SGOU
Higher Education for All

INTERNET OF THINGS

COURSE CODE: B21CA24DE
Bachelor of Computer Applications
Discipline Specific Elective Course
Self Learning Material



SGOU

SREENARAYANAGURU OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SREENARAYANAGURU OPEN UNIVERSITY

SGOU

Vision

To increase access of potential learners of all categories to higher education, research and training, and ensure equity through delivery of high quality processes and outcomes fostering inclusive educational empowerment for social advancement.

Mission

To be benchmarked as a model for conservation and dissemination of knowledge and skill on blended and virtual mode in education, training and research for normal, continuing, and adult learners.

Pathway

Access and Quality define Equity.

Internet of Things

Course Code: B21CA24DE

Semester - VI

Discipline Specific Elective Course
Undergraduate Programme
Bachelor of Computer Applications
Self Learning Material

(With Model Question Paper Sets)

SGOU

SREENARAYANAGURU
OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SREENARAYANAGURU
OPEN UNIVERSITY

INTERNET OF THINGS

Course Code: B21CA24DE
Semester- VI
Discipline Specific Elective Course
Bachelor of Computer Applications

Academic Committee

Dr. M.V. Judy
Dr. Aji S.
Dr. Vishnukumar S.
Rajesh R.
Dr. Rafidha Rehiman K.A.
P.M. Ameera Mol
Dr. Ajitha R.S.
Dr. Bindu Lal T.S.
Dr. Sreeja S.

Development of the Content

Dr. John T. Abraham

Review

Dr. Shanu N.

Edit

Dr. Shanu N.

Proofreading

Dr. Shanu N.

Scrutiny

Shamin S.
Subi Priya Laxmi S.B.N.
Dr. Kanitha D.K.
Greeshma P.P.
Aswathy V.S.
Sreerekha V.K.

Cover Design

Jobin J.

Co-ordination

Director (IC), MDDC :
Dr. Sajeevkumar G.
Coordinator, Development:
Dr. Anfal M.
Coordinator, Distribution:
Dr. Abdul Razak Kunnathodi

SGOU

Scan this QR Code for reading the SLM on a digital device.

Edition
October 2026

Copyright
© Sreenarayanaguru Open University

ISBN 978-81-903304-6-6

9 788190 330466

All rights reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from Sreenarayanaguru Open University. Printed and published on behalf of Sreenarayanaguru Open University by Registrar, SGOU, Kollam.

www.sgou.ac.in

Visit and Subscribe our Social Media Platforms

MESSAGE FROM VICE CHANCELLOR

Dear

With immense joy and excitement, I extend my heartfelt greetings to all of you and warmly welcome you to Sreenarayanaguru Open University.

Established in September 2020 as a state-driven initiative, Sreenarayanaguru Open University is dedicated to advancing higher education through open and distance learning. Our vision is guided by the principle of "access and quality define equity," laying the foundation for a celebration of excellence in education. I am delighted to share that we are steadfast in our commitment to uphold the highest standards and refrain from compromising on the quality of education we offer. The university draws its inspiration from the legacy of Sreenarayana Guru, a revered figure in the Indian renaissance movement. His name serves as a constant reminder for us to prioritize quality in all our academic endeavors.

Sreenarayanaguru Open University operates within the practical framework of the widely recognized "blended format." Acknowledging the constraints faced by distance learners in accessing traditional classroom settings, we have curated a pedagogical approach centered on three main components: Self Learning Material, Classroom Counselling, and Virtual Modes. This comprehensive blend is poised to deliver dynamic learning and teaching experiences, maximizing engagement and effectiveness. Our unwavering commitment to quality ensures excellence across all aspects of our educational initiatives.

The University aims to offer you an engaging and stimulating educational environment that fosters active learning. The SLM is designed to offer a comprehensive and cohesive learning experience, fostering a deep interest in the study of technological advancements in IT. Careful consideration has been given to ensure a logical progression of topics, facilitating a clear understanding of the discipline's evolution. The curriculum is thoughtfully crafted to provide ample opportunities for students to navigate through the current trends in information technology. Furthermore, this course is designed to provide essential insights into computer hardware, software classification, and foundational HTML concepts crucial for web development.

We assure you that the university student support services will closely stay with you for the redressal of your grievances during your studentship. Feel free to write to us about anything that seems relevant regarding the academic programme.

Wish you the best.



Regards,
Dr. Jagathy Raj V.P.

01-10-2026

SGOU

Contents

BLOCK 1 IoT – Applications & Architectures **1**
UNIT 1 Introduction to IoT and its key features 2
UNIT 2 IoT Applications 14
UNIT 3 Convergence of IT and OT 22
UNIT 4 IoT Architecture 31

BLOCK 2 Engineering IoT Networks **41**
UNIT 1 Smart Objects 42
UNIT 2 Communication Criteria 50
UNIT 3 Communication Protocols 57
UNIT 4 IoT Access Technologies 64

BLOCK 3 IP as IoT Network Layer **71**
UNIT 1 Business Case for IP 72
UNIT 2 IP Adoption or Adaptation for IoT 85
UNIT 3 Need for Optimization of IP 99
UNIT 4 Optimizing IP For IoT 111

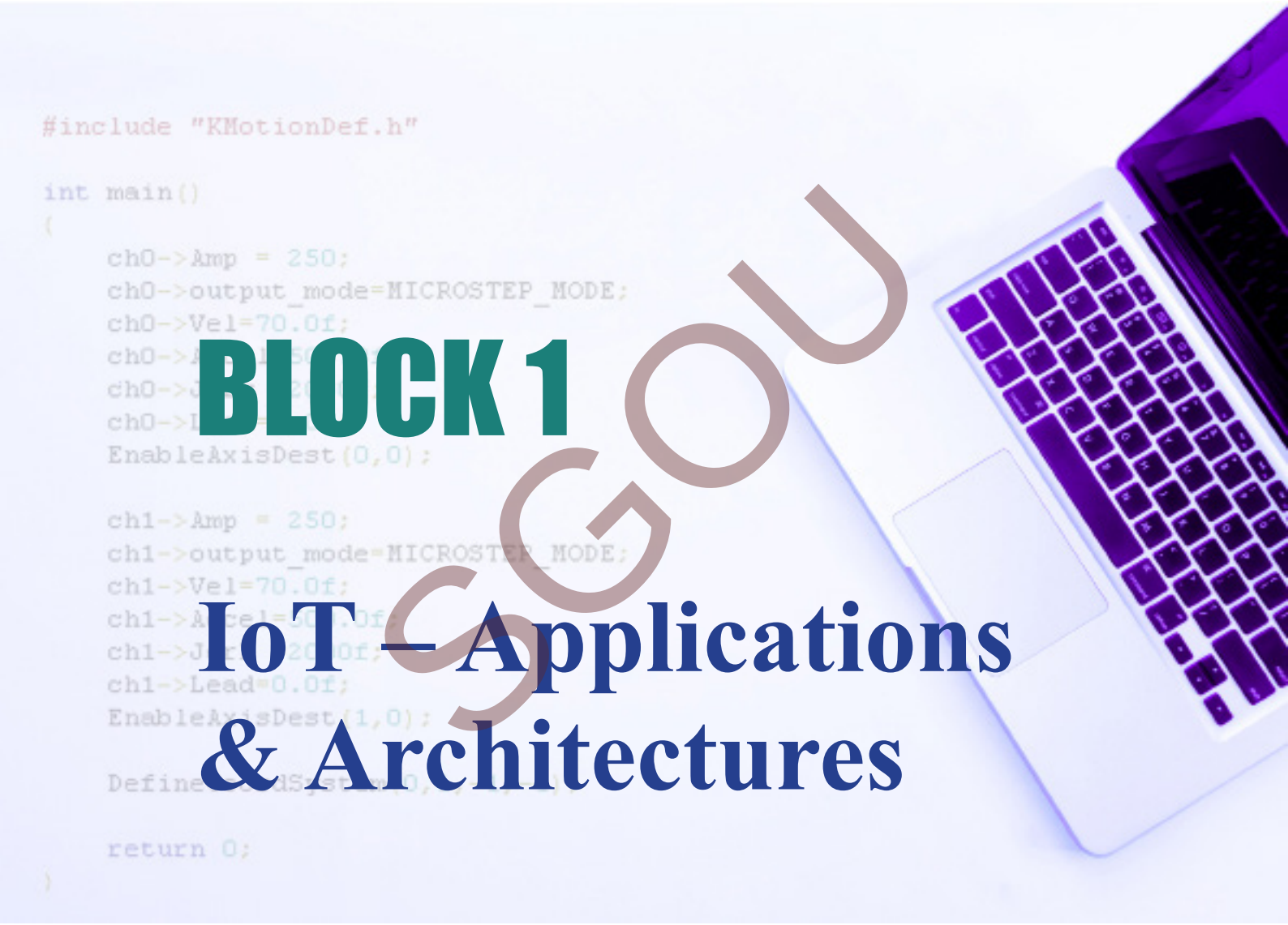
BLOCK 4 Transport Layer & Application Protocols **121**
UNIT 1Transport Layer 122
UNIT 2 Supervisory Control and Data Acquisition (SCADA) 132
UNIT 3 Generic Web-Based Protocols 141
UNIT 4 IoT Application Layer Protocols 151

BLOCK 5 Data and Analytics for IoT **160**
UNIT 1 Introduction to Data Analytics for IoT 161
UNIT 2 Big Data Analytics Tools and Technology 172
UNIT 3 Edge Streaming Analytics 183
UNIT 4 Network Analytics 193

BLOCK 6 IoT Security & Introduction to Arduino and Raspberry Pi **203**
UNIT 1 Common Challenges in Operational Technology (OT) Security 204
UNIT 2 Formal Risk Analysis Structures – OCTAVE & FAIR 234
UNIT 3Arduino – Basics on Integration of Sensors and Actuators 267
UNIT 4 Implementation of IoT using Raspberry Pi 305

Model Question Paper Sets **330**

SGOU



BLOCK 1

IoT – Applications & Architectures

SGOU



UNIT 1

Introduction to IoT and its Key Features

Learning Outcomes

After completion of this unit, learner will be able to:

- define the Internet of Things and recall its defining characteristics
- identify the basic building blocks of a typical IoT system
- explain the four-layer architecture used to organise IoT systems
- recognize the key technologies that enable IoT communication
- describe common real-world areas where IoT is applied

Prerequisite

Think about the last time you woke up to a phone alarm that also switched on your room lights, used a fitness band that quietly counted your steps throughout the day, or controlled your home air conditioner through an app while you were away. You may have also seen a car using sensors to assist with parking. These devices can sense their surroundings, exchange information, and perform actions with little direct human involvement.

This everyday experience of "things" communicating and working with one another is based on a simple idea: the Internet of Things (IoT). Before learning the concept formally, look around you and identify five devices that may be collecting or exchanging data without requiring your constant attention.

Examples include:

1. A smartphone synchronizing notifications and updates.
2. A Wi Fi router managing connections between devices.
3. A smart television downloading updates or streaming data.
4. A smartwatch tracking health information and sending it to a mobile application.

SGOU

5. A smart electricity meter transmitting usage data to the utility provider.

Many connected devices exchange information in the background, often without users noticing the communication.

Keeping these examples in mind will make the concepts in this unit easier to understand, because you can connect each new IoT concept with devices and situations that you encounter in everyday life.

Keywords

Sensor, Actuator, Embedded System, Connectivity, Cloud Computing, Gateway, Machine-to-Machine (M2M), RFID

Discussion

1.1.1 What is the Internet of Things?

The Internet of Things (IoT) refers to a network of physical objects, including devices, vehicles, appliances, machines, and even everyday items. These objects are equipped with sensors, software, and communication hardware that enable them to collect data, exchange it across a network, and often respond to that data with little or no direct human involvement. The term was coined in 1999 by Kevin Ashton, a British technologist, while describing how RFID tags could allow computers to manage inventory without people having to type in data manually. Since then, the idea has grown far beyond inventory tags to cover almost every category of device that can be made "smart" by adding sensing and connectivity to it.

A useful way to understand IoT is to contrast it with a traditional computer network. A traditional network mainly connects computers and mobile phones that are operated directly by people. An IoT network, in contrast, mainly connects physical objects that were traditionally considered "dumb," such as a streetlight, refrigerator, irrigation valve, or factory motor. It enables these objects to collect, generate, and share data independently, without requiring constant human input or supervision.

In an IoT system, humans move from being active operators of every device to becoming supervisors who occasionally check dashboards or receive alerts. The devices themselves can handle routine tasks such as sensing, communication, and, in many cases, decision-making.

Key Features of an IoT System

An IoT system generally has four key features. These features help us understand how an IoT system collects data, communicates, processes information, and produces useful results.

SGOU

1. **Sensing :** The device can measure or detect physical conditions such as temperature, motion, light, pressure, humidity, or location. Sensors collect information from the surrounding environment.

2. **Connectivity :** The device can send and, in most cases, receive data through a network. Communication may take place using technologies such as Bluetooth, Wi-Fi, cellular networks, or LPWAN.

3. **Processing :** The collected data is processed or analysed to obtain useful information. This processing may take place on the device itself, known as edge processing, or on a remote server or cloud platform, known as cloud processing.

4. **Action or Insight :** The results of processing can lead to an automatic action or provide useful information to a person. For example, an actuator may switch a device on or off, or the system may send an alert, generate a report, or display information on a dashboard.

According to Cisco Internet Business Solutions Group (Cisco IBSG, 2012), the number of connected devices worldwide was estimated to have reached 12.5 billion in 2010, compared with a global population of approximately 6.8 billion. Thus, by 2010, the number of connected devices had already exceeded the number of people in the world.

The term "Internet of Things" (IoT) is relatively recent. The phrase was first used by Kevin Ashton in 1999 in a presentation to Procter & Gamble, in the context of using RFID and the Internet to improve supply-chain information.

1.1.2 A Brief Look at the Evolution of IoT

IoT did not emerge overnight. Instead, it developed through decades of progress in three separate fields: embedded computing, wireless communication, and the Internet itself. Over time, advances in these areas came together and made the growth of IoT possible.

Early machine to machine, or M2M, systems in the 1980s and 1990s enabled individual machines, such as vending machines and utility meters, to send simple status information through telephone lines. As wireless networks became more widespread, sensors became less expensive, and microcontrollers became more affordable during the 2000s, it became economically practical to place a small computer, a sensor, and a radio inside everyday objects.

The rapid growth of smartphones after 2007 provided millions of people with a simple and familiar way to interact with connected devices through mobile applications. This accessibility significantly accelerated the adoption of IoT technologies among consumers.

SGOU

Table 1.1.1 Milestones in the Evolution of IoT

Period	Development
1980s–1990s	Early M2M systems enabled machines such as vending machines and utility meters to exchange data through telephone and dedicated lines.
1999	Kevin Ashton coined the term **"Internet of Things"** while working on RFID-based inventory tracking.
2000s	Falling costs of sensors, microcontrollers, and wireless communication components made it affordable to add sensing and connectivity to ordinary objects.
2008–2010	The number of connected devices exceeds the world's human population for the first time.
2010s	Cloud platforms, smartphone apps and low-power wide-area networks (LPWAN) enable large-scale consumer and industrial IoT.
2020s	IoT combines with artificial intelligence and 5G to support real-time, large-scale applications such as smart cities and connected factories.

1.1.3 Building Blocks of an IoT System

An IoT system, whether it is a small fitness band or a large citywide traffic network, consists of four main functional blocks that work together in a continuous cycle, as shown in Fig. 1.1.1. These four blocks are Sensing, Connectivity, Data Processing, and Action and Insight. Each block performs a specific function, and together they allow physical objects to collect data, communicate with other systems, analyse information, and respond to changing conditions.

1. Sensing

Sensing is the first stage of an IoT system. At this stage, sensors collect information about the physical environment or the condition of an object. A sensor is a device that detects or measures a physical quantity and converts it into data that can be understood and processed by a computer system.

Different IoT applications use different types of sensors depending on what needs to be monitored. For example, a temperature sensor measures temperature, a humidity sensor measures moisture in the air, a motion sensor detects movement, a light sensor measures the intensity of light, and a GPS sensor or module can provide location

SGOU

information. Wearable devices can use sensors to measure parameters such as heart rate, movement, and physical activity.

The sensor continuously or periodically collects data from the environment. The frequency of data collection depends on the application. For example, an autonomous vehicle may need sensor data many times per second, while a smart electricity meter may collect and transmit readings at longer intervals.

For example, consider a smart agriculture system. Soil-moisture sensors are placed in the field to measure the amount of moisture present in the soil. When the soil becomes dry, the sensor detects the change and produces a corresponding data value. This data is then passed to the next stage of the IoT system.

2. Connectivity

After data is collected by the sensors, it must be transmitted to another device or system for further processing. This is the role of the connectivity stage. Connectivity provides the communication link between the IoT device and other components such as a smartphone, gateway, edge device, server, or cloud platform.

IoT devices can use different communication technologies depending on the application, distance, power requirements, and amount of data that needs to be transmitted. Common technologies include Wi-Fi, Bluetooth, cellular networks, Zigbee, Ethernet, and Low-Power Wide-Area Networks (LPWAN).

For example, a fitness band may use Bluetooth to send activity data to a smartphone. A smart home device may use Wi-Fi to communicate with a home network. A smart electricity meter installed in a remote location may use a cellular or LPWAN connection to transmit meter readings to a utility provider.

In some IoT systems, the sensor device does not directly communicate with the cloud. Instead, the data is first sent to a gateway. The gateway collects data from one or more IoT devices and forwards it to a server or cloud platform. It may also perform some initial processing before sending the data further.

3. Data Processing

Once the sensor data reaches the appropriate device, gateway, server, or cloud platform, it must be processed and analysed to make it useful. This stage is known as data processing.

Raw sensor data may contain large amounts of information, including unnecessary or repeated values. Processing helps to organise the data, remove unnecessary information, identify important values, and determine whether a particular condition has occurred.

Data processing can take place in different locations. When processing is performed directly on the IoT device, it is known as device-level processing. When processing is performed close to the source, such as on a gateway or edge device, it is called edge processing. When data is sent to a remote server or cloud platform for processing, it is known as cloud processing.

SGOU

The type of processing depends on the requirements of the application. Applications that require a very fast response may perform processing close to the device. Applications that require large-scale data storage and analysis may use cloud platforms.

For example, in a smart agriculture system, the collected soil-moisture reading can be compared with a predefined moisture level. If the moisture level is below the required value, the system can identify that the soil is dry. The processed result can then be passed to the next stage.

4. Action and Insight

The final stage is Action and Insight. After the data has been processed, the resulting information can be used either to perform an automatic action or to provide useful information to a human user.

An action occurs when the IoT system automatically responds to a particular condition. This is usually performed using an actuator. An actuator is a device that converts a control signal into a physical action. Examples include motors, switches, valves, relays, and other control devices.

For example, in a smart agriculture system, if the processed data shows that the soil is too dry, the system can automatically activate a water pump through an actuator. When the required moisture level is reached, the system can turn the pump off.

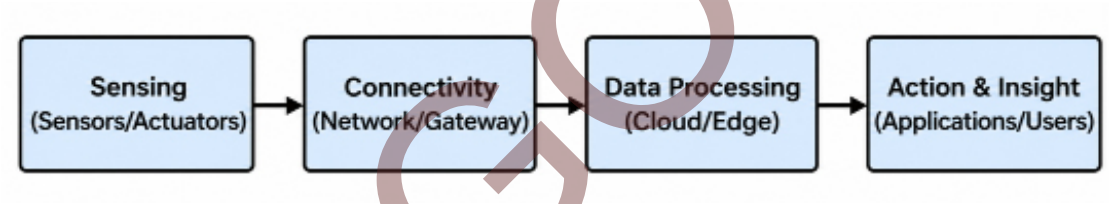


Fig. 1.1.1 Working Principle of an IoT System

It is important to note that this process usually does not happen only once. In most real-world IoT systems, sensing, data transmission, processing, and action take place continuously. Depending on the application, this cycle may occur many times per second, as in an autonomous vehicle, or once every few minutes, as in a smart electricity meter.

By continuously collecting and analysing fresh data, an IoT system can respond to changing conditions in real time. This continuous operation helps the system make decisions based on current information rather than outdated data. Thus, the continuous interaction between sensing, connectivity, processing, and action enables IoT systems to monitor physical environments and respond intelligently with minimal human intervention.

1.1.4 The Layered Architecture of IoT

To design, build and troubleshoot IoT systems in a disciplined way, engineers organise their components into layers, much as computer networks are organised using the OSI model. While several layered models exist in the literature, a simple and widely taught version uses four layers, shown in Fig. 1.1.2. the Perception Layer, the Network Layer, the Processing (or Middleware) Layer, and the Application Layer.

SGOU

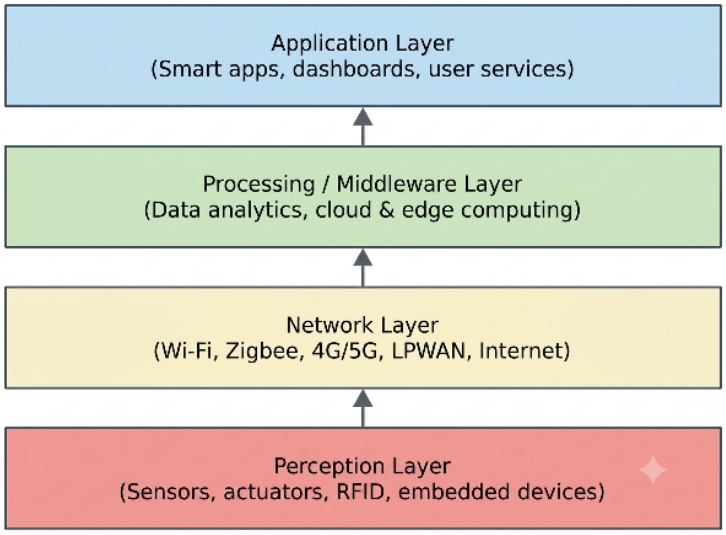


Fig. 1.1.2 The Four-Layer Architecture of IoT

1.1.4.1 Perception Layer

The perception layer, which is also known as the sensing layer, is the part of an IoT system that is closest to the physical environment. It includes the sensors and actuators that directly interact with the surroundings being observed or controlled. Examples include a temperature sensor in a greenhouse, an accelerometer in a wearable device, an RFID tag attached to a warehouse pallet, and a motor that operates an irrigation valve.

This layer is responsible for collecting data from the physical world and initiating actions when required. As a result, it serves as the primary link between the real environment and the rest of the IoT system.

This layer converts physical, real-world quantities into electrical signals that the rest of the system can work with, and it converts digital commands back into physical action through actuators.

1.1.4.2 Network Layer

The network layer is responsible for carrying the data collected at the perception layer to wherever it needs to be processed. Depending on the distance involved and the amount of power available to the device, this layer may use short-range technologies such as Bluetooth, Zigbee or Wi-Fi, or longer-range technologies such as 4G/5G cellular networks and Low-Power Wide-Area Networks (LPWAN) like LoRaWAN and NB-IoT. A device that monitors a single room may only need Wi-Fi, while a device that tracks a shipping container across a country will typically rely on cellular or satellite connectivity.

1.1.4.3 Processing (Middleware) Layer

Raw sensor readings are rarely useful on their own; the processing layer is where this data is filtered, aggregated, stored and analysed. Some of this analysis happens close to the sensor itself, an approach called edge computing, which reduces delay and saves

SGOU

bandwidth. The remaining data is typically sent to cloud computing platforms, which provide significantly greater storage capacity and processing power. These platforms can collect and analyze data from thousands of devices simultaneously. For example, they can identify traffic patterns across an entire city rather than focusing only on the behavior of a single vehicle.

1.1.4.4 Application Layer

The application layer is where the processed information finally becomes useful to a person or an organisation. This layer includes the dashboards, mobile apps, alerts and automated business rules that let a farmer check soil moisture from a phone, let a hospital track the location of medical equipment, or let a factory manager receive an alert the moment a machine starts to vibrate abnormally. Because this layer is what users actually see and interact with, its design has a major influence on whether an IoT system is judged useful in practice.

1.1.5 Enabling Technologies of IoT

IoT is made possible by several technologies that work together to connect physical objects, collect data, communicate information, and provide useful services. These technologies support the different functions of an IoT system.

1. **Sensors and Actuators :** Sensors collect information from the physical environment, such as temperature, humidity, motion, pressure, and light. Actuators perform actions based on instructions, such as switching on a motor, opening a valve, or turning on a light. Many of these devices use Micro-Electro-Mechanical Systems (MEMS), which help make sensors and actuators small and suitable for use in IoT devices.

2. **Identification Technologies :** Technologies such as RFID (Radio Frequency Identification) and NFC (Near Field Communication) help identify and exchange information with objects without requiring a physical connection. For example, RFID can be used to identify products in a warehouse.

3. **Communication Protocols :** IoT devices need communication protocols to exchange data with other devices and systems. Protocols such as MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol) are designed to work efficiently with devices that have limited processing power, memory, and battery capacity.

4. **Cloud and Edge Computing :** Cloud computing provides large-scale storage and computing resources for storing and analysing IoT data. Edge computing processes data closer to where it is generated, which can reduce delay and support faster responses. Together, cloud and edge computing help IoT systems handle large amounts of data efficiently.

1.1.6 Advantages and Challenges of IoT

IoT provides several advantages by enabling devices to continuously collect and exchange data with minimal human involvement. This can improve efficiency, safety,

SGOU

convenience, and decision-making in many areas. However, connecting a large number of devices also creates challenges related to security, privacy, data management, connectivity, and system complexity. Table 1.1.2 summarises the major advantages and challenges associated with IoT.

Table 1.1.2 Advantages and Challenges of IoT

Advantages	Challenges
Continuous, automated monitoring reduces manual effort.	Large numbers of devices increase the attack surface for security threats.
Faster, data-driven decisions based on real-time information.	Managing and storing the huge volume of data generated is demanding.
Improved efficiency and reduced operating costs over time.	Devices from different manufacturers may use incompatible standards.
Enables new services such as predictive maintenance and remote care.	Limited battery life and processing power on many small devices.

1.1.7 Application Domains of IoT: A Preview

The four-layer architecture and enabling technologies introduced in this unit can be applied in many different areas using suitable sensors, devices, and applications. For example, connected roadways use sensors and cameras to monitor traffic and improve transportation. Connected factories use sensors to monitor machines and identify possible problems before failures occur. Connected buildings use sensors to manage lighting, heating, security, and other systems more efficiently.

IoT is therefore used in a wide range of environments to improve efficiency, safety, convenience, and decision-making. Unit 2 of this block focuses on three important application domains: Connected Roadways, Connected Factories, and Connected Buildings. It explains how the basic concepts introduced in this unit are applied in these environments and how IoT helps address their specific requirements and challenges.

Recap

- IoT connects physical objects using sensors, software and communication hardware so that they can exchange data with little human involvement.
- The term "Internet of Things" was coined by Kevin Ashton in 1999 in the context of RFID-based inventory tracking.
- A genuine IoT system shows four properties together: sensing, connectivity, processing, and resulting action or insight.

SGOU

- IoT evolved from early machine-to-machine (M2M) telemetry through the falling cost of sensors and wireless chips.
- Every IoT system, regardless of size, repeats a loop of sensing, transmission, processing and action.
- The IoT architecture is commonly divided into four layers: Perception, Network, Processing (Middleware) and Application.
- The Perception Layer contains sensors and actuators that interact directly with the physical environment.
- The Network Layer carries data using technologies ranging from Wi-Fi and Zigbee to cellular and LPWAN.
- The Processing Layer uses edge computing and cloud computing to analyse and store data.
- The Application Layer delivers dashboards, alerts and automated actions to end users.
- Enabling technologies for IoT include MEMS sensors, RFID/NFC identification, lightweight protocols such as MQTT/CoAP, and cloud platforms.
- IoT offers advantages such as automation and efficiency, but also introduces challenges such as security risk and device incompatibility.
- Application domains of IoT include connected roadways, connected factories and connected buildings, among many others.

SGOU

Objective Type Questions

1. Who coined the term "Internet of Things" and in what year?
2. What are the four properties that together define a genuine IoT system?
3. What is the name given to the layer that contains sensors and actuators?
4. Which layer of the IoT architecture is responsible for carrying data across a network?
5. What term describes data processing carried out close to the sensor rather than in the cloud?
6. Which two identification technologies allow objects to be recognised without physical contact?
7. Name one lightweight communication protocol designed for low-power IoT devices.

8. What is the term for machines communicating with each other without human involvement?
9. Which layer of the IoT architecture delivers dashboards and alerts to end users?
10. What technology reduced the physical size and cost of modern sensors?
11. Which decade saw the number of connected devices first exceed the world's population?
12. What do the initials LPWAN stand for?

Answers

1. Kevin Ashton, 1999.
2. Sensing, connectivity, processing, and action/insight.
3. Perception Layer.
4. Network Layer.
5. Edge computing.
6. RFID and NFC.
7. MQTT (or CoAP).
8. Machine-to-Machine (M2M).
9. Application Layer.
10. MEMS (micro-electromechanical systems).
11. 2008–2010.
12. Low-Power Wide-Area Network.

SGOU

Assignments

1. Explain the meaning of the Internet of Things using an example from your own daily life, and identify the sensing, connectivity, processing and action elements in it.
2. Describe, with a diagram, the four-layer architecture of IoT and explain the role of each layer.

3. Compare edge computing and cloud computing as approaches to processing IoT data, giving one situation where each would be preferred.
4. Trace the evolution of IoT from early machine-to-machine systems to present-day large-scale deployments.
5. Discuss any four enabling technologies of IoT and explain why each is important.
6. Analyse two advantages and two challenges of deploying IoT at a large scale, such as across a city.
7. Differentiate between RFID and NFC as identification technologies used in IoT systems.

Suggested Reading

1. Greengard, Samuel. *The Internet of Things*. MIT Press, 2015.
2. Hanes, David, et al. *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Cisco Press, 2017.
3. Buyya, Rajkumar, and Amir Vahid Dastjerdi, editors. *Internet of Things: Principles and Paradigms*. Morgan Kaufmann, 2016.
4. Bahga, Arshdeep, and Vijay Madisetti. *Internet of Things: A Hands-On Approach*. Universities Press, 2015.

Reference

1. Ashton, Kevin. "That 'Internet of Things' Thing." *RFID Journal*, 2009.
2. Atzori, Luigi, et al. "The Internet of Things: A Survey." *Computer Networks*, vol. 54, no. 15, 2010, pp. 2787–2805.
3. Gubbi, Jayavardhana, et al. "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions." *Future Generation Computer Systems*, vol. 29, no. 7, 2013, pp. 1645–1660.

SGOU



IoT Applications

Learning Outcomes

After completion of this unit, learner will be able to:

- identify the key components of a connected roadway system
- explain how IoT supports predictive maintenance in a connected factory
- describe the role of a Building Management System in a connected building
- recognize common benefits and challenges shared across IoT application domains

Prerequisite

Recall the four-layer architecture and the sensing-connectivity-processing-action loop you studied in Unit 1. Now picture three everyday scenes. First, a commuter stuck at a signal that stays red even though no cross-traffic is present. Second, a factory worker who discovers a motor has failed only after production has already stopped. Third, an office where the air-conditioning runs at full strength in an empty conference room over a weekend.

In each of these situations, a modest amount of sensing capability and automated decision making, which are the core elements you have already studied, could have prevented the loss of time, the disruption of production, or the unnecessary consumption of electricity.

This unit takes the general IoT architecture you already know and shows exactly how it is applied to roads, factories and buildings, so that you begin to see the same four layers repeating in very different settings.

Keywords

Smart Traffic Management, Vehicle-to-Infrastructure (V2I), Industrial IoT (IIoT), Predictive Maintenance, Building Management System (BMS), Smart Lighting, Adaptive Signal Control, Asset Tracking

SGOU

Discussion

1.2.1 Introduction to IoT Application Domains

The same building blocks introduced in Unit 1, including sensors in the perception layer, suitable communication networks, data processing at the edge or in the cloud, and applications that present information to users, can be found in almost every IoT deployment. What mainly varies across different application domains is the type of sensors used, the scale at which the system operates, and the specific decisions that are automated.

This unit explores three widely recognized IoT application domains: connected roadways, connected factories, and connected buildings. Each domain demonstrates how the core principles of IoT are applied in practice while addressing different requirements, priorities, and implementation choices.

1.2.2 Connected Roadways

A connected roadway uses sensors embedded in or alongside the road, combined with communication between vehicles and infrastructure, to manage traffic more intelligently than fixed-time signals or manual monitoring ever could. The central idea is Vehicle-to-Infrastructure (V2I) communication, in which vehicles share information such as speed and position with roadside equipment, and the roadside equipment shares information such as signal timing or hazard warnings back with vehicles.

1.2.2.1 Components of a Connected Roadway

As Fig. 1.2.1 shows, a connected roadway typically combines vehicle-mounted sensors such as GPS receivers and on-board diagnostics (OBD) units, roadside units carrying traffic cameras and inductive-loop or radar sensors, a communication network linking these to a central point, and a Traffic Management Centre where cloud-based analytics decide how to adjust signals or issue alerts.

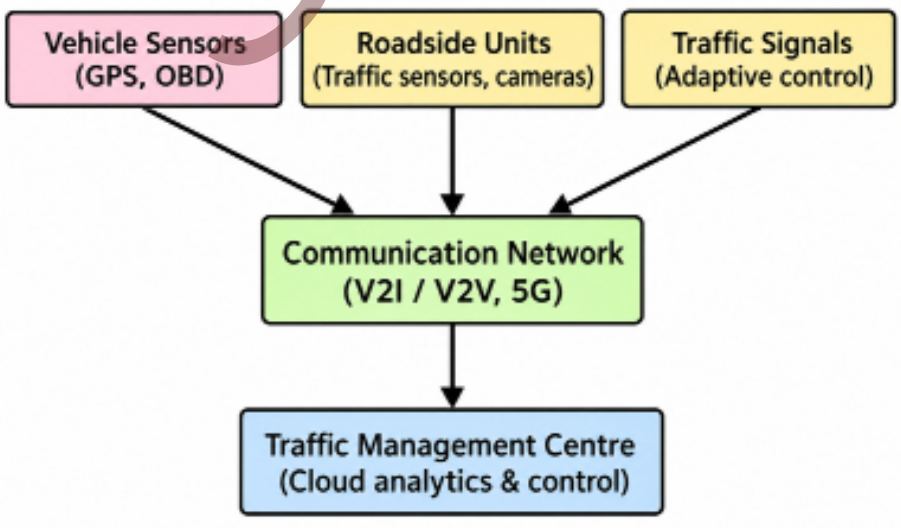


Fig. 1.2.1 Concept of a Connected Roadway System

SGOU

1.2.2.2 Illustrative use Cases

Adaptive traffic signal control changes signal timing in real time based on actual vehicle counts, rather than a fixed schedule, which reduces waiting time at low-traffic hours. Smart parking systems use ground sensors to detect empty parking bays and guide drivers directly to them through a mobile app, cutting down the time spent circling for parking. Automatic incident detection uses cameras and sensor data to spot stopped vehicles or sudden braking patterns, alerting emergency services faster than a member of the public might.

1.2.3 Connected Factory (Industrial IoT)

When IoT is applied inside a manufacturing plant, it is generally referred to as Industrial IoT, or IIoT. Here, sensors are attached to machines, robots and production lines to continuously track variables such as temperature, vibration and running hours, which were traditionally checked only occasionally by a maintenance technician walking the factory floor.

1.2.3.1 Components of a Connected Factory

Fig.1.2.2 shows how sensors fitted on machines and robots feed data through Programmable Logic Controllers (PLCs) or dedicated sensor modules to an industrial gateway, which forwards it to a cloud or edge analytics platform. The results are then shown on a dashboard that plant managers use to plan maintenance and monitor overall production health.

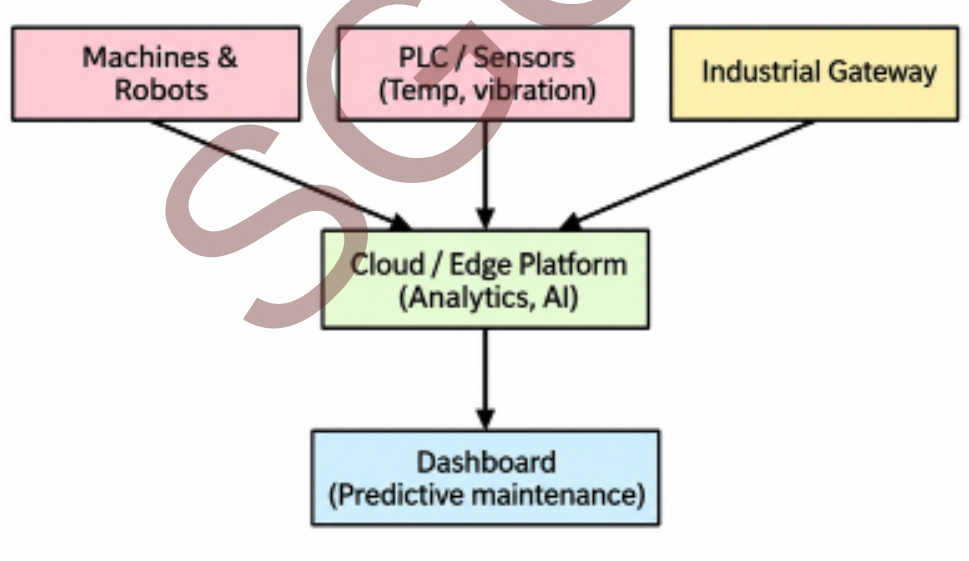


Fig. 1.2.2 Concept of a Connected Factory (Industrial IoT)

1.2.3.2 Illustrative use Cases

Predictive maintenance analyses vibration and temperature trends to flag a machine that is likely to fail in the coming days, allowing repairs to be scheduled before an unplanned breakdown halts production. Asset tracking uses RFID or Bluetooth tags to locate tools, materials and finished goods within a large facility in real time. Automated

SGOU

quality control uses cameras and sensors on the production line to detect defective items immediately, rather than after a batch has already been packaged.

Note: Manufacturers who adopt predictive maintenance report reducing unplanned equipment downtime by a significant margin compared with traditional "fix it after it breaks" or fixed-schedule maintenance approaches, because failures are anticipated using sensor trends rather than discovered after the fact.

1.2.4 Connected Buildings

A connected, or "smart," building uses sensors and automated control systems across facilities that were traditionally managed separately, such as heating and cooling, lighting, security, and fire safety. These systems are integrated through a Building Management System (BMS), allowing the entire building to operate in a more efficient, coordinated, and safe manner.

By bringing these functions together, the building can monitor conditions continuously, optimize energy usage, improve occupant comfort, enhance security, and respond more effectively to potential safety issues.

1.2.4.1 Components of a Connected Building

As shown in Fig.1.2.3, sensors distributed across HVAC (heating, ventilation and air-conditioning) equipment, lighting fixtures, access-control doors and fire-detection systems all report into a central Building Management System, which in turn presents summarised information and controls through a facility manager's dashboard.

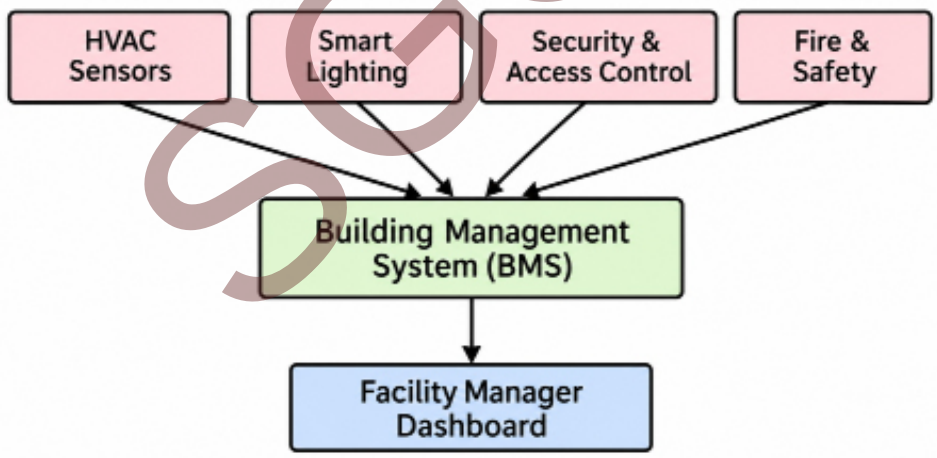


Fig. 1.2.3 Concept of a Connected (Smart) Building

1.2.4.2 Illustrative use Cases

Occupancy-based HVAC control reduces heating or cooling in rooms detected to be empty, cutting energy use without any change in comfort for occupants who are actually present. Smart lighting dims or switches off lights automatically based on daylight availability and occupancy sensors. Integrated security combines access-control logs, motion sensors and camera feeds so that unusual activity after hours can be flagged automatically rather than relying solely on a security guard noticing it.

SGOU

1.2.5 A Comparative View of the Three Domains

Although connected roadways, connected factories and connected buildings serve very different purposes, Table 1.2.1 shows how each still maps cleanly onto the sensing, connectivity, processing and action pattern introduced in Unit 1.

Table 1.2.1 Comparing Connected Roadways, Factories and Buildings

Domain	Typical Sensors	Primary Goal
Connected Roadway	GPS, inductive loops, traffic cameras	Reduce congestion and improve road safety
Connected Factory	Vibration, temperature, RFID tags	Reduce downtime and improve product quality
Connected Building	Occupancy, temperature, motion sensors	Reduce energy use and improve occupant safety

1.2.6 Common Benefits and Challenges

Across all three domains, the greatest benefit of connecting previously isolated systems is the ability to act on real-time information rather than on fixed schedules or occasional manual inspection, which generally improves efficiency, safety and cost savings. The challenges are similarly shared: a large number of connected sensors and controllers increases the number of points that must be secured against interference or attack, and combining equipment from different vendors, each following different communication standards, can make integration difficult without careful planning.

SGOU

Recap

- The sensing, connectivity, processing and action pattern is commonly used across connected roadways, factories and buildings.
- Connected roadways use Vehicle-to-Infrastructure (V2I) communication to improve traffic management and road safety.
- Adaptive traffic signals, smart parking and automatic incident detection are common connected-roadway applications.
- Industrial IoT (IIoT) connects machines, robots and production systems to monitor and improve factory operations.
- Predictive maintenance uses sensor data to identify possible machine failures before they occur.
- Asset tracking and automated quality control help improve efficiency and product quality in connected factories.

- Connected buildings integrate HVAC, lighting, security and fire safety through a Building Management System (BMS).
- Occupancy-based HVAC control, smart lighting and integrated security are common connected-building applications.
- Across these domains, IoT helps improve efficiency, safety, resource usage and decision-making using real-time data.
- Common challenges include security risks, system integration and interoperability between devices from different vendors.

Objective Type Questions

1. What does the abbreviation V2I stand for?
2. Which term describes IoT applied inside a manufacturing plant?
3. What is the name given to maintenance that is scheduled based on sensor trends rather than a fixed calendar?
4. Which device collects location data from a vehicle in a connected roadway system?
5. What is the central system that unifies HVAC, lighting and security in a smart building called?
6. Which connected-roadway use case guides drivers directly to an empty parking bay?
7. What kind of sensors help detect an empty conference room for HVAC control?
8. Which identification technology is commonly used for asset tracking on a factory floor?
9. Name one primary goal shared by connected roadways, factories and buildings.
10. What challenge arises when equipment from different vendors follows different communication standards?

SGOU

Answers

1. Vehicle-to-Infrastructure.
2. Industrial IoT (IIoT).
3. Predictive maintenance.
4. GPS (receiver).
5. Building Management System (BMS).
6. Smart parking.
7. Occupancy sensors.
8. RFID.
9. Efficiency (or real-time, data-driven decision-making).
10. Interoperability problems.

Assignments

1. Describe, with a diagram, the components of a connected roadway system and explain how V2I communication supports adaptive signal control.
2. Explain how predictive maintenance in a connected factory differs from traditional scheduled maintenance, using a suitable example.
3. Discuss the role of a Building Management System in coordinating HVAC, lighting and security within a smart building.
4. Compare connected roadways, connected factories and connected buildings in terms of their primary goals and typical sensors.
5. Analyse the security challenges that arise when a large number of IoT devices are deployed across a city-scale roadway network.
6. Suggest two additional use cases, beyond those discussed in this unit, for IoT in either a factory or a building setting.
7. Explain how asset tracking using RFID improves efficiency on a factory floor.
8. Evaluate the benefits and limitations of occupancy-based HVAC control in an office building.

SGOU

Suggested Reading

1. Hanes, David, et al. *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Cisco Press, 2017.
2. Bahga, Arshdeep, and Vijay Madisetti. *Internet of Things: A Hands-On Approach*. Universities Press, 2015
3. Buyya, Rajkumar, and Amir Vahid Dastjerdi, editors. *Internet of Things: Principles and Paradigms*. Morgan Kaufmann, 2016.
4. Sinha, Rajeev, and Yashwant Singh. *Industrial Internet of Things and Cyber-Physical Systems*. IGI Global, 2019.

Reference

1. Gubbi, Jayavardhana, et al. "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions." *Future Generation Computer Systems*, vol. 29, no. 7, 2013, pp. 1645–1660.
2. Sisinni, Emiliano, et al. "Industrial Internet of Things: Challenges, Opportunities, and Directions." *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, 2018, pp. 4724–4734.
3. Minoli, Daniel, and Benedict Occhiogrosso. "Practical Aspects for the Integration of 5G Networks and IoT for Smart Cities." *Internet of Things*, vol. 14, 2021.

SGOU



Convergence of IT and OT

Learning Outcomes

After completion of this unit, learner will be able to:

- define Information Technology (IT) and Operational Technology (OT) in the context of industrial systems
- identify the key differences between IT systems and OT systems
- explain the drivers and benefits of IT–OT convergence in modern industrial environments
- describe the major challenges faced while deploying and scaling Internet of Things (IoT) systems
- recognize the security and interoperability issues that IT–OT convergence and large-scale IoT deployments introduce

Prerequisite

Picture a large water-treatment plant. On the factory floor, pressure gauges, pumps and chlorine dosing units are watched over by control-room engineers who trust dials, alarms and paper log-books built to run non-stop for decades. A few corridors away, in the administration block, the IT team manages email servers, billing software and the plant's website, refreshing it with the newest security patch every month. The two worlds rarely speak to each other – until the night a slow leak in an underground valve goes unnoticed for six hours, because the pressure-drop reading sitting quietly on a control-room screen never reaches the analytics dashboard that could have flagged it instantly. Somewhere between the machine that senses the problem and the software that could act on it, a wall exists. Why does this wall exist, what happens when organisations try to knock it down, and what new problems appear on the other side? This unit takes you into that space where the world of machines meets the world of computers.

Keywords

Information Technology, Operational Technology, SCADA, Industrial Internet of Things, Edge Computing, Interoperability, Legacy Systems, Cyber-Physical Systems

SGOU

Discussion

1.3.1 Two Worlds Inside Every Industry: IT and OT

Every modern industrial organisation, whether it runs a power grid, a chocolate factory or a metro railway, actually operates two parallel technology worlds side by side. The first is Information Technology (IT): the servers, databases, enterprise software and office networks that store records, run payroll, send email and generate business reports. The second is Operational Technology (OT): the programmable logic controllers (PLCs), supervisory control and data acquisition (SCADA) systems, sensors, actuators and robotic arms that directly sense and control physical machinery on the shop floor. For most of the twentieth century these two worlds developed independently, built by different engineers, governed by different priorities, and almost never connected to the same network.

1.3.1.1 Information Technology (IT)

Information Technology refers to the hardware, software and networks that an organisation uses to create, process, store and exchange digital information. Typical IT assets include desktop computers and laptops, database servers, enterprise resource planning (ERP) software, email systems and cloud applications. IT departments are chiefly concerned with the confidentiality, integrity and availability of data, and they operate in a comparatively predictable environment: machines are replaced every few years, operating systems are patched frequently, and a short outage for maintenance is usually tolerable.

1.3.1.2 Operational Technology (OT)

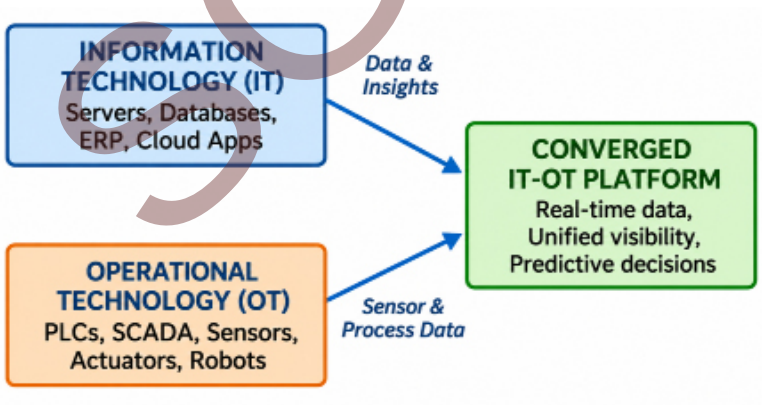


Fig. 1.3.1 Convergence of IT and OT in Industrial IoT

Operational Technology refers to the hardware and software that directly monitors and controls physical devices, processes and events in industries such as manufacturing, energy, water treatment and transportation. OT systems include PLCs that switch motors on and off, SCADA systems that give operators a live view of an entire plant, distributed control systems (DCS) that manage complex chemical processes, and remote terminal units (RTUs) that relay field data over long distances. Unlike IT equipment, OT

SGOU

equipment is often installed once and expected to run continuously for fifteen to thirty years with minimal downtime, because stopping a power turbine or a blast furnace even briefly can be expensive or dangerous.

1.3.1.3 Comparing IT and OT

Although both IT and OT ultimately serve the same organisation, they differ sharply in their objectives, their tolerance for risk, and their approach to change, as summarised in Table 1.3.1

Table 1.3.1 Comparison of Information Technology and Operational Technology

Aspect	Information Technology (IT)	Operational Technology (OT)
Primary concern	Confidentiality and integrity of data	Safety and continuity of the physical process
Typical lifespan	3 to 5 years	15 to 30 years
Update frequency	Frequent patches and upgrades	Rare; changes are tightly controlled
Tolerance to downtime	Short outages usually acceptable	Very low; downtime can be costly or unsafe
Example devices	Servers, laptops, routers	PLCs, SCADA, RTUs, sensors, actuators
Common protocols	TCP/IP, HTTP, SQL	Modbus, PROFINET, DNP3, EtherNet/IP

1.3.2 Why Are IT and OT Converging?

The rise of the Industrial Internet of Things (IIoT) and the broader movement often called Industry 4.0 has made it commercially attractive, and sometimes essential, to connect OT devices to IT networks. When a factory's sensors can feed their readings directly into cloud analytics platforms, managers gain a single, real-time view of both the business and the shop floor. Maintenance can shift from a fixed calendar schedule to predictive maintenance, where a machine is serviced only when its own vibration or temperature data suggests a fault is developing. Inventory levels, energy consumption and production throughput can all be tracked and optimised together instead of in separate silos. This blending of the two historically separate technology stacks into one connected, data-driven platform is what we call IT–OT convergence.

In 2010, a sophisticated piece of malware named Stuxnet was discovered targeting industrial control systems used in uranium enrichment. It specifically manipulated PLCs while feeding false, normal-looking readings back to the SCADA monitoring screens, so operators saw nothing unusual while the equipment was being damaged. Stuxnet is widely cited as the moment the world realised that once OT equipment becomes reachable through IT-style networks, it inherits IT-style cyber risks too – a lesson that lies at the heart of every IT–OT convergence project today.

SGOU

1.3.2.1 Benefits of IT–OT Convergence

- **Real-time plant-wide visibility:** IT and OT data can be combined to provide a complete view of production and business operations through a common dashboard.
- **Predictive maintenance:** Data collected from machines can be analysed to identify possible equipment problems early, allowing maintenance to be performed before a failure occurs and helping to reduce unplanned downtime.
- **Faster data-driven decision-making:** Real-time information helps teams in design, production, and supply-chain management make faster and more informed decisions.
- **Better resource utilisation:** Energy, materials, and labour can be monitored together, helping organisations identify waste and use resources more efficiently.

1.3.2.2 Challenges in Achieving IT–OT Convergence

Bringing IT and OT together is rarely simple. OT engineers and IT administrators often come from different professional cultures: one prioritises uninterrupted physical safety, the other prioritises data security and rapid patching, and the two priorities can conflict. OT devices frequently use proprietary, decades-old protocols that were never designed to be exposed to a shared network, so gateways and protocol translators are needed before any integration can happen. Perhaps most importantly, connecting a previously isolated OT network to the wider IT network dramatically increases the attack surface available to a cyber-criminal, which is why convergence projects must always be paired with a strong security strategy rather than treated as a purely technical wiring exercise.

1.3.3 IoT Challenges

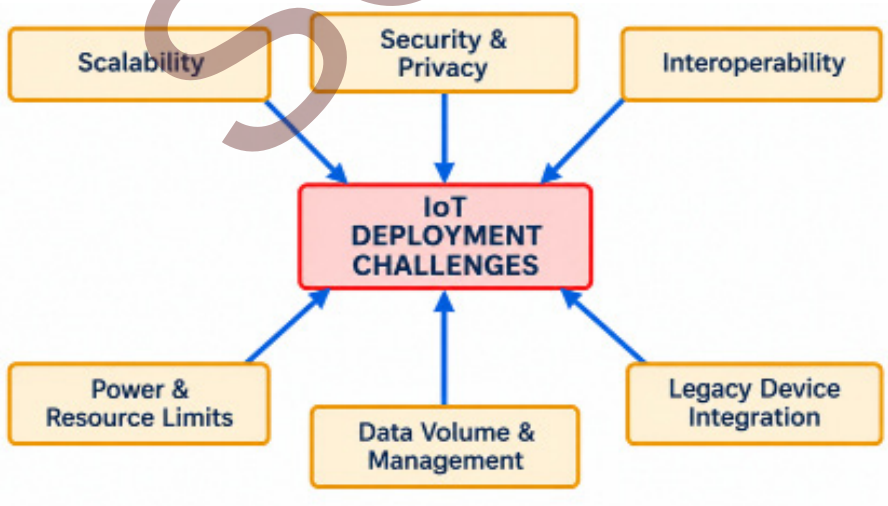


Fig. 1.3.2 Major Challenges in IoT Deployment

Beyond the specific problem of IT–OT convergence, the wider adoption of the Internet of Things introduces a set of challenges that any organisation must plan for before rolling out a large-scale deployment. These challenges arise because IoT systems

SGOU

combine an enormous number of devices, many of which are small, battery-powered and inexpensive, with the demanding requirement of moving and analysing their data reliably and securely.

1.3.3.1 Scalability

An IoT deployment may need to support anywhere from a handful of sensors in a smart home to billions of devices spread across an entire smart city. Addressing schemes, network capacity and backend databases that work comfortably for a pilot project of a hundred devices can collapse under the weight of a million. Designing an architecture that scales smoothly, without needing to be redesigned at every stage of growth, is one of the first challenges any IoT planner faces.

1.3.3.2 Security and Privacy

Many IoT devices are inexpensive and resource-constrained, which often means they ship with weak default passwords, cannot run full encryption, and rarely receive security updates once deployed. This makes them attractive targets for attackers looking for an easy entry point into a larger network. At the same time, the sheer volume of personal and operational data that IoT devices collect – from home occupancy patterns to factory production rates – raises serious privacy questions about who can access that data and how long it is retained.

> In October 2016, a piece of malware called Mirai turned tens of thousands of ordinary IoT devices – mostly home routers and internet-connected security cameras – into a botnet simply by trying a short list of default factory passwords. The botnet was then used to launch one of the largest distributed denial-of-service (DDoS) attacks recorded at that time, temporarily knocking major websites offline. The incident remains a textbook example of why weak default security on inexpensive IoT devices is a challenge for the entire internet, not just for the device owner.

1.3.3.3 Interoperability and Standardisation

IoT devices are manufactured by thousands of different vendors, each often favouring its own communication protocol, data format or cloud platform. Wi-Fi, Bluetooth Low Energy, Zigbee, LoRaWAN, MQTT and CoAP are only a few of the many technologies in common use. Without agreed standards, devices from different manufacturers may be unable to talk to one another, forcing organisations to either lock themselves into a single vendor's ecosystem or invest heavily in gateways that translate between protocols.

1.3.3.4 Power and Resource Constraints

A great many IoT endpoints, such as soil-moisture sensors scattered across a farm or fitness trackers worn on the wrist, must run for months or years on a small battery or even on harvested energy. This severely limits the processing power, memory and radio strength available to the device, which in turn limits the complexity of the security measures and communication protocols it can support. Designing lightweight protocols and duty-cycled communication schedules that conserve energy without sacrificing reliability is a continuing engineering challenge.

SGOU

1.3.3.5 Data Volume and Management

A modern IoT deployment can generate a continuous, high-velocity stream of data from thousands of sensors simultaneously. Transporting all of this raw data to a central cloud for processing can overwhelm network bandwidth and delay time-critical decisions. Organisations must therefore decide what data needs to be processed instantly close to the source, what can be summarised before transmission, and what should be archived centrally for long-term analysis.

1.3.3.6 Legacy Device Integration

Many industrial sites still rely on equipment installed decades ago that was never designed with network connectivity in mind. Replacing this equipment purely to add IoT capability is often too costly to justify, so organisations instead retrofit legacy machines with external sensors and protocol-translating gateways that can read old signals and forward them, in a modern format, onto the IoT network.

Table 1.3.2 summarises these challenges together with a typical mitigation approach used by practitioners.

Table 1.3.2 Summary of Major IoT Challenges and Mitigation Approaches

Challenge	Typical Mitigation Approach
Scalability	Hierarchical, addressable network design; IPv6 addressing
Security and privacy	Device authentication, encryption, regular firmware updates
Interoperability	Adoption of open standards and protocol-translating gateways
Power and resource limits	Lightweight protocols; duty cycling; energy harvesting
Data volume	Edge and fog processing to filter data before it reaches the cloud
Legacy integration	Retrofit sensors and protocol gateways for older equipment

SGOU

Recap

- Information Technology (IT) handles data processing, storage and communication, while Operational Technology (OT) senses and controls physical processes.
- IT and OT differ in lifespan, tolerance to downtime, update frequency and communication protocols.
- IT–OT convergence is driven by the rise of the Industrial Internet of Things and the demand for real-time, data-driven decision-making.

- Benefits of convergence include predictive maintenance, plant-wide visibility and better resource utilisation.
- Convergence also introduces challenges such as cultural differences between IT and OT teams and an expanded cyber-attack surface.
- Major IoT deployment challenges include scalability, security and privacy, interoperability, power constraints, data volume, and legacy device integration.
- Weak default security on inexpensive IoT devices, as seen in the Mirai botnet incident, can be exploited on a massive scale.
- Edge and fog processing help manage the huge volume of data generated by large IoT deployments.

Objective Type Questions

1. What does the abbreviation OT stand for?
2. Which technology primarily deals with the confidentiality and integrity of digital data?
3. Which type of device is a PLC an example of?
4. What term describes the blending of IT and OT into a single connected platform?
5. Which 2010 malware specifically targeted industrial control systems?
6. Which maintenance strategy uses sensor data to service equipment before it fails?
7. Name the 2016 malware that turned insecure IoT devices into a botnet.
8. What is the typical operational lifespan of OT equipment?
9. Which challenge refers to an IoT system's ability to grow from a few devices to millions of devices?
10. Which term describes the ability of devices from different vendors to work together?
11. What kind of computing is used to process IoT data close to the source?
12. What must be added to old industrial machines to give them network connectivity?
13. Which two properties are severely limited in battery-powered IoT sensors?
14. Which communication protocol is commonly used for industrial automation, unlike TCP/IP used in IT?

SGOU

Answers

1. Operational Technology
2. Information Technology (IT)
3. An Operational Technology (OT) device / controller
4. IT–OT convergence
5. Stuxnet
6. Predictive maintenance
7. Mirai
8. 15 to 30 years
9. Scalability
10. Interoperability
11. Edge (or fog) computing
12. Sensors and protocol-translating gateways
13. Processing power and memory (battery/energy)
14. Modbus (or PROFINET/DNP3)

Assignments

1. Explain the difference between Information Technology and Operational Technology with suitable examples.
2. Discuss the drivers and benefits of IT–OT convergence in an industrial setting.
3. What challenges does an organisation face while trying to converge its IT and OT systems? Explain.
4. Describe, with an example, how the Stuxnet incident illustrates the security risks of IT–OT convergence.
5. List and explain any five major challenges faced during large-scale IoT deployment.
6. How does the Mirai botnet incident demonstrate the security weaknesses of consumer IoT devices?

SGOU

7. Explain how edge and fog computing help address the data volume challenge in IoT systems.
8. Discuss the problem of legacy device integration in IoT and suggest possible solutions.

Suggested Reading

1. Hanes, David, et al. *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Cisco Press, 2017.
2. Greengard, Samuel. *The Inte*
3. Buyya, Rajkumar, and Amir Vahid Dastjerdi, editors. *Internet of Things: Principles and Paradigms*. Morgan Kaufmann, 2016.

Reference

1. Sabella, Rik, et al. *Orchestrating and Automating Security for the Internet of Things*. Cisco Press, 2018.
2. Yokogawa Electric Corporation. "IT/OT Convergence – Integrating Business and Manufacturing." *Yokogawa Digital Infrastructure Wiki*, 2023.
3. Amos, Zac. "IoT Is the Essential Mediator for Secure IT/OT Convergence." *IoT For All*, 2 Dec. 2024.

