PROTOCOLS: HTTP, HTTPS, FTP, TELNET AND OTHERS

LO 4.3 Explain the functions of Application Layer protocols—HTTP, HTTPS, FTP, Telnet and Ports

TCP/IP suite consists of a number application layer protocols. For example, HTTP, HTTPS, FTP, Telnet and others. A port uses a protocol for sending and receiving messages. A TCP/IP message must be sent from the right port at the transmission end and to the right port at the receiver end, else the receiver port does not listen.

4.6.1 HTTP and HTTPS Ports

Hyper Text Transfer Protocol (HTTP) port number is 80. A web HTTP server listens to port 80 only and responds to port 80 only. An HTTP port sends application data stack at the output to the lower layer using the HTTP protocol.[5]

An HTTP port uses a URL like http://www. mheducation.com/. The default port is taken as 80. The port number can be specified after the TLD. For example, after '.com' in URL http://www. mheducation.com:80/.

HTTPS (HTTP over Secure Socket Layer or TLS) port number is 443. An HTTPS port sends a URL; for example, https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers. Here, TLD is .org, domain name is wikipedia.org and Subdomain name is en. Resource URL is at /wiki/List_of_TCP_and_UDP_port_numbers.

The port receives the data stack at the input at the receiver end. Each port at the application layer uses a distinct protocol. A port is assigned a number according to protocol used for transmission and reception.

The important features of HTTP are:

- HTTP is the standard protocol for· requesting a URL defined web-page resource, and for sending a response to the web server. An HTTP client requests an HTTP server on the Internet and the server responds by sending a response. The response may be with or without applying a process.
- HTTP is a stateless protocol. This is because for an HTTP request, the protocol assumes a fresh request. It means there is no session or sequence number field or no field that is retained in the next exchange. This makes a current exchange by an HTTP request independent of the previous exchanges. The later exchanges do not depend on the current one. E-commerce like application needs a state management mechanism. The stateless feature of HTTP is compensated by a method as: A cookie is a text file which creates during a particular pair of exchanges of HTTP request and response. The creation is either at a CGI or processing program. For example, JavaScript or script or at a client. A prior exchange may then depend on this cookie. The cookie thus provides an HTTP state management mechanism.

[5] http://www.w3.org/Protocols/

- Basically, HTTP is a file transfer-like protocol. We use it more efficiently than the FTP (a protocol for a file transfer on the Internet) because in FTP, we have to give a certain command. Then, a communication establishes between two systems just to retrieve a specified file. On the other hand, HTTP is simple. There are no command line overheads. This makes it easy to explore a website URL. A request (from a client) and reply (from a server) is the paradigm.
- The HTTP protocol is very light (a small format) and thus speedy as compared to other protocols, such as FTP. HTTP is able to transfer any type of data to a client provided it is capable of handling that data.
- HTTP is flexible. Assume during a client web connection, the connection breaks. The client can start by re-connecting. Being a stateless protocol, HTTP does not keep track of the state as FTP does. Each time a connection establishes between the web server and the client, both these interpret this connection as a new connection. Simplicity is a must because a webpage has the URL resources distributed over a number of servers.
- HTTP protocol is based on Object Oriented Programming System (OOPS). Methods are applied to objects identified by a URL. It means as in the normal case of an Object Oriented Program, various methods apply on an object.
- Following features have been included from HTTP 1.0 and 1.1 version onwards: (a) Multimedia file access is feasible due to provision for the MIME (Multipurpose Internet Mail Extension) type file definition.
- Eight HTTP specific specified methods and extension methods included from HTTP 1.1 version onwards. The HTTP specific methods are as follows. 1. GET. 2. POST. 3. HEAD. 4. CONNECT. 5. PUT. 6. DELETE. 7. TRACE. 8. OPTIONS. (Last four from 1.1.)
- Earlier versions, GET follows a space and then the document name. The server returns the documents and closes the connection. The POST method permits form processing as using it the client transmits the form data or other information to the server from 1.1. Also, the server does not close the connection after response and thus response can be processed before it is sent.
- A provision of user authentication exits besides the basic authentication.
- Digest Access Authentication prevents the transmission of username and password as HTML or text from HTTP 1.1 version onwards.
- A host header field adds to support those ports and virtual hosts that do not accept or send IP packets. An Error report to Client when an HTTP request is without a Host header field from HTTP 1.1 version onwards.
- An absolute URL is acceptable to the server. In the earlier version, only a proxy server accepted that.
- *Message Headers* uses are: A message during a request from a client or during a response from a server consists of two parts—a start-line, none or several message-headers (fields) and empty line, and body of the message. Message headers are:

- *Common (general) headers* are added when requesting a server and when responding to a client. A header includes MIME version, OPTIONS, cacheable or not cacheable, and transfer to close or not close the connection.
- *Request headers* are for a request and client information to a server. The header includes acceptable media or preferred specification—specifies about acceptability of HTML or HTML or text or any other type informs if a special type of character set acceptable.
- *Entity headers* contain information about the entity body contained in the message or in case the body is not present then information about the entity is not present its body. For example, information for content-length in bytes.
- *Response headers* are present in the response for server information to a client.

- Status codes add in the response and caching of a resource provided at a server (and proxy). For example, status code returned as response when 400 means a bad request (the request unrespondable), 401 unauthorized request, 402 means request requires a payment before response feasible, 403 means request is for a forbidden resource and 404 means URL resource not found by the server.
- Byte range specification helps an HTTP server to send large response in parts. Length specification helps in presentation in chunks.
- Selection among various characteristics on retrieval by the client is feasible when a server sends a *response* to the client *request.* For example, the two characteristics, *language* and *encoding* can be specified in the server environment variables while the client sends the request header for retrieving a resource. The resource then retrieves in that *language* and with that *encoding.* The contents sent to the client do not change, only the way in which these are presented to the client change. The environment variables for the way in which a resource retrieves could not be changed at the server in the earlier HTTP versions before 1.1.

4.6.2 Other Ports

TCP/UDP registered ports and unregistered ports

The IANA reserves the port numbers. The numbers are between 0 and 1023. They are well-known ports. System process or software connects to them. Port number 0 means the host itself. Internet Assigned Number Authority (IANA) registers a registered port number which is according to a specific protocol. Registered numbers are between 1024 and 49151. These are registered ports. 2^{12} numbers between 49152 are unregistered, left for assignment and use by users. System process or software connects to them. A user unregistered server has a port number above 5000.

A list of widely used ports is available at https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers.

Example 4.4

Problem

Give examples of registered port numbers and the protocols which each port uses.

Solution

Examples are as follows:

Port Number	17	21	23	25	53	79	80	108	110
Protocol	Users List	FTP	Telnet	SMTP	DNS	Finger	HTTP	SNA-GAS	PoP3

Port Number	161	389	443	547	1293	2095	5222	5223	5683
Protocol	SNMP	LDAP	HTTPS	DHCPServer	IPsec	cPanel (default use is for web mail)	XMPP	XMPP-SSL	CoAP

FTP is a file transfer protocol. It is a stateful protocol. Telnet is for remote connection to a computer. SMTP is for mail transfer and PoP3 for mail retrieval from a mail server.

A port supports multiple logical connections between a source at an IP address and destination at another IP address. A socket consisting of port and IP sends application data stack on the Internet to another socket at another IP address and port.

Reconfirm Your Understanding

- HTTP, HTTPS, FTP, Telnet, SMTP, DNS, DHCP, PoP3, SNMP, LDAP, XMPP, XMPP-SSL, CoAP, LWM2M and IPSec are protocols for the application layer/application-support layer ports.
- A URL specifies the protocol when sending a request for a resource. For example, http://, coap://, https:// and ws:// before a domain name.
- HTTP is stateless communication. A method is used to save states during communication in web applications.
- An HTTP client is used for a web server for requesting a webpage as response from an HTTP server.
- An HTTPS port is for an HTTP message over a secure socket layer or TLS.
- A port sends a TCP/IP message to a receiver port. IANA assigns a port number for each protocol. A receiver port listens only if the sender port is using the right port and sends to the right port.
- A port number associates a protocol at the application/application-support layer. For example, port number 80 associates HTTP.

Self-Assessment Exercise

1. What are the functions of HTTP, HTTPS, FTP, Telnet, CoAP and LWM2M ports? ★
2. What is the advantage of a receiver port not listening to a message unless sent using the right port at transmission end and received at the right port at the receiver end? ★★
3. What do the terms *language* and *encoding* specify in the server environment variables while client sends the request header for retrieving a resource? ★
4. List the features of HTTP. ★★★
5. List the usages of eight HTTP specific specified methods. ★
6. What are the message header types and the uses of each in HTTP 1.1? ★★
7. What are the advantages of a server response consisting of a status code? ★
8. How does function of a port differ from that of a socket? ★★
9. List the 16 port numbers along with their protocols. ★★★
10. Why is it that IANA registers a port number and assigns it a specific protocol? ★★★

Key Concepts

- 6LoWPAN
- ARP
- Class A, B and C
- Data segment
- DHCP
- DNS
- Header
- Header fields
- Host
- HTTP
- HTTPS
- IP
- IP router
- IPv6
- MAC address
- MTU
- Packet
- PDU
- Port
- RARP
- RPL
- Subnet
- TCP
- TCP header
- TCP/IP
- UDP
- UDP header

Learning Outcomes

LO 4.1

- Internet communication is through the routing of packets. A packet has a maximum 2^{16} B. Packet routing for Internet communication is as follows: Each router has information about the path to the destination for the packets. When a number of paths are available at an instance, then number of packets of same source simultaneously follows different paths from a router. Destination-end transport layer reassembles the packet according to their sequences in source transport layer data streams. Then, the reassembled data segment transfers to the IoT application layer at the destination end.

- Four layers for Internet communication are application, transport, Internet and data-link layers. These layers use the TCP/IP suite of protocols for Internet communication. TCP/IP suite is a set of protocols with layers for the Internet.
- Application layer protocol is HTTPS, HTTP, MQTT, XMPP, SOAP, FTP, TFTP, Telnet, PoP3, SMTP, SSL/TLS or other when a TCP stream communicates. The protocol is DNS, TFTP, Bootpc, Bootps, SNMP, DHCP, CoAP, LWM2M or other when a datagram communicates using UDP.
- Application layer security protocol is TLS or DTLS.
- Transport layer protocol is TCP or UDP or RSVP, DCCP or other. TCP feature is acknowledged data flow and ensuring delivery of all sequences in the segment. UDP is unacknowledged data flow.
- Internet layer protocol is IP protocol (IP v4/IPv6/ RPL)/ICMP/ICMPv6/IPSec or other. Internet layer uses an IP protocol for packet forwarding.
- Data-link layer protocol example is Ethernet, MAC, PPP, ARP, RARP or NDP.
- Adaptation layer protocol is 6LoWPAN for IEEE 802.15.4 IoT/M2M devices.
- RPL is a non-storing routing mode protocol. IoT/M2M low power lossy environment uses RPL protocol. Internet layer IPv6 receives and transmits from/to data-adaptation layer when using IEEE 802.15.4 WPAN devices

LO 4.2

- Internet communication between routers uses IP addresses. IPv4 address is 32-bit address and IPv6 of 128-bits.
- A subnet is a network of devices, nodes, machines or computers, which connect using a common IP address. Internet Service Provider allocates a common IP address to group of subnets and their hosts. Each subnet in the group has a subnet mask.
- Three types of classes—A, B and C network address corresponds to a group of subnets and hosts.
- IPv6 address is 128-bit address, which has 32 hexadecimal digits. IPv6 addresses are classified into three classes: *unicast*, *anycast* and *multicast* addresses.
- DNS is a domain name system to translate IP address into names and names into IP addresses.
- IP address can be static or dynamic. Dynamic IP generates using DHCP client and server.
- Internet layer IP address using ARP finds the node (device) MAC address at data-link layer. The node (device) data link layer with a MAC address connects Internet using an IP address RARP finds the IP address

LO 4.3

- A port sends a TCP/IP message to a receiver port. An HTTP or HTTPS port is for sending a request and receiving the response.
- IANA assigns a port number for each protocol for the Internet. A receiver port listens to only if sender port is using right port and sends to right port.
- HTTP, HTTPS, FTP, Telnet, SMTP, DNS, DHCP, PoP3, SNMP, LDAP, XMPP, XMPP-SSL, CoAP, LWM2M and IPSec are the protocols for the application layer/application support layer ports.
- A URL specifies the protocol when sending request for a resource, for example, http://, coap://, https:// and ws:// before a domain name.
- A port number associates a protocol at the application/application-support layer. For example, port number 80 associates HTTP.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. TCP/IP protocol suite for IoT application/service/business process consists of: ★
   (a) Application, application-support, security, transport, network, adaptation and data-link layer protocols
   (b) Security, transport, network, data-link layer protocols
   (c) Transport and Internet layer protocols
   (d) Security, transport and Internet layer protocols
2. TCP protocol (i) almost guarantees successful reception of all packets, (ii) specifies that if within a specified interval, there is no acknowledgement of last sent sequence number for the messages or the acknowledged sequence number is less than the last sequence number, then the unacknowledged sequences retransmits. (iii) The acknowledged sequence number of data stack may be received after the interval, (iv) receiver end not receiving packets, (v) header field provisions for the multiple acknowledgements, and (vi) window size is large. Duplicate and triplicate acknowledgements in TCP are because of: ★★★
   (a) (ii)
   (b) (ii) and (iii)
   (c) (i)
   (d) (i) and (vi)
3. (i) Transport layer protocol is UDP when the application layer protocol is DNS, TFTP, Bootpc, Bootps, SNMP, DHCP, CoAP or LWM2M, (ii) Internet layer protocol is IPv4/IPv6/RPL/ICMP/ICMPv6/IPSec. (iii) Transport and IP layers use a header field for error detection. (iv) Data-link layer uses CRC bits for error detection. ★★
   (a) All except (i)
   (b) All except (iv)
   (c) All except (iii)
   (d) All
4. UDP (i) is Internet layer protocol, (ii) is Transport layer protocol for a CoAP client for communication IoT sensor data, (iii) sends a datagram of size up to 2^{16} B, (iv) packetizes the data stack of Application layer, (v) acknowledged data flow. ★★
   (a) All except (i)
   (b) All except (ii) and (v)
   (c) All except (iii)
   (d) (ii) and (iii)
5. (i) The data flow directs downwards in an RPL instance from root at transport layer to child nodes and from child node to leaf node at physical layer device node. (ii) The data flow directs upwards in an RPL instance from a leaf or child node to other child node and then to the root. (iii) RPL is IoT/M2M low power loss environment uses RPL protocol. (iv) Internet layer IPv4 receives and transmits from/to adaptation layer when using IEEE 802.2 LAN devices. (v) ROLL is used when network has ★

unconstrained nodes. (vi) When network required high data transfer rate, relative to IP low packet delivery rate, unstable links. (vii) RPL functions in storing mode.
(a) All except (i) and (vii)
(b) All except (ii) and (v)
(c) (i), (ii) and (iii)
(d) (i), (ii) and (iv)

6. IPv6 features are (i) provisioning a larger addressing space, (ii) permitting the hierarchical address allocation, (iii) route aggregation across the Internet, (iv) limit the expansion of routing tables, (v) provisions additional optimisation for the delivery of services using routers, subnets and interfaces, (vi) manages device mobility, security, and configuration aspects and (vii) provisions expanded and simple use of multicast addressing (viii) provisions 2^{16} B data grams. ★
   (a) All except (ii) and (vii)
   (b) All except (viii)
   (c) (i) to (v)
   (d) (i), (ii) and (iv)
7. HTTP 1.1 supports HTTP specific specified methods (i) GET (ii) POST (iii) HEAD (iv) CONNECT (v) PUT (vi) DELETE (vii) TRACE (viii) OPTIONS, and (ix) HTTP 1.1 supports the extension methods included from HTTP 1.1 version onwards. ★
   (a) All except (iii) and (vii)
   (b) All except (ix)
   (c) (i) to (v)
   (d) All
8. 6LoWPAN device node frame size is: ★
   (a) Same is Ethernet frame
   (b) 256 B
   (c) 127 B
   (d) 2^{16} B
9. Port (i) is an interface to the network using a protocol, (ii) sends a data stack at the output to the lower layer, (iii) Port sends a data stack at the output to the Internet layer, (iv) receives data stack at the input from the transport layer, (v) receives data stack at the input from the TLS layer, and (vi) is assigned number according to protocol used for transmission and reception. ★
   (a) All
   (b) All except (iii) and (v)
   (c) All except (ii)
   (d) All except (i)

Short-Answer Questions

1. Why do the header-fields add at a layer before a data stack transfer to a lower layer? ★ **[LO 4.1]**
2. What are the header fields in 6LoWPAN? Why is the application data from a device node is 33B or 54 B? ★★ **[LO 4.1]**
3. Explain, why IoT device nodes use RPL in place of IPv6 or IPv4, why a CoAP client in place of HTTP client and 6LoWPAN at the adaptation layer in place of the MAC. ★ **[LO 4.1]**

4. How are header fields formatted and used in unicast IPv6 address? **[LO 4.1]** ★★
5. Why and how do DODAGs used in RPL? **[LO 4.1]** ★★★
6. A network address is 198.136.x.x. Recognize the class of the network. What can be the maximum and minimum values for first decimal number of the network address? What can be the maximum and minimum values for second decimal number of the network address? How will the network address 198.136.x.x.used to find the subnet address and host? **[LO 4.2]** ★★★
7. How do the subnets enable usage of a limited number of IP addresses in an enterprise network? **[LO 4.2]** ★★
8. What do the terms network interface, host, subnet, port, socket and subnet mean? **[LO 4.2]** ★
9. How does the subnet expand a network? **[LO 4.2]** ★★
10. Explain the commonalities and contrast between HTTP and CoAP. **[LO 4.3]** ★★★
11. What are uses of four types of headers, environmental variables and status codes in HTTP? **[LO 4.3]** ★★

Review Questions

1. How does data communicate from a gateway using an IP address after formatting as packets? [**LO 4.1**] ★
2. IoT applications and services layer is using TCP/IP suite application protocols. How does the source-end network layer connect through a set of IP routers with the receiving-end application layer? [**LO 4.1**] ★★
3. When and how does the UDP datagram communicate from source-end transport layer to destination-end application-support layer? [**LO 4.1**] ★★
4. When and how does TCP stream communicate from source to end using IP packets? [**LO 4.1**] ★
5. What are the features of RPL? How do they enable routing in a lossy environment? **[LO 4.1]** ★★★
6. Show and explain the header fields in a 6LoWPAN frame, IP packet, TCP stream. **[LO 4.1]** ★★
7. Describe class A, B and C networks. **[LO 4.2]** ★★★
8. What is MAC address? How does a MAC address assign to an IoT node? How does address resolved to enable packets in IP network reach the node? [**LO 4.2**] ★★★
9. How are the port and port number used? **[LO 4.3]** ★
10. Describe the features of HTTP. [**LO 4.3**] ★★

Practice Exercises

1. Redraw Figure 4.1 for source-end IoT/M2M IoT applications and services layer connected through a set of IP routers to send data packets using an IP address and communicating to the devices network layer IP address using TCP/IP suite of application protocols. **[LO 4.1]** ★
2. Create a table giving usages of IP, IPv6, RPL, 6LoWPAN, TCP and UDP protocols. **[LO 4.1]** ★
3. Show by diagram, the fields in 127 B data stack which communicate between the IoT device and 6LoWPAN adaptation layer. **[LO 4.1]** ★★

4. List the RPL instances in Figure 4.4 for communication from each device node data transfer to an access point. **[LO 4.1]** ★★★
5. Create a table for a number of bits in data stack, usages and fields in an IPv4 address, IPv6 address, routing prefix, subnet address, subnet mask, subnet id, host identifier, and MAC addresses. **[LO 4.2]** ★
6. Make a table giving in each row the subnet mask, class of the network address, subnet ID, subnet addresses and lsbs of host. Assume four rows have subnet masks for individual subnets = (1111 1111 1111 1111 0001 0000 0000 0000), (1111 1111 1111 1111 0010 0000 0000 0000), (1111 1111 1111 1111 1110 0000 0000 0000). (1111 1111 1110 1100 0000 0000 0000 0000). The table gives the identifiers addresses. **[LO 4.2]** ★★
7. Compare the port, socket and WebSocket by examples. **[LO 4.3]** ★★★
8. Create a table for understanding which port number to use when, for each port. Use the port numbers listed at https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers. **[LO 4.3]** ★★★

5

Data Acquiring, Organising, Processing and Analytics

Learning Objectives

LO 5.1 Apply the data-acquiring and data-storage functions for IoT/M2M devices data and messages

LO 5.2 Classify ways of organising data

LO 5.3 Summarise the transactions on stored data, functions for business processes and business intelligence, and the concepts of IoT applications—integration and services architecture

LO 5.4 Identify the functions and usage of data analytics and data visualisations for IoT applications and business processes

LO 5.5 Explain knowledge discovery, knowledge management and knowledge-management reference architecture

Recall from Previous Chapters

Lessons for IoT/M2M architectural layers and functions learnt in previous chapters, are—devices communicate, first over a local network or WPAN and then send the physical layer data to data-adaptation and gateway layer. The gateway connects to the Internet, and communicates the data packets. The packets communicate over the Internet through a set of routers. Application and application-support layers use the acquired and collected data for IoT applications. Applications can also control and monitor the functions of devices. The application messages, commands and data communicates to devices through the gateway using the Internet.

5.1 INTRODUCTION

Having learnt about devices, devices-network data, messages and packet communication to the Internet, let us understand the functions required for applications, services and business processes at application-support and application layers. These functions are data acquiring, data storage, data transactions, analytics, results visualisations, IoT applications integration, services, processes, intelligence, knowledge discovery and knowledge management.

Let us first discuss the following terms and their meanings used in IoT application layers.

Application refers to application software or a collection of software components. An application enables a user to perform a group of coordinated activities, functions and tasks. Streetlights control and monitoring is an example of an application. Software for tracking and inventory control are other examples of applications. Tracking applications use tags and locations data of the RFIDs.

An application enables a user to withdraw cash using an Automatic Teller Machine (ATM). An umbrella sending warning messages for weather (Example 1.1), a waste container management, health monitoring, traffic lights control, synchronisation and monitoring are other examples of IoT applications.

Service denotes a mechanism, which enables the provisioning of access to one or more capabilities. An interface for the service provides the access to capabilities. The access to each capability is consistent with constraints and policies, which a service-description specifies. Examples of service capabilities are automotive maintenance service capabilities or service capabilities for the Automatic Chocolate Vending Machines (ACVMs) for timely filling of chocolates into the machines.

Service consists of a set of related software components and their functionalities. The set is reused for one or more purposes. Usage of the set is consistent with the controls, constraints and policies which are specified in the service description for each service. A service also associates a *Service Level Agreement* (SLA).

A service consists of a collection of self-contained, distinct and reusable components. It provides logically grouped and encapsulated functionalities. Traffic lights synchronising service, automobile maintenance service, devices location, detection and tracking service, home security-breach detection and management service, waste containers substitution service, and health-alerts service are the examples of IoT services.

Service Oriented Architecture (SOA) is a software architecture model, which consists of services, messages, operations and processes. SOA components are distributed over a network or the Internet in a high-level business entity. New business applications and applications integration architecture in an enterprise can be developed using an SOA.

Message means a communicating entity or object.

Operation means action or set of actions. For example, actions during a bank transaction.

Transaction (trans + action) refers to two inter-related sets of operations or actions or instructions. For example, a transaction may be access to sales data to select and get the annual sales in a specific year in return. One operation is access to sales data and other is annual sales in return. Another example of a transaction is a query transaction with a Database Management System (DBMS).

Query is a command for getting select values from a database which in return transfer the answer to the query after its processing. A query example is command to ACVMs database for providing sales data of ACVMs on Sundays near city gardens in a specific festival period in a year. Another example is query to service center database for providing the list of automobile components needing replacement that have completed expected service-life in a specific vehicle.

Query Processing is a group of structured activities undertaken to get the results from a data store as per the query.

Key Value Pair (KVP) refers to a set of two linked entities, one is the key, which is a unique identifier for a linked entity and the other is the value, which is either the entity that is identified or a pointer to the location of that entity. A KVP example is birthday-date pair. KVP is birthday: July 17, 2000. Birthday is the key for a table and date July 17, 2000 is the value. KVP applications create the look-up tables, hash tables and the network or device configuration files.

Hash Table (also called hash map) refers to a data structure which maps the KVPs and is used to implement an associative array (for example array of KVPs). A hash table may use an index (key) which is computed using a hash function and key maps to the value. Index is used to get or point to the desired value.

Bigtable maps two arbitrary string values into an associated arbitrary byte array. One is used as row key and the other as column key. Time stamp associates in three-dimensional mapping. Mapping is unlike a relational database but can be considered as a sparse, distributed multi-dimensional sorted map. The table can scale up to 100s to 1000s of distributed computing nodes with ease of adding more nodes.

Business Transaction (BT) in database theory, refers to a (business) process that requests information from or that changes the *data* in a *database*. One operation in a BT is a command 'connect' that connects a DBMS and database, which in turn also connects with the DBMS. Similarly, BTs are processes using commands 'insert', 'delete', 'append', and 'modify'.

Process means a composition of a group of structured activities or tasks that lead to a particular goal (or that interact to achieve a result). For example, streetlights control

process of the purchase process for an airline ticket. A process specifies activities with relevance rules based on data in the process.

Process Matrix is a multi-element entity, each element of which relates a set of data or inputs to an activity (or subset of activities).

Business Process (BP) is an activity or series of activities or a collection of inter-related structured activities, tasks or processes. A BP serves a particular goal or specific result or service or product. The *BP* is a representation or process matrix or flowchart of a sequence of activities with interleaving decision points; interleaving means putting in between. Decision point means an instance in a series of activities when decisions are taken for further activities.

A web[1] definition states, "a BP is a specific event in a chain of structured business activities that typically change the state of data and/or a product and generate some type of output". Examples of BPs include finding the annual sales growth and managing the supplies. Another definition[2] of BP is that "business process is an activity or set of activities that will accomplish a specific organizational goal". One more definition[3] of BP states, "BP is a series of logically related activities or tasks (such as planning, production, or sales) performed together to produce a defined set of results."

Business Intelligence (BI) is a process which enables a business service to extract new facts and knowledge, and then undertake better decisions. These new facts and knowledge follow from earlier results of data processing, aggregation and analysis of these results.

Example 5.1 clarifies the meanings of application, service, SOA, BP and BI for an Internet connected chain of ACVMs to enable the understanding of what these terms do mean.

Example 5.1

Problem

Assume a connected chain of ACVMs spread all over a city. Each ACVM delivers chocolate at each instance as per the user's choice among one of the 5 flavours, viz. FL1, FL2, FL3, FL4 and FL5. A chosen chocolate delivers on insertion of coins of appropriate amount as per the cost. A display unit displays user interfaces and takes part in user interaction when a child wishes to buy and get the chocolate of her/his choice. The ACVM also displays advertisements for chocolates, news, weather reports and events in the city whenever not in use. Each ACVM connects to the Internet for its management and services. What do application, service, SOA, business process and business intelligence mean in the Internet of ACVMs?

Solution

Application refers to software that provides for the management of ACVMs. When a new ACVM installs, then the Manager updates the database of machine IDs. The manager programs the ACVMs for the services and also does the diagnosis of each ACVM at regular intervals. This helps in locating the improper functioning

[1] http://www.webopedia.com/TERM/B/business_process.html
[2] http://searchcio.techtarget.com/definition/business-process
[3] http://www.businessdictionary.com/definition/business-process.html

ACVMs. When the Manager finds a specific ACVM faulty, he/she initiates the required steps. For example, sending SMS to owner; informing the ACVM maintenance service which sends a mechanic; informing the chocolate-filling service, and so on.

Service refers to software, which initiates filling of the ACVMs with chocolates of distinct flavours. Each service has a service description for usage by a Manager application. Service includes a Service Level Agreement (SLA) with the Manager. Service initiates actions on message from the Manager. For example, Manager periodically delivers the messages for events and alerts for the sell up to the specified threshold levels at each ACVM for each flavour. This enables service to optimally plan the filling of the ACVMs.

ACVMs manager can deploy number of services, like collection of coins service. Each service has the service description (desc) and SLA. Service software distinguishes itself from an application due to its usages, not only specific to one application but can also be to many applications. A service selection uses desc and enters into an SLA between the service and application.

SOA is architecture models all the ACVM services, their interactions and the initiation of each service on a distinct message from an application, service or process to another. SOA describes the services, other components, their interactions, the sequences of messages, operations and processes.

Process refers to series of activities such as acquiring data from the ACVMs for the counts of each flavour sold during programmed intervals, analysing the acquired data, initiating messages for Fill service for each flavour through Fill process. Another series of activities such as acquiring data for the amount collected at each of the ACVMs at programmed intervals, analysing the acquired data; and initiating Collect service for each flavour through Collect process.

Business process refers to series of related processes, such as Fill, Collect and Display at appropriate intervals is a business process in ACVMs chain.

Business intelligence is a process which enables the ACVM business service to extract new facts and knowledge. This enables the undertaking of better decisions for new option(s) to maximise profits. For example, dropping or reduced loading of the ACVMs with specific flavour(s) in select regions or introducing new alternative flavours or advertising for the flavours getting less favour from the children, or relocating ACVMs in specific areas or adopting enterprise strategy for intimacy with children such as free chocolate on birthdays or awarding loyalty points.

New facts and knowledge follows from the earlier acquired results of data processing and aggregation, and the region-wise, segmentation-wise, flavour-wise, week-wise and festival-wise sales analytics.

Refer Oracle's IoT architecture framework (Figure 1.5). IoT devices connect through the Internet. The data acquires, organises, processes and is analysed for the applications, services, enterprise applications, BPs and BI. The connectivity to the Internet of IoT/M2M devices, such as sensors, streetlights, ATMs, RFIDs and automobiles, is for various applications and services. IoT/M2M applications, services and business processes use the messages and data packets received from the devices over the Internet.

Following sections describe the data acquiring, organising, analytics, visualisations for IoT applications, services, business processes and knowledge discovery and management.

5.2 DATA ACQUIRING AND STORAGE

LO 5.1 Apply the data-acquiring and data-storage functions for IoT/M2M devices data and messages

Following subsections describe *devices data*, and steps in acquiring and storing data for an application, service or business process.

5.2.1 Data Generation

Data generates at devices that later on, transfers to the Internet through a gateway. Data generates as follows:

- ***Passive devices data:*** Data generate at the device or system, following the result of interactions. A passive device does not have its own power source. An external source helps such a device to generate and send data. Examples are an RFID (Example 2.2) or an ATM debit card (Example 2.3). The device may or may not have an associated microcontroller, memory and transceiver. A contactless card is an example of the former and a label or barcode is the example of the latter.
- ***Active devices data:*** Data generates at the device or system or following the result of interactions. An active device has its own power source. Examples are active RFID, streetlight sensor (Example 1.2) or wireless sensor node. An active device also has an associated microcontroller, memory and transceiver.
- ***Event data:*** A device can generate data on an event only once. For example, on detection of the traffic or on dark ambient conditions, which signals the event. The event on darkness communicates a need for lighting up a group of streetlights (Example 1.2). A system consisting of security cameras can generate data on an event of security breach or on detection of an intrusion. A waste container with associate circuit can generate data in the event of getting it filled up 90% or above. The components and devices in an automobile generate data of their performance and functioning. For example, on wearing out of a brake lining, a play in steering wheel and reduced air-conditioning is felt. The data communicates to the Internet. The communication takes place as and when the automobile reaches near a Wi-Fi access point.
- ***Device real-time data:*** An ATM generates data and communicates it to the server instantaneously through the Internet. This initiates and enables Online Transactions Processing (OLTP) in real time.
- ***Event-driven device data:*** A device data can generate on an event only once. Examples are: (i) a device receives command from Controller or Monitor, and then performs action(s) using an actuator. When the action completes, then the device sends an acknowledgement; (ii) When an application seeks the status of a device, then the device communicates the status.

5.2.2 Data Acquisition

Data acquisition means acquiring data from IoT or M2M devices. The data communicates after the interactions with a data acquisition system (application). The application interacts

and communicates with a number of devices for acquiring the needed data. The devices send data on demand or at programmed intervals. Data of devices communicate using the network, transport and security layers (Figure 2.1).

Applications configure the devices for acquiring data

An application can configure the devices for the data when devices have configuration capability. For example, the system can configure devices to send data at defined periodic intervals. Each device configuration controls the frequency of data generation. For example, system can configure an umbrella device to acquire weather data from the Internet weather service, once each working day in a week (Example 1.1). An ACVM can be configured to communicate the sales data of machine and other information, every hour. The ACVM system can be configured to communicate instantaneously in event of fault or in case requirement of a specific chocolate flavour needs the Fill service (Example 5.1).

Application can configure sending of data after filtering or enriching at the gateway at the data-adaptation layer. The gateway in-between application and the devices can provision for one or more of the following functions—transcoding, data management and device management. Data management may be provisioning of the privacy and security, and data integration, compaction and fusion (Section 2.3).

Device-management software provisions for device ID or address, activation, configuring (managing device parameters and settings), registering, deregistering, attaching, and detaching (Section 2.3.2).

Example 5.2 gives the process of acquiring data from the embedded component devices in the automobiles for Automotive Components and Predictive Automotive Maintenance System (ACPAMS) application.

Example 5.2

Problem

Internet of ACPAMS application—How does an ACPAMS application acquire data from the embedded devices in the automobile components in a car?

Solution

A number of components, such as engine control system, axle, steering system, brake linings, wipers, air conditioners, battery and shockers, need predictive maintenance. Each component embeds computing hardware, software and interface to a network in an automobile. Each embedded device in the automobile communicates to a central computing system using the Controller Area Network (CAN) bus. Consolidated data then communicates thorough the Internet to the ACPAMS center (Example 5.2).

An application at the system, first manages each embedded device. This means it allots the device ID (address), activates, configures (manages device parameters and settings), registers, deregisters, attaches and detaches. System gateway software communicates to a service-centre application.

The gateway application in event of an automobile in the vicinity of the hotspot, communicates the acquired data aggregated over preset required intervals for each embedded component device. The communication is through Wi-Fi (Section 2.2). As and when the automobile happens to be near a Wi-Fi hotspot, the device data communicates to the application.

The acquired data stores in the databases at a data store (Section 5.2.6). The application uses analytics and advanced analytics at scheduled intervals. The analytics predicts the maintenance needs for each automotive component after predefined intervals (Section 5.5). The application messages the automobile dashboard the needed preventive maintenances at regular intervals.

Similar to the above example, the industrial plants use applications for acquiring data from machines. Analytics of data enable necessary predictive or prescriptive maintenances (Section 5.5.1).

5.2.3 Data Validation

Data must be validated before storing

Data acquired from the devices does not mean that data are correct, meaningful or consistent. Data consistency means within expected range data or as per pattern or data not corrupted during transmission. Therefore, data needs validation checks. Data validation software do the validation checks on the acquired data. Validation software applies logic, rules and semantic annotations. The applications or services depend on valid data. Then only the analytics, predictions, prescriptions, diagnosis and decisions can be acceptable.

Data aggregation, adaptation and enrichment is done before communicating to the Internet

Large magnitude of data is acquired from a large number of devices, especially, from machines in industrial plants or embedded components data from large number of automobiles or health devices in ICUs or wireless sensor networks, and so on. Validation software, therefore, consumes significant resources. An appropriate strategy needs to be adopted. For example, the adopted strategy may be filtering out the invalid data at the gateway or at device itself or controlling the frequency of acquiring or cyclically scheduling the set of devices in industrial systems. Data enriches, aggregates, fuses or compacts at the adaptation layer.

5.2.4 Data Categorisation for Storage

Services, business processes and business intelligence use data. Valid, useful and relevant data can be categorised into three categories for storage—data alone, data as well as results of processing, only the results of data analytics are stored. Following are three cases for storage:

1. Data which needs to be repeatedly processed, referenced or audited in future, and therefore, data alone needs to be stored.
2. Data which needs processing only once, and the results are used at a later time using the analytics, and both the data and results of processing and analytics are stored. Advantages of this case are quick visualisation and reports generation without reprocessing. Also the data is available for reference or auditing in future.
3. Online, real-time or streaming data need to be processed and the results of this processing and analysis need storage.

Data from large number of devices and sources categorises into a fourth category called Big data. Data is stored in databases at a server or in a data warehouse or on a Cloud as Big data.

5.2.5 Assembly Software for the Events

A device can generate events. For example, a sensor can generate an event when temperature reaches a preset value or falls below a threshold. A pressure sensor in a boiler generates an event when pressure exceeds a critical value which warrants attention.

Each event can be assigned an ID. A logic value sets or resets for an event state. Logic 1 refers to an event generated but not yet acted upon. Logic 0 refers to an event generated and acted upon or not yet generated. A software component in applications can assemble the events (logic value, event ID and device ID) and can also add Date time stamp. Events from IoTs and logic-flows assemble using software.

5.2.6 Data Store

A data store is a data repository of a set of objects which integrate into the store. Features of data store are:

- Objects in a data-store are modeled using Classes which are defined by the *database schemas*.
- A data store is a general concept. It includes data repositories such as database, relational database, flat file, spreadsheet, mail server, web server, directory services and VMware
- A data store may be distributed over multiple nodes. Apache Cassandra is an example of distributed data store.
- A data store may consist of multiple schemas or may consist of data in only one scheme. Example of only one scheme data store is a relational database.

Repository in English means a group, which can be related upon to look for required things, for special information or knowledge. For example, a repository of paintings of artists. A *database* is a repository of data which can be relied upon for reporting, analytics, process, knowledge discovery and intelligence. A flat file is another repository.

Flat file means a file in which the records have no structural interrelationship (Section 5.3). Section 5.5.1 explains the spreadsheet concept. VMware uses data store to refer to a file that stores a virtual machine.

5.2.7 Data Centre Management

A data centre is a facility which has multiple banks of computers, servers, large memory systems, high speed network and Internet connectivity. The centre provides data security and protection using advanced tools, full data backups along with data recovery, redundant data communication connections and full system power as well as electricity supply backups.

Data centre is meant for data storage, data security and protection

Large industrial units, banks, railways, airlines and units for whom data are the critical components use the services of data centres. Data centres also possess a dust free, heating, ventilation and air conditioning (HVAC), cooling, humidification and dehumidification equipment, pressurisation system with a physically highly secure environment.

The manager of data centre is responsible for all technical and IT issues, operations of computers and servers, data entries, data security, data quality control, network quality control and the management of the services and applications used for data processing.

5.2.8 Server Management

Server management means managing services, setup and maintenance of systems of all types associated with the server. A server needs to serve around the clock. Server management includes managing the following:

- Short reaction times when the system or network is down
- High security standards by routinely performing system maintenance and updation
- Periodic system updates for state-of-the art setups
- Optimised performance
- Monitoring of all critical services, with SMS and email notifications
- Security of systems and protection
- Maintaining confidentiality and privacy of data
- High degree of security and integrity and effective protection of data, files and databases at the organisation
- Protection of customer data or enterprise internal documents by attackers which includes spam mails, unauthorised use of the access to the server, viruses, malwares and worms
- Strict documentation and audit of all activities.

5.2.9 Spatial Storage

Consider goods with RFID tags. When goods move from one place to another, the IDs of goods as well as locations are needed in tracking or inventory control applications. Spatial storage is storage as spatial database which is optimised to store and later on receives queries from the applications.

Suppose a digital map is required for parking slots in a city. Spatial data refers to data which represents objects defined in a geometric space. Points, lines and polygons are common geometric objects which can be represented in spatial databases. Spatial database can also represent database for 3D objects, topological coverage, linear networks, triangular irregular networks and other complex structures. Additional functionality in spatial databases enables efficient processing.

Internet communication by RFIDs, ATMs, vehicles, ambulances, traffic lights, streetlights, waste containers are examples of where spatial database are used.

Spatial database functions optimally for spatial queries. A spatial database can perform typical SQL queries, such as select statements and performs a wide variety of spatial operations. Spatial database has the following features:

- Can perform geometry constructors. For example, creating new geometries
- Can define a shape using the vertices (points or nodes)
- Can perform observer functions using queries which replies specific spatial information such as location of the centre of a geometric object
- Can perform spatial measurements which mean computing distance between geometries, lengths of lines, areas of polygons and other parameters
- Can change the existing features to new ones using spatial functions and can predicate spatial relationships between geometries using true or false type queries.

Reconfirm Your Understanding

- Data generates from the active or passive devices. Data can generate on events at devices. Data can also generate in real time.
- An application interacts for data acquisition. It can configure the devices for data when devices have configuration capability. For example, the devices can be configured to send data at defined periodic intervals, on specific events or in real time.
- Large magnitude of data acquired from a large number of devices, especially, from machines in industrial plants or embedded components data from large number of automobiles or health devices in ICUs or wireless sensor networks. Acquired data of the devices needs validation that data are correct, meaningful or consistent, are within expected ranges or are as per expected patterns and have not become corrupted during transmission.
- Data storage systems store the data. Data is stored in databases at a server, cloud, warehouse or as big data on the cloud.
- Data store is a data repository of a set of objects, which integrate into the store.
- Objects in a data store are modelled using Classes, which the database schemas define.
- Data store includes database, relational database, flat file, spreadsheet, mail server, web server, directory services and VMware. Data store may be distributed over multiple nodes.
- A data store may consist of multiple schemas or may consist of data in only one scheme, such as relational database.
- Data is stored on a server for short reaction times, optimised performance and high security.
- A data centre stores data for data security and protection using the advanced tools, full data backups along with data recovery, redundant data communication connections and full system power. Data store requires data centre management or server management.
- Spatial storage is storage through a spatial database which is optimised to store, enable querying of the data objects defined in a geometric space, and which is a database for 2D and 3D objects, topological coverage, linear networks, triangular irregular networks or other complex structures.

Self-Assessment Exercise

1. List the different types of data which is generated at the devices. ★
2. What does data acquisition mean? What are the benefits of data acquisition by an application after data aggregation, compaction or fusion and enrichment of data takes place from a number of devices? ★★
3. What does data validation mean? When does a data acquisition application consider data invalid? How can an application compensate for the missing or invalid data? ★★
4. How does an application or service support software acquired data of industrial plant machines? Show diagrammatically the in-between physical cum data-link, adaptation, network, transport layers. ★★★
5. What do you mean by data store? What are the different schemas for a data store? ★
6. List the features of a data centre and the activities of a data centre manager. ★★
7. What does server management mean? ★
8. What does spatial database mean? What are additional data fields that spatial data possess? ★★★

5.3 ORGANISING THE DATA

LO 5.2 Classify ways of organising data

Data can be organised in a number of ways. For example, objects, files, data store, database, relational database and object oriented database. Following subsections describe these ways of organising and querying methods.

5.3.1 Databases

Required data values are organised as database(s) so that select values can be retrieved later.

Database

One popular method of organising data is a database, which is a collection of data. This collection is organised into tables. A table provides a systematic way for access, management and update. A single table file is called flat file database. Each record is listed in separate row, unrelated to each other.

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

Relational Database

A relational database is a collection of data into multiple tables which relate to each other through special fields, called keys (primary key, foreign key and unique key). Relational databases provide flexibility. Examples of relational database are MySQL, PostGreSQL, Oracle database created using PL/SQL and Microsoft SQL server using T-SQL.

Object Oriented Database (OODB) is a collection of objects, which save the objects in objected oriented design. Examples are ConceptBase or Cache. Example 5.3 shows the advantages of using relational databases.

Example 5.3

Problem

Recall Example 5.1. Show the advantages of relational databases taking the example of Internet of ACVMs.

Solution

A Manager application receives ACVMs information. It send requests to ACVMs for the chocolates sold, and number required. The request is sent each hour from Manager to Fill service. A Fill service executes on receipt of the requests from ACVMs for chocolate requirement every hour. Application, service and process use the common relational database RDBACVM tables. Table 5.1 is for ACVMs information, and pending service requests Num1, Num 2, Num2, Num 3 and Num 4 for the five flavours available at an ACVM. Table 5.2 is for ACVMs Fill Request Information, and Table 5.3 for chocolates Fill service actions (Tables 5.1 to 5.3).

Table 5.1 RDBAVCM Table A—ACVMs information

Machine ID	Region	Address	Installation Date	Maintenance Schedule	Fill Service Address	Pending Request Number 1	Pending Request Number 2	Pending Request Number 3	Pending Request Number 4

Table 5.2 RDBAVCM Table B—ACVMs fill request information

Service Request Number	Machine ID	Request Receipt DateTime	Number FL1 Request	Number FL 2 Request	Number FL 3 Request	Number FL 4 Request	Number FL5 Request

Table 5.3 RDBAVCM Table C—ACVMs fill service actions

Service Request Number	Service DateTime	Number FL1 Sent	Number FL 2 Sent	Number FL 3 Sent	Number FL 4 Sent	Number FL5 Sent

Common key fields between A, B and C are machine id or service request number. The relationships between fields of A, B and C are maintained by RDBMS. For example, when number of each flavour requested equals the number sent to a machine after processing a service request, then corresponding Num in A becomes 0.

Assume three flat-file databases maintained separately—*A′* for ACVMs information, *B′* for ACVMs Fill Service Request Information and database and *C′* for Fill Service Process. Then every time a service request raises or service request processes, corresponding fields of A′, B′ and C′ in three flat file separate databases require update for maintenance of consistency of database in each application, service and process. Further, each one needs all fields due to no system for the maintenance of the relationships between *A′*, *B′* and *C′*.

Database Management System

Database Management System (DBMS) is a software system, which contains a set of programs specially designed for creation and management of data stored in a database. Database transactions can be performed on a database or relational database.

Atomicity, Data Consistency, Data Isolation and Durability (ACID) Rules

The database transactions must maintain the atomicity, data consistency, data isolation and durability during transactions. Let us explain these rules using Example 5.3 as follows:

Atomicity means a transaction must complete in full, treating it as indivisible. When a service request completes, then the pending request field should also be made zero.

Consistency means that data after the transactions should remain consistent. For example, sum of chocolates sent should equal the sums of sold and unsold chocolates for each flavour after the transactions on the database.

Isolation means transactions between tables 5.1 and 5.2, 5.2 and 5.3 and 5.3 and 5.1 are isolated from each other.

Durability means after completion of transactions, the previous transaction cannot be recalled. Only a new transaction can affect any change.

Distributed Database

Distributed Database (DDB) is a collection of logically interrelated databases over a computer network. Distributed DBMS means a software system that manages a distributed database. The features of a distributed database system are:

- DDB is a collection of databases which are logically related to each other.
- Cooperation exists between the databases in a transparent manner. Transparent means that each user within the system may access all of the data within all of the databases as if they were a single database.
- DDB should be 'location independent', which means the user is unaware of where the data is located, and it is possible to move the data from one physical location to another without affecting the user.[4]

[4] http://www.csee.umbc.edu/portal/help/oracle8/server.815/a67784/ds_ch1.htm

Consistency, Availability and Partition-Tolerance Theorem

Consistency, Availability and Partition-Tolerance Theorem (CAP theorem) is a theorem for distributed computing systems. The theorem states that it is impossible for a distributed computer system to simultaneously provide all three of the Consistency, Availability, Partition tolerance (CAP) guarantees.[5] This is due to the fact that a network failure can occur during communication among the distributed computing nodes. Partitioning of a network therefore needs to be tolerated. Hence, at all times either there will be consistency or availability.

Consistency means 'Every read receives the most recent write or an error'. When a message or data is sought the network generally issues notification of time-out or read error. During an interval of a network failure, the notification may not reach the requesting node(s).

Availability means 'Every request receives a response, without guarantee that it contains the most recent version of the information'. Due to the interval of network failure, it may happen that most recent version of message or data requested may not be available.

Partition tolerance means 'The system continues to operate despite an arbitrary number of messages being dropped by the network between the nodes'. During the interval of a network failure, the network will have two separate set of networked nodes. Since failure can always occur therefore, the partitioning needs to be tolerated.

5.3.2 Query Processing

Query means an application seeking a specific data set from a database. For example, a query at a relational database at bank server may be for the ATM transactions made in a month by a specific customer ID (Example 2.3). Other examples are: most-liked chocolate flavour in the city by children of age group 6 to 10 (Example 5.1); number of times a vehicle visited at the ACPAMS center (Example 5.2) and service was rendered with satisfaction level of 5 out of 5.

Query Processing

Query processing means using a process and getting the results of the query made from a database. The process should use a correct as well as efficient execution strategy. Five steps in processing are:

1. Parsing and translation: This step translates the query into an internal form, into a relational algebraic expression and then a Parser, which checks the syntax and verifies the relations.
2. Decomposition to complete the query process into micro-operations using the analysis (for the number of micro-operations required for the operations), conjunctive and disjunctive normalisation and semantic analysis.
3. Optimisation which means optimising the cost of processing. The cost means number of micro-operations generated in processing which is evaluated by calculating the costs of the sets of equivalent expressions.

[5] https://en.wikipedia.org/wiki/CAP_theorem

4. Evaluation plan: A query-execution engine (software) takes a query-evaluation plan and executes that plan.
5. Returning the results of the query.

The process can also be based on a heuristic approach, by performing the selection and projection steps as early as possible and eliminating duplicate operations.

Distributed Query Processing

Distributed Query Processing means query processing operations in distributed databases on the same system or networked systems. The distributed database system has the ability to access remote sites and transmit the queries to other systems.

5.3.3 SQL

SQL stands for Structured Query Language. It is a language for viewing or changing (update, insert or append or delete) databases. It is a language for data querying, updating, inserting, appending and deleting the databases. It is a language for data access control, schema creation and modifications. It is also a language for managing the RDBMS.

SQL was originally based upon the tuple relational calculus and relational algebra. SQL can embed within other languages using SQL modules, libraries and pre-compilers. SQL features are as follows:

- *Create Schema* is a structure that contains descriptions of objects created by a user (base tables, views, constraints). The user can describe and define the data for a database.
- *Create Catalog* consists of a set of schemas that constitute the description of the database.
- *Use Data Definition Language* (DDL) for the commands that depict a database, including creating, altering and dropping tables and establishing constraints. The user can create and drop databases and tables, establish foreign keys, create view, stored procedure, functions in a database.
- *Use Data Manipulation Language* (DML) for commands that maintain and query a database. The user can manipulate (INSERT, UPDATE or SELECT the data and access data in relational database management systems.
- *Use Data Control Language* (DCL) for commands that control a database, including administering privileges and committing data. The user can set (grant or add or revoke) permissions on tables, procedures, and views.

5.3.4 NOSQL

NOSQL stands for No-SQL or Not Only SQL that does not integrate with applications that are based on SQL. NOSQL is used in cloud data store. NOSQL may consist of the following:

- A class of non-relational data storage systems, flexible data models and multiple schemas

- Class consisting of uninterpreted key and value or 'the big hash table'. For example in [Dynamo (Amazon S3)]
- Class consisting of unordered keys and using the JSON. For example in PNUTS
- Class consisting of ordered keys and semi-structured data storage systems. For examples in the BigTable, Hbase and Cassandra (used in Facebook and Apache)
- Class consisting of JSON (Section 2.3). For example in MongoDb[6] which is widely used for NOSQL)
- Class consisting of name and value in the text. For example in CouchDB
- May not require a fixed table schema

NOSQL systems do not use the concept of joins (in distributed data storage systems). Data written at one node replicates to multiple nodes, therefore identical and distributed system can be fault-tolerant, and can have partitioning tolerance. CAP theorem is applicable. The system offers relaxation in one or more of the ACID and CAP properties. Out of the three properties (consistency, availability and partitions), two are at least present for an application.

- *Consistency* means all copies have same value like in traditional DBs.
- *Availability* means at least one copy available in case a partition becomes inactive or fails. For example, in web applications, the other copy in other partition is available.
- *Partition* means parts which are active but may not cooperate as in distributed databases.

5.3.5 Extract, Transform and Load

Extract, Transform and Load or ETL is a system which enables the usage of databases used, especially the ones stored at a data warehouse. *Extract* means obtaining data from homogeneous or heterogeneous data sources. *Transform* means transforming and storing the data in an appropriate structure or format. *Load* means the structured data load in the final target database or data store or data warehouse.

All the three phases can execute in parallel. Data extraction takes longer time. Therefore, the system while pulling data, executes another transformation processes on already received data and prepares the already transformed data for loading. As soon as data are ready for load into the target, the data load starts. It means next phase starts without waiting for the completion of the previous phases.

ETL system usages are for integrating data from multiple applications (systems) hosted separately.

5.3.6 Relational Time Series Service

Time series data means an array of numbers indexed with time (date-time or a range of date-time). Time series data can be considered as time stamped data. It means data carries along with it the date and time information about the data values. For example, sales of chocolates in Internet of ACVMs (Example 5.1) are different on different dates and times.

[6] http://www.w3resource.com/mongodb/introduction-mongodb.php MongoDb

The sales need indexing with a range between two dates or indexed with date-time. A time series of sales is called sale profile of the ACVMs. A time series of log of chocolate sales is called chocolate sales trace.

Time series is any data-set that is accessed in a sequence of time. Software programs and an analytics program analyses the set in a time series, meaning analyses in a chronological order. IoT devices, such as temperature sensors, wireless sensor network nodes, energy meters, RFID tags, ATMs, ACVMs generate time-stamped or time series data.

Time Series Database (TSDB) is a software system which implements a database that optimally handles mathematical operations (profiles, traces, curves), queries or database transactions on time series.

Conventional database systems, Relational Database System (RDMS) or flat file database software may not be modelled for time series handling and may not therefore function efficiently for time series data with complex logic or business rules and need of high transaction throughout time series data.

IBM Informix TimeSeries software expanded database functionality by adding efficient storage, faster load of data to enable fast query processing and transactions, fast performance, sophisticated support for managing time series. Server can have built-in time series Informix software for the handling of IoT time series data.

5.3.7 Real-Time and Intelligence

Decision on real-time data is fast when query processing in live data (streaming) has low latency. Decision on historical data is fast when interactive query processing has low latency. Low latencies are obtained by various approaches: Massively Parallel Processing (MPP), in-memory databases and columnar databases.

TeraData Aster and Pivotal Greenplum are examples of MPP. In-memory and on-store both transaction methods exist for the databases. SAP Hana and QClick view are examples of in-memory databases. SAP Sybase IQ and HP Vertica are examples for columnar databases for faster Analytics.

Reconfirm Your Understanding

- A database is a collection of data which is organised as tables. A relational database is a collection of data organised as multiple tables, which relate to each other through special fields. Object Oriented Database (OODB) is a collection of objects, which saves the objects in objected oriented design.
- Database Management System is a software system, which contains a set of programs specially designed for creation, management and transactions of data stored in a database.
- Database transaction is the execution of a specific set of operations on a database. Relational database transaction is the execution of interrelated instructions using relations. A transaction is a sequential execution of a specific set of relational operations on relational database.

- Database transaction models state that transactions must maintain transaction atomicity, data consistency, data isolation and durability.
- Query means to an application or service seeking a specific data set from a database. Query processing means using a process and getting the results of the query made from a database.
- CAP theorem applies to distributed computing nodes. CAP theorem states that for distributed computing systems, it is impossible for a distributed computer system to simultaneously provide all three, consistency, availability, partition tolerance (CAP), guarantees.
- Distributed database is a collection of logically interrelated, cooperating databases over a computer network. A distributed database system has the ability to access remote sites and transmit queries.
- Distributed query processing means query processing operations in distributed databases on same system or networked systems.
- SQL is a language for data access control, schema creation and modifications. It is a language for managing the RDBMs. It is a language for data definition, data manipulation and data control instructions.
- NOSQL stands for Not Only SQL or No-SQL and no integration with applications that are based on SQL. It is used in Cloud Data Store. NOSQL consists of classes of non-relational data storage systems, flexible data models and multiple schemas.
- Time series data means an array of numbers indexed with time (date-time or a range of date-time).
- Decision on real-time data is fast when query processing in live data (streaming) has low latency. The decision on historical data is fast when interactive query processing has low latency.

Self-Assessment Exercise

1. What does a relational database mean? ★
2. List the differences between flat-file and relational databases. ★★
3. Consider relational database tables, A, B and C. Here is A bank information (name, address, phone numbers, IFSC code), B is customers' information (IDs, names, addresses, phone numbers, Account Types, Account Numbers) and C is Bank Passbook Transactions details (Date of Transaction, Credit, debit, Debit amount, Credit amount and Balance). What will be the keys used? How does the data in the tables relate? C relates to which fields in A and B. ★★★
4. What are three essential features when using distributed databases? ★
5. What does TSDB mean? ★
6. What are the features of SQL? ★
7. How does SQL differ from NOSQL? ★★
8. List the differences between time-series database system and RDBMS in construction and usages. ★★★

5.4 TRANSACTIONS, BUSINESS PROCESSES, INTEGRATION AND ENTERPRISE SYSTEMS

LO 5.3 Summarise the transactions on stored data, functions for business-processes and business intelligence, and the concepts of IoT applications—integration and services architecture

A transaction is a collection of operations that form a single logical unit. For example, a database connect, insertion, append, deletion or modification transactions. Business transactions are transactions related in some way to a business activity.

5.4.1 Online Transactions and Processing

Recall Example 2.3—OLTP means process as soon as data or events generate in real time. OLTP is used when requirements are availability, speed, concurrency and recoverability in databases for real-time data or events. Example 5.4 gives the uses of OLTP in the application and network domain in Internet of ATMs (ATM of a bank) connected to a bank server.

Example 5.4

Problem

What are the usages of OLTP in the application and network domain in Internet of ATMs (ATM of a bank) connected to a bank server?

Solution

Server applications need processing and update-intensive database management with a high throughput. The requirements in these applications are availability, speed, concurrency and recoverability, and reduced paper trails. Therefore, the transactions at ATMs need OLTP.

Batch Transactions Processing

Batch transactions processing means the execution of a series of transactions without user interactions. Transaction jobs are set up so they can be run to completion. Scripts, command-line arguments, control files, or job control language predefine all input parameters.

Batch processing means a transaction process in batches and in an non-interactive way. When one set of transactions finish, the results are stored and a next batch is taken up. A good example is credit card transactions where the final results at the end of the month are used. Another example is chocolate purchase transactions. The final results of sell figures from ACVMs can communicate on the Internet at the end of an hour or day.

Streaming Transactions Processing

Examples of the streams are log streams, event streams and twitter streams. Query and transactions processing on streaming data need specialised frameworks. Storm from Twitter, S4 from Yahoo, SPARK streaming, HStreaming and flume are examples of frameworks for real-time streaming computation frameworks.

Interactive Transactions Processing

Interactive transactions processing means the transactions which involve continual exchange of information between the computer and a user. For example, user interactions during e-shopping and e-banking. The processing is just the opposite of batch processing.

Real-time Transactions Processing

Real-time transaction processing means that transactions process at the same time as the data arrives from the data sources and data store. An example is ATM machine transactions. In-memory, row-format records enable real-time transaction processing. Row format means few rows and more columns. The CPU accesses all columns in single accesses in SIMD (single instruction multiple data) streams processing.

Event Stream Processing and Complex Event Processing

Event Stream Processing (ESP) is a set of technologies, event processing languages, Complex Event Processing (CEP), event visualisation, event databases and event-driven middleware. Apache S4 and Twitter Storm are examples of ESPs. SAP Sybase ESP and EsperTechEsper are examples of CEPs. ESP and CEP does the following:

- Processes tasks on receiving streams of event data
- Identifies the meaningful pattern from the streams
- Detects relationships between multiple events
- Correlates the events data
- Detects event hierarchies
- Detects aspects such as timing, causality, subscription membership
- Builds and manages the event-driven information systems.

Complex Event Processing

CEP has many applications. For example, IoT event processing applications, stocks algorithmic-based trading and location-based services. A CEP application in Eclipse are used for capturing a combination of data, timing conditions and efficiently recognise the corresponding events over data streams.

5.4.2 Business Processes

A business process consists of a series of activities which serves a particular specific result. It is used when an enterprise has a number of interrelated processes which serve

a particular result or goal. The results enable sales, planning and production. The BP is a representation or process matrix or flowchart of a sequence of activities with interleaving decision points.

Internet of RFIDs enables a business process called tracking of RFID labelled goods (Example 2.2) which also enables inventory control process.

IoT/M2M enables the devices' data in databases for business processes. The data supports the process. For example, consider a process, streetlights control and management (Example 1.2). Each group of streetlights sends data in real time through the gateways. The gateways connect to the Internet. The control and management processes streetlights real time databases and group databases.

5.4.3 Business Intelligence

Business intelligence is a process which enables a business service to extract new facts and knowledge and then undertake better decisions. The new facts and knowledge follow from the earlier results of data processing, aggregation and then analysing those results. Example 5.5 shows business intelligence for Internet of ACVMs, whereas Example 5.6 shows business processes, intelligence and BP architecture reference model in Automotive Maintenance Application at the Service Centre.

Example 5.5

Problem

Recall Internet of ACVMs (Example 5.1). What are the new facts and knowledge that follow from the earlier results of business processes?

Solution

Consider Fill service for the ACVMs. The BI extracts the knowledge about the required Fill service frequency from the facts. Also extract the strategy to service the filling of chocolates in ACVMs in specific areas of the city such that all flavours can be simultaneously dispatched. BI lies in the service to the machines in the area when cost incurred is minimum on one hand and each machine is able to serve on user demand any flavour without fail.

Example 5.6

Problem

Recall automotive maintenance application at a service centre (Example 5.2). Draw BI/BP architecture for automobile components service process at ACPAMS.

Solution

The predictive analytics of the acquired data in databases enables the service to extract the knowledge. It gets the prediction of components needing service from the facts. The service to a set of automobile components has to be timely and preventive. BI lies in the service to the automobile when components are serviced or replaced timely with least number of visits to the service centre. Figure 5.1 shows model architecture for BI and BP at automobile service centre.

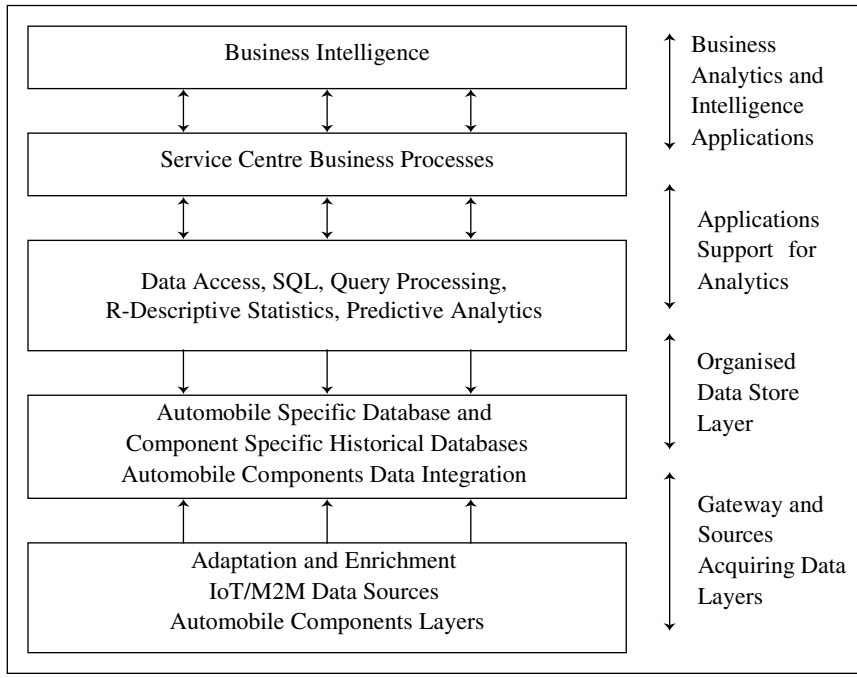


Figure 5.1 Architecture reference model for the business intelligence and business processes at ACPAMS

5.4.4 Distributed Business Process

Several times, business processes need to be distributed. Distribution of processes reduces the complexity, communication costs, enables faster responses and smaller processing load at the central system. For example, recall Example 1.2, distribution of control process for each group of lights at the gateway itself reduces complexity, communication costs, faster responses and smaller processing load at the central system.

Distributed Business Process System (DBPS) is a collection of logically interrelated business processes in an Enterprise network. DBPS means a software system that manages the distributed BPs. DBPS features are:

DBPS is a collection of logically related BPs like DDBS. DBPS exists as cooperation between the BPs in a transparent manner. Transparent means that each user within the system may access all of the process decisions within all of the processes as if they were a single business process.

DBPS should possess 'location independence' which means the enterprise BI is unaware of where the BPs are located. It is possible to move the results of analytics and knowledge from one physical location to another without affecting the user.[1]

Example 5.7 shows distributed business processes in an automobile enterprise.

Example 5.7

Problem

Recall automotive maintenance application at an ACPAMS centre (Examples 5.2 and 5.6). Enterprise business intelligence lies in predictive and prescriptive analytics based services to the automobile components, which are serviced or replaced timely with minimum number of visits to service centres. Draw business intelligence and business processes architecture for automobile components service process.

Solution

Figure 5.2 shows model architecture for distributed BI and BP at an automobile enterprise. Two business processes are at enterprise layer, viz. one at network layer and one at devices and gateway layer.

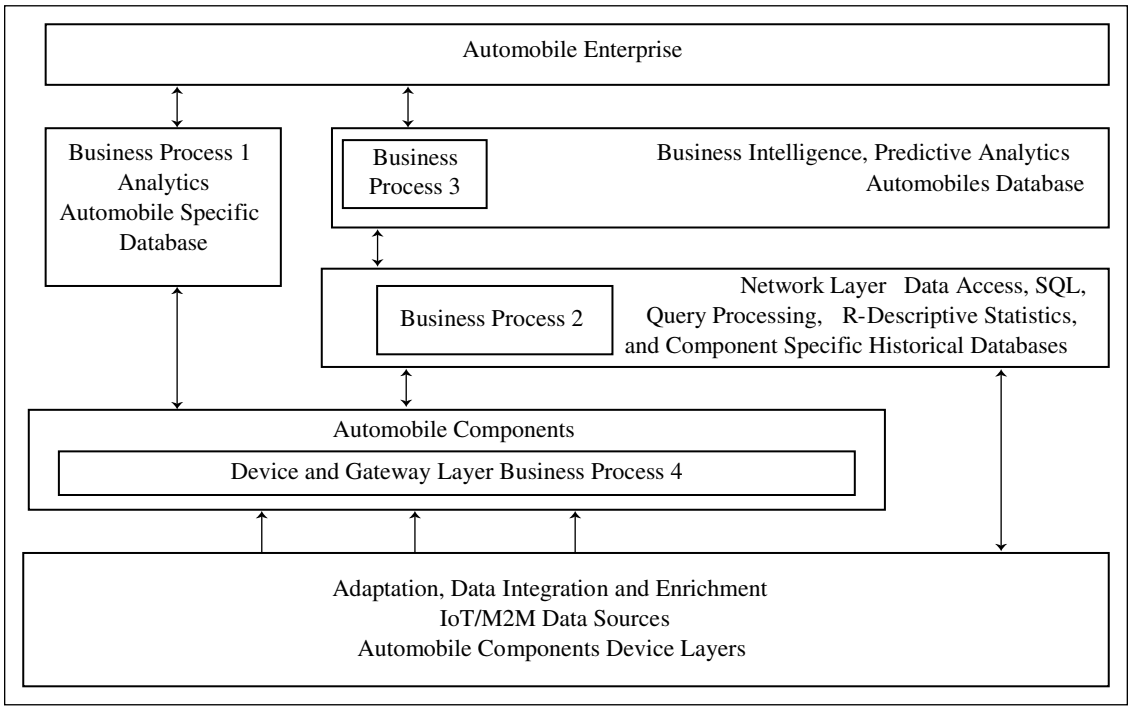


Figure 5.2 Distributed interrelated business intelligence and processes at the enterprise, network, and devices and gateway layers

Interrelationships in distributed BPs are:

- Enterprise layer business process 1 (EBP1) interrelates directly with the device and gateway layer Business process 4 (DGBP4) thus with the device data, adaptation, data integration and enrichment layer for a specific automobile. EBP1 analytics is with non-historical data.
- Network layer Business Process 2 (NBP2) interrelates directly with the enterprise layer business process 3 (EBP3). NBP2 uses the data access, SQL, query processing, R- descriptive statistics and component specific historical databases. NBP 2 has access to data of number of automobiles of same the model as one sending data to EBP1. EBP3 analytics is with the other automobile databases and historical databases, and enables predictive and prescriptive analytics.
- NBP2 interrelates directly with the DGBP4. This enables updating the database for the NBP2.

5.4.5 Complex Applications Integration and Service Oriented Architecture

An enterprise has number of applications, services and processes. Heterogeneous systems have complexity when integrating them in the enterprise.

Following are the standardised business processes, as defined in the Oracle application integration architecture:

- Integrating and enhancing the existing systems and processes
- Business intelligence
- Data security and integrity
- New business services and products (web services)
- Collaboration and knowledge management
- Enterprise architecture and SOA
- e-commerce
- External customer services
- Supply chain automation and analytics results visualisation
- Data centre optimisation

IoT applications, services and processes enhance the existing systems in a number of enterprises. For example, an automobile enterprise has a number of divisions. Each division has Sales, Customer Relations Management, Automobile Maintenance Services, and Accounting. IoT-based services help in business intelligence, processes and systems, such as post-sales services and supply chain automation and analytics results in visualisation enhancement of the services from an enterprise.

Complex application integration means integration of heterogeneous application architectures and number of processes. SOA consists of services, messages, operations and processes.

SOA components distribute over a network or the Internet in a high-level business entity. New business applications can be developed using a SOA.

5.4.6 Integration and Enterprise Systems

Figure 5.3 shows complex applications integration architecture and SOA of cloud-based IoT services, web services, cloud services and services.

Process orchestration means a number of business processes running in parallel and a number of processes running in sequence. The process matrix provides the decision points which indicate which processes should run in parallel and which in sequence. An SOA models the number of services and interrelationships. Each service initiates on receipt of messages from a process or service.

The service discovery and selection software components select the services for application integration. Service orchestration software coordinates the execution of the number of services, cloud services, cloud IoT services and web services. Services run in parallel and a number of processes in sequences.

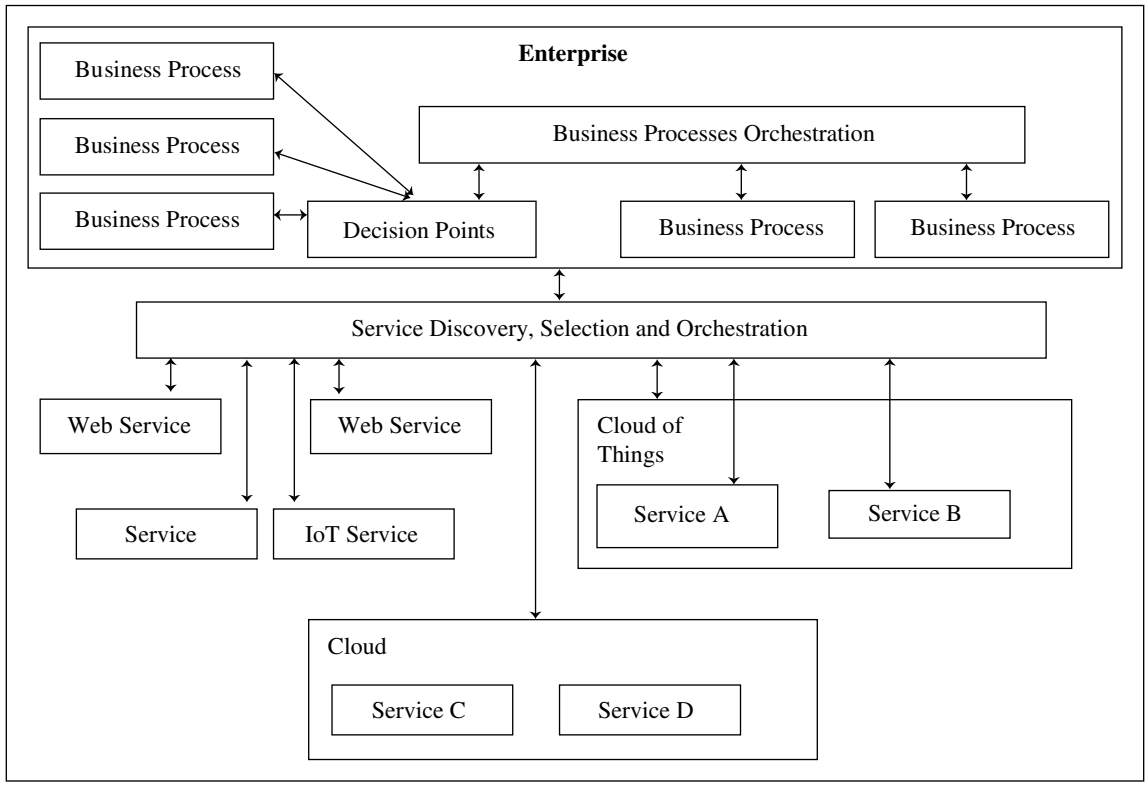


Figure 5.3 Complex applications integration architecture and SOA of cloud-based IoT services, web services, cloud services and services

Reconfirm Your Understanding

- A database transaction is a collection of operations that form a single logical unit of database. A transaction means operations such as connect, insertion, append, deletion or modification in a unit of database. *Business transactions* are transactions related in some way to a business activity.
- OLTP stands for Online Transactions Processing, refers to processing of data or events in real time. Batch transaction processing means transaction processing in batches and in a non-interactive way. Stream transactions processing on streaming data need specialised frameworks. Real-time transaction processing means that transaction processing is done at the same time as the data arrives from the data sources and data stores.
- Complex event processing application uses are for capturing a combination of data, timing conditions and efficiently recognising the corresponding events over data streams.
- A business process consists of a series of activities. The process may consist of a collection of interrelated structured activities or tasks or processes which follow a logical sequence.
- A process matrix has a number of elements. Each element may represent a series of operations and activities on a given set of inputs that perform a specific task leading to a decision point in the process.

- Business intelligence is a process that enables a business service to extract new facts and knowledge and then undertake better decisions. The new facts and knowledge follow from the earlier results of data processing, aggregation and then analysing those results.
- Distribution of processes reduces complexity, communication costs, and enables faster responses and smaller processing load at the central system.
- Distributed business process system (DBPS) is a collection of logically interrelated business processes in an enterprise network.
- Complex application integration means integration of heterogeneous application architectures and number of processes.
- SOA consists of enterprise and service discovery, selection and orchestration layers for services, messages, operations and processes.

Self-Assessment Exercise

1. What do batch transactions, streaming transactions and real-time transactions processing mean? ★★
2. List the tasks which event stream processing and complex stream processing do. ★
3. What does a process matrix mean? ★
4. List the steps in tracking a business process in Internet of RFIDs. Draw diagrammatically the actions at physical cum data-link, data adaptation, network, transport, application-support and application layers. ★★★
5. What is the benefit of device and gateway layer business process in distributed business processes in automobile enterprise? ★★★
6. List the standardised business processes in the Oracle application integration architecture. ★★
7. Why does OLTP operation run fast in row format? ★
8. Draw waste container management business intelligence and business processes architecture for Internet of Waste containers. Assume that each container generates an event on getting 90% filled up and communicates the event along with the container ID. ★★★

5.5 ANALYTICS

LO 5.4 Identify the functions and usage of data analytics and data visualisations for IoT applications and business processes

Organised data after acquiring from the devices can be used for multiple purposes. Applications usually use the data of devices in two ways—for monitoring, reporting and rule-based actions. For example, in Internet of Streetlights applications just do that (Example 1.2); for analytics, new

facts and taking decisions based on those facts. For example, Internet of ACVMs can use analytics, new facts are found and those facts enable taking of the decisions for new option(s) to maximise the profits from the machines (Example 5.1).

Example 5.8 gives the uses of analytics in the application and network domain in Internet of ATMs (ATM of a bank) connected to a bank server.

Example 5.8

Problem

What are the usages of analytics in the application and network domain in Internet of ATMs connected to a bank server?

Solution

An ATM machine generates income from transaction charges, charges for advertisements at the machine locations and idle-state advertisements of bank products and bank services at the display screen. Each machine's usage is analysed during each hour on a daily basis in each segment. Each machine's expenses and incomes are used for cost-benefit analysis. Analytics enable faster and more accurate scheduling. It enables timely actions, enables optimum scheduling region wise for the cash supply service, special scheduling on special days, such as days near dates of receiving salary, festivals and holidays. The analytics also enable each region machines the maintenance, scheduling and ATM location relocation.

An enterprise creates sections and unit-wise analytics. The analytics enable fact-based decision making in place of intuition-drive decision making. Analytics provides business intelligence. It is a key for the success of an enterprise business.

Analytics require the data to be available and accessible. It uses arithmetic and statistical, data mining and advanced methods, such as *machine learning* to find new parameters and information which add value to the data. Analytics enable building models based on selection of right data. Later the models are tested and used for services and processes.

5.5.1 Analytics Phases

Analytics has three phases before deriving new facts and providing business intelligence. These are:

1. *Descriptive analytics* enables deriving the additional value from visualisations and reports.
2. *Predictive analytics* is advanced analytics which enables extraction of new facts and knowledge, and then predicts or forecasts.
3. *Prescriptive analytics* enables derivation of the additional value and undertake better decisions for new option(s) to maximise the profits.

Descriptive Analytics

Descriptive analytics answers the questions about what happened in the past. Descriptive analytics means finding the aggregates, frequencies of occurrences, mean values (simple

or geometric averages) or variances in values or groupings using selected properties and hence applying these. Descriptive analytics enable the following:

- Actions, such as Online Analytical Processing (OLAP) for the analytics
- Reporting or generating spreadsheets
- Visualisations or dashboard displays of the analysed results
- Creation of indicators, called key performance indicators.

Descriptive Analytics Methods

- ***Spreadsheet-based reports and data visualisations:*** *R*esults of descriptive analysis can be presented in a spreadsheet format before creating the data visuals for the user. Spreadsheet enables user visualisation of *what if*. For example, if sales of chocolates of specific flavour drop by 5% on specific set of ACVMs, how it will influence the profitability? A spreadsheet is a table. The values are in the cells in the rows and columns. Each value can have a predefined relationship to the other values. For example, a value in cell C_jR_i (cell at j^{th} column and i^{th} row) can be related to another cell or a set of cells through a formula or Boolean relation or statistically analysed value.
- ***Descriptive statistics-based reports and data visualisations:*** Descriptive analysis can also use descriptive statistics. Statistical analysis means finding peak, minima, variance, probabilities, and statistical parameters. Formulae are used for the data sets to enable the data showing variations understandable.
- ***Data mining and machine learning methods in analytics:*** Data mining analysis means use of algorithms which extract hidden or unknown information or patterns from large amounts of data. Machine learning means modelling of the specific tasks.

  SAS and SPSS are two tools. *R* is a programming language and software environment for statistical computing and graphics. The language is also the core of many open source products. Descriptive analytics enable intelligence for further actions.

Example 5.9

Problem

a. Recapitulate Examples 5.1, 5.3 and 5.5. How are the descriptive analytics used in Internet of ACVMs?
b. How is the spreadsheet method used in Internet of ACVMs?

Solution

a. Descriptive analytics looks at past performance and the performance is evaluated by mining historical data. The analytics find the reasons behind past performance or success or failure. Reports for management use this type of analysis. For example, report the analysis of the sales in individual areas, sales on individual occasions, analysing children preferences, analysing flavour preferences, expenses and incomes from individual regions, incomes from various sources, chocolate sales, advertisements during idle state displays at the machines, or chocolate sales on the machines, and so on.
b. Recall Example 5.3. Spreadsheet design is as follows: A cell in a row is for an ACVM id and another for the period under consideration. Five other cells are for the numbers sold during the period for the

flavours FL1 and FL5. A predefined formula calculates the profitability from the values in the cells and values of the purchase and sell-prices for each flavour, FL1 and FL5. If a value given in one of the five cell changes then new profitability figure automatically calculates using the predefined formulae in each row and the summing row. The spreadsheet analytics can be graphically visualised.

- ***Online analytical processing (OLAP) in analytics:*** OLAP enables viewing of analysed data up to the desired granularity. It enables view of rollup (finer granulites data to coarse granulites data) or drill down (coarser granulites data to finer granulites data). OLAP enables obtaining summarized information and automated reports from large volume database. Results of queries are based on Metadata. Metadata is data which describes the data. Pre-storing calculated values provide consistently fast response.

  OLAP uses the analysis functions which are not possible to code in SQL. The data structure is designed from the users perspective, using Spreadsheet like formulae.

  OLAP is a significant improvement over query systems. OLAP is an interactive system to show different summaries of multidimensional data by interactively selecting the attributes in a multidimensional data cube.[7]

  OLAP enables analysing data in multiple dimensions in a structure called data cube. Each dimension represents a hierarchy. Each dimension has a dimension attribute which defines the dimension and summary of measure attribute.

  A *slice* of a data-cube can be viewed when values of multiple dimensions are fixed. A *dice* of a data-cube can be viewed with variable values in multiple dimensions. *Slicing and dicing functionalities* mean selecting specific values for these attributes, which are then displayed on top of the cross-tables.

  A slice means a data relationship in the analysed multiple dimensional data. A slice of a data relationship between two attributes can be individually visualised. For examples, monthly sales versus flavours sold at the chain of ACVMs in Example 5.1 after the analysis.

  A cubical dice has six faces, each face marked distinctly. Face 1 has one dot, face 2 two, and so on. Sixth face has six dots. Similarly, six different cross referenced tables can be created during OLAP for three-dimensional structure for analysing data. An n-dimensional structure will have 2-*n* faces (tables). Each table and corresponding visual gives a relationship between two attributes. The tables are cross referenced.

  Dicing is a process of creating cross referenced tables, each viewable separately on n-dimensional structure faces.

OLAP can be one of the three types: multidimensional OLAP (MOLAP), relational OLAP (ROLAP) and hybrid OLAP (HOLAP).

Example 5.10 explains multidimensional data-cube analytics.

[7] https://en.wikipedia.org/wiki/Online_analytical_processing

Example 5.10

Problem

How are the OLAP used for analytics in Internet of ACVMs (Example 5.1)?

Solution

Consider Internet of ACVMs (Example 5.1). First dimension can be for time intervals, varying in hierarchy from hour, day, week, month and year. Second dimension can be number of installed machines with hierarchically aggregated values from 10s, 50s, 100s, and so on. The third dimension can be number of chocolates sold, with hierarchically aggregated values, 100s, 1000s, 10000s, and so on. Fourth dimension can be number of individual chocolate flavours sold, 50s, 100s, 150s, 200s and so on of a specific flavour. Similarly, fifth, sixth and other dimensions can be specified for other flavours.

OLAP uses the following steps:

Identify the dimensions, each with an attribute and hierarchy. For example, identifying dimensions: 1 number of time intervals, 2 number of installed machines, 3 total number of chocolates of all five flavours sold.

Analyse cross tabulations (Row header with one attribute, column header with another attribute and cell having the aggregate value or calculated value according to a formula or analysis). For example, table of number of chocolates sold and time intervals as well as table of number of chocolate sold and number of machines.

Visualise n-dimensional cube-data; cube means integration of fact table with cross-dimensional tables, visualising the slices and dice faces.

When visualising analysed results, first take a whole, then region wise and then individual ACVMs means drilling down views (from coarse granularity of analysis to finer granularity analysis). Next when visualising analysed results, first consider flavour wise and then as a whole means rolling up views (from finer granularity of analysis to coarser granularity analysis).

Advanced Analytics: Predictive Analytics

Predictive analytics is advanced analytics in which the user interprets the outputs of descriptive analytics

Predictive analytics answer the question "What will happen?" Predictive analytics is advanced analytics. The user interprets the outputs from advanced analytics using descriptive analytics methods, such as data visualisation. For example, output predictions are visualised along with the yearly sales growth of past five years and predicts next two years sales. Another example, output predictions for the next cycle of automobile sells are visualised along yearly cycles of sales growth and fall in past ten years. Visualising can show the effects to increased competition for a product in years ahead and take decisions, such as need to change product mix and introducing new car models.

Predictive analytics uses algorithms, such as regression analysis, correlation, optimisation, and multivariate statistics, and techniques such as modelling, simulation, machine learning, and neural networks. The software tools make the predictive analytics easy to use and understand. The examples are as follows:

- Predicting trends
- Undertaking preventive maintenance from earlier models of equipment and device failure rates

- Managing the campaign with integrated marketing strategy from previous studies of effect of campaigns with respect to media types, regions, targeted age group
- Predicting by identifying patterns, clusters with similar behaviour
- Predicting based on anomalous characteristics, anomaly detection

The results of predictions need verifications from a domain knowledge, and view from multiple angles.

Prescriptive Analytics

Prescriptive analytics answers not only what is anticipated or what will happen or when it will happen, but also why it will happen based on the input from descriptive analytics and business rules. This final phase, additionally to the prediction also suggests actions for deriving benefits from predictions, and shows the implications of the decision options or the optimal solutions or new resource allocation strategies or risk mitigation strategies. Prescriptive analytics suggest best course of actions in the given state or set of inputs and rules.

5.5.2 Event Analytics

Events definable options are unique, non-interaction or interaction options for the events. Event analytics use event data for events tracking and event reporting. An event has the following components:

- *Category*–an event of chocolate purchase in ACVM example belongs to one category and event of reaching predefined threshold of sell for specific chocolate flavour which belongs to other category
- *Action*–sending message from ACVM on completing predefined sell is the action taken on the event
- Label (optional)
- Value (optional)–on event, messaging the number of chocolate of that flavour sold or remaining.

Event analytics generate event reports using event metric, such as event counts for a category of events, events acted upon, event pending action, rate of new events generation in that category.

5.5.3 In-memory Data Processing and Analytics

In-memory option of row or column formats can be selected in certain databases, for example, Oracle dual format architecture database that enables to run the real-time, ad-hoc, analytic queries on IoTs' data.

In-memory and On-store Row Format Option (Few Rows and Many Columns)

Consider the transactions of the type, ATM transactions or sales order transactions. Each row has separate record. For example, separate record for each ACVM or each bank customer or each sales order. The columns have data associated with the record. A row

format enables quick access of all columns for a record. OLTP operations run fast in the row format. There are fewer rows and more columns. For example, updates, inserting new transactions or querying the transactions of specific amount. A row format can be optimized for OLTP operations. The operations access only few rows and need quick access to the columns.

A row format, allowing row data, will be brought into the CPU with a single memory reference. Data for each record is together in-memory and on-store. There is a single copy of the table on storage. Recall Example 5.1 for Internet of ACVMs. For example, required chocolates of each flavour for distinct ACVMs in row format in-memory database enable faster querying.

In-memory and On-store Column Format Option (Few Columns and More Rows)

Consider analytics of the type, monthly sales of chocolates on the ACVMs, enterprise yearly profits. Analytical workloads access few columns but scan the entire data set. Analytics therefore run faster on column format, more rows and few columns. Fast for processing needs few columns and many rows. They typically require aggregation or fusion or compaction also. A columnar format allows for much faster data retrieval when only a few columns in a table are selected because all the data for a column is kept together in-memory in column format option. A single memory access will load many column values into the CPU. It also lends itself to faster filtering and aggregation, making it the most optimised format for analytics.

5.5.4 Real-time Analytics Management

Real-time analytics management means ensuring faster OLTP as well as OLAP. Real-time analytics works both as direct querying using an OLTP database and in a data warehouse and OLAP on queried results. Queries return fast, databases such as Oracle database provides in-memory row format option large speedups for OLTP applications and in-memory column format option for large speedups for OLAP applications.

Example 5.11

Problem

Give example of dual format in-memory architecture of databases.

Solution

In-memory option of row or column formats can be selected in certain databases. For example, Oracle dual format architecture databases for in-memory columnar as well as row formats.

Oracle's unique dual format architecture allows data to be stored in both row and column format simultaneously. The Oracle database in-memory option is designed to be completely compatible with and transparent to the existing Oracle applications and is trivial to deploy. This eliminates the trade-offs required in those databases which offer only one format option, format for faster access during OLTP or faster access during analytics. Such databases needs generation of second copy for analytics, thus delaying costs, additional storage costs and synchronisation issues.

The Oracle optimiser is in-memory aware. It has been optimized to automatically run analytic queries using the column format, and OLTP queries using the row format. Oracle's in-memory columnar technology is a pure in-memory format. The in-memory columnar format does not persist on storage.

5.5.5 Analytics using Big Data in IoT/M2M

Big data is data of high volume, variety and velocity, and may also include veracity

Big data means extreme amount of data. Big data also means data of high volume, variety and velocity (3Vs) or one which also includes veracity (4Vs).

Volume means data received from number of sources of data, including data sets with sizes beyond the ability of commonly used software tools to acquire, manage and process data within a tolerable elapsed time.

Variety means structured as well as unstructured data in different formats, variety of data on which no SQL (Structured Query Language) applicable.

Velocity means data received with higher rates due to use of number of sources of data.

Veracity means variation in data quality for analytics. The analytics need the trustable data, filtered data after removing—anomalous data, non-standard and not cross referencing data.

Big data also refers simply to the use of predictive analytics or other certain advanced methods which extract the value from data. Big data seldom relates to just a particular size of data set. Extreme amount of data means data with additional information—situational information. For example, information from analysis of data of time period, festival days, holidays, and data of different locations, and contextual information, which means data gathered in specific contexts.

5.5.6 Big Data Analytics

Big data is multistructured data while RDMS maintain more structured data. The open source software Hadoop and MapReduce are from Apache Software. They enable storage and analyse the massive amounts of data. Hadoop File System (HDFS), Mahout, a library of machine learning algorithms and HiveQ, a SQL like scripting language software are used for Big data analytics in the Hadoop ecosystem. MapReduce is a programming model and a core of Hadoop. Large data sets process onto a cluster of nodes using MapReduce. Same node runs the algorithm using the data sets at HDFS and processing is at that node itself.

Hadoop is an open-source framework. The framework stores and processes big data. The clusters of computing nodes process that data using simple programming models. Processing takes place in a distributed environment. The framework scales up from single server to thousands of processing machines and servers, each offering environment of local storage and processing. Hadoop accesses data in sequential manner and performs batch processing. A new data set results from input data set that also processes sequentially.

HBase is an example of columnar format data storage which enables read or write access in real time for very large tables distributed in Hadoop File System (HDFS). HBase is database for big data. Data access is random access. Therefore, it provides fast look-up from large tables and access latency is small. HBase uses big hash tables. HBase can be considered similar to Google's BigTable.

5.5.7 Data Analytics Architecture and Stack

Analytics architecture consists of the following layers:

- Data sources layer
- Data storage and processing layer
- Data access and query processing layer
- Data services, reporting and advanced analytics layer

Figure 5.4 shows an overview of a reference model for analytics architecture. Figure 5.4 also shows on the right-hand side the layers in the reference model.

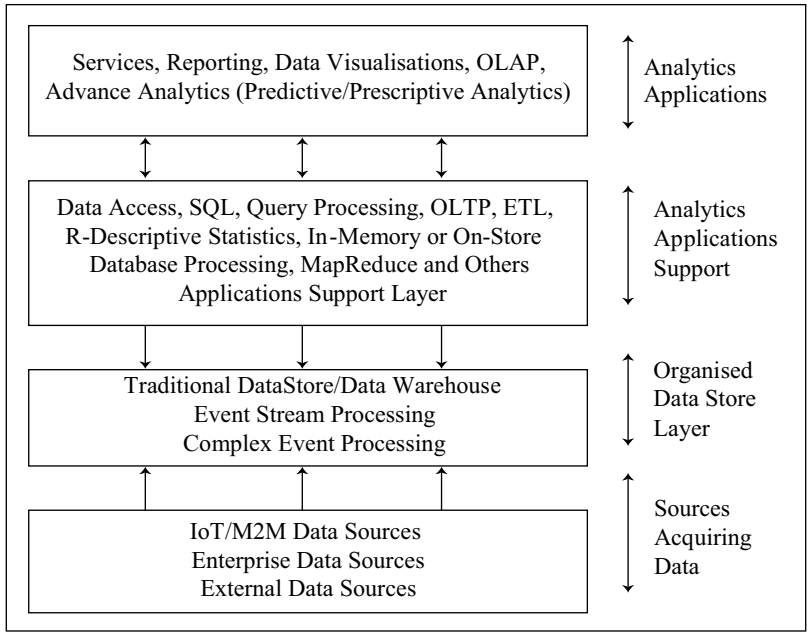


Figure 5.4 Analytics Architecture Reference Model

Analytical sandbox means analytics tools and analytics environment for predictive analytics on multistructured data. Mesos v0.9 is a resources management platform which enable multiple frameworks sharing of cluster of nodes and which is compatible with open analytics stack [data processing (Hive, Hadoop, HBase, Storm), data management (HDFS)].

Berkeley Data Analytics Stack (BDAS) consists of data processing, data management and resource management layers.

Applications, AMP-Genomics and Carat run at the BDAS. Data processing software component provides in-memory processing which processes the data efficiently across the frameworks. AMP stands for Berkeley's Algorithms, Machines and Peoples Laboratory.

Data processing combines *batch*, *streaming* and *interactive* computations.

Resource management software component provides for sharing the infrastructure across the frameworks.

Figure 5.5 shows an overview of BDAS architecture which is a reference model for analytics architecture. Figure 5.5 also shows on right-hand side the file system, library of machine learning algorithms and SQL like scripting language software for the Big data analytics in Hadoop ecosystem.

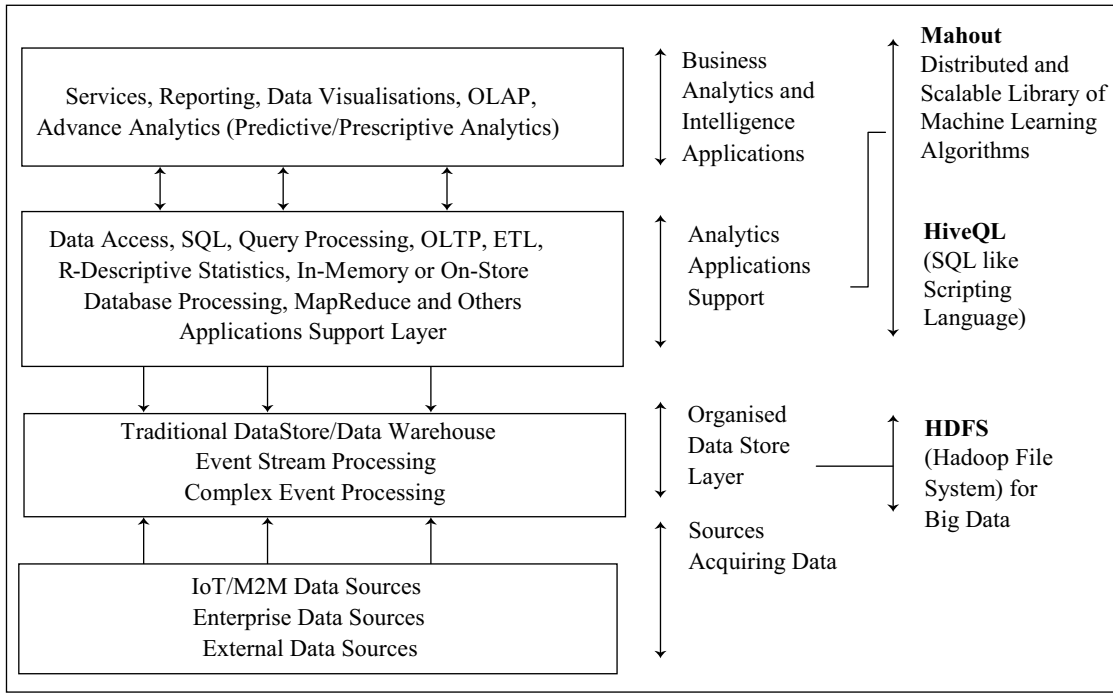


Figure 5.5 Berkeley data analytics stack architecture

Reconfirm Your Understanding

- Organised data in database or data store is used for analytics, new facts and decision taking on those facts. Analytics has three phases before deriving new facts and provide *business intelligence*—descriptive, predictive and prescriptive analytics.

- Analytics uses statistical methods to find new parameters, which add value to the data. Analytics enable building models based on selection of right data. Later the models are tested and used for services and processes.
- Analytics architecture consists of the following layers—data sources, data storage and processing, data access and query processing and data services, reporting and advanced analytics layer.
- Descriptive analytics, statistics, data mining and machine learning are analytics tools. Analytics enable the actions, reporting and generating spreadsheet, data visualisations and KPIs.
- Descriptive analytics looks at past performance. The performance is evaluated from mining the historical data.
- OLAP uses following steps—identify the dimensions, each with an attribute and hierarchy, analyse cross tabulations.
- OLAP runs faster on column format, more rows and few columns. OLTP runs faster on row format: few rows and more columns. Dual in-memory formats provide advantage of faster real-time query processing as well as analytics.
- OLAP enables viewing of analysed data up to desired granularity, viewing slices and cross-referenced tables, each viewable separately on *n*-dimensional structure faces using dicing functions.
- Predictive analytics answer the question, "What will happen?" Predictive analytics is advanced analytics. The user interprets the outputs from advanced analytics using descriptive analytics methods, such as data visualisation.
- Prescriptive analytics answers not only what is anticipated or what will happen or when it will happen, but also why it will happen based on the input from descriptive analytics and business rules.
- Event analytics use event data, for events tracking and event reporting. Event analytics generate event reports using event metric (event counts, events acted up on, event pending action, rate of new events generation) in each category of events.
- Analytics architecture consists of the following layers: Data Sources, Data Storage and Processing, Data Access and Query Processing, Data Services, Reporting and Advanced Analytics Layers.
- Berkeley Data Analytics Stack consists of data processing, data management and resource management layers.

Self-Assessment Exercise

1. List the uses of analytics. ★
2. Explain spreadsheet-based reports and data visualisation with example of a daily sales database for ACVMs. ★★
3. List the advantages of descriptive analytics of ACVMs data. ★★★
4. What does OLAP mean? ★
5. List the usages of slicing and dicing functionalities at automobile service centre in Internet of automotive components. ★★★
6. How do predictive and prescriptive analytics differ? ★★
7. Why do OLTP operations run faster in row format in-memory database? Why do OLAP operations run faster in column format in-memory database? ★★

8. What is big data? ★
9. How does big data analytics differ from structured RDMS analytics? ★
10. How are the analytics architecture layers used in automobile service centre for Internet of automotive components? ★★
11. Explain Berkeley Data Analytics Stack layer software components. ★★★
12. How does analytics lead to business intelligence? ★★

5.6 KNOWLEDGE ACQUIRING, MANAGING AND STORING PROCESSES

LO 5.5 Explain knowledge discovery, knowledge management and knowledge-management reference architecture

Data Information Knowledge Wisdom (DIKW)[8] forms a pyramid. Information is an enriched set of data values when considered in a context and that can be queried upon. Visualisation of data gives information. Spreadsheet gives information. Analytics gives information.

"Only 5% streetlights are switched on in night in Internet of Streetlights" is information (Example 1.2). "All ATMs active" is information (Example 2.3). "All ACVMs filled with chocolates of all five flavours at the moment" is information (Example 5.1). "An enterprise is showing consistent growth and profits in successive five years" is information about the functioning of an enterprise.

Information in the given context is an answer to the query or set of queries. The answer comes from processing the data and querying. For example, a balance sheet data is the enriched set of data values using the analytics. "Is the data in enterprise balance sheet showing consistent growth in preceding five years?" is querying of the balance sheet. The answer is the information obtained from the balance sheet.

Knowledge, according to an English dictionary is sharable information and understanding about a subject or context. Information that All ACVMs filled at most times gives understanding and knowledge that "Fill service is prompt in attending to the service requests". Information about the time series data of sale figures for chocolates gives understanding and knowledge that "ACVMs give good sales and profit during festive the days near the gardens." Knowledge comes from researching gives existing information. An enterprise consistent growth is as a result of installation of new machinery five years back, full production during these years and enterprise sales force performance.

IoT data sources continuously generate data, which the applications or processes acquire, organise and integrates or enriches using analytics. *Knowledge discovery tools*

[8] J. Rowley, "The wisdom hierarchy: Representations of the DIKW hierarchy", Journal of Information Science, 33(2), pp. 163-19, 2007

provide the knowledge at particular point of time as more and more data is processed and analysed. Knowledge is an important asset of an enterprise.

Knowledge Management

Knowledge management[9] (KM) is managing knowledge when the new knowledge is regularly acquired, processed and stored. Knowledge management also provisions for replacing the earlier gathered knowledge and managing the life cycle of stored knowledge. 'Fill service is prompt' is temporal knowledge. It may change at a later date. 'Enterprise consistent growth' is temporal knowledge. It may change in sixth year of operations.

A management tool role is to create, control, use, monitor and delete. A KM tool has processes for discovering, using, sharing, replacing with new, creating and managing the knowledge database and information of the enterprise.

Wisdom

Sensible and reasonable decisions are made using advanced tools which enable wise decision. *Wisdom*, according to an English dictionary is "Ability to use the experience and knowledge in order to make sensible and reasonable judgment and decisions". Judgment from the experience and knowledge that "ACVMs chain needs adaptation of loyalty point scheme" is wisdom. Judgment from the knowledge of clients of a particular bank, "Operating a free dispensary will improve heath of the clients", then they will earn more and consequently bank expects to attract bigger deposits is wisdom.

5.6.1 Knowledge-Management Reference Architecture

Figure 5.6 (a) shows a reference architecture for knowledge management. Figure 5.6(b) shows correspondences with the ITU-T reference model four layers and OSI model layers.

The lowest layer has sublayers for devices data, streaming data sources which provide input for analytics and knowledge. Databases, Business Support Systems (BSSs), Operational Support Systems (OSSs) data can also be additional inputs.

Next higher layer has data adaptation and enrichment sublayers. Adaptation and enrichment sublayers adapt the data from the lowest layer in appropriate forms, such as database, structured data and unstructured data so that it can be used for analytics and processing.

Next higher layer has processing and analytics sublayers. These sublayers are input to information access tools and knowledge discovery tools.

The highest layer has knowledge acquiring, managing, storing and knowledge life-cycle management; sublayers for managing, storing and knowledge life-cycle management. Knowledge acquires from the use of information access tools and knowledge discovery tools.

[9] https://en.wikipedia.org/wiki/Knowledge_management

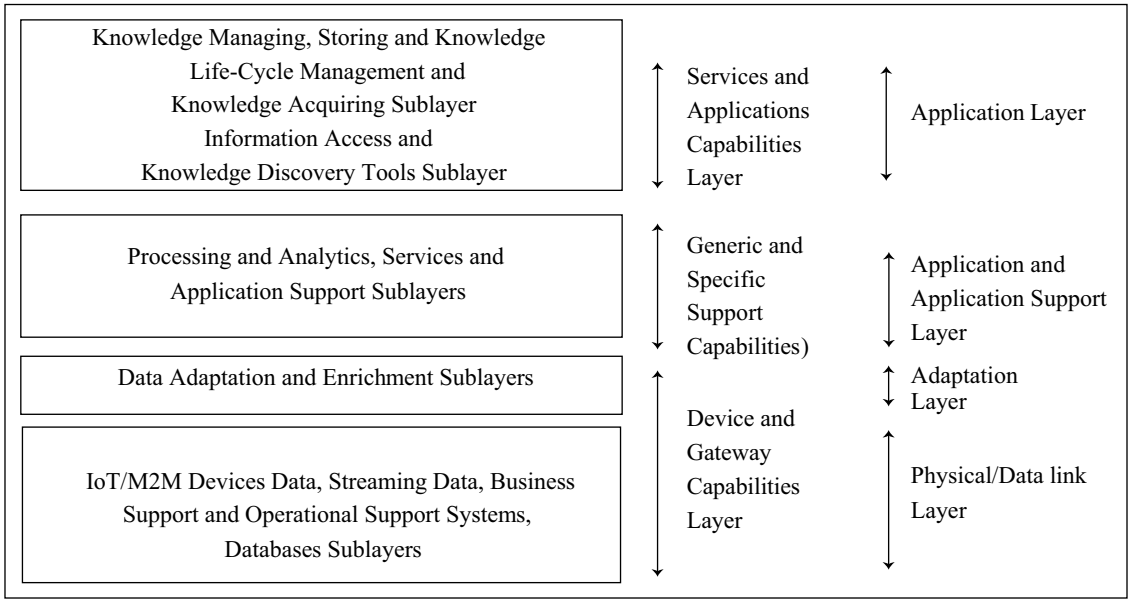


Figure 5.6 (a) A reference architecture for the knowledge management (left-hand side) and (b) Correspondence in terms of ITU-T reference model and OSI layers for IoT/M2M (middle and right-hand side)

Reconfirm Your Understanding

- Data Information Knowledge Wisdom forms a pyramid.
- Information is an *enriched set of data values* when considered in a given context that can be queried upon. Visualisation of data gives information. Spreadsheet gives the information. Analytics gives information.
- Knowledge gathers from researching up on the information. Knowledge is sharable information and understanding about a subject or context.
- Knowledge management is managing knowledge when new knowledge is acquired, processed and stored.
- Knowledge management architecture highest layer is knowledge managing, storing, knowledge life-cycle management, knowledge acquiring, information access and knowledge discovery tools.
- Wisdom is the ability to use the experience and knowledge in order to make sensible and reasonable judgments and decisions.

Self-Assessment Exercise

1. Explain Data Information Knowledge Wisdom pyramid. ★
2. Show diagrammatically, software components at the layers in reference-architecture for knowledge management of services at Internet of automotive components. ★★
3. Why does knowledge management include functions for replacing the earlier gathered knowledge, usage of knowledge discovery tool periodically and managing the life cycle of stored knowledge? Take example of ACPAMS. ★★★
4. Explain knowledge management by example of applications, services and processes for Internet of Automatic Chocolate Vending Machines, Internet of ATMs and Internet of automobile components. ★★★

Key Concepts

- Big data
- Business intelligence
- Business process
- CEP
- Data acquiring
- Data generation
- Data source types
- Data storage
- Data store
- Data visualisation
- Database
- DBMS
- Descriptive analytics
- Dicing functionalities
- Information
- In-memory database
- Knowledge
- NOSQL
- OLAP
- OLTP
- Predictive analytics
- Query
- Query processing
- RDBMS
- Relational database
- Spreadsheet
- SQL
- Transaction
- Time-series database
- Wisdom

Learning Outcomes

LO 5.1

- Data, real-time data, events or event-driven data generate from the active or passive devices and other data sources.
- A data acquisition application interacts and acquires data from the interactions. Large magnitude of data is acquired from a large number of connected devices, especially from machines in industrial plants or embedded components data from large number of automobiles or health devices in ICUs or wireless sensor networks.
- Data storage systems store the data after validation. Data store can be database, relational database at a server or cloud. Data store can be at data warehouse or Big data at a Cloud.
- Data store may consist of multiple schemas or may consist of data in only one scheme, such as a relational database. Data store at server for short reaction times, optimised performance and high security.

- Spatial storage is storage as spatial database which is optimised to store, enables querying the data objects defined in a geometric space, and which is a database for 2D and 3D objects.

LO 5.2

- A database is a collection of data. A relational database is a collection of data into multiple tables which relates to each other through special fields.
- Database Management System is a software system, which contains a set of programs specially designed for creation and management of data stored in a database.
- Database transaction is the execution of a specific set of operations on a database. Transactions can be performed on a database. Relational database transaction is the execution of interrelated instructions using relations.
- Query processing means using a process, getting the results of the query made to a database.
- Distributed database is a collection of logically interrelated, cooperating databases over a computer network. Distributed query processing means query processing operations in distributed databases on the same system or networked systems.
- SQL is language for data access control, schema creation and modifications. NOSQL stands for or Not Only SQL or No-SQL and no integration with applications that are based on SQL. It is used in cloud data store.
- Time series data means an array of numbers indexed with time (date-time or a range of date-time).

LO 5.3

- A *transaction* is a collection of operations that form a single logical unit of database. OLTP process begins as soon as data or events generate in real time. Batch transactions processing, stream transactions processing, real-time transaction processing, and complex event processing are process methods for data and events of the data sources and data stores.
- A business process consists of a series of interrelated structured activities or tasks or processes which follow a logical sequence.
- A process matrix represents series of operations and activities on a given set of inputs that perform a specific task leading to a decision point in the process.
- Business intelligence is a process that enables a business service to extract new facts and knowledge and then undertake better decisions.
- Distribution of processes reduces the complexity and communication costs, and enables faster responses and smaller processing load at the central system.
- Complex application integration integrates heterogeneous application architectures and number of processes.
- SOA is a software architecture model consisting of services, messages, operations and processes.

LO 5.4

- Organised data is used for analytics, new facts and decision taking on those facts. Analytics has three phases before deriving new facts and provide *business intelligence*: descriptive, predictive and prescriptive analytics.
- Analytics uses statistical methods and finds new parameters which add value to the data.
- Analytics architecture consists of the following layers: Data Sources, Data Storage and Processing, Data Access and Query Processing and Data Services, Reporting and Advanced Analytics Layers.

- Descriptive analytics, statistics, data mining and machine learning are analytics tools. Analytics enable actions, reporting and generating spreadsheet, data visualisations and KPIs.
- OLAP runs faster on column format, i.e. more rows and few columns.
- Predictive analytics answer the question "What will happen?" Predictive analytics is advanced analytics. The user interprets the outputs from advanced analytics using descriptive analytics methods, such as data visualisation.
- Prescriptive analytics answers not only what is anticipated or what will happen or when it will happen, but also why it will happen based on inputs from descriptive analytics and business rules.
- Event analytics use event data for event tracking and event reporting.
- Analytics architecture consists of the following layers—Data Sources, Data Storage and Processing, Data Access and Query Processing, Data Services, Reporting and Advanced Analytics Layers.

LO 5.5

- Data Information Knowledge Wisdom forms a pyramid.
- Knowledge gathers from researching existing new information. Knowledge is sharable information and understanding about a subject or context.
- Wisdom is the ability to use experience and knowledge in order to make sensible and reasonable judgment and decisions.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. An IoT application or service software components at the applications and services layer are (i) Devices data or event or message generation, (ii) Data acquiring, collection, assembling and storage, (iii) Transactions processing, (iv) IoT applications integration with services (v) Business processes, (vi) Complex Event Processing, (vii) Business intelligence, (viii) Analytics, (xi) Data analytics stack, (ix) Intelligence, (x) Knowledge discovery and (xi) Knowledge management. Which is correct? ★
   (a) All except (i) to , (vi) to (xi)
   (b) (iv) to (viii)
   (c) All except (x) and (xi)
   (d) All except (i)
2. A service means (i) a mechanism which enables the provisioning of access to one or more capabilities, (ii) has an interface for the service which provides access to capabilities, (iii) initiates on message from another service, (iv) consists of a set of related software components and their functionalities, (v) accesses server database, (v) access to each capability is consistent with the constraints and policies, (vi) has a service description, (vii) can advertise for its capabilities, and (vii) can be used with the complex applications integration. Which is correct? ★★

(a) All except (iii) to (v) and (vii)
(b) All except (iii) to (v) and (vii)
(c) All except (v)
(d) All except (vii)

3. (i) Objects in a data store model using classes which the database define, (ii) data store includes data repositories such as database, relational database, flat file, (iii) Data store includes data repositories such as spreadsheet, mail server, web server, directory services and VMware, (iv) Data store may be distributed over multiple nodes, and (v) A data store may consist of multiple schemas or may consist of data in only one scheme. Which is correct? ★★
   (a) (ii) to (v)
   (b) (ii) and (v)
   (c) (i), (ii) and (iv)
   (d) All
4. Relation database examples are (i) MySQL, (iii) PostGreSQL, (iv) Oracle database created using PL/SQL, (v) Microsoft SQL server using T-SQL, (vi) HBase, (vii) MongoDB, (viii) CouchDB, (ix) a database in which data organised into multiple tables which relates to each other through special fields, and (x) a database in which data organised into flat file table which enables systematic way for the access. Which is correct? ★
   (a) All except (x)
   (b) All except (vi), (vii), (viii) and (x)
   (c) All except (v)
   (d) (i) and (ix)
5. Server management functions are (i) Monitoring of all critical services with SMS and email notifications, (ii) the security of systems and the protection, (iii) maintaining confidentiality and privacy of data, (iv) high degree of security and integrity and effective protection of data, files and databases at the organisation, (v) protection of customer data or enterprise internal documents by attackers which includes spam, (vi) mails, unauthorized uses of the access to the server, viruses, malwares and worms, and (vii) strict documentation and audit of all activities. Which is correct? ★
   (a) All except (i), (vi) and (vii)
   (b) All
   (c) (i) to (v)
   (d) (ii) to (vii)
6. SQL is a language for (i) data querying, updating, inserting, appending and deleting the databases, (ii) data access control, schema creation and modifications, (iii) access to server (iv) querying the file, and (v) service. Which is correct? ★★
   (a) (i) and (ii)
   (b) All except (ii)
   (c) All except (iii)
   (d) (i), (ii) and (iv)
7. Decision on real-time data is fast when query processing in data has low latency due to (i) massively parallel processing, (ii) relational databases, (iii) time series database, (iv) in-memory database, and (v) columnar database. Which is correct? ★
   (a) (iii), (iv) and (v)
   (b) All except (ii)

(c) (i) to (iv)
(d) (i), (iv) and (v)

8. NOSQL is (i) used in cloud data store, (ii) used in data mining, (iii) means no database schema, (iv) no integration with applications that are based on SQL, and (v) consists of classes of non-relational data storage systems, flexible data models and multiple schemas. Which is correct? ★
   (a) All except (ii)
   (b) All except (i)
   (c) (i), (iv) and (v)
   (d) All
9. OLTP is used in (i) Internet of Streetlights, (ii) Internet of Automatic Chocolate Vending Machines, (iii) Internet of ATMs, (iv) Internet of Automotive Components for Service Centre Predictive Analytics, (v) Internet of RFIDs, (vi) Complex event processing, and (vii) in case of applications when requirements are availability, speed, concurrency and recoverability in databases for real-time data or events. ★★★
   (a) All
   (b) (iii), (iv), (vi) and (vii)
   (c) (iii) and (vii)
   (d) (iii) to (vii)
10. A layer in business intelligence and business processes architecture reference model for internet of automotive components for service centre consists of the following: ★★
    (a) Datagram transport layer security layer
    (b) Data access, SQL, query processing, R-descriptive statistics, predictive analytics layer
    (c) Data integration layer
    (d) Transactions processing layer
11. SOA (i) models the number of services and interrelationships, (ii) is a software architecture model which consists of services, messages, operations and processes, (iii) components distribute over a network or the Internet in a high-level business entity, and (iv) new business applications and applications integration architecture in an enterprise can be developed using a SOA. Which is correct? ★
    (a) All except (iii)
    (b) All
    (c) All except (iv)
    (d) (ii)
12. Descriptive analytics enable (i) actions, (ii) reporting or generating spreadsheets, (iii) statistical analysis, (iii) visualisations from different perspectives of the analysed results, and (iv) creation of key performance indicators, (v) data visualisation of slices and dices from cross reference tables, (vi) creation of in-memory row-format database, and (vii) later on predictions using predictive analytics. Which is correct? ★
    (a) (i) to (iv)
    (b) All except (ii) and (iii)
    (c) All
    (d) All except (vi)
13. Big data means (i) cloud data, (ii) data received from number of sources of data, (iii) data sets with sizes beyond the ability of commonly used software tools to acquire, manage and process data within a tolerable elapsed time, (iv) unstructured data in ★★

predefined formats, and (v) data on which NOSQL (Structured Query Language) applicable.
(a) All except (iv)
(b) (iii) to (vii)
(c) All except (i) and (vi)
(d) (iii) to (vii)

14. An Oracle Database has (i) in-memory row format option and (ii) in-memory column format option. Which is correct? ★
(a) Real-time analytics need both options
(b) Real-time analytics need option (ii)
(c) OLTP needs (ii) option
(e) OLAP needs (i) option
15. Analytics architecture and Berkeley Data Analytics Stack Architecture consist of (i) Data sources layer, (ii) Data storage and processing layer, (iii) Data access and query processing layer, and (iv) Data services, reporting and advanced analytics layer. Which is correct? ★★
(a) (i) to (iv) in analytics architecture only
(b) All except (i) and (ii)
(c) All except (i)
(d) All
16. Highest layer at knowledge management reference architecture consists of (i) knowledge managing, (ii) knowledge storing, (iii) knowledge life-cycle management, (iv) KNOWLEDGE acquiring, (v) information access and knowledge discovery tools, (vi) analytics (vii) processing and (viii) services and databases. Which is correct? ★
(a) All except (viii)
(b) All except (iii)
(c) (i) to (v)
(d) All

Short-Answer Questions

1. What do the sensor data, real-time data, periodic intervals data, event data, and event initiated data mean? Give an example of each in IoT/M2M applications. **[LO 5.1]** ★
2. What are the checks, which validate the data? **[LO 5.1]** ★★
3. What are the methods for data storage, which can be used for analytics later? **[LO 5.1]** ★★
4. How do the database and relation database differ? **[LO 5.2]** ★★★
5. What are the functions which spatial database can perform? **[LO 5.2]** ★
6. How do you select a chocolate flavour FL1 sales on festive days from transactions with the data store? **[LO 5.2]** ★★
7. Why do the database transactions follow rules of atomicity, consistency, isolation, and durability? **[LO 5.2]** ★★
8. List the performed actions during the event-stream processing and complex-event processing. **[LO 5.2]** ★★★
9. How do mathematical operations (profiles, traces, and curves), queries or database transactions on time series implement in a Time Series Database (TSDB)? **[LO 5.2]** ★★★

10. What are the data types for which CAP theorem holds true? Answer why CAP theorem states that out of three properties (consistency, availability and partitions), two are at least present for a service or process? **[LO 5.2]** ★★
11. What are the types of business processes? **[LO 5.3]** ★
12. What are the software components for descriptive analytics? **[LO 5.3]** ★★
13. When are streaming transactions and batch transactions performed? **[LO 5.3]** ★★
14. What are the operations preceding business intelligence? **[LO 5.3]** ★★★
15. List reasons for distributed databases reduced complexity, communication costs, faster responses and smaller processing load at the central system. **[LO 5.3]** ★★
16. What are the different visualisations from the OLAP? **[LO 5.4]** ★
17. Why does knowledge require replacement and needs use of knowledge discovery tools at specified periodic intervals? Take an enterprise which services automobiles using descriptive and prescriptive analytics as an example. **[LO 5.5]** ★★★

Review Questions

1. Describe data generation from IoT/M2M devices. [**LO 5.1**] ★
2. List the features of relational time series service. [**LO 5.2**] ★★
3. What is NOSQL and what are the NOSQL usages? [**LO 5.2**] ★★
4. Describe different types of transaction processing on databases, streaming data and events. [**LO 5.3**] ★★★
5. List the features of distributed business process and distributed business process system (DBPS). [**LO 5.3**] ★★
6. List OLAP functionalities. [**LO 5.4**]
7. Describe in-memory row format and column format database features and usages. [**LO 5.4**] ★
8. Explain using an example of Data Information Knowledge Wisdom pyramid from the sales data acquiring, organising and analytics. [**LO 5.5**] ★★★

Practice Exercises

1. List data types which communicate from RFIDs in Internet of RFIDs. **[LO 5.1]** ★
2. List ways in which sensor nodes configures in Internet of Streetlights. **[LO 5.1]** ★★
3. List the SQL functionalities. **[LO 5.2]** ★★
4. Which relational database tables in Example 5.2 for Internet of Automobile Components are used in ACPAMS Centre? **[LO 5.2]** ★★★
5. Consider Example 5.1 and list the process matrix and interleaved decision points. ★★
6. List the usages of Internet of Automatic Chocolate Vending Machines after analytics using three relational database tables.(Example 5.3) **[LO 5.3]** ★★
7. List the business intelligence obtained in Internet of ATMs. **[LO 5.3]** ★
8. Workout the n-dimensional structure of 2-n cross-referenced tables in Internet of Automatic Chocolate Vending Machines. **[LO 5.4]** ★★★
9. Assume n devices each in ICU from ten patients communicate data and events onto the Internet. List the series of activities and software tools at each layer required for acquiring, storing and analytics. Draw the analytics reference architecture. **[LO 5.5]** ★★★
10. List the processes for knowledge creation. **[LO 5.5]** ★

6

Data Collection, Storage and Computing Using a Cloud Platform

Learning Objectives

LO 6.1 Outline cloud computing paradigm for data collection, storage and computing services

LO 6.2 Describe cloud computing service models in a software architectural concept, everything as a service (XAAS)

LO 6.3 Explain the usage of cloud platforms for IoT applications and services with examples of Xively (Pachube/COSM) and Nimbits

Recall from Previous Chapters

The concepts learnt in previous chapters can be summed up as follows—large magnitude of data is generated from large number of devices, especially from machines in industrial plants, embedded components data from large number of automobiles, health devices in ICUs or wireless sensor networks, and so on. The applications acquire the data, collect and store in servers, data centres or data warehouses. The data is organised in a relational database or other structured or unstructured formats. IoT applications, services and processes use data for computations, transactions, OLTP, OLAP, analytics, business processes, business intelligence and knowledge discovery.

6.1 INTRODUCTION

A few conventional methods for data collection and storage are as follows:

IoT data collection and storage methods

- Saving devices' data at a local server for the device nodes
- Communicating and saving the devices' data in the files locally on removable media, such as micro SD cards and computer hard disks
- Communicating and saving the data and results of computations in a dedicated data store or coordinating node locally
- Communicating and saving data at a local node, which is a part of a distributed DBMS (Section 5.3.1)
- Communicating and saving at a remote node in the distributed DBMS
- Communicating on the Internet and saving at a data store in a web or enterprise server
- Communicating on the Internet and saving at data centre for an enterprise

Cloud is a new generation method for data collection, storage and computing. Section 6.2 describes cloud computing paradigm for data collection, storage, computing and services. Section 6.3 describes cloud-computing service models in a software architectural concept, 'everything as a service'. Section 6.4 describes IoT specific cloud-based services, Xively, Nimbits. Section 12.2.4 describes platforms such as AWS IoT, Cisco IoT, IOx and Fog, IBM IoT Foundation, TCS Connected Universe (TCS CUP).

6.2 CLOUD COMPUTING PARADIGM FOR DATA COLLECTION, STORAGE AND COMPUTING

LO 6.1 Outline Cloud computing paradigm for data collection, storage and computing services

Different methods of data collection, storage and computing are shown in Figure 6.1. The figure shows (i) Devices or sensor networks data collection at the device web server, (ii) Local files, (iii) Dedicated data store at coordinating node, (iii) Local node in a distributed DBMS, (iv) Internet-connected data centre, (v) Internet-connected server, (vi) Internet-connected distributed DBMS nodes, and (vii) Cloud infrastructure and services.

Cloud computing paradigm is a great evolution in Information and Communications Technology (ICT). The new paradigm uses XAAS at the Internet connected clouds for collection, storage and computing.

Following are the key terms and their meanings, which need to be understood before learning about the cloud computing platform.

Resource refers to one that can be read (used), written (created of changed) or executed (processed). A path specification is also a resource. The resource is atomic (not-further divisible) information, which is usable during computations. A resource may have multiple instances or just a single instance. The data point, pointer, data, object, data store or method can also be a resource.

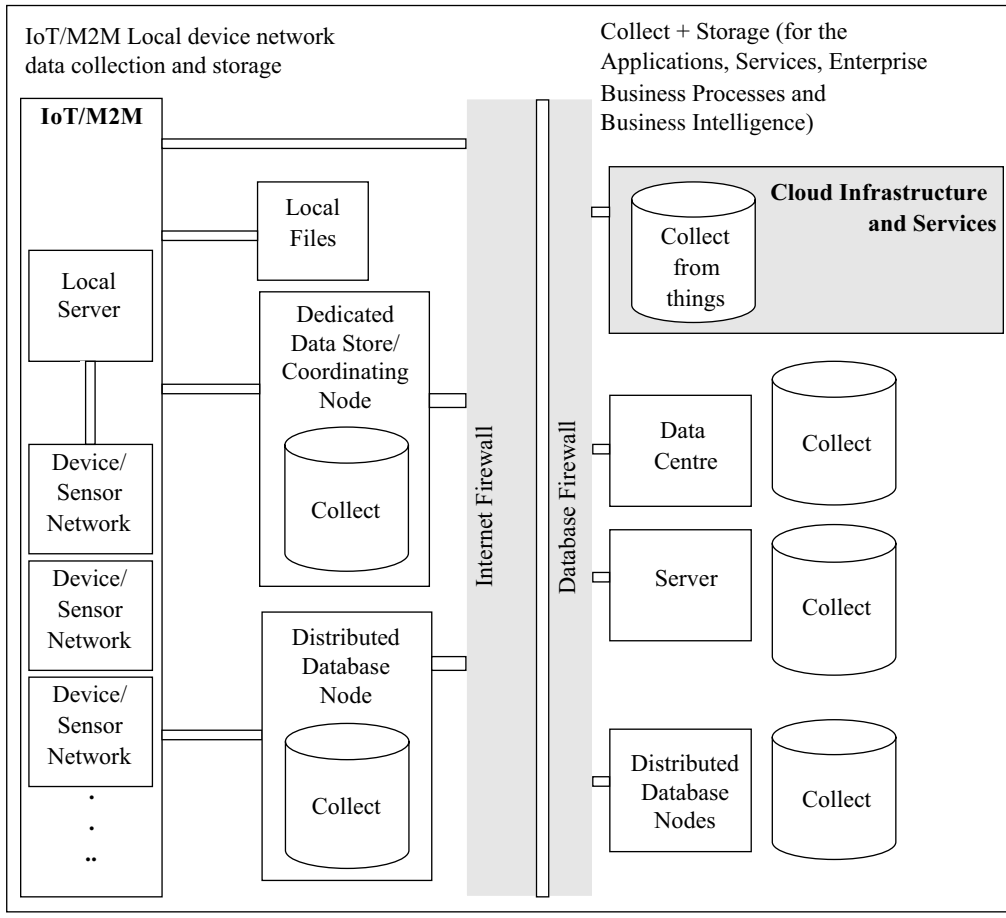


Figure 6.1 Devices or sensors network data collection at a device local-server, local files, dedicated data store, at a coordinating node, a local node of a distributed DBMS, Internet-connected server of data centre, server or distributed database nodes or a cloud infrastructure

System resource refers to an operating system (OS), memory, network, server, software or application.

Environment refers to an environment for programming, program execution or both. For example, *cloud9* online provides an open programming environment for BeagleBone board for the development of IoT devices; *Windows* environment for programming and execution of applications; *Google App Engine* environment for creation and execution of web applications in Python or Java.

Platform denotes the basic hardware, operating system and network, and is used for software applications or services over which programs can be run or developed.[1]

[1] Based on definition at https://www.techopedia.com/definition/3411/platform

A platform may provide a browser and APIs which can be used as a base on which other applications can be run or developed.

Edge computing is a type of computing that pushes the frontier of computing applications, data and services away from centralised nodes to IoT data generating nodes, that means at logical extremes of the network.[2] IoT device nodes are pushed by events, triggers, alerts, messages and data is collected for enrichment, storage and computations from the remote centralised database nodes. Pushing the computations from centralised nodes enables the usage of resources at device nodes, which could be a requirement in case of low power lossy networks. The processing can also be classified as edge computing at local cloud, grid or mesh computing. The nodes may be mobile or of a wireless sensor network or cooperative distributed in peer-to-peer and ad-hoc networks.

Distributed computing refers to computing and usage of resources which are distributed at multiple computing environments over the Internet. The resources are logically-related, which means communicating among themselves using message passing and transparency concepts, and are cooperating with each other, movable without affecting the computations and can be considered as one computing system (location independent).

Service is a software which provides the capabilities and logically grouped and encapsulated functionalities. A service is called by an application for utilising the capabilities. A service has a description and discovery methods, such as advertisement for direct use or through a service broker. The service binds to Service Level Agreement (SLA) between service (provider end point) and application (end point). One service can also use another service.

Web Service, according to the W3C definition, is an application identified by a URI, described and discovered using the XML based Web-Service Description Language (WSDL). A web service interacts with other services and applications using XML messages and exchanges the objects using Internet protocols.

Service-oriented architecture consists of components which are implemented as independent services which can be dynamically bonded and orchestrated, and which possess loosely coupled configurations, while the communication between them uses messages. Orchestrating means a process which predefines an order of calling the services (in sequences and in parallel) and the data and message exchanges.

Web computing refers to computing using resources at computing environment of web server(s) or web services over the Internet.

Grid computing refers to computing using the pooled interconnected grid of computing resources and environments in place of web servers.

Utility computing refers to computing using focus on service levels with optimum amount of resources allotted when required and takes the help of pooled resources and environments for hosting applications. The applications utilise the services.

[2] https://en.wikipedia.org/wiki/Edge_computing

Cloud computing refers to computing using a collection of services available over the Internet that deliver computational functionality on the infrastructure of a service provider for connected systems and enables distributed grid and utility computing.

Key Performance Indicator (KPI) refers to a set of values, usually consisting of one or more raw monitored values including minimum, average and maximum values specifying the scale. A service is expected to be fast, reliable and secure. The KPIs monitor the fulfillment of these objectives. For example, a set of values can relate to Quality of Service (QoS) characteristics, such as bandwidth availability, data backup capability, peak and average workload handling capacity, ability to handle defined volume of demand at different times of the day, and the ability to deliver defined total volume of service. A cloud service should be able to fulfill the defined minimum, average and maximum KPI values agreed in the SLA.

Localisation means cloud computing content usage is monitored by determining localisation of the QoS level and KPIs.

Seamless cloud computing means during computing the content usages and computations continue without any break when the service usage moves to a location with similar QoS level and KPIs. For example, continue using same cloud platform when developer of software shifts.

Elasticity denotes that an application can deploy local as well as remote applications or services and release them after the application usage. The user incurs the costs as per the usages and KPIs.

Measurability (of a resource or service) is something which can be measured for controlling or monitoring and enables report of the delivery of resource or service.

Homogeneity of different computing nodes in a cluster or clusters refers to integration with the kernel providing the automatic migration of the processes from one to other homogeneous nodes. System software on each computing node should ensure same storage representation and same results of processing.[3]

Resilient computing refers to the ability of offering and maintaining the accepted QoS and KPIs in presence of the identified challenges, defined and appropriate resilience metrics, and protecting the service.[4] The challenges may be small to big, such as misconfiguration of a computing node in a network to natural disasters. An English Cambridge dictionary gives the meaning of resilience as the power or ability to return to the original form, position, etc., after being bent, compressed or stretched.

Scalability in cloud services refers to the ability using which an application can deploy smaller local resources as well as remotely distributed servers and resources, and then increase or decrease the usage, while incurring the cost as per the usage on increasing scales.

[3] http://www.netlib.org/utk/papers/practical-hetro/node3.html
[4] https://en.wikipedia.org/wiki/Resilience_(network)

Maintainability in cloud services refers to the storage, applications, computing infrastructure, services, data centres and servers maintenances which are responsibilities of the remotely connected cloud services with no costs to the user.

XAAS is a software architectural concept that enables deployment and development of applications, and offers services using *web* and *SOA*. A computing paradigm is to integrate complex applications and services (Section 5.3.5) and use XAAS concept for deploying a cloud platform.[5]

Multitenant cloud model refers to accessibility to a cloud platform and computing environment by multiple users who pay as per the agreed QoS and KPIs, which are defined at separate SLAs with each user. Resource pooling is done by the users but each uses pays separately.

The following subsections describe the cloud computing paradigm and deployment models.

6.2.1 Cloud Computing Paradigm

Cloud computing means a collection of services available over the Internet. Cloud delivers the computational functionality. Cloud computing deploys infrastructure of a cloud-service provider. The infrastructure deploys on a utility or grid computing or web-services environment that includes network, system, grid of computers or servers or data centres.

Just as we—users of electricity—do not need to know about the source and underlying infrastructure for electricity supply service, similarly, a user of computing service or application need not know how the infrastructure deploys or the details of the computing environment. Just as the user does not need to know Intel processor inside a computer, similarly, the user uses the data, computing and intelligence in the cloud, as part of the services. Similarly, the services are used as a utility at the cloud.

Cloud Platform Services

Cloud platform offers the following:

- Infrastructure for large data storage of devices, RFIDs, industrial plant machines, automobiles and device networks
- Computing capabilities, such as analytics, IDE (Integrated Development Environment)
- Collaborative computing and data store sharing

Cloud Platform Usages

Cloud platform usages are for connecting devices, data, APIs, applications and services, persons, enterprises, businesses and XAAS.

[5] Reader can refer to standard books for details of Cloud Computing and SOA, such as *Mastering Cloud Computing* by Rajkumar Buyya, Christian Vecchiola and S. Thamarai Selvi from McGraw-Hill Education (2013).

The following Equation 6.1 describes a simple conceptual framework of the Internet Cloud:

Internet Cloud + Clients = User applications and services with 'no boundaries and no walls' **… 6.1**

An application or service executes on a platform which includes the operating system (OS), hardware and network. Multiple applications may initially be designed to run on diversified platforms (OSs, hardware and networks). Applications and services need to integrate them on a common platform and running environment.

Cloud storage and computing environment offers a *virtualised environment*, which refers to a running environment made to appear as one to all applications and services, but in fact physically two or more running environments and platforms may be present.

Virtualisation

Virtualisation enables provisioning for storage, network functions, server and desktop in execution environment of multiplatforms and servers

A characteristic of virtualised environment is that it enables applications and services to execute in an independent execution environment (heterogeneous computing environment). Each one of them stores and executes in isolation on the same platform, though in fact, it may actually execute or access to a set of data centres or servers or distributed services and computing systems. The applications or services which are hosted remotely and are accessible using the Internet can easily be deployed at a user application or service in a virtualised environment, provided the Internet or other communications are present.

Applications need not be aware of the platform, just Internet connectivity to the platform, called cloud platform, is required. The storage is called cloud storage. The computing is called cloud computing. The services are called cloud services in line with the web services which host on web servers.

Virtualisation of storage means user application or service accesses physical storage using abstract database interface or file system or logical drive or disk drive, though in fact storage may be accessible using multiple interfaces or servers. For example, Apple iCloud offers storage to a user or user group that enables the sharing of albums, music, videos, data store, editing files and collaboration among the user group members.

Network Function Virtualisation (NFV) means a user application or service accesses the resources appearing as just one network, though the network access to the resources may be through multiple resources and networks.

Virtualisation of server means user application accesses not only one server but in fact accesses multiple servers.

Virtualised desktop means the user application can change and deploy multiple desktops, though the access by the user is through their own computer platform (OS) that in fact may be through multiple OSs and platforms or remote computers.

Cloud Computing Features and Advantages

Essential features of cloud storage and computing are:

- On demand self-service to users for the provision of storage, computing servers, software delivery and server time
- Resource pooling in multi-tenant model
- Broad network accessibility in virtualised environment to heterogeneous users, clients, systems and devices
- Elasticity
- Massive scale availability
- Scalability
- Maintainability
- Homogeneity
- Virtualisation
- Interconnectivity platform with virtualised environment for enterprises and provisioning of in-between Service Level Agreements (SLAs)
- Resilient computing
- Advanced security
- Low cost

Cloud Computing Concerns

Concerns in usage of cloud computing are:

- Requirement of a constant high-speed Internet connection
- Limitations of the services available
- Possible data loss
- Non delivery as per defined SLA specified performance
- Different APIs and protocols used at different clouds
- Security in multi-tenant environment needs high trust and low risks
- Loss of users' control

6.2.2 Cloud Deployment Models

Following are the four cloud deployment models:

1. ***Public cloud:*** This model is provisioned by educational institutions, industries, government institutions or businesses or enterprises and is open for public use.
2. ***Private cloud:*** This model is exclusive for use by institutions, industries, businesses or enterprises and is meant for private use in the organisation by the employees and associated users only.
3. ***Community cloud:*** This model is exclusive for use by a community formed by institutions, industries, businesses or enterprises, and for use within the community organisation, employees and associated users. The community specifies security and compliance considerations.

A cloud deployment model may be public, private, community or hybrid

4. ***Hybrid cloud:*** A set of two or more distinct clouds (public, private or community) with distinct data stores and applications that bind between them to deploy the proprietary or standard technology.

Cloud platform architecture is a virtualised network architecture consisting of a cluster of connected servers over the data centres and Service Level Agreements (SLAs) between them. A cloud platform controls and manages resources, and dynamically provisions the networks, servers and storage. Cloud platform applications and network services are utility, grid and distributed services. Examples of cloud platforms are Amazon EC2, Microsoft Azure, Google App Engine, Xively, Nimbits, AWS IoT, CISCO IoT, IOx and Fog, IBM IoT Foundation, TCS Connected Universe Platform.

Example 6.1

Problem

Recall Example 5.1 and list the features of the Microsoft cloud platform.

Solution

The Microsoft cloud platform consists of: (i) Microsoft finished services and solutions; (ii) Microsoft building block services (Dynamic CRM Online, Sharepoint Online; Exchange Online, dot Net, SQL, ASP dot Net, Office Live and Windows Live Services); (iii) Cloud infrastructure services (Windows Azure for compute, storage and resource management services); and (iv) Global foundation services (Hardware, networking, deployment and operations services). The platform includes development tools (Visual Studio, Windows server, Visual C++, Visual Basic, visual C# and dot Net).

Reconfirm Your Understanding

- Computations need resources, computing environment and platform.
- Applications and services may require multiple platforms and environments, and multisource resources.
- The computing paradigms which are deployed for applications and services, are edge computing, distributed computing, grid computing, utility computing and cloud computing.
- A cloud service connects devices, data, APIs, applications, services, processes, persons, enterprises, businesses and everything as a service.
- Cloud services offer a virtualised environment, needs QoSs and defined KPIs in the SLAs with the users.
- Features of cloud services are *on demand self-service*, *resources pooling*, *network broad accessibility*, *elasticity* and *measurability,* massive scale, scalability, maintainability, homogeneity, virtualisation, interconnectivity platform, and resilient computing.
- Cloud services need to provide advanced security for storage, applications, computing infrastructure, services, data centres and servers at remotely connected cloud services independently at low cost.
- Cloud services utilisation requires high-speed Internet connectivity.
- Cloud computing deployment models are *public*, *private*, *community* and *hybrid clouds*.
- Amazon EC2, Microsoft Azure, Google App Engine, Xively and Nimbits are the examples of cloud services.

Self-Assessment Exercise

1. List the merits of each method for a data store shown diagrammatically in Figure 6.1. ★★
2. List five key features of cloud computing. ★
3. List the usages of virtualisation functions for data store, networks and servers. ★★★
4. Compare the relative merits of four cloud-deployment models. ★
5. What are the concerns in using cloud computing for applications? ★★
6. How do you define cloud computing? How does it differ from distributed computing? ★★★

6.3 EVERYTHING AS A SERVICE AND CLOUD SERVICE MODELS

LO 6.2 Describe cloud computing service models in a software architectural concept, everything as a service (XAAS)

Cloud connects the devices, data, applications, services, persons and business. Cloud services can be considered as distribution service—a service for linking the resources (computing functions, data store, processing functions, networks, servers and applications) and for provision of coordinating between the resources. Figure 6.2 shows four cloud service models and examples.

Cloud computing can be considered by a simple equation:

Cloud Computing = SaaS + Paas + IaaS + DaaS … 6.2

Four service models: Software, Platform, Infrastructure and Data as the Services

SaaS means Software as a Service. The software is made available to an application or service on demand. SaaS is a service model where the applications or services deploy and host at the cloud, and are made available through the Internet on demand by the service user. The software control, maintenance, updation to new version and infrastructure, platform and resource requirements are the responsibilities of the cloud service provider.

PaaS means Platform as a Service. The platform is made available to a developer of an application on demand. PaaS is a service model where the applications and services develop and execute using the platform (for computing, data store and distribution services) which is made available through the Internet on demand for the developer of the applications. The platform, network, resources, maintenance, updation and security as per the developers' requirements are the responsibilities of the cloud service provider.

IaaS means Infrastructure as a Service. The infrastructure (data stores, servers, data centres and network) is made available to a user or developer of application on demand. Developer

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

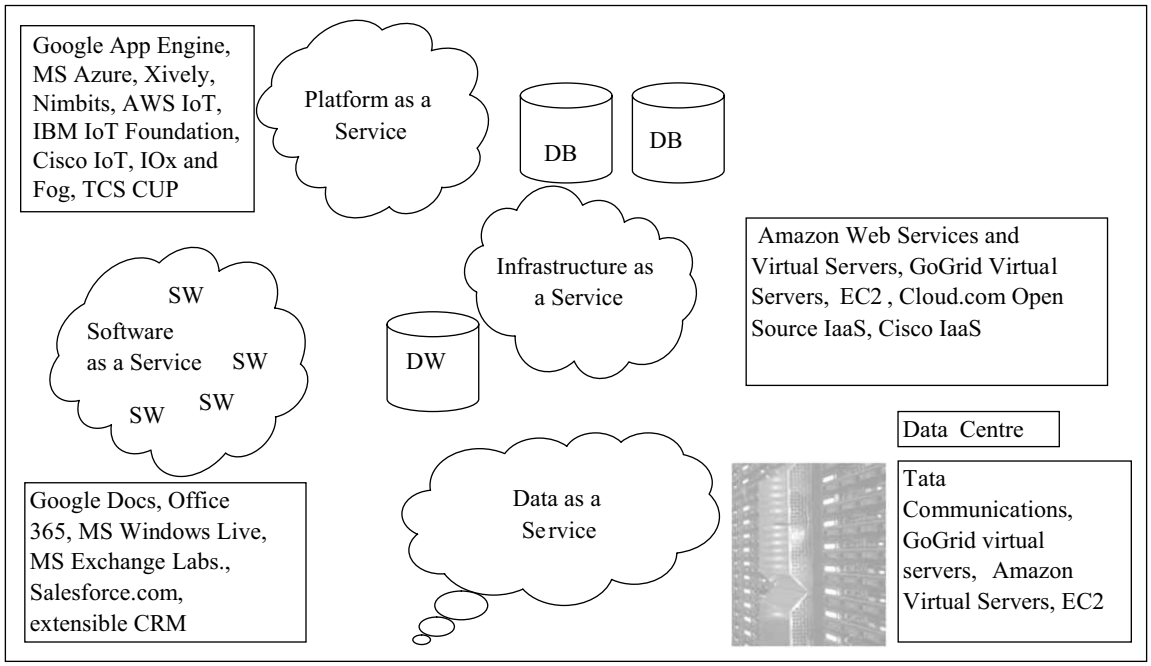


Figure 6.2 PaaS, SaaS, IaaS and DaaS Cloud Service model

[DW—Data Warehouses; DB—Databases; EC2—Elastic Computing Cloud; SW—Software; MS—Microsoft; CRM—Customer Customer Relations Management]

installs the OS image, data store and application and controls them at the infrastructure. IaaS is a service model where the applications develop or use the infrastructure which is made available through the Internet on demand on rent (pay as per use in multi-tenancy model) by a developer or user. IaaS computing systems, network and security are the responsibilities of the cloud service provider.

DaaS means Data as a Service. Data at a data centre is made available to a user or developer of application on demand. DaaS is a service model where the data store or data warehouse is made available through the Internet on demand on rent (pay as per use in multi tenancy model) to an enterprise. The data centre management, 24×7 power, control, network, maintenance, scale up, data replicating and mirror nodes and systems as well as physical security are the responsibilities of the data centre service provider.

Example 6.2

Problem

Give examples of Saas, PaaS, IaaS, DaaS service models for cloud computing.

Solution

- **SaaS** Applications of Google Docs for online office, MS Windows Live for online office applications, MS Exchange Labs, TCS iON (Integrated IT-as-a-Service), Salesforce.com for extensible Customer Relations Management (CRM) system.

- **PaaS** SuiteFlex for business process development NetSuite tools, MS Azure for Windows applications programming and execution environment, server platforms of EC2 and GoGrid, application platforms of Force Com, Google App engine for scalable execution environment for web applications, and TCS platform BPO solutions, Xively, Nimbits, AWS IoT, IBM IoT Foundation, Cisco IoT, IOx and Fog, TCS CUP.
- **IaaS** infrastructure services—Amazon Virtual Servers, GoGrid virtual servers, Elastic Computing Cloud (EC2), Cloud.com open source IaaS, TCS Transformation Solutions, Cisco IaaS, and IBM BlueCloud shared infrastructure service that automate fluctuating demands for the IT resources.
- **DaaS** Data storage platforms of Tata Communications 10 X, Apple and Cisco for the DaaS.

Reconfirm Your Understanding

- A computing paradigm is *Everything as a Service*. It means each of the software, application, infrastructure, platform and computing environment is considered as a service.
- Service-oriented architecture binds the services through the SLAs.
- Four service models for the cloud services are: Software, Platform, Infrastructure and Data as Services (SaaS, PaaS, IaaS and DaaS).
- A number of organisations, such as CISCO, Oracle, IBM, Google, Amazon, Microsoft, TCS, and Tata Communications, offer the cloud PaaS.

Self-Assessment Exercise

1. Recall Examples 2.3 and 5.4 for Internet of ATMs. List the applications and corresponding cloud service models required for Internet of ATMs. ★
2. List the service models deployed for the applications in IBM BlueCloud. ★★
3. Why is an SLA necessary to bind the services at the cloud? ★
4. How will you deploy the cloud service model for Waste Container Management, Business Intelligence and Business Processes in Internet of waste containers? ★★★

6.4 IoT CLOUD-BASED SERVICES USING THE XIVELY, NIMBITS AND OTHER PLATFORMS

LO 6.3 Explain the usages of cloud platforms for IOT applications and services with the examples of Xively (Pachube/COSM) and Nimbits

A user is an application or service. The user obtains responses or feeds from the application or service. An IoT cloud-based service provides for data collection, data points, messages and calculation objects. The service also provisions for the generation and communication of alerts, triggers and feeds to the user. A server can be deployed at the edges (device nodes) which communicates the feeds to the cloud service.

Cloud service deploys a server instance at the IoT sensor network

A data point generates a new data each time a new sensed value (data) is recorded. A feed means a set of data points or objects or data streams or messages which generate and communicate after application of rules for filter, calculation, compaction, fusion, compression, analysis or aggregation. The feed may also be for the alerts on the programmed triggers or alarms. The feed can be data stream generating and communicating at the present time intervals. The feed is for an application or service. Feeds means required data instances at predefined intervals for an application or service using the push/subscribe or any other mode.

Example 6.3

Problem

A server provisions for generation and communication of data points, objects, streams, alerts, triggers and feeds applying the rules for filtering and calculations. Consider streetlights Internet Example 1.2. Assume that each streetlight device node generates the following data: (i) *streetlightID*, (ii) *ambientLightCondition* aLC (= dark or daylight), (iii) *trafficDensity* tD (= VehiclesPassingPerSec), (iv) *functionality func (*= streetlight *functional* or *nonFunctional*) which streams to the streetlights group server locally.

Assume that a streetServer provides a local control and feeds the data streams to IoTStreetLightsServer (ILS) at a cloud. Three applications (services), central controller, Maintenance Service and Traffic_Lights_Control_Service connect to the cloud for feeds of messages and alerts from an ILS hosted at the cloud. Suggest the data points, datastreams, triggers and feeds for the ILS at the cloud.

Solution

DataPoints or each streetlight are (i) aLC (= 1 or 0), (ii) trafficPresence (= 1 or 0), (iii) tD (iv) funct (= 1 or 0) and (v) *activate*.

The streetServer controls locally the deactivation of the individual lights when aLC = daylight and when tD = 0 even when a trigger *activate* (Central controller activates the streetlight yes or no) is received from the controller. This is an energy-saving measure. The server receives in the feeds trigger *activate* for each *streetlightID* from the controller. The server feeds the controller datastreams for the (i) streetID, streetServerID and streetTrafficDensity messages and (ii) *streetlightID*. A functionality alert feeds when the func = 0 and that means the streetlight needs maintenance.

Central controller receives the feeds from the clouds. The controller generates the following data points, (i) *streetlightID* (ii) *activate* (Central Controller command activate the streetlight yes or no) (iv) *tD* (v) *func* .

The controller generates the feeds as follows:

1. feed_ID, streetServerID, streetlightID and activate for the streetServer
2. feed_ID, streetlightID and functionality Alert for Maintenance Service.
3. feed_ID, streetServerID and streetTrafficDensity datastream for Traffic_Lights_Control_Service.

Following subsections describe the Xively, Nimbits and other cloud platforms as a service to the users.

6.4.1 IoT Cloud-based Data Collection, Storage, Computing using Xively

Pachube is a platform for data capture in real-time over the Internet. Cosm is a changed domain name, where using a concept of console, one can monitor the feeds. Xively is the latest domain name. Xively is an open source platform for Arduino which is an open-source prototyping platform that provides connectivity with web deploying Internet.

Xively is a commercial PaaS for the IoT/M2M.[6] It is used as a data aggregator and data mining website often integrated into the Web of Things. Xively is an IoT PaaS for services and business services. The platform supports the REST, WebSockets and MQTT protocols and connects the devices to Xively Cloud Services. There are native SDKs for Android, Arduino, ARM mbed, Java, PHP, Ruby and Python languages. Developers can use the workflow of prototyping, deployment and management through the tools provided by Xively.[3]

Xively PaaS services offers the following features:

- It enables services, business services platform which connects the products, including collaboration products, Rescue, Boldchat, join.me, and operations to the Internet.
- Data collection in real-time over the Internet.
- Data visualisation for data of connected sensors to IoT devices.
- Graphical plots of collected data.
- It generates alerts.
- Access to historical data.
- It supports Java, Python and Ruby, and Android platform.
- It generates feeds which can be real-world objects of own or others.
- It supports the ARM mBed™-based, Arduino-based and other hardware-platform-based IoT devices, and HTTP-based APIs which are easy to implement on device hardware acting as clients to Xively web services, and connect to the web service and send data.
- It supports REST.

A user creates an account with Xively when deploying Xively APIs for the data collection and other functions. An API key from *my settings* is necessarily copied.

Xively APIs enable interface with Python, HTML5, HTML5 server, tornado, webSocket, webSocket Server and WebSockets and interface with an RPC (Remote Procedure Call).

Devices get an online presence. For example, an Arduino climate logging client that can be accessed via a browser or mobile using Xively. Arduino is an open-source prototyping platform which is based on ATmega microcontroller for embedded applications and IoT/M2M. Another platform is mBed™. It is based on ARM Cortex-microprocessors. The mBed™ is also for embedded applications and IoT/M2M (Chapter 8). Xively is an open source platform that enables IoT devices or sensors network to connect the sensor data to the web.[7,8] Xively provides services for logging, sharing and displaying sensor data of all kinds using an HTTP based API.

[6] https://dzone.com/articles/how-to-use-xively-platform-in-iot-project
[7] http://blackstufflabs.com/2011/10/08/ arduino-pachubes-api-v2/?lang=en
[8] https://developer.mbed.org/cookbook/Pachube) which is easy to implement with mbed

Xively is based on the concept of users, feeds, data streams, data points and triggers. A feed is typically a single location (e.g. a house), and data streams are of individual sensors associated with that location (for example, ambient lights, temperatures, power consumption).

Pull or Push (Automatic or Manual Feed) Methods for IoT Devices Data

Xively provides two modes for data capture, viz. a pull method (automatic feed type) where data is collected from an http server, and a push method (manual feed type) where data is written to Xively using an http client.

Data Formats and Structures

Number of data formats and structures enable interaction, data collection and services with Xively. The support exists for JSON (Section 3.2.3), XML (Section 3.3.3) and CSV (a tabular, spreadsheet, excel, data numbers and text with a comma-separated values in file).

Private and Public Data Access

A free account supports up to 10 sensor feeds updated in near real time and the data is stored for up to 3 months. Interactive graphs provided by the service can be embedded onto the mobile giving the application access to data anywhere. The application can even use other user's data feeds as inputs.

Data streams, Data points and Triggers

Xively enables data streams, data points and triggers. Data stream means continuous sensed data flow over the Internet. Data points mean data values, whereas trigger means action on a state change. For example, ambient light sensing below a threshold value near a group of streetlights (Example 1.2) or data point reaching at threshold, maximum or minimum or set point.

Example 6.4 shows create, read, update and delete data stream functions and usage of data points and triggers using Xively APIs V2 API.[9]

Example 6.4

Problem

How does Xively APIs V2 create, read, update and delete data stream? How do the APIs use data points? How do the APIs use triggers?

Solution

Firstly the API is registered. Then a procedure is followed for data stream, data points and triggers.

Data Streams

Xively v2 API enables the following data stream functionalities:

1. Create new datastream using POST /v2/feeds/<feed_id>/ datastreams/ <datastream_id>

[9] http://api.pachube.com/v2/

2. Read a datastream using GET /v2/feeds/<feed_id>/datastreams/ <datastream_id>
3. Update a datastream using PUT /v2/feeds/<feed_id>/datastreams/ <datastream_id>
4. Delete a datastream using DELETE /v2/feeds/<feed_id>/datastreams/ <datastream_id>

Each function uses

1. *@param stream_id Stream ID text.
2. *@param value value.
3. * @return Return code from a web server.

Data Points

Xively v2 API enables the following data point functionalities:

1. Create new datapoints using POST /v2/feeds/<feed_id>/datastreams/ <datastream_id>/datapoints,
2. Read datapoints using GET / v2/feeds/<feed_id>/ datastreams/ <datastream_id>/ datapoints/<timestamp>
3. Update datapoints using PUT /v2/feeds/<feed_id>/datastreams/ <datastream_id>/ datapoints/<timestamp>
4. Delete datapoints using DELETE /v2/feeds/<feed_id>/datastreams/<datastream_id>

Triggers

Xively v2 API (http://api.pachube.com/v2/) enables the following trigger functionalities:

1. Creating new triggers using POST /v2/triggers
2. Listing of all available triggers using GET /v2/triggers
3. Reading a trigger using GET /v2/triggers/<trigger_id>
4. Updating a trigger using PUT /v2/triggers/<trigger_id>
5. Deleting a user using DELETE /v2/triggers/<trigger_id>

Triggers can be a text/number, for example, Triggers_Max_Min. Trigger_id can be text/number, such temperatureMax12345.

Creating and Managing Feeds

Sensors or IoT devices network, such as a group of streetlights (Example 1.2), can feed (also called environment) data to Xively over the Internet. Individual devices can send the data streams for the sensed parameters to Xively over Internet.

Visualising Data

Xively is a platform which captures data over the Internet in real time and provides graphing, alerts and historical data access. Xively enables visualising data of feeds and data streams. Xively allows the manual as well as automatic feeds.

Following Example 6.5 shows the usage of Xively v2 API.

Example 6.5

Problem

How do the feeds create, list, read, update and delete using the Xively API V2? How do the users create, list, read, update and delete using the Xively v2 API?

Solution

Xively v2 API enables the following feed functionalities:

1. Creating new feed using POST /v2/feeds.
2. Listing of all available feeds using GET /v2/feeds
3. Reading feed using GET /v2/feeds/<feed_id>
4. Update a feed using PUT /v2/feeds/<feed_id>
5. Delete a feed: DELETE /v2/feeds/<feed_id>

Feed_id can be a text/number, for example, streetlights_group12345. Feeds can be text/number, such street_lightsFeed.

Xively v2 API (http://api.pachube.com/v2/) enables the following user functionalities:

1. Creating new users using POST /v2/users
2. Listing of all available users using GET /v2/users
3. Reading user using GET /v2/users/<user_id>
4. Update a user using PUT /v2/users/<user_id>
5. Delete a user using DELETE /v2/users/<user_id>

Users can be for example, Controllers_Monitors. User_id can be text or number, such controller12345.

6.4.2 IoT Cloud-based Data Collection, Storage and Computing Services Using Nimbits

Nimbits enables IoT on an open source distributed cloud. Nimbits cloud PaaS deploys an instance of Nimbits Server at the device nodes. Nimbits functions as an M2M system data store, data collector and logger with access to historical data. Nimbits architecture is a cloud-based Google App Engine. Nimbits server is a class hierarchy com.nimbits.server.system.ServerInfo of java.lang.Object. Nimbits PaaS services offer the following features:

- Edge computing locally on embedded systems, built up of local applications. It runs the rules and pushes important data up to the cloud running when connected over Internet and an instance of Nimbits Server hosts at the device nodes which is then enabled.
- It supports multiple programming languages, including Arduino, new Arduino library, push functions from Arduino cloud, JavaScript, HTML or the Nimbits.io Java library.
- Nimbits server functions as a backend platform. Nimbits data point can relay data between the software systems, or hardware devices such as Arduino, using the cloud as a backend.
- An open source Java library called nimbits.io enables easy development of JAVA, web and Android solutions (Nimbits data, alerts, messages on mobile).
- It provides a rule engine for connecting sensors, persons and software to the cloud and one another. Rules can be for calculations, statistics, email alerts, xmpp messages (Section 3.3.3), push notifications and more.

- It provides a data logging service and access, and stores the historical data points and data objects.
- Storage in any format that can be serialised into a string, such as JSON or XML.
- It filters the noise and important changes sent to another larger central instance.
- It processes a specific type of data and can store it.
- Time- or geo-stamping of the data.
- Nimbits clients provide over Internet, data collection in real time, charts, chart and graphical plots of collected data and data entry.
- Data visualisation for data of connected sensors to IoT devices.
- Supports the alerts subscription, generation and sending in real time over the Internet.
- It creates streams of data objects and stores them in a data point series.
- Data accessibility and monitoring from anywhere, and is used to shape the behaviour of connected devices and software.
- It supports the mBed™, Arduino, Raspberry Pi based and other hardware platform based IoT devices.
- Web service APIs are easy to implement on device hardware acting as clients to Nimbits web services, and connect to the web service and send data.
- It deploys software on Google App Engine, any J2EE server on Amazon EC2 or on a Raspberry Pi.

Figure 6.3 shows connected devices, sensor nodes, network data points, Nimbits server, deployment at the device network nodes, and networked with the Nimbits Server (PaaS, SaaS and IaaS services) at cloud for applications and services.

Architecture shown in Figure 6.3 shows a NimbitsServerL which deploys at each device node and is an instance of the NimbitsSeverS at the cloud. Each NimbitsServerL of the device node generates the calculation objects for device nodes.

Each node also hosts an XMPPServerL, an instance of the XMPPSeverC at the cloud.

XMPPServerL deploys at each device node and generates the data feed channels for the XMPP messages and alerts.Each XMPPServerL sends feeds to XMPPSeverC.

Data Points

A data point means a collected value of a sensor in a group of sensors. Data points organise the data in a number of ways. For example, points can have child points (child points mean subpoints; for example, if light level is a data point then light on or off is a child point and light level above or below the threshold can be another child point.)

Points can be in the folders. The folders can go as deep as like in a tree (Tree means a folder having several subfolders, a subfolder having several subfolders, till the leaf subfolder.)

Any type of document can upload and organise them with the points. Files can be shared publicly or with the connections.

A subscription data feed is a special point for each user that logs system messages, events, alerts from other points which are subscribed by a service and more.

Data Channels

A user can create a data feed channel which shows the system events and messages that also shows data alerts which are subscribed to show up in the feed. The user can subscribe to the data point of other users also, and configure the subscription(s) to send messages to the feed. The user can observe the idle, high or low alerts here in real time. The user data feed is just another Nimbits data point.

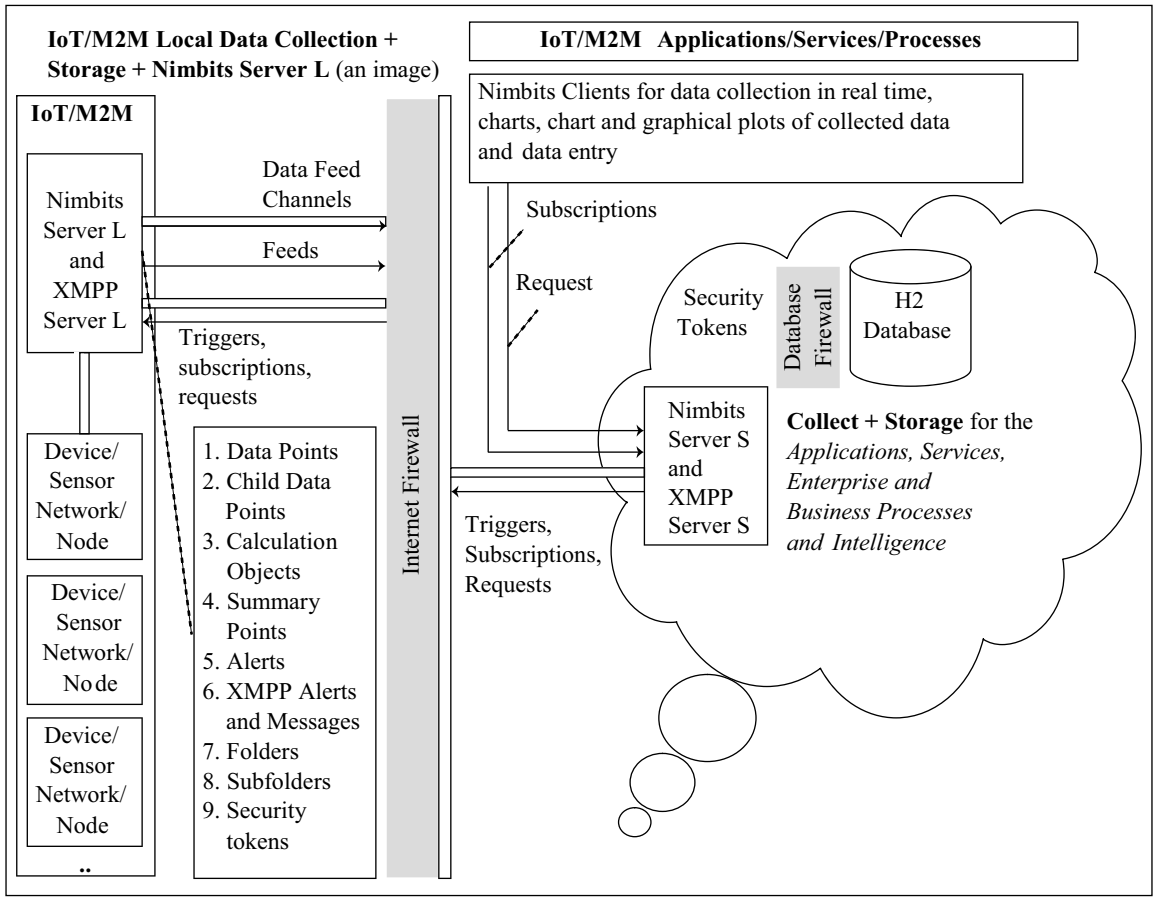


Figure 6.3 Connected devices, sensor nodes, network data points, Nimbits server, deployment at the device network nodes, and networked with the Nimbits Server (PaaS, SaaS and IaaS services) at cloud for applications and services.

Using Advanced Features

An application can create a connection to another Nimbits application or service.[10] The application sends an invitation and if the invitee approves, then the application can see

[10] http://bsautner.github.io/com.nimbits/

invitee's points and data in their tree (if they set the objects permission level to public or connection viewable).

Nimbits 3.6.6 introduced H2 database engine. Nimbits 3.8.10 includes H2 database engine. H2 is Java SQL database. APIs are in pure Java, The main features of H2 are:

- Very fast, open source, JDBC API
- Embedded and server modes; in-memory databases
- Encrypted database
- ODBC driver
- Full text search
- Multi-version concurrency
- Browser-based console application
- Small footprint (around 1.5 MB jar file size)

MySQL is not in pure Java and have no provisions for in-memory or encrypted databases. Footprint (DLL) is nearly 4 MB. (JAR means Java archive while DLL stands for dynamically linked library.)

Security Tokens

Nimbits 3.9.6 provides security tokens in a new way.

Breakthrough Performance and Data Integrity

Nimbits server 3.9.10 version launched in June 2015 provisions for the breakthrough performance and data integrity.

Alerts

A filter means applying some rule to get new data for a data point. The filter item in the tree called "ah" is for XMPP alerts (Section 3.3.3). A user application can now have many JIDs (Jabber IDs) for a single point—alerts and messages can be sent over XMPP using the custom JID of points.

Jabbing

Jabbing means pushing the alerts or messages down quickly or pushing repeatedly. Each type of alert or message is assigned a Jabber ID, called JID. Each JID consists of three main parts, viz. the node identifier (optional), domain identifier (required) and resource identifier (optional). A JID is written in notations as <JID>:= [<node>"@"]<domain>["/"<resource>.

Subscriptions

A user can create many subscriptions for a single point. It may subscribe to one of the points, other user, or anyone's public point to get the alerts. The user gets alerts when the point goes into an alarm state, or receives new data. Alarm state means reaching preset value. For example, a pressure boiler reaching critical pressure necessitating action or water reaching boiling point necessitating action.

Subscriptions are alternative to the configurations of alert (an alert-configuration means specifying when a point became idle, high, low using the point property menu). A subscription to a point creates on configuring the subscription—how an application is programmed to get alerts (XMPP, Twitter, Email or other) and what events trigger the alert (new data, high or other alerts). Refer to a tutorial.[11]

Example 6.6 for Internet of ACVMs clarifies the data points, child points for the system events and messages, subscriptions, JIDs for the alerts to XMPP server and subscriptions to NimbitsServerC at the cloud. It assumes the architecture shown in Figure 6.3.

Example 6.6

Problem

Consider Internet of Automatic Chocolate Vending Machines (ACVMs) (Examples 5.1, 5.3 and 5.8). Assume that each ACVM device nodes generates the following data points, child points and further children up to the leaf and summary points.
Suggest the data points and child points for the system events and messages, subscriptions for the alerts to XMPP server and subscriptions to NimbitsServerC at the cloud.

Solution

Data generated on each ACVM is (i) *ACVM_ID* for machine Id, (ii) *TimeDate* (iii) *ActiveStatus*, (iv) to (viii) *ChocFl1Sold*, *ChocFl2Sold*, *ChocFl3Sold*, *ChocFl4Sold*, *ChocFl5Sold*, each time a chocolate sells among five flavours Fl1, Fl2, Fl3, Fl4 and Fl5.

Data Points for each ACVM are (1) *ACVM_ID*, (2) *TimeDate* (3) ChocolateUnsold and (4) ChocolateReqPending for number of chocolates requested and pending filling at the machine. Data point can possess a number of child points and a child point can have further children in a tree format.

Child points are as follows:

- Child points of each (1) ACVM_ID are (1.1) *Region*, (1.2) *Address*, (1.3) *Installation date*, (1.4) *MaintenanceSchedule,* (1.5) *Fill Service address*, and (1.6) *ActiveStatus* (for ACVM status = active or inactive), and (1.7) ACVM_Nodedomainadress.
- **Child points of (2)** TimeDate are (2.1) to (2.5) HourOnDate, DayOnDate, WeekOnDate, MonthOnDate, and YearOnDate for which hour, which day (1^{st} day, 2^{nd} day, …, or 365^{th} day), which week (1^{st} week, 2^{nd} week, …, or 52^{nd} week), which month (1^{st} month or 2^{nd} month …), and which year (1^{st} year or 2^{nd} year, …).
- **ChildPoints of (3)** ChocolateUnsold are (3.1) to (3.5) ChocFl1UnSold, ChocFl2UnSold, ChocFl3Sold, ChocFl4unSold, ChocFl5UnSold for unsold flavour of each chocolate flavour.
- **ChildPoints of (4)** ChocolateReqPending **are** (4.1) to (4.5) ChocFl1ReqPending, ChocFl2ReqPending, ChocFl3ReqPending, ChocFl4ReqPending, ChocFl5ReqPending, for each chocolate flavour requested and pending from fill service.
- A child point can have further children in a tree format. ChildLeafPoints are as follows:
- ChildLeafPoints of (1.4) *MaintenanceSchedule* are (1.4.1) Last_MaintenaceDate, and (1.4.2) *Next_ Preventive_Maintenance_Date* for last and next preventive maintenance dates.

[11]http://nimbits.blogspot.in/.

- ChildLeafPoints of (1.6) *ActiveStatus* are (1.6.1 to 1.6.5) ChocolateFl1Present, ChocolateFl2Present ChocolateFl3Present, ChocolateFl4Present and ChocolateFl5Present for the number of each chocolate flavour present.

An application can have many subscriptions to a point. It may subscribe to another user's point to which the user is connected or find a point using the search engine on nimbits.com, or may subscribe to a public point from another user who is not even known.

Summary point

A user creates a summary point which can compute averages, minimum, maximum, standard deviations, variance and sums of another point on a specific time interval basis.

Calculations

A user can create calculation objects for a point. The objects can organise in a tree and a user can apply many formulas for a single data point. For example, in temperature sensor data point, one formula is for increase over the last value while the other is for increase over a normal value, each time a new temperature data point is recorded.

Example 6.7 for Internet of ACVMs clarifies the summary points, data feed channels for the system events and messages, subscriptions, JIDs for the alerts to XMPP server and subscriptions to NimbitsServerC at the cloud. It assumes the architecture shown in Figure 6.3.

Example 6.7

Problem

Nimbits ServerL generates the calculation objects. XMPPServerL registers for the subscriptions of the resources (messages and alerts). XMPPServerC receives the data channel feeds the events and messages from the multiple instances of XMPPServerL for the ACVM nodes for the device nodes at different addresses. Each NimbitsServerL sends data channel feeds to the NimbitsServerC and XMPPServerC at the Cloud.
An application connects to the NimbitsServerC at the cloud service. Nimbits Client of the application sends subscriptions for the events and messages to XMPPServerC.
Refer to Example 6.6. Suggest the summary points, data feed channels for the system events and messages, subscriptions for the alerts to XMPP server and subscriptions to NimbitsServerC at the cloud.

Solution

Summary points are generated from the DataPoints at the ACVM device nodes itself. Summary points are as follows: (s1) ACVM_ID, (s2) HourDate, (s3.1 to s3.5) Fl1HourlySell, Fl2HourlySell, Fl3HourlySell, Fl4HourlySell, and Fl5HourlySell, summary points communicates to NimbitsServerL through data feed channel every hour.

Calculation Objects are the calculated objects at the NimbitsServerL, an instance of Nimbits Server S at the cloud. CalculationObjects deploys the data channel feeds received for the summary points from the device nodes.

Consider a calculation object *c*. Each calculation object has following resources:

1. (*c*.1) ACVM_ID, (*c*.1.1) ACVM_Node_domainAddress.

2. (*c.*2) HourDate, and (*c.*2.1 to *c.*2.1.5) Fl1FillHourlyStatus, Fl1FillHourlyStatus, Fl1FillHourlyStatus, Fl1FillHourlyStatus, Fl1FillHourlyStatus for the percentage of chocolates of each flavour remaining unsold at the CVM with respect to maximum fill.
3. (*c.*3) DayDate, and (*c.*3.1 to *c.*3.5) are Fl1DaySell, Fl2DaySell, Fl3DaySell, Fl4DaySell, and Fl5DaySell for daily sell of each ACVM.
4. (*c.*3.6 to *c.*3.10) are Fl1DaySellAmounnt, Fl2DaySellAmount, Fl3DaySellAmount, Fl4DaySellAmount, and Fl5DaySellAmount for daily sell amount collected, and (*c.*3.11) DayTotalSellAmount.
5. (*c.*4) WeekDate, (*c.*4.1 to *c.*4.5) WeeklyTotalFl1SellAmount..... to WeeklyTotalFl5SellAmount for the amount of Sell of each ACVM flavours in one week, and (*c.*4.6) WeekTotalSellAmount.
6. (*c.*5) MonthDate, (*c.*5.1 to *c.*5.5) MonthlyTotalFl1SellAmount, ... to MonthlyTotalFl5SellAmount for the amount of Sell of each ACVM flavours in one month, and (c5.6) MonthlyTotalSellAmount.
7. (*c.*6) YearDate, (*c.*6.1) to (*c.*6.5) YearlyTotalFl1SellAmount, ...to YearlyTotalFl1SellAmount for the amount of Sell of each ACVM flavours in one year, and (c6.6) YearTotalSellAmount.

Data feed channel communicates the system events and messages, subscriptions and XMPP alerts. Subscription is for the resources at the JIDs. A JID is <JID>:= ACVM_ID@ ACVM_Node_domainAddress/ <resourceName>.

Nimbits clients at applications subscribe to the data feed channels for the XMPPServerS. Following alerts generates on subscriptions to XMPPServerL

1. When (1.6) *ActiveStatus* (for ACVM status = active or inactive) = 0 due to some ACVM device node functioning problem.
2. Hourly alerts as long as (1.6) *ActiveStatus* = 0 or *Next_Preventive_Maintenance_Date* = DayDate
3. When (1.4.2) Next_Preventive_Maintenance_Date < TimeDate
4. Daily alerts as long as (1.4.2) TimeDate > N*ext_Preventive_Maintenance_Date*
5. When (*c.*2.1 to *c.*2.1.5) Fl1FillHourlyStatus, Fl1FillHourlyStatus, Fl1FillHourlyStatus, Fl1FillHourlyStatus, Fl1FillHourlyStatus falls below the 20% of the maximum fill space for each flavour chocolate, falls below 10% and falls below 5%.
6. When any of the (3.1) to (3.5) falls below the 1% of the maximum fill space for each flavour chocolate.

Messages generate and communicate on subscriptions to XMPPServerL for the required calculation object resources (c.1) to (c.6) and their data fields. XMPPServerS receives the data feed channels from each XMPPServerL for the alerts and calculation objects for each ACVM device nodes.

Each Nimbits Client at application receives the data channel feeds from NimbitsServerS and XMPPServerS.

6.4.3 Using Public Cloud IoT Platforms

Many cloud PaaS and SaaS platforms are now available for IoT. Table 6.1 gives the examples of cloud-based platforms.

Table 6.1 Cloud PaaS and SaaS Platforms

Cloud Platform	Features
Spark	Distributed, cloud-based IoT operating system and web-based IDE includes a command-line interface, support for multiple languages and libraries for working with many different IoT devices
OpenIoT	Open source middleware, enables communication with sensor clouds and enables cloud-based sensing as a service and developed use cases for smart agriculture, intelligent manufacturing, urban crowd sensing, smart living and smart campuses
Device Hub	*Open source backbone* for IoT, a cloud-based service that stores IoT-related data, *provides* data visualisations, *allows* web-page-based console to control IoT devices, *enables* developers to create applications such as tracking of vehicular data, monitoring weather data.

Section 12.2.4 describes AWS IoT, IBM IoT Foundation, Cisco IoT, IOx and Fog, TCS CUP cloud PaaS, which have many features but not open source for enterprises IoT applications and services.

Reconfirm Your Understanding

- IoT/M2M client application communicates, subscribes and sets the triggers and receives the responses and feeds deploying a cloud service.
- Cloud service deploys an instance of cloud server at edge computing node at sensor nodes network provisions for generation and communication of data points, objects, tree, streams, alerts, and feeds on requests, subscriptions and triggers.
- Feeds consist of time/geo-stamped data points, streams and alerts. Feed for messages and alerts generates on application of rules for filtering and calculations.
- Xively (Pachube/COSM) Cloud PaaS service deploys an instance of server at Arduino and other IoT sensor nodes platforms with provision for real-time data collection, data visualisation, graphical plots, HTTP based APIs and feeds.
- Nimbits Cloud PaaS service deploys an instance of Nimbits server at Arduino, Raspberry Pi or other IoT sensor nodes platforms that provides for real-time data collection, store at an H2 Database, edge computing, Nimbits data, alerts, messages, application programmed to get alerts (XMPP, Twitter, Email or other), data visualisation, analytics, rule engine, calculation objects, summary points, alerts and feeds on subscriptions, triggers and request.
- Spark, OpenIoT, DeviceHub and Intel IoT Cloud Analytics Kit are also cloud platforms which can be deployed for data collection, store, organising and analytics by clients/applications.

Self-Assessment Exercise

1. What are the methods in architecture shown in Figure 6.1 that are used for the data collection, storage and computing in cloud computing architecture in Figure 6.3? ★
2. When are the push, pull and subscription methods used in cloud computing architecture in which instances of server deploys at the device nodes or networks? ★
3. List the merits of deploying instances of server at the IoT device nodes or networks during cloud computing. ★★
4. What are the methods for generating data points from the data generated at the device nodes? List the latest features in Xively (Pachube/COSM) cloud platform. ★★
5. List the latest features in Nimbits cloud platform. ★★
6. List the merits in usages of feeds for messages and alerts from the cloud server instances to a cloud server. ★★★
7. List the merits of an H2 database over MySQL. Recall Q5.3.8. ★
8. List the datapoints, summary points, alerts, data channels, subscriptions for alerts and messages required in Internet of Waste Containers Management. ★★★

Key Concepts

- Amazon EC2
- Calculation object
- Child point
- Cloud common features
- Cloud computing
- Cloud essential features
- DaaS
- Data feed channel
- Data point
- Device Hub
- Distributed computing
- Edge computing
- Elasticity
- Everything as a service (XAAS)
- Feed
- Grid computing
- IaaS
- Microsoft Azure
- Network virtualisation function
- Nimbits Cloud
- PaaS
- SaaS
- Server instance
- Subscription
- Summary points
- TCS iON
- Trigger
- Utility computing
- Virtualisation
- Xively (Pachube/COSM)
- XMPP Server

Learning Outcomes

LO 6.1

- Conventional data collection and storage methods are (i) save at a devices *web server*, (ii) communicate and save the *files* locally, (iii) communicate and save data or results of

computations at a dedicated data store or coordinating node locally, (iv) communicate and save data at *a local node of a distributed DBMS*, (v) communicate and save at a remote node in a distributed DBMS, (vi) communicate on the Internet and save *at an enterprise server*, and (vii) communicate on the Internet and save *at data centre for an enterprise*.

- Cloud presents a virtualised environment for network, data store and computing.
- Usage of cloud is a method for data storage and computing with features of on demand self-service, resources pooling, network broad accessibility, elasticity and measurability.
- The cloud has massive scale, scalability, maintainability, homogeneity, virtualisation, interconnectivity platform and resilient computing features.
- Cloud deployment models are public cloud, private cloud, community cloud and community cloud.

LO 6.2

- Cloud computing model is everything as a service model.
- Four models are SaaS, PaaS, IaaS and DaaS.
- Example of cloud infrastructure services are Amazon Virtual Servers, GoGrid virtual servers, Elastic Computing Cloud (EC2), Cloud.com open source IaaS, TCS Transformation Solutions, Cisco IaaS, and IBM Bluemix Cloud shared infrastructure service that automate fluctuating demands for the IT resources.

LO 6.3

- IoT cloud-based services using the Xively, Nimbits, other platforms, such as TCS Connected Universe deploy a cloud server instance (device agent) at the device node network.
- The server then collects the data, generates the data points and data point trees, calculation of objects using filtering and applying rules.
- The server communicates feeds on requests, triggers and subscriptions.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. Cloud computing means a collection of (i) services available over the Internet that deliver computational functionality, (ii) infrastructure service of a service provider, (iii) infrastructure deployments for a utility or grid computing or web services environment, (iv) services for distributed computing, (v) grid of computers, (vi) servers, (vi) data centres, and (vii) application, application-support, security, transport, network, adaptation and data-link layer protocols. ★★
   (a) All
   (a) (i) to (iii)
   (b) All except (vii)
   (c) (iii) to (vi)

2. User application, service or process access to resources appearing as just one network, though in fact the network access to the resources may be through multiple resources and networks, is called (i) cloud network, (ii) network function virtualisation, (iii) application virtualisation or (iv) Internet access. ★
   (a) (i)
   (b) (i) and (iv)
   (c) (i)
   (d) (ii)
3. IoT cloud platform provides (i) infrastructure for large data storage of devices, RFIDs, industrial plant machines, automobiles and device networks, (ii) computing capabilities, such as analytics, IDE and data visualisation, (iii) provides collaborative computing and data store sharing, (iv) databases, (v) in-memory databases for on-line analytics, (vi) devices data collection and connectivity to services, applications, (vii) data calculation objects, and (viii) data feed channels. ★★
   (a) All except (ii), (vii) and (viii)
   (b) All except (iii) and (v)
   (c) All except (ii), (iii) and (viii)
   (d) All except (v)
4. Essential features of cloud services for data collection, store and computing are (i) on demand self-service, (ii) resources pooling, (iii) measurability, (iv) massive scale, (v) no maintenance costs to the user, (vi) homogeneity, and (vii) advanced security for storage, applications, computing infrastructure, services, data centres and servers at remotely connected cloud services independently at low cost. ★★★
   (a) All except (i)
   (b) All except (ii) and (v)
   (c) (i), (ii) and (iii)
   (d) (ii) to (v) and (vii)
5. Nimbits 3.8.10 H2 database has the following features (i) Java SQL database, (ii) very fast, open source, JDBC API, (iii) embedded and server modes; in-memory databases, (iv) encrypted database, (v) ODBC driver, (vi) full text search, (vii) multi-version concurrency, (viii) desktop-based console application, and (ix) footprint around 10 MB jar file size. ★
   (a) All except (viii) and (ix)
   (b) All except (v) to (viii)
   (c) (i) to (iv))
   (d) (i), (ii), (iv) and (vii)
6. (i) Jabbing means pushing the alerts or messages down quickly or pushing repeatedly, (ii) JID is assigned for a group of alerts or messages from a set of domains, (iii) subscription means requesting pushing of a resource, (iv) a feed means a set of data points or objects or data streams or messages which generate and communicate after application of rules for filter, calculation, compaction, fusion, compression, analysis or aggregation, and (v) the feed may also be for the alerts on the programmed triggers or alarms. ★★
   (a) All except (i)
   (b) All except (ii)
   (c) (i), (ii) and (iii)
   (d) (i), (ii) and (iv)

7. DeviceHub is (i) an open source backbone for IoT, (ii) a hybrid cloud-based service that stores IoT-related data, (iii) provides data visualisations, (iv) allows web-page-based console to control IoT devices, (v) provides descriptive and predictive analytics, and (vi) enables developers to create applications such as tracking of vehicular data, monitoring weather data. ★★★
   (a) (iii) to (vi)
   (b) All except (ii) and (v)
   (c) All except (v)
   (d) All
8. Examples of clouds are (i) Nimbits, (ii) DeviceHub, (iii) Amazon Virtual Servers, (iv) GoGrid virtual servers, (v) Elastic Computing Cloud (EC2), (vi) Cloud.com open source IaaS, (vii) TCS Transformation Solutions, (viii) Cisco IaaS, and (ix) IBM Bluemix Cloud shared infrastructure service that automate fluctuating demands for the IT resources. ★
   (a) All
   (b) All except (ix)
   (c) (i) to (v)
   (d) All except Nimbits and DeviceHub
9. A data feed channel communicates the system (i) events, (ii) messages, (iii) subscriptions and (iv) XMPP Alerts. ★
   (a) (i) to (iii)
   (b) (i) and (ii)
   (c) All
   (d) (iv)

Short-Answer Questions

1. List the service models of cloud computing. **[LO 6.1]** ★
2. Why is virtualisation required in cloud computing? **[LO 6.1]** ★★
3. How does cloud computing differ from grid computing? **[LO 6.1]** ★★★
4. List the usages in IoT deploying 'everything as a service' model of cloud services. **[LO 6.2]** ★
5. List the merits of using the cloud in IoT applications. **[LO 6.2]** ★★
6. Draw the architecture deploying the edge and cloud computing for Internet of Automobile Components. **[LO 6.3]** ★★★
7. Show the architecture deploying edge and cloud computing in Example 6.6 for IoT of ACVMs. **[LO 6.3]** ★★
8. How is cloud used as a database in IoT applications? **[LO 6.3]** ★★
9. What are the commonalities and contrast between Nimbits and Xively clouds for IoT applications? **[LO 6.3]** ★
10. What are the merits of usage of a data point tree for the device nodes or networks in place of individual data points? **[LO 6.3]** ★★
11. XMPP server enables device nodes or network messages and alerts as a common resource for the applications. **[LO 6.3]** ★★★
12. How is a calculation object created from data point trees? **[LO 6.3]** ★★★

Review Questions

1. Describe the virtualisation concept in usage of cloud services. **[LO 6.1]** ★★
2. What are merits and concerns when using cloud services for IoT applications? **[LO 6.2]** ★
3. What are deployment models for cloud services for IoT applications? **[LO 6.2]** ★★
4. Explain SaaS, PaaS, IaaS and DaaS service models of clouds. **[LO 6.2]** ★
5. Describe usage of each cloud—the Spark, OpenIoT, DeviceHub, Intel IoT Cloud Analytics kit, Xively and Nimbits. **[LO 6.3]** ★★★
6. What are the functionalities in Nimbits for IoT applications? **[LO 6.3]** ★★★

Practice Exercises

1. Give one example each for understanding the terms, 'on-demand self-service', 'resources pooling', 'network broad accessibility', 'elasticity' and 'measurability'. **[LO 6.1]** ★★
2. List the services offered at Amazon EC2 and TCS clouds. **[LO 6.2]** ★
3. What are the cloud service models other than the four main service models, viz. SaaS, PaaS, IaaS and DaaS? **[LO 6.2]** ★
4. List the main cloud services from Oracle, Google, Amazon, Microsoft, Tata Communications, GoGrid and NetSuite. **[LO 6.2]** ★★★
5. Recall Example 5.2. Show the architecture for usage of cloud at Internet of Automobile Components in automobile service centre applications diagrammatically. **[LO 6.3]** ★★
6. Show the architecture for usage of cloud. applications at Internet of ACVMs diagrammatically. **[LO 6.3]** ★★
7. Recall Examples 1.2 and 6.3 for Internet of Streetlights. Show the architectures when using the cloud services and when using the gateways at each street with a group of streetlights and web server. List the communicating of data from each device node, data points and feeds for an IoT device network in each street for controlling and monitoring. **[LO 6.3]** ★
8. Recall Example 5.2 for Internet of Automobile Components communicating the data, data points, summary points and data feed channels for building automobile service centre H2 database. List the data points, child points, calculation objects, summary points, data feed channels for the system events and messages, subscriptions for the alerts to XMPP server and subscriptions for building the H2 database atNimbitsServerC at the cloud. **[LO 6.3]** ★★★
9. Recall Example 5.3 RDBAVCM Tables A, B and C and Example 6.6. List the data points, child points, calculation objects, summary points, data feed channels for the system events and messages, subscriptions for the alerts to XMPP server and subscriptions to NimbitsServerC at the cloud corresponding to the tables. **[LO 6.3]** ★★★

7

Sensors, Participatory Sensing, RFIDs and Wireless Sensor Networks

Learning Objectives

- **LO 7.1** Elucidate sensor technology for sensing the real world using analog and digital sensors, and examples for sensing devices for IoT and M2M
- **LO 7.2** Define the concepts of participatory sensing, industrial IoT and automotive IoT
- **LO 7.3** Describe the uses of actuators in devices
- **LO 7.4** Describe the uses of data communication using serial bus protocols
- **LO 7.5** Explain the Radio Frequency Identification (RFID) technology
- **LO 7.6** Explain the Wireless Sensor Network (WSN) technology

Recall from Previous Chapters

From Chapters 1 and 5 we have learnt that sensors play an important role in gathering data for IoTs and M2M. One of the components of IoT devices is hardware which consists of a microcontroller, firmware and the sensor(s) and/or actuator(s). The automobile components embed the sensors and circuits consisting of microcontrollers and firmware in a car. The sensing devices network communicates the status of each component and sensed data to the applications at an automobile service centre. An application at the centre plans the service schedule and generates alerts for the urgent actions required and the need for specific actions.

Actuators are devices used for actions, such as switching ON a set of streetlights or delivering a chocolate in an automatic chocolate vending machine and then communicating information on the Internet for a service provider application.

RFIDs and their sensors play an active role in Internet of RFIDs. RFIDs are used for identifying objects using RF. Two examples of application of Internet of RFIDs are tracking parcels, goods and delivery, and supply chain management.

WSN can be defined as 'a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations.'

7.1 INTRODUCTION

IoT and M2M applications need a large magnitude of data which is generated from a large number of devices, ATMs, sensors at parking slots, health devices in ICUs, machines in industrial plants, embedded components in automobiles, RFIDs, or wireless sensor networks.

IoT and M2M data is generated using sensors, embedded devices and systems

Data is generated using sensors, embedded devices and systems at the physical layer in the IoT architecture. Thereafter, the data communicates through the data-link, data-adaptation, network, application-support and application layers to the applications of IoT.

Data is used for analytics, visualisation, intelligence and knowledge discovery or controls and monitoring. Control systems use the sensors for monitoring and the actuators for actions.

Prototyping and designing of IoT need embedded device platforms, which provide connectivity to the Internet and can communicate with applications using the Internet. The applications in IoT monitor and control devices, systems and machines using the actuators.

This chapter describes technological aspects of sensing, concepts of participatory sensing in M2M devices, industrial IoT and automotive IoT. It also introduces the role of actuators and describes details of use of RFIDs and WSNs.

Section 7.2 describes sensor technology and analog and digital sensors which are used in IoT and M2M. Section 7.3 describes participatory sensing concept, industrial IoT and automotive IoT. Section 7.4 describes use of actuators. Section 7.5 describes use of data communication protocols (UART, I2C, LIN, CAN, USB and MOST) by IoT and M2M devices. Sections 7.6 and 7.7 describe technological aspects of RFIDs and WSNs.

7.2 SENSOR TECHNOLOGY

LO 7.1 Elucidate sensor technology for sensing the real world using analog and digital sensors, and examples for sensing devices for IoT and M2M

Sensor technology is a technology used for designing sensors and associated electronic readers, circuits and devices. A sensor can sense a change in physical parameters, such as temperature, pressure, light, metal, smoke and proximity to an object. Sensors can also sense acceleration, orientation, location, vibrations or smell, organic vapours or gases. A microphone senses the voice and changes in the sound, and is used to record voice or music.

Sensor technology uses sensors and electronic circuits, readers and devices

A sensor converts physical energy like heat, sound, strain, pressure, vibrations and motion into electrical energy. An electronic circuit connects to the input at a sensor. The circuit receives the output of the sensor. The output is according to the variation in physical condition. A smart sensor includes the electronic circuit within itself, and includes computing and communication capabilities.

The circuit receives energy in form of variations through currents, voltages, phase-angles or frequencies. Analog sensors measure the variations in the parameters with respect to a reference or normal condition and provide the value of sensed parameter after appropriate calculations.

The change of states with respect to a reference or normal condition senses the states in the form of 0s and 1s in digital sensors.

7.2.1 Sensing the Real World

Sensors are electronic devices that sense the physical environment

Electronic components can function as sensors. Sensor is an electronic device in a circuit that senses a physical environment or condition. The sensor sends signals to an electronic circuit, which interconnects to a serial port interface at a microcontroller or controller or computing device.

A sensor senses a specific physical condition when it exhibits a measurable change in a characteristic circuit parameter on the change in the specific physical condition or environment.

Example of Resistive, Capacitive, Diode and Transistor-based Sensors

Example 7.1 gives the characteristic circuit parameters which change with the physical conditions.

Example 7.1

Problem

Assume R is resistance, C is capacitance, I_{rev} is reverse saturation current in a p-n diode and I_{sat} is saturation current in a bipolar junction transistor (BJT). How do the characteristic circuit parameters R, C, I_{rev} and I_{sat}, sense the physical environment? How are they used for measuring the surrounding condition?

Solution

- Resistance R, of a specially designed wire wound on a coil is used as temperature sensor. R functions as a component. R shows variation with temperature in metal wires such as platinum. A touch-screen senses the resistive variation in R with the touched position. A strain sensor resistance shows the variation in R with the strain. Change in R is proportional to change in length due to strain; for example, 12 cm per 10 kilo-Ohm. A flex-sensor senses the bending on applying a force by change in R. Sensor resistance R varies on flexing with the path or local deflection profile change which senses the flex.

  Resistance of a sensor of organic solvent vapours shows measurable drop in the vapours concentration in the vicinity. The conductivity (reciprocal of resistance) of sensor increases depending on the vapour or gas concentration in the air near the sensor. Gas sensor is a metal oxide coated sensor whose resistance varies with vapour adsorption; for example, Sensor (TGS2620) of Figaro Company.

  Resistance of a photo-conductor shows measurable drop in the presence of light. The conductivity (reciprocal of resistance) of the sensor increases depending on the radiation intensity. The sensor is basically a special semiconductor sensor, which shows conductivity variation exponentially with the received light intensity.
- *Capacitance*, *C,* is used as proximity sensor when the capacitance of sensing component shows variation with proximity to a specific object, such as a metal part or a finger. A level sensor capacitor shows variation with level of filler in a container.

  A touched position sensor (capacitor) shows variation when a finger touches or is in the proximity or vicinity to the touch position on a screen. The variation in C is as per the touched position. A position when corresponds to a menu item displayed to the user, then the selected application runs on touching.
- Reverse saturation current I_{rev} of a *p-n* junction diode is used as a temperature sensor when that shows measurable variations within the temperature range of the study.
- A specially made *p-n* diode with a window-entry for radiation at the junction can be used as a photo sensor. The sensor functions as an incident radiation-intensity sensor when the diode reverse-saturation current I_{rev} shows a measurable variation with radiation energy.
- A specially made BJT with a window-entry for radiation at the junction is used as a photo sensor (phototransistor). Saturation current I_{sat} between collector-emitter in the BJT shows measurable variation within incident infrared or light radiation intensity.

Mobile phones use a number of sensors, such as finger touch and gesture via a sensor on touchscreen

A characteristic parameter of a circuit changes with the physical conditions. Technology that facilitates such changes due to sensing is also used in mobile phone. A mobile phone can sense surrounding conditions. The touchscreen of a mobile phone can senses a finger touch and gestures. Smartphones have resistive and capacitive sensors, photodiode current-based sensors, and acceleration, gyroscope, temperature and pressure sensors. The sensors enable the functioning of applications and games.

A microcontroller is an associate computing device with a sensor circuit which calculates the touched position and maps it to a user command when a resistive-based touchscreen is used. Then the mobile phone takes further actions as per the command.

Analog Sensors

Analog sensors generate analog outputs as per the physical environmental parameters such as temperature or strain

Analog sensors use a sensor and an associated electronic analog circuit. Analog sensors generate analog outputs as per the physical environmental parameters, such as temperature, strain, pressure, force, flex, vapours, magnetic field, or proximity. Resistance of the sensing component may show measurable changes with surrounding pressure or strain or magnetic field or humidity. Resistance of a pressure sensor increases on pressure which creates a strain on the sensor. A flex sensor, for example, of 2.2 inch or 4.5 inch length, shows that its resistance across the sensor strip increases on flexing due to a changed path and deflection of the sensing resistor.

The measurement of analog output from a sensor circuit is performed as follows—the sensor output is given to the input of a signal conditioning-cum-amplifying circuit (SC). The SC output is the input to an Analog-to-Digital Converter (ADC). The ADC gives a digital output; for example, 8 or 12 bits. This output is read using a microcontroller. Microcontroller reading and computation gives the value of the sensed parameter value and shows the physical condition around the sensor.

Reading Temperature from Resistance Sensor

Example 7.2 shows how an analog sensor and a sensing resistor with associated electronics enables the sensing and measuring of temperature.

Example 7.2

Problem

Assume an electronic circuit with a sensing component resistance as a physical object. How are temperatures sensed?

Solution

A resistor in the form of a wire or a component can be a part of the electronic circuit. Ohm's law states that resistance remains constant only as long as physical conditions remain the same. The resistor functions as a sensor when its value changes measurably within the required temperature range for sensing.

An equation or a table relates the sensor resistance with the sensed and measured parameter value at a given physical condition

The measurements can be first made using two standard or reference temperature points, such as 0°C and 100°C. An equation or table can be prepared for the sensing component resistance R values as a function of the temperature T in °C. When changes are linearly related to the change in the physical environment then the equation is used. When changes are nonlinearly or

exponentially related to the change in the physical environment, then use of the table is preferred.

When temperature, such as of oil or coolant plate in an M2M or IoT device in an automobile needs to be sensed then a simple electronic circuit uses the sensing component at the sensed object. The associate computing device calculates the temperature value at the measuring instance. Figure 7.1 shows a circuit consisting of a resistance bridge. The bridge has one sensing resistor at the sensing object and three fixed (standard) resistors. The figure shows a microcontroller using an electronic circuit with a port connected to sub-circuits, serial port interface, ADC, signal conditioning amplifier and resistance bridge.

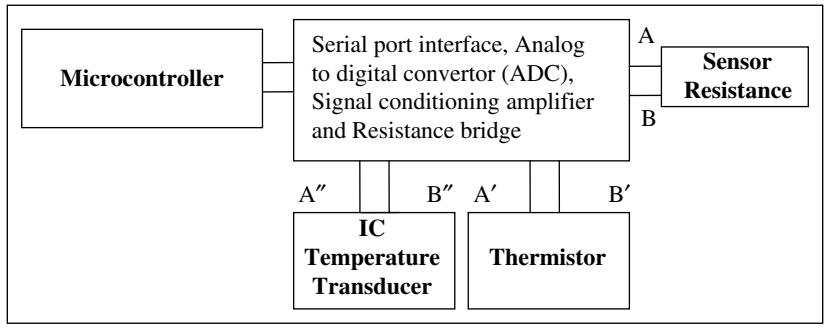


Figure 7.1 Microcontroller serial port connected to sub-circuits—serial port interface, ADC, signal conditioning amplifier, resistance bridge and sensor resistance outputs A ad B. Alternatively circuit connects to a thermistor output at A′ and B′ or IC based temperature-transducer output at A″ and B″

A transducer induces current or voltage. The output changes as per a change in the physical energy at input. An IC-based circuit for a temperature-transducer induces current in the output according to the heat energy, represented by the temperature.

Microcontroller is a computing device which reads the input at its ports, saves the reading in memory and then the reading is used for computations and communication.

Reading from Capacitive Sensor

Example 7.3 shows how a sensing capacitor enables sensing using a capacitance bridge.

Example 7.3

Problem

Assume an electronic circuit with a sensing component capacitor as a physical object. How are the changes of capacitance sensed when using an electronic circuit with a sensing capacitor as a physical object?

Solution

Consider two cases:

1. When a metal part is present in the vicinity of a parallel plate capacitor, the capacitance *C* changes and proximity distance is sensed.
2. When a finger reaches near a screen, and the screen has a metallic grid at its base, then *C* will change depending upon the touched position or *C* varies with time when the finger is approaching towards the vicinity to a menu item on the screen.

C of a sensor object is a part of a capacitive bridge. The associate computing device calculates the proximity distance in case 1 and touched position in case 2 as well as the successive variations. Microcontroller is a computing device which reads the input at its ports, saves the input at memory, and then uses the data for communication over the Internet.

Figure 7.2 shows a circuit using a capacitance bridge. The bridge consists of the sensing capacitor (object) and three fixed (standard) capacitors. The figure shows a microcontroller-based electronic circuit with port connected to four sub-circuits, serial port interface, ADC, signal conditioning amplifier, diode and capacitance bridge.

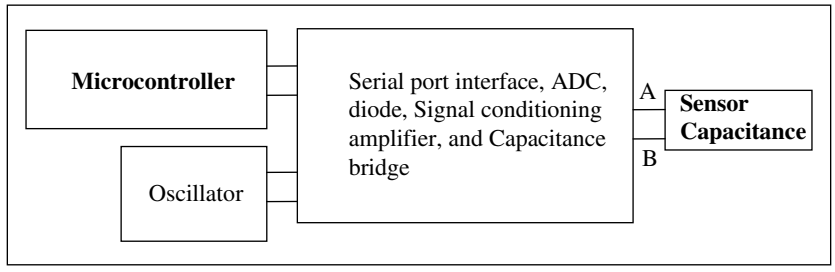


Figure 7.2 Microcontroller electronic circuit; port connected to sub-circuits; serial port interface, ADC, diode, signal conditioning amplifier and capacitance bridge, and sensor capacitance at A and B

Serial Port Interface

A serial port interface with the ADC has an advantage that the ADC 8 or 10 or 12 bit output is input to the interface, and the interface sends the input to the serial port at the microcontroller. Serial port interface has just two terminals in the output (Section 7.5).

Analog to Digital Converter

A microcontroller may consist of an in-circuit ADC or multiple inputs ADC. It processes the digital output from the in-circuit ADC. Alternatively, a port accepts the digital input consisting of 1s and 0s through an external ADC. An 8-bit port accepts the 8-bit input which corresponds to 0 to 255 decimal (255_d). The port

Analog sensor circuit connects to a signal conditioner amplifier, then to an analog to digital converter

accepts the 12-bit input in two cycles, viz. once 8 bits and then 4 bits which correspond to 0 to 4096 decimal (4096_d). The analog output from a signal conditioning amplifier connects to the ADC input, V_{in}.

The decimal value of ADC digital output relates V_{in} and is in the form of the binary bits at the input. The decimal value of binary bits is proportional to the ratio of analog input V_{in} and reference voltage V_{ref}. Assume a microcontroller functions with V_{DD}, the microcontroller power supply + ve input and V_{SS} microcontroller power supply -ve input.

Assume ($V_{DD} - V_{SS}$) = 5 V. Half of the reference voltage = ($V_{ref}/2$) is applied to ADC. Half reference voltage can be 1.65 V ($V_{ref} < 5V$. ($V_{ref}/2$) << ($V_{DD} - V_{SS}$).

Assume an ADC is of 8 bits. [$2^n - 1 = 255_d$] Then ADC digital output = $V_{in} \times 255_d / V_{ref}$, because V_{ref} is the maximum input which can be applied to the ADC which gives maximum digital output, 11111111_b, which means 255_d.

Sampling ADC

Sampling means that an ADC accepts input signals at specified periodic intervals and converts them into digits. The interval is set as per the signal frequency and other needs. The applications of sampling ADC are many. For example, while recording voice or music, the sampling ADC receives signals from the microphone for the recording sensor.

Signal Conditioning Amplifier

An SC amplifies the signal at the input as well as adds or subtracts an offset voltage in such a way that minimum V_{in} (min) and maximum V_{in} (max) values of the sensed physical parameter equal to 0 V and V_{ref}, respectively, at the SC outputs. Example 7.4 clarifies how to use the SC and ADC and compute the temperature.

Example 7.4

Problem

How are the voltage inputs to ADC and signal conditioning amplifier designed when using a sensor for temperature? How is the sensed temperature computed? Assume that the sensor measures temperatures between –10°C and +40°C.

Solution

Output of the signal conditioning amplifier (SC) in the circuit shown in Figure 7.1 is set such that the voltages which apply to ADC inputs are always between 0 V and V_{ref} (reference voltage input) at the ADC. Since temperature range is between (T_{min}) and (T_{max}), when temperatures to be sensed are between –10 °C (T_{min}), and + 40°C (T_{max}), then the ADC input = SC output should be 0V when sensor is at –10 °C and input to ADC should be V_{ref} when sensor is at 40°C. (T_{min}) and (T_{max}) difference is 50°C.

Software in the microcontroller uses a predefined table or equation to compute the sensed temperature, T. The table saves T = 40°C when ADC output is 255_d (= 11111111 binary). The sensed temperature is [50°C $\times$ $n/255$ –10°C], when the ADC output in decimal equals n. The sensed temperature is –10°C when the ADC output equals 0_d.

The same approach of using SC, ADC and computations are followed not only for temperature but also for other sensors, such as, strain, pressure, force, flex, vapours, magnetic field or proximity distances sensors.

Digital Sensors

Digital sensor uses the sensor and an associated electronic digital circuit to read 1s and 0s

A specific electronic component or circuit gives digital output 1 or 0 (on-off state) or output of 1s and 0s as a binary number (corresponding to a set of on-off states). A digital sensor uses the sensor and has an associated electronic circuit which gives digital output. The output 1 or 0 (1s and 0s) is read through a port in a microcontroller. This circuit can be used for sensing a sudden change in specific physical state or condition or can be used for sensing a sudden change in specific set of physical states or conditions.

Sensing of an On-Off State

A number of conditions needs detection using the concept of digital output of on-off state. Example 7.5 gives four cases on how to sense an on-off state, which means binary 1 and 0 output for reading by a circuit or microcontroler.

Example 7.5

Problem

(a) How is the switch on state sensed? How does one-bit digital output, 1 or 0 generate? (b) How does a streetlight sensor sense ambient light condition? (c) How does a rotating wheel state sense that it has reached at a specific orientation? (d) How does a linearly moving part state sense that it has reached at a specific linear position? Assume that the digital output is the input of a microcontroller port.

Solution

(a) Figure 7.3 shows that the circuit (top left) consists of one end A of the resistance connected to ground potential (negative end of the supply). The other end B connects to the end C of the switch. A switch can have two states on or off and gives two outputs 1 or 0. The other end D connects to the supply voltage V+ (= 5V) terminal. Microcontroller port pins connect C and A.

Alternative (top right) circuit is that base end, B of BJT which connects to one end of the switch. The emitter end, E of BJT connects the ground potential. The collector end, C connects to +ve end of supply Vcc (= 5V) through a resistance. The switch one end connects to the microcontroller port pin. Microcontroller port pin other end connects to C.

Another alternative (bottom middle) circuit is that switch one end B connects to MOSFET (Metal-Oxide Field Effect Transistor) source end S. The drain end D of MOSFET connects to the supply voltage + terminal, VDD through a resistance. The gate G connects to end A of the switch. The inputs at the microcontroller port pins are from B and D.

The circuits give 1s and 0s at port pins when switch state is off and on.

The right side top circuit voltage at collector C < 0.8 V is taken as 0 and > 2.8 V is taken as 1 and VCC is at 5V .

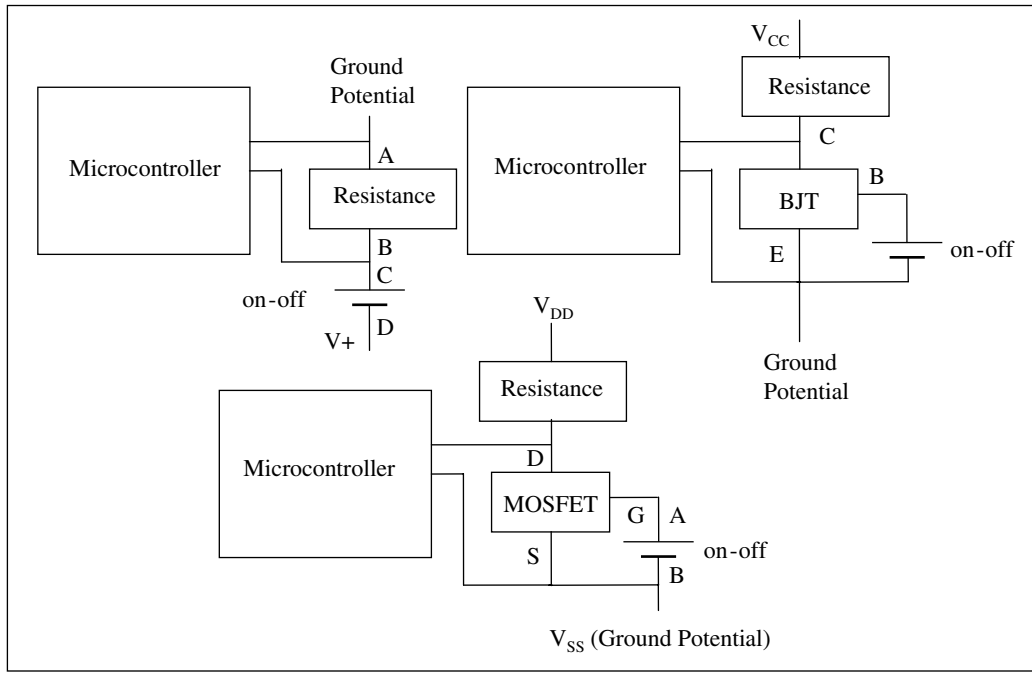


Figure 7.3 Three microcontroller electronic circuits for sensing On and Off states on generation of output 1 and 0 for the port pin

The middle bottom circuit voltage at drain D (with respect to supply terminal, VSS) less than (1/3) × (VDD – VSS) is considered as 0 and when greater than (2/3) × (VDD – VSS) that is taken as 1.]

(b) An environment ambient condition sensor at a streetlight senses the ambient light condition using a phototransistor, FPT. FPT detects the environment ambient light intensity above a threshold. Figure 7.4 shows a circuit. The circuit gives 1 in output when ambient light intensity is below threshold and gives 0 in output ambient light intensity is above threshold.

(c) A rotating wheel gives two outputs 1 or 0, when it has a pair of LED or IR-LED (light emitting diode or infrared LED) and phototransistor (FPT) is on two sides of a slot in the wheel. Figure 7.5 shows the circuit. The circuit gives 1 in output when a slot of rotating wheel reaches near the FPT on completing a rotation and gives 0 in outputs till that is revolving towards the completion of the next revolution. The light to FPT does not block till the wheel completes the revolution.