SGOU



IoT Architecture

Learning Outcomes

After completion of this unit, learner will be able to:

- identify the key drivers that make traditional network architectures unsuitable for large-scale IoT deployments
- recall the standardised IoT architectural models proposed by industry consortia
- explain the seven layers of the IoT World Forum (IoTWF) reference model
- describe the components of a simplified IoT architecture
- recognise the roles of edge, fog and cloud computing in managing IoT data

Prerequisite

Imagine you have been asked to design the network for a new smart city: traffic signals, streetlights, parking sensors, water meters and pollution monitors, numbering in the hundreds of thousands, all reporting data continuously. Your only prior experience is designing the network for a company office of two hundred employees, where every device is a laptop or a desktop plugged into a stable wired or Wi-Fi connection, and where a two-minute network outage at midnight would bother nobody. Would the office network blueprint survive being stretched across an entire city, where devices sit in the rain, run on tiny batteries for years, and where a lost reading from a single flood sensor could matter at 3 a.m.? Before we can compare or choose an IoT architecture, we first need to understand exactly why the old blueprint breaks down at this new scale – and that is where our journey in this unit begins.

Keywords

Scalability, oneM2M, IoT World Forum, Reference Model, Fog Computing, Edge Computing, Core Functional Stack, Data Abstraction

SGOU

Discussion

1.4.1 Why IoT Needs a New Network Architecture

Designing a network is much like designing a building: without a blueprint that anticipates the number of occupants, the load on the structure and the possibility of future expansion, the building will eventually fail to serve its purpose. Traditional enterprise IT networks were designed around a comparatively small and predictable number of well-behaved clients, such as employee laptops and servers, connected over reliable wired or Wi-Fi links within a single building or campus. IoT networks break almost every one of these assumptions: they may contain a vastly larger number of devices, many of which are inexpensive, constrained in power and processing ability, and connected over unreliable wireless links spread across wide geographic areas. This mismatch is what drives the search for new, purpose-built IoT network architectures.

1.4.1.1 Scale

Analysts estimate that tens of billions of connected devices will be in use worldwide, and a single deployment, such as a smart city or a large agricultural farm, may itself involve hundreds of thousands of sensors. Addressing, routing and managing a network of this size requires an architecture built for massive scale from the outset, rather than one that was designed for a few thousand clients and later stretched to fit.

1.4.1.2 Security

Every additional connected device is a potential entry point for an attacker. Because many IoT devices are physically accessible in public or remote locations and cannot support heavyweight security software, an IoT architecture must build in security at every layer, from the device itself through the network to the application, rather than relying on a single perimeter firewall as many traditional IT networks do.

1.4.1.3 Constrained Devices and Networks

Many IoT endpoints have limited processing power, memory and battery life, and they often communicate over low-bandwidth, high-latency or intermittently available wireless links. An architecture designed for high-powered, always-connected computers cannot simply be reused; the protocols and data formats must be made lightweight enough to run on constrained hardware and tolerant of unreliable connectivity.

1.4.1.4 Data

IoT devices can generate a continuous, high-volume stream of data, and much of the value of IoT lies in analysing this data quickly. Traditional architectures, built around occasional file transfers and database queries, are not designed to ingest, filter and analyse continuous sensor streams at scale, so newer architectures must add explicit layers for data accumulation, abstraction and analytics.

1.4.1.5 Legacy Device Support

Because industrial and infrastructure equipment is often used for decades, a new IoT

SGOU

architecture cannot assume that every device will be modern and IP-capable. It must accommodate older equipment through gateways and protocol translators, allowing legacy devices to participate in the same network as the newest IoT sensors. Table 1.4.1 summarises these drivers and their architectural implications.

Table 1.4.1 Drivers Behind New IoT Network Architectures

Driver	Implication for Architecture
Scale	Hierarchical, highly addressable design supporting millions of devices
Security	Security built into every layer, not just the network perimeter
Constrained devices and networks	Lightweight protocols tolerant of low power and unreliable links
Data	Dedicated layers for data accumulation, abstraction and analytics
Legacy device support	Gateways and protocol translation for older equipment

1.4.2 Comparing IoT Architectural Models

As IoT adoption grew, several industry groups proposed standardised reference architectures to give designers a common vocabulary and a consistent starting point. Two influential examples are the oneM2M standardised architecture and the IoT World Forum (IoTWF) reference model.

1.4.2.1 The oneM2M Standardised Architecture

oneM2M is a global standards body formed by telecommunications organisations to create a common machine-to-machine (M2M) and IoT service layer. Its architecture is organised around three broad domains: an application layer where IoT applications reside, a services layer that offers common functions such as device management, security and data storage to any application, and a network layer that provides underlying connectivity. By separating a reusable services layer from the applications built on top of it, oneM2M allows different IoT applications to share the same underlying infrastructure instead of each rebuilding common functions from scratch.

1.4.2.2 The IoT World Forum (IoTWF) Reference Model

In 2014, an architectural committee at the IoT World Forum, backed by companies including Cisco, IBM and Rockwell Automation, published a seven-layer reference model intended to become a widely accepted, vendor-neutral description of IoT systems. Unlike the three broad domains of oneM2M, the IoTWF model breaks the journey of data, from a physical sensor to a business decision, into seven distinct layers, described in Table 1.4.2

SGOU

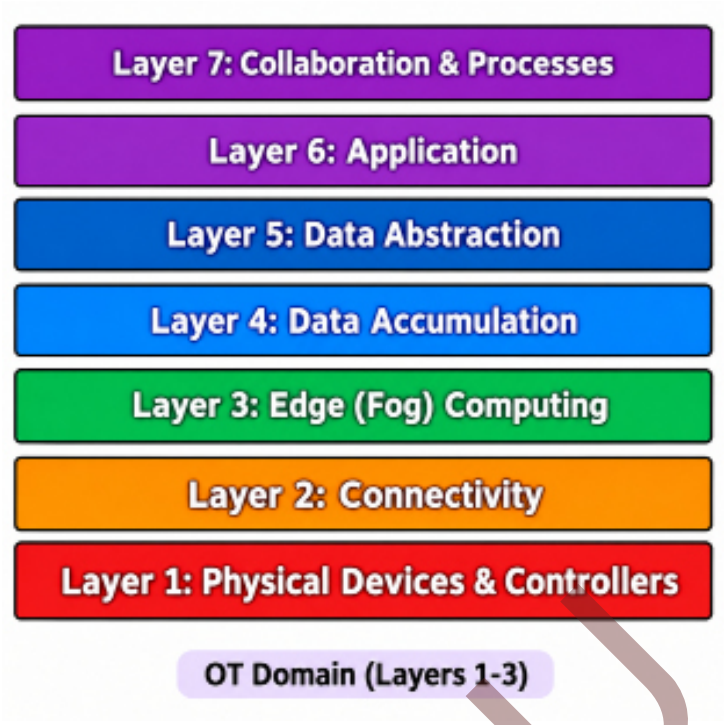


Fig. 1.4.1 IoT World Forum (IoTWF) Seven-Layer Reference Model

Table 1.4.2 Function of Layers in Seven-Layer Reference Model

Layer	Name	Function
1	Physical Devices and Controllers	The actual sensors, actuators and controllers – the "things" that sense and act on the physical world
2	Connectivity	Reliable, timely transmission of data between devices and the network
3	Edge (Fog) Computing	Initial data processing, filtering and aggregation performed close to the devices
4	Data Accumulation	Capturing and storing data so that it can be used by higher-layer applications
5	Data Abstraction	Reconciling, formatting and combining data from multiple sources for easier access
6	Application	Software that interprets data to generate reports, alerts or automated control actions
7	Collaboration and Processes	People and business processes that act on the insights delivered by the application layer

SGOU

1.4.2.3 IT and OT Responsibilities in the Reference Model

The IoTWF model conveniently maps onto the IT–OT distinction introduced earlier in this course: the lower three layers (Physical Devices, Connectivity and Edge Computing) generally fall under OT responsibility, since they deal directly with sensing and controlling physical equipment, while the upper four layers (Data Accumulation, Data Abstraction, Application, and Collaboration and Processes) generally fall under IT responsibility, since they deal with storing, interpreting and acting on information. This is precisely the boundary where IT–OT convergence, discussed in Unit 3, takes place in practice

Table 1.4.3 draws together a brief side-by-side comparison of the two standardised models discussed above.

Did You Know?

The IoT World Forum reference model was not designed by a single company acting alone. It was the outcome of an architectural committee bringing together Cisco, IBM, Rockwell Automation, Intel and several other technology leaders at the 2014 IoT World Forum, with the explicit goal of giving the whole industry a shared vocabulary for describing IoT systems, rather than leaving every vendor to invent its own incompatible terminology.

Table 1.4.3 Comparison of the oneM2M and IoTWF Architectural Models

Feature	oneM2M	IoT World Forum (IoTWF)
Number of layers/ domains	Three (Application, Services, Network)	Seven
Primary focus	Reusable service layer for M2M/IoT applications	Complete data journey from device to business process
Backed by	Global telecom standards bodies	Cisco, IBM, Rockwell Automation and partners
Best suited for	Telecom-oriented M2M service platforms	Enterprise and industrial IoT solution design

1.4.3 A Simplified IoT Architecture

While the seven-layer IoTWF model is thorough, many practitioners find it convenient to work with a simplified, three-layer view of an IoT system for day-to-day

SGOU

design and discussion. This simplified architecture groups the seven detailed layers into two complementary stacks: the Core IoT Functional Stack, which describes how data moves from a physical sensor to a useful application, and the IoT Data Management and Compute Stack, which describes where along that journey the data is actually processed.

1.4.3.1 The Core IoT Functional Stack

The Core IoT Functional Stack condenses the IoTWF layers into three practical layers, shown in Fig 1.4.2.

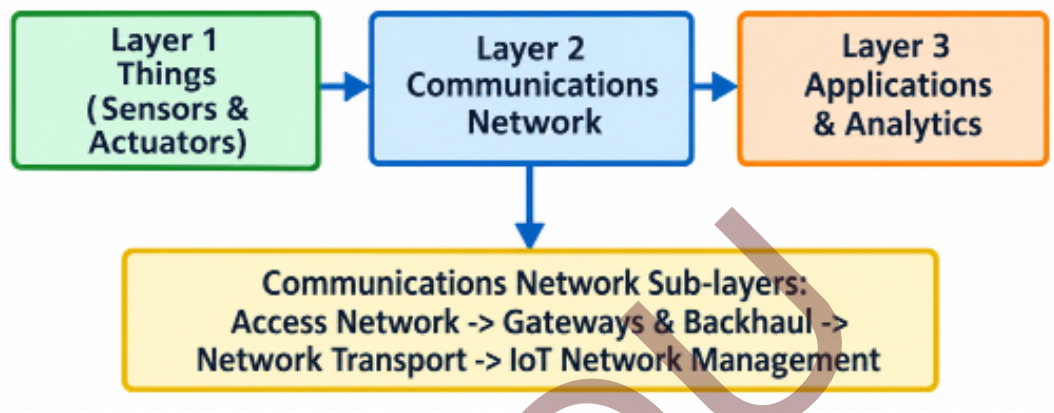


Fig. 1.4.2 Simplified IoT Architecture: Core IoT Functional Stack

- **Layer 1 – Things: Sensors and Actuators Layer:** the physical devices that sense conditions (temperature, motion, pressure) or act on the environment (valves, motors, switches).
- **Layer 2 – Communications Network Layer:** carries data from the things to the applications, and is itself commonly divided into an access-network sub-layer (the immediate wireless or wired link to the device), a gateways and backhaul sub-layer (which aggregates and forwards traffic toward the core network), a network-transport sub-layer (which carries traffic across the wider network), and an IoT network-management sub-layer (which monitors and manages the health of the network itself).
- **Layer 3 – Applications and Analytics Layer:** the software that turns raw data into meaningful information, ranging from simple monitoring dashboards to advanced predictive analytics and automated control applications.

1.4.3.2 The IoT Data Management and Compute Stack

Deciding where data should be processed is just as important as deciding how it should flow. The IoT Data Management and Compute Stack describes a hierarchy of computing locations, moving outward from the device toward a centralized data centre, as illustrated in Fig 1.4.3.

SGOU

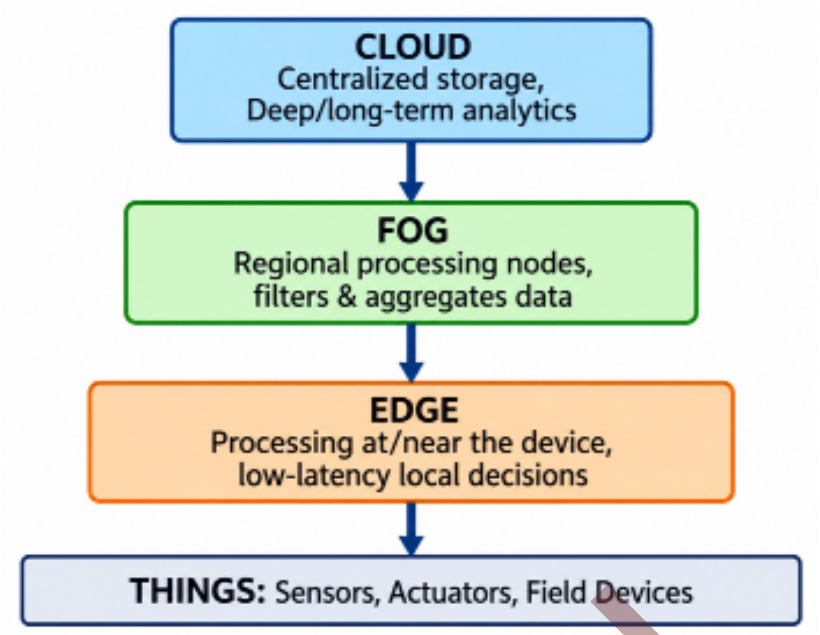


Fig. 1.4.3 Hierarchy of Edge, Fog, and Cloud Computing in IoT

Edge computing performs processing at or very close to the device itself, enabling extremely fast, local decisions, such as an industrial safety system that must shut off a machine within milliseconds of detecting a fault, without waiting for a distant server to respond. Fog computing extends this idea outward to regional nodes, often located within the same building or local network, that can aggregate, filter and analyse data from many nearby devices before forwarding only the most relevant results onward. Cloud computing sits at the top of the hierarchy, offering centralised, virtually unlimited storage and computing power for tasks that are not time-critical, such as long-term trend analysis, machine-learning model training, and organisation-wide reporting.

Note : The term "fog computing" was popularised by Cisco around 2014 to describe computing that happens close to the ground, as opposed to the cloud far above. The playful naming captures the core idea neatly: fog, like cloud computing, provides storage and processing resources, but it sits much closer to where the data is actually generated.

Together, the Core IoT Functional Stack and the IoT Data Management and Compute Stack give designers a simplified, easy-to-communicate architecture that still captures the essential ideas found in the more detailed IoTWF and oneM2M models: sense the physical world, move the data reliably, decide wisely where to process it, and turn it into meaningful action.

SGOU

Recap

- Traditional enterprise network architectures are not well suited to IoT because of differences in scale, security exposure, device constraints, data volume and legacy equipment.
- The five key drivers behind new IoT architectures are scale, security, constrained devices and networks, data, and legacy device support.
- oneM2M organises IoT systems into three domains: application, services and network layers.
- The IoT World Forum (IoTWF) reference model divides IoT systems into seven layers, from physical devices to collaboration and processes.
- In the IoTWF model, the lower three layers generally correspond to OT responsibilities and the upper four layers to IT responsibilities.
- A simplified IoT architecture groups these ideas into a Core IoT Functional Stack (Things, Communications Network, Applications and Analytics) and an IoT Data Management and Compute Stack.
- Edge computing processes data closest to the device for the fastest response; fog computing aggregates data regionally; cloud computing provides centralised, long-term storage and analytics.
- The term "fog computing" was popularised by Cisco to describe processing that happens close to the ground rather than in a distant cloud.

Objective Type Questions

1. What analogy is commonly used to describe the importance of planning a network architecture?
2. Which driver of new IoT architectures relates to billions of connected devices?
3. Which driver relates to devices with limited battery, memory and processing power?
4. How many layers does the oneM2M architecture broadly divide an IoT system into?
5. In which year was the IoT World Forum seven-layer reference model published?
6. Which layer of the IoTWF model represents the physical sensors and controllers?
7. Which layer of the IoTWF model is responsible for reliable and timely data transmission?

SGOU

8. Which layer of the IoTWF model performs initial local data processing near the devices?
9. Which layers of the IoTWF model generally fall under OT responsibility?
10. What are the three layers of the Core IoT Functional Stack?
11. Which type of computing processes data closest to the device for the fastest response?
12. Which type of computing aggregates data from nearby devices at a regional level?
13. Which type of computing provides centralised, long-term storage and analytics?
14. Which company popularised the term "fog computing"?

Answers

1. Designing a building using a blueprint
2. Scale
3. Constrained devices and networks
4. Three
5. 2014
6. Layer 1: Physical Devices and Controllers
7. Layer 2: Connectivity
8. Layer 3: Edge (Fog) Computing
9. Layers 1 to 3
10. Things; Communications Network; Applications and Analytics
11. Edge computing
12. Fog computing
13. Cloud computing
14. Cisco

SGOU

Assignments

1. Explain why traditional enterprise network architectures are unsuitable for large-scale IoT deployments.
2. Discuss the five key drivers behind the need for new IoT network architectures.
3. Describe the oneM2M standardised architecture and its three domains.
4. Explain the seven layers of the IoT World Forum (IoTWF) reference model with a neat diagram.
5. How do IT and OT responsibilities map onto the IoTWF reference model? Explain.
6. Describe the Core IoT Functional Stack of the simplified IoT architecture.
7. Explain the hierarchy of edge, fog and cloud computing in IoT data management.
8. Compare the oneM2M and IoTWF architectural models, highlighting their key differences.

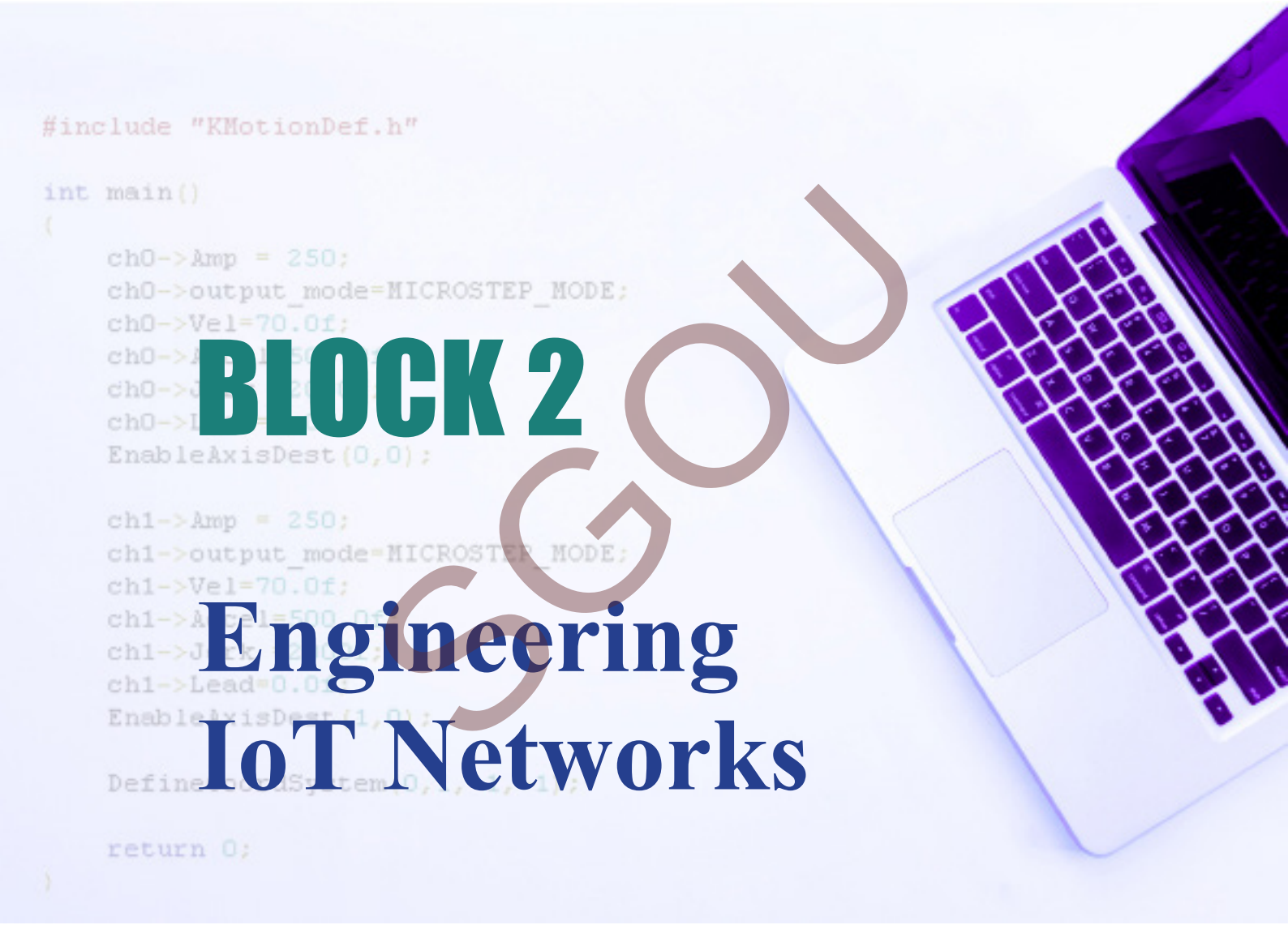
Suggested Reading

1. Hanes, David, et al. *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Cisco Press, 2017.
2. Bahga, Arshdeep, and Vijay Madisetti. *Internet of Things: A Hands-on Approach*. Universities Press, 2015.
3. Greengard, Samuel. *The Internet of Things*. MIT Press, 2015.

Reference

1. Alshohoumi, Fatma, et al. "Systematic Review of Existing IoT Architectures." *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 6, 2019.
2. Cisco Systems. "The Internet of Things Reference Model." *IoT World Forum*, 2014.
3. AltexSoft. "IoT Architecture Layers and Components Explained." *AltexSoft Blog*, 2023.

SGOU



BLOCK 2

Engineering IoT Networks

SGOU



Smart Objects

Learning Outcomes

After completion of this unit, learner will be able to:

- identify the main components of a smart object
- explain the functions of sensors and actuators in IoT systems
- describe how sensor nodes communicate observations through a sensor network
- explain common sensor-network topologies and basic energy considerations
- recognize the role of gateways in connecting local sensing environments to wider networks

Prerequisite

Think about an automatic water tank controller. A level sensor notices how full the tank is, a small controller interprets the reading, and a pump is switched on or off. Now imagine hundreds of such devices distributed across a farm, factory or hospital, each reporting local conditions. The interesting question is no longer simply "Can a device measure something?" but "How can many devices sense, communicate and act together without constant human supervision?" This unit builds that bridge from familiar automatic systems to networked IoT objects.

Keywords

Smart Object, Sensor, Actuator, Sensor Node, Gateway, Sensing, Topology, Wireless Sensor Network

SGOU

Discussion

2.1.1 From Ordinary Objects to Smart Objects

An ordinary object becomes interesting in IoT when it can participate in an information flow. A smart object typically has a physical identity, a sensing or actuation interface, some processing capability and a communication interface. The object does not have to be powerful; it has to perform its intended task reliably and exchange useful information with other parts of the system.

Consider a smart room. A temperature sensor observes the room, a controller interprets the observation, and an actuator may operate a fan. If the device also sends measurements to a gateway, the local event becomes part of a larger information system. The same pattern appears in agriculture, logistics, buildings and industrial monitoring.

The key idea is integration. Sensing answers "what is happening?", processing helps answer "what does it mean?", communication answers "who needs to know?", and actuation answers "what should happen next?". When these functions cooperate, the object becomes an active participant rather than a passive component.

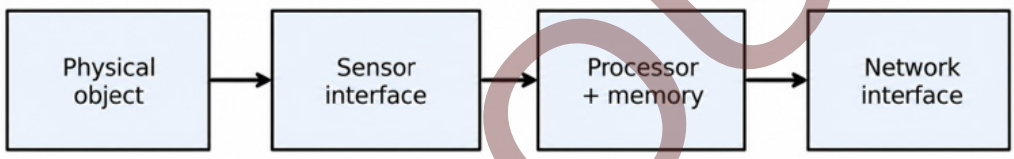


Fig. 2.1.1 From ordinary objects to smart objects (conceptual illustration)

2.1.2 Sensor: Sensing as the Starting Point

A sensor converts a physical phenomenon into a usable signal. Temperature, light, pressure, humidity, motion, vibration and proximity are common examples. The physical quantity being observed is often called the measurand. The sensor interface produces an electrical or digital representation that can be processed by the IoT device.

The quality of an IoT application depends heavily on the quality of its observations. Accuracy describes closeness to the true value, while precision describes consistency among repeated measurements. Resolution indicates the smallest meaningful change that can be distinguished. Response time indicates how quickly the sensor reflects a change. These properties matter differently in different applications.

For example, a room-monitoring application may tolerate modest response delay, whereas vibration monitoring of a rotating machine may require rapid sampling. A sensor should therefore be selected from the application requirement rather than from a specification sheet considered in isolation.

2.1.3 Sensor Interfaces and Signal Handling

A sensing element rarely communicates directly with an application. The signal may need conditioning, conversion, filtering and interpretation. In a simple device,

SGOU

the controller may read an analogue voltage using an analogue-to-digital converter. In another device, the sensor may already contain signal processing and provide a digital interface.

Signal handling is important because measurements contain uncertainty. Electrical noise, environmental interference, drift and installation conditions can influence readings. Simple techniques such as averaging repeated observations can reduce random variation, while calibration helps compensate for systematic error. The purpose is not to make a measurement magically perfect, but to make it sufficiently trustworthy for the intended decision.

Physical quantity → sensing element → signal → conversion/conditioning → digital value → application meaning.

This chain prevents the common mistake of treating a sensor value as if it were automatically the same thing as the physical reality.

2.1.4 Actuators: Turning Information into Action

An actuator performs a physical action in response to a control signal. Motors, relays, valves, heaters, pumps, buzzers and electronically controlled switches are common examples. In an IoT system, the actuator is the point where digital information produces a change in the physical environment.

Suppose a greenhouse system detects excessive heat. The sensor reports temperature; software compares the observation with the desired condition; and the actuator may open a vent or operate a fan. The actuator must be appropriate for the load, power supply and safety requirements. A correct software command cannot compensate for an incorrectly rated physical device.

Sensors and actuators are therefore complementary. A sensor provides evidence about the state of the environment, while an actuator changes that state. Many IoT applications form a feedback loop in which sensing, decision-making and action repeat continuously.

2.1.5 Smart-Object Architecture

A useful conceptual architecture contains five cooperating elements: the physical object, sensor/actuator interfaces, processing and memory, communication, and power. Not every device exposes all elements as separate hardware blocks, but the functions are present conceptually.

The processor may perform local filtering, threshold detection, scheduling or control. Memory stores configuration, temporary data and sometimes recent observations. The communication interface exchanges data with another node, gateway or service. Power management determines how long the object can operate, particularly when it is battery powered.

This architecture highlights a design principle: intelligence can be distributed. A device can perform some decisions locally and send only useful information upstream.

SGOU

Such local processing can reduce communication traffic and energy consumption while improving response time.

2.1.6 Sensor Nodes and Sensor Networks

A sensor node is a networked device that observes one or more physical variables and communicates its observations. When many nodes cooperate, they form a sensor network. Nodes may send readings directly to a gateway or forward data through neighboring nodes, depending on the network arrangement.

A sensor network is more than a collection of sensors. The nodes must coordinate addressing, timing, communication, energy use and data movement. The network may also need to tolerate nodes that temporarily fail or lose connectivity. These requirements make networking an essential part of sensing rather than an afterthought.

In a farm, for instance, soil-moisture nodes can be placed across fields. A gateway collects their readings and passes summarized information to an application. The network transforms many local measurements into a view of the larger environment.

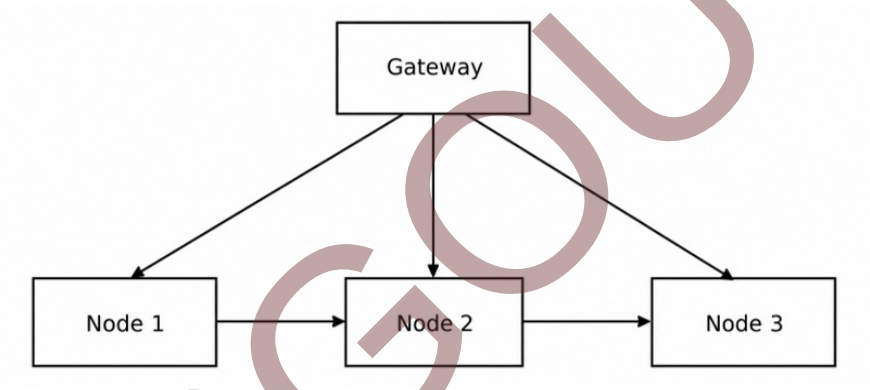


Fig. 2.1.2 Sensor nodes and sensor networks (conceptual illustration)

2.1.7 Network Topologies

Topology describes how nodes are arranged and how communication paths are formed. In a star arrangement, nodes communicate through a central coordinator or gateway. In a tree arrangement, communication follows hierarchical branches. In a mesh arrangement, nodes may have multiple neighboring paths, improving path flexibility but increasing coordination requirements.

No topology is universally best. A small indoor deployment may benefit from a simple star because management is easy. A wide industrial area may need multiple paths so that the loss of one link does not isolate a region. The correct choice depends on range, energy, expected traffic, maintenance and failure conditions.

Topology also influences troubleshooting. A central gateway can simplify diagnosis, whereas a distributed mesh can require more careful observation of link quality and routing behavior. Understanding topology therefore helps learners predict how a network will behave before implementation.

SGOU

2.1.8 Energy and Lifetime

Energy is a central constraint in many sensor networks. A battery-powered node may need to operate for months or years without physical access. Communication can consume a significant portion of the available energy, so the network should avoid unnecessary transmissions.

Duty cycling is a common strategy. A node can remain asleep when no measurement is required and wake for sensing, processing and communication. Local aggregation is another strategy: instead of transmitting every raw sample, a node may send a summary or an event when a meaningful change occurs.

Energy awareness must not compromise the application objective. If a safety system needs rapid alerts, excessive sleep intervals may be unacceptable. The engineering task is to balance measurement frequency, communication frequency, latency and battery life.

2.1.9 Gateways and Local Intelligence

A gateway connects a local IoT or sensor environment to another network. It may translate between communication technologies, aggregate data, enforce local rules, buffer information during temporary outages and provide a management point for devices.

A gateway is especially useful when many constrained nodes should not each maintain a direct connection to a cloud service. The gateway can collect local measurements and forward compact, meaningful information. It may also support local control so that an actuator continues to respond even when the wider network is unavailable.

This leads to a practical distinction between sensing and system intelligence. A sensor node may focus on measurement and low-power communication, while the gateway can carry out more demanding coordination. The division of responsibilities should follow the application requirements.

A smart object combines physical interaction with processing and communication. Sensors observe physical conditions, while actuators produce physical actions. Sensor nodes cooperate to form sensor networks, and gateways connect local deployments to wider systems.

Important concepts include measurement quality, signal handling, topology, energy management, duty cycling, local processing and data aggregation. Star, tree and mesh arrangements offer different trade-offs in simplicity, resilience and management.

In this unit, always connect the physical and digital sides of IoT. A useful mental model is sense → process → communicate → decide → act. The quality of the overall system depends on every link in that chain.

SGOU

Recap

- A smart object combines physical interaction with processing and communication.
- Sensors observe physical conditions, while actuators produce physical actions.
- Sensor nodes cooperate to form sensor networks, and gateways connect local deployments to wider systems.
- Important concepts include measurement quality, signal handling, topology, energy management, duty cycling, local processing and data aggregation.
- Star, tree and mesh arrangements offer different trade-offs in simplicity, resilience and management.
- In this unit, always connect the physical and digital sides of IoT. A useful mental model is sense → process → communicate → decide → act. The quality of the overall system depends on every link in that chain.

Objective Type Questions

1. Which component observes a physical quantity?
2. What component produces a physical action?
3. What is a networked sensing device called?
4. Which device commonly connects a local sensor network to a wider network?
5. What arrangement uses a central coordinating point?
6. Which topology can provide multiple paths?
7. What technique allows a battery device to sleep between tasks?
8. What process helps correct systematic measurement error?
9. What function interprets sensor data locally?
10. What is the physical quantity being measured called?

SGOU

Answers

1. Sensor
2. Actuator
3. Sensor node
4. Gateway
5. Star
6. Mesh
7. Duty cycling
8. Calibration
9. Processing
10. Measurand

Assignments

1. Explain the architecture of a smart object with a neat labelled diagram and discuss the role of each functional block.
2. Differentiate sensors and actuators with at least four IoT examples and explain how they form a feedback loop.
3. Describe a sensor network for agricultural monitoring, including nodes, gateway, topology and data flow.
4. Compare star, tree and mesh sensor-network topologies and discuss one suitable application for each.
5. Explain why energy management is important in battery-powered sensor networks. Discuss duty cycling and aggregation.
6. Discuss the role of a gateway in an IoT system and explain why local intelligence can be useful during connectivity failures.
7. Prepare a short case study of a smart-building sensing system and identify its sensors, actuators, communication path and control decisions.
8. Explain how measurement quality, calibration and signal handling affect the reliability of an IoT application.

SGOU

Suggested Reading

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Hersent, Olivier, David Boswarthick, and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Wiley, 2012.
3. Raj, Pethuru, et al. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017.

Reference

1. David Hanes et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Olivier Hersent, David Boswarthick, and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Wiley, 2012.
3. Pethuru Raj, Anupama C. Raman, and Harihara Subramanian. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017.

SGOU



Communication Criteria

Learning Outcomes

After completion of this unit, learner will be able to:

- identify the major criteria used to select an IoT communication technology
- explain the relationship among range, data rate, latency and reliability
- describe how energy, cost, scalability and deployment environment affect communication choices
- compare communication requirements for representative IoT applications

Prerequisite

Imagine two IoT devices. One is a battery-powered soil sensor buried far from a gateway and sends a small reading every fifteen minutes. The other is a camera that must move large amounts of data across a local network. Both are "IoT devices," but clearly they should not be given the same communication strategy. Before learning individual protocols, you need a way to judge whether a communication technology fits a particular job. This unit gives you that decision-making vocabulary.

SGOU

Keywords

Range, Data Rate, Latency, Reliability, Energy Efficiency, Scalability, Cost, Coverage

Discussion

2.2.1 Why Communication Criteria Matter

Communication is the bridge between an IoT device and the rest of the system. The choice of link affects whether information arrives on time, how much energy is consumed, how far devices can be placed, and how many devices can be supported. A technology should therefore be selected from requirements rather than popularity alone.

A useful first step is to describe the application in measurable terms. How far must a node communicate? How much data is generated? How quickly must the receiver know? How often does communication occur? What happens if a packet is lost? How difficult is maintenance? These questions convert a vague requirement into an engineering profile.

The criteria interact. Increasing data rate may increase energy demand. Extending range may require greater transmit power or a different frequency band. Strong reliability mechanisms may add overhead. Good IoT engineering is therefore a balancing exercise.

2.2.2 Bandwidth

Bandwidth is an important criterion used to evaluate the quality and efficiency of a system. It refers to the capacity of a communication channel to transmit data within a given period of time. Bandwidth is usually measured in bits per second (bps) and its higher units such as Kbps, Mbps, and Gbps.

The required bandwidth depends on the type and amount of data generated by IoT devices. For example, a temperature sensor may generate only a small amount of data at regular intervals and therefore requires low bandwidth. In contrast, an IoT-based surveillance camera continuously transmits video data and requires much higher bandwidth. Selecting appropriate bandwidth helps an IoT network transmit data efficiently without unnecessary resource usage. Therefore, while designing an IoT network, the bandwidth requirement of devices, data transmission frequency, number of connected devices, and application requirements should be considered.

2.2.3 Data Rate and Message Size

Data rate describes how quickly information can be transferred. IoT traffic can vary from a few bytes of periodic sensor data to continuous streams such as audio or video. A low-data-rate sensor does not need the same throughput as a multimedia device.

Message size and frequency matter together. A small message sent once per hour creates very different traffic from a small message sent every second. Similarly, a large message sent occasionally may require a short burst of higher throughput.

The important lesson is to match capacity to the application. Excess capacity can add cost or energy use, while insufficient capacity creates delay and congestion. Data-rate requirements should therefore be derived from the actual information flow.

2.2.4 Latency and Response Time

Latency is the time between sending information and its useful arrival or processing. Some IoT applications tolerate seconds or minutes, while control applications may require much faster response.

Consider two examples. A weather logger can store a measurement and upload it later; a safety controller may need to react immediately when a dangerous condition is detected. The second application places a much stronger requirement on latency and predictable delivery.

SGOU

Latency is not identical to data rate. A link can have high throughput but still introduce delays due to contention, processing or retransmission. IoT engineers should therefore examine both capacity and timing.

2.2.5 Reliability and Availability

Reliability describes how consistently communication succeeds, while availability concerns whether the service is accessible when required. Wireless links can be affected by interference, obstacles, congestion, fading and device failure.

Applications should define the consequences of communication failure. Losing one routine temperature reading may be harmless; losing a safety alarm may be unacceptable. Reliability mechanisms can include acknowledgements, retransmissions, redundancy, multiple paths and local fallback behavior.

A robust system does not assume that every message will always arrive. It defines what happens when communication is delayed or absent. Local buffering and local control can make an IoT system more resilient to temporary network problems.

2.2.6 Power Consumption

Power consumption is an important criterion when engineering IoT networks because many IoT devices operate using batteries or limited energy sources. It refers to the amount of electrical energy consumed by an IoT device during activities such as sensing, data processing, transmission, and reception. Since IoT devices may be deployed in remote or difficult-to-access locations, frequent battery replacement or recharging may not be practical.

For example, consider a smart agriculture system in which soil-moisture sensors are installed across a large field. If each sensor consumes a large amount of power while continuously transmitting data, its battery will be depleted quickly. A low-power communication method can allow the sensor to remain operational for a longer period by transmitting data only when necessary or by entering a sleep mode when it is not active. Therefore, IoT network design should aim to achieve reliable communication with minimum power consumption, thereby increasing battery life, reducing maintenance requirements, and improving the overall efficiency of the network.

2.2.7 Scalability and device density

Scalability describes how well a communication arrangement continues to function as the number of devices increases. A design that works for ten nodes may become congested when thousands of devices share the same environment.

High device density increases contention for shared resources. Addressing, channel access, scheduling, gateway capacity and management overhead become increasingly important. A scalable design also considers provisioning, firmware updates and fault diagnosis.

SGOU

Scalability should be considered from the beginning. If expansion is expected, the communication plan should leave room for additional devices and traffic instead of assuming that the initial deployment will remain small.

2.2.8 Cost, Complexity and Maintenance

Cost includes more than the purchase price of a radio module. Antennas, gateways, subscriptions, power systems, installation, testing, maintenance and replacement all contribute to the total cost of ownership.

Complexity also has a cost. A sophisticated network may provide advanced capabilities but require more configuration and specialized maintenance. For a simple application, a simpler technology may be the better engineering choice.

A useful decision compares technical performance with lifecycle practicality. The cheapest component is not necessarily the cheapest system if it requires frequent battery replacement or difficult field maintenance.

Table 2.2.1 Communication criteria for IoT selection

Criterion	What it asks	Typical implication
Range	How far?	Coverage and placement
Data rate	How much data?	Capacity requirement
Latency	How quickly?	Response behavior
Reliability	How consistently?	Failure tolerance
Energy	How much power?	Battery life
Scalability	How many devices?	Congestion and management
Cost	What is the total expense?	Lifecycle feasibility

2.2.9 Security and operating environment

Communication choices are also influenced by the need to protect information and control access. IoT devices can operate in homes, hospitals, factories, public spaces and remote locations, each with different threat and safety considerations.

The physical environment matters as well. Metal structures, moving machinery, underground locations and dense buildings can affect radio performance. Temperature, moisture and electromagnetic interference may also influence device operation.

Security and environmental constraints should be treated as requirements alongside range and data rate. A communication technology is suitable only when it can operate safely and appropriately in the actual deployment context.

SGOU

2.2.10 Comparing application profiles

A periodic environmental sensor usually emphasizes low energy use, modest data rate and long operational life. A smart-home controller may emphasize low latency, convenient installation and local reliability. An industrial monitoring system may emphasize robustness, predictable response and resistance to interference.

These examples show why no single criterion should dominate every decision. The "best" technology changes with the application profile.

A simple selection method is to list the requirements, rank them by importance, eliminate technologies that fail essential constraints, and then compare the remaining choices on cost and implementation effort.

2.2.11 Decision Natrix and Worked Example

Suppose a remote water-level sensor sends one short message every ten minutes. Required range is several hundred metres, latency can be a few seconds, and battery life should be long. A high-throughput multimedia link would provide unnecessary capacity, while a low-power technology with adequate coverage may be more appropriate.

A decision matrix can score candidate technologies against range, data rate, energy, latency, reliability, cost and scalability. The scores should be based on the actual application requirements, not on arbitrary preferences.

The worked example demonstrates a general rule: start with constraints, then compare technologies. This prevents the common error of selecting a protocol first and trying to force the application to fit it.

Communication criteria provide a systematic way to compare IoT technologies. The most important criteria include range, data rate, latency, reliability, energy efficiency, scalability, cost, security and environmental suitability.

Criteria interact, so trade-offs are unavoidable. A long-range link may require more power; higher throughput may increase energy demand; stronger reliability may add overhead.

A sound selection process begins with the application profile and eliminates technologies that violate essential requirements before comparing secondary advantages.

Recap

- Communication criteria provide a systematic way to compare IoT technologies. The most important criteria include range, data rate, latency, reliability, energy efficiency, scalability, cost, security and environmental suitability.
- Criteria interact, so trade-offs are unavoidable. A long-range link may require more power; higher throughput may increase energy demand; stronger reliability may add overhead.

SGOU

- A sound selection process begins with the application profile and eliminates technologies that violate essential requirements before comparing secondary advantages.

Objective Type Questions

1. Which criterion describes physical communication distance?
2. Which criterion describes the amount of data transferred per unit time?
3. What term describes communication delay?
4. Which criterion concerns successful and consistent delivery?
5. Which criterion is critical for battery-powered nodes?
6. What property describes growth from few to many devices?
7. What broader expense includes installation and maintenance?
8. Which factor describes the physical surroundings affecting communication?
9. What happens when shared communication resources become crowded?
10. 10. Which mechanism can improve resilience by using alternative paths?

Answers

1. Range
2. Data rate
3. Latency
4. Reliability
5. Energy efficiency
6. Scalability
7. Lifecycle cost
8. Environment
9. Congestion
10. Redundancy

SGOU

Assignments

1. Explain the major communication criteria used in IoT technology selection and illustrate their interdependence.
2. Compare range, data rate and latency using two contrasting IoT applications.
3. Discuss reliability and availability and explain what an IoT system should do when communication fails.
4. Explain the relationship between communication frequency and battery life in a sensor node.
5. Develop a conceptual decision matrix for selecting a communication technology for a remote water-level sensor.
6. Discuss scalability and the problems that can arise when a network grows from tens to thousands of devices.
7. Explain why lifecycle cost and maintenance should be considered along with the purchase price of communication hardware.
8. Write a reasoned note explaining why application requirements should be defined before choosing a communication technology.

Suggested Reading

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Hersent, Olivier, David Boswarthick, and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Wiley, 2012.
3. Raj, Pethuru, et al. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017.

Reference

1. David Hanes et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Olivier Hersent, David Boswarthick, and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Wiley, 2012.
3. Pethuru Raj, Anupama C. Raman, and Harihara Subramanian. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017.

SGOU



Communication Protocols

Learning Outcomes

After completion of this unit, learner will be able to:

- explain the basic communication ideas behind Wi-Fi, RFID, Bluetooth, RF and satellite links
- identify suitable IoT use cases for each communication technology
- compare the technologies using range, throughput, energy and deployment considerations
- recognize the difference between a general radio concept and a specific protocol or technology family

Prerequisite

You already use several communication technologies without thinking about the engineering behind them. A phone may connect to a wireless network, a smartwatch may communicate with a nearby device, a shop may identify an item without a line-of-sight scan, and a navigation receiver may obtain information from satellites. These experiences are useful starting points. The challenge is to understand why each technology is used for a different purpose and why one cannot simply replace another in every IoT application.

Keywords

Wi-Fi, RFID, Bluetooth, Radio Frequency, Satellite Communication, NFC, Access Point, Tag

SGOU

Discussion

2.3.1 A Family of Communication Choices

IoT communication technologies differ in physical coverage, data capacity, energy requirements, device complexity and infrastructure. Wi-Fi is commonly associated with local wireless networking; Bluetooth is designed for short-range personal-area connectivity; RFID identifies or tracks tagged objects; RF is a broad description of radio-frequency communication; satellite systems support very wide-area connectivity.

The purpose of this unit is not to memorize isolated specifications. Instead, you should learn to recognize the operating idea and the type of problem each technology addresses.

A protocol or technology should be viewed as part of a complete system that includes antennas, devices, access points or readers, gateways and applications.

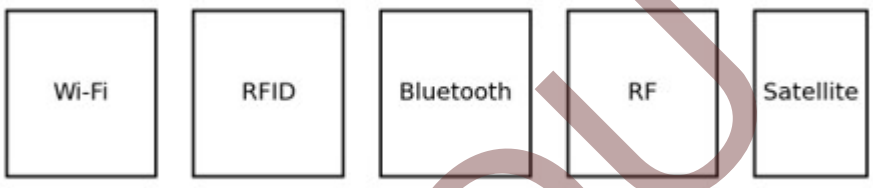


Fig. 2.3.1 A family of communication choices (conceptual illustration)

2.3.2 Wi-Fi in IoT

Wi-Fi is a family of wireless networking technologies used to connect devices to local networks. In an IoT deployment, a sensor or controller may communicate through an access point to reach local services or a wider network.

Wi-Fi is attractive when relatively high data rates, existing infrastructure and convenient IP connectivity are important. It is useful for smart appliances, cameras, gateways and devices that have access to a suitable power source.

Its limitations must also be considered. Continuous connectivity can consume more energy than low-power alternatives, and coverage can be affected by walls, interference and access-point placement.

2.3.2.1 Wi-Fi Architecture and Operation

A simple Wi-Fi IoT arrangement contains stations and an access point. The access point provides wireless access to a local network, while devices contend for the shared radio medium. Security, authentication and network configuration determine how devices are admitted and protected.

For self-learning, distinguish the device from the network service. A sensor may be physically small, but its communication behavior depends on the access point, channel conditions and network configuration.

SGOU

Wi-Fi is therefore most compelling when the deployment can support its energy and infrastructure requirements. It is less attractive when nodes are extremely power constrained and transmit only tiny messages infrequently.

2.3.3 RFID: Identification Through Radio

RFID uses radio communication between a reader and a tag. The tag contains identification information and may be passive or active depending on the system. Passive tags can obtain operating energy from the reader field, while active tags use their own power source.

RFID is especially useful when the main goal is identification, tracking or access control rather than continuous high-volume communication. Examples include inventory management, library systems, asset tracking and contactless identification.

The important conceptual distinction is that RFID is often an identification mechanism within a larger IoT system. The reader can forward tag observations to software that maintains an inventory or triggers an action.

2.3.4 Bluetooth and Short-range Connectivity

Bluetooth is designed for short-range wireless communication and is widely used for personal and device-to-device connectivity. In IoT, it can connect sensors, wearables, controllers and nearby mobile devices.

Bluetooth Low Energy is particularly relevant to battery-powered applications because it supports low-duty-cycle communication patterns. A device can advertise its presence, establish a connection when required, and exchange relatively small amounts of information efficiently.

Bluetooth is attractive when the physical distance is modest and low-power operation is important. The actual suitability depends on the application, device behavior, interference environment and required throughput.

2.3.5 RF as a Broad Engineering Concept

Radio frequency refers broadly to the portion of the electromagnetic spectrum used for radio communication. Many IoT technologies operate using RF energy, but “RF” by itself does not identify one single protocol.

An RF link involves a transmitter, antenna, propagation path and receiver. Frequency, modulation, transmit power, bandwidth, antenna design and environmental conditions influence performance.

Understanding RF at this level helps learners interpret why different technologies behave differently. Two devices may both use radio waves but employ different frequencies, protocols, channel-access methods and power budgets.

SGOU

2.3.6 Satellite Communication

Satellite communication uses spacecraft as part of the communication path, enabling connectivity over very large geographic areas. It is valuable where terrestrial infrastructure is limited or absent, such as remote monitoring, maritime applications and isolated installations.

The strengths are coverage and reach. The trade-offs can include higher equipment and service costs, greater propagation delay in some systems, antenna requirements and power considerations.

For IoT, satellite links are often best viewed as a connectivity option for remote assets rather than as a replacement for short-range local networks. A local sensor network may still collect data before a gateway forwards selected information through a satellite link.

2.3.7 Comparative View

The following table summarizes the conceptual position of the technologies. The exact performance of a particular implementation varies, so the table should be used for selection logic rather than as a rigid specification.

Table 2.3.1 Conceptual comparison of communication technologies

Technology	Primary role	Typical IoT emphasis	Key caution
Wi-Fi	Local networking	Connectivity and higher data demand	Energy/infrastructure
RFID	Identification	Tracking and access	Reader/tag arrangement
Bluetooth	Short-range link	Personal-area and low-power devices	Short physical range
RF	Radio communication concept	Flexible wireless links	Not a single protocol
Satellite	Wide-area connectivity	Remote locations	Cost and propagation considerations

Combining technologies in one IoT system

A real deployment can use more than one communication technology. For example, a set of local sensors may use Bluetooth or another short-range link to a gateway, while the gateway uses Wi-Fi or another wide-area connection to a server. A remote gateway might use satellite connectivity when terrestrial access is unavailable.

SGOU

This layered combination separates local sensing needs from backhaul needs. It also allows each link to be selected for the role it performs best.

The resulting system is often more practical than forcing every sensor to communicate directly with the cloud.

Recap

- Wi-Fi provides local wireless networking and is useful where infrastructure and power support higher communication demand.
- RFID focuses on radio-based identification and tracking.
- Bluetooth supports short-range device communication, with low-energy operation especially relevant to IoT.
- RF is a broad physical concept, while satellite communication supports very wide-area connectivity.
- These technologies can also be combined. A gateway may translate or bridge between local links and a wider network.

Objective Type Questions

1. Which technology provides local wireless networking through access points?
2. Which technology is commonly used for radio-based identification?
3. Which short-range technology is common in personal-area connectivity?
4. What broad term describes radio-frequency communication?
5. Which technology supports very wide-area connectivity through spacecraft?
6. What device provides local Wi-Fi access to a network?
7. What is the identifying object in an RFID system called?
8. Which Bluetooth family is especially relevant to low-power IoT?
9. What component reads RFID tags?
10. What type of communication path uses a spacecraft?

SGOU

Answers

1. Wi-Fi
2. RFID
3. Bluetooth
4. RF
5. Satellite
6. Access point
7. Tag
8. Bluetooth Low Energy
9. Reader
10. Satellite link

Assignments

1. Explain the basic role of Wi-Fi in IoT and discuss its strengths and limitations.
2. Describe RFID operation and distinguish passive and active tags at a conceptual level.
3. Explain Bluetooth and discuss why low-energy operation is important for IoT devices.
4. Explain why RF is a broad engineering term rather than a single IoT protocol.
5. Discuss satellite communication for remote IoT applications and explain its principal trade-offs.
6. Prepare a comparison table for Wi-Fi, RFID, Bluetooth, RF and satellite communication using range, data demand, energy and typical application.
7. Explain how a gateway can combine multiple communication technologies in a single IoT deployment.
8. For the following three scenarios, justify the most suitable communication technologies: warehouse tracking, wearable connectivity, and remote environmental monitoring.

SGOU

Suggested Reading

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Hersent, Olivier, David Boswarthick, and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Wiley, 2012.
3. Raj, Pethuru, et al. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017.

Reference

1. David Hanes et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Olivier Hersent, David Boswarthick, and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Wiley, 2012.
3. Pethuru Raj, Anupama C. Raman, and Harihara Subramanian. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017.

SGOU



IoT Access Technologies

Learning Outcomes

After completion of this unit, learner will be able to:

- explain the purpose of IEEE 802.15.4 in low-rate wireless networking.
- describe the main ideas and device roles in Zigbee networks
- explain the motivation for IEEE 802.11ah in IoT access
- compare 802.15.4/Zigbee and 802.11ah at a conceptual level
- identify suitable application situations for these access technologies

Prerequisite

Suppose a building contains hundreds of small sensors: room temperature, door state, energy meters and occupancy indicators. These devices may send only small messages, but they must operate for long periods and coexist with many neighbors. A conventional high-throughput wireless network is not automatically the best answer. You need an access technology designed around the realities of constrained devices. This unit introduces two important directions: the low-rate personal-area networking family associated with IEEE 802.15.4 and Zigbee, and the sub-GHz Wi-Fi approach represented by IEEE 802.11ah.

Keywords

IEEE 802.15.4, Zigbee, IEEE 802.11ah, Low-Rate Wireless, Coordinator, Router, End Device, Sub-GHz Wi-Fi

SGOU

Discussion

2.4.1 Why access Technologies are Needed

An access technology defines how devices obtain and use a communication medium. For IoT, the access layer must often support many devices, modest data volumes, limited energy and deployment environments that make wiring expensive.

IEEE 802.15.4 and Zigbee are associated with low-rate, low-power wireless networking. IEEE 802.11ah extends the Wi-Fi family into sub-GHz operation to address longer-range and large-scale IoT-oriented connectivity.

These technologies should be understood in terms of their roles: the problems they address, how devices communicate, the types of networks they support, and the trade-offs they introduce.

2.4.2 IEEE 802.15.4: Conceptual Role

IEEE 802.15.4 specifies a low-rate wireless personal-area networking foundation. It is designed for applications where devices exchange relatively small amounts of information and where power efficiency is important.

The standard addresses lower layers of communication, providing mechanisms for the physical and medium-access aspects of the network. Higher-layer networking and application functions can be supplied by technologies built on top of it.

This layered view is important. A standard at one layer does not by itself describe every behavior of a complete IoT solution. A complete system combines physical communication, network organization and application semantics.

2.4.3 Device Roles and Network Organization

A low-power wireless network can include a coordinating device and simpler end devices. Some network arrangements also use routers that forward traffic for other nodes. The exact roles depend on the higher-level technology and topology.

The reason for differentiated roles is practical. A constrained end device can focus on sensing and sleep most of the time, while a more capable router remains available to forward traffic.

This division allows the network to combine battery-powered sensing with a communication structure that can cover a larger area than a single direct link.

2.4.4 Zigbee as a Networking Technology

Zigbee builds networking capabilities around low-rate wireless communication and is widely associated with automation, sensing and control applications. It provides a framework in which devices can form organized networks rather than operating as isolated point-to-point links.

SGOU

Zigbee networks can use coordinator, router and end-device roles. The coordinator helps establish and manage the network; routers can forward traffic; end devices can perform sensing or control with lower power requirements.

The practical value of Zigbee lies in combining low-power communication with network organization suitable for distributed devices.

2.4.5 Zigbee Topologies

Common conceptual arrangements include star, tree and mesh forms. In a star, end devices communicate through a central coordinator. In a tree, routers can extend the network through branches. In a mesh, multiple paths may be available among participating nodes.

Mesh networking can improve path flexibility because traffic may be able to move through alternative neighbors. However, additional routing and coordination also increase network complexity.

Topology should therefore be selected according to physical layout, reliability requirements, device capability and expected traffic.

2.4.6 Zigbee Communication and Energy Behavior

Low-power IoT devices often spend much of their time inactive. End devices can sleep between measurement or communication events, while other network devices may remain more continuously available.

This creates an important trade-off. A sleeping device can save energy, but it cannot receive arbitrary messages while fully asleep. Application timing and network design must therefore coordinate wake-up and communication behavior.

Battery life is consequently a system property. It depends on sensing frequency, radio activity, processing, sleep duration and the number of messages handled.

2.4.7 IEEE 802.11ah: The Motivation

IEEE 802.11ah is a Wi-Fi amendment designed for operation in sub-GHz bands, targeting applications that benefit from longer coverage and support for many low-power devices.

The sub-GHz operating environment can provide useful propagation characteristics compared with conventional higher-frequency Wi-Fi deployments, making it attractive for IoT scenarios where coverage is important.

The technology retains a relationship with the broader Wi-Fi ecosystem while addressing requirements that differ from ordinary high-throughput local-area use.

2.4.8 802.11ah Architecture and IoT use

A conceptual 802.11ah deployment can include an access point and numerous stations such as meters, sensors and controllers. The access point coordinates access to the wireless medium and provides a connection toward the rest of the network.

SGOU

Potential applications include smart metering, building monitoring, agricultural sensing and large collections of distributed devices. The suitability depends on range, traffic pattern, energy requirements and deployment conditions.

As with every access technology, the correct question is not "Is 802.11ah better?" but "Does its operating model match this application?"

2.4.9 Comparing 802.15.4/Zigbee and 802.11ah

The following table gives a conceptual comparison. It deliberately avoids fixed numerical claims because actual performance depends on implementation, regulatory domain and deployment conditions.

Table 2.4.1 Conceptual comparison of IoT access technologies

Aspect	IEEE 802.15.4 / Zigbee	IEEE 802.11ah
Primary orientation	Low-rate, low-power networking	Sub-GHz Wi-Fi for IoT-oriented coverage
Network role	Zigbee adds organized networking above 802.15.4	Wi-Fi access point and stations
Typical traffic	Small sensor/control messages	IoT traffic with Wi-Fi ecosystem compatibility
Topology	Star/tree/mesh possibilities	Infrastructure-oriented access
Key emphasis	Constrained devices and networking	Coverage and large device populations

2.4.10 Interesting note: Layers are Complementary

A common source of confusion is treating IEEE 802.15.4 and Zigbee as identical terms. A useful mental model is that 802.15.4 supplies a lower-level wireless foundation, while Zigbee adds networking and application-oriented organization above that foundation.

Similarly, 802.11ah is an amendment within the Wi-Fi family, not a completely unrelated radio technology. Thinking in layers makes it easier to compare technologies without mixing physical, link and higher-level functions.

2.4.11 Application-Based Selection

For a small low-power mesh of sensors and control devices, a Zigbee-oriented solution may be attractive because the networking model fits distributed automation. For a large IoT deployment seeking longer coverage while remaining within a Wi-Fi family, 802.11ah may be considered.

SGOU

The final choice still depends on device cost, energy, required throughput, installation environment, security, ecosystem and maintenance. Technology selection is therefore a multi-criteria decision.

Recap

- IEEE 802.15.4 provides a low-rate wireless networking foundation
- Zigbee builds a broader networking framework around low-power communication
- IEEE 802.11ah extends Wi-Fi into sub-GHz operation for IoT-oriented coverage and scale.
- Coordinator, router and end-device roles help distribute responsibilities.
- Star, tree and mesh topologies provide different trade-offs in simplicity, reach and resilience.
- The central lesson is to compare access technologies by application requirements and by the layer of the communication stack they address.

Objective Type Questions

1. Which IEEE standard provides a low-rate wireless foundation?
2. Which technology builds networking capabilities around 802.15.4?
3. Which Wi-Fi amendment targets sub-GHz IoT operation?
4. Which Zigbee role coordinates the network?
5. Which Zigbee role can forward traffic?
6. Which Zigbee role is typically more constrained?
7. Which topology uses a central node?
8. Which topology can offer alternative paths?
9. Which layer distinction helps prevent confusion between 802.15.4 and Zigbee?
10. What kind of operation is important for battery-powered IoT nodes?

SGOU

Answers

1. 802.15.4
2. Zigbee
3. 802.11ah
4. Coordinator
5. Router
6. End device
7. Star
8. Mesh
9. Layering
10. Low-power operation

Assignments

1. Explain the conceptual role of IEEE 802.15.4 in low-rate wireless IoT networking.
2. Describe Zigbee coordinator, router and end-device roles with a labelled topology diagram.
3. Compare star, tree and mesh arrangements in a Zigbee-oriented network.
4. Explain why sleep and duty-cycle behavior are important in low-power access technologies.
5. Discuss the motivation for IEEE 802.11ah and explain why sub-GHz operation is relevant to IoT.
6. Compare IEEE 802.15.4/Zigbee and IEEE 802.11ah from the viewpoint of architecture, coverage, energy and application fit.
7. Prepare a case study for a smart-metering deployment and justify a suitable access technology.
8. Explain the importance of protocol layering when distinguishing a lower-level standard such as IEEE 802.15.4 from a broader networking technology such as Zigbee.

SGOU

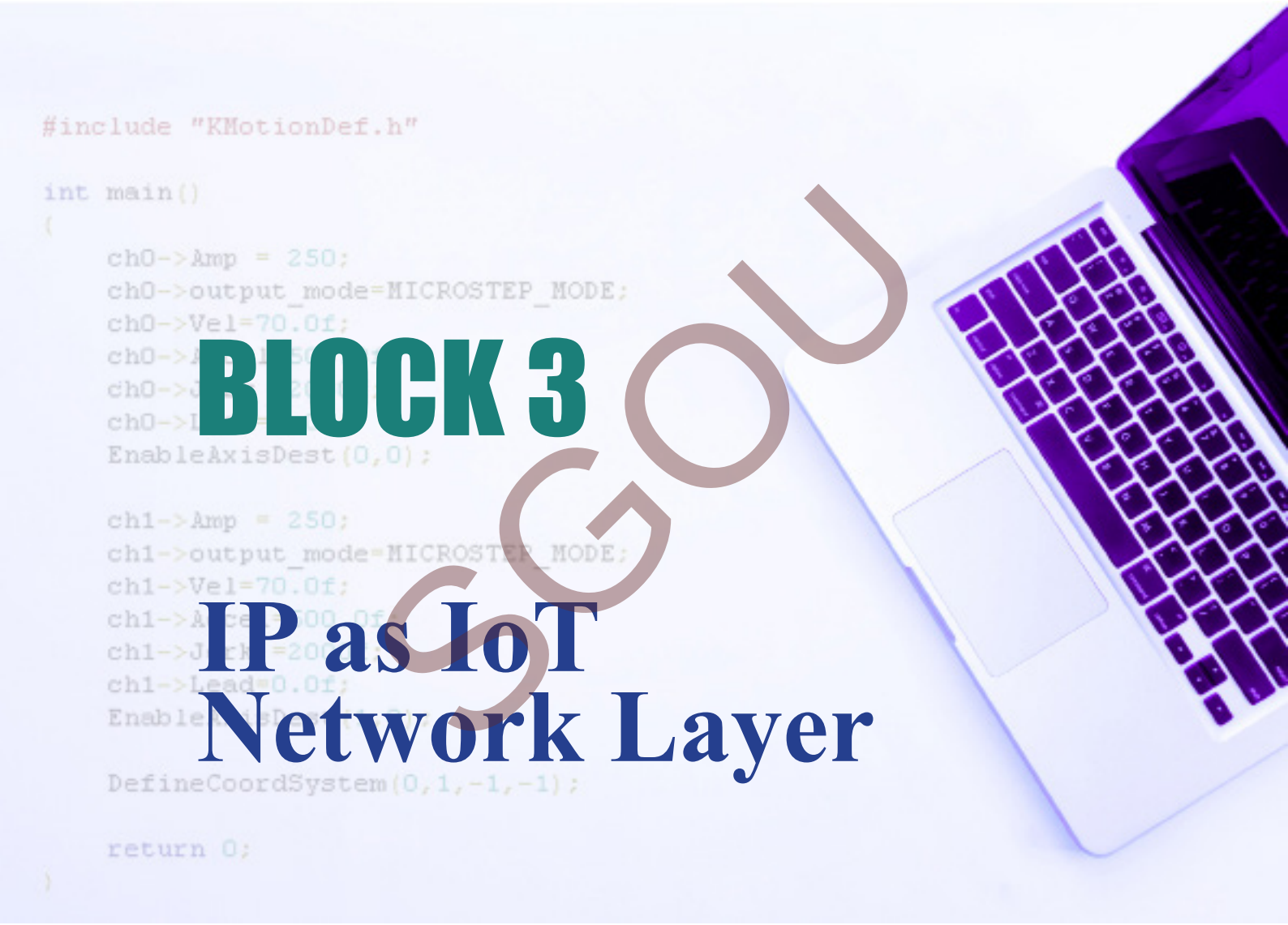
Suggested Reading

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Hersent, Olivier, David Boswarthick, and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Wiley, 2012.
3. Raj, Pethuru, et al. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017.

Reference

1. David Hanes et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Olivier Hersent, David Boswarthick, and Omar Elloumi. The Internet of Things: Key Applications and Protocols. Wiley, 2012.
3. Pethuru Raj, Anupama C. Raman, and Harihara Subramanian. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017.

SGOU



BLOCK 3

IP as IoT
Network Layer

SGOU



Business Case for IP

Learning Outcomes

After completion of this unit, learner will be able to:

- explain why IP is valuable for IoT connectivity
- identify major technical and business advantages of IP-based networking
- describe the roles of IPv4 and IPv6 in large-scale connected systems
- recognize how end-to-end connectivity, interoperability and routability support IoT growth

Prerequisite

Think about the number of digital systems you already use in a normal day: a phone, a smart television, a Wi-Fi printer, a payment terminal, a vehicle, a cloud service and perhaps a sensor in a building. They may be built by different manufacturers, use different hardware and serve very different purposes, yet many of them can exchange information through familiar Internet networking. The interesting question is not simply how they connect, but why a common network language makes such a diverse ecosystem manageable. As you study this unit, keep one idea in mind: an IoT deployment becomes much easier to scale when devices can use a common, widely understood networking foundation.

Keywords

Internet Protocol (IP), IPv4, IPv6, interoperability, end-to-end connectivity, routability, scalability, IoT networking

SGOU

Discussion

3.1.1 The IoT Connectivity Challenge

IoT changes the scale and diversity of networking. A conventional enterprise network may contain a manageable number of computers and servers, but an IoT deployment can contain sensors, meters, controllers, appliances, vehicles and gateways spread across homes, factories, farms and cities. These devices generate small but frequent messages, and many of them operate with limited memory, processor capacity and battery energy. A networking foundation therefore needs to be more than a method for moving packets; it must support growth, interoperability and operational continuity.

IP is valuable because it is already the common network-layer technology of the Internet. Instead of creating a separate networking language for every class of sensor, an organization can use a common addressing and routing model and then choose appropriate technologies underneath and above IP. This separation of concerns is important. A temperature sensor does not need to know whether its message will eventually travel over Ethernet, Wi-Fi, cellular connectivity or a low-power wireless link. The network layer provides a stable abstraction.

The business consequence is significant. Common networking reduces the risk of building an isolated technology island. A device can potentially communicate with existing infrastructure, management platforms and cloud services without requiring a completely separate network architecture. That does not mean every IoT device should connect directly to the public Internet; security, gateways and policy controls remain essential. The point is that IP gives the architecture a common vocabulary. See fig. 3.1.1

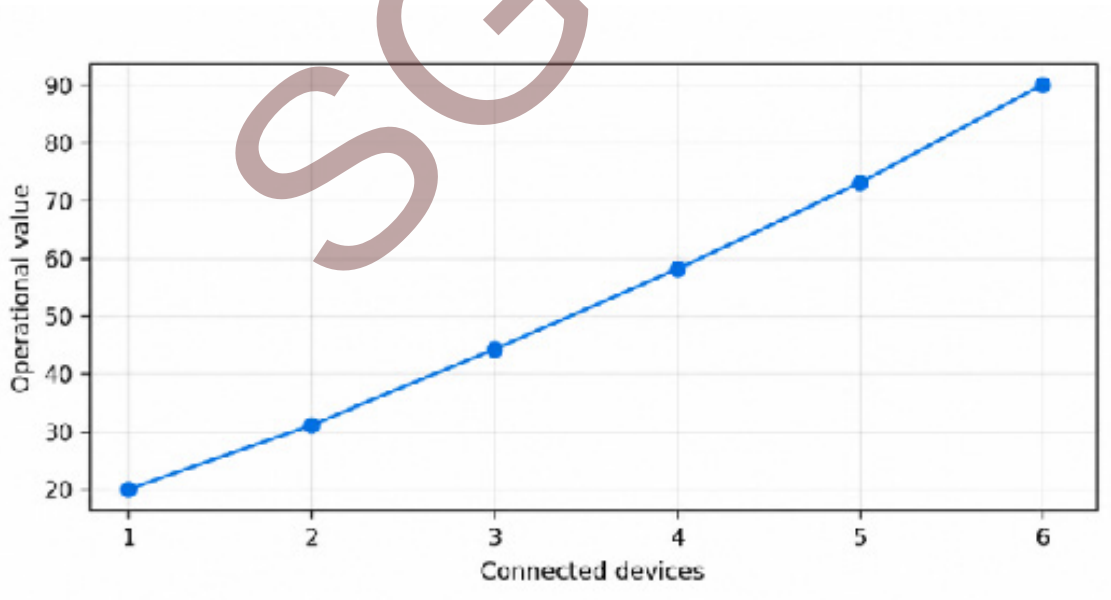


Fig. 3.1.1 Illustrative relationship between connected-device scale and value from a common IP foundation.

SGOU

3.1.2 What is the Business Case for IP?

A business case asks whether a technology creates enough value to justify its cost, risk and operational effort. For IP in IoT, the answer is strongly connected to reuse. Organizations already possess IP expertise, tools, monitoring practices, security mechanisms and network infrastructure. Extending those capabilities toward IoT can reduce the need to maintain a completely independent networking ecosystem.

Interoperability is one of the strongest arguments. IoT projects frequently combine components from multiple vendors. When a common network-layer technology is used, the organization can change a sensor supplier or communication medium without necessarily redesigning the whole application architecture. This flexibility is especially useful during long-lived deployments because individual devices may be replaced many times during the lifetime of the service.

Another business advantage is scalability. A prototype with a few devices may work with a proprietary protocol, but a production system often grows in geographic coverage and device count. IP's established addressing and routing model provides a foundation for that growth. The organization can also reuse familiar operational practices such as monitoring, troubleshooting, segmentation and access control.

Finally, IP supports gradual evolution. An IoT system can begin with local connectivity and later integrate with enterprise systems or cloud platforms. The network architecture does not have to be discarded simply because the application becomes larger. The business case therefore comes from a combination of technical reuse, supplier flexibility, operational familiarity and long-term scalability.

3.1.3 Interoperability and Open Networking

Interoperability is one of the strongest reasons for considering IP as the network layer for IoT. An IoT installation rarely consists of devices from a single manufacturer. Sensors, gateways, controllers, routers, cloud platforms and management systems may come from different vendors and may use different link technologies. A common network-layer approach reduces the amount of custom networking logic needed to make these components communicate.

Think of an IoT system as a conversation between many different speakers. Each speaker may have its own accent, but communication becomes easier when everyone follows a common language at an appropriate layer. IP plays a similar role. It provides common rules for addressing and forwarding packets while allowing the underlying communication technologies to vary. For example, one part of a deployment may use Ethernet, another Wi-Fi, and a constrained sensor segment may use a low-power wireless technology. The application can still be organized around IP connectivity.

Open networking also reduces dependence on a single proprietary networking environment. Suppose an organization initially purchases sensors from one supplier and later needs to replace them because of cost, availability or product lifecycle issues. When the network architecture uses widely adopted standards, replacing one class of device does not necessarily require the entire communication architecture to

SGOU

be redesigned. This does not guarantee plug-and-play interoperability: application protocols, data formats, security credentials and device capabilities must still agree. However, a common IP foundation removes one important source of incompatibility.

Interoperability can be considered at several layers. At the network layer, devices need compatible addressing and packet-forwarding mechanisms. At higher layers, they may require compatible transport protocols, application protocols and data models. Therefore, saying that two devices are "IP-compatible" does not mean that they automatically understand each other's application data. IP provides a common networking foundation; other standards complete the communication chain.

For IoT, this layered approach is especially useful because device lifetimes are often long. A building sensor installed today may remain in service for many years while the surrounding network infrastructure evolves. A standards-based network layer gives the deployment a better chance of accommodating future links, gateways and services without changing every device at the same time.

From a business perspective, interoperability can reduce integration effort, support multi-vendor procurement and simplify long-term expansion. From a technical perspective, it encourages modular architecture. From an operational perspective, it allows organizations to reuse established networking skills and management practices. These benefits explain why open, standards-based networking is an important part of the business case for IP.

Table 3.1.1 Interoperability considerations in proprietary and IP-based IoT networking.

Aspect	Proprietary approach	Open IP-based approach
Vendor dependence	Often higher	Can support multi-vendor choices
Network reuse	May require specialized infrastructure	Can reuse established IP infrastructure
Scalability	Depends on the proprietary ecosystem	Benefits from mature IP addressing and routing
Integration	May require custom gateways	Common network layer can simplify integration

3.1.4 End-to-End Connectivity

The end-to-end principle is an important idea in Internet architecture: functions that are meaningful only to the communicating endpoints are often best handled at the endpoints, while the network focuses on moving packets. IoT systems do not follow this principle rigidly in every deployment, because gateways, proxies and edge services can be necessary for security, protocol translation and local processing. Even so, an IP-centered architecture makes end-to-end communication a natural possibility.

SGOU

For example, a soil sensor may send a reading to an application service. The sensor does not need to understand the entire path between itself and the service. Routers forward packets based on network-layer information. This separation allows network topology to change without requiring the application to understand every intermediate link.

End-to-end connectivity also supports distributed applications. A device can communicate with another device, an edge service or a cloud platform using a common network layer. In industrial systems, this can support local control while still allowing selected data to reach enterprise analytics.

However, end-to-end reachability must not be confused with unrestricted exposure. Security architecture should determine who is allowed to communicate, which ports or services are accessible, and where traffic should be filtered. Network segmentation and firewalls may intentionally prevent direct communication.

The key learning point is that IP provides a general connectivity model, while security and policy determine the permitted communication relationships. Together, these ideas make IoT architectures both flexible and controllable.

3.1.5 Addressing and Scale

Every networked endpoint needs an identity that allows traffic to be delivered to the correct destination. Addressing becomes especially important in IoT because the number of connected objects can grow far beyond the number of conventional computers. IPv4 and IPv6 differ greatly in address-space size, and IPv6 was designed with a very large address space suitable for massive networking.

A large address space does not by itself solve every deployment problem. Devices may still be organized into subnets, and networks still need routing, address management, security policies and monitoring. Nevertheless, abundant addresses simplify the conceptual model: devices can be assigned network-layer identities without relying on scarce public addresses.

IPv6 also introduces mechanisms that are important for modern network design, including automatic address configuration approaches and standardized support for extension headers and neighbor discovery. In constrained IoT networks, not all of these mechanisms can be carried in their ordinary form without adaptation, which will become a major theme in later units.

Scalability has a physical dimension as well. A city-scale sensor deployment may span many administrative domains and communication technologies. A common IP addressing and routing model allows those domains to be connected using established Internet technologies.

For the learner, it is useful to separate three ideas: address capacity, routing scalability and operational scalability. IPv6 primarily addresses the first, while hierarchical network design, routing protocols and management practices address the others. Good IoT architecture requires all three.

SGOU

Table 3.1.2 Selected IPv4 and IPv6 characteristics relevant to IoT.

Aspect	IPv4	IPv6
Address size	32-bit	128-bit
IoT scale	More constrained	Very large address space
Constrained-link role	Legacy/coexistence	Central to modern IoT standards

3.1.6 IPv4 and IPv6 in IoT

IPv4 remains widely deployed and continues to be important in enterprise and Internet infrastructure. IPv6, however, is particularly significant for IoT because its address space is vastly larger and because current standards for constrained networking are centered on IPv6. The choice is therefore not simply a matter of replacing one version with another overnight; real deployments often need coexistence strategies.

IPv4 uses 32-bit addresses, while IPv6 uses 128-bit addresses. The difference is enormous. For IoT, this means IPv6 can support addressing at a scale that is difficult to approach with IPv4's address space. IPv6 also provides standardized mechanisms for address configuration and multicast that are useful in many networked environments.

A learner should avoid the misconception that IPv6 is automatically lightweight. The basic IPv6 header is larger than the IPv4 header, and constrained links may have small frame sizes. This is one reason optimization mechanisms are needed. 6LoWPAN, discussed later, allows IPv6 communication over low-power wireless personal area networks by introducing an adaptation layer and compression mechanisms.

In practice, IPv4 and IPv6 may coexist through dual-stack operation or translation mechanisms. IoT architects choose among these approaches based on device capabilities, infrastructure, application requirements and lifecycle considerations.

The important conclusion is that IPv6 is a strategic foundation for scalable IoT, but its successful use on constrained links depends on adaptation and optimization.

3.1.7 Routing and Reachability

IP networks are useful not only because they assign addresses but because routers can use those addresses to forward traffic across interconnected networks. IoT deployments often contain many network segments, gateways and wireless links. Routing allows packets to travel beyond the local link.

Reachability can be viewed as a chain of decisions. A device generates a packet, the local network determines the next hop, and routers continue forwarding the packet toward a destination prefix. Each router needs enough information to make that decision efficiently. In small IoT networks, routing may be simple; in larger low-power and lossy networks, specialized routing protocols are needed.

SGOU

This distinction leads to an important architectural lesson. IP provides a common network-layer abstraction, but the routing algorithm must match the characteristics of the environment. Later, RPL will be introduced as a routing protocol designed for low-power and lossy networks.

Routing also supports fault tolerance. If one path becomes unavailable, routing mechanisms may find another path. In wireless sensor networks, link quality can change because of interference, obstacles, battery conditions or environmental factors. A robust routing system must therefore react to changing conditions without generating excessive control traffic.

Thus, IP's value is not limited to addressing. It creates a scalable framework in which routing protocols can provide reachability across heterogeneous networks.

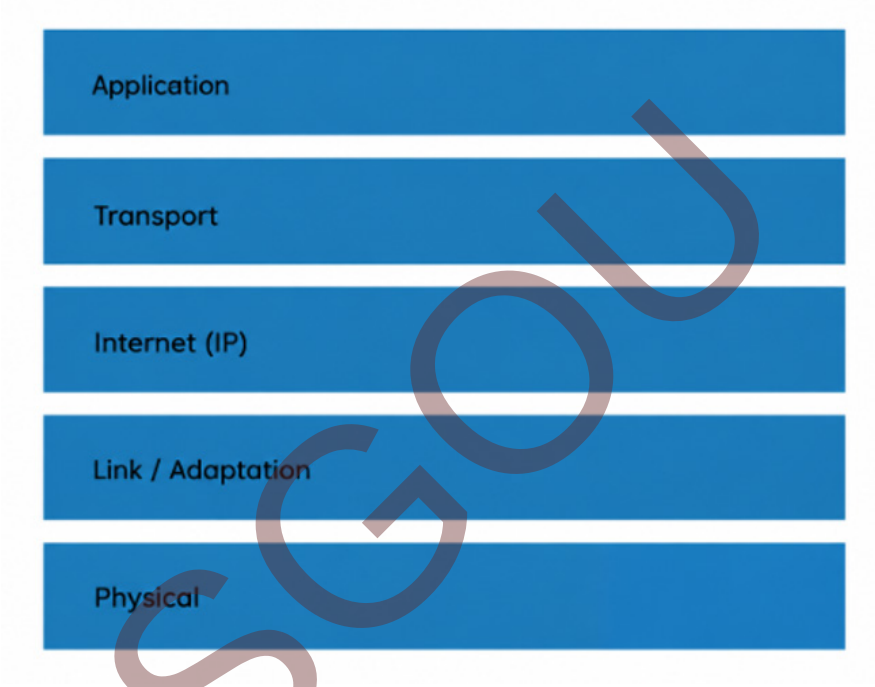


Fig. 3.1.2 IP-Centered view of an IoT Communication stack

3.1.8 Operational and Management Advantages

A network that works in a laboratory but cannot be monitored is difficult to operate in the real world. IoT deployments require visibility into device connectivity, packet loss, latency, addressing, routes and failures. IP's widespread adoption means that many operational concepts and tools already exist.

Network administrators understand familiar concepts such as addresses, prefixes, routing tables, ports and traffic flows. Monitoring systems can collect network-level information, while security tools can enforce segmentation and access policies. These capabilities can be extended toward IoT, although the constraints of small devices require careful implementation.

SGOU

IP also supports layered troubleshooting. If a sensor cannot reach a server, the problem can be considered in stages: Is the device powered? Is the local link available? Does it have a valid address? Is a route present? Is the transport service reachable? Is the application responding? A layered model helps narrow the problem.

Operational reuse can reduce training and support costs. An organization does not need a completely separate mental model for every connected device category.

At the same time, IoT management is not identical to conventional IT management. Devices may sleep, move, have extremely limited memory or remain deployed for many years. Therefore, IP gives a reusable foundation, but IoT-specific management practices must be added. The business advantage comes from combining the two rather than assuming one completely replaces the other.

3.1.9 IP and Architectural Flexibility

IoT architectures vary widely. Some applications use direct device-to-cloud communication, some place intelligence at an edge gateway, and others use local device-to-device interactions. An IP-based foundation can support all of these patterns because it is not tied to a single application topology.

Consider a factory. A vibration sensor might send raw measurements to an edge computer because local decisions must be made quickly. The edge computer may then send selected information to a cloud service for long-term analytics. The same network-layer technology can support both local and remote communication, while application and transport choices vary according to need.

Architectural flexibility also matters when a system evolves. An organization may initially deploy a gateway-based model because its sensors are highly constrained. Later, newer devices may become capable of direct IP communication. A common network foundation makes such migration easier because the application architecture does not need to be completely redesigned.

IP therefore functions as a stable abstraction across changing hardware. This is particularly valuable for IoT, where hardware generations can change faster than the intended service lifetime.

The correct way to think about IP is not as the only technology in an IoT stack, but as a common layer that allows many underlying and overlying technologies to coexist. Its value is greatest when the architecture uses this modularity deliberately.

3.1.10 Economic Effects of Standardization

Standards can influence cost in several ways. First, a common protocol increases the number of potential suppliers and implementations. Competition can reduce acquisition costs or improve product choice. Second, engineers can transfer knowledge between projects because core networking concepts remain familiar. Third, integration teams can reuse software libraries, test methods and operational procedures.

SGOU

There are also indirect benefits. A standardized network foundation can reduce the cost of future changes. Suppose a building owner wants to add occupancy sensors five years after the original installation. If the new devices can participate in the existing IP network, the integration challenge is smaller than if a completely separate network architecture is required.

Standardization can also improve longevity. IoT systems often remain in service for many years, while individual components may be replaced. Using widely supported protocols reduces the risk that a discontinued proprietary technology becomes the single point of failure for the entire system.

However, standardization is not free. Implementations must be tested, secured and maintained, and constrained devices may need adaptation layers. The business case should therefore compare total lifecycle cost rather than only initial purchase price.

The central conclusion is that IP's economic value comes from ecosystem effects: reuse, competition, skill availability, integration flexibility and long-term maintainability.

3.1.11 Limits and Trade-offs of IP

A balanced study must also identify what IP does not solve. IP does not define the meaning of sensor data, guarantee application-level security, provide perfect reliability, or remove the constraints of low-power radio links. It is a network-layer foundation, not a complete IoT architecture.

One trade-off is overhead. A general-purpose protocol carries information needed for interoperability and routing, but a tiny sensor may have very little useful payload. Later units will show why compression and adaptation are needed.

Another issue is security. An IP address does not prove that a device is trustworthy. Authentication, authorization, confidentiality and integrity require appropriate security mechanisms and operational controls. An exposed device can remain vulnerable even when its network protocol is standardized.

There is also a management trade-off. Large numbers of addresses and devices increase the need for automation. Manual configuration becomes impractical at scale.

Therefore, the correct business case is not “IP solves IoT.” A more accurate statement is that IP provides a strong common networking foundation, after which IoT-specific adaptation, optimization, security and management are required. This distinction prevents unrealistic expectations and leads to better architecture decisions.

3.1.12 Worked Scenario: Smart Campus

Consider a university campus with smart lighting, environmental sensors, energy meters, access systems and parking sensors. Each subsystem generates different kinds of information, but all of them need network connectivity.

An IP-centered architecture can provide a common network foundation while allowing different link technologies to be used where appropriate. A building may use

SGOU

Ethernet for controllers, Wi-Fi for mobile devices, and low-power wireless links for battery-operated sensors. Gateways or routers connect the different network segments.

The benefit becomes visible when the campus adds a new application. If the parking system can use the same network services and operational infrastructure, the campus does not need to create an entirely new networking environment. Monitoring, segmentation and troubleshooting practices can be reused.

Now consider lifecycle change. A sensor supplier is replaced after several years. If the replacement supports the required IP-based communication model, the network architecture can remain largely stable.

This scenario demonstrates a recurring pattern: IP creates a common layer; the physical and link technologies solve local connectivity problems; routing joins the networks; transport and application protocols provide the required services. IoT becomes manageable because the architecture is layered rather than monolithic. See fig.3.1.2

At this point, pause and connect the major ideas. IP is valuable to IoT because it provides a widely adopted network-layer foundation for addressing and forwarding. Its strongest advantages are interoperability, scalability, routability, reuse of operational knowledge and flexibility across heterogeneous links. IPv6 is especially important because its address space supports very large deployments, although constrained links require adaptation.

A useful mental model is a city road system. The road language and addressing rules provide a common framework, while different vehicles and local streets solve different physical needs. In IoT, IP plays a similar organizing role. It does not determine every detail, but it allows many parts of the system to fit together.

When evaluating an IoT architecture, ask: Can devices use a common network layer? Can the system grow without redesigning everything? Can administrators monitor and secure it? Can new suppliers be introduced? Can different communication technologies coexist? These questions translate the technical advantages of IP into business value.

Remember also that standardization introduces no magic. Constrained devices still need efficient mechanisms, security still needs deliberate design, and application interoperability still requires shared data meanings. These limits lead naturally to the next unit: whether IP should simply be adopted as it is, or adapted to the special conditions of IoT.

Recap

- IP provides a common network-layer foundation for heterogeneous IoT devices.
- Interoperability reduces integration barriers across vendors and technologies.
- IPv6 provides a very large address space suited to large-scale IoT.

SGOU

- IP supports addressing, routing and end-to-end communication while allowing layered architectures.
- Standardization can reduce lifecycle cost through reuse of skills, tools and infrastructure.
- IP alone does not solve application interoperability, security or constrained-link efficiency.

Objective Type Questions

1. What network-layer protocol provides a common foundation for heterogeneous IoT networks?
2. Which IP version uses 128-bit addresses?
3. What property allows devices from different vendors to communicate through common protocols?
4. Which function determines the path packets take between networks?
5. What term describes communication capability between endpoints across an IP network?
6. Which IP version is especially important for massive addressable IoT deployments?
7. What business benefit comes from reusing existing IP skills and tools?
8. What does IPv6 provide in very large IoT deployments?
9. Which architectural principle separates network forwarding from many endpoint-specific functions?
10. What does IP not automatically provide at the application layer?

SGOU

Answers

1. IP
2. IPv6
3. Interoperability
4. Routing
5. Reachability
6. IPv6

7. Reuse
8. Address space
9. End-to-end
10. Application semantics

Assignments

1. Explain the business case for using IP as a common network layer in IoT.
2. Discuss interoperability and supplier flexibility in an IP-based IoT architecture.
3. Explain how IPv6 supports large-scale IoT addressing.
4. Describe end-to-end connectivity and its limits in secure IoT systems.
5. Compare the technical and economic advantages of IP with a proprietary networking approach.
6. Explain the role of routing in connecting heterogeneous IoT networks.
7. Discuss the operational benefits of reusing IP-based monitoring and troubleshooting skills.
8. Analyze a smart-campus scenario and justify an IP-centered architecture.

Suggested Reading

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Olsson, Jerry. IPv6 Essentials. O'Reilly Media.
3. Kurose, James F., and Keith W. Ross. Computer Networking: A Top-Down Approach. Pearson.
4. Forouzan, Behrouz A. Data Communications and Networking. McGraw-Hill.
5. Shelby, Zach, and Carsten Bormann. 6LoWPAN: The Wireless Embedded Internet. Wiley.

SGOU

Reference

1. Shelby, Zach, et al. The Constrained Application Protocol (CoAP). RFC 7252, Internet Engineering Task Force, 2014.
2. Winter, Tim, et al. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, Internet Engineering Task Force, 2012.
3. Thubert, Pascal. 6LoWPAN Routing Header. RFC 8138, Internet Engineering Task Force, 2017.
4. Kushalnagar, Nandakishore, et al. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, IETF, 2007.

SGOU



IP Adoption or Adaptation for IoT

Learning Outcomes

After completion of this unit, learner will be able to:

- explain the difference between adopting IP directly and adapting IP for constrained networks
- describe why IPv6 is important for large IoT deployments
- explain the role of adaptation layers such as 6LoWPAN
- Identify suitable lightweight protocols for IoT environments.

Prerequisite

Imagine trying to send a long letter through a very small delivery tube. The language of the letter is perfectly understandable, but the tube cannot carry the whole letter in one piece. You could abandon the language, or you could keep the language and introduce a method for packaging, dividing and reassembling the message. IoT networking faces a similar choice. IP offers a powerful common language, but many IoT links are tiny, slow, energy-conscious or lossy. This unit explores how engineers preserve the benefits of IP while adapting its use to the realities of constrained networks.

Keywords

6LoWPAN, constrained node, constrained link, IPv6, adaptation layer, fragmentation, interoperability

SGOU

Discussion

3.2.1 Adoption versus Adaptation

The phrase "adoption or adaptation" captures a central IoT networking decision. Adoption means using IP mechanisms largely as they are used in conventional networks. Adaptation means retaining IP as the architectural foundation while changing how packets are represented, transported or routed over constrained environments.

The distinction is important because IoT links can be dramatically different from ordinary Ethernet. A battery-powered wireless sensor may have a small maximum transmission unit, low data rate, limited memory and a higher probability of packet loss. Simply placing ordinary protocol traffic on such a link can waste energy and bandwidth.

Adaptation does not mean abandoning interoperability. In fact, the purpose of an adaptation layer is usually to preserve the IP model while translating it into a form suitable for the underlying link. This is the role played by technologies such as 6LoWPAN for IPv6 over low-power wireless personal area networks.

Think of adaptation as using the same language with a more compact writing system when space is limited. The meaning is preserved, but the representation is changed. This idea will appear repeatedly in IoT networking: compress what can be inferred, fragment what cannot fit, and choose mechanisms that respect constrained resources.

3.2.2 Why IPv6 Matters for IoT

IPv6 matters to IoT primarily because of its enormous address space and its role in modern constrained-network standards. A connected-world architecture may require addresses for sensors, controllers, appliances, vehicles and infrastructure. The 128-bit IPv6 address space provides an exceptionally large pool.

IPv6 also supports mechanisms that fit the architecture of large interconnected networks. Stateless address configuration can simplify the process of assigning addresses, while multicast can support group communication patterns. Neighbor discovery provides local network functions, although constrained environments may need specialized optimization.

A common mistake is to assume that IPv6 automatically makes communication efficient. It does not. An IPv6 packet carries a larger base header than IPv4, and constrained links may have small frame sizes. Therefore, IPv6 must be adapted and optimized when used over such links.

This is why IoT networking standards often preserve IPv6 while introducing a layer between IPv6 and the low-power link. The adaptation layer can compress headers, handle fragmentation and provide other functions needed to make the IPv6 packet fit the link.

Thus, IPv6 provides the scalable network-layer identity, while adaptation provides the practical bridge to constrained devices.

SGOU

3.2.3 Constrained Nodes

A constrained node is a device whose resources are significantly limited compared with conventional computers or servers. The limitations may involve processing power, memory, energy, storage or communication capability. A small sensor does not necessarily have the resources to maintain large buffers or execute complex protocols.

Constraints influence protocol design. A device powered by a coin cell battery must minimize radio use because wireless transmission and reception consume valuable energy. A node with very little RAM may not be able to buffer many packets. A slow processor may struggle with computationally expensive operations.

The constraints are not identical for every device. One sensor may have sufficient memory but limited energy; another may have a stronger processor but a low-rate radio. Engineers therefore classify devices according to their actual capabilities rather than assuming that all IoT nodes are equally constrained.

Constrained nodes can still participate in IP networks. The important question is how much protocol functionality should run directly on the node and how much should be handled by gateways or edge devices. Adaptation and lightweight protocol choices help move this boundary toward direct IP communication without exceeding device capabilities.

Understanding constraints is the first step toward making a sensible adoption or adaptation decision.

3.2.4 Constrained Links

A constrained link is a communication path with limitations such as low bandwidth, small frame size, variable quality, high energy cost or frequent packet loss. Low-power wireless technologies are common examples.

The maximum transmission unit, or MTU, is particularly important. If an IP packet is larger than the link can carry, it cannot simply be transmitted unchanged. The packet must be fragmented or otherwise handled so that it fits. Fragmentation introduces additional state, transmission overhead and opportunities for failure.

Low bandwidth also changes the economics of protocol overhead. On a fast wired network, an extra few bytes may have little practical impact. On a low-rate wireless link used by thousands of sensors, those bytes can consume meaningful airtime.

Lossiness creates another challenge. A lost fragment may cause a larger packet to become unusable. Retransmission can consume more energy and increase latency. Protocol design must therefore balance reliability with resource consumption.

The lesson is that adaptation is driven by the characteristics of the link as much as by the capabilities of the node. An efficient IoT network treats the link as a first-class design constraint.

SGOU

3.2.5 The 6LoWPAN Idea

6LoWPAN stands for IPv6 over Low-Power Wireless Personal Area Networks. The technology family provides an adaptation approach that allows IPv6 packets to operate over constrained wireless links based on IEEE 802.15.4.

The key insight is that many IPv6 header fields can be compressed because information is predictable from the local context. Instead of transmitting every field in full, a compressed representation can indicate which values can be inferred by the receiver. This reduces the number of bytes placed on the constrained link.

6LoWPAN also addresses fragmentation because an IPv6 packet may be larger than the underlying link frame. The adaptation layer can divide a packet into fragments and provide information that allows the receiver to reconstruct it.

These functions do not replace IPv6. They make IPv6 practical over a link that cannot efficiently carry ordinary IPv6 packets. This is a classic example of adaptation preserving an architectural principle while changing its representation.

The broader lesson extends beyond 6LoWPAN: successful IoT networking often depends on inserting a carefully designed adaptation layer between a general-purpose network architecture and a highly constrained physical technology.

> 6LoWPAN is an adaptation approach: it does not replace IPv6 with a new network-layer addressing system. It helps IPv6 fit a constrained link.

3.2.6 Header Compression

Header compression reduces the number of bytes needed to transmit protocol headers. In IoT, this can be important because the payload of a sensor message may be tiny. If the packet contains only a few bytes of useful data, transmitting a comparatively large header can dominate the communication cost.

Compression is possible because protocol fields are often predictable. Addresses may share prefixes, some fields may remain unchanged between packets, and certain values can be inferred from link-layer context. A compressed packet therefore carries only the information that cannot be safely inferred.

The receiver reconstructs the original header from the compressed representation. This must be done consistently and unambiguously. Compression is useful only when the receiver has enough context to restore the required information.

Header compression illustrates a broader principle: optimization should remove redundancy rather than remove meaning. The network remains IP-compatible, but the constrained link carries a more compact representation.

Compression also reduces airtime. Less airtime can mean lower energy consumption and lower probability of collision on a shared wireless channel. Thus, a small reduction in bytes can create benefits beyond bandwidth alone.

SGOU

3.2.7 Fragmentation and Reassembly

Fragmentation is needed when a packet is too large for the maximum frame size of the underlying link. The adaptation layer divides the packet into smaller fragments that can be transmitted separately. The destination reassembles those fragments into the original packet.

Fragmentation solves a size mismatch, but it introduces cost. Each fragment needs additional information, and the loss of one fragment can prevent the complete packet from being reconstructed. More fragments also mean more opportunities for interference and retransmission.

For that reason, engineers should not think of fragmentation as a free feature. It is a necessary mechanism when packets do not fit, but reducing the need for fragmentation can improve efficiency. Header compression is valuable partly because it can make packets smaller.

A useful analogy is moving a large object through a narrow doorway. Cutting the object into pieces makes transport possible, but each cut creates assembly work and the loss of one piece prevents completion. Good design therefore considers both the ability to fragment and the value of keeping packets compact.

This trade-off becomes central when optimizing IP for IoT.

Table 3.2.1 Constraint-to-response mapping.

Constraint	Typical effect	Adaptation response
Small MTU	Packet may not fit	Fragmentation
Tiny payload	Headers become significant	Header compression
Low energy	Radio use is costly	Fewer/smaller transmissions
Lossy link	Retransmissions increase cost	Careful routing/reliability

3.2.8 Addressing and Neighbor Discovery

IPv6 addressing provides the identity needed for communication, but a constrained link cannot always afford the same amount of control traffic as a conventional network. Neighbor discovery mechanisms can generate messages that are useful on ordinary networks but expensive when thousands of small devices share a low-power link.

IoT adaptations therefore seek to reduce unnecessary communication. Some information can be inferred from link-layer identifiers or local context. Other mechanisms can be optimized to reduce the number or size of messages exchanged.

The objective is not to remove essential functions. A node still needs to know how to reach relevant neighbors or routers, and addresses still need to be resolved appropriately.

SGOU

The goal is to achieve those functions with resource-aware signaling.

This distinction between function and representation is important. The function "identify a neighbor" may remain necessary even when the message format and frequency are changed.

As you continue, keep asking two questions: What function does this protocol message provide? And can the same function be achieved with less energy, bandwidth or memory? Those questions are the foundation of IoT optimization.

3.2.9 Adoption Strategies in Real Networks

There is no single adoption strategy for every IoT project. A deployment may use direct IPv6 connectivity for capable devices, adaptation for constrained wireless nodes, gateways for legacy equipment, or dual-stack operation where IPv4 remains necessary.

A practical strategy begins with device capabilities. If a device can support IPv6 efficiently, direct adoption may be reasonable. If the device uses a highly constrained link, an adaptation layer may be required. If an old proprietary controller cannot support IP, a gateway may translate between the legacy protocol and an IP-based architecture.

The application also matters. A closed industrial system with strict latency requirements may make different choices from a city-scale environmental sensing network. Security requirements, device lifetime and maintenance capability also influence the decision.

A good architecture therefore avoids the false choice that "everything must be IP directly" or "constrained devices should never use IP." Instead, it uses IP where it creates value and adaptation where constraints require it.

The result is a layered ecosystem in which different technologies can coexist while retaining a common network-layer direction.

3.2.10 Worked Scenario: Smart Agriculture

Imagine a farm with hundreds of battery-powered soil-moisture sensors. Each sensor sends a small reading several times an hour. The radio link is low power and may experience interference. A conventional IP packet representation would contain information that is useful but expensive to transmit repeatedly.

An adapted IPv6 solution can reduce header overhead and fragment packets when necessary. A gateway may connect the low-power network to a higher-capacity IP network, while routing mechanisms maintain reachability within the field.

The business objective is not merely "use IPv6." It is to achieve reliable, scalable sensing while preserving battery life and keeping the system interoperable with the farm's wider information infrastructure.

This example shows why adoption and adaptation are complementary ideas. The farm can adopt IP at the architectural level while adapting its representation and behavior at the constrained edge.

SGOU

When analyzing such a system, identify the node constraints, link constraints, addressing needs, routing requirements and gateway responsibilities. These dimensions usually reveal where adaptation is needed.

3.2.11 Benefits and Limits of Adaptation

Adaptation can reduce packet size, support fragmentation, improve compatibility with constrained links and allow a common IP architecture to reach resource-limited devices. It can also simplify integration because applications can continue to use IP-based interfaces while the lower layers handle local constraints.

However, adaptation adds complexity. Compression and fragmentation require additional processing and state. Interoperability depends on consistent implementations. Troubleshooting can become more complicated because engineers must understand both the IP layer and the adaptation layer.

Security must also be considered. An adaptation layer does not remove the need for authentication, confidentiality or integrity. Engineers must ensure that optimization mechanisms do not create unexpected security weaknesses.

The right conclusion is not that adaptation is always better than direct adoption. The choice depends on constraints. If the link and node can handle ordinary IP efficiently, additional adaptation may be unnecessary. If the environment is highly constrained, adaptation may be essential.

This is a recurring engineering principle: introduce complexity only when it solves a real constraint, and then measure whether the benefit justifies the added mechanism.

3.2.12 Lightweight Application Protocols

IoT devices often operate with limited processing power, memory, battery capacity and network bandwidth. Conventional communication protocols may introduce additional overhead in such environments. Lightweight protocols are designed to reduce communication overhead and support efficient data exchange among connected devices. CoAP, MQTT and MQTT-SN are application-layer protocols that address different communication requirements in IoT environments.

3.2.12.1 Constrained Application Protocol (CoAP)

The Constrained Application Protocol (CoAP) is a lightweight application-layer protocol developed by the Internet Engineering Task Force (IETF) for resource-constrained devices and networks. It was standardized in 2014 through RFC 7252. CoAP follows a request-response communication model similar to HTTP and is suitable for exchanging small amounts of data in constrained IoT environments.

In CoAP, a client sends a request to a server to access or modify a resource, and the server returns an appropriate response. CoAP commonly uses the User Datagram Protocol (UDP) to reduce communication overhead. Since UDP does not guarantee message delivery, CoAP provides its own message-level reliability mechanism. Confirmable messages require acknowledgements and can be retransmitted if an acknowledgement

SGOU

is not received. Non-confirmable messages do not require acknowledgements and are suitable when occasional message loss is acceptable. CoAP can also operate over TCP, TLS and WebSockets, as specified in RFC 8323.

Characteristics of CoAP

- **UDP-based communication :** CoAP commonly uses UDP, which does not require connection establishment and has a smaller transport header than TCP. CoAP provides its own reliability mechanism when required.
- **HTTP-like request methods :** CoAP supports GET, POST, PUT and DELETE methods to retrieve, submit, update and remove resources.
- **Multicast support :** CoAP supports multicast requests, allowing a client to communicate with multiple devices simultaneously. Multicast requests use Non-confirmable messages.
- **Low communication overhead :** CoAP uses a compact message format with a fixed header of four bytes, helping reduce the number of bytes transmitted.
- **Resource-oriented communication :** Device information and services are represented as resources identified by Uniform Resource Identifiers (URIs). Clients access these resources using CoAP request methods.

3.2.12.2 Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) is a lightweight messaging protocol designed for machine-to-machine (M2M) and IoT communication. It is suitable for networks with limited bandwidth, intermittent connectivity and resource-constrained devices. MQTT generally operates over TCP/IP and requires an ordered, reliable, bidirectional transport. It can also operate over WebSockets and over TCP connections secured using TLS.

Unlike CoAP's request-response model, MQTT uses a publish-subscribe communication model managed by a central broker. A publisher sends a message to a topic through the broker. The broker forwards the message to clients that have subscribed to that topic. This arrangement allows devices to exchange information without maintaining direct communication with one another.

For example, temperature sensors in a building can publish readings to a topic named building/temperature. A monitoring application subscribed to this topic receives the readings through the MQTT broker.

Characteristics of MQTT

- **Publish-subscribe communication :** MQTT uses publishers, subscribers and a broker. Publishers send messages to topics, while subscribers receive messages from their subscribed topics. This supports loosely coupled communication between devices.
- **Lightweight messaging :** MQTT uses a compact message format and reduces communication overhead, making it suitable for bandwidth-constrained networks.

SGOU

- **Quality of Service (QoS):** MQTT provides three message delivery levels. QoS 0 provides at-most-once delivery, QoS 1 provides at-least-once delivery, and QoS 2 provides exactly-once delivery at the MQTT protocol level. These levels allow applications to select an appropriate balance between delivery assurance and overhead.

- **Retained messages:** The broker can retain the latest message published to a topic. When a new client subscribes to that topic, the broker can send the retained message without waiting for a new publication.

- **Last Will and Testament:** A client can configure a Will Message when connecting to the broker. If the broker detects an unexpected disconnection, it publishes the Will Message to the specified topic, allowing other clients to be informed.

- **Session management:** MQTT supports session mechanisms that can preserve subscription information and store eligible messages for delivery after a temporary disconnection, depending on the session configuration.

3.2.12.3 MQTT for Sensor Networks (MQTT-SN)

MQTT for Sensor Networks (MQTT-SN) is a lightweight messaging protocol designed for wireless sensor networks (WSNs). While conventional MQTT generally uses TCP/IP, MQTT-SN does not require TCP/IP and can operate over constrained wireless technologies such as Zigbee. It is intended for sensor devices with limited processing power, memory, energy and communication bandwidth.

MQTT-SN retains the publish-subscribe model of MQTT but introduces mechanisms suited to sensor networks. Communication between MQTT-SN clients and an MQTT broker can be handled through an MQTT-SN gateway. The gateway acts as an intermediary, enabling communication between the sensor network and conventional MQTT infrastructure.

SGOU

Characteristics of MQTT-SN

- **MQTT-based communication model:** MQTT-SN retains the publish-subscribe approach and supports features such as publishing, subscribing and Will Messages. A gateway can connect MQTT-SN clients to an MQTT broker.

- **Reduced message overhead:** MQTT-SN uses short topic identifiers instead of transmitting complete topic names in every message. A registered topic can be represented by a two-byte identifier, reducing transmission overhead.

- **Support for sleeping devices:** MQTT-SN provides mechanisms for devices that periodically enter sleep mode to conserve energy. The gateway can store messages for a sleeping client and deliver them when the client becomes active, subject to the protocol's sleep and message-handling procedures.

- **Gateway discovery:** Clients can discover available gateways through announcement and search procedures. This helps establish communication between sensor devices and gateways.

Differences between MQTT and MQTT-SN

Although MQTT and MQTT-SN share the publish-subscribe communication model, they differ in their transport requirements and mechanisms for constrained devices.

- **Transport requirements:** MQTT generally operates over TCP/IP, whereas MQTT-SN can operate over non-TCP/IP sensor networks.
- **Topic representation:** MQTT uses topic names, whereas MQTT-SN can use compact topic identifiers to reduce message size.
- **Will Message management:** MQTT-SN provides separate operations for updating the Will Topic and Will Message. In MQTT, the Will Message is configured during connection establishment.
- **Security considerations:** MQTT-SN uses a different connection mechanism from MQTT and does not define the same username and password fields in its CONNECT message. Therefore, deployments should provide suitable authentication, confidentiality and integrity mechanisms. Depending on the network architecture, security may be implemented through the underlying network or protocols such as Datagram Transport Layer Security (DTLS).

CoAP, MQTT and MQTT-SN support different communication requirements in IoT environments. CoAP provides compact request-response communication, MQTT supports broker-based publish-subscribe messaging, and MQTT-SN adapts publish-subscribe communication for constrained wireless sensor networks.

The selection of a protocol depends on the communication model, device capabilities, network characteristics, energy requirements and application needs. These protocols help IoT systems exchange information efficiently while operating within the limitations of constrained devices and networks.

Related mechanism: SCHC

Static Context Header Compression and Fragmentation (SCHC) is a lightweight adaptation-layer mechanism designed for networks with very limited frame sizes, particularly Low-Power Wide-Area Networks (LPWANs). It reduces packet overhead by compressing headers using shared context rules. If a compressed packet is still too large for the underlying network, SCHC can divide it into fragments for transmission and reassemble them at the receiving end.

Note: SCHC should be discussed separately from CoAP, MQTT and MQTT-SN because it does not provide an application-layer messaging model. Instead, it optimizes the transmission of packets generated by protocols such as IPv6 and UDP.

SGOU

Recap

- Adoption uses IP directly where device and link capabilities permit; adaptation modifies representation or behavior for constraints.
- IPv6 is central to scalable modern IoT networking.
- Constrained nodes may have limited energy, memory, processing or storage.
- Constrained links may have small MTUs, low bandwidth and variable reliability.
- 6LoWPAN provides an adaptation approach for IPv6 over IEEE 802.15.4 low-power links.
- Header compression reduces overhead; fragmentation handles packets that cannot fit.
- CoAP is a lightweight application-layer protocol that uses a request-response model and commonly operates over UDP.
- MQTT uses a broker-based publish-subscribe model and provides three Quality of Service (QoS) levels for message delivery.
- MQTT-SN adapts MQTT communication for wireless sensor networks using compact topic identifiers, gateway discovery and support for sleeping devices.
- SCHC is an adaptation-layer mechanism that reduces packet overhead through header compression and supports fragmentation and reassembly in constrained networks.

SGOU

Objective Type Questions

1. What does adaptation mean in constrained IoT networking?
2. Which technology adapts IPv6 for IEEE 802.15.4 low-power links?
3. What is the purpose of header compression?
4. What problem does fragmentation solve?
5. Which type of device has severe limits on memory or energy?
6. What does MTU stand for?
7. Why can ordinary protocol headers be costly for tiny sensor payloads?
8. What layer preserves the IP architecture while handling link-specific constraints?

9. What factor determines whether direct IP adoption is practical?
10. What is one reason IPv6 is important for IoT?
11. Which protocol is commonly used for lightweight request-response communication in IoT?
12. Which transport protocol is commonly used by CoAP?
13. Which component manages message exchange in MQTT?
14. Which protocol is designed for publish-subscribe communication in wireless sensor networks?

Answers

1. Adaptation
2. 6LoWPAN
3. Overhead reduction
4. Packet-size mismatch
5. Constrained node
6. Maximum Transmission Unit
7. Airtime
8. Adaptation layer
9. Device/link capability
10. Address space
11. CoAP
12. UDP
13. Broker
14. MQTT-SN

SGOU

Assignments

1. Explain adoption and adaptation in the context of IoT networking.
2. Discuss why IPv6 is important for IoT.
3. Explain the characteristics of constrained nodes and constrained links.
4. Describe the purpose and functions of 6LoWPAN.
5. Explain header compression with a suitable IoT example.
6. Explain fragmentation and reassembly and discuss their limitations.
7. Compare direct IPv6 adoption, adaptation and gateway-based integration.
8. Analyze a smart-agriculture scenario and recommend an appropriate networking approach.
9. Explain the role of lightweight application protocols in IoT communication. Describe the architecture, working principles and major characteristics of CoAP, MQTT and MQTT-SN.
10. Compare MQTT and MQTT-SN with respect to their communication models, transport requirements and suitability for resource-constrained devices. Briefly explain the role of SCHC in reducing communication overhead in constrained IoT networks.

Suggested Reading

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Olsson, Jerry. IPv6 Essentials. O'Reilly Media.
3. Kurose, James F., and Keith W. Ross. Computer Networking: A Top-Down Approach. Pearson.
4. Forouzan, Behrouz A. Data Communications and Networking. McGraw-Hill.
5. Shelby, Zach, and Carsten Bormann. 6LoWPAN: The Wireless Embedded Internet. Wiley.

SGOU

Reference

1. Shelby, Zach, et al. The Constrained Application Protocol (CoAP). RFC 7252, Internet Engineering Task Force, 2014.
2. Winter, Tim, et al. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, Internet Engineering Task Force, 2012.
3. Thubert, Pascal. 6LoWPAN Routing Header. RFC 8138, Internet Engineering Task Force, 2017.
4. Kushalnagar, Nandakishore, et al. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, IETF, 2007.

SGOU



Need for Optimization of IP

Learning Outcomes

After completion of this unit, learner will be able to:

- explain why ordinary IP communication can be costly for constrained IoT systems
- identify sources of communication overhead in IoT networks
- describe the effects of bandwidth, energy, memory, MTU and packet-loss constraints
- explain why optimization must preserve interoperability while reducing unnecessary overhead

Prerequisite

Suppose a battery-powered sensor wakes for only a few milliseconds, measures temperature, and sends a tiny reading. If the useful measurement is only a few bytes but the communication process repeatedly exchanges larger control information, energy is spent on communication rather than sensing. Now imagine thousands of such devices sharing a low-rate wireless link. A small inefficiency repeated many times becomes a system-level problem. This unit develops the reasoning needed to see where such overhead comes from and why optimization is not merely a matter of making packets smaller.

Keywords

overhead, MTU, fragmentation, latency, energy efficiency, bandwidth, packet loss, constrained network

SGOU

Discussion

3.3.1 Why Optimization Matters

Optimization becomes necessary when the cost of communication is large compared with the useful information being delivered. IoT devices often send short measurements, yet the complete communication process can involve headers, control messages, routing information, acknowledgements and retransmissions.

Consider a battery-powered sensor that sends a five-byte measurement. If the network adds several layers of overhead, the radio may transmit far more bytes than the application actually produced. This does not mean the headers are useless; they provide addressing, delivery and protocol functions. The engineering challenge is to reduce unnecessary repetition and represent required information efficiently.

Optimization is also a scale problem. An inefficiency that is negligible for one device can become significant when repeated by ten thousand devices. Airtime becomes scarce, collisions increase, batteries drain faster and network management becomes more difficult.

Therefore, optimization should be understood as system engineering. It is not simply "make the packet smaller." The goal is to preserve required functionality while reducing resource consumption and maintaining interoperability.

3.3.2 Sources of Overhead

Communication overhead comes from several sources. Network headers identify destinations and provide forwarding information. Transport headers support services such as multiplexing and checks. Control traffic supports discovery, routing and maintenance. Security mechanisms add fields and processing needed for protection. Retransmissions occur when packets or fragments are lost.

The amount of overhead depends on the application and network. A device sending large measurements may have a relatively small overhead percentage, while a sensor sending a few bytes may have a much larger percentage.

It is useful to distinguish fixed overhead from variable overhead. A fixed header may appear in every packet, while retransmission overhead appears only when loss occurs. Routing control traffic may increase when topology changes.

Optimization targets each source differently. Header compression reduces representation overhead. Fragmentation strategies address packet-size mismatch. Routing optimization reduces unnecessary control traffic. Application batching can sometimes reduce the number of packets.

The most effective optimization therefore begins with measurement. Engineers should identify which part of the communication cost dominates before selecting a mechanism. See fig. 3.3.1

SGOU

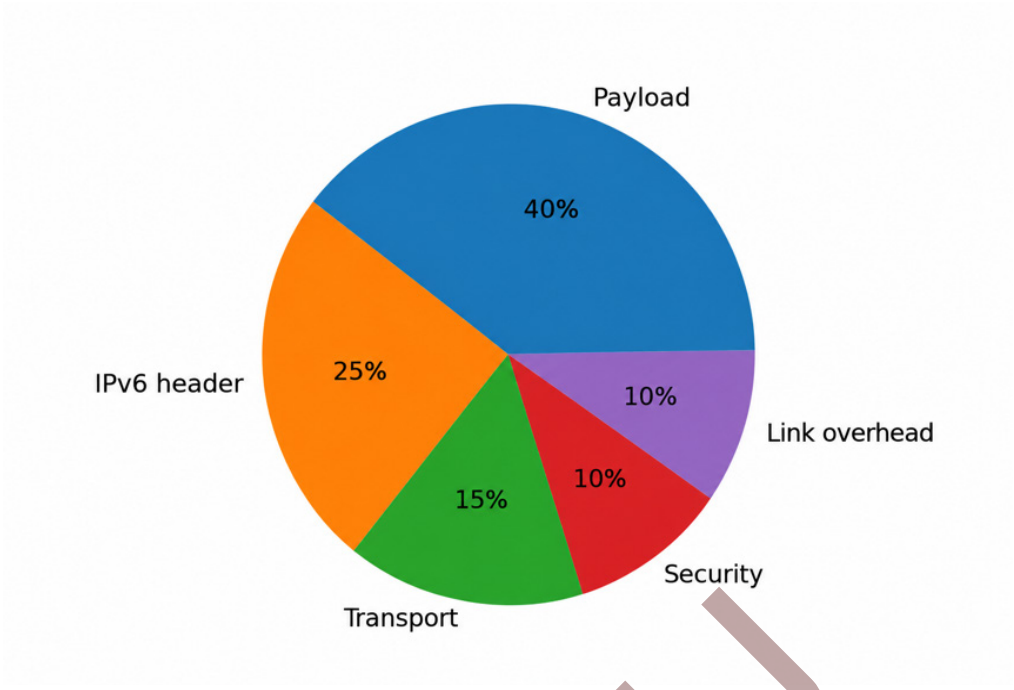


Fig. 3.3.1 Illustrative communication overhead; actual proportions vary by protocol and application.

3.3.3 Header Compression in IoT Networks

Header compression is an important optimization technique used in Internet of Things (IoT) networks to reduce the amount of data transmitted over low-power communication links. IoT devices commonly exchange small amounts of application data, such as temperature readings, humidity values and sensor status information. However, every transmitted packet contains protocol headers that carry essential information required for communication, including source and destination addresses, packet length, protocol identification and other control information. When the size of these headers is large compared with the actual application data, a considerable portion of the available network capacity is consumed by communication overhead.

This problem is particularly significant in constrained IoT networks because devices often communicate through low-power wireless technologies such as IEEE 802.15.4. These technologies have limited frame sizes and bandwidth compared with conventional networks. For example, a sensor may transmit only a few bytes of measurement data, while the IPv6 header alone occupies 40 bytes. Additional headers from other protocol layers may further increase the packet size. As a result, transmitting the complete headers repeatedly can consume valuable network resources and increase energy consumption.

Header compression reduces this overhead by representing protocol header information using fewer bits. Instead of transmitting all header fields in their original form, the compression mechanism identifies information that is repeated, predictable or already known to the communicating devices. Such information can be represented using shorter fields or omitted from the transmitted packet when it can be reliably reconstructed at the receiving end. The receiver uses the available context and

SGOU

decompression rules to reconstruct the original header before forwarding the packet to the appropriate protocol layer.

For example, consider two IoT devices that frequently exchange IPv6 packets. Their communication may use the same source address, destination address and other predictable header values for multiple transmissions. Sending all these values repeatedly creates unnecessary overhead. Header compression allows the devices to represent such information in a compact form, reducing the number of bytes that must be transmitted over the wireless link. The receiver then reconstructs the required header using the information already available to it.

In IoT networks, 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) provides mechanisms for compressing IPv6 headers so that IPv6 communication can operate efficiently over constrained networks such as IEEE 802.15.4. It reduces the size of IPv6 and certain related protocol headers by eliminating or shortening fields that can be inferred from the communication context. This makes it possible to transmit IPv6 packets more efficiently over wireless links with limited frame capacity.

The major benefits of header compression include:

1. **Reduced communication overhead:** It decreases the number of bytes required to represent protocol headers, allowing a greater proportion of the transmitted packet to contain useful application data.
2. **Improved bandwidth utilization:** Smaller packets consume less transmission capacity, allowing constrained wireless networks to use their available bandwidth more efficiently.
3. **Reduced energy consumption:** Transmitting and receiving fewer bytes reduces radio activity, which helps extend the battery life of resource-constrained IoT devices.
4. **Reduced fragmentation:** Smaller compressed packets are more likely to fit within the available frame size of low-power wireless technologies, reducing the need to divide packets into multiple fragments.
5. **Improved communication efficiency:** By reducing unnecessary header information, devices can exchange data with less communication overhead while retaining the essential functions of IP-based communication.

However, header compression also introduces some processing requirements. The sending device must compress the header, while the receiving device must decompress it to reconstruct the original information. Both devices must also maintain or obtain the necessary information required for correct decompression. Therefore, the compression mechanism should be designed to minimize processing overhead while providing effective reduction in packet size.

Header compression helps overcome the limitations of low-power IoT networks by reducing the overhead associated with IP communication. It enables constrained devices to exchange IP packets more efficiently, conserve energy and make better use of limited wireless bandwidth.

SGOU

3.3.4 Bandwidth Constraints

Bandwidth determines how quickly information can be transmitted over a link. Low-power IoT links may provide much less capacity than typical wired or Wi-Fi networks. When many devices share the same channel, available capacity must be divided among them.

Bandwidth affects both payload and overhead. Every byte consumes airtime. If a packet contains a small sensor reading plus a large amount of repeated protocol information, the effective application throughput can be poor.

Traffic bursts can be especially harmful. A network that handles periodic readings may struggle when thousands of devices transmit simultaneously after a scheduled event. This can produce congestion, collisions and increased packet loss.

Optimization can involve reducing packet size, reducing packet frequency, distributing transmissions over time and avoiding unnecessary control traffic. These techniques operate at different layers, so the architecture should coordinate them.

The important lesson is that bandwidth is not simply a measure of maximum speed. In a shared IoT network, it is a scarce resource whose use affects latency, reliability and energy consumption.

3.3.5 Energy Constraints

Energy is one of the defining constraints of many IoT systems. A sensor installed in a field or inside a structure may be expected to operate for months or years without battery replacement. Communication can consume a significant portion of its energy budget.

The radio is affected by both transmission and reception. A device may therefore benefit not only from sending shorter packets but also from reducing the number of times it must wake and listen.

Optimization can reduce energy use indirectly. Header compression reduces transmitted bytes. Efficient routing reduces unnecessary forwarding. Avoiding fragmentation reduces the number of frames that must be sent. Local processing can sometimes reduce the amount of data that travels over the network.

However, energy optimization must not sacrifice reliability blindly. A lost packet may require retransmission, which can consume more energy than the original transmission. The best strategy balances packet size, reliability and communication frequency.

A useful rule is: every radio operation has a cost. Good IoT protocols try to obtain the required communication outcome with as few costly operations as practical.

3.3.6 Memory and Processing Constraints

A constrained node may have only a small amount of RAM and limited processing capacity. Protocols that require large routing tables, extensive buffering or complex state can exceed those resources.

SGOU

Fragmentation illustrates the issue. A node that receives multiple fragments may need memory to store them until reassembly is complete. If several packets are being reassembled at once, memory demand can increase quickly.

Processing also consumes energy. Compression and decompression are beneficial when the communication savings exceed their computational cost. In many low-power systems this trade-off is favorable, but it should be considered rather than assumed.

IoT protocol design therefore aims for small state, predictable memory use and manageable computation. This is one reason specialized routing and adaptation mechanisms are important.

The broader principle is that optimization is multidimensional. A solution that minimizes bytes but requires excessive memory may not be optimal. Engineers must consider bandwidth, energy, memory, processing and reliability together.

3.3.7 MTU and Packet Size

The maximum transmission unit, or MTU, is the largest packet size that a link can carry without requiring fragmentation at that layer. IoT links may have much smaller frame capacities than ordinary IP networks.