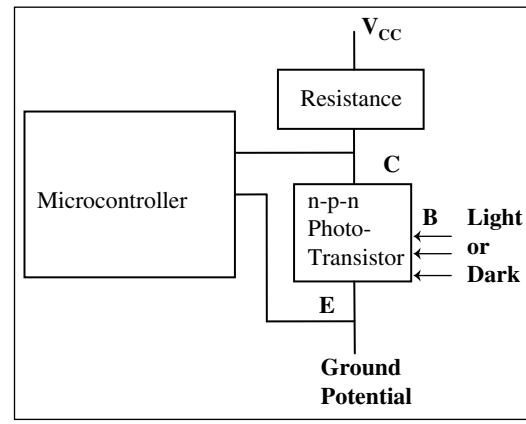


Figure 7.4 Microcontroller electronic circuit for a streetlight environment ambient condition sensor which senses two states and generates two outputs 1 or 0 for the port pin

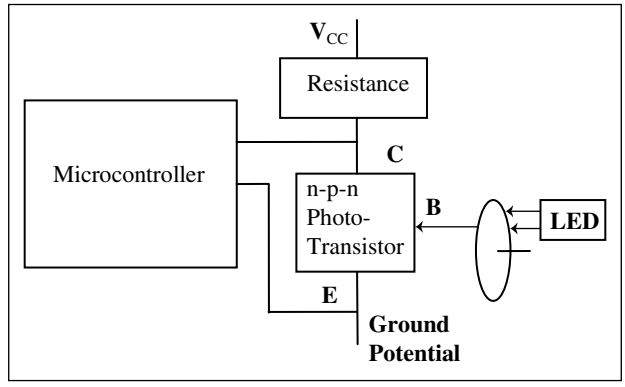


Figure 7.5 Microcontroller electronic circuit for a rotating wheel, rotation-completion sensing which senses two states (completion or incomplete) and generate two outputs 1 or 0 for the port pin

A Microcontroller port P_0 pin receives the input 1 or 0. Number of 1s within a specific time interval divided by the interval gives the rotational speed revolutions per minute (rpm). The values of the rpm and circumference of the tyre enable the computation of speed in km/hour in the automobile, as rotator motion converts to linear motion in the automobile.

(d) A moving part reaching at a specific location is sensed as follows: The part is attached with a light blocker. When the part reaches the empty space between a pair of LED (light emitting diode) and FPT, the blocker blocks the light falling on the FPT, and the circuit gives 1 in output. The circuit gives output = 0 when the part is not in the vicinity and is not blocking the light to FPT. Figure 7.6 shows the circuit.

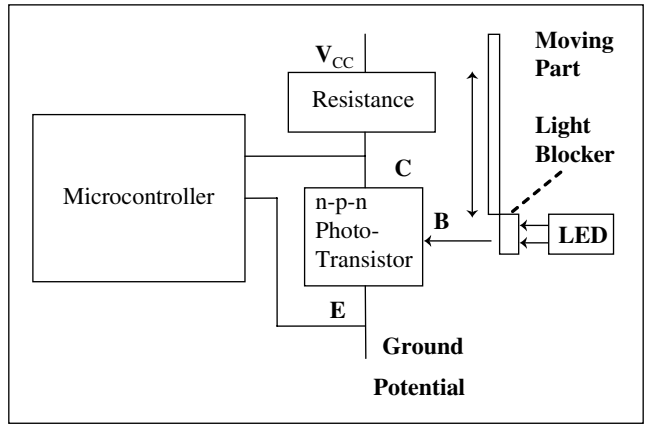


Figure 7.6 Microcontroller electronic circuit for sensing when a moving part reaches at a specific location and sensing two states (reached or yet to reach) and generate two outputs 1 or 0 for the port pin

A number of other applications exist for digital on-off sensing. For example, sensing of presence of traffic on a street, sensing the filling of a waste container up to a certain preset level that sends an alert on the Internet to the city waste management service, sensing the presence of organic vapours which generates an alert when sensed, sensing gas leakage or fire and generating an emergency alert.

Sensing a Set of On-Off States

Digital sensor circuit connects to the microcontroller port

A number of conditions together need detection in many applications. A circuit generates digital output for a set of On-off states. A specific electronic component or circuit gives digital output, such as, a set of 4 or 8 or 16 states consisting of 1s and 0s for sensing a set of discrete changes in a specific set of physical states or conditions. The output connects to a port input of a microcontroller, which reads the input at a given instance. Example 7.6 clarifies this.

Example 7.6

Problem

(a) How does an array of 8 switches generate a set of 1s and 0s? (b) How is the "communication of a number of chocolates for each type of flavours that remain unsold at each instance of 10% sale of each type" in Internet of Automatic Chocolate Vending Machines done? (c) How does the data of unoccupied parking slots be identified among the number of parking slots and used in Internet of Parking Spaces?

Solution

(a) An array of 8 switches generates 8-bit input for a port. An output port pin in a microcontroller sends a sense signal = 1 and the 8 return lines give the input to the microcontroller port. The 8-bit port reads the input to sense 8 on-off states of the switch. Figure 7.7(a) shows the circuit. When a keyboard is used, then eight sense lines and eight return lines are used to read the input and sense which key is pressed at a given instance.

(b) Assume five arrays of ten FPT-LED pairs, one for each 10% level in each chocolate filler assembly for each type of chocolate flavour. Five flavour assemblies mean fifty sensors. Each sensor generates an on state signal 1 for 10% filled level of unsold chocolate type. When a specific flavour sells 80%, then eight sensors give eight 1s and two 0s in an array. Total filler levels are 10 for each chocolate flavour. Total sum of 1s and 0s is 80_d. Five sets of five-filler assemblies for five chocolate flavours are read cyclically at the preset intervals. The data of all 50-filler levels is transmitted to the Internet and then a chocolate vending service application communicates regularly the presence of unsold chocolate percentages to the service. Figure 7.7(b) shows the circuit.

(c) Assume that there are 64 parking slots. A photoconductor as a sensor for laser diode light beam block detection can be used at each parking slot. Each sensor generates an on state signal 1 whenever a slot gets vacant. An array of eight slots is selected by three encoded lines from microcontroller. A set of eight-slots status in an array is read cyclically. Each array is cyclically selected. Data of all 64 slots is transmitted to the Internet and then a parking service application communicates regularly the presence of vacant slots to the vehicles passing through the traffic lights nearby to the parking facility. Figure 7.7(c) shows the circuit.

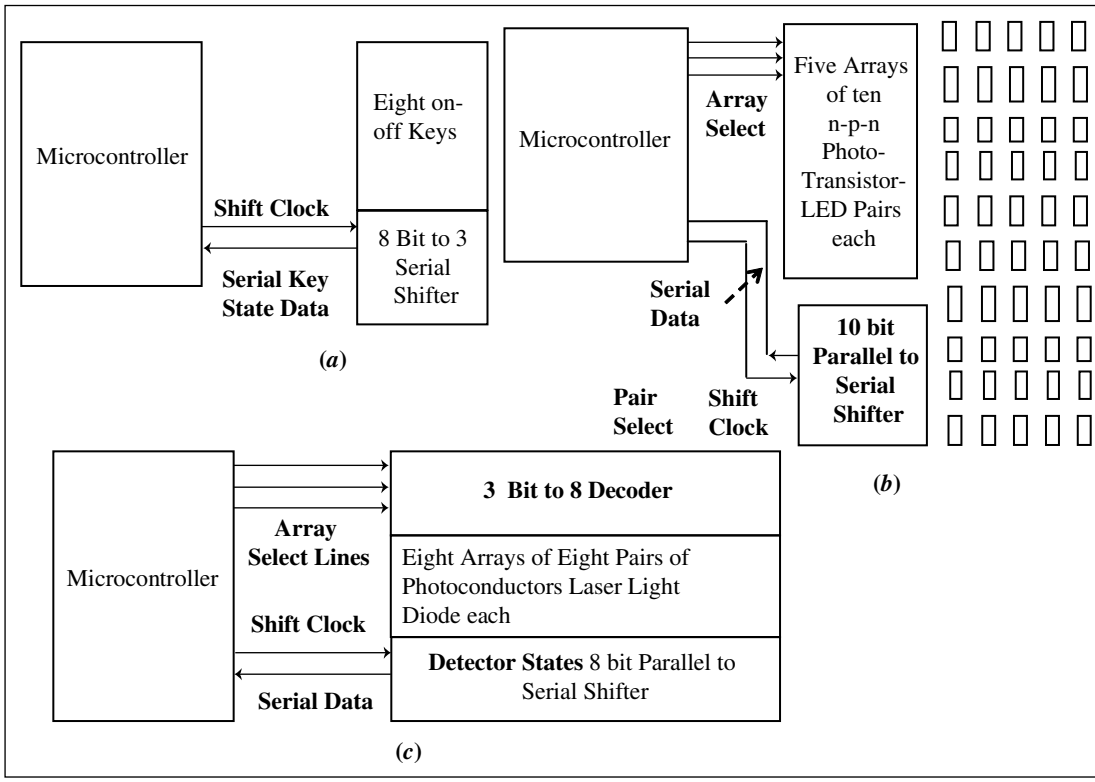


Figure 7.7 Microcontroller electronic circuits for sensing on-off states of (a) an array of 8 switches, (b) two-dimensional array of 50 sensors for 10 sales level when automatic chocolate vending machine sells 5 flavours, and (c) array of 64 sensors at parking slots with a sensor for each slot

7.2.2 Examples of Sensors

Temperature

A component called thermistor, shows larger changes in resistance within narrow environment temperature range (120°C to –90°C). An NTC thermistor shows negative temperature coefficient which means a drop in the resistance value with rise in temperature.

Thermistor finds applications in home automation or in sensing the clouds. The output of thermistor connects to the circuit of a signal conditioning amplifier, ADC and then to microcontroller serial port, similar to the circuit in Figure 7.1 except that in place of a resistance-bridge, a thermistor circuit is used.

A temperature sensor is called PTC, when it exhibits a Positive Temperature Coefficient. Resistance value of a PTC resistor rises with rise in temperature. A thin wire of platinum or other metallic alloys shows linear changes with its temperature. These can be used for sensing temperature and measuring the values over very wide ranges of temperatures, say (0–1600°C).

Example 7.2 described the measuring principle and procedure of using a resistance bridge, signal conditioning amplifier and ADC for generating input to a microcontroller. Figure 7.1 shows the circuit.

Certain ICs, such as AD590, function as temperature transducer and generate 1 μA for every 1°C rise in temperature. The output of the transducer connects to a signal conditioning amplifier, ADC and then to the serial port interface at the microcontroller. This circuit is similar to the circuit in Figure 7.1 except that in place of a resistance bridge, an IC transducer is used.

Readily available temperature sensors have in-built circuitry with three terminals, viz. two for 5 V or 3.6 V supply + and – terminals and one for ADC input, V_{in}.

Humidity

Use of capacitance changes for measuring RH%

Humidity is measured in percentage. It is the relative percentage ratio (RH%) of content of water vapours in air compared to one in a situation of maximum possible water vapour content for the air temperature at the instance of measurement. Greater than 90% humidity signifies it is a rainy day. A capacitor sensor shows change in capacitance as a percentage of relative humidity changes.

Readily available humidity sensors show output voltage proportional to RH% (Humidity sensor available from Sparkfun, an US company). The sensor is given input supply at + and – (ground) potential terminals and it generates output V_{RH} as a function of RH%.

Figure 7.2 shows a circuit for sensing using capacitive sensors, except that in place of capacitance bridge output, the sensor circuit is in-built in readily available sensors. V_{RH} is directly given to the ADC, and ADC output to the serial port interface at the microcontroller. The circuit can be used for measuring RH%. The computations give RH%.

Distance

Use of time intervals between IR pulses and reflected pulses for measuring the distance of an object

Infrared (IR) sensor is useful for a 0.15 m to 0.8 m range of object. IR sensor works on the principle that when a narrow beam IR LED sends radiation at an inclined angle, the nearby phototransistor FPT receives the reflected radiation after travelling two times the object distance. The reflected radiation delay (= 2 × 3.3 ns per m) between transmitted and reflected signal is proportional to the distance. The distance can be measured for object from 0.1 m to 0.8 m. Above 0.8 m, the reflected intensity may be insufficient for detection and below 0.15 m, the time interval is less than 1 ns, which inhibits the detection.

Readily available distance-based (IR) sensors shows output voltage proportional to distance (Sparkfun distance IR sensor). The sensor LED is given input supply at + and – (ground) potential terminals and IR-FPT along with internal circuitry generate output V_{dis} as a function of distance. V_{dis} is directly given as input to the ADC and ADC output to the microcontroller serial port. The computations give the distance.

Use of time intervals between ultrasonic pulses and their echoes for measuring distance from an object

Alternatively, ultrasonic sensors send the pulses. The frequencies of ultrasonic waves are of few kilocycles. The detection of echo from ultrasonic pulses and an associated circuitry generate a signal proportional to the distance. Ultrasonic wave delay is 2×3 millisecond/meter in air as the speed of sound in air is 330 m/s. Long-range distances and any obstacles nearby can be detected using ultrasonic sensors. These sensors are used in industrial automation, rail tracks and oil pipeline faults.

Light

Use of photoconductor, p-n junction photodiode or phototransistor for measuring the intensity of light

Example 7.1 shows that a photoconductor can be used to detect light in the vicinity. The sensor shows a drop in resistance with surrounding light. Alternatively, the p-n junction photodiode or phototransistor can be used to measure incoming radiation intensity incoming from a particular direction. Figure 7.1 shows that the output of the sensor circuit connects the signal conditioning amplifier, ADC and microcontroller.

Acceleration

Use of MEMS for measuring the acceleration components in three axes by capacitive variations in three axes

A Micro-Electro-Mechanical Sensor (MEMS) detects linear accelerations a_x, a_y and a_z along three axes x, y and z, respectively. An MEMS moves when a mass moves along a direction. A mechanical movement has three components. The variations cause the variation in three capacitance values, C_x, C_y and C_z. The value of each C depends on the space between two plane surfaces, which varies on acceleration along an axis. These capacitances are part of an electronic circuit and the resulting voltage variations give the a_x, a_y and a_z.

An accelerometer sensor is used in new generation mobile phones. The display screen image and menu items rotate and align horizontally or vertically on detecting the three components using the sensor when it rotates along with the phone. The accelerometer also detects up/down, right/left and front/back accelerations given to the device by the user.

Sparkfun ADX335 is a readily available accelerometer sensor. The accelerometer is given input supply at + and – (ground) potential terminals and it generates three outputs, V_x, V_y and V_z as a function of time. The computations give a_x, a_y and a_z. Figure 7.8 shows the accelerometer circuit.

Vibrations and Shocks

MEMS measures the vibrations and shocks by changes due to piezoelectric effect

Alternatively, MEMS may use piezoelectric effect in place of capacitive change effects. The effect observed in certain specific materials is accumulation of electric changes on surfaces due

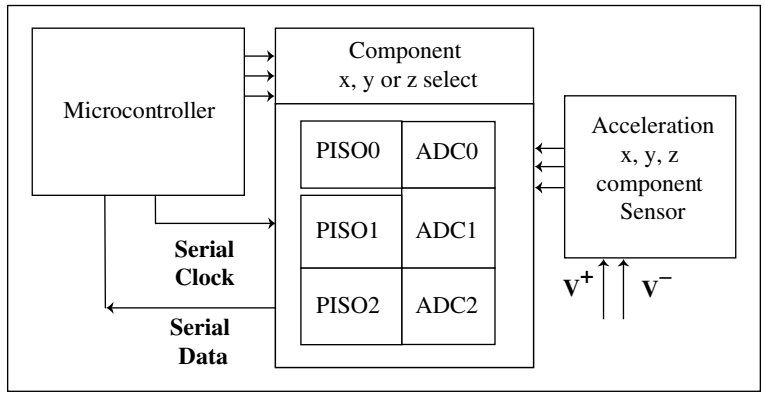


Figure 7.8 Microcontroller electronic circuit for measuring three acceleration components from the accelerometer sensor

to mechanical compression of the piezoelectric material. The rate of change of number of charges with time implies a flow of current. Vibrations create repeated compression and decompression. An associated electronic circuit generates output according to the intensity of vibrations. The circuit also senses the mechanical shocks. A user initiates the vibrations or shocks, or the device shakes when it falls, then the in-built sensor in the mobile senses these changes and the system takes action as programmed.

Angular Acceleration and Change in Direction (Angle)

Gyroscope is a sensor which measures the change in angular velocity (angular acceleration) and the change in direction (angle). An application takes measurements using a gyroscope or accelerometer and the system initiates actions as programmed. For example, mobile gaming application uses in-motion gestures of a player when deploying a gyroscope.

Orientation and Direction Compass

A gyroscope can be used as an electronic compass or a digital compass as it shows the change in direction (angle).

Alternatively a digital compass can also be used. A compass shows the directions—North, South, East and West. It also shows the direction in which an object is inclined. The compass is a very simple device used for navigation. It has a magnetic strip which aligns towards the Earth's magnetic north under the influence of the Earth's magnetic field. The compass shows by how many degrees ϕ the device's *north* is inclined from the actual Earth's magnetic north in a clockwise direction. Mobile device's *north* means the screen's upper direction, *south* the lower, *west* the left and *east* the right.

A digital compass is an electronic device used to detect the orientation or direction with respect to the *north* in degrees. The compass gives a digital output in terms of a sequence of 1s and 0s. A readily available digital compass sensor sends 1s and 0s to the

microcontroller serial port. The sensor is given input supply at + (V_{CC}) and – (ground) potential terminals and it connects serial-port terminals, SCL (Serial Clock) and SDA (Serial Data) (Section 7.5). The sequence of serial bits corresponds to a byte (8-bit binary number). For that number computation give angle ϕ with respect to the north direction.

Magnetic Sensors/Magnetometer

Example 7.7 explains sensing principle and procedure.

Example 7.7

Problem

How does a magnetometer show the direction and changes in the magnetic field?

Solution

A magnetometer present in a device enables three-dimensional interactions between a tiny magnet M1 in the device and a nearby iron magnetised piece M2 without touch. It uses the magnetic field created in the environment of the device. The magnetometer is also used as an orientation, proximity and distance sensor for iron or steel objects in industrial automation.

One application also identifies presence of iron or steel objects and switches off the phone upon detection. Another application monitors the changes in magnetic fields of M1 and identifies the gestures of the user.

Electric Current

Alternating Current (AC) is detected by a miniature transformer and its associated circuitry. A Direct Current (DC) flows in one direction at all instances. It detects using a sensor circuit, which detects the magnetic field by flowing current. Readily available electric current i and voltage v or power (product of i and v) are used. They can connect to the microcontroller using its associated circuitry. The computation can be done in the microcontroller. A wireless transmitter can transmit the data through Wi-Fi to the utility company.

Sound

A microphone is used to sense sound. A readily available electronic board with a microphone connects to the microcontroller, which can control an actuator for actions based on the sensed sound, or recognise the voice and then take required action, such as dialing a number using the actuator circuit or switching on the car.

Sensing the Things

Reading Barcodes

Bar coding uses 1D or 2D code

A barcode is a representation of data. The data relates to the object where the printed code strip is attached. The code is read by an optical scanner. Earlier a barcode would systematically

represent data by varying the widths and spacing of parallel lines. The code was a linear or one-dimensional code (1D). Later, barcodes evolved into rectangles, dots, hexagons and other geometric patterns in two dimensions (2D). The 2D system use symbols which are also referred to as barcodes.

A barcode reader is a scanner for a printed code called the barcode. An electronic device reads the output for a port of microcontroller or computing device or computer. The scanner has a light source. When it is switched on, the light impulses pass through a lens and focuses on the black and white spaces of the barcode. The light source can be laser based or LED based.

Reflected light sensor or CCD (Charge Coupled Device) detector at the scanner along with an associated decoder circuit converts the optical impulses into electronic pulses and analyses the barcode's image data. The resolution commonly used is of dimension 0.33 mm of the printed code. The sensor sends the contents of the barcode as 1s and 0s at the input port of the computing device.

QR Code

QR code is an abbreviation for Quick Response Code. It was first used in automotive industry. Its applications are product identification, tracking, marketing and document management.

The QR code uses standardised encoding modes, such as numeric, alphanumeric, byte/binary or other. The code stores the data efficiently and is extendable. It is now popular in industries other than the automotive industry. It is read faster and the data stored is more than that using a standard Universal Product Code (UPC).

The QR code consists of black square dots arranged in a square grid format on a white background. The required data is at patterns in both horizontal and vertical components of the image. A scanner or camera reads the code and the data is processed using an error-correction method called Reed Solomon method. The processing takes place till the process results in appropriate interpretation of the data.

Motion Sensors for Moving Objects

Motion or speed is measured in m/s. The sensor measures delay between successive reflected IR light pulses. An LED source is an IR light source and a phototransistor is an IR sensor. Alternatively, ultrasonic wave echoes can be used to sense the motion of light. The sensor measures the delay between successive echoes.

Readily available motion sensors, such as Sparkfun sensor show output voltage proportional to motion. The sensor is given input supply at + and – (ground) potential terminals and it generates output V_m as a function of m/s.

The computations give variation in the speed of nearby objects and also show vibrations. Security systems use the motion sensors to communicate data on the Internet and raise security alarms.

Pressure Sensors

Pressure sensors use the resistance changes due to strain. Alternatively, they use piezoelectric effect

Pressure P is measured as force per m^2. Pressure can be sensed in a number of ways. The sensor is called pressure transducer, pressure transmitter, pressure sender or pressure indicator. Piezometer pressure transducer uses a piezoelectric object between two surfaces. The compression creates electric charges on the opposite surfaces of the object. The flow of charges generates current and voltages, which provide the measure of pressure.

A resistive sensor also measures variation of resistance with force. The force creates stress and thus the strain.

Readily available pressure sensors, such as Sparkfun sensor show output voltage proportional to pressure. The sensor is given input supply at + and – (ground) potential terminals and it generates output V_P as a function of P.

Application of pressure sensor can be explained as, a tyre pressure monitoring system uses the pressure sensors on each tyre. The sensors communicate the tyre pressure in each tyre, and a corresponding monitoring circuit sends alerts on the dashboard of the vehicle.

Figure 7.1 shows the circuit for sensing using resistance sensors, except that in place of resistance bridge output, the sensor circuit is in-built using readily available sensors. V_P is directly given to the ADC, and ADC output to the serial port interface at the microcontroller. The circuit can be used for measuring P. The computations give the pressure, P.

Environmental Monitoring Sensor

Environment parameters are temperature, humidity, barometric pressure and light. A collective use of these parameter sensors enable monitoring of the environment. The data of these sensors adapts to the requirement and sends communication on the Internet to the cloud or web for the environment monitoring applications. For example, light environment on the streets monitors the lights (Examples 1.2 and 5.5).

Location Data

Determining location of an object means finding its distance from several fixed locations which are in multiple directions and also measuring the intensity of light or IR or ultrasonic waves enables computations for a location, in case the source location intensities and attenuation per metre is known.

GPS

Location determination can be done using a Global Positioning System (GPS), also known as Geographical Positioning System. A user can receive the location from a service provider. The service provider finds the GPS location through signals from satellites. The service provider finds the user location with respect to the service provider through its base stations.

Camera

Camera is an image sensor. The camera uses CCD, which consists of a large number of pixels, exposed to the light from the image. It accumulates charges on each of the pixel present at a large number of horizontal and vertical coordinates. The charge accumulation is as per the intensity of light at the corresponding pixel coordinate in the image. Coloured camera has set of R, G and B (Red, Green and Blue) light intensity components at each pixel coordinate.

The camera generates a file from the R, G and B intensities at each image pixel. The file gets saved in the memory after compression in jpg or gif or any other standard format. A computer communicates the saved file to the Internet for applications such as industrial IoT, home security systems, ATM security systems, automotive IoT or participatory sensing IoT.

LIDAR

LIDAR (Light + Radar) [Laser Imaging, Detection and Ranging] sensors and laser 3D imaging technology enables remote sensing and imaging. It finds distances by throwing light using laser on target. The sensor senses the reflected light which enables computations of distance.

Laser 3D Imaging

3D imaging is feasible using laser 3D imaging technology. The technology uses both scanning and non-scanning systems. 3D gated viewing laser radar is a non-scanning system using pulse laser and fast gated camera.

Reconfirm Your Understanding

- Sensors can sense temperature, humidity, acceleration, angular acceleration, object distance, orientation angle with respect to a fixed direction, magnetic object proximity, touch and gestures of users, motion, sound, vibrations, shocks, electric current and environment conditions.
- Many IoT applications need data generated from sensing devices. A sensor converts physical energy like heat, sound, strain, flex, pressure, vibrations and motion into electrical energy. An electronic circuit is associated with sensor. The circuit receives the output from the sensor. The output is according to the variation in physical parameters or state.
- Sensors are of two types—analog and digital.
- A circuit parameter can be used as a sensor when it shows measurable variation with heat, sound, strain, pressure, vibrations and motion.
- A complex sensor includes the output processing circuit and computing and communication capabilities. Digital sensing needs a circuit to generate 1 or 0 or binary output of 1s and 0s for storing, computing and communication device. A sensor can use a phototransistor-LED pair and digital circuit to generate 1s and 0s.
- A set of digital sensors sense the number of chocolates of each type of flavour remaining unsold and communicate in IoT application of automatic chocolate vending machines (Example 5.5).

- Digital sensors sense unoccupied parking slots and digital output identifies the unoccupied slots among the number of parking slots. These are used in Internet of Parking Spaces. Sensor uses photoconductor-light beam pair and digital circuit.
- MEMS is a microelectromechanical sensor that detects linear accelerations a_x, a_y and a_z along three axes x, y and z, respectively.
- Things get sensed and identified by barcode readers or QR codes, and associate circuit communicates the object identity and object document information to the Internet.
- The identity, location, document information, pressure, motion, environment parameters and other characteristics data communicate to the Internet for IoT applications.

Self-Assessment Exercise

1. What are the characteristic parameters which change with physical environment and therefore are used for sensing applications? ★
2. What are the circuits for measuring variation in resistance with respect to temperature? Compare these and describe their application. ★★
3. Draw a circuit for measuring variation in capacitance with respect to an organic solvent vapour concentration using a microcontroller? ★
4. When do you use an analog sensor and a digital sensor? ★
5. What is a smart sensor? What are the capabilities of a smart sensor? ★★★
6. Why and when is ADC required? What does a signal-conditioning amplifier require before the ADC? ★★
7. How do you measure temperature in range –20°C and +100°C with a resolution of 1°C? ★★★
8. Tabulate the sensors for temperature, level of filling in a container, pressure, tyre-pressure, humidity, ambient light intensity, traffic in proximity, acceleration, distance of in-front object, location with respect to fixed locations, object proximity, metallic object proximity, iron object proximity, orientations with respect to north, sound, vibrations and empty parking spaces and their characteristic parameters which enable sensing the changes and their values? ★★
9. How can a sensor for sensing four states of filling a waste container, viz., below 80%, up to 80%, up to 90% and full 100%, be designed for a city waste management service? ★★★
10. What is an MEMS? What are the physical entities which are sensed using an MEMS? ★★
11. What is piezoelectric effect? What are the physical entities which are sensed using piezoelectric effect? ★★

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

12. How is sound sensed and voice recognised? ★
13. How do you sense the things and identify product information using a barcode reader? What are the applications of IoTs when using barcode sensors? ★★
14. What is QR code? How do you use QR code in the applications of IoTs? ★★
15. How is the motion of a moving object sensed? ★

7.3 PARTICIPATORY SENSING, INDUSTRIAL IoT AND AUTOMOTIVE IoT

LO 7.2 Define the concepts of participatory sensing, Industrial IoT and Automotive IoT

Information collected from sensors of multiple heterogeneous sources can lead to knowledge discovery after analytics and data visualisation. A web source defines Participatory Sensing (PS) as "sensing by the individuals and groups of people contributing sensory information to form a body of knowledge". Deborah Estrin, University of California, Los Angeles now at Cornell University, defines participatory sensing as, "Participatory sensing is the process whereby individuals and communities use evermore-capable mobile phones and cloud services to collect and analyse systematic data for use in discovery."

A participant of a PS process can be sensors used in mobile phones. Mobile phones have camera, temperature and humidity sensors, an accelerometer, a gyroscope, a compass, infrared sensors, NFC sensors, bar or QR code readers, microphone and GPS. Mobiles communicate on the Internet the sensed information with time, date and location stamps.

Applications of PS include retrieving information about weather, environment information, pollution, waste management, road faults, health of individuals and group of people, traffic congestion, urban mobility, or disaster management, such as flood, fire, etc. Participatory sensing has many challenges such as—security, privacy, reputation and ineffective incentives to participating entities.

Figure 7.9 (a) shows the sources of data in the PS process for IoT applications. Figure 7.9 (b) shows the phases of a PS process. Phase 1 is coordination, in which the participants of a PS process organise after identifying the sources. Next two phases, i.e. phases 2 and 3 involve data capture, communication and storage on servers or cloud. Next two phases, i.e. phases 4 and 5 involve PS data processing and analytics, visualisation and knowledge discovery. Last phase, i.e. phase 6 is for initiating appropriate actions.

7.3.1 Industrial IoT

Industrial Internet of Things (IIoT) involves the use of IoT technology in manufacturing. IIoT involves the integration of complex physical machinery M2M communication with the networked sensors and use of software, analytics, machine learning and knowledge discovery. Example of the functions of IIoT are refining the operations for manufacturing or maintenance, or refining the business model of an industry.

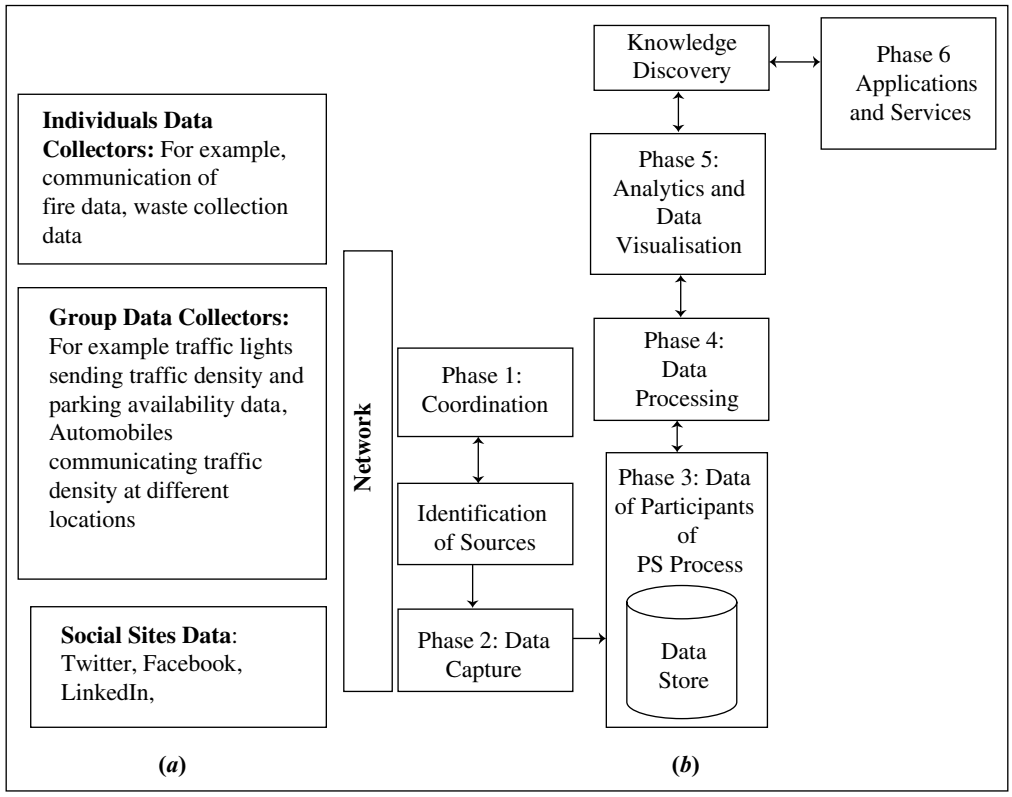


Figure 7.9 (a) Sources of data in the PS processes and (b) Phases of a participatory sensing process for IoT applications and services

IIoT applications are in the manufacturing, railways, mining, agriculture, oil and gas, utilities, transportation, logistics and healthcare services.

Example 7.8 explains use of IoT technologies in manufacturing and industrial processes.

Example 7.8

Problem

How is IIoT technology used in optimising the bicycle manufacturing process?

Solution

The sensors at each manufacturing stage in a bicycle industry communicate information on completion at each stage for each bicycle. An IIoT application analyses that data on completion of each activity at each stage, including data of breakdowns, work stoppages and failures at the stages. The application enables the company to take steps and synchronises various actions to remove any bottlenecks due to the components supply or the manufacturing stage machinery or human failures. Bicycle manufacturing is thus optimised. Figure 7.10 shows IIoT phases in the bicycle manufacturing process.

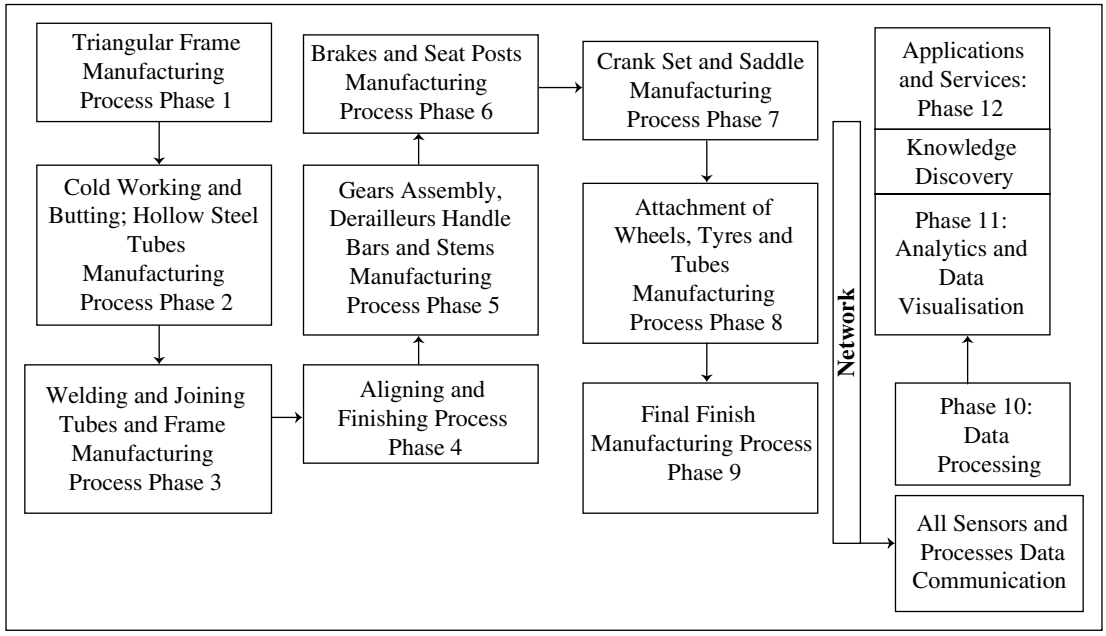


Figure 7.10 IIoT phases in the bicycle manufacturing process

Example 7.9 elucidates IIoT application in predictive maintenance of industrial processes.

Example 7.9

Problem

How is IIoT technology useful in predictive maintenance of industrial processes?

Solution

Consider an application for predictive railroad service centre. Ultrasonic sensors, infrared temperature sensors and microphone sensors along railway tracks sense and communicate the captured data for each train passing through each segment track. The application predicts the failures. This enables undertaking of preventive maintenances.

Similarly, IIoT finds applications in predictive maintenance of aircraft parts, gas pipelines and machines used in production.

Service-Oriented Cross-Layer Infrastructure for Smart Embedded Devices (SOCRADES) project developed an integration architecture which integrates shop floor industrial machines with enterprise systems. Three-level architecture has three levels—device management, service management and application interface for the systems.

Example 7.10 describes software for IIoT and M2M applications.

Example 7.10

Problem

How is IIoT technology useful in predictive maintenance of industrial processes?

Solution

- GE Industrial Analytics Software Predix provides an IIoT platform. The platform provisions for sensor-based computing and predictive analytics.
- An advanced cloud-based service and software from Axeda Company manages connected products and machines. The software enables implementation of innovative M2M and IoT applications.
- OSI soft is software for real-time data management for sensor-driven computing. The sensors data is from the manufacturing processes and utilities companies such as electricity, phone and mining.

Industrial Internet Consortium (2014) is body which has been founded for creation of standards, open interoperability and the development of architectures for Industrial Internet of Things (IIoT).

7.3.2 Automotive IoT

Automotive IoT enables the connected cars, vehicles-to-infrastructure technology, predictive and preventive maintenances and autonomous cars.

Connected Cars Technology

Automotive vehicles can drive through roads with little or no effort at all. A connected car with the combination of GPS tracking and an Internet connection enables applications such as:

1. Display for driver that enables driving through the shortest route, avoiding the congested route, etc.
2. Customisation of functioning of the vehicle to meet the driver's needs and preferences
3. Get notifications about traffic
4. Protecting cars against theft
5. Weather and enroute destinations
6. Keeping a tab on driver's health and behaviour.

Vehicle-to-Infrastructure Technology

Automotive IoT enables Vehicle-to-Infrastructure (V2I) technology. A vehicle communicates with other vehicles, the surrounding infrastructure and a Wi-Fi LAN. Examples of V2I applications are:

- Alerts and warnings for forward collision
- Information about blind spots
- Notification about a vacant parking space
- Information about traffic congestion on route to destination
- Stream live music and news.

Predictive and Preventive Maintenances

Example 7.11 shows automotive IoT application for predictive and preventive maintenances of automobiles by service centre application.

Example 7.11

Problem

How is an automotive IoT technology useful in predictive maintenance of an automobile by a service centre application?

Solution

Consider Internet of connected automotive components (Example 5.2). A number of sensors for statuses and conditions of components are used. Examples are engine movements unit, axle, steering unit, brake linings, wipers, air conditioners, battery, tyre movements, coolant and shockers. The statuses and conditions data are needed for predictive maintenance. A component embeds computing hardware and for ultrasonic sensors, IR sensors, sound sensors, seat alignment sensors, height sensors, driver acceleration sensors at start, during running and driver braking characteristics and road friction, and microphone sensors. The sensors capture the data for noise, vibration and harsh driving and actions of the vehicle.

The sense data communicates in real-time or stores and transmit when the automobile reaches a Wi-Fi node. The service centre application schedules maintenance alerts and predicts the failures and alerts for the actions. Figure 7.11 shows Internet of connected car components for predictive and preventive maintenances of automobile by a service centre.

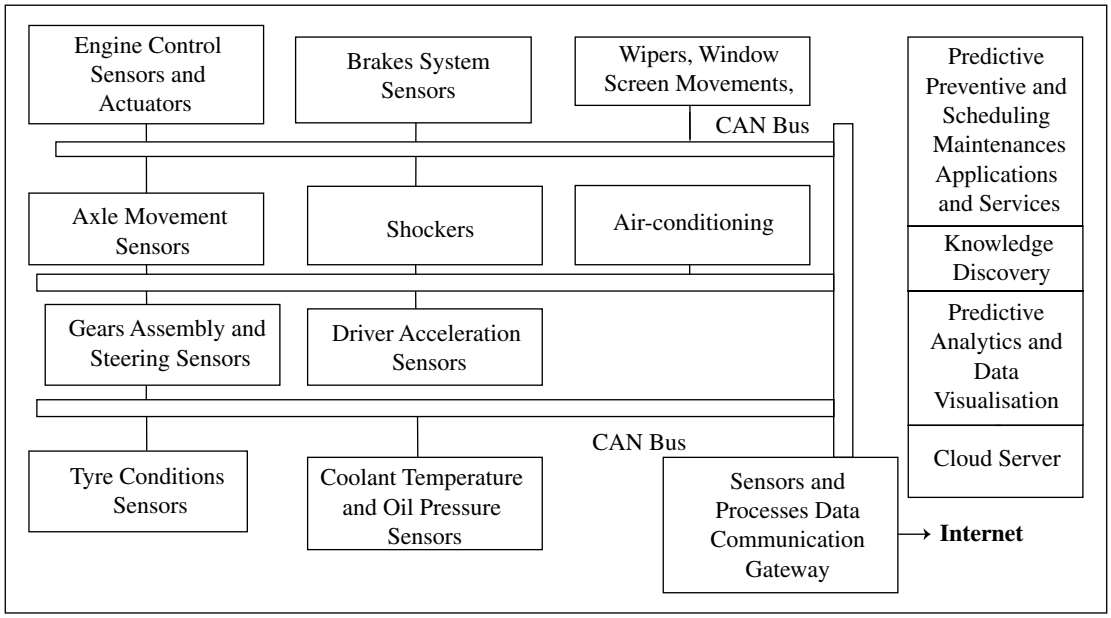


Figure 7.11 Internet of connected car components for predictive and preventive maintenances of automobile by service centre

Autonomous Cars

Driverless cars (also known as autonomous cars or robotic cars) have become a reality. These deploy LIDAR and laser 3D imaging technology.

Reconfirm Your Understanding

- Participatory sensing (PS) is sensing by individuals and groups of people contributing sensory information to form a body of knowledge.
- Applications of PS are many, such as retrieving information about the weather and environment, pollution, waste management, road faults, individual and group of personals health, traffic congestion, urban mobility and disaster management, such as flood, fire.
- Industrial Internet of Things (IIoT) is the use of IoT technologies in manufacturing and predictive maintenance. IIoT involves the integration of complex physical machinery M2M communication with the networked sensors, and use of software, analytics, machine learning and knowledge discovery.
- Automotive IoT enables connected cars, vehicle to infrastructure technology, predictive and preventive maintenances and autonomous cars.

Self-Assessment Exercise

1. List the merits of participatory sensing. ★
2. How does PS enable traffic congestion reports? Show an architectural diagram of the process. ★★★
3. List the applications of industrial IoT. ★
4. How is industrial IoT used for predictive maintenance of machines in the industry? ★★★
5. List the sensors needed in a car for automotive IoT. ★
6. Draw an architectural diagram of the processes in automotive IoT for predictive and preventive maintenance. ★★

7.4 ACTUATOR

LO 7.3 Describe the uses of actuators in devices

An actuator is a device that takes actions as per the input command, pulse or state (1 or 0), or set of 1s and 0s, or a control signal. An attached motor, speaker, LED or an output device converts electrical energy into physical action. Examples of applications of actuators are:

- Light sources
- LEDs

- Piezoelectric vibrators and sounders
- Speakers
- Solenoids
- Servomotor
- Relay switch
- Switching on a set of streetlights
- Application of brakes in a moving vehicle
- Ringing of alarm bell
- Switching off or on a heater or air-conditioner or boiler current in a steam boiler in a thermal plant.

Light Source

Traffic lights are examples of function of light sources as actuators controlled by the inputs.

LED

LED is an actuator which emits light or infrared radiation. Uses of different colour LEDs, RGB (Red-Green-Blue) LEDs, intensity variation of LED and colours, graphic and text display using big screens are actions which are controlled using the inputs. RGB LED has three inputs to control, i.e. R, G and B components and thus the composite colour. Pulse width modulated pulses control the LED light emission intensity. A microcontroller is used for generating PWM outputs.

Piezoelectric Vibrator

Piezoelectric crystals when applied in varying electric voltages at the input generate vibrations.

Piezoelectric Speaker

A piezoelectric speaker enables synthesised music tunes and sounds. The appropriately programmed pulses generate the music, sounds, buzzers and alarms when they are the input to the speaker. A microcontroller is used for generating PWM outputs for actions using speakers.

Solenoid

A solenoid is an actuator consisting of a number of cylindrically wound coils. The flow of current creates a magnetic field in proportion to the number of turns in the solenoid and the current in it. If a shaft made of iron is placed along the axis, then its motion can be controlled by the input current, pulses and variations of current with time. It can create a sharp forward push, backward push and repeated to and fro motion. It can also create rotator motion from linear motion by using a Cam.

A cam with linear shaft assembly is used in an engine to convert rotatory motion into linear motion and vice versa. A cam is an especially cut mechanical rotating object such

that the radius linearly increases (from the centre) between 0° and 180° rotation and decreases between 180° and 360°. When a cam assembly rotates, then a linear shaft moves in forward and backward motions in each rotation.

Motor

A motor can be DC (direct current controlled) or AC (alternating current controlled). IO modules are readily available to receive the control digital inputs of 1s and 0s and deliver high currents. The dc or ac rotates the motor. A cam also converts rotator motion into linear motion when it rotates using a motor.

Servomotor

Servomotor is a geared DC motor for applications such as robotics. It rotates the shaft of a motor. The shaft of the motor can be controlled and positioned or rotated through 180° (+90°) degrees. The shaft's angular position is controlled through 180°, between –90° and +90° degrees.

Example 7.12

Problem

How does angular position in a servomotor shaft control using input pulse widths?

Solution

A servomotor has just three terminals, viz. two for voltage supply + and – terminals and one for a pulse width modulated (PWM) input. A servomotor is conveniently controlled by PWM. The motor has three terminals. Two are for the supply voltage and ground. A position control PWM signal to the motor is given at the third terminal.

Figure 7.12 shows an interface circuit for controlling servomotor angular position. The pulse repeats 50 times (means at 20 ms intervals) to the motor input. The input is through a PWM control output at a port pin of microcontroller. It controls the movement as well as the angle of a servomotor between –90° and +90°.

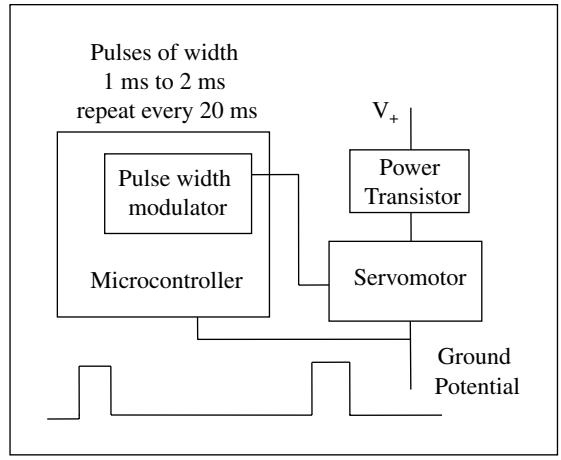


Figure 7.12 Microcontroller circuit for controlling servomotor angular position

The microcontroller has an internal circuit to generate pulses of different widths as per value, *p* loaded in the PWM register.

When the port output PWM pulse is of width 1.5 ms, then it places the shaft at a mid-angular position. Let us assume that when the pulse width is 1 ms, the angle is –90°. When the width is 2 ms, then the angular position is +90°. Varying the width, *pw* ms between 1

and 2 ms using the PWM register value will change the angular position, $\phi = [(pw - 1.5) \times 180°]$ When $pw = 1.5$ ms, then, $\phi = 0°$ (mid angular position), when 1 ms, then –90°; when 2 ms, then +90°.

Relay Switch

An electronic switch can be controlled by the input 1 or 0 from the port pin of a microcontroller or through a push button switch and battery. The current flows through the switch or voltage applies through the switch depending upon the input state 1 or 0.

A relay switch makes mechanical contact when the input circuit magnetises with a control circuit and pulls a lever to make the contact. The current flows through the switch or voltage applies through the switch depending upon the input state 1 or 0 from the port pin of a microcontroller or through a push-button switch and battery.

Reconfirm Your Understanding

- An actuator is a device which takes actions as per the control inputs or command using 1s and 0s, pulse or state (1 or 0) or set of 1s and 0s or width of the input pulses of 1s and 0s.
- Examples of actuators are light sources, LEDs, piezoelectric vibrators and sounders, speakers, solenoids servomotor, relay switch, switching on a set of streetlights, application of brakes of a moving vehicle, ringing of alarm bell, and switching off or on a heater or an air-conditioner or a boiler current in a steam boiler in a thermal plant.

Self-Assessment Exercise

1. List uses of LEDs and RGB LEDs in control and monitoring using IoT. ★
2. How does a piezoelectric vibrator function? ★★
3. How does a solenoid enable linear and angular motions? ★★★
4. How can servomotor find use in automotive IoT? ★

7.5 SENSOR DATA COMMUNICATION PROTOCOLS

LO 7.4 Describe the uses of data communication using serial bus protocols

A serial interface uses a protocol for serial communication. A microcontroller includes interfaces for serial communication. UART and several other protocols are popularly used. Wired communication can be:

- Serial asynchronous communication, such as, using UART communication. A RFID reader uses a 125 kHz RFID UART module. A GPS device also sends serial data using the UART.

- Synchronous serial-communication devices, such as sensors which can communicate serial data using I2C or SPI interfaces in wired bus communication.

Automotive sensors communicate serial data using LIN, CAN, MOST, IEEE 1394 serial protocols (Section 2.3.2).

7.5.1 Serial Bus

A bus gives the following advantages[1] in sensor circuits and connecting devices:

1. Simplifies number of interconnections compared to direct connections between them
2. Provides a common way (protocol) of connecting different or same type of I/O devices
3. Can add a new device or system's interface that is compatible with a system's I/O bus
4. Provides flexibility, allowing a system to support many different I/O devices depending on the needs of its users and allowing users to change the I/O devices that are attached to a system as their needs change.

Widely used communication protocols for serial bus communication are Inter-Integrated Circuit (I2C), Universal Serial Bus (USB) and FireWire (is now a standard and known as IEEE 1394).Widely used serial bus communication protocols for serial communication in automobiles' internal sensors and other embedded circuits are Local Interconnect Network (LIN), Controller Area Network (CAN) and Media Oriented System Transport (MOST).[2] A serial interface communicates serially with another device serial interface for:

- Sending the sensor data on a network, for example in an automobile.
- Receiving commands to control the actuator actions through the microcontroller serial output.

A microcontroller program prepares the interface. Preparation means configuring the device for serial interfacing. The configuration also sets the data rate at the beginning of the data receive or send program. A program reads the bytes from the serial port inputs, and sends the outputs at serial or parallel ports.

UART for Serial Bus

A UART device sends 8-bit data at successive intervals, called baud intervals. A start bit precedes the data (characters). Start bit means 1 becoming 0 for certain interval, called baud interval before the 8-bit data. A stop bit is 1 for a minimum interval equal to baud interval. The bit succeeds the 8-bits data which means 10 bauds per character. Each digit or command communicates 8 bits. Each character communicates 8 bit and is coded as per the ASCII (American Standard Code for Information Interchange) code.

Software Serial Library

Integrated Development Environment (IDE) for a microcontroller system provides a software serial library. A library consists of number of programs. A software serial library

[1] http://www.webopedia.com/TERM/B/bus.html and https://en.wikipedia.org/wiki/Serial_communication

[2] Raj Kamal, *Embedded Systems- Architecture, Programming and Design* 3/e, McGraw-Hill Education, New Delhi, 2014, pp.169-174.

has distinct programs for each serial interface protocol. A program enables a user to directly use a protocol. For example, a library program is used for reading an RFID tag. Another can be used for sending data to a USB port. A USB port is used for onward transmission to the Internet.

Using UART Communication for a RFID Tag

A header character is sent before the tag. A tag ID has ten digit characters. An end character consists of 1 byte. It succeeds the 10 digits of the tag. The total number of digits communicated is equal to 12.

Reciprocal of baud interval is called baud rate. The 125 kHz corresponds to 125 kbaud/s (kilo bauds per second) which means 12.5 k characters per second. 'Baud' is a German word that means *drop,* such as raindrop. Bits in UART protocol communicate like bauds.

A library program for RFID using UART serial interface protocol enables its direct use for an RFID tag.

Using the I2C protocol for a Serial Bus

Integrated circuits, sensors or actuators can be interconnected with the serial synchronous interface, I2C. A library program for I2C serial interface protocol enables direct use of a sensor IC. I2C bus means different integrated circuits using I2C interface communication over the same set of wires.

I2C is inter-integrated circuit bus. Sensors using I2C bus protocol consists of two active signal terminals, i.e. Serial Data (SDA) and Serial Clock (SCL) lines and one ground line terminal. Uses of I2C, pronounced as I square C (I2C) are for serial bus communication of the number of device circuits. For example, one for flash memory, touch screen, one for measuring temperature and one for measuring pressure in a number of processes in a plant. These ICs mutually network through a common synchronous serial bus. Figure 7.13 shows bus SCL and SDA lines for serial synchronous data communication using I2C protocol.

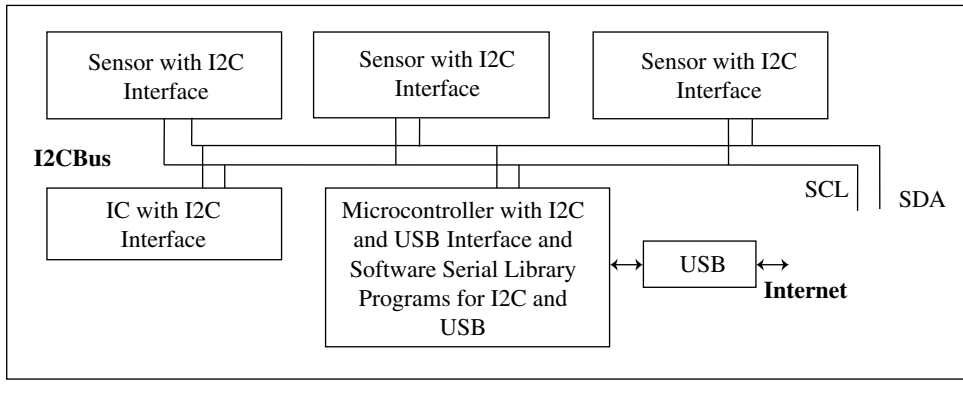


Figure 7.13 Bus SCL and SDA lines for serial synchronous data communication using I2C protocol

Three I2C bus standards are industrial 100 kbps I2C, 100 kbps SM I2C and 400 kbps I2C. I2C is serial computer bus which communicates single ended (one-end communicating at an instance), multi-master (number of master devices can connect the bus) and multi-slave (number of slave devices can connect the bus). A communicating device sends signals using SCL and SDA and can function as a master. Another device receiving signals SCL and SDA is the slave. The receiving device acknowledges use input SCL of master for sending SDA output.

Using the LIN Serial Bus

Local Interconnect Network (LIN) is a serial bus network protocol for communication between automobile circuits, sensors and actuator circuits, components and systems, such as window movements, seat movements and wipers. The protocol is simpler to use compared to CAN in automobiles.

LIN communication is single master with maximum 15 slaves with no bus arbitration. No bus arbitration means the bus does not select the destination, when number of devices request for the bus.[3]

LIN features are (i) use of single wire using which communication up to 19.2 kbit/s for 40 m bus length can be done, (ii) variable length of data frame (2, 4 and 8 byte) and (iii) configuration flexibility.

Using the CAN Protocol for Serial Bus

Embedded controllers with sensors and actuators are networked and are controlled through the CAN bus. Figure 7.14 shows serial bidirectional line network of number of CAN controllers and devices.

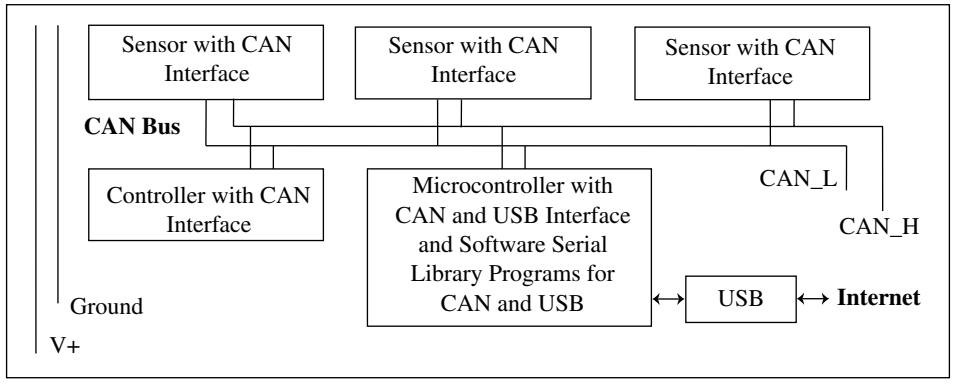


Figure 7.14 CAN bus serial bidirectional line network of number of CAN controllers and devices on a CAN bus

[3] https://en.wikipedia.org/wiki/Local_Interconnect_Network

A library program for CAN serial interface protocol enables direct use for embedded circuits with a sensor. CAN bus means different controller circuits using CAN communicate with the same set of wires.

A number of devices are located and are distributed in a Vehicular Control Network (VCN). An automobile uses a number of distributed controllers. The controllers provide the controls for brakes, engine, electric power, lamps, temperature, air conditioning, car-gate, front display panel, meter display panels and cruising. Medical electronics and industrial-plant serial communication also use the CAN bus.

CAN bus has a serial bidirectional line. A CAN device receives or sends a bit at an instance by operating at the maximum rate of 1 Mbps. It employs a twisted pair connection to each node. The pair runs up to a maximum length of 40 m. A CAN version also functions up to 2 Mbps. CAN bus is a standard bus in a distributed network.

Using USB Bus

Universal Serial Bus (USB) is a bus between a host system and a number of interconnected peripheral devices. Maximum 127 devices can connect to a host. It provides a fast (up to 12 Mbps) and as well as a low-speed (up to 1.5 Mbps) serial transmission and reception between the host and serial devices. Both the host and devices can function in a system. Three standard protocols for USB are USB 1.1 (a low speed 1.5 Mbps 3 meter channel along with a high speed 12 Mbps 25 meter channel), USB 2.0 (high speed 480 Mbps 25 meter channel), and wireless USB (high speed 480 Mbps 3 m).

A library program for USB serial interface protocol enables direct use for USB bus connected host system and serial devices.

USB bus uses tree topology. A host connects to the devices or nodes using a USB port driving software and a host controller. The host computer or system has a host-controller, which connects to a root hub. A hub is one that connects to other nodes or hubs. A tree-like topology forms in which the root hub connects to the hub and node at level 1 and a hub at level 1 connects to the hub and node at level 2 and so on.

Using the IEEE 1394 Bus Standard (FireWire) Protocol

Digital video cameras, digital camcorders, Digital Video Disk (DVD), set-top boxes, high-definition audio-video and music systems multimedia peripherals, latest hard disk drives, and printers need a high-speed bus standard interface directly to a personal computer. IEEE 1394 is a standard for 800 Mbps serial isosynchronous data transfer. Isosynchronous means that bits in data frame communicate synchronously, but frames in-between time interval can be variable.

Using the MOST Protocol

The communication functionality is provided by a driver software called MOST network services. The services are service programs. The software for processing the MOST protocol communication between a MOST Network Interface Controller (NIC) and devices.

The services enable a user to directly use these service programs, such as communicating media files using MOST NIC.

MOST protocol enables high-speed serial bus for synchronous data communication. It forms a multimedia network optimised for automotive and other industries. The MOST bus uses a ring topology.

Reconfirm Your Understanding

- Each serial interface uses a serial communication protocol. A microcontroller includes serial communication interfaces using UART and serial protocols, such as I2C, CAN.
- Sensor circuits send serial data on a serial bus using UART, SPI or I2C or CAN protocols.
- Automotive sensors communicate serial data using LIN, CAN and MOST serial protocols.
- A UART device sends 10-bits for each 8-bit data (character or digit or command) at successive intervals.
- RFID UART interface communicates 12 characters, i.e. one start, 10 for ID and one end character.
- A software serial library is provided in the Integrated Development Environment (IDE) for a microcontroller system. The library consists of a number of programs. A distinct program exists for use during communication using a serial interface protocol.
- Bus enables a number of interfaces connected through a common set of lines.
- A USB is used for serial synchronous communication to computer, tablet, mobile and other devices.
- MOST protocol communication enables multimedia file transfers in automobiles.

Self-Assessment Exercise

1. How many characters of 8-bit each can transfer in 1 s when baud rate in UART serial communication is 1.2 kbaud per second? ★
2. How many characters communicate when an RFID UART communicates the ID of 10 ASCII characters to a reader? ★
3. How is a software serial library used? ★★
4. List the features of the following serial protocols: UART, I2C, USB, CAN, LIN, IEEE 1394 and MOST. ★★★
5. Draw a diagram for using a microcontroller I2C bus interface and I2C bus for environment parameters temperature, barometric pressure, humidity and organic solvent vapour leakage sensors. ★★
6. List the LIN applications in an automobile network of electronic control units. ★
7. How and when is a USB bus used? ★
8. When are the following serial protocols: UART, I2C, USB, CAN, LIN, IEEE 1394 and MOST used? ★★★

7.6 RADIO FREQUENCY IDENTIFICATION TECHNOLOGY

LO 7.5 Explain the Radio Frequency Identification (RFID) technology

Section 1.5.2 introduced RFID, its functions, role and IoT applications. RFID is an identification system using tagging and labelling of objects. The following subsections give a detailed description of RFID technology.

7.6.1 RFID IoT Systems

Section 2.3.1 described the RFID wireless communication. A tag enables identification of an object at different locations and times. A product, parcel, postal article, person, bird, animal, vehicle or object can have a tag or label in order to make the identification feasible.

The reader circuit of an ID can use UART or NFC protocol to identify the tag, when the RFID tag is at a distance less than 20 cm. An active NFC device/mobile generates an RF field which induces the currents in RFID and generates enough power for RFID. Using that power, the RFID transmits the identification of tag contents.

Passive device drives power from the electrical current induced in its antenna by the incoming RF signals from a reader or hotspot, and then transmits the tag information back. The active device has an in-built power source (battery) and transmits the information on its own.

A hotspot consists of a wireless transceiver or Wi-Fi transceiver for Internet connectivity. It receives signals from a number of RFID tags in an organisation and transmits the data to the web server over the Internet. The hotspot connects to the Internet for IoT services, applications and business processes. A mobile or wireless nearby the device can also function as a hotspot.

RFIDs form an IoT network. They connect to the Internet and then to an IoT server. An IoT server consists of RFID identity manager, device manager, data router, analyser, storage and database server and services.

Principle of RFID

RFID device functions as a tag or label which may be placed over an object

A tag is an electronic circuit which transmits its ID using RF signals. The ID transmits to a reader, then that transmits along with the additional information to a remote server or cloud connected through the Internet. The additional information is as per the application. For example, for a tracking application, it is location and time-stamped data along with the ID. Example 7.13 gives the methods for communication of an ID from a tag and its reading by a reader.

Example 7.13

Problem

How does an RFID tag communicate and is read by a reader?

Solution

- Method 1: Communication is according to UART protocol of 10-bit per character. Reading RFID tag for processing is as follows—the tag circuit communicates first a header, which corresponds to ASCII code for a carriage return. The code is 13_d. Then ten digits (bytes) communicate which correspond to unique tag ID. The end byte is for communicating the finishing of data. The byte corresponds to new line ASCII code. The code is 10_d. Twelve bytes communicate serially.
- Method 2: Communication is according to NFC or another protocol, for which the ID reader is adapted.

An RFID tag has an advantage over a barcode or QR code in terms of simpler processing of the RFID data. It can also be made invisible to a person. This is because it uses short-range RF transceivers instead of light or laser. The tag transmits back a short string of data to reader. The RFID reader picks the RF and communicates to the Internet or remote web server or cloud server. An active system can transmit to the reader at longer range compared to a passive system. An active system can receive commands, process and then send information compared to no actions except sending the ID in the passive system.

RFID IoT Network Architecture

Example 2.2 explained a four layered ITU-T reference model for the Internet of RFIDs, individual capabilities of the layers and data interchange. Fourth layer capabilities are for IoT/M2M services and applications. RFID technology has many applications.

RFID IoT Applications

Examples are tracking and inventory control of goods, supply chain systems, business processes such as for payment, leasing, insurance, and quality management, access to buildings and road tolls or secured store centre entries, and devices such as RFID based temperature or any other parameter sensor. New applications of RFID network have been found in designing a factory, protecting a brand and anti-counterfeiting measures.

Components of an RFID System

Figure 7.15 shows the components needed in a system for IoT applications and services.

The components of an RFID system are:

Components of an RFID system are the tag, transceiver, reader, data processing subsystem, middleware, and applications and services

RFID is a tiny chip which functions as a tag or label onto an object. The chip is one of three types—passive, active and battery-powered passive (battery switches when reader is nearby).

A *transceiver* is in-built at the chip. It communicates in a range 10 cm to 200 m according to the chip. The chip does UART communication to the reader either using RF link or does NFC

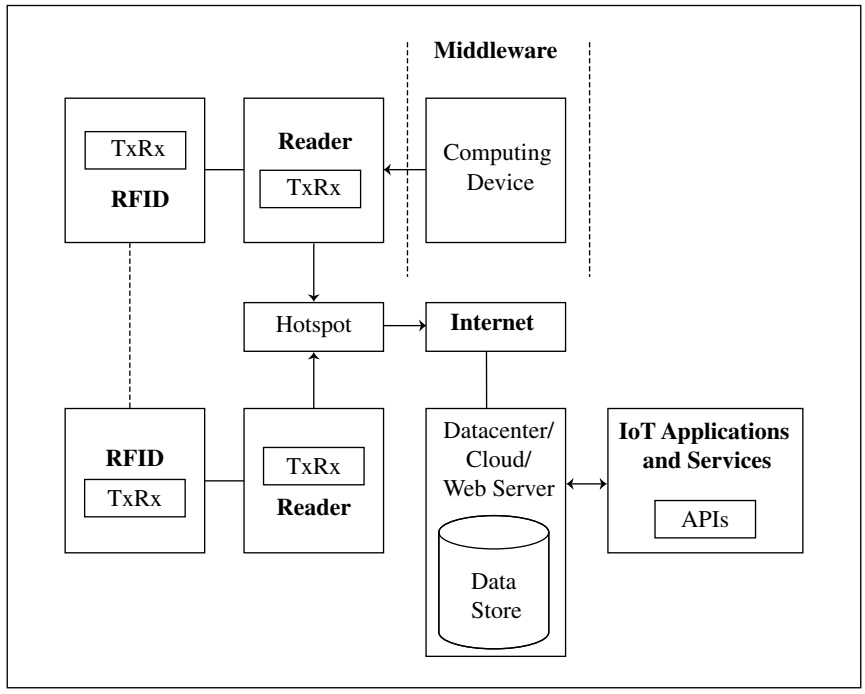


Figure 7.15 Components needed in a system for RFID IoT applications and services

communication to the reader within 20 cm range; standard frequency range used can be between 120 kHz to 150 kHz, 13.56 MHz, 433 MHz, higher when using UHF and microwave frequencies. Transceiver using RF frequencies recommended by Regulator has the data transfer rate of 115 kbps using carrier RF signals from 915 MHz to 868 MHz, 315 MHz or 27 MHz.

A nearby RFID reader for receiving ID uses the transceiver within it. It receives the header which consists of 1 start byte, then 10 byte ID and then one end byte when using the UART protocol. Hotspot, mobile or computer with wireless transceiver or Wi-Fi transceiver transmits and receives signals from the RFID tag.

Data processing subsystem: A reader associates a data processing subsystem which consists of a computing device and a middleware and provides connectivity to the Internet, directly or through a gateway which includes a data adaptation sublayer. The subsystem is a backend system. A reader circuit may send data directly or through a computer, mobile or tablet to the Internet. The computations for transmission (of the contents information of tagged device) are usually little. Example of a reader is SparkFun SEN-08419 for prototype developments.

Middleware: Middleware are software components used at the reader, read manager, data store for the transaction data store and APIs of the applications.

Applications and services and other associated applications software use the data store at the cloud or web server.

Issues

The issues are:

- ***Design issue:*** Designing a unique ID system needs a standard global framework.
- ***Security issue:*** A tag is read only. It can thus interact with any reader and thus allows automated external monitoring. A tag can thus be tracked without authority. A privacy issue arises when a tag and reader need not to be authenticated before their use. Full implementation of privacy and security needs data processing at the tag and reader with access encryption and authentication algorithms. Another issue is that the RFID system can be vulnerable to external virus attacks.
- ***Cost issue:*** RFID tag and reader become costly with data processing and security enhancing technology.
- ***Protection issue:*** The tag needs protection from the adverse weather condition which may damage the tag.
- ***Recycling issue:*** Recycling of the tags can be an environmental concern.
- ***Active life issue:*** Active RFID, which consists of battery, has limited life of up to 2 to 4 years.

7.6.2 EPCglobal Architecture Framework

MIT Auto-ID Labs is a group which consists of research laboratories of seven universities. The group has designed an architectural framework for IoTs. The group works in the field of networked RFIDs and emerging sensing technologies. The group works together with EPCglobal research group.

EPCglobal is a standard architecture framework, which uniquely identifies physical objects, loads, locations, assets and other entities

The group has suggested the Electronic Product Code (EPC) standards, roles and architecture. The goal of the EPCglobal architecture framework is assignment of a unique identity. The framework facilitates business processes, applications and services uniquely identifying physical objects, loads, locations, assets and other entities.

EPCIS and ONS, Design Issues

EPC Information Services (EPCIS) is a design of EPC global standards which enables EPC-related data sharing within and across enterprises.

The EPC global architecture defines the following:

- EPCIS Capturing Application (ECA) for capturing EPC-related data required for business processes
- EPCIS Accessing Application (EAA) for enterprise business processes supported by data captured using ECA
- EPCIS-enabled Repository for storing records of events and for retrieving information using queries from EAA

- Partner applications such as postal tracking system connected with payment systems.

Object Name Service (ONS) version 2.0.1 is a standard recommended in 2013. The ONS performs the lookup functions which are based upon the DNS. A DNS name enables connectivity to the web server using the Internet. ONS implements that function using a distributed set of servers. DNS is Domain Name System governed by IETF. Lookup function refers to looking at the DNS name for enabling the web server connectivity.

ONS enables web connectivity to web server by lookup functions based on the DNS

ONS characteristics are robustness, less needs of power and memory at the tag circuit and less wireless bandwidth need for the data communication. Functions of data communication use a backend network, such as a hot-spot or Wi-Fi device. Transfer of the data communication task to the backend network saves the wireless bandwidth.

Design issues include governance model and architecture of the ONS. The concerns involved are:

- Political for control over information
- Capability issue of loss of domestic and strategic capability
- Security issue of collecting business intelligence
- Commercial issues
- Innovation control issues

This joint control by countries of ONS model can handle and mitigate the risks which associate the single-country control.

Technological Challenges

RFID technology challenges are as follows:

- Interference: When an organisation uses a number of wireless systems, since RFID hotspot also requires wireless installation, the frequencies may interfere among the systems. The systems require effective mitigation from interference.
- Effective implementation at data processing subsystem consisting of reader and tag protocols, middleware architecture and EPC standards
- Need of low cost tags and RFID technology
- Design robustness
- Data security

Security Challenges

The issues associated with RFID security are:

- Discovery of foreign attacks (intrusions) and maintain overall data integrity
- Unauthorised disabling of a tag by a reader which is external, thus making the tag useless
- Unauthorised tag manipulation by a reader which is external, thus making the tag useless
- Cloning of the tag by an unauthorised entity

- Eavesdropping, which means setting up an additional reader pretending to be a reader belonging to the system
- Man-in-the-Middle attack: When an external object pretends to be either a tag or reader between system tags and readers

Solutions can be encryption, tag deactivation on detection of intrusion, mutual authentication between the tag and reader, detection of the tag owner, use of read data analyser and data cleaning. Example 7.14 gives the methods for solving security issues.

Example 7.14

Problem

How does an RFID tag securely communicate and read by a reader? How is an RFID tag secured?

Solution

Method 1: An RFID tag may have a processor and memory. Then it computes one-way hash function called meta ID on tag. When the RFID reader uses the meta-ID, then only the tag communication unlocks and the tag becomes readable. After the reader finishes reading, the tag gets locked. This disables reading by unknown entities to the system.

Method 2: The tag can self-destruct if under attack.

IP for an RFID in Internet of RFIDs

Data from the RFID reader after filtering, aggregation and routing get stored at an IP address. This data is in XML format. Data accesses using HTTP and SOAP protocols.

Internet protocol (IP), IPv6 is 128-bit IP address, which is required for routing data on the Internet. It needs to be mapped with the 96-bit EPC. The EPC is header, manufacturer, product and serial number bits.

A method for secure IPv6 communication is the use of Cryptographically Generated Addresses (CGAs). A host suffix/interface identifier generates a cryptographic one-way hash function. The function input is a public key as binary input. CGA is meant to be globally unique in a statistical sense. CGAs may, however, not be workable routing addresses at the IP layer for RFID IoT system.

Another method is Overlay Routable Cryptographic Hash Identifiers (ORCHID). It is a newly suggested class of identifiers. The identifier is based on CGAs. The ORCHID format is compatible to IPv6-like address format.

The ORCHID identifier is used just by the APIs and applications. It differs for IPv6 address. It does not identify the network locations, while IPv6 has locator which is 64-bit network prefix.

7.6.3 Web of Things of RFIDs

Web of Things (WoT) means making objects a part of the World Wide Web. WoT software, architecture styles such as JSON, REST, JSON, and programming patterns such as web sockets, makes this feasible. WoT data store of objects is similar to web pages store. The

web (application layer) receives and sends data using the Internet (IP network layer). WoT enables IoT applications for the RFIDs also as RFID enables identification of each object distinctly on the web.

Reconfirm Your Understanding

- A tag is an electronic circuit which transmits its ID using RF signals. The ID transmits to a reader which associates a computing device and connects to the Internet.
- RFID forms the IoT network using the computing system which connects to the Internet and then to IoT server.
- RFID system components for IoT applications and services are RFID tag with transceiver; reader with associated computing device and transceiver; hotspot or mobile of computer for Internet connectivity; middleware, and applications and services.
- Middleware consists of RFID identity manager, device manager, data router, analyser, storage and database server and services.
- Examples of IoT applications of RFIDs are for tracking and inventory control, identification in supply chain systems, business processes, RFID based temperatures or any other parameter sensor, in factory design, brand protection, and anti-counterfeiting and business processes.
- Design, security, cost, protection, recycling and active life are issues in IoT RFID applications and services.
- EPCglobal architecture framework assigns a unique identity for business processes. Applications and services uniquely identify physical objects, loads, locations, assets and other entities using the framework.
- EPC Information Services (EPCIS) is a design of EPCglobal standards which enables EPC-related data sharing within and across enterprises.
- ONS performs the lookup functions based upon the DNS. DNS name enables web server Internet connectivity using HTTP, REST, webSockets and Internet protocols.
- Technological challenges are wireless interference, implementing data processing subsystems, reducing system components' costs, robust system design and security of the system components.
- Data security solutions can be encryption, tag deactivation on intrusion detection, mutual authentication between the tag and reader, detections of the tag owner, reader analyser and data cleaning.
- RFID IoT system 128-bit IPv6 address can be generated using CGA.
- ORCHID is a newly suggested class of identifiers with a format compatible to IPv6-like address format.
- Web of Things means making objects data and object applications and services a part of the World Wide Web.

Self-Assessment Exercise

1. How is an RFID identified for IoT applications? ★★
2. Why does the RFID tag not create a network on its own? Why does it need a nearby reader for creating a network of RFIDs? ★★★
3. Show the RFID IoT system components for the parcel tracking applications and services using a diagram. ★

4. List the examples of RFID tags IoT applications and services. ★★
5. List the middleware functions required for the RFID IoT applications and services. ★
6. What are the technological issues in RFID IoT system design? ★
7. List the security issues in using the RFIDs IoT. ★
8. What are the features of EPCglobal architecture framework? ★
9. Why is EPCIS required?
10. How does an object data communicate and store at a web server using ONS? ★★
11. How is the IPv6 used for the RFID objects using ORCHID? ★★★
12. How is the web of things implemented? ★★★

7.7 WIRELESS SENSOR NETWORKS TECHNOLOGY

LO 7.6 Explain the Wireless Sensor Network (WSN) technology

A set of sensors can be networked using a wireless system (Section 1.5.3). They cooperatively monitor the physical and environmental conditions, such as temperature, sound, vibration, pressure, motion, or hazardous gas-leaks and pollutants at different locations. A WSN acquires data from multiple and remote locations. For example, in Internet of Waste Containers, the sensors wirelessly communicate the waste containers statuses in a waste management system in a smart city; sensors in Internet of Streetlights acquire data of ambient light conditions; and a WSN communicates the nearby traffic densities data for the control and monitoring of traffic signals.

Each node of the WSN has an RF transceiver. The transceiver functions as both, a transmitter and receiver.

Following subsections define WSN and describe its history, context, and architecture, and nodes, connecting the nodes, networking of the nodes and securing the communication. The subsections also describe establishment of a WSN infrastructure, data-link layer MAC protocols, routing protocols, various integration approaches, challenges of security, QoS and configuration, and WSN specific IoT applications.

7.7.1 WSN Concepts

Definition

WSN is defined as a network in which each sensor node connects wirelessly and has the capability of computation, for data compaction, aggregation and analysis. Each one also has communication as well as networking capabilities. A WSN consists of spatially distributed autonomous devices (sensors).

WSN History

Sound-waves-based surveillance and tracking systems for enemy submarines were used in the 1950s. Wireless-based networked radars were also used during this time. Research on the Distributed Sensor Networks (DSN) using network communication started before 1980. The DSNs are in use since then. Many spatially distributed sensing nodes collaborate and network autonomously on best effort basis. Research and applications of Low Power Wireless Integrated Micro-Sensors (LWIM) use in DSNs is ongoing since 1998.

WSNs have a large number of IoT applications. Examples are smart homes and smart city. The WSN nodes can sense and communicate, using the Internet, wirelessly the data from remote locations, such as industrial plant machines, forests, lakes, gas or oil pipelines, which may sometimes not be easily accessible.

Context-based Node Operations

The dictionary meaning of the word 'context' is 'the circumstances that form the setting of an event, statement, or idea, and in terms of which it can be fully understood. A WSN node can adapt, re-program or do another task using a sensor and associated circuit, computations, networking capabilities and *context* at that node.

When nodes can adapt their operations in response to changes in the environment, then it is called context-dependent sensing, networking and computing. The application layer programs help the node to differentiate the tasks to be undertaken on a changed context.

The selection of data, memory, power and routing path management, the routing protocol, users, devices and application interfaces in the WSN system can be programmed to function as per the context, and consider the circumstances during networking and computations.

Context can be a physical, computing, user, structural or temporal context. Following may correspond to the context for re-programming of the actions of the WSN nodes:

- Past and present surrounding situations
- Actions such as the present network
- Surrounding devices or systems
- Changes in the state of the connecting network
- Physical parameters such as present time of the day
- Nearest connectivity currently available
- Past sequence of actions of the device user
- Past sequence of application or applications
- Previously cached data records
- Remaining memory and battery power at present

A WSN reprogramming can be Over-The-Air (OTA), which means wirelessly modifying the codes in flash memory through an access point by the gateway, application or service.

Example 7.15 shows how nodes, coordinators and routers take into account and adapt their tasks according to a context.

Example 7.15

Problem

How is the context taken into account by the WSN nodes, network coordinators and routers?

Solution

Assume a set of WSN nodes sensing the environment conditions, which includes the detection of hazardous goods or gases in an industry. As soon as a sensor senses a hazardous good, the node discovers the paths and protocols available for broadcasting the newly sensed information. The nodes coordinate with each other to send the information towards the destination.

Assume a WSN in a forest tied with a bird or mule or another object. The node can follow any direction and reach any location at different instances. When the node finds another node in the WSN, it shares the information. When any node of a WSN senses and discovers wireless radiation from an access point, it off-loads the collected data. The access point sends that on the Internet or to another coordinating WSN with another set of node. Further, the node battery can also accumulate the charge from the radiations of the access point.

Assume a wireless sensor node may extend the interval for temperature or other data transmission when it senses low memory or power availability. When the node completes the off-loading of the data and recharges at the access point or coordinating node of WSN clusters, it may reduce the interval for temperature measurements.

7.7.2 WSN Architecture

WSN Node Architecture

Figure 7.16 shows the three-layer architecture of a node. The three layers are application layer, network layer (serial link with data-link MAC), physical cum data-link layer (MAC + Physical Layer).

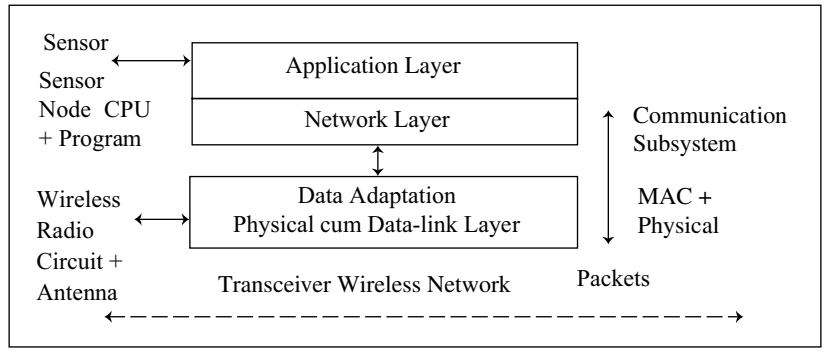


Figure 7.16 Architecture of a wireless sensor node

The application layer software components are sensor management, sensor query and data dissemination, task assignment, data advertisement and application-specific protocols.

Sensor, CPU and program sensor node constitute the application and network layers. Network layer links serially to the data-link layer, and may include the coordination or routing software. A serial link interconnects the layers to a wireless radio circuit and antenna. The radio circuit is at physical cum data-link layer. Communication subsystem uses MAC and physical protocols.

Architecture for Connecting Nodes

Figure 7.17 shows two architectures for connecting WSN nodes, fixed connecting infrastructure of WSN nodes, coordinators, relays, gateways and routers, and mobile ad-hoc network of WSNs, access points, routers, gateways and multi-point relays.

An access point is a fixed point transceiver to provide the accessibility to nodes present nearby or nodes reaching in the wireless range. A multipoint relay connects to other networks such as the Internet or mobile service provider network. A router's role is to select a path for packet transmission among the presently available paths in the network. A coordinator provides the link between the two networks.

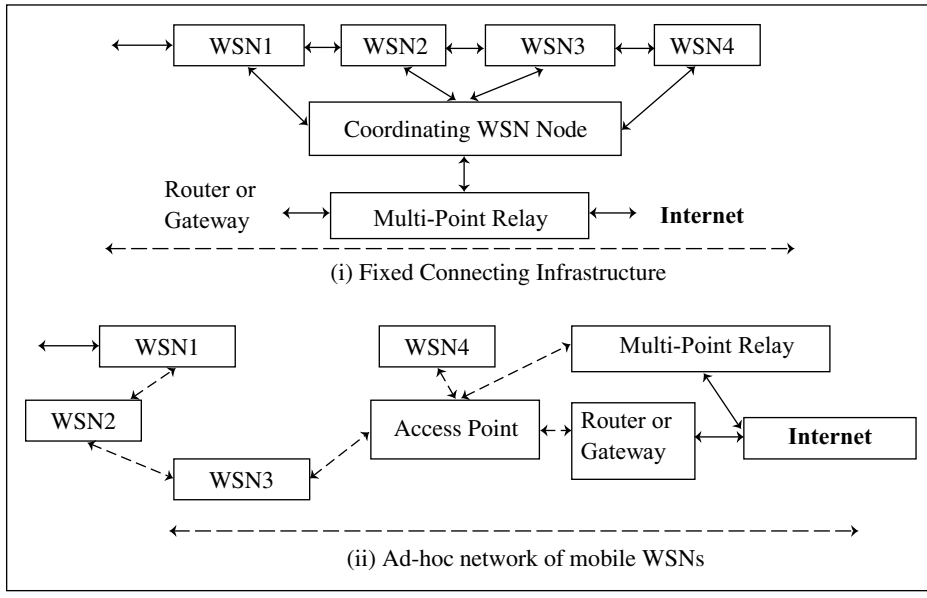


Figure 7.17 Architecture for connecting WSN nodes: (i) Fixed connecting infrastructure of WSN nodes, coordinators, relays, gateways and routers and (ii) Mobile ad-hoc network of moving WSNs

Fixed infrastructure example is a smart home network consisting of WSNs at security surveillance points, refrigerator, air conditioner, microwave, TV and computers with Wi-Fi access point. Another example is WSN nodes, access points, routers, gateways and multi-point relays. At an industrial plant, multiple machine locations, stores, offices, sales and receipts and other plant locations.

Ad-hoc infrastructure example is mobile WSNs tied with birds or animals for habitat monitoring.

Architecture for Networking of the Nodes

Two basic architectures for networking of the nodes are layered architecture and multi-cluster architecture.

Wireless Multi-Hop Infrastructure Network Architecture (MINA)

MINA is a layered architecture. The WSN nodes have data sensing as well as capabilities of forwarding towards the access point (base station). The nodes can be mobile and have coverage and mobility range for communication to remote access points.

Access points possess data gathering and processing capabilities and have connectivity with a larger network, such as the Internet. Figure 7.18 shows layered architecture.

Each node connects to a short-distant neighbour. When the node moves to longer distances then it communicates through 2 or 3 hops to the access point (base station). Each node has low-power transceivers to the nearest neighbouring layer WSNs.

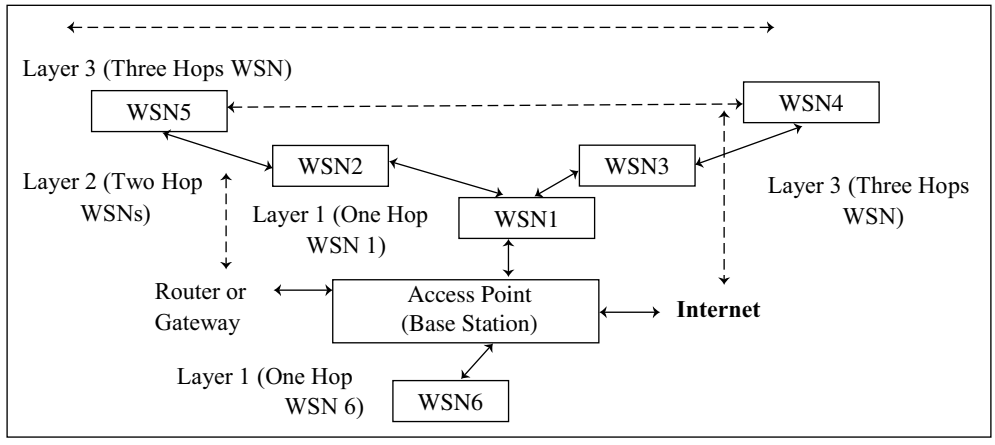


Figure 7.18 Layered Architecture for network of nodes

Assume that the base station is surrounded by three layers of WSNs. Layer 1 WSNs directly connect. Layer 2 WSNs first connect to layer 1 WSNs functioning as coordinators and then directly connect. Layer 3 WSNs first connect to layer 2 WSNs functioning as coordinators, then connect to layer 1 WSNs and then connect to the access point.

The figure shows that layer 1 WSN1 and WSN6 connect directly to the access point. It means hop count = 1. The figure also shows WSN 2 and WSN 3 at layer 2. They connect by one hop to WSN1 and next hop through WSN1. Hop count = 2 for layer 2. It means that WSN 2 connects to W**S**N1 which has connectivity with the access point (base station).

The figure also shows WSN 4 and WSN 5 at layer 3. They connect by three hops, one to layer 2 WSN and then layer 1 WSN and then to the access point. Hop count = 3 for layer 3. It means that WSN 5 connects to WSN2, then to WSN1 and then the access point (base station). WSN 4 connects to WSN3, then to WSN1 and then the access point (base station). Access points connect the clusters using wireless LAN (802.11b) protocol. The access points enable connectivity to the Internet. Sensor data is archived and can be queried in real-time, and users with mobile devices and remote clients can access the data.

Multiple Clusters Architecture

Each cluster has a gateway node. A set of clusters with a gateway each has one cluster with a cluster-head gateway. Multi-cluster architecture has a number of clusters, which associate a cluster-head gateway.

Cluster-head enables a tree-like topology of the clusters in a multi-cluster architecture. The cluster formation and election of cluster-heads is autonomous in distributed WSNs and WSN clusters. Figure 7.19 shows a multi-cluster architecture. The figure shows two clusters. The figure also shows a cluster-head gateway. Head gateway connects to the Internet or cellular of other networks and provides connectivity to WSNs in the multiple clusters.

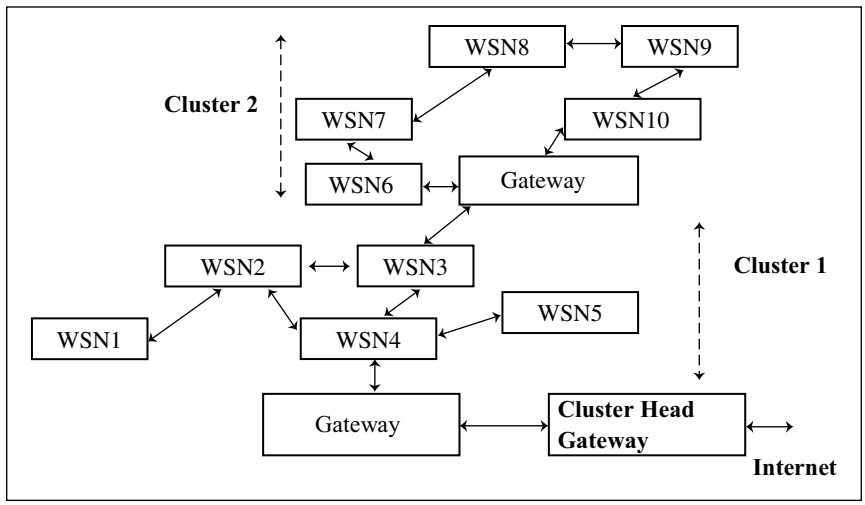


Figure 7.19 Multi-cluster architecture for network of nodes

A number of clusters associated with the cluster-head gateway depends on the coverage required in the network. When a cluster-head exists then a gateway interconnects two clusters. Each node connects to a short-distant neighbour. The node connects to a gateway node through one, two or more hops when it connects to the WSN of another cluster. When a node moves to longer distances then it communicates to the neighbouring cluster through the gateway.

Cluster 1 is an ad-hoc network of mobile WSNs (WSN 1 to WSN5). The WSNs in a cluster may have mesh architecture or layered architecture. Each WSN connects to another WSN in its wireless range. Cluster 2 is an ad-hoc network of mobile WSNs (WSN 6 to WSN10).

Clustered architecture enables data compaction or fusion and aggregation. A gateway communicates compacted or aggregated data to another cluster. A cluster-head further aggregates, compacts or fuses the data before communication to the web server or cloud using the Internet.

7.7.3 WSN Protocols

Network protocols have design goals as: (i) limit computational needs, (ii) limit use of battery power and thus bandwidth limits, operation in self-configured ad-hoc setup mode and (iv) limit memory requirement of the protocol.

The physical layer does adaptive RF power control at the transceiver; reduces power in case of nearby node, increases the power when the nearest node is at a greater distance, uses CMOS low power ASIC circuits and energy-efficient codes.

Data-link Layer Media Access Control (MAC) Protocol

S-MAC (Sensor-MAC) protocol can be deployed at the data-link layer. S-MAC nodes go to sleep for prolong periods. They need to synchronise at periodic intervals.

S-MAC protocol enables use of the energy-efficient, collision-free transmission, and intermittently synchronises the operations. Collision-free transmission results from scheduling of a channel. Each node can be allocated the channel. Channels are reused such that collision-free transmission takes place and retransmission is not resorted to.

Routing Protocols

The network layer uses multi-hop route determination, energy-efficient routing, route caching and directed diffusion of data.

Routing protocols are either proactive or reactive. Proactive protocols keep route cache and determine the route in advance. Reactive protocols determine the route on demand. Routing protocols are table-driven when a routing table guides the paths available. Routing protocols are demand-driven when a source demands the route and it guides the paths available.

Fixed Wireless Sensor Networks may use Cluster-head Gateway Switch Routing (CGSR) which guides the paths using heuristic routing schemes.

7.7.4 WSN Infrastructure Establishment

When a WSN infrastructure establishes the following steps which need consideration:

- Sensors with associated CMOS low power ASIC circuits, their radio ranges and energy-efficient coding

- Infrastructure selection, (i) fixed connecting infrastructure of nodes, coordinators, relays, gateways and routers, or (ii) mobile ad-hoc network of mobile WSNs with limited or unspecified mobility region
- Network topology and architecture according to the applications and services, wireless multi-hop infrastructure network architecture or multi-cluster architecture
- Network nodes self-discovering, self-configuring and self-healing protocols, localisation, mobility range, security, data-link and routing protocols, link quality indicator (LQI) (packet reception ratio), coverage range, and QoS required according to applications and services
- Clusters, cluster gateways, cluster-heads and clusters hierarchy
- Routing, data aggregation, compaction, fusion and direct diffusion
- Synchronisation of times at intermittent intervals as time is used as reference in inter-node distance estimation, accounting for the delays, localisation, ranging and location services.

WSN Various Integration Approaches

WSN needs integrated approaches for:

- Nodes design and provisioning of resources
- Nodes localisation
- Nodes mobility
- Sensor connecting architecture
- Sensor networking architectures
- Data dissemination protocols
- Security protocols
- Data-link layer and routing protocols
- Integration with sensors data, other than wireless sensors data, for IoT applications and services

Quality of Service

Quality of Service (QoS) is an average weighted QoS metric over the lifetime of a network. Several metrices are as follows: (i) Average delay: Measure of the time taken in generation of sensor data and its delivery up to the destination, (2) Lifetime: Time up to which the WSN functions effectively or given energy resources of the modes will last, (iii) Throughput: Bytes per second delivered up to the destination. Low throughput means high delays. It also relates to the bandwidth of the network, and (iv) Link Quality Indicator (LQI) which means packets delivered/packet transmitted from the nodes.

Real-time applications of communication over the WSNs sensor networks need provision of guaranteed specifications for maximum delay, minimum bandwidth or other QoS parameters.

Challenges of high QoS are (i) routing through nodes of high throughputs, lower delay and paths which use low energy resources, (ii) link and (ii) maintaining the product of priority and delay, which means higher priority packets take lower delay paths and lower priority packets take higher delay paths.

A metric is the coverage of a sensor network. The coverage depends on the density and locations of the nodes in the region, communication range and sensitivity of the nodes. Another metric is percentage of times the network cover occurrence of an event in case of surveillance systems, hazard chemicals or fire-detection systems.

Configuration

Challenges of configuring in view of resource constraints with the nodes statically or dynamically or self-automatic are the following: (i) locations and mobility range of the WSN nodes, (ii) clusters, (iii) gateways, (iv) cluster-heads, (v) sampling rate of the sensed parameters and (vi) their aggregation, compaction and fusion.

Challenges are to be met with limited battery power, storage, computation capability, bandwidth and scalability limit on the nodes.

7.7.5 WSN Nodes Secure Communication

Data privacy and integrity needs and two protocols SNEP and μ-TESLA

Sensor networks need secure communication for data privacy and integrity. Authentication ensures data from the sensing node only, maintains data integrity and disables the communication of messages from unauthenticated sources. Privacy ensures data secrecy and eavesdropping (entering without permission).

Berkeley laboratories suggested SPINS (Security Protocols in Network of Sensors). SPINS use the symmetric cryptographic protocol because of the following reasons—an asymmetric cryptographic method which has high overhead in term of memory and computations for digital signatures, key-generation and verification. It also has high memory requirements and communicates higher number of bytes compared to the symmetric method.

SPINS is a suite of security protocols for the sensor networks which are: (i) Secure Network Encryption Protocol (SNEP) and (ii) micro-Tesla (m-Tesla). SPINS use block cipher encryption functions.

SNEP enables secure point-to-point communication. It ensures data privacy and integrity. It secures communication after authentication process. It does not require message replay, thus message remains fresh.

Access point distributes session key to A and B using SNEP. Two nodes, A and B, share six keys. The encryption keys, K_{AB} and K_{BA}, Cipher-block-chaining Message Authentication Code (MAC) keys, K_{AB} and K_{BA}, and counter keys, C_A, C_B are the shared keys. MAC ensures message integrity and privacy. .

- μ-TESLA is light-weight version of TESLA. It enables authenticated broadcasting. The authentication is micro-timed and efficient. It results in stream-loss tolerant secure authentication.

  An access point authenticates using μ-TESLA. First a packet is listened and considered as a parent. It is authenticated later. This makes secure authentication stream-loss tolerant.

Distributed sensor networks use a key management scheme, which can be (i) based on probabilistic key sharing or (ii) random key pre-distribution key sharing.

Localised Encryption and Authentication Protocol (LEAP)

Different packets when using LEAP use different keying mechanisms. Security needs decide the mechanism. A node uses four keys: individual key, group key, cluster key and share a pair of key with the neighbour.

A challenge for highly-distributed architecture with localised coordination is the implementation of goals of a system for applications and services.

Implementation needs are autonomous operation, self-organisation, self-configuration, adaptation, energy conservation at physical, MAC, link, route, application layer, design of scalable node density, number and types of networks.

Network is data-centric network and route nodes have no addressability. Three challenges, viz. security, QoS and configuring of nodes are as follows:

Security

Security challenges are:

- Hello flood attack: An attacker node sends hello messages repeatedly, and thus drains the energy of the attacked node.
- Sybil attack is an attack where a single node, presents itself as different entities at different times.
- Selective forwarding attack is an attack when the attacker node does not forward the attacked node messages on receiving.
- Sinkhole attack is when an attacked node behaves as an access point and receives the messages without forwarding them.
- Wormhole attack is an attack where the attacker node gives false information of distances of the destinations, thus forcing the attacked node to take longer paths. The longer path has high latency and thus high delivery delays of packets.

7.7.6 WSN IoT Applications

WSNs are increasingly being used as a subsystem in IoT-based applications and services. WSN can function as a source of data along with other systems and connect through a gateway and access point to the Internet.

An example of WSN specific IoT application is smart home control and monitoring system (Sections 1.5.3 and 12.5.1). A connected home has the following applications deployed in a smart home:

1. Mobile, tablets, IP-TV, VOIP telephony, video conferencing, video-on-demand, video-Surveillance, Wi-Fi and Internet
2. WSN nodes and wireless actuator nodes, which can be built using ZigBeeIP (Section 2.3.1) are nodes for home security access control and security alerts, lighting control,

home health care, fire detection, leak detection, energy efficiency, solar panel monitoring and control, temperature monitoring and HVAC control and automated meter reading.

Figure 7.20 shows source-end cluster of ZigBee WSN and actuator nodes, coordinators and routers connected through gateway and set of RPL routers for data packets from IPv6 addresses and communicating with IoT and M2M IoT applications and services layer for data packets from IPv6 addresses.

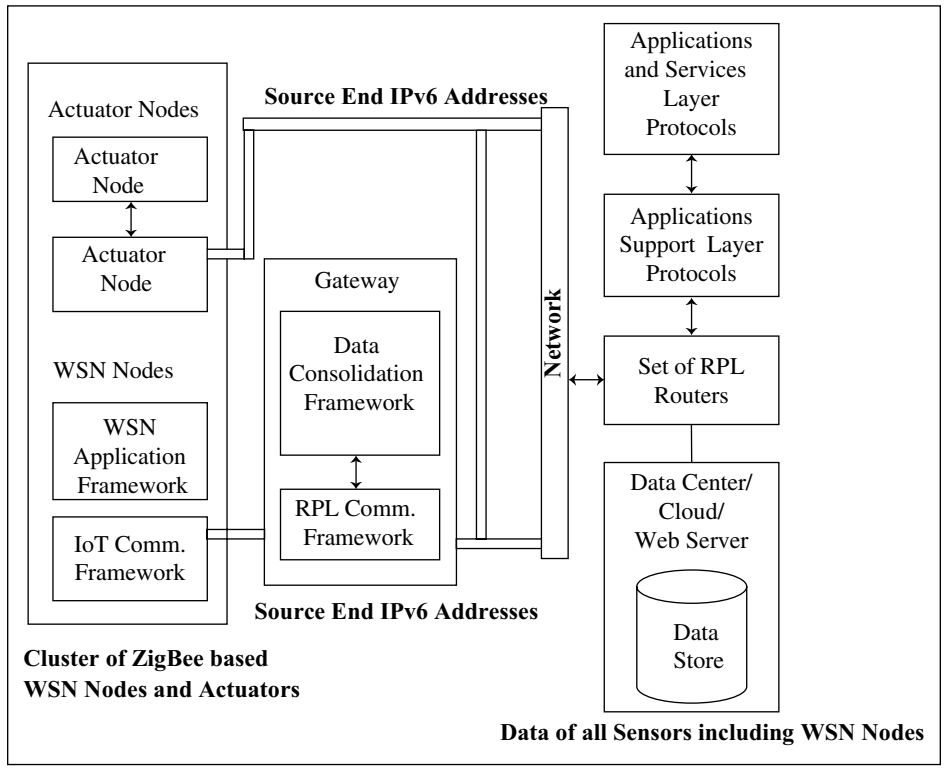


Figure 7.20 Source-end cluster of ZigBee WSN and actuator nodes, coordinators and routers connected through gateway and set of RPL routers for data packets from IPv6 addresses and communicating with IoT and M2M IoT applications and services layer

ZigBee devices form WPAN (Wireless Personal Area Network) devices in home network. WSN nodes, coordinating nodes and routing nodes can also use ZigBee protocol for physical layer wireless connectivity. ZigBee is IEEE 802.15.4 standard protocol for physical/data-link layer (Phy/MAC).

ZigBee IP is an enhancement for IPv6 connectivity (Section 2.3.1). ZigBee IP is enhancement with RFD. RFD (Reduced Function Device) means one that functions for the 'sleepy' battery-operated device. Sleepy means one that wakes up infrequently, sends data

and then goes back to sleep. 6lowpan refers to IPv6 over low power wireless personal area network which permits communication of IPv6 packets in a sensor network.

Reconfirm Your Understanding

- A set of sensors can be networked using the wireless. They do cooperative monitoring of physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or hazardous gas-leaks and pollutants, waste containers, home appliances and surveillance systems at different locations.
- Sound-waves-based surveillance and tracking systems for enemy submarines were used in the 1950s. Research and applications of low power wireless integrated micro-sensors and WSN uses are ongoing since 1998.
- WSNs are subsystems which connect to the Internet using gateways. The subsystems have a number of IoT applications and services, such as for smart homes and smart cities.
- WSN nodes associate with CMOS low power ASIC or microcontroller circuits, transceiver of adaptable radio ranges and coding for processing needs to be energy-efficient.
- WSN infrastructure can be fixed connecting infrastructure of nodes, coordinators, relays, gateways and routers, or an ad-hoc connecting network of mobile WSNs with limited or unspecified mobility region.
- The architecture of a WSN is according to applications and services. Two basic architectures are wireless multi-hop infrastructure network architecture or multi-cluster architecture using clusters, cluster gateways, cluster-heads and clusters hierarchy.
- WSN nodes and gateway do the routing, data aggregation, compaction, fusion and direct diffusion.
- Network is a data-centric network and route nodes have no addressability.
- Network nodes use self-discovering, self-configuring and self-healing protocols.
- Three challenges are the WSN security, QoS and configuring of the nodes.
- WSN issues are localisation, mobility range, security, data-link and routing protocols, link quality indicator, coverage range, and required QoS according to applications and services.
- A WSN IoT application is smart home control and monitoring system (Sections 1.5.3 and 12.5.1). A connected home has number of applications deployed in the smart home.

Self-Assessment Exercise

1. How does a wireless sensor circuit differ from that of a sensor? ★★
2. How does a wireless sensor node circuit differ from that of an RFID? ★★
3. List the context changes on which a WSN node may be configured to re-program. ★
4. Draw the three-layer architecture of a wireless sensor node. ★
5. What are the advantages of fixed WSN nodes infrastructure? ★
6. What are the advantages and characteristics of infrastructure for ad-hoc WSN nodes? ★★

7. How is a fixed WSN architecture used in IoT application to waste containers management system? ★★★
8. How is an ad-hoc WSN architecture used in IoT application for habitat monitoring of birds? ★★★
9. What are the advantages of layered multi-hop WSN nodes? ★★
10. List the security attacks in WSNs. ★
11. List the advantages of using SNEP and micro-TESLA for privacy and data integrity in WSNs over security protocol, such as TLS. ★★★
12. List the design challenges of WSN security, QoS and configuring of the nodes. ★★★
13. List the security attacks in WSN. ★
14. List the considerations for establishing infrastructure when deploying a WSN. ★

Key Concepts

- Accelerometer
- Access point
- Ad-hoc network
- Analog sensor
- Analog to digital converter
- Automotive IoT
- Barcode
- CAN protocol
- Digital sensor
- EPC
- Fixed infrastructure
- Gyroscope Hotspot
- I2C protocol
- Industrial IoT
- Microcontroller port
- Multi-cluster architecture
- Multi-hop architecture
- NFC
- On-off states
- Participatory sensing
- Pressure sensor
- QR code
- Quality of service
- RFID
- RFID reader
- Security
- Sensor node
- Serial port interface
- Set of on-off states
- Smart home
- Software library
- Temperature sensor
- Transceiver
- UART
- Wireless sensor network

Learning Outcomes

LO 7.1

- Many IoT applications need data generated from sensing devices. Sensors for number of physical environments, parameters and conditions convert physical energy into electrical energy.
- A complex sensor includes an output processing circuit, which possesses computing and communication capabilities. Sensing needs the microcontroller for storing, computing and communication and sensor electronic circuits.
- Sensors are of two types—analog and digital. Sensors can sense temperature, humidity, acceleration, angular acceleration, object distance, orientation angle with respect to a fixed direction, magnetic object proximity, touch and gestures of users, motion, sound, vibrations, shocks, electric current and environment conditions.