When an IPv6 packet exceeds the available capacity, fragmentation may be required. The problem is not only the size of the original packet. Headers and adaptation information also consume space. Therefore, efficient header representation can reduce the likelihood that fragmentation is necessary.

Packet size also affects reliability. If a large packet is divided into several fragments, the complete message depends on successful delivery of all required pieces. A single loss may force retransmission or cause the packet to be discarded.

Applications can sometimes help by choosing suitable payload sizes. Transport and application protocols can also avoid generating unnecessarily large messages.

The practical lesson is simple: know the link MTU. A protocol stack that ignores the physical constraints of the underlying network will eventually pay for that mismatch through fragmentation, delay or loss.

Note : If a payload is tiny, which is likely to matter more: reducing one byte of payload or reducing repeated overhead across every packet? Explain your reasoning before reading on.

3.3.8 Latency and Responsiveness

Latency is the time between sending information and receiving the required result. IoT applications vary greatly in their tolerance for delay. A periodic temperature monitor may tolerate seconds, while an industrial control loop may require much faster response.

Optimization can influence latency in several ways. Smaller packets may require less airtime. Fewer retransmissions reduce waiting. Efficient routing can shorten paths.

SGOU

On the other hand, aggressive compression, sleep schedules or buffering can add processing or waiting time.

There is therefore no universal "minimum latency" configuration. Engineers must identify the application's response requirement and then choose networking mechanisms that satisfy it.

An important distinction is between average latency and worst-case behavior. A system may have a low average delay but occasional long delays caused by retransmissions or route repair. Safety-critical applications may care more about bounded behavior than about average performance.

The key idea is that optimization is goal-dependent. The best IoT network is not necessarily the one with the smallest packets; it is the one that meets application requirements within resource limits.

3.3.9 Packet Loss and Reliability

Wireless links can experience interference, fading, obstacles and temporary connectivity problems. Packet loss therefore has direct consequences for IoT performance.

Loss interacts strongly with fragmentation. If a packet is divided into five fragments and one fragment is lost, the receiver may not be able to reconstruct the original packet. The larger the number of fragments, the more opportunities there are for loss.

Retransmission can improve reliability but costs energy and time. A protocol should therefore avoid both extremes: transmitting unreliable data without protection and retransmitting so aggressively that the network becomes overloaded.

Routing also matters. A route with a slightly longer path but more reliable links may provide better overall performance than a short but unstable route.

The correct optimization target is often "use the fewest resources needed to achieve acceptable reliability." This requires understanding application priorities. A lost humidity reading may be harmless; a lost control command may not be.

3.3.10 Control Traffic

IoT networks must exchange control information to discover neighbors, maintain routes, synchronize state and recover from changes. Control traffic is essential, but excessive control traffic consumes the same bandwidth and energy resources as application data.

A low-power network therefore needs careful control-message behavior. Messages should be sent often enough to maintain useful state but not so often that they dominate traffic.

Routing protocols for low-power and lossy networks address this challenge by organizing control information and adapting to link conditions. Later, RPL will be discussed as a major example.

SGOU

Optimization can also involve aggregation. If several measurements can be represented by one message without losing required meaning, packet count can be reduced. This is an application-level decision rather than a purely network-layer mechanism.

The lesson is to treat control traffic as part of the system's cost. A network is not efficient merely because its application packets are small.

3.3.11 Reliability versus Efficiency

Efficiency and reliability can pull in different directions. Additional acknowledgements, retries and redundancy can improve delivery but consume resources. Aggressive compression and small packets can reduce cost but may increase the number of packets.

The right balance depends on the application. For environmental monitoring, occasional missing measurements may be acceptable if the next reading arrives soon. For an actuator command, stronger delivery assurance may be required.

This is why IoT optimization should begin with requirements. Ask what must be guaranteed, what can be approximate, and what can be delayed. Only then should protocol parameters be selected.

Layering helps because different mechanisms can address different needs. Routing can seek reliable paths; transport can provide an appropriate delivery model; the application can decide whether a measurement must be retransmitted.

Thus, optimization is a trade-off problem, not a race toward the smallest possible packet.

3.3.12 Security Overhead

Security is essential in IoT, but security mechanisms also consume resources. Authentication tags, encryption processing, key management messages and secure handshakes can increase packet size and energy use.

This does not justify removing security. Instead, it highlights the need for efficient security design. A secure system should protect the data and device without adding unnecessary communication.

Security overhead can also interact with fragmentation. If protected packets become larger, the probability of fragmentation may increase. Designers should therefore consider packet size after security mechanisms are applied, not before.

The right approach is integrated optimization. Security requirements should be treated as system requirements from the beginning, and network efficiency should be assessed with the required protections enabled.

A secure but unusable system is not a successful IoT system, and an efficient but insecure system is equally unacceptable. Good engineering finds a practical balance.

SGOU

3.3.13 Worked Analysis: Battery Sensor

Suppose a sensor sends a small measurement every minute. The device has a limited battery and uses a low-rate wireless link. Initially, the system sends one large packet for each reading, and occasional losses trigger retransmission.

The first optimization question is whether the packet contains information that can be compressed or inferred. If so, header compression can reduce airtime. The second question is whether the payload size causes fragmentation. If it does, reducing unnecessary overhead may eliminate some fragments. The third question concerns routing and control traffic: is the device receiving or forwarding more traffic than the application requires?

Finally, consider reliability. If the application can tolerate an occasional missing reading, aggressive retransmission may be unnecessary. If every reading is essential, stronger delivery mechanisms may be justified.

This analysis demonstrates a method rather than a single answer. Measure the traffic, identify the dominant costs, and select the smallest set of mechanisms that addresses those costs.

3.3.14 Check Your Progress and Summary

Optimization is needed because IoT networks operate under constraints that conventional networks may not face as strongly: low bandwidth, limited energy, small memory, low processing capacity, small MTUs, variable links and packet loss.

The most important sources of cost include headers, control traffic, fragmentation, retransmissions and security processing. Optimization techniques should be chosen according to measured bottlenecks. Header compression targets representation overhead; fragmentation handles packets that cannot fit; routing optimization reduces unnecessary forwarding and controls traffic.

Remember the trade-off rule: reducing one resource cost can increase another. Compression requires computation, reliability requires control traffic, and fragmentation increases state and loss sensitivity.

The next unit turns these principles into a practical toolbox. We will examine how 6LoWPAN, header compression, fragmentation, RPL and lightweight transport choices work together to optimize IP for IoT.

Recap

- Optimization is necessary because IoT resources are scarce and communication overhead can be large relative to useful payload.
- Header compression reduces the size of protocol headers by representing repeated or predictable information using fewer bits, thereby reducing communication overhead in constrained IoT networks.

SGOU

- 6LoWPAN supports IPv6 header compression, improving bandwidth utilisation, reducing energy consumption and minimising packet fragmentation in low-power wireless networks.
- Bandwidth, energy, memory, processing, MTU, latency and packet loss are interconnected constraints.
- Fragmentation can increase transmission cost and loss sensitivity.
- Control traffic and retransmissions are part of total network cost.
- Security is essential and its overhead must be included in performance reasoning.
- Optimization should begin with requirements and measurement, not with a favorite protocol.

Objective Type Questions

1. What is the main purpose of optimizing IP for IoT?
2. Which resource is especially important in battery-powered sensors?
3. What does MTU specify?
4. Which IoT adaptation technology provides IPv6 header compression for low-power wireless networks?
5. Which type of information is used by the receiver to reconstruct a compressed protocol header?
6. What happens when a packet is larger than the link can carry?
7. Which traffic is necessary for route maintenance but also consumes resources?
8. What event can cause a packet to be transmitted again?
9. Which constraint concerns the amount of RAM available to a node?
10. Why can fragmentation increase reliability problems?
11. Which performance measure describes delay?
12. What should optimization begin with: measurement or protocol preference?

Answers

1. Resource efficiency
2. Energy
3. Maximum Transmission Unit

SGOU

4. 6LoWPAN
5. Context
6. Fragmentation
7. Control traffic
8. Retransmission
9. Memory
10. Fragment loss
11. Latency
12. Measurement

Assignments

1. Explain why optimization of IP is necessary for IoT.
2. Discuss bandwidth and energy constraints in low-power networks.
3. Explain the effect of MTU on packet transmission.
4. Discuss how fragmentation affects efficiency and reliability.
5. Explain the role of control traffic in IoT network cost.
6. Analyze the trade-off between reliability and energy efficiency.
7. Discuss how security mechanisms influence packet size and processing cost.
8. Develop a checklist for identifying the dominant performance constraints in an IoT deployment.

SGOU

Suggested Reading

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Olsson, Jerry. IPv6 Essentials. O'Reilly Media.
3. Kurose, James F., and Keith W. Ross. Computer Networking: A Top-Down Approach. Pearson.
4. Forouzan, Behrouz A. Data Communications and Networking. McGraw-Hill.

5. Shelby, Zach, and Carsten Bormann. 6LoWPAN: The Wireless Embedded Internet. Wiley.

Reference

1. Shelby, Zach, et al. The Constrained Application Protocol (CoAP). RFC 7252, Internet Engineering Task Force, 2014.
2. Winter, Tim, et al. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, Internet Engineering Task Force, 2012.
3. Thubert, Pascal. 6LoWPAN Routing Header. RFC 8138, Internet Engineering Task Force, 2017.
4. Kushalnagar, Nandakishore, et al. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, IETF, 2007.

SGOU



Optimizing IP For IoT

Learning Outcomes

After completion of this unit, learner will be able to:

- explain major mechanisms used to make IP practical for constrained IoT networks
- describe IPv6 header compression and fragmentation in low-power links
- explain the purpose of RPL in low-power and lossy networks
- recognize how transport, routing and adaptation choices work together in an optimized IoT stack

Prerequisite

You have already seen that IP provides a common foundation and that constrained networks introduce special limitations. Now imagine an engineer assembling a real IoT network: the devices need addresses, packets must cross a low-power wireless link, routes must adapt to changing conditions, and applications need dependable communication without exhausting batteries. No single technique solves all of these problems. Instead, optimization comes from choosing a set of complementary mechanisms that each remove a particular obstacle. This unit brings those mechanisms together and shows how they cooperate.

Keywords

header compression, RPL, UDP, fragmentation, low-power and lossy network, routing, optimization

SGOU

Discussion

3.4.1 Optimization as a Layered Strategy

Optimizing IP for IoT is best understood as a coordinated set of mechanisms rather than one algorithm. A constrained network may need compact headers, efficient fragmentation, specialized routing and a transport protocol that matches the application.

At the adaptation layer, 6LoWPAN can compress IPv6 headers and handle fragmentation for low-power IEEE 802.15.4 networks. At the routing layer, RPL is designed for low-power and lossy networks. At the transport layer, UDP can provide a lightweight datagram service when the application does not require TCP's connection-oriented behavior.

These mechanisms solve different problems. Compression reduces bytes. Fragmentation resolves size mismatch. Routing establishes paths. Transport provides end-to-end service semantics. The stack works well when each layer performs its intended function without duplicating unnecessary work.

Optimization also requires measurement. A mechanism should be selected because it addresses a real constraint, not because it is fashionable. The objective is an efficient, interoperable and maintainable network.

3.4.2 IPv6 Header Compression

IPv6 headers contain fields needed for addressing, forwarding and protocol processing. On a constrained link, transmitting all of those fields repeatedly can consume substantial airtime. Header compression reduces this cost by exploiting information that can be inferred from context.

For example, neighboring devices may share address prefixes, and some header fields may remain constant across a sequence of packets. Instead of transmitting the full value every time, a compressed representation can indicate that the receiver should reconstruct it from known context.

The important point is that compression is lossless with respect to the protocol meaning. The receiver must be able to recover the necessary header information correctly.

Compression is especially useful for small sensor payloads. If a measurement contains only a few bytes, saving several header bytes can materially improve efficiency.

However, compression depends on shared context. If context changes frequently or cannot be inferred safely, fewer fields can be compressed. Good implementations therefore adapt the compression level to the communication state.

SGOU

Table 3.4.1 Optimization mechanisms and the problems they address.

Mechanism	Main problem addressed	Typical benefit
Header compression	Repeated header information	Fewer transmitted bytes
Fragmentation	Packet larger than link frame	Allows transmission
RPL	Low-power/lossy routing	Structured reachability
UDP	Unnecessary transport state	Lightweight datagrams

3.4.3 6LoWPAN Adaptation Layer

The 6LoWPAN adaptation layer sits between IPv6 and an IEEE 802.15.4-based low-power link. Its purpose is to make IPv6 communication practical on a link with small frames and constrained resources.

Three ideas are particularly important: header compression, fragmentation and frame adaptation. Compression reduces the amount of information that must be transmitted. Fragmentation divides packets that exceed the available frame capacity. Together, these mechanisms bridge the mismatch between a general IP packet format and a constrained link.

This is a strong example of layered architecture. IPv6 remains the network-layer protocol, while the adaptation layer handles link-specific constraints. Applications above the network layer do not need to understand every detail of the radio.

The adaptation approach also supports interoperability. A constrained device can participate in an IP-based architecture without requiring a proprietary network-layer protocol.

For learners, the key phrase is “preserve the IP model, adapt the representation.” That phrase captures the purpose of 6LoWPAN.

3.4.4 Fragmentation in Practice

When an IPv6 packet cannot fit into the underlying link frame, fragmentation allows it to be transmitted in smaller pieces. The adaptation layer divides the packet and includes information needed for reassembly.

Fragmentation should be used carefully. More fragments mean more transmissions and more opportunities for loss. Reassembly also requires memory and state at the receiving side. In a low-power network, these costs can be significant.

SGOU

This is why header compression is complementary to fragmentation. If compression reduces the packet enough to fit into a single frame, fragmentation can be avoided. If the packet remains too large, fragmentation provides a way to complete the transfer.

A useful engineering question is therefore: “Can I reduce the packet before I divide it?” This simple question often leads to better resource use.

Fragmentation also highlights why packet-loss characteristics matter. If one fragment is lost, the receiver may have to discard or recover the entire packet depending on the mechanisms available. Thus, packet size and reliability are tightly connected.

3.4.5 RPL for Low-Power and Lossy Networks

RPL, the IPv6 Routing Protocol for Low-Power and Lossy Networks, was designed for environments where links can be unreliable and devices have limited resources. Its design reflects the fact that routing in such networks differs from routing in stable wired networks.

RPL organizes nodes into a directed acyclic graph rooted at a destination or collection point. Nodes select preferred parents according to routing objectives and link information. This structure supports upward traffic toward a root and can also support downward communication using appropriate mechanisms.

Routing decisions can consider metrics such as link quality, hop count or energy-related objectives. The exact objective depends on the network's requirements.

RPL is important because it demonstrates that using IP does not mean using exactly the same routing assumptions as a conventional enterprise network. The IP layer remains common, but the routing protocol is adapted to the environment.

The broader lesson is that optimization often occurs through specialized protocols that understand the constraints of the target network.

> RPL is not simply a smaller version of a conventional routing protocol. Its design reflects the instability, resource limits and traffic patterns of low-power and lossy networks.

3.4.6 RPL Concepts: DODAG, Rank and Parent

A RPL DODAG is a Destination-Oriented Directed Acyclic Graph. The structure provides a way for nodes to organize routes toward a root.

Rank represents a node's position relative to the root according to the routing objective. It is not simply a universal physical distance; its meaning depends on the objective function and protocol rules. Nodes select preferred parents that provide suitable routes toward the root.

SGOU

This organization helps avoid arbitrary route formation in a low-power network. Instead, nodes maintain relationships within a structured topology.

The topology can change. Wireless conditions vary, devices may fail, and energy conditions may shift. RPL therefore includes mechanisms for maintaining and repairing the routing structure.

For learners, imagine a hill with a base station at the bottom. Each node chooses a neighboring point that offers a suitable path toward the base. The analogy is imperfect, but it helps visualize why rank and parent selection matter.

3.4.7 UDP and Lightweight Transport

UDP provides a connectionless datagram transport service with less protocol state than TCP. This can be useful in IoT applications where messages are small and the application can tolerate occasional loss or implement reliability in another way.

UDP does not provide the same reliability, ordering and congestion-control behavior as TCP. Therefore, choosing UDP is not simply choosing "faster communication." The application must provide any additional reliability or ordering mechanisms it actually needs.

For constrained IoT, the lower state and smaller transport overhead can be valuable. CoAP, for example, is an application-layer protocol commonly associated with constrained devices and can operate over UDP.

The correct design question is: what service does the application require? If it needs a full reliable byte stream, TCP may be appropriate. If it exchanges independent request-response datagrams and can use application-level mechanisms, UDP may be more suitable.

Transport selection is therefore another form of optimization.

3.4.8 Application-Layer Efficiency

Network optimization does not stop at the IP layer. Applications can reduce communication cost by sending only useful information, using compact representations and avoiding unnecessary polling.

For example, a sensor does not always need to report every measurement if the value has not changed significantly and the application can tolerate event-based reporting. Local aggregation can also combine several readings before transmission.

However, application optimization should preserve semantics. Compressing data so aggressively that the receiver cannot interpret it reliably is not useful. Likewise, reducing reporting frequency may be unacceptable for a safety-critical application.

This demonstrates a central principle of layered optimization: each layer should contribute where it has the best information. The application knows what information matters; the network knows how to route it; the adaptation layer knows the link constraints.

SGOU

The strongest IoT systems coordinate these choices rather than optimizing each layer in isolation.

3.4.9 Integrated Example: Smart Building

Consider a smart building with battery-powered occupancy sensors, lighting controllers and an edge gateway. Occupancy sensors use a low-power wireless network and send small messages when room state changes.

6LoWPAN can adapt IPv6 communication to the constrained link. Header compression reduces repeated information. If a packet still exceeds the frame size, fragmentation can be used. RPL can maintain routes through the building toward the gateway, while UDP can support lightweight application exchanges where appropriate.

The gateway connects the low-power network to a higher-capacity IP network. From the building's perspective, the edge network and enterprise network form a layered system rather than two unrelated islands.

Now imagine a wireless interference problem on one floor. RPL may select alternative parents, while application traffic continues through the network. This illustrates the benefit of separating routing adaptation from application logic.

The scenario shows that optimization is a stack-level activity. No single mechanism is responsible for all performance improvements.

3.4.10 Comparing Optimization Mechanisms

It is helpful to compare mechanisms by the problem they address. Header compression primarily reduces packet representation overhead. Fragmentation addresses packet-size limitations. RPL addresses routing in low-power and lossy environments. UDP can reduce transport state when the application does not require TCP's service model.

These mechanisms are complementary, not competing replacements. A network may use all of them because each operates at a different layer.

The choice of mechanism should also consider failure modes. Compression depends on context, fragmentation increases loss sensitivity, routing requires topology maintenance, and UDP does not itself provide reliable delivery.

A good engineering table therefore asks four questions: What constraint is being addressed? What resource is saved? What new complexity is introduced? What application requirement could be affected?

Thinking in this structured way helps avoid selecting a mechanism simply because it is widely known.

3.4.11 Optimization and Security

Optimization must be performed without weakening security. An IoT network may use encryption and authentication at different layers depending on the architecture.

SGOU

Security mechanisms can increase packet size and processing cost, so the optimized design should be evaluated with security enabled.

Routing security is also important because a compromised node could attempt to disrupt route formation. Network access control, device authentication and secure updates are part of a complete IoT security strategy.

Compression itself does not provide security. A compressed packet may still require integrity protection and confidentiality. Similarly, fragmentation does not guarantee that fragments are authentic.

Therefore, the correct sequence is not "optimize first, secure later." Security and efficiency should be considered together so that the final packet format, processing load and communication pattern are realistic.

3.4.12 Design Reasoning Checklist

When examining an IoT network, begin with the application. What information is sent? How often? How quickly must it arrive? What happens if a message is lost?

Next examine the device. How much memory, processing capacity and battery energy are available? Can it support IPv6 directly? Does it need an adaptation layer?

Then examine the link. What is the MTU? What is the data rate? How variable is link quality? Is the channel shared by many devices?

Finally examine the network architecture. Which routing protocol fits the environment? Is a gateway required? Which transport service is appropriate? How will security be applied?

This checklist converts the abstract idea of optimization into an engineering workflow. It encourages you to select mechanisms based on constraints and requirements rather than on protocol names alone.

The major optimization mechanisms can now be connected. 6LoWPAN provides an adaptation layer for IPv6 over low-power IEEE 802.15.4 networks. Header compression reduces repeated network information. Fragmentation allows packets larger than the link's frame capacity to be transmitted. RPL provides routing designed for low-power and lossy networks. UDP can offer a lightweight transport service when the application does not need TCP's full service model.

The key idea is cooperation across layers. Adaptation handles link constraints, routing handles topology and reachability, transport handles end-to-end service, and the application determines what information is worth sending.

Optimization should always be judged against requirements. A smaller packet is useful only if the application still receives the required information securely and reliably. A route with fewer hops is useful only if it remains sufficiently stable. A lightweight transport is useful only if its service model matches the application.

With these principles, you can reason about an IoT stack rather than memorizing isolated protocol names.

SGOU

Recap

- Optimization works across layers rather than through one mechanism.
- 6LoWPAN adapts IPv6 to constrained low-power links.
- Header compression reduces repeated header information.
- Fragmentation permits oversized packets to cross small-frame links.
- RPL provides routing designed for low-power and lossy networks.
- UDP can provide lightweight datagram transport when its service model fits the application.
- Security and optimization should be considered together.

Objective Type Questions

1. Which adaptation technology supports IPv6 over IEEE 802.15.4 low-power links?
2. What mechanism reduces repeated IPv6 header information?
3. What mechanism handles packets that exceed link frame capacity?
4. Which routing protocol is designed for low-power and lossy networks?
5. What does RPL organize for routing toward a root?
6. Which transport protocol provides lightweight datagram service?
7. What layer decides what information is actually worth sending?
8. What must be considered together with optimization to protect IoT systems?
9. What is the main purpose of a gateway in a heterogeneous IoT architecture?
10. Which mechanism primarily addresses routing rather than packet representation?

SGOU

Answers

1. 6LoWPAN
2. Header compression
3. Fragmentation
4. RPL

5. DODAG
6. UDP
7. Application
8. Security
9. Interconnection
10. RPL

Assignments

1. Explain optimization as a layered strategy for IoT.
2. Describe IPv6 header compression and its benefits.
3. Explain the role of 6LoWPAN in optimizing IPv6 over low-power links.
4. Discuss fragmentation and its effect on reliability.
5. Explain the purpose and basic concepts of RPL.
6. Discuss the role of UDP in constrained IoT applications.
7. Analyze an integrated smart-building networking scenario.
8. Compare header compression, fragmentation, RPL and UDP in terms of the problems they address.

SGOU

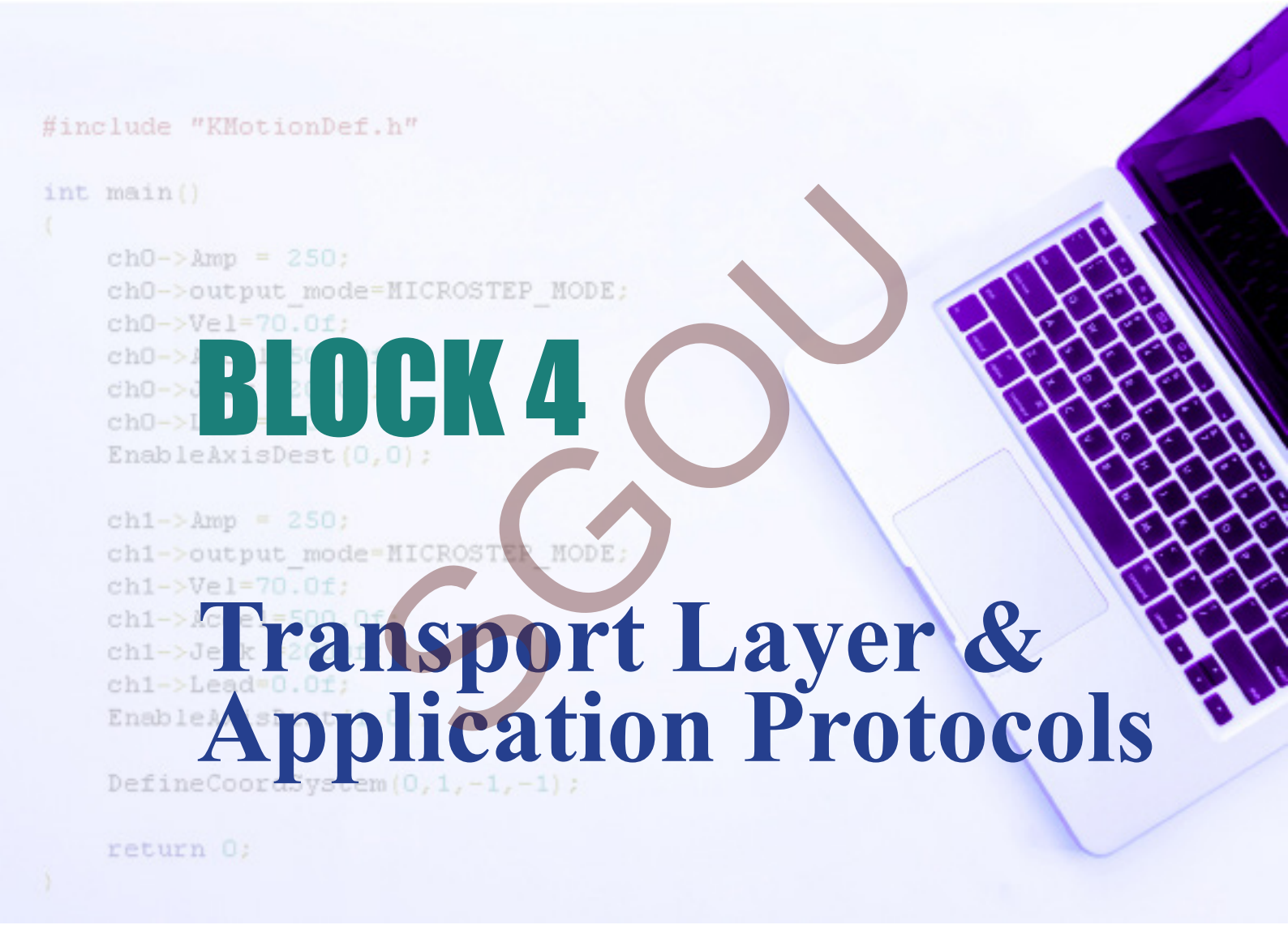
Suggested Reading

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Olsson, Jerry. IPv6 Essentials. O'Reilly Media.
3. Kurose, James F., and Keith W. Ross. Computer Networking: A Top-Down Approach. Pearson.
4. Forouzan, Behrouz A. Data Communications and Networking. McGraw-Hill.
5. Shelby, Zach, and Carsten Bormann. 6LoWPAN: The Wireless Embedded Internet. Wiley.

Reference

1. Shelby, Zach, et al. The Constrained Application Protocol (CoAP). RFC 7252, Internet Engineering Task Force, 2014.
2. Winter, Tim, et al. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, Internet Engineering Task Force, 2012.
3. Thubert, Pascal. 6LoWPAN Routing Header. RFC 8138, Internet Engineering Task Force, 2017.
4. Kushalnagar, Nandakishore, et al. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, IETF, 2007.

SGOU



BLOCK 4

Transport Layer & Application Protocols

SGOU



Transport Layer

Learning Outcomes

After completion of this unit, learner will be able to:

- explain the role of the transport layer in end-to-end communication for IoT systems
- identify the main characteristics of TCP and UDP and distinguish their typical IoT uses
- describe ports, sockets, segmentation, flow control, congestion control, and reliability in transport communication
- recognize the importance of transport choices for constrained devices, gateways, and cloud-connected IoT applications
- explain common transport-layer considerations for secure and efficient IoT communication

Prerequisite

Imagine a temperature sensor sending a reading to a cloud service. The sensor has collected the value, the network knows where the destination is, and yet another question remains: how should the data be delivered from one application process to another? What happens if a packet is lost? What if several applications are using the same device? What if the network becomes crowded? These are the kinds of practical questions that make the transport layer important. You already know that IoT devices communicate through networks and that IP addresses help identify end-points. In this unit, we move one step upward and examine how communication is organized between applications. You will see why a lightweight sensor may prefer a different transport service from a banking application, and why reliability is not always synonymous with better IoT performance.

Keywords

Transport Layer, TCP, UDP, Port Number, Socket, Segmentation, Flow Control, Congestion Control

SGOU

Discussion

4.1.1 End-to-End Communication

The transport layer sits between application processes and the underlying network service. Its central concern is not simply moving a packet from one address to another; it is enabling one application process to communicate with another application process across an internetwork. In an IoT system, a sensor application may generate a measurement, a gateway may forward it, and a cloud application may consume it. The transport layer provides mechanisms that help these processes exchange data in an organized manner.

A useful way to picture the relationship is to imagine a postal system. The IP layer is concerned with getting an envelope to the correct building, while the transport layer helps ensure that the envelope reaches the intended room or service inside that building. In networking terms, the room is associated with a port number. This distinction becomes particularly useful on IoT gateways, where many applications may share a single network interface and IP address.

Transport services can differ in reliability, ordering, connection behavior, and overhead. The Internet architecture therefore does not force every application to use exactly the same service. Instead, applications select a transport protocol whose characteristics match their communication needs. For example, a small monitoring message may tolerate an occasional lost update, whereas a firmware image normally requires complete and ordered delivery.

The important lesson is that transport is an application-facing communication service. Good IoT engineering begins by asking what the application actually needs rather than assuming that one transport protocol is always appropriate.

4.1.2 Ports, Sockets and Multiplexing

An IP address identifies a network interface, but an IoT device may run several communication services at the same time. A gateway, for instance, might receive sensor data, provide a management service, and maintain a diagnostic connection. Port numbers allow the transport layer to distinguish these services. A transport endpoint is commonly described using an IP address together with a port number and transport protocol.

The term socket is often used for the software endpoint through which an application communicates. A client application can open a socket, specify a destination, and exchange data through the selected transport service. At the receiving side, the operating system examines the destination port and delivers the data to the appropriate application process. This process is called demultiplexing; the reverse operation at the sender is multiplexing.

Consider an IoT gateway with one Ethernet interface and one IP address. A telemetry service may listen on one port, an administrative service on another, and a local diagnostic application on a third. The network does not need three separate IP addresses simply

SGOU

because three services are active. Transport ports provide the additional distinction required by applications.

For learners, it is useful to separate three ideas: address answers 'which network endpoint?', port answers 'which service?', and socket answers 'which software communication endpoint?'. Together they make concurrent network applications practical.

4.1.3 Transport Segmentation and Reassembly

Applications often generate data in units that are larger than a single network packet. The transport layer can divide application data into smaller pieces suitable for transmission. This process is commonly called segmentation. At the destination, the pieces can be reassembled into the original application data when the selected transport protocol provides the required service.

Segmentation is important because networks have limits on packet size. Sending a very large application message as one indivisible unit could create inefficiency or require lower layers to fragment it. Transport protocols can attach their own control information so that the receiving side can understand how the transmitted data belongs together.

In a reliable, connection-oriented protocol such as TCP, sequence numbers help the receiver identify the position of bytes in the overall data stream. If segments arrive out of order, the protocol can use the sequence information to organize the stream. If data is missing, retransmission mechanisms can request or trigger recovery. In contrast, UDP preserves message boundaries at the transport interface and does not itself provide reliable stream reconstruction.

For constrained IoT devices, the cost of transport headers, buffering, retransmission, and memory usage matters. A sensor transmitting a short reading every minute has a very different data pattern from an industrial gateway transferring a large configuration file. Thus, segmentation and reassembly should be understood not only as technical mechanisms but also as factors affecting energy, memory, latency, and network utilization.

4.1.4 TCP: Connection-Oriented Transport

Transmission Control Protocol (TCP) provides a connection-oriented, reliable byte-stream service. Before application data is exchanged, TCP establishes a logical connection between the endpoints. During the connection, TCP uses sequence numbers, acknowledgements, retransmission and other mechanisms to provide ordered delivery and recovery from loss.

The connection establishment process is commonly described using a three-way handshake. The initiating endpoint sends a synchronization request, the receiving endpoint acknowledges it and sends its own synchronization information, and the initiator acknowledges the response. This exchange establishes the shared state needed

SGOU

for the connection. The exact packet fields are less important at first than the concept: both endpoints agree on a transport session before normal data transfer proceeds.

TCP's reliability is useful when losing or reordering data would make the application result incorrect. Examples include transferring an IoT device firmware image, uploading a configuration database, or exchanging a control file where every byte matters. However, reliability introduces state, acknowledgements and possible retransmissions. On a low-power sensor with tiny messages and intermittent connectivity, this additional work may not be desirable.

TCP should therefore be viewed as a service with a cost-benefit profile. It is not simply 'better' than UDP. Its strengths become valuable when ordered, complete delivery is important, while its overhead may be less attractive when the application can tolerate loss or already provides its own recovery.

Note : In many IoT systems, the protocol that looks simplest at the application level may still require careful choices about reliability, security, buffering and failure recovery.

4.1.5 TCP Reliability and Flow Control

Reliability in TCP is achieved through several cooperating mechanisms. Sequence numbers identify positions in the byte stream, acknowledgements inform the sender about received data, and retransmission allows lost data to be sent again. A receiver also advertises how much additional data it can accept, which supports flow control.

Flow control protects a receiver from being overwhelmed by a sender that can transmit faster than the receiver can process. The receiver communicates an advertised window, and the sender limits outstanding data accordingly. This is particularly relevant when an IoT gateway communicates with a small embedded device or when a cloud service receives data from devices with limited memory.

TCP also incorporates congestion control. Flow control is about the receiver's capacity; congestion control is about the condition of the network. When the network appears congested, TCP adjusts its sending behavior. Mechanisms such as slow start and congestion avoidance are intended to prevent persistent overload and to share network capacity more responsibly.

The distinction is worth remembering. Flow control protects the communicating endpoints, whereas congestion control protects the network from being overwhelmed by excessive traffic. Together with acknowledgements and retransmissions, these mechanisms make TCP reliable and robust for many applications.

However, these same mechanisms also cause TCP performance to vary when network latency, packet loss, or connectivity conditions change. Such variations are common in wireless IoT environments, where communication links may be less stable and more unpredictable than traditional wired networks.

SGOU

4.1.6 UDP: Lightweight Datagram Transport

User Datagram Protocol (UDP) provides a connectionless datagram service with minimal transport-layer overhead. It does not establish a connection before sending data, and it does not provide TCP-style guarantees of delivery, ordering, or retransmission. Each UDP datagram is handled as a separate message at the transport interface.

This simplicity can be valuable in IoT. A device may periodically report a measurement where the newest value is more useful than an old value waiting for retransmission. A monitoring application might prefer low overhead and low latency, especially when occasional loss is acceptable. UDP can also support one-to-many communication patterns through mechanisms available at the IP layer.

The absence of reliability does not mean that UDP is unreliable in every application. It means the transport protocol does not supply those guarantees. If an application requires acknowledgements, sequence information, duplicate detection, or retransmission, it can implement suitable mechanisms above UDP. This approach allows the application designer to tailor reliability to the actual data semantics.

A practical question for learners is: "Which is more problematic: losing a single sensor reading or delaying all readings while attempting to recover a lost packet?" There is no single answer that applies to every situation. The appropriate choice depends on the specific application and its requirements.

For this reason, the selection of a transport protocol in an IoT system should be based on factors such as the importance of the data, timing requirements, available energy resources, and the characteristics of the network.

4.1.7 TCP and UDP: A Practical Comparison

TCP and UDP represent two different transport service philosophies. TCP emphasizes reliable, ordered byte-stream delivery with connection state and congestion/flow control. UDP emphasizes a simple datagram service with low protocol overhead and no built-in reliability guarantee.

For an IoT application, the choice should be tied to the meaning of the data. Suppose a smart meter periodically reports current consumption. If a later reading supersedes an earlier one, the application may tolerate occasional loss. A lightweight datagram service can therefore be attractive. Now consider a gateway receiving a firmware package. Missing bytes cannot simply be ignored, so a reliable transport becomes useful. Table 4.1.1 summarizes the main differences.

Table 4.1.1 Comparison of TCP and UDP

Feature	TCP	UDP
Connection	Connection-oriented	Connectionless
Delivery guarantee	Reliable, ordered byte stream	No built-in delivery guarantee

SGOU

Retransmission	Supported	Not provided by UDP itself
Flow/congestion control	Supported	Not provided by UDP itself
Overhead	Higher	Lower
Typical IoT fit	Reliable transfers and control/data streams	Lightweight or loss-tolerant exchanges

4.1.8 Transport Choices in Constrained IoT Networks

IoT networks often contain constrained nodes with limited processor speed, memory, battery capacity and bandwidth. These constraints change how transport mechanisms should be assessed. A protocol that is inexpensive on a desktop may impose noticeable energy or memory costs on a tiny sensor.

Latency is another important factor. Interactive control applications may need timely communication, while environmental monitoring can often tolerate delay. Reliability is similarly application-dependent. A single lost temperature sample may have little effect on a long-term trend, but a lost command to change an actuator state may be significant.

Gateways can reduce constraints by performing protocol translation, buffering, aggregation or security functions. A gateway may collect many sensor readings and send summarized information to a cloud service. This can reduce the number of external transmissions and allow small devices to remain simple. However, the gateway becomes a critical point whose buffering and connection-management policies must be carefully designed.

When choosing a transport service, learners should ask four practical questions: How important is every message? How quickly must it arrive? How much device and network overhead can be tolerated? Where, if necessary, will reliability be implemented? These questions lead to more defensible design choices than choosing a protocol merely because it is familiar.

Note: Do not memorize a protocol name without understanding its communication model. Ask what is communicating, who consumes the data, how frequently it changes, and what happens when communication fails.

4.1.9 Transport Performance: Latency, Throughput and Overhead

Transport performance is multidimensional. Latency describes how long data takes to travel through the system; throughput describes the amount of data delivered per unit time; overhead represents the additional bytes and processing associated with communication. IoT applications may prioritize these measures differently.

SGOU

A battery-powered sensor sending 20 bytes every few minutes has a very different performance target from a video-enabled surveillance device. For the first, energy and overhead may dominate. For the second, sustained throughput and predictable delay may matter more. Reliability mechanisms can improve correctness while also increasing traffic and delay when retransmissions occur.

A simple performance analysis can compare an application payload with the total transmitted bytes. If a 20-byte reading requires several layers of headers, the protocol overhead is proportionally significant. Aggregating several measurements into one application message can sometimes improve efficiency, although aggregation may increase waiting time. The correct choice depends on the application's timing requirements.

Performance should also be measured under realistic network conditions. Wireless links can experience variable signal quality, interference and temporary disconnection. A transport strategy that performs well in a laboratory may behave differently in a crowded deployment. Therefore, transport performance is best understood as a relationship among application needs, protocol behavior and actual network conditions.

4.1.10 Reliability versus Efficiency

Reliability and efficiency are two important but sometimes competing communication requirements.

A highly reliable communication mechanism may introduce additional overhead because it may require acknowledgements, retransmissions, ordering and connection management. These mechanisms improve delivery assurance but may increase delay, bandwidth consumption and energy usage.

An efficient lightweight mechanism may reduce overhead and latency but may provide fewer delivery guarantees.

Table 4.1.2 Reliability versus Efficiency

Aspect	Higher Reliability	Higher Efficiency
Delivery guarantee	Stronger	Usually weaker
Overhead	Higher	Lower
Delay	May increase	Generally lower
Retransmission	Supported	Usually handled by application if needed
Energy consumption	May be higher	Can be lower
Suitable for	Critical data, files, configuration	Periodic sensor data, time-sensitive updates
Example	TCP	UDP

SGOU

4.1.11 Choosing TCP or UDP in IoT

Transport selection should consider:

Reliability : Does every message have to reach the destination?

Latency : Can the application tolerate delay?

Bandwidth : Is network capacity limited?

Energy : Is the device battery-powered?

Message size : Are the messages small or large?

Loss tolerance : Can an application tolerate losing an occasional message?

For example, a smart irrigation sensor that reports soil moisture every minute may tolerate occasional loss because the next reading will provide updated information. On the other hand, a firmware update requires complete and ordered delivery.

Therefore, **there is no universally best transport protocol**. The appropriate protocol depends on the communication requirements of the application.

Recap

- The transport layer provides process-to-process communication services above IP.
- Port numbers identify application services, while sockets represent software communication endpoints.
- TCP provides a reliable, ordered byte-stream service using sequencing, acknowledgements and retransmission.
- TCP flow control protects the receiver, while congestion control responds to network conditions.
- UDP provides a lightweight datagram service without TCP-style delivery guarantees.
- IoT transport choices depend on reliability, latency, energy, memory, payload size and network conditions.
- TLS and DTLS can protect transport-oriented communication, but secure deployment also requires authentication and authorization.
- A single IoT system may legitimately use different transport strategies for different data flows.

SGOU

Objective Type Questions

1. Which layer provides process-to-process delivery?
2. What identifies an application service at a host?
3. Which transport protocol provides a reliable byte stream?
4. What is the purpose of a TCP acknowledgement?
5. Which protocol provides a lightweight datagram service?
6. What does flow control protect?
7. What does congestion control respond to?
8. Which security protocol is commonly associated with TCP-based protected communication?
9. Why might a periodic sensor tolerate occasional loss?
10. What is a socket used for?

Answers

1. Transport layer
2. Port number
3. TCP
4. Confirms received data
5. UDP
6. Receiver capacity
7. Network congestion
8. TLS
9. A later reading can replace an earlier reading
10. Application communication endpoint

SGOU

Assignments

1. Explain the role of the transport layer in an IoT architecture with a suitable communication scenario.
2. Compare TCP and UDP in terms of connection behavior, reliability, overhead and IoT suitability.
3. Describe TCP connection establishment, sequencing, acknowledgements and retransmission.
4. Explain the difference between flow control and congestion control with an IoT example.
5. Discuss how ports, sockets and multiplexing support multiple IoT services on one gateway.
6. A smart irrigation sensor sends soil-moisture readings every minute. Recommend a transport strategy and justify your reasoning.
7. Discuss transport-layer security considerations for constrained IoT devices.
8. Prepare a comparison table showing how reliability, latency, energy and payload size influence transport selection.

Suggested Reading

1. Hanes, David, et al. IoT Fundamentals. Cisco Press, 2017.
2. Kurose, James F., and Keith W. Ross. Computer Networking: A Top-Down Approach. Pearson, 2021.
3. Forouzan, Behrouz A. Data Communications and Networking. McGraw-Hill, 2013.
4. RFC 9293: Transmission Control Protocol. Internet Engineering Task Force.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Kurose, James F., and Keith W. Ross. Computer Networking: A Top-Down Approach. 8th ed., Pearson, 2021.
3. Forouzan, Behrouz A. Data Communications and Networking. 5th ed., McGraw-Hill, 2013.
4. Tanenbaum, Andrew S., and David J. Wetherall. Computer Networks. 5th ed., Pearson, 2011.

SGOU



Supervisory Control and Data Acquisition (SCADA)

Learning Outcomes

After completion of this unit, learner will be able to:

- explain the purpose and major components of SCADA systems
- identify the roles of sensors, actuators, RTUs, PLCs, HMIs, historians, and supervisory servers
- describe SCADA communication, monitoring, alarming, data acquisition, and control functions
- recognize the relationship between SCADA and modern IoT architectures
- explain basic security, reliability, and operational considerations in connected SCADA environments

SGOU

Prerequisite

Think about a water-treatment plant spread across a large area. Pumps, valves, tanks and chemical measurements must be observed continuously, yet an operator cannot stand beside every device at every moment. A control room therefore needs a way to collect measurements, display conditions, raise alarms and send commands to remote equipment. This is the setting in which SCADA becomes meaningful. You may already have seen dashboards that show temperature, pressure, motor status or tank level. In an industrial environment, such information is not merely for viewing; it can support decisions that affect physical processes. In this unit, you will connect familiar ideas about sensors, networks and control with the structured world of industrial supervisory systems.

Keywords

SCADA, RTU, PLC, HMI, Historian, Telemetry, Alarm Management, Industrial Control

Discussion

4.2.1 Introduction to SCADA

Supervisory Control and Data Acquisition, commonly abbreviated as SCADA, refers to systems used to monitor and supervise industrial or infrastructure processes. The phrase contains two important ideas. 'Supervisory control' means that operators and supervisory software observe processes and may issue high-level commands. 'Data acquisition' means collecting measurements and status information from remote equipment.

SCADA is commonly associated with sectors such as electrical power, water distribution, manufacturing, oil and gas, transportation and building infrastructure. The physical process is represented by measurements such as temperature, pressure, flow, voltage, level and equipment status. Operators see these values through a human-machine interface (HMI), while specialized devices at the field level connect sensors and actuators to the control system.

A SCADA system is not simply a collection of sensors. It is an organized architecture that combines field equipment, communication networks, computing systems, visualization and operational logic. The architecture must also cope with long distances, communication failures, timing requirements and the need for safe operation.

Modern IoT technologies increasingly interact with SCADA environments, but the two should not be treated as identical. SCADA is strongly tied to monitoring and control of physical processes, whereas IoT is a broader networking and computing paradigm. Their integration can provide additional analytics and remote services while preserving the operational requirements of industrial control.

4.2.2 Functions of SCADA

SCADA systems perform several important functions in industrial and infrastructure environments.

1. Data Acquisition

SCADA collects information from sensors, meters, PLCs, RTUs and other field devices. The collected information may include temperature, pressure, flow rate, voltage, current, tank level and equipment status.

2. Monitoring

The collected information is presented to operators through an HMI or supervisory interface. Operators can observe the current condition of the process.

3. Supervisory Control

SCADA allows authorized operators to issue supervisory commands. For example, an operator may start or stop a pump or change an operating parameter.

SGOU

4. Alarm Management

The system generates alarms when important process variables move outside configured limits. This helps operators identify abnormal conditions.

5. Data Logging and Historical Storage

SCADA systems can store time-stamped process information. Historical data is useful for troubleshooting, performance analysis and maintenance.

6. Trending

Trend displays show how process variables change over time. A trend may reveal whether temperature, pressure or flow is increasing, decreasing or remaining stable.

7. Reporting

SCADA data can be used to generate operational reports, production reports, maintenance information and performance summaries.

4.2.3 SCADA Architecture

A typical SCADA architecture can be viewed as a hierarchy. At the field level are sensors and actuators. Local control equipment such as programmable logic controllers (PLCs) or remote terminal units (RTUs) collects signals and executes control logic. Communication networks connect these devices to supervisory servers. Operators interact through HMIs, while historians store process data for later analysis.

The hierarchy is useful because it separates time-critical local control from supervisory functions. A PLC can execute a control loop close to the machine even if a higher-level server is temporarily unavailable. The SCADA layer can then provide broader visibility, alarm management, historical trends and operator commands.

A simplified information flow is: physical process → sensors/actuators → PLC/RTU → communication network → SCADA server → HMI/historian. Control commands travel in the reverse direction. The actual implementation varies by industry, but this conceptual flow helps learners understand the responsibilities of each component.

Table 4.2.1 Key components and functional roles in SCADA Systems

Component	Main role	Typical information
Sensor	Measures physical condition	Temperature, pressure, level
Actuator	Changes physical process	Valve, motor, switch
PLC	Local control and automation	Inputs, logic, outputs
RTU	Remote telemetry and control	Remote field measurements
HMI	Operator visualization and interaction	Status, trends, alarms
Historian	Historical storage	Time-stamped process data

SGOU

4.2.4 Sensors, Actuators and Field Devices

Sensors convert physical conditions into signals that can be measured by a control system. A temperature sensor may produce a voltage, resistance change or digital value. A pressure sensor represents pressure in a form that an input module can interpret. Actuators perform the opposite kind of role: they receive commands and influence the physical process, for example by opening a valve, switching a motor or adjusting a control element.

Field devices must operate in environments that may contain vibration, dust, moisture, electromagnetic interference or temperature extremes. Industrial design therefore pays attention to signal conditioning, electrical isolation, calibration and maintenance. A measurement is only useful when the system can trust its relationship to the real physical variable.

SCADA commonly distinguishes between analogue and discrete information. An analogue value such as tank level can vary across a range, whereas a discrete status such as pump ON/OFF represents a limited set of states. Understanding this distinction helps learners interpret how process data is represented and how alarms are generated.

4.2.5 RTUs and PLCs

Remote Terminal Units and Programmable Logic Controllers are important field-level components in SCADA. An RTU is designed for collecting remote signals and communicating with supervisory systems. It is particularly useful where equipment is geographically distributed, such as pumping stations or electrical substations. A PLC is a programmable industrial controller widely used to execute deterministic control logic near machines and processes.

The distinction is not absolute in modern systems. Many contemporary devices combine functions and can communicate using multiple industrial protocols. The important conceptual difference is the role each component commonly plays. RTUs emphasize remote telemetry and communication, while PLCs emphasize local control and automation.

A PLC may read sensors, execute a control program, and update actuator outputs within a repeating scan cycle. This local behavior allows a physical process to continue operating even when supervisory communication is interrupted. SCADA can then supervise the process, display status and provide operator-level commands.

Note : In many IoT systems, the protocol that looks simplest at the application level may still require careful choices about reliability, security, buffering and failure recovery.

4.2.6 HMI and Operator Interaction

The Human-Machine Interface provides the visual and interactive layer through which operators understand the process. An HMI screen may show equipment status, numerical values, trends, alarm conditions and controls. A good HMI reduces cognitive load by presenting the most important information clearly rather than filling the screen with decorative graphics.

SGOU

Operators need to distinguish normal conditions from abnormal ones quickly. Consistent symbols, meaningful labels, clear units and well-designed alarm indications are therefore important. An HMI should support decision-making without hiding critical information.

In a water plant, for example, an operator might see tank levels, pump states and flow rates. If a level crosses a configured limit, the system can raise an alarm. The operator may acknowledge the alarm and, depending on the control philosophy and authorization, issue a command. The HMI is therefore part of an operational workflow, not merely a display.

4.2.7 Data Acquisition and Historians

Data acquisition involves collecting process values and status information from field devices and making the information available to supervisory applications. A historian is a specialized storage system used to retain time-related process data. Historical records can support troubleshooting, maintenance, compliance, performance analysis and long-term planning.

Suppose a pump repeatedly fails during high-demand periods. Historical temperature, vibration, pressure and motor-current trends may help engineers identify a pattern that is difficult to notice from a single live display. Time-stamped data allows events to be correlated and compared across operating conditions.

Not every value needs to be stored at the same frequency. High-frequency signals may require more storage, while slowly changing variables can be sampled less often. Good data management therefore balances resolution, storage cost, query performance and analytical value.

4.2.8 Alarms, Events and Trends

An alarm indicates that a process condition requires attention. An event records something significant that occurred, such as a state change, operator action or system message. A trend displays how a process variable changes over time. Together, these mechanisms help operators move from raw values to operational understanding.

Alarm design is particularly important. If every small fluctuation creates an alarm, operators may face alarm flooding and begin ignoring alerts. A meaningful alarm should correspond to a condition that requires a response. Priority, limits, delay and acknowledgement behavior should be defined according to the operational context.

Trends provide a complementary perspective. An operator who sees a pressure value of 8 bar may not know whether it is stable or rapidly increasing. A trend can reveal the direction and rate of change. Thus, SCADA systems combine present-state information with historical context.

SGOU

4.2.9 SCADA Communication

SCADA devices need communication mechanisms that match industrial requirements. Depending on the deployment, communication may use Ethernet, serial links, radio, cellular networks, fiber or other media. At the protocol level, industrial systems may use protocols such as Modbus, DNP3, IEC 60870-5-104 or vendor-specific mechanisms.

Protocol choice is influenced by device capabilities, interoperability requirements, timing, bandwidth, existing infrastructure and security. Legacy systems may contain older serial equipment, while newer installations may use IP-based industrial networks. Gateways can translate between technologies so that modernization does not require replacing every field device at once.

Reliable communication is important, but SCADA systems also need to handle communication failure explicitly. Timeouts, stale-data indicators, redundant links and fallback control strategies can prevent operators from mistaking missing information for a normal process state.

4.2.10 Modern Trends in SCADA

Modern SCADA systems are increasingly connected with IoT, cloud computing, edge computing, analytics and artificial intelligence.

- **SCADA and IoT :** IoT technologies can extend SCADA by connecting operational information with cloud platforms, analytics systems and enterprise applications.

  A gateway may collect selected SCADA information and forward it to an IoT platform. The IoT platform can then support large-scale analysis and visualization.

- **Cloud-Based SCADA** : Cloud technologies can provide scalable storage, remote monitoring and centralized analytics. However, time-critical control should not automatically be moved to the cloud because network delay or communication failure may affect physical processes.

- **Edge Computing :** Edge computing processes information close to the source of data. In industrial environments, an edge device can analyse sensor information locally and send only important results to a central or cloud system.

- **Artificial Intelligence and Machine Learning :** AI and ML can analyse historical SCADA data to identify unusual behaviour, predict equipment failure and support predictive maintenance.

- **Digital Twins :** A digital twin is a digital representation of a physical system. SCADA data can contribute real-time information to a digital twin, allowing engineers to understand and analyse the behaviour of physical equipment.

SGOU

- **Industrial Internet of Things :** The integration of industrial systems with IoT technologies is often associated with the Industrial Internet of Things (IIoT). IIoT can improve monitoring, analytics, interoperability and maintenance while maintaining appropriate local control.

4.2.11 SCADA and IoT Integration

IoT technologies can extend SCADA by connecting operational data to analytics platforms, enterprise applications and cloud services. A gateway can collect process information from a SCADA environment, normalize it and forward selected data to an IoT platform. Analytics can then identify patterns across many sites or correlate operational data with maintenance and business information.

Integration must respect the difference between monitoring and control. Sending a temperature history to a cloud analytics service is generally less time-critical than sending a command that changes a valve position. Many architectures therefore keep fast control local and expose carefully selected information to higher-level systems.

A layered approach is useful: field control remains close to the process, SCADA provides supervision, and IoT/cloud services provide broader analytics and integration. This arrangement can improve visibility without unnecessarily placing time-critical control functions on an external network.

Recap

- SCADA combines supervisory control with data acquisition for physical processes.
- Sensors measure process variables and actuators influence physical equipment.
- PLCs commonly execute local industrial control logic, while RTUs support remote telemetry and control.
- HMIs present process information and support authorized operator interaction.
- Historians preserve time-stamped process data for analysis and troubleshooting.
- Alarms identify conditions that require attention, while trends reveal changes over time.
- Reliability, redundancy, maintenance and communication-failure handling are central SCADA concerns.
- IoT can extend SCADA with analytics and integration while keeping time-critical control appropriately local.
- SCADA security requires authentication, authorization, segmentation, monitoring and controlled maintenance.

SGOU

Objective Type Questions

1. What does SCADA stand for?
2. Which device commonly executes local industrial control logic?
3. What does an RTU primarily support?
4. Which component provides operator visualization?
5. What is a historian used for?
6. What is the purpose of an alarm?
7. Which device converts a control command into physical action?
8. What does a trend show?
9. Why is redundancy used in SCADA?
10. Why is SCADA security important?

Answers

1. Supervisory Control and Data Acquisition
2. PLC
3. Remote telemetry and control
4. HMI
5. Historical process data
6. To alert an operator to an abnormal condition
7. Actuator
8. Change of a process variable over time
9. To improve availability
10. Because compromise can affect physical processes

SGOU

Assignments

1. Explain the architecture of a SCADA system and describe the role of each major component.
2. Compare PLCs and RTUs in terms of their typical roles in industrial automation.
3. Describe how an HMI, historian and alarm system work together in a control room.
4. Explain the importance of data acquisition and time-stamped historical data.
5. Discuss common SCADA communication requirements and give suitable examples.
6. Explain how IoT can extend SCADA without unnecessarily moving time-critical control to the cloud.
7. Discuss SCADA security threats and basic protective measures.
8. Prepare a case study of SCADA use in water, power, manufacturing or transportation infrastructure.

Suggested Reading

1. Stouffer, Keith, et al. Guide to Industrial Control Systems (ICS) Security. NIST.
2. Clarke, Gordon, and Deon Reynders. Practical Modern SCADA Protocols. Elsevier.
3. Hanes, David, et al. IoT Fundamentals. Cisco Press.
4. NIST Cybersecurity Framework and ICS security guidance.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Boyes, Walt, et al. The Handbook of Industrial Automation. 2nd ed., CRC Press, 2020.
3. Clarke, Gordon, and Deon Reynders. Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems. Elsevier, 2004.
4. Stouffer, Keith, et al. Guide to Industrial Control Systems (ICS) Security. NIST Special Publication 800-82, National Institute of Standards and Technology.

SGOU



Generic Web-Based Protocols

Learning Outcomes

After completion of this unit, learner will be able to:

- explain why web technologies are useful for connected IoT applications
- identify the roles of URI, HTTP methods, status codes, headers, resources, and representations
- describe REST-style communication and the basic operation of web APIs
- explain the purpose of WebSocket and related real-time web communication patterns
- recognize practical issues such as caching, authentication, payload formats, and secure web communication in IoT

Prerequisite

When you open a weather website, your browser requests information from a server and receives a response that can be displayed immediately. Behind that familiar experience are simple but powerful ideas: identifying resources, sending requests, receiving responses, representing information in standard formats, and securing communication. IoT systems can use the same web foundation to expose device data, accept commands, connect dashboards and integrate services. You do not need to be a web developer to understand these ideas. If you can imagine a device as a source of resources and a client as a consumer of those resources, you already have the starting point for this unit.

Keywords

HTTP, URI, REST, Web API, GET, POST, WebSocket, JSON

SGOU

Discussion

4.3.1 Web Technologies and IoT

Web technologies provide standardized ways for software systems to exchange information. IoT applications benefit because devices and services often need to communicate with dashboards, mobile applications, enterprise systems and cloud platforms. Instead of inventing a completely different communication model for every application, developers can use widely supported web concepts.

HTTP is one of the most familiar application protocols on the Internet. A client sends a request to a server, and the server returns a response. In IoT, the client may be a gateway or application, while the server may expose device data through a web API. The communication can be used for telemetry retrieval, configuration, device registration or management.

Web-based communication is attractive because tools, libraries and infrastructure are widely available. However, IoT devices can be constrained, so the overhead and connection behavior of conventional web communication must be considered. A gateway often helps by translating between device-oriented protocols and web APIs.

The central idea is interoperability. If an IoT service exposes resources through well-defined web semantics, many different clients can access it without needing to understand the internal implementation of the device.

4.3.2 The HTTP Request-Response Model

HTTP follows a request-response model. A client sends a request that contains information such as a method, target resource and headers. The server processes the request and returns a response containing a status code, headers and, when appropriate, a message body.

Consider a smart-home dashboard requesting the current room temperature. The dashboard may send a GET request for a resource representing the room or sensor. The server responds with a representation of the current measurement, perhaps encoded as JSON. If the dashboard later sends a configuration update, it may use a different HTTP method.

The request-response model is simple and stateless at the protocol level: each request can be understood in relation to the information it carries, while application systems may maintain state separately. This property supports scalability, but repeated polling can create traffic and delay when a client wants frequent updates. For near-real-time needs, other patterns such as WebSocket may be more appropriate.

SGOU

Table 4.3.1 Transport / SCADA / Web / Application concept comparison (illustrative)

Method	Common purpose	IoT example
GET	Retrieve representation	Read latest sensor data
POST	Submit/process data or trigger action	Submit telemetry/event
PUT	Replace or create representation at known URI	Replace device configuration
PATCH	Partially modify representation	Change one setting
DELETE	Remove resource	Remove registered device record

4.3.3 HTTP Methods

HTTP methods communicate the intended action on a resource. GET is commonly used to retrieve a representation. POST is commonly used to submit data or request processing that may create a new resource or trigger an action. PUT is commonly associated with replacing or creating a representation at a known URI, while PATCH is commonly used for partial modification. DELETE requests removal of a resource where the server permits it.

The exact semantics should be respected because predictable methods improve interoperability. For example, a device API might expose a temperature resource through GET and a configuration resource through GET and PUT. A command endpoint might use POST when a request represents an action rather than simply replacing stored data.

Choosing methods carefully also makes an API easier to understand and test. A client should not have to guess whether a request retrieves information, updates configuration or initiates an operation. Clear semantics are especially useful in IoT environments where many devices may be produced by different vendors.

4.3.4 HTTP Status Codes and Headers

An HTTP response includes a status code that summarizes the outcome of a request. Codes in the 2xx range generally indicate success, 4xx codes indicate that the request cannot be fulfilled because of a client-side issue, and 5xx codes indicate server-side problems. Common examples include 200 for successful retrieval, 201 for successful creation, 400 for a malformed request, 401 for missing or invalid authentication, 404 for an unknown resource, and 500 for an internal server error.

Headers provide additional metadata. They can describe content type, caching instructions, authentication information, connection behavior and other aspects of the request or response. For IoT APIs, the Content-Type header can indicate that the payload is JSON, XML or another representation.

SGOU

Good status codes and headers make failures easier to diagnose. If a device API returns a generic error for every problem, clients cannot easily distinguish an invalid command from temporary service failure. Clear protocol behavior therefore improves both interoperability and operational support.

4.3.5 URI, Resources and Representations

A Uniform Resource Identifier (URI) identifies a resource. In a web-based IoT API, resources might represent devices, sensors, measurements, locations, configurations or events. A URI does not need to reveal how the resource is stored internally; it provides a stable identifier through which clients can interact with it.

For example, an API could expose a resource for a particular sensor and another for its latest reading. The representation returned by the server may be JSON, while another client could request a different representation if the API supports content negotiation.

This separation between resource and representation is important. The resource is the conceptual thing being accessed; the representation is the data format used to communicate its current state. Understanding the distinction helps learners design clearer APIs and avoid confusing a database record with the resource itself.

4.3.6 REST-Style IoT APIs

REST, or Representational State Transfer, is an architectural style associated with web-based systems. RESTful design emphasizes resources, standard operations, stateless interactions, and a uniform interface. Not every API that uses HTTP is automatically RESTful, but REST-style principles can make IoT services easier to understand and integrate.

Imagine a fleet-management service exposing resources for vehicles, sensors and alerts. Clients can retrieve a vehicle representation, submit information, update configuration and request deletion where appropriate. The API uses HTTP semantics rather than inventing separate action names for every operation.

Stateless interaction means that each request contains the information needed for the server to process it, rather than requiring the server to remember conversational state between requests. This can simplify scaling. However, application state can still exist as data; statelessness does not mean that the system has no state at all.

4.3.7 JSON and Other Payload Formats

IoT web APIs need a representation format for structured data. JSON is widely used because it is text-based, easy for many programming languages to process, and relatively compact compared with some older textual formats. A temperature reading might be represented using fields such as device identifier, timestamp, value and unit.

XML is another structured format with strong expressive capabilities, while binary formats may be more efficient for constrained networks. The appropriate choice depends on interoperability, payload size, parsing resources and application requirements.

SGOU

A good payload should be explicit enough to avoid ambiguity. A numeric value without a unit can be dangerous in engineering applications. Including a timestamp and device identifier can also make data more useful when messages are delayed, duplicated or aggregated. Thus, payload design is part of communication design, not merely a formatting detail.

4.3.8 Web APIs for Device Management

Web APIs can expose device-management functions such as registration, configuration retrieval, configuration update, firmware metadata and health information. A device-management service may provide resources that describe capabilities and current status.

Security and authorization are particularly important for management APIs. Reading a temperature value may have a different risk from changing an actuator configuration. APIs should therefore distinguish permissions according to operation and resource sensitivity.

A gateway can also expose a web API while hiding the details of the underlying field protocols. This allows enterprise applications to interact through a consistent interface even when the connected devices use different industrial or IoT protocols.

4.3.9 Polling, Long Polling and WebSocket

A simple HTTP client can repeatedly poll a server for new data. Polling is easy to implement but can produce unnecessary requests when nothing has changed. Long polling keeps a request open until new information becomes available or a timeout occurs, reducing some unnecessary traffic.

WebSocket provides a persistent, full-duplex communication channel over a connection that begins with an HTTP-based handshake. After establishment, both sides can send messages without the strict request-response pattern of ordinary HTTP. This makes WebSocket useful for live dashboards, interactive control interfaces and applications requiring frequent updates.

The choice depends on the communication pattern. A dashboard that needs a reading once every few minutes may work well with ordinary HTTP retrieval. A dashboard displaying rapidly changing equipment status may benefit from a persistent channel. The simplest mechanism that satisfies the timing and scalability requirements is usually preferable.

4.3.10 HTTPS – Secure HTTP

HTTPS (Hypertext Transfer Protocol Secure) is HTTP communication protected using TLS (Transport Layer Security).

HTTP by itself does not provide encryption for the data exchanged between a client and server. If sensitive information is transmitted over an unprotected connection, an attacker who can observe the communication may be able to read or manipulate it.

SGOU

HTTPS addresses this problem by establishing a protected communication channel.

In an IoT application, HTTPS can be used when a device, gateway or application communicates with a web server or cloud service.

For example, suppose a smart-home gateway sends device configuration information to a cloud server. Using HTTPS helps protect the communication from unauthorized observation and modification.

4.3.10.1 Security Services Provided by HTTPS

HTTPS with TLS can provide important security properties.

Confidentiality

Encryption helps prevent unauthorized parties from reading the transmitted information.

Integrity

Protection mechanisms help detect unauthorized modification of transmitted information.

Authentication

Digital certificates can help a client verify the identity of the server.

These properties are particularly important when IoT systems transmit credentials, configuration information, control commands or sensitive sensor data.

4.3.10.2 Digital Certificates

HTTPS commonly uses digital certificates as part of server authentication.

A certificate contains information about the identity of the server and is associated with a cryptographic public key. Certificates are normally issued by trusted certificate authorities.

When an IoT application connects to a secure web service, it can verify whether the server's certificate is valid and trusted.

Certificate management is therefore an important part of secure IoT communication.

4.3.10.3 HTTP and HTTPS Comparison

Table 4.3.2 HTTP and HTTPS

Feature	HTTP	HTTPS
Basic protocol	HTTP	HTTP over TLS
Encryption	Not provided by HTTP itself	Yes, through TLS

SGOU

Confidentiality	Not guaranteed	Protected
Integrity protection	Not provided by HTTP itself	Provided through TLS
Server authentication	Not provided by HTTP itself	Supported through certificates
IoT use	Non-sensitive or controlled environments	Secure device/cloud/API communication

HTTPS generally uses **TCP port 443**, whereas traditional HTTP commonly uses **TCP port 80**.

4.3.11 HTTP versus WebSocket

HTTP normally follows a request-response interaction. The client sends a request and the server sends a response.

WebSocket provides a persistent, full-duplex communication channel. After the connection is established, both the client and server can send messages.

Table 4.3.3 HTTP versus WebSocket

Feature	**HTTP**	**WebSocket**
Communication model	Request-response	Full-duplex
Persistent communication	Not normally required for each interaction	Yes
Server can send independently	Not in the normal request-response model	Yes
Suitable for	Web APIs, resource retrieval	Live monitoring and interactive applications
Example IoT use	Retrieve sensor data	Real-time equipment dashboard

SGOU

Recap

- Web technologies provide interoperable communication mechanisms for IoT applications.
- HTTP follows a request-response model using methods, status codes, headers and message representations.
- GET commonly retrieves resources, while POST, PUT, PATCH and DELETE support other operations.

- URIs identify resources and representations carry their communicated state.
- REST-style design emphasizes resources, stateless interactions and a uniform interface.
- JSON is widely used for structured web payloads, although other formats may be appropriate.
- Polling is simple but can create repeated traffic; WebSocket supports persistent full-duplex communication.
- HTTPS uses TLS to protect HTTP communication, while authentication and authorization control access.
- Web APIs can hide device-specific technologies behind consistent application interfaces.

Objective Type Questions

1. What protocol commonly carries web API requests?
2. Which HTTP method is normally used to retrieve a resource?
3. Which method is commonly used for partial modification?
4. What does a 404 status indicate?
5. What does a URI identify?
6. Which data format is widely used in web APIs?
7. What does REST emphasize?
8. Which mechanism provides a persistent full-duplex web channel?
9. What does HTTPS add to HTTP?
10. Why can excessive polling be inefficient?

SGOU

Answers

1. HTTP
2. GET
3. PATCH
4. Resource not found
5. A resource
6. JSON
7. Resources and a uniform interface
8. WebSocket
9. TLS protection
10. It generates repeated requests even when no new data exists

Assignments

1. Explain the HTTP request-response model using an IoT sensor example.
2. Describe GET, POST, PUT, PATCH and DELETE and give one IoT use for each.
3. Explain the importance of HTTP status codes and headers in device APIs.
4. Discuss URI, resource and representation with a suitable smart-home example.
5. Explain REST-style communication and why it can support interoperable IoT services.
6. Compare polling, long polling and WebSocket for a live monitoring application.
7. Discuss HTTPS, authentication and authorization requirements for an IoT web API.
8. Design a conceptual web API for a smart building and specify at least five resources and their operations.

SGOU

Suggested Reading

1. Fielding, Roy T. Architectural Styles and the Design of Network-Based Software Architectures.
2. Richardson, Leonard, and Mike Amundsen. RESTful Web APIs. O'Reilly Media.
3. Hanes, David, et al. IoT Fundamentals. Cisco Press.
4. RFC 9110: HTTP Semantics. Internet Engineering Task Force.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Fielding, Roy T. Architectural Styles and the Design of Network-Based Software Architectures. University of California, Irvine, 2000.
3. Richardson, Leonard, and Mike Amundsen. RESTful Web APIs. O'Reilly Media, 2013.
4. Kurose, James F., and Keith W. Ross. Computer Networking: A Top-Down Approach. 8th ed., Pearson, 2021.

SGOU



IoT Application Layer Protocols

Learning Outcomes

After completion of this unit, learner will be able to:

- explain the purpose of application-layer protocols in IoT communication
- describe the basic concepts and message patterns of MQTT, CoAP, AMQP, and related IoT protocols
- identify the roles of brokers, publishers, subscribers, topics, resources, and messages
- compare IoT application protocols according to communication pattern, overhead, reliability, and deployment context
- recognize practical protocol-selection and security considerations for IoT applications

Prerequisite

Imagine hundreds of sensors reporting to a monitoring platform. Some devices need to publish measurements without knowing which application will consume them. Others may be extremely constrained and need a compact request-response protocol. Still others may be part of an enterprise workflow where reliable messaging and queuing are important. The transport layer alone cannot express all these application requirements. Application-layer protocols provide the language that IoT software uses to exchange meaningful messages. In this unit, you will meet several important protocols and, more importantly, learn how to reason about when each communication style is useful.

Keywords

MQTT, CoAP, Broker, Publisher, Subscriber, Topic, AMQP, Application Layer

SGOU

Discussion

4.4.1 Role of the IoT Application Layer

The application layer is where network communication is connected to the meaning of an IoT application. A temperature value is not merely a sequence of bytes; it represents a measurement with a device identity, time, unit and context. Application-layer protocols define how such information is organized, addressed and exchanged.

Different IoT environments require different communication patterns. A publish-subscribe system allows a device to publish data without knowing every consumer. A request-response system allows a client to ask for a resource and receive a response. A message-queuing system can decouple producers and consumers through intermediate infrastructure.

This unit focuses on MQTT and CoAP as important IoT-oriented protocols and introduces AMQP as a messaging protocol commonly encountered in enterprise integration. The aim is not to memorize packet fields but to understand the communication models and the circumstances in which each model is useful.

Application protocols operate above transport services. Therefore, protocol selection should be considered as a stack: the application needs a communication model, the transport must support the required behavior, and the network must provide reachability. Security and operational management must span the complete path.

4.4.2 MQTT: The Publish-Subscribe Model

Message Queuing Telemetry Transport (MQTT) is a lightweight messaging protocol widely used in IoT. Its central model is publish-subscribe. Publishers send messages to topics, and subscribers receive messages for topics to which they are subscribed. A broker manages the distribution between publishers and subscribers.

The broker decouples the two sides. A temperature sensor does not need to know the address of every dashboard or analytics service. It publishes a message to an appropriate topic, and the broker distributes it to authorized subscribers. This is particularly useful when many devices and applications interact.

A topic is a structured name used for routing messages. Organizations can design topic hierarchies to represent sites, devices and measurements. The topic design should be clear and stable because it becomes part of the application's interface.

MQTT's lightweight design and simple publish-subscribe model make it attractive for constrained devices and unreliable networks, provided the deployment is engineered with suitable security, session and quality-of-service settings.

SGOU

Table 4.4.1 Application concept comparison (illustrative)

Protocol	Communication model	Typical strength	Typical context
MQTT	Publish-subscribe	Lightweight telemetry distribution	Sensor fleets, cloud telemetry
CoAP	Resource request-response	Constrained communication	Embedded devices and constrained networks
AMQP	Message-oriented middleware	Queues and enterprise routing	Enterprise integration

4.4.3 MQTT Broker, Publisher and Subscriber

The MQTT architecture contains three conceptual roles. A publisher produces messages, a subscriber consumes messages, and a broker receives publications and forwards them to matching subscribers. A single client can perform both publishing and subscribing.

Consider a smart-farm deployment. Soil sensors publish measurements to topics associated with field zones. A dashboard subscribes to the measurement topics. An alert service also subscribes and generates notifications when values cross thresholds. The sensors do not need direct connections to either application; the broker provides the distribution point.

This decoupling improves scalability and flexibility. New subscribers can be added without modifying every sensor. Publishers can remain unaware of the number of consumers. At the same time, the broker becomes operationally important and must be protected, monitored and, where necessary, made highly available.

MQTT therefore illustrates a major IoT design pattern: separate data producers from consumers through an intermediary that manages distribution.

4.4.4 MQTT Quality of Service

MQTT defines three commonly discussed Quality of Service levels. QoS 0 uses an at-most-once delivery approach, so a message is delivered according to the underlying connection without a protocol-level delivery guarantee. QoS 1 provides at-least-once delivery, which means a message may be delivered more than once. QoS 2 provides exactly-once delivery semantics through a more involved exchange.

These levels represent different trade-offs. QoS 0 has low overhead and may suit frequently refreshed measurements where an occasional lost value is acceptable. QoS 1 can be useful when a message should not be silently lost, provided the application can tolerate duplicates. QoS 2 provides stronger delivery semantics but introduces additional protocol work.

SGOU

The phrase 'exactly once' should be interpreted carefully. It refers to the protocol's delivery semantics, not a guarantee that the physical action represented by a message can never happen twice. Applications controlling actuators should still be designed with safe command handling, identifiers and idempotent behavior where appropriate.

4.4.5 MQTT Sessions, Retained Messages and Last Will

MQTT includes features that support practical IoT deployments. A retained message allows the broker to store the most recent retained message for a topic and provide it to a new subscriber. This can be useful when a dashboard needs the latest known state immediately after subscribing.

MQTT also supports a Last Will and Testament mechanism. A client can specify a message that the broker publishes if the client disconnects unexpectedly according to the protocol conditions. This can help other applications recognize that a device may no longer be online.

Session behavior determines how subscriptions and pending messages are handled across connections. These features should be selected based on application needs and broker capabilities. They are useful precisely because IoT devices can sleep, disconnect, reconnect or operate over intermittent networks.

4.4.6 CoAP: Web-Style Communication for Constrained Devices

The Constrained Application Protocol (CoAP) is designed for constrained nodes and networks. It follows a resource-oriented request-response style inspired by web concepts while using a compact message format. CoAP commonly operates over UDP and can support methods such as GET, POST, PUT and DELETE.

A CoAP server exposes resources identified by paths. A constrained client can request a resource, submit information or modify a resource. This makes CoAP a natural fit for devices that benefit from a compact protocol while still using concepts familiar from web communication.

Because CoAP commonly uses datagram transport, it includes its own mechanisms for message reliability and duplicate detection at the application-protocol level. Confirmable messages can be acknowledged, while non-confirmable messages can be used when the application does not require the same reliability behavior.

CoAP therefore combines web-like resource interaction with mechanisms suited to constrained environments.

4.4.7 CoAP Methods, Messages and Observe

CoAP messages carry information about the type of interaction, message identity and resource being accessed. A client can use GET to retrieve a resource, POST to submit information or trigger processing, PUT to update or create a resource, and DELETE to remove a resource where supported.

SGOU

CoAP also provides an Observe mechanism that allows a client to register interest in a resource and receive notifications when the resource changes. This can reduce the need for repeated polling. For a sensor whose value changes periodically, Observe can provide a useful event-like interaction pattern.

As with MQTT, application semantics remain important. A notification that a temperature changed does not automatically define what an application should do with the information. The protocol transports the interaction; the application defines its meaning and safety requirements.

4.4.8 AMQP and Message-Oriented Middleware

Advanced Message Queuing Protocol (AMQP) is a messaging protocol associated with reliable message-oriented middleware and enterprise integration. It supports concepts such as exchanges, queues, bindings and consumers. These mechanisms allow producers and consumers to be decoupled and can support routing and reliable processing workflows.

AMQP is often more feature-rich than lightweight IoT-oriented protocols. That can be an advantage when integrating IoT data with enterprise applications, business processes and complex messaging infrastructure. The additional capabilities may also mean greater implementation and operational overhead than a tiny device can comfortably support.

A common architecture therefore places an IoT gateway between constrained devices and an enterprise messaging platform. The gateway can translate lightweight device messages into richer enterprise messages while maintaining a clear boundary between operational and business systems.

4.4.9 MQTT, CoAP and AMQP Comparison

MQTT, CoAP and AMQP are all application-layer communication protocols, but their models differ. MQTT is centered on publish-subscribe messaging through a broker. CoAP provides resource-oriented request-response communication for constrained environments. AMQP provides rich message-oriented middleware features suited to reliable enterprise messaging.

No protocol wins in every situation. A large sensor fleet sending telemetry to many consumers may benefit from MQTT. A constrained node exposing a small set of resources may fit CoAP. An enterprise integration layer may use AMQP where queues and routing are important.

4.4.10 Selecting an Application Layer Protocol

There is no single application-layer protocol that is best for every IoT application.

Protocol selection should consider:

SGOU

Communication Model

Does the application require request-response, publish-subscribe or continuous two-way communication?

Device Constraints

Does the device have limited memory, processing power and battery capacity?

Reliability

Does the application require guaranteed or controlled delivery?

Latency

How quickly must information reach its destination?

Scalability

How many devices and applications will communicate?

Bandwidth

Is the available network capacity limited?

Security

Does the communication contain sensitive information or control commands?

Interoperability

Must the system integrate with existing web, enterprise or industrial systems?

4.4.11 IoT Application Protocol Selection – Example

Consider a smart agriculture system.

Soil-moisture sensors may need to send periodic measurements to a central monitoring platform. MQTT can be suitable because its publish-subscribe model allows many sensors to publish measurements while different applications subscribe to the required information.

A highly constrained field device exposing resources may use CoAP.

A web dashboard may communicate with a server using HTTP or WebSocket. HTTP is suitable for ordinary resource retrieval, while WebSocket can support live updates.

An enterprise application processing agricultural business information may use AMQP.

If the agricultural system contains a real-time distributed control component requiring sophisticated QoS, DDS may be considered.

This example shows that one IoT system may use multiple protocols at different communication boundaries.

SGOU

4.4.12 Security of IoT Application Protocols

Security must be considered when selecting and deploying application-layer protocols.

Important security requirements include:

- Authentication
- Authorization
- Encryption
- Integrity protection
- Secure key management
- Access control
- Secure device identity
- Monitoring and logging

For example, MQTT deployments can use secure transport and authentication mechanisms. CoAP deployments can use appropriate security mechanisms for constrained environments. HTTP-based communication should use HTTPS when protection is required.

Security requirements become particularly important when application-layer protocols are used to send control commands, not merely sensor readings.

SGOU

Recap

- Application-layer protocols express the communication model and meaning of IoT messages.
- MQTT uses publish-subscribe communication through a broker and topics.
- MQTT QoS levels provide different delivery semantics and overhead trade-offs.
- CoAP provides compact resource-oriented communication for constrained devices and commonly uses UDP.
- CoAP supports methods such as GET, POST, PUT and DELETE and includes Observe for change notifications.
- AMQP provides message-oriented middleware concepts such as exchanges and queues for enterprise integration.

- Protocol selection should be based on application requirements, device constraints and operational conditions.
- Security, credential lifecycle and authorization are essential parts of application-protocol deployment.
- A complete IoT architecture may combine multiple application protocols through gateways.

Objective Type Questions

1. Which protocol uses a publish-subscribe model?
2. What component distributes MQTT messages?
3. What is an MQTT topic?
4. Which MQTT QoS provides at-least-once delivery?
5. Which protocol is designed for constrained web-style communication?
6. What HTTP-like method retrieves a CoAP resource?
7. What CoAP feature supports resource change notifications?
8. What protocol is associated with exchanges and queues?
9. Why might a gateway translate between IoT protocols?
10. What should guide IoT protocol selection?

SGOU

Answers

1. MQTT
2. Broker
3. Message routing name
4. QoS 1
5. CoAP
6. GET
7. Observe
8. AMQP
9. Different segments have different requirements
10. Application requirements

Assignments

1. Explain the publish-subscribe model of MQTT with a smart-home example.
2. Describe the role of an MQTT broker, publisher, subscriber and topic.
3. Explain MQTT QoS 0, QoS 1 and QoS 2 and discuss suitable uses.
4. Describe CoAP resources, methods, messages and Observe.
5. Compare MQTT and CoAP in terms of communication model and constrained-device suitability.
6. Explain AMQP concepts and its role in enterprise IoT integration.
7. Discuss security requirements for MQTT, CoAP and AMQP deployments.
8. Design a multi-protocol smart-agriculture architecture and justify where MQTT, CoAP and/or AMQP would be used.

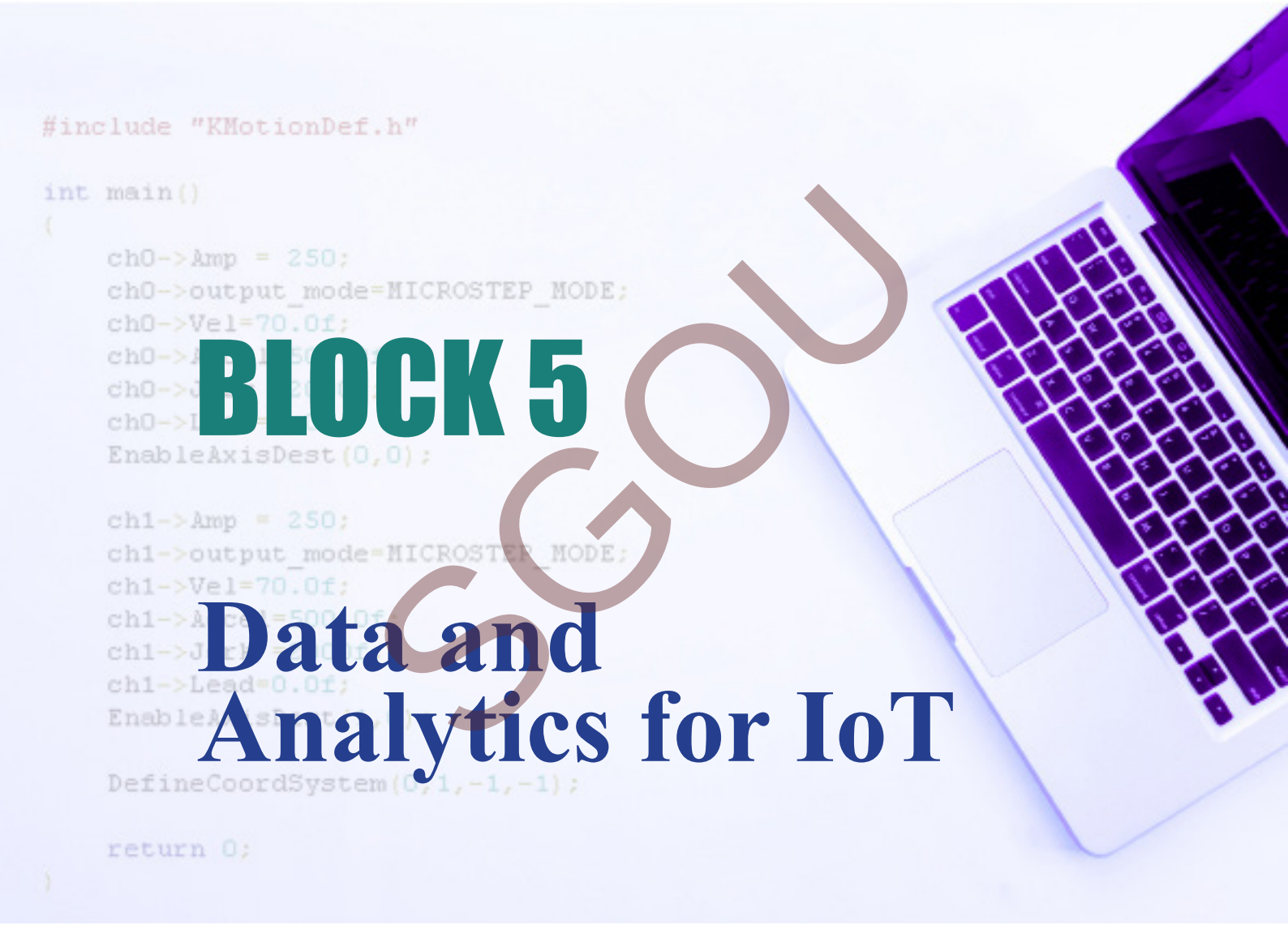
Suggested Reading

1. Hanes, David, et al. IoT Fundamentals. Cisco Press.
2. Shelby, Zach, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252.
3. OASIS. Advanced Message Queuing Protocol Version 1.0.
4. MQTT Version 5.0 specification, OASIS.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Light, Roger A. 'Mosquitto: server and client implementation of the MQTT protocol.' Journal of Open Source Software, 2017.
3. Shelby, Zach, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, IETF, 2014.
4. OASIS. Advanced Message Queuing Protocol Version 1.0. OASIS Standard.

SGOU



BLOCK 5

Data and Analytics for IoT

SGOU



Introduction to Data Analytics for IoT

Learning Outcomes

After completion of this unit, learner will be able to:

- explain why analytics is essential in an IoT environment
- identify the major stages of the IoT data lifecycle
- describe common forms of IoT data and basic data-quality concerns
- explain descriptive, diagnostic, predictive and prescriptive analytics
- distinguish batch analytics from streaming analytics in practical IoT situations

Prerequisite

Think of a smart irrigation field. A soil-moisture sensor reports a number, a weather service provides a forecast, and a pump controller records whether water was delivered. The numbers by themselves do not tell the farmer what to do. The real value appears when the readings are cleaned, compared, interpreted and connected to an action. You already encounter this pattern in fitness bands, vehicle dashboards and smart electricity meters. This unit builds on that familiar experience and shows how raw IoT observations become useful evidence. A basic understanding of sensors, communication networks, databases and simple averages is enough to begin.

Keywords

IoT data, Data lifecycle, Data quality, Descriptive analytics, Diagnostic analytics, Predictive analytics, Prescriptive analytics, Streaming analytics

SGOU

Discussion

"A sensor does not make a decision. It only provides an observation. Analytics is the bridge between the observation and the decision."

An Internet of Things system can contain hundreds or millions of sensing points. Each point may produce a small record, but the records accumulate over time and often arrive at different speeds. A useful analytics process therefore begins by asking a simple question: what decision are we trying to support? If a cold-storage operator wants to know whether food has been exposed to unsafe temperature, the relevant data may include temperature, door state, power status and time. If the purpose is maintenance, vibration and motor current may matter more. The analytical question determines what data is collected, how it is prepared and which method is appropriate.

IoT analytics is not restricted to advanced machine learning. A threshold rule, an average, a trend line or a comparison between two time periods can already be valuable. More sophisticated models become useful when the relationship is difficult to describe with simple rules. The important idea is that analytical sophistication should follow the decision requirement rather than become an end in itself.

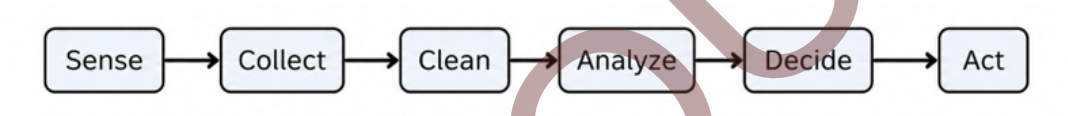


Fig. 5.1.1 From IoT observations to action

The lifecycle shown in Fig.5.1.1 is best understood as a loop rather than a one-way pipeline. Decisions generate new observations, those observations reveal whether the decision worked, and the resulting evidence can improve later decisions. In a smart building, for example, an energy-saving rule may reduce cooling during lightly occupied periods. Subsequent temperature and occupancy readings reveal whether comfort was maintained. Thus analytics supports learning from operations, not merely producing a report.

5.1.1 Why Data matters in IoT ?

Connected devices are often described as the eyes and ears of an IoT system. Yet an eye that only records images without interpretation is of limited practical value. Data analytics supplies that interpretation. It helps transform repeated observations into summaries, relationships, warnings and predictions that people or automated systems can use.

Consider a fleet of delivery vehicles. GPS readings show where vehicles are, engine sensors indicate operating conditions, and order systems provide delivery requirements. A basic dashboard can describe the current situation. Further analysis can explain why some routes are consistently delayed. Predictive methods can estimate which vehicles are likely to experience a fault. Prescriptive logic can then suggest a maintenance appointment or route change. The same raw data can therefore support several levels of decision-making.

SGOU

The usefulness of analytics also depends on context. A temperature of 28 °C may be ordinary in one environment and abnormal in another. A burst of network traffic may indicate a software update rather than an attack. An isolated sensor reading may be noise, while the same reading repeated across several sensors may be meaningful. Good IoT analytics therefore combines numerical evidence with time, location, device identity and operational knowledge.

Note : When an analytical result looks surprising, do not immediately assume that the algorithm is wrong. First check the measurement, units, timestamp, device identity and surrounding context.

Table 5.1.1 Questions answered by different analytical stages

Question	Useful evidence	Possible result
What is happening now?	Recent sensor readings	Current status
Why did it happen?	Events, trends, related variables	Likely cause
What may happen next?	Historical patterns and current state	Forecast
What should we do?	Forecast plus constraints	Recommended action

A mature IoT solution normally combines these views. Descriptive information keeps operators informed, diagnostic analysis supports troubleshooting, predictive analysis provides early warning, and prescriptive analysis connects evidence to choices. The four forms are complementary rather than competing alternatives.

5.1.2 The IoT Data Lifecycle

The data lifecycle describes the path followed by information from its origin to its eventual use and retention. Although real architectures differ, a practical IoT lifecycle can be described through sensing, collection, transmission, ingestion, validation, storage, processing, analysis, visualization and action. The stages may overlap, and some processing can occur at the edge before data reaches a central platform.

Sensing is the point at which the physical world is observed. Sensors may measure temperature, pressure, light, vibration, position, current or many other quantities. Collection gathers these observations, often through gateways or device-management services. Transmission moves the records through a communication network. At ingestion, the platform accepts the incoming records and checks whether they follow the expected structure.

Validation is especially important because an analytics system can process incorrect data very efficiently. A sensor may report an impossible value because of calibration failure, a timestamp may be missing, or a device may unexpectedly change its unit.

SGOU

Validation can flag such records for later inspection rather than silently treating them as facts.

Storage preserves information for operational use, historical analysis or compliance requirements. Processing then transforms the stored or incoming data. Examples include converting units, removing duplicates, calculating rolling averages and joining sensor readings with equipment metadata. Analysis applies statistical or machine-learning methods to answer the business or operational question. Visualization communicates the result through a dashboard, report or alert, while action closes the loop.

A Useful Habit

Before collecting large amounts of data, write down the decision the data must support. This simple step often reveals which measurements are essential and which are unnecessary.

5.1.3 Forms and Characteristics of IoT Data

IoT data is often time-stamped and associated with a device, place or event. A temperature record might contain device ID, timestamp, temperature, unit and location. A vehicle message may contain speed, latitude, longitude, fuel level and status flags. Camera systems may produce images or short video segments rather than a single number. Industrial systems may also produce logs, alarms and maintenance records.

The variety of formats matters because different data types require different handling. Structured records fit naturally into tables. Semi-structured messages such as JSON can contain changing fields. Unstructured material such as images, audio and free-text maintenance notes needs different storage and processing approaches. IoT platforms therefore commonly use more than one representation rather than forcing every source into one rigid format.

Scale is another defining characteristic. Volume refers to how much data is produced; velocity concerns how quickly it arrives; variety describes differences in type and source. Veracity draws attention to uncertainty and quality, while value asks whether the data contributes to a useful purpose. These dimensions help explain why a seemingly small device message can become an engineering challenge when multiplied across a large population.

Table 5.1.2 Common characteristics of IoT data

Characteristic	Meaning in IoT	Example
Volume	Amount of data accumulated	Years of temperature records
Velocity	Rate at which data arrives	Vibration samples every second

SGOU

Variety	Different formats and sources	Numbers, logs, images
Veracity	Trustworthiness and uncertainty	Noisy or drifting sensor
Value	Practical usefulness	Early fault warning

A good analytics design respects these characteristics from the beginning. High-velocity data may need streaming processing. High-volume historical data may require distributed storage. Varied data may need an integration layer. Uncertain data may require validation and quality indicators. The goal is not to collect everything simply because collection is possible; the goal is to preserve the evidence needed for meaningful decisions.

5.1.4 Data Collection and Data Quality

Data quality is a foundation rather than a final polishing step. An analytical model cannot reliably correct every defect introduced by sensing or communication. Common problems include missing readings, duplicated records, impossible values, incorrect units, clock drift and sudden changes caused by sensor replacement. These problems are particularly common in IoT because devices operate in physical environments and may be exposed to heat, vibration, interference or unstable connectivity.

Suppose a water meter normally reports litres per minute but one device begins reporting gallons per minute. The numerical values may look perfectly reasonable. Without unit information, however, the downstream analysis may conclude that consumption has changed dramatically. This example shows why metadata is part of data quality. Device identity, units, timestamp and calibration information can be as important as the measured value itself.

Quality checks should be proportionate to the application. A smart display that shows room temperature can tolerate occasional missing readings. A safety-related controller requires much stronger validation and fail-safe behaviour. Useful checks include range validation, continuity checks, timestamp validation, duplicate detection and cross-sensor comparison. Suspicious records can be flagged rather than automatically deleted so that analysts can later determine what happened.

Data preparation can also involve normalization, aggregation and enrichment. Normalization may convert different units to a common scale. Aggregation can turn second-by-second readings into minute-level summaries. Enrichment may attach equipment type, location or maintenance history. These operations reduce ambiguity and make later analysis easier to interpret.

Practical Check

If a sensor suddenly reports an extreme value, ask four questions: Was the device working? Was the unit correct? Did nearby sensors show the same change? Did a real-world event occur at that time?

SGOU

5.1.5 Descriptive and Diagnostic Analytics

Descriptive analytics answers the question, "What happened?" It summarizes observed data using counts, averages, minimum and maximum values, distributions, trends and dashboards. In an IoT setting, a building manager might examine hourly energy consumption by floor. A factory operator might view the number of alarms raised by each machine during a shift.

A useful description does more than display numbers. It selects measures that match the operational question and preserves the time context. An average can hide short periods of extreme behaviour, so a dashboard may also show peaks, ranges or percentiles. A trend can reveal gradual drift that is invisible in a single snapshot. Descriptive analytics is therefore the first layer of understanding, not merely a decorative report.

Diagnostic analytics goes one step further by asking, "Why did it happen?" The answer may require comparing variables, looking for temporal relationships, examining event logs or grouping observations by device. Suppose energy use increases sharply at 2 p.m. A diagnostic analysis may reveal that a cooling system started, occupancy increased or a control schedule changed. The analysis does not automatically prove causation; it identifies evidence that helps explain the event.

Table 5.1.3 Four analytical perspectives

Analytics	Central question	Typical IoT output
Descriptive	What happened?	Trend, summary, dashboard
Diagnostic	Why did it happen?	Cause indicators, comparisons
Predictive	What is likely next?	Forecast or risk score
Prescriptive	What action is preferable?	Recommendation or policy

Note : Correlation can be useful evidence for diagnosis, but correlation alone does not establish cause. Operational knowledge and controlled investigation may be needed.

5.1.6 Predictive and Prescriptive Analytics

Predictive analytics uses historical and current information to estimate a future or unknown outcome. In predictive maintenance, vibration, temperature, load and operating hours can be used to estimate whether a machine is approaching an abnormal state. In demand forecasting, previous consumption patterns and external factors can support a forecast of future electricity use.

Prediction is always subject to uncertainty. A forecast should therefore be interpreted as an estimate, not a promise. Model performance depends on the quality and representativeness of the training data, the stability of the environment and the suitability of the variables. When conditions change, a model may need monitoring or retraining.

SGOU

Prescriptive analytics asks a different question: given the evidence and the available choices, what action should be considered? A smart building might compare several cooling schedules while respecting comfort limits. A fleet system might balance delivery time, fuel consumption and vehicle availability. Prescriptive logic may use optimization, rules, simulation or a combination of methods.

The distinction is subtle but important. A prediction says that a pump has a high probability of failure within a future period. A prescription says that the pump should be inspected now because inspection is cheaper and safer than waiting. The second statement introduces an action and its consequences. That means prescriptive analytics must consider constraints, costs and operational priorities in addition to predictive accuracy.

Worked Example

A sensor indicates rising motor vibration. Descriptive analytics shows the trend; diagnostic analysis compares it with load and temperature; predictive analytics estimates the probability of failure; prescriptive logic recommends inspection or controlled shutdown. The four stages form one decision chain.

5.1.7 Batch and Streaming Analytics

Batch analytics works on accumulated data. It is appropriate when an answer can wait for a scheduled computation, such as producing a monthly energy report or recalculating historical statistics. Batch jobs can process large datasets efficiently and can be repeated when a transformation or analytical method changes.

Streaming analytics processes records as they arrive or in very small intervals. It is appropriate when delay reduces the usefulness of the result. A fire alarm, equipment protection rule or intrusion alert may need to react within seconds. Streaming systems commonly use windows, event-time concepts and continuous transformations to summarize an unending flow of records.

The choice is not always either-or. A modern IoT platform may stream recent events for immediate alerts while simultaneously storing the same events for later batch analysis. The streaming path answers "What needs attention now?" and the batch path answers "What can we learn from the history?"

Table 5.1.4 Batch and streaming analytics

Aspect	Batch	Streaming
Input	Accumulated dataset	Continuous event flow
Latency	Minutes to hours or longer	Seconds or milliseconds when required
Typical use	Historical reporting	Alerts and live monitoring
Strength	Large-scale repeated computation	Timely reaction
Challenge	Stale results if delayed	State, ordering and reliability

SGOU

A sensible architecture chooses the processing mode according to the decision deadline. If a result is needed once a quarter, a low-latency stream is unnecessary. If a machine must be stopped immediately when a dangerous pattern appears, a purely nightly batch process is clearly inadequate.

5.1.8 Mini Case Study: Smart Agriculture

Consider a smart farm divided into several irrigation zones. Each zone has soil-moisture sensors, a weather feed and a pump controller. The immediate goal is to provide enough water for healthy crops while avoiding unnecessary irrigation. The analytical system receives soil moisture, recent rainfall, forecast information and pump activity.

The first task is data validation. Sensor readings are checked for plausible ranges and missing intervals. The second task is aggregation: several readings in the same zone can be summarized so that one noisy measurement does not dominate the decision. Descriptive analytics can show moisture trends by zone. Diagnostic analysis can investigate why one area remains dry even though the pump is operating; the cause might be a blocked pipe or a faulty sensor.

Predictive analytics can estimate when soil moisture is likely to fall below an acceptable range by combining recent measurements with forecast weather. Prescriptive logic can then consider whether to irrigate immediately, delay because rain is expected, or inspect the irrigation equipment. The system can also send selected historical data to a central platform for long-term comparison across seasons.

Case Insight

The value of the system does not come from having more sensors. It comes from connecting trustworthy measurements to a clear agricultural decision.

This case also illustrates the importance of human oversight. A recommendation should be understandable enough for an operator to question it when field conditions are unusual. For example, a newly planted section may require different moisture thresholds from an established crop. Analytics supports expertise; it does not eliminate the need for it.

The same reasoning applies to smart buildings, transportation, manufacturing and healthcare monitoring. In every case, the central design question remains the same: what observation is needed, how trustworthy is it, what analysis is appropriate, and what decision should follow?

SGOU

Recap

- IoT analytics converts device observations into information, predictions and decisions.
- The data lifecycle includes sensing, collection, ingestion, validation, storage, processing, analysis, communication and action.
- IoT data can exhibit high volume, velocity and variety, while veracity and value remain essential considerations.
- Data quality problems include missing values, duplicates, impossible readings, wrong units and inaccurate timestamps.
- Descriptive analytics summarizes what happened, while diagnostic analytics investigates why.
- Predictive analytics estimates likely future outcomes and must be interpreted with uncertainty in mind.
- Prescriptive analytics connects analytical evidence with possible actions and operational constraints.
- Batch analytics is suited to accumulated historical workloads; streaming analytics supports time-sensitive decisions.
- Analytics should be designed around a decision rather than around a fashionable tool.
- Combining several analytical perspectives often provides a stronger operational picture than using one technique alone.

SGOU

Objective Type Questions

1. Why is analytics necessary in an IoT environment?
2. Which stage of the lifecycle checks whether sensor records are plausible and consistent?
3. What does volume mean when discussing IoT data?
4. Which type of analytics asks what happened?
5. Which type of analytics investigates why an event occurred?
6. Which type of analytics estimates a future outcome?
7. Which type of analytics connects evidence to a preferred action?
8. Which approach processes accumulated historical data?

9. Which approach handles data continuously as it arrives?
10. Why can an extreme sensor value be misleading?

Answers

1. To turn observations into useful information and decisions.
2. Data validation.
3. The amount of data.
4. Descriptive analytics.
5. Diagnostic analytics.
6. Predictive analytics.
7. Prescriptive analytics.
8. Batch analytics.
9. Streaming analytics.
10. It may be noise, a fault, an incorrect unit, or a genuine event.

Assignments

1. Explain the IoT data lifecycle with a suitable smart-building or smart-agriculture example.
2. Differentiate descriptive, diagnostic, predictive and prescriptive analytics, using one IoT example for each.
3. Discuss the major dimensions of IoT data scale and quality and explain why each matters.
4. Compare batch and streaming analytics and identify an application in which each is appropriate.
5. A sensor produces occasional extreme values. Explain a systematic method for deciding whether the values are genuine or erroneous.
6. Explain how two or more IoT data sources can be combined to support a practical operational decision.
7. Why does a large volume of data not automatically guarantee useful analytics? Discuss with an example.
8. Prepare a short analytical plan for an IoT application of your choice, identifying the question, data, analytics type and expected decision.

SGOU

Suggested Reading

1. Buyya, Rajkumar, and Amir Vahid Dastjerdi, editors. Internet of Things: Principles and Paradigms. Morgan Kaufmann, 2016
2. Han, Jiawei, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. 3rd ed., Morgan Kaufmann, 2011
3. Provost, Foster, and Tom Fawcett. Data Science for Business. O'Reilly Media, 2013
4. James, Gareth, et al. An Introduction to Statistical Learning. 2nd ed., Springer, 2021.

Unit Completion Check: Can you explain, in your own words, how a raw sensor observation becomes a trustworthy analytical result and then supports an operational decision? If yes, proceed to the next unit.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A Survey." Computer Networks, vol. 54, no. 15, 2010, pp. 2787–2805.
3. Kleppmann, Martin. Designing Data-Intensive Applications. O'Reilly Media, 2017.
4. Karau, Holden, et al. Learning Spark: Lightning-Fast Data Analytics. O'Reilly Media, 2015.

SGOU



Big Data Analytics Tools and Technology

Learning Outcomes

After completion of this unit, learner will be able to:

- explain why IoT workloads often require big-data technologies
- describe distributed storage and parallel processing
- explain the basic roles of Hadoop, MapReduce and Apache Spark
- identify common ingestion, database and cloud technologies used with IoT
- describe the stages of a scalable IoT analytics pipeline

Prerequisite

Imagine a city with thousands of connected streetlights, parking meters, traffic sensors and environmental monitors. A single device may send only a small message, but thousands of devices reporting every few seconds produce a stream that grows quickly. Trying to store and process everything on one computer becomes inconvenient and eventually impractical. This situation introduces the central idea of big-data technology: divide storage and computation across resources so that a large workload can be handled reliably. Familiarity with files, databases, networks and simple programming ideas will help.

Keywords

Big data, Hadoop, MapReduce, Apache Spark, Data ingestion, Distributed storage, NoSQL, Cloud analytics

SGOU

Discussion

"A large IoT system is not simply a small IoT system repeated many times. Scale changes the engineering problem because storage, computation, communication and failure become collective concerns."

Big data in IoT emerges from the combination of device count, sampling rate, message size, retention period and application scope. A few thousand sensors reporting once per hour may be manageable on modest infrastructure. Millions of devices reporting every second can produce a workload that requires distributed storage, parallel processing and carefully designed ingestion.

The practical objective is not to use the largest technology stack available. It is to match the architecture to the workload. A small deployment may need only a database and a dashboard. A city-scale platform may need message brokers, distributed storage, stream processing, batch processing, multiple databases and cloud services. The tools become meaningful only when their roles are understood.



Fig. 5.2.1 Logical flow of IoT

The above figure shows a common logical flow. Devices generate records, an ingestion layer accepts them, storage preserves them, distributed processing transforms them, and analytical services produce results for users or applications. In practice, some processing may occur at the edge and the same event may be routed to several consumers.

5.2.1 Why IoT creates Big Data

Volume is the most obvious dimension of IoT scale, but it is not the only one. Velocity can become the dominant challenge when devices report frequently. Variety appears when a platform combines numerical readings with logs, images, device metadata and external information. Veracity becomes important because physical sensors are not perfect. Value prevents the discussion from becoming purely technical: storing a petabyte of data is not useful if nobody can explain why it should be retained.

Distributed systems address scale by dividing storage and computation among multiple machines. The benefit is horizontal growth: capacity can often be increased by adding nodes rather than replacing one machine with a much larger machine. Distribution can also improve resilience when data and workloads are replicated or recoverable. However, distribution introduces coordination, partitioning and failure-handling challenges.

Consider a year of temperature records from many buildings. The data can be partitioned by building or by time period. Several workers can calculate local summaries simultaneously, after which a coordinating stage combines those summaries. The result is the same logical calculation, but the work has been divided so that more resources can participate.

SGOU

Table 5.2.1 Big data dimensions in IoT

Dimension	IoT example	Engineering implication
Volume	Years of sensor history	Distributed storage
Velocity	Hundreds of events per second	Streaming ingestion
Variety	Telemetry plus images and logs	Multiple data models
Veracity	Drift and missing readings	Validation and quality flags
Value	Useful operational insight	Purpose-driven retention

Note : Big-data technology is fundamentally about organizing large workloads so that storage and computation can grow, remain available and produce results within the required time.

5.2.2 Distributed Storage and Processing

Distributed storage divides data across several nodes while maintaining enough information to locate and retrieve the pieces. Depending on the system, data may be replicated so that the loss of one node does not make the information unavailable. Partitioning and replication are therefore central ideas in large-scale IoT storage.

Distributed processing follows a similar principle. A large computation is divided into smaller tasks that can run in parallel. For example, a system calculating daily average temperature for ten thousand devices can distribute devices among workers. Each worker calculates local sums and counts, and a final stage combines them. The method is simple, but it demonstrates why parallelism matters.

Fault tolerance becomes increasingly important as the number of components grows. If one machine fails in a small system, the impact may be obvious. In a large cluster, occasional failures are expected. A resilient platform therefore employs mechanisms such as replication, retries, checkpointing, and task recomputation. The goal is to maintain service availability even when individual components fail. Distributed systems also create trade-offs. Moving data between nodes costs network bandwidth. Synchronizing state can add delay. Debugging a problem across several services can be more difficult than debugging a single program. Good engineering therefore balances scalability with operational simplicity.

Note : Adding more machines is not automatically a solution. If the workload is badly partitioned or spends most of its time waiting for network communication, additional nodes may provide little benefit.

5.2.3 Hadoop and the MapReduce Idea

The Hadoop ecosystem popularized a practical approach to storing and processing large datasets across clusters of commodity machines. For learners, two ideas are

SGOU

particularly useful: distributed storage and distributed batch computation. Hadoop demonstrates how a large job can be organized as coordinated work rather than as one sequential program.

MapReduce separates a computation into a map stage and a reduce stage. During mapping, input records are transformed into intermediate key-value pairs. Related keys are grouped, and the reduce stage combines their values. The pattern is powerful because many map operations can run independently. A later stage then assembles the partial results.

Suppose a platform stores records containing device ID, timestamp and temperature. A mapping step can emit the device ID together with its temperature. The reduce step can calculate a sum and count for each device. Dividing the sum by the count produces the average. The example is intentionally simple: the important lesson is how a familiar calculation becomes a distributed workflow.

Table 5.2.2 Simplified MapReduce example

Stage	IoT example	Output
Map	Read device ID and temperature	(device, temperature)
Group	Collect values by device	All readings for each device
Reduce	Compute sum and count	Summary per device
Final	Calculate average	Average temperature

MapReduce is especially suited to large batch computations. Repeated iterative workloads may be less efficient when intermediate results repeatedly need to be written and read. This limitation helps explain why other processing engines, including Spark, became attractive for many analytical tasks.

5.2.4 Apache Spark and In-Memory Analytics

Apache Spark is a general-purpose distributed data-processing engine that supports batch processing, interactive analysis and streaming workloads. One of its notable characteristics is the ability to keep intermediate data in memory when appropriate, reducing repeated disk access for some iterative calculations. Spark can therefore be useful when an IoT workflow requires several transformations or analytical passes over related data.

A typical IoT workload might clean incoming records, group readings by location, calculate hourly statistics, join the result with occupancy information and identify unusual patterns. Spark provides higher-level programming interfaces that allow developers to describe transformations and actions while the engine manages much of the distributed execution.

It is misleading to describe Spark simply as a replacement for Hadoop. Hadoop and Spark represent different processing approaches and ecosystems, and real deployments

SGOU

can combine technologies. The right question is whether the workload is primarily batch, interactive, streaming or a mixture. Tool choice should follow that requirement.

Note : Do not begin with the product name. Begin with the workload: How much data? How quickly does it arrive? How long may the computation take? How often must it repeat?

Table 5.2.3 Matching workload needs with technology roles

Need	Possible technology role
Distributed batch storage	Hadoop-style distributed storage
Large batch computation	MapReduce or Spark
Iterative analytics	Spark
Continuous event processing	Streaming engine / broker
Flexible operational records	NoSQL or document store

Understanding these roles gives an IoT engineer a durable foundation even when specific products change over time.

5.2.5 Data Ingestion and Messaging

Before analytics can begin, data must enter the platform. IoT devices may communicate through lightweight protocols such as MQTT, while larger systems often use message brokers or streaming platforms. The ingestion layer provides a controlled entry point and can separate producers from consumers.

A message broker acts as an intermediary. Sensors publish messages without needing to know exactly which applications will consume them. A live dashboard, storage service and anomaly-detection process can therefore subscribe to the same event stream. This loose coupling makes the architecture easier to extend because a new consumer can be added without changing every sensor.

Ingestion is also a data-quality boundary. The platform can check whether a message contains a device identifier, timestamp and expected fields. It can detect malformed records, unexpected units or unsupported message versions. Early validation prevents downstream applications from having to solve the same basic problem repeatedly.

> If three applications need the same sensor event, direct device-to-application connections create unnecessary dependencies. A messaging layer can receive the event once and distribute it to the required consumers.

In high-volume systems, ingestion also needs to manage bursts. A sudden event, software update or network reconnection can cause many devices to transmit simultaneously. Buffering and back-pressure mechanisms help prevent downstream components from being overwhelmed. This is why ingestion should be treated as part of analytics engineering rather than as mere network plumbing.

SGOU

5.2.6 Databases for IoT Analytics

No single database model is ideal for every IoT workload. Relational databases are valuable when structured relationships, transactions and well-defined schemas are central. Time-series databases are designed around measurements indexed by time and are often convenient for sensor histories. NoSQL systems can support flexible or very large datasets, while object storage is useful for economical retention of large files and raw historical data.

Table 5.2.4 Storage choices for IoT workloads

Storage style	Strength	Typical IoT use
Relational	Structured relationships and transactions	Device inventory, configuration
Time-series	Efficient time-based queries	Sensor history and monitoring
NoSQL / document	Flexible schema and scalable access	Semi-structured device records
Object storage / data lake	Large, economical retention	Raw historical data and batch analysis

A time-series query may ask for the average temperature of a device between 2 p.m. and 4 p.m. A relational query may ask which devices belong to a particular building and who manages them. A document store may hold changing device attributes. The access pattern should guide the choice.

Large IoT platforms often use several storage technologies. Recent operational data may be kept in a fast store, while older records move to economical object storage. The important skill is not memorizing brand names but recognizing the relationship between data characteristics, query patterns and operational constraints.

> Storage design is part of analytics design. If records are stored without the identity, timestamp and units needed for later analysis, the problem cannot be solved merely by buying a faster database.

5.2.7 Cloud Platforms and Scalable Analytics

Cloud computing allows organizations to provision storage and processing resources without owning the entire physical infrastructure. This is attractive for IoT because device populations and workloads may change over time. A cloud architecture can scale resources upward during heavy workloads and reduce them when demand falls, although scaling still requires careful design and cost control.

SGOU

A cloud-based IoT analytics platform may include device connectivity, message ingestion, databases, object storage, stream processing, batch processing and dashboards. Services can be combined according to the application. A small deployment might use only a few managed services; a large enterprise platform may have separate environments for operational data, historical analytics and machine learning.

Cost deserves attention from the beginning. Sending every raw reading to a remote service may be unnecessary when simple filtering or aggregation can occur near the source. Retaining all data forever may also create avoidable expense. Retention policies should distinguish raw data, cleaned data and derived summaries, and should state how long each category is needed.

Cloud responsibility is shared. A provider may manage physical infrastructure, but the application owner still has responsibilities for access control, credentials, encryption choices, data classification, monitoring and service configuration. Cloud convenience does not remove the need for sound security and governance.

Note : Use cloud elasticity where it creates value, but reduce unnecessary data movement before paying to store and process information that will never be used.

5.2.8 Building a Scalable IoT Data Pipeline

A scalable pipeline connects device observations to analytical results through a sequence of controlled stages. A useful conceptual pipeline is ingestion, validation, transformation, storage, processing and presentation. Each stage should have a clear responsibility and a defined failure behaviour.

Transformation may include unit conversion, timestamp normalization, aggregation, filtering and enrichment. For example, different devices may report temperature in Celsius or Fahrenheit. A normalization step can convert them to one standard before analysis. Enrichment can attach location, device type or maintenance information. The pipeline should preserve enough metadata to make results traceable.

Observability is essential. Operators need to know whether messages are arriving, whether processing is delayed, whether storage is filling and whether data quality has deteriorated. Metrics such as ingestion rate, processing latency, error rate and backlog help reveal problems before users notice them.

Table 5.2.5 Controls across an IoT analytics pipeline

Pipeline stage	Main question	Example control
Ingestion	Are records arriving?	Message rate and connection status
Validation	Are records plausible?	Range and schema checks
Transformation	Are units and time consistent?	Normalization
Storage	Can data be retrieved reliably?	Retention and indexing

SGOU

Processing	Is computation keeping up?	Latency and backlog
Presentation	Can users act on results?	Dashboard and alerts

Scalability is therefore more than processing power. It includes throughput, reliability, maintainability, data quality and the ability to explain what the platform is doing.

5.2.9 Data Governance and Mini Case Study

Data governance provides the rules that make data understandable, controlled and responsibly used. In an IoT platform, governance can include ownership, access rights, retention periods, naming conventions, quality rules and audit records. These practices matter because sensor data may reveal sensitive information about locations, equipment or patterns of activity.

Consider an industrial plant with vibration sensors, temperature probes and production counters. The ingestion layer receives events from the devices. A message broker distributes them to a storage service, a real-time dashboard and an anomaly-detection pipeline. A distributed processing engine produces daily summaries and joins sensor data with maintenance records. Operators receive alerts when unusual patterns appear.

The architecture must answer practical questions. Who can see raw vibration data? How long should it be retained? What happens if the broker is unavailable? How are duplicated messages handled? Which records are allowed into the machine-learning dataset? These questions are not separate from analytics; they determine whether the analytical results can be trusted.

Case Insight

A technically scalable pipeline is still incomplete if nobody can explain where the data came from, who is allowed to use it, how long it is retained, and what happens when a component fails.

The case also illustrates a useful division of labour. Edge devices may perform immediate filtering. Brokers handle ingestion and distribution. Storage systems preserve history. Distributed engines perform large-scale computation. Dashboards and applications present results. Governance and monitoring operate across the entire pipeline.

Recap

- IoT becomes a big-data workload through combinations of volume, velocity, variety, veracity and value.
- Distributed storage spreads data across nodes, while distributed processing divides computation among workers.

SGOU

- MapReduce organizes distributed batch computation into map and reduce stages.
- Apache Spark supports distributed batch, interactive and streaming workloads and can keep intermediate data in memory when appropriate.
- Message brokers decouple IoT producers from analytical consumers and help manage ingestion.
- Relational, time-series, NoSQL and object-storage approaches serve different IoT access patterns.
- Cloud platforms provide elastic resources but still require cost, access-control, security and retention planning.
- A scalable pipeline includes ingestion, validation, transformation, storage, processing and presentation.
- Observability measures such as rate, latency, errors and backlog are essential for operating a large platform.
- Data governance makes analytical data understandable, controlled, traceable and responsibly used.

Objective Type Questions

1. Why can IoT become a big-data workload?
2. What is the main idea of distributed storage?
3. What is the purpose of parallel processing?
4. Which programming model uses map and reduce stages?
5. What is Apache Spark used for?
6. Why is a message broker useful in an IoT platform?
7. Which database style is well suited to time-stamped sensor measurements?
8. Why can a cloud platform be attractive for IoT analytics?
9. What is data enrichment?
10. Why is monitoring important in a scalable pipeline?

SGOU

Answers

1. Large device populations, high sampling rates and long retention can generate large and fast datasets.
2. It divides data across multiple nodes.
3. It divides a large computation into tasks that can run concurrently.
4. MapReduce.
5. Distributed data processing and analytics.
6. It decouples producers from consumers and distributes events.
7. A time-series database.
8. It provides flexible, scalable storage and processing resources.
9. Adding useful contextual information such as location or device type.
10. It reveals delays, failures, backlogs and quality problems.

Assignments

1. Explain why an IoT platform may need distributed storage and processing rather than a single server.
2. Describe the MapReduce model with a suitable IoT example.
3. Compare Hadoop-style batch processing and Apache Spark in terms of their processing approach and suitable workloads.
4. Explain how a message broker improves the architecture of a multi-consumer IoT system.
5. Compare relational, time-series, NoSQL and object storage for IoT applications.
6. Discuss the main cost and governance considerations of a cloud-based IoT analytics platform.
7. Describe a scalable IoT data pipeline and explain the responsibility of each stage.
8. Prepare a conceptual architecture for an industrial IoT platform and justify the placement of ingestion, storage, processing and analytics.

SGOU

Suggested Reading

1. White, Tom. Hadoop: The Definitive Guide. 4th ed., O'Reilly Media, 2015.
2. Karau, Holden, et al. Learning Spark: Lightning-Fast Data Analytics. O'Reilly Media, 2015.
3. Kleppmann, Martin. Designing Data-Intensive Applications. O'Reilly Media, 2017.
4. Buyya, Rajkumar, and Amir Vahid Dastjerdi, editors. Internet of Things: Principles and Paradigms. Morgan Kaufmann, 2016.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A Survey." Computer Networks, vol. 54, no. 15, 2010, pp. 2787–2805.
3. Kleppmann, Martin. Designing Data-Intensive Applications. O'Reilly Media, 2017.
4. Karau, Holden, et al. Learning Spark: Lightning-Fast Data Analytics. O'Reilly Media, 2015.

SGOU



Edge Streaming Analytics

Learning Outcomes

After completion of this unit, learner will be able to:

- explain why analytics may be performed close to IoT data sources
- distinguish edge, fog and cloud processing
- describe event streams, event time and windowing
- explain filtering, aggregation, feature extraction and event detection
- discuss resource, reliability and security considerations in edge analytics

Prerequisite

Picture a machine whose vibration suddenly rises. If the machine needs to be stopped quickly, sending every sample to a distant cloud, waiting for a response and then returning a command can introduce unnecessary delay. A nearby gateway can detect the abnormal pattern and act locally while still sending selected information to a central platform. You already know the roles of sensors, gateways and networks; this unit builds on them and asks a practical question: where should computation happen when time matters?

Keywords

Edge computing, Streaming analytics, Event stream, Windowing, Fog computing, Event time, Real-time processing, Edge intelligence

SGOU

Discussion

"The fastest network is still not instantaneous. When the decision deadline is very short, moving computation closer to the source can be more effective than trying to move data faster."

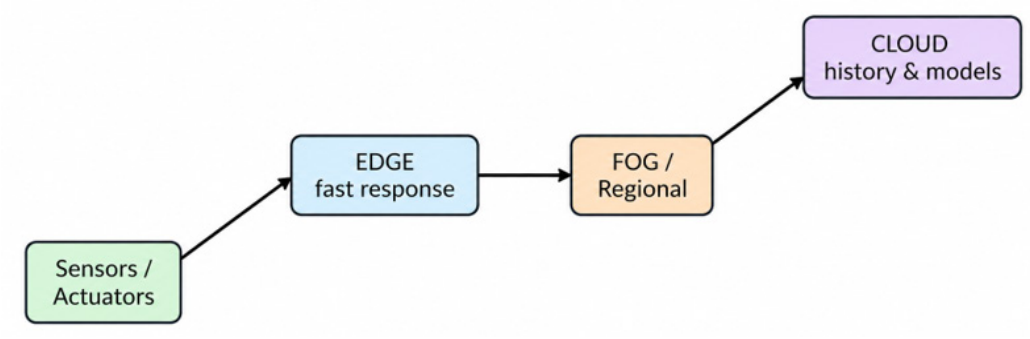


Fig. 5.3.1 Typical edge–fog–cloud placement

Edge computing places some computation near the physical source of data. The edge may be a gateway, industrial computer, smart camera or capable device. The purpose is not simply to reduce cloud usage. Local processing can reduce latency, save bandwidth, continue during connectivity interruptions and keep selected information within a local environment.

Streaming analytics treats data as a flow of events arriving over time. Instead of waiting for a complete dataset, the system continuously filters, transforms, aggregates and evaluates incoming records. Edge streaming combines these ideas: selected analytical work is performed close to the source while other work remains centralized.