- Digital sensing needs a circuit to generate 1 or 0 or binary output of 1s and 0s for storing, computing and communication. Digital sensors find wide applications. For example, sense the number of chocolates of each type of flavours remaining unsold and communicate that number to chocolate fill service. Another example is sense unoccupied parking slots.
- MEMS is a micro electromechanical sensor which detects linear accelerations a_x, a_y and a_z along three axes x, y and z, respectively.
- Things get sensed, identified by barcode readers or QR code, and the associated circuit communicates the object identity and object document information to the Internet.

LO 7.2

- Participatory sensing means participation of physical sensors as well other means, such as social media. Applications of participatory sensing are many, such as retrieving weather and environment information. It can be used to retrieve information about pollution, waste management, road faults, individual and group of personal health, traffic congestion, urban mobility or disaster management such as flood, fire etc.
- Industrial Internet of Things (IIoT) is the use of IoT technologies in manufacturing and predictive maintenance. IIoT integrates complex physical machinery M2M communication with the networked sensors, and uses of software, analytics, machine learning and knowledge discovery.
- Automotive IoT enables connected cars and vehicles data communication on the Internet for maintenance, services and a number of applications.

LO 7.3

- The examples of actuators are light sources, LEDs, piezoelectric vibrators and sounders, speakers, solenoids servomotor, relay switch, switching on a set of streetlights, application of brakes of a moving vehicle, ringing of alarm bell, and switching off or on the heater or an air conditioner or a boiler current in a steam boiler in a thermal plant.

LO 7.4

- Each serial interface uses a serial communication protocol, UART, SPI or I2C or CAN protocols. Automotive sensors communicate serial data using LIN, CAN and MOST serial protocols
- A software serial library is provided in the Integrated Development Environment (IDE) software for a microcontroller system for distinct programs for use during communication using a serial interface protocol and for serial synchronous communication to computer, tablet, mobile and other devices.

LO 7.5

- RFIDs form the IoT network using the computing system which connects to the Internet and then to IoT server.
- RFID system components for IoT applications and services are (i) RFID tag with transceiver, (ii) A reader with associated computing device and transceiver, (iii) Hotspot or mobile of computer for Internet connectivity, (iii) Middleware and (iv) Applications and services.
- Examples of applications of Internet of RFIDs are tracking and inventory control, identification in supply chain systems, business processes, RFID based temperatures or any other parameter sensor, in factory design, brand protection, and anti-counterfeiting and business processes.

- EPCglobal architecture framework assigns a unique identity for business processes. ONS performs the lookup functions based upon the DNS. DNS name enables web server Internet connectivity using HTTP, REST, webSockets and Internet protocols.
- Data security solutions can be encryption, tag deactivation on intrusion detection, mutual authentication between tag and reader, detections of tag owner, a read data analyser and data cleaning.
- RFID IoT system 128-bit IPv6 address can be generated using CGA.

LO 7.6

- A set of sensors can be networked using the wireless. They cooperatively monitor the physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or hazardous gas leaks and pollutants, waste containers, home appliances, and surveillance systems at different locations.
- WSNs are subsystems which connect to the Internet using access points and gateways. The subsystems have a number of IoT applications and services, such as for smart home, and smart city.
- Architecture of a WSN is according to applications and services. Two basic architectures are wireless multi-hop infrastructure network architecture or multi-cluster architecture using clusters, cluster gateways, cluster heads, and clusters hierarchy.
- Network is a data-centric network and route nodes have no addressability. Three challenges are security, QoS and configuring of nodes.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. Sensors use the parameters of (i) resistance, (ii) capacitance, (iii) reverse diode saturation current, (iv) saturation current between collector-emitter in photo-transistor, (v) current in LED, (vi) piezoelectric current or (vii) magnetic field variation with time for sensing the physical environment or conditions. ★★
   (a) (i) and (ii)
   (a) (i) to (vi)
   (b) (i) to (iv)
   (c) All except (v)
2. Relative humidity sensor senses using: ★
   (a) Capacitance change between the parallel plates due to water vapours in air
   (b) Resistance of air varying with moisture
   (c) Piezoelectric effect in air
   (d) Magnetic flux changes
3. Distance of a car in front is measured by measuring delay of (i) ultrasonic pulses echoed back, (ii) microwaves reflected, (iii) infrared reflected, (iv) light reflected and (v) image from the car in front. ★★
   (a) (i) to (iii)

(b) (iv)
(c) (i)
(d) (v)

4. Accelerometer sensor uses (i) MEMS, (ii) ultrasonic waves, (iii) IR and has (iv) three, (v) five terminals and (vi) six terminals for voltage outputs corresponding to three components of acceleration. ★★
   (a) (iii) and (iv)
   (b) (i) and (v)
   (c) (ii) and (vi)
   (d) (i) to (iii) and (vi)
5. Participatory sensing is the sensing by (i) individuals, groups of people and communities using mobile phones with multiple sensors and cloud services, (ii) collaborative sensor circuits and (iii) multiple parameters sensing by individuals and groups of sensors, contributing, collecting and analysing sensory information to form a body of knowledge. ★
   (a) All
   (b) (iii)
   (c) (ii)
   (d) (i)
6. Industrial Internet of Things (IIoT) is the use of IoT technologies in (i) refining the operations of manufacturing, (ii) maintenance, (iii) refining business models. IIoT involves the uses of (iv) integration of complex physical machinery, (v) M2M communication with the networked sensors, (vi) software, (vii) analytics, (viii) machine learning, and (ix) knowledge discovery. ★★
   (a) All except (ii), (iii) and (ix)
   (b) All (iii) and (ix)
   (c) All
   (d) (i) and (vi) to (ix)
7. Automotive IoT enables (i) connected cars, (ii) vehicle-to-infrastructure technology, (iii) predictive and preventive maintenances, (iv) descriptive and predictive analytics, (v) manufacture to develop the improved components and (vi) use of autonomous cars. ★★★
   (a) All
   (b) All except (ii) and (v)
   (c) All except (vi)
   (d) All except (iii) and (vi)
8. An actuator is a device which takes actions as per the input command using (i) 1s and 0s, (ii) pulse or state (1 or 0) or set of 1s and 0s or (iii) width of the input pulses of 1s and 0s and (iv) control inputs. ★
   (a) All except (iii) and (iv)
   (b) All except (iii)
   (c) All
   (d) All except (iv)
9. A software serial library is provided in (i) the integrated development environment for a microcontroller system and (ii) in library programs of the user. The library has (iii) the distinct program for each serial interface protocol, (iv) a program for ★★★

set of serial interface protocols, (v) the program for all interface protocols and (vi) a program for sending data to USB port for onward transmission on the Internet.
(a) (i), (iii) and (vi)
(b) (i) and (v)
(c) All except (vi)
(d) All

10. The Web of Things (WoT) means making (i) objects (ii) objects including RFIDs and (iii) JSON as a part of the World Wide Web. (iv) WoT data store of objects is dissimilar to web pages store. (v) The Web (Application Layer) receives and sends data using the Internet (IP Network Layer). (vi) WoT enables IoT applications for the RFIDs also as RFID enables identification of each object distinctly on the web. ★★★
    (a) All
    (b) (i), (ii), (v) and (vi)
    (c) All except (vi)
    (d) All except (iii)
11. SPINS SNEP and micro-TESLA protocols for WSN security has advantages as: (i) Uses asymmetric cryptographic method, (ii) high overhead in term of memory and computations, (iii) uses digital signatures, (iv) has high memory requirements and (v) communicates higher number of bytes compared to symmetric cryptographic methods method. ★
    (a) (i)
    (b) (ii)
    (c) (iii)
    (d) None
12. Challenge for highly-distributed architecture with localised coordination is the implementation of goals of a system for applications and services using WSNs implementation needs are (i) autonomous operation, (ii) self-organisation, (iii) self-configuration, (iv) adaptation, (v) energy conservation at physical, MAC, link, route, application layer, (vi) design of scalable node density, number and types of the networks, (vii) security and (viii) quality of service. ★★
    (a) (iii) to (vii)
    (b) All
    (c) All except (vi)
    (d) (ii) to (viii)

Short-Answer Questions

1. How do sensors measure sound intensity, strain, flex, pressure, vibrations and motion? **[LO 7.1]** ★
2. Why is an equation or table required to find the sensed parameter value from the sensor circuit output? **[LO 7.1]** ★★
3. How are distances sensed? **[LO 7.1]** ★★
4. What are the uses of a phototransistor-LED-pair-based digital sensor in an automobile? **[LO 7.1]** ★★★
5. How is participatory sensing used for city traffic densities management using IoT? **[LO 7.2]** ★★
6. How can industrial IoT be defined? **[LO 7.2]** ★

7. What are the automotive IoT applications of vehicle communication with other vehicles, surrounding infrastructure and Wi-Fi local area network? **[LO 7.2]** ★★★
8. What are the actuators used in Internet of Streetlights? **[LO 7.3]** ★
9. Why is a serial CAN bus deployed for connecting sensor circuits in an automobile? **[LO 7.4]** ★★
10. When are the barcodes and QR codes used? **[LO 7.5]** ★
11. How is the security maintained in RFID IoT system? **[LO 7.5]** ★
12. What are the differences in nodes and network topology for RFID IoT and WSN IoT applications? **[LO 7.6]** ★★★

Review Questions

1. What are the uses of an analog sensor? What are the uses of a digital sensors? **[LO 7.1]** ★
2. How do sensors for temperature, pressure and humidity function and communicate data on Internet? **[LO7.1]** ★★★
3. Why do many sensors have three or four terminals for communication to the microcontroller? How are such sensors used for IoT applications and services? **[LO 7.1]** ★★★
4. How do the digital compass and accelerometer function? **[LO 7.1]** ★
5. What are the merits and demerits of participative sensing? **[LO 7.2]** ★★
6. How is industrial IoT used? **[LO 7.2]** ★★
7. Give a table of applications of analog and digital sensors in automotive IoT. **[LO 7.2]** ★
8. What are the uses of LEDs when given constant 1 or 0 inputs, pulse width modulated inputs and variable interval 1s and 0s inputs? **[LO7.3]** ★★★
9. How are actuators used for IoT based control and monitoring? **[LO 7.3]** ★
10. What are the signals and their uses in serial bus UART, I2C and CAN protocols? **[LO 7.4]** ★★
11. What is software library required for using the protocols for communication of sensors data? **[LO7.4]** ★★
12. How is RFID tag ID, location and time information communicated to the server for IoT applications? **[LO7.5]** ★★★
13. How is the RFIDs infrastructure established for IoT? **[LO7.5]** ★
14. What are the data-link, network, security and application layer protocols used in the WSNs? **[LO7.6]** ★
15. Why are the data aggregation, compaction and fusion needed before transmitting data from a WSN gateway and cluster head? **[LO7.6]** ★★★
16. How is infrastructure for IoT, which includes WSN infrastructure established? **[LO7.6]** ★★

Practice Exercises

1. Write the procedure for using a circuit using the resistance bridge, signal conditioning amplifier, ADC, serial port and microcontroller to sense change in resistance and thus the physical environment? **[LO 7.1]** ★

2. How is a capacitance bridge used to sense change in capacitances on a level detector in a tank? **[LO 7.1]** ★★
3. Using a diagram, explain how an array of 8 photo-transistors is used to sense the linear position of a linear moving machine part? **[LO 7.1]** ★★
4. How is an array of 8 photo-transistors used to sense the instantaneous orientation angle of a rotating shaft with precision of (360°/256)? **[LO 7.1]** ★★★
5. Describe the procedure for using ultrasonic sensing method for detecting railway track faults for prescriptive maintenance. **[LO 7.1]** ★★
6. Write the procedure for automotive IoT monitoring the driver's health during driving. **[LO 7.2]** ★★
7. Draw diagram for controlling a servomotor in an IoT application. **[LO 7.3]** ★
8. Write the sequence of bits and bytes when using UART for communication RFID ten characters ID. **[LO 7.4]** ★
9. List the steps for secure communication in Internet of RFIDs. **[LO 7.5]** ★
10. How can Internet of RFIDs be used in logistics? Explain using a diagram the stages used and communication between them. **[LO 7.5]** ★★
11. Draw a diagram showing Internet-connected wireless sensors network in a smart-home for remotely controlling, monitoring and providing the services. ★★★
12. Explain when and why is fixed network infrastructure and ad-hoc network infrastructure used in WSNs. **[LO 7.6]** ★★
13. State the applications and reasons for use of a multi-hop layered architecture. **[LO 7.6]** ★★★
14. State the applications and reasons for use of an ad-hoc network architecture in WSNs. **[LO 7.6]** ★★★

8

Prototyping the Embedded Devices for IoT and M2M

Learning Objectives

LO 8.1 Explain the basic concept of embedded systems

LO 8.2 Categorise the embedded device platforms, mobiles and tablets for prototyping and designing of IoT and M2M based systems

LO 8.3 Elaborate how the devices can always remain connected to the Internet and cloud

Recall from Previous Chapters

Section 1.1.1 gave the definition of IoT. Physical objects or things embedded with electronics, software and sensors connect to the Internet. They communicate data to applications and services. IoT enables achieving greater value and service by exchanging data with the manufacturer, operator and connected devices. Each thing is uniquely identifiable through a computing system embedded into that and each system is able to interoperate within the existing Internet infrastructure.

Recall the examples of applications of IoT or M2M systems from the previous chapters. Examples were given for applications and services for streetlights, smart homes, and traffic signalling control and monitoring, ATMs, automatic chocolate vending machines, predictive analytics and maintenance of vehicles, parking and waste container management services.

Each system needs electronic circuits for computation and communication. The circuits use the sensors and actuator devices, which embed the computing hardware and software. An embedded device means a device, which embeds software into a computing platform and performs the computations and communication for specific systems.

8.1 INTRODUCTION

Refer to IoT and M2M architecture concepts (Section 1.3). Devices generate data. The data is generated using embedded devices, sensors and systems at the physical layer. The data needs computations at a data-adaptation gateway. Enriched data communicates through the Internet for analytics, visualisation, knowledge discovery, applications and services.

A system needs electronic circuits for computation and communication. The circuits use sensors and actuator devices, which embed the computing hardware and software. Prototyping and designing require the embedded device platforms for data generation. This also requires connectivity to the Internet through computations, adaptation and networking. The software at application and application-support layer monitor and control the embedded devices, systems and machines using the actuators at the devices.

Following are some key terms, which need to be understood when learning principles of prototype designing for creating the devices for IoTs and M2M applications:

Embedded system denotes a system that embeds software into a computing platform. The system is dedicated for either an application(s), specific part of an application, product or a component of a large system.

Embedded device refers to a device, which embeds software into a computing platform and performs the computations and communication for specific systems.

Microcontroller unit (MCU) means a single-chip VLSI unit (also called microcomputer), which may be having limited computational capabilities. The MCU possesses memory, flash, enhanced input-output capabilities and a number of on-chip functional units.

Timer refers to a device which enables initiating new action(s) on timer start, on the clock inputs, time outs or when the number of clock inputs equal to a preset value.

Port refers to a device that enables input output (IO) communication between the MCU and another device such as a sensor or actuator or keypad or with an external computing device.

USB port connects the device hardware to a computer, downloads the developed codes into the device from the computer, or sends the codes from the device to the computer. USB port can also provide the power for charging the battery of the connected platform, thus an external charger is not needed.

GPIO pins refer to General Purpose Input-Output pins. A pin that can be used in addition to digital input and output for other purposes, such as Rx and Tx or SDA and SCK, PWMs, analog inputs, outputs or timer outputs. The Rx and Tx pins are used during UART protocol-based reception and transmission, SDA and SCK are used during use of I2C protocol-based serial data and clock communication.

Board is an electronic hardware—an electronic circuit board with MCU or SoC, circuits and connectors, which provide the connections to other ICs and circuit components. The ICs and circuit components can also be inserted or joined or put in place onto the board, by surface mount technology. The board may also have battery, power supply, voltage regulator or connections for the power.

Platform denotes a set consisting of computing and communication hardware, software and operating system (OS). A platform enables working with different software, APIs, IDE and middleware. A platform may enable the development of codes at the development stage. It may also enable prototype development for an application(s) or specific parts of an application.

Module (hardware) is smaller form-factor hardware which can be placed onto a board. The module may embed the software. It may enable use of the board circuit with shorter form factor. An example is RF module placed onto an electronic board.

Shield means a supporting circuit with connection pins, socket(s) and supporting software. The supporting circuit enables the connectivity of a board or computing platform to external circuits. The circuit connects the elements that can be plugged onto a board or platform. Usage of the supporting circuit provides extra features, such as connectivity to wireless devices, such as ZigBee, ZigBee IP, and Bluetooth LE, Wi-Fi or GSM or RF module or to a wired device, such as Ethernet shield. The Ethernet shield enables wired connection of a platform to the Ethernet controller and through an external standard LAN socket enables connectivity to a wired or Wi-Fi modem for the Internet. Shield is the term used in Arduino hardware for the supporting circuits.

Header means plastic-coated strip or plastic-capped plug-in which is placed on top of the pin holes when making connection of the wires without electronic soldering. A header also provides jumpers. Six-pin header means plastic 6-pin plug-in that connects the 6 pin holes. This header is a component, distinct from enveloping words in a data stack for communication at a layer according to a protocol.

Jumper denotes a wire with a solid tip at each end which is normally used to interconnect the components on an electronic-circuit breadboard. Jumpers are used for transferring IOs or signals to or from the pins.

Interrupt means an action in which a running program interrupts an hardware signal, such as timer timeout or on execution of a software instruction for interrupt. For example, a program interrupts for sending the data or for accepting a newly added device in the system or on execution of the INT instruction.

Integrated Development Environment (IDE) means a set of software components and modules which provide the software environment for developing and prototyping.

Operating system (OS) is a system software which facilitates the running of processes, allocation of memory, system calls to the IOs, facilitates the use of network subsystems, and which does devices management, priority allocations of processes and threads, and

enables multitasking and running of number of threads. The OS enables many system functions using the given computing device hardware.

Section 8.2 describes the embedded computing basics, i.e. embedded devices hardware and software. Sections 8.3 describes the popular embedded devices platforms (Arduino, Intel Galileo and Edison, Raspberry Pi, Beagle Bone, and mBed™), mobiles and tablets for prototyping and designing for IoT and M2M applications and services. Section 8.4 describes how the devices remain always connected to the Internet and cloud.

8.2 EMBEDDED COMPUTING BASICS

LO 8.1 Explain the basic concept of embedded systems

Embedding means embedding function software into a computing hardware to enable a system function for the specific dedicated applications. A device embeds software into the computing and communication hardware, and the device functions for the applications.

A reader can study in detail the architecture, programming and designing of the embedded systems in this book[1].

8.2.1 Embedded Software

Software consists of instructions, commands and data. A computing and communicating device needs software. Software does the bootloading and enables the applications and services. The software includes an OS. A device embeds software which also includes the device APIs and middleware which enable the device to perform computing and communication functions.

Bootloader

Bootloader is a program which runs at the start of a computing device, such as an MCU. A bootloader initiates loading of system software (OS) when the system power is switched on, and power-on-self test completes.

Bootloader may also facilitate the use of system hardware and networking capabilities. Bootloader loading of the OS may be from an external source. Alternatively, bootloader can itself function as a system software, when the codes for IO functions and OS basic system functions happen to be from the same source. Booting is said to complete when normal, operational runtime environment is attained.

Operating System

An operating system (OS) facilitates the use of system hardware and networking capabilities. When a load of the OS into RAM completes then the MCU starts the normal

[1]Raj Kamal, *Embedded Systems—Architecture, Programming and Design*, 3rd edition, McGraw-Hill Education, New Delhi, 2014.

operational runtime environment. When the device is executing multiple tasks or threads, then also an OS is required. The OS controls the multiple processes and device functions. Process, task and thread are the set of instructions which run under the control of the OS. The OS enables memory allocation to different processes, and prioritising of the processes enables the use of network hardware and device hardware functions and execution of software components and processes. The OS is at flash memory of the device. It may be required to load functions at the device RAM.

An OS may be open source, such as Linux or its distribution. Linux distribution means a package or set of software components and modules bundled together for specific functions or for specific hardware, and distributed for wider usages and applications. For example, Arduino Linux distribution runs in Arduino circuit boards and the Linux functions enable developing the application programs for using the Arduino.

Real-Time Operating System

Real-Time Operating System (RTOS) is an OS that enables real-time execution of processes on computing and communication hardware. RTOS uses prioritisation and priority allocation concept to enable the execution of processes in real-time.

8.2.2 Integrated Development Environment

Application software codes are perfected by a number of cycle of runs and tests. A cycle in the development phase consists of editing-testing-debugging. The cycles repeat during different development phases till the system has thoroughly tested and debugged the software. A system develops in a larger time-frame than the hardware circuit design.

Integrated Development Environment (IDE) is a set of software components and modules which provide the software and hardware environment for developing and prototyping. An IDE enables the codes development on a computer, and later on enables download of the codes on the hardware platform. IDE enables software that communicates with the Internet web server or cloud service.

IDE consists of the device APIs, libraries, compilers, RTOS, simulator, editor, assembler, debugger, emulators, logic analyser, application codes' burner for flash, EPROM and EEPROM and other software components for integrated development of a system. IDE may be open source. For example, Arduino has open source IDE from the Arduino website.

The IDE or prototype tool enables a prototype design. IDE is used for developing embedded hardware and software platforms, simulating, and debugging. IDE is a tool for software development of embedded devices, which makes application development easy. For example, a software serial library is provided in an IDE for a microcontroller system. The library consists of a number of programs. The library has programs for each serial-interface protocol, which can be used in the device. The program enables using the

protocol-specific programs directly, such as using a program for reading an RFID tag or using a program for sending data to the USB port for onward transmission on the Internet (Section 7.5.1).

Simulator

Simulator is software that enables development on the computer without any hardware, and then prototyping hardware can be connected for embedding the software and further tests.

APIs and Device Interfaces

Recall Section 1.4.1. Major components of IoT devices are software. Software consists of device Application Programming Interfaces (APIs) and device interface for communication over the network and communication circuit/port(s) which also includes a middleware. The middleware creates IPv4, IPv6, 6LowPAN, MQTT, COAP, LWM2M, REST and other communication protocol stacks.

Device Interfaces

A connectivity interface consists of communication APIs, device interfaces and processing units. Software commands the action on the message or information received followed by hardware port outputs for the actuators.

8.2.3 Embedded Hardware Units

The hardware includes the following:

- Single VLSI (very-large integrated) chip
- A core in an application specific instruction set processor (ASIP), called MCU
- A core in an application specific integrated circuit (ASIC) core
- A core in System-On-Chip (SoC) or an SoC chip with an SD card for embedded software and operating system (OS) software.

Following subsections describe MCU, SoC and selection of platform for prototyping.

Microcontroller Unit

An MCU is a single-chip VLSI unit (also called microcomputer) which though having limited computational capabilities, possesses enhanced input-output capability and has a number of on-chip functional units, such as Internal RAM, flash, IO ports, GPIOs, serial interfaces, timers, serial ports and timers.

Application-specific MCUs have additional on-chip functional units, such as PWM circuits (1, 2 or 3), ADC (1, 2, 4 or higher) and other functional units. Figure 8.1 shows the on-chip functional units in a microcontroller.

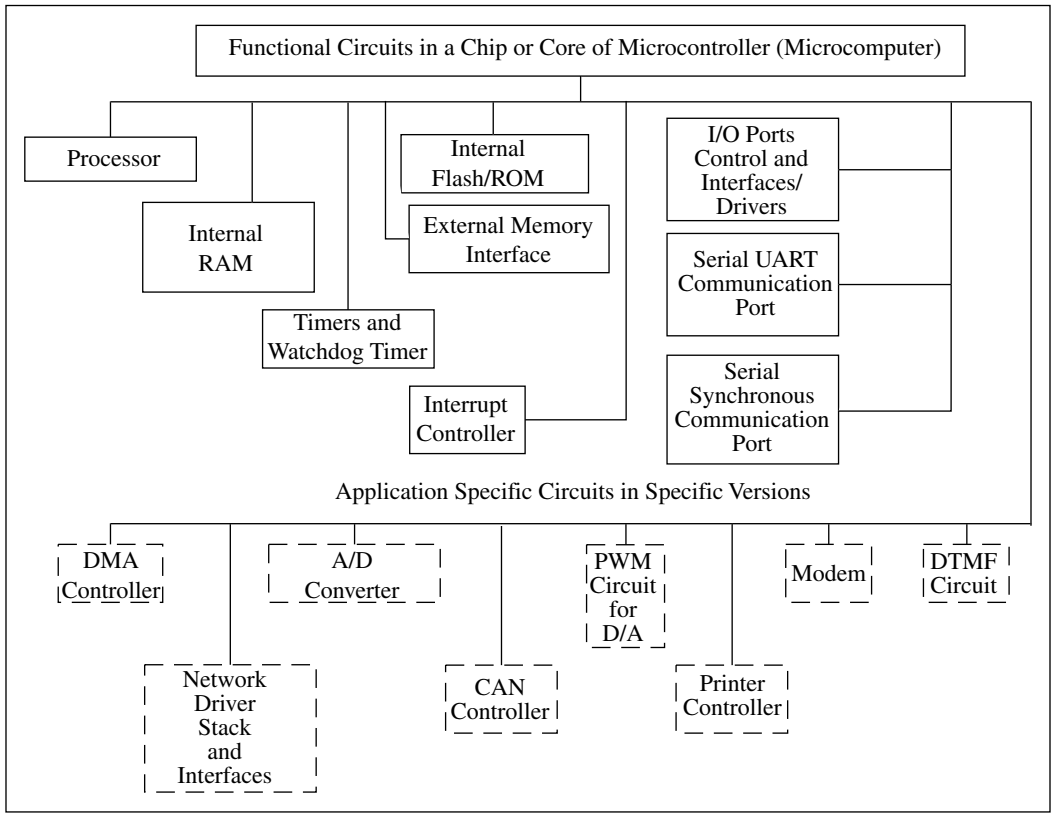


Figure 8.1 Microcontroller, on-chip functional units and application specific units

Microcontroller chips are from different sources and are of different families, groups and versions

An MCU is an IC chip, available from a number of sources, such as ATMEL, Nexperia, Microchip, Texas Instruments or Intel. A source may manufacture different types, families and groups of MCUs, like ATMEL manufactures AVR®8 and AVR®32 families of MCUs. A family of MCUs can have different versions. Following are the considerations when using a specific MCU version from a source, family or group of MCUs.

- MCU can be of 8-bit, 16-bit or 32-bit family.
- MCU clock frequency can be 8 MHz, 16 MHz, 100 MHz, 200 MHz or higher. The clock frequency depends on the version and family. Performance defines number of instructions executed per sec that primarily depends on the clock frequency also. A metric for performance is Million Instructions Per Second (MIPS). Another metric for performance is Million Floating Point Operations Per Second (MFLOPS).
- MCU includes RAM which can be 4 kB, 16 kB, 32 B or higher. RAM read and write of byte takes an instruction cycle each. RAM is used for temporary variables, stacks and run-time need of the memory.

- MCU includes EEPROM and flash memory, which may be 512 B, 1 kB, 2 kB, 4 kB, 16 kB, 64 kB, 128 kB, 512 kB or higher. Flash stores the programs, data, tables and required information during building and testing stages, and then stores a final version of the application program in the embedded device.
- MCU includes timers, I/O ports, GPIO pins, serial synchronous and asynchronous ports and interrupt controllers.
- MCU includes several functional units in specific version, such as ADC, multi-channel ADC or ADC with programmable positive and negative reference voltage pins, PWMs, RTC, I2C, CAN and USB ports, LCD interface, ZigBee interface, Ethernet, modem or other functional units, depending on a specific source, family, group and version.

Example 8.1 gives the basic hardware units in AVR®8 MCUs and required functional units for many applications.

Example 8.1

Problem

What are the basic hardware units in AVR®8 group of microcontrollers? What are the advanced features in the AVR®32?

Solution

ATMEL manufactures AVR® MCUs. Basic hardware units in an AVR®8 are:

- AVR® 8 is an 8-bit MCU.
- It has instructions for executing in single-clock cycle.
- Three groups of AVR MCUs are tiny, mega and xmega. Tiny has 512 B to 16 kB program memory, mega has 4 kB to 512 kB program memory and xmega has 16 kB to 384 kB program memory.
- Application-specific additional controller units for LCD, CAN, USB, ZigBee and advanced PWM, and fuse programmable additional FPGA (5 to 40 k gates).
- It has on-chip RAM (SDRAM).
- Serial SPI for synchronous serial communication, USART for serial synchronous-asynchronous receiver and transmitter. Two-wire serial interface (TWI) and in-circuit serial programming (ICSP) through serial interface.
- ADC and analog voltage comparator.
- Three timer-counters and watchdog timer.
- Four ports with programmable port pins as input or output.
- Additional units as per the version.
- Serial Peripheral Interface (SPI) is serial communication in master slave mode. SPI signals are master in slave out (MISO), master out slave in (MOSI), CLK (serial clock from master and SS (slave select).
- AVR®32 has an 8-bit arithmetic logic unit (ALU), 32-bit registers, digital signal processing (DSP) instructions, single instruction multiple data (SIMD) instructions. These features enable support for video processing.

System-on-Chip

Complex embedded devices, such as mobile phones, consist of a circuit which is designed on a single silicon chip. The circuit consists of multiple processors, hardware units and software. An SoC is a system on a VLSI chip that has multiple processors, software and all the needed digital as well as analog circuits' on chip. An SoC embeds processing circuit with memory and is specific to dedicated application(s) and may have the analog circuits.

SoC may associate an external SD card in a mobile phone. The card stores the external programs and operating system and enables the use of the chip distinctly for distinct purposes. Secure Digital Association created the SDIO (Secure Data Input-Output) card. The card consists of standard, mini, micro or nano form. It consists of flash memory and communication protocol. An SoC can be from different sources, for examples, Raspberry Pi and BeagleBone.

Selection of Embedded Platform

Selection among the number of different available platforms depends on a number of factors like price, open source availability, ease of application development and needed capabilities, performance required from IoT device and suitability for developing and using for prototyping and designing.

Hardware

The choice, beside the price, of embedded hardware depends on the following:

- Processor speed required which depends upon the applications and services. For example, image and video processing need the high-speed processors
- RAM need which may be 4 kB or higher depending upon the OS and applications. For example, requirement is 256 kB for using the Linux distribution. New generation mobile phones have over one GB RAM.
- Connection needs to ZigBee, ZigBee IP, Bluetooth LE, Wi-Fi or Wired Ethernet for networking using a supporting circuit (shield)
- USB host
- Sensor, actuator and controllers interfacing circuits, such as ADC, UART, I2C, SPI, CAN single or multiples
- Power requirements, V- and V+, 0 V and 3.3 V or 0 V and 5 V or other.

Software Platforms and Components

Selection among a number of different available software depends on hardware platform, open source availability of software components, cost of availability or development of other required components for applications and services.

The choice of embedded software, beside the price, depends on the following: IDE with device APIs, libraries, OS or RTOS, emulator, simulator and other environment components, middleware with communication and Internet protocols, and Cloud and sensor-cloud platform for applications development, data storage and services.

Example 8.2 explains a few open source software.

Example 8.2

Problem

Give examples of IoT open source frameworks for implementation tools, web services, middleware and cloud services.

Solution

Open Source Framework for IoT Implementation

Refer Section 1.4.3. Eclipse IoT (www.iot.eclipse.org) provides open source implementations of a number of standards including MQTT CoAP, LWM2M and services and frameworks that enable an Open Internet of Things software. Eclipse tool work with Lua. An IoT programming language is Lua. IoT can be programmed in any open source language like Python or Java or PHP also using the tools.

Arduino development tools provide a set of software that includes an IDE and the Arduino programming language for a hardware specification for interactive electronics that can sense and control more of the physical world (Section 9.3 for details).

Middleware

OpenIoT is an open source middleware. It enables communication with sensor clouds and enables cloud-based 'sensing as a service', IoTSyS is middleware. It enables provisioning of communication stack for smart devices using IPv6 and a number of other standard protocols (Section 1.4.4).

Web Services

Web server or cloud server or clients use SOAP, REST, JSON, HTTP, HTTPS, web sockets, web APIs, URI. (Section 3.4) These provide the building blocks (software components) for writing the codes for Web Services.

Cloud Applications Development Platforms

Refer Section 6.4. The cloud-based development platforms are also widely used for IoT because of world wide availability, storage and platform specific features.

Xively (Pachube/Cosm) is another cloud application development platform for sensor data. Xively is open source for basic services (Section 6.4.1). It is software and server platform for data capture, data visualization real-time and other features over the Internet. It is an open source platform for Arduino open-source electronics prototyping platform connectivity with web.

Nimbits enables IoT on an open source distributed cloud. PaaS at the Nimbits cloud deploys an instance of Nimbits server at the device nodes (Section 6.4.2).

Reconfirm Your Understanding

- IoT design needs hardware, sensors, actuators, embedded platform, interfaces, firmware, communication protocols, Internet connections, data storage, analytics and machine learning tools.
- Prototype design for IoT embedded systems and circuits for sensors and actuators need the devices for computations and communications. A device consists of hardware which consists of a microcontroller, ASIC or SoC. The device software embeds for its functions.

- Microcontroller has processor, internal RAM, ROM or flash, timers, serial interfaces, IO ports and number of application-specific functional units.
- SoC consists of multiple processors, software and all the needed digital as well as analog circuits.
- An embedded device platform needs an IDE and development tools for prototyping and designing.
- Selection among number of different available software depends on hardware platform, open source availability of software components, embedded platform and software integration tools, cost of availability or development of other required components for applications and services.
- Open source software can be used. Examples are OpenIoT and Eclipse IoT stack.
- Cloud-based application development platform, such as Xively or Nimbits, Device Hub, Cloud.com, Cisco IoT, IOx and Fog, IBM Bluemix, TCS CUP can be used.

Self-Assessment Exercise

1. What are the roles of major components of IoT devices? ★
2. List the microcontroller functional units in AVR®8. ★
3. What are the advantages of using SoC with external SD micro or nano card? ★★
4. List the functional programs in an IDE. Why is IDE an important tool for embedded device development? ★★
5. List the features in open IoT and Eclipse IoT. ★
6. Give examples of open source frameworks for IoT Implementation tools, web services, middleware and cloud services for using embedded computing platforms. ★★★

8.3 EMBEDDED PLATFORMS FOR PROTOTYPING

LO 8.2 Categorise the embedded device platforms, mobiles and tablets for prototyping and designing of IoT and M2M based systems

Designing a product needs a prototype development first. A standard source board enables prototyping, an easy task for number of IoT and M2M devices. This is because of the open source availabilities of IDE, middleware and software components from number of sources and forums for the board.

Several standard popular boards, modules and supporting circuits (shields) are available from a number of sources. Following subsections describe the features and usages of Arduino, Intel® Galileo and Edison, Raspberry Pi, BeagleBone and mBed boards. A board uses MCU as embedding platform for creating the IoT, M2M and wearable devices.

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

8.3.1 Arduino

Arduino boards AVR MCU based products, modules and shields are popular and use the for easy prototype development

Arduino boards, modules and shields are popular AVR® MCU-based products. Each board has clear markings on the connection pins, sockets and in-circuit connections. Thus, Arduino boards are easy to work for DIY (do-it-yourself) and simplify the prototyping of embedded platforms for IoTs.

The IDEs are open source. Arduino boards are thus easy to program. For example, Arduino Uno board is a robust as well as widely used board to get started with electronics and coding. Uno is most used and documented board of the whole Arduino family at present.

The board's analog input pins and PWM pins can connect sensors, actuators and analog circuits. The board's digital I/O pins can connect On-Off states, set of On-Off states, digital inputs from sensors, digital outputs to actuators and other digital circuits. A board with a shield inserted into it makes a wireless connection to a ZigBee, Bluetooth LE, WiFi, GSM, or RF module or a wired connection to Ethernet LAN for the Internet.

Development boards for IoT devices are the Arduino Ethernet, Arduino Wi-Fi and Arduino GSM shields. Development boards for the wearable devices are Arduino Gemma, LilyPad, LilyPad Simple/SimpleSnap and LilyPad USB.

A board has pre-programmed bootloader in the MCU. Bootloader software program embeds onto the AVR® MCU chip. Bootloader can also be downloaded using USB connection to a computer or tablet. Bootloader enables use of the AVR platform with the Arduino IDE. Bootloader enables the board functions. The board needs no OS after bootloading by default. An OS can be embedded when required, for example, when system needs to do multiple tasks or needs to run multiple threads or processes. When a programmer develops the codes using an IDE, the codes are pushed into the MCU using USB port of the board. The codes are pushed after developing-testing-debugging cycle (s).

A programmer develops the codes using the editor in an IDE. Then these are downloaded onto the board, tested and debugged. This constitutes one cycle. Downloading is through a USB port interconnecting the board and external computer or tablet with an IDE. The cycle needs to be repeated till the codes are finalized for a prototype.

Figure 8.2 shows architecture of Arduino Fio board with Ethernet shield.

IoT Applications

An Arduino application can be an embedded-device data-connectivity to the Internet and data store on cloud. Example of an Arduino application for IoT is Internet of streetlights (Example 1.2). Arduino board find applications where the device does not require intensive computing and graphics. The applications are using *things* which are light-emitting devices, wearable devices, health monitoring or fitness devices, watches, sensors and actuators connected smartly through the Internet. The developments tools are open sources that use the computer with Windows, Arduino Linux distribution or a MAC.

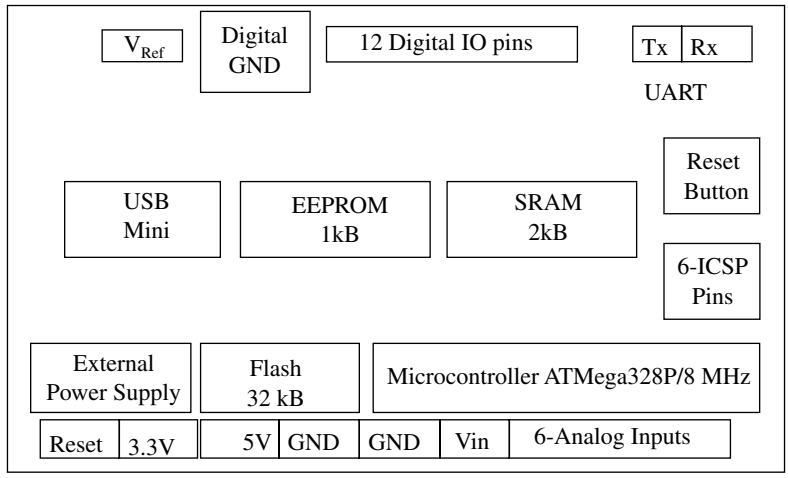


Figure 8.2 Architecture of Arduino Fio board for development IoT devices development

Features

Arduino Uno

Arduino Uno is a reference model for the Arduino platform. The Uno is an MCU board. It includes an ICSP 6-pin header, which enables embedding (burning) of programs. The process of programming an EEPROM/ROM is called burning. ICSP is in-circuit serial programming, which means burning the code through connectivity with the ICSP header.

Uno board includes a USB connection, a power jack and reset button (to start the board and run Bootloader). USB is also used for pushing programs on to the board after development using IDE on a computer.

Arduino Uno, and IoT and Wearable Devices Arduino Boards

Table 8.1 lists the features of Uno and IoT and wearable device boards as the examples. Additionally, the features of Arduino ARM based platform are also given.

Table 8.1 Features of UNO and other IoT device and wearable device boards

Board/ Shield	Application	AVR® Microcontroller/ Clock	Operating/ input V	EEPROM/ SRAM/ Flash	Analog In/ Out/Digital IOs/*n*-bit PWM	USB/ UART	Ethernet/ Wi-Fi/ GSM
Due	Fast computations, ARM based MCU	ATSAMSX8I	3.3 V/ 7 V–12 V	0 kB/ 96 kB/ 512 MB	12/2/54/ 12	2 micro/4	0/0/0

(*Contd.*)

UNO	Getting started with electronics and coding	ATMega328/ 16 MHz	5 V/7 V–12 V	1 kB/2 kB/ 32 kB	6/0/14/6	Standard/ 1	0/0/0
Yun	IoT	(i) ATmega 32U4/ 16MHz (ii) AR9331 Linux/ 400 MHz	5 V	(i) 1 kB/ 2.5 MB/ 32 MB (ii) 1 kB/ 16 MB/ 64 MB	(i) 20/7/ 12/0	Micro/1	0/0/0
Ethernet	IoT	ATMega328/ 16 MHz	5 V/ 7 V–12 V	1 kB/2 kB/ 32 kB	6/0/14/4	Standard/ 0	1/0/0
Fio	IoT	ATMega328P/ 8 MHz	3.3 V / 3.7 V–7 V	1 kB/2 kB/ 32 kB	8/0/14/6	Mini/1	0/0/0
Gemma	Wearable	ATtiny85/8 MHz	3.3 V/ 4–16 V	512 B/512 B/8 kB	1/0/3/2	Micro/0	0/0/0
LilyPad	Wearable	ATmega168V/ 8 MHz ATmega328P/ 8 MHz	2.7–5.5 V/ 2.7–5.5 V	512 B/1 kB/ 16 kB	6/0/14/6	0/0	0/0/0
LilyPad SimpleSnap	Wearable	ATmega328P/ 8 MHz	2.7–5.5 V/ 2.5–5.5 V	512 B/ 512 B/8 kB	4/0/9/4	0/0	0/0/0
LilyPadUSB	Wearable	ATmega32U4/ 8 MHz	3.3 V/ 3.8–5 V	1 kB/ 2.5 kB/32 kB	4/0/9/4	Micro/0	0/0/0

Table 8.2 lists the features of ARM based Arduino for IoT devices and wearable devices board, 'due' for fast computations and communication.

Table 8.2 Features of ARM-based Arduino for IoT device and wearable device board

Board	Application	ARM Microcontroller/ Clock	Operating/ input V	EEPROM/ SRAM/ Flash	Analog In/ Out/Digital IOs/*n*-bit PWM	USB/ UART	Ethernet/ Wi-Fi/ GSM
Due (ARM based MCU)	Fast computations,	ATSAMSX8I 84 MHz	3.3 V/ 7 V–12 V	0 kB/ 96 kB/ 512 MB	12/2/54/ 12	Two micro USB/4	0/0/0

Two boards are Arduino-R3 and Arduino Yun. Example 8.3 gives the special features of these.

Example 8.3

Problem

List the special features in Arduino-R3 and Arduino Yun boards.

Solution

Arduino Uno-R3 is an enhancement of Uno. It uses ATmega16U2 in place of the 8U2. It requires no drivers needed for Linux or Mac. IDE has the ability to have the Uno show up just like a keyboard or mouse. I2C signals SDA and SCL pins added. IOREF pin and future use pin added. IOREF specifies that Board IO voltages as reference to the shield. The Uno R3 is compatible to all existing shields and adapts to new shields which use new pins.

Arduino Yun combines the Arduino-based board with Linux. This is because the two processors are ATmega32u4 for support to Arduino and Atheros AR9331 for running Linux. IoT applications enablers are Wi-Fi, Ethernet support, a USB port, micro-SD card slot, three reset buttons and more.[2] The Yun can be controlled from anywhere with any Internet-connected web browser without assigning an IP address to the board. WebSockets can also be used for providing real time full-duplex communication over TCP.

Features which make Arduino boards widely used are:

1. Prototyping ease
2. Flexibility and ease of assembling modules on the board
3. Open extensible source code, schematics, software, middleware and IDE; the AVR-C codes extend on coding in C++, and the libraries can be added with additional programs, AVR-C codes means C commands and statements for using AVR ports, serial interfaces and other functional units of the MCU
4. IDE latest version and appropriate OS are open source[3] IDE and software runs on multiple environments, Linux, Windows and Mac OS-X
5. Number of times programmability of the board during the editing-testing-debugging cycles, and for development of number of new prototype using the same
6. Hardware open source and extensible using the modules, shields and other circuits with open version of IDE, software modules and codes from other designers

8.3.2 Intel®Galileo

Intel®Galileo Gen 2 boards are Arduino certified boards for development and prototyping. A Galileo is based on the Intel® Pentium architecture which includes features of single threaded, single core and 400 MHz constant speed processor. An example is use of Intel Quark SoC X1000 Application processor. No separate graphic and video processors support is included in the board. The Galileo is hardware and software pin-to pin compatible with shields the designed for Arduino Uno R3 and Arduino IDEs.

Intel® Galileo Gen 2 boards use Intel Pentium architecture processor, 8 MB flash and SD card, Arduino R3 compatible connections, IOs and drivers, and run Linux distributions

[2] http://asynkronix.se/internet-of-things-with-arduino-yun-and-yaler/
[3] http://www.arduino.cc/en/Main/Software

Galileo additionally provides large 8 MB SPI flash to store firmware (Bootloader) and enables the users to incorporate Linux firmware calls in their Arduino sketch programming. Intel Galileo Arduino SW (IDE and drivers) are given at Intel communities website.[4] Galileo permits boot off and store drivers in the SD card. Galileo[5] supports a set of 30 sensors and accessories for Arduino. The usages can be learnt from the demonstration of usages and Linux images.[6]

IoT Applications

Examples of Galileo board applications are making smart everyday '*things*' such as health monitoring or fitness devices, watches, sensors and cameras. The board has the facility of development of the codes on a personal computer (PC) with the board running Linux. The development tools and IDE are at the connected computer. The codes run using Windows, Arduino Linux distribution, Linux, or MAC on the PC or tablet.

Galileos best IoT applications for the "things" are health-monitoring or fitness devices, cameras, which require the facility of personal computer board running Linux

Features

Figure 8.3 show architecture of Intel Galileo Gen 2 board for advanced computer functionalities with network connectivity for development of IoTs.

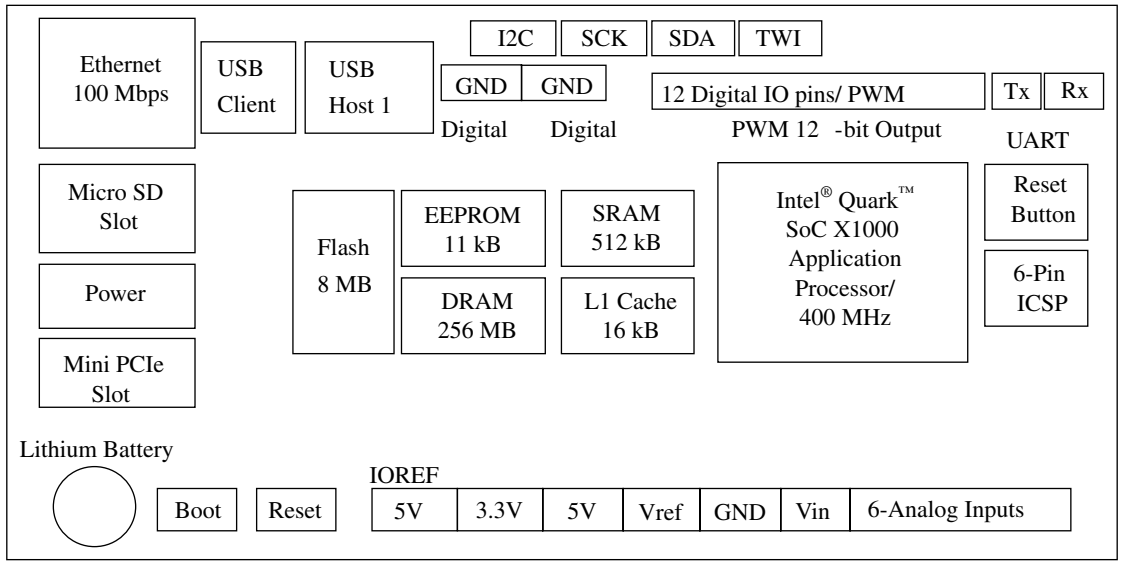


Figure 8.3 Architecture of Intel Galileo Gen 2 board for advanced computer functionalities with network connectivity for development of IoTs

[4] https://communities.intel.com/docs/DOC-22226
[5] http://www.dfrobot.com/index.php?route=product/product&product_id=725.
[6] http://www.intel-software-academic-program.com/pages/courses

Table 8.3 lists the features of Intel® Galileo Gen 2 board.

Table 8.3 Features of Intel® Galileo Gen 2 board

Board/ Shield	Application	SoC Processor/ Clock	Operating/ input V	L1 cache/ EEPRO/ SRAM/ Flash/ DRAM	Analog In/ Out/Digital IOs/12-bit PWM/Micro SD/IOREF (for shield IOs)	USB host/ USB/ Client/ UART Rx and Tx /I2C SDA-SCK/ RS232	Ethernet port/ Wi-Fi Adapter/ GSM
Intel® Galileo Gen 2	Advanced compute functiona-lity and network connecti-vity	Intel® Quark™ SoC X1000 application processor/ 400 MHz	5 V/ 7 V–12 V	16 kB/ 11 kB/ 512 kB/ 8 MB/ 256 MB	6/0/14/6/1/1	Standard host/ Standard client// 1 pair/1 pair/ 1	100 Mbps/ N-2200 Mini-PCIe /0/

Features which makes Galileo boards widely used are:

1. Prototyping ease with single board computations and networking support
2. Node.js and C programming languages, Arduino codes open extensible source code, schematics, software, middleware and IDE; the AVR-C codes extend on coding in C++, and the libraries can be added with additional programs, AVR-C codes means C commands and statements for using AVR ports, serial interfaces and other functional units of the MCU
3. IDE latest version and appropriate OS from open source
4. IDE and software runs on multiple environments, Linux, Windows and Mac OS X
5. Programmability number of times on downloading of codes through USB port, which enables the number of times download occurs during edit-test and debug cycles
6. On-board 8 MB NOR Flash, IOREF for 5V IOREF to shield in place of 3.3 V IOs, 12-bit pulse width modulation (PWM), console UART1 redirection to Arduino compatible headers, 12V Power-over-Ethernet (PoE) capability, a power regulation system that accepts power supplies from 7V to 15V, reset button, integrated Real-Time Clock (RTC) with optional 3V 'coin cell' battery for operation between turn on cycles and reset button to reset the sketch and any attached shields
7. Flexibility and ease of connecting the extended memory and hardware connectivity board to external a full-sized mini-PCI Peripheral Connect Interconnect Express (PCIe) slot (which also functions as Wi-Fi adapter), Ethernet port socket, Micro-SD slot
8. Extended interfacing capabilities using SPI, several PC industry standard I/O ports and features to expand native usage and capabilities beyond the Arduino shield additions, 6-pin ICSP, 3.3V USB TTL UART header, USB host port, USB client port, and I2C port

Applications of Intel Galileo are in IoT devices needing intensive computations for examples, camera in smart homes and IoT of ATMs.

8.3.3 Intel® Edison

Intel® Edison[7] is a high performance computation and communication module. It includes processor core in the SoC that is dual core, dual threaded Intel ATOM x86 CPU running at 500 MHz, while in Galileo the core is Intel Quark X1000 400 MHz single threaded, single core. RAM is 1 GB in size which is four times that of Galileo. Edison includes Wi-Fi and Bluetooth LE communication interfaces. The interfaces enable seamless device internetworking and device-to-cloud communication. The interface thus enables rapid prototype development produce IoT and wearable computing devices.

Edison can be used for compatibility with Arduino board as well as independently with smaller form factor board. It enables creation of prototypes and fast development of prototyping projects.

Edition includes tools for collect, store, and process data in the cloud, trigger alerts when using advanced analytics and OLTP and OLAP of the data streams. It has higher performance.[8]

8.3.4 Raspberry Pi

Raspberry Pi 3 (RPi 3) model B is the latest (February 2016) single board SoC based computing and communication board. An RPi runs on the OSes (Windows 10 IoT Core, RISC OS, FreeBSD, NetBSD, Plan9, Inferno, AROS and additional distributions of Linux such as Raspbian Ubuntu) on the board. The RPi includes hardware and software provides high performance computing and graphics.

Raspberry Pi 2 model B+ runs on quad core ARM Cortex and runs Window 10 IoT Core and additional distributions of Linux Operating systems

RPi is for home automation and drones, for devices which need an OS that is different from that of traditional PCs, such as Ubuntu Core (also known as Snappy). The core is a stripped down version of Ubuntu, designed to run securely on autonomous machines, M2M and IoT devices.

The SoC at RPi board uses the processor (ARM Cortex quad core) plus a graphic processor (Broadcom VideoCore IV) for graphics and video. The power required is 4 W. Memory on-board 1 GB, plus in the multimedia card module memory support plus SD card (model B) or MicroSDHC card (model B+) slot for external SD and microSD cards. RPi provisions for no real-time clock (RTC) and therefore an external chip is used for including an RTC.

[7] http://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison_pb_331179002.pdf
[8] http://stackoverflow.com/questions/26978356/compare-intel-galileo-and-intel-edison

IoT Applications

An RPi application is media server IoT devices. RPi board functions as a personal computer. Its applications are in networked security camera systems in home automation or ATMs applications and services. Example 8.4 gives Raspberry Pi applications in IIoT.

Example 8.4

Problem

Give an example of Raspberry Pi applications for Industrial IoT.

Solution

Raspberry Pi applications for Industrial IoT can use ready-to-run software for RPi called 'Echelon'. The Echelon IIoT platform has software for the core requirements for the IIoT that include the REST APIs for web connectivity, load control devices, autonomous control, security, reliability, wired or wireless devices, physical sensors data collection and send to other devices.

Features

RPi clock speed is about 40 times that in Arduino and larger RAM compared to Arduino. Table 8.4 lists the features of the Raspberry Pi 2 model B + board.

Table 8.4 Features of Raspberry Pi 2 model B + board

Board/ Shield	Application	SoC Processor/ Clock	Opera-ting/ Input V	L1 Cache/ L2 Cache/ RAM	Digital IO/ PWM/ Micro SD/	USB hosts/ UART Rx and Tx/I2C SDA-SCK/ SPI	Ether-net port/ HDMI out/ Camera Port/ GSM
Raspberry Pi 2 model B+	Advanced compute, and graphic and network connectivity functionality like a single board computer	Quadcore ARM Cortex-A7/ 900 MHz	5 V/ 7 V–12 V	16 B/ 128 kB/ 1 GB RAM	40/1/1	Standard hosts 2/ 1 pair/1 pair/1 with two chip select signals	1/ 1 /1/0

Features besides very low performance to cost ratio, which makes RPi boards widely used are:

- Computer-like prototyping ease for developing media server and home or ATM surveillance systems IoT applications and services

- Coding in Python, C++, and the libraries
- Software runs on multiple environments, Python, Scratch, Squeek, IDLE, C, Linux and BSD OSes, Windows 10 and several OSes with external keyboard and display monitor
- Flexibility and ease of connecting the hardware to external systems, connectivity of RPi takes place through two USB hosts hub and Ethernet connector
- Connectivity to the extended memory through a micro-SD slot
- Extended interfacing capabilities using, SPI, UART, I2C, 40 GPIO pins, supports Wi-Fi module, stereo audio, video with Pi Camera module, and Stream of High-definition HDMI output.

The comparisons of performances of processing and graphics between RPi and Galileo show the Galileo performance higher though both have single core processors and Galileo has 400 MHz processor while RPi 700 MHz.[9]

RPi best usages are the media applications for IoT such as photos or video. Galileo on the other hand is best when requiring sensors, high memory and processing power and a RTC.

8.3.5 BeagleBone

BeagleBone-X15 runs SoC which uses the fast processor and DSP processor for graphics and video

Texas Instruments BeagleBone-X15 (BB-X15) is the latest (November 2015) single-board computer for computing and communication. BB runs on the OS Linux, RISC OS, FreeBSD, OpenBSD and additional distributions of Linux such as Ubuntu boards. The SoC uses the processor (ARM Cortex A15 core) and DSP processor (TMS320C64x plus multimedia 4 GB eMMC) for graphics and video. The power required is 2 W. Memory on-board is 2 GB, plus in the memory support plus and microSD cards.

IoT Applications

BB-X15 best applications are for IoT applications requiring fast media, 2D and 3D graphics, and video streaming multimedia

BB-X15 is used when high performance is required. BB applications are for media, 2D and 3D graphics, and video servers. Performance is nearly two times faster than RPi 2.[10] Table 8.5 lists the features of BeagleBone-X15 board.

Features which make BB boards highly useful for media and video processing are listed as:

- Single-board computer and communication board
- Prototyping ease for media, graphics and video servers needing IoT applications
- IDEs for BB includes environment for code development in Python, Scratch, Squeek, Cloud9/Linux, C, Linux and BSD OSes

[9] http://www.mouser.com/applications/open-source-hardware-galileo-pi/
[10] http://makezine.com/2013/04/15/arduino-uno-vs-beaglebone-vs-raspberry-pi/

Table 8.5 Features of BeagleBone X15 board

Board/ Shield	Application	SoC Processor/ Clock	Operating/ input V	SDRAM memory/ multi-media card (MMC) Memory	Analog In/Out/ Digital IOs/ 12-bit PWM/ Micro SD	USB 3.0/ UART Rx and Tx /I2C SDA-SCK/ CAN/SPI	Ethernet port/ PCIe 2-Ch/ Audio Adapter
Texas Instruments Beagle Bone-X15	Advanced compute functionality and network connectivity with single-board computer	Dual core ARM A15/ 1.5 GHz; Dual Core graphics SGX 544 3D/512 MHz; DSP dual C66x/ 700 MHz,	5 V/ 7 V–12 V	DDR L3 2 GB/ eMMC 4 GB	1/0/157/1/1	three hubs/ 1 pair/ 1 pair/1/ 1	1 Gbps × 2/ 1 /1

- Programmability for number of times downloading of codes takes place through USB port during edit-test-debug cycles of development
- Pulse width modulation (PWM), CAN/SPI//DVI-D, Integrated real-time clock (RTC) and Reset button to reset the sketch and any attached modules
- Flexibility and ease of connecting the extended memory and hardware connectivity board to external sockets for the 2-channel PCI Peripheral Connect Interconnect Express (PCIe) slot (which also functions as Wi-Fi adapter), stereo audio, Ethernet x2 and Micro-SD slot
- Extended interfacing capabilities using 157 GPIO pins
- Flexibility and ease of connecting the hardware connectivity board to external through three USB 3.0 and USB hub with two USB 3.0 ports, stereo audio, video and stream of high-definition HDMI output.

The comparisons between Intel Galileo and BeagleBone shows that BeagleBone is high performance single-board computer along with 2 GB DDR L3 and 4 GB eMMC, LCD, audio, video, multi-media, 3D graphics support.

8.3.6 mBed^{TM}

ARM mBed^{TM} board is open source. The board is extensible using modules, shields and other circuits. The board enables the use of an open version of modules, support of software library for peripheral components, sensors, radios, protocols and cloud service APIs under the Apache License 2.0.

An ARM-based platform is ARM® mBed™ IoT device platform since 2014. ARM's software framework provides the cloud development environment, mBed OS and mBed device server. A forum for development is mBed development forum, https://developer.mbed.org/forum/.

Following C++ APIs for mBed are also open source: (i) REST for administration, security, data flow, device, multitenancy, authentication, directory and subscription management APIs, (ii) Publish-Subscribe APIs (iii) L2M2M APIs, (v) device interfacing components, CoAP-SMS, CoAP-MQ, CoAP, HTTP, MQTT and (vi) device security components DTLS and TLS for end-to-end IP security across the communication channels.

IoT Applications

An mBed community ecosystem enables the development of the secure and efficient IoT applications. A platform includes an online web-browser-based programming and development environments for IoT devices.

Ecosystem means detailed framework with combination of device and infrastructure components for developing applications. A developer focuses on the applications using the framework. Community ecosystem means developer community exchanging ideas, codes, raising questions, providing answers and sharing new codes, hardware and applications.

ARM complete mBed ecosystem for Internet of Things provides a detailed set of tools to drive IoT.[11] It offers free OS for Cortex-M series processors, server-side software to connect and manage devices, and a website to serve as a focus for the developer community.

IoT Device Platform *i*s a self-contained framework. The development becomes easy due to combination of device and infrastructure components. A developer focuses on the applications using the framework.

mBed applications use IBM IoT Foundation for Internet cloud data storages and IoT applications. The applications are using "things" with lights, health monitoring or fitness devices, watches, sensors and actuators smartly through Internet. The developments tools are open sources and are at a connected computer with Windows, Arduino Linux distribution or MAC.

Features

Table 8.6 lists the features of mBed boards and starter kit.

IoT starter kit includes actuators, RGB LED, PWM connected speaker, PWM connected sensors, temperature sensor, 3 Axis -1 1.5g accelerometer, magnetometer, user interfaces-controls: 2 push-buttons, 2 x potentiometers, 5 way joystick and 128x32 graphics LCD.

Following are the features which foster wider use of mBed boards:

- Prototyping ease with popular ARM Cortex MCUs

[11] http://www.v3.co.uk/v3-uk/news/2373552/arm-unveils-complete-mbed-ecosystem-for-the-internet-of-things

Table 8.6 Features of mBed boards and starter kit

Board/ Shield	Application	SoC Processor/Clock	Operating/ Input V/ Perating Volate/ USB power pin output	SRAM/ Flash Memory	Analog In/Out/ Digital IOs/ PWM/ IOREF	USB host /UART Rx and Tx /I2C SDA-SCK /SPI /CAN	Ethernet port/W-Fi/ GSM/ ZigBe/RF/ Bluetooth
NXP LPC 1768 mBed	IoT and wearable ARM MCU based devices	Cortex-M3, 96MHz	3.3 V/ 4.9 V to 7 V/ 5 V	8 kB/ 32 kB	6/0/26/6/0	1 / 3 pairs/ 2 / 2 / 1	100 Mbps / 0/0/0/0/0
EA LPC 4088		Cortex-M4/120 MHz		96 kB/ 512 kB	6/0/26/6/0	1 / 3 pairs/ 2 / 2 / 1	100 Mbps / 0 /0/0/0/0
Wi-Fi Dip Cortex		Cortex-M3/72 MHz		12 kB/ 64 B	6/0/26/6/0	1 / 3 pairs/ 2 / 2 / 1	0/1/0/0/0/0
U Blox C027		Cortex-M3/96 MHz8		32 kB/ 512 kB	6/0/26/6/0	1/ 3 pairs/ 2 / 2 / 1	0/0/1/0/0/0
FRDM-K64F mbed IoT kit IBM IoT Foundation	Starter kit with USB/ Ethernet and ArduinoTM R3 compatible I/O s	MK64FN1M0VLL12 Cortex-M4 core/ up to 120 MHz	5V/ 7V–12V	256 kB/ 1 MB	6/0/14/6/1	micro-B USB host-cum-client/1/ 3 pairs/ 2 / 2 / 1	1/0/0/0/0/0
mBed RF module	RF for the kit						0/0/0/0/1/0
mBed BL LE module	Bluetooth for the kit						0/0/0/0/0/1
mBed shield	ZigBee Application						0/0/0/1
mBed shield	Wi-Fi Application						0/1/0/0 (RN-XV)

- Open extensible source code, schematics, software, middleware and online SD (IDE), with mbed C/C++ software platform tools for creating MCU firmware core libraries, the MCU peripheral drivers, networking, RTOS runtime environment, build tools and test and debug scripts
- mBed online IDE enables IoT/M2M applications development on the mBed platform
- Open source code editor with web-based C/C++ programming environment
- Open source web browser using the cloud ARMCC C/C++ compiler
- IDE latest version and appropriate OS are open source Eclipse with GCC ARM Embedded tools
- Built-in USB drag and drop FLASH programmer

mBed OS provides a C++ application framework with the ARM mBed Community libraries to create device applications, and component architecture and facilitation of automated power management, threads, Bootstrap, FOTA (firmware over the air) and connectivity protocol stack support for the Bluetooth®LE, Wi-Fi®, Zigbee IP, Zigbee NAN, 6LoWPAN, Cellular, IPv4, IPv6 and Ethernet.

ARM® mBed™ IoT device platform support IBM IoT Foundation software which makes the development of IoT applications and services an easy task.

8.3.7 Computing Systems (Mobiles and Tablets)

New generation mobiles other than regular phones also have the following features:

- Sensors and devices such as temperature sensors, accelerometer, gyroscope, magnetometer, camera, microphone, GPS
- Actuators, such as audio output speaker, vibrator elements, video, LEDs
- Connectivity, such as NFC, Bluetooth, USB, Wi-Fi, cellular
- Text communication such as SMS
- Multimedia communication such as MMS.

Tablets provide computing and network capabilities like a computer. Therefore, the mobiles and Tablets can also be deployed for IoTs. For example, participatory sensing.

Mobiles and tablets can however be used for wiring the new circuits for prototyping, as input is through USB connection, which can also be used for power to a prototype board. They do not have easy and direct connectivity for external sensors, actuators and health monitoring devices.

Android Development Kit

An ARM based Arduino board 'Due' is compatible with Android Development Kit (ADK). Android platform provides software for accessing the IO pins on the board and Internet communication.

Reconfirm Your Understanding

- Arduino boards are easy to work for DIY (do-it-yourself) and prototyping the embedded platforms for IoTs.
- Arduino IDEs are open source.
- Prototype development boards for IoT devices are the Arduino Ethernet, Arduino Wi-Fi and Arduino GSM shields. Development boards for wearable devices are Arduino Gemma, LilyPad, LilyPad Simple/ SimpleSnap and LilyPad USB.
- Arduino best applications can be the Internet cloud data storages and IoT applications, such as IoT of Streetlights which does not require intensive computing and graphics.
- Intel Galileo board includes an SoC, controlling and monitoring functions on SoC, store drivers and provisions for additional memory with an SD card.
- Galileo is Arduino board compatible.
- Edison has high performance compared to Galileo due to use of dual core double threaded 700 MHz processor, communication interface for Wi-Fi and Bluetooth LE and tools for OLTP, OLA, trigger alerts in advanced analytics of the data streams.
- Processor at the SoC has Intel Pentium architecture. The board provides large 8 MB SPI flash and supports a set of 30 sensors and accessories for Arduino.
- Galileos' best applications can be to make smart everyday *things* with health monitoring or fitness devices, watches, sensors, cameras, with facility of a personal computer board running Linux. The development tools are at the connected computer with Windows, Arduino Linux distribution, Linux, or MAC.
- Edison provides higher performance due to dual core, two-threaded 700 MHz CPU.
- Edison includes tools for collect, store and process data in the cloud, trigger alerts when using advanced analytics and OLTP and OLAP of the data streams.
- Raspberry Pi boards provide functions as single-board computer with media server applications and the intensive computations. The boards run at a speed of about 40 times that in Arduino and have larger RAM compared to Arduino.
- Raspberry Pi IoT applications are for media server required in systems such as networked security camera systems or ATMs.
- BeagleBone-X15 runs SoC which uses fast processor and DSP processor. BB-X15 has multimedia 4 GB eMMC for graphics and video and runs OSes, Linux, RISC OS, FreeBSD, OpenBSD and additional distributions of Linux such as Ubuntu.
- BB-X15 applications are as media, 2D and 3D graphics and video servers in IoT applications. BB best applications is its high performance compared to RPi.
- ARM-based mBed boards run faster than Arduino AT boards and are for applications using Internet cloud data storages and IoT applications of the sensors, actuators and health monitoring or fitness devices, watches, sensors and actuators smartly through the Internet with open sources. The board connects to a computer with Windows, Arduino Linux distribution or MAC or cloud during development phases.
- Mobiles and tablets have number of sensors, such as camera, temperature sensor, accelerometer and gyroscope which can be used in IoT applications and services.

Self-Assessment Exercise

1. List the features which are common in Arduino boards. ★
2. List the common and distinct features and applications of Due, Uno and LilyPad SimpleSnap. ★★
3. List the common and distinct features and applications of Arduino, Galileo and Edison boards. ★★★
4. What are the on-board functional units in Intel Galileo? ★
5. List the common and distinct features and applications of Galileo and Raspberry. ★
6. What are the features in Raspberry Pi 2 that enable it to function as a single-board computer and perform video processing? ★★
7. Draw the architecture of Raspberry Pi. ★★
8. What are the features which make BeagleBone-X15 highly suitable for IoT applications requiring fast media, 2D and 3D graphics and video streaming multimedia? ★★★
9. What are the features of mBed that distinguish it from Arduino? ★★
10. Write the IoT applications for Arduino R-3, Intel Galileo, Raspberry Pi and BeagleBone. ★★★

8.4 THINGS ALWAYS CONNECTED TO THE INTERNET/CLOUD

LO 8.3 Elaborate how the devices can remain always connected to Internet and cloud

8.4.1 Connecting Things to Arduino, Galileo, Edison, RPi, BB and mBed

Figure 8.4 shows things, devices and circuit boards connected to the Internet and cloud or web servers for applications and service.

Figure 8.4 shows that *things* first connect to an embedded computing device 1, 2, 3, … The data adapts for communication at this layer. The device may be an electronic circuit or prototype circuit at an Arduino, Galileo, Edison, RPi, BB, mBed or any other board.

Each connected device on LAN has a MAC address of 48 bit at the data-link layer. (Section 4.5). The MAC can be assigned in the program, in case not available, as in an Arduino board.

The device communicates using devices 6LowPAN, Bluetooth (low energy), ZigBee wireless or other wired technologies. When device communicates on the Internet, it uses 32-bit IP address or 128-bit IPv6 address. ARP translates an IP address to MAC and RARP translates a MAC address to IP address. Translation needs a subnet mask. Connected device may use a domain name, which translates to IP address in a DNS. A cloud or

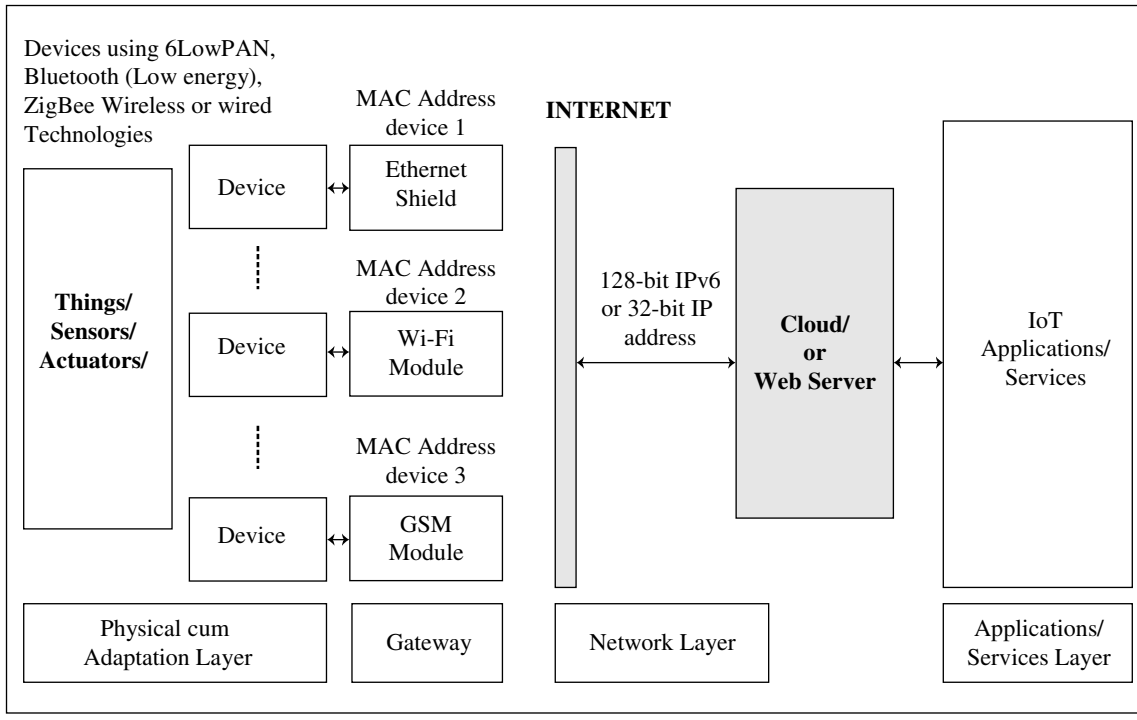


Figure 8.4 'Things', Sensors, actuators, devices and circuit boards connected to Internet and cloud or web-server for applications and services

server end at the web uses another IPv6 or IPv4 address. The gateway and addresses are specified in the program for communication to the Internet.

8.4.2 Internet Connectivity

Web connectivity of server or client ends use connected-device network protocol, gateway protocol or SOAP, REST, HTTP RESTful and WebSockets. Applications and services access the cloud or web server, using a protocol, such as HTTP, HTTPS, SOAP or web socket (Section 3.4)

Connecting Arduino USB to Internet

Arduino board IDE supports USB. USB port connects to a mobile or computer or tablet with an Internet connection using IDE USB port functions. A computer then connects to the Internet using network interface cards.

Connecting Arduino to Internet

Arduino IDE supports Ethernet protocol Library. Library means a set of codes which enable use of functions of the library for specific purposes. Ethernet LAN connects to the

network router directly or through Wi-Fi adapter socket of a wireless modem to connect to the network router. When the shield sends data of the device to the cloud, then the Ethernet client mode of shield is used. When the shield sends data of the device to the computer, then the Ethernet server mode is used. Example 8.5 gives the header files in the Ethernet Library preinstalled with the IDE.

Example 8.5

Problem

List header files in for Arduino Ethernet Library which are included as preprocessor statements in C codes and provide the functions for Ethernet LAN.

Solution

```
#include <SPI.h> /*Serial IO functions between Arduino SPI port and Ethernet shield*/
#include <util.h> /*IO utility functions*/
#include <EthernetUdp.h> > /* UDP protocol for sending datagram to the web server.*/
 #include <EthernetClient.h> /* Ethernet client end computing device connection. The shield is then used as client to the remote cloud or web server. */
#include <EthernetServer.h> /* Ethernet server end computing device connection. Ethernet shield is then used as server to the connected computer with the board. */
#include <Ethernet.h> /* Ethernet LAN functions */
#include <Dns.h> /* DNS protocol requests a IP address from the domain name.*/
#include <Dhcp.h> /* DHCP protocol requests a dynamic IP address from the IP router*/
```

Connecting Arduino to WiFi

Arduino IDE supports WiShield Library. WiShield connects to a network router without wires. However, sufficient energy for Wi-Fi communication is required from the power supply.

Example 8.6 gives the header files in the WiShield Library.[12] Use of WiShield needs that the program predefines (i) network type, infrastructure fixed network or ad-hoc network, (ii) security type, 1 (WEP), 2 (WPA) or 3 (WPA2), (iii) WEP key (wired equivalent security key or 128-bit when security type = 1 (means WEP) and (iv) SSID (Service Set identifier), a unique ID of 32 characters and is used for naming wireless networks. Service set means a set of networked interconnected devices which communicate using SSID of the set.

[12] http://github.com/asynlabs/WiSieldnpreinstalled

Example 8.6

Problem

List header files in for Arduino WiShield Library which are included as preprocessor statements in C codes and provide the functions for Wi-Fi.

Solution

```
#include <util.h> /*IO utility functions*/
#include <SPI.h> >
#include <WiServer.h> /* WiFi server end computing device connection. WiFi shield
is then used as server to the connected WiFi of another Service Set.*/
```

Reconfirm Your Understanding

- Things, devices and circuit boards connect to the Internet and cloud or web servers for applications and services. A MAC address assigns to an Ethernet shield or computer. A subnet mask enables resolution between an IP address and MAC address. DNS translates as domain name to the IP address. An IPv6 or IPv4 address is used for communication from the web socket, server or client using TCP or UDP on the Internet.
- Web connectivity of client ends when the server or cloud uses the connected-device network protocol, gateway protocol, SOAP, REST, HTTP RESTful and web sockets.
- Arduino Ethernet shield connects and provides Internet connectivity through wired adapter of modem or Wi-Fi modem on the LAN. When sending data of the device to the cloud, then Ethernet client mode of Shield is used. When sending data of the device to the computer then Ethernet server mode is used.
- Arduino Ethernet and WiShield libraries have preinstalled functions with IDE for Internet accesses to a remote IP address, means cloud or web server.

Self-Assessment Exercise

1. Show using a diagram, the simplest way for connecting Arduino USB to the Internet. ★
2. How does Arduino Ethernet shield connect to the Internet? ★
3. List the header files required from Arduino Ethernet Library. ★★
4. When are the Ethernet client and server used? ★★★
5. List the WiShield Library header files and parameters requiring preprocessor definitions. ★★

Key Concepts

- Analog input
- Analog output
- Arduino
- Arduino Linux distribution
- ARM microcontroller
- ARM® mBed IoT device platform
- AVR microcontroller
- BeagleBone-X15
- Bootloader
- Cortex
- Digital IO
- Embedded device
- Ethernet library
- Flash
- I2C
- IBM IoT Foundation
- IDE
- In-circuit-programming
- Intel® Galileo/Edison
- LilyPad
- Microcontroller
- microSD
- Operating system
- PWM
- Raspberry Pi 2/Pi 3
- SD card
- Serial port
- SoC
- SPI
- Uno
- Video processor
- WiShield library

Learning Outcomes

LO 8.1

- Things, sensors and actuators need devices for computations and communications.
- A device consists of MCU, ASIC or a SoC. The device software embeds in the memory for enabling the device functions. MCU has processor, internal RAM, ROM or flash, timers, serial interfaces, IO ports and number of application-specific functional units.
- SoC consists of multiple processors, software and all the needed digital as well as analog functional units.
- An MCU AVR® 8 is for use in embedded device platforms.
- Each platform needs an IDE and development tools for the prototyping and designing.
- Selection amongst number of different available software depends on the hardware platform.
- Open source software are also available for the embedded device platforms.

LO 8.2

- IoT, M2M and wearable devices can be prototyped and developed using Arduino boards. An Arduino board is easy to work for DIY (do-it-yourself) and prototyping and have open source IDEs.
- Arduino best applications can be the Internet cloud data storages and IoT Applications, such as IoT of streetlights which don't require intensive computing and graphics.
- Intel Galileo boards have SoC with the SD card for controlling and monitoring the functions of SoC, store drivers, and for additional memory and Arduino boards compatible.
- Intel® Edison is a high performance computation and communication module which the SoC, 1 GB RAM and Wi-Fi and Bluetooth LE communication interfaces.
- Raspberry Pi boards provide functions as single-board computer with media server applications and the intensive computations.
- Raspberry Pi's best IoT applications for media server requiring applications, such as networked security camera systems or ATMs.
- BeagleBone-X15 runs SoC which uses the fast processor and DSP processor.
- BB-X15 applications are as media, 2D and 3D graphics, and video servers in IoT applications.

- ARM based mBed boards run faster than Arduino AT boards and are used for applications using Internet cloud data storages and IoT applications through open sources. The board connects to a computer with Windows, Arduino Linux distribution or MAC or cloud.
- Mobiles and tablets have number of sensors, such as camera, temperature sensor, accelerometer, gyroscope which can be used in IoT applications and services.

LO 8.3

- *Things*, devices and circuit boards connect to the Internet and cloud or web servers for applications and service. A MAC address assigns to an Ethernet shield or computer.
- A subnet mask enables resolution between an IP and MAC address. DNS translated domain name to the IP address. An IPv6 or IPv4 address is used for communication from WebSocket, server or client using TCP or UDP on the Internet.
- Arduino USB, Arduino Ethernet shield and Arduino Wi-Fi module use the USB, Ethernet and WiShield Library functions which are preinstalled functions within the IDE.
- Library functions enable coding for Internet accesses of devices using USB, Ethernet or Wi-Fi to the remote IP address cloud or web server.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. Things, sensors and actuators need a device for Internet connectivity that consists of (i) software, (ii) microprocessor with cache, SRAM and flash, or MCU, (iii) MCU with external memory, (iv) SoC with external SRAM, (v) SoC with external SD or microSD card, (vi) functional units including analog inputs, PWM for analog outputs and digital IOs, and (vii) RGB LEDs. ★
   (a) (i), either (ii) or (v) and (vi)
   (b) All except (vii)
   (c) (i), (v) and (vi)
   (d) All except (v)
2. A connected device platform needs (i) either a MCU, SoC with external SD card or ASIC, (ii) software program embedded into the device, (iii) IDE embedded into the device, (iv) required IDE functions embedded into the device, (v) embedding of codes after code-development, simulating and debugging, (vi) embedding through a device programmer or code burner, (vii) Bootloader and OS both, and (viii) Bootloader compulsorily and OS optionally for running multiple tasks and additional system functions. ★★
   (a) All
   (b) All except (iii) and (vii)
   (c) All except (iii) and (viii)
   (d) All except (iv) and (vi)
3. Bootloader (i) is system software which loads on to a chip or computing platform to let system start the functions, (ii) loads a system software or operating system, ★★

(iii) enables the normal, operational runtime environment on starting of a system, (iv) preferably pre-installed in an MCU or SoC, (v) is must in a system and (vi) may also facilitate the use of system hardware and networking capabilities.
(a) (i)
(b) (i) to (iv)
(c) All
(d) All except (ii)

4. Performance for computations (i) in MIPS depends on the clock rate, (ii) in MIPS primarily depends on the clock rate, (iii) in MFLOPS depends on the clock rate as well as on presence of a unit for the floating-point operations, (iv) in MIPS depends on RAM capacity, whether 4 kB, 16kB, 32B, 512 kB or 1 GB higher and (v) in MFLOPS on the flash memory capacity. ★
   (a) All except (ii) and (v)
   (b) All except (iv) and (v)
   (c) (i), (ii) and (iii)
   (d) (ii) and (iii)
5. Selection among the number of different available platforms depends on number of factors—(i) price, (ii) open source availability, (iii) ease of application development, (iv) needed capabilities, performance required from the IoT device and (v) suitability for developing and using for prototyping and designing. ★
   (a) All
   (b) All except (ii) and (v))
   (c) (i) to (iv)
   (d) (i) to (iii)
6. Arduino features are (i) prototyping ease, flexibility and ease of assembling modules on the board, (ii) ARM based platform, (iii) open extensible source code, schematics, software, middleware and IDE, (iv) libraries can be added with additional programs, (v) IDE and software runs on multiple environments, Linux, Windows and Mac OS X, (vi) Open source and extensible hardware using the modules, shields and other circuits with open version of modules and codes from other designers and (vii) suitability for IoT applications which don't require intensive computations and media processing. ★
   (a) All except (ii)
   (b) All except (vi) and (vii)
   (c) (i) to (iv)
   (d) (i), (iii) and (iv)
7. Intel Galileo is best for IoT applications for the *things* (i) such as health monitoring or fitness devices, cameras, media server and streaming video, (ii) connect for computations and Internet connectivity using the facility of single-board personal computer board with Pentium architecture and single core processor, (iii) connected circuit require computations running Linux, and (iv) hardware and software pin-compatible with shields designed for the Arduino Uno R3 and development phase use the Arduino IDEs. ★★
   (a) All except (i) and (iv)
   (b) All except (i)
   (c) All except (iii)
   (d) All except (iv)

8. Raspberry Pi 2 board applications are for IoT applications requiring (i) hardware and software for high performance computing and graphics, (ii) media server IoT devices, (iii) networked security camera systems, and (iv) ATM applications and services. Raspberry Pi 2 B+ board consists of (v) The ARM Cortex quad core SoC processor plus a graphic processor for graphics and video, (vi) Memory on-board 512 kB, plus in the multimedia card module memory support and (vii) no SD card or MicroSDHC card. ★★★
   (a) All are,
   (b) All except (ix)
   (c) (i) to (v)
   (d) All except (v)
9. BB-X15 best applications are for IoT applications requiring (i) fast media, (ii) 2D and 3D graphics, (iii) video streaming multimedia, (iv) PWM, CAN/SPI//DVI-D, USB 2.0 port and integrated RTC, (v) multimedia 1 GB eMMC, 1 GB on-board memory plus micro SD support, (vi) 2-channel PCIe, (vii) stereo audio, and (viii) Ethernet x4. ★★
   (a) (i), and (iv) to (vi)
   (b) (i) to (vii)
   (c) All
   (d) All except (iv), (v) and (viii)
10. Features of an a mBed device platform are (i) mBed online IDE enables IoT/M2M applications development, (ii) open source code editor with web-based C/C++ programming environment, (iii) open source Eclipse with GCC ARM Embedded tools, (iv) PWM, CAN/SPI//DVI-D, USB 2.0 port and integrated RTC, (v) Open extensible source code, schematics, software, middleware and online IDE, and (vi) codes for creating MCU firmware core libraries, the MCU peripheral drivers, networking, RTOS runtime environment, build tools and test and debug scripts, on the mBed platform. ★★★
    (a) (i), and (iv) to (vi)
    (b) (i) to (vi) except (ii)
    (c) All
    (d) All except (iv)

Short-Answer Questions

1. Why is an IDE required for prototyping the embedded device platform? **[LO 8.1]** ★
2. What are the software components required for connecting sensors and actuators to the Internet? **[LO 8.1]** ★★
3. How does a microprocessor compare with a microcontroller? **[LO 8.1]** ★
4. Outline the benefits of using a SoC with an SD card for embedded device prototyping. **[LO 8.1]** ★
5. Write the similarities and dissimilarities in usages of Arduino boards with Intel Galileo boards. **[LO 8.2]** ★★★
6. Draw the architecture of Intel Galileo board and list the usages of each component on the Intel Galileo board. **[LO 8.2]** ★★
7. Draw the architecture of Raspberry Pi 2 model B + board using the features? **[LO 8.2]** ★
8. What are the possible IoT applications of BeagleBone-X15 board? **[LO 8.2]** ★★

9. What are the demerits of usages of Arduino for IoT embedded image devices development compared to Edison, RPi or BB? **[LO 8.2]** ★★★
10. What are the additional benefits of using Raspberry Pi compared to Arduino-R3 for IoT embedded device development? **[LO 8.2]** ★★
11. What are the benefits of using the Eclipses tools, services and frameworks that enable an Open Internet of Things? **[LO 8.2]** ★★★
12. How does the library help in configuring Arduino for the Wi-Fi connectivity to Internet? **[LO 8.3]** ★★★

Review Questions

1. Describe the AVR®8 features which enable a generation of sensors data for the Internet. **[LO 8.1]**
2. How is an embedded-device platform hardware and software selected for M2M applications? **[LO 8.1]** ★★
3. How is an embedded-device hardware and software selected for IoT applications? **[LO 8.1]** ★★
4. What are merits in Arduino boards for the IoT, M2M and IIoT applications and services? **[LO 8.2]** ★
5. What are features which makes Intel Edison boards suitable for IoT, M2M and IIoT compared to Arduino? **[LO 8.2]** ★★★
6. Compare the features of Intel Galileo, Raspberry and mBed boards for the IoT, M2M and IIoT. **[LO 8.2]** ★★
7. Describe usages of Raspberry and BeagleBone boards for IoT applications. **[LO 8.2]** ★
8. How do the sensors and actuators in the things connect securely to the Internet? **[LO 8.3]** ★★★

Practice Exercises

1. Draw architecture of AVR®8 that reflects the microcontroller features. **[LO 8.1]** ★
2. List the required IDE features which help in selecting right embedded hardware and software. **[LO 8.1]** ★★
3. List the differences in Arduino boards for IoTs and wearable devices. **[LO 8.2]** ★★
4. Write the differences between Arduino, Intel Galileo, Edison and mBed boards for the IoT devices. **[LO 8.2]** ★★★
5. Show the architecture for BeagleBone diagrammatically. **[LO 8.2]** ★
6. Compare Raspberry Pi 3 and BeagleBone-X15 features in a table. **[LO 8.2]** ★★
7. Show the architectures of sensors and actuators with Arduino boards connected to Gateway using Wi-Fi for Internet of Streetlights applications. **[LO 8.3]** ★★★
8. How will you modify the architectures shown in exercise 7 for Internet of Traffic Control Lights. Lights are controlled as well as delay lights are synchronised according to traffic density at incoming and outgoing roads. Apply sensors and actuators with Arduino boards connected to gateway using Wi-Fi for Internet of Streetlights applications. **[LO 8.3]** ★★★

9

Prototyping and Designing the Software for IoT Applications

Learning Objectives

LO 9.1 Develop the codes, design and test the embedded devices for IoT and M2M using IDEs and development platforms

LO 9.2 Analyse and program the devices, gateways, Internet connectivity, web and cloud applications using the open-source implementations of Eclipse IoT stack

LO 9.3 Describe prototyping methods of the online components for IoT application APIs

Recall From Previous Chapters

Section 1.4 listed the following five entities for the technology behind IoT systems:

1. Device platform consists of device hardware and software using a microcontroller (or SoC or custom chip), and software for the device APIs and web applications
2. Connecting and networking (Connectivity protocols and circuits) enabling the internetwork of devices and physical objects, called 'things'; enabling Internet connectivity to remote servers
3. Server and web programming enabling web applications and web services
4. Cloud platform enabling storage, computing, prototyping and product development platforms
5. Online transactions processing, online analytics processing, data analytics, predictive analytics and knowledge discovery enabling large number of applications of an IoT system.

The previous chapter defined embedding as a process of embedding software into a computing hardware to enable system functioning for the specific dedicated application(s). A device-platform embeds software into computing and communication hardware, and it functions for specific application(s). The chapter covered the hardware and prototype development using a device platform, such as, Arduino, Galileo, Edison, Raspberry Pi, BeagleBone, mBed. The chapter described basics of designing the embedded devices. The descriptions included the following:

- Uses of embedded software functions, device APIs and middleware; the middleware enables a device to perform the functions of the computing and communication.
- Connectivity interfaces consist of processing units, device interfaces and APIs for communication. They perform the computing and communication functions.
- Integrated Development Environment (IDE) enables the coding on a computer connected with the embedding platform, downloading of the codes from the computer and development of error free codes using a number of edit test-run, and debug cycles (Section 8.2).

9.1 INTRODUCTION

Figures 1.3 to 1.5 showed that in order to develop the IoT software five levels are needed: (i) Gather + Consolidate, (ii) Connect, (iii) Collect + Assemble, (iv) Manage and Analyse and (v) Applications and Services. This chapter will explain the five levels and required software in detail.

The present chapter focuses on methods of developing software, which are used at levels of IoT devices, gateways, Internet connectivity and web and cloud applications.

Recall the meaning of the key terms, such as app, application, framework, XML, API, script, java script, object, client, server, URL, HTTP, HTML5, web service, WebSocket and bootloader, Application Programming Interface (API) Operating System (OS), Real-Time OS (RTOS), process, task, thread, IDE and library from Chapters 3 and 8. Some of these key terms are re-explained along with new terms. The understanding of these terms will enable the reader ease in learning the development processes for software including the APIs:

Component is an abstraction for a core set of frameworks, services or software functions that can be reused after reconfiguring them for providing solutions.

Software framework is an abstraction which provides reusable software environment and software with generic functionalities. A framework helps a user to selectively change and add new functionalities when reusing the functions that create new applications, services, products and solutions. For example, assume a framework which provides generic functionalities for graphics and drawing the objects. A drawing application developer selects and reuses the framework functions, selecting the drawing and visualising functions, like ellipse or circle and adding the required ellipse or circle parameters such as size, position, colour, shading and fill patterns. An application runs and draws the object

as per the parameters and uses the generic functions provided in the drawing framework. The analogy can be taken from a candle building framework which is used in the candle making industry. Putting the threads in appropriate holes and filling of molten wax makes a bundle of candles.

Application framework is a framework that supports a fundamental structure and provides a skeleton which leads to development of applications, and provides reusable development environment with generic functionalities. The framework helps a user to selectively change, add new functionalities and build the application.

Web-application framework is a framework which supports the fundamental structure and skeleton, which leads to the development of web-applications including, web APIs and resources. The framework provides reusable web-application development-environment with generic functionalities. The framework helps user of framework selectively change and add new functionalities and build web application or web service and APIs.

Community is a software development initiative or model, which uses collective efforts for software development. A community forms on *initiative* of universities, organisations and institutions for an open source project. The community is an approved open-source independent foundation for distribution, like OSGi. A foundation or community may also hold copyright and commit to follow distribution practices, contributor agreements and licensing, for example, Apache foundation, for the Apache server.

OSGi (Open Service Gateway Initiative) is a development framework for Java that is used for deploying modular software programs, libraries and bundles.

OSGi bundle refers to a tightly coupled collection of classes, jars, and configuration files. A bundle is dynamically loadable. The external dependencies are explicitly declared, when existing.

Software stack is a set of frameworks and services that is minimum need for an intended complete solution. Examples are Eclipse IoT Stack for IoT and Berkeley Data Analytics Stack (BDAS) for analytics solutions. A stack may be community supported. For example, Open Services Gateway initiative (OSGi) supports Eclipse IoT Stack. Stack provides a set of software functions, frameworks, packages, modules or subsystems. The stack creates a complete platform to support the applications or services when installed and configured on individual systems or added to the templates for automatic installation.

Sandbox server[1] is a server with data from the source code distributions and other collections of contents, data or sets of the codes, proprietary or public, and is protected from changes intentional or unintentional. The server functions as a working directory, test server or development server. A sandbox is also used for testing a developed application. Sandbox client uses the server's minimal functionalities, same environment variables or access to an identical database of the server for the development of specific functionalities and application(s).

[1] https://en.wikipedia.org/wiki/Sandbox_(software_development)

Linux distribution means a package or set of software functions and modules of Linux, bundled together for specific functions or for specific hardware, and distributed for wider usages and applications.

Bootloader is system software which loads or embeds into a microcontroller chip or computing platform to let the system start its functions.

OS is system software. It manages the processes, memory, IOs, network subsystems and devices. It enables prioritisation, multitasking, concurrent running of multiple threads and many system functions using given computing device hardware. An OS may be open source, for example, Linux or its distribution. A Linux-based OS distribution is Debian Squeeze Arch Linux Fedora for Raspberry Pi or BeagleBone using an ARM processor.

Multitasking or *multithreading* means execution of multiple tasks or threads, according to some plan, such as active or unblocked tasks run according to a sequence or one after one in allotted time slices or according to the priorities assigned for each or run first called task first. The OS supervises the running of the tasks and threads.

Real-time Operating System (RTOS) is also a system software and is an OS that enables running of real-time processes on computing and communication hardware.

Script is a piece of code in text form, which runs using an *Interpreter*, which interprets them at the runtime. Examples of the scripting languages are JavaScript, JSON, Perl and PHP.

C is a high-level programming language, used for coding the embedded platforms. The codes compile and generate the executable codes for that platform or computer. C is a procedure-oriented language.

Function (method) is a named entity with arguments in between the brackets () and has the statements in C or other programming language. The arguments are specified within the brackets or quotes. The arguments associate with the function, pass the initial values and return the results. For example, assume a function `chocolates_Sold()` for calculating the chocolates sold during the specific interval using a sales table. Then three arguments are start and end dates, annual sales table that will be passed in the function for the execution of the statements and fourth argument returns the value = total number of chocolates sold during the interval after the calculations.

Data values are passed from a (calling) function to a called function and the results after executing the statements can be sent (returned) for use by a function needing those results. Each language has some element similar to C function. For example, a function is called a method in Java for the similar purpose.

Exception is a signal or object which represents occurrence of an event, exceptional condition or conditions on which another set of codes need to run. This set of codes may be at a `callback()` or `catch()` function. Exception is thrown (sent to the system) on the occurrence. It is thrown when an exceptional condition arises during the execution of a function or a set of statements.

Catch is a method (function) which runs only once on the occurrence of an event. A thrown exception is said to be caught by the `catch()`, also called `callback()` method. The event object resets or removes (points to null) at start of the `catch()`.

C++ is a high-level object-oriented programming language, which includes classes, objects as well as C codes. C++ codes also compile and generate executable codes. A modified version of standard C/C++ used at the device platforms includes additional software libraries, and that may subtract certain functions. Subtracting may be due to need of limiting the usages of system memory in small memory platforms or due to no necessity of them for specific problems or hardware. For example, Arduino platform C++ compiler uses avr-gcc, which does not provide for handling of exceptional conditions and try-catch blocks which are used in standard C++. Examples of additional libraries are specific programs for distinct actions, which provide for programs, such as sending data to a USB port for onward transmission onto the Internet, and for the usages of protocols, ports, serial ports, USB ports, GPIO pins, timers, Ethernet and hardware units at a device platform. The *avr* stands for AVR® microcontroller at the Arduino boards. The *gcc* stands for GNU C/C++/Objective C compiler. The avr.gcc enables usage of codes, which run on the AVR.

Compiler is the software for debugging and listing errors, if any, and then converting the codes into running executable codes on a specific computing platform and environment. Languages, such as C, C++ need compilers to enable running the codes on a computing platform and run-time environment.

Compiled virtual-machine language refers to a language such as Java or C# that compiles codes, which are platform and runtime environment independent. The codes need a Virtual Machine (VM) or a framework that enables running on a computing platform and run-time environment.

Application Programming Interface (API) is a software which enables access to an app, application or service. An API runs on a local or remote computing platform, which sends messages from the client (user) end for the server (application) end, and receives server messages for the client.

Message is a general term, which refers to communicating data, request, query or response during communication from one end, say a client, server, script, function, method or API to the other end.

Graphic User Interfaces (GUIs) refers to computer screen display tools, such as status bar, task bar, buttons, check-boxes, menus and dialog boxes which enable easy interactions of APIs between the user and app, application or service software.

Graphical software means software with the GUIs.

Cross-platform software refers to software which can be developed on one system, such as a computer for another computing platform, such as Arduino.

Lightweight software function denotes a function which requires limited additional resources, and is not dependent on the OS functions or functional resources of another source.

Client is a piece of code which makes a request. Example is a CoAP client sending request to a CoAP server or an HTTP client sending request to an HTTP server.

Server is one which generates the response to client request or which publishes messages or responses for the clients, which send the requests or subscriptions.

Broker is an intermediate server which receives the published data, message, a client request or subscription from one end and which also functions as a server to send the messages/responses for another client at the other end. For example, MQTT broker receives data/messages published from the device platform using node.js MQTT client running an application and sends response to the other end Java MQTT client at the application running at a computing system or phone using subscription.

Java is a widely used object-oriented, high-level programming language with large open source libraries, communities and distributions. Java compiles into byte codes. The codes can run anywhere on a computing platform, provided the platform includes a JVM (Java Virtual Machine). A JVM is a software to translate the byte codes into native machine codes. Java is thus a compiled VM language. Machine means a computing platform consisting of hardware, OS and Software.

C# is an object-oriented high-level programming language and a compiled VM language that compiles CIL (common runtime language) codes, which are called the managed codes. The managed codes are CPU neutral and platform independent, and therefore can run anywhere on a computing platform provided the platform has the runtime environment and framework, such as dot Net.

Perl, a scripting language, the usages of which are for *handling and parsing the XML, and parsing* of the strings of character data and messages of the client for the server and the messages of server for the client (Section 3.3). Perl also has vast libraries. Perl enables easy coding of the scripts, even for big websites. A set of script codes execution is slower than C++ and need greater runtime memory for parsing.

Python, a high-level language in which codes are interpreted at runtime, expresses code in fewer lines compared to Java or C++, and supports big libraries and open source codes on a computing platform or computer.[2] A version is Cpython, which is a community, Python software foundation manages.

Raspberry Pi, BeagleBone or other platforms also run Python. There are many advantages of using Python. Examples are easy sending requests and receiving responses from the Internet services, regular expression coding for short expressions for easy handling and parsing of the strings of character data, such as weather data (Example 1.1), message strings to or from a server or cloud, self-memory management and easy connecting and querying of databases. The codes execute little slower than C++ and need greater memory.

[2] https://en.wikipedia.org/wiki/Python_(programming_language)

Ruby is a programming language which is used for development of a web application and service.

Event model enables the functions, methods, scripts or APIs run on eventing. The scripts can run in multitasking mode on multiple events. Assume, if four events occur then for each event a script or function runs and four tasks, each executing multiple scripts which run on eventing.

*callback() i*s a method (function) which runs on the event. For example, on receiving a message or the data or a request or a change in a parameter value behind a set-limit or on occurrence of certain action or exception. A `callback()` runs on the event and therefore, an `eventListner()` function need to run (explicitly or implicitly in background) for listening the occurrence of that event and finding the need of running `callback()`.

Node.js is a framework for creating the JavaScript codes for running on a cross platform environment, including browser, such as Chrome. The framework enables faster creation of the scalable network applications for distributed devices using a browser. The code development is fast when compared to creating that from JavaScript directly. Scalable means can be made big or small.

Node.js programming is based on the event model for running of scripts in an application. The scripts at the `callback()` functions run at the multiple distributed devices on the events. Node.js framework thus enables development of real time web applications and games using the browsers where input-output needs to be processed fast.

Processing language is a programming language for a development environment. Processing is open source. A computer programmer can download the environment from the website for the processing language.[3] The language also enables creation of an elegant GUI-based programming interface.

Wiring,[4] an open source module consists of a set of hardware and software which is similar to Arduino and has programming environment for an Arduino like device platform.

Sketch in Arduino means an open source unit of code (a program unit), which uploads on the device platform/board and runs. An example is *blink* sketch, which is the code for blinking at the board when run. Another example of *sketch* is for analog reading and serial printing (or displaying on serially connected computer) at successive intervals or RFID tag ID reading or I2C device data reading.

Firmware refers to software permanently residing at a circuit Read Only Memory (ROM) or flash. Firmware provides specific functionalities of that circuit, such as bootloader, a start-up program on power on and ROM BIOS (basic input-output system) functions.

Lua extension language/scripting language, is a dynamic type language for a number of host platforms. Lua has features of co-routines (multiple tasks running at distributed applications, dynamic loading of the modules, usages of first class functions (supporting

[3] http://www.processing.org
[4] http://www.wiring.org.co

passing functions as arguments to other functions, returning them as the values from other functions or *assigning* the values to the variables or values in data structure).[5]

Section 9.2 describes the ways of developing software for the embedded device platforms using the IDE. Section 9.3 describes the ways of developing gateway software for communication between the devices and applications and services at a cloud or server using Eclipse IoT stack. Section 9.4 describes ways of developing the online component APIs, web APIs and WebSocket APIs for message exchanges between embedded devices and applications and services at the server or cloud.

9.2 PROTOTYPING EMBEDDED DEVICE SOFTWARE

LO 9.1 Develop the codes, design and test the embedded devices for IoT and M2M using the IDEs and development platforms

The previous chapter described prototyping and development boards—Arduino, Intel Galileo, Edison, Raspberry Pi, BeagleBone and mBed. Prototype development of the programs requires bootloader, OS and IDE. Software embeds into a device platform.

First level in IoT architectural concept is gathering (data from devices/sensors) + consolidating (enriching). Second level is connection to the Internet. An IDE enables development of software for functions at first and second levels. IDE may also enable usages of the OS or RTOS functions at an embedded device.

IDE enables development of software at first level and second level for embedding into device platform

Bootloader firmware stores at flash/ROM of a microcontroller in a device and enables communication with a computer having an IDE. The IDE, in general, consists of the APIs, libraries, compilers, RTOS, simulator, editor, assembler, debugger, emulators, logic analyser, code burner, and other software for integrated development of the system. An IDE may be an open source. For example, Arduino has an open source IDE which is downloadable from the Arduino website.

IDE enables the development of codes on a computer, and later on downloading (pushing) of codes on to embedded device, such as Arduino or microcontroller board. The code-burner places codes into flash memory or EEROM or EPROM. The specific application codes are thus embedded into the device.

9.2.1 Programming Embedded Device Arduino Platform using IDE

Arduino board can be programmed using avr-gcc tools.[6] The Arduino board has a pre-installed bootloader embedded into the firmware.

Arduino IDE open source from http://www.arduino.cc provides multitasking simplicity and enables the development of the codes, their simulation and upload (embedding) on to the device

[5] https://en.wikipedia.org/wiki/First-class_function
[6] http://www.nongnu.org/avr-libc

Arduino programmer develops the codes using a graphical cross-platform IDE. Arduino provides simplicity. IDE of Arduino board also has simplicity, is based on processing language and makes the programming easy. The board connects to a computer which runs the IDE. The bootloader program hand overs the control and enables running of the loader, which loads the required OS functions and software into the system hardware and networking capabilities into the board.