5.3.1 Why Analytics moves toward the Edge ?

IoT devices often produce data at the same place where a decision must be made. A safety sensor may need to trigger a shutdown. A smart camera may need to detect a person entering a restricted area. A vehicle may need to react to an obstacle before a remote service can respond. In such cases, latency becomes an architectural concern.

Bandwidth is another reason. A video camera can produce much more data than is necessary for a simple event such as "person detected." An edge processor can examine the stream locally and transmit only event metadata or selected evidence. This reduces network traffic and can also lower storage requirements.

Resilience can improve because local decisions do not always depend on continuous cloud connectivity. A factory may continue a safe operating rule while its connection to a central service is temporarily unavailable. The design should, however, make clear which decisions are allowed locally and which require central coordination.

Privacy can also influence placement. Processing information locally may reduce the need to transmit raw data. This is not a guarantee of privacy, because the edge device itself must still be secured and the information may still be sensitive. Placement is therefore one part of a broader security and governance strategy.

SGOU

Design Question

For every analytical task, ask: How quickly is the result needed? How much data is produced? What happens if the network fails? Is the raw data sensitive? These questions help determine where processing belongs.

5.3.2 Edge, Fog and Cloud

Edge, fog and cloud describe different positions in a distributed architecture. Edge refers to computation close to the data source. Fog is commonly used for an intermediate layer between edge devices and centralized cloud resources. Cloud computing provides centralized and scalable storage and processing. The three can work together rather than compete.

Consider a smart-city traffic system. A camera at an intersection may detect vehicles locally. A nearby regional node may combine information from several intersections to estimate congestion. A cloud platform may store historical data and analyze city-wide patterns. Each layer performs work appropriate to its latency, scale and information scope.

Table 5.3.1 Comparing processing locations

Layer	Typical role	Strength
Edge	Immediate filtering, detection, control	Low latency
Fog / regional	Combine nearby sources, coordinate local services	Local aggregation
Cloud	Historical storage, fleet-wide analysis, model training	Scale and broad context

The choice should be driven by the decision requirement. A small sensor may need only a gateway rule. A safety-critical machine may require substantial local inference. A strategic planning application can often tolerate centralized processing. Adding more layers can improve flexibility but also increases the number of components that must be secured, updated and monitored.

"Edge" is a placement concept, not a single product. A gateway, industrial computer or smart device can all act as an edge node if it performs computation near the source.

SGOU

5.3.3 Streaming Data and Event Time

A stream is a sequence of records arriving over time. Each record may carry an event timestamp representing when the physical event occurred. Processing time is different: it is when the analytics system receives or processes the record. The distinction becomes important when networks delay messages.

Suppose a sensor measures vibration at 10:00:01, but the message reaches the processing system at 10:00:05. If the system uses only processing time, it may place the record in the wrong interval. Event-time processing preserves the time of the original observation and allows the system to reason about late arrivals.

Streaming systems therefore need rules for ordering, lateness and state. Some applications can tolerate a few seconds of delay. Others require immediate action and may use simplified processing. There is no universal setting; the acceptable delay depends on the operational requirement.

Mini Example

A bus-location message carries event time 08:15:10 but arrives at 08:15:14. The four-second difference matters if the system is calculating arrival statistics for narrow time windows.

A well-designed stream pipeline defines how timestamps are interpreted, how much lateness is accepted and what happens when an event arrives too late. These decisions affect the correctness of aggregates, alerts and dashboards.

5.3.4 Stream Processing and Windows

An IoT stream is often unbounded, so an analytics system cannot wait for the "end" of the data. Windows provide a practical way to analyze manageable portions of the stream. A tumbling window divides time into fixed, non-overlapping intervals. A sliding window moves continuously and may overlap previous intervals. Session windows group events according to periods of activity.

Table 5.3.2 Common streaming windows

Window type	Basic idea	Example
Tumbling	Fixed, non-overlapping intervals	Average temperature every 1 minute
Sliding	Fixed length that moves at a chosen step	5-minute average updated every minute
Session	Group events separated by inactivity	Vehicle trip activity
Count-based	Fixed number of records	Average of every 100 readings

SGOU

Windowing allows an edge device to maintain only the state needed for a decision. For example, a gateway can keep the last five minutes of vibration summaries instead of storing every historical reading locally. The central platform can retain the full history if needed.

Window size influences sensitivity. A very short window may react quickly but also be affected by noise. A longer window can smooth fluctuations but may delay detection. The correct choice depends on the physical process and the cost of false alarms and missed events.

> Windowing is not merely a programming trick. It is a way of expressing what amount of recent context is relevant to the decision.

5.3.5 Filtering, Aggregation and Feature Extraction

Filtering removes records that are irrelevant to the immediate task. A gateway may discard readings that are unchanged within a useful tolerance or forward only events that cross a meaningful threshold. Filtering reduces traffic and processing load, but it must be designed carefully so that important information is not lost.

Aggregation summarizes several observations. Average, minimum, maximum, count and standard deviation are common examples. An industrial gateway might calculate one-second vibration statistics from high-frequency samples and send the summary upstream. Aggregation reduces data volume while retaining useful characteristics of the signal.

Feature extraction transforms raw measurements into indicators that an analytical model can use. For vibration, features might include average amplitude, variance or the energy within a frequency band. For a vehicle, features might include recent speed variability, acceleration events and distance travelled. Features should be selected because they carry information relevant to the decision.

Table 5.3.3 Common edge-stream operations

Operation	Purpose	Edge example
Filtering	Reduce irrelevant records	Send only threshold crossings
Aggregation	Summarize observations	One-minute average
Feature extraction	Represent useful signal characteristics	Vibration variance
Compression	Reduce representation size	Compact event record

SGOU

Practical Principle

Reduce data only when you understand what information the decision requires. A smaller message is not automatically a better message.

5.3.6 Event Detection and Real-Time Decisions

Event detection identifies patterns that require attention. The simplest approach is a threshold: if temperature exceeds a limit, raise an alert. More robust systems may require several conditions, persistence over time or a combination of signals. These safeguards reduce false alarms caused by brief noise.

Suppose a pump normally operates between 40 and 60 °C. A single reading of 61 °C may not justify a shutdown. If temperature remains above the limit while vibration and current also rise, the evidence becomes stronger. A rule can therefore require persistence and corroboration before triggering a high-priority action.

Real-time systems also need alert management. Repeating the same alarm every second can overwhelm an operator. Debouncing, cooldown periods and event correlation can turn many low-level records into one meaningful incident. The system should distinguish between detection, notification and control: detecting a condition does not always mean that an automatic action is safe.

For high-consequence control, analytics should be combined with explicit safety logic and fail-safe behaviour. A statistical signal alone should not be treated as an unquestionable command.

The quality of a real-time decision depends on latency, accuracy, context and reliability. A highly accurate model that responds too slowly may be useless for a fast event. Conversely, a very fast rule that generates frequent false alarms may cause operators to ignore genuine warnings.

5.3.7 Resource, Reliability and Security Considerations

Edge devices usually have fewer computing, memory, storage and power resources than centralized servers. An algorithm that is comfortable in a cloud environment may be too heavy for a small gateway. Edge analytics therefore requires attention to model size, processing time, memory use and energy consumption.

Reliability is equally important. A remote site may lose network connectivity, restart after a power interruption or operate for long periods without physical access. Local software should recover predictably, preserve necessary state and fail safely. Graceful degradation means that useful and safe functions continue even when some services are unavailable.

SGOU

Security risks increase when devices are physically accessible. An attacker may attempt to modify software, steal credentials or interfere with stored data. Secure boot, signed updates, strong authentication, least-privilege access and encrypted communication are common protective measures. Remote management must be designed so that updates can be applied without creating a larger security problem.

Table 5.3.4 Edge deployment considerations

Concern	Typical question
Resources	Can the device run the algorithm within its CPU and memory limits?
Power	How much energy does continuous processing consume?
Reliability	What happens during network or power loss?
Security	Can an unauthorized person modify the device?
Maintenance	How will software be updated and monitored remotely?

An edge architecture is successful only when its local intelligence remains manageable over the full life of the device, not merely when the first prototype works.

5.3.8 Model Inference at the Edge and Case Study

Edge inference means running a trained analytical or machine-learning model close to the data source. Training may still occur centrally because training can require substantial historical data and computing resources. After training, a compact model can be deployed to gateways or devices for fast local predictions.

Consider predictive maintenance for a motor. A sensor measures vibration and temperature at high frequency. The edge device calculates selected features and feeds them into a trained model. If the resulting risk score crosses a carefully chosen threshold, the device can raise a local warning and send a summary to the central platform. The cloud can retain the evidence, compare the machine with similar assets and improve the model later.

The architecture separates fast local action from long-term learning. It also limits the amount of raw high-frequency data that must cross the network. However, model drift remains possible. Machine behaviour can change after maintenance, component replacement or a change in operating conditions. Monitoring the model and its inputs is therefore necessary.

SGOU

Case Insight

The best edge design is often hybrid: local processing handles time-sensitive decisions, while centralized systems provide history, fleet-wide comparison, governance and model improvement.

This hybrid approach reflects a broader principle of distributed intelligence. Computation should be placed where it provides the best combination of latency, data availability, reliability, privacy, cost and maintainability. No single location is ideal for every task.

Recap

- Edge analytics places selected computation close to the data source.
- Edge, fog and cloud can form a cooperative architecture rather than mutually exclusive choices.
- Streaming analytics processes records continuously and must account for event time and late arrivals.
- Windows make an unbounded stream manageable for continuous analysis.
- Filtering reduces irrelevant data; aggregation summarizes observations; feature extraction preserves useful signal characteristics.
- Real-time event detection should account for noise, persistence, repeated alerts and operational context.
- Edge devices face tighter resource and reliability constraints than centralized systems.
- Security and remote management are essential because edge devices may be physically accessible and geographically distributed.
- Edge inference runs trained models locally, while centralized systems can support training, history and fleet-wide analysis.
- Hybrid edge–cloud architectures balance response time with broader analytical context.

Objective Type Questions

1. What is the main reason for moving analytics toward the edge?
2. Which layer is closest to the physical data source?

SGOU

3. What is an event stream?
4. What is event time?
5. Which window uses fixed non-overlapping intervals?
6. What operation summarizes several observations?
7. What is edge inference?
8. Why is local processing useful when connectivity is unreliable?
9. Why are cooldown periods useful in alerting?
10. Name one important security concern for physically accessible edge devices.

Answers

1. Lower latency and local decision-making.
2. Edge.
3. A sequence of records arriving over time.
4. The timestamp associated with the physical event.
5. Tumbling window.
6. Aggregation.
7. Running a trained model locally.
8. Local decisions can continue during network outages.
9. They reduce repeated alerts for the same incident.
10. Device compromise or unauthorized modification.

Assignments

1. Explain the reasons for performing analytics at the edge.
2. Compare edge, fog and cloud processing using a smart-city example.
3. Explain event time, processing time and the problem of late-arriving events.

SGOU

4. Describe tumbling, sliding, session and count-based windows with suitable IoT examples.
5. Explain how filtering, aggregation and feature extraction can reduce network traffic.
6. Discuss the design considerations for a real-time IoT alerting system.
7. Explain resource, reliability, security and remote-management challenges at the edge.
8. Prepare a conceptual edge-streaming architecture for predictive maintenance and explain the role of each layer.

Suggested Reading

1. Shi, Weisong, and Schahram Dustdar. "The Promise of Edge Computing." Computer, vol. 49, no. 5, 2016, pp. 78–81.
2. Cao, Weisong, et al. "Edge Computing: Vision and Challenges." IEEE Internet of Things Journal, vol. 3, no. 5, 2016, pp. 637–646.
3. Buyya, Rajkumar, and Amir Vahid Dastjerdi, editors. Internet of Things: Principles and Paradigms. Morgan Kaufmann, 2016.
4. Karau, Holden, et al. Learning Spark: Lightning-Fast Data Analytics. O'Reilly Media, 2015.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A Survey." Computer Networks, vol. 54, no. 15, 2010, pp. 2787–2805.
3. Kleppmann, Martin. Designing Data-Intensive Applications. O'Reilly Media, 2017.
4. Karau, Holden, et al. Learning Spark: Lightning-Fast Data Analytics. O'Reilly Media, 2015.

SGOU



Network Analytics

Learning Outcomes

After completion of this unit, learner will be able to:

- explain the purpose of network analytics in an IoT environment
- identify important network performance and telemetry measures
- describe flow and topology analysis for IoT networks
- explain QoS, anomaly detection and security analytics
- describe how network analytics supports capacity planning and operational decisions

Prerequisite

Think about a campus with smart lights, access systems, environmental sensors, cameras and connected laboratory equipment. When everything works, the network is almost invisible. When one gateway becomes overloaded, packets are lost or a device begins contacting an unusual destination, the effect can spread across applications. You already know that IoT devices communicate through networks; this unit turns the network itself into a source of evidence. By observing traffic, topology, latency and failures, an operator can understand not only whether the network is working, but why it behaves as it does.

Keywords

Network telemetry, Latency, Packet loss, Throughput, Flow analysis, Topology, QoS, Anomaly detection

SGOU

Discussion

"A device may be healthy while the network path to it is unhealthy. Network analytics helps separate device problems from communication problems."

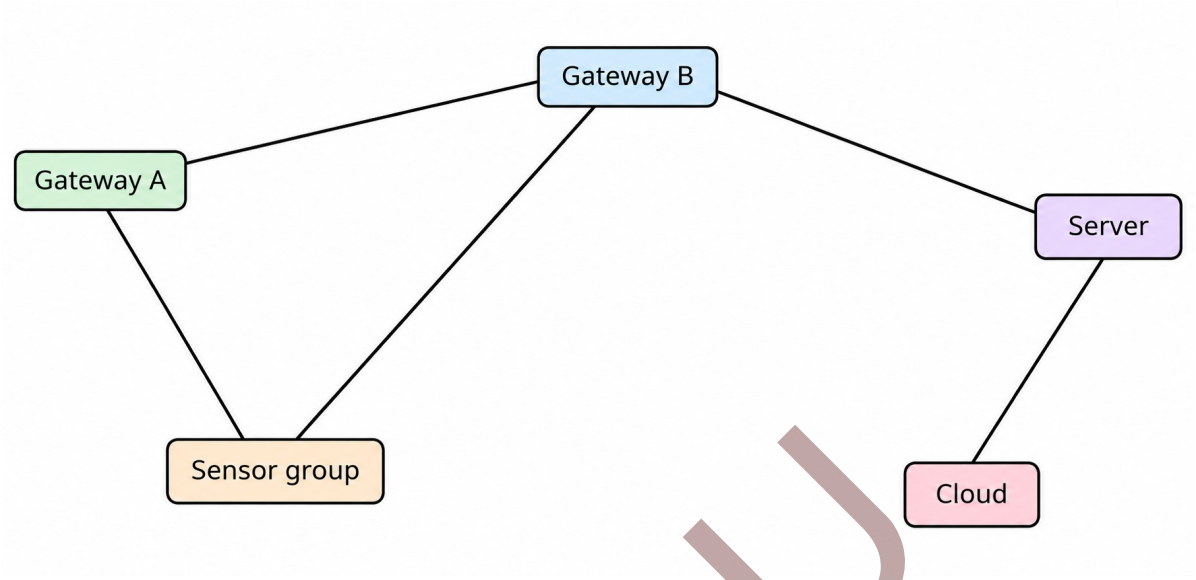


Fig. 5.4.1 A small IoT network viewed through analytics

Network analytics uses measurements and communication records to understand how an IoT network behaves. The purpose may be troubleshooting, performance optimization, reliability improvement, security monitoring or capacity planning. Because IoT deployments often contain constrained devices and shared wireless links, network conditions can strongly affect application behaviour.

The central idea is observation with context. A latency increase is useful evidence, but its meaning depends on where it occurs, when it occurs and which applications are affected. Combining performance metrics with flows, topology and device information produces a more complete operational picture.

5.4.1 Network Telemetry and Performance

Network telemetry refers to measurements that describe communication behaviour. Common measures include latency, packet loss, throughput, utilization, retransmissions, connection counts and error rates. These measurements help operators understand whether a network is carrying traffic efficiently and reliably.

Latency is the time taken for information to travel between endpoints or for a request to receive a response. Packet loss indicates that packets fail to reach their destination successfully. Throughput describes the amount of data carried per unit time. Utilization indicates how much of an available resource is being used. These measures are related but not interchangeable.

SGOU

Table 5.4.1 Basic network analytics measures

Measure	Meaning	Why it matters
Latency	Communication delay	Time-sensitive control
Packet loss	Unsuccessful packet delivery	Reliability and retries
Throughput	Data carried per unit time	Capacity and performance
Utilization	Resource usage level	Congestion risk
Retransmission	Repeated delivery attempts	Wireless/link quality

A useful dashboard should avoid presenting metrics without context. For example, 60 percent link utilization may be healthy for one application and problematic for another. Likewise, a small packet-loss rate may be tolerable for periodic environmental sensing but unacceptable for a control loop. Network analytics becomes meaningful when measurements are connected to application requirements.

> The question is not simply "Is the network busy?" It is "Is the network providing the level of service required by the applications using it?"

5.4.2 Flow Analysis

A network flow is a summary of communication between endpoints over a period. Instead of examining every packet, flow records can describe source, destination, protocol, volume and timing information. Flow analysis is useful for understanding who is communicating with whom and how much traffic is being exchanged.

In an IoT environment, flow information can reveal unexpected communication patterns. A temperature sensor that normally sends small messages to one gateway may suddenly transmit large volumes of data to an unfamiliar destination. The event does not prove compromise, but it is a useful signal for investigation.

Flow analysis also supports performance troubleshooting. If a gateway is overloaded, flows can show which device groups or applications contribute most of the traffic. An operator can then examine whether the traffic is expected, whether messages are being duplicated, or whether a configuration change has created unnecessary communication.

Practical Example

A video-enabled security system and hundreds of low-rate environmental sensors share one wireless network. Flow analysis can show that a small number of video streams consume most of the capacity, explaining why sensor messages are delayed.

SGOU

Flow records are especially useful when full packet capture would be too expensive or inappropriate. They provide a higher-level view that can be retained and analyzed over longer periods. For deeper investigation, packet-level evidence may still be necessary, but flow analytics is often an efficient first step.

5.4.3 Topology and Dependency Analysis

Topology describes how network components are connected. In an IoT deployment, the topology may include devices, gateways, access points, routers, servers and cloud endpoints. Dependency analysis goes a step further by asking which applications depend on which components.

Suppose several classrooms lose sensor connectivity at the same time. If the affected devices all connect through one gateway, the gateway becomes a strong candidate for investigation. If devices on different gateways fail simultaneously but share a common upstream router, the shared dependency may explain the event. Topology turns a large collection of alarms into a smaller set of plausible failure points.

Table 5.4.2 Using topology to narrow a fault

Observation	Possible inference	Next step
One device offline	Device or local link problem	Check device and local connection
Many devices on one gateway offline	Gateway or access problem	Inspect gateway
Several gateways affected	Shared upstream dependency	Inspect router or backbone
Network healthy, application failing	Higher-layer problem	Check service or data pipeline

Topology analytics is also useful for resilience planning. A component that serves many critical devices may represent a concentration of risk. Redundancy, alternate paths and failover arrangements can then be considered. Thus topology analysis supports both troubleshooting and design decisions.

> The most important node is not always the busiest node. A quiet component that sits on many critical paths may be a much more important dependency.

5.4.4 QoS Analytics

Quality of Service, or QoS, concerns how network resources are provided to different types of traffic. IoT deployments may carry safety-related control messages, ordinary telemetry, firmware updates, video and administrative traffic at the same time. Treating every packet identically may not provide the service level required by the most time-sensitive applications.

SGOU

QoS analytics compares observed network behaviour with application requirements. A control message may require low latency and low loss. A historical sensor upload may tolerate delay. Video may require sustained throughput. Analytics can therefore identify whether the network is meeting the service expectations associated with each traffic class.

Table 5.4.3 Illustrative QoS requirements

Traffic type	Likely priority need	Useful metric
Safety/control	Very low delay, high reliability	Latency and loss
Telemetry	Reliable delivery	Loss and throughput
Video	Sustained throughput	Throughput and jitter
Firmware update	High volume but delay tolerant	Throughput and completion time

QoS is not merely a configuration exercise. Conditions change as device populations grow, applications are added and usage patterns shift. Analytics provides evidence about whether the chosen priorities continue to work. If low-priority traffic consumes capacity needed by critical applications, the issue can be detected and addressed.

> A network can be technically fast and still provide poor service if the wrong traffic receives the available capacity at the wrong time.

5.4.5 Network Anomaly Detection

An anomaly is a pattern that differs from expected behaviour. In network analytics, anomalies may appear as unusual traffic volume, unexpected destinations, repeated connection failures, sudden latency changes or a device communicating at an unusual time.

Simple anomaly detection can use thresholds. For example, an alert may be raised when traffic from a gateway exceeds a defined level. More adaptive methods can learn normal behaviour and identify deviations. However, context matters. A traffic spike during a scheduled firmware update is different from an unexplained spike at an unusual time.

False positives are a major practical challenge. If operators receive too many alerts, they may begin to ignore them. Good systems therefore combine signals, include context and assign appropriate severity. A device sending unusual traffic may be suspicious, but the investigation should also check whether a legitimate configuration change occurred.

SGOU

Operational Rule

Anomaly detection should answer two questions: "What is unusual?" and "Why might this be unusual in this context?" The second question reduces unnecessary investigation.

Historical baselines can be built by device, site, application or time of day. A campus gateway may normally be busy during working hours but quiet at night. A single baseline for the entire day would produce misleading alerts. Contextual baselines are therefore often more useful than one global threshold.

5.4.6 Security Analytics

Network security analytics uses communication evidence to identify possible threats or compromised devices. Useful signals include unexpected destinations, unusual connection rates, repeated authentication failures, abnormal traffic volumes and changes in communication patterns.

IoT security is complicated by device diversity. Some devices are powerful computers; others have limited memory and communicate infrequently. A normal pattern for one device type may be abnormal for another. Security analytics should therefore incorporate device identity, role and expected behaviour.

Consider a smart camera that normally communicates with a local video server. If it suddenly begins making repeated outbound connections to an unfamiliar destination, the network event should be investigated. The network evidence can be combined with device logs, identity information and recent maintenance records. This multi-source view is stronger than relying on one indicator.

Table 5.4.4 Security analytics signals and context

Signal	Possible interpretation	Context to check
Unexpected destination	Possible compromise or misconfiguration	Recent software update
Traffic spike	Large transfer or attack	Scheduled backup/ update
Repeated login failures	Credential issue or attack	Administrative activity
New communication peer	New service or compromise	Configuration change

An analytical signal is not automatically proof of an attack. It is evidence that should be interpreted, correlated and investigated.

SGOU

5.4.7 Capacity Planning

Capacity planning uses historical and expected demand to determine whether network resources will remain sufficient. IoT growth can be gradual, but small devices can accumulate into a significant load. New cameras, additional telemetry points, higher sampling rates or software updates can change traffic patterns.

Planning should consider more than bandwidth. Wireless airtime, gateway CPU and memory, routing capacity, broker limits, storage, power and upstream links can all become bottlenecks. A network may have enough nominal bandwidth while a gateway is overloaded by connection management or message processing.

Historical analytics can reveal growth trends and recurring peaks. Forecasting can estimate when a resource may cross an operational threshold. Scenario analysis can then ask what happens if the number of devices increases by a given amount or if sampling frequency doubles. The purpose is to identify constraints before they become outages.

Practical Example

A campus plans to add 2,000 environmental sensors. Capacity planning should estimate not only their average traffic but also reconnection bursts, firmware updates, gateway processing and peak periods.

Capacity decisions should be tied to service objectives. If critical traffic requires a latency target, the plan should consider the resource level needed to preserve that target under expected load. This makes capacity planning an operational activity rather than a simple hardware-counting exercise.

5.4.8 Integrated Campus IoT Case

Imagine a university campus with smart lighting, room-temperature sensors, access-control devices, cameras and laboratory monitoring equipment. Network analytics collects telemetry from gateways and access points, records flow summaries and maintains a topology view. The operations team wants to reduce outages while identifying unusual device behaviour.

During one afternoon, latency increases in several buildings. Telemetry shows higher utilization on a shared wireless segment. Flow analysis reveals that several video streams are consuming unusually high throughput. Topology analysis shows that the affected buildings share the same upstream path. QoS analytics indicates that environmental telemetry is receiving less timely service than expected.

The immediate response is to investigate the video traffic and protect the required service level for critical telemetry. Security analytics can run in parallel to check whether any device is sending unexpected traffic. After the incident, capacity analysis can determine whether the network needs additional resources or whether application configuration should change.

SGOU

Case Insight

Performance, flow, topology, QoS and security analytics answer different questions. Their combination provides a much stronger explanation of an incident than any one metric.

The case also illustrates the value of history. If similar utilization spikes occur every Monday afternoon, the problem may be predictable and manageable. If the pattern appears suddenly and only once, a different investigation may be appropriate. Network analytics therefore supports both immediate response and long-term improvement.

Recap

- Network analytics uses telemetry and communication evidence to understand IoT network performance, reliability and behaviour.
- Latency, packet loss, throughput, utilization and retransmission describe different aspects of network performance.
- Flow analysis summarizes communication patterns and can support both troubleshooting and security investigation.
- Topology and dependency analysis help identify shared points of failure and critical paths.
- QoS analytics connects network behaviour with the service requirements of different applications.
- Anomaly detection identifies deviations from expected behaviour and should use context to reduce false positives.
- Security analytics can correlate unexpected destinations, connection patterns and device context to identify possible threats.
- Capacity planning should consider bandwidth as well as wireless, gateway, processing and upstream constraints.
- Historical patterns and forecasts can help organizations act before capacity becomes an outage.
- Combining several analytical views produces a more reliable basis for operational decisions.

Objective Type Questions

1. What does network telemetry provide?
2. Which measure describes communication delay?

SGOU

3. Which measure indicates unsuccessful packet delivery?
4. What does throughput measure?
5. What does a network flow summarize?
6. What does topology analysis examine?
7. What is the purpose of QoS analytics?
8. What is an anomaly in network analytics?
9. Why should capacity planning consider more than bandwidth?
10. What is the benefit of combining performance, flow, topology and security analytics?

Answers

1. Measurements describing network behaviour.
2. Latency.
3. Packet loss.
4. Data successfully carried per unit time.
5. A summary of communication between endpoints.
6. Connections and dependencies.
7. Matching network treatment to application needs.
8. A pattern that differs from expected behaviour.
9. Processing, wireless and gateway resources can also become bottlenecks.
10. It provides a more complete operational picture.

Assignments

1. Explain the role of network telemetry in operating an IoT environment.
2. Differentiate latency, packet loss, throughput and utilization with suitable examples.
3. Describe how flow analysis can support both performance and security investigations.

SGOU

4. Explain how topology and dependency analysis help diagnose IoT network failures.
5. Discuss QoS analytics using an example of mixed critical and non-critical IoT traffic.
6. Explain how anomaly detection can be used in an IoT network and why context matters.
7. Discuss security analytics signals that may indicate a compromised IoT device.
8. Prepare a network-analytics plan for a campus IoT deployment, including telemetry, analysis, alerts and capacity planning.

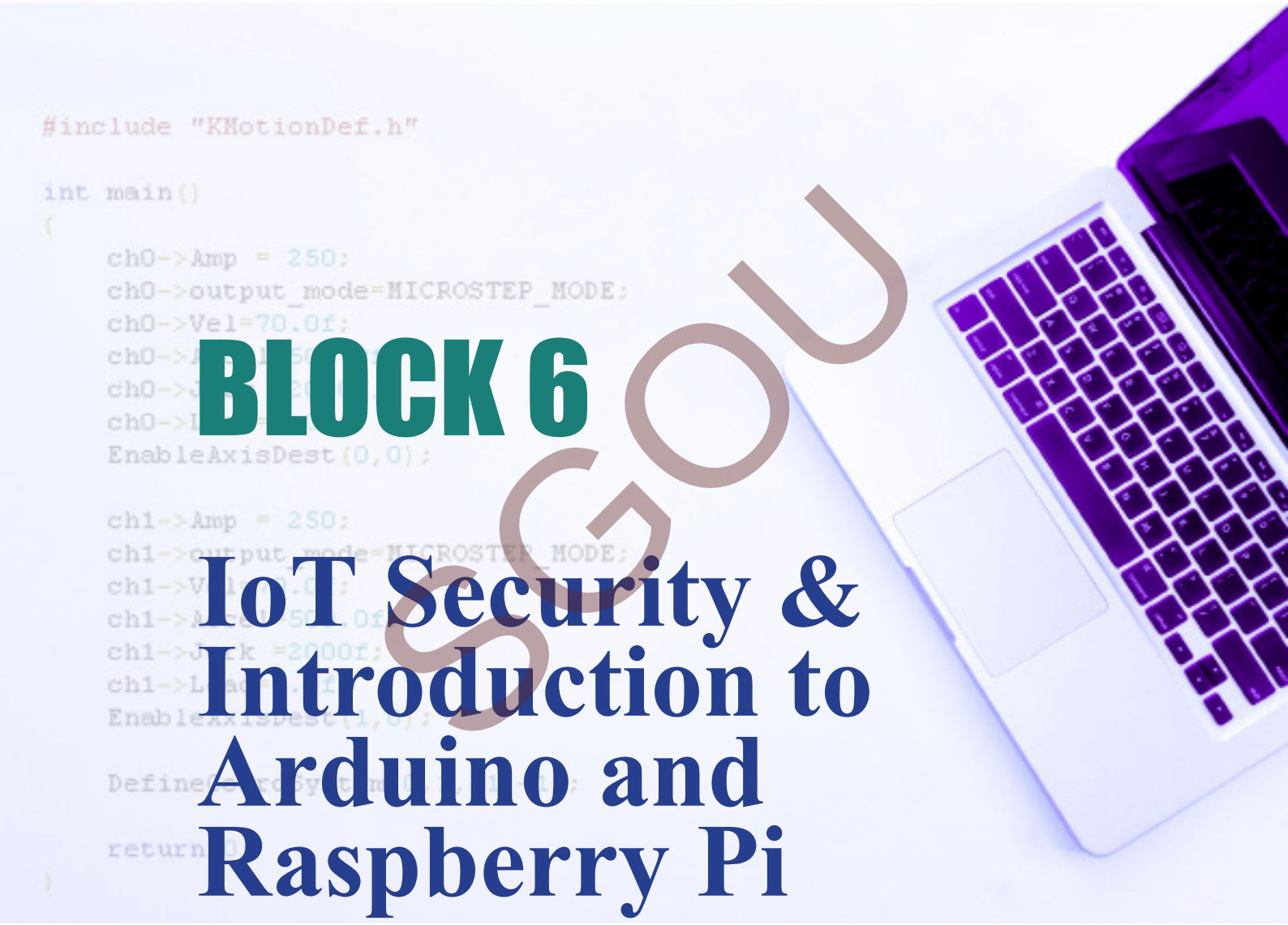
Suggested Reading

1. Odom, Wendell. CCNA 200-301 Official Cert Guide, Volume 1. Cisco Press, 2020.
2. Stallings, William. Data and Computer Communications. 10th ed., Pearson, 2013.
3. Buyya, Rajkumar, and Amir Vahid Dastjerdi, editors. Internet of Things: Principles and Paradigms. Morgan Kaufmann, 2016.
4. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A Survey." Computer Networks, vol. 54, no. 15, 2010, pp. 2787–2805.
3. Kleppmann, Martin. Designing Data-Intensive Applications. O'Reilly Media, 2017.
4. Karau, Holden, et al. Learning Spark: Lightning-Fast Data Analytics. O'Reilly Media, 2015

SGOU



BLOCK 6

IoT Security & Introduction to Arduino and Raspberry Pi

SGOU



Common Challenges in Operational Technology (OT) Security

Learning Outcomes

After completion of this unit, learner will be able to:

- define Operational Technology (OT) and OT Security
- Identify the major Vulnerabilities in OT environments.
- explain the differences between Information Technology (IT) and Operational Technology (OT)
- recognize common cyber threats affecting industrial control systems
- describe basic security practices used to protect OT infrastructures

Prerequisite

Imagine that you are travelling by metro rail. Every train arrives on time, the automatic doors open and close correctly, signalling systems guide the trains safely, and electricity is continuously supplied. Have you ever wondered how these systems operate without interruption?

Behind the scenes, thousands of industrial devices continuously monitor sensors, control machines, and communicate with one another. These systems belong to Operational Technology (OT). Unlike ordinary computer systems used for office work, OT systems directly control physical equipment such as power plants, factories, transportation systems and water treatment plants.

Now imagine what would happen if a cyber attacker manipulated the control system of a power station or altered the signals controlling railway traffic. Such attacks could affect not only computers but also human safety and critical national infrastructure.

In this unit, we will explore why securing Operational Technology has become one of the most important responsibilities of modern engineers.

Before studying this unit, you should have a basic understanding of:

- Computer networks
- Internet of Things (IoT)

SGOU

- Sensors and actuators
- Basic cybersecurity concepts

Do not worry if some of these ideas are new to you. We will explain every concept step by step.

Keywords

Operational Technology (OT), Industrial Control System (ICS), SCADA, PLC, Cyber Attack

Discussion

6.1.1 Introduction to Operational Technology

When we hear the word technology, we often think about computers, laptops, smartphones or the Internet. However, many technologies operate silently behind the scenes, controlling the machines and processes that keep our daily lives functioning.

For example:

- Electricity generation in power stations
- Manufacturing in factories
- Railway signalling systems
- Oil and gas pipelines
- Water purification plants
- Smart traffic management systems

These systems are collectively known as Operational Technology (OT).

Operational Technology consists of hardware and software used to monitor and control physical devices, industrial equipment and infrastructure. Unlike Information Technology (IT), whose primary goal is processing and storing information, OT directly interacts with the physical world.

For instance, in a smart manufacturing plant, sensors continuously measure temperature, pressure and machine speed. Controllers analyse this information and automatically adjust industrial equipment to maintain safe and efficient operation. Without OT, modern industries would be unable to automate production or ensure operational safety.

SGOU

Definition

Operational Technology (OT) refers to the hardware and software systems that monitor, control and automate industrial processes and physical devices in critical infrastructure and manufacturing environments. See figure 6.1.1 Operational Technology in Everyday Life

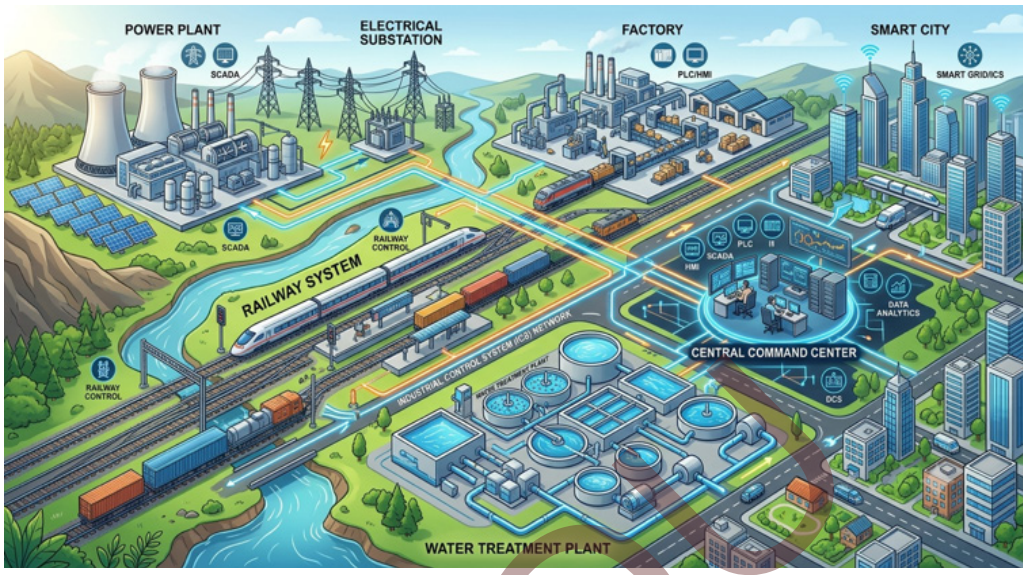


Fig. 6.1.1 Operational Technology supports critical infrastructure such as power generation, manufacturing, transportation and water management.

Example

Consider an automated bottling plant.

- Sensors detect the position of bottles.
- Controllers regulate conveyor belt speed.
- Actuators fill bottles with the required quantity of liquid.
- Cameras inspect product quality.
- The supervisory computer monitors the entire production process.

Every operation is performed automatically using Operational Technology.

Did You Know?

More than 80% of critical infrastructure systems, including electricity generation, water supply and transportation, depend on Operational Technology. Protecting these systems is therefore essential for ensuring public safety and economic stability.

SGOU

6.1.2 Information Technology (IT) versus Operational Technology (OT)

Many learners initially assume that Information Technology (IT) and Operational Technology (OT) are the same because both involve computers and networks. In reality, their objectives and priorities differ significantly.

Information Technology focuses on creating, processing, storing and transmitting information. Operational Technology focuses on controlling physical equipment and industrial processes.

For example, if an office email server becomes unavailable for an hour, employees may experience inconvenience. However, if the control system of a chemical plant fails for even a few minutes, the consequences may include production loss, equipment damage or serious safety hazards.

The fundamental priority of IT is Confidentiality, Integrity and Availability (CIA) of information. In OT environments, the order of priority changes. Safety and Availability are the foremost concerns because interruptions can directly affect human lives and industrial operations.

Table 6.1.1 Comparison between IT and OT

Feature	Information Technology (IT)	Operational Technology (OT)
Primary Objective	Data processing	Control of physical processes
Main Assets	Computers, servers, databases	PLCs, sensors, actuators, industrial controllers
Priority	Confidentiality, Integrity, Availability	Safety, Availability, Reliability
Downtime	Usually acceptable for short periods	Often unacceptable
Environment	Offices, educational institutions	Industries, power plants, transportation systems
Typical Examples	Banking software, ERP, email	SCADA, PLC, DCS, industrial robots

Industry Insight

Modern industries increasingly integrate IT and OT systems to enable real-time monitoring, predictive maintenance and cloud-based analytics. While this integration improves operational efficiency, it also expands the cybersecurity attack surface, making robust OT security indispensable.

SGOU

6.1.3 Need for Operational Technology (OT) Security

In the previous section, we learned that Operational Technology (OT) controls industrial equipment and critical infrastructure such as power plants, manufacturing industries, transportation systems, oil refineries and water treatment facilities. These systems form the backbone of a nation's economy and public services.

For many years, OT systems operated in isolated environments and were not directly connected to external networks. Consequently, cyber threats were relatively limited. However, with the rapid adoption of the Industrial Internet of Things (IIoT), cloud computing, remote monitoring and wireless communication, OT systems have become increasingly interconnected. While this connectivity has improved operational efficiency and enabled intelligent decision-making, it has also introduced significant cybersecurity risks.

Today, cyber attackers no longer target only office computers and corporate databases. They also attempt to compromise industrial control systems because successful attacks can disrupt essential services, damage expensive equipment and endanger human lives.

Therefore, securing OT environments has become a critical responsibility for engineers, cybersecurity professionals and industrial organizations.

6.1.3.1 Why is OT Security Important?

Operational Technology security is essential for several reasons.

Protecting Human Safety

Unlike traditional IT systems, OT directly controls machines and industrial equipment. A cyber attack that manipulates industrial processes may lead to serious accidents.

Example

If an attacker modifies the control logic of a chemical processing plant, the temperature or pressure inside reactors may exceed safe limits, resulting in explosions or toxic gas leakage.

Human safety is therefore the highest priority in OT security.

Ensuring Continuous Operation

Many industrial facilities operate continuously throughout the year.

Examples include:

- Power generation plants
- Oil refineries
- Steel manufacturing units

SGOU

- Water treatment plants
- Telecommunication networks

Unexpected interruptions may cause enormous financial losses.

Protecting Critical Infrastructure

Critical infrastructure includes systems whose failure would seriously affect society.

Examples include:

- Electricity distribution
- Railway signalling
- Air traffic control
- Water supply
- Healthcare services

Cybersecurity helps maintain uninterrupted operation of these essential services.

Preventing Financial Losses

Industrial shutdowns caused by cyber attacks often result in:

- Production delays
- Equipment damage
- Contract penalties
- Recovery expenses
- Loss of customer confidence

Preventive security measures are considerably less expensive than recovering from a successful cyber attack.

Regulatory Compliance

Governments and international organizations require industries to comply with cybersecurity standards and regulations.

Examples include:

- IEC 62443
- NIST Cybersecurity Framework
- ISO/IEC 27001

Compliance improves both security and organizational reputation. See figure 6.1.2 Importance of OT Security

SGOU

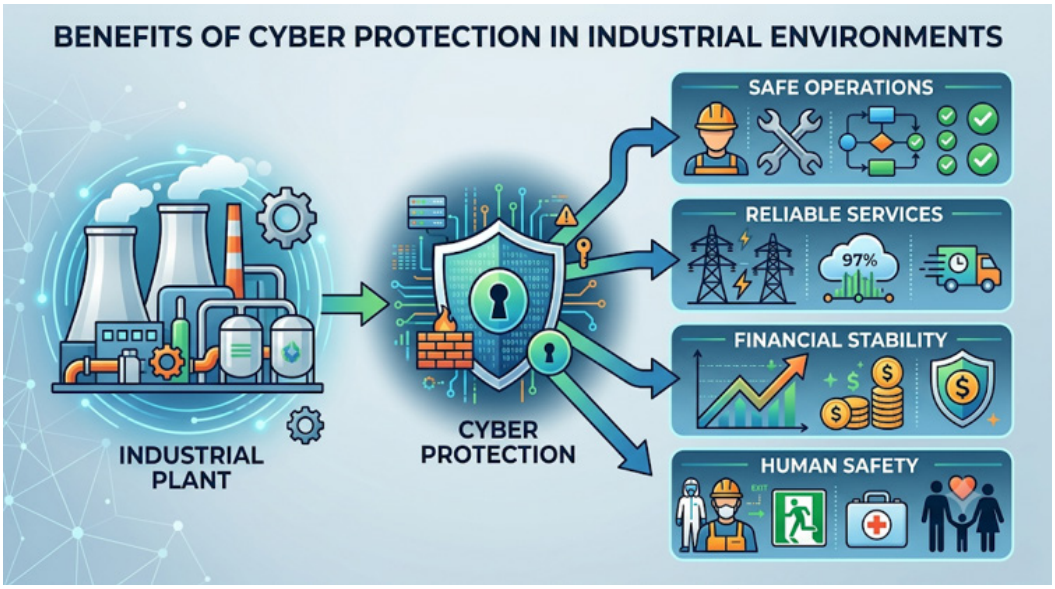


Fig. 6.1.2 Importance of Operational Technology security in protecting industrial systems.

OT Security Protects More Than Computers

When we think about cybersecurity, we usually imagine protecting computers and confidential information. In Operational Technology, cybersecurity also protects machines, industrial equipment, production processes, public utilities and human lives. This makes OT security fundamentally different from conventional IT security.

6.1.4 Industrial Control Systems (ICS)

Industrial operations are managed using specialized control systems collectively known as Industrial Control Systems (ICS).

An Industrial Control System continuously performs three important activities:

1. Collects information from sensors.
2. Processes the collected information.
3. Controls machines using actuators.

ICS forms the operational core of modern industries.

Definition

An Industrial Control System (ICS) is an integrated system of hardware, software, sensors, controllers and communication networks used to monitor and control industrial processes.

SGOU

6.1.4.1 Major Components of ICS

A typical Industrial Control System consists of:

- Sensors
- Controllers
- Human Machine Interface (HMI)
- Communication Network
- Supervisory Computer
- Actuators

See figure 6.1.3 Components of an Industrial Control System

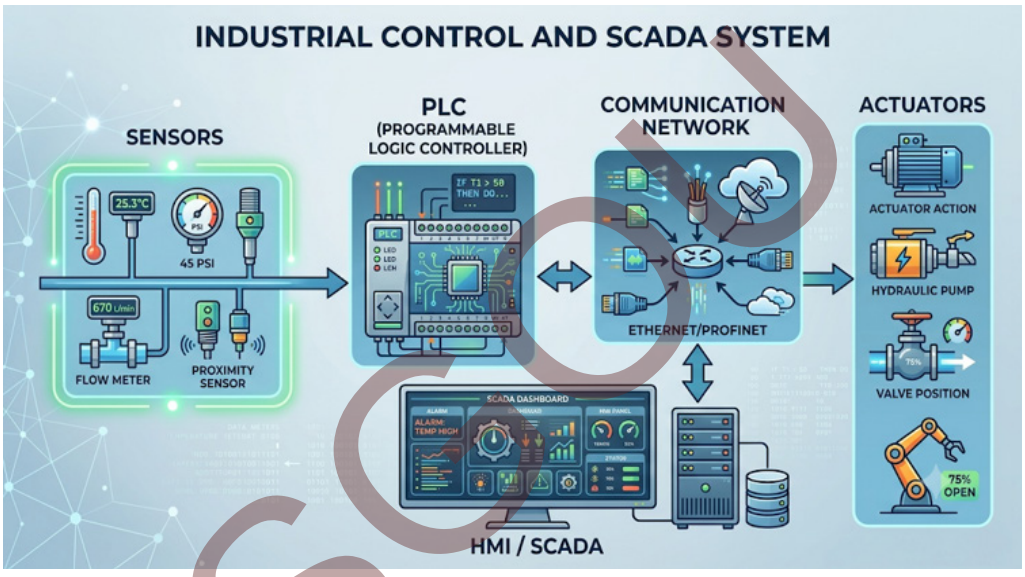


Fig. 6.1.3 Basic components of an Industrial Control System.

6.1.5 Types of Industrial Control Systems

Industrial environments employ several categories of control systems depending on operational requirements.

The most common systems include:

- SCADA
- Distributed Control System (DCS)
- Programmable Logic Controller (PLC)

6.1.5.1 Supervisory Control and Data Acquisition (SCADA)

SCADA is one of the most widely used Industrial Control Systems.

Its primary function is to monitor geographically distributed industrial processes.

SCADA continuously collects information from remote locations and presents it to operators using graphical interfaces.

Applications

SCADA is extensively used in:

- Electricity transmission
- Water distribution
- Oil and gas pipelines
- Railway signalling
- Smart grid management

Working Principle

The operation of a SCADA system can be understood through the following sequence:

1. Sensors measure process variables such as temperature, pressure or flow rate.
2. Remote Terminal Units (RTUs) or PLCs collect sensor information.
3. Communication networks transfer the information to the central SCADA server.
4. Operators monitor the process through Human Machine Interface (HMI) software.
5. Control commands are transmitted back to field equipment when required. See figure 6.1.4 SCADA Architecture.

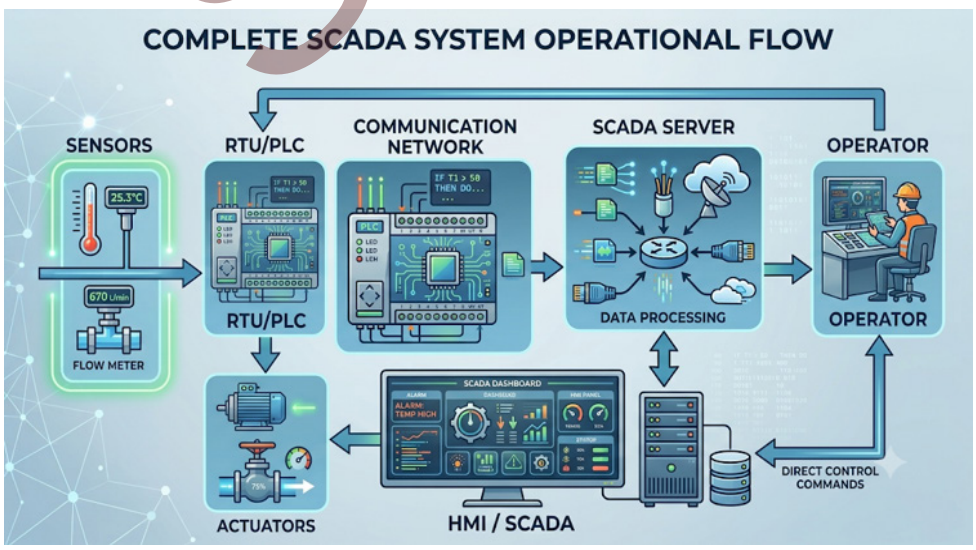


Fig. 6.1.4 General architecture of a SCADA system.

SGOU

Advantages of SCADA

- Remote monitoring
- Real-time data collection
- Centralized supervision
- Faster fault detection
- Improved operational efficiency

Limitations

- Higher implementation cost
- Dependence on communication networks
- Increasing cybersecurity risks

6.1.5.2 Programmable Logic Controller (PLC)

A Programmable Logic Controller (PLC) is a rugged industrial computer designed to automate machinery and manufacturing processes.

Unlike ordinary computers, PLCs operate reliably under harsh industrial conditions involving:

- Heat
- Dust
- Vibration
- Electrical noise
- Moisture

Functions of PLC

A PLC performs the following sequence repeatedly:

- Reads sensor inputs.
- Executes the control program.
- Generates control outputs.
- Repeats the cycle continuously.

Applications

PLCs are widely used in:

- Conveyor belt control

SGOU

- Packaging systems
- Industrial robots
- Elevator control
- Traffic signal systems.

See figure 6.1.5 PLC Operation

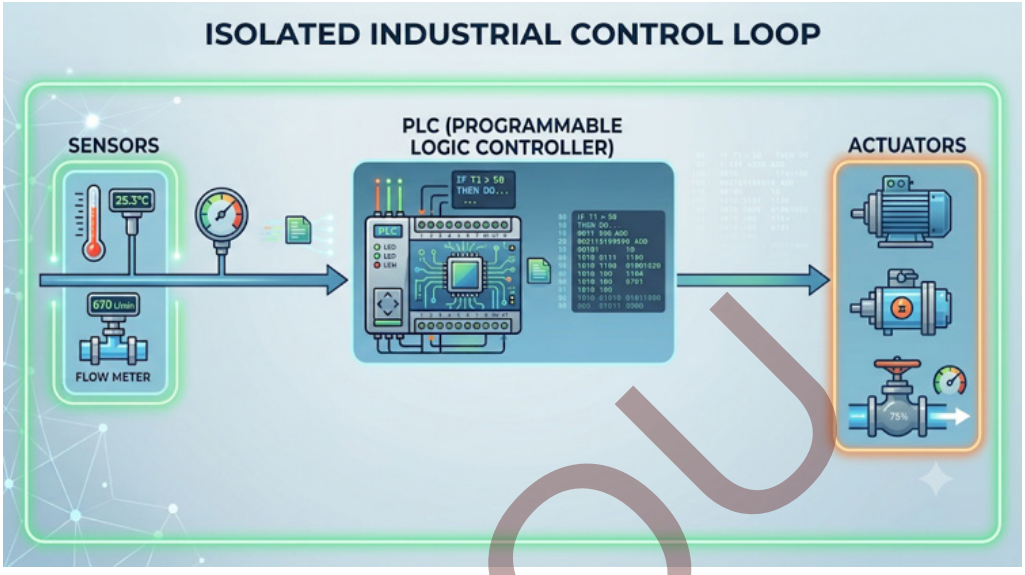


Fig. 6.1.5 Basic operation of a Programmable Logic Controller.

6.1.5.3 Distributed Control System (DCS)

Large industrial plants often contain thousands of sensors and actuators.

Managing all these devices using a single controller becomes difficult.

A Distributed Control System (DCS) distributes control functions among multiple controllers located near different sections of the plant.

Applications

- Chemical industries
- Petroleum refineries
- Pharmaceutical manufacturing
- Thermal power plants
- Cement industries

Advantages

- Improved reliability

SGOU

- Fault isolation
- Better scalability
- Faster process control
- Easier maintenance

Table 6.1.2 Comparison of SCADA, PLC and DCS

Feature	SCADA	PLC	DCS
Primary Function	Supervisory Monitoring	Machine Control	Process Control
Coverage	Wide Geographic Area	Individual Machines	Large Industrial Plants
Response Time	Moderate	Very Fast	Fast
Scalability	High	Moderate	Very High
Typical Applications	Power Grid	Conveyor System	Chemical Plant

Industry Insight

Modern smart factories often combine SCADA, PLC, and DCS within a single Industrial Internet of Things (IIoT) architecture. PLCs control individual machines, DCS manages plant-wide processes, while SCADA provides centralized monitoring and remote supervision.

Solved Illustration

Problem

A water distribution company wants to monitor pumping stations located across different cities from a single control centre.

Which Industrial Control System is most appropriate?

Solution

Requirements

- Remote monitoring
- Geographically distributed locations
- Centralized supervision
- Real-time communication

SGOU

Suitable System: SCADA

Reason: SCADA enables centralized monitoring and control of geographically dispersed industrial facilities, making it ideal for water distribution networks.

6.1.6 Industrial Internet of Things (IIoT)

Until recently, most industrial control systems functioned as isolated environments. Machines communicated only with nearby controllers, and information rarely left the factory premises. Today, industries are becoming increasingly connected through the Industrial Internet of Things (IIoT), enabling intelligent monitoring, predictive maintenance and data-driven decision making.

The Industrial Internet of Things (IIoT) refers to the application of IoT technologies in industrial environments. By integrating sensors, controllers, communication networks, cloud platforms and data analytics, IIoT enables industries to collect and analyse operational data in real time.

Unlike consumer IoT applications such as smart homes or wearable devices, IIoT emphasizes reliability, safety, availability and precision, as failures can have significant financial and safety consequences.

Definition

Industrial Internet of Things (IIoT) is the integration of industrial equipment, sensors, controllers, communication networks and cloud technologies to monitor, control and optimize industrial processes. See figure 6.1.6 Architecture of an Industrial IoT System.

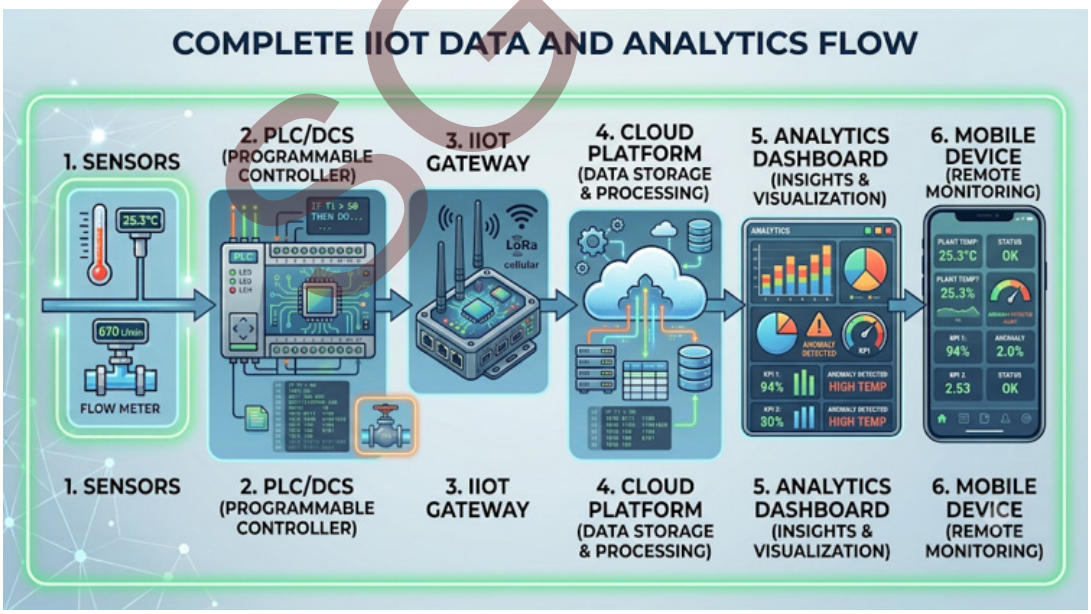


Fig. 6.1.6 Typical architecture of an Industrial Internet of Things (IIoT) system.

SGOU

Advantages of IIoT

Industrial IoT offers several benefits to modern organizations.

- Continuous monitoring of industrial equipment
- Predictive maintenance of machines
- Improved production efficiency
- Reduced operational costs
- Better resource utilization
- Faster fault detection
- Remote monitoring of industrial plants
- Improved decision-making through analytics

Applications of IIoT

Industrial IoT is widely adopted in:

- Smart manufacturing
- Smart energy systems
- Oil and gas industries
- Transportation
- Agriculture
- Water management
- Healthcare
- Mining

Industry Insight

Many manufacturing industries now employ Digital Twins, which are virtual representations of physical machines. Data collected through IIoT continuously updates the digital model, enabling engineers to simulate operations, detect faults and optimize performance without interrupting actual production.

6.1.7 Common Cyber Threats in OT Environments

As industrial systems become interconnected, they also become attractive targets for cybercriminals. Unlike attacks on traditional IT systems, attacks on OT environments can directly affect machinery, production processes and public safety.

Some of the most common cyber threats are discussed below.

SGOU

6.1.7.1 Malware

Malware refers to malicious software intentionally developed to damage, disrupt or gain unauthorized access to computer systems.

Examples include:

- Viruses
- Worms
- Trojans
- Spyware

In industrial environments, malware may alter controller programs, interrupt production or corrupt operational data.

Example

A malware infection in a manufacturing plant may stop automated assembly lines, resulting in production delays and financial losses.

6.1.7.2 Ransomware

Ransomware is one of the fastest-growing cyber threats affecting industries.

After infecting a system, ransomware encrypts important files and demands payment before restoring access.

Industrial organizations are particularly vulnerable because production downtime often results in significant financial losses. See figure 6.1.7 Ransomware Attack Process.

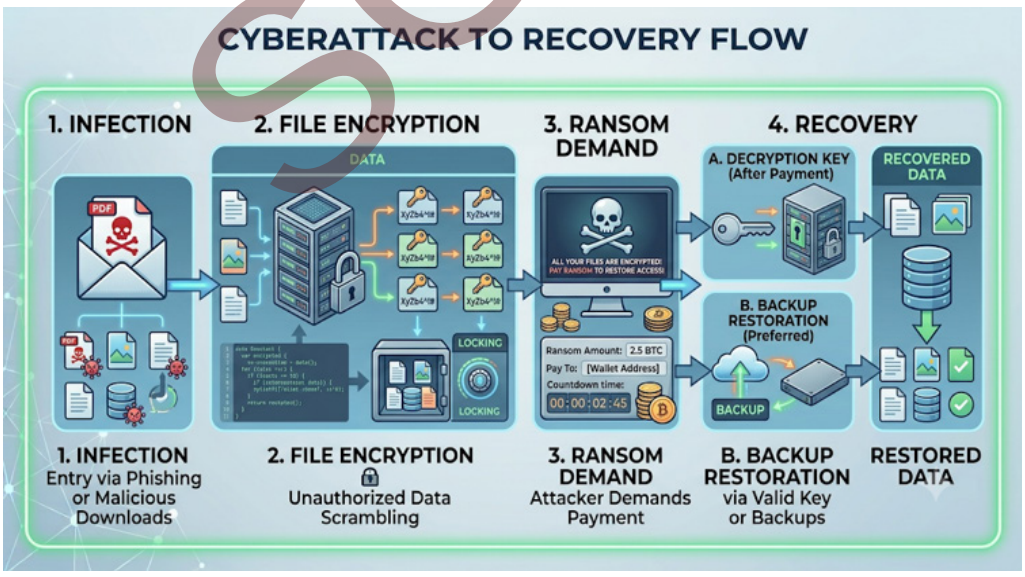


Fig. 6.1.7 General stages of a ransomware attack.

SGOU

6.1.7.3 Insider Threats

Not every cyber attack originates from external hackers.

Sometimes employees, contractors or vendors accidentally or intentionally compromise industrial systems.

Examples include:

- Weak passwords
- Unauthorized software installation
- Mishandling confidential information
- Deliberate sabotage
- Human error

Insider threats remain one of the most challenging security issues because insiders often possess legitimate system access.

6.1.7.4 Denial-of-Service (DoS) Attacks

A Denial-of-Service (DoS) attack attempts to make industrial systems unavailable by overwhelming communication networks or servers with excessive requests.

Consequences may include:

- Loss of communication
- Delayed control signals
- Interrupted monitoring
- Production downtime

6.1.7.5 Unauthorized Remote Access

Modern industries increasingly permit remote maintenance and monitoring.

While remote connectivity improves operational efficiency, weak authentication mechanisms may allow attackers to gain unauthorized access to industrial control systems.

Proper authentication and secure communication protocols are therefore essential.

The Cost of Industrial Downtime

Studies indicate that an hour of unplanned downtime in a large manufacturing facility can cost thousands or even millions of dollars, depending on the industry. Consequently, preventing cyber incidents is often far less expensive than recovering from them.

SGOU

6.1.8 Vulnerabilities in Operational Technology (OT) Environments

Many industrial facilities continue to operate equipment and control systems that were installed twenty or thirty years ago. These systems include industrial controllers, sensors, monitoring devices and Supervisory Control and Data Acquisition (SCADA) systems. They were primarily designed to ensure reliable and continuous industrial operations rather than to address modern cybersecurity threats. At the time of their development, industrial networks were generally isolated from external networks, and security features such as encryption, advanced authentication and continuous threat monitoring were not considered essential. As a result, many legacy industrial systems have security limitations that make them vulnerable to cyberattacks.

Securing Operational Technology (OT) environments against modern cyber threats is particularly challenging because these systems have specific operational requirements that differ from those of conventional IT environments. OT systems are designed to control physical processes and ensure the continuous operation of industrial facilities. Therefore, implementing security measures requires careful consideration of operational availability, personnel safety, equipment limitations and system reliability. The major challenges associated with OT security are explained below.

1. 24/7 Availability Requirements

OT systems are expected to operate continuously, often throughout the day and night, without interruptions. Industrial processes such as electricity generation, water treatment, manufacturing and transportation depend on the uninterrupted functioning of control systems. Even a short period of downtime may interrupt production, affect essential services or result in significant financial losses.

However, implementing cybersecurity measures in OT environments may require temporarily stopping certain operations. For example, installing software updates or security patches may require restarting a computer, controller or other connected equipment. Such interruptions may not be acceptable in industrial environments where continuous operation is essential. This creates a challenge for organizations because delaying security updates may leave systems exposed to known vulnerabilities, while applying updates immediately may affect operational continuity.

To address this challenge, organizations should plan security improvements carefully and implement them in stages. Updates should be tested in a separate environment before being introduced into operational systems. Backup equipment, redundant controllers and failover mechanisms can also help maintain operations when a component becomes unavailable. In addition, continuous monitoring and rapid incident response help identify security problems early and reduce the impact of unexpected disruptions.

2. Safety Risks

Unlike conventional IT systems, OT systems directly interact with physical equipment and industrial processes. Therefore, a cybersecurity incident or an incorrectly implemented security measure may create risks to human life, equipment

SGOU

and the surrounding environment. Industrial processes often operate within specific safety limits, such as temperature, pressure, speed and chemical concentration. Any unauthorized modification or interruption of these control parameters may lead to dangerous situations.

For example, disabling an industrial alarm or interfering with a pressure monitoring system may prevent operators from receiving warnings about an unsafe increase in pressure inside an industrial vessel. Similarly, blocking communication between a controller and a safety monitoring device may affect the ability of the system to detect hazardous conditions. Consequently, security controls must not interfere with essential safety functions or prevent operators from responding to emergencies.

To minimize these risks, cybersecurity professionals and industrial safety engineers must work together when designing and implementing security measures. Every proposed security change should be evaluated for its possible impact on industrial processes and safety requirements. Security mechanisms should be tested carefully to ensure that they protect the system without affecting essential control operations. Therefore, OT security requires an approach that considers both cybersecurity and industrial safety rather than directly applying conventional IT security practices.

3. Weak Identity and Access Management

Identity and Access Management (IAM) refers to the processes used to identify users, verify their identities and control their access to systems and resources. Effective IAM ensures that only authorized individuals can access industrial devices and perform permitted operations. However, many legacy OT systems have limited or outdated authentication and access control capabilities.

In some industrial environments, multiple operators and maintenance personnel may use shared usernames and passwords to access control systems. User accounts may not be properly documented, and access permissions may not be reviewed regularly. In addition, some devices may not support individual user authentication or advanced security mechanisms. These limitations make it difficult to determine who accessed a particular device or modified an industrial configuration. They may also allow unauthorized users to obtain excessive privileges.

For example, if several maintenance employees use the same administrator account to access a SCADA workstation, identifying the individual responsible for an unauthorized configuration change becomes difficult. Similarly, an employee who no longer requires access may continue to use an active account if account permissions are not updated.

Organizations should maintain an accurate record of users, devices and their access permissions. Individual user accounts, strong passwords, role-based access control and periodic permission reviews can improve access security. Network zoning and physical access restrictions should also be implemented to prevent unauthorized access to critical industrial equipment.

4. Difficulties in Asset Management

Asset management involves identifying, recording and maintaining information

SGOU

about all hardware, software and network components within an organization. In OT environments, effective asset management is essential because industrial networks may contain a large number of interconnected devices, including PLCs, sensors, SCADA servers, engineering workstations, communication gateways and monitoring systems.

However, maintaining an accurate inventory of OT assets can be difficult, particularly in large industrial facilities. Some devices may have been installed many years ago, while others may be located at remote sites or connected through third-party networks. Organizations may not have complete information about the devices in use, their software versions, communication protocols or security configurations. As a result, identifying outdated devices, known vulnerabilities and unauthorized connections becomes difficult.

For example, an organization may be aware of the main control systems in its manufacturing facility but may not have a complete record of older controllers or remote monitoring devices connected to the industrial network. These unidentified devices may contain outdated software or insecure communication services that attackers could exploit.

Organizations should conduct regular OT asset discovery and maintain an updated inventory of all industrial devices and their configurations. Each asset should be classified according to its importance, security condition and operational role. Network segmentation can restrict unnecessary communication between devices and reduce the possibility of attacks spreading across the environment. Air-gapped systems, which are physically isolated from other networks, can also reduce external attack paths when properly implemented and maintained.

5. Weak Threat Monitoring and Detection

Threat monitoring involves continuously observing network communication, system activities and security events to identify suspicious behaviour or potential cyberattacks. In conventional IT environments, organizations commonly use antivirus software, endpoint detection systems and network monitoring tools to identify malicious activities. However, these tools may not be suitable for every OT device because many industrial systems have limited computing resources and operate using specialized hardware and software.

Legacy OT devices may not support the installation of modern security applications. In addition, some industrial communication protocols were designed primarily for reliable data exchange rather than security. Their communication patterns may be unfamiliar to conventional IT security tools, making it difficult to identify malicious activities. Consequently, unauthorized commands, abnormal network traffic and suspicious changes in industrial operations may remain undetected.

For example, an attacker may send unusual commands to a PLC through an industrial communication protocol. A conventional security monitoring tool may not recognize these commands as suspicious because it does not understand the normal communication behaviour of that industrial device. This can delay the detection of an attack and increase its potential impact.

SGOU

Organizations should use security monitoring solutions specifically designed for Industrial Control Systems (ICS) and SCADA environments. These solutions can analyse industrial network traffic, understand commonly used OT protocols and identify unusual communication patterns. Intrusion Detection Systems (IDS) configured for industrial environments can also help detect unauthorized access attempts, abnormal commands and suspicious network activities. Continuous monitoring, proper event logging and timely security alerts enable organizations to identify and respond to threats more effectively.

The security of OT environments is influenced by several challenges that arise from their continuous operational requirements and direct interaction with physical processes. Maintaining 24/7 availability, ensuring personnel safety, managing user identities, identifying all connected assets and detecting threats are important considerations in OT security. Organizations must adopt security measures that are compatible with industrial operations and avoid introducing additional risks to system reliability and safety. A combination of careful security planning, asset management, access control, network segmentation and specialized threat monitoring helps strengthen OT security while maintaining uninterrupted industrial operations.

Table 6.1.3 Common OT Security Challenges

Challenge	Description	Possible Impact
Legacy Equipment	Old devices without security features	Increased vulnerability
Weak Authentication	Poor password practices	Unauthorized access
Lack of Encryption	Data transmitted in plain text	Data interception
Continuous Operation	Difficult to shut down systems	Delayed security updates
IT–OT Integration	Expanded attack surface	Increased cyber risk
Human Error	Incorrect configurations	Operational disruption

6.1.9 Convergence of IT and OT

Traditionally, Information Technology (IT) and Operational Technology (OT) functioned independently.

Today, organizations integrate these environments to achieve:

- Centralized monitoring
- Cloud connectivity
- Data analytics

SGOU

- Artificial Intelligence
- Predictive maintenance
- Business intelligence

Although IT–OT convergence enhances operational efficiency, it also introduces additional cybersecurity risks because attackers may exploit weaknesses in IT systems to access industrial control networks.

Consequently, organizations increasingly adopt Zero Trust Security, network segmentation and continuous monitoring to protect integrated environments. See figure 6.1.8 IT–OT Convergence.

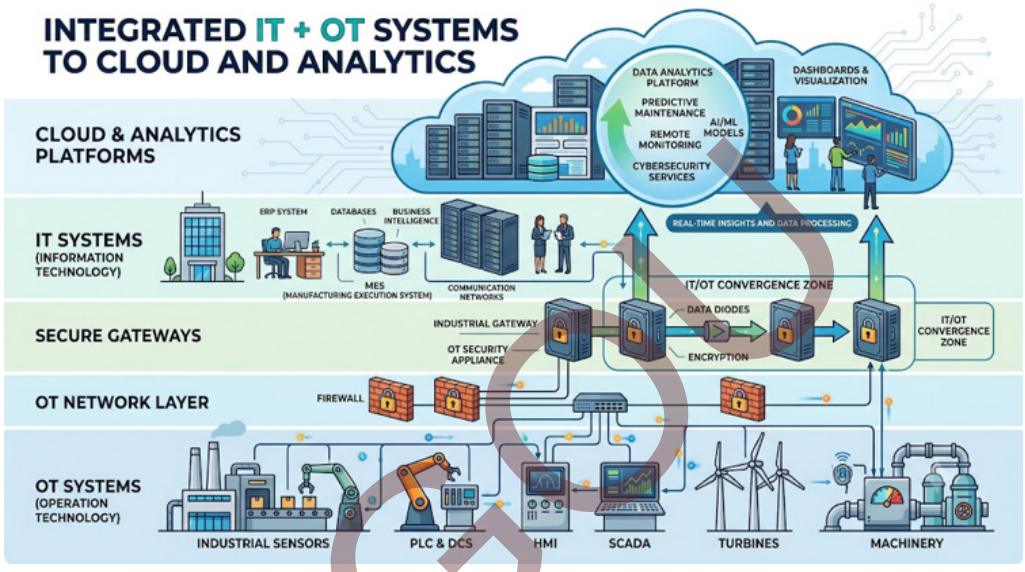


Fig. 6.1.8 Integration of Information Technology and Operational Technology.

Case Study

Cyber Attack on a Water Treatment Facility

A municipal water treatment facility enabled remote access for maintenance engineers. An attacker obtained valid login credentials through phishing and gained unauthorized access to the control network. The attacker attempted to alter chemical dosing parameters.

Fortunately, plant operators observed unusual readings through the Human Machine Interface (HMI) and immediately restored normal operating conditions before public safety was affected.

Lessons Learned

- Strong authentication is essential.
- Multi-factor authentication should be implemented.
- Continuous monitoring detects abnormal activities early.

SGOU

- Employee cybersecurity awareness is equally important.
- Remote access should be strictly controlled.

Problem

A manufacturing company uses several PLCs that cannot be upgraded because production must continue 24 hours a day.

How can the organization improve cybersecurity without replacing the existing PLCs?

Solution

Instead of replacing the PLCs immediately, the organization can:

- Isolate the PLC network using firewalls.
- Restrict remote access.
- Implement network segmentation.
- Continuously monitor industrial traffic.
- Apply security patches to connected systems whenever possible.
- Train employees in secure operational practices.

These measures significantly reduce cybersecurity risks while maintaining uninterrupted production.

6.1.10 Fundamental Security Principles for OT Systems

Operational Technology systems require a security strategy that prioritizes human safety, continuous operation, and system reliability. Unlike conventional IT systems, where protecting information is often the primary objective, OT systems must ensure that industrial processes continue to operate safely even during cyber incidents.

The following principles form the foundation of OT security.

6.1.10.1 Defense in Depth

Defense in Depth is a layered security approach where multiple security controls protect industrial systems. If one security mechanism fails, additional layers continue to safeguard the system.

Typical layers include:

- Physical security
- Network security
- Endpoint protection
- User authentication

SGOU

- Continuous monitoring
- Backup and recovery

See figure 6.1.9 Defense in Depth.

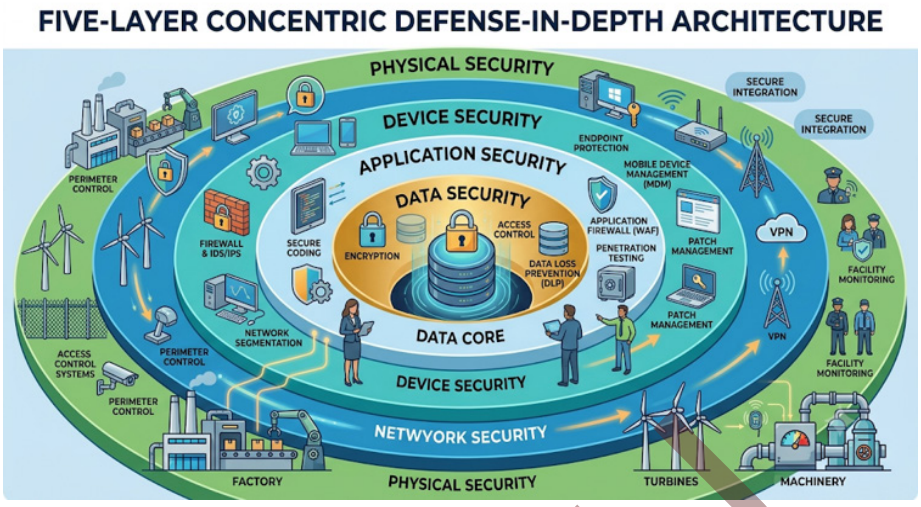


Fig. 6.1.9 Multiple security layers provide stronger protection than relying on a single defense mechanism.

6.1.10.2 Network Segmentation

Instead of connecting every device to the same network, industries divide networks into smaller, isolated segments.

For example:

- Office Network
- Industrial Control Network
- Safety Instrumented System
- Remote Access Network

If malware infects one network segment, segmentation helps prevent its spread to critical industrial systems.

6.1.10.3 Least Privilege Principle

Every user should receive only the permissions necessary to perform assigned duties.

For example:

- Operators should control machines but not modify security policies.
- Engineers may configure controllers but should not access payroll systems.
- Visitors should receive temporary, restricted access.

SGOU

This principle significantly reduces the possibility of accidental or intentional misuse.

6.1.10.4 Continuous Monitoring

Modern industries continuously monitor:

- Network traffic
- Device status
- User activities
- Login attempts
- Controller behaviour

Continuous monitoring enables early detection of abnormal activities before they escalate into serious cyber incidents.

6.1.10.5 Incident Response Planning

Even the best security measures cannot guarantee complete protection. Organizations therefore prepare Incident Response Plans (IRPs) that define how cyber incidents should be handled.

A typical incident response process includes:

1. Detection
2. Reporting
3. Containment
4. Investigation
5. Recovery
6. Post-incident review

Preparedness minimizes operational disruption and recovery time.

Cybersecurity is a Continuous Process

Cybersecurity is not a product that can be purchased once and forgotten. It is an ongoing process involving technology, organizational policies, employee awareness, regular audits and continuous improvement.

6.1.11 Best Practices for Securing OT Environments

Organizations should adopt comprehensive security practices to reduce cyber risks.

Recommended Best Practices

- Implement strong authentication mechanisms.

SGOU

- Use Multi-Factor Authentication (MFA) wherever possible.
- Regularly update software and firmware.
- Disable unused communication ports.
- Maintain secure backups.
- Conduct periodic vulnerability assessments.
- Train employees in cybersecurity awareness.
- Restrict remote access.
- Encrypt sensitive communication.
- Maintain detailed security logs.
- Perform periodic penetration testing.
- Develop disaster recovery plans.

Together, these practices significantly strengthen the resilience of OT systems against cyber threats.

Industry Insight

Many industrial organizations now establish Security Operations Centers (SOCs) that continuously monitor industrial networks, analyse security events and respond rapidly to cyber incidents. By integrating artificial intelligence and threat intelligence, SOCs can identify unusual activities before they affect industrial operations.

6.1.12 Emerging Trends in OT Security

Operational Technology security continues to evolve as industries adopt digital transformation technologies.

Some important emerging trends include:

6.1.12.1 Artificial Intelligence for Threat Detection

Artificial Intelligence (AI) analyses large volumes of industrial network traffic to identify unusual patterns and activities. It uses machine learning techniques to distinguish normal operations from suspicious behaviour that may indicate cyber attacks. AI based threat detection helps security teams identify potential threats quickly and respond to them effectively.

6.1.12.2 Zero Trust Security

Zero Trust Security follows the principle that no user or device should be trusted automatically, even if it is connected to the internal network. Every access request must be verified through authentication, authorization and continuous security checks, where appropriate. This approach reduces the risk of unauthorized access and limits the spread of attacks within industrial networks.

SGOU

6.1.12.3 Cloud Based Security Monitoring

Cloud based security monitoring uses cloud platforms to collect and analyse security data from multiple industrial systems. It enables centralized monitoring, threat detection and the identification of suspicious activities across connected devices and networks. It can also support predictive maintenance by analysing operational data to identify potential equipment failures.

6.1.12.4 Digital Twins

Digital Twins are virtual representations of physical industrial systems, equipment or processes that use data to reflect their real world behaviour. They allow engineers to simulate cyber attacks and evaluate possible security weaknesses in a controlled virtual environment. The results help organizations assess security improvements before implementing them in actual industrial systems.

6.1.12.5 Industrial Threat Intelligence

Industrial threat intelligence involves collecting, analysing and sharing information about cyber threats that target industrial systems. This information may include attack techniques, malicious indicators, known vulnerabilities and emerging threats. Sharing reliable threat intelligence helps organizations strengthen their security measures and improve their ability to prevent, detect and respond to cyber attacks.

Case Study

Protecting a Smart Manufacturing Plant

A manufacturing company connected hundreds of industrial robots, PLCs and quality inspection systems to a centralized monitoring platform.

Initially, every device communicated through a single network.

After conducting a cybersecurity assessment, engineers implemented:

- Network segmentation
- Firewalls
- Multi-factor authentication
- Continuous monitoring
- Secure remote access
- Employee awareness programmes

Within one year:

- Malware incidents reduced significantly.
- Network performance improved.

SGOU

- Unauthorized access attempts were detected early.
- Overall operational reliability increased.

This example demonstrates how layered cybersecurity significantly strengthens industrial resilience.

Recap

- Operational Technology (OT) controls physical processes in industries and critical infrastructure.
- OT differs from Information Technology because its primary objectives are safety, availability and reliability.
- Industrial Control Systems (ICS) include SCADA, PLC and Distributed Control Systems (DCS).
- Industrial Internet of Things (IIoT) connects industrial devices for intelligent monitoring and automation.
- Increasing connectivity has introduced new cybersecurity challenges for OT environments.
- Common cyber threats include malware, ransomware, insider threats, denial-of-service attacks and unauthorized remote access.
- Legacy industrial systems often lack modern cybersecurity mechanisms.
- IT–OT convergence improves operational efficiency but also expands the cyber attack surface.
- Defense in Depth, network segmentation and the Principle of Least Privilege strengthen OT security.
- Continuous monitoring and incident response planning enable organizations to detect and recover from cyber incidents efficiently.
- Emerging technologies such as Artificial Intelligence, Digital Twins and Zero Trust Security are transforming OT cybersecurity.
- Protecting Operational Technology is essential for ensuring industrial productivity, public safety and national security.

SGOU

Objective Type Questions

1. What is Operational Technology?
2. Which systems are collectively known as Industrial Control Systems?
3. What is the primary objective of OT?
4. Which Industrial Control System is widely used for geographically distributed monitoring?
5. What is the function of a Programmable Logic Controller?
6. What does IIoT stand for?
7. Which malware encrypts files and demands payment?
8. What is meant by an insider threat?
9. Why are legacy OT systems vulnerable?
10. What is network segmentation?
11. Which security principle grants only minimum required permissions?
12. What is Defense in Depth?
13. Why is continuous monitoring important?
14. What is the purpose of an Incident Response Plan?
15. Name two emerging technologies used in OT security.

SGOU

Answers

1. Technology that monitors and controls industrial processes.
2. SCADA, PLC and DCS.
3. Safe and reliable control of physical processes.
4. SCADA.
5. To automate industrial equipment.
6. Industrial Internet of Things.
7. Ransomware.
8. A threat originating from authorized users.
9. They lack modern security features.
10. Dividing networks into isolated sections.
11. Least Privilege Principle.

12. Multiple layers of security.
13. To detect abnormal activities early.
14. To manage and recover from cyber incidents.
15. Artificial Intelligence and Digital Twins (or Zero Trust Security).

Assignments

1. Explain the concept of Operational Technology and distinguish it from Information Technology with suitable examples.
2. Discuss the architecture and applications of Industrial Control Systems, including SCADA, PLC and DCS.
3. Explain the importance of Industrial Internet of Things (IIoT) in modern industries and discuss its cybersecurity implications.
4. Describe the major cyber threats affecting Operational Technology environments with appropriate industrial examples.
5. Discuss the cybersecurity challenges associated with legacy industrial systems and IT–OT convergence.
6. Explain the principles of Defense in Depth, Least Privilege and Network Segmentation in OT security.
7. Discuss best practices that industries should adopt to improve Operational Technology security.
8. Analyse the role of Artificial Intelligence and Zero Trust Security in strengthening future OT environments.

SGOU

Suggested Reading

1. Langner, Ralph. Robust Control System Networks. Momentum Press, 2017.
2. Krutz, Ronald L. Industrial Control Systems Security and Resiliency. Wiley, 2016.
3. Stallings, William. Network Security Essentials: Applications and Standards. Pearson, 2020.
4. Easttom, Chuck. Modern Industrial Security. McGraw-Hill Education, 2022.
5. NIST. Guide to Operational Technology (OT) Security. National Institute of Standards and Technology.

Reference

1. Hanes, David, et al. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Cisco Press, 2017.
2. Knapp, Eric D., and Joel Thomas Langill. Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems. 2nd ed., Syngress, 2015.
3. Stouffer, Keith, et al. Guide to Industrial Control Systems (ICS) Security. National Institute of Standards and Technology (NIST), Special Publication 800-82.
4. Macaulay, Tyson, and Bryan Singer. Cybersecurity for Industrial Control Systems. CRC Press, 2016.
5. Boyes, Hugh, and Tim Watson. Industrial Internet of Things Security. Institution of Engineering and Technology, 2021.

SGOU



Formal Risk Analysis Structures – OCTAVE & FAIR

Learning Outcomes

After completion of this unit, learner will be able to:

- define information security risk and risk analysis.
- identify the components of a formal risk assessment process.
- explain the OCTAVE framework for cybersecurity risk assessment.
- describe the FAIR framework and its major concepts.
- recognize the importance of formal risk analysis in Operational Technology environments.

Prerequisite

Imagine that you are responsible for managing the computer systems of a large hospital. Hundreds of patients depend on digital medical records, diagnostic equipment, and life-support systems every day. Suddenly, you receive information that hackers are attempting to attack the hospital's network.

Should you spend all available resources protecting every system equally?

Probably not.

Some systems, such as email servers, can tolerate short interruptions. However, systems controlling intensive care units or emergency services require immediate protection because their failure could threaten human lives.

This simple situation introduces an important concept in cybersecurity—Risk Analysis. Instead of trying to protect everything with the same level of effort, organizations identify their most valuable assets, understand possible threats, estimate risks, and prioritize security measures accordingly.

In this unit, we will learn how internationally accepted frameworks such as OCTAVE and FAIR help organizations make informed cybersecurity decisions.

Before studying this unit, you should be familiar with:

SGOU

- Basic cybersecurity concepts
- Operational Technology (OT)
- Industrial Control Systems (ICS)
- Computer networks

Keywords

Risk, Asset, Threat, Vulnerability, Impact, OCTAVE, FAIR, Risk Assessment

Discussion

6.2.1 Introduction to Risk Analysis

Every organization faces uncertainty. A manufacturing company may experience equipment failure, a bank may face cyber attacks, and a hospital may encounter ransomware incidents. While it is impossible to eliminate every threat completely, organizations can identify risks and reduce their likelihood and impact.

This systematic process is known as Risk Analysis.

Risk analysis enables organizations to understand what could go wrong, how likely it is to occur, and what consequences may result. Based on this understanding, organizations can allocate resources efficiently and implement appropriate security controls.

In cybersecurity, risk analysis is an essential activity because resources such as budget, personnel and time are always limited. Effective risk analysis ensures that the most critical assets receive the highest level of protection.

Definition

Risk Analysis is the systematic process of identifying, evaluating and prioritizing risks so that appropriate measures can be implemented to reduce their likelihood and impact.

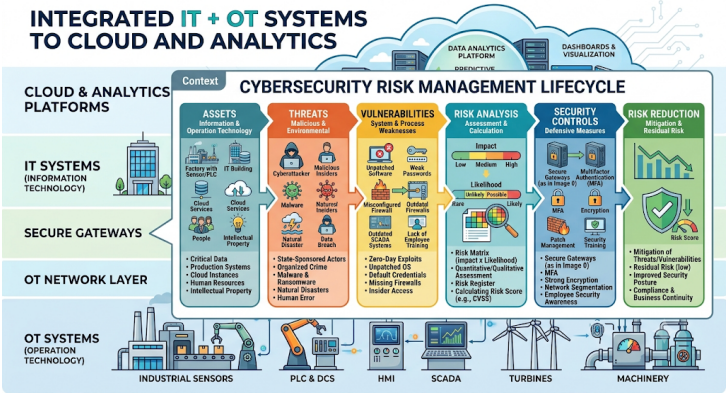


Fig. 6.2.1 General process of cybersecurity risk analysis.

SGOU

Example

Consider a smart manufacturing plant.

The organization owns:

- Industrial robots
- PLC controllers
- Production databases
- Network infrastructure
- Employee information systems

If ransomware infects the production network, manufacturing operations may stop completely.

Risk analysis helps determine:

- Which assets are most valuable?
- Which threats are most dangerous?
- Which vulnerabilities exist?
- Which security controls should be implemented first?

Industry Insight

Organizations that conduct regular cybersecurity risk assessments generally experience fewer security incidents because they proactively identify weaknesses before attackers exploit them.

6.2.2 Understanding Information Security Risk

Information security risk exists whenever valuable organizational assets are exposed to threats that exploit existing vulnerabilities.

Risk is not simply the existence of a threat.

Instead, risk arises when three important elements exist simultaneously:

- Valuable asset
- Existing vulnerability
- Potential threat

If any one of these elements is absent, the overall risk becomes significantly lower.

SGOU

Example

Suppose an industrial company stores production data on an isolated computer that has no network connection.

Although cyber attackers exist, they cannot directly access the isolated computer through the Internet.

Consequently, the cybersecurity risk is much lower than that of an Internet-connected production server.



Fig. 6.2.2 Risk exists when threats exploit vulnerabilities associated with valuable assets.

6.2.3 Components of Risk

Understanding the components of risk is essential before performing formal risk assessment.

Asset

An asset is anything valuable to an organization.

Examples include:

- Industrial equipment
- Production data
- Software applications
- Network infrastructure
- Customer information
- Intellectual property

Threat

A threat is any event or factor capable of causing harm.

Threats may originate from:

- Cybercriminals
- Insider attacks
- Natural disasters
- Hardware failures
- Human errors

Vulnerability

A vulnerability is a weakness that can be exploited by threats.

Examples include:

- Weak passwords
- Outdated software
- Misconfigured firewalls
- Unpatched operating systems
- Inadequate employee training

Impact

Impact refers to the consequences experienced if a threat successfully exploits a vulnerability.

Possible impacts include:

- Financial loss
- Operational disruption
- Equipment damage
- Legal penalties
- Reputational damage
- Safety incidents

Likelihood

Likelihood represents the probability that a particular threat will successfully exploit a vulnerability.

SGOU

Organizations typically classify likelihood as:

- Very Low
- Low
- Medium
- High
- Very High

Table 6.2.1 Components of Risk

Component	Description	Example
Asset	Valuable organizational resource	PLC Controller
Threat	Potential source of harm	Malware
Vulnerability	Weakness in security	Weak password
Impact	Consequence of attack	Production shutdown
Likelihood	Probability of occurrence	High

Not Every Vulnerability Creates High Risk

Many organizations discover hundreds of vulnerabilities during security assessments. However, not all vulnerabilities require immediate attention. A vulnerability affecting an isolated test computer usually presents much less risk than a similar vulnerability affecting the controller of a power plant.

Effective risk analysis helps prioritize corrective actions.

6.2.4 Need for Formal Risk Analysis

Organizations cannot rely solely on intuition when making cybersecurity decisions.

Formal risk analysis provides a structured methodology for identifying risks, estimating their significance and selecting appropriate mitigation strategies.

Some important reasons include:

Better Decision-Making

Managers allocate resources based on measurable risks rather than assumptions.

Improved Compliance

Formal risk assessment supports compliance with international standards such as:

SGOU

- ISO/IEC 27001
- IEC 62443
- NIST Cybersecurity Framework

Efficient Resource Utilization

Organizations invest first in protecting high-risk assets instead of treating every asset equally.

Business Continuity

Risk assessment helps prevent operational disruptions by identifying critical systems requiring additional protection.

Enhanced Cyber Resilience

Organizations become better prepared to detect, respond to and recover from cyber incidents.



Fig. 6.2.3 Benefits of conducting formal cybersecurity risk analysis.

Problem

A manufacturing company identifies two cybersecurity issues:

- Weak password policy affecting office computers.
- Outdated firmware affecting PLC controllers used in production.

Which issue should receive higher priority?

Solution

Although both issues require attention, the outdated PLC firmware presents greater

organizational risk because successful exploitation could directly disrupt production and compromise operational safety.

Therefore, updating and protecting the PLC controllers should receive higher priority.

6.2.5 Risk Assessment Process

Risk assessment is one of the most important activities in cybersecurity management. It enables organizations to systematically identify risks, evaluate their severity and decide how to manage them effectively.

Risk assessment is not a one-time activity. Since technologies, threats and business operations change continuously, organizations should perform risk assessments periodically to maintain a strong security posture.

A standard risk assessment process generally consists of the following steps:

1. Asset Identification
2. Threat Identification
3. Vulnerability Assessment
4. Risk Analysis
5. Risk Evaluation
6. Risk Treatment
7. Continuous Monitoring and Review



Fig. 6.2.4 Continuous cybersecurity risk assessment lifecycle.

SGOU

6.2.5.1 Asset Identification

The first step is identifying the assets that require protection.

An asset is anything valuable to an organization that supports its business operations.

Examples include:

Physical Assets

- Industrial robots
- PLC controllers
- SCADA servers
- Network switches
- Manufacturing equipment

Digital Assets

- Databases
- Software applications
- Design documents
- Cloud storage
- Customer information

Human Assets

- Employees
- Engineers
- Operators
- Contractors

Without identifying assets, it becomes impossible to determine what needs protection.

Example

In a power generation plant, important assets include:

- Turbine control systems
- SCADA servers
- Generator monitoring systems
- Operator workstations
- Communication networks

SGOU

These assets directly influence electricity production and therefore receive the highest security priority.

6.2.5.2 Threat Identification

After identifying assets, organizations determine the threats that could potentially affect them.

Threats may arise from many sources.

External Threats

- Hackers
- Cybercriminal organizations
- Nation-state attackers
- Malware campaigns

Internal Threats

- Employee negligence
- Insider attacks
- Configuration mistakes
- Unauthorized activities

Environmental Threats

- Floods
- Earthquakes
- Fire
- Power failures

Technical Threats

- Hardware failure
- Software bugs
- Communication failures
- System overload

Industry Insight

Modern organizations increasingly use Cyber Threat Intelligence (CTI) platforms that collect information about newly discovered attack techniques and emerging cyber threats worldwide. This intelligence enables organizations to strengthen their defenses before attacks occur.

SGOU

6.2.5.3 Vulnerability Assessment

After identifying threats, organizations examine weaknesses that attackers might exploit.

Common vulnerabilities include:

- Weak passwords
- Missing software updates
- Open communication ports
- Misconfigured firewalls
- Insecure remote access
- Lack of employee awareness
- Unsupported operating systems

Organizations frequently perform:

- Vulnerability scanning
- Security audits
- Penetration testing
- Configuration reviews

to identify security weaknesses before attackers exploit them.



Fig. 6.2.5 A threat becomes dangerous when it exploits an existing vulnerability.

6.2.5.4 Risk Analysis

Risk analysis estimates the seriousness of identified risks.

Most organizations evaluate two important factors:

SGOU

- Likelihood – How likely is the attack?
- Impact – What would happen if the attack succeeds?

Risk increases when both likelihood and impact are high.

Simple Risk Formula

Although organizations use different models, a simplified expression is:

Risk = Likelihood × Impact

This formula helps compare multiple risks and prioritize mitigation efforts.

Example

Suppose ransomware attacks have become common in a manufacturing company.

Likelihood = High

Impact = Very High

Overall Risk = Very High

The organization should immediately implement additional security controls.

6.2.5.5 Risk Evaluation

Risk evaluation determines whether identified risks are acceptable or require immediate action.

Organizations typically classify risks into categories such as:

- Low
- Medium
- High
- Critical

Critical risks require urgent mitigation, while lower risks may be monitored until resources become available.

Table 6.2.2 Typical Risk Categories

Likelihood	Impact	Risk Level
Low	Low	Low
Low	High	Medium
Medium	Medium	Medium
High	High	High
Very High	Very High	Critical

SGOU

Every Risk Does Not Require Immediate Action

Some organizations mistakenly attempt to eliminate every possible risk. This is neither practical nor economically feasible.

Instead, organizations prioritize high-risk issues while continuously monitoring lower-risk situations.

6.2.5.6 Risk Treatment

Once risks have been evaluated, organizations decide how to manage them.

There are four commonly accepted risk treatment strategies.

Risk Avoidance

The organization eliminates the activity causing the risk.

Example:

A company decides not to expose industrial controllers directly to the Internet.

Risk Mitigation

Additional security controls reduce either the likelihood or impact of the risk.

Examples include:

- Firewalls
- Multi-factor authentication
- Encryption
- Network segmentation
- Security awareness training

Risk Transfer

Responsibility for certain risks is transferred to another organization.

Examples include:

- Cyber insurance
- Outsourced security services
- Managed Security Service Providers (MSSPs)

Risk Acceptance

Some risks are sufficiently small that organizations decide to accept them rather than invest in costly mitigation measures.

Accepted risks should still be monitored periodically.

SGOU



Fig.6.2.6 Common strategies for cybersecurity risk treatment.

6.2.6 Qualitative and Quantitative Risk Assessment

Organizations assess risks using two major approaches.

Qualitative Risk Assessment

Qualitative assessment uses descriptive terms rather than numerical values.

Typical ratings include:

- Very Low
- Low
- Medium
- High
- Very High

This method is widely used because it is simple and requires relatively little data.

Advantages

- Easy to understand
- Fast implementation
- Suitable for most organizations

SGOU

Limitations

- Subjective
- Depends on expert judgment

Quantitative Risk Assessment

Quantitative assessment uses numerical values and financial estimates.

Examples include:

- Expected monetary loss
- Annual loss expectancy
- Probability calculations
- Statistical models

Advantages

- Objective
- Supports financial decision-making
- Useful for cost-benefit analysis

Limitations

- Requires extensive data
- More time-consuming
- Complex calculations

Table 6.2.3 Qualitative vs Quantitative Risk Assessment

Feature	Qualitative	Quantitative
Measurement	Descriptive	Numerical
Complexity	Low	High
Cost	Lower	Higher
Accuracy	Moderate	Higher
Suitable For	General organizations	Large enterprises

SGOU

Problem

A company identifies two security risks:

Risk	Likelihood	Impact
Weak employee passwords	High	Medium
Failure of backup generator	Low	Very High

Which should receive higher priority?

Solution

Although the backup generator has a very high impact, its likelihood is low.

Weak employee passwords have a high likelihood and can be exploited frequently.

Therefore, both risks require attention, but password security should be addressed immediately through stronger authentication and employee awareness, while contingency planning should mitigate the backup generator risk.

Industry Example

A smart factory performs annual cybersecurity risk assessments.

During one assessment, engineers discover that:

- Several PLCs still use default passwords.
- Remote maintenance software lacks multi-factor authentication.
- Some operator workstations run unsupported operating systems.

Based on the risk assessment, management prioritizes:

1. Changing default passwords.
2. Implementing MFA.
3. Upgrading obsolete operating systems.

The result is a significant reduction in cybersecurity risk without replacing expensive industrial equipment.

6.2.7 OCTAVE Framework (Operationally Critical Threat, Asset, and Vulnerability Evaluation)

Organizations today face an increasing number of cybersecurity threats. Simply installing antivirus software or firewalls is no longer sufficient. Organizations must understand which assets are most critical, what threats exist, and how these threats can affect business operations.

SGOU

To address this challenge, researchers at the Software Engineering Institute (SEI), Carnegie Mellon University, developed the OCTAVE Framework.

OCTAVE is a self-directed, asset-based risk assessment methodology that helps organizations identify, evaluate, and manage cybersecurity risks. Unlike purely technical assessment methods, OCTAVE considers organizational objectives, operational processes, business requirements and human factors while assessing risks.

Definition

OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) is a structured information security risk assessment framework that enables organizations to identify critical assets, evaluate threats and vulnerabilities, and develop risk mitigation strategies based on organizational priorities.

Objectives of OCTAVE

The OCTAVE framework aims to:

- Identify critical organizational assets.
- Evaluate security threats affecting those assets.
- Identify vulnerabilities that attackers could exploit.
- Assess the impact of security incidents.
- Develop effective risk mitigation strategies.
- Support informed management decisions.

Unlike vulnerability scanning tools that focus mainly on technical weaknesses, OCTAVE provides a business-driven approach to risk assessment.



Fig. 6.2.7 OCTAVE focuses on protecting organizational assets by evaluating threats and vulnerabilities.

SGOU

6.2.7.1 Key Characteristics of OCTAVE

OCTAVE possesses several important characteristics that distinguish it from many traditional security assessment methods.

Asset-Centric Approach

The framework begins by identifying the organization's most valuable assets rather than focusing solely on technical vulnerabilities.

Self-Directed Assessment

Organizations perform the assessment themselves using internal experts who understand business operations, technologies and organizational processes.

Business-Focused

Risk assessment is aligned with organizational objectives and business priorities rather than only technical considerations.

Comprehensive Analysis

OCTAVE evaluates:

- Assets
- Threats
- Vulnerabilities
- Operational practices
- Organizational policies
- Technology infrastructure

Risk-Based Decision Making

Security investments are prioritized according to business risk rather than simply addressing every identified vulnerability.

Why Asset-Based Assessment?

Two organizations may use identical software systems but have completely different security priorities.

For example:

- A hospital considers patient records and medical equipment as critical assets.
- A bank considers financial transactions and customer accounts as critical assets.

SGOU

OCTAVE recognizes that different organizations require different security priorities.

6.2.8 OCTAVE Methodology

The OCTAVE methodology consists of three major phases.

Phase 1 – Build Asset-Based Threat Profiles

This phase identifies the organization's most important assets and determines the threats affecting them.

Major activities include:

- Identifying critical assets
- Understanding business requirements
- Identifying security requirements
- Developing threat profiles

Examples of critical assets include:

- Industrial control systems
- Customer databases
- Intellectual property
- Cloud services
- Financial systems



Fig. 6.2.8 Building asset-based threat profiles.

SGOU

Phase 2 – Identify Infrastructure Vulnerabilities

After identifying critical assets, organizations evaluate the technological infrastructure supporting those assets.

Activities include:

- Network assessment
- System configuration review
- Vulnerability identification
- Security control evaluation
- Infrastructure documentation

This phase helps identify technical weaknesses that attackers might exploit.

Phase 3 – Develop Security Strategy and Risk Mitigation Plans

The final phase combines information collected during the previous phases.

Organizations:

- Evaluate risks.
- Prioritize risks.
- Develop mitigation plans.
- Allocate security resources.
- Monitor implementation.

This phase produces a practical cybersecurity improvement plan.

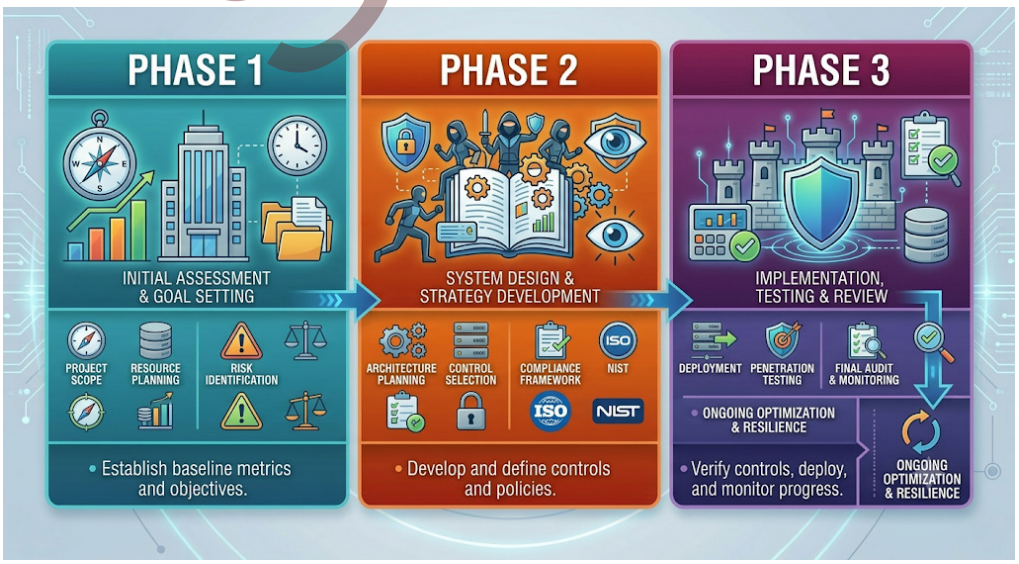


Fig. 6.2.9 Three phases of the OCTAVE framework.

SGOU

Table 6.2.4 OCTAVE Phases

Phase	Purpose	Major Activities
Phase 1	Identify Critical Assets	Asset identification, threat profiling
Phase 2	Evaluate Infrastructure	Vulnerability assessment
Phase 3	Risk Management	Mitigation planning and security strategy

6.2.9 OCTAVE Allegro

As technology evolved, organizations required a simpler and faster risk assessment approach.

This led to the development of OCTAVE Allegro.

Unlike the original OCTAVE methodology, OCTAVE Allegro places greater emphasis on information assets and requires less technical analysis, making it suitable for organizations with limited cybersecurity resources.

Major Features of OCTAVE Allegro

- Simplified methodology
- Faster implementation
- Information asset focus
- Lower implementation cost
- Suitable for small and medium organizations
- Reduced dependence on technical infrastructure analysis

Table 6.2.5 Comparison between OCTAVE and OCTAVE Allegro

Feature	OCTAVE	OCTAVE Allegro
Assessment Scope	Organization-wide	Information assets
Complexity	Higher	Lower
Technical Analysis	Extensive	Limited
Time Requirement	Longer	Shorter
Suitable Organizations	Large enterprises	Small and medium organizations

Industry Insight

Many organizations begin with OCTAVE Allegro to establish a basic cybersecurity risk management process. As cybersecurity maturity improves, they adopt the full OCTAVE methodology for more comprehensive organizational assessments.

SGOU

Advantages of OCTAVE

The OCTAVE framework offers several important benefits.

- Asset-oriented methodology
- Business-driven decision making
- Comprehensive risk assessment
- Encourages organizational participation
- Flexible implementation
- Suitable for various industries
- Supports long-term cybersecurity planning

Limitations of OCTAVE

Despite its strengths, OCTAVE has certain limitations.

- Requires experienced personnel.
- Time-consuming for large organizations.
- Extensive documentation is needed.
- Results depend on organizational expertise.
- Less suitable for rapid technical vulnerability assessments.

Case Study

Risk Assessment in a Smart Manufacturing Company

A manufacturing company wants to strengthen the cybersecurity of its Industrial Internet of Things (IIoT) infrastructure.

Using the OCTAVE framework:

Phase 1

Critical assets identified:

- PLC controllers
- SCADA servers
- Production databases
- Industrial robots

SGOU

Phase 2

Security weaknesses identified:

- Weak administrator passwords
- Unpatched servers
- Lack of network segmentation
- Inadequate employee training

Phase 3

Mitigation plan:

- Implement Multi-Factor Authentication.
- Segment industrial networks.
- Upgrade obsolete operating systems.
- Conduct employee awareness programmes.
- Establish continuous security monitoring.

Within one year, cybersecurity incidents decreased significantly while production reliability improved.

Problem

A hospital wants to perform a cybersecurity assessment focusing primarily on protecting its most valuable assets rather than beginning with vulnerability scanning.

Which risk assessment framework is more appropriate?

Solution

The OCTAVE Framework is the appropriate choice because it begins by identifying organizational assets, evaluating threats affecting those assets, and then developing mitigation strategies based on business priorities.

6.2.10 FAIR Framework (Factor Analysis of Information Risk)

As organizations become increasingly dependent on digital technologies, managers need to understand cybersecurity risks not only from a technical perspective but also in terms of business and financial impact. Questions such as "How much money could we lose if this cyber attack succeeds?" or "Is investing in a new security solution financially justified?" are common in boardroom discussions.

To answer such questions, cybersecurity professionals use the FAIR Framework.

The Factor Analysis of Information Risk (FAIR) framework is an internationally recognized methodology that provides a quantitative approach to information risk

SGOU

analysis. Instead of relying only on descriptive terms such as "high risk" or "medium risk," FAIR estimates risk using measurable factors, helping organizations make informed business decisions.

Definition

FAIR (Factor Analysis of Information Risk) is a quantitative risk analysis framework that measures information security risk by estimating the probable frequency and probable magnitude of future loss.

Objectives of FAIR

The FAIR framework aims to:

- Quantify cybersecurity risks.
- Estimate the financial impact of cyber incidents.
- Support business decision-making.
- Prioritize cybersecurity investments.
- Improve communication between technical experts and business managers.

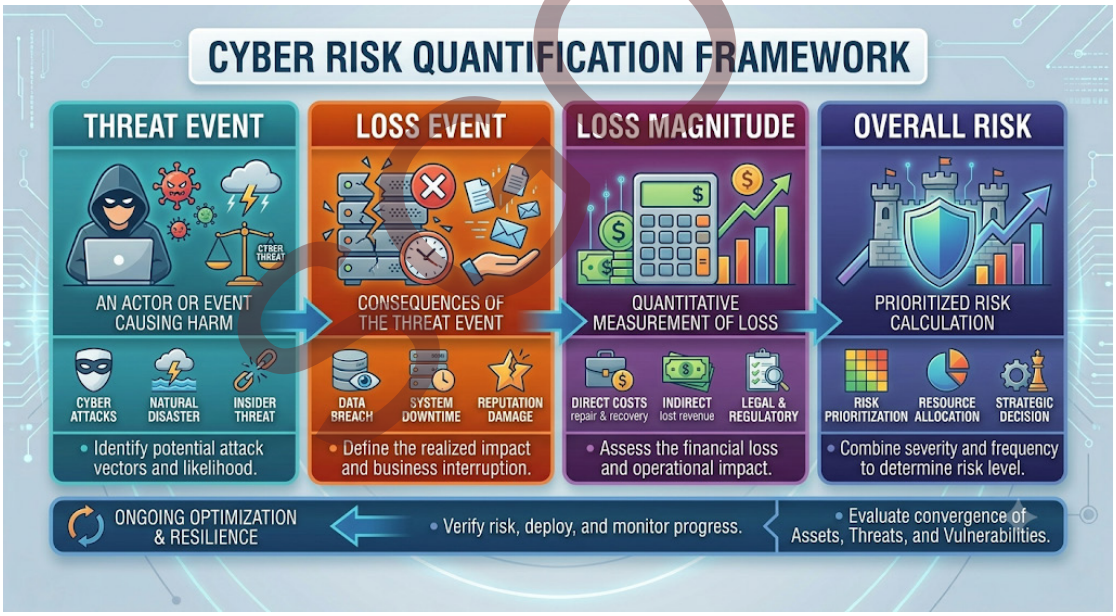


Fig. 6.2.10 FAIR estimates cybersecurity risk by analysing the likelihood and financial impact of loss events.

6.2.11 Key Concepts in FAIR

The FAIR model is based on several important concepts.

Asset

An asset is anything valuable to an organization.

Examples include:

- Industrial control systems
- Customer databases
- Cloud infrastructure
- Financial records
- Production servers

Threat

A threat is any entity capable of causing harm.

Examples include:

- Cybercriminals
- Malicious insiders
- Hackers
- Malware
- Natural disasters

Threat Event

A threat event occurs when a threat attempts to act against an asset.

Example:

A hacker attempts to gain unauthorized access to an industrial control server.

Loss Event

A loss event occurs when the threat successfully compromises the asset.

Example:

The attacker successfully disables a production server, causing manufacturing downtime.

Loss Magnitude

Loss magnitude represents the expected financial consequences of a successful attack.

Possible losses include:

- Revenue loss
- Equipment damage
- Legal penalties

SGOU

- Recovery costs
- Reputation damage
- Productivity loss

Threat Event Frequency

Threat Event Frequency (TEF) estimates how often threat events are expected to occur.

For example:

- Daily phishing attempts
- Weekly malware scans
- Annual ransomware campaigns

Vulnerability

In FAIR, vulnerability represents the probability that a threat will successfully exploit an asset under specific conditions.

Why Quantify Cybersecurity Risk?

Senior managers often understand financial values better than technical terminology. Saying that a cyber incident could cost an organization ₹5 crore provides clearer guidance for investment decisions than simply describing the risk as "high."

6.2.12 FAIR Risk Analysis Process

The FAIR methodology follows a systematic process.

Step 1 – Identify the Asset

Determine which valuable asset requires protection.

Example:

- SCADA Server
- PLC Controller
- Production Database
- Customer Information System

Step 2 – Identify Threat Scenarios

Determine possible threats.

Examples:

- Malware
- Insider attack
- Ransomware
- Unauthorized remote access

SGOU

Step 3 – Estimate Threat Event Frequency

Estimate how frequently threats are likely to occur.

Step 4 – Estimate Vulnerability

Determine the probability that the identified threat will successfully exploit the vulnerability.

Step 5 – Estimate Loss Magnitude

Estimate potential losses.

Examples include:

- Financial loss
- Operational downtime
- Equipment damage
- Customer dissatisfaction
- Legal penalties

Step 6 – Calculate Risk

Risk is estimated by combining:

- Probability of occurrence
- Expected loss

This provides a measurable estimate of organizational risk.

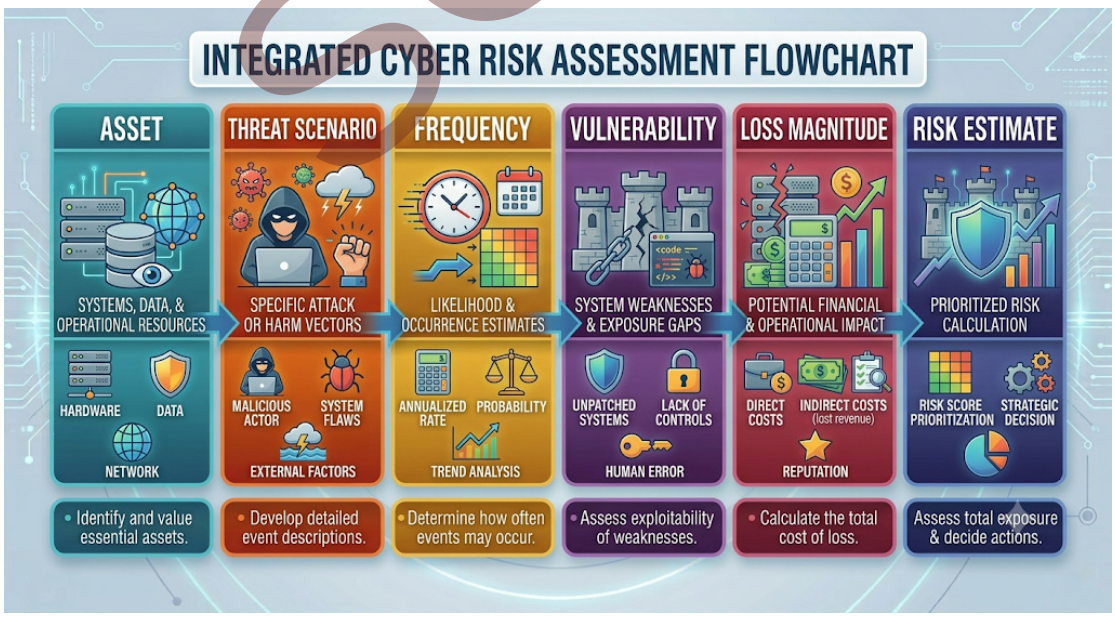


Fig. 6.2.11 FAIR quantitative risk assessment process.

SGOU

6.2.13 Advantages of FAIR

The FAIR framework offers several important advantages.

- Quantitative methodology
- Supports financial analysis
- Improves executive decision-making
- Enables cost-benefit analysis
- Prioritizes cybersecurity investments
- Widely accepted internationally
- Consistent and repeatable assessment process

Limitations of FAIR

Some challenges include:

- Requires reliable organizational data
- More complex calculations
- Financial estimation may be difficult
- Requires experienced analysts
- Time-consuming for beginners

Table 6.2.6 OCTAVE vs FAIR

Feature	OCTAVE	FAIR
Approach	Qualitative, Asset-Based	Quantitative, Financial
Primary Focus	Organizational assets	Financial risk
Developed By	Carnegie Mellon SEI	FAIR Institute
Complexity	Moderate	Higher
Risk Measurement	Descriptive	Numerical
Best For	Organizational security planning	Financial risk analysis
Output	Risk mitigation strategy	Quantified business risk

SGOU

Industry Insight

Many multinational organizations combine OCTAVE and FAIR in their cybersecurity programmes. OCTAVE helps identify and prioritize critical assets, while FAIR estimates the financial consequences of identified risks. Together, they provide a comprehensive approach to cybersecurity risk management.

Case Study

Cyber Risk Assessment in a Smart Manufacturing Plant

A smart manufacturing company operates multiple production lines controlled by SCADA systems and PLCs. Management plans to invest in advanced cybersecurity measures but wants to determine whether the investment is economically justified.

Using OCTAVE

The security team:

- Identifies critical assets.
- Examines vulnerabilities.
- Prioritizes operational risks.
- Develops mitigation strategies.

Using FAIR

The finance and cybersecurity teams estimate:

- Expected ransomware frequency.
- Probability of successful attacks.
- Estimated production downtime.
- Financial losses due to interrupted manufacturing.

The FAIR analysis predicts that a major ransomware incident could result in losses of several crores of rupees. Based on this analysis, management approves additional investments in network segmentation, security monitoring, and employee awareness training.

This case demonstrates how OCTAVE and FAIR complement each other in practical cybersecurity planning.

Problem

A financial institution wants to estimate the monetary impact of ransomware attacks before deciding whether to purchase cyber insurance.

Which framework is most appropriate?

SGOU

Solution

The FAIR Framework is the appropriate choice because it focuses on quantifying cybersecurity risk by estimating the probability and financial impact of future loss events. This enables informed decisions regarding cyber insurance and security investments.

Recap

- Risk analysis helps organizations identify, evaluate, and prioritize cybersecurity risks.
- Risk arises when valuable assets are exposed to threats that exploit vulnerabilities.
- A formal risk assessment process includes asset identification, threat identification, vulnerability assessment, risk analysis, evaluation, treatment, and continuous monitoring.
- Risk treatment strategies include avoidance, mitigation, transfer, and acceptance.
- Qualitative risk assessment uses descriptive ratings, while quantitative assessment uses numerical estimates.
- The OCTAVE Framework is an asset-based, qualitative methodology that aligns cybersecurity decisions with organizational objectives.
- OCTAVE consists of three phases: building asset-based threat profiles, identifying infrastructure vulnerabilities, and developing mitigation strategies.
- OCTAVE Allegro simplifies risk assessment by focusing on information assets.
- The FAIR Framework provides a quantitative approach to measuring cybersecurity risk using probable frequency and probable loss magnitude.
- OCTAVE and FAIR complement each other and are often used together to support effective cybersecurity governance and business decision-making.

Objective Type Questions

1. Define risk analysis.
2. What is an organizational asset?
3. What is a vulnerability?

SGOU

4. What are the four risk treatment strategies?
5. What is qualitative risk assessment?
6. Expand OCTAVE.
7. Who developed the OCTAVE framework?
8. Name the three phases of OCTAVE.
9. What is OCTAVE Allegro?
10. Expand FAIR.
11. What is Threat Event Frequency?
12. What is Loss Magnitude?
13. Which framework estimates financial risk?
14. Differentiate between OCTAVE and FAIR.
15. Why are formal risk analysis frameworks important?

Answers

1. Systematic identification and evaluation of risks.
2. Anything valuable to an organization.
3. A weakness that can be exploited.
4. Avoid, Mitigate, Transfer, and Accept.
5. Assessment using descriptive ratings.
6. Operationally Critical Threat, Asset, and Vulnerability Evaluation.
7. Software Engineering Institute (SEI), Carnegie Mellon University.
8. Asset-Based Threat Profiles, Infrastructure Vulnerabilities, Security Strategy.
9. A simplified version of OCTAVE.
10. Factor Analysis of Information Risk.
11. Frequency of expected threat events.
12. Estimated financial consequences of a successful attack.
13. FAIR.

SGOU

14. OCTAVE is qualitative and asset-based; FAIR is quantitative and financially oriented.

15. They help organizations prioritize cybersecurity risks and allocate resources effectively.

Assignments

1. Explain the concept of cybersecurity risk analysis and discuss its importance in Operational Technology environments.
2. Describe the complete risk assessment process with suitable industrial examples.
3. Explain the OCTAVE framework and discuss its three phases in detail.
4. Discuss the features, advantages, and limitations of OCTAVE Allegro.
5. Explain the FAIR framework and describe its major concepts with suitable examples.
6. Compare OCTAVE and FAIR with respect to methodology, objectives, and applications.
7. Discuss how OCTAVE and FAIR can be used together to improve organizational cybersecurity.
8. Explain the importance of formal risk analysis frameworks in Industrial Internet of Things (IIoT) environments.

SGOU

Suggested Reading

1. FAIR Institute. *FAIR™ Book of Knowledge (FAIR-BOK)*.
2. Carnegie Mellon University, Software Engineering Institute. *OCTAVE Allegro User Guide*.
3. Stallings, William. *Effective Cybersecurity: A Guide to Using Best Practices and Standards*. Pearson.
4. Whitman, Michael E., and Herbert J. Mattord. *Principles of Information Security*. Cengage Learning.
5. NIST. *Framework for Improving Critical Infrastructure Cybersecurity*.

Reference

1. Alberts, Christopher, and Audrey Dorofee. *OCTAVE® Method Implementation Guide*. Carnegie Mellon University, Software Engineering Institute.
2. Jones, Jack A., and Jack Freund. *Measuring and Managing Information Risk: A FAIR Approach*. Butterworth-Heinemann, 2014.
3. National Institute of Standards and Technology. *Guide for Conducting Risk Assessments (NIST SP 800-30 Rev. 1)*.
4. ISO. *ISO/IEC 27005: Information Security Risk Management*. International Organization for Standardization.
5. Stoneburner, Gary, et al. *Risk Management Guide for Information Technology Systems*. NIST SP 800-30.

SGOU



Arduino – Basics on Integration of Sensors and Actuators

Learning Outcomes

After completion of this unit, learner will be able to:

- define Arduino and explain its significance in embedded system development
- describe the architecture and major components of an Arduino board
- explain the features and advantages of Arduino
- install and use the Arduino Integrated Development Environment (IDE)
- develop simple Arduino programs
- select appropriate Arduino boards for different IoT applications

Prerequisite

Imagine you want to build an automatic street light that turns ON when it becomes dark and switches OFF automatically during daylight. Alternatively, consider designing a smart irrigation system that waters plants only when the soil becomes dry.

Can an ordinary desktop computer perform these tasks directly?

The answer is No.

A desktop computer is designed mainly for general-purpose computing. It cannot directly interact with sensors and actuators placed in the physical environment.

To solve such problems, engineers use microcontroller boards. Among the most popular microcontroller platforms is Arduino, which has transformed the way students, hobbyists, researchers, and industries develop embedded systems and Internet of Things (IoT) applications.

Arduino makes electronics and programming simple, affordable, and accessible. It enables users to connect sensors, read environmental data, control actuators, and communicate with other devices with minimal hardware and software complexity.

In this unit, we will explore Arduino from the basics and learn how it integrates sensors and actuators to create intelligent IoT systems.

SGOU

Before beginning this unit, you should be familiar with:

- Basic computer programming concepts
- Digital electronics fundamentals
- Sensors and actuators
- Internet of Things basics

Keywords

Arduino, Microcontroller, IDE, Digital Pin, Analog Pin, Sketch, GPIO, Embedded System

Discussion

6.3.1 Introduction to Arduino

The rapid growth of embedded systems and IoT has created a demand for simple, low-cost, and flexible hardware platforms. Arduino fulfills this need by providing an open-source electronics platform consisting of programmable microcontroller boards and easy-to-use software.

Arduino enables users to design interactive electronic systems without requiring advanced knowledge of hardware engineering.

Thousands of educational institutions worldwide use Arduino because it allows beginners to quickly develop real-world electronic applications.