The Arduino bootloader provisions for multitasking by the usage of interrupt (analogous to eventing) handing functions for each task. Multitasking is done by assigning multiple values of a number *n* for the tasks ($n > 0$). When an instruction for interrupt; for example, INT *n* executes, then interrupt-handing function *n* is called for execution. Each task or thread can have the number *n* associated with it. Interrupt-handing function, similar to a `callback(n)` executes on event *n* or similar to catch function on exception *n*.

The IDE consists of a set of software modules, which provide the software and hardware environment for developing and prototyping the software for a specific device platform. First, a computer downloads an appropriate IDE version, as per the computer OS. A computer usually runs Windows or Mac OS X or Linux.

The bootloader enables the computer to push the developed codes into a board using the Arduino IDE through a USB cable or a labelled serial port. Arduino bootloader *need not initiate the upload of OS*, as done in a computer where bootloader loads the OS from the secondary disk.

The Arduino IDE is available from the website of Arduino.[7] A programmer downloads the required IDE version. IDE runs on the computer and enables the development of the codes, their simulation and upload (embedding) on to the device platform.

The Arduino IDE includes a C/C++ library. The library is called *Wiring* for a project of the same name with open source module at a website.[4] The *Wiring* library functions make coding easy for the Arduino IO operations.

Development of the Codes

Arduino IDE functions as a file editor for the codes using the *Processing* environment and library functions. The editor provides automatic indentation, highlights the syntax of the codes and matches braces. The edited file compiles, checks and lists errors, if no errors, enables pushing of codes for embedding onto the board through serial or USB port.

Simplicity of Arduino is clear from the fact that only two functions are necessary to define executable program functions for the board, namely, `setup()` and `loop()`. The function `setup()` runs at the start and is used for initialising settings, and function `loop()` has a program in endless loop using statement 'while (true) {*statements* ;}' which runs till power off.

A *serial monitor* at the IDE enables messages from the embedded software for the microcontroller into the computer screen where the IDE is setup. The messages are required during testing and debugging the downloaded software during test stages.

[7] http://www.arduino.cc

Using GPIO Pins

Assume that a programmer is well versed in C/C++ programming. Example 9.1 shows how to program GPIO pins and serial monitor at the IDE (for computer screen display) using Arduino IDE C/C++. The example is for switching on only of north and south pathways TLs. Next example will describe full control of traffic lights for all four pathways sequentially and also use of an internal LED for testing the successful running of the program.

Example 9.1

Problem

Programming for Arduino controlled traffic-control lights (TLs) at a road junction:

Assume Arduino Uno board as an embedded device platform (Section 8.4.1) for the following project: three TLs—Red, Yellow and Green needs to be controlled on each of the four north, east, south and west clockwise pathways.

Let twelve GPIO pins on Uno connect twelve number externally connected LEDs, R_0, Y_0, G_0, R_1, Y_1, G_1, R_2, Y_2, G_2, R_3, Y_3, and G_3, (four sets of three R, G, Y LEDs each). The port LEDs represents the TLs during the prototype-development and testing-stage.

How can the port LEDs be On-Off programmed so that north and south pathways directed roads and traffic is switched on and east and west pathways traffic switched off?

Solution

First step is declaring the data types, constants, variables and functions used. Second and third steps are coding for `setup()` and `loop()`. Example of program codes, which embeds onto Uno are as follows:

```
/*Assume twelve digital IO ports assigned to external LEDs are port 2 to 12 and
port 14. The ports are programmed according to TLs switching ON or OFF./
/* Pin 13 connects the board LED, and be used for indicating successful running
of the developed codes during testing phases.*/
int internalLED = 13;
/* Variables are written using a lower case first character */
int ledR0, ledY0, ledG0, ledR1, ledY1, ledG1, ledR2, ledY2, ledG2, ledR3,
ledY3, ledG3;
Assign the pins to the respectively connected LEDs */
ledR0 = 2; ledY0 = 3; ledG4 = 4; ledR1 = 5; ledY1 = 6;ledG1 = 7; ledR2 = 8;
ledY2 = 9; ledG2 = 10; ledR3 = 11; ledY3 = 12; ledG3 = 14;
/* Declare Functions for sequences of traffic lights ON-OFF as follows: */
     void north_south_Green() {
digitalWrite (ledR0, LOW); digitalWrite (ledY0, LOW); digitalWrite (ledG0,
HIGH);
digitalWrite (ledR2, LOW); digitalWrite (ledY2, LOW); digitalWrite (ledG2,
HIGH);
     };
/* Function Switch RED ON for East and West pathways*/
    void east_west_Red() {
```

```
digitalWrite (ledR1, HIGH); digitalWrite (ledY1, LOW);digitalWrite (ledG1, LOW);
digitalWrite (ledR3, HIGH); digitalWrite (ledY3, LOW); digitalWrite (ledG3, LOW);
      };
/*********************************************************/
void setup ( ) {
/* GPIO pins 2 to 12 and 14 are thus assigned port numbers corresponding to 12 external LEDs, R0, Y0, G0, R1, Y1, G1, R2, Y2, G2, R3, Y3, and G3.*/
/* Assign mode of each pin as output */
pinmode (ledR0, OUTPUT); // Constants are written in Upper Cases
pinmode (ledY0, OUTPUT);
.
pinmode (ledY3, OUTPUT);
pinmode (ledG3, OUTPUT);
/* Let Pin 13 be used for indicating successful running of the developed codes during testing phases. Initialise internal Port 13 Digital IO Pin LED for test.*/
pinmode (internalLED, OUTPUT);
/* Initialise start of the Board and Sequences. */
digitalWrite (internalLED, HIGH);
/* Display the settings of Digital IO pins at serial display-monitor on the computer where IDE is setup. */
/*Let UART mode baud rate = 9600*/
Serial.begin (9600);
Serial.println ("Arduino project.Program for controlling three traffic signals-Red, Yellow and Green at four pathways");
Serial.println ("Arudino board LED glows when cycle starts for the sequences of lights turning high and turns off for brief interval in order to indicate the successful completion of the cycle");
Serial.println ("Twelve 12 external LEDs, R0, Y0, G0, R1, Y1, G1, R2, Y2, G2, R3, Y3, and G3 corresponds to 12 traffic lights at north, east, south, west four pathways")
}
/*********************************************************************/
Step: Loop function which endlessly runs
void loop () {
/* Assume no right turn from pathways or left turn from pathways permitted, just for simplicity and for learning the basics. */
/*Switch Green ON for North and South pathways */
/*Switch RED ON for East and West pathways*/
// Run Functions
```

```
 north_south_Green();
 east_west_Red ();
}
```

Example 9.2 is for a complete program for switching on and off of north and south pathways and east and west pathways, after intervals of 10 s each. The LEDs in ON states remain so for 30 s period. Function `delay()` provides the period of 30 s and also the intervals of 10 s between switching on of traffic along north and south pathways and switching on of traffic along east and west pathways. Test LED switch off for 6s between the successive repeat cycles.

Example 9.2

Problem

Programming for Arduino controlled traffic-control lights (TLs) at a road junction with control of on intervals: Assume the same set up and LEDs with the same connection as described in Example 9.1. Assume that setup remains the same as before. Therefore, the steps 1 and 2 of Example 9.1 are same in the present example as well.

How can the port LEDs be on-off programmed codes for Step 3: Function loop ( ). The new codes are for signalling and lights control of all four pathways. Assume delays of 10 s each between successive states of LEDs and steady state for 30 s for a pair of pathways.

Write the Function test ( ) to find that the software is functioning as programmed on the board. Assume that test function uses on-board LED at GPIO pin 13 during the period of running of codes.

Solution

Program codes in Step 1 for pre-processing declarations, which embeds onto Uno, are same as in Example 9.1.

```
/*  Assume twelve digital IO ports assigned to external LEDs are port 2 to 12
.
ledY2 = 9; ledG2 = 10; ledR3 = 11; ledY3 = 12; ledG3 = 14;
/* Declare Functions for sequences of traffic lights ON-OFF as follows: */
     void north_south_Green() {
digitalWrite (ledR0, LOW); digitalWrite (ledY0, LOW); digitalWrite (ledG0,
HIGH);
digitalWrite (ledR2, LOW); digitalWrite (ledY2, LOW); digitalWrite (ledG2,
HIGH);
     };
/* Function Switch RED ON for East and West pathways*/
    void east_west_Red() {
digitalWrite (ledR1, HIGH); digitalWrite (ledY1, LOW);digitalWrite (ledG1,
LOW);
digitalWrite (ledR3, HIGH); digitalWrite (ledY3, LOW); digitalWrite (ledG3,
LOW);
      };
```

```
/* Function Switch Yellow ON for North and South  pathways forbrief interval
before change of state */
    void north_south_Yellow (){
digitalWrite (ledR0, LOW); digitalWrite (ledY0 HIGH); digitalWrite (ledG0,
LOW);
digitalWrite (ledR2, LOW); digitalWrite (ledY2, HIGH); digitalWrite (ledG2,
LOW);
      };
 /* Function Switch Green ON for East and West pathways*/
     void east_west_Green (){
digitalWrite (ledR1, LOW); digitalWrite (ledY1, LOW);digitalWrite (ledG1, HIGH);
digitalWrite (ledR3, LOW); digitalWrite (ledY3, LOW);digitalWrite (ledG3, HIGH);
      };
/* Function Switch RED ON for North and South pathways*/
      void north_south_Red (){
digitalWrite (ledR0, HIGH); digitalWrite (ledY0, LOW); digitalWrite (ledG0,
LOW);
digitalWrite (ledR2, HIGH); digitalWrite (ledY2, LOW);digitalWrite (ledG2,
LOW);
      };
/* Function Switch Yellow ON for East and West pathways */
   void east_west_Yellow  () {
digitalWrite (ledR1, LOW); digitalWrite (ledY1, HIGH);digitalWrite (ledG1,
LOW);
digitalWrite (ledR3, LOW); digitalWrite (ledY3, HIGH); digitalWrite (ledG3,
LOW);
      };

/*****************************************************************/
```

Program codes in Step 2 for `setup()` same as in Example 9.1 as setting is unchanged.

```
void setup ( ) {
/* GPIO pins 2 to 12 and 14 are thus assigned port numbers corresponding  */
.
Serial.println (" Twelve 12 external LEDs, R0, Y0, G0, R1, Y1, G1, R2, Y2, G2,
R3, Y3, and G3 corresponds to 12 traffic lights at north, east, south, west
four pathways")
}

/*****************************************************************/
```

Modification in Step 3 function `loop()` are as follows:

```
void loop () {
/*Assume no right turn from pathways or left turn from pathways permitted, just
for simplicity and for learning the basics*/
/*Function Switch Green ON for North and South pathways */
north_south_Green();
/*Function Switch RED ON for East and West pathways*/
east_west_Red();
//Function delay (30000)
delay (30000); //Wait for 30 secomd
/*Function Switch Yellow ON for North and South  pathways forbrief interval
before change of state */
  north_south_Yellow ();
//Function delay (10000);
delay (10000); //Wait for 10 secomd
/*Function Switch Green ON for East and West pathways*/
     east_west_Green ();
/*Function Switch RED ON for North and South pathways*/
   north_south_Red ();
//Function delay  (30000);
delay (30000); //Wait for 30 secomd
/*Function Switch Yellow ON for East and West pathways */
east_west_Yellow  ();
//Function delay  (10000);
delay (10000); //Wait for 10 secomd
/*Testing to show completion of one cycle of traffic light
Let internal board LED flash OFF for 6s before the sequences of lights repeat */
//Statements for  test;
digitalWrite (internalLED, LOW); delay (6000); // Wair 6 s
Serial.println (" One Sequence completed" );
digitalWrite (internalLED, HIGH);
}
```

Emulator and Debugging of Software

Emulation means to do the actions similar to a real entity. Emulator software emulates the running of an embedded device platform, known as target board. Software emulates the embedded device platform onto the computer. The developer observes the actions on the computer.

Emulating and debugging of software ensures reliable, tested and bug-free software

In-circuit Emulation (ICE) means hardware debugging by emulation after connecting the target board to the computer. The computer sets the break points in the embedded software. The results at the end of each break point enable a programmer to find the bug (erroneous code or code-section). Computer can initiate single stepping through the codes. The screen displays the registers in the microprocessor or microcontroller, the variables values at the end of each step or at the end of a specified section of codes. Computer display also displays the contents of memory addresses at each break point.

When embedded Linux is used on the board, then the developer can use a utility, called gdb (GNU debugger), which is open source at GNU website.[8] Connection of target board is through a serial or USB port of computer. Ethernet is also used for device platforms, which support networking.

JTAG (Joint Test Action Group) standard enables testing of the target board onto the computer when JTAG software embeds into hardware, and JTAG connector connects the hardware to computer.

9.2.2 Reading from the Sensors and Devices

Using ADC Analog Input

Assume a temperature sensor is used for measuring between 0 degree and 100 degree Celsius. A sensor sends analog output at an analog input of a 10 bit ADC (Section 7.2.1). An ADC output converts to serial by a parallel input to serial-output (PISO) converter. The serial output connects to the serial SPI input pin at Arduino Uno board. An RH% sensor can also be used in a similar manner where measured value is in RH% in place of degree Celsius (Section 7.2.2).

The ADC output for a sensor at 100 degrees is decimal 1023 (=binary 1111111111) and decimal 0 (=0000000000) for 0 degree. Example 9.3 explains the usage of analog read functions for the Arduino.

Example 9.3

Problem

Programming of Arduino for usages of analog sensor devices at SPI port:

A temperature sensor analog output is given to an SPI at Arduino through a 10-bit ADC and PISO. How will the data be read from the SPI input every hour into an Arduino board from the sensor?

How will the one hour wait loop be programmed and how does the test performed by LED on-off states using a blinking program blink at each 3 s interval?

Solution

Step 1 is declaring the data types, constants, variables and functions used. Second and third steps are coding for setup() and loop(). Example of program codes, which embeds onto Uno are as follows:

```
#include <SPI.h> /*Include the Serial IO functions between Arduino SPI port and
serial input from ADC. */
```

[8] http://www.gnu.org/software/gdb

```
#include <util.h> /*IO utility functions. Includes UART interface, which connects to computer for display of messages on computer-screen. IDE software provides the functions for display using serial interface output of board. */
/* Sensed parameter, say, temperature Sensor Input at A0 pin. Define initial input  = 0. */
#define TempSensorADCinput 0
/*calibCoeff = output change in parameter per unit rise in temerature by 1 degree Celsius.  For example, calibCoeff  = temperature change per unit change in converted digital value of Vinput from the Sensor.*/
/* Define calibCoeff = 0.097752. */
#define calibCoeff 0.097752 /*The calibCoeff is 100/1023 = 0.097752 when sensed parameter analog value is between 0 degree Celsius to 100 degree Celsius and observed digital value is from 0 to 1023 from 10-bit ADC*/
float observedV, parameter; /*observedV = Vinput from Sensor. The parameter = sensed parameter value 0….1023*/
 /*Let internal LED at digital IO Pin 13 be used for indicating successful measurement and  wait for next reading */.
int internalLED = 13; /* initialise internal Port 13 Digital IO Pin LED for test Function. */
/* Name the unit, whether degree Celsius or RH% */
char [ ] unit;
unit = "degree Celsius"; /* Declare the unit of sensed parameter unit*/

/*****************************************************************/
```

Step 2 is as follows for `setup()` statements of the program for setting the board and parameters for measurement of sensed V.

```
void setup() {
/* Declare the unit of sensed parameter initial value and initial observed V*/
parameter = 0.0; // Declare the initial value of the parameter
observedV= 0.0; // Declare the observedV = 0 at minimum
Serial.begin (9600); //Let UART mode baud rate = 9600.
Serial.println ("Arduino Program for input for ADC input at the board");
Serial.println ("Check that Arduino pin A0 connects ADC input from the sensor ");
Serial.println ("Check that Arduino 3.3 V pin connects ADC reference pin at the sensor");
 pinMode (internalLED, OUTPUT);
// Indicate start of the Board and Sequences
digitalWrite (internalLED, HIGH);
// Display the results at serial display monitor on computer where IDE is setup
Serial.println (" Arudino board LED glowing starts when hourly cycle starts for the sequences of LED lights turning high and turns off for successive 3s intervals in order to indicate that system is waiting for the next hourly reading");
```

```
}
/*****************************************************************/
```

Step 3 is as follows for `loop()` statements of the program for measuring the ADC input and parameters for the measurement of sensed V at successive interval.

```
void loop () {
// ADC uses analog reference voltage and it is internally set.
//A reference voltage is 3.3V at the Uno.
// The value is used to convert the acquired value into appropriate reading,
// for example, temperature value in degree Celsius units or into RH%
//10-bit ADC means analog voltage resolution is (Vref/ 1024)
observedV = analogRead (TempSensorADCinput);
//parameter is found from the calibration coefficient
 parameter = calibCoeff*obesrvedV *1023/3.3;
Serial.print ("Temperature ="); //Assume sensed parameter is temperature
Serial.println (parameter, unit);
test ( );
}
/*****************************************************************/
```

Step 4 includes a `test()`. The statements for test function are as follows:

```
void test ( ){ //Start blinks every three second
/* One Hour wait for next reading. Run the loop 600 times. Each wait for 3+ 3
second. 6 s × 600 = 1 hour and LED blinks at 3s ON-OFF intervals*/
digitalWrite (internalLED, HIGH);}
for ( int i = 1, i<=600) , i++) {
delay (3000);
digitalWrite (internalLED, LOW);
delay (3000); // Wair 6 s
digitalWrite (internalLED, HIGH);
}
```

Using the Libraries

Sections 7.5 described software serial libraries and their usages for data communication using serial bus protocols, UART, I2C, USB and CAN. Section 8.3.6 described mBed device platform libraries. The library functions can directly be used during coding.

Using the Timers

Timer functions are required in a number of applications. A number of timer libraries are available. A set of timer functions library is available online.[9] MsTimer stands for millisecond timers with two states. It holds a simple usage (Example 9.4). It has two

[9] http://www.arduino.cc/playground/Main/MsTimer2

functions `set()` and `start()`. First one sets the timer for interrupt after a preset interval and second one to start running the timer.

Example 9.4

Problem

Programming of Arduino for uses of timer library functions:

How will you use the Mstimer2 library to call a function action() after successive intervals which are preset at 3 s?

Solution

Following are the statements which calls `action()` every 3s using Mstimer2.

Step 1: Statements for preprocessor commands, declarations of data types and functions and inclusion of required library files.

```
#include <MsTimer2.h>
void action ( ) {
/* Write statements for actions on preset time over, for example change of
output at an IO pin*/
}
/*****************************************************************/
```

Step 2: Statements are for pin configuring and initial set up statements as in Examples 9.1 to 9.3. A statement specifies a preset value for the interval after which timer MsTimer2 will interrupt and call interrupt handing function. Let function be specified and named `action()`.

```
void setup ( ) {
/* Write pins and initial setup statements. */
/* Set the millisecond timer to execute the function action ( ) after 3000 ms*/
MsTimer2: : set (3000, action);
/* Start the millisecond timer*/
MsTimer2: : start ();}
/*****************************************************************/
```

Step 3: Statements are for codes for actions during 3s and which are repeatedly done, if stop condition not defined inside the loop.

```
loop ( ){
}
```

Using Software Serial Library

A serial interface library has functions to read and write as per serial protocols. A protocol-based communication first transmits the header bits. These may include the address of the device or registers (control, data, status or other registers). The data bits transmit after header bits and control bits, if any. End-bits transmit in the end. Each serial protocol has specific ways of formatting and sequencing during communication.

UART protocol based communication uses two signals, denoted by Tx or TxD (for serial transmission of header, data and other bits) and Rx or RxD (for serial reception). Assume that baud rate is 2400. Then, 240 times a new set of 8-bits data transmits in one second. 10 bits transmit for each set for data, character, command or address. A byte represents a character in a string, a sensed data and command for receiving device or address of destined or receiving device.

Arduino board has pins 0 (Rx) and 1 (Tx) for UART serial communication. A computer connects to the board using these pins. Software serial library has functions, which read and write serial data using Arduino's board. The data input-outputs can be at two pins of digital IO pins, other than 0 and 1, viz. pin 2 and pin 3 as the Rx and Tx, respectively.

UART functions in software serial library set the pins modes as Tx and Rx. The communication initiates RFID Rx and Tx pins at the RFID IC, whenever RFID pins connect to digital IO Tx and Rx pins, respectively. The IC TX becomes low from high when communication initiates from Arduino. RFID IC detects that and first sends a header, which represent a preset address of the device.

The Arduino serial interface reads the header before accepting the tag ID. ID is of 10 characters and the IC sends an end-character from the RFID at the end. End character ASCII code is 13.

Software serial library enables data communications using simple uses of serial read and write functions. Example 9.5 explains the serial read of an RFID tag using UART with the help of the library functions.

Example 9.5

Problem

Programming of Arduino for usages of RFID ID serial-data reading using UART port:

An RFID Integrated circuit (IC) sends serial data to an Arduino board using UART protocol Tx and Rx inputs at the RFID tag pins. How will serial data be read at pins 2 and 3 as Rx and Tx, respectively using Arduino board?

Solution

Following is a program for reading serial data for the sensed value and stored 16-bit coefficients at IC.

Step 1: Preprocessor commands, declarations of data types, functions and constants

```
/* UART communication has 1 start bit. The first serial bit becomes LOW from
HIGH for a period equal to baud interval. Then, 8 serial bits transmit in 8
successive baud intervals. A stop bit then transmits for 1 baud interval. Total
10-bits communicate for each serial write or read.*/


#include <UARTSerial.h> /*Serial IO functions for Arduino UART Serial Interface*/
#include <util.h> /*IO utility functions*/
/* Assume that IC uses address 0x08 for device header which includes address.
Let the IC send 12 bytes, one for the header (address), ten bytes for the sensed
data or RFID tag or ID and one byte for the end character. (Data sheet of the
```

```
device specifies the actual address and format of header bits and tag or ID
bits.)*/
#define UARTDEVICE_HEADER 0x08
int nID 10 /* number of characters in the string for the ID or number of bytes
of sensed data from the device*/
char [ ] ID_characters "" //A String Variable for ID_characters
 int header //Variable for the header received from the device
int end // Variable for end character received
/*Declare internalLED digital IO pin number which shows status of the running
of the Board*/
int internalLED = 13
/*Declare Arduino IO pin numbers which the IC uses to transmit or receive
serially from the board*/
int SerialRxPin 3
int SerialTxPin 4
class SerialRxRequest {
protected: bool loop ( );
private: bool serialRx_request;
 }
SerialRxRequest : : SerialRxRequest (bool serialRx_request) {
/* Set the serialRx_request. Assign true to the request to serial reception of
sensed data or RFID tag*/
this -> serialRx_request = true; }
/*****************************************************************/
```

Step 2: Setup statements to declare the modes of pins and initial states.

```
void setup ( ) {
/* Read data from UART DEVICE. An IC used as RFID tag or IC based sensor may be
using UART communication. */
pinmode (SerialRxPin, INPUT);
pinmode (SerialTxPin, OUTPUT);
pinMode (internalLED, OUTPUT);
digitalWrite (internalLED, HIGH);
digitalWrite (SerialTxPin, LOW);//enable Serial reception
}
/*****************************************************************/
```

Step 3: Continuously running loop till stop condition encounters.

```
bool SerialRxRequest : : loop () {
/*Serial Reception of characters as long as SerialTxPin = = LOW or SerialRx_
request = = true*/
if (SerialTxPin = = HIGH || serialRx_request = = false) (
SerialRx.stop ( ); serialTxPin = HIGH; }
```

```
If (SerialTxPin = = LOW || SerialRx_request = = true) {
SerialRx.begin (4800); /*Assume it communicates at baud rate = 4800. */
ID_characters [0] = SerialRx.read (); // Reader Header character
/*Check header character = 0x08, if match then read sensed data ten bytes and
end character*/
If (ID_characters [0] = = UARTDEVICE_HEADER) {
 for (int i= 1; i< nID +2); i++) {
 ID_characters [i] = SerialRx.read ();
SerialTxPin = HIGH}
// Statements for Serial Print of the result at Serial Monitor
 .
/* Statements for communicating ID Characters or sensed characters on to
Internet or Ethernet or USB*/

 // Call test () for functioning of the serial IOs
  test ( );
 SerialTxPin = HIGH; //stop further read, loop terminate
 return false;)
}
```

Using I2C Serial Protocol

I2C protocol is communication protocol for a serial-bus that communicates in synchronous mode. A device transfers data as a master or slave. Master means the device can address and communicate with a number of slaves. Master sends clock pulses to the slave devices for synchronisation. Maximum connected devices can be up to 127.

At one instance, one device functions as a master and another as slave, and functioning depends on the internal circuit for I2C functions at the device.

Assume that a sensor has built-in ADC as well as I2C interface, and communicates the bytes over I2C SDA and SCL pins. Example 9.6 explains the usages of software serial library for communication of bytes on I2C bus using Arduino. For details, refer to author' book on embedded systems.[10]

Example 9.6

Problem

Programming of Arduino for usages of synchronous serial-data read using I2C bus:

How will the data and coefficients be read from IC based sensor at Arduino serial port? Use the software serial library functions at IDE for reading I2C bus data.

Solution

[10] Raj Kamal, 'Embedded Systems—Architecture, Programming and Design, 3rd edition, McGraw-Hill Education, New Delhi, pp. 169–170, 2014.

The serial pin A4 is for SDA bits and A5 for SCL bits. First connect the sensor IC with A4 and A5. Functions at software serial library enable uses of read and write for serial data.

Assume Arduino functions as master device and an IC based sensor as a slave device. Assume that the sensor IC has addresses for control and data registers. The registers save at EEPROM memory in the IC. A measured 16-bit value saves at two memory addresses, which save the data registers. A programming of control register enables the working of the IC in several modes. The IC uses the data registers and communicates two bytes. One is for higher and the other for lower bytes. Memory also saves and communicates the calibration coefficients for communication as I2C serial data from a set of addresses. Assume device calibration data has 22 addresses in memory for eleven values of 16-bit calibration coefficients. Following is the program for reading serial data for sensed value and stored 16-bit coefficients at the device.

Step 1: Program codes for pre-processing declarations, data types and functions

```
#include <Serial.h> /* Include functions between Arduino and Serial Monitor l*/
#include <Wire.h> /* include Wiring functions*/
#include <util.h> /*IO utility functions and include functions between Arduino
and input-output devices*/
/*I2C protocol envelopes the data with header bits. First seven bits are for the
address of destination device. Define the I2C serial device address for use for
the sensor IC communication.  Assume that sensor uses address 0x08 for device
address and Device Data 1 Address 0x16 and data 2 address 0x17 (Data sheet of
a device specifies the actual addresses).*/
#define I2CDEVICE_ADDRESS 0x08
/*Assume control command is at address 0x02*/
#define char I2DEVICE_CONTROLDATA 0x02
/*Assume Data 1 address is at 0x06 and 2 at 0x0A. */
#define I2DEVICE_DATA1_ADDRESS 0x06
#define I2DEVICE_DATA2_ADDRESS 0x0A
/*Declare two integers 16-bit variables, observedValue1 and observedValue2 */
int observedValue1, observedValue2
/*Declare two float (four bytes each) variables , value one for value1 after
using calibration coefficients and another for value2 after using calibration
coefficients */
float observedCValue1, observedCValue2
/*Declare unit of sensed parameter value, for example, degree Celsius*/
#define char [ ] unit "……..."
/*Declare internalLED digital IO pin number which shows status of the running
Arduino*/
int internalLED = 13;
/*Assume that IC based sensor uses addresses from 0x18 to 0x2C, two addresses
per 16-bit integer coefficient. (Data sheet of the device specifies the actual
address where calibration coefficients can store).*/
#define CALIBCOEFF_ADDRESS [ ] 0x18, 0x1A, 0x1C, 0x1E, 0x20, 0x22, 0x24, 0x26,
0x28, 0x2A, 0x2C
/*Declare array for integer values of calibration coefficients stored at the
```

```
IC memory*/
int calibCoeff [11]
 void I2CDeviceCalibration ( ){ //read calibration coefficients
 int j=0;
/* I2CSerialReadInt is a function to read two bytes, higher and lower bytes.
The function returns a 16-bit integer value*/
 for (int j= CALIBCOEFF_ADDRESS [0], j <= CALIBCOEFF_ADDRESS [10], j = j +2) {
calibCoeff [j] = I2CSerialReadInt (CALIBCOEFF_ADDRESS [j]);
  }
float DeviceSensedResult (observedValue ) {
//Write statements for using calibration coefficients
.
return observedCValue; .
}
test ( ) {
//Write statements for testing the result .
.
return observedCValue; .

/*************************************************************************/
```

Step 2: Program codes for `setup()`

```
void setup ( ) {
  int observedValue0, observedCValue0,
Serial.begin (9600); //Let UART mode baud rate = 9600 for serial monitor
Serial.println (" Arduino Program for input for sensor device using I2C protocol"
Serial.println (" Check that Arduino pin SDA bits at pin A4 and serial clock
SCL pin A5. ");
pinMode (internalLED, OUTPUT);
digitalWrite (internalLED, HIGH);
//Set I2C serial standard clock rate 100 kHz.
// Write data at I2DEVICE
 Wire.begin ();
 /* Get the eleven calibration Coefficients from the IC*/
I2CDeviceCalibration ( ); //Read integers for 11 calibration coefficients
 /* Declare initial offset, for example, Sensor value when temperature is 0
degree Celsius or pressure is atmospheric pressure or photo-current when dark.
*/
observedValue0 = 0; // Declare the initial value of the parameter
/* Calculate sensed Offset value result using calibCoeffs. */
observedCValue0 = DeviceSensedResult (observedValue0);
```

```
}
/*************************************************************************/
```

Step 3: Function `loop()` for the sensor readings and get the temperature values from read value using calibration coefficient

```
 void loop () {
//Read Integer at data 1 address
//Read Integrate at data 2 address
observedValue1 = I2CSerialReadInt (I2DEVICE_DATA1_ADDRESS);
observedValue2 = I2CSerialReadInt (I2DEVICE_DATA2_ADDRESS);
//find the result for sensed data
observedCValue1 = DeviceSensedResult (observedValue1);
observedCValue2 = DeviceSensedResult (observedValue1);
/* Statements for Serial Print of the result at Serial Monitor. */
 .
.
/* Write Statements for delay as per intervals between successive data 1 and
data 2 readings*/
.
.
/* Test function Statements for the test of status of running the program */
 test ( );
}
```

Using Cloud Library at Xively

Paho, earlier known as Cosm, is now called Xively. Xively cloud platform is an application development platform. The cloud platform has library functions, which enable programming for web services, applications and APIs. The applications and APIs may be written in C/C++, Java, Python, dot Net, JavaScript, Perl, PHP or Ruby.

Using an OS

OS is a system software for managing the processes, memory, IOs, network subsystems and devices management. The OS consists of system functions for priorities, multitasking or running a number of threads, and many system functions using given computing device hardware.

Linux distribution libraries are for using Linux OS as a package or set of functions and modules bundled together for specific functions or for specific hardware and distributed for wider usages and applications. For example, in Arduino, mBed, microcontroller circuits and embedded device platforms.

Arduino Yun board has loaded Linux-distribution for Arduino into it. Computer used for Arduino IDE downloads the *Arduino Linux distribution* for running at the Arduino, and then use it as the OS for developing Arduino application programs. Around 256 kB of RAM is required for this purpose.

A code developer can use FreeRTOS or DuinOS for Arduino versions due to their smaller memory needs.

Multitasking

A multitasking environment consists of multiple processes, tasks or threads. A process, task or thread consists of a set of instructions that runs under OS control (supervision). An OS has multitasking (multithreading) functions and many other system functions.

Using Threads

Consider four delay instructions in the program given in Example 9.2. During the period for delay, the system is just waiting for a preset period to elapse. Consider a set of instructions before each delay as one thread, and other sets of instructions and other threads. When the OS is multitasking, then the delay period is utilised in running other in sequence or next to present one priority thread(s). The delay function is a system function and is called `sleep()` or `OS_Delay()` function of the OS. The sleep function makes a system call to OS to block that thread from running for the preset delay period. The sleep function argument specifies that period.

The OS threads can run in sequences when each thread has normal priority. OS can also be made to run such threads for a specified time slice in each cycle of thread run sequences.

Using Real-time Thread Scheduling Library

An RTOS consists of threads, each thread is assigned with a specific priority level, and when a higher priority thread activates, then the system pre-empts the lower priority thread.

A set of built-in functions for real-time scheduling (switching) enable the threads scheduling. A real-time thread-scheduling library is provided in system software. An example of the scheduling library is Atomthreads library which is downloadable online.[11]

Using GNU C Library Version

A version of GNU C library enables Linux OS platforms at the microcontroller (μC) in an embedded device. An example is uClib, which is downloadable from the university website.[12]

Example 9.7 shows the usage of OS multithreading functions with each thread assigned normal priority.

[11] http://www.atomthreads.com
[12] http://www.uclibc.com

Example 9.7

Problem

Programming of Arduino controlled traffic-control lights using multi-threading OS functions:

Reconsider Example 9.2. How will the OS multithreading functions be programmed?

Solution

Step 1: Program codes for pre-processing declarations for inclusions and declarations of data types and functions.

```
/* include Posix Thread library functions in Linux distribution or include
functions of OS being used */
# include <pthread.h>
// declare thread delete request false to disable delete of the thread(s)
bool delete_request false
// Define class named LightsThread for Traffic Lights Project
/*-------------------------------------------------------------------------*/
```

Step 2: Program codes for declaring class for a thread and creation of threads.

```
class SeqThread : public PThread {
// Each SeqThread has an integer variable ID
public SeqThread (int threadID);
// Declare loop ( ) either returns a Boolean variable: true or false
protected bool loop ( );
// threadID is an integer variable specific to a specific SeqThread object
private int threadID;
 }
SeqThread : : SeqThread (int threadID) {
 this -> threadID = threadID; // Assign threadID number to this SeqThread
}
/*-------------------------------------------------------------------------*/
```

Step 3: Program codes for `setup()` for creating a main_thread_list consisting of *numseqthread* sequentially executing threads and write statements in each thread.

```
void setup () {
/* Create specified number of seqthreads.  LightsThread class objects run at
loop () and return true or false */
//Define nine Created seqThread (1), seqThread (2), ……seqThread (9)
/* Each thread has ID from 1 to 9.*/
int numseqthread = 9; //declare number of sequential running threads
/*Write statements for the eight SeqThread objects of class SeqThread  for the
functions in Examples 9.1 and 9.2*/
/* SeqThread (1) waits of signal 1 and runs Function north_south_Green () and
Function east_west_Red () */
.
```

```
/*SeqThread (2) runs Function delay (30000) and sends signal to enable start
of next thread (3) */
.
/*SeqThread (3) waits of signal 3 and runs Function north_south_Yellow () */
..
/*SeqThread (4) runs Function delay (10000) and send signals to enable start
of next thread (5) */
.
/* SeqThread (5) waits of signal 5, runs Function east_west_Green () and
Function north_south_Red () */
.
/*SeqThread (6) runs Function delay (30000) and sends signal to enable start
of next thread (7)*/
.
// SeqThread (7) waits for signal 7 and runs Function east_west_Yellow ()
.
/*SeqThread (8) waits for signal 8, runs Function sleep (10000) and at the end
sends signal to enable start of next thread (9) */
.
/*SeqThread (9) waits for  signal 9, runs, calls Function test ( ) next and at
the end send signal to enable start of thread (1) in next sequences cycle */
.
/*Use a Boolean variable delete_request false to create the threads and run
loop */
if (delete_request = = false ) {
  for (int i = 1; i <= numseqthread; i++) {
  main_thread_list -> add_thread (new SeqThread (i))};
 }
/*delete_request if returned true from the loop then delete the threads to free
the memory */
if (delete_request = = true) {
 for (int i = 1; i <= numseqthread; i++) {
 pthread_delete (SeqThread (threadID)); }
 }
 }
/*-----------------------------------------------------------------------*/
```

Step 4: Follows for `loop()` statements of the program running all nine threads at the main_thread_list

```
bool SeqThread :: loop () {
/* OS runs all nine threads at the main_thread_list as long as loop return
Boolean variable is true*/
// check for delete_request true
if (delete_request) return false;
```

```
/*loops keep running and show status until returns the loop ( ) returns false
 internalLED keep flashing Low from High after the end of a sequence cycle
due to the test Seqthread (9) */
}
/*-------------------------------------------------------------------------*/
```

9.2.3 Programming Embedded Galileo, Raspberry Pi, BeagleBone and mBed Device Platforms

Development of programs is done using an IDE. The cycles of edit-test-debug are used until the simulated results show the successful test runs of the programs. Following are the details for popular device platforms.

Intel Galileo IDE

Intel Galileo IDE for Windows is downloadable from the Galileo website.[13] Intel Galileo and Arduino combined features are:

- Similarity of IDE with Arduino: Galileo adds Arduino X86 board and new IDE also enables firmware upgrading the board.
- Compatibility: Compatible with Arduino shields and similar ICSP header, serial port, 14 digital I/O pins and 6 analog inputs
- Ethernet Library: Compatible in usability with Arduino's Ethernet library and can connect with HTTP connection using standard WebClient example
- PCI Express Mini Cards: Can connect to GSM, Bluetooth, WiFi cards using Arduino's WiFi library.
- USB Host Port: USB is a dedicated port and usable with Arduino USB Host library and keyboard or mouse connection to computers.
- TWI/I2C, SPI Support, Serial Connectivity and MicroSD Support: Compatible with standard Arduino libraries.
- Linux distribution running on board: Just 8 MB flash enables node.js (for web projects), sound tools: ALSA, video tools: V4L2, Python, SSH, computer vision openCV

Raspberry Pi IDE

RPi programming using Adafruit, BlueJIDE or NinjaIDE

Raspberry Pi (RPi) provides a smart-card-size computing system with media co-processors. A developer can work with the standard keyboard and display. The Pi supports number of languages including Python. Readily available libraries PyPi are for RPi programming using Python. RPi supports *OS Raspbian Ubuntu distribution of Linux* (also known as Snappy) and has features of securely running the autonomous machines, M2M and IoT devices.

[13] https://downloadcenter.intel.com/download/24355/Intel-Arduino-IDE-1-6-0

Number of IDEs is available for developing computing devices with media capabilities. Following are the popular ones:

AdafruitIDE is a web-based IDE.[14] The IDE provides a web-based development environment. RPi connects to LAN and coding can be done using Python, JavaScript or Ruby languages. Adafruit also provides a customised OS in place of Raspbian that functions with keyboard and displays monitor connected with RPi. A secure shell (SSH) for cryptographic network protocol for the application layer enables remote login and remote connections of the systems, keyboard and monitor.

BlueJIDE is a Java-based IDE at BlueJIDE.[15] BlueJIDE runs with the help of Java Development Kit, and apart from this also runs the Java programs including Java 8 on RPi using PI4J library.

Ninja-IDE source[16] is an IDE which stands for not-just-another IDE. Ninja enables the uses of file functions as well building of applications in Python. It runs in cross-platform environment, RPi as well as Linux, Windows and Mac OSX. Its features are:

- Ability to create plugins, thus extend code editor functionalities, assistance for project development and load several plugins' widgets for multi-purposes.
- Code editing with multiple languages, code assistant for code completion, code navigation, web navigation and imports from anywhere, contextual menu in tabs to perform quick actions, code jumps, breakpoints and bookmarks navigation implementation, profile manager, code error finder and static error files detection.
- Code locator for fast and direct access to a class, function or file with pop up over the text fields.
- Symbol exporter, files and file usages finder, web inspector and virtualenv have been added, which can be specified in the project creation from Ninja IDE or for an existing project in Project Properties.
- Supports use of a console for embedded Python. Console manages Python project automatically. It allows the user to perform file management related task in the IDE itself and saves the description of the project files itself.

Section 9.3 will describe Java coding using Eclipse Pi4J. Example 9.10 will explain how to program RPi board using eclipse Pi4J implementation.

BeagleBone IDE

Recapitulate Section 8.3.5 for details of BB board for development. BB capabilities extend using capes, just as Arduino by shields. BB extensions provide two 46-pin dual-row expansion headers, motor control, prototyping, battery power, Ethernet, MicroHDMI, VGA, LCD, USB host and USB client for power and communication and other functions. One can work with a standard keyboard.

OS is built-in Ångström distribution of Linux. The OS is pre-loaded with BB boards. BB supports autonomous machines, M2M and IoT devices. BB has features of running

[14] https://learn.adafruit.com/webide/overview
[15] http://www.bluej.org/raspberrypi/index.html
[16] http://ninja-ide.org/

cloud-based IDE and Node.js web BoneScript library functions.[17] The BB supports Ubuntu, Android and Debian. Node.js is a JavaScript platform for building scripts using rich libraries and networking and real-time capabilities.

BeagleBone development supports usages C/C++ Toolchain with a gcc Cross Compiler from linaro.org and the Eclipse IDE in Windows.

mBed IDE

Recapitulate Section 8.3.6 mBed ARM® has online IDE, an open extensible source code which includes schematics, software, middleware and C/C++ software platform tools. A web browser is used on computer hosting the IDE that uses ARMCC C/C++ compiler at cloud. The tools enable use of MCU firmware core libraries, peripheral drivers, networking, RTOS runtime environment, build tools and test and debug scripts. The IDE enables a developer to develop the applications on mBed platform for the IoT/M2M. IDE latest version and appropriate OS are open source Eclipse and include GCC ARM embedded tools.

The mBed OS provides a C++ application framework with the ARM mBed community libraries to create device applications, component architecture and facilitation of automated power management.

9.2.4 Programming Embedded Device Platforms for Internet Connectivity

Using the Ethernet and WiFi Libraries

Arduino Ethernet Library and header files in the library enable the usage of Serial IO functions between Arduino SPI port, IO utility, Ethernet shield, Ethernet client, Ethernet server, DNS, vDHCP and UDP protocol functions.

Using IP Library

IP library is used when a set of built-in codes and the codes which enable use of the functions of the library for creating stack for the TCP/IP protocols based communication are needed. Examples of IP library functions are (i) lwIP (lightweight Internet Protocol) which enables TCP/IP communication with the little memory requirements (nearly 40 kB flash/ ROM and few tens of kB RAM). The library is downloadable from library website[18] and (ii) μIP (micro Internet protocol library functions enable TCP/IP communication with very little memory requirements (few kB RAM) and for use when buffering of packets is not required. When a small amount of data communicates then buffering is not required. Usage of buffering in communication is for incoming packets and for outgoing unacknowledged packets.

[17] https://www.c9.io
[18] http://www.savannah.nongnu.org/projrcts/lwip/

Using Cryptographic Library

Library functions for secure communication for the IoT

Security of data from embedded device platforms is of high importance in web applications and services. Assume that the device-end has a user ID, say A1 for authentication at another end. A cloud, web application or service end has an application ID, say A2 for authentication at the user's end.

The A1 needs to store at the application or service end database and A2 needs to store at the user-end database. A1 communicates for authentication at the other end. The application matches received A1 with the database stored A1. If a match is successful, then the user end stands authenticated.

Two security risks are—one is during communication to the other end, some system in-between, such as, a switch or router or server S1 reads A1 or A2 received from another end and resends the same to the authentication end. Another security risk is that some system modifies A1. The application will authenticate the S1 authentication code in place of the user end. Similarly, the user end can wrongly authenticate the application or get a modified text.

Two basic requirements are thus authentication and encryption of data. Cryptographic library is used when a set of built-in codes and the codes which enable use of the cryptographic functions, like OAuth 1.0 protocol functions are needed. The OAuth 1.0 functions enable securing the data from modifications or replaying that by some other software.

A library, called *cryptosuite* is downloadable online.[19] Encryption key length can be 128, 192 or 256. Data encryption can use AES256 (Advanced Encryption Standard 256-bit encryption key length) or DES (Data Encryption Standard). Arduino AES 256 library is downloadable and used for Arduino boards.

Adding Security and Authentication

Securing Communication of Authentication Code (Secret Key)

Authentication using a secret key and a hash function in place of communicating a plain text string

Security adds in the authentication code A1 when hash of A1 H (A1) communicates to the application end and application matches the stored H(A1) with the received H(A1).

Security adds by communicating hash of A1 or secret key (a text string) in place of communicating plain text string. A function, called hash function creates a fixed length string, called hash of the input string, for example of A1. The H(A1) uses the original string, say A1. User communicates H(A1) in place of original A1.

The application or service end stores the H(A1) for identifying string A1. A1 cannot be identified from stored H(A1) when that mismatches with the received H(A1) at that end. Assume H(A1) changes to H'(A1) due to in-between modification. The received H'(A1)

[19] http://code.google.com/p/cryptosuite

will not match with stored H(A1). Therefore, the authentication code of user A1 will not be identified.

Examples of the usages of hash function are SHA1/SHA256 secured hash algorithms or MD5 message digest algorithm. Example 9.8 lists the steps involved in the method.

Example 9.8

Problem

Using Hash algorithm for secure communication of authentication codes:

How will the authentication code be created for communicating securely from an embedded device platform using SHA1 hash function?

How does the SHA1 used at the application?

Solution

(i) The statements for authentication code from an embedded device platform using a hash function are as follows:

Step 1: Statements are preprocessor commands, declarations of data types and functions and include the required library files.

```
/*Cryptographic library functions */
/*Include <sha1.h>, <sha256.h> or <md5.h> header files*/
#include <sha1.h>
/*IO utility functions*/
#include <util.h>
/* The authentication codes declares as set of unsigned integer of 8-bit each
at a pointed address */
uint8_t *authcode
/* The hash authentication codes declares as set of unsigned integer of 8-bit
each at a pointed address*/
uint8_t *hashauthcode
// Assign Values to authcode
.
.
// Write other preprocessor statements
.
.
}
/*************************************************************************/
```

Step 2 is as follows for `setup()` statements for initialising SHA1, creating hash code and setup GPIO pin modes.

```
void setup ( ) {
sha1.init ( );// initializes SHA1
hashauthCode = sha1.result ( ); // creates hash of authentication code, authCode
```

```
pinMode (internalLED, OUTPUT);
digitalWrite (internalLED, HIGH);
//Write statement for display on Serial Monitor of authentication code
Serial.begin (9600);
.
.
/* Write Statements for display of hash of the authentication code a */
..
}
/*************************************************************************/
```

Step 3 is write `loop()` statements for communication of the code, and test.

```
void loop () {
 /* Write Statements for communication of hash of the authentication code */
.
.
 test ( );
}
/*************************************************************************/
```

(ii) The application receives hashauthCodeNew in place of hashauthCode. Then, a Boolean variable mismatch will be equal to true when the authentication fails. The preprocessor and `loop()` statements are:

```
/* First include the functions from Cryptographic library using pre-processor
statement */
#include <sha1.h>/* or #include <sha256.h> or <md5.h>*/
/*IO utility functions*/
#include <util.h>
/* The hash of authentication code used to check the match and new hash received
are sets of unsigned integer of 8-bit each at a pointed addresses*/
uint8_t *hashauthcode, *hashauthcodeNew
/*-------------------------------------------------------------------------*/
loop ( ) {
/* Write statements for receiving the hashauthCodeNew */
.
/* Write statements for matching the hashauthCodeNew with stored hashauthcode*/
 .
/* Write statements for receiving the hashauthCode*/
If (match_request = true) {
if (hashauthCodeNew == hashauthCode) {
mismatch = false;
```

```
match_request = false; /* Matching is only once and thus cancel the match
request */
} else
{mismatch = true;}
. }
```

Data Encryption and Decryption

Data needs protection from read and use by an in-between system. Encryption ensures the protection needs. Uses of standard algorithms, such as AES128, AES192, AES256 or DES enable the encryption and decryption. Example 9.9 explains the statements in C/C++ for using the AES256 encrypting and decrypting algorithms.

Example 9.9

Problem

Encryption of device end messages and decryption at application end:

(i) How is the encrypted message or sensed data created for secure communication from an embedded device platform?

(ii) How does the message decrypt at the application end?

Solution

(i) The statements for authentication code from an embedded device platform using a hash function are:

Step 1: Preprocessor commands, declarations of data types and functions and include the required library files.

```
/* First include the functions from Cryptographic library using pre-processor
statement */
#include <aes256.h>
/*IO utility functions*/
#include <util.h>
/* Contextual parameters Data type declaration */
aes256_context context /* Number of contextual parameters required that enables
AES algorithm execute. These save at the pointed address context*/
.
.
// Write other preprocessor declarations
.
.
}
/*************************************************************************/
```

Step 2 is as follows for `setup()` statements for 256-bit key and message string.

```
void setup ( ) {
uint8_t key [ ] = {…, …., …..., ……, ……}/* Curly bracket has key of thirty two
8-bit unsigned integer numbers, each number separates by a comma. */
aes256.init (&context, key);// initializes AES256
char *message = "……"// Assign Message characters for communication
/*Write statement for display on Serial Monitor of authentication code */
aes256_encrypt_ecb (&context, (uint8_t) message);

Serial.begin (9600);
.
.
/* Write Statements for display of encrypted message */
..
}
/*************************************************************************/
void loop () {
 /* Write Statements for communication of encrypted message */
.
.
 test ( );
}
```

(ii) Assume that the application receives the encrypted message. The statements for decrypting can be:

```
#include <aes256.h> /* First include the functions from Cryptographic library
using pre-processor statement */
#include <util.h> /*IO utility functions*/
aes256_context context /* Number of contextual parameters required during
execution of AES algorithm. These save at the pointed address context*/
char *message = "……"// Assign Message characters for communication



.
// Write other preprocessor statements
.
.
}
/*************************************************************************/
```

Step 2 is as follows for `setup()` statements for 32 bit key.

```
void setup ( ) {
```

```
uint8_t key [ ] = {…, …., …..}/* Curly bracket has key of thirty two 8-bit
unsigned integer numbers, each number separated by comma. */
aes256.init (&context, key) ;// initializes AES256

Serial.begin (9600);
.
.
}

loop ( ) {
// Write statements for receiving the message for decryption
.
.
 aes256_decrypt_ecb (&context, (uint8_t) message);
aes256_done (&context); /* It ends the initialized AES256*/
.
/* Write Statements for display of decrypted message on serial monitor*/
..
// Write statements for test ( ).
}
```

Reconfirm Your Understanding

- IDE enables development of embedded device software. Arduino IDE has simplicity and is based on processing language and wiring library functions.
- Simplicity of Arduino is clear from the fact that only two functions are necessary to define the executable program functions for the device, namely, `setup()` and `loop()`. A serial monitor at the IDE consisting computer enables the results from the serial port at a device platform. The developed software at the computer also downloads into the device using the port.
- A traffic-lights monitoring program shows how to (i) program `setup()` and `loop()` functions using Arduino IDE C/C++ for programming the GPIO pins, (ii) control the intervals between three states of each pathway using delay() function, and (iii) *test* using serial monitor and testing of the program functions.
- Number of library functions provide ease in programming. Usages of timer and software serial libraries show how to program for (i) the timer functions, (ii) reading of RFID ID tag using UART functions, and (iii) I2C communication between sensor communicating sensed data using I2C and Arduino I2C port.
- When multiple threads of a program execute on a device platform, an OS provides for system calls for these functions. Rewriting of Arduino controlled traffic control lights monitoring program shows how to program using multi-threading OS functions.
- Intel-Arduino IDE functions enable development of embedded device software for Intel Galileo device platform.

- OS Raspbian Ubuntu distribution of Linux, Snappy enables programming for secure running of autonomous machines, M2M and IoT devices using RPi.
- A web-based IDE Adafruit enables development of embedded device software for RPi as well as BB.
- BlueJIDE is a Java based IDE for RPi.
- Ninja-IDE is IDE for RPi enabling development of embedded device software at Windows, Mac OX and Linux cross platform environments. The Ninja enables file functions as well for building applications in Python using an embedded Python console.
- IDE and OS distributions for BB enable the embedded software development.
- Things communicate to the Internet using functions at Ethernet, WiFi and IP.
- Things communicate securely using crypto-library functions.
- The crypto-library functions enable the programming, secure communication of data for IoT. Two security risks are taken care by (i) using a secret key and its secure communication, (ii) using encryption and decryption functions, such as AES128, AES192, AES256 or DES).

Self-Assessment Exercise

1. Why does prototyping embedded software become easy using an IDE for a device platform? ★
2. List the steps used in the traffic light program, function loop() in Example 9.1. ★★
3. Write a program to read three digital inputs, viz. 1-bit for streetlight working status (Yes or No), 1-bit for ambient surrounding light status (insufficient or not insufficient) and three bits for traffic density level using Arduino IDE and device platform. ★★★
4. How will you use the library functions to call a function action1() after a preset interval of 1s and another after 3s from the start? Use Example 9.4 as hint. ★
5. List the soft serial library functions for Arduino and write their usages. ★
6. When does a device platform use UART, I2C and SPI bus for communicating with a sensor? ★★
7. Using Example 9.7, write the advantages of using the threads functions of the OS. ★★★
8. List additional features in Intel Galileo device platform over Arduino. ★★
9. Tabulate for comparing the usages and features of IDEs for Raspberry Pi. ★★★
10. Show the steps in software for communicating the data stores at the device platform over the Internet using Ethernet library functions using Example 8.5. ★
11. List security risks when communicating a secret key over Internet. ★
12. List and write usages of the cryptolibrary functions for secure communication of the data over Internet. ★★★

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

9.3 DEVICES, GATEWAYS, INTERNET AND WEB/CLOUD SERVICES SOFTWARE-DEVELOPMENT

LO 9.2 Analyse and program the devices, gateways, Internet connectivity, web and cloud applications using the open-source implementations of Eclipse IoT stack

Recall Chapter 3—connected devices in IoT/M2M use the CoAP and LWM2M web-communication protocols and messaging-protocols, such as message-cache, Message Queue Telemetry Transport (MQTT), and Extensible Messaging and Presence (XMPP). MQTT is a publish/subscribe (Pub/Sub) protocol. The devices connect, network and communicate over the web. They use the communication gateway, SOAP, REST, RESTful HTTP and WebSockets. Figure 3.1 showed the connected devices, protocols and usages of the Internet in IoT/M2M applications and services.

Figure 9.1 shows five levels for software development for applications and services in the IoT or M2M. The software needs are for the devices, local network, gateway, cloud/web connectivity and web/cloud APIs.

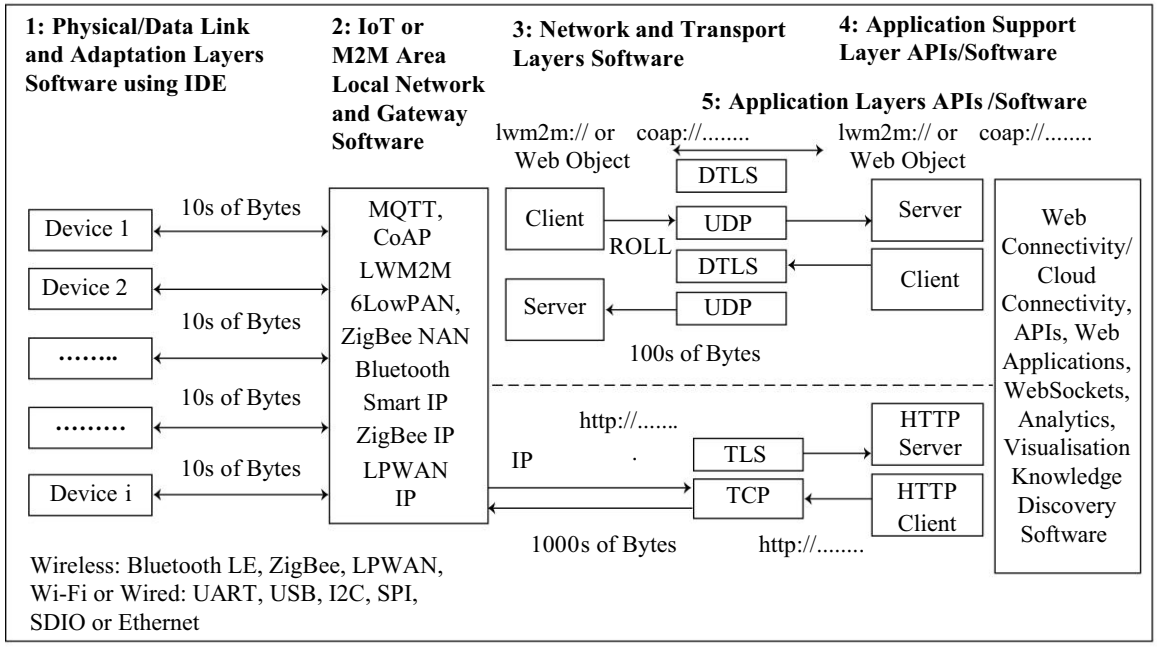


Figure 9.1 Five levels for software development for applications and services for IoT or M2M

Software, such as *Eclipse IoT*, enable the development of software for the first, second and third levels. The software enables the device gateways connectivity to the Internet and cloud server. Eclipse IoT enables open source implementations of IoT protocols. The implementable protocols include MQTT

Eclipse IoT stack for five levels of software development for the applications and services in IoT/M2M

CoAP, OMA-DM and OMA LWM2M and Internet connectivity protocols (Sections 3.2, 3.4 and 4.2).

9.3.1 Use of Software Stack for an Intended Complete Solution

Section 9.2 discussed the first-level software. Now consider the software for higher levels. Each level has characteristic complexity and fragments. Recall Sections 3.3 to 3.5. The connected devices use a variety of protocols, such as LWM2M, CoAP, MQTT, and methods for connecting to the web. Web communication uses the Gateway, SOAP, REST, RESTful HTTP and WebSockets functions.

A stack is a full set, consisting of frameworks, applications and services that are minimum needs for intended complete solution. Following sections describe Eclipse IoT (www.iot.eclipse.org) stack for end-to-end IoT/M2M solutions.

9.3.2 End-to-End IoT Solutions with Java using Eclipse IoT Stack

Open Services Gateway Initiative (OSGi) (now OSGi Alliance) provides and maintains open standard specifications. OSGi describes the specification of management of Java packages/classes in a modular system, which enables the implementation of a complete and dynamic component model. A component means software which can reuse a core set of frameworks and services for provisioning the solutions. The components and applications deploy in the form of bundles and can be remotely installed, started, stopped, updated, and uninstalled without requiring system reboot.[20]

OSGi in addition gives specifications for service platforms in Java language. The component or application life-cycle management uses a set of APIs. When a service registers, then service bundles detect the deletion or addition of new services and get adapted.

OSGi in addition to original focus, now has evolved specifications far beyond the gateways, such as *eventing*, *configuration management*, *clock*, *crypto* (aes, base64, sha-1), *geolocation data, cloud* and application servers and services for the connect and manage functions.

OSGi service functions for Clock, Crypto (AES, base64, SHA-1), Geolocation, Data and Cloud Services

OSGi thus find applications for the management of Java packages and classes. Examples of applications and services are fleet management, automobiles, automobiles, smart phones and now in IoT/M2M solutions.

Eclipse IDE is an application using OSGi specifications. Eclipse open IoT stack is a set of Java frameworks, protocols, development tools and OSGi services. The features of Eclipse IoT stack are:

- Provides open source specifications which are as per open OSGi standard specifications

[20] https://en.wikipedia.org/wiki/OSGi

- Provisions for simpler open source implementations, and programs, services and bundles development using the open source Java frameworks and services
- Consists of the components and frameworks for IoT solutions. The stack takes care of the complexity of creating IoT solutions and enables fast development of the solutions
- Provisions for open source technologies which enables easy programming in Java for the device platforms, and running the codes in JVM or Eclipse Concierge (a lightweight implementation of OSGi runtime)
- Enables usages of protocol functions provided in lightweight M2M (OMA M2M standard), MQTT (OASIS IoT standard), CoAP (IETF IoT standard) and standard network protocols. The functions provide the connectivity and interoperability of the device gateways
- Provisions the IoT gateway services for remote management and applications management
- Provisions for new solutions for devices connectivity to the Internet and the remote management and application management functions and APIs for using server or cloud provided functions
- Support of large number of institutions including Cisco and IBM

Eclipse IoT stack includes device platform for physical/data-link and adaptation layers. Table 9.1 gives the implementations and frameworks for (*Gather + Consolidate*) and local network and gateway software (*connect*) levels.

Table 9.1 Eclipse implementations and frameworks included in Eclipse IoT stack

Eclipse Implementation/Framework	Description
Eclipse Pi4J	A framework based on WiringPi and PiFace, Gertboard and other shields, which support the I2C/SPI/GPIO interfaces in sensors, actuators and device platforms, for the RPi. Similar framework is for BB platform and capes used with BB.
Eclipse Koneki	A set of functions for embedded Lua language based development of device applications.
Eclipse Mihini	An embedded Lua runtime which provides provisions for hardware abstraction and other services.
Eclipse Krikkit	Provisions a system of rules when programming edge devices. For example, when configuring the device platforms.

It is beyond the scope here to cover each Eclipse IoT component for device platforms. Following section describes the usages of Pi4J components.

Eclipse.Pi4J

RPi board has 8 × GPIO plus the following, which can also be used as GPIO: UART, I2C bus, SPI bus with two chip selects, I2S audio, +3.3 V, +5 V and ground pins. Example 9.10 gives an *Eclipse Pi4J* implementation of GPIO/I2C/SPI protocols using RPi device platform.

Example 9.10

Problem

Programming of RPi board using the eclipses Pi4J implementation:

Assume RPi board (Section 8.3.1) as embedded device platform. Assume that GEN0 to GEN6 are the GPIO dedicated pins in a given board.

How are GPIO pins setup using Pi4J? When a GPIO_01 pin connects status LED, how will the port GEN_1 LED be programmed? When a GPIO pin connects traffic red, yellow and green lights, how will the port GPIO_02, GPIO_03 and GPIO_04 be programmed?

Solution

Example of program codes which embeds onto RPi are as follows:

Step 1 statements are constructors and methods.

```
import // Import Pi4J from www.eclipse.org
/* A gpio object specified. */
GpioController gpio = GpioFactory.getInstance ( );
/* gpio object method for RPi pin GPIO_01 named status LED setting in state
HIGH. */
GpioPinDigitalOutputPin pin01 = gpio.provisionDigitalOutputPin (RaspiPin,
GPIO_01, "statusLED", PinState.HIGH);
/* pin01 to be set LOW after 2000 ms) */
Thread.sleep (2000);
pin01.low ( ); ;
/* Return */
gpio.shutdown ( );
/*****************************************************************/
```

Step 2 statements are gpio object specifications using constructors, and methods of sleep, low and high.

```
/* Four gpio pin objects, pin02, pin03 and pin04 specified. */
GpioController gpio = GpioFactory.getInstance ( );
/* gpio object initialization methods for RPi pin GPIO_02, named as REDLED
setting in state HIGH and Other two in LOW states at the start. */
GpioPinDigitalOutputPin pin02 = gpio.provisionDigitalOutputPin (RaspiPin,
GPIO_02, "REDLED", PinState.HIGH);
```

```
GpioPinDigitalOutputPin pin03 = gpio.provisionDigitalOutputPin (RaspiPin,
GPIO_03, "YELLOWLED", PinState.LOW);
GpioPinDigitalOutputPin pin04 = gpio.provisionDigitalOutputPin (RaspiPin,
GPIO_04, "REDLED", PinState.LOW);
 /* pin02 to be set LOW after 30000 ms and pin03 to be set HIGH) */
Thread.sleep (30000);
pin02.low ( );
pin03.high ( );
/* pin03 to be set LOW after 5000 ms and pin04 to be set HIGH) */
Thread.sleep (5000);
pin03.low ( );
pin04.high ( );
/* One Cycle completes on one pathway */
```

Eclipse IoT stack includes gateway, network, transport and application-support layers. Table 9.2 gives the implementations and frameworks for gateway software (connect) and network and transport layers software (Connect + Assemble + Manage).

Table 9.2 Eclipse implementations and frameworks included in Eclipse IoT stack for gateway, network, transport and Application-Support layers

Eclipse Implementation/ Framework	Functions for IoT/M2M	Description
Eclipse Wakkamma	LWM2M clients with LWM2M server	A set of library functions for LWM2M in a 'C' implementation.
Eclipse Californium	CoAP clients. Secure DTLS and CoAP server	An implementation in Java of CoAP, including DTLS for IoT security.) A Californium-based sandbox server can register the CoAP clients. The server (CoAP://iot.eclipse.org:5683) also interacts with CoAP clients.
Eclipse Lehshan	LWM2M clients. Secure DTLS and LWM2M sandbox server	Java implementation of LWM2M for device management in Java and includes DTLS for IoT security. A Lehshan-based sandbox server can register the LWM2M clients. The server (coap://iot.eclipse.org:5684) interacts with the client Web UI and REST API. The server interacts with registered LWM2M clients.[21]

(*Contd.*)

[21] http://www.iot.eclipse.org/lwm2m

Eclipse Moquette	C MQTT clients at devices and applications in pub/ sub mode using TCP and Sandbox server	An implementation in 'C' of the publish/subscribe protocol MQTT using TCP. A sandbox server (tcp:// m2m.eclipse.org:1883).at cloud/web interacts with MQTT clients at an application running at a computer/tablet/ mobile phone.
Eclipse Paho	MQTT clients at devices and applications in pub/ sub mode using a MQTT Broker	A Java implementation of the MQTT client and Moquette which uses a Java MQTT broker (m2m. eclipse.org/paho); Paho also uses the JavaScript, Lua, Python, dot Net, dot net compact, dot net micro, Windows Phone, Android.
Eclipse OM2M	M2M TEST APIs network, gateway and sandbox server interactions	An implementation of the ETSI M2M standard. It provides a horizontal service capability layer (SCL) that can be deployed in an M2M network, gateway or device. M3DA sandbox server (http://iot. eclipse.org:44900) interacts with REST API (http:// iot.eclipse.org/m3da)
Eclipse Ponte	M2M CoAP and MQTT gateways and MQTT-MQTT broker and MQTT-CoAP broker	A framework based on Java and OSGi services for IoT and M2M Gateways. Ponte gateway includes bridge between the M2M/IoT protocols, for example, between MQTT and CoAP protocols using devices to the Web. Ponte broker functions connect the MQTT device platform applications network at one end with the CoAP device platform applications network at another end (Section 3.3).
Eclipse Jetty	WebSockets bi-directional communication	A set of Java methods and annotations for WebSocket objects creation and sessions (Sections 3.4.5 and 9.4)*
Eclipse Kura	Gateways. Services, cloud connectivity, management of device, network configuration and application	A set of OSGi based application framework and services for building IoT and M2M gateways. Services include (i) device abstraction, CAN bus for automotive embedded devices, configuration management, and integrated device cloud functions, and (ii) provides application portability, modularity and application management and provisions for built-in OSGi services for IoT apps.

*English meaning of annotation is addition of notes to a page or diagram or picture or painting for better understanding.

Covering all components of Eclipse IoT here is beyond the scope of this book. The following section describes the usages of Paho.

Eclipse.Paho

Program implementation in Pub/Sub mode for the MQTT clients and MQTT broker using eclipse Paho

Recall Section 3.3.1. A communication mode is request-response mode. An HTTP client makes a request to a server which sends a response. The client is said to pull the required messages (data) in response. When physical devices, such as streetlights exchange data between each other, or when other system, such as a central server or controller, controls them, then data exchanges are just 10s of bytes. The devices need a lightweight method in place of the HTTP.

A communication mode is pub-sub mode. MQTT protocol provides for three objects: MQTT clients at the devices, MQTT broker and MQTT clients at the application(s). Assume that a device, such as a streetlight deploys MQTT client for sending the sensed data and receives commands from the application end through a broker (intermediate server). MQTT-client, unlike HTTP, does not have to pull the messages (commands) which the device needs for its control. An MQTT broker pushes the messages to the client, provided a client has subscribed to those. A broker functions like a dashboard, where the messages first reach from the sources and from there the messages then dispatch to the subscribers (ones who makes a request for the messages being dispatched).

Each MQTT client may be required to use an *always connected* UDP (or TCP) connection with a broker. The MQTT broker uses buffers for all messages. This facilitates the dispatch when a client reconnects after an interruption. An interruption is always a possibility in ROLL environment.

Recall the example of Internet of Streetlights (Example 1.2) and devices connectivity using five levels (Figure 9.1). Consider use of MQTT Pub/Sub protocol for the connectivity between devices and the application layers. Figure 9.2 shows the connected streetlights publishing the sensed data and subscribing for the control data, and using MQTT clients during communication with the application layer. The figure shows that MQTT clients communicate through an MQTT broker with other clients at the devices or at the applications and central controller. The figure also shows sequence 1, 2, 3 and 4 for message exchanges.

Example 9.11 gives an *Eclipse Paho* Java implementation of MQTT clients broker architecture.

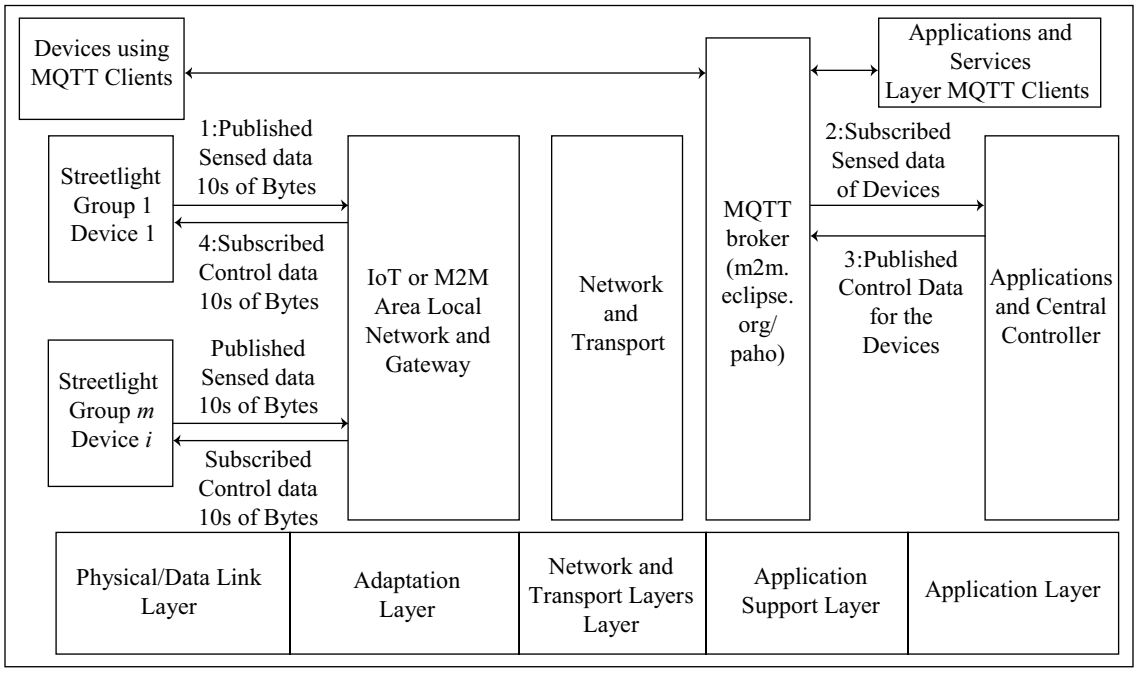


Figure 9.2 Connected streetlights publishing the sensed data, subscribing for the control data and using MQTT clients during communication with the application layer MQTT clients through an MQTT broker

Example 9.11

Problem

Communication of sensed data to the applications and central controller for the control of streetlights (Example 1.2) using the Eclipse Paho implementation of MQTT:

(i) Import Java packages for MQTT clients, broker and applications and the Publisher and Subscriber classes for constructing the objects for publishing sensed data of the devices, such as streetlights.

(ii) Implement the codes for publisher class for control data from the central controller and sensed data from the devices.

(iii) Implement the codes for subscriber class for subscribing of sensed data by the application/central controller.

(iv) Implement the codes for subscriber class for subscribing of control data by the device clients

(v) Implement the codes for observing 'Last Will and Testament' (LWT) message on disconnection/failure of communication of messages on topic sensed data and control data and communicating again on reconnection using MQTT broker.

(vi) Implement the codes for threads for publishing the messages for the sensed data. For example, for traffic presence and lighting needs in Internet of Streetlights.

(vii) Implement the codes for publisher and subscriber threads at the devices and application in example of Internet of Streetlights

Solution

Assume that a Java package is used for 'Internet of Streetlights':

Step 1: The program implements in Paho that imports following packages.

```
Package org.eclipse.pahoIoTStreetLights;
import org.eclipses.paho.client.mqttv3.MqttClient;
import org.eclipses.paho.client.mqttv3.MqttException;
import org.eclipses.paho.client.mqttv3.MqttMessage;
```

Following are the sets of coding required:

(i) Publisher class for constructing the objects for publishing sensed data of the devices, such as streetlights

```
/*Let first step is that within the class Publisher, declare URL of the MQTT
Broker and create instance of class MQTTClient. Use MAC Address as Client_ID
for publisher ID and the initialization of the MqttClient instance is as per
following code. Use -pubDevice as a suffix for the Client_Id. Following is
the Java code for that.*/
public class Publisher {
public static final String MQTT_BROKER_URL = "tcp://broker.mqttdashboard.
com:1883";
/* client is the instance of MqttClient */
private MqttClient client;
 public Publisher()
 {
String p1 = "-pubDevice";
/* Declare clientDevice_ID using method getMacAddress ( ). */
String clientDevice_Id = Utils.getMacAddress() + p1;
try {
mqttClient = new MqttClient(MQTT_BROKER_URL, clientDevice_Id);
}
catch (MqttException e) {
/* Write codes for callback method */
/* On exeception, Assume that callback method is print the stack trace. */
e.printStackTrace();
System.exit(1);
     }
  }
}
/*********************************************************************/
```

(ii) Publisher class for publishing control data at the application/central controller

Let next step be use MAC Address as Application ID for subscriber ID and the initialisation of the MqttClient instance is almost the same as above publisher above code, except that use-pubApp as a suffix for the Client_Id.

(iii) Subscriber class for subscribing of sensed data by the application/central controller

Let next step be use MAC Address as Application ID for subscriber ID and the initialisation of a MqttClient instance is as per publisher codes, except that use -subApp as a suffix for the Application_ Id. Following is the Java code for that.

```
public class Subscriber {
public static final String MQTT_BROKER_URL = "tcp://broker.mqttdashboard.
com:1883";
/* client is the instance of MqttClient */
private MqttClient client;
 public Subscriber ()
 {
String s1 = "-subApp";
/* Declare client at the Application_ID using method getMacAddress ( ). */
String Application_ID = Utils.getMacAddress() + s1";
try {
mqttClient = new MqttClient(MQTT_BROKER_URL, Application_Id);
}
catch (MqttException e) {
/* Write codes for callback method */
/* On exeception, Assume that callback method is print the stack trace. */
e.printStackTrace();
System.exit(1);
     }
  }
}
/*****************************************************************/
```

(iv) Subscriber class for subscribing of control data by the device clients

Let the MAC Address be used as Client_ID for subscriber ID and the initialisation of the MqttClient instance is almost the same as above subscriber code, except use -subDevice as a suffix for the Client_Id.

(v) Observing LWT message on disconnection/failure of communication of messages on topic sensed data and control data

A disconnection is feasible. A broker needs to detect disconnection by a client. If the clientDevice has set a subscription to a message for a topic (for example, SensedData), then when the device reconnects, the broker sends 'LWT' message string to the topic 'SensedData' on reconnection. Similarly, If the clientApplication has set a subscription to a message for a topic (for example, ControlData), then when application reconnects, the broker sends 'LWT' message string to the topic 'ControlData' on reconnection.

The following code implementation enables the client to get the LWT string and notify disconnection or required message to other clients, as per the requirement. LWT means Last Will and Testament; will means what is requested on connection death (failure).

```
MqttConnectOptions options = new MqttConnectOptions();
options.setCleanSession(false);
/* Set a will (sought message on a connection death (failure) consisting of two bytes on the reconnection after the disconnection. Here SensedData denotes the topic and LWT is sent as the message on that topic. */
/* SensedData observes the SensedData/LWT message to detect disconnection of SensedData client.*/
 options.setWill(clientDevice.getTopic("SensedData/LWT"),
" SensedData Lost".getBytes(), 2, true);
clientDevice.connect(options);
/* ControlData observes the ControlData/LWT message to detect disconnection of ControlData client message.*/
options.setWill(clientApp.getTopic("ControlData/LWT"),
" ControlData Lost".getBytes(), 2, true);
clientApp.connect(options);
/*********************************************************************/
```

(vi) Declaring threads for publishing the messages on the topic sensed data

A client publishes the sensed data. A step is declaring the threads for the topic and messages of each TOPIC which the client publishes from each streetlight device platform. A string for Topic/Message from each streetlight in each group of streetlights publishes the *light need*, *working status* and *traffic presence* values (Example 1.2 and Figure 9.2). Assume that *light need* value needs to communicate every 16 m, *working status* every day and *traffic presence* each minute. Following is the code for declaring the TOPIC strings, messages and threads.

```
/* Set topics and message strings */
public static final String TOPIC_LIGHTNEED = "SensedData/deviceID, groupID, lightneed";
 /* Assume that LightNeed = true is the message which need to communicate every 16 minutes whenever the ambient light condition demands that streetlight should be switched ON.*/
/* Create a thread for publishing lightneed every 16 m */
 public class LightNeed extends Thread {
    publishLightNeed();
    for ( int i=1, i=9600, i++){sleep (1000); } /* Wait 16 m*/
    }
  public static final String TOPIC_WORKINGSTATUS = "SensedData/deviceID, groupID, workingstatus";
/* Create a thread for publishing working status every day */
  public class WorkingStatus extends Thread {
    publishWorkingStatus ();
```

```
        for ( int i=1, i=24800, i++){sleep (30000); } /* Wait 1 day*/
           }
 public static final String TOPIC_TRAFFICPRESENCE = "SensedData/deviceid,
groupid, trafficpresence ";
 /* Create a thread for publishing TrafficPresence every 1m */
 public class TrafficPresence extends Thread {
    publishTrafficPresence ();
    sleep(30000); sleep(30000); /* Wait 1 m*/ .
    }
/*********************************************************************/
```

(vii) (a) Publishing the messages by clients at each streetlight on the topic sensed data

```
/*Next step is publishing of messages by the clients at each streetlight
device platform. Message string publishes for lightneed, workingstatus and
traffic-presence from each streetlight in each group of streetlights.*/
    private void publishLightNeed() throws MqttException {
   /* Declare two numbers for creating a random number*/
   final int n1= 100; final int n2=110;
   final MqttTopic lightneedTopic = client.getTopic(TOPIC_LIGHTNEED);
   final int lightneedNumber = Utils.createRandomNumberBetween(100, 110);
   final String lightneed = lightneedNumber + "Requesting ON";
   lightneedTopic.publish(new MqttMessage(lightneed.getBytes()));
   }
 private void publishWorkingStatus() throws MqttException {

 /* Write code as above for workingstatus */
 }
 private void publishTrafficPresence() throws MqttException {

 /* Write similar code for trafficpresence */
 }

/*************************************************************/
```

(b) Publishing the messages by application/central controller on the topic 'control data'

```
/*Next step is publishing of messages by the Application for each
streetlight device platform. Message string publishes for control message
for each streetlight in each group of streetlights.*/

  private void publishControlMessage() throws MqttException {
   /* Specify codes for sending controlmessage */
```

```
  final MqttTopiccontrolmessageTopic = client.getTopic(TOPIC_CONTROLDATA);
            controlmessageTopic.publish(new    MqttMessage(controlmessage.
getBytes()));
    }
/*************************************************************/
```

(c) Client application subscription of the messages on the topic sensed data

A coding step is codes for a client subscription, which is reading the values on the topic SensedData. Each client device sends on the topic SensedData publishes three messages to the broker.

"SensedData /lightneed" ,

"SensedData /workingstatus" and

"SensedData/trafficpresence"

A *MqttCallback* interface has three abstract methods:

1. messageArrived() implementation when runs then the Broker sends the new message to specific client device in a group of devices
2. deliveryComplete() implementation runs when message with specified QoS = 1 or 2 at the Broker which sends to publisher on successful delivery to subscriber, and
3. connectionLost() implementation runs when the connection is unexpectedly closed form the MQTT broker.

(d) Use a wildcard '#' instead of subscription to 3 different topics, namely, TOPIC_LIGHTNEED, TOPIC_WORKINGSTATUS, TOPIC_TRAFFICPRESENCE,

mqttClient.setCallback(new SubscribeCallback());

mqttClient.connect();

mqttClient.subscribe("SensedData/#");

mqttClient.subscribe("ControlData/#");

public class SubscribeCallback implements MqttCallback

{

@Override

[@Override means as follows: A compile will cause annotation checks of the method for the override. Override method means a method having the same name, same type and same number of arguments but has statements which use new arguments and override the earlier statements for that method. When annotated, a compile will cause an error if the method is not found in one of the parent classes or implemented interfaces.]

```
public void connectionLost(Throwable cause) {}
  @Override
  public void messageArrived(MqttTopic SensedData, MqttMessage deviceID,
groupID, lightneed) {
/* Write Codes for actions on receiving the lightneed*/
.
;}
```

```
public void messageArrived(MqttTopic SensedData, MqttMessage deviceID,
groupID, workingstatus) {
/* Write Codes for actions on receiving the workingstatus */
.
;}
public void messageArrived(MqttTopic SensedData, MqttMessage deviceID,
groupID, trafficpresence) {
/* Write Codes for actions on receiving the workingstatus */
.
;}
@Override
  public void deliveryComplete(MqttDeliveryToken token) {
 /* Write Codes for actions on delivery completion */
.
; }

ClientApplication subscription of the messages on the topic ControlData
     mqttClient.setCallback(new SubscribeCallback());
mqttClient.connect();
mqttClient.subscribe("ControlData/controlmessage");
public void messageArrived(MqttTopic ControlData, MqttMessage deviceID,
groupID, controlmessage);
{
/* Write Codes for actions on receiving the wcontrolmessage */
.
;}
 @Override
  public void deliveryComplete(MqttDeliveryToken token) {
 /* Write Codes for actions on delivery completion */
.
; }
```

Figure 9.3 shows the connected sensor and actuator which are publishing the sensed data and subscribing for the control data using MQTT Android clients' communication with application layer MQTT clients through an MQTT broker.

Example 9.12 gives application on Android MQTT client device platform using Paho.

Example 9.12

Problem

A programme implementation for communication with an Android phone using Paho and ADT:

Connected sensor and actuator, which are publishing the sensed data, and subscribing for control messages using MQTT Android clients

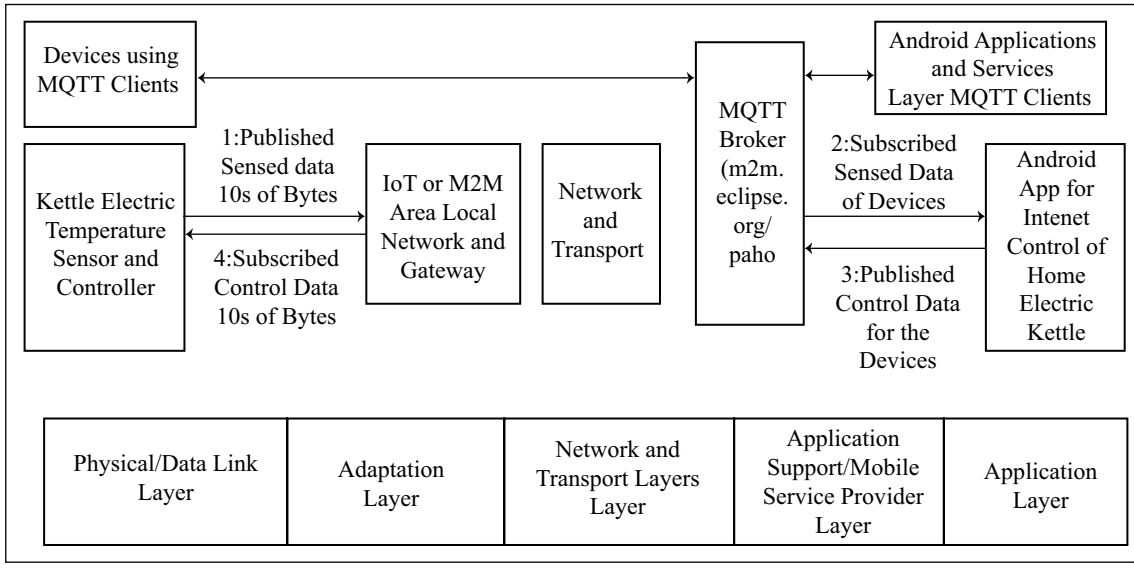


Figure 9.3 Connected sensor and actuator, which are publishing the sensed data, and subscribing for data control using MQTT Android clients' communication with the application layer MQTT clients through an MQTT broker

Implement codes for Subscriber class for subscribing to the sensed data (for example, electric kettle temperature sensor) by the Android phone. Use Eclipse Moquette Paho implementation of MQTT using TCP?

Solution

Assume that Java JDK version 6 or higher package is used for implementation in Paho. Also Android Software Development Kit (SDK) is installed and Android Development Toolkit (ADT) is added as plugin to the Eclipse.

Package org.eclipse.pahoAndroidApplicationExample;

import org.eclipses.paho.client.mqttv3.MqttClient;

import org.eclipses.paho.client.mqttv3.MqttException;

import org.eclipses.paho.client.mqttv3.MqttMessage;

Coding is similar to Example 9.11. Following code shows the analogy.

Implement the Subscriber class for subscribing to the sensed data (for example, electric kettle temperature sensor) by the Android phone as follows:

```
/*Let first step is that within the class Subscriber, declare URL of the MQTT
Broker and create instance of class MQTTClient. Use MAC Address as Client_ID
for publisher ID and the initialization of the MqttClient instance is as per
following code. Use -subAndroid as a suffix for the Client_Id. */
public class Subscriber {
public static final String MQTT_BROKER_URL = "tcp://broker.mqttdashboard.
com:1883";
```

```
/* client is the instance of MqttClient */
private MqttClient client;
 public Subscriber()
 {
String a1 = "-subAndroid";
/* Declare clientDevice_ID using method getPhoneNumber( ). */
String clientDevice_Id = Utils.getPhoneNumber () + a1;
try {
mqttClient = new MqttClient(MQTT_BROKER_URL, clientDevice_Id);
}
catch (MqttException e) {
/* Write codes for callback method */
/* On exeception, Assume that callback method is print the stack trace. */
e.printStackTrace();
System.exit(1);
     }
  }
}
```

Eclipse IoT Stack Applications for Projects

The Eclipse IoT stack includes application layer software (applications and services) for OM2M, smart home and SCADA projects. Table 9.3 gives Eclipse implementations and frameworks included in the Eclipse IoT stack for application layer software.

Table 9.3 Eclipse implementations and frameworks included in the Eclipse IoT stack for application layer software

Eclipse Implementation/ Framework	Description
Eclipse SmartHome	A set of implementations for projects for smart homes using wireless and wired protocols and network protocols
Eclipse SCADA	A set of SCADA (Supervisory Control and Data Acquisition) functions for usages, such as in factory automation, industrial processes buildings and health systems
Eclipse OM2M	An open source implementation of SmartM2M and oneM2M standard that enables developing services on a horizontal M2M service platform, the implementation is independently of the underlying network and facilitates the deployment of vertical applications and heterogeneous devices.

Reconfirm Your Understanding

- Needs of developing the software at five levels: (i) Gather + Consolidate, (ii) Connect, (iii) Collect + Assemble, (iv) Manage and Analyse, and (v) applications and services in IoT/M2M applications and services.
- Eclipse open IoT stack is a set of Java frameworks, protocols, development tools and implementations including usages of OSGi services, such as *Clock*, *Crypto* (AES, base64, SHA-1), *Geolocatio*n, *Data* and *Cloud* Services for the connect and manage functions for IoT solutions.
- Eclipse IoT stack is for open source technologies for device platforms and running the codes in JVM or Eclipse Concierge (a lightweight implementation of OSGi runtime).
- The stack enables the usages of protocol functions for connectivity and interoperability of device gateways.
- The stack enables the usages of lightweight M2M (OMA M2M standard), MQTT (OASIS IoT standard), CoAP (IETF IoT standard) and standard network protocols for IoT gateway services for remote management and applications management.
- Eclipse Pi4J, Eclipse Koneki, Eclipse Mihini, Eclipse Krikkit provide physical cum data-link and adaptation layers software.
- Eclipse Moquette, Eclipse Paho, Eclipse Wakkamma, Eclipse Californium, Eclipse Lehshan, Eclipse OM2M, Eclipse Ponte and Eclipse Kura provide gateway, network, transport, application support layer software.
- An example showed the usages of Paho for streetlights IoT where streetlight devices publish the sensed data using Paho Java MQTT clients. Applications publish control data and subscribe to the sensed data. An Eclipse Paho MQTT broker provides a gateway between the devices and applications.
- An example showed the usages of Paho for Android MQTT clients for controlling an electric kettle.
- Application layer implementations for smart home and SCADA projects and for M2M can use Eclipse SmartHome, Eclipse SCADA and Eclipse OM2M for software development.

Self-Assessment Exercise

1. List five levels of software which need to be developed. ★
2. Write the features of Eclipse IoT stack. ★★
3. Which are the Eclipse stacks for implementations using LWM2M and CoAP clients and LWM2M and CoAP servers in client-server communication modes? ★
4. What are the software for functions implemented using Eclipse Pi4J, Eclipse Koneki, Eclipse Mihini and Eclipse Krikkit? ★★
5. List the Eclipse stack implementations for devices and interactions with application clients through a broker. ★★
6. When is the LWM2M or CoAP client-server in communication mode? ★★
7. List the features of Eclipse Kura. ★★
8. What are for the OSGi service functions used in the implementation of Eclipse IoT stack? ★★
9. Write the subscriber and publisher classes for IoT/M2M applications using the pub/sub mode of communication. Use the MQTT device clients–broker–application clients. ★★★

9.4 PROTOTYPING ONLINE COMPONENT APIs AND WEB APIs

LO 9.3 Describe prototyping methods of the online components for IoT application APIs

Application Programming Interface (API) is a way of obtaining input from one face (user-end or a program) and sending the request or message or data to another face (an application program). An API initiates the message exchanges. The API initiates action when it receives inputs, it calls the execution of application codes (set of codes) on input events and initiate running of functions for actions [callback ( )] on them. The API initiates and enables desired actions using the computing system, and may also access a service which is local or remote on occurrence of event of input to the interface. The API can also initiate sessions between the user or client and the applications or service functions.

A web application uses UI and web API. A web API accesses a service on the web using the Internet and web protocols. A UI of a web API gets input from the user or client and accesses a service, such as MSN weather service using MSN weather server. Figure 9.4 shows mobile phone weather-application UIs, APIs and web API interactions and ten message-exchange sequences.

API enables implementation of an application or service on the event of input from one end

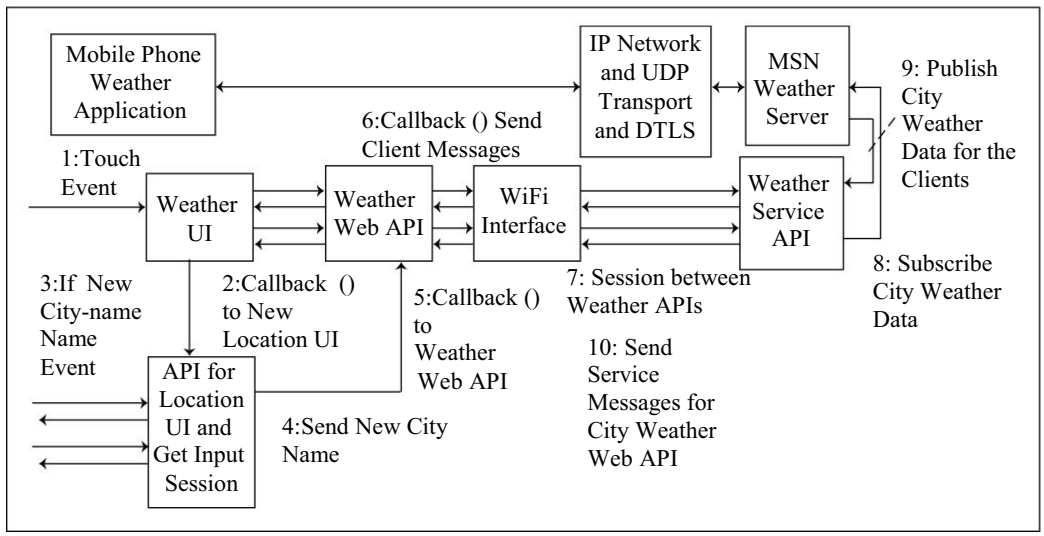


Figure 9.4 The mobile phone weather-application UIs, APIs and web API interactions, and ten message-exchange sequences

9.4.1 Using the Number of Mashed Up APIs in an Application or Service

A number of APIs can be mashed to get the desired result. Figure 9.4 shows number of mashed up APIs in the weather application.

Example 9.13 clarifies the concept of API, web API and mashing of APIs.

Example 9.13

Problem

The concept of API, web API and mashing of APIs:

How does a weather-application access the weather-service in a mobile phone?

Solution

The weather-application in a mobile phone uses a UI (a screen API) which displays onto screen to seek user input. The phone screen shows up a touch-option with label 'weather'. Following are the sequential actions:

- When user touches 'weather', the touch messages transfer to another UI (User Interface) which appears on the screen next (sequences 1 and 2 in the Figure 9.4).
- UI is user interface for location API. It displays a message "Welcome! Please choose your default location". The screen also displays a text box with entry 'Search for a city'. A cursor points at the beginning of the text box. The user enters the input into the textbox i.e. name of the city using a touch keypad, which also shows up at the bottom (Sequences 3 and 4 in the figure).
- When the location API gets the name of the city and the user taps at the textbox, the UI display disappears, the phone communicates with the MSN weather site. Weather service gets the messages and sends for display onto the 'weather' (Sequences 5 to 10).
- The 'weather' is a UI for display and hourly refresh. The API displays, "Mon, sign for fully Sunny, 30°, 18°; Tue, sign for small part sunny, rest cloudy, 30°, 19°, ...", and the city name shows up at the bottom line. If the weather UI is touched, then callback function initiates another API. The called API UI is full screen, which shows full details of the weather at the selected city. When user touches the UI screen, then the weather messages refresh.

The weather web application at the phone is using a number of APIs and callback functions.

- The screen UI gets input (touch) from the user. The UI sends in output weather tile touch message to the location API.
- The location API for weather gets input for the name of the city. When input is given then it callbacks a back-end code for weather access. The code accesses to the MSN weather site using the Internet over WiFi network to which the phone is connected. The location API sends the city name.
- Weather web-API callback functions interact with weather-API service for weather service messages. It subscribes to the MSN weather server. The server publishes weather messages for the subscribing clients (Sequences 5 to 7).
- Weather service API sends service messages as a response which communicate over the Internet to the weather client of weather web API. The service also sends in response presence and next two days predications, and expected maximum and minimum temperatures (Sequence 8 to 10).

Earlier Example 9.13 used mobile phones as the device platform. Refer to Example 1.1 of IoT. The example considered a smart umbrella as a device platform, which enables the umbrella, and behaves like living entity through computing. An embedded small device interacts with the weather web-service and mobile of the owner through the Internet. Figure 9.5 shows weather web-API for an umbrella, extreme weather-message web-API, mobile phone web-API, and weather-service API, and their interactions, and fourteen sequences of message exchanges.

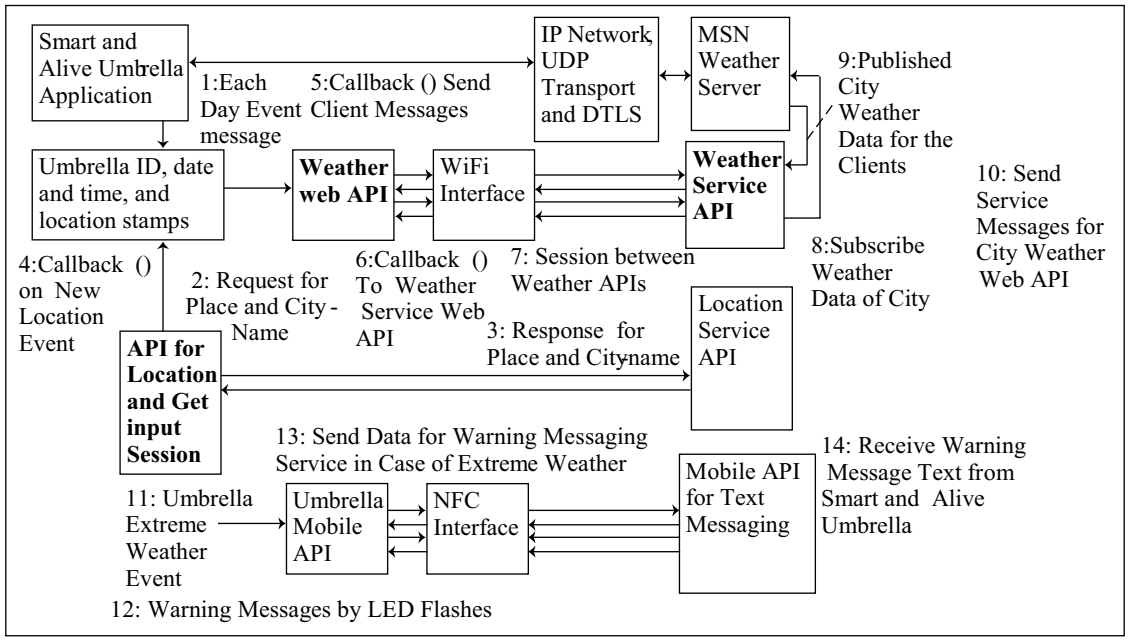


Figure 9.5 Umbrella weather web API, location API, extreme weather warning message umbrella mobile API, mobile phone text messaging API, weather service API and their interactions and fourteen sequences of message exchanges

Example 9.14 clarifies the concept of APIs and web APIs for Internet of Smart and Alive umbrella.

Example 9.14

Problem

The concept of APIs and web-APIs for Internet of Smart and Alive Umbrella:

How does a 'Weather' API at an umbrella access a weather service and communicate an expected extreme weather event, such as rain or hot sunny day to a mobile phone API by a text message?

Solution

Assume that an umbrella consists of an embedded device for an IoT weather application. The device application uses a mash of APIs as follows:

- When a start of the day event, say 8 a.m., ticks at the clock (Sequence 1 in Figure 9.5), a callback() function executes and sends a client-request to the location server (Sequence 2), and pulls the location data (sequence 3). Sequences 2 and 3 find the location and generate event message on location change.
- Location API sends message for new location in case of device-location change (Sequence 4).
- The messages communicate in JSON Object format (Section 3.2.3). The umbrella ID, date, time and location stamps communicate on the day-start event to the weather API. Sequences 5 to 10 are similar to that in Example 9.13 for mobile phone web APIs.
- An event, if generates an extreme weather warning, then a callback function executes. The warning can be by LED flashes at the device or by a text message on the mobile.

The weather web application at the umbrella is using number of APIs and callback functions. These are:

- Location API gets the name of the city from the location server. (Sequences 2 and 3).
- The weather web-API callback functions interact with the weather API for the weather service messages. The API subscribes to MSN or another weather Server. The server publishes weather messages for the subscribing clients (Sequence 5 to 8).
- Weather service API sends service messages as a response which communicates over the Internet to the weather-client of the web API. The weather service also communicates in response the present and next two days predications, and expected maximum and minimum temperatures (Sequences 9 and 10).
- Umbrella mobile web API sends message for owner on mobile using NFC protocol (Sequences 11 and 13 in the figure).

Implementing the APIs in an Application or Service

An application or service consists of mashed APIs. Following are the steps for implementing the APIs:

1. Make an implementation table for sequences of API actions. A session of message exchanges between two ends of API may consist of number of actions.
   - Column 1 of each row has the actions in sequences which occur one by one.
   - Column 2 may specify for each action, the authentication code or method (device platform ID, such as MAC address) for secured communication to and from other end (server or application). Column 2 has authentication method for use that may involve user/password communication. User in case of IoT device platform is device platform ID, such as MAC address. Password can be some code internally generated at device platform using some algorithm using a secret key as input (Example 9.8).
   - Column 3 specifies the API inputs for initiating the action on event.
   - Column 4 specifies the API outputs for the inputs. The outputs communicate to other end and initiate the execution of methods, callback functions, generate requests or send responses.
2. Use the secure communication protocols, such as DTLS/TLS.
3. Use the standard formats for object or message exchanges: JSON, TLV or XML or REST style of URIs or URLs for message format (Section 3.2).
4. Use for access to remote methods standard protocols, such as SOAP (for XML-RPC) or JSON-RPC. RPC stands for remote procedure call. Procedure also means *function* in C/C++ or method in Java (Sections 3.3 and 3.4).
5. Use the standard client-server protocols, such as MQTT when using publish/subscribe or such as XMPP models of message or object exchanges (Section 3.2).
6. Use the standard client and server protocol for client-server http, or CoAP or ws protocol using client-response model of message or object exchanges (ws:// used in place of http:// for bi-directional message exchanges) (Section 3.4).
7. Use the standard methods for web object or message exchanges. For example, REST request-response model methods, such as HTTP GET/POST methods or WebSocket methods for bi-direction message exchanges (Section 3.4).

8. Use the scripts, for example, JavaScript or Node.js framework for creating set of codes in event model or codes for running at distributed computing systems (device platforms or services at web/cloud). Event model calls a `callback()` `[onEventAction()]` functions on inputs. The function may initiate sending a client data or sending a server response. Function runs once on each input-event.
9. Use the language or scripting language framework which provides wider support of libraries, community and test tools.

Example 9.15 explains the table for easy implementation of API.

Example 9.15

Problem

Drawing an implementation table for weather web APIs:

How does a table make it easy writing codes for weather web APIs? Use the sequences shown in Figure 9.4.

Solution

Table 9.4 gives the actions, inputs and outputs for weather APIs.

Table 9.4 Implementation table for weather web APIs

Sequence Number and Action	Authentication	Inputs	Outputs
1: New connection to weather service API	MAC address or deviceID/ ApplicationID/ API_ID	Location (City Name)	Message to weather web client to enable client connect to send request for weather messages
2: New connection to weather Service	MAC address or weather service API ID	Messages (city name, ClientID, time stamp)	Subscribe to the weather service, if a new city name (messaged from the location server)
3: New connection to weather service	MAC address or weather service API ID	Messages of weather web-service API on new city name	Subscription message(s) on new city to the weather service and messages (clientID, weather information with with time stamp from messaged data)
3: New response of weather service	MAC address or weather service ID	Weather messages and warnings for this and predictions of next two days for the city from repository	Response message(s)

9.4.2 APIs Implementation Using REST Methods

When implementing a REST API or using REST methods in client or server, first making a table makes the coding easy.

- Column 1 of each row has the URL for communication for implementing an action.
- Column 2 may specify the action (REST methods), such as PUT/GET/POST/DELETE for implementation.
- Column 3 may specify for each action, the authentication code or method (device platform ID, such as MAC address) for secured communication to and from other end (server or Application). Column 3 has authentication method for use at server/destination end that may involve user/password session. User in case of IoT device platform is device platform ID, such as MAC address. Password can be some code internally generated at device platform using some algorithm using a secret key as input (Example 9.8).
- Column 4 specifies the API inputs/parameters for initiating the action.
- Column 5 specifies the API outputs for the inputs. The outputs communicate to other end and initiate the execution of methods, callback functions, generate Cookie or response.

9.4.3 API Implementation in WebSockets

WebSocket enables bi-directional communication at the same instances between two ends. The WebSocket needs lesser header-size when compared to the HTTP header size (Figure 3.9) HTTP client first pulls data from one end to another and later other end-clients in other direction (Section 3.4.5).

Eclipse Jetty is a WebSocket implementation in Java and is open source from a weblink https://www.eclipse.org/websocket. The implementation uses Jetty WebSocket (String). Maximum message size declares as WebSocket (maxTextMessageSize). WebSocket Client and server both use the callback listeners onOpen, onMessage, OnClose, onError. A Java implementation imports following:

```
import org.eclipse.jetty.websocket.client.WebSocketClient;
import org.eclipse.jetty.websocket.api.Session;
import org.eclipse.jetty.websocket.api.StatusCode;
import org.eclipse.jetty.websocket.api.annotations.OnWebSocketOpen;
import org.eclipse.jetty.websocket.api.annotations.OnWebSocketConnect;
import org.eclipse.jetty.websocket.api.annotations.OnWebSocketMessage;
import org.eclipse.jetty.websocket.api.annotations.OnWebSocketClose;
import org.eclipse.jetty.websocket.api.annotations.OnWebSocketOnError;
```

Packages, classes, methods, variables and parameters may be annotated in Java. An annotation in Java is a form of syntactic metadata that can be added to a Java source

code. Syntactic metadata refers to the format instructions for storage of the values or class or method.