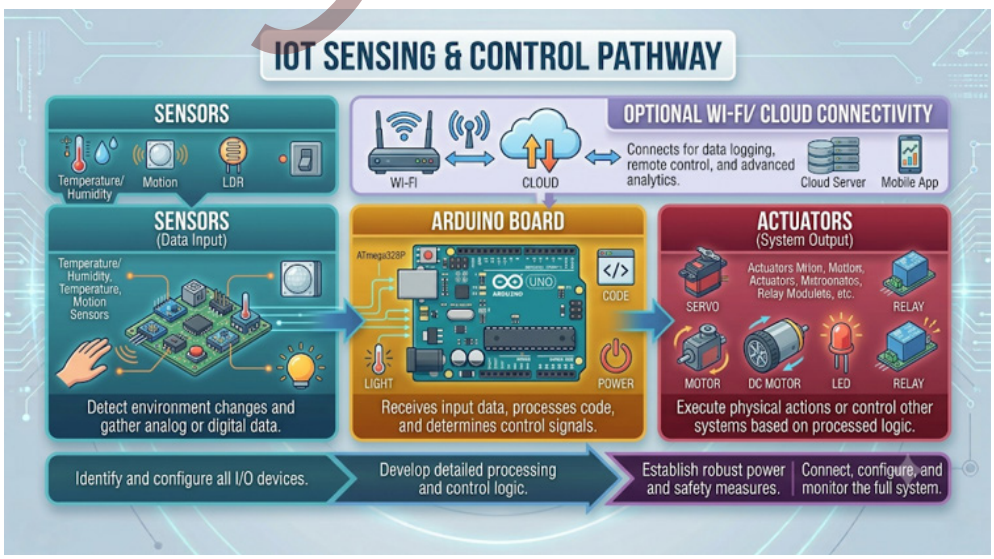


Fig.6.3.1 Arduino-Based Embedded System

SGOU

Definition

Arduino is an open-source electronics platform based on programmable microcontroller boards and software that enables users to create interactive electronic systems by integrating sensors, actuators, communication modules, and control programs.

Figure 6.3.1 Arduino acts as the central controller connecting sensors, actuators, and communication modules.

Why is Arduino Popular?

Arduino has become one of the world's most widely used embedded development platforms because it offers:

- Open-source hardware
- Open-source software
- Low cost
- Easy programming
- Extensive community support
- Large number of compatible sensors and modules
- Cross-platform software support
- Ideal platform for IoT prototyping

Applications of Arduino

Arduino is widely used in:

- Smart homes
- Smart agriculture
- Healthcare monitoring
- Industrial automation
- Robotics
- Environmental monitoring
- Wearable devices
- Educational laboratories
- Internet of Things (IoT) systems

SGOU

Industry Insight

Many successful IoT startups begin their product development using Arduino because it enables rapid prototyping. Once the design is validated, the prototype is often converted into a custom embedded hardware platform for commercial production.

6.3.2 History and Evolution of Arduino

Arduino originated in 2005 at the Interaction Design Institute Ivrea (IDII) in Italy.

The objective was to develop a low-cost microcontroller platform that students could easily program without expensive development tools.

The project was led by Massimo Banzi, along with David Cuartielles, David Mellis, Tom Igoe, Gianluca Martino, and others.

Since its introduction, Arduino has evolved into one of the world's largest open-source hardware ecosystems.

Today, Arduino supports:

- Artificial Intelligence projects
- Internet of Things applications
- Robotics
- Smart agriculture
- Industrial automation
- Edge computing
- STEM education

Table 6.3.1 Timeline of Arduino Development

Year	Development
2005	Arduino project introduced
2010	Arduino Uno prototype developed and Large community growth
2012	IoT applications increased
2015	Arduino MKR series introduced
2020–Present	AI, IoT, Edge Computing and Industry 4.0 applications

SGOU

Did You Know?

The name Arduino comes from a historic bar named Bar di Re Arduino in Ivrea, Italy, where the founders frequently met during the development of the platform.

6.3.3 Features of Arduino

Arduino offers several features that distinguish it from traditional embedded development platforms.

Open-Source Platform

Both the hardware design and software are openly available, allowing users to modify and customize the platform.

Easy Programming

Arduino uses a simplified version of C/C++, making programming easy for beginners.

Cross-Platform Support

The Arduino IDE runs on:

- Windows
- Linux
- macOS

Large Community

Millions of developers worldwide contribute tutorials, projects, libraries, and technical support.

Extensive Library Support

Thousands of ready-made libraries simplify the use of:

- LCD displays
- Wi-Fi modules
- Bluetooth
- GPS
- RFID
- Temperature sensors
- Motor drivers

SGOU

Low Cost

Most Arduino boards are inexpensive, making them suitable for educational institutions and prototype development.

Rapid Prototyping

Arduino enables engineers to develop and test new electronic systems quickly without designing complex hardware.

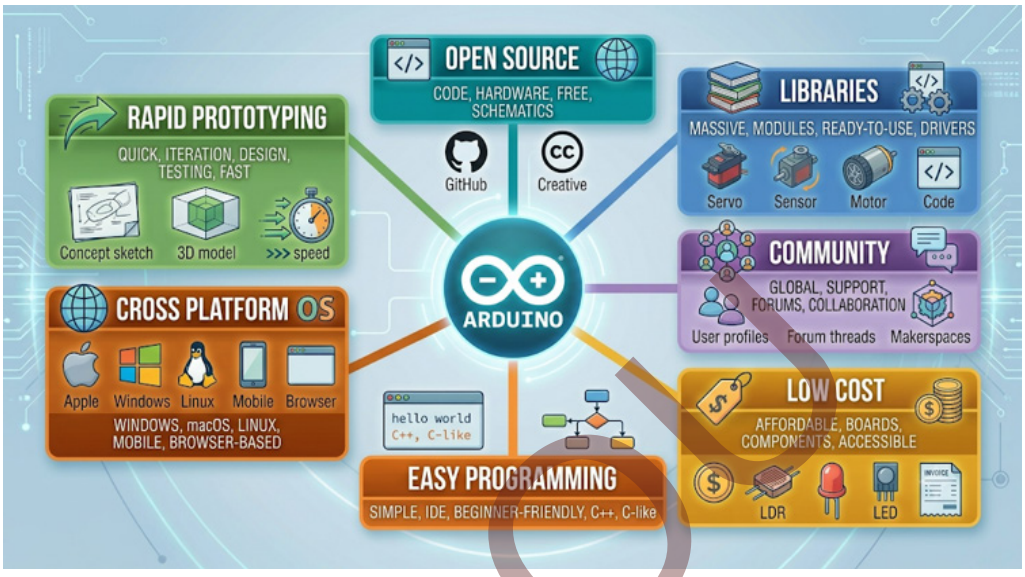


Fig. 6.3.2 Major features contributing to Arduino's popularity.

6.3.4 Hardware Architecture of Arduino

Although several Arduino models are available, they share a common architecture.

An Arduino board typically consists of:

- Microcontroller
- Digital Input/Output Pins
- Analog Input Pins
- USB Interface
- Power Supply Circuit
- Crystal Oscillator
- Voltage Regulator
- Reset Circuit
- Communication Interfaces

SGOU

The microcontroller acts as the brain of the board, executing user programs and controlling connected devices.

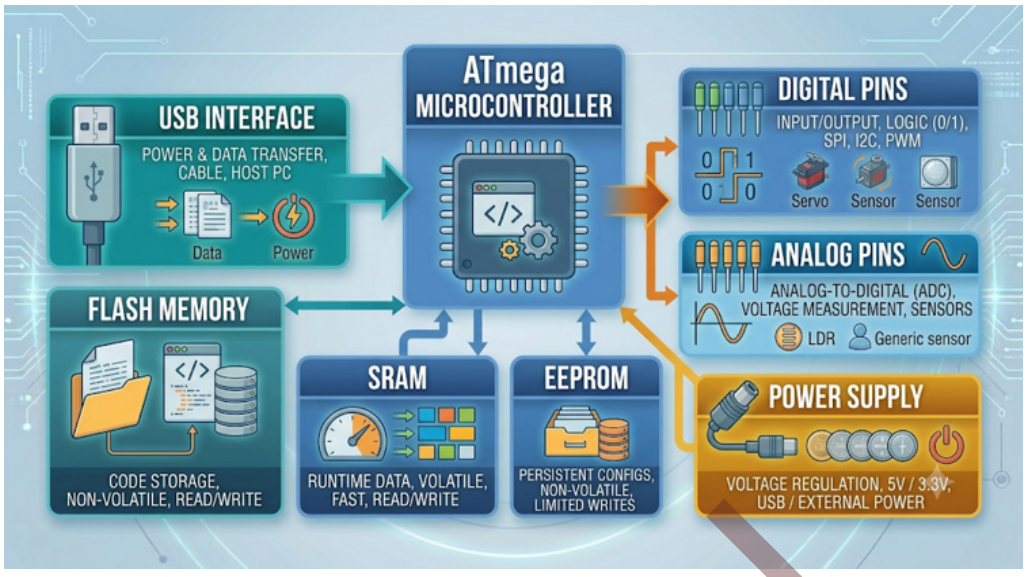


Fig.6.3.3 General architecture of an Arduino board.

Main Hardware Components

Microcontroller

The microcontroller executes the uploaded program (called a Sketch) and controls all board operations.

Many Arduino Uno boards use the ATmega328P microcontroller.

Digital Input/Output Pins

Digital pins can operate in two states:

- HIGH (5V)
- LOW (0V)

Applications include:

- LED control
- Relay switching
- Push-button input
- Buzzer control

Analog Input Pins

Analog pins measure continuously varying voltages.

SGOU

Applications include:

- Temperature sensing
- Light sensing
- Soil moisture sensing
- Gas sensing

USB Port

The USB interface is used for:

- Program uploading
- Serial communication
- Power supply

Power Supply

Arduino boards can be powered through:

- USB cable
- External adapter
- Battery
- DC power supply

Solved Illustration

Problem

A student wants to build a temperature monitoring system using Arduino.

Which Arduino pin should be used to connect the analog output of a temperature sensor?

Solution

Since the temperature sensor provides a continuously varying voltage, it should be connected to one of the Analog Input Pins (A0–A5) on the Arduino Uno. The microcontroller reads the analog signal using its built-in Analog-to-Digital Converter (ADC) and processes the temperature data.

6.3.5 Arduino Uno Board in Detail

Among all Arduino development boards, the Arduino Uno is the most popular because of its simplicity, affordability, and extensive community support. It is widely used by students, researchers, hobbyists, and industries for developing embedded systems and Internet of Things (IoT) applications.

SGOU

The Arduino Uno is based on the ATmega328P microcontroller and provides all the essential features required for learning embedded programming and developing real-time control systems.

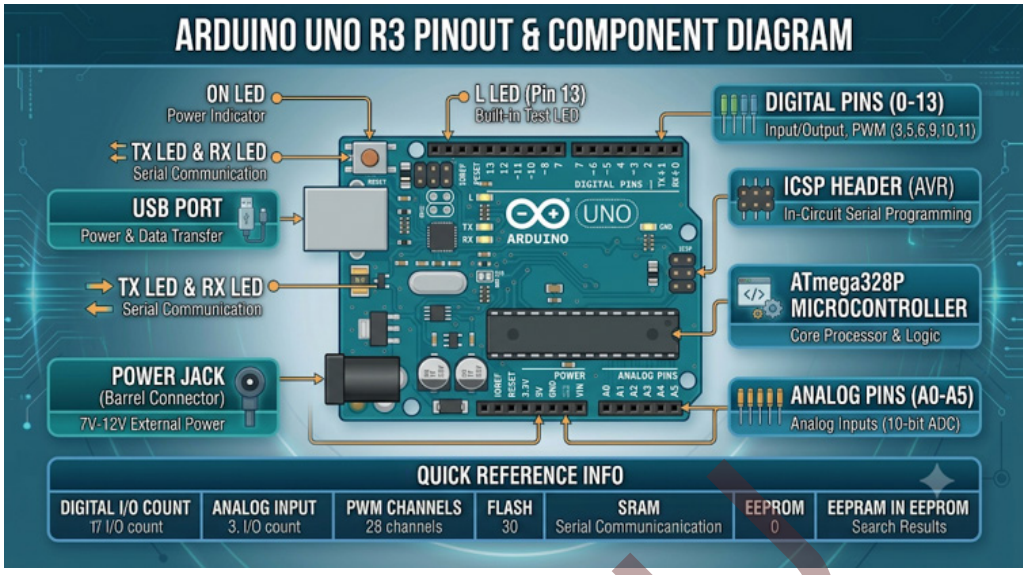


Fig. 6.3.4 Major components of the Arduino Uno development board.

Table 6.3.2 Technical Specifications of Arduino Uno

Feature	**Specification**
Microcontroller	ATmega328P
Operating Voltage	5 V
Input Voltage (Recommended)	7–12 V
Digital I/O Pins	14 (6 PWM)
Analog Input Pins	6
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
USB Interface	USB Type-B
Communication	UART, SPI, I²C

SGOU

Advantages of Arduino Uno

- Beginner-friendly
- Easy programming
- Low cost
- Large online community
- Wide sensor compatibility
- Extensive software libraries
- Ideal for IoT prototyping

6.3.6 Pin Configuration of Arduino Uno

The Arduino Uno contains several categories of pins, each designed for specific purposes.

Digital Input/Output Pins

Arduino Uno provides 14 digital pins, numbered 0 to 13.

These pins can function either as:

- Digital Inputs
- Digital Outputs

Digital pins operate using two logic levels:

- HIGH (5V)
- LOW (0V)

Applications

- LED control
- Push-button input
- Relay switching
- Buzzer control
- Digital sensors

PWM Pins

Some digital pins support Pulse Width Modulation (PWM).

SGOU

PWM Pins:

3, 5, 6, 9, 10 and 11

PWM enables simulation of analog output by rapidly switching digital signals ON and OFF.

Applications

- LED brightness control
- Motor speed control
- Servo motor operation

Analog Input Pins

Arduino Uno provides six analog pins (A0–A5).

Unlike digital pins, analog pins measure continuously varying voltages.

The ATmega328P converts analog signals into digital values using its built-in Analog-to-Digital Converter (ADC).

Applications

- Temperature sensor
- Soil moisture sensor
- Gas sensor
- Light sensor (LDR)
- Potentiometer

Power Pins

Power pins include:

- 5V
- 3.3V
- GND
- Vin

These pins supply power to sensors and external modules connected to the Arduino.

Reset Pin

The Reset pin restarts the microcontroller and begins execution of the uploaded program from the beginning.

SGOU

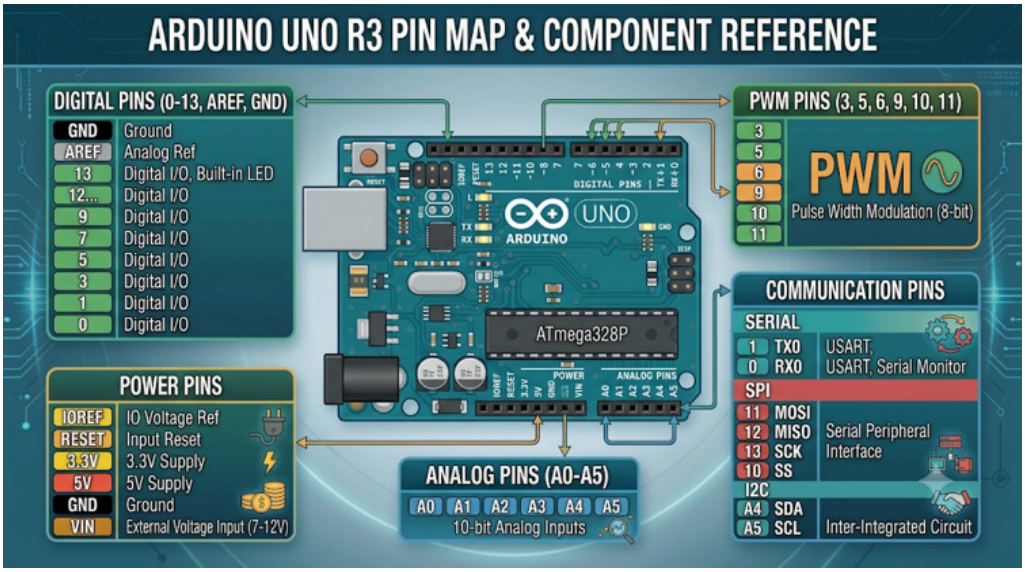


Fig.6.3.5 Pin configuration of the Arduino Uno board.

6.3.7 Memory Organization

Like a computer, the Arduino Uno uses different types of memory to perform various tasks.

Flash Memory

Flash Memory stores the program (Sketch) permanently.

Characteristics:

- Non-volatile
- Retains data even after power is removed
- Stores user programs

Capacity:

32 KB

SRAM (Static Random Access Memory)

SRAM stores temporary variables while the program is running.

Characteristics:

- Volatile
- Contents disappear after power is switched OFF
- Used for calculations and intermediate data

SGOU

Capacity:

2 KB

EEPROM (Electrically Erasable Programmable Read Only Memory)

EEPROM stores small amounts of permanent data.

Examples:

- Device settings
- Calibration values
- User preferences

Characteristics:

- Non-volatile
- Data remains after power is removed

Capacity:

1 KB

Table 6.3.3 Arduino Memory Comparison

Memory	Capacity	Volatile	Typical Use
Flash	32 KB	No	Program storage
SRAM	2 KB	Yes	Variables during execution
EEPROM	1 KB	No	Permanent settings

Why is SRAM Important?

Many beginners focus only on Flash memory because it stores the program. However, most Arduino projects fail due to insufficient SRAM, especially when large arrays, strings, or multiple sensor libraries consume the limited 2 KB available on the Arduino Uno.

6.3.8 Power Supply Options

Arduino Uno supports multiple power supply methods.

USB Power

The simplest method.

SGOU

Advantages:

- Easy
- Convenient
- Used during programming

Voltage:

5 V

DC Barrel Jack

External power adapter:

Recommended:

7–12 V

Suitable for standalone projects.

Vin Pin

External regulated voltage may be supplied directly through the Vin pin.

Battery Supply

Portable Arduino applications often use:

- 9V battery
- Lithium-ion battery
- Power bank
- Solar power system

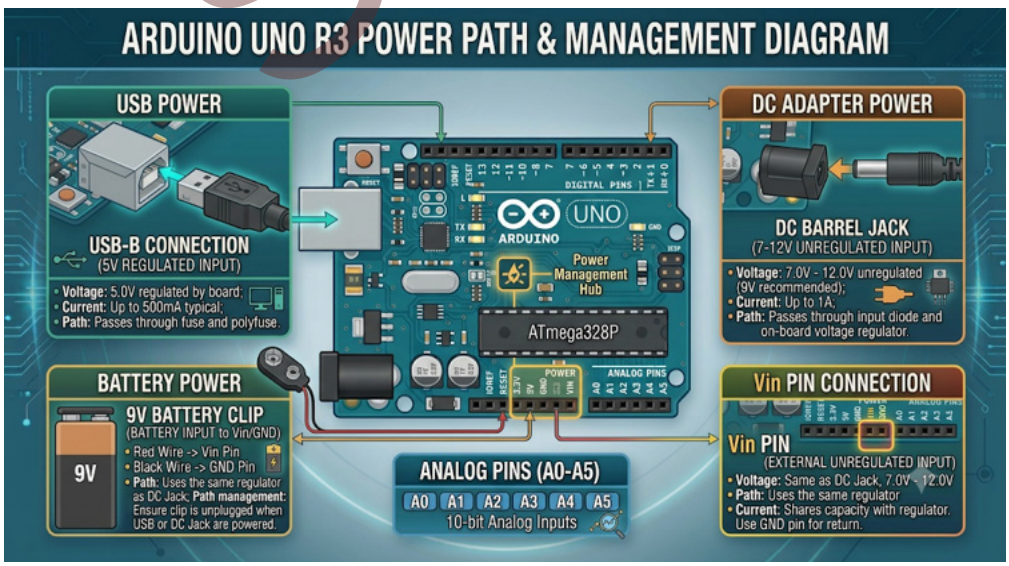


Fig. 6.3.6 Different methods of powering an Arduino Uno.

SGOU

6.3.9 Clock System

The Arduino Uno uses a 16 MHz crystal oscillator.

The clock synchronizes all internal operations.

Functions include:

- Instruction execution
- Timer operation
- Serial communication
- Pulse generation

A higher clock frequency generally enables faster program execution.

6.3.10 Communication Interfaces

Modern IoT systems require communication with sensors, displays, wireless modules and other microcontrollers.

Arduino Uno supports three important communication protocols.

UART (Universal Asynchronous Receiver Transmitter)

UART enables serial communication between Arduino and computers or other devices.

Applications

- Serial Monitor
- Bluetooth modules
- GSM modules
- GPS receivers

Pins

- TX (Pin 1)
- RX (Pin 0)

SPI (Serial Peripheral Interface)

SPI provides high-speed communication.

Applications

- SD cards
- RFID modules

SGOU

- TFT displays
- Ethernet modules

SPI Pins

- MOSI
- MISO
- SCK
- SS

I²C (Inter-Integrated Circuit)

I²C enables communication with multiple devices using only two wires.

Pins

- SDA
- SCL

Applications

- OLED displays
- RTC modules
- EEPROM
- Environmental sensors

Table 6.3.4 Comparison of Communication Interfaces

Interface	Speed	Wires	Typical Applications
UART	Moderate	2	Serial communication
SPI	High	4	SD Cards, Displays
I²C	Moderate	2	Sensors, RTC, OLED

Solved Illustration

Problem

A student wants to connect a temperature sensor, OLED display, and SD card to the same Arduino Uno.

Which communication interfaces should be used?

SGOU

Solution

- Temperature Sensor: Analog pin (A0) or I²C, depending on the sensor model.
- OLED Display: I²C (SDA and SCL).
- SD Card Module: SPI (MOSI, MISO, SCK, SS).

Using the appropriate interface ensures efficient communication and minimizes wiring complexity.

Industry Insight

Although advanced IoT boards such as ESP32 and Raspberry Pi Pico include built-in Wi-Fi and Bluetooth, the Arduino Uno remains one of the most widely used teaching and prototyping platforms because of its simplicity, reliability, and extensive ecosystem of compatible sensors and modules.

6.3.11 Arduino Integrated Development Environment (IDE)

Writing a program is only one part of developing an embedded system. The program must also be compiled, checked for errors, and uploaded to the Arduino board. These tasks are performed using the Arduino Integrated Development Environment (IDE).

The Arduino IDE is a free, open-source software application that allows users to write, edit, compile, upload, and debug Arduino programs, commonly called Sketches.

The IDE is designed to be simple and user-friendly, making it suitable for beginners while still offering powerful features for experienced developers.

Definition

Arduino IDE is a software development environment used to create, compile, upload, and monitor Arduino programs.

Major Features of Arduino IDE

- Open-source software
- Easy-to-use graphical interface
- Built-in code editor
- Error checking during compilation
- One-click uploading
- Serial Monitor
- Serial Plotter
- Library Manager

SGOU

- Board Manager
- Cross-platform support

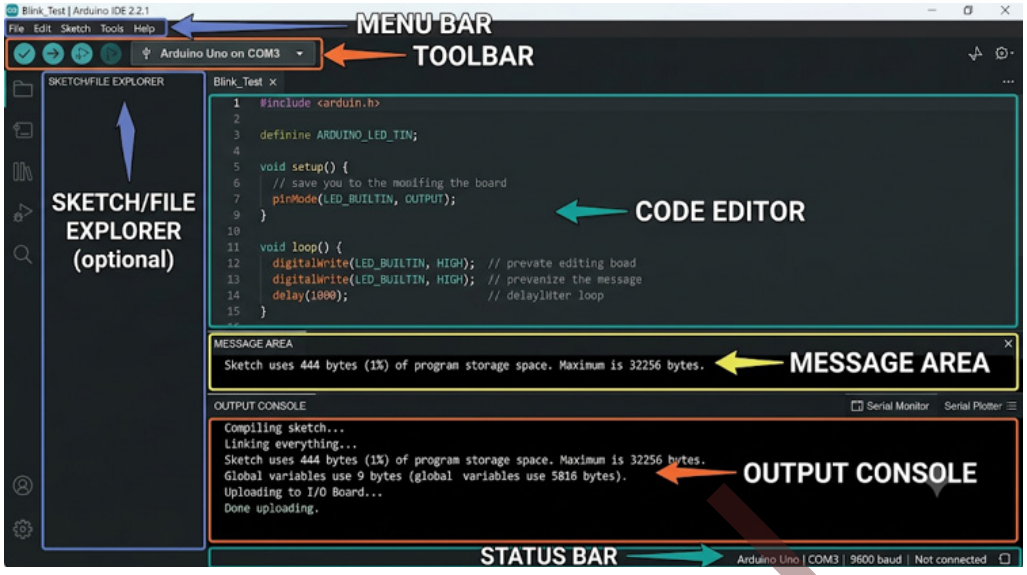


Fig.6.3.7 Main interface of the Arduino Integrated Development Environment.

Installing Arduino IDE

The Arduino IDE can be downloaded free of cost from the official Arduino website.

Installation Steps

1. Download the latest Arduino IDE.
2. Install the software.
3. Connect the Arduino board using a USB cable.
4. Install the required USB drivers (if necessary).
5. Open the Arduino IDE.
6. Select the correct Board.
7. Select the appropriate COM Port.
8. Create a new sketch.

After installation, the IDE is ready for developing Arduino applications.

Industry Insight

Professional embedded engineers often use advanced development environments such as PlatformIO, Keil, or Microchip Studio for commercial products. However, the Arduino IDE remains one of the most widely used platforms for education, rapid prototyping, and IoT research due to its simplicity and extensive library support.

SGOU

6.3.12 Arduino IDE Interface

The Arduino IDE consists of several important sections.

Menu Bar

Provides access to:

- File
- Edit
- Sketch
- Tools
- Help

Toolbar

Contains shortcut buttons for frequently used operations.

Common buttons include:

- Verify (Compile)
- Upload
- New Sketch
- Open
- Save
- Serial Monitor

Code Editor

The central area where Arduino programs are written.

Output Console

Displays:

- Compilation results
- Upload progress
- Error messages
- Memory usage

Status Bar

Displays board information and communication status.

SGOU

6.3.13 Arduino Program Structure

Every Arduino program follows a standard structure consisting of two mandatory functions.

```
void setup()
{
}
void loop()
{
}
```

setup () Function

The setup() function executes only once when the Arduino starts or resets.

Typical tasks include:

- Initializing pins
- Starting serial communication
- Configuring sensors
- Setting communication speed

loop () Function

The loop() function executes continuously until power is removed.

Typical tasks include:

- Reading sensors
- Controlling LEDs
- Operating motors
- Sending data
- Displaying information

SGOU

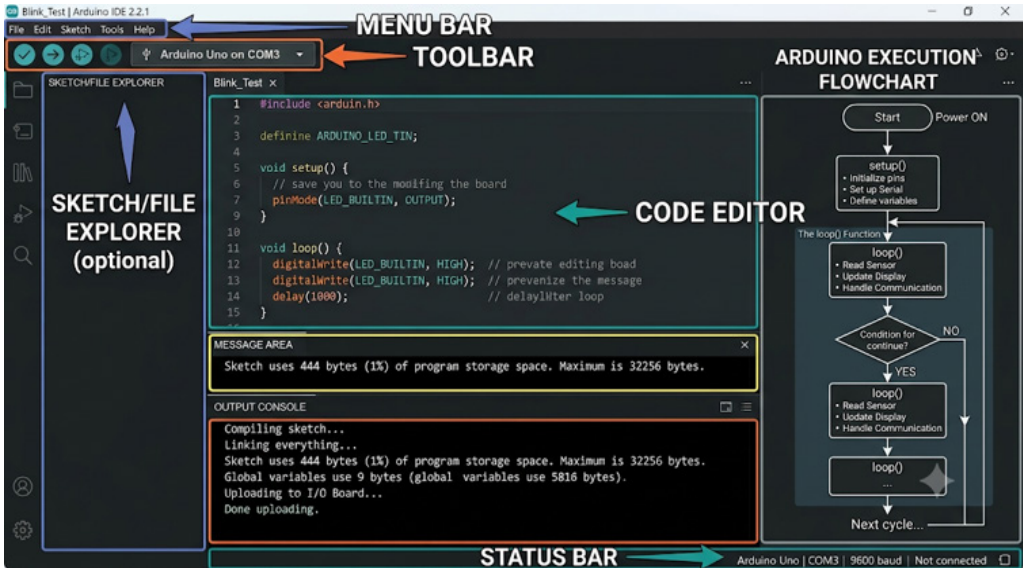


Fig.6.3.8 Execution sequence of an Arduino program.

6.3.14 Writing the First Arduino Program

One of the simplest Arduino programs is the Blink Program, which repeatedly turns an LED ON and OFF.

```
void setup()
{
   pinMode(13, OUTPUT);
}
void loop()
{
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
}
```

Program Explanation

pinMode()

Configures a pin as either:

- INPUT
- OUTPUT

SGOU

digitalWrite()

Sets the output state.

- HIGH
- LOW

delay()

Pauses program execution.

delay(1000);

means wait for 1000 milliseconds (1 second).

Expected Output

The built-in LED connected to pin 13 blinks continuously with a one-second interval.

Solved Illustration

Problem

Modify the Blink program so that the LED remains ON for 2 seconds and OFF for 3 seconds.

Solution

```
void setup()
{
  pinMode(13, OUTPUT);
}
void loop()
{
  digitalWrite(13, HIGH);
  delay(2000);
  digitalWrite(13, LOW);
  delay(3000);
}
```

6.3.15 Variables and Data Types

Variables store data during program execution.

SGOU

Table 6.3.5 Common Data Types

Data Type	Description	Example
int	Integer	25
float	Decimal number	25.6
char	Character	'A'
boolean	True/False	true
String	Text	"Arduino"

Example

```
int temperature = 30;
float voltage = 4.85;
boolean motor = true;
```

6.3.16 Operators

Arduino supports several categories of operators.

Arithmetic Operators

Operator	Purpose
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Comparison Operators

Operator	Purpose
==	Equal
!=	Not Equal
>	Greater Than
<	Less Than

SGOU

>=	Greater Than or Equal
<=	Less Than or Equal

Logical Operators

Operator	Purpose
&&	AND
||	OR
!	NOT

6.3.17 Digital Input Programming

Arduino can receive input from switches and digital sensors.

Example:

```
int button = digitalRead(2);
```

If the button is pressed,

```
button == HIGH
```

Otherwise,

```
button == LOW
```

Applications

- Push buttons
- IR sensors
- Motion sensors
- Door sensors

6.3.18 Digital Output Programming

Digital outputs control external devices.

Example:

digitalWrite(8, HIGH);

Applications:

- LED

SGOU

- Relay
- Buzzer
- Motor Driver

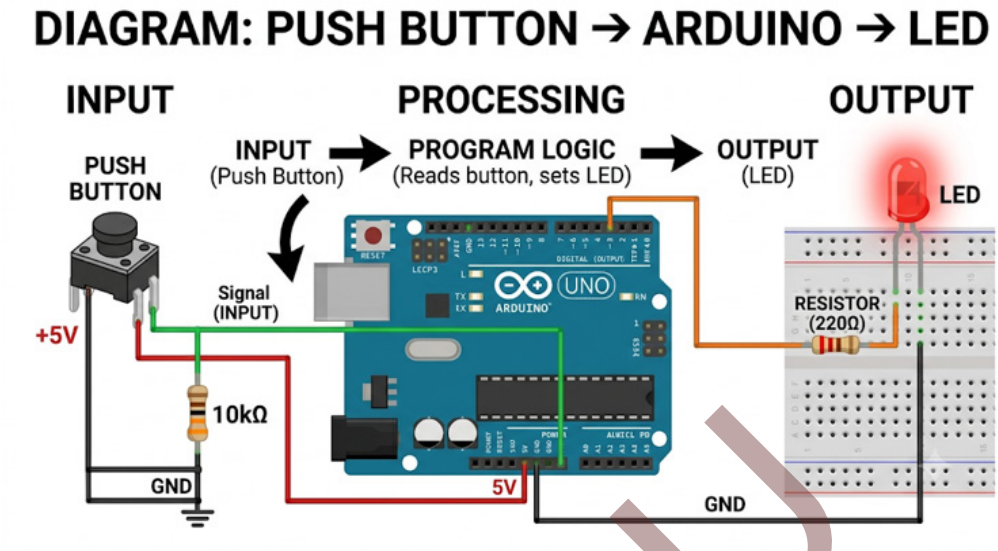


Fig.6.3.9 Reading digital input and controlling digital output.

6.3.19 Serial Communication

The Arduino communicates with computers using Serial Communication.

Initialization:

```
Serial.begin(9600);
```

Display message:

```
Serial.println("Hello World");
```

Read incoming data:

```
Serial.read();
```

Applications

- Sensor monitoring
- Debugging
- Communication with PC
- Bluetooth communication
- GSM modules

Why 9600 Baud?

The value 9600 represents the communication speed in bits per second (bps). Although higher baud rates such as 115200 bps are available, 9600 bps is commonly

SGOU

used because it offers reliable communication for most educational and prototype projects.

6.3.20 Compiling and Uploading a Sketch

The typical workflow in Arduino development is:

1. Write the program.
2. Save the sketch.
3. Verify (Compile).
4. Correct any errors.
5. Connect the Arduino board.
6. Select the Board and COM Port.
7. Upload the program.
8. Observe the output.

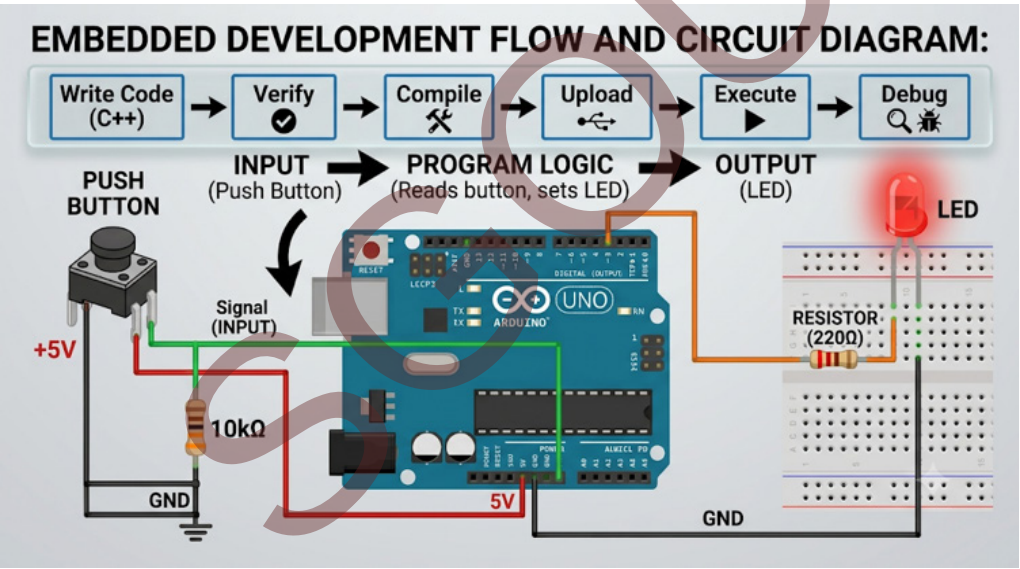


Fig.6.3.10 Typical Arduino software development process.

Case Study

Automatic Street Light System

A municipality wants to reduce electricity consumption by automating street lights.

Components Used

- Arduino Uno
- LDR (Light Dependent Resistor)

- Relay Module
- LED (Street Light Model)

Working

- During daytime, the LDR detects high light intensity, and the Arduino turns the street light OFF.
- At night, light intensity decreases. The Arduino reads the sensor value and activates the relay, turning the street light ON automatically.

Benefits

- Energy conservation
- Reduced operational cost
- Automatic operation
- Minimal human intervention

6.3.21 Integration of Sensors with Arduino

The true power of Arduino lies in its ability to interact with the physical world through sensors. Sensors collect information from the surrounding environment and convert physical quantities into electrical signals that the Arduino can process.

By integrating different sensors, Arduino-based systems can monitor environmental conditions, detect motion, measure distance, sense light intensity, and much more.

What is a Sensor?

A sensor is an electronic device that detects physical parameters such as temperature, light, humidity, pressure, motion, or distance and converts them into electrical signals for processing by a microcontroller.

General Sensor Integration Process

The following steps are typically followed when connecting a sensor to an Arduino:

1. Connect the sensor to the appropriate power supply (5V or 3.3V).
2. Connect the sensor output to the appropriate Arduino input pin.
3. Write the Arduino program to read the sensor value.
4. Process the received data.
5. Display or use the data to control an actuator.

SGOU

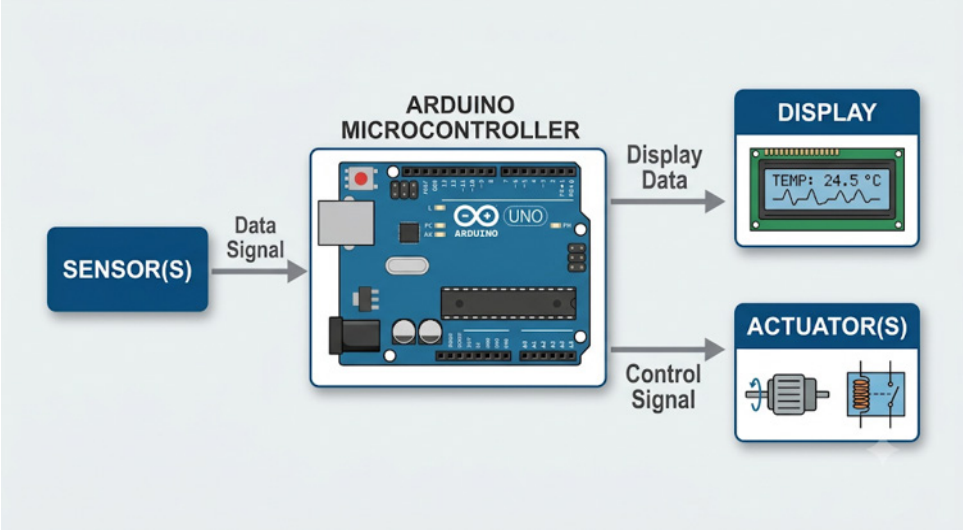


Fig.6.3.11 General process of integrating a sensor with an Arduino.

6.3.22 Common Sensors Used with Arduino

Temperature Sensor

Temperature sensors measure ambient or object temperature.

Applications include:

- Weather stations
- Smart agriculture
- Industrial monitoring
- Home automation
- Healthcare

Example

An LM35 temperature sensor produces an analog voltage proportional to temperature.

Arduino reads the analog voltage using an analog input pin and converts it into a temperature value.

Light Sensor (LDR)

A Light Dependent Resistor (LDR) changes its resistance according to the intensity of light.

Applications include:

- Automatic street lighting
- Smart classrooms

SGOU

- Greenhouse automation
- Solar tracking systems

Ultrasonic Sensor

Ultrasonic sensors measure distance by transmitting ultrasonic waves and measuring the time taken for the reflected signal to return.

Applications include:

- Obstacle detection
- Robot navigation
- Smart parking
- Water level measurement

Soil Moisture Sensor

This sensor measures the moisture content of soil.

Applications include:

- Smart irrigation
- Precision agriculture
- Plant monitoring

Gas Sensor

Gas sensors detect the presence of harmful gases.

Applications include:

- Industrial safety
- Fire detection
- Air quality monitoring
- LPG leakage detection

Table 6.3.6 Common Arduino Sensors

Sensor	Measures	Typical Applications
LM35	Temperature	Weather station
LDR	Light intensity	Street lights
HC-SR04	Distance	Robotics

SGOU

Soil Moisture	Soil water content	Smart farming
MQ Gas Sensor	Gas concentration	Safety systems

Sensor Calibration

Two identical sensors may produce slightly different readings due to manufacturing variations. Therefore, many industrial systems perform sensor calibration to improve measurement accuracy before deployment.

6.3.23 Integration of Actuators with Arduino

While sensors collect information, actuators perform physical actions based on commands generated by the Arduino.

Definition

An actuator is a device that converts electrical signals into physical movement or action.

General Actuator Control Process

Sensor → Arduino → Decision → Actuator

Common Actuators

- LED
- Buzzer
- Relay
- Servo Motor
- DC Motor
- Stepper Motor
- Solenoid Valve

LED

LEDs are the simplest actuators.

Applications :

- Status indication
- Signal lamps
- Warning systems

SGOU

Buzzer

Buzzers generate audible alerts.

Applications:

- Security alarms
- Fire alarms
- Timer systems

Relay

Relays allow Arduino to control high-voltage electrical appliances safely.

Applications:

- Home automation
- Industrial control
- Smart irrigation
- Street lighting

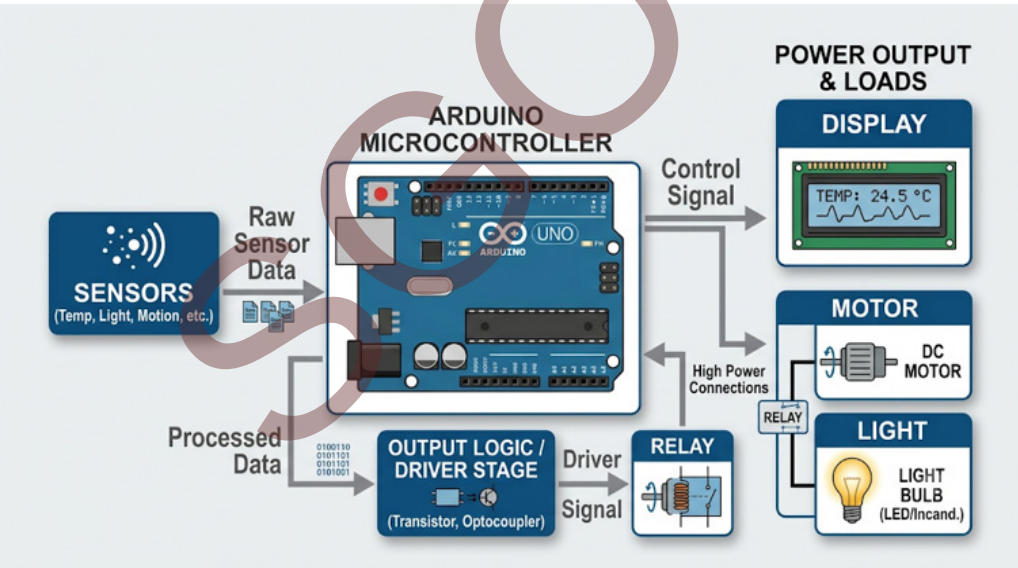


Fig.6.3.12 Arduino processes sensor data and controls actuators.

Servo Motor

Servo motors rotate to precise angular positions.

Applications:

- Robotic arms
- Camera positioning

- Smart door locks
- Industrial automation

6.3.24 Example IoT Applications

Automatic Street Light

Components

- Arduino Uno
- LDR
- Relay
- LED

Working

The Arduino continuously reads the LDR value.

- Bright daylight → LED OFF
- Darkness → LED ON

Smart Irrigation System

Components

- Soil Moisture Sensor
- Arduino Uno
- Relay
- Water Pump

Working

- Soil dry → Pump ON
- Soil wet → Pump OFF

Advantages:

- Water conservation
- Reduced labour
- Improved crop productivity

SGOU

Home Security Alarm

Components

- PIR Motion Sensor
- Arduino
- Buzzer
- LED

Working

Motion detected:

- Alarm activated
- LED flashes

Industrial Temperature Monitoring

Components

- LM35
- Arduino
- LCD
- Cooling Fan

Working

When temperature exceeds the preset limit, Arduino automatically activates the cooling fan.

Industry Insight

Many commercial IoT products begin as Arduino prototypes. Once the functionality is verified, manufacturers often redesign the hardware using dedicated microcontrollers for improved performance, lower power consumption, and reduced production cost.

6.3.25 Best Practices for Arduino Projects

To improve reliability and maintainability, the following best practices are recommended:

- Use descriptive variable names.
- Write modular code using functions.
- Add comments to explain logic.
- Avoid unnecessary delays.

SGOU

- Use appropriate power supplies.
- Verify sensor connections before uploading programs.
- Protect circuits against reverse polarity.
- Use external power for high-current actuators.
- Test individual modules before integrating the complete system.

Case Study

Smart Greenhouse Monitoring System

A horticulture research institute developed a smart greenhouse using Arduino.

Sensors Used

- Temperature sensor
- Humidity sensor
- Soil moisture sensor
- Light sensor

Actuators Used

- Water pump
- Ventilation fan
- LED grow lights

Working

The Arduino continuously collects environmental data.

When:

- Soil moisture becomes low → Water pump starts.
- Temperature rises → Cooling fan operates.
- Light intensity decreases → LED lights switch ON.

Benefits

- Higher crop yield
- Reduced water consumption
- Automated environmental control
- Improved energy efficiency

SGOU

Solved Illustration

Problem

A student wants to build an automatic water tank controller.

The pump should start when the water level becomes low and stop automatically when the tank becomes full.

Identify suitable sensors and actuators.

Solution

Sensor:

- Ultrasonic sensor or Water Level Sensor

Actuator:

- Relay
- Water Pump

Working:

1. Arduino continuously measures the water level.
2. When the water level falls below the lower threshold, the relay switches ON the pump.
3. When the tank becomes full, Arduino switches OFF the relay.

This automation conserves water and prevents overflow.

Recap

- Arduino is an open-source microcontroller platform widely used for embedded systems and IoT development.
- The Arduino Uno, based on the ATmega328P microcontroller, is the most popular development board for beginners and prototyping.
- The Arduino IDE is used to write, compile, upload, and monitor Arduino programs.
- Every Arduino program contains two essential functions: setup() and loop().
- Arduino supports digital and analog input/output operations, enabling communication with a variety of sensors and actuators.
- Common communication interfaces include UART, SPI, and I^2C.

SGOU

- Sensors such as LM35, LDR, HC-SR04, soil moisture sensors, and gas sensors enable Arduino to monitor environmental conditions.
- Actuators including LEDs, buzzers, relays, and servo motors allow Arduino to perform physical actions based on sensor inputs.
- Arduino is widely used in IoT applications such as smart homes, smart agriculture, industrial automation, environmental monitoring, and robotics.
- Following good programming and hardware practices improves the reliability, maintainability, and scalability of Arduino projects.

Objective Type Questions

1. What is Arduino?
2. Which microcontroller is used in the Arduino Uno?
3. What is the purpose of the Arduino IDE?
4. Differentiate between digital and analog pins.
5. Name the two mandatory Arduino functions.
6. What is the function of pinMode()?
7. Which communication protocol uses SDA and SCL lines?
8. Name three sensors commonly interfaced with Arduino.
9. What is the purpose of a relay module?
10. Which sensor is used for distance measurement?
11. What is a servo motor?
12. Mention two applications of Arduino in IoT.
13. Why is sensor calibration necessary?
14. State two best practices for Arduino programming.
15. Why is Arduino considered suitable for rapid prototyping?

SGOU

Answers

1. An open-source electronics platform based on programmable microcontrollers.

2. ATmega328P.
3. To write, compile, upload, and debug Arduino programs.
4. Digital pins handle HIGH/LOW signals; analog pins read continuously varying voltages.
5. setup() and loop().
6. It configures a pin as INPUT or OUTPUT.
7. I²C.
8. LM35, LDR, HC-SR04 (or Soil Moisture Sensor, MQ Gas Sensor).
9. To safely control high-voltage devices.
10. HC-SR04 ultrasonic sensor.
11. A motor that rotates to precise angular positions.
12. Smart irrigation and home automation (or robotics, environmental monitoring).
13. To improve measurement accuracy.
14. Use descriptive variable names and modular code (or add comments, test modules individually).
15. Because it is low-cost, easy to program, and supports a wide range of hardware modules.

Assignments

1. Explain the architecture and features of the Arduino Uno board with a neat diagram.
2. Discuss the role of the Arduino IDE in embedded system development.
3. Explain the structure of an Arduino program and illustrate the functions of setup() and loop().
4. Describe the integration of sensors with Arduino using suitable examples.
5. Explain how actuators are controlled using Arduino and discuss their applications.
6. Design an Arduino-based smart irrigation system and explain its working.
7. Discuss the role of Arduino in Internet of Things (IoT) applications.
8. Explain the communication interfaces supported by the Arduino Uno and compare UART, SPI, and I²C.

SGOU

Suggested Reading

1. Blum, Jeremy. Exploring Arduino. Wiley, 2019.
2. Evans, Brian. Beginning Arduino Programming. Apress, 2020.
3. Ibrahim, Dogan. Arduino Projects Handbook. Elektor, 2021.
4. Monk, Simon. 30 Arduino Projects for the Evil Genius. McGraw-Hill Education, 2020.
5. Banzi, Massimo. Making Things Talk. Maker Media, 2021.

Reference

1. Banzi, Massimo, and Michael Shiloh. Getting Started with Arduino. 4th ed., Maker Media, 2022.
2. Margolis, Michael. Arduino Cookbook. 3rd ed., O'Reilly Media, 2020.
3. Monk, Simon. Programming Arduino: Getting Started with Sketches. 3rd ed., McGraw-Hill Education, 2021.
4. Boxall, John. Arduino Workshop. No Starch Press, 2019.
5. Arduino. Arduino Documentation. Arduino, https://docs.arduino.cc/.

SGOU



UNIT 4

Implementation of IoT using Raspberry Pi

Learning Outcomes

After completion of this unit, learner will be able to:

- explain the basic concepts and features of Raspberry Pi.
- identify the major hardware components and interfaces of Raspberry Pi.
- demonstrate basic Raspberry Pi programming and GPIO interfacing using Python.
- describe communication protocols and data transmission methods used in Raspberry Pi based IoT systems.
- discuss the applications of Raspberry Pi in IoT, Edge Computing, healthcare, agriculture, and industrial automation.

SGOU

Prerequisite

Suppose you want to build a smart home surveillance system that not only detects motion but also captures images, stores them locally, uploads them to cloud storage, and sends an email notification to the homeowner.

Can a simple microcontroller perform all these tasks efficiently?

While microcontrollers like Arduino excel at reading sensors and controlling devices, applications requiring multitasking, networking, multimedia processing, and operating systems are better handled by Single Board Computers (SBCs).

One of the most widely used SBCs is the Raspberry Pi. It combines the capabilities of a desktop computer with the flexibility of embedded systems, making it an ideal platform for IoT, robotics, artificial intelligence, edge computing, and automation. Unlike traditional computers, Raspberry Pi is inexpensive, compact, energy-efficient, and capable of running a full-fledged operating system. In this unit, we will explore the Raspberry Pi platform, understand its architecture, and learn how it supports modern IoT applications.

Keywords

Raspberry Pi, Single Board Computer (SBC), GPIO, Linux, Raspberry Pi OS, ARM Processor, Edge Computing

Discussion

6.4.1 Introduction to Raspberry Pi

The Raspberry Pi is a compact, affordable, and powerful single-board computer designed to promote computer science education and embedded system development. Despite its small size, Raspberry Pi functions like a complete computer. It supports an operating system, storage devices, networking, multimedia applications, and programming environments. Its ability to interface with sensors, cameras, displays, and communication modules makes it one of the most popular IoT development platforms.

Raspberry Pi is a low-cost, credit-card-sized Single Board Computer (SBC) developed by the Raspberry Pi Foundation that runs a Linux-based operating system and supports programming, networking, multimedia processing, and Internet of Things (IoT) applications.

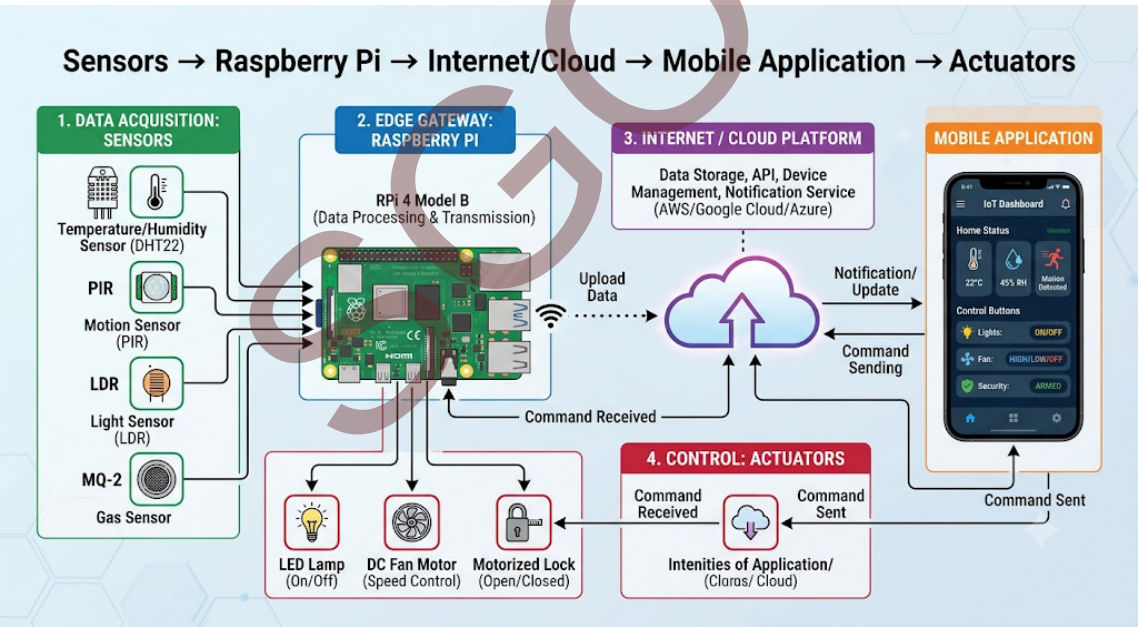


Fig. 6.4.1 Raspberry Pi functioning as the central controller in an IoT system.

Why Raspberry Pi is Popular

The Raspberry Pi has gained worldwide popularity because it provides:

- Affordable computing
- Linux operating system support

SGOU

- High processing capability
- Networking features
- USB connectivity
- HDMI display output
- Camera interface
- GPIO pins for hardware interfacing
- Excellent community support

Raspberry Pi is widely used in:

- Internet of Things (IoT)
- Robotics
- Artificial Intelligence
- Smart Home Automation
- Industrial Automation
- Surveillance Systems
- Smart Agriculture
- Edge Computing
- Educational Laboratories
- Digital Signage

Many organizations use Raspberry Pi as an edge computing device, processing sensor data locally before transmitting essential information to cloud servers. This reduces network traffic, lowers latency, and improves the responsiveness of IoT systems.

6.4.1.1 History and Evolution of Raspberry Pi

The Raspberry Pi was developed by the Raspberry Pi Foundation, a UK-based charitable organization established to promote computer science education. The first Raspberry Pi was released in 2012 with the goal of making computing accessible and affordable for students.

Over the years, the platform has evolved significantly, offering faster processors, increased memory, wireless communication, and improved graphics capabilities. Today, Raspberry Pi powers millions of educational, research, industrial, and commercial projects worldwide.

SGOU

Table 6.4.1 Major Milestones

Year	Development
2012	Raspberry Pi Model B launched
2014	Raspberry Pi Model B+ introduced
2016	Raspberry Pi 3 with built-in Wi-Fi and Bluetooth
2019	Raspberry Pi 4 with USB 3.0 and up to 8 GB RAM
2023	Raspberry Pi 5 launched with enhanced CPU and graphics performance

The Raspberry Pi Foundation originally intended to sell only a few thousand boards for educational purposes. Today, tens of millions of Raspberry Pi boards have been sold worldwide, making it one of the most successful single-board computer platforms ever developed.

Features of Raspberry Pi

The Raspberry Pi provides a wide range of features that make it suitable for both education and industrial applications.

1. **Single Board Computer:** Unlike a microcontroller, Raspberry Pi functions as a complete computer capable of running a full operating system.
2. **Compact Size:** Its small size enables integration into embedded systems, robots, drones, and portable devices.
3. **Low Power Consumption:** Raspberry Pi consumes much less power than conventional desktop computers, making it suitable for continuous operation.
4. **Powerful Processor:** Modern Raspberry Pi models use multi-core ARM processors capable of multitasking and multimedia processing.
5. **Networking Support:** Most recent Raspberry Pi models include:

   Wi-Fi

   Bluetooth

   Ethernet

   These features simplify IoT application development.

6. **Multimedia Support**

   Supports:

   HD video

   HDMI displays

SGOU

Audio output

Camera modules

7. **Programming Support**

Raspberry Pi supports multiple programming languages, including:

Python

C

C++

Java

Scratch

Python is the most widely used language for Raspberry Pi-based IoT applications due to its simplicity and extensive library support.

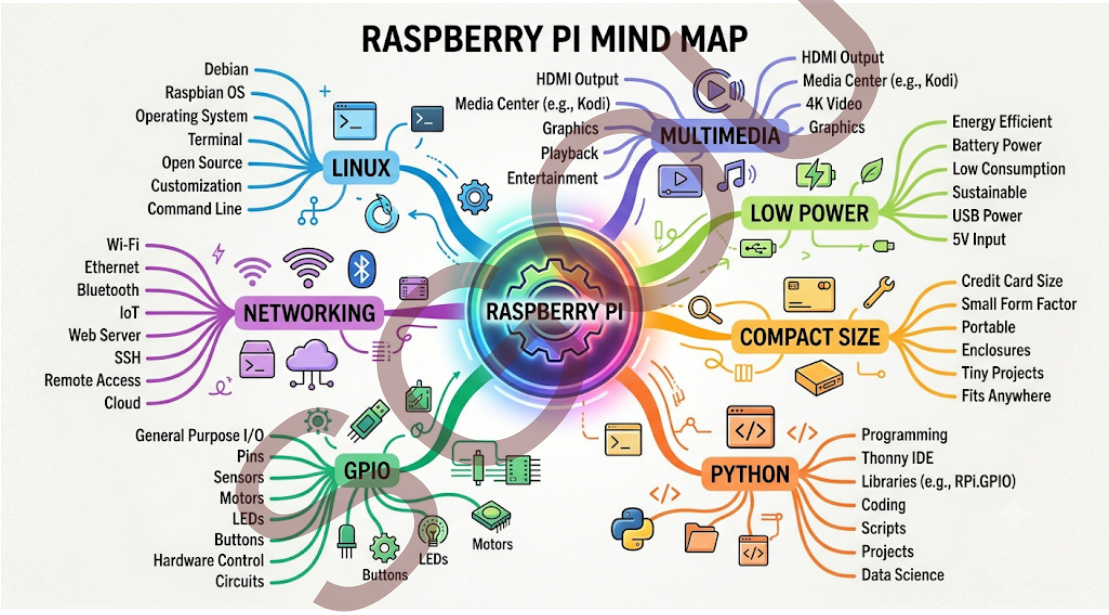


Fig. 6.4.2 Major features of the Raspberry Pi platform.

6.4.2 Setup of Raspberry Pi

Setting up the Raspberry Pi is the first step in implementing an IoT application. The setup includes preparing the hardware, installing the operating system, configuring the Raspberry Pi, connecting it to a network, and installing the software required for the IoT application.

6.4.2.1 Hardware Components Required

The Raspberry Pi integrates several hardware components onto a single circuit board.

Major components include:

- ARM Processor
- RAM
- GPIO Header
- USB Ports
- HDMI Port
- Ethernet Port
- Wi-Fi Module
- Bluetooth Module
- MicroSD Card Slot
- CSI Camera Connector
- DSI Display Connector
- Power Input

Each component performs a specific function, enabling Raspberry Pi to act as a complete computing platform.

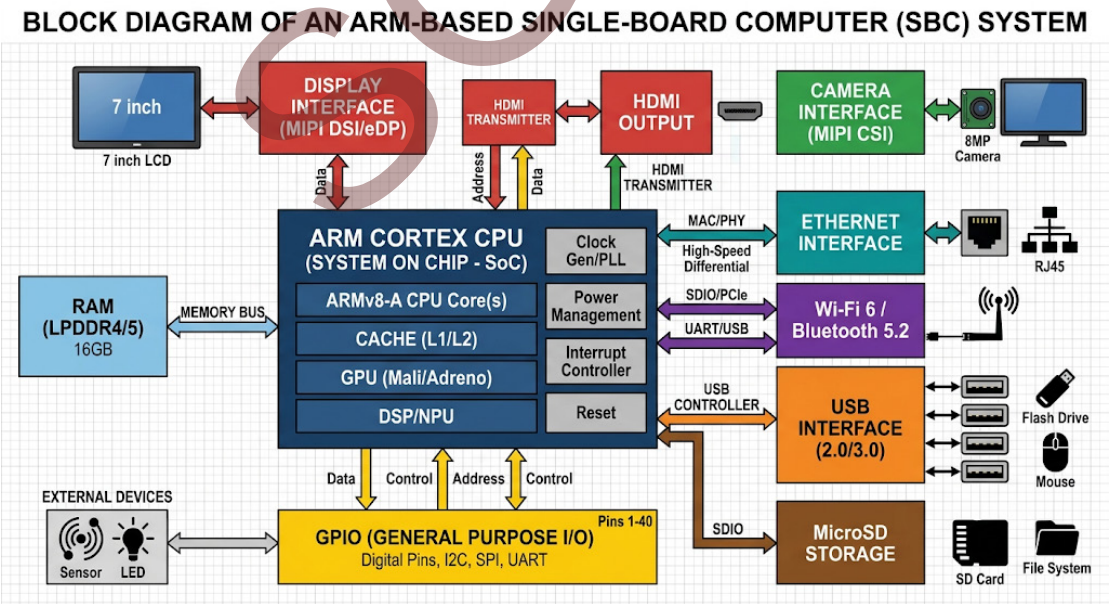


Fig. 6.4.3 General hardware architecture of a Raspberry Pi board.

SGOU

Major Components of Raspberry Pi

1. ARM Processor

The ARM processor is the central processing unit (CPU) of the Raspberry Pi.

Functions include:

- Executing programs
- Managing memory
- Processing sensor data
- Running the operating system
- Performing mathematical operations

2. Random Access Memory (RAM)

RAM stores temporary data while programs are executing.

Higher RAM capacity allows more applications to run simultaneously.

3. MicroSD Card

Unlike desktop computers that use hard