Following are the callback WebSocket methods:

- onWebSocketConnect()
- onWebSocketMessage()
- onWebSocketClose()

Open source Paho Go Client at Eclipse Paho consists of added WebSocket (ws) clients to connect to an MQTT broker. Open source Eclipse Ponte consists of added MQTT-over-ws clients in JavaScript.

Open source client-server WebSocket methods for implementation on Arduino device platform can be used from the weblink https://github.com/krohling/ArduinoWebsocket and https://github.com/djsb/arduino-websocket. Krohling Arduino ws client and ws server (bi-directional) implementation is as follows:

- Declare MAC address as follows: byte mac48BitAddress (six bytes separated by comma)
- Declare WebSocket server URL as follows: char server [] = "echo.websocket.org" for initialization
- Declare serverSocket (8080) for using HTTP alternate web server for initial creation of WebSockets using echo. Echo means sending the message and receiving same message back when WebSockets successfully create.http web server port is 80. When a secondary or alternate web server hosted the port number 8080 is used. 8080 is commonly used for proxy and caching
- Declare object wsClient of class WebSocketClient
- Declare wsClientDA object of class dataArrivedWebSocketClient for web object of arrived data
- Declare callback method dataArrived (WebSocketClient wsClientDA, String message) on an end receiving string from WebSocket

The HTTP client sends the request or messages to the HTTP server and waits for a response. The server sends response or messages. A WebSocket client is that which initiates the connection to peer. A WebSocket server is published and waits for the connections from the peers.

Figure 9.6 shows the sequences of sessions for message exchanges when using WebSocket objects.

Following are methods wsclient.available(), wsclient.connect(Server), wsClient. connection(), wsClient.connected(), wsClient.send(), wsClient.setDataArrived() and wsClient.setDataArrivedDelegate(String dataArrived) for communication after creating WebSocket clients for bi-directional exchange of messages.

When implementing a WebSocket API or ws client or ws server, first making a table makes the coding easy. Following are the columns:

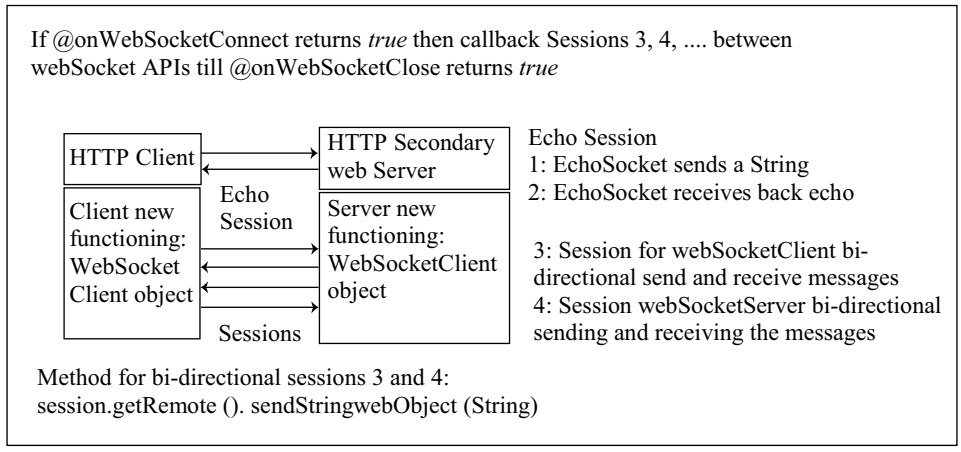


Figure 9.6 Sequences of sessions for message exchanges when using WebSockets

1. Column 1 of each row has the URI or URL of message destination end for communication for implementing an action.
2. Column 2 may specify the action such as following callback WebSocket method: onWebSocketOpen, OnWebSocketSession, onWebSocketConnect, onWebSocketMessage String Message, Session session), onWebSocketClose or following method WebSocket (maxTextMessageSize) when using Eclipse Jetty WebSocket implementations.
3. Column 3 may specify for each action, the authentication code or method (device platform ID, such as MAC address) for secured communication from source end to destination end. Column 3 has authentication method for use at the destination-end (server or application/service) that may involve user/password communication. User in case of IoT device platform is device platform ID, such as MAC address. Password can be some code internally generated at device platform using some algorithm using a secret key as input (Example 9.8).
4. Column 4 specifies the API inputs/parameters for initiating the action.
5. Column 5 specifies the API outputs for the inputs. The outputs communicate to other end and initiate the execution of methods, callback functions, generate cookie or response.

Example 9.16 explains a WebSocket APIs implementation table method.

Example 9.16

Problem

Implementation table for the weather and weather service APIs using Eclipse webSocketClient library:

How is implementation table drawn for ease of writing codes for weather and weather service APIs? Use the library at Eclipse website.[22]

[22] http:// jetty.websocket.client.WebSocketClient.eclipse.org

Solution

Table 9.5 gives the actions, inputs and outputs for weather and weather service APIs.

Table 9.5 Implementation table drawn for ease of writing codes for a weather and weather service APIs

Sequence number and URL	Method	Authentication	Inputs	Outputs
1: URL of weather service API	onWebSocketConnect()	MAC address or deviceID/ ApplicationID/ API_ID	—	Creation of WebSockets
2: URL weather service	onWebSocketMessage()	MAC address or weather service API ID	callback(), Messages (city name, ClientID, time stamp)	Request to weather service
3: URL of weather service API	onWebSocketMessage()	MAC address or weather service ID	callback(), Messages of weather web-service	Response of service for messages (clientID, weather information with time stamp)
4: URL of weather Web API	onWebSocketMessage()	MAC address or weather service API ID		Weather messages and warnings for this and predictions of next two days for the city

Reconfirm Your Understanding

- An API is a way of obtaining input from one face and sending the request/message/data to another face (other set of codes). The API enables initiations of events, obtaining inputs, execution of application codes (set of codes) on input events and running the functions (callback functions) on input event. Web API is an API which uses web protocols for web applications.
- Example of weather application in a mobile phone and the smart and alive umbrella shows the uses of APIs and web APIs.
- Example of smart and alive umbrella shows the uses of APIs and web APIs.
- APIs implementation using standard message or object formats, protocols and models.
- APIs implementation using standard message format, protocols, language or script framework with wider library, community and test tools.
- API implementation is easy when a table is made for each action of the sessions and interactions, specifying the authentication methods, inputs and outputs.

- Message exchanges between APIs, webAPIs, webService APIs, webClient and webServer use REST APIs and WebSockets.
- Eclipse Jetty, Paho and Ponte implementation provide the library functions for WebSockets and WebSocket APIs.

Self-Assessment Exercise

1. Define the functions of API. ★
2. Why are APIs used for communicating with application/service at both server and client ends? ★★
3. Draw a diagram showing sessions between location API and location service API. ★
4. Draw a table of actions and sessions for implementing the APIs in example of Internet of Smart and Alive Umbrella. ★★★

Key Concepts

- Application
- Application programming interface
- Authentication
- Callback functions
- Client
- CoAP
- Crypto functions
- Development framework
- Distribution
- Eclipse IoT stack
- Embedded devices
- Encryption
- Event model
- Gateway methods
- Hash algorithm
- Integrated development environment
- IoT stack
- Kura
- Library functions
- LWM2M
- MAC address
- Mashed up APIs
- MQTT broker
- MQTT client
- OM2M
- OSGi services
- Paho
- Ponte
- REST methods
- Server
- Software serial library
- Service
- Test functions
- WebSockets

Learning Outcomes

LO 9.1

- A number of device platforms of IDEs, such as Arduino, Intel Galileo, RPi, BB and mBed, provide development tools, libraries and framework, and are used for the development of embedded Software.

- A number of programming examples explain the process of developing the codes and test codes.
- Arduino IDE is simpler and embedded software requires building upon just two functions: setup() and loop(). Bootloader enables simple multitasking by the usages of interrupt handlers.
- A serial monitor at the IDE enables computer obtaining results from the serial port of device platform for display on monitor.
- Number of library functions provides the ease in programming. For example, usage of timer and software serial libraries.
- Examples explained the usages of: (i) timer functions, (ii) reading of RFID tag using UART functions and (iii) I2C communication between sensor communicating sensed data using I2C and device I2C port.
- An example showed how the multiple threads of a program execute on a device platform.
- Things communicate to the Internet using functions for Ethernet, WiFi and IP.
- Things communicate data securely using cryptolibrary functions, such as AES128, AES192, AES256 or DES.

LO 9.2

- Eclipse IoT stack implementations enable the software components, development device, gateway, Internet and web/cloud application service.
- The development of software components takes place at five levels: (i) Gather + Consolidate, (ii) Connect, (iii) Collect + Assemble, (iv) Manage and Analyse, and (v) Applications and Services in IoT/M2M applications and services using Eclipse open IoT stack.
- Eclipse IoT stack of the device platforms is for open source technologies, usages of protocol functions, connectivity and interoperability of the device gateway, lightweight M2M (OMA M2M standard), MQTT (OASIS IoT standard), CoAP (IETF IoT standard), IoT gateway services, remote management and applications management.
- Eclipse IoT stack implementations use the Eclipse Pi4J, Eclipse Koneki, Eclipse Mihini, Eclipse Krikkit, Eclipse Moquette, Eclipse Paho, Eclipse Wakkamma, Eclipse Californium, Eclipse Lehshan, Eclipse OM2M, Eclipse Ponte and Eclipse Kura provide gateway, network, transport, application-support layers' software components.
- A number of Java programming examples explain the process of usage of Paho MQTT clients publishing the messages and subscribing for the messages through an MQTT Broker. Eclipse Paho MQTT broker provides a gateway between the devices and applications.
- Paho enables usage of Android MQTT clients for controlling the devices through the Android phone clients.
- Eclipse stack IoT and M2M solution frameworks for application layer implementations enable applications software components development for smart home, SCADA and OM2M projects.

LO 9.3

- APIs and web APIs enable message exchanges between embedded devices and applications/ services at the server/cloud.
- API obtains input from one face and sends the request or message or data to another face (other set of codes). The API enables initiations, obtaining inputs, execution of application codes (set of codes) on input events and running functions (callback functions) on input event. Web API is an API which uses web protocols.
- A number of examples explained the implementations using standard message or object formats, protocols and models.

- Way of message exchanges between APIs, webAPIs, webService APIs, web client and web server explained with usages of the REST APIs and WebSockets.
- Eclipse Jetty, Paho and Ponte implementation provide the library functions for WebSockets and WebSocket APIs.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. The levels for developing the software are: (i) Gather + Consolidate, (ii) Connect, (iii) Compacting, (iv) Manage and Analyse, and (v) Querying databases of devices data by the applications and services in IoT/M2M applications and services. ★
   (a) (i) to (v) except (iii)
   (b) (i) to (v)
   (c) (i) to (iv)
   (d) All except (iii) and (v)
2. The Arduino IDE includes: (i) C/C++ library, (ii) Java Library, (iii) JavaScript, (iv) Wiring library, (v) Processing language, (vi) setup() and loop() functions, (vii) Arduino device platform has bootloader embedded into that and usages of the OS is not a must and may add OS Linux distributions. ★★
   (a) All except (ii) and (iii)
   (b) (i) to (vii)
   (c) (i) to (vi)
   (d) (i) to (v) except (iii)
3. Which of the following statement is correct? (i) Pins, Tx or TxD (for serial transmission of header, data and other bits) and Rx or RxD (for serial reception of data) are used for I2C bus communication, (ii) Pins SDA and SCL are used for the UART bus communication, (iii) RFID tag communicates to the reader using UART, (iv) UART BAUD rate is always 9600, (v) I2C bus is asynchronous bus, (vi) Two Arduino boards can communicate with each other using I2C. ★★★
   (a) All except (ii) and (iii)
   (b) (i) to (vi)
   (c) (iii) and (vi)
   (d) (i) to (v) except (iii)
4. AdafruitIDE is for Raspberry Pi and can be updated for BeagleBone, (ii) RPi connects to LAN when coding, (iii) Coding can be using Python, JavaScript or Ruby languages, (iv) Adafruit also OS for functions of keyboard and display monitor, (v) Includes a secure shall (SSH) for cryptographic network protocol for the Application layer, (vi) enables remote login, and (vii) remote connections of the systems, keyboard and monitor. ★
   (a) All except (ii) and (iii)
   (b) (i) to (vii)

(c) All except (v)
(d) (i) to (vi)

5. IoT security risks are: (i) an in-between switch or router or server S reads messages between two ends and replays the same for authentication at a communicating end for obtaining access to communication between two ends, (ii) S modifies the data of one end and other end can wrongly authenticate the application or get the modified text, (iii) *Two requirements* for IoD data secure communication are authentication and encryption of data. AES128, AES192, AES256 or DES functions are used for (iv) encryption and (v) authentication. ★
   (a) (ii) to (iv)
   (b) (i) to (iii)
   (c) (i) to (iv)
   (d) All except (v)
6. The features of Eclipse IoT stack components and frameworks for the IoT solutions are: (i) Provides open source and specifications, (ii) Specifications as per OSGi, (iii) Simpler open source implementations and program development using the open source Java frameworks and services, (iv) C implementation in Paho and Californium, (v) Python and Java Script implementations. The stack enables usages of (v) LWM2M and MQTT, (vii) CoAP, (viii) TCP, (ix) UDP, (x) DTLS, and (xi) IoT gateway services for remote management and applications management. ★★★
   (a) (ii) to (xi)
   (b) (i) to (viii)
   (c) All except (iv) and (v)
   (d) All except (xi)
7. An application programming interface: (i) obtains from one end and sending the request or message or data to another end API, or (ii) application or service end or (iii) database server end, (iv) Broker end, (v) web server end, (vi) The API enables initiations on obtaining inputs, call for execution of application codes (set of codes) on input events and initiate running the functions (callback functions) on input event. ★
   (a) All
   (b) All except (ii) to (iv)
   (c) All except (ii) and (iv)
   (d) All except (iv)
8. The API (i) initiates and (ii) enables desired actions using the computing system, (iii) may also access a service which is local, or (iv) remote on the event of input to the interface. The API can have (v) initiate sessions between user or client, and (vi) the application functions, or (vii) service functions. ★
   (a) (ii) to (vi)
   (b) (i) to (viii)
   (c) All except (iv)
   (d) (i) to (v)
9. Which of the following statement is correct? (i) HTTP enables one-direction communication at an instance, client end to server end or server end to client end, (ii) HTTPS enables secure communication at an instance, (iii) WebSocket connection can create first by using an HTTP session, (iv) WebSockets are used for chat room ★★★

APIs, (v) WebSocket communicate larger header than http, and (vi) WebSockets are real time communication only and cannot be used for pulling responses on requests.
(a) (i) to (vi)
(b) (i) to (iv)
(c) All except (iv) and (v)
(d) All except (iii)

10. Which of the following statement is correct? (i) REST methods are PUT/GET/POST/DELETE implementation in API using REST style communication architecture, (ii) REST APIs use the URL for access to a resource, (iii) Callback(), (iv) object method() (v) C functions WebSocket methods are onWebSocketConnect(), onWebSocketMessage(), and onWebSocketClose(). ★
(a) (i) to (iv)
(b) (i), (ii), (iv) and (v)
(c) (i) to (iii)
(d) All except (iv) and (v)

Short-Answer Questions

1. What are the five levels in software development for IoT applications and services? **[LO 9.1]** ★
2. Why is it necessary to write the test functions? Why are debugger and emulators used? **[LO 9.1]** ★★
3. List and write usages of library functions for sensing the ADC readings given in the automobile embedded devices Example 9.3. **[LO 9.1]** ★★★
4. What are the UART port library functions needed for communicating the IDs of RFIDs ? **[LO 9.1]** ★
5. When is UART and I2C communication used? **[LO 9.1]** ★★
6. What are the IDEs for Raspberry Pi? **[LO 9.1]** ★
7. What are the security risks taken care of when: (i) using a secret key and its secure communication, (ii) using encryption and decryption functions, such as, AES128, AES192, AES256 or DES? **[LO 9.1]** ★★★
8. List the Ethernet and Wi-Fi functions for IoT systems. **[LO 9.1]** ★★
9. When do you use CoAP, MQTT and HTTP client? **[LO 9.2]** ★
10. Why are the device_ID, deviceGroupID and Application_ID needed, and time stamped data-communication needed for sending data to an application or service? **[LO 9.2]** ★★
11. When do you use MQTT clients and MQTT broker, and CoAP clients and servers for communicating the devices data? **[LO 9.2]** ★★★
12. What are the benefits obtained on using the Eclipse sandbox servers? **[LO 9.2]** ★
13. What are the differences in usages of Eclipse Paho and Ponte for IoT gateway message-exchanges between device clients and the application or service? **[LO 9.2]** ★★★
14. Why is MAC address usable for authenticating a device platform at the application/service APIs? **[LO 9.3]** ★
15. List the steps for message-exchanges using APIs. **[LO 9.3]** ★★
16. Explain usages of HTTP client, HTTP server, HTTP secondary server, HTTPS client-server, WebSocket Client and WebSocket server. **[LO 9.3]** ★★★

Review Questions

1. What are the features of Arduino IDE that enable the programming tasks simpler at Arduino platform? **[LO 9.1]** ★
2. How are the pins programmed for digital IO and UART serial IOs at Arduino platform? **[LO 9.1]** ★
3. What are the features of IDE for Intel Galileo devices in addition to compatibility with Arduino? **[LO9.1]** ★
4. Describe the IDEs for Raspberry Pi and BeagleBone? **[LO 9.1]** ★★
5. What are different software components of Eclipse IoT stack for developing software for embedded devices? **[LO9.2]** ★★
6. What are the different software components of Eclipse IoT stack for developing connectivity software? Explain the uses and functions of LWM2M, CoAP and MQTT protocols with diagrams. **[LO9.2]** ★★★
7. What do you mean by IoT gateway services for remote applications management? What are the functions of Eclipse Kura? **[LO9.2]** ★★
8. What are the different software frameworks of Eclipse IoT stack for developing IoT solutions? **[LO9.2]** ★★
9. Why are APIs used? Describe the ways of implementing APIs for device platform clients and a server for application/service ends. **[LO 9.3]** ★★★
10. When are the REST APIs used? When are WebSockets between the device platform and web APIs used? Describe differences in APIs for REST and WebSocket functions. **[LO 9.3]** ★★★

Practice Exercises

1. An embedded device platform is Arduino Uno. Write a program to drive a moving display of 24 LEDs, which are arranged in the shape of a garland. How will the four outputs at port pins be multiplexed and programmed? **[LO9.1]** ★
2. Sensor data readings are the input to an ADC. The ADC coverts the analog input of sensor to 10-bits output. The output converts to a serial input using a parallel to serial converter. The serial input is read by the Uno serial port, which receives data input at 0.8333 ms intervals. Write a program for reading of sensor data at a serial input in Arduino Uno. **[LO9.1]** ★★
3. List the functions needed in a timer library. Show how will you use the library to call a callback function after successive intervals of 2 s each. **[LO9.1]** ★
4. An Arduino board has SDA bits at pin A4 and SCL bits at pin A5. How will the software serial library enable the serial readings from 8 temperature sensor circuits on an I2C bus? Each sensor has addresses starting from 0x10 to 0x17. **[LO9.1]**. ★★★
5. Write the sequences of bits and bytes when using an UART interface, when it communicates an ID of an RFID consisting of 10-characters. **[LO 9.1]** ★
6. List the steps for secure communication of an RFID tags ID. **[LO 9.1]** ★★
7. Write a program for running four threads at successive intervals of 16.666 ms. Use the multithreading functions of an OS. **[LO9.1]** ★★
8. Draw a diagram showing how the connected streetlights send the sensed data. The clients at streetlights device platform are using CoAP protocol when they communicate with a CoAP server. The server sends data to applications at a central ★★★

controller. Californium is a Java implementation of CoAP including DTLS. How can Eclipse Californium be used for IoT security using DTLS? **[LO9.2]**

9. Draw a diagram showing the sequence and functions for the devices data in automobiles. The data communicates using the Internet to a central automotive service (Example 5.2). Use Eclipse IoT stack implementations. **[LO 9.2]** ★
10. Draw a functional diagram for streetlights using APIs for IoT sensors data and Control_ Application. Show the interactions between the two sets of APIs. Also show the sequences of message-exchanges. **[LO 9.3]** ★★
11. How is a query sent and responded by an API at a database server? **[LO9.3]** ★
12. How is data of a group of sensors aggregated at a gateway and communicated using an HTTP client and HTTP server? The data is to be added with time stamps, groupIDs and sensorIDs. **[LO9.3]** ★★★

10

IoT Privacy, Security and Vulnerabilities Solutions

Learning Objectives

LO 10.1 Explain the requirements of privacy and security, vulnerabilities from threats, and need of threat-analysis in IoT

LO 10.2 Familiarise with modelling using use cases and misuse cases

LO 10.3 Outline security tomography of large networks and layered attacker model

LO 10.4 Examine functions for source identity-management, identity-establishment, device messages access-control, message-integrity, message non-repudiation and availability in IoT applications and services

LO 10.5 Describe security model, security profiles and security protocols for IoT

Recall from Previous Chapters

Chapter 1 introduced IEEE P2413 standard which provisions for reference architecture that builds upon a reference model. The reference architecture covers the definition of basic architectural building blocks and their integration capability into multitiered systems. The framework includes the provisions for the quality 'quadruple' trust that includes protection, security, privacy and safety.

The chapter also introduced communication modules for IoT that include a cryptographic protocol TLS or DTLS for secure Internet communication, and DeviceHive for M2M platforms and their integration that includes web-based management software which creates security rules-based networks and monitors the devices.

Chapter 2 introduced data privacy and security steps. For example, a device data can be encrypted at physical (device) layer for security on the network. The devices can opt for link-level as well as service-level security, just the service level or unsecured level. Example of service level is use of TLSv1.2 public key (RSA and ECC) and PSK cipher suite for providing the transport level security end-to-end security protocols. End-to-end means application layer to physical layer. The chapter also introduced usage of Wireless Protected Access (WPA) and Wired Equivalent Privacy (WEP) security sublayer because wireless communication needs *security*, *integrity* and *reliability*.

Chapter 3 described features of DTLS. The DTLS has provisions for three types of security services, viz. *integrity*, *authentication* and *confidentiality*. LWM2M OMA specification for device gateway to Internet has provisions for M2M Authentication Server (MAS) for security, root key data store, and devices and data authentication. XMPP protocol includes layer Simple Authentication and Security Layer (SASL) for security.

Chapter 4 introduced IPv6 at IP layer which includes headers for managing device mobility, security and configuration aspects. IPv6 uses AES-128 security = 21 B header.

Chapter 5 introduced data centres provision for data security and protection using advanced tools, full data backups along with data recovery and redundant data communication connections, servers and data centres high security standards high degree of security and integrity and effective protection of data, files and databases at the organisation.

Chapter 6 introduced data security at cloud servers in a multi-tenant environment. Data needs high trust and low risks, and security from the loss of users' control. Security provisions as per the developer requirements are the responsibilities of cloud service providers. Nimbits cloud servers for the device provisions usage of security tokens and provisions data integrity.

Chapter 7 introduced sensor level security and participatory sensing challenges of security, privacy and reputation. Section 7.6 described security challenges in RFID usages. RFID IDs need provisioning for full implementation of privacy and security needs and data processing at the tag and reader with access encryption and authentication algorithms. A security issue in an RFID network is a vulnerability to external virus attacks. Section 7.6.2 described methods of securing communication for WSNs, including usages of Berkeley laboratories suggested SPINS Symmetric Cryptographic Protocols [Secure Network Encryption Protocol (SNEP), and micro-Tesla (μ-Tesla)].

Chapter 8 introduced: (i) Arduino using WiShield Library[1] preinstalled with the IDE for Wi-Fi interface security; (ii) RPi IDE security interfaces; (iii) mBed device platform; and (iv) REST APIs for administration, security, data flow, device, multitenancy, authentication, directory and subscription management.

Chapter 9 introduced security of data for embedded device platforms. The chapter described the authentication process and uses of result of the hash algorithmic functions or message digest operation. The chapter gave examples of using crypto-library functions in the cryptosuite.[2]

Chapter 9 also described encryption process using AES or DES algorithms for encryption. The key length can be 128, 192 or 256 bits. Data encryption can use AES256 (Advanced Encryption

[1] http://github.com/asynlabs/WiSield
[2] http://code.google.com/p/cryptosuite

Standard 256-bit encryption key length) or DES (Data Encryption Standard). Arduino AES 256 library can be downloaded and used for Arduino boards. Authentication can be by using SHA1/SHA256 secured hash algorithms or MD5 message digest algorithm. Eclipse Californium has DTLS security implementations.

Previous chapters described security protocols and standard implementations at device to application and service levels. Security, privacy, authentication, encryption and protection are must for end-to-end solutions from device level to network and application/service levels.

10.1 INTRODUCTION

International organisations are making a number of efforts towards ensuring that IoT design must ensure *trust*, data *security* and *privacy*.

Design must ensure trust, data security and privacy in IoT

Trust is important. For example, consider the messages and video clips of the operations from the ATMs to server. A user places the trust in the bank that sensitive information will not be disclosed which can harm the user. When things communicate in an analogous manner, then trust of safe use of data exists. When one tweets on Twitter, trust must exist that no one will harm him/her due to his tweets. Tweets expose ones' behaviour, thinking about the actions, inactions and policies of government officials and functionaries. Trust in IoT context means dependability, accuracy, quality of data from multiple sources for the intended applications and services.

An organisation, Open Trust Alliance[3] established IoT Trustworthy Group (ITWG) for recognising the priority (must in designs for security and privacy) from the onset of product development and addressing holistically.

An organization, 'I am the Cavalry'[4] has prescribed an Oath for connected medical devices on the lines similar to Hippocratic Oath which is attestation by physicians to provide care in the best interest of patients. The medical devices should have commitments to capabilities that preserve patient safety, as well as trust in the process of delivering care itself.

Security is important. For example, consider the ATM messages. They should communicate on Internet securely. The security distortions can lead to serious consequences.

A smart city security is also important. The city deploys smart health, public safety, transport, and deploys IoT and smart home applications and services. An organization[5] has taken initiative for solving cyber security problems in smart cities.

Privacy is important. The video clips communicate on the Internet in a smart home security application. If the clips reach unrelated entities, it can lead to serious breach of home security.

[3] https://www.otalliance.org
[4] https:www.iamthecavalry.org
[5] http://securingsmartcities.org

Industrial Internet Consortium[6] (IIC) was formed by Intel, IBM, CISCO, GE and AT&T (2014) for coordinating efforts and initiatives to connect and integrate objects. Another organisation, AllSeen Alliance[7] established by Linux Foundation is a collaborative project of cross-industry consortium which is dedicated for enabling the interoperability of billions of IoT devices and applications/services.

Following are the terms, whose meanings must be understood before learning the topics covered in this chapter.

Message is a string that represents data or client-request or server-response which communicates between sender and receiver objects.

Hash refers to a collection or bundle which gives an irreversible result after many operations on data and the operations are just one way. For example, when wheat crop is ripe and cut, hashing process separates out the grains which are used for consumption and the resultant waste is discarded. When data such as user ID and password needs secret communication for the purpose of authentication, then it is communicated after a set of standard operations on the usage of an algorithm, called secure hash algorithm. The algorithm generates a fixed size, say, 128, or 256-bit value using a secret key. Only the hash value communicates. Receiver-end retrieves the hash value, and compares that with stored hash value. If both are equal then the sender message is authenticated.

Digest is a process which gives the irreversible result involving many operations. A standard algorithm called MD5 (Message Digest 5) is also used for digest, similar to the hash value. Receiver-end stores the digest value expected to be obtained after the MD5 operations, and compares that with received value. If both are equal then the sender message is authenticated.

Encryption is a process of generating new data using a secret key known only to a receiver. Before sending the encrypted data, the sender and receiver, both identify each other and know the key that will be used by them. The encryption uses a 128, 192 or 256-bit key for encrypting the data.

Decryption is a process which retrieves the data from the encrypted data.

Use Case means a list of event steps or actions which define the interactions between two ends, in which one is playing the role and other is the system. The steps accomplish a task or goal or mission. One end is called *actor* in Unified Modelling Language (UML), while other end is the *system*. Use Case is a software engineering term. For example, an API is playing the role of obtaining inputs (events) and generation of outputs which interact with the system such as a web server or web API or service or web application using a callback() function as per the output (Sections 9.2 and 9.4). Use cases define the required behaviour of software under development. Use cases describe the details of usages of software and its normal behaviour.

Misuse Case can be understood as reverse sense of Use case. Misuse case defines the behaviour which is not required from the software under development. A Misuse case

[6] https://www.iiconsortium.org
[7] https://www.allseenalliance.org

defines the behaviour which should not happen. This in turn specifies the threats also. Misuse case gives information and renders help in identifying the requirement of new Use cases for prevention of attack and find out what should not happen.

Layer means a stage during a set of actions at which the action is taken as per the specific protocol or method and then the result passes to the next layer until the set of actions completes (Figures 2.1 to 2.3). A design using the layers' model enables representation of a set of systematic actions which are followed sequentially for accomplishing a task.

Sublayer is a layer consisting of various sublayers in a model to provide set of actions sequentially taking place at the layer.

Firewall is a software interface, which interconnects networks with differing trusts, and is immune to penetration and provides perimeter defence. It functions as a choke point for controlling and monitoring. It does auditing and provides controlled accesses. It allows only authorised traffic and imposes restrictions on network services. It can raise alarms for abnormal behaviours.

Following sections describe the processes and actions for ensuring security and privacy that are taken at the device, network, transport, application and service levels. Section 10.2 describes IoT vulnerabilities, security requirements and method of analysis of the threats to privacy and security. Section 10.3 describes Use cases and Misuse case methods. Section 10.4 describes the security tomography uses in large networks, such as RFID networks and WSNs. This section also describes layer model of attacks. Section 10.5 describes how things (embedded devices, sensor or actuators) establish their identity before the messages are used by them after set of message exchanges on the Internet. This section also describes practical aspects of access control, message integrity, non-repudiation and availability. Section 10.6 describes security model for IoTs.

10.2 VULNERABILITIES, SECURITY REQUIREMENTS AND THREAT ANALYSIS

LO 10.1 Explain the requirements of privacy and security, vulnerabilities from threats and threat analysis in IoT

10.2.1 Privacy

Message privacy means that the message should not reach into the hands of the unrelated entities. When data or messages communicate from the things (device platforms), those are meant only for the applications or services and for targeted goals only.

Privacy also means no interference or disturbance from other. Consider an example of messages from embedded devices in an automobile using the Internet to an automobile service centre (Example 5.2). Privacy means the messages reach only the centre and used only by the services of the centre. Another automobile company on whose hands the data falls may face serious business consequences.

IoT necessarily need privacy policy. A privacy policy needs to determine that 'how much of the IoT devices data and which data need absolute privacy and which need limited privacy'. Company authorities need the support for accessing the data which may be private for individuals. The authorities also need to respect the individual customer needs of privacy and understand that privacy is a legitimate human need. Privacy policy vendors should take privacy seriously. They must respect their customers enough to understand that the privacy is a legitimate human need.

National Institute of Standards and Technology (NIST), USA is developing the standards for privacy. A system may be secure but may inadvertently breach the privacy of an individual. A tracking service may track a vehicle while does not want his/her movements to be tracked.

Security authorities and agencies need support for accessing data which may be private for individuals. The authorities also need to respect the individual's needs.

Privacy standards for IoT data are under development. A community organisation is 'I am The Cavalry'.[8] 'Cavalry' means a group of soldiers who used horses in earlier centuries or formed part of army that used fast armoured vehicles. The Cavalry focuses on computer security issues where these interact with public safety, such as medical devices, automobiles, public infrastructure and home electronics. The organisation strives to ensure that these technologies are worthy of the trust.

IoT system vulnerabilities are numerous, due to participation of a number of layers, and hardware and software components

10.2.2 Vulnerabilities of IoT

Vulnerability means weak without complete protection, weakness to defend oneself or can be easily influenced from surrounding unwanted things from itself. An IoT security article describes that there are many vulnerabilities, due to participation of the number of layers, hardware sublayers and software in applications and services.[9] The nature of IoT also varies. For example, sensors, machines, automobiles, wearables, and so on. Each faces different kind of vulnerabilities and has complex security and privacy issues.

IoT network can be vulnerable to eavesdropping. Eavesdropper creates security issues. An eavesdropper, say *E*, listens to the messages and commands in the network during communication and obtains confidential messages. A server at *E* sends fake commands which a server *S* for the devices data assumes that are from the devices or applications. *S* issues responses for the device operations in response to requests from *E*. *E* listens these responses. A fake device at *E* can be used to send the device data, such as sensor data, requests and commands from *E* for disrupting the control system. Use of secret key encryption can protect the messages to and from device, server, application or service.

[8] https:www.iamthecavalry.org
[9] http://www.networkcomputing.com/internet-things/iot-security-privacy-reducing-vulnerabilities/807681850

The key is a device software-generated string which can be cracked by trying large number of combinations. Device unique ID and authentication issues exist in negligible user interaction scenario.

Security features needs to be incorporated in a standard format recommended for IoT. For example, a standard for electronic products architecture is from a developing group, EPCglobal. The group is responsible for creation and maintenance of privacy policy products.

Open Web Application Security Project[10] (OWASP) has undertaken the associated security issues of IoT for the purpose of helping developers, manufacturers and consumers. OWASP is open source and has free to use licensing policy. Project is community-model-based software development initiative. A community model is making collective efforts and initiative by universities, organisations and institutions in an open source project. OWASP has undertaken a number of security related subprojects, such as ones for defining, the 'Top Vulnerabilities', 'Attack Surface Areas' and Testing Guides'.

OWASP has identified top ten vulnerabilities in IoT applications/services as follows:

- Insecure web interface
- Insufficient authentication or authorisation
- Insecure network services
- Lack of transport encryption/integrity verification
- Privacy concerns
- Insecure cloud interface
- Insecure mobile interface
- Insufficient security configurability
- Insecure software or firmware
- Poor physical security

Reader can refer to the OWASP project definitions of IoT attack surface areas. Attack surface areas mean the areas in software or hardware which are vulnerable to attack (Example 10.1).

Example 10.1

Problem

What are the attack surface areas in Device Web Interface (DWI) and Cloud Web Interface (CWI) defined in OWASP?

Solution

Following are attack surface areas for DWI: SQL injection, cross-site scripting, cross-site request forgery, account lock out, username enumeration, weak passwords and known default credentials.

Following are attack surface areas for CWI: SQL injection, cross-site scripting, cross-site request forgery, account lock out, username enumeration, weak passwords and known default credentials, same as ones for DWIs plus transport encryption, encrypted personally identifiable information (PII) sent, unencrypted PII sent, device information leaked and location leaked and cloud user data disclosure, user/device location disclosure and differential privacy.

[10] https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

CERT Coordination Centre (CERT/CC) at Carnegie Mellon University (CMU) has taken the initiative for coordinating the vulnerabilities in IoT devices.[11] CERT stands for Computer Emergency Response Team at CMU Software Engineering Institute. The team researches on software bugs that impact software and Internet security.

Refer to work of Danj K. Klinedinst web hosted in January 2016.[12] The team completed a survey for vulnerabilities in common home Wi-Fi routers in 2015 and reported a number of vulnerabilities database.

10.2.3 Security Requirements

IoT reference architecture means a guide for one or more concrete architects. IoT reference architecture is a set of three architectural views—functional, information, and deployment and operational. A functional view is from F. Carrez and co-workers[13]. Security is one of the functional groups (FG) of the functional view. FG for security consists of security functions between the application and device.

Security FG contains five sets of functions which are required for ensuring security and privacy. Large number of devices, applications and services communicate in IoT. Five functional components (FCs) of security are defined in IoT reference architecture.

Following are five functional components (FCs):

1. Identity management (IdM)
2. Authentications
3. Authorisation
4. Key exchange and management
5. Trust and reputation

Figure 10.1 lists the functions of a security function group in functional view in IoT reference architecture.

10.2.4 Threat Analysis

A threat-analysis tool first generates the threats and analyses a system for threat(s). Threat analysis means uncovering the security design flaws after specifying the *stride* category, data flow diagram, elements between that the interactions occuring during the stride, and processes which are activated for analysis. Stride means a regular or steady course, pace or striding means, passing over or across in one long step (set of statements when considering a threat to a software component). Stride means taking a long step for dainty little steps. Example 10.2 shows the case of a threat-analysis tool for analysis during a stride.

[11] https://insights.sei.cmu.edu/cert/2016/01/coordinating-vulnerabilities-in-iot-devices.html
[12] http://www.kb.cert.org/vuls/
[13] F. Carrez and co-workers, 2013, as referred in book A Jan Holler and co-workers, "*M2M to Internet of Things—Introduction to a New Age of Intelligence*", *Chapter 8* Academic Press/Elsevier, 2014.

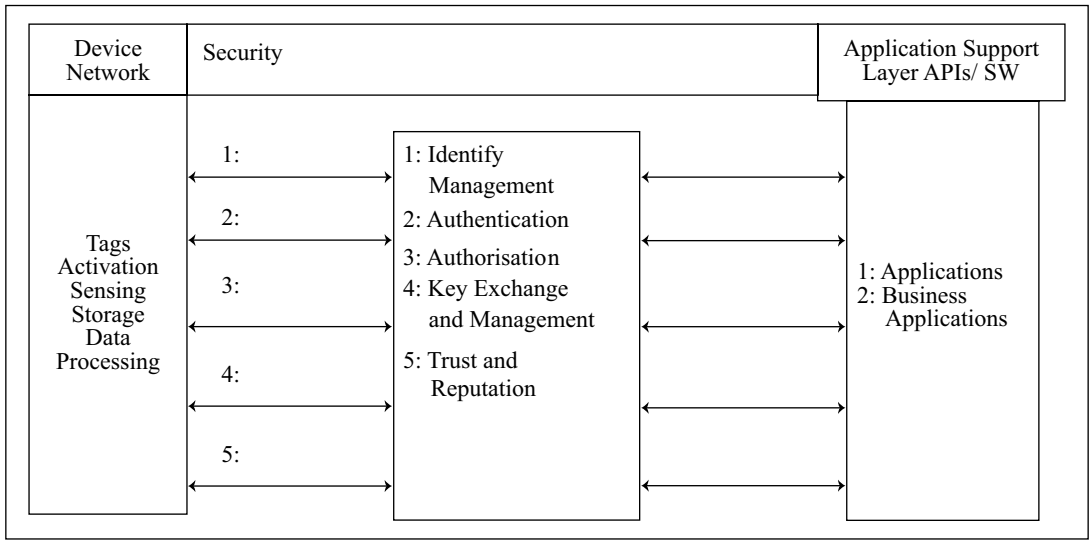


Figure 10.1 Security function group components in functional view in IoT reference architecture

Example 10.2

Problem

How is the threat analysis for a system performed using Microsoft Threat Modelling Tool 2014?

Solution

A model is designed for threat analysis with definitions of strides and elements. The interacting elements are processes, data store, flow, boundary or may be external specified elements in the system. The tool has three components: Getting started guide, Create a model and Open a model. 'Create a model' is used after 'Getting started guide'. The tool component 'open the model' precedes the threat analysis for each stride.

The tool also provisions for definitions of new threats using a stride category. A stride category is first created which generates a list of threats which are active and based on interactions between the elements. The list is as per the definitions in the model open for analysis.

Figure 10.2 shows an example of tool usage for threat analysis during a web service interaction between the application and web.

Examples of element types are process, data store, flow, boundary and an external specified element. Tool predefines a number of threat categories. A new category can also be created. The tool suggests threat definitions and generates the mitigation solutions automatically.

The tool messages on display (analysis view) show the vulnerabilities and the data flow diagram. For example, data flow between the device and application or service.

The analysis view shows threats which are active and which inactive. An example of threat is 'data store inaccessible'. The view also shows the category in which it is active and inactive. A stride category example is 'spoofing' and threat is 'spoofing the web context process'. Another example of threat is 'denial of service category'.

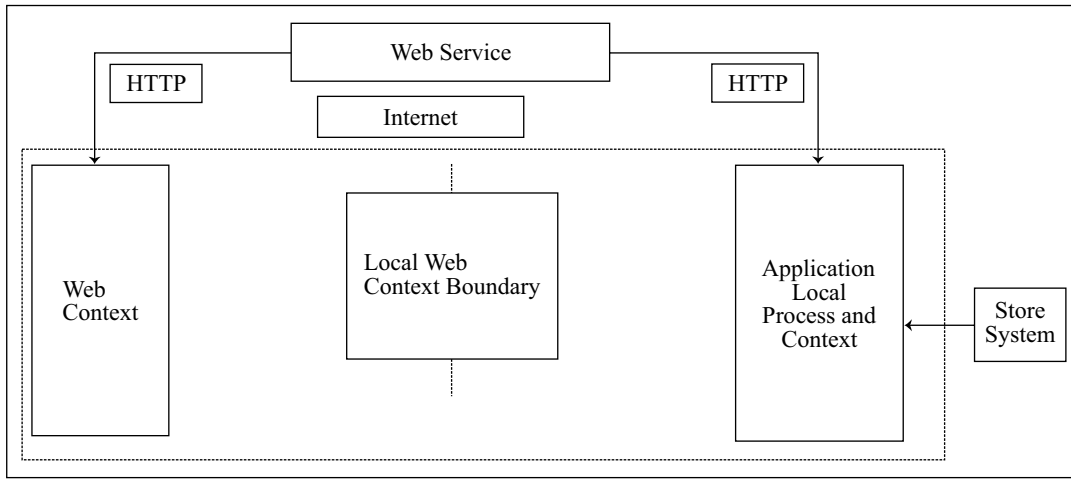


Figure 10.2 An application threat model in Microsoft threat modelling tool display when 'diagram' item selected

When a search item is for the element 'process' then the view displays the processes which are active and inactive among the OS process, Thread, Kernel Thread, Native Application or Managed Application, Thick Client, Browser Client, Browser, Active-X Plug-ins, Web Server, Web Application and Win32 Server.

Reconfirm Your Understanding

- Message privacy functions ensure that messages do not reach into the hands of unrelated entities and reach from the things to the applications or services only.
- IoT has a large number of vulnerabilities. This is due to participation of the number of layers, hardware sublayers, FCs, applications and services.
- Secuity features must incorporate in a standard format recommended for IoT.
- OWASP has identified vulnerabilities in detail and listed top ten vulnerabilities in IoT applications/services. They have also defined the attach surface areas for vulnerabilities.
- CERT at CMU defined the vulnerabilities in common home WiFi routers and reported a number of vulnerabilities and listed them in a database.
- An IoT reference architecture has three—functional, information, deployment and operational.
- Five functional components (FCs) in security functional group in-between the application/service and device are identity management, authentications, authorisation, key exchange and management, trust and reputation.
- A threat analysis tool uncovers the security design flaws after specifying the *stride* category, data flow diagram, elements between which the interactions occur during the stride, and specifying the processes which are activated for analysis.

Self-Assessment Exercise

1. How do you define message privacy? ★
2. What do you mean by trust? ★
3. List the top ten vulnerabilities for attack. ★
4. List the focus areas of 'I am the Cavalry' for computer security issues. ★★
5. List the focus areas of OWASP and importance of each in IoT security. ★★
6. Write the meaning of SQL injection, cross-site scripting, cross-site request forgery, account lock out, username enumeration, weak passwords, known default credentials, transport encryption, encrypted personally identifiable information (PII). ★★★
7. Write the usage of five functions components in the security group of functions. ★★
8. List the steps in threat analysis when using Microsoft Threat Analysis Tool 2014. ★★★

10.3 USE CASES AND MISUSE CASES

LO 10.2 Familiarise with modelling using use cases and misuse cases

UML notation use case diagrams are required for the FCs of a security FG. A use case analysis enables the requirement analyses. Use cases are key features of many models and frameworks for processes development. Oracle Unified Method (OUM) and IBM Rational Unified Process (RUP) are examples of frameworks for software development process models and requirements analyses.

Figure 10.3 shows a simple use case diagram for generating and communicating a key.

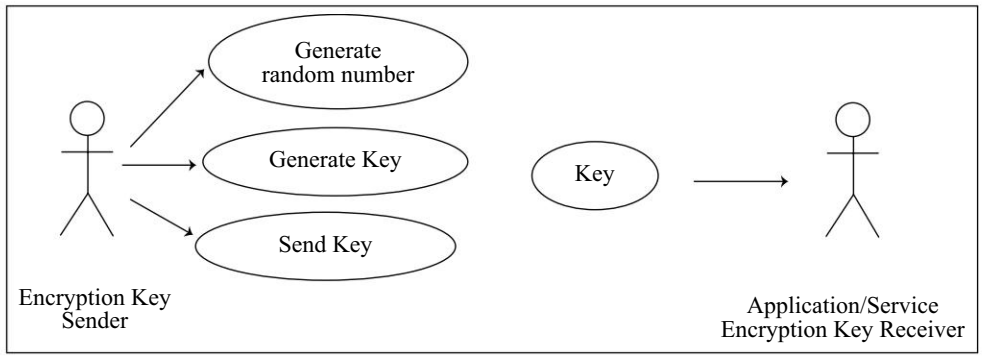


Figure 10.3 Simple use case diagram for generating and communicating a key

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

Example 10.3 gives the use cases for security and privacy in IoT applications and services.

Example 10.3

Problem

What are the examples of security issues for which use cases are required in IoT?

Solution

Use cases depict the requirements as follows:

- *Boot process protection:* Boot process runs initial programs. If it corrupts then the system fails.
- *Secure software and firmware update:* When software and firmware update, then secure updates are needed.
- *Stored data encryption and integrity protection:* Data storage needs protection from hackers.
- *Platform integrity verification:* The device platform integrity needs verification, else fake device platform communicates and the fake sends the command spoofing as the IoT device platform commands.
- *Key generation*, management and exchanges need security as these are prerequisite for authentication, authorisation and messages encryption.
- *Authentication* of sender and receiver is must for secure message communication.
- *Secured communication through a network:* Required for secure message exchanges between IoT devices and applications or services.
- *Lifecycle management:* All messages, keys and data have specific life and need to be deleted from the system.

Firstly, the actors and collaborating actors are identified. Then, misuse cases are developed for each of them. Purpose of this is creating specifications for communicating the potential risks and rationale for security-relevant decisions to the device platform, application or service. Example 10.4 elucidates the misuse cases for security and privacy in IoT application/services.

Example 10.4

Problem

What are the examples of security issues for which misuse cases are required in IoT so that the requirements of new use cases can be analysed?

Solution

Misuse cases depict the requirements as follows:

- Identity misuse cases
- Non-repudiation
- Crypto offloads: The transfer of cryptographic keys and messages need to be protected from eavesdropper or hackers
- Eavesdropping
- Fake server
- Fake device platform
- Flooding of TCP/SYN
- Unauthorised access to a data store
- Audit and accountability: Each transaction needs to be securely stored for future auditing and fixing accountabilities.

Reconfirm Your Understanding

- UML notation use-case-diagrams are required for FCs of a security FG.
- Use cases analysis enables requirements analyses. Use cases are key feature of many models and frameworks for processes development.
- Examples of use cases need are boot process protection, secure software, firmware update, stored data encryption and integrity protection. Data storage need protection from hackers.
- Misuse cases define the required behaviour of components under development which should not happen. That highlights the threats to enable designing of new use cases for prevention of attack and find out what should not happen.
- Example of misuse cases requirements are identity misuse cases and non-repudiation.

Self-Assessment Exercise

1. What is the purpose of use cases? ★
2. Explain the meaning of actors and collaborating actors in use cases. ★★
3. What is the purpose of misuse cases? ★
4. Write the definition of each of the following terms: communicating the potential risks, rationale for security-relevant decisions, Misuse cases for key management. ★★
5. Draw a use case diagram for creating authentication code using SHA1 algorithm. ★★★
6. Write meanings of misuse cases for identity, non-repudiation, crypto offloads, eavesdropping, fake server and fake device platform. ★★★

10.4 IoT SECURITY TOMOGRAPHY AND LAYERED ATTACKER MODEL

LO 10.3 Outline security tomography of large networks and layered attacker model

10.4.1 Security Tomography

Computational tomography means a computing method of producing a three-dimensional picture of the internal structures of an object, by observation and recording of the differences in effects on passage of energy waves impinging on those structures.

Computational security in complex set of networks utilises the network tomography procedures of identifying the network vulnerabilities. This enables design of efficient attack strategies.

Computational security in complex set of networks and system utilises the security tomography procedures of identifying the vulnerabilities

A complex set of networks may be distributed or collaborative. Network tomography refers to the study of vulnerabilities and security aspects for network monitoring in a complex system, such as WSNs, RFIDs or IoT networks and allocating resources and ensuring network reliability and security.

Monitoring of individual nodes is not fast acting and is also impractical. Network tomography helps in observing each network section (for example, a WSN nodes network between two access points) and subsections. The security tomography means finding attack vulnerable sections/subsections from the observations for behaviours using a finite number of objects or threats in a complex set of subsystems.

10.4.2 Layered Attacker Model

Figure 10.4 shows a layered attacker model and possible attacks on the layers.

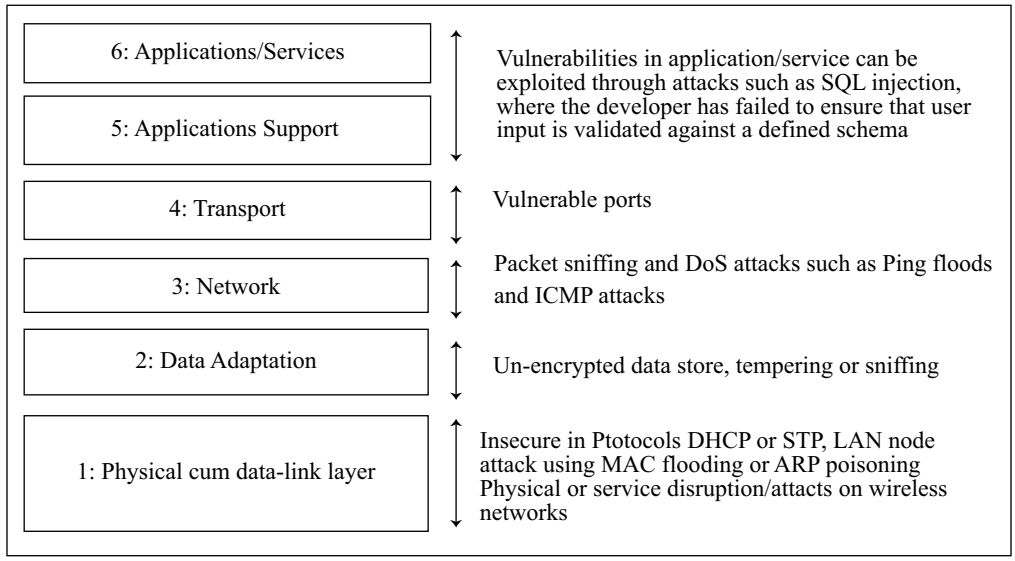


Figure 10.4 Layered attacker model and possible attacks using IETF six-layer modified model for IoT/M2M

Following are the suggested solutions for mitigating the attacks on the layers (OSI modified six layers IoT Architecture).

Layer 1 Attacks Solution

Solution depends on the devices used. For example, link-level provisioning of security uses—BT LE link level AES-CCM 128 authenticated encryption algorithm for confidentiality and authentication, and ZigBee at link-level security using AES-CCM-128.

Layer 2 Attacks Solution

Programming the network switches to prevent internal node attacks during use of DHCP or Spanning Tree Protocol (STP). Additional controls may include ARP inspection, disabling unused ports and enforcing effective security on VLAN's (Virtual LAN) to prevent VLAN hopping. VLAN refers to a group of end stations with a common set of requirements, independent of a physical location.[14] VLANs have the same attributes as a physical LAN but allow you to group end stations even if they are not located physically on the same LAN segment.

LWM2M OMA specification for device gateway to the Internet has provisions for MAS for security, root key data store, and devices and data authentication.

Layer 3 Attacks Solution

Use of temper resistant router, use of packet filtering and controlling routing messages and packets data between layers 3 and 4 through a firewall reduces the risks.

Layer 4 Attacks Solution

Port scanning method is a solution which identifies the vulnerable port. A solution is the opening of network ports and configuring effectively the firewall, and locking down ports only to those required. A solution is DTLS between layers 5 and 4. The DTLS has provisions for three types of security services, viz. *integrity*, *authentication* and *confidentiality*. A solution is include SASL (Simple Authentication and Security Layer) for security when using the XMPP protocol.

Layers 5 and 6 Attacks Solution

Above layer 4, we are looking primarily at application-level attacks which are results of poor coding practices. Assume an attacker injects the SQL input to extract data from the database (e.g. SELECT * from USERS). When the application fails to validate the injection, the query extracts the data.

Web applications/services can use HTTPS communication link. The features of S-HTTP (Secure HTTP) are as follows:

- Application-level security (HTTP specific)
- Content privacy domain header
- Allows use of digital signatures and encryption, various encryption options
- Server-client negotiations
- Cryptographic scheme is a property assigned for the link
- Specific algorithm is the value assigned
- Direction specification is done, one-way or two-way security

[14] http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12 and http://www.cisco.com/c/en/us/td/docs/2SX/configuration/guide/book/vlans.html

CISCO suggested layered framework provisions for following solutions:

- Layers 1–6: Role-based security
- Layers 1–4 Anti-temper and detection-based security
- Layers 1–6: Data protection and confidentiality
- Layers 1–6: IP protection

Reconfirm Your Understanding

- Monitoring of individual nodes is not fast acting and is also impractical. Network tomography helps in observing each network section (for example, a WSN nodes network between two access points) and subsections.
- Security tomography enables finding the attack vulnerable sections/subsections on observations for behaviours using a finite number of objects or threats in a complex set of subsystems.
- Data between the things and application/service communicate through six layers and their sublayers. Layered attacker model shows the vulnerabilities at each layer, so that necessary solutions can be given for each layer attack. For example, using a temper resistant router, packet filtering and controlling routing messages and packets data between layers 3 and 4 through a Firewall reduce the risks.
- S-HTTP (Secure HTTP) enables secure communication between the client and server sockets.

Self-Assessment Exercise

1. Why does security tomography enable fast detection in case of complex set of subsystems or networks? ★
2. List the physical cum data-link layer attacks in layered attack model. ★
3. What are transport layer attacks in layered attack model? ★
4. How do the port scanning and DTLS features enable mitigation of the attacks at layer 4. ★★★
5. List and explain the features of HTTPS. ★★
6. What are the meanings of ARP inspection and disabling unused ports and how are they used for layer 2 attack mitigation? ★★★

10.5 IDENTITY MANAGEMENT, AND ESTABLISHMENT, ACCESS CONTROL AND SECURE MESSAGE COMMUNICATION

LO 10.4 Examine functions for source identity-management, identity-establishment, device messages access-control, message-integrity, messages non-repudiation and availability in IoT applications and services

Source of message needs to specify an identity (ID) when sending the messages. The receiver can thus know that from where the messages have been received. Number of ways exist for specifying identity (ID). The messages can be from several sensors, actuators and platforms and those may be for several applications and services. Id management and establishment for IoT are therefore basic requirements.

A MAC address can specify identity of a computing device platform. However, the platform may connect several sensors and actuators. An application layer may consist of number of applications and services. An URI (Universal Resource Identifier) can be used on the Internet. Many devices however do not use the URI. An Object Identifier (OID) in IoT can have the following identifiers:

- Types of things (for example, streetlight, vehicle, ATM, WSN, RFID)
- Class identifier, since it refers to a class (or type, or category) of things; for example, make and model
- Instance identifier; for example, VIN (Vehicle identity number) for vehicles

Subsection 10.5.1 describes identity management and identity establishment mechanisms. Subsection 10.5.2 describes access control mechanisms. Message integrity, non-repudiation and availability are important requirements for security and privacy. Subsections 10.5.3 to 10.5.5 describe these.

10.5.1 Identity Management and Establishment

Identity Management (IDM) for the devices, applications and services is an FC of security FG. IdM means managing different identities, pseudo-names, hierarchies of group IDs as well as IDs for message senders and receivers. The FC anonymously manages the IDs. Example 10.5 gives four examples of uses of IDM functions.

Example 10.5

Problem

Give examples of usages identity management for establishing identities of device and application/services.

Solution

IETF transport layer does the device IDM and identity registry functions.

Oracle's IoT architecture (Figure 1.5) provisions for device IDM functions at the manage sublayer between stream and acquire layers.

OMA-DM model suggests use of a DM server. One of the functions of a DM server is assigning the device ID or address, and activating, configuring (managing device parameters and settings), subscribing to device services, opting out of the device services, and configuring the device modes (Section 2.4.2).

Reader uses an RFID identity manager when reading from the RFIDs. EPCglobal architecture framework is used for assignment of a unique identity for business processes, applications and services, and uniquely identify the physical objects, loads, locations, assets and other entities. ONS performs the lookup functions based upon the DNS. DNS name enables web server Internet connectivity using HTTP, REST, WebSockets and Internet protocols (Section 7.6.2).

NIST suggests the specifications for cyber-physical systems and for the asset identification specifications, and common platform enumeration specifications for naming, name matching, dictionary and applicability language.

ITU-T X.660 suggests the procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree. ITU-T X.672 suggests an OID (object identifier) resolution system. ITU-T OID Flyer suggests object identifiers and their registration authorities: your solution to identification.

Communication between the device and application/services is after each one establishes the identity of the other securely, using authentication and authorisation and other functions.

10.5.2 Access Control

Three FCs in a security FG for ensuring security and privacy are:

- Authentications
- Authorisation
- Key exchange and management

Authentication

ID establishment and authentication are essential elements of access control. A hash function or MD5 gives the irreversible result after many operations on that and the operations are just one way. The algorithm generates a fixed size, say, 128 or 256-bit hash or digest value using authentication data and secret key. Only the hash or digest value communicates. The receiver-end receives the value, and compares that with a stored value. If both are equal then the sender is authenticated.

Hash function characteristic features are pre-image resistance, hash function should not alter, before or after communication and should be as per the previous image (original message), second pre-image resistance: hash function should not be altered by an in-between entity (called eavesdropper), should remain the same as one for the previous image (original message) should be collision-resistance and should not be the same for any form of altered message.

Authorisation

Access control allows only an authorised device or application/service access to a resource, such as web API input, IoT device, sensor or actuator data or URL. Authorisation model

is an essential element of secure access control. The standard authorisation models are as follows:

- Access Control List (ACL) for coarse-grain access control
- Role-Based Access Control (RBAC) for fine-grain access control
- Attribute-Based Access Control (ABAC) or other capability-based fine grain access control

An access control server and data communication gateway can be centrally used to control accesses between application/service and IoT devices. The server central control can be on a cloud server. Each device can access the server and communicate data to another server.

Alternatively, a distributed architecture enables:

- Each device to request access to the server and the server grants application/service access token
- Each application/service to request access to the server and the server grants device access token for the device.

Key Exchange and Management

Key of sender messages needs to be known to receiver for accessing the received data. Key of respondent of messages needs to be known to sender for accessing the responses. The keys, therefore, need to be exchanged before the communication of authentication code, authorisation commands and encrypted messages. Since each application/service component and device data application or service may need unique and distinct keys, an FC provisions for the functions of key management and exchanges. Example 10.6 gives the steps for key exchanges and encryption and decryption.

Example 10.6

Problem

Show and list the steps for the key generation and exchange for authentication and authorisations followed by secure communication of an application/service message to the device/gateway.

Solution

Figure 10.5 shows the steps of using FCs. The steps are for key exchanges and management, authentication and authorisations. The steps follow secure communication of an application or service message to the gateway and device.

Steps for designing use case for key exchanges and encrypting and decrypting the messages are:

1. 1 and 2: Device/gateway *D* generates secret key *K*
2. 3 and 4: Application/service *A* generates secret key *K*
3. 5 and 6: *D* exchanges key *K* and *K′*
4. 7 and 8: Authentication and authorisation of *D* and *A*,
5. 9: Message given to encryption algorithm
6. 10: Encrypted message using *K′* to *D*
7. 11: Decryption algorithm decrypts encrypted message using *K′* for *D*
8. 12: Message *M* retrieves at *D*

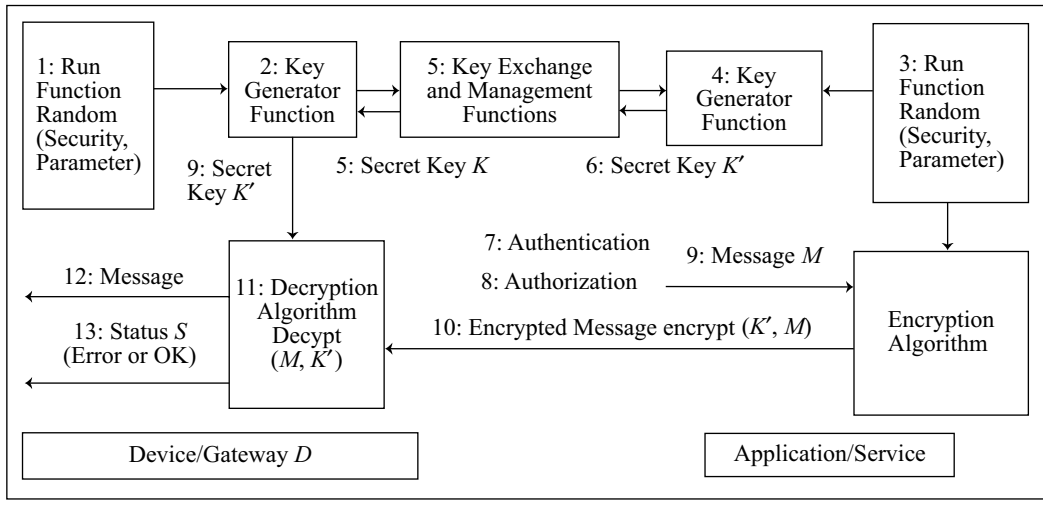


Figure 10.5 Steps during key exchanges and management, authentication and authorisations followed by secure communication of application/service message to the device/gateway

9. 13: Status code sends 'Error' or 'OK'. As per the status of data exchanges for authentication, authorisations and message communication.

Similarly, message encrypts at *D* using *K* and decrypts at *A* using *K*.

10.5.3 Message-Integrity

An important aspect of system design is message integrity (data integrity), which means the message remains unaltered. A message should not be altered during its communication. The encrypted data after decrypting should be identical to one before encryption.

Message integrity check involves the following steps:

- Hash function or digest algorithm calculates 128 or 192 or 256 Hash value h_0, taking the message M_0 and K as inputs
- Appends the h_0 along with the message
- Communicate or store h_0

Integrity Check

Integrity check steps are as follows:

1. Retrieve *M* any time later. Assume that retrieved message is M_1.
2. Calculate 128 or 192 or 256 Hash value h_1, taking the message M_1 and *K* as inputs.
3. Compare h_1 and h_0.
4. Message is unchanged if $h_1 = h_0$, and integrity check passes else fails.

Message or data integrity means maintaining and assuring the accuracy and consistency over its entire lifecycle.

10.5.4 Message Non-Repudiation

Non-repudiation means an assurance that the source of a message once having communicated the data to a sender, cannot deny it later, that the message was not sent from the source and is not the same as sent earlier. It means data is signed and the signature put at the source cannot be denied. Digital signature is a method which ensures non-repudiation.

The service provides proof of the message's origin as well as its integrity. A digital certificate asserts the origin using a public key infrastructure. A digital signature is certified by a trusted digital certifying service [trusted third party (TTP) service]. TTP protects the private (secret) key and issues the certificate that a message was sent using this specific private (secret) key of source incase the private key is lost and used by some other source of message. Only the TTP is permitted to be the repository for public key certificates.

Example 10.7 gives the steps for signing, issue of digital certificates to a signed message and verification of a signed message.

Example 10.7

Problem

Show and list the steps for the use of TTP service and steps for signing, issue of digital certificates to a signed message and verification of a signed message.

Solution

Figure 10.6 shows the use of TTP service and steps for signing, issue of digital certificates to a signed message and verification of a signed message.

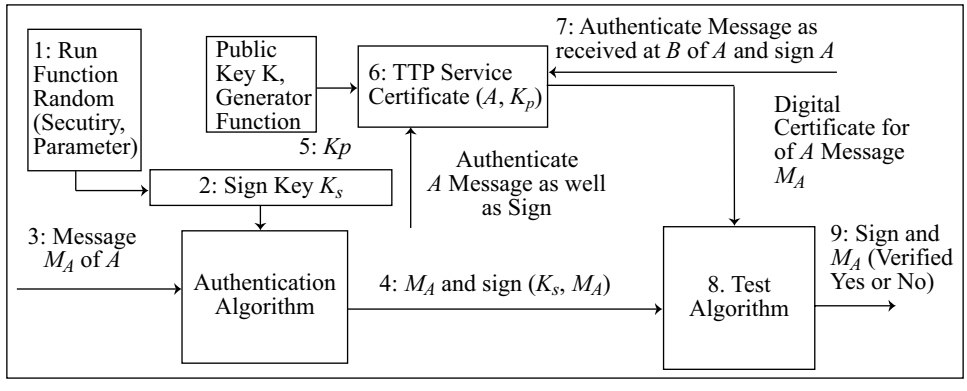


Figure 10.6 Use of TTP service and steps for signing, issue of digital certificates to a signed message and verification of signed message

Steps for designing use cases are:

- 1 and 2: End TTP and A generate keys K_p and K_s
- 3 and 4: Message M_A and sign generator function sign (K_s, M_A) executes and sent to End B
- 5 and 6: K_p to TTP Service and generation of certificate (A, K_p)

- 7 and 8: Digital certificate and certificate of messages received at End *B* are verified and result sent to test algorithm.
- 9: Sign and M_A verified to be same as received at *B*, Yes or No.

Similarly messages received at End *B* can be verified.

10.5.5 Message-Availability

Message-availability affects when Denial-of-Service (DoS) attack occurs. This is because source-end message (of device or network or application/service resource) is unavailable to the intended destination-end on DoS. Examples of DoS attacks are:

- ICMP flooding which repeatedly sends control messages to the destination and thus denies the path to source end
- An SYN flood means the attacker sends a flood of TCP/SYN messages (packets) using a forged address, and the destination repeatedly sends TCP/SYN packets assuming the packet is from an actual source. The service to the source message becomes unavailable and the original message becomes unavailable at the destination
- Peer-peer attack
- Application layer messages flooding

An attack can be prevented using specific methods for specific type of attacks. Firewall is one method for preventing attacks on messages from untrusted networks.

Reconfirm Your Understanding

- MAC, URI or OID can be used as ID for the things, application or service. An OID includes types of things, class identifier and instance identifier.
- 'ID Management' is an FC of security FG.
- Authentication, authorisation and key exchanges and management are also FCs of security FG.
- ID establishment and authentication are essential elements of access control. A hash function or MD5 gives an irreversible result after many operations on that and the operations are just one way. Hash algorithm calculates hash h_0 for authentication code using a secret key. h_0 value communicates to the other end. If data store hash value h_1 matches with communicated hash h_0, then the sender stands authenticated.
- An access control server and data communication gateway can be centrally used to control accesses between application/service and IoT devices. Authorisation functions authorise access for messages at one end from the other end. Authorisation models specify coarse-grain access control and fine-grain access control.
- Key of sender of messages needs to be known to the receiver for accessing the received data. Key of respondent of messages needs to be known to the sender for accessing the responses. The keys, therefore, need to be exchanged before the authentication code, authorisation commands and encrypted messages communicate.
- A requirement is message-integrity. Sending the hash along with the message enables the message-integrity check.

- Non-repudiation assures that the source of a message once having communicated the data to a sender, cannot deny it later, that the message was not sent from the source and is not the same as sent earlier. Digital sign functions enable signed messages. A TTP service issues a digital certificate for a message, and saves it in its repository. At any time a receiver can verify the signed message from the digital certificate of that message.
- Message availability ensures prevention of DoS attacks, which can be due to a number of reasons, for example TCP/SYN flooding.

Self-Assessment Exercise

1. What are the methods of specifying an ID of an application/service or device application? ★
2. What does key management mean? ★★
3. List steps to authenticate two end-points communicating the messages. ★
4. What do you mean by coarse-grain and fine-grain authorisations? ★★★
5. Why is authentication not sufficient and authorisation needed for communicating messages? ★★★
6. List the steps for message integrity check. ★★
7. Why are the ends exchange keys? ★
8. What is the need of non-repudiation? ★
9. Write the sources for DoS which affect message availability. ★★
10. List the functions of firewall in ensuring message availability. ★

10.6 SECURITY MODELS, PROFILES AND PROTOCOLS FOR IoT

LO 10.5 Describe security model, security profiles and security protocols for IoT

An IETF recommended draft recommends the following security models for five security profiles. Table 10.1 gives the details.

Table 10.1 Security model for differing security profiles

Security Profile	Usage	Description	Security Model
SecProf_0	6LowPAN/CoAP	No security	No temper resistant (no provision for prevention of tempering)
SecProf_1	Home Usage	Operations between things without central device	1. No temper resistant 2. Sharing of keys between layers

(*Contd.*)

SecProf_2	Managed Home Usage	Operations between things and local device-central device interaction possible	1. No temper resistant 2. Sharing of keys between layers
SecProf_3	Industrial Usage	Operations between things enabled and relies on local or back-end device for security	1. Temper resistant 2. Key and process separation
SecProf_4	Advanced Industrial Usage	Ad-hoc operations between enabled things and relies on central device or a collection of control devices for security. Distributed and centralised (local and/or back-end) security architecture	1. (No) temper resistant 2. Sharing of keys between layers/ key and process separation sandbox[15]

Some applications target powerful devices aimed at more exposed applications and need security parameters such as keying materials, and the certificates must be protected in the things. For example, by using tamper-resistant hardware.

Sharing of keys has the following features:

- Needed across a networking stack of devices.
- Provides authenticity and confidentiality in each networking layer, minimise the number of key establishment/agreement handshake, needs less overhead for constrained thing for example, applications with resource constraints (for example, temperature and humidity sensor)

Key separation at different networking layers:

- Needed in advanced applications
- May also possibly use process separation and sandboxing to isolate one application from another.

CISCO IoT secure environment framework has four FCs:

1. Authentication
2. Authorisation
3. Network-enforced policy
4. Secure analytics: visibility and control

10.6.1 Security Protocols

Open Trust Protocol (OTrP) is a protocol that manages security configuration in a Trusted Execution Environment (TEE) and is used for installing, updating and deleting applications and services.

[15] https://en.wikipedia.org/wiki/Sandbox_(software_development) [for usages of sandbox for software development]

Following DTLS and X.509 security protocol details can be referred from Section 3.2.1.

- DTLS (Datagram Transport Layer Security) protocol is for maintaining the privacy during the datagram which communicate when using the CoAP or L2M2M clients and servers. It enables protection from eavesdropping, tampering or message faking. The basis of DTLS protocol is Transport Layer Security (TLS) protocol for data segment communication using the transport layer.
- X.509 protocol refers to issue of a digital certificate with a trust based on TTP authorised certification authority. It deploys a Public Key Infrastructure (PKI). The PKI manages the digital certificates and public-key encryption. It is a subunit of TLS protocol used for securing communication with the web.

Reconfirm Your Understanding

- IETF draft recommends five security profiles and the security model for each profile.
- Profiles 0, 1, and 2 are for 6LowPAN/CoAP use, home usages and managed home usage.
- Profiles 0, 1, and 2 security profiles use no temper resistant, and sharing of keys based message communication.
- Profiles 3 and 4 are industrial and advance industrial usages and these use temper-resistant, key and process separation or sandbox.
- CISCO suggests a security framework based on authentication, authorisation, network enforced policy and secure analytics: visibility and control.

Self-Assessment Exercise

1. Why does 6LowPAN/CoAP communication need no temper resistance? ★
2. Why do home usage with no central device or with central device communication need sharing of keys between layers and no provision for prevention of tempering? ★
3. Why do industrial usage operations between things that enable and rely on local or back-end devices for security need sharing of keys between layers and key and process separation? ★★★
4. Why do ad-hoc operations between things that enable and rely on central device or a collection of control devices for security in the advanced industrial usages and using distributed and centralised (local and/or backend) security architecture need no provision for prevention of tempering and sharing of keys between layers/key and process separation sandbox. ★★★
5. What are the features of key sharing? ★★
6. What are the four FCs in CISCO IoT secure environment framework? ★

Key Concepts

- Attack surface
- Authentication
- Authorisation
- Decryption
- Digest
- Digital certificate
- Encryption
- Hash
- Key exchanges
- Key generator
- Layered attack model
- Layered attack solution
- Link level security
- Non-repudiation
- Privacy
- Secure hash algorithm
- Security
- Security model
- Security tomography
- Signed message
- Threat analysis
- Trust
- Trusted party certificate
- Vulnerability

Learning Outcomes

LO 10.1

- OWASP gives definitions of vulnerabilities and highlights top ten IoT vulnerabilities
- Trust, security and privacy are required in IoT
- Message-privacy need should ensure that messages do not reach unrelated entities and reach from the things to the desired applications/services only.
- Security features incorporate in number of standards for IoT.
- OWASP specifies the attack surface areas for vulnerabilities.
- An IoT reference architecture specifies functional view. A functional view has security function group (FG). Five functional components (FCs) in the FG between the application/service and device are identity management (IDM), authentication, authorisation, key exchange and management, trust and reputation.
- A threat analysis tool can uncover flaws in any security design.

LO 10.2

- UML notation use case diagrams are required for FCs of security FG.
- Use cases analysis enables requirements analysis. Use cases are a key feature of many models and frameworks for developing the processes.
- Examples of need of use cases are boot-process protection, secure software and firmware updates, stored data encryption and integrity protection. Data store needs protection from hackers
- Misuse cases define the required behaviour of components under development which should not happen. This highlights the threats to enable designing new use cases for preventing attacks and find out what should not happen.

LO 10.3

- Security tomography technique enables fast detection of attack-vulnerable sections/subsections in a complex set of networks such as network of WSNs.
- Security tomography means finding attack-vulnerable sections/subsections by observations for behaviours using a finite number of objects or threats in a complex set of subsystems.
- Data between the things and application/service communicate through six layers and their sublayers.
- Layered attacker model specifies the vulnerabilities at each layer.

LO 10.4

- A message communicates using the ID of the device platform/application/service.
- An FC, 'ID Management' FC of security FG manages the IDs.
- Authentication, authorisation, key exchanges and management are also FCs of the Security FG.
- ID establishment and authentication are essential elements of access control. A hash function or MD5 enables authentication.
- Access control server and data-communication gateway can be centrally used to control accesses between application/service and IoT devices. Authorisation functions authorise the access for messages at one end from the other end. Authorisation models specify coarse-grain access control and fine-grain access control.
- The messages communicate using keys. Keys need to be exchanged before the authentication-code, authorisation commands and encrypted messages communicate.
- Hash algorithm enables message integrity checks.
- Non-repudiation assures security when signed messages are used, and a digital certificate issuing TTP is used for digital certificates.
- Message-availability ensures prevention of DoS attacks, which can be due to a number of reasons.

LO 10.5

- IETF draft recommends five security profiles and a security model for each profile.
- Profiles 0, 1, and 2 are for 6LowPAN/CoAP use, home usages and managed home usage and Profiles 3 and 4 are industrial and advance industrial usages and these use temper-resistant, key and process separation/sandbox.
- CISCO also suggests a security framework based on authentication, authorisation, network enforced policy and secure analytics: visibility and control

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. Encryption is a (i) process of generating new data or (ii) new authentication code using a secret key known only to a receiver (iii) Sender and receivers need not identify each other and know the keys that will be used by them. (iv) The encryption uses 32-bit, 64 bit, 128 bit, 192 bit or 256-bit key or MAC address for encrypting. ★
   (a) (i)
   (b) (ii) and (iv)
   (c) All
   (d) (i), (ii) and (iii)
2. OWASP has identified top ten vulnerabilities in IoT applications/services as (i) secure web interface, (ii) insufficient authentication or authorisation, (iii) secure network services, (iv) lack of transport encryption/integrity verification, (v) privacy concerns, (vi) insecure cloud interface, (vii) insecure mobile interface, ★★

(viii) insufficient security configurability, (ix) insecure software or firmware and (x) poor physical security.
(a) All except (i)
(b) All except (i) and (iii)
(c) All except (ii) and (iv)
(d) (ii) to (ix)

3. Functional Components (FCs) of security are defined in IoT reference architecture and they are (i) identity management, (ii) authentications, (iii) authorisation, (iv) key exchange and management, (v) trust and reputation, (vi) physical device security, and (vii) security configurability. ★
(a) (ii) to (v)
(b) (ii) and (iii)
(c) (i) to (v)
(d) (ii) to (vii)

4. OWASP definitions for cloud web interface attack surface area contains the following attacks: (i) SQL injection, (ii) cross-site scripting, (iii) cross-site request forgery, (iv) account lock out, (v) username enumeration, (v) weak passwords, and (vi) known default credentials, (vii) transport encryption, (viii) encrypted personally identifiable information (PII) sent unencrypted PII, (ix) sent device information leaked and (x) location leaked and (xi) cloud user data disclosure. ★★
(a) All
(b) All except (vii) to (xi)
(c) (v) to (xi)
(d) (i) to (v)

5. Microsoft Threat Analysis Tool 2014 analysis view shows the (i) threats which are active and (ii) which are inactive. (iii) 'Data Store accessible' is a threat. The view also shows the (iv) process category, (v) stride category, (vi) 'spoofing' is a stride category, and (vii) 'denial of service' category threat. ★★★
(a) All except (iii) and (ivi)
(b) All except (iii) and (iv)
(c) (ii) to (vii)
(d) (i) to (vi)

6. (i) Use cases use UML notation. (ii) Use cases analysis enables requirements analyses. (iii) Use cases are key feature of many models and frameworks for (iv) processes, and (v) API development. (v) Oracle Unified Method (OUM) and IBM Rational Unified Process (RUM) are examples of software components and module development using use case. ★★
(a) All are except (ii)
(b) All except (ii) and (v)
(c) All
(d) All except (iii) and (v) true

7. An object identifier (OID) in IoT can have the following identifiers: (i) URI, (ii) MAC address, (iii) IP address, (iv) types of things, (v) class identifier, and (vi) instance identifier. ★
(a) All except (iv)
(b) All except (ii) and (iii)

(c) All except (i)
(d) (iv) to (vi)

8. Trusted Third Party (TTP) service (i) certifies a digital signature, (ii) protects the private (secret) key and (iii) issues the certificate that message was sent using this specific private (secret) key of source, (iv) shares key with all and (v) only one permitted to be the repository for public key certificate and keys. ★★
   (a) All except (ii) and (iv)
   (b) All except (ii) and (v)
   (c) All except (iv) and (v)
   (d) (i) and (v)
9. Access control allows only authorised (i) device or (ii) application/service access a resource, such as (iii) web API input, (iv) IoT device, (v) sensor or (vi) actuator data or (vii) URL. (viii) Authorisation model is an essential element of secure access control. (ix) Coarse-grain authorisation means web application access authorisation. ★★★
   (a) All except (i)
   (b) to (viii)
   (c) All except (viii) and (ix)
   (d) All except (vii)
10. Layer 2 attack solutions are (i) network switches programmed to prevent internal node attacks during use of (ii) DHCP, (iii) STP (Spanning Tree Protocol) or (iv) IP. (v) Additional controls may include ARP inspection, (vi) disabling unused ports and (vii) when using LWM2M client device gateway to the Internet, then have provisions for MAS (M2M Authentication Server) for security, (viii) root key data store and (ix) devices and data authentication. ★★
    (a) All except (iv)
    (b) All except (iv) and (viii)
    (c) All except (v)
    (d) All except (ii) and (v) true

Short-Answer Questions

1. What are the focuses of OWASP? **[LO 10.1]** ★
2. Compare requirements of trust, privacy and security. **[LO 10.1]** ★★
3. What are the meanings and kinds of functional view, functional group and functional component in IoT reference architecture? **[LO 10.1]** ★★★
4. What does device management and device data integrity refer to? **[LO 10.1]** ★
5. When and why is threat analysis performed? **[LO 10.1]** ★★
6. Why is management of keys important? **[LO 10.2]** ★★
7. When is security tomography useful? **[LO 10.3]** ★★★
8. When is SQL injection a potential threat? **[LO 10.3]** ★
9. What are the layer 1 attack solutions for wireless devices? **[LO 10.3]** ★★
10. How is the access of a web API controlled in a web service? **[LO 10.4]** ★★★
11. When do you use authentication and authorisation? **[LO 10.4]** ★
12. How does the layer attack model differ from the IETF recommended security model for security profile requirements? **[LO 10.5]** ★
13. When and why do we need to use OTrP for IoT applications and services? **[LO 10.5]** ★★★
14. Why do we need TEE in IoT? **[LO 10.5]** ★★

15. When do we use DTLS and when X.509?` **[LO 10.5]** ★

Review Questions

1. What are attack surface areas in DWI and CWI defined in OWASP? **[LO 10.1]** ★
2. How is the display view during threat analysis for a system when using Microsoft threat modelling tool 2014? **[LO 10.1]** ★★★
3. What are the security requirements during message exchanges between devices to applications/services? **[LO 10.1]** ★★
4. Why are misuse cases important for security threats? **[LO 10.2]** ★★
5. How does the Misuse Cases help in secure design? **[LO 10.2]** ★
6. Draw a diagram showing possible attacks at the layers. Take OSI modified six-layer IoT model as reference. **[LO 10.3]** ★
7. Describe the steps for verification of message and digital signature for non-repudiation. **[LO 10.4]** ★★★
8. Show steps for message integrity check. **[LO 10.4]** ★★
9. Describe access-authorisation methods. **[LO 10.4]** ★★
10. What are the security model profiles and protocols for the IoT specified in the IETF draft? **[LO 10.5]** ★★★

Practice Exercises

1. Search https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project for OWASP subprojects and list them. **[LO 10.1]** ★
2. Search the reported results of CERT Coordination Centre (CERT/CC) at Carnegie Mellon University initiative for coordination the vulnerabilities in IoT devices. **[LO 10.1]** ★★★
3. Show use case diagram for streetlight sensors for ambient light, traffic density and streetlight working status communication to a gateway. **[LO 10.2]** ★
4. Tabulate the attacks in IoT applications/services at each layer. **[LO 10.3]** ★
5. Tabulate the layer attack solutions for each layer. **[LO 10.3]** ★★
6. Draw and show the steps for sending encrypted data after authentication. **[LO 10.4]** ★★
7. Show the steps by diagrams for identity management. **[LO 10.4]** ★★★
8. Redraw Figure 10.5 and show steps during key exchanges and management, authentication and authorisations followed by secure communication of message from to the device/gateway to application/service. **[LO 10.4]** ★★
9. What do you mean by security model for industrial usages in IoT? **[LO 10.5]** ★

11

Business Models and Processes Using IoT

Learning Objectives

LO 11.1 Outline the building blocks needed for a business model and understand the importance of innovation in the business model

LO 11.2 Summarise value propositions and value creation using IoT

LO 11.3 Explain IoT based business model scenarios

Recall from Previous Chapters

Earlier chapters described number of examples of IoT/M2M applications in industry and business. Sections 1.1.3, and 1.5.2 and 7.6 described Internet-connected RFIDs for logistics, tracking, inventory control, sales, security, supply-chain management and many applications.

Sections 1.5.3 and 7.7.6 dealt with applications of Internet connected complex extensively interconnected WSNs. Section 1.6.1 discussed about M2M communication in extensively interconnected and Internet-connected M2M applications and services, such as coordinated movement of tools, robots, drones, refinery operations, and sequential control of machines.

Applications of IoT architecture and Internet of ATMs for generating income from transaction charges, charges for advertisements at the machine locations, and idle-state advertisements of bank products and bank services at the display screen were discussed in Examples 2.3 and 5.8. Internet-connected chain of Automatic Chocolate Vending Machines (ACVMs) with associated business activities such as displaying advertisements for chocolates, news, weather reports and events in the city whenever not in use, and ACVM supply chain maintenance services and new business activities using predictive analytics were dealt with in Examples 5.1, 5.5, 5.9 and 6.6.

Examples 5.2 and 5.7 discussed Internet-connected automobile for automotive service centre maintenance and timely servicing or replacements of the components, with minimum number of visits to the service centre using predictive analytics.

Section 5.4 discussed business intelligence, business processes and distributed business-processes having wide applications, such as in automotive maintenance applications/services.

Section 7.3 discussed IIoT (Industrial IoT) applications in manufacturing, maintenance and industry. IIoT enables the integration of complex physical machinery, industry and business.

Example 7.8 explained applications of IIoT technologies in optimising the processes for bicycle manufacturing.

Example 7.9. explained IIoT applications for railroad service centre using Internet-connected sensors on the rails and predictive analytics.

11.1 INTRODUCTION

IoT has many applications in different domains such as applications/services of RFIDs in logistics, WSNs, predictive maintenance of railroads, oil pipelines, monitoring of transport units in logistics, detecting machine faults in industry, and deploying these IoT based applications and services. Industry 4.0 is a new paradigm which will enable intelligent connected manufacturing and will focus on uses of interconnected IoT and IIoT.

Business is driving force for growth of industry and technologies. This chapter focuses on design of business models, need of innovation in a model, value propositions in a model and value creation using a model with reference to IoT and IIoT.

Following are the lessons learnt earlier and new key terms, which require understanding for learning business model design, innovation and value creation in business processes.

Application or App means software for application, such as software for creating and sending an SMS, measuring and sending the measured data, or receiving message from a specified sender. Application means application software. App is the abbreviation popular for application in the device whenever only one specific task is executed following user interaction(s).

Service means a mechanism, which enables the provisioning of access to one or more capabilities. An interface for a service provides the access to the capabilities. The access to each capability is consistent with the constraints and policies, specified by a *service description*.

Business intelligence (BI) is a process that enables a business service to extract new facts and knowledge, and then undertake better decisions. The new facts and knowledge follow from the earlier results of data processing, aggregation and then analysing those results.

Business process (BP) is an activity or series of activities or a collection of inter-related structured activities, tasks or processes. A BP serves a particular goal or specific result or service or product. The BP is a representation or process matrix or flowchart of a sequence of activities with interleaving decision points (Section 5.1).

Distributed BP (DBP) is a collection of logically interrelated business processes. DBP reduces the complexity and communication costs, and enables faster responses and smaller processing load at the central BP system. DBP is similar to the distribution of control processes for each group of lights at the gateway itself with reduced complexity, communication costs, faster responses and smaller processing load at the central system of lighting control (Example 1.2).

DBP management is management of DBPs in an enterprise network.

BPs integration is integration of BPs to reduce the complexity and communication costs, and enable faster responses and smaller processing load at the central BP system (for example in an enterprise system).

Industry 4.0 can be defined in several ways as:

- Industry 4.0 refers to a collective term embracing a number of contemporary automation, data exchange and manufacturing technologies.[1]
- Industry 4.0 is the vision of tomorrow's manufacturing deploying intelligent factories, machines, raw materials, and products in which the Internet plays a vital role.[2]
- Industry 4.0 is the 'fourth industrial revolution' with use of Cyber Physical Production System (CPPS), which is a merger of virtual and real world.[3]
- Industry 4.0 is the intelligent solution for connected manufacturing.[4]

Following sections describe the business models and processes when using IoT. Section 11.2 describes business models and business-model innovation; Section 11.3 describes value creation and Section 11.4 describes business model scenario for IoT.

11.2 BUSINESS MODELS AND BUSINESS MODEL INNOVATION

L0 11.1 Outline the building blocks needed for a business model and understand the importance of innovation in the business model

Business model is a concept in business. Following subsections describe the concepts of business models and business model innovation in business processes.

[1] https://en.wikipedia.org/wiki/Industry_4.0
[2] https://www.youtube.com/watch?v=HPRURtORnis
[3] www.deloitte.com/.../ch-en-manufacturing-industry-4-0-24102014.pdf
[4] https://www.bosch-si.com/solutions/manufacturing/industry-4-0/industry-4-0.html)

11.2.1 Business Models

A business model is an abstraction for a range of informal and formal descriptions to represent core aspects of a business

Business models continue to grow and innovate since ancient times. Ancient time models were 'barter resource with one another. Later models were purchase, add value and sale, and use of currency, 'plan, purchase raw material, manufacture a product on bigger scale, distribute and sale and profit', and the innovations still continue. A business model nowadays takes into account many factors, such as competitive advantage, experience curves, value chain, theory of portfolio of products and services, core competencies of business organisation and generic strategies. Portfolio means mix of two or more products or services. For example, producing watches and jewellery studded watches of simple to sophisticated designs. Another example is offering predictive analytics for automobile service centres as well as oil pipelines.

Several definitions and descriptions of 'business model' exist in literature. A *business model* can be defined as a conceptual structure supporting the viability of a business, including its purpose, its goals and its ongoing plans for achieving them.[5]

A business model can be defined as abstract representation of an organisation and this representation may be conceptual, textual, and/or graphical. Representation is for the all the core inter-related architectural, co-operational, and financial arrangements. Architecture includes organisational infrastructure and technological architecture. Representation includes many activities of present and future, and core products and/or services the organisation offers, or will offer.

The term 'business model' refers to 'uses of a range of informal and formal descriptions to represent core aspects of a business, business process, strategy, practices and operational processes and policies including culture.

A business model may focus not only on financial goals but also on the business sustainability or establishing a corporate culture when offering value to customers.

Working on a Business Model

Business model canvas is a visual chart with a number of elements, which describe building blocks of a company business

Documentation for a business model has many benefits, such as maintaining a focus on corporate goals and reviewing operational practices. A popular way of generating or working on a business model is using a canvas which is a visual template for developing new or documenting existing business model. The canvas was designed and developed by Alexander Osterwalder[6] and his co-workers. The canvas is a single reference model. Its basis is conceptualisation similarities of a wide range of business models.

[5] http://www.whatis.techtarget.com%20business%20model/

[6] A. Osterwalder, Y. Pigneur, A. Smith et al., "*Business Model Generation*", Wiley, 2010

A 'Business Model Canvas' is a visual chart with elements. The elements describe the companies or organisations product's value proposition, infrastructure, customers and finances. Figure 11.1 shows nine building blocks of this canvas.

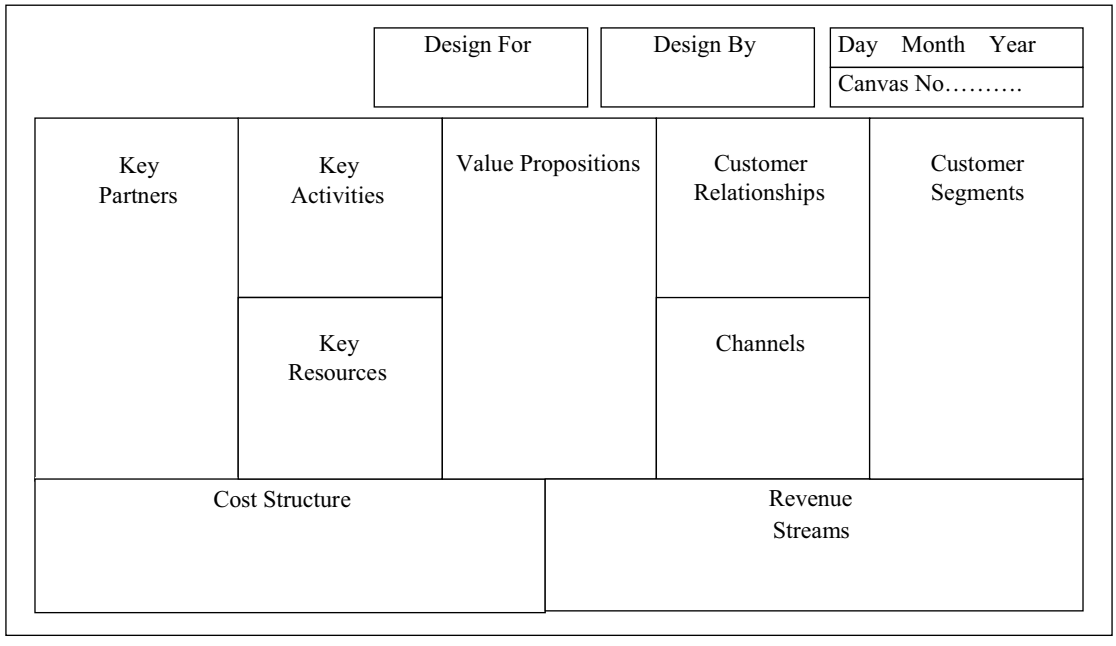


Figure 11.1 Nine building blocks of a business model canvas

Osterwalder Business Model Canvas has nine business model building blocks. Three building blocks are for business infrastructure which consist of:

1. ***Key partners:*** Strategic alliances between competitors or non-competitors to optimise the operations and reduce risks of a business model
2. ***Key activities:*** Key activities to execute a company's value proposition
3. ***Key resources:*** Key resources for sustaining and supporting the business, and necessary for creation of value to customers. Examples of resources are financial, physical, human and intellectual, customer segments, platforms, markets and diversifications of innovative products.

Four building blocks are business offerings which consist of:

1. ***Value propositions:*** Products and services offered, their features such as performance, efficiency, accessibility, price, cost, convenience, usability and design, and how they differ from competitors fall in this category
2. ***Customer relationships:*** Identified type of relationship of the company to be created with their customers and targeted segments
3. ***Customer segments:*** Identified sets of customers, segments, client groups and diverse groups based on the value propositions offered

4. ***Channels:*** Effective, fast, efficient and cost-effective channels to deliver value proposition to its targeted customers

Two building blocks are for business finances which consist of:

1. ***Cost structure:*** Cost constituents, such as the input raw material, manufacturing, maintenance, packaging, logistics, machinery replacement, to be considered in offering the value propositions and services, and considerations of scopes of economies in the operations
2. ***Revenue stream:*** Identified types of income sources, such as income from sales of product and physical goods, usage charges for the services, charges for subscription, sales income, service usage charges, subscription charges, relationship of the company to be created with their customers and targeted segments

Canvas Format

Two canvas formats are:

1. Large surface printed sheet to enable sketching, creativity, analysis, understanding or discussion on business model elements
2. Web-based interface for canvas where the entries of elements can be made and creative suggestions can be placed, understanding can be obtained, analysis can be done or discussion on business model elements can be done.

Example 11.1

Problem

A business model canvas for services: Suggest a business model canvas for a bank for offering banking services using Internet of ATMs.

Solution

Figure 11.2 shows nine building blocks of the canvas.

Notes:

1. Bank company is mutually dependent partner for ATM manufacturing, installation, maintenance and services. Reason is that bank services to customers crucially depend on their products and services, and the company has a business model, which depends on the bank and bank customers.
2. Model can be modified when bank enters into strategic partnerships with other banks to serve the other banks' customers as well.

Subscription Business Model

One business model is subscription business model. A customer pays for accessing the service or product at periodic intervals. For example, Internet data services, cloud platform services, data centre services etc. The customers can also make members who pay the membership fees for an organisation offering the services. Example 11.2 elucidates subscription business model.

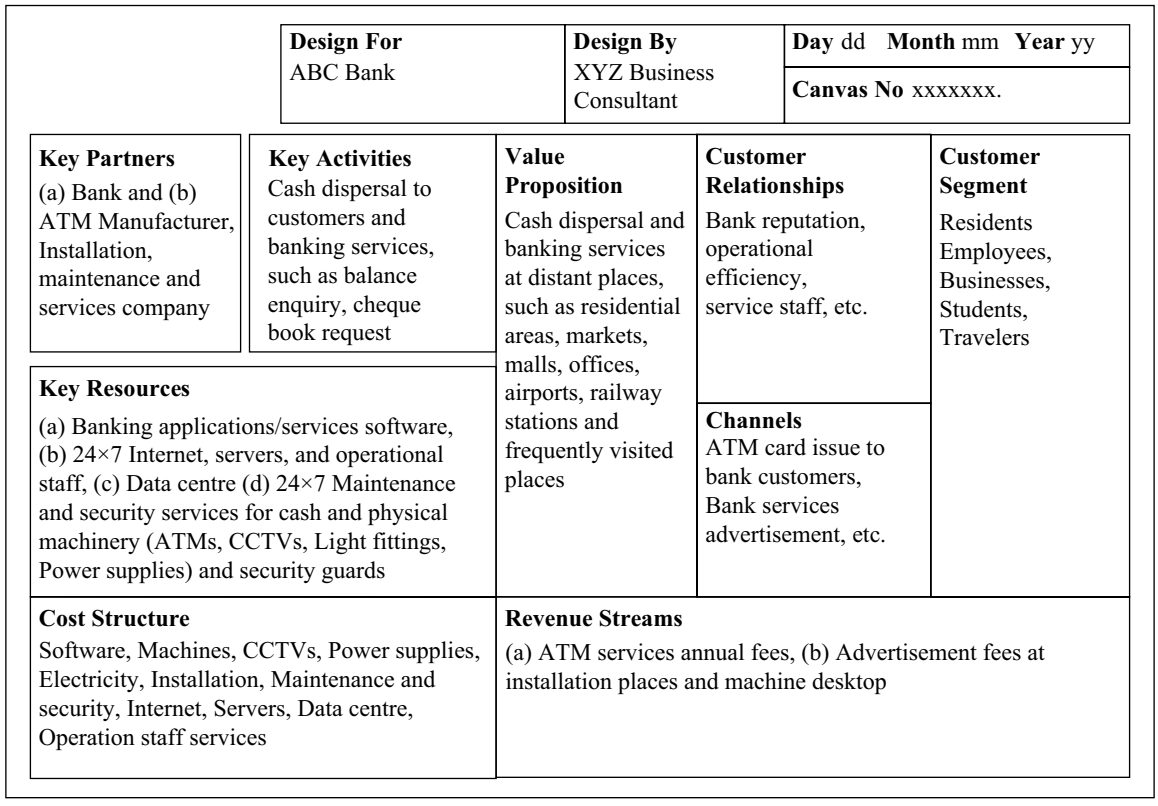


Figure 11.2 Business model canvas for offering banking services using Internet of ATMs

Example 11.2

Problem

Suggest a business model canvas for Internet of Streetlights and controlling of operations service.

Solution

Figure 11.3 shows nine building blocks of the canvas for Internet of Streetlights and control of operations with minimum electricity use.

Customisation Business Model

One business model is customisation business model. Each customer pays for customisation of a product/service. The customers can also make payment once or in part once and later, after product usage training and further improvement costs. Example 11.3 show a customised business model.

Design For Internet of Streetlights Service	**Design By** IoT Products Design Company	**Day** dd **Month** mm **Year** yy **Canvas No** xxxxxxx.

Key Partners Streetlights Service	**Key Activities** Controlling of operations of Streetlights with minimum electricity usages	**Value Proposition** Energy Efficient Control and Efficient 24× 7 Services for Streetlights, Installation s, Gateway Servers and Central Server	**Customer Relationships** Company Reputation for efficient and reliable services	**Customer Segment** Smart city services company or Municipal corporation
Key Resources (a) Sensors data for surrounding light intensity and traffic presence and density, streetlight functioning status, (b) Internet of Streetlights services and control software, (c) 24×7 Internet, Gateway, Servers, (d) Cloud or data centre maintenance service			**Channels** None	
Cost Structure Turnkey installation, Operations and maintenances for streetlights, installations, Gateway servers and Central server		**Revenue Streams** Annual subscription of municipal corporation, Subscription from traffic signalling and control service, and Subscriptions of advertisers on streetlight installations		

Figure 11.3 Subscription business model canvas for offering Internet of streetlight services using Internet of Streetlights and control of operations with minimum electricity use

Example 11.3

Problem

Suggest a customised business-model canvas for Internet-based railroad predictive analytics and maintenance scheduling.

Solution

Figure 11.4 shows nine building blocks of the canvas for customised Internet-based railroad predictive analytics and maintenance scheduling.

11.2.2 Business Model Innovation

Business model innovation is the development of new and unique concepts supporting an organization's financial viability, including its mission, and the processes for bringing those concepts to fruition. The primary goal of business model innovation is to realise new revenue sources by improving product value, and how the products are delivered to customers.[7]

[7] http://searchcio.techtarget.com/definition/business-innovation

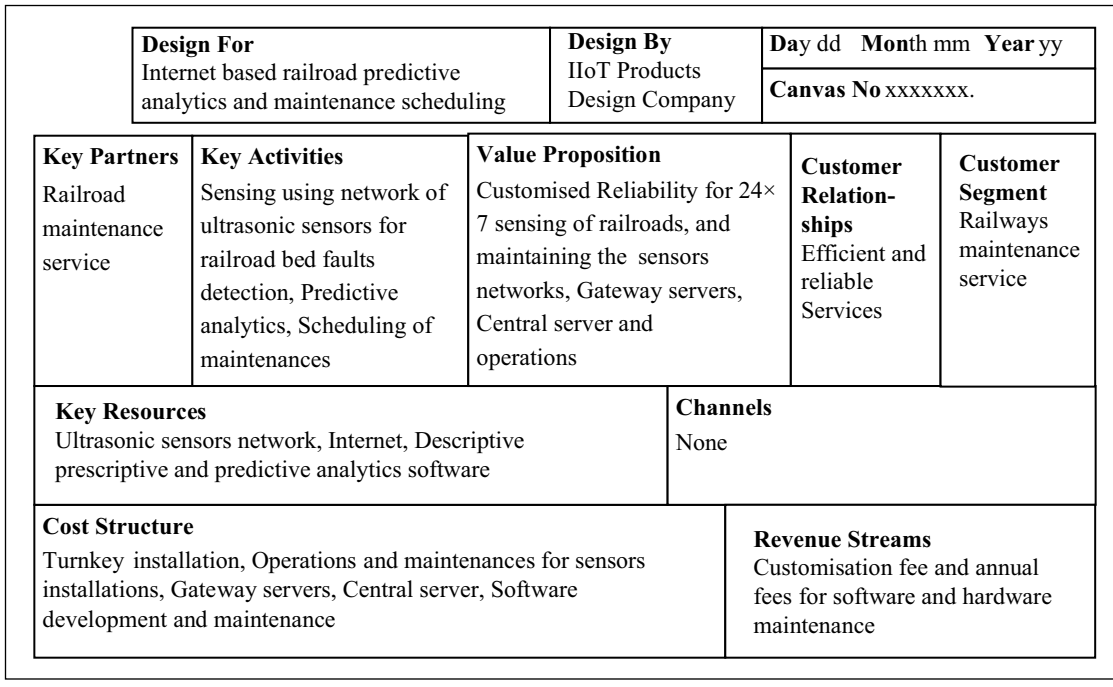


Figure 11.4 Canvas for Business model of Customisation offering Internet based railroad predictive analytics and maintenance scheduling

A business model needs innovation because:

- *New access path* to businesses and direct interaction with the customers have come into existence. The access has become fast and easy. New communication channels are now fast due to use of Internet, more powerful with low cost of computers, mobile communication and cloud computing.
- *Direct interactions* at lower costs require modifications in marketing channels deployed earlier to reach customers. Customers also have faster and multichannel information access and better search options.
- *Business transactions* have become easier and partly automated. Information about customers has become a valuable asset.
- *New price* models need to be worked out in new competitive environment with wider use of ICT.
- *Decentralised models* with the customer at the centre stage are building up and therefore, business models have to be customer-centric. The model should have flexible value chain which reacts fast according to varying customer requirements.
- *Faster introduction of new products and new portfolio strategies* due to better search options with the customers

Business models of an earlier era had innovation as an *interaction* dimension. This has shifted to the *creation* dimension.

Business model innovation is becoming a decisive factor for successful business. Five innovation drivers (Capgemini Consulting) are:

1. Customer neighbourhood production units for fast deliveries and local sales, and customer involvement from the beginning
2. Revenues from new sales instead of maintenance
3. Decentralised production and service
4. Client contribution, open-source design and procurement
5. Significantly reduced capital expenditure

Example 11.4 shows use of IoT leading to business-model innovation.

Example 11.4

Problem

How does use of IoT lead to business model innovation for railroad analytics and maintenance?

Solution

Figure 11.4 showed Internet-based railroad predictive analytics and maintenance scheduling. This example shows a business model innovation with uses of IoT, data analytics and visualisation technologies. Key resources of new business models are uses of an ultrasonic sensors network, Internet, software for descriptive, prescriptive and predictive analytics and visualisation using dices and slices.

Earlier, a business model consisted of unautomated collection of sensors data, no use of Internet, and some use of descriptive analysis of collected data, and analysis of previously observed fault-intervals to design maintenance schedules.

New business model innovation marks a shift to the *creation* dimension by use of key resources shown in the figure. The use of new technologies and communication channels provide fast real-time sensors network, data collection and analytics.

Reconfirm Your Understanding

- A business model is a concept which can be defined in a number of ways. A web source defines a business model as an abstract representation of an organisation. Representation may be conceptual, textual, and/or graphical. Representation is for all core inter-related architectural, co-operational, and financial arrangements. Architecture includes organisational infrastructure and technological architecture. Representation includes many activities of present and future. Representation includes core products and/or services the organisation offers or will offer in future.
- Alexander Osterwalder and his co-workers' business model canvas is a popular way of generating or working on a business model. It can be used as a visual template for developing new or documenting existing business model. The canvas is a visual chart of nine building blocks—key partners, key activities, key resources, value propositions, and customer relationships: customer segments, channels that deliver value proposition to its targeted customers cost structure and revenue stream.
- Examples of business model canvas: Internet of Streetlights and Internet-based railroad sensing and predictive-analytics-based maintenance scheduling.
- Business model innovation is the development of new, unique concepts supporting an organisation's financial viability, including its mission, and the processes for bringing those concepts to fruition.
- Business model innovation must in the new era shift from the interaction dimension to the creation dimension.

Self-Assessment Exercise

1. What is meant by a business model? ★
2. What are the benefits provided by a business model? ★★
3. What are the building blocks of a business model canvas suggested by Osterwalder? ★★
4. What are the key resources in Internet of Streetlight in Example 11.2? ★
5. What are the revenue streams in Internet of Streetlights service because of business model innovation of linking with traffic signalling and control service in Example 11.2? ★
6. What are the revenue streams in case of a customised business model for a sensor network in railroad maintenance Example 11.3? ★★
7. What does business-model innovation mean? ★
8. List and explain five innovation drivers suggested by Capgemini Consulting. ★★★

11.3 VALUE CREATION IN THE INTERNET OF THINGS

LO 11.2 Summarise value propositions and value creation using IoT

Value proposition means producing product or provisioning of a platform or service. For example, using RFIDs in tracking service for the goods is a value proposition. Value creation means creation of a 'smart tracking and logistics service' from the sensed IDs of the RFIDs communication on Internet, data analytics, data visualisations and mobile communication for provisions for SMS to receiver and delivery confirmation to the sender.[8]

Value creation is the expansion of relationships enabled by a disruptor media (Internet) and the creation of new behaviours as a result.

Value creation is the heart of any business model. It involves performing activities that *enhance the value of a company's product or service* (offering) and encourages customer willingness to pay.

Features of value creation using IoT are:

- IoT enables addressing the emergent needs and real-time needs using predictive analytics.

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

[8] https://hbr.org/2014/07/how-the-internet-of-things-changes-business-models

- Information convergence creates new experiences for current products. Information enables innovative services.
- IoT enables offering of product and services which can be updated using the Internet (such through as Wi-Fi) and create synergic value for the product.
- IoT enables value capture and thus recurrent revenue.
- Adds personalisation and context, and uses networked products/services.
- Faster ecosystem functioning where multiple companies establish loose relationships among themselves or establish relationships with big companies.

Value chain means series of actions for value creation. The base of IoT value chain is data collected using APIs for the sensors/sensor networks/M2M data or from multiple information sources. The chain includes actions using web APIs, open data, data from mobile-services network and corporate databases.

Figure 11.5 (a) shows value proposition in production/manufacture-driven value chain Figure 11.5 (b) shows value creation on deploying a value chain using information-driven IoT value chain in production/manufacture/service. Figures reflect analogy between the two.

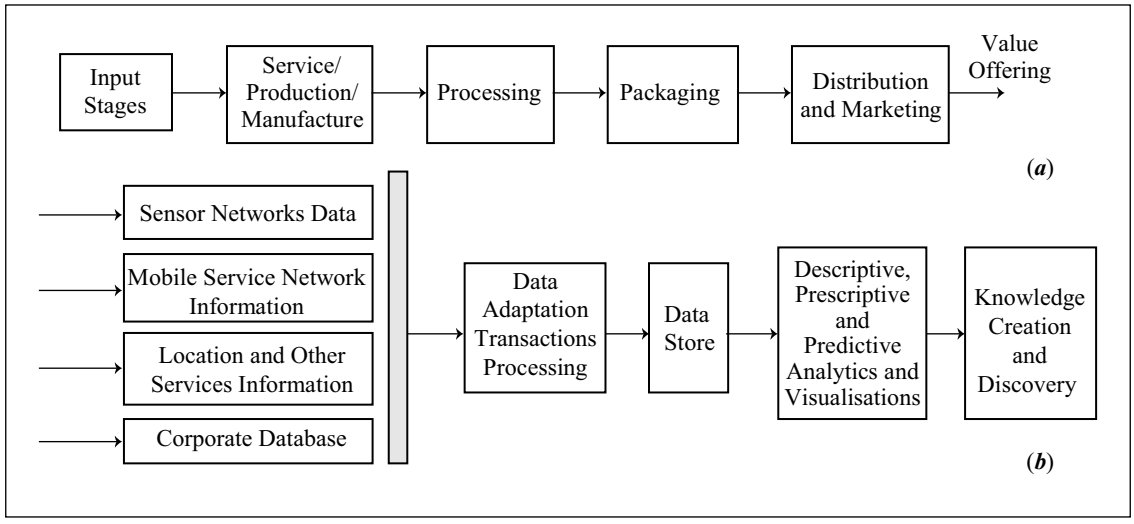


Figure 11.5 (a) Value proposition in production/manufacture-driven value chain (b) Value creation using information-driven IoT value chain in production/manufacture-driven value chain

Example 11.5

Problem

How does Internet of Streetlights create value, expand relationships and help in creating new behaviours when the business model is represented by the canvas at Figure 11.3 in Example 11.2?

Solution

Value created is real-time energy-efficient street lighting and control. The key resources in the model are smart streetlights, sensors network data for surrounding light intensity and traffic presence and density, streetlight functioning status. Traffic presence and density data enable switching off the light in no traffic condition in the vicinity.

The sensor network data expands the relationships to traffic signalling and control service in the city. The relationship creates a new behaviour. Traffic signalling links with the data network. Traffic signalling and control service, control the signalling systems at pathway junctions in the city. The service also generates traffic density reports for GPS navigation and driving in the city.

The subscriptions from the linked services generate additional revenue to the streetlights service company.

Reconfirm Your Understanding

- Value creation is the expansion of relationships enabled by a disruptor media (Internet) and the creation of new behaviours as a result.
- Value creation involves performing activities that *enhance the value of a company's product or service* (offering) and encourages customer willingness to pay.

Self-Assessment Exercise

1. Write the definition of value proposition and value creation. ★
2. List features of value creation using IoT? ★
3. How does information enable innovative services? ★★
4. Explain the analogy between value creation using information-driven IoT value chain and using production/manufacture-driven value chain from Figures 11.5(a) and (b). ★★★

11.4 BUSINESS MODEL SCENARIOS FOR INTERNET OF THINGS

LO 11.3 Explain IoT based business model scenarios

Sensors, M2M, sensor networks data and the data using web APIs for multiple information sources data, open data, mobile services network information data, corporate database and knowledge database are at the input stages. The data from multiple sources and services are part of the key resources in business model scenarios for IoT.

Real-time monitoring applications are used in maintenance scheduling, predictive maintenance, or in fault or anomaly detection. Real-time M2M, IoT sensors and sensors network data can be used to generate events and monitor the systems. Real-time anomalies and fault detection are possible during the service, production or maintenance. Event analytics enables insight into business processes and distributed processes. This enables efficient processes and streamlines existing business processes.

Figure 11.6 shows the real-time monitoring scenario in maintenance scheduling, predictive maintenance or in fault or anomaly detection using IoT.

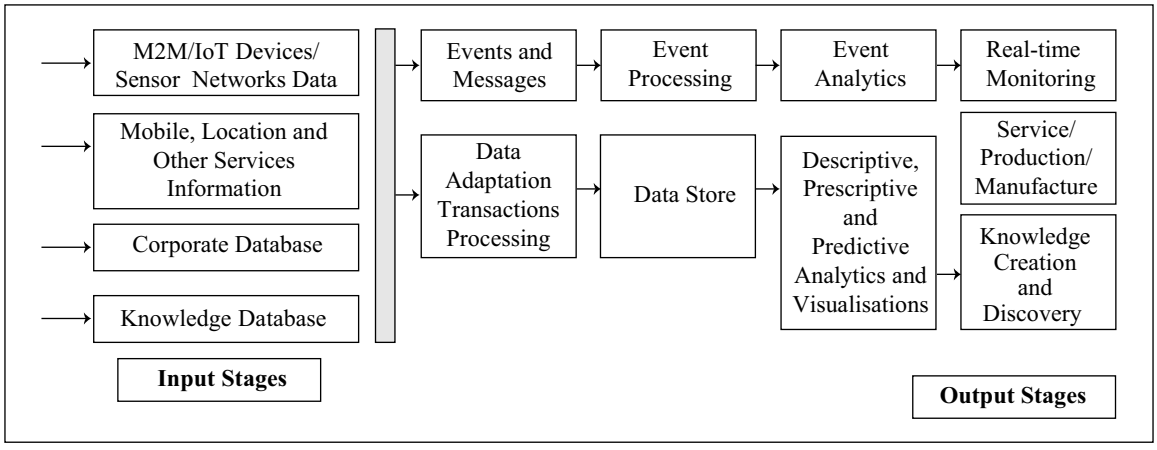


Figure 11.6 Real-time monitoring scenario in maintenance scheduling, predictive maintenance, or in fault or anomaly detection in real-time using IoT

IIoT deploys connectivity to all parts of the production process: machines, products, systems, and people using the cloud and big data technology. The machines and products communicate, and thus manage themselves and each other. Software-based systems and service platforms, data analytics and visualisation, will, therefore, play a major role in tomorrow's manufacturing.[9]

Example 11.6

Problem

How does IIoT deploy connectivity to all parts of the production process—machines, products, systems; and how do the machines and products communicate, and thus manage themselves using data analytics and visualisation in manufacturing of portfolio mix of automobile models?

[9] https://www.bosch-si.com/solutions/manufacturing/data-analytics/manufacturing-analytics.html

Solution

Figure 11.7 shows a coordinated manufacturing and production process for portfolio mix of automobiles with the uses of analytics and visualisation, sale services, customer feedbacks and maintenance centres information.

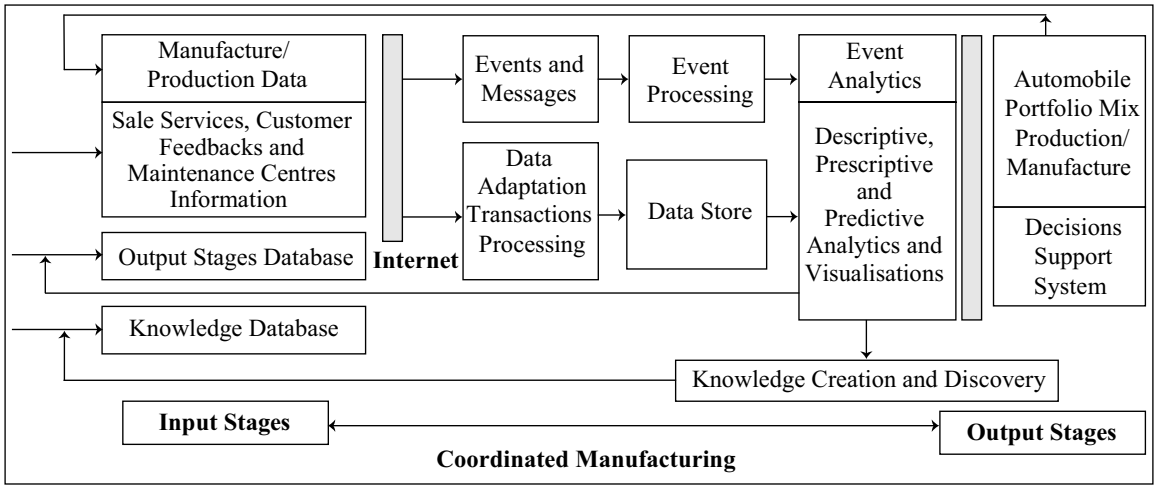


Figure 11.7 Coordinated manufacturing and production processes for portfolio mix of automobiles with the uses of analytics and visualisation, sale services, customer feedbacks and maintenance centres information

When customer choices or sales data alters for specific automobile models then manufacturing unit alters the portfolio mix.

Industry 4.0 (Industries 4.0) is the fourth industrial revolution in which IIoT is an important component. The Industry 4.0 will extensively deploy IIoT and will change manufacturing.[10]

Reconfirm Your Understanding

- IoT-based business model scenarios use sensors, M2M, sensor network data and data using web APIs for multiple information sources data, open data, mobile services network information data, corporate database and knowledge database at the input stages.
- The data from multiple sources and services are a part of the key resources in business model scenarios for the IoT driven value chain.
- Use of M2M data along with the sales, customer and maintenance services data enables coordinated manufacturing and production process for portfolio mix of automobiles with the use of analytics and visualisation, sale services, customer feedback and maintenance centre information.

[10] www.zdnet.com/article/industry-4-0-its-all-about-information-technology/).

Self-Assessment Exercise

1. How does IoT based business model scenarios differ from earlier scenarios? ★
2. How does IIoT facilitate coordinated manufacturing? ★★
3. Why is IoT and multiple sources of data communication on the Internet helpful and how do analytics and visualisations help? ★★★

Key Concepts

- Business model
- Business model canvas
- Business model innovation
- Coordinated manufacturing
- Cost structure
- Customer relationships
- Customer segments
- Industry 4.0
- Internet of RFIDs
- Internet of streetlights
- Key activities
- Key partners
- Key resources
- Knowledge creation
- Portfolio mix
- Predictive analytics
- Real-time monitoring
- Revenue stream
- Sensors network
- Subscription business model
- Value chain
- Value creation
- Value proposition
- Visualisation

Learning Outcomes

LO 11.1

- A business model is a concept which can be defined in a number of ways. A web definition states that business model is an abstract representation of an organisation. Representation may be conceptual, textual, and/or graphical. Representation is for all core inter-related architectural, co-operational and financial arrangements. Architecture includes organisational infrastructure and technological architecture
- Alexander Osterwalder and his co-workers business model canvas is a visual chart of nine building blocks: key partners, key activities, key resources, value propositions, customer relationships, customer segments, channels: to deliver its value proposition to its targeted customers cost structure, and revenue stream.
- Business model innovation is the development of new, unique concepts supporting an organisation's financial viability, including its mission, and the processes for bringing those concepts to fruition
- Business model innovation must, in the new era, shift from *interactions* dimension to the *creation* dimension.

LO 11.2

- Value creation is the expansion of relationships enabled by a disruptor media (Internet) and the creation of new behaviours as a result.

- Value creation involves performing activities that enhance the value of a company's product or service (offering) and encourages customer willingness to pay.
- IoT driven value chain uses sensors data, sensor network data, web services data, multiple source data and analytics, and knowledge discovery tools to add value in the chain.

LO 11.3

- IoT based business model scenarios use sensors, M2M, sensor networks data and the data using web APIs for multiple information sources data, open data, mobile services network information data, corporate database and knowledge database at the input stages.
- The data from multiple sources and services are part of the key resources in business model scenarios for IoT driven value chain.

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. A business model is: (i) an abstract representation; (ii) financial representation of an organisation, representation may be (iii) conceptual, (iv) textual, (v) meta data based and/or (vi) graphical. Representation is for the (vii) selected core inter-related architectural, co-operational and financial arrangements. Representation includes (vii) core products, (viii) future products and/or services. ★
   (a) (i) and (vi)
   (b) All except (i), (ii) and (vii)
   (c) All except (ii) and (vii)
   (d) (iii) to (viii)
2. Business model canvas is a visual chart of the following building blocks: (i) key partners, (ii) key activities, (iii) key resources, (iv) value propositions, (v) customer relationships, (vi) customer segments, (vii) channels which deliver its value proposition to its targeted customers (viii) cost structure, (ix) profit anticipated and (ix) revenue streams. ★
   (a) (i) to (ix)
   (b) All except (i)
   (c) All except (ii) and (vii)
   (d) (i) to (viii) and (x)
3. Features of value creation using IoT are: ★★
   (i) IoT enables addressing to the emergent needs and real-time needs using predictive analytics.
   (ii) Information convergence creates new experiences for current products. Information enables innovative services.
   (iii) IoT enables offering of product and services which can be updated using the Internet (such as through Wi-Fi) and create synergic value for the product.

(iv) IoT enables value capture and thus enables recurrent revenue.
(v) Adds personalisation and context, and uses networked products/services.
(vi) Faster ecosystem functioning where multiple companies establish loose relationships among themselves or establish relationships with big companies.
(a) (i), (iii) (iv) not true
(b) All true
(c) Only (v) and (vi) true
(d) All true except (ii)

4. Innovation drivers according to Capgemini Consulting are (i) customer neighbourhood production units for fast deliveries and local sales, (ii) customer feedback, (iii) new sales revenues instead of maintenance, (iv) decentralised production and service, (vi) client contribution, open-source design and procurement, and (vii) significantly reduced capital expenditure. ★★
   (a) All except (ii)
   (b) All except (iii)
   (c) All except (iv)
   (d) All true
5. Value creation using information-driven IoT value chain consists of the following series of actions: (i) input of location and other services data, (ii) input data of sensors, M2M and sensor networks data, (iii) events and messages, (iv) data adaptation, combination and transaction processing (v) events processing, (vi) data store (vii) descriptive, prescriptive and predictive analytics, (viii) data visualisations (ix) using dicing and slices, and (x) knowledge discovery. ★★★
   (a) (i) to (viii) true
   (b) All except (i), (iv) and (v)
   (c) All except (x)
   (d) All
6. IIoT deploys: (i) connectivity to all parts of the production process (ii) machines, (iii) products, (iv) systems, and (v) people. IIoT business scenario (vi) use the cloud and big data technology. The machines and products communicate, and thus (vii) manage through remote control. IIoT business scenario use (viii) software-based systems and (ix) service platforms. ★★
   (a) All except (ix)
   (b) All except (vii)
   (c) All except (vi)
   (d) All except (v)

Short-Answer Questions

1. What does a business model concept represent? **[LO 11.1]** ★
2. What are the revenue streams in Internet of ATMs? **[LO 11.1]** ★★
3. Why is innovation in a business model necessary? **[LO 11.1]** ★★★
4. What are the building blocks for business finance in the canvas for business model? **[LO 11.1]** ★
5. When is subscription business model useful and when is it not useful? **[LO 11.1]** ★
6. What is the meaning of value proposition, value creation and value chain? **[LO 11.2]** ★

7. List the features of value creation using IoT. **[LO 11.2]** ★★
8. What are the stages in value chain driven by a manufacturing business model and in value chain driven by IoT/M2M devices data and information from multiple sources? **[LO 11.2]** ★
9. Which new business scenarios are emerging from uses of IIoT? **[LO 11.3]** ★★
10. How will industry 4.0 era use the IIoT and coordinated manufacturing and knowledge sharing extensively? **[LO 11.2]** ★★★

Review Questions

1. Show by diagram nine building blocks of a business model canvas. [**LO 11.1**] ★
2. What are the building blocks in Example 11.1 for business model canvas for offering banking services using Internet of ATMs? [**LO 11.1**] ★★
3. Explain canvas for subscription business model offering Internet of Streetlight services using Internet of streetlights and controlling of operations with minimum electricity uses. **[LO11.1]** ★★★
4. Why does a business model need innovation? [**LO 11.1**] ★★
5. What do you mean by value chain? Explain by drawing series of actions for IoT data driven, events and messages driven and information driven value chain. [**LO 11.2**] ★
6. How does IoT enable service, production, manufacture, real-time monitoring and maintenance scheduling in business scenarios? **[LO 11.3]** ★★★

Practice Exercises

1. Draw a business model canvas for Internet-connected automotive for services of automotive service centre for maintenance using predictive analytics in Examples 5.2 and 5.7. **[LO 11.1]** ★★
2. What shall be the cost structure and revenue streams in Internet-connected chain of Automatic Chocolates Vending Machines (ACVMs) and supply chain maintenance services and business activities using predictive analytics in Examples 5.1? **[LO 11.1]** ★★★
3. What are the new business innovations possible using IoT devices data, M2M data and predictive analytics? **[LO 11.1]** ★★
4. Draw a diagram showing input stages, value chain and output stages in Internet of Streetlights as in Example 11.2. **[LO 11.2]** ★
5. Assume a connected chain of Automatic Chocolate Vending Machines (ACVMs) spread all over a city (Example 5.1). Each ACVM delivers a chocolate of user's choice, i.e. among one of the five flavours. A chosen chocolate is delivered on insertion of coins of appropriate amount as per the cost. A display unit displays user interfaces, and takes part in user interaction when a child wishes to buy and get the chocolate of her/his choice. The ACVM also displays advertisements for chocolates, news, weather reports and events in the city whenever not in use. Each ACVM connects to the Internet for its management, and services. How does the Internet of ACVMs create value, expand relationships and help in creation of new behaviours in the business model? **[LO 11.2]** ★★★
6. Explain Figure 11.7. How do manufacturing and production processes coordinate with the use of analytics and visualisation, sale services, customer feedback and maintenance centres information? **[LO 11.3]** ★★★

12

IoT Case Studies

Learning Objectives

LO 12.1 Examine design complexity levels and use of connected Platform-as-a-Service (PaaS) cloud for accelerating design, development and deployment of M2M, IoT and IIoT applications and services.

LO 12.2 Familiarise the design approaches for IoT/IIoT globally trending usages in core business categories/areas—premises, supply-chain and customer monitoring.

LO 12.3 Outline a new concept 'connected car' with the example of Tesla.

LO 12.4 Summarise the design approaches to IoT applications for smart homes, smart cities, smart environment monitoring and smart agriculture using illustrative examples

LO 12.5 Undertake new IoT project design challenges from understanding of a case study, 'Smart city streetlights control and monitoring'.

Recall from Previous Chapters

IoT/M2M/IIoT application examples:

Chapters 1 to 11 described number of examples of IoT—an Internet-connected umbrella; Internet-connected streetlights; Internet-connected RFIDs; Internet of WSNs; Internet-connected projects, such as wearable watch, smart home and smart city; Internet-connected ATMs; Internet-connected chain of Automatic Chocolate Vending Machines (ACVMs); Internet-connected automotive to an automotive service centre for servicing; IIoT technologies for optimising the bicycle manufacturing processes; a railroad maintenance and service centre using Internet-

connected sensors on the rails and deploying IIoT predictive analytics; Internet-coordinated manufacturing and production processes with information from the sale services, customer feedbacks and maintenance centres.

IoT/M2M communication protocols:

Chapters 3 described a connectivity framework for the devices and network domains and layers, and how the connected devices use the CoAP, LWM2M, XML, message queues, MQTT, XMPP, SOAP, RESTful HTTP, WebSockets and other protocols for web connectivity to services, applications and processes.

Chapter 4 described the Internet communication protocols and application-layer port protocols.

IoT/M2M data processing and analytics in the applications and services:

Chapter 5 introduced the terms data acquiring, validating, organising, processing, analytics, visualisation, business processes, processing intelligence, knowledge discovery and knowledge management.

Design using cloud PaaS in an IoT project development and deployment:

Chapter 6 described cloud-based service models, and PaaS as a development platform.

Design of embedded devices with the sensors, actuators, bus communication protocols, RFIDs and WSNs:

Chapter 7 described sensor technology, analog and digital sensors, the use of actuators, use of data communication protocols on buses, ports and interfaces (UART, I2C, LIN, CAN, USB and MOST) and technological aspects of RFIDs and WSNs.

Design of embedded devices with Arduino, Raspberry, Intel Galileo and Edison, mBed or other boards and IDEs:

Chapter 8 described embedded devices hardware and software, popular embedded devices platforms (Arduino, Intel Galileo and Edison, Raspberry Pi, Beagle Bone, and mBed), mobiles and tablets for prototyping and designing for IoT and M2M.

D*evelopment of IoT software using IDEs, IoT stacks, online components and APIs:*

Chapter 9 described the ways of developing software components using IDE. Section 9.3 described Eclipse IoT stack for developing the data acquiring, validating, communicating and gateway software components. Section 9.4 described the online component APIs, web APIs and WebSocket APIs for message exchanges between embedded devices and applications/services on the server/cloud.

IoT security and privacy provisioning software requirements:

Chapter 10 described IoT vulnerabilities, the security requirements, methods of analysis of threats to privacy and security, use and misuse case methods, and the practical aspects of access control, message integrity, non-repudiation and availability. Section 10.6 described security models for IoTs.

12.1 INTRODUCTION

An IoT project design involves a prototype design. A design uses number of technologies due to number of layers through which data communicates, processes and is analysed (Figures 1.2 to 1.5). Production of an IoT system is planned after the prototype is satisfactorily tested, simulated and debugged.

Following are key terms which need to be understood learn project case studies for IoT systems.

Service refers to a set of related software functionalities/components. The set is reused for one or more purposes. The set enables the provisioning of access to one or more capabilities. An interface of a service provides the access to capabilities. The access to each capability is consistent with the constraints and policies, as specified by a service description. Examples of service capabilities are provisioning of security, bank transactions, device IDs management and device or vehicle tracking.

Web service refers to a service using web protocols, web objects or WebSockets. For example, location, weather reporting, traffic density reporting or streetlights monitoring and controlling service.

Cloud computing is a collection of services available over the Internet that deliver computational storage, software or infrastructure functionalities, such as a service provider for connected systems, and for enabling distributed, grid and utility computing.

PaaS is a service model in which the platform is made available to a developer of application, service or process on demand. PaaS developed applications, services and processes which execute using the platform for computing, data store and distribution computing nodes are made available through Internet on-demand. The platform, network, resources, maintenance, updating, and security as per the developer requirements are responsibility of the cloud service provider.

Examples of services are communication management functions, a deployment server for sending and receiving messages from the device agents, diverse sources and devices data store, big data store, server data and DBMS management functions, events processing, message caching and routing, OLTP, OLAP, data analytics for IoT /M2M or other applications/services/business processes, intelligence, and knowledge discovery.

Cloud-connected device platform refers to a cloud services platform for the devices connectivity, control, monitoring, applications, services and other functions.

Cloud-connected Universe Platform (CUP) refers to a cloud services platform for the multi-sources and systems data and devices connectivity, a controls, monitoring, applications, services and other functions. The CUP provisions number of functions and access to connected devices and data sources, such as customers, mobile applications and social network.

Device agent refers to the software which provides the reporting and real-time information to a deployment server (enterprise or applications or cloud server). The agent also refers to software for controlling the device that is installed onto devices/mobile devices. The agent communicates with a control deployment server and carries out the instructions it receives from the server.

Multi-tenant platform refers to a platform that can be used by multiple developers, companies and service deployments. The platform charges recurring expenses as per used/rendered services by the platform.

Domain agnostic platform refers to a platform which functions and renders services on virtually any business domain.

Predictive analytics is an advanced analytics technique that is used to answer the question—What will happen? The answer is based on interpretations of the outputs and results from advanced descriptive analytics, data visualisation and other methods. For example, Caterpillar Inc. has been running over 3 million construction equipment machines across the globe. Predictive analytics of the data over the Internet from the machines predicts and reports the failures to Caterpillar service and maintenance centres.[1] Other examples are predicting (i) expected fault from stress build-up data from sensors placed all along an oil pipeline, (ii) expected component failure from embedded sensors data in a connected car, or (iii) customer behavioural changes and preferences for advance planning of production or logistics.

Data visualisation refers to examining and viewing the results from an analytics process, or process for intelligence or knowledge on a dashboard, infographic, or other applications. Visualisation may be for data dices and slices (Section 5.5.1 and Example 5.10).

Infographic refers to information graphic or graphic visual representation of information, data or knowledge that enables easy viewing or use and clear recognition of view or trends or patterns.[2]

Automated dashboard or infographic refers to creation of dashboard or infographic by software, such as analytics or business process.

Connected car refers to a car that is equipped with Internet access, and usually also with a WLAN. The car shares its Internet access with other devices both on the inside and outside, and is also outfitted with special technologies that tap into the Internet or WLAN, and provide additional benefits to the driver. Another definition from McKinsey for connected car is 'a vehicle which optimises its own operation and maintenance as well as the convenience and comfort of passengers using on-board sensors and Internet connectivity.'[3]

---

[1] http://www.caterpillar.com/en/news/caterpillarNews/

[2] https://en.wikipedia.org/wiki/Infographic

[3] mhttp://www.mckinsey.com/industries/automotive-and-assembly/our-insights/whats-driving-the-connected-car and http://www.autoconnectedcar.com/2014/04/152-million-connected-cars-in-2020-with-14-5-billion-in-revs-very-big-data/

R is a statistical computing and graphics language and environment that includes number of statistical linear and nonlinear modelling, time series analysis, statistical tests, and classifications and clustering.

CVS (Concurrent Versions System) client refers to a software client for revision control for a development software. The client tracks all work and all changes in a set of files that allows collaboration among the developers, separated in space and time.

Git client refers to a built-in open source GUI tool used during browsing and sending a request for a response to a server. A Git-client user gets a platform-specific experience, such as Windows experience.

PyDev refers to a Python IDE for Eclipse code development in Python, Jthyon or IronPython.

Over-The-Air Programming (OTP) means program memory is reprogrammed through wireless transceiver when the functioning of an embedded software circuit needs modification. For example, for extending capabilities or installing a new version or path with the program.

Low Power Wide Area Network (LPWAN) refers to a type of wireless telecommunication network designed to allow long-range communications that communicate at a low bit rate among things (connected objects), such as WSNs and battery-operated sensors. A number of standards are included in LPWAN. Examples are LTE Advanced for Machine Type Communications (LTE-MTC) for connected thing, and LoRaWAN (Low power and range WAN) specification released in 2015. WAN stands for Wide Area Network.

IFTTT refers to free web-service from a company named as IFTTT.[4] Its name is derived from the service which enables a service user to create a sequential set of conditional statements (If This Then That), called applets which trigger actions by change to other web services, such as Facebook, Twitter, Gmail. The web APIs control the actions. A *trigger* corresponds to 'this' and *action* corresponds to 'that'. A *trigger* can provide *ingredient* (basic data, such as picture or e-mail sender address, subject, body, attachment, receive date) for *action*. The *action* task can be communicating content on a social network. Presently IFTTT operating system versions are Android 4.1 onwards and iOS7 onwards which enable services such as smart home controls and automation using mobile phones or tablets.

Section 12.2 describes design complexity levels and design approach of using a connected Platform-as-a-Service (PaaS) cloud for accelerating design, development and deployment of M2M, IoT, IIoT applications and services. Section 12.3 describes the applications of development platforms for IoT/IIoT in globally trending usages in premises, supply chain, customer and product monitoring areas. Section 12.4 describes a new concept 'connected car' with the example of Tesla. Section 12.5 describes approaches to IoT applications for smart homes, smart cities, smart environment monitoring, smart agriculture and smart production using illustrative examples. Section 12.6 describes a case study of smart streetlights in a smart city project.

[4] https://en.wikipedia.org/wiki/IFTTT

12.2 DESIGN LAYERS, DESIGN COMPLEXITY AND DESIGNING USING CLOUD PaaS

LO 12.1 Examine design complexity levels and use of connected Platform-as-a-Service (PaaS) cloud for accelerating design, development and deployment of M2M, IoT, IIoT applications and services.

An IoT prototype developer designs a project in stages. A design has various levels of complexities. A design approach uses connected devices PaaS or connected universe PaaS.

Following subsections describe the prototype design stages, complexities and usages of open source tools and connected PaaS.

12.2.1 Design Layers and Phases during Development and Deployment

Design stages 1 to 3 for a simple IoT system and increasing number of stages from 5 up to 11 for complex IoT/M2M/IIoT applications and services

Equation 1.1 conceptually described a simple IoT system, such as, Internet of Umbrella. The system consists of a physical object or sensor and/or actuator, controller and Internet for connectivity to a web service and/or mobile service provider.

The system needs designing for the following layers:

Layer 1: Physical object(s), sensor(s) and/or actuators (s)

Layer 2: Intranet, Internet or mobile service provider

Layer 3: Controller/monitor

Equation 1.2 (Figure 1.5) conceptualised Oracle architectural model. The IoT system or the product needs designing at the following layers between data gathering from sensors or other data sources and data organising and analytics.

Layer 1: Gather

Layer 2: Enrich + Stream

Layer 3: Manage

Layer 4: Acquire at server or cloud and organise

Layer 5: Analyse + Intelligence

Equation 1.3 (Figure 1.3) conceptualised IBM framework. The IoT system or product needs designing at following layers between sensors or other sources of data and analytics and intelligence.

Layer 1: Gather

Layer 2: Consolidate

Layer 3: Connect + Collect

Layer 4: Assemble

Layer 5: Manage, Analyse

Layer 6: Enterprise integration, Complex Applications Integration and SoA

Figure 7.9 showed participation of four sources of data in the participatory processes in a complex IoT system and product. The IoT system or a product needs designing at the following phases:

Phase 1: Gather data from individual group and collective groups of sensors and data sources, such as CRM, ERP or social media sources through a coordination process between them

Phase 2: Data Capture, Consolidate, Connect + Collect

Phase 3: Data Consolidation

Phase 4: Connect, Data Processing and Validation

Phase 5: Analytics and Data Visualisation

Phase 6: Applications, Services

Phase 7: Applications Integration and SoA

Figure 7.10 depicted IIoT industrial processes groups of sensors and data sources, such as CRM, ERP or social media sources, that participate in a complex IIoT system or product. The IoT system or product needs designing at following phases:

Phase 1–9: IIoT industrial processes and use of groups of sensors and data sources such as CRM, ERP or social media sources for data capturing and gathering, consolidation

Phase 10: Connect, Collect and Data Processing

Phase 11: Analytics and Data Visualisation

Phase 12: Applications and Services

Figure 11.5 depicted a sophisticated IoT system in which data gathers from multiple sources and services, and is part of the key resources. Figure 11.6 depicted a highly sophisticated IoT system for coordinated manufacturing which addresses emergent and real-time needs using predictive analytics and intelligence and where information convergence creates new experiences for current products. Information enables innovative services.

12.2.2 Design Complexity Levels in IoT Prototypes and Product Development and Deployment

Six design complexity levels for IoT/M2M/IIoT applications and services

A complexity level can be assigned with increasing IoT prototypes and product design complexities. Security and privacy are common concerns during the communication in all designs. Six complexity levels can be assigned as:

Complexity Level 1: Internet of Umbrella (Example 1.1) has least complexity in design as it consists of a single physical object, and depends on a single weather web service. Equation 1.1 can be used to represent an IoT system. Internet of other home items in smart homes, such as refrigerator and home lighting networks also have the least complexity in design.

Application or service program and system does not use the cloud or server platform.

Complexity Level 2: Internet of Streetlights (Examples 1.2 and 2.1) has level 2 complexity. Figure 1.1 depicted a group of streetlights connected to a group controller using the intranet. The group controllers coordinate and connect to the Internet and applications and services of a central coordinating server. The IoT system can be conceptualised by Equation 1.2 or 1.3 with little analytics. 'Internet of drip irrigation points in channels' also has the complexity level 2 in its design.

System of complexity level 2 uses the cloud or server database platform for acquiring and organising, an application or service program such as controlling and monitoring is used.

Complexity Level 3: Internet of RFIDs (Example 2.2) has level 3 design complexity. Platform provisions for a number of applications and services, such as tracking, security, inventory control and supply chain management. The IoT system can be conceptualised by equation 1.2 or 1.3 with additional complex analytics and an additional stage for visualisation. Internet of ATMs (Example 2.3) also has level 3 design complexity.

Level 3 complexity system uses a cloud or server platform. The platform acquires and organises enriched data points, event triggers and alerts at a database, analyses the data with analytics and visualises the analysed data.

Complexity Level 4: 'Internet of WSNs' has level 4 design complexity. Smart home and city systems (Section 1.7.2) can be conceptualised by Figures 1.5, 1.11 and 7.9 reference models. The system requires server platform for acquire, organise, analyse and visualise steps of multiple sources data, and connects to a number of networks. 'Internet of rail track sensors' in rail track fault prediction and detection, 'Internet of oil pipeline sensors' and 'Internet of information about weather, environment information, pollution, waste management, road faults, individual and group of people health and traffic congestion' (Section 7.3) also have design complexity level 4.

A level 4 complexity system uses networked devices, coordinators and a centralised server cloud platform. The platform acquires and organises enriched data points, event triggers and alerts at a database, analyses the data with analytics, visualises the analysed data, processes intelligence and provisions for a number of applications and services.

Complexity Level 5: Internet of ACVMs (Example 5.1) which gathers multiple ACVMs data has complexity level 5. 'Internet of Automotive Components and Predictive Automotive Maintenance Application and Service (ACPAMS)' also have this complexity level.

Level 5 complexity system uses multi-input data sources and a cloud platform. The system cloud-server platform acquires, organises, performs data, events, triggers and streams processing and OLAP, visualises multiple sources analysed data and provisions for applications and services. The system extracts intelligence, may deploy machine learning and may perform knowledge discovery and knowledge management.

Complexity Level 6: Industrial IIoT systems and products involve the integration of complex physical machinery M2M and IoT communication. They have design complexity level 6.

Recapitulate Figure 7.10, which shows IIoT phases in the bicycle manufacturing process. Industrial Internet of Things (IIoT) is an application of IoT technology in manufacturing. These systems and products analyse data points, triggers, events and alerts from networked sensors and multiple data sources.

Examples of the functions of IIoT are refining the operations for manufacturing or maintenance, or refining the business model of an industry.

Another example of complexity level 6 system is value creation system using information-driven IoT value chain in production/manufacture-driven value chain (Figure 11.5). Another example is real-time monitoring scenario in maintenance scheduling, predictive maintenance, or in fault or anomaly detection in real time (Figure 11.6).

Complexity level 6 system uses multi-input data sources and multi-server cloud platform. A system cloud-server platform acquires, organises, performs OLTP, events and streams processing and OLAP, visualises multiple sources data and may connect to a number of servers. The system extracts intelligence, deploys machine learning and performs knowledge discovery, management and provisions for numerous applications and services.

A complexity level 1 IoT prototype and product design needs embedded development platform and IDEs (Sections 8.2 and 8.3). Embedded platform enables system hardware and IDE enables software development. The code writing, system design and test of the embedded devices for IoT and M2M are done using the IDEs and development platforms (Section 9.2). Software development for end-to-end connectivity can deploy Eclipse implementations and frameworks included in Eclipse IoT stack (Section 9.3.2).

Complexity levels 2 to 4 IoT prototype and product designs need connected devices and device network Platform-as-a-Service (PaaS) cloud.

Complexity levels 5 and 6 IoT prototype and product designs need connected Platform-as-a-Service (PaaS) cloud, such as TCS Connected Universe Platform (CUP).

12.2.3 Tools, Projects and Platforms

A number of tools and a number of open source tools, projects and platforms are available to the developers of IoT applications and services

Profitbricks describes top 49 tools for IoT project design and development.[5] Postscapes describes a number of open source tools, projects and platforms.[6] Following are examples of four open source tools, projects and platforms available for IoT prototype designs, development and deployment.

Eclipse IoT IDE is most popular Java IDE with excellent user interfaces (UIs), Windows Builder, integration with XML editor, Git client, CVS client, PyDev and other workspace with an extensible plug-in system (Section 9.3). Plug-ins enable development of applications in C, C++, JavaScript, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework) and other programming

[5] https://blog.profitbricks.com/top-49-tools-internet-of-things/
[6] http://www.postscapes.com/internet-of-things-award/open-source/#

languages. It provides multi-projects environment in same window, number of plug-ins and debug viewing.

Oracle IoT development platform refers to provisioning of Oracle Java SE Embedded, Oracle Java ME Embedded, Oracle Java Embedded Suite and Oracle Event Processing solutions that facilitate seamless communications between all elements of the IoT architecture (Figure 1.5). Oracle PaaS is a large and big data platform, which delivers, integrates, secures and retrieves, and analyses comprehensive data from millions of device endpoints. Oracle solutions enable real-time response and data capture. The solutions integrate with the IT systems. Oracle M2M platform offers middleware capabilities and communications between devices.

KaaIoT refers to an IoT development platform which is a multi-purpose middleware platform with inclusion of built-in end-to-end data encryption solutions and software for monitoring, management and configuration of connected devices using communication protocols. It provides an elastic server-architecture deployable at cloud, a fast-growing ecosystem of compatible hardware, analytics and data processing systems, distributed IoT systems, and services which include smart home, connected car, fleet management, operate across a broad variety of devices, and enterprise applications.

Microsoft Research Lab of Things (MSRLT) refers to a platform for innovative solutions, applications and source code samples that have key features of enabling the interconnection of devices, provide HomeOS client side component/source code platform, codes, sensors and devices deployable on Windows-based PC (HomeHub). The platform enables implementation, deployment, monitoring the field studies and analysis of experimental data in areas, such as healthcare, energy management and home automation.

12.2.4 Connected Platform-as-a-Service (PaaS) Cloud

Following are connected devices PaaS and connected universe PaaS.

Xively Connected Devices Platform

Xively PaaS capabilities of elastic and scalable server, management and routing of the messages in real time, and libraries of devices as well as business CRM and ERP objects

Section 6.4 described IoT development and deployment using Xively, Nimbits and others platforms. *Xively* (Section 6.4.1) is PaaS model cloud platform, free for developers. Xively permits many languages and platforms. Xively cloud is elastic and scalable, manages and routes the message in real time, has lifecycle management capabilities, does time series archiving, generates conditional triggers and assigns fine-grain permissions, and enables many devices', provisioning, activation and management. The platform provides a developer-workbench and device-management console.

Xively is a PaaS model cloud platform that permits RESTful API, multiple data formats, including JSON, XML and CSV. Xively enables development using searchable libraries of devices as well as business CRM and ERP objects, clients for iOS, Android, JavaScript and more, and server libraries for programs development Ruby, Python, Java and more.

Nimbits Connected Devices Server and Platform

Nimbits PaaS capabilities of offering distributed cloud and Nimbits providing a downloadable server platform on chips, Raspberry Pi, web server

Nimbits (Section 6.4.2) is a PaaS on a cloud platform (Figure 6.3). Nimbits provides a downloadable server which functions as the distributed-server nodes, Nimbits Server-L and XMPP Server-L. Developer downloads the servers-L onto a chip, Raspberry Pi and web server from the Nimbits cloud platform.

Nimbits provides hardware and software solutions to a developer with an advantage that only application required data, messages, alerts, triggers and notification need to communicate on the Internet between the servers-L node and the Nimbits cloud server-S and XMPP Server-S, as shown in Figure 6.3.

The applications and services communicate with the servers-S. The platform enables connectivity, recording and processing with geo and time-stamping of data and enables the platform communication with the application. The node enables retrieval of large amounts of data, (from devices), event alerts and triggers for the application. The platform also enables complex analytics.

Nimbits PaaS is an open source for IoT applications and services, and offers distributed cloud (Section 6.4.2). Nimbits provides a downloadable server platform on chips, Raspberry Pi, web server compatible with most J2EE servers (such as Jetty Server or Apache Tomcat,) and clouds using Linux, Amazon EC2, Google App Engine.

IBM Internet of Things Foundation Connected Devices Platform

IBM IoT foundation and Bluemix platform capabilities of connect, collect, assemble, manage, analyse and data visualisation deploying IBM Watson analytics

IBM Internet of Things Foundation (IITF) is a fully managed, cloud-hosted service for devices registration, connectivity and control, rapid visualisation and storage of data derived from IoT. Applications can connect using the HTTP API. IITF Node-RED is a visual editor for wiring the IoT application.

IBM Bluemix is a cloud platform. IITF along with Bluemix provides access to the applications to the devices data, fast composition of analytics and visualisation dashboards. IITF features at various stages in IBM conceptual framework (Figure 1.3) are:

- *Connect:* Provides registration, connectivity end-to-end in pub/sub mode using MQTT protocol, and HTTP API for devices remotely monitors the device connectivity using MQTT clients or HTTP APIs, provides secure connectivity of devices to the development boards and cloud, and connects using open-source device code from Eclipse Paho.
- *Collect:* Collect and manage a time-series view of devices data, near real-time IoT data visualisation and OLTP, optimised business results with near real-time decision making
- *Assemble:* Events assembled into logic flows with the help of Bluemix
- *Manage:* Connections and subscriptions to be managed

- *Analyse:* IBM Watson Analytics solution enables analytics, OLAP, predictive analytics and data visualisation with automated dashboard and infographic creation.

CISCO IoT (CIOT) System

The system allows a developer to work in the familiar Linux application environment. The developer can develop the applications in a number of languages, use a number of programming models and open-source development tools.

CISCO IoT, iOx and Fog application platform for capabilities of highly secure connectivity, fast and reliable application integration and functions of controls within the distributed network nodes

CIOT provides the network connectivity, cyber and physical security and data analytics.

Application development platform, CISCO IOx combines IoT application execution within the CISCO Fog applications. IOx technology offers highly secure connectivity, fast and reliable application integration with sensors and cloud, near real-time, automated and high volume of data.

CISCO Fog provides an ecosystem with the ability to transform sensor data and perform the control functions within the distributed network nodes. This enables development of applications such as site asset management, energy monitoring and smart parking infrastructure and connected cities.

AWS IoT Platform

AWS IoT is a PaaS for secure and bi-directional communication software between sensors, actuators, embedded systems and boards, and the AWS Services at the cloud.[7] The platform enables the collection, storage and analysis of data from multiple devices, enabling development of applications that control these devices from the tablets or phones.

AWS IoT device SDK, IOT and services capabilities of collect, store, analyse and using the data from multiple devices and development of applications

AWS IoT is a PaaS that provides the devices connectivity to AWS and other devices. The PaaS enables secure data and interactions, and processing of device data, and enable offline and online interactions of the applications with the devices.

Figure 12.1 shows the components of AWS IoT device SDK (Software Development Kit), AWS IoT and AWS services and data flow diagram and architecture for developers of large number of IoT applications and services using AWS IoT and AWS Services PaaS.

TCUP capabilities of distributed computing and data processing functions and usages of data analytics and visualisation in business processes, intelligence, knowledge discovery and big data enabled architecture

TCS Connected Universe Platform

A Connected Universe Platform (CUP) offers PaaS for connectivity of the IoT/M2M devices as well as customer, mobile apps and

[7] https://aws.amazon.com/iot-platform/how-it-works/AWS IoT Features

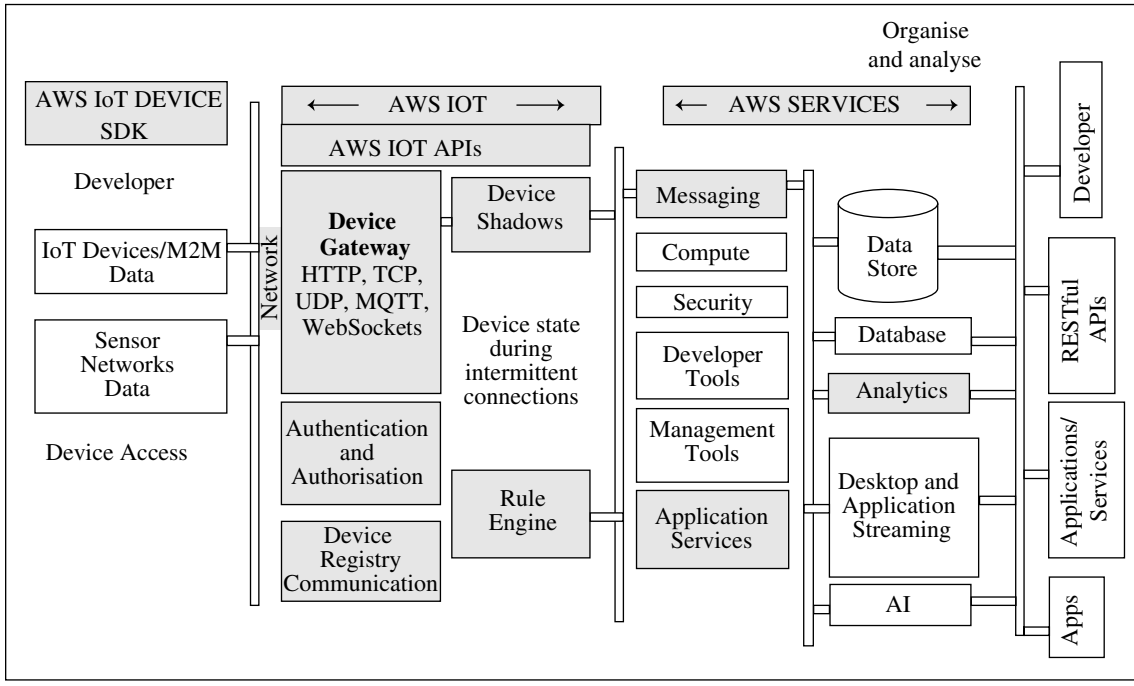


Figure 12.1 Components of AWS IOT device SDK, AWS IOT and AWS services and data flow diagram and architecture for developers of large number of IoT applications and services using AWS IOT and AWS PaaS

other data. CUP also offers connectivity to applications and services. CUP provides the data processing functions and usages of data analytics in business processes, intelligence and knowledge discovery.

TCS offers a Connected Universe Platform (TCUP) which is a highly scalable platform for sensor integration, sensor data storage, analytics (real-time and Big Data processing), rich query capabilities and visualisation.[8] Figure 12.2 shows the CUP architecture and data-flow diagram and architecture when using the TCUP cloud server for PaaS.

Features of cloud-based TCUP are:

1. Offers a scalable cloud PaaS which has secure architecture
2. Offers a domain agnostic multi-tenant platform which optimises the network traffic
3. Gathers, stores and analyses data captured at the embedded sensors, events and diversified sources
4. Enables device management, sensors data acquisition and storage, and analytics
5. Provides Sensor Web Enablement (SWE) services which span sensor description, discovery, integration, sensor observation and measurement capture, storage, and query
6. Can deploy solutions across heterogeneous and interoperable devices, sensors and applications

[8] http://www.tcs.com/about/research/Pages/TCS-Connected-Universe-Platform.aspx

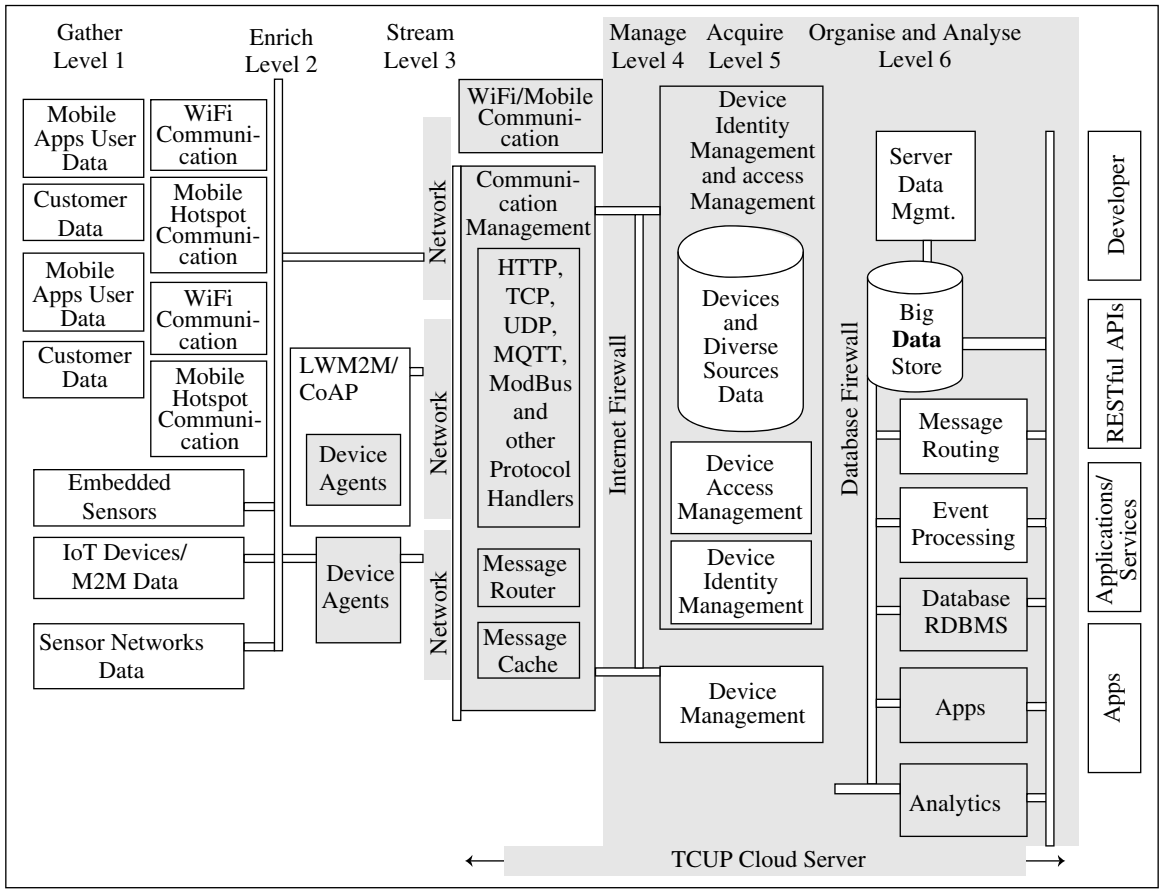


Figure 12.2 CUP architecture and data-flow diagram and architecture when using the TCUP cloud server for PaaS for the developers and large number of IoT applications and services

7. Ensures meeting of future performance needs even as the numbers of devices and users grow
8. Offers device software modules and a set of web services for the application developers to create highly scalable, intelligent and predictive analytics driven IoT applications
9. Allows IoT applications to be built for virtually any devices/sensors and virtually any business domain
10. Integrates IoT data with enterprise systems easily and with ease
11. Makes it easy to develop, deploy and administer IoT, M2M/IIoT software applications
12. Offers a single platform for creating innovative products, intelligent infrastructure, enhancing operational efficiencies, and craft new customer experiences and helping companies offer unique services to their customers
13. Offers remote operations and monitoring
14. Provides big data enabled architecture and distributed computing
15. Offers secure communication and privacy.

Reconfirm Your Understanding

- Six design complexity levels are there in IoT prototypes development and deployment.
- A number of open source tools and connected device platforms are available for IoT prototypes development and deployment
- Oracle IoT platform for Java embedded solutions for the devices and functions for data gathering, enriching, streaming, managing, acquiring, organising, analysing and enterprise and applications integration.
- Microsoft Research Lab of Things for implementation, deployment, monitoring field studies and analyse experimental data in number of areas
- IBM Internet of Things Foundation (IITF) and IBM Bluemix cloud platform connect, collect, assemble and manage; and IBM Watson analytics provide predictive analytics and data visualisation.
- CISCO IoT System application development and deployment platform provides network connectivity, data analytics, and cyber and physical security. CISCO IOx technology offers highly secure connectivity, and fast and reliable application integration with sensors and cloud, near real time, automated and high volume of data. CISCO Fog provides ability to transform sensor data and perform the control functions within the distributed network nodes.
- AWS IoT device SDK (Software Development Kit), AWS IoT and AWS services enable secure data and interactions, processing of device data, and enable interactions of the applications with the devices and AWS services, such as developer, application services, AI, analytics and management tools, database, storage, processing and desktop and application streaming,
- TCS connected universe platform is a cloud PaaS that is scalable and has secure architecture. TCUP gathers, stores and analyses data captured by the embedded sensors, events and diversified sources.
- TCUP based solution can be deployed across heterogeneous and interoperable devices, sensors and application.
- TCUP offers device agents, device software modules and a set of web services for the application developers to create highly scalable, intelligent and predictive analytics driven IoT applications.
- Offers a single platform for creating innovative products, intelligent infrastructure, enhancing the operational efficiencies, and craft new customer experiences and helping companies offer unique services to their customers.

Self-Assessment Exercise

1. List the new features added at each complexity level when design complexity increases. ★
2. What are the capabilities and features of the Oracle IoT/M2M plaform? ★
3. Why do we prefer open source tools for IoT development? What are the limitations of open source tools for IoT development? ★★
4. How do the connective devices PaaS and connected universe platforms help in accelerated project design? ★★★

Note: ★ Level 1 & Level 2 category
★★ Level 3 & Level 4 category
★★★ Level 5 & Level 6 category

5. List the benefits of Xively features of elasticity and scalability, managability and routability of the messages in real-time features. ★★
6. What are the advantages of Nimbits provision of a downloadable platform as the distributed-server node? ★★
7. How is Nimbits distributed-server nodes used for multi-location smart parking applications? ★★★
8. What are the functions of Connect stage in the IBM IoT foundation? ★★
9. How does CISCO IoT, IOx and Fog enable control functions within the distributed network nodes? ★★★
10. How is the smart parking infrastructure application used with CISCO Fog platforms? ★★★
11. How does data flow between AWS IoT, AWS services and applications represent Equation 1.3? ★
12. How does data flow in TCUP represent Equation 1.2? ★
13. How do the components of TCUP function as per Oracle IoT architecture reference model (Figure 1.5)? ★
14. What does creation of 'highly scalable, intelligent and predictive analytics driven IoT applications using TCUP' mean? ★★
15. Draw a table to show a comparison between Nimbits cloud architecture and TCUP architecture. ★★★
16. List the TCUP cloud server sub-units and their usages. ★★★

12.3 IoT/IIoT APPLICATIONS IN THE PREMISES, SUPPLY-CHAIN AND CUSTOMER MONITORING

LO 12.2 Familiarise the design approaches for IoT/IIoT globally trending usages in core business categories/ areas—premises, supply-chain and customer monitoring.

Tata Consultancy Services (TCS) is a world renowned nearly about $16-20 billion annual revenue generating organisation for IT services, consulting and business solutions. TCS made a global trend study 'Internet of Things: The Complete Reimaginative Force' and web-published it in July 2015.[9] TCS categorised IoT/IIoT business usages into *four high-level categories of core business areas* for tracking global trends:

- ***Premises monitoring:*** Examples are Internet-connected cameras, sensors and other devices at banks, ATMs, airports, shopping centres and so on. Example 2.3 describes the architecture of Internet of ATMs in banking applications.
- ***Supply-chain monitoring:*** Examples are Internet-connected RFIDs, sensors, cameras and other devices in the production, distribution and services, and supply-chain order verification, automated reordering and shipping (SCOVARS) *:*
- ***Customer monitoring:*** Examples are Internet-connected digital devices, mobile apps, wearable devices such as wristband for tracking customers behaviours, preferences,

[9] http://www.tcs.com/SiteCollectionDocuments/White%20Papers/Internet-of-Things-The-Complete-Reimaginative-Force.pdf).

locations, systems or components health for applications such as business planning, analytics, health and other services.

- ***Product monitoring:*** Examples are Internet-connected embedded hardware, software, devices network and IIoT technologies which help in product optimisation during the manufacturing process such as bicycle manufacturing process (Example 7.8). Example 11.6 describes IoT applications for coordinated manufacturing and production processes for portfolio mix of automobiles.

Following subsections describe the case studies of projects in premises monitoring system; connected ATMs physical security system; supply chain monitoring; RFID integration into the supply chain order verification; automated reordering and shipping (SCOVARS); and customer monitoring system involving; tracking of customers' carrying Internet-connected digital devices (TCCICDD).

12.3.1 Case Study: Connected ATM Premises Monitoring Project

Steps in designing a premises monitoring system—connected ATMs physical security system

Banks, bank-ATM machines, offices, stores, company's business places, residences, hotels and others need to monitor their premises remotely using sensors, digital cameras and other devices connected to the Internet. An ATM system deploys a number of subsystems. Consider a project of designing surveillance subsystem for each ATM. A design process is divided into a number of steps as given below.

Abstraction

A developer first abstracts the suspicious activity detection as a system event for the activities that endangers the physical safety of the machine and its cash. On the events at the premise, the triggers, messages and data communicates to the banking system in real time using the Internet as a communication channel. The developer abstracts the video clips recorded from the camera as data files from the system that communicate to data store at a surveillance-system server for a group of bank ATMs. A software developer abstracts the hardware as source of events and data files and abstracts software as communicating process for the triggers, messages, data and data files.

Reference Model for Surveillance System

Two domains and their high-level service capabilities in the surveillance system IoT architecture reference model are:

(1) Device and gateway domain deploy digital camera, spatially arranged vibration sensors at an ATM premise devices and media-server gateway for the surveillance. The domain high-level capabilities are:
   - Enriches the sensors and camera data into the events, data files, and processes the events generated, data filters, time stampings and encrypting
   - Media Server Gateway communicates the enriched data deploying the TCP/IP.

(2) Applications and network domain deploys the applications and services and have high-level capabilities as follows—surveillance management functions using the accessed data. The functions include initiation of actions on detection of security breaches at the ATM, and data storage and organising functions for the video files.

Figure 12.3 shows the data-flow diagram and domain architecture reference model for the ATM premise monitoring and the surveillance system management functions and services.

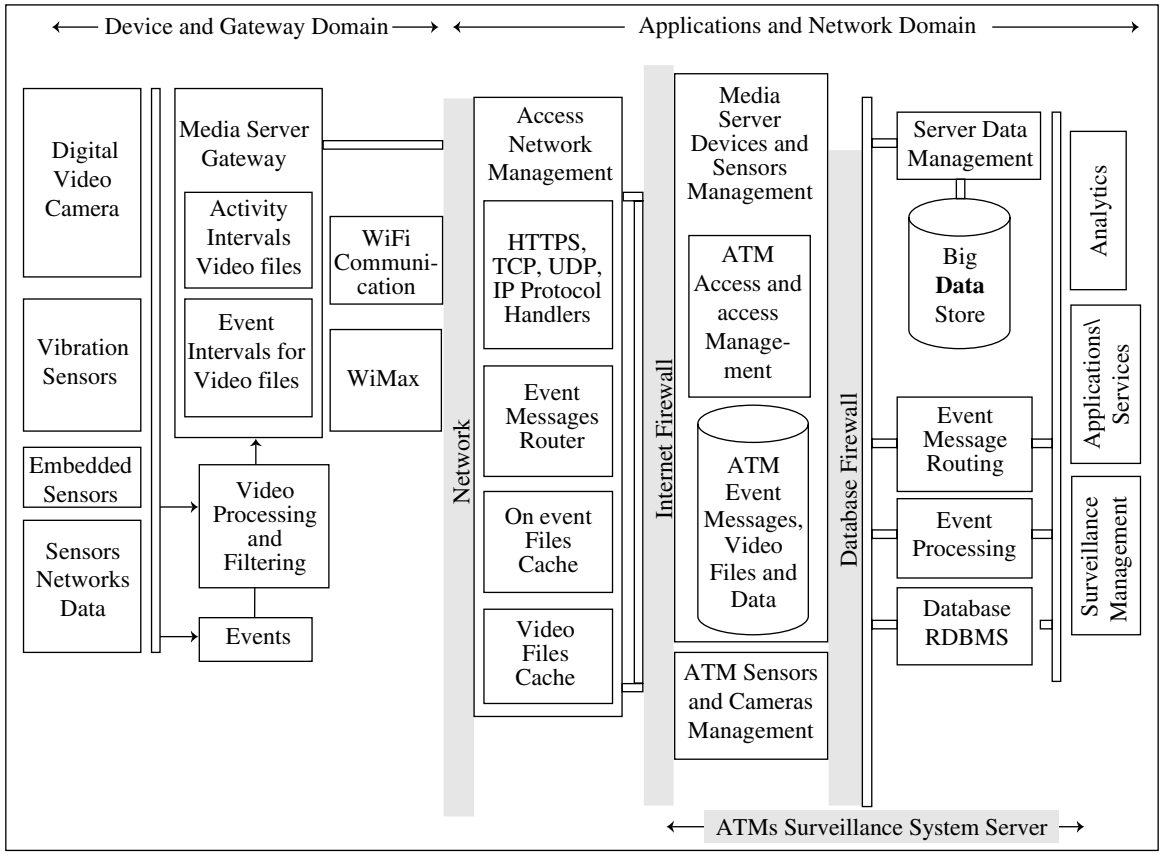


Figure 12.3 Data-flow diagram and domain architecture reference model for the ATM premise monitoring and the surveillance system management functions and services

Identifying Requirements of Device and Gateway Domain

Devices hardware design components are 24 × 7 active digital video cameras; number of spatially-distributed embedded vibration-sensors at the ATM premises; sensors data processing for detection of suspicious activities; video-processing and filtering hardware; communication network connectivity for the events, and on the event, real-time videos and communication network connectivity for the large voluminous data of video clips of transaction activities of users.

Software design modules at the device domain are software components for embedded device such as the distributed vibration sensors processing their data; filtering and extraction of events; communication on the events and on the event videos in real time; embedded camera device video filtering out irrelevant video clips of inactivity intervals data and extracting the relevant intervals video files; and media server gateway for communication of events and on-event video files to the bank server in real time. Large voluminous data of video clips communicate to the data store during periods of inactivity at the ATM.

Identifying Requirements of Network Sub-domain

Network hardware and software design components are Wi-Fi/WiMax access network, core IP network, and surveillance system server. Software design components are network management functions to ensure secure communication network between the device and gateway domain and applications/services.

Identifying Requirements of Application Sub-domain

Software design components are data store organising functions; and surveillance management functions are real-time events processing and 'on the event' video file, activating the alarms (SMS alerts to bank authorities and city police authorities).

Design Implementation of Device and Gateway Domain Hardware and Software

An implementation of ATM premise circuits and embedded sensor devices software needs high computing power for video and multimedia processing. Let us recall Section 8.3.4. Raspberry Pi 2 model B+ (RPi 2) can be deployed for the prototype embedded real-time systems for 24×7 active digital video cameras; spatially-distributed embedded vibration sensors; sensors data processing; video processing and filtering; and communication network.

RPi 2 CPU is quad-core ARM Cortex-A7/ 900 MHz and a graphic processor (Broadcom VideoCore IV) for graphics and video, 1 GB RAM, plus multimedia card module, MicroSDHC card. The RPi 2 provides high-performance computing and graphics.

Section 9.2.3 described the programming of Raspberry Pi platform. AdafruitIDE, a web-based IDE can be deployed for prototype development. Open source ARM videoCore driver enables driving of the camera hardware. Open source Raspbian provisions a Linux distribution using OS.

Section 9.3 described Eclipse IoT stack-based end-to-end IoT solutions with Java and OSGi. A developer can code using following Eclipse stack components for abstractions, software and gateway software in embedded sensors and devices:

1. Eclipse Mihini hardware abstraction and other services in embedded Lua runtime
2. Eclipse.Pi4J for the development of embedded sensor devices software using the framework based on WiringPi and PiFace, Gertboard and other shields
3. Eclipse Koneki functions for development of device applications in embedded Lua language
4. Eclipse Krikkit rules system for configuring the device platforms

5. Eclipse Kura for the Raspberry Pi and the Kura development environment, Gateway. Services, cloud connectivity, management of device, network configuration and applications.

Design Implementation of Application and Network Domain Software

The developer uses OLTP for data store to activate interval time and location stamped (Section 5.4.1) video files. Software developer uses event analytics (Section 5.5.2) for developing surveillance management functions.

Testing and Validation

The implanted hardware/software of both domains needs to be thoroughly tested in the laboratory environment by using sensors and cameras.

12.3.2 Case Study: Connected RFIDs Supply Chain Monitoring Project

Steps in designing a supply-chain monitoring system—connected RFIDs system supply chain order verification, automated reordering and shipping (SCOVARS)

Supply-chain monitoring process is important for companies, distributors and manufacturers. The IoT applications and services consist of Supply-Chain Order Verification, Automated Reordering and Shipping (SCOVARS) operations. The operations involved are planning and scheduling of production, scheduling deliveries, shipping, delivery confirmation from customer, automated reordering from customers, order verification, acknowledgement operations which repeat in each cycle.

An example of applications of SCOVARS is sale of toys, such as LEGO at stores and planning and scheduling of production of toys. Inventory control of each kind of toys uses the RFID labels at the toys, packages and containers for shipping. Whenever a toy sells at a store, such as WalMart, the inventory auto adjusts. Reordering of the toys sold during a week takes place using an app for automated reordering at the customer's end. When a reordering event occurs, then that processes at a server and the event messages route to the ERP applications at the LEGO enterprise.

Section 1.1.3 introduced RFID, its functions, role and uses in IoT applications. Sections 1.5.2 and 7.6.1 explained connected RFID devices and RFID readers at both the customer end and enterprise end. A developer divides the design process into a number of steps as follows.

Abstraction of SCOVARS

Production End (PE) assigns an identity to a root node (level 0) of data tree. Supply End (SE) assigns identities at daughter data-nodes (level 1) for each shipment. Shipping Node (SN) assigns identities at container data nodes (level 2). Shipping End (SHE) consists of identities of group of containers data nodes (level 3). Sales Organisation End (SOE) receiving the group of toy containers consists of identities of containers at data nodes

(level 4). Point of Sale End (POSE) assigns the identities at leaf nodes. A code designer abstracts the PE to POSE end-to-end communication as communication of data trees. Each data tree has data nodes with Universal Resource Identifiers (URIs), messages, alerts and triggers. Data nodes are distributed nodes in the PE-POSE supply chain network.

The code designer abstracts the POSE to PE end-to-end communication as communication of events, alerts, triggers, messages and data files. Each file has reordering information.

Architecture Reference Model for SCOVARS

Designer considers ITU-T four-layers (Figure 2.2) architecture as the reference. The layers have generic management, specific management and management and security capabilities. Four layers for Internet of RFIDs for SCOVARS are:

Layer 1: Device layer capabilities and gateway capabilities are present in RFID physical device-cum-RFID reader, which uses the URI at each node.

Layer 2: Transport and network capabilities use the protocol handlers and Internet connectivity.

Layer 3: Services and application-support layer capabilities at a server node are RFID devices URI registry, access management, URI management and URI time series, server node database, events processing and data analytics.

Layer 4: Services and applications capabilities perform the tracking, plan and schedule production, schedule deliveries, shipping, order verification, and acknowledgement operations.

Design Implementation of Layer Hardware/Software

Arduino board for prototyping the devices between PE and POSE ends

Design implementation and prototyping of the devices between PE and POSE ends needs Arduino-like computing power. Prototype development can be done using Arduino Yun which combines the Arduino-based board with Linux. This is because of the two processors, ATmega32u4 for support to Arduino and Atheros AR9331 for running Linux. IoT application enablers are Wi-Fi, Ethernet support, a USB port, micro-SD card slot, 3 reset buttons and more. The Yun can be controlled from anywhere with any Internet-connected web browser without assigning an IP address to the board. WebSockets can also be used for providing real-time full-duplex communication over TCP.[10]

Refer Section 8.2—Arduino board can be programmed using downloadable tools. Arduino IDE is the tool for the software development of embedded devices that makes application development easy. For example, a software serial library is provided in the IDE for a microcontroller system. The library consists of number of programs. The distinct program exists for each serial interface protocol. The program enables a user to directly use these protocol specific programs, such as using a program for reading an RFID tag or using a program for sending data to USB port for onward transmission on the Internet.

[10] http://asynkronix.se/internet-of-things-with-arduino-yun-and-yaler/

A software serial library is provided in the Integrated Development Environment (IDE) for a microcontroller system. The library consists of a number of programs.

Section 9.3 described Eclipse IoT stack-based end-to-end IoT solutions with Java and OSGi. Software developer can use Eclipse Kura development environment, gateway, services, cloud connectivity, management of device, network configuration and applications.

Design Implementation of Applications in SCOVARS

The developer can use connected devices or connected universe platform. The software requirements are event processing and transaction processing, database functions and event analytics (Section 5.5.2) for the development.

Testing and Validation

The implanted hardware/software at each end-point between POSE and PE needs to be thoroughly tested in a laboratory environment. Arduino IDE is open source and provides embedded hardware and software platform, simulating and debugging.

12.3.3 Case Study: Customer monitoring in IoT Applications/Services Project

Customer tracking and customer database of behaviours, preferences, locations, usage patterns and product health for the innovative applications and services

Data from tracking of customers and customer database provides the behaviours, preferences, locations, usage patterns and product health. Applications such as business planning, analytics, health, services and manufacturing use this data. Consider 'tracking of customers carrying Internet-connected digital devices' (TCCICDD) project for the IoT applications and services case study.

Tracking is done using customer's Internet-connected mobile apps and wearable digital devices, customer databases, customer-end embedded devices and sensors. Information from customer feedbacks, the sale services and maintenance centres also enable tracking. Tracking customers and their information enable creation of innovative products, intelligent infrastructure, enhancing the operational efficiencies, craft provisioning of new customer experiences, and offer unique services to their customers.

An example is usage of customer behaviour, preferences, locations analytics and visualisation for sale services, customer feedbacks and preference for creating information for controlling supply chain and boosting sales.

TCCICDD design process can be divided into a number of steps as follows.

Abstraction for TCCICDD

A designer abstracts the sensors and devices data as device messages. The designer abstracts the customer data for TCCICDD as devices messages, events, alerts and triggers. The sensors are put at the places of customer's visits, such as malls and company sales centres.

The messages include customer ID, location and time stamps. The designer abstracts contents as devices database. The database deploys the time-series information of the customer feedbacks, the sale services and maintenance centres.

The data store at the server for time series and location-stamped customer data, customer database and network accessed data, messages and events.

Communication gateway and the Internet abstract as network. Network consists of communication between the customer data, customer database and device messages and a connected universe or other PaaS.

Architecture Reference Model for TCCICDD

Let TCCICDD project designer deploy Oracle reference model architecture (Figure 1.5) for the applications of IoTs in services and business processes. Five layers in project reference model are as follows:

Layer 1 (Gather): A Mobile app of the company and apps at wearable and other devices connected devices send customer data to the gateway for change of locations, product health, preferences, usages of the product and feedbacks. Customer database contents are ID, account number, options, age, name and other information. The layer gathers the customer data, customer database, sensors and device data and sends that to layer 2 gateway. The software at embedded devices, sensors at malls and other public places also gather and send the data to gateway.

Layer 2 (Enrich and Stream): Gateway enriches the data by generating time series and location-stamped data and adapting for communication using IPv4 protocol on the network. Gathered and enriched data stream to the server using the network.

Layer 3 (Manage): Communication management functions access and perform ID management, data and messages routing and caching functions.

Layer 4 (Acquire and Organise at server or cloud): Devices and diverse sources data store and acquires the data routes. Data is organised as big data store and database RDBMS.

Layer 5 (Analyse + Intelligence): Applications perform analytics and data visualisation, and extract intelligence

Layer 6 (Enterprise Integration, Complex Applications Integration and SoA)

Figure 12.4 shows data-flow diagram and architecture for information, customers' data, IoT/M2M devices, sensors using a connected devices PaaS and TCCICDD applications and services.

Identifying Requirements at Six Layers at TCCICDD

Requirements at six layers for TCCICDD are as follows:

Layer 1 (Gather): The applications and services install at the mobile and wearable devices embed hardware and software for gathering the customer data. An embedded device

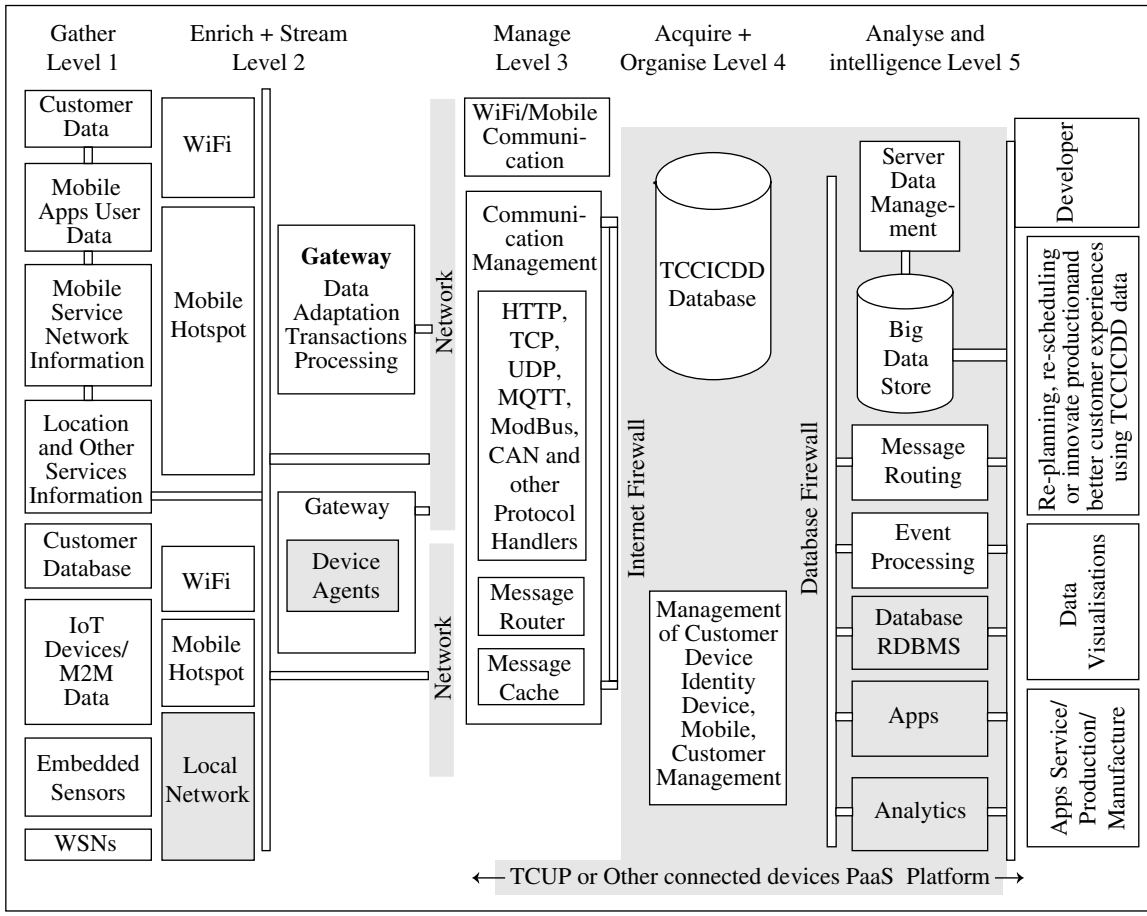


Figure 12.4 Data flow diagram and architecture for information, customers' data, IoT/M2M devices, sensors using a connected devices PaaS and TCCICDD applications/services

gathers the customer data for acquiring at server database. The embedded sensors and device hardware and software gather the data and communicate that to the gateway.

Layer 2 (Enrich and Stream): Gateway software and processor accesses the data of layer 1 and enriches the data by generating time series and location-stamped data and adapts the data for sensing it to the network. Network card or shield streams to the server using network protocols.

Layer 3 (Manage): Communication management functions does access and ID management, customer data and messages routing and caching.

Layer 4 (Acquire and Organise): Server data store acquires the customer, events and data of devices and diverse sources.

Layer 5 (Analyse and Intelligence): Data is organised as big data store and database RDBMS. Data is analysed using event processing, message routing and analytics

Layer 6 (Enterprise Integration, Complex Applications Integration and SoA): Applications perform analytics and data visualisation, and extract intelligence for service, production and manufacturing, re-planning, re-scheduling or innovating the production and providing better customer experiences using TCCICDD data.

Design Implementation of Hardware, Software, Applications and Services

Design implements layer 1 using mobile apps and embedded sensors and devices software. Design of sensors/devices needs Arduino and Raspberry Pi like computing power depending on the sensor. Example 9.12 explained Internet-connected smartphones for weather and location services using the APIs for web applications. Similarly a mobile app tracks the location data of the customer.

A program sends data to USB/MMC port for onward transmission on the Internet. Section 9.3 described Eclipse IoT stack-based end-to-end IoT solutions with Java and OSGi. A software developer can use Eclipse Kura development environment, Gateway. Services, cloud connectivity, management of device, network configuration and applications.

Project developer can use TCUP or other PaaS for communication management. Applications developer uses TCUP or other server platform data store, database, events and streams processing, OLTP and OLAP software, database functions, event analytics, analytics and data visualisation, and extract intelligence for service, production and manufacturing, re-planning, re-scheduling or innovating the production.

Reconfirm Your Understanding

- Premises monitoring deploys a surveillance system, which communicates source of events, triggers, messages, data and data files between the ATMs and the bank server.
- Reference model architecture for ATM surveillance system can be considered as ETSI two-domains model—device and gateway, and network and applications domains.
- A design process has number of steps, viz. abstraction, design of hardware and software in a reference model, identifying requirements of embedded hardware and software modules for the domains, design implementation for the domains hardware and software, and testing and validation.
- Prototype development board for premises monitoring system can be Raspberry Pi 2 Model B+. The board includes quad-core ARM Cortex-A7/ 900 MHz CPU, a graphic processor (Broadcom VideoCore IV) for graphics and video, 1 GB RAM, multimedia card module. OS Raspbian Ubuntu distribution of Linux, Snappy) enables programming for secure running of autonomous machines, sensors and video cameras.
- Developer implements the design for domains hardware and software deploying Eclipse Stack, events and streams processing, OLTP and event analytics.
- A supply-chain monitoring system is SCOVARS. Architecture reference model can be IETF/ITU-T reference model for RFIDs supply-chain applications for planning and scheduling production, scheduling deliveries, shipping, order verification, acknowledgement operations and customer applications for delivery confirmation from customer's and automated reordering.
- SCOVARS uses RFID devices, readers and gateway, network, server and applications.

- SCOVARS design process and prototype development has number of steps—abstraction, design of hardware and software in a reference model, identifying requirements of layer 1 embedded hardware and software modules and layer 2 to 4 software, design implementation for the domains hardware and software, and (v) testing and validation
- Arduino board suits prototype RFIDs at the customer and company ends. Arduino IDE and tools enable programming for a user to directly use these protocol-specific programs, such as using a program for reading an RFID tag or using a program for sending data to a USB port for onward transmission on the Internet.
- A developer can implement the design for the hardware and software using the IDE and Eclipse Kura module in Stack.
- Developer uses event processing and OLTP software, database functions and event analytics for the development of applications at the company end.
- Arduino IDE enables embedded hardware and software platform, simulating and debugging.
- An architecture based on Oracle reference model is used for TCCICDD.
- A customer monitoring system is TCCICDD. It uses mobile apps, embedded sensors/devices, gateway, network, server and applications.
- Applications for data visualisation and service, production or manufacture re-planning, re-scheduling or innovating the production and providing better customer experiences using TCCICDD data.
- TCCICDD design process and prototype development have a number of steps—abstraction, design of hardware and software in a reference model, identifying requirements of layers 1 embedded hardware and software modules and layers 2 and 3 gateway software and processor, design implements using a server platform, and testing and validation.
- A developer can implement the design for hardware and software using the IDE and Eclipse Kura module in Stack.
- The project uses TCUP or other server platforms for communication management, data store and database.
- A developer can use TCUP or other server platform, event processing and OLTP software, database functions and event analytics for the development of applications for a company using TCCICDD data.

Self-Assessment Exercise

1. Make a table of abstraction of the hardware/software units in surveillance system in Internet of ATMs. ★
2. What are the hardware and software capabilities in ETSI architecture device and gateway for surveillance system for Internet of ATMs? ★
3. What are the hardware and software in ETSI architecture network and applications domain for surveillance system in Internet of ATMs? ★★
4. List the functions in Eclipse IoT stack which can be used in the surveillance system. ★★★
5. Why does Raspberry Pi 2 Model B+ suit the prototype development of ATM devices, sensors and cameras? ★★
6. List the functions of each component and module in ATMs surveillance system server. ★★★

7. Make a table of abstraction of the hardware/software units in supply chain monitoring system SCOVARS. ★
8. What are the hardware and software requirements in IETF architecture layers 1 and 2 for SCOVARS at the company and customer ends? ★
9. What are the functions for the application-support and applications layers 3 and 4 for SCOVARS? ★★
10. List the functions of IDE needed at layer 1 in SCOVARS. ★★★
11. Why does Arduino Yun suit the prototype development of RFIDs layer 1 for SCOVARS? ★★
12. List the functions at layer 3 for SCOVARS server. ★★★
13. Make a table of abstraction of the hardware/software units in TCCICDD. ★
14. What are the hardware and software requirements in Oracle architecture layers 1 and 2 for TCCICDD for customer ends? ★
15. What are the functions for the layers 2, 3 and 4 for TCCICDD? ★★
16. How do the applications use the data gathered at layer 1 for TCCICDD? ★★★
17. Why do mobile apps and embedded devices/sensors suit the prototype development of TCCICDD for layer 1? ★★★
18. List the functions at layers 5 and 6 for TCCICDD server. ★★

12.4 CONNECTED CAR AND ITS APPLICATIONS AND SERVICES

LO 12.3 Outline a new concept 'connected car' with the example of Tesla

Following subsections describe the connected car. The in-car components network among themselves. The car connects to the service centre, manufacturing units, traffic-monitoring systems and other services.

12.4.1 Connected Car

Connected in-car components network and Internet-connected cars enabling number of services

Recall Examples 5.2 and 5.7. Internet-connected cars enable the services of automotive service centres and predictive maintenance using predictive analytics.

Section 12.1 introduced the concept of a connected car. Figure 12.5 shows an overview of in-car subunits in a connected car.

In-Car ECUs Cluster Network

A connected car generates data using in-car Electronic Control Units (ECUs) cluster consisting of digital embedded devices/product heath devices/sensors. The cluster uses Bluetooth and NFC for inter-devices wireless communication, Controller Area Network (CAN) and Local Interconnect Network (LIN) for wire communication. Multimedia devices use Media Oriented System Transport (MOST).

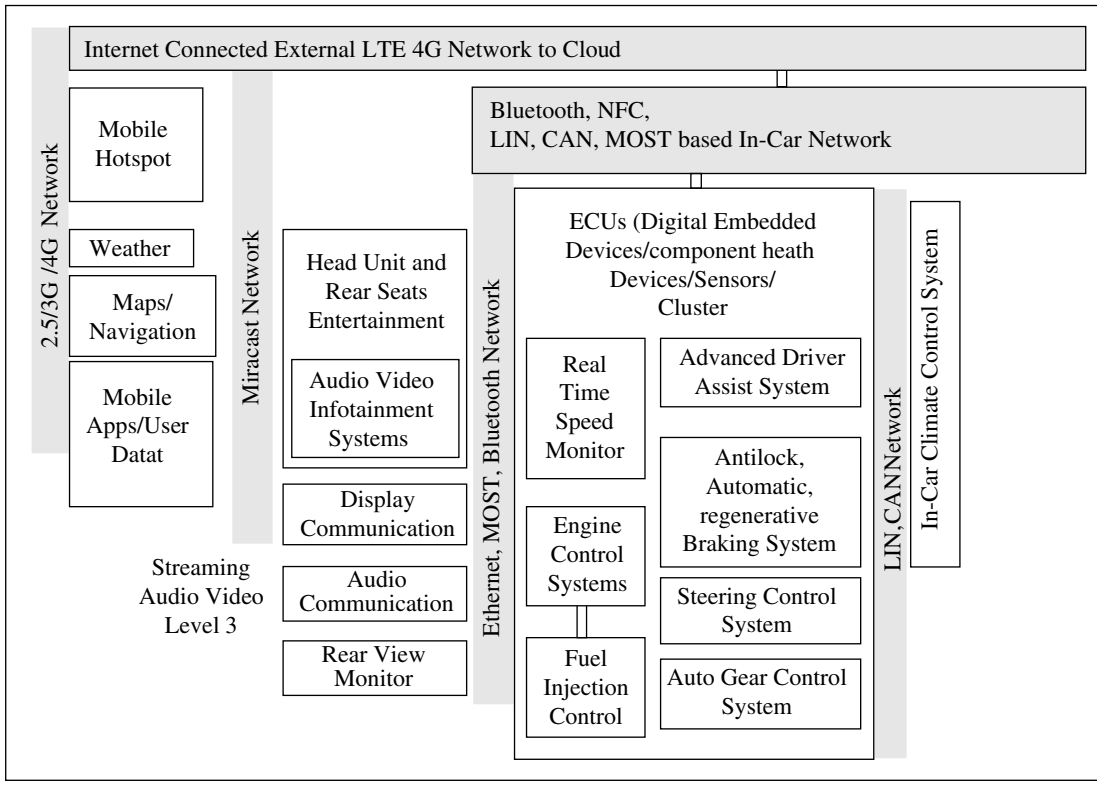


Figure 12.5 Overview of an Internet-connected car

Cluster

The cluster consists of the following systems:

- Engine control
- Speed control and brake system, antilock braking, automatic braking and regenerative braking
- Safety systems
- Seat and pedal controls
- Car environment controls: Car climate and light controllers control the air-conditioning, heater, ventilation, windows, light and temperature (ACHVWLT).
- Automobile status monitoring: Automobile status monitoring, tyre pressure, real-time vehicle mileage, data loggers and driving assisting devices management, steerable head lights, windshield IR camera, windscreen heads up display, night-vision assistance, and so on.
- System interfaces for commands, voice activation, and Advanced Driver Assistance System (ADAS) and interfacing (System interfaces are soft programmable buttons, voice commands, and wireless personal area connectivity in the automobile).

Route and Traffic Monitors

Route and traffic controls and monitoring are done by mobile API based car location and surrounding area maps, cached traffic reports for real time traffic monitoring, guided route programming and route planners.

Infotainment Systems

Infotainment systems are as follows—displayed text-to-speech converters. Bluetooth connected music player, VCD/DVD, car mobile phones, audio CD player, LCD screen, touch panel screen, Miracast devices for streaming Internet radio and video. Auto-managed infotainment features.

Ethernet Connectivity

Next generation in-car network uses Ethernet bus and Ethernet audio-video bridge. The bridge is between the connected number of subsystems and units. The examples of usages of bridge are connectivity of other in-car networks with the head unit and rear-seats entertainment, driver information centre, advanced diver assistance system, and around view monitoring and gateways at ECUs.

Internet Connectivity

A connected car uses 2.5G/3G/HSPA+/4G or Wi-Fi network for connectivity of weather, maps, navigation and mobile apps and user data. Navigation is using data feed of near real-time traffic conditions to the car. Maps APIs may use Google maps. Location APIs use real-time GPS locations from location services. Internet connectivity can not only be through mobile data services of service providers but also from Wi-Fi nodes installed at the junctions in smart cities, where the network stacked for the cloud platform transmits from a nearby connected car in vicinity. Latest generation connected cars use 4G LTE for streaming audio-video (Internet radio and mobile TV), and for multimedia over the air updates for their in-car infotainment system.

Connected car IoT enables seamless connectivity of 'in-car' systems with the Internet, driver assist/alerts, infrastructure usages, open source infotainment audio-video updates, and the company computational platforms. The Internet-connected car analytics enables diagnostics, prognosis, car maintenance and company services, comfort and convenience.

12.4.2 Automotive Components Predictive Automotive Maintenance Service (ACPAMS) and Re-planning manufacturing process (RPMP)

Two applications of the connected car are:

1. Automotive Components Predictive Automotive Maintenance Service (ACPAMS): Optimally-preparing and more effective maintenance, automatic detection of service requirements by direct transfer of service-relevant data to car maintenance and service centre, and driver/car user server and automatic reminder of servicing appointments shorten the service visits.

2. Re-planning manufacturing process (RPMP): The data are the customer ID and location and time stamped data as follows:
   - In-car network data from ECUs Cluster of digital embedded devices/ car-health devices/sensors
   - In-car climate control system
   - Streaming audio video
   - Maps, navigation, weather, mobile apps and user data

Applications and services ACPAMS and RPMP

ACPAMS is a car maintenance service. A developer can design a number of service centre applications, such as remote diagnostics, roadside assistance, car malfunction and car-location-based emergency services and reporting for the service-centre support, remote care, fleet management, fuel and eco-initiatives. Connected-car advanced analytics with in-depth views including car-health monitoring, car-health trend analysis, condition-based maintenance triggers, detecting dependencies amongst car-health problems, depreciation analysis and driving behaviour analytics, analysis, detecting patterns of dependencies among vehicle health issues, location-based geospatial analysis.

RPMP enables re-planning, re-scheduling or innovating the production and manufacturing of automotive components and the car. RPMP also enables design for better customer experiences using the organised and analysed data.

Figure 12.6 shows data-flow diagram and architecture of connected car using TCUP for system designing, developers, manufacturing units and maintenance service units and IoT applications/services.

Identifying Requirements at Six Layers at ACPAMS and RPMP

Requirements at six layers for *ACPAMS* and *RPMP* applications and services are:

Layer 1 (Gather): The apps install at the mobile and wearable devices embed hardware and software for gathering the car location, weather, traffic, navigation and car-health data. ECUs cluster gathers the data using buses. The embedded sensors and device hardware and software gather the data and communicate that using the bus.

Layer 2 (Enrich and Stream): Software at car central computer access the data of layer 1. The data collects using CAN, MOST, Ethernet and Miracast Wi-Fi devices and display board. Software enriches the data by generating time series and location-stamped data and adapts the data for sending it to the network. The central-computer has smart gateway for ECUs cluster and provides cryptographic authentication, integrity and confidentiality functions for car cluster security.

A car may use Miracast devices Wi-Fi alliance certified software implementation and Wi-Fi display technical specification. Wi-Fi alliance defines Miracast usages protocol to provide 'HDMI over Wi-Fi', replacing the cable from the computer to the display and in-car infotainment systems. Adapters plug into HDMI or USB ports that enable non-Miracast devices to connect using the Miracast.

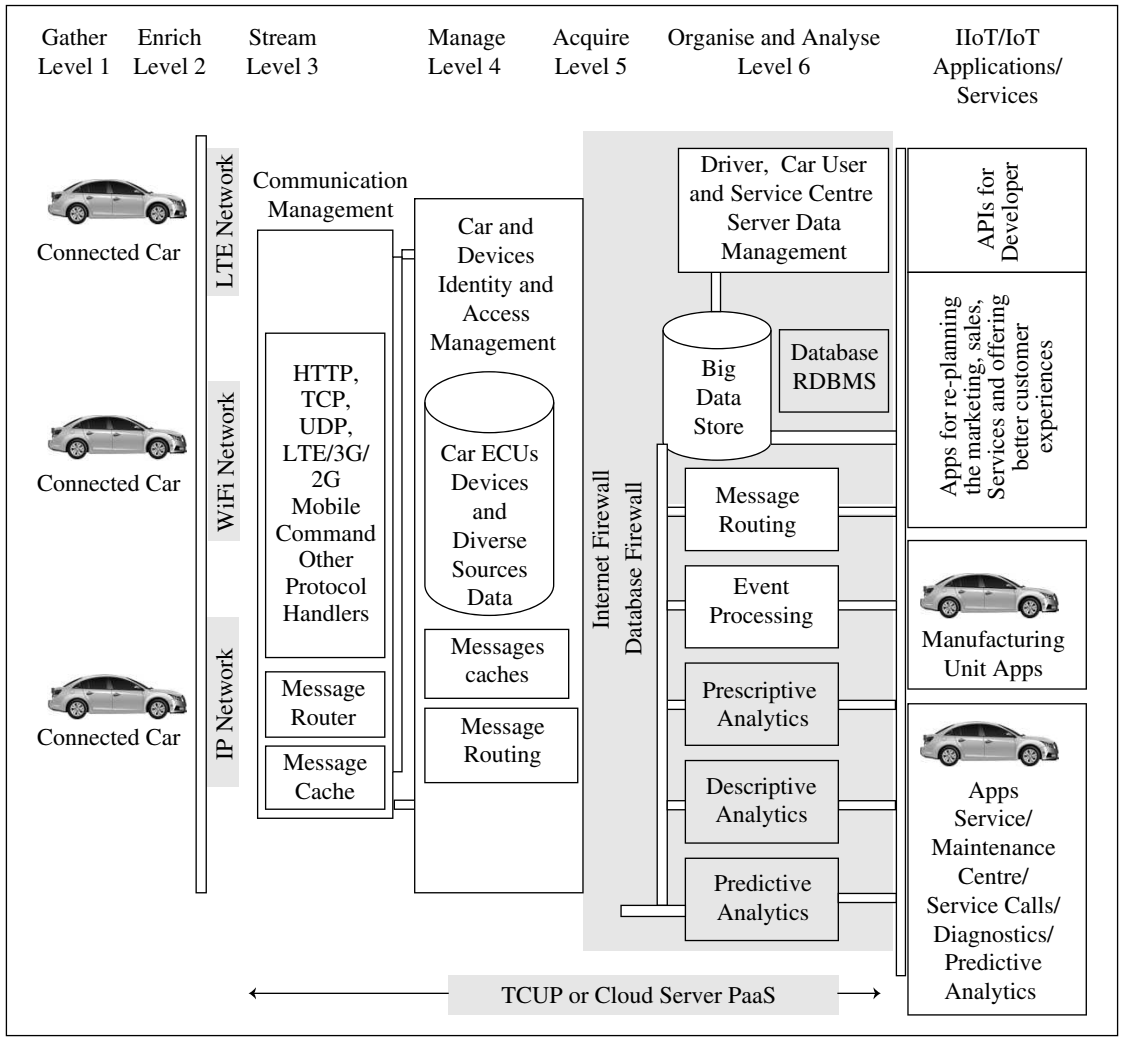


Figure 12.6 Data-flow diagram and architecture of connected cars using TCUP for system designing, developers, manufacturing units and maintenance service units

The in-car computer streams data to the network using mobile services or 2.5G/3G/ HSPA+/4G or Wi-Fi network protocols.

Layer 3 (Manage): TCUP or a cloud server PaaS consists of communication management functions for the accesses, and ECU cluster, infotainment and other systems management, data and messages routing and caching.

Layer 4 (Acquire and Organise): The platform acquires the ACPAMS and RPMP data, events and data of devices and diverse sources.

Layer 5 (Analyse and Intelligence): The platform organises data as big data store and database RDBMS. Data is analysed using the event processing, message routing, analytics and AI.

Layer 6 (Enterprise Integration, Complex Applications Integration and SoA): Services and applications run on the processors for the service, production and manufacture, data visualisation, re-planning, re-scheduling or innovating the production and better customer experiences using ACPAMS, RPMP and services data.

Design Implementation of Layers 1 and 2 Hardware/Software

Design implements layer 1 using mobile apps and embedded sensors and devices software. Design of ECUs/embedded devices/sensors/car-components health devices need Arduino AVR or ARM-based boards with the sensors. Infotainment systems prototype can use Raspberry Pi 2 Model B+ or BB boards. The boards provide high computing power, graphic, video and audio processing needs. A program sends data to USB/MMC port for onward transmission on Internet.

Eclipse IoT stack-based end-to-end IoT solutions with Java and OSGi: A software developer can use Eclipse Kura development environment, gateway. Services, cloud connectivity, management of device, network configuration and applications.

Design Implementation of Layers 3, 4, 5 and 6

The project uses TCUP or other server platform for communication management, data store, and database.

TCUP or Other Cloud Server PaaS Platform

The IoT/IIoT data stores at TCUP or other PaaS cloud. The data are used for number of purposes such as service centre maintenance, re-planning the marketing, sales, service and product development functions, manufacturing processes, and personalising and deepening customer experiences and by developing additional value-added services.

12.4.3 Tesla Car with Semi-Autonomous Drive Assist

A Tesla car is electric car running on lightweight lithium battery with more than 200 km range with autopilot mode. Autopilot refers to semi-autonomous Advanced Driver Assist System (ADAS). Tesla car has ADAS, twelve ultrasonic sensors in the front and rear bumpers, eight radar cameras mounted at the top of the windshield and forward looking radar. Tesla car hardware provides full self-driving capability at a safety level (SAE Level 5).

ADAS automates, adapts and enhances automobile systems for safety (SAE level 5) and better driving with features, such as incorporate GPS and traffic warnings and or alert about the blind spots.

Safety features refers to collisions avoidance, accidents prevention, automate braking, and taking over control of the vehicle.

The in-car system alerts the driver about potential problems, to other cars or dangers, and keeps the driver in the correct lane.

The Tesla system operates in a shadow mode which means processing without taking action but for sending data back to Tesla production unit to improve the car abilities and functionalities. Later the software can be upgraded over-the-air. Earlier, the car did not have automatic emergency braking and other features. Tesla expects full self-driving by 2017 end.

Reconfirm Your Understanding

- IoT applications and services enable car maintenance service, service centre applications, for example, remote diagnostics, roadside assistance, car malfunction and location car based emergency services and reporting for service centre support, remote care, fleet management, fuel and eco initiatives.
- Internet connected car has in-car network of ECUs. The architecture can be based on Oracle reference model for ACPAMS, RPMP and other services.
- Connected car generates data using in car ECUs cluster consisting of digital embedded devices/component health devices/sensors
- LIN, NFC, CAN, MOST, Bluetooth, Ethernet and Miracast protocols enable in-car connectivity.
- Connected car uses mobile apps for maps, navigation, location data and wearable devices or embedded health devices for health data. Connected car communicate to Internet using 4G LTE/3G/2.5 G mobile data or Wi-Fi over IP network.
- A central in-car computer enriches the generated data. In-car infotainment systems use Miracast devices for in-car network.
- TCUP or other PaaS cloud enables the developer to develop new applications.
- The platform runs the applications for—maintenance by Service Centre; re-planning the marketing, sales, service and product development functions, manufacturing processes; and personalising and deepening customer experiences and by developing additional value-added services.
- Tesla is an electric car with autopilot mode. It will possess full self-driving capability with safety in future.

Self-Assessment Exercise

1. Make a table of abstraction of the hardware/software units in ACPAMS and RPMP. ★
2. What are the hardware and software requirements in Oracle architecture layers 1 to 6 for ACPAMS and RPMP for connected-car end? ★
3. Why do Arduino, Raspberry Pi Model B+ and mBed boards suit the development of the embedded devices/sensors for prototype development of ACPAMS and RPMP for layer 1? ★★
4. List the functions at layers 5 and 6 ACPAMS and RPMP PaaS (Figure 12.6). ★★★
5. List the applications a developer uses for the car maintenance service centre and driver/car user server databases. ★★
6. How do the car ECUs, devices and diverse sources data enable RPMP? ★★★

12.5 IoT APPLICATIONS FOR SMART HOMES, CITIES, ENVIRONMENT-MONITORING AND AGRICULTURE

LO 12.4 Summarise the design approaches to IoT applications for smart homes, smart cities, smart environment monitoring and smart agriculture using illustrative examples

Smart homes, smart cities, smart environment monitoring and smart agriculture are illustrative examples of IoT applications and services. Subsections below describe design approach for designing the projects in these areas.

12.5.1 Smart Home

Home automation enabling on deploying openHAB, a Java and OSGi services based open-source software development platform

Section 1.7.2 introduced the smart home concept and listed a number of smart home services, such as home lighting control, control and monitoring of appliances, security, intrusion detection, video surveillance, access control and security alerts, Wi-Fi, Internet, and remote cloud access, control and monitoring.

Smart-home Devices Development and Deployment using an Open-source Software

All smart home devices can communicate using openHAB.[11] HAB drives from Home Automation Bus. The developer deploys Java and OSGi services (Section 9.3.2) and uses the open source development environment and deployment platform. Its accompanying cloud platform[12] my.openHAB provides communication between that with the cloud. The my.openHAB cloud-connector also provides REST (Section 3.4.4) and cloud-based services, such as IFTTT (Section 12.1). The operating system versions Android 4.1 onwards and iOS7 onwards for IFTTT, enable services such as smart home controls and automation using mobile phones or tablets. OpenHAB computing environment is Java. GUI clients are designed and can be used as downloads from git.[13] IDE, guidelines, bindings for code development are provided for openHAB.[14] Figure 12.7 shows architectural layers in openHAB development environment.

A *service* in figure refers to service capabilities, which can be called upon whenever needed. The figure shows the following:

1. Core openHAB objects—REST service and repository; base library
2. openHAB add-on objects—Item provider, protocol bindings, automation logics, user interfaces and libraries
3. OSGi framework services—Configuration administration, event administration service, declarative services, log back, runtime and HTTP services

[11] http://www.openhab.org/
[12] https://my.openHAB.org
[13] https://git-scm.com/downloads
[14] http://docs.openhab.org/developers/developers/development/ide.html, /development/guidelines.html, /development/bindings.html

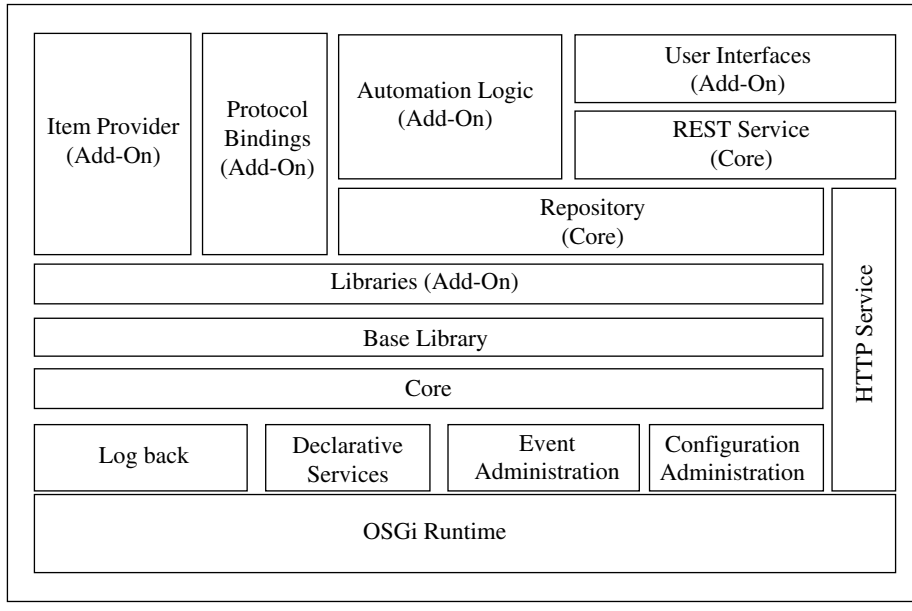


Figure 12.7 Architectural layers in openHAB development environment

4. OpenHAB deploys event administration service of OSGi with pub/sub mode.
5. A stateful repository is for querying and for use by automation logics. Some functions are stateless and do not depend on previous action(s). Remaining actions are stateful and dependent on previous chain of actions. State of items in repository is as per the actions.

Two domains and their high-level service capabilities in the home automation system in IoT architecture reference model are:

1. *Device and Gateway Domain*: Assume that the system deploys *j* lighting devices, each with a proximity sensor. *Automation logic* provides that if no change is found in proximity due to presence of person(s) then the devices switch off. Assume that the system also deploys *k* intrusion sensors and *l* appliances. Automation logic provides on intrusion, communicate trigger(s) to a local or remote web-service as per configuration setting at the configuration administration service of OSGi framework.
2. *Application and Network Domain*: Applications and network domain deploys applications and services and have high-level capabilities.

Domain Architectural Reference Model

Figure 12.8 shows the data-flow diagram and domain architecture reference model for home automation lighting, appliances and intrusion monitoring services.

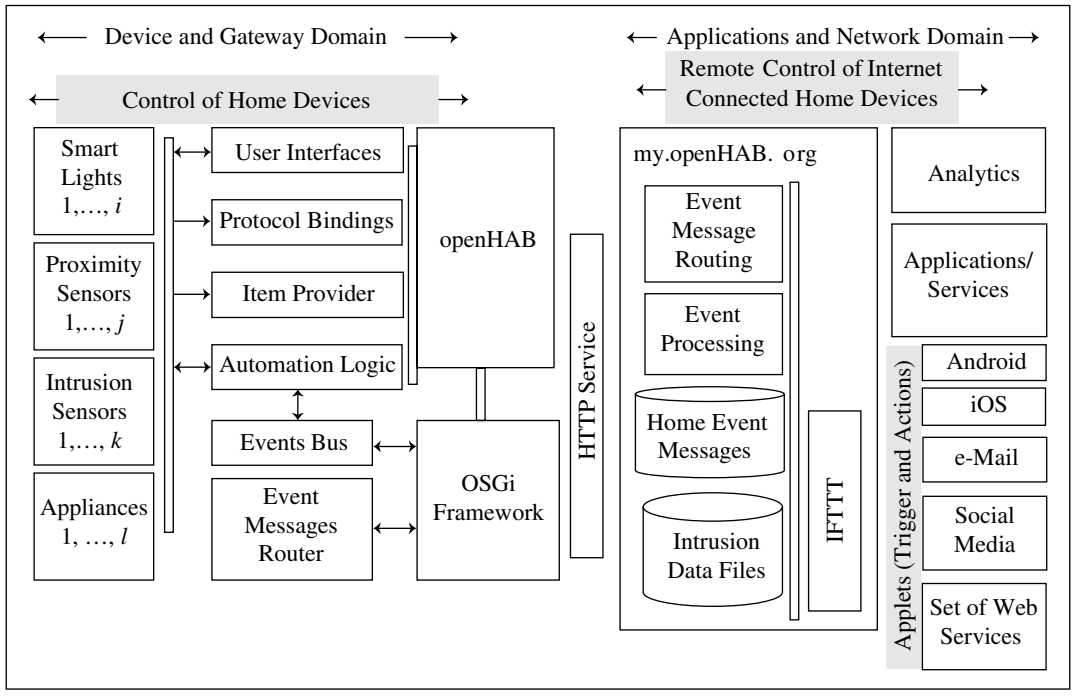


Figure 12.8 Data-flow diagram and domain architecture reference model for home automation lighting, appliances and intrusion monitoring services

Figure shows that openHAB has an event bus. The bus is asynchronous. The event bus refers to a communication bus for all protocol bindings. The bindings link to the hardware. The event bus is the base service of openHAB. Example of the event is command, which triggers an action or a state change of some item or device. Another example of event is status update which informs about a status change of some item or device. For example, in response to a command.

The openHAB service is integration-hub between such devices and bindings between different protocols used for networking the home devices, OSGi and HTTP service. Usually just one instance of openHAB runs on a central coordinator (computer) at home. Event Administration Service of OSGi service is used for remote services. Several distributed openHAB instances can connect and deploy the event bus.

Code Development Environment, Development, Debugging and Deployment

The openHAB development uses the Eclipse IDE (Section 9.3.1) for Java Developers. Development environment can be selected as Windows (32 or 64), Linux or Mac. The Eclipse Installer adds all required plug-ins, provides encoding settings for workspace, pre-configured code formatters and automatically prepares the IDE. Workspace compiles and provides runtime by opting" openHAB_Runtime" configuration.

The openHAB platform is neutral to hardware and interfacing protocols. For example, a security camera device may be on Raspberry platform and lighting devices on Arduino. The automation logics can connect different systems. Bus systems, devices and protocols have dedicated bindings to openHAB. Each binding User Interface (UI) design can have unique look and feel. A binding sends and receives the commands, and status updates on openHAB event bus.

The openHAB brings together different bus systems, hardware devices and interface protocols. The dedicated bindings facilitate this. The openHAB solution aims for provisioning a universal integration platform for the devices. The codes are written in Java and are fully based on OSGi. The Equinox OSGi runtime and Jetty as a web server build the core foundation of the runtime.

Devices hardware design components are 24 × 7 active digital video cameras for intrusion detection, number of spatially-distributed embedded proximity sensors, the home premises; sensors data processing for detection of suspicious activities; video-processing and filtering hardware; communication network connectivity to the events bus, and on the event, communication network connectivity.

Software design modules at the device domain are software components for embedded device distributed proximity sensors data processing; filtering and extraction of events; and communication on the events, and media server gateway for communication of events.

Identifying Requirements of Network Sub-domain

Network hardware and software design components are Wi-Fi/WiMax access network, core IP network, and server. Software design components are network management functions to ensure secure communication network between device and gateway domain and applications/services.

The openHAB cloud connector connects the local openHAB runtime to a remote openHAB cloud, such as my.openHAB, instance from openHAB foundation.[15]

Design Implementation of Device and Gateway Domain Hardware and Software

Section 7.2.2 described the sensors which can be deployed for a number of home automation applications. An implementation of home premise intrusion circuits and embedded sensor devices software needs high computing power for intrusion detection. Section 8.3.4 described Raspberry Pi 2 model B+ (RPi 2) which can be deployed due to high computing power.

An implementation of home premise lighting and appliances embedded sensor devices software needs computing power for lighting automation and Arduino or RPi boards can be deployed.

The openHAB can be used for end-to-end solutions for smart home applications and services. Section 9.3.2 described Eclipse IoT stack-based end-to-end IoT solutions with

[15] https://github.com/openhab/openhab/wiki/openHAB-Cloud-Connector

Java and OSGi. A developer can code using Eclipse stack components for abstractions, software and gateway software in embedded sensors and devices. Sample projects using openHAB are given at Github.[16]

12.5.2 Smart City

Smart city secure deploying multiple ICT and IoT solutions, such as smart parking, traffic solutions, street-lighting, health and waste management services

Smart city applications and services connect people, process, data and things. A smart city can be defined as a vision which integrates multiple ICT and IoT solutions in a secure fashion to manage a city's assets such as information systems, schools, libraries, transportation systems, hospitals, power plants, water supply networks, waste management, law enforcement and other community services. Sectors that have been developing smart city technology include government services, transport and traffic management, energy, health care, water, innovative urban agriculture and waste management.[17]

Smart-city solutions integrate a number of city services and can include the following[18]:

1. Smart parking spaces
2. Smart street lightings and smart lighting solutions, such as SimplySNAP smart lighting solution which enables the development, control, and optimisation of a smart lighting implementation, developed by Synapse Wireless partnering with ThingWorx
3. Smart traffic solutions, such as smart energy management, smart parking, smart waste bins, smart street lighting, and security and surveillance, developed by Tech Mahindra partnering with ThingWorx
4. Smart water management, such as AquamatiX for monitoring and optimizing a city's water and sewage services
5. Smart connected bike share services
6. Smart health services
7. Smart structures (building, bridges and historical monuments) health, vibrations and material conditions monitoring, analysing and managing structures health data to improve energy usage, maintenance, operations, and comfort solution. For example using solutions developed by WiseUp partnering with ThingWorx platform
8. Smart city system integrator, such as Pactera from ThingWorx

Architectural View

Figure 1.12 shows four-layer architectural framework developed at CISCO cloud IoT for a city. Layers consists of (i) devices network and distributed nodes, (ii) distributed

[16] https://github.com/openHAB/openHAB/wiki/Projects-using-openHAB
[17] https://en.wikipedia.org/wiki/Smart_city
[18] https://www.thingworx.com/about/news/ptc-demonstrates-industry-leadership-smart-cities-smart-city-expo-world-congress-2016/

data capture, processing and analysing, (iii) data centres and cloud and (iv) applications, such as waste containers monitoring. Figure 12.9 shows data-flow diagram and domain architecture reference model for the smart city applications and services.[19]

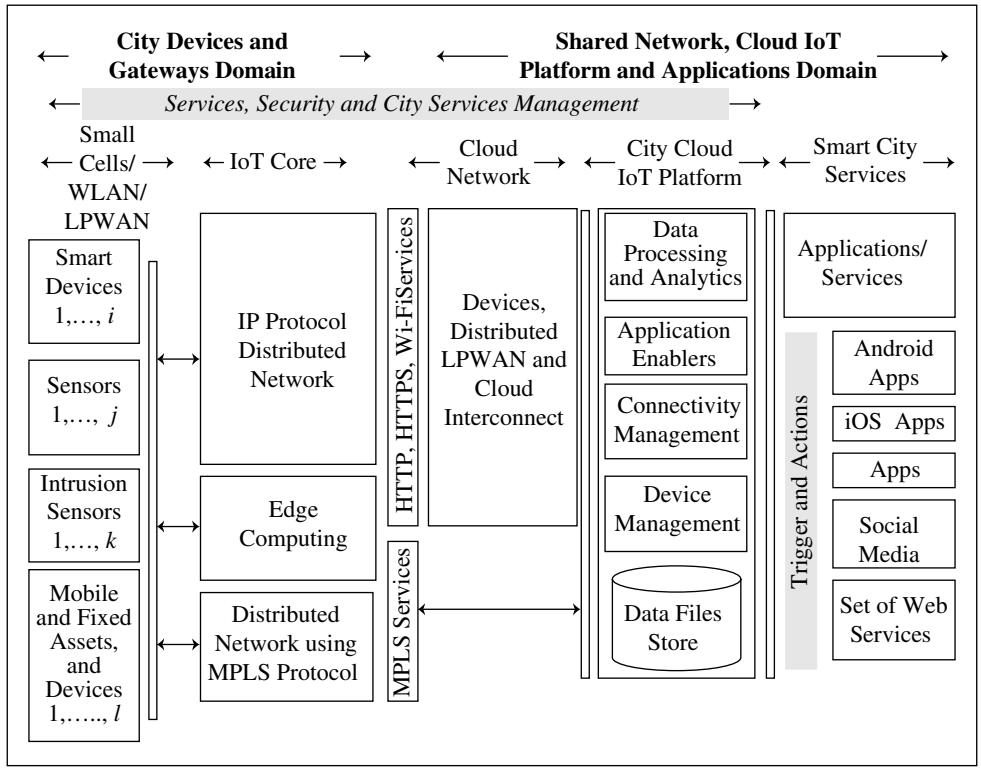


Figure 12.9 Data flow diagram and domain architecture reference model for the smart City Applications and Services

Two domains are (i) City devices and Gateways domain, and (ii) shared network, cloud IoT platform and applications domains. Services, security and city services management are cross-domain functions. Assume the edge sensors and devices consist of, say *i*-smart devices, *j*- sensors, *k*-intrusion sensors and *l*- mobile and fixed assets and devices whereas *i*, *j*, *k* and *l* can be very large numbers.

The edge sensors and devices wirelessly connect within small cells, systems connect with WLAN (Wireless LAN). They communicate using LPWAN. The distributed network of edge-computing systems connects using IP protocols or using the Multiprotocol Label Switching (MPLS). The MPLS assigns the labels to data packets and forward the labels to city cloud IoT platform.

City cloud IoT platform collects messages, triggers, alerts and data files at data store. The platform does device and connectivity management functions, application enabler

[19] https://networks.nokia.com/smart-city

functions, and data processing and analytics. The platform generates triggers which follows actions, such as connect to social media, set of web services, applications, iOS apps and Android apps. The platform connects to a number of city applications and services.

Smart city applications and services can deploy CISCO IoT, IOx and Fog. This is because of usages of shared networks and distributed access point nodes, and the need of an ecosystem with ability to transform sensor data and perform the control functions within the distributed network nodes. This enables development of applications, such as site asset management, energy monitoring, and smart parking infrastructure and connected cities.

Alternatively, smart city solutions can deploy ThingWorx IoT platform, which is an intelligent management platform for all connected things (IMPACT). IMPACT platform provides devices and connectivity management, application enablement, data and analytics with secure end-to-end access.

Sensors, Devices, Hardware Edge Systems Deployment

Section 7.2.2 described the sensors which can be deployed for a number of city applications. Section 8.3 described embedded platforms Arduino, Edison, RPi, BeagleBone and mBed which can be used for prototype development of edge systems.

An open-source prototyping platform, openHAB and hardware RPi or Arduino can be deployed at the edge systems. Bosch presented at CES 2017 a prototyping platform for IoT.[20] The platform connects the devices and provides a number of solutions. Edge hardware is Bosch XDK 110 development kit, which includes devices [eight number microelectromechanical system, accelerometer, magnetometer and gyroscope, coupled with relative-humidity (RH), pressure (P), temperature (T), acoustic and digital light-sensors], functional extendibility with a 26-pin extension port, Bluetooth and Wi-Fi connectivity.[21] XDK workbench includes its IDE.

Code Development Environment, Development, Debugging and Deployment

Section 9.3.2 described end-to-end IoT Solutions with Java and Eclipse IoT Stack. Section 12.5.1 described openHAB software platform for devices, sensor and edge systems at a smart home. Java, Eclipse IoT Stack and openHAB can be used for end-to-end solutions for smart city applications and services.

XDK workbench is for code development environment for the prototype kit XDK 110. The workbench includes the IDE, software for all system component drivers, a debugger port, and high and low level APIs for sensors. The bench also includes libraries for sensor readings, CoAP and LWM2M client and server, MQTT client and broker using Eclipse Paho connector, and HTTP client request and response server. Also included are virtual SDK applications for virtual mobile iOS and Android GUIs for monitoring and control of the sensors and WLAN and Bluetooth LE protocol stacks. XDK provides access to XDK developer community for online technical support. The community offers platform for discussions of ideas, exchange of information and innovation.

[20] http://www.eetimes.com/document.asp?doc_id=1331136
[21] http://xdk.bosch-connectivity.com/overview

Smart City Parking

A growing problem in cities is of vehicular traffic congestion and parking spaces. A modern city, therefore, provides a number of multilevel parking spaces which spread all over the city. A driver needs a mobile app. Significant fuel saving can result from provisioning of smart parking spaces in a city.

A smart parking-service should enable the following:

1. Guides the drivers for the available parking slots and spaces
2. Provides a mobile app, and the app assists a driver and enables him/her to obtain the appropriate parking-slot information remotely. The information includes location of the parking utility, its cost, advance reservation facility, direction guidance and the time to reach an available slot. The app accesses the slots availability in real-time data in pub/sub mode.
3. Publishes messages in real time for available slots and alerts for slot unavailability at the parking utility
4. Consists of a central supervisory control and monitoring system (CSS) which connects the edge sensors and devices, accurately senses the slots available for occupancy of vehicles in real time, and predicts the expected availability time in case of non-availability of slots
5. Optimises the usages of parking spaces and reaching time
6. Provides display boards at road traffic junctions for status of availability
7. Provides good parking experience to users
8. Adds value for all parking stakeholders, drivers and service providers
9. Enables intelligent decisions using data and historical analytics reports at city cloud data store, and enables planning for traffic flow in the city by predictive analytics

Sensors play vital role in the smart parking. The application is ranked as topmost among 50 sensor-applications for a smarter world.[22]

Domain Architecture Reference Model

Figure 12.10 shows data-flow diagram, domains and architecture reference model for smart parking applications and services.

The figure shows four layers at two domains. Parking spaces are at layer 1. They are sensed using coordinators at each level in multilevel parking spaces. An actuator for the light at each slot is used. Lighting control module at the coordinator actuates the parking space lights. The lightings can be switched on and off as per requirement for each space.

A Parking Assistance System (PAS) is at layer 2. This includes CCS and three modules for monitoring, control and display. Layers 1 and 2 communication protocols are ZigBee, LWM2M and UDP.

The CSS maintains a real-time database of time-series data of the parking spaces. The system connects layer 3, which includes the SMS gateway and cloud IoT platform.

[22] http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking/

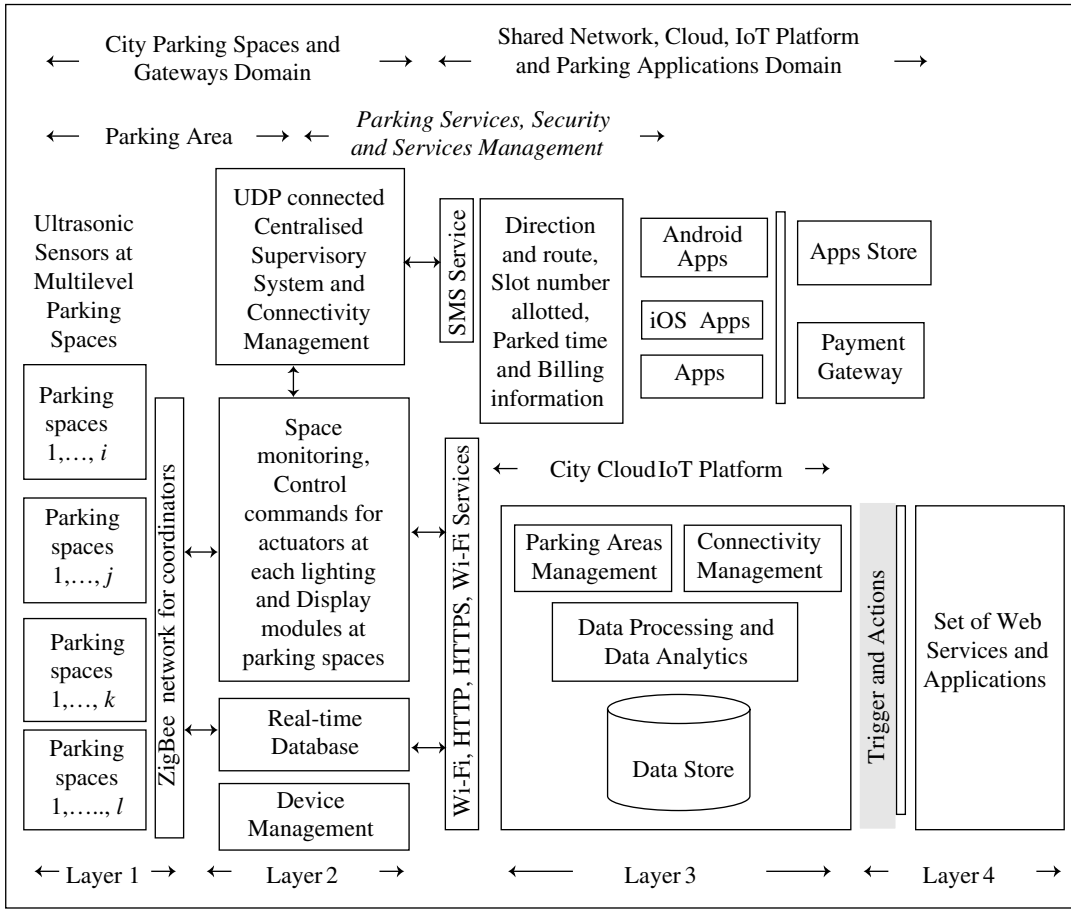


Figure 12.10 Data flow diagram, domain architecture reference model and four layers for the design of smart parking applications and services

Layer 2 connects CSS with all coordinators. Layer 2 includes a real-time time-series database. Layer 2 also connects with three modules for (i) displaying, (ii) space monitoring and (iii) control commands for actuators for each parking slot.

Layer 3 consists of SMS gateway and City cloud IoT platform that connects CSS, modules and database using the Wi-Fi, HTTP and HTTPS services. The platform has data store, data processing and analytics, and parking areas and connectivity management modules.

The CSS sends the UDP packets using MPLS and uses a SMS service to communicate with the mobile app. The SMS service communicates parking information. A packet provides information such as slot available, slot allocated, time parked, billing information and directional and parking space route details to the user's mobile phone

The user downloads a PAS app from the App store. The user's mobile also connects to a payment gateway for parking service bill payment.

Layer 4 web services connect the cloud data store, and use the PaaS cloud for the analytics.

Hardware Prototype Development and Deployment

Section 7.2.2 described sensors for ultrasonic pulse detectors. When a car parking slot is occupied, then the parked car reflects back ultrasonic pulses to the source. The sensor measures the reflected directional intensity and delay period for the reflections. The advantage of ultrasonic waves is that the distance and therefore, identification of reflector car surface is also found from the delay in reflected pulses.

The coordinator updates the parked slots status on each alert from a circuit. The sensor associated circuit at coordinator alerts the CSS for status change and generates time-series messages from the sensor data and communicates to the CSS for saving at the real-time database.

Figure 12.11 shows the design principle for a set up for identifying vacant spaces and slot IDs using ultrasonic pulses and back reflections to the transceiver (emitter and sensors) at the coordinator.

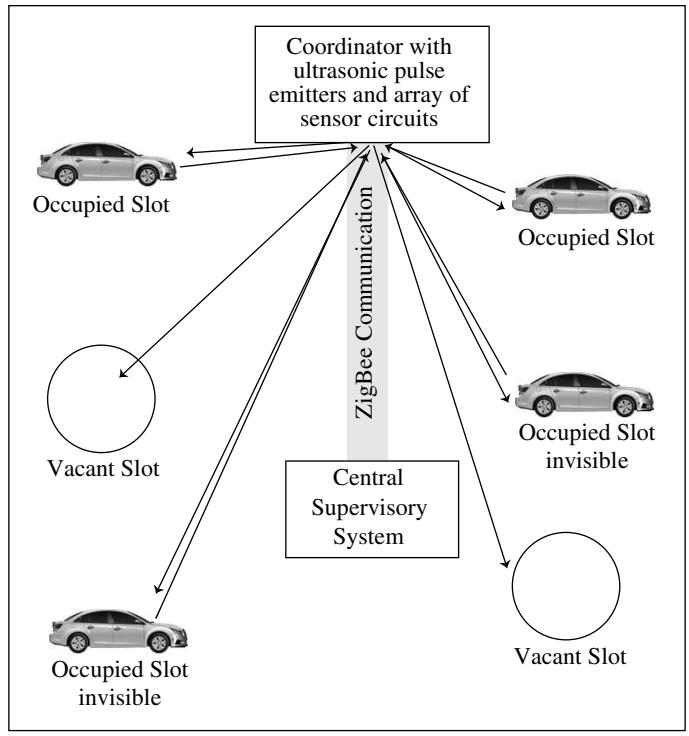


Figure 12.11 Design principle for the set up for identifying vacant spaces using ultrasonic pulses and back reflections from cars to the transceiver at the coordinator

Assume that a set of four coordinators each are placed at each level. A coordinator consists of a transceiver, which emits ultrasonic pulses and receives reflected signals using an array of sensors. Sensors have associated circuits at each level of parking. Assume that each transceiver's-coverage area at each coordinator spans to one fourth of the slots at that level. Therefore, total sixteen coordinators are located at four levels 1, 2, 3, and 4. Four coordinators at level 1 are C_{i1}, C_{i2}, C_{i3} and C_{i4}, four at level 2 are C_{j1}, C_{j2}, C_{j3} and C_{i4}, four at level 3 are C_{k1}, C_{k2}, C_{k3} and C_{k4} and four at level 4 are C_{l1}, C_{l2}, C_{l3} and C_{l4}.

The coordinators process the status information and time-series data transmits to CSS. The CSS receives information of parking slots from the coordinators as UDP datagram. CSS then relays the parking information to city-cloud platform data-store and to users, which seeks that from mobiles or card dashboard computer. Each coordinator sends all messages with slot ID and observation instances onto the ZigBee network as time-series data. A real-time database updates at regular intervals, say of 1 m each.

The edge sensors and devices wirelessly connect within small cells, systems connect with ZigBee. They communicate using LPWAN. The distributed network of edge-computing systems connects using IP protocols or using the Multi Protocol Label Switching (MPLS). The MPLS assigns the labels to data packets and forward the labels to City cloud IoT-platform.

Code Development Environment, Development, Debugging and Deployment

Section 9.3.2 described end-to-end IoT Solutions with Java and Eclipse IoT Stack. Section 12.5.1 described openHAB software platform for devices, sensor and edge systems at a smart home. Java, Eclipse IoT Stack and openHAB can be used for end-to-end solutions for each parking utility in the city.

Smart city parking solutions can also be developed using the intelligent management platform IMPACT.

12.5.3 Smart Environment-Monitoring

Environment monitoring refers to actions that are required for characterising and monitoring the quality of the environment. A smart environment monitoring system should enable the following:

1. Preparations for assessment of environment impact
2. Establish the trends in environmental parameters and current status of the environment
3. Interpretation of data and evaluate environmental quality indices
4. Monitor the air, soil and water quality parameters
5. Monitor harmful chemicals, biological, microbiological, radiological and other parameters

Weather Monitoring System

Uses of sensors for T, RH and P_{atm} parameters, WSNs, access points, gateways and a cloud platform for smart weather monitoring service

A smart weather monitoring system should enable the following:

1. Each measuring node for weather parameters is assigned an ID. Each lamppost deploys a wireless sensor node. Each node measure the T, RH and other weather parameters at assigned locations. A group of WSNs communicates using ZigBee and forms a network. Each network has an access point, which receives the messages from each node. Figures 7.17 and 7.18 depicted the WSNs. They depicted interconnections between nodes, coordinators, routers and access points. Each access point associates a gateway.
2. The nodes communicate the parameters up to the access point using WSNs at multiple locations.
3. Forward and store the parameters on an Internet cloud platform
4. Publishes weather messages for the display boards at specific locations in the city and communicates to weather API at mobile and web users
5. Publishes the messages in real time and send alerts using a weather reporting application
6. Analyse and assess the environment impact
7. Enables intelligent decisions using data and historical analytics reports at city cloud weather data store

Two domains and their high-level service capabilities in the weather monitoring services in IoT architecture reference model are:

1. *Device and Gateway Domain*: Assume that the system deploys *m* weather-sensor embedded devices, each with a location-data sensor and *n* access-points for the WSNs. A sensor node does minimum required computations, gathers sensed information and communicates with other connected nodes in the network.[23]

   A data-adaptation layer for the data, messages, triggers and alerts does the main computations and puts the result in real time updated database. The items identified for communication from gateway are queried from the database. The items communicate from gateway using network protocols and HTTP/HTTPS services.

   (i) *Device subdomain:* Hardware WSN board consists of sensors for weather parameters. A board example is Waspmote.[24] Following are the sensor circuit features: ultralow power dissipation; multiple transceiver interfaces, such as ZigBee and Wi-Fi (for medium range), RFID, NFC, Bluetooth 2.1 or BL LE (for short range) and LPWAN, 4G, 3G (for long range); OTA programmability, AES, RSA, MD5, SHA, Hash (as encryption libraries) and bus protocols, such as CAN and RS232C.

   (ii) *Gateway Subdomain:* The parameters and alerts communicate to a local or remote web service, time-and location-stamping service, item provider, protocol bindings and 6LowPAN/IPv6 modules as per configuration setting at the configuration

[23] https://en.wikipedia.org/wiki/Sensor_node
[24] http://www.libelium.com/products/waspmote/overview/

administration service of OSGi framework. The bindings between ZigBee LANs, 6LowPAN and LPWAN and IPv6 protocols are used for networking of the devices, WSNs, OSGi with the HTTP/HTTPS services.

2. *Application and Network Domain:* Applications and network domain deploys the applications and services and has high-level capabilities, such analytics, data visualisation, display-board feeds, weather reporting application, and IFTTT triggers and actions. The cloud platform can be IBM Bluemix, AWS IoT or TCUP.

Domain Architectural Reference Model

Figure 7.17 depicted layered architecture for network of nodes with fixed connecting infrastructure and the mobile WSNs using coordinators, relays, gateways and routers. Figure 7.18 depicted a layered architecture for network of nodes with multilevel hops and router or gateway and access point. Figure 12.12 shows the data-flow diagram and domain architecture reference model for the WSN based monitoring services.

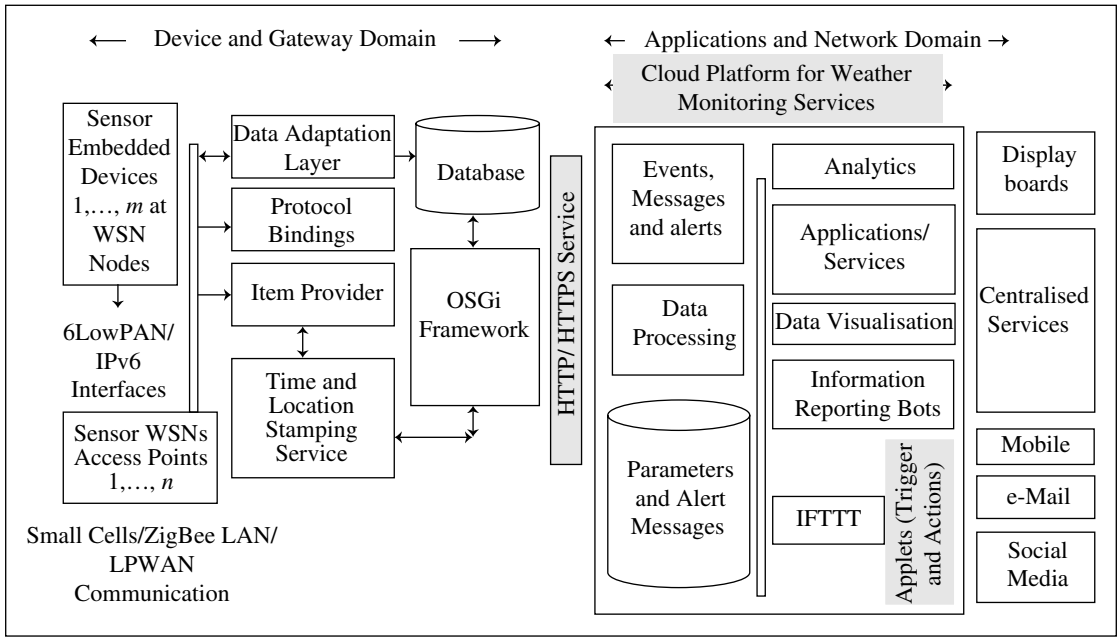


Figure 12.12 Data-flow diagram and domain architecture reference model for the WSNs based monitoring services

Devices Hardware Design and Code Development Environment, Development, Debugging and Deployment

A microcontroller circuit consists of memory, over the air programmability (OTP) and transceiver associated with each sensor or node. The weather monitoring circuit deploys sensors for T, RH and atmospheric pressure (P_{atm}) and may include solar visible radiation, wind speed and direction, and rainfall.

Hardware design of the sensor and WSN node can use Arduino board with ZigBee shield. Alternatively Waspmote, an open source wireless sensor platform for autonomous wireless low power including WSN nodes, called MOTEs (mobile terminals). Waspmote node battery power can be between 1 year and 5 years depending on the transceiver, frequency and the radio used. The platform is compatible with the Arduino IDE and has a community support.

The edge devices and WSNs codes development uses IDE such as the Arduino or Eclipse IDE (Section 9.3.1) for Java Developers.

Weather Reporting Bot

Uses of weather bots for autonomous weather reporting service

A bot is an application that runs automated or semi-automated scripts a for specific set of tasks and communicates the results over the Internet.[25] A bot generally performs the task which are simple and structurally repetitive, such as a, weather reporting bot. The word 'bot' is derided from the word *robot*.

A bot can communicate with an API using Instant Messaging (IM) or Internet Relay Chat (IRC) or to Twitter or Facebook. A bot can also chat and give responses to the questions from user API.

A widely used bot is an application in which the script fetches, analyses and files information from a web server. A server may specify in a text file, called robots.txt for rules on bot behaviour on that server. A server can use software to deny access in case the script does not observe the rules specified at the file.

A weather bot is multitasking. That can be used to communicate a report on a mobile. The bot fetches, analyses and communicates information to a report seeking API. The bot uses the weather parameters and generates the alert messages from the database and messages for forecast by a cloud analytics service.

A mobile app can display the report in two succeeding frames repeatedly. The first frame shows the weather condition of the current day, as:

1. First line: condition such as clear, rain, partly rain, cloudy or partly cloudy
2. Second line, first part text gives the day current T and four or five spaces
3. Second line, second part text gives superscripted text for the maximum T expected and subscripted text for minimum T expected, followed by four or five spaces
4. Second line, third part text gives superscripted text for current RH% value and subscripted text for wind-speed in kmph (kilometer per hour).

Thus, first reporting frame displays the current condition and day's forecast for T_{max} and T_{min}. Second frame shows the forecast for today, tomorrow and day after for weather as:

1. First line: "Sat Sun Mon"
2. Second line shows a symbol which is completely unfilled circle for sunny, or cloud image with sun for partly cloudy, or cloud sign for fully cloudy below each day

[25] https://en.wikipedia.org/wiki/Internet_bot

Sat, Sun and Mon. After the sign, a superscripted word gives maximum *T*, and a subscripted word, the minimum expected on that day.

Thus, forecast for three days is reported, viz. today, tomorrow and day after.

Example of creating a weather bot is Slackweatherbot API.[26] The bot uses the codes given a Farnciskim site.[27] The API is a node.js module for the bot. It displays Second frame shows the forecast for today, tomorrow and day after days for weather, for example as follows:

1. First line: Bot name Current Runtime (such as 09:15 A.M.)
2. Second line shows "Condition for *City Name*, *Place Name*, *Current Time*, *Standard* (such as IST, GMT)
3. Third line shows "Today (such as SAT): *T*, *condition* (such as sunny, partly cloudy, cloudy or rain)"
4. Fourth line shows "Tomorrow (such as SUN) Current: *T*, *condition*"
5. Fifth line shows "Day after (such as MON) Current: *T*, *condition*"

Air Pollution Monitoring

Uses of sensors for CO, CO_2 and T, RH and P_{atm} parameters, WSNs, access points, gateways and a cloud platform for smart air pollution monitoring service

A growing problem for all residents is air pollution from cars, toxic gases generated in factories and farms, such as carbon monoxide (CO). Pollution needs monitoring and to ensure the safety of workers and goods inside chemical plants. The monitoring does the following tasks:

1. Monitoring and measuring levels of CO, a gas dangerous above 50–100 ppm level; carbon dioxide (CO_2), a gas causes which greenhouse effect; and ozone (O_3), a gas dangerous above 0.1 mg/per kg air level, for controlling air pollution
2. Monitoring and measuring levels of hydrogen sulfide (H_2S), a highly toxic gas. It is a greenhouse gas so its increase may contribute to global warming as well.
3. Monitoring and measuring levels of hydrocarbons, such as ethanol, propane.
4. Measure T, RH and P_{atm} parameters for calibrations of sensed gaseous parameters of each node
5. Investigate air quality and the effects of air pollution.
6. Compute Air Quality Index (AQI) from the parameters, such as hourly or daily averages of air pollutant concentration, particulate matter (such as dust or carbon particle)
7. Compute source and spatial dispersion of pollutants as a function of day conditions, wind-speed and direction, air temperature and air temperature gradient with altitude and topography using analytics.
8. Data visualisation
9. Report the pollution status to monitoring authorities

[26] https://github.com/franciskim/Slackweatherbot

[27] https://franciskim.co/how-to-create-a-weather-bot-for-slack-chat/

Sensors play a vital role in air-quality monitoring. The application has eleventh ranking among 50 sensor-applications for a smarter world.[22]

A data-flow diagram and domain architecture reference model for air pollution monitoring services are similar to Figure 12.12. Two domains and their high-level service capabilities in the air quality and pollution monitoring services in IoT architecture reference model are:

1. *Device and Gateway Domain*: Assume that the system deploys *m* gas sensor embedded devices at each WSN with a location-data sensor and *n* access-points for the WSNs (Figures 7.17 and 7.18). The data-adaptation layer at gateway does the aggregation, compaction and fusion computations for each sensor node data. The queries gather sensed information from the database and the items selected communicate using HTTP/HTTPS/MPLS services.

   WSN board IO ports connect the sensors for gaseous, particulate matter and weather parameters. Each sensor node is configured by assigning a node ID. A node ID maps with the GPS location found earlier from GPS modules at the data-adaptation layer at the gateway.

   A sensor ID is configured for each sensor at the node. Each sensor associated circuit is also configured for frequency of measurements every day and interval between two successive measurements. The sensor circuit is configured to activate only for measurement duration at a measuring instance followed by long inactive intervals.

   Refer Example 9.3. The example gives the codes for uses of the calibration coefficients of the temperature sensor. Refer Example 9.6 which gives the codes for uses of the calibration coefficients of sensed data communicated by I2C serial bus. The coefficients enable to sense parameters with enhanced accuracy.

   The coefficients for a gas-sensor output depend on the T, RH and P_{atm} (THP). Therefore, these parameters are also measured along with gas sensor outputs. Mapping of a sensor output with the THP coefficients at the adaptation layer will result into accuracy in the measurements of values of sensed parameters.

   An example is Waspmote board[24] which can be used with sensors such as city pollution CO, NO, NO_2, O_3, SO_2 and dust particles sensors and air-quality finding sensors for SO_2, NO_2, dust particles, CO, CO_2, O_3 and NH_3. The Arduino or Eclipse IDE can be used to develop codes for the Waspmote.
2. *The Applications and Network Domain:* The applications and network domain deploys the applications and services and have high-level capabilities, such as events, messages, alerts and data processing, databases, applications and services, analytics, data visualisation, display-board feeds, pollution reporting applications and services, and IFTTT triggers and actions. The cloud platform can be TCUP, AWS IoT, IBM Bluemix or Nimbits.

Uses of sensors for T, RH, CO, CO_2 and infrared light, WSNs, alarm algorithm, access points, gateways, real time analytics and cloud platform for smart forest-fire detection and mapping of affected areas

Forest Fire Detection

A big problem for countries with large forest areas is forest fires. A fire monitoring service does the following tasks:

1. Uses OTP features for programmable WSNs and gateways
2. Measures and monitors the T, RH, CO, CO_2 and infrared light (fire generated) intensity in real time at preset intervals
3. Each WSN uploads the program and preset measured intervals of t1 (say, 300 s) each and the preset measured intervals of t2 (say, on 1 or 5 s) on sensed parameters values exceeding thresholds can instantaneously trigger the fire-alarm algorithm
4. Configures the data-adaptation layers with calibration parameters
5. Communicates the WSN messages at the preset intervals to the access point associated for specific network area
6. Communicates alerts, triggers, messages and data at data-adaptation layer using an uploaded program at associated gateway
7. Uploads connectivity programs for gateways
8. Runs at the data-adaptation layer the faulty or inaccessible sensors at periodic intervals
9. Integrates data with the node locations found from mapping with node IDs, compute, and activate the alarms using an algorithm, input-sensed and calibrated coefficients
10. Processes the layer data and database information, and communicates instantaneously to nearest mobiles and fire-fighting services near the access point gateway
11. Updates the database and communicates to a cloud platform, such as Nimbits, my.openHAB, TCUP, AWS or Bluemix platform
12. Modifies the preset measured intervals to t2 on activation of the fire alarm after value changes above the configured threshold values
13. Uses analytics to evaluate reliability index of the preset, threshold and configuration values and need to update alarm-algorithm and if needs improvement then upload new algorithms
14. Uses analytics to generate and communicate topological maps for the currently fire-infected forest area and reachability maps for fire-fighting service equipments

Sensors play a vital role in forest-fire monitoring. The application has tenth ranking among 50 sensor-applications for a smarter world.[22]

Figure 12.13 shows a data-flow diagram and domain architecture reference model for the monitoring service.

The figure shows that the service deploys m embedded-sensor devices at each of n WSN associated with x access points. Device and gateway domain functions in the fire monitoring service for forests in IoT architecture reference model is as follows:

A lookup table enables mapping of two entities. Location-data stamping uses sensor IDs at a lookup table. Data adaptation of each sensor is at the layer. Data aggregates, compacts and fuses, computes, gathers sensed information and the algorithms use that for alarm and faulty sensor identification and configuration management. Data store at the database, updates in real time. The alerts and messages communicate to IoT cloud platform.

Hardware WSN board and sensors can use Waspmote board.[24] Each WSN communicates to access points using a multiprotocol wireless router.

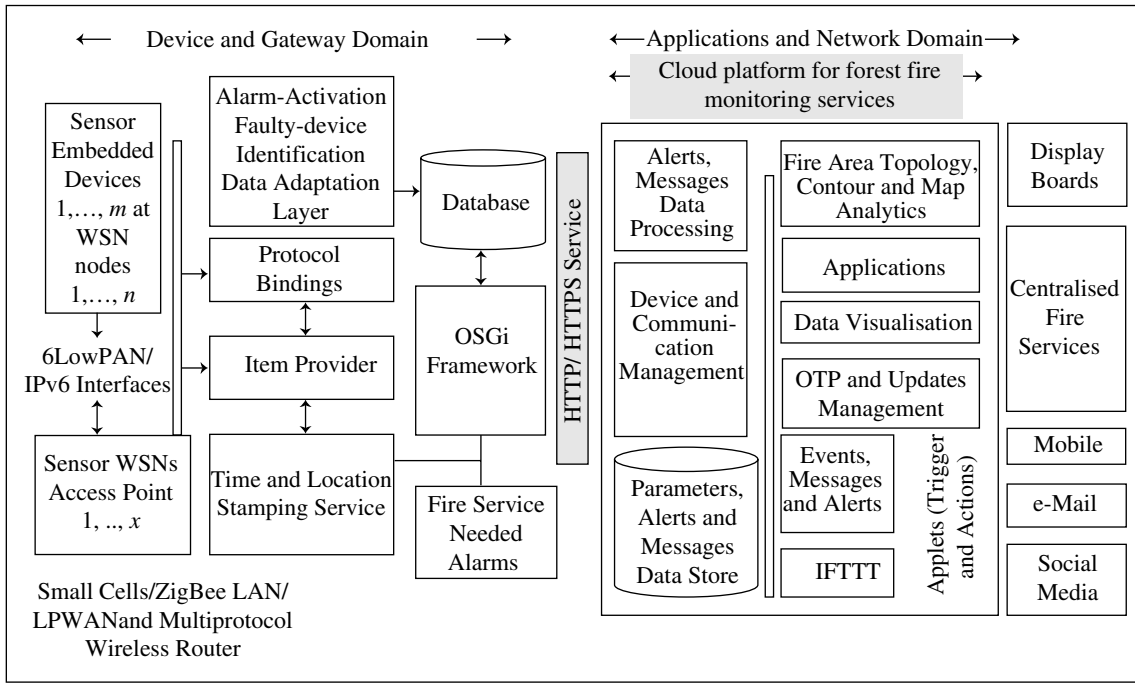


Figure 12.13 Data flow diagram and domain architecture reference model for the WSNs based Forest Fire monitoring service

12.5.4 Smart Agriculture

Following section describes two applications viz., smart irrigation in crop fields and smart wine quality enhancing.

Smart Irrigation

Uses of soil moisture sensors, actuators for water channels, WSNs, LPWAN, access points, gateway and cloud platform for smart irrigation

Smart irrigation deploys sensors for moisture. A smart irrigation monitoring service does the following tasks:

1. Sensors for moisture and actuators for watering channels are used in smart irrigation.
2. Uses soil moisture sensors with a sensor circuitry board with each one installed at certain depth in the fields.
3. Uses an array of actuators (solenoid valves) which are placed along the water channels and that control deficiencies in moisture levels above thresholds during a given crop period.
4. Uses sensors placed at three depths for monitoring of moisture in fruit plants such as grapes or mango, and monitors evapotranspiration (evaporation and transpiration)
5. Measures and monitors actual absorption and irrigation water needs
6. Each sensor board is in a waterproof cover and communicates to an access point using ZigBee protocol. An array of sensor circuits forms a WSN.

7. Access point receives the data and transfers it to an associated gateway. Data adapts at the gateway and then communicates to a cloud platform using LPWAN.
8. The cloud platform may be deployed such as Nimbits, my.openHAB, AWS or Bluemix.
9. Analytics at the platform analyses the moisture data and communicates to the actuators of water irrigation channels as per the water needs and past historical data
10. Measurements at the sensors are at preset intervals and actuators activate at analysed required values of the intervals.
11. The platform uploads the programs to sensors and actuators circuitry and sets preset measurement intervals of T1 (say, 24 hour) each and the preset actuation interval of t2 (say, on 120 hour)
12. Sensed moisture values when exceed preset thresholds then trigger the alarm
13. An algorithm uploads and updates the programs for the gateways and nodes.
14. Runs at the data-adaptation layer and finds the faulty or inaccessible moisture sensors at periodic intervals
15. Open source SDK and IDE are used for prototyping the monitoring system.

Smart Wine Quality Enhancing

The sensors monitor the soil moisture and trunk diameter in vineyards. The monitoring controls the sugar content in grapes and health of grapevines.

Data-flow diagram and domain architecture reference-model for the monitoring service are similar to ones shown in Figure 12.12.

Device and Gateway Domain

A WSN measures moisture and other parameters and has an ID. Each node is a WSN. Each WSN measures at assigned places in a crop or vineyard at certain depth(s) inside the soil. Sensors at three equally spaced depths are used for the vineyard grapes sugar-control. A group of WSNs communicate among themselves using ZigBee and form a network. Each network has an access point, which receives the messages from each node using LPWAN. Figures 7.17 and 7.18 show the WSNs. They show interconnections between nodes, coordinators, routers and access points. Each access point associates a gateway. Each gateway communicates to the cloud using LPWAN.

Reconfirm Your Understanding

- Prototypes for applications and services can be developed using the data-flow diagram, two-domain four-layers architectural reference model. Design of smart applications and services need designing and prototype development for the four architectural layers: (i) sensor and device networks, (ii) data adaptation and gateway, (iii) cloud platform, which includes data store and analytics, and (iv) applications and services.
- IDE (layer 1) is used for programming the sensors and devices.
- ZigBee, 6LowPAN, IPv6, WLAN and LPWAN are used for communication from sensor and device networks.

- Eclipse IoT stack and OSGi services (layers 1 and 2) can be used for Java programming at architectural layers 1 and 2.
- An access point receives the data and transfers it to the associated gateway. Data adapts at the gateway and then communicates to a cloud platform using LPWAN.
- A cloud platform (layer 3), such as Nimbits, my.openHAB, AWS, Bluemix or TCS CUP, is deployed for processing the messages, events, alerts, triggers and data and storing, analytics, and visualising the results, and using the platform-specific features.
- Applications, services and Mobile apps (layer 4) enable a number of services using the cloud.
- Illustrative examples of numerous IoT applications and services are smart homes, smart cities, smart environment monitoring and smart agriculture.
- Smart home services enable home-lighting control, control and monitoring of appliances, security, intrusion detection, video surveillance, access control and security alerts, Wi-Fi, Internet and remote cloud access for control and monitoring.
- An open source openHAB is the development environment and deployment platform. The platform is neutral to hardware and interfacing protocols. For example, a security camera device may be on Raspberry platform and lighting devices on Arduino. The automation logics can connect different systems.
- A cloud platform is my.openHAB. The my.openHAB cloud connector also includes REST and cloud based services, such as IFTTT.
- IFTTT service enables a developer to create a sequential set of conditional (If This Then That) statements, called applets, which trigger actions by change to other web-service, such as Facebook, Twitter, Gmail. APIs of the application control the actions using the triggers.
- Smart city integrates multiple ICT and IoT solutions in a secure fashion to manage a city's assets, such as information systems, schools, libraries, transportation systems, hospitals, power plants, water-supply networks, waste management, law enforcement, and other community services.
- Smart city edge sensors and devices wirelessly connect, and networks of the devices communicate using LPWAN, the systems connect using IP protocols or using the Multiprotocol Label Switching (MPLS).
- Smart city solutions can deploy CISCO IoT cloud. Alternative is ThingWorx IoT platform for intelligent management of things. City cloud IoT platform collects messages, triggers, alerts and data files at data store. The platform uses events, triggers and data store and analytics. The platform connects to a number of city applications and services, and triggers the actions, such as connect to social media, set of web services, applications and mobiles apps.
- Smart parking service is a city service, which guides the drivers to the vacant parking slots and spaces, and also provides a mobile app. The app assists a driver and enables the driver to obtain appropriate parking-slot information remotely.
- Smart environment monitoring refers to actions that are required for characterising and monitoring the quality of environment, such as air, soil and water.
- Weather-monitoring systems measure the T, RH, P and other weather parameters; communicate those using WSNs at multiple areas, which communicate up to the access points and associated gateways. The gateway adapts, stores and forwards the parameters to the Internet cloud platform.
- Weather-monitoring service publishes weather-messages for display boards at specific locations in the city and communicates to weather APIs at mobiles and web users.
- A bot fetches analyses and communicates server information repeatedly for the report-seeking APIs. The bots communicate the reports on a mobile app or web application. A weather-bot is a multitasking JavaScript or node.js scripts for the weather reports autonomously.

- Smart air-pollution monitoring-service measures the levels of CO, CO_2, particulate matter and other parameters. The service computes AQI from the parameters, such as, hourly or daily averages of air pollutant concentration and particulate matter (such as dust or carbon particle)
- The service computes source and spatial dispersion of pollutants as a function of day conditions, wind-speed and direction, air temperature and air temperature
- Smart forest fire monitoring service deploys number of network of WSNs, interconnected access points and associated gateways. The gateways connect with Internet cloud platform enables forest fire detection and map the affected area.
- Smart irrigation deploys sensors for moisture at a depth in the crop fields and actuators for watering channels. Smart quality monitoring deploys sensors placed at three depths for monitoring of moisture in fruit plants such as vineyard or mango, and monitors evaporation and transpiration.
- Smart irrigation controls deficiencies in moisture levels above thresholds during a given crop-period.

Self-Assessment Exercise

1. What are the programming layers in development of prototypes? ★
2. List the roles performed by my openHAB cloud platform for smartHome services? ★★
3. What are the design complexities of smart home, smart city parking, weather monitoring and air-quality index monitoring services? What are the limitations of open source tools for IoT development? ★★
4. How does Eclipse IDE help in prototyping the sensor and device circuits? ★★★
5. Draw the objects and layers in openHAB architecture, and explain the use of each object and layer. ★★
6. What are the objects and their uses at my.openHAB.org platform for openHAB based development? ★
7. What are the functions of IoT core in smart city applications and services? ★★
8. How are CISCO IoT, IOx and Fog used for smart city applications and services? ★★★
9. What are the objects and their uses at city parking spaces and gateways domain? ★★★
10. What are the objects and their uses at city cloud IoT platform for the parking service? ★★
11. How does provisioning of IFTTT service help in smart home and smart city applications and services? ★
12. How does delay period evaluation in receiving the reflected pulses enable identification of a parking slot ID by the coordinator? ★★
13. List the tasks of a weather monitoring service. ★
14. Why is the OTP feature useful in air-pollution monitoring circuits, network and gateway? ★
15. List the similarities in prototype development approach for weather, air-pollution and forest-fire monitoring services. ★★★
16. List the tasks of a air-pollution monitoring service. ★

17. What are the objects and their uses at cloud platform for forest-fire monitoring services? ★★★
18. List the tasks of a smart irrigation monitoring service. ★
19. How does smart irrigation result in efficient uses of water? ★★
20. Draw a table of similarities and differences between uses of IoT cloud platforms for smart home, smart city, smart parking, smart weather, smart air-pollution and smart forest-fire monitoring services. ★★★

12.6 CASE STUDY: SMART CITY STREETLIGHTS CONTROL AND MONITORING

LO 12.5 Undertake new IoT project design challenges from understanding of a case study, 'Smart city streetlights control and monitoring'

Figure 1.1 in Example 1.2 depicted the application of Internet of Things concept for the city streetlights. A lamppost hosts a streetlight, WSN actuator and sensors. The sensors enable messages for lamp functioning status, ambient light and traffic. The actuator enables switching the lights on and off.

When ambient light is above threshold, then the lights are switched on. The WSN deploy sensors for detecting traffic presence and traffic density so that when traffic is not present, then lights are switched off. This results in saving of energy. The traffic density messages communicate to traffic-signal monitoring service.

The WSN transceiver can also accept data from other services, such as Wi-Fi service, security service or traffic signalling service and retransmit onto the network of WSNs and then to access points. City services can deploy lampposts in streetlighting systems as information networks. A lamppost can be an active node in the services network.

Each transceiver at the lamppost can receive and retransmit in real time. Events, messages, alerts, triggers and notifications from a number of services can transmit for services such as smart parking, traffic signalling, waste management, air-quality index monitoring services, security services for home, banks and important public places, emergency services and hospitals.

Smart city streetlights control and monitoring: when traffic is not present then lights are switched off as well as lampposts function as information network nodes

A control and monitoring service for city streetlights does the following tasks:

1. Measures and monitors the streetlights and measures traffic parameters in real time at preset intervals
2. Each WSN is uploaded by the program for configuring and communicating within the WSN network
3. The network connects a coordinator which deploys the data adaption, store, time, location, IDs stamping and gateway interfaces
4. Communicates the WSN network messages

5. Messages transmit at the preset intervals to the access point, which connects a coordinator.
6. Coordinator generates and communicates alerts, triggers, messages and data after aggregating, compacting and processing at data-adaptation layer.
7. Coordinator creates and updates in real time a database which transfers to the cloud for processing and for cloud data store.
8. Uses the OTP features and uploads the programs at the WSNs and gateways. An OTP module at the cloud node provides OTP management and uploads connectivity programs for gateways
9. Runs at the data-adaptation layer for faulty or inaccessible sensors at periodic intervals
10. Integrates data, and activates the alerts and triggers
11. Cloud node provides platform for processes, analyses and visualisation of the data and database information. The node provides analytics and AI for optimising monitoring and control functions.
12. Cloud platform can be CISCO IoT, IOX and Fog, Nimbits, my.openHAB, TCUO, AWS or Bluemix platform with Watson analytics.

Architecture Reference Model Two Domains

Figure 12.14 shows data-flow diagram and domain architecture reference model for the monitoring service.

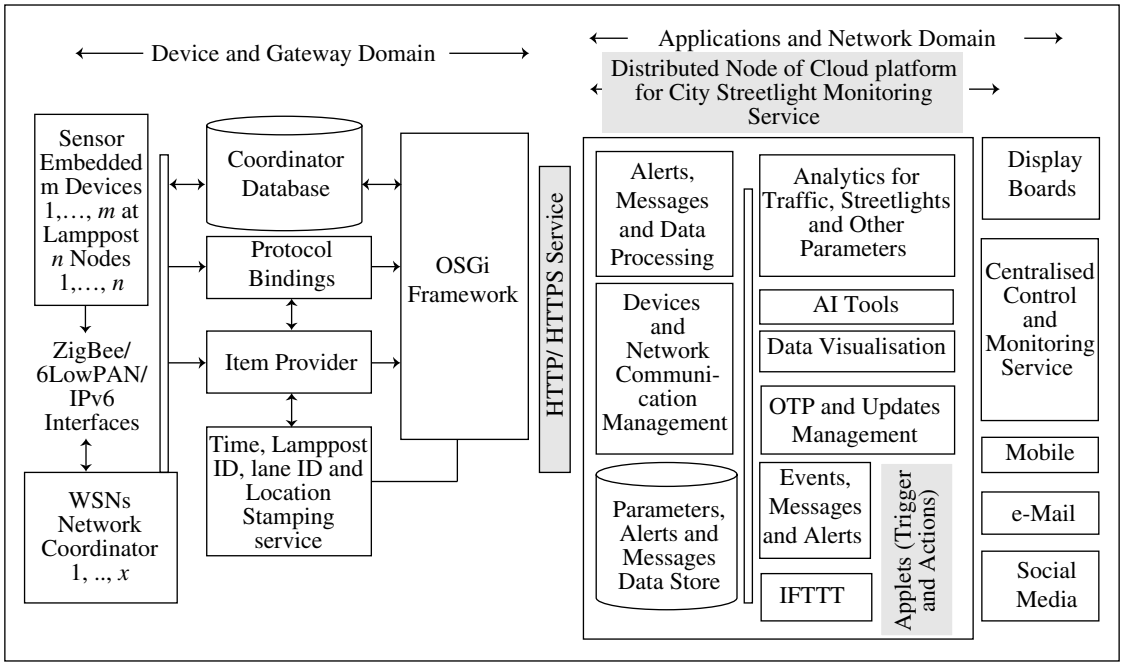


Figure 12.14 Data-flow diagram and domain architecture reference model for the WSNs networks of city streetlights central control and monitoring service

Device and Gateway Domain

Hardware and software components and modules are as follows:

Hardware

Hardware consists of m embedded-devices at a WSN. The n WSN node networks communicate between them using ZigBee/6LowPAN/IPv6 protocol. The city streetlight service deploys x coordinators. Each network communicates with a coordinator using LPWAN or ZigBee IP wireless interfaces. Coordinator functions as data store, protocol binder, item provider and gateway.

Each lamppost deploys a WSN. Each node senses a set of sensors data. Sensor circuits can deploy Arduino boards with ZigBee or ZigBee IP shields. Each WSN interfaces with other WSNs and forms a network of ZigBee devices.

A WSN measures the following parameters: (i) ambient light condition, whether above or below a preset threshold, (ii) presence or absence of traffic in vicinity, (iii) traffic density and (iv) lamppost status, whether non-functional or not

Each lamppost need not measure traffic parameters. Each WSN configures the sensing devices so that a measurement activates or deactivates as per commands from the coordinator and central monitor service. Configuring the node enables each parameter measurement at different preset intervals. Each WSN configures the actuator for enabling the lights on and off as per commands.

A group of WSNs communicate among themselves using ZigBee and form a network. Each network has an access point, which receives the messages from each node using LPWAN. Figures 7.17 and 7.18 show the WSNs. They show interconnections between nodes, coordinators, routers and access points. Each access point associates a gateway. Each gateway communicates to the cloud using LPWAN.

Software

Open source IDE or Eclipse IoT stack which include OSGi can be used for software development at devices and gateway domain.

Each WSN is assigned sensor-IDs, lamppost-ID, lane-ID, subgroup-ID (left and right sides traffic). A subgroup of wireless sensor nodes form a WSN network and an assigned network-ID. Each coordinator is assigned a coordinator-ID.

Each coordinator has three modules: (i) protocol binding module, (ii) item provider module for communication of queried items, alerts, messages and data, and (iii) time, lamppost ID, lane ID and location stamping service. The coordinator can use an open source OSGi framework for Java codes. A database at coordinator stores in associated streetlights, lanes and lane subgroups data.

Applications and Network Domain

Cloud platform for city streetlight monitoring service deploys a number of distributed nodes. Internet connectivity is using HTTP/HTTPS service. The IP protocol network

routers connect each coordinator with a distributed node. The distributed node platform provides the:

(i) Alerts, messages and data processing module
(ii) Devices network and communication management module
(iii) Analytics tools for traffic, streetlights and other parameters
(iv) Data store for parameters, alerts and messages
(v) AI tools
(vi) Data visualisation tools
(vii) Coordinators, networks and nodes update management using OTP
(viii) Event messages, triggers and alerts for central control and monitoring services
(ix) IFTTT for communication to mobile, e-mail, social media and web services and applications.

12.6.1 Streetlights Control and Monitoring Coding Examples

Following are the examples of communication of parameters from a lamppost to the coordinator in Java.[28]

```
Class Main
package com.main.execution.files;
import java.sql.Time;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap; // JSON Object is subclass of java.util.HashMap
//import org.json.simple.JSONObject;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import com.bean.classes.PathClass;
import com.bean.classes.StreetLight;
import LpostCommunication.com.trafic.communication.ResponseFromServer;

public class MainClass {
    public static void main(String[] args) {
          Configuration cfg=new Configuration();
          cfg.configure("resourcefiles/hibernate.cfg.xml");
          SessionFactory sf=cfg.buildSessionFactory();
          Session s=sf.openSession();
          StreetLight lpostDetails=new StreetLight("Lpost_1", "Lane1");
          ResponseFromServer response=new ResponseFromServer();
          response.getResponse(request,    isFaulty,    isAmbLightCondition,
```

[28] Code through courtesy Dr. Mrs. Preeti Saxena

```
            isTrafficCondition, status);
            response.getResponse(lpostDetails, 0, 1, 1, 0);
            PathClass finalobject=response.action(lpostDetails);
            s.save(finalobject);
            s.beginTransaction().commit();
            s.close();
            sf.close();
            System.out.println("Done");
    }
}
```

Object JSONText Out *to string*

Following is the code in Java for adding a new sensor for traffic density.

```
JSONObject obj=new JSONObject();
    obj.put ("name","SensTrafficDensity1");
     obj.put ("trafficDensitySensor",new Integer(100));//Sensed Traffic
Density is Integer
     obj.put ("numberOftrafficDensitySensors", new Double(1000.21));
    obj.put ("is_newSensorAdded",new Boolean(true)); //Returns Boolean for
new Sensor Addition
 StringWriter out = new StringWriter();
    obj.writeJSONString(out);
 String jsonText = out.toString();
 System.out.print (jsonText);
```

```
Result: {"Name": "SensTrafficDensity1": "TrafficDensity": 10"
"numberOfSensors":1000000, "newSensorAdded": null, }
```

Lamppost WSN Streetlight Class

Following is the code in Java for class StreetLight with a lamppost ID, laneID. status,

```
package com.bean.classes;
public class StreetLight {
 public StreetLight(String lpostId, String laneId) {
// To do Auto-generated constructor stub
    this.lpostId = lpostId;
    this.laneId = laneId;
          }
    private int status;// status of functioning, traffic presence and ambient
light at a lamppost
    private String lpostId;
    private String laneId;
private int isFaulty; // Street Light if faulty value is 1 (true)
    private int isAmbLightCondition; // sensor value , if ambient light
conditions OK then value is 1 //(true)
 private int isTrafficCondition; // Traffic present the is 1 (true)
```

```
    public int getStatus() {
    return status;
    }
 public void setStatus(int status) {
    this.status = status;
    }
    public String getLpostId() {
    return lpostId;
    }
    public void setLpostId(String lpostId) {
    this.lpostId = lpostId;
    }
    public String getLaneId() {
          return laneId;
    }
    public void setLaneId(String laneId) {
    this.laneId = laneId;
    }
    public int getIsFaulty() {
    return isFaulty;
    }
    public void setIsFaulty(int isFaulty) {
    this.isFaulty = isFaulty;
    }
    public int getIsAmbLightCondition() {
    return isAmbLightCondition;
    }
    public void setIsAmbLightCondition(int isAmbLightCondition) {
    this.isAmbLightCondition = isAmbLightCondition;
    }
    public int getIsTrafficCondition() {
    return isTrafficCondition;
    }
    public void setIsTrafficCondition(int isTrafficCondition) {
    this.isTrafficCondition = isTrafficCondition;
    }
}
```

Class for Lamppost Communication of traffic information with a Coordinator (Server) at an instance

```
package LpostCommunication.com.trafic.communication;
import java.text.SimpleDateFormat;
import java.util.Date;
import com.bean.classes.PathClass;
import com.bean.classes.StreetLight;
public class ResponseFromServer {
public StreetLight getResponse(StreetLight request, int isFaulty, int
isAmbLightCondition, int isTrafficCondition,int status) {
    request.setIsFaulty(isFaulty);
    request.setIsAmbLightCondition(isAmbLightCondition);
    request.setIsTrafficCondition(isTrafficCondition);
    request.setStatus(status);
    return request;
    }
public PathClass action(StreetLight response) {
 if (response.getIsAmbLightCondition() == 1 && response.getIsFaulty() == 1
    && response.getIsTrafficCondition() == 1) {
 System.out.println("Light Condition :" + response.getIsAmbLightCondition() +
" Faulty "
                              + response.getIsFaulty() + " Trafic Condition
" + response.getIsTrafficCondition());
 } else if (response.getIsAmbLightCondition() == 1 && response.getIsFaulty()
== 1
    && response.getIsTrafficCondition() == 0) {
    System.out.println("Light Condition :" + response.
getIsAmbLightCondition() + " Faulty "
                              + response.getIsFaulty() + " Trafic Condition
" + response.getIsTrafficCondition());
    } else if (response.getIsAmbLightCondition() == 1 && response.
getIsFaulty() == 0
     && response.getIsTrafficCondition() == 1) {
     System.out.println("Light Condition :" + response.
getIsAmbLightCondition() + " Faulty "
          + response.getIsFaulty() + " Trafic Condition " + response.
getIsTrafficCondition());
          } else if (response.getIsAmbLightCondition() == 1 && response.
getIsFaulty() == 0
          && response.getIsTrafficCondition() == 0) {
    System.out.println("Light Condition :" + response.
getIsAmbLightCondition() + " Faulty "
          + response.getIsFaulty() + " Trafic Condition " + response.
getIsTrafficCondition());
```

```
            } else if (response.getIsAmbLightCondition() == 0 && response.
getIsFaulty() == 1
            && response.getIsTrafficCondition() == 1) {
     System.out.println("Light Condition :" + response.
getIsAmbLightCondition() + " Faulty "
                                + response.getIsFaulty() + " Trafic Condition
" + response.getIsTrafficCondition());
      } else if (response.getIsAmbLightCondition() == 0 && response.
getIsFaulty() == 1
      && response.getIsTrafficCondition() == 0) {
     System.out.println("Light Condition :" + response.
getIsAmbLightCondition() + " Faulty "
            + response.getIsFaulty() + " Trafic Condition " + response.
getIsTrafficCondition());
            } else if (response.getIsAmbLightCondition() == 0 && response.
getIsFaulty() == 0
            && response.getIsTrafficCondition() == 1) {
     System.out.println("Light Condition :" + response.
getIsAmbLightCondition() + " Faulty "
            + response.getIsFaulty() + " Trafic Condition " + response.
getIsTrafficCondition());
            } else {
     System.out.println("Light Condition :" + response.
getIsAmbLightCondition() + " Faulty "
            + response.getIsFaulty() + " Trafic Condition " + response.
getIsTrafficCondition());
            }
            return this.InsertOperations(response);
     }
Class for Received Lamppost Info insertion into Database at the Coordinator
     public PathClass InsertOperations(StreetLight map)
     {
            PathClass object=new PathClass();
            SimpleDateFormat format=new SimpleDateFormat("DD/MM/YYYY");
            SimpleDateFormat tformat=new SimpleDateFormat("HH:MM:SS");
            Date date=new Date();
            object.setAvgOnPeriod(10.0);
            object.setAvgTraficDensity(10.0);
            object.setAvgTraficPrsence(10.0);
            object.setDate(format.format(date));
            object.setFaulty(map.getIsFaulty());
            object.setLightFunctionality(""+map.getIsAmbLightCondition());
```

```
            object.setRoadPathId(map.getLaneId());
            object.setStatus("”+map.getStatus());
            object.setSubGroup("Left/rigth");
            object.setTime(tformat.format(date));
            object.setTimeSlotNumber(1);
            return object;
        }
    }
```

Mode of communication between the lamppost nodes and coordinator is pub/sub mode. Each streetlight lamppost MQTT client transfers the data to an MQTT broker. A coordinator also includes a MQTT client and subscribes for the sensed data of each lamppost. Each lamppost includes the MQTT broker to receive the sensed data from the host lamppost and the neighbouring lampposts. Example 9.11 codes can be used for coding for the communication between lampposts and the coordinator.

Reconfirm Your Understanding

- An application of Internet of Things concept is smart streetlights in the city. A lamppost can be an active node in the services network. City services can deploy lampposts at the street lighting systems as information network for security services for home, banks and public installations, hospitals, parking spaces, waste container management.
- Each lamppost for street lighting functions as a WSN. Each WSN interfaces with other WSNs and forms a network of ZigBee devices.
- The sensors at WSN measure the following parameters: (i) ambient light condition, whether above or below a preset threshold, (ii) presence or absence of traffic in vicinity, (iii) traffic density and (iv) lamppost status, whether non-functional or not
- Each lamppost communicates to a coordinator serving a lane in the city road network.
- ZigBee, 6LowPAN, IPv6, WLAN and LPWAN are used for communication from sensor and device networks. Eclipse IoT stack and OSGi services (layers 1 and 2) can be used for Java programming at architectural layers 1 and 2.
- Each access point receives the data and transfers it to an associated gateway. Data adapts at the gateway and is saved in the server database. The database communicates with a cloud platform using LPWAN.
- A distributed node cloud platform (layer 3) deploys cloud such as CISCO IoT, AWS, Bluemix or TCS CUP for processing the messages, events, alerts, triggers, data and storing, analytics, AI and visualising tools.
- Central control and monitoring service applications use the cloud data store and results of events processing.
- Examples of Java coding for communication of lamppost status and parameters to the coordinator are given. The codes can be used for development of full set of codes.

Self-Assessment Exercise

1. List the functions of a lamppost for the street lighting project. ★
2. What are the functions of central control and monitoring service? ★★
3. Tabulate the similarity between smart air-pollution monitoring and smart street-lighting project monitoring. ★★★
4. Why and how do the database contents at the coordinators communicate with the cloud used by the monitoring services? ★★
5. What are the actions to be taken at the cloud platform in the project? ★
6. List the usages of each tool by the central control and monitoring service. ★★
7. Explain the codes for Class Lamppost Communication. ★
8. What are the PathClass InsertOperations which create the columns and rows of SQL databases at the coordinator. ★★★

Key Concepts

- Analytics
- Application and network domain
- Arduino
- AWS
- Big data
- Bot
- CISCO IoT
- Cloud platform
- Cloud server
- Communication management
- Connected car
- Connected universe platform
- Customer data
- Customer monitoring
- Data Store
- Device and gateway domain
- Design abstraction
- Design complexity
- Distributed cloud node
- Domain architecture
- Eclipse IoT stack
- Eclipse Kura
- Ethernet
- ECUs
- Enriched data
- ETSI reference architecture
- Fog
- Gateway
- Gather phase
- IBM Bluemix
- IDE
- IETF IoT architecture
- IFTTT service
- In-car network
- Internet of ATMs
- IOx
- ITU-T reference model
- Java Coding
- LPWAN
- Maintenance centre
- Message cache
- Miracast
- Oracle IoT architecture
- PaaS
- Predictive analytics
- Premises monitoring
- Production monitoring
- Raspberry Pi
- Reference model
- Services
- Smart air quality monitoring
- Smart agriculture
- Smart environment monitoring
- Smart irrigation
- Smart parking
- Smart Street lights
- Smart weather monitoring
- Stream
- Supply-chain monitoring
- TCUP
- Value chain
- Waspmote
- WSN
- ZigBee

Learning Outcomes

LO 12.1

- Ability to conceptualise the design complexity level and use the connected Platform-as-a-Service (PaaS) cloud for accelerating design, development and deployment of M2M, IoT, IIoT applications and services
- Ability to develop IoT prototypes development and deployment
- Development and deployment using a number of open-source tools and connected-device platforms available for IoT prototypes for applications and services
- Development and deployment of applications and services using PaaS platforms, such as Microsoft Research Lab of Things, IBM Internet of Things Foundation (IITF), IBM Bluemix cloud platform, CISCO IoT, IOx and Fog technology, Xively, Nimbits, AWS IoT device SDK (Software Development Kit), and TCS Connected Universe Platform
- Knowledge that a cloud PaaS needs to provide scalable and secure architecture, gather, store, analyse data captured by the embedded sensors, events and diversified sources.
- Cloud PaaS solution needs deployment across heterogeneous and interoperable devices, sensors and applications.
- A project design can deploy TCUP device agents, device software modules and a set of web services for application developers to create highly scalable, intelligent and predictive analytics driven IoT applications, creating innovative products, intelligent infrastructure, enhancing the operational efficiencies, and craft new customer experiences and helping companies offer unique services to their customers.

LO 12.2

- Familiarised with the design approaches for IoT/IIoT globally trending usages in core business categories/areas—premises, supply chain and customer monitoring
- Premises monitoring and surveillance system need to communicate with the source of events, triggers, messages, data and data files between the ATMs and the bank server.
- Knowledge of number of steps in the design process—Abstraction, design of hardware and software in a reference model, identifying requirements of embedded hardware and software modules for the domains, design implementation for the domains hardware and software, and testing and validation
- Familiarised with reference-model architecture for the ATM surveillance example
- Knowledge of uses of prototype development boards, hardware and software deploying Eclipse Stack, events and streams processing, OLTP and event analytics
- Knowledge of design approach based on IETF/ITU-T reference model in a supply-chain monitoring system and RFIDs supply-chain applications
- Usage of Arduino board suitability for prototype RFIDs at the customer and company ends
- Usage of IDE and Eclipse Kura module in stack-supply-chain monitoring system
- IoT application for a customer-monitoring system is TCCICDD using mobile apps, embedded sensors/devices, gateway, network, server and applications, data visualisation and service/production/manufacture re-planning, re-scheduling or innovate the production and provide better customer experiences

LO 12.3

- Knowledge of 'connected car' concept and its IoT applications and services
- The connected car internal devices and systems enable car maintenance service, service centre applications: For example, remote diagnostics, roadside assistance, car malfunction and location car-based emergency services and reporting for service centre support, remote care, fleet management, fuel and eco initiatives.
- A central in-car computer enriches the generated data. In-car infotainment systems use Miracast devices for in-car network.
- Project design can use TCUP or other cloud server PaaS to enable a developer to develop new applications.
- The platform helps in applications such as maintenance by service centre; re-planning the marketing, sales, service and product development functions, manufacturing processes; and personalising and deepening customer experiences by developing additional value-added services.
- Knowledge of Tesla electric car with autopilot mode, ADAS, full self-driving capability and safety

LO 12.4

- Knowledge of the design approaches for IoT applications in smart homes, smart cities, smart environment monitoring and smart agriculture using illustrative examples
- Knowledge of development of prototypes for applications and services using data-flow diagram, two-domain four-layers architectural reference model
- Conceptualised the design of smart applications and services using four architectural layers: (i) sensor and device networks, (ii) data adaptation and gateway, (iii) cloud platform, which includes data store and analytics, and (iv) applications and services.
- IDE (layer 1) usages for programming the sensors and devices and ZigBee, 6LowPAN and IPv6 for communication from sensors in device networks, and the WLAN and LPWAN for wireless access point and gateways.
- Usage of data adaptation at the gateway and then communication to a cloud platform using WLAN or LPWAN
- A cloud platform (layer 3), such as Nimbits, my.openHAB, AWS, Bluemix or TCS CUP, is deployed for processing the messages, events, alerts, triggers and data and storing, analytics, and visualising the results and using the platform specific features.
- Applications and services (layer 4) enable number of services using the cloud.
- Illustrative examples of numerous IoT applications and services are smart homes, smart cities, smart environment monitoring and smart agriculture.
- Usage of an open-source openHAB is the development environment and deployment platform in smart home automation services
- Usages of REST and cloud-based services, such as IFTTT, for triggering the actions by change to other web services, such as Facebook, Twitter, and Gmail. APIs of the application control the actions using triggers.
- Knowledge that smart city services integrates multiple ICT and IoT solutions in a secure fashion to manage a city's assets such as information systems, schools, libraries, transportation systems, hospitals, power plants, water supply networks, waste management, law enforcement and other community services.
- Applicability of CISCO IoT cloud, IOx and Fog for smart city solutions

- Design approach for smart parking service, smart environment monitoring, and smart characterising and monitoring the quality of the environment, such as air, soil and water.
- Design approach for weather monitoring system and use of bots.
- Knowledge of smart air-pollution monitoring service to compute AQI from the parameters, such as, hourly or daily averages of air-pollutant concentration and particulate matter (such as dust or carbon particle)
- Familiarised with smart irrigation for control of deficiencies in moisture levels above thresholds during a given crop period and monitoring of moisture in fruit plants such as vineyard or mango, and monitoring evaporation and transpiration

LO 12.5

- Understanding an IoT project design using a case study, 'Smart city streetlights control and monitoring'
- A lamppost can be an active node in the services network. City services can deploy lampposts at the street lighting systems as information network for security services for home, banks and public installations, hospitals, parking spaces, waste container management etc.
- Project design as lamppost for street lighting functions as a WSN. Each WSN interfaces with other WSNs and forms a network of ZigBee devices.
- Usage of a distributed node form cloud platform for processing the messages, events, alerts, triggers, data and storing, analytics, AI and visualising tools.
- Ability to write coding in Java; learning from examples of coding for creating string from sensor observations and communication of lamppost status and messages

Exercises

Objective Questions

Select one correct option out of the four in each question.

1. Consider Internet of Umbrella (Example 1.1), Internet of Streetlights (Example 1.2), Internet of ATM Surveillance system (Example 2.3), and Internet of Automatic Chocolate Vending Machines (Example 5.1]. The design complexity levels are: ★
   (a) 1, 2, 3 and 5
   (b) 1, 2, 3 and 4
   (c) 2, 4, 3 and 5
   (d) 2, 3, 4 and 5
2. Consider CISCO IoT, IOx and Fog application development platforms, the IOx (i) combines IoT application execution within the Watson analytics applications (ii) enables highly secure connectivity, fast and reliable network and (iii) near real-time, automated and high volume of data of sensors and cloud. ★★

   Fog provides (iv) ability to transform sensor data, and perform the control functions (v) in sensor nodes (vi) within the distributed network nodes. This enables development of applications such as site asset management, energy monitoring, and smart parking infrastructure and connected cities.
   (a) (i) to (v)

(b) All except (i) and (v)
(c) All except (iii)
(d) (i), (ii) and (vi)

3. Features of cloud-based TCUP are: (i) offering a cloud Platform-as-a-Service (PaaS) which is non-scalable, and has secure architecture, (ii) offers a domain agnostic multi-tenant platform which optimises the network traffic, (iii) gathers, stores and analyses data captured by the embedded sensors, events, and diversified sources, (iv) enables device management, sensors data acquisition and storage, and analytics, and (v) makes it easy to develop, deploy and administer IoT, M2M/IIoT software applications. ★
   (a) (ii) to (v) true
   (b) All true
   (c) All true except (ii)
   (d) (i) to (iv) true
4. Cloud services platform functions are (i) communication management, (ii) deploying server for sending and receiving messages from the device agents, (iii) providing diverse sources and devices data Store, (iv) deploying big data store, (v) deploying an application(s) server and RDBMS, (vi) event processing, (vii) message caching and routing, (viii) OLTP and OLAP, (ix) data analytics, and (x) applications/services/ business processes, intelligence, and knowledge discovery software. ★
   (a) (iii) to (ix)
   (b) All
   (c) All except (iv)
   (d) (i) to (ix)
5. Surveillance system devices consist of (i) digital video camera, (ii) spatially arranged vibration sensors, (iii) software for enriching the sensors and camera data, (iv) generating the event messages, (v) on the events, process the data and video files real-time communication, (v) filtering, (vi) location and time stampings, (vii) encrypting, and (viii) media-server gateway communication. ★★
   (a) (i), (iii) to (vii)
   (b) All true except (ii)
   (c) All true
   (d) (i) to (vii)
6. Developer for surveillance system software for Internet of ATMs uses Eclipse Kura for (i) the Raspberry Pi, (ii) development environment, gateway services, cloud connectivity, management of device, network configuration and applications, (iii) hardware abstraction and other services in embedded Lua runtime, (iv) development of embedded sensor devices software using the framework based on WiringPi and PiFace, Gertboard and other shields, and (v) functions for development of device applications in embedded Lua language. ★
   (a) (ii) to (v)
   (b) (i) and (ii) true
   (c) All except (iii) true
   (d) All true except (iv)

7. Connected RFIDs supply chain applications are (i) plan and schedule the production, (ii) schedule the deliveries and shipping, (iii) verify the customer order, (iv) acknowledge the order, (v) confirm the customer delivery, and perform automated reordering. Connected RFIDs device and gateway capabilities (vi) are present in RFID physical device-cum-RFID reader which acquires the ID data. ★★★
   (a) (i) to (iv) true
   (b) (ii), (iv) and (v)
   (c) All true
   (d) (ii) to (v) true
8. The connected RFIDs supply-chain gateway communicates to the Internet the enriched data using a (i) wireless protocol to an access point, (ii) IP protocol to TCP/IP network, (iii) CoAP client and (iv) MQTT client. ★
   (a) All true
   (b) (iii) and (iv)
   (c) (i) and (ii)
   (d) All except (iv)
9. Consider an architectural reference model for tracking customer and monitoring services. The (i) layer 2 gateway enriches the data by generating time series and location stamped data, (ii) adapts data for communication using IP4 protocol on the *network*, (iii) layer 3 streams the gathered and enriched data stream to the server using the *network*, (iv) Layer 5 acquires the devices and diverse sources data at the data store and route to layer 6, and (v) layer 6 organises data at big data store and database RDBMS, analyse data using the event processing, message routing, and analytics, (vi) Layer 6 includes applications for creation of innovative products, intelligent infrastructure, and enhancement of operational efficiencies. ★★
   (a) All except (i)
   (b) (iii) and (iv)
   (c) (i) to (iv)
   (d) (i) to (vi)
10. Consider tracking customer service. Tracking is done using (i) customer's Internet-connected mobile apps and (ii) wearable digital devices, (iii) customer databases, (iii) embedded devices and sensors at customer's places of visit, (iv) information from customer feedback, and (v) information from the sale services and maintenance centres. The examples of IoT applications/services are creation of (vi) innovative products and (vii) provisioning for new customer experiences, and offering unique services to their customers. ★★★
    (a) (iii) to (v) true
    (b) All except (ii) and (v)
    (c) All except (ii) and (iii)
    (d) All true
11. Consider reference architecture for IoT applications for connected-car embedded devices clusters. Data from the layer 1 generates using (i) cluster of in-car ECUs, (ii) embedded devices, car-health devices and sensors, (iii) in-car infotainment systems, and (iv) mobile apps. In-car network (v) uses Bluetooth and NFC for inter-devices wireless communication to the Internet, (vi) CAN (Controller Area Network) for real-time devices and sensors data to the central computer, (vii) LIN (Local Interconnect ★★★

Network) for intra-car seats wire communication to the Internet, and (viii) MOST for multimedia devices.
(a) All except (iv) and (v)
(b) All except (iii), (v) and (vii)
(c) All except (ii) and (vii)
(d) All except (iii) and (iv)

12. Applications of gathered data using connected-car include (i) re-planning the marketing, sales, (ii) re-planning the company services, (iii) offering better customer experiences, (iv) service calls, diagnostics and predictive analytics at the service and maintenance centre, (v) triggers for the maintenance as per the car's condition, (vi) detecting dependencies amongst car-health problems, depreciation analysis and driving behaviour analytics, (vii) detecting patterns of dependencies among vehicle-health issues, (viii) location-based geospatial analysis. ★★★
(a) All true
(b) All except (i), (v) and (vi)
(c) (iv) to (vi)
(d) All except (vi) and (vii)

13. Consider architectural layers in an openHAB development environment for smart home applications and services. The prototype uses (i) openHAB add-on objects, (iii) openHAB core objects, (iv) OSGi framework services, (v) cloud platform and (vi) can deploy IFTTT service. ★★★
(a) All true
(b) All except (iii)
(c) (i) to (vi)
(d) All except (iv) and (vi)

14. IFTTT service enables (i) a developer to create a sequential set of conditional (If This Then That) statements, (ii) call Java applets, (iii) trigger actions by change to other web service, (iv) actions enable data communication from cloud to Facebook, Twitter, Gmail. ★★★
(a) (iii) and (iv)
(b) All except (ii)
(c) (i) to (iii)
(d) All except (iii)

15. Consider smart air quality monitoring service. The service tasks are (i) computes air quality index, (ii) measures T, RH and P, (iii) computes calibration parameters of gas sensors, (iii) measures CO, (iv) measures CO_2 and (v) particles suspended in air. The service (vi) connects and communicates with a cloud platform. The service deploys (vii) WSN networks, (viii) distributed computing nodes IP network, and (ix) LPWAN for sensor nodes network communication with a gateway. ★★★
(a) All except (v) and (ix)
(b) All except (ii) and (v)
(c) All true
(d) All except (ii)

16. Consider Internet of Streetlights application for central control and monitoring service. A lamppost communicates to another one in (i) a WSN network, (ii) an IP network, (iii) an LPWAN, (iv) and uses BT LE or ZigBee within the device network. ★★★

The network nodes (v) exchange messages, alerts and triggers, and communicate with (vi) a coordinator (server), (vii) IoT cloud, and (viii) central control and monitoring service for streetlights.

(a) All true
(b) All except (i), (iv) and (vi)
(c) (i), (iv) and (vi)
(d) (i), (iv) to (vi)

Short-Answer Questions

1. Justify assignment of complexity level 4 for development and deployment of smart air pollution and air quality monitoring applications and services. **[LO12.1]** ★
2. Explain project development by taking an example RFID based inventory control using Nimbits cloud platform. **[LO12.1]** ★★★
3. Why is cloud platform offering PaaS preferred over standalone dedicated application servers for developing and using the IoT applications/services? **[LO12.1]** ★★
4. When does a developer need the use of distributed platforms, such as CISCO IoT, IOx and Fog? **[LO12.1]** ★
5. What are the TCUP features which enable creation of highly scalable, intelligent and predictive analytics driven IoT applications, and allows IoT applications to be built for virtually any devices/sensors and virtually any business domain? **[LO12.1]** ★★★
6. What are the features of AWS? **[LO 12.1]** ★★★
7. What are the advantages of gathering, enriching, streaming and creating data store for the data captured by the embedded sensors, events, as well as diversified sources, such as customers, other enterprise databases and social media? **[LO12.2]** ★
8. How does the ETSI domains partitioning as device and gateway, and network and applications offer a suitable reference model for Internet of ATMs based surveillance system? **[LO12.2]** ★★★
9. Why are spatially distributed vibration sensors used in addition to digital cameras at an ATM in the surveillance system in Internet of ATMs? **[LO12.2]** ★★
10. Why is Raspberry Pi 2 model B+ (RPi 2) preferred over Arduino board for the prototype embedded real-time systems for 24×7 active digital video cameras? **[LO12.2]** ★
11. List the commonalities and differences in cloud server platforms shown in Figure 12.4 and 12.6. **[LO12.2]** ★
12. How does customer data monitoring and sensing along with the RFID usages improve the customer services and experiences in Internet-connected RFIDs and customers? **[LO12.2]** ★★★
13. What are the advantages of using Ethernet and Miracast nodes in in-car network? **[LO12.3]** ★
14. How does Internet-connected car improve maintenance and services using predictive analytics? **[LO12.3]** ★★★
15. What are the modules and actions at the gateway in home automation? **[LO12.4]** ★★
16. How do the tasks in a weather-monitoring system differ from that in an air-pollution-monitoring system? **[LO12.4]** ★
17. What are the changes required when light pulses are to be used instead of ultrasonic pulses for sensing vacant parking spaces? **[LO12.4]** ★★

18. What is the advantage of using an Internet bot in smart weather-monitoring service? **[LO12.4]** ★★
19. How can the quality of vineyard grapes be enhanced by using smart agriculture? **[LO12.4]** ★
20. What are the changes in codes required to use TLV format for sensors, IDs and values? **[LO12.5]** ★★★

Review Questions

1. Decribe design-level complexity levels with an example each. **[LO 12.1]** ★
2. Compare IBM cloud platform IoTF, Bluemix, Watson analytics predictive analytics and data visualisation; and Cisco IoT, IOx and Fog cloud platforms. **[LO 12.1]** ★★★
3. How does TCS connected universe platform gather, enrich, stream, manage, acquire, organise and analyse? **[LO 12.1]** ★★
4. Describe new features in TCUP which enable easy development and deployment of IoT/M2M/IIoT applications and services. **[LO 12.1]** ★
5. Describe device and gateway domain functions in the surveillance system in the Internet of ATMs**.** **[LO12.2]** ★★
6. Describe network and applications domain functions in the surveillance system in the Internet of ATMs**.** **[LO 12.2]** ★★
7. A toy company stores the inventory of each kind of toys and labels using RFID labels at the toy packages and containers for shipping. Show a data-flow diagram and architectural reference model for the steps between shipping of toys and automated re-ordering of the toys? **[LO 12.2]** ★★★
8. Describe suitability of Oracle reference model architecture for the applications of IoTs in services and business processes based on Internet tracking of customers carrying digital devices. **[LO 12.3]** ★
9. Describe the applications feasible using the data store for Internet-tracked customers carrying connected digital devices. **[LO12.3]** ★★★
10. Describe in-car network of ECUs and infotainment devices in Internet-connected cars. **[LO 12.3]** ★
11. Describe the applications which are feasible using the driver/car user data server for Internet-connected cars. **[LO 12.3]** ★★★
12. Explain usages of IFTTT cloud service. **[LO 12.4]** ★★★
13. Describe the applications which are feasible using the vacant parking slots data server for Internet-connected cars. **[LO 12.4]** ★★★
14. How is the forest fire service facilitated by using the WSN networks, gateway and cloud platform? **[LO 12.4]** ★
15. Why do air pollution sensors measure the T, RH and P also along with the gases and particulate matter? **[LO 12.4]** ★
16. Explain JSON object coding in Java for a smart street-lighting service. **[LO 12.5]** ★★

Practice Exercises

1. An Arduino board is attached with ZigBee, GPS, Ethernet shields and a location tracker is developed to display location information. What is the complexity level for the design? **[LO 12.1]** ★

2. Draw a diagram for the devices, modules and reference model for device and gateway domain for the location tracking system described in Practice exercise 1. **[LO 12.1]** ★★
3. How will a developer use the Oracle IoT development platform for automatic chocolate vending machine services? **[LO 12.1]** ★★
4. How will a developer use Nimbits hardware and software solution for smart air quality index monitor service? How do the application-required data, messages, alerts, triggers and notification communicate on the Internet between the servers-L node and the Nimbits cloud server-S and XMPP Server-S (Figure 6.3)? **[LO 12.1]** ★★★
5. How will a developer use AWS IoT device SDK (Software Development Kit), AWS IoT and AWS service for smart air quality index monitoring service? **[LO 12.1]** ★★★
6. How will a developer use TCUP for monitoring ATM premises? **[LO 12.2]** ★★
7. What are the possible mobile apps for an Internet-based RFIDs tracking database by a toys manufacturing company at the customer and company ends? **[LO 12.2]** ★★
8. Draw an architectural view of RFIDs IDs data transfer for supply chain applications for containers tracking. **[LO 12.2]** ★
9. A famous company LEGO applies labels which also contain the labels for inlays (pieces that a child assembles and dissembles) at the manufacturing facility. The inlay labels are not as per EPC. The RFIDs based systems need EPC, (electronic product code) standards, roles and architecture. RFID based technologies effective implementation at data processing subsystem consist of reader and tag protocols, middleware architecture and EPC standards accepted by Walmart. How does four layer ITU-T architecture model the solution which would let LEGO warehouse workers apply RFID enabled labels to shipping packages of the toys and the LEGO tagging process for the toy inlays would not disrupt the manufacturing facility, order-picking and shipping process at the distribution facility? **[LO 12.2]** ★★★
10. Redraw an architectural view of usages of TCUP for customer preferences and customer services monitoring in Internet-connected cars. **[LO 12.2]** ★
11. Draw an architectural reference model of usages of a cloud platform and predictive analytics for automotive car maintenance of connected-car components. **[LO 12.3]** ★
12. How is the architectural view for connected-cars re-deployable for connected waste containers' servicing company? Assume that the company optimises the number of trips for the servicing using descriptive and predictive analytics. **[LO12.3]** ★★★
13. Compare Eclipse IoT stck platforms and AWS IoT device SDK (Software Development Kit), AWS IoT and AWS service for smart home automation and home-intrusion monitoring service. **[LO 12.4]** ★★★
14. An autonomous parking system deploys a web service for real-time reports from parking slot services. The service collects information about available parking slots. How is the openHAB development platform and my.openHAB.org cloud architectural usable for the service? **[LO12.4]** ★★
15. Section 12.5 describes a number of IoT applications and services. List the applications and services where embedded devices and sensors can deploy Raspberry Pi 2 model B+. prototype development board. List the processors and memory and OS used in RPi2. **[LO12.4]** ★
16. Build two-domains four-layer architecture reference model for smart air quality index monitoring service using openHAB. **[LO 12.4]** ★★

17. How does the smart street lighting service simultaneously provide an information network for waste containers' management service? Draw an architectural view for water containers' management service that is similar to the lighting monitoring service? ★★★
[LO 12.5]

18. Recall Example 5.1. The ACVM also displays advertisements for chocolates, news, weather reports and events in the city whenever not in use. Describe two domain architectural reference models, and show what modification in the codes for the smart street lighting service simultaneously provide information network for automatic chocolate machine supply chain management service? **[LO 12.5]** ★★★

19. Modify the code for communication of street light sensor data for additional data from five other services data, for their separate coordinators using the same WSN? ★★
[LO 12.5]

20. Describe the complete design of a location tracker using Arduino board attached with ZigBee, GPS, Ethernet shields to display and track location information. **(**Accompanying online content with the book gives the design and full codes). ★★★
[LO 12.5]

Solutions to Multiple Choice Questions

CHAPTER 1

1. (a) 2. (b) 3. (b) 4. (d) 5. (b) 6. (b) 7. (c)

CHAPTER 2

1. (a) 2. (b) 3. (b) 4. (b) 5. (d) 6. (a) 7. (c) 8. (b) 9. (a) 10. (c)
11. (b) 12. (b) 13. (a) 14. (d) 15. (d) 16. (b)

CHAPTER 3

1. (d) 2. (b) 3. (b) 4. (d) 5. (a) 6. (b) 7. (b) 8. (a) 9. (c) 10. (b)
11. (c) 12. (c)

CHAPTER 4

1. (a) 2. (b) 3. (d) 4. (d) 5. (c) 6. (b) 7. (b) 8. (c) 9. (b)

CHAPTER 5

1. (d) 2. (c) 3. (a) 4. (b) 5. (b) 6. (a) 7. (d) 8. (c) 9. (c) 10. (b)
11. (b) 12. (d) 13. (a) 14. (a) 15. (d) 16. (c)

CHAPTER 6

1. (b) 2. (d) 3. (d) 4. (c) 5. (a) 6. (b) 7. (b) 8. (a) 9. (c)

CHAPTER 7

1. (d) 2. (a) 3. (c) 4. (b) 5. (d) 6. (c) 7. (a) 8. (c) 9. (a) 10. (b)
11. (d) 12. (b)

CHAPTER 8

1. (a) 2. (b) 3. (c) 4. (d) 5. (a) 6. (a) 7. (b) 8. (c) 9. (d) 10. (d)

CHAPTER 9

1. (d) 2. (a) 3. (c) 4. (b) 5. (d) 6. (c) 7. (a) 8. (b) 9. (b) 10. (d)

CHAPTER 10

1. (a) 2. (b) 3. (c) 4. (a) 5. (b) 6. (c) 7. (d) 8. (c) 9. (b) 10. (a)

CHAPTER 11

1. (c) 2. (d) 3. (b) 4. (a) 5. (d) 6. (b)

CHAPTER 12

1. (a) 2. (b) 3. (a) 4. (d) 5. (c) 6. (b) 7. (c) 8. (c) 9. (d) 10. (d)
11. (b) 12. (a) 13. (a) 14. (b) 15. (c) 16. (d)

Bibliography

I. Print Books and E-Books

1. Adrian Mcewen, Hakin Cassimally, *Designing The Internet Of Things*, Wiley, 2015.
2. Alasdair Gilchrist, *Industry 4.0—The Industrial Internet of Things*, Apress, 2016
3. Arshdeep Bahga, Vijay Madisetti, *Internet of Things—A Hands-on Approach*, Universities Press, 2015.
4. Charalampos Doukas, *Building Internet Of Things with the Arduino: V.10,* CreateSpace/ Amazon, 2012.
5. Cuno Pfister, *Getting Started with the Internet Of Things—Connecting Sensors and Microcontrollers to the Cloud,* Maker Media Inc. O'Reilly, 2011.
6. Daniel Kellmereit, Daniel Obodovski, *The Silent Intelligence—The Internet Of Things*, DND Ventures LLC, 2013.
7. Daniel Minoli, *Building with Internet of Things with IPv6 and MPIv6—The Evolving World of M2M Communications,* Wiley, 2013.
8. Dave Evans, *The Internet of Things How the new Evolution of Internet Changing Everything*, CISCO, 2011.
9. Dieter Uckelmann, Mark Harrison, Florian Michahelles (Eds.), *Architecting The Internet Of Things*, Springer, 2011.
10. Dominique D Guinard, Vlad M Trifa, *Building the Web of Thing*s, Manning, 2016.
11. Emily Gertz, Patrick Di Justo, *Environmental Monitoring with Arduino—Building Simple Devices to Collect Data About the World Around Us*, Maker Media, Inc. O'Reilly, 2012.
12. Harvé Chabanne, Pascal Urien, Jean-Ferdinand Susini (Eds.), *RFID and The Internet of Things*, Wiley, 2013.
13. Hakima Chaouchi (Ed.), *The Internet Of Things: Connecting Objects*, Wiley, 2010.
14. Hakima Chaouchi (Ed.), *The Internet Of Things: Connecting Objects To The Web*, Wiley 2013.

15. Honbo Zhou, *The Internet of Things in the Cloud: A Middleware Perspective*, CRC Press, 2013.
16. Ian G Smith, *The Internet of Things, New Horizons*, IERC- Internet of Things European Research Cluster, 2012.
17. Jan Höller, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, *From Machine-To-Machine To The Internet Of Things: Introduction to a New Age of Intelligence,* Elsevier, 2014.
18. Jeremy Blum, *Exploring Arduino: Tools And Techniques for Engineering Wizardry*, Wiley, 2013.
19. Kris Jamsa, *Cloud Computing-SaaS, PaaS, IaaS, Virtualizatiom, Business Models, Mobile, Security, and More,* Jones & Bartlett, 2013.
20. Marco Schwartz, *Internet of Things with the Raspberry Pi: Build Internet of Things Projects using the Raspberry Pi Platform (eBook),* Open Home Automation, 2014.
21. Marco Schwartz, *Internet of Things with the Arduino Yún- Projects to help you Build a World of Smarter Things*, Packet Publications, 2014.
22. Mike Kuniavsky, *Smart Things: Ubiquitous Computing User Experience Design*, Elsevier, 2010.
23. Naveen Balani (author), Rajeev Hathi (Ed.), *Enterprise IoT: A Definitive Handbook*, CreateSpace/Google Books, 4th Ed. 2016
24. Norris, *Internet Of Things: Do-It-Yourself At Home Projects For Arduino, Raspberry Pi And Beaglebone*, McGraw-Hill, 2015.
25. Olivier Hersent, David Boswarthick, Omar Elloumi, *The Internet of Things: Key Applications and Protocols*, 2nd Ed., Wiley, 2012.
26. Peter Semmelhack, *Social Machines-How to Develop Connected Products That Change Customers*, Wiley, 2013.
27. Peter Waher, *Learning Internet of Things*, PACKT, 2015.
28. Raj Kamal, *Embedded Systems—Architecture, Programming and Design*, 3rd Ed., McGraw-Hill Education, 2015.
29. Raj Kamal, *Internet and Web Technology,* Tata McGraw Hill, 2002.
30. Raj Kamal, *Mobile Computing*, 3rd Ed. (in press), Oxford University Press, 2017.
31. Rajkumar Buyya, Amir Vahid Dastjerdi (Eds.) *Internet of Things—Principles and Paradigms*, Morgan Kaufman imprint, Elsevier, 2016
32. Rajkumar Buyya, Christian Vecchiola, S. Thamarai Selvi, *Mastering Cloud Computing,* McGraw-Hill, 2013.
33. Rob Faludi, *Building Wireless Sensor Networks- with ZigBee, XBee, Arduino, and Processing*, O'Reilly, 2010.
34. Robert Stackowiak, Art Licht, Venu Mantha, Louis Nagode. *Big Data and The Internet of Things Enterprise Information Architecture for A New Age*, Apress, 2015.
35. Samuel Greengard, *The Internet of Things*, The MIT Press Essential Knowledge series, 2015.
36. Tero Karvinen, Kimmo Karvinen, *Make: Sensors: A Hands-On Primer* For Monitoring The Real World With Arduino And Raspberry Pi, Maker Media Inc., 2014.
37. Tom Igoe, *Making Things Talk - Using Sensors, Networks, and Arduino—to See, Hear, and Feel Your World*, 2nd Ed., O'Reilly, 2011.

II. Website References

1. http://www.tcs.com/SiteCollectionDocuments/White%20Papers/Internet-of-Things-The-Complete-Reimaginative-Force.pdf. [Internet of Things: The Complete Reimaginative Force, TCS Global Trend Study, July 2015]
2. http://www.gartner.com/newsroom/id/2970017. [Gartner, "Gartner Says by 2020, a Quarter Billion Connected Vehicles Will Enable New In-Vehicle Services and Automated Driving Capabilities", Jan., 2015]
3. http://www.caterpillar.com/en/news/caterpillarNews/history/110-years-pass-since-the-first-test-of-the-steam-powered-track-type-tractor.html. [Caterpillar, "110 Years Pass Since the First Test of the Steam-Powered Track-Type Tractor"]
4. http://www.tcs.com/.../Brochures/TCS-Connected-Car-Solutions-1014-1.pdf
5. http://fortune.com/2015/03/05/the-race-to-the-internet-of-things/ [Fortune, "The race to the Internet of things," March, 2015]
6. http://postscapes.com/internet-of-things-books [A presentation of Books on Internet of Things]

III. Print and e-Journals

1. http://www.iotjournal.com/
2. http://www.rfidjournal.com/
3. www.iot-j.ieee.org, IEEE Internet of Things Journal, Joint Publications of IEEE Sensors Council, IEEE Communications Society, IEEE Computer Society, IEE Signal Processing Society
4. http://www.journals.elsevier.com/journal-of-network-and-computer-applications/ Elsevier Journal of Network and Computer Applications

IV. e-Journal Paper References

1. http://www.emeraldinsight.com/loi/bpmj, *Internet of Things: Applications and Challenges in Smart Cities: a Case Study of IBM Smart City Projects*, Business Process Management Journal, Volume 22, Issue 2, pp 263, 2016.
2. http://iot-analytics.com/product/iot-platforms-market-report-2015-2021-3/?utm_source=Google%20AdWords&utm_medium=CPC&utm_content=Textanzeigen&utm_campaign=SEA%20-%20iot%20platform%20market%20report, IoT Platforms: Market Report 2015-2021, 2016.
3. http://www.ibm.com/internet-of-things/trial.html?cm_mmc=search-gsn-_-unbranded-watson-iot-search-_-internet%20of%20things%20(Broad%20and%20Phrase%20Match%20Only)-phrase-_-ROW-iot-mkt-oww Start Developing with IBM Watson IoT Platform.
4. http://www.itu.int/en/ITU-T/Workshops-and-Seminars/iot/201402/Pages/default.aspx, Internet of Things – Trends and Challenges in Standardization, Geneva, Switzerland, 2014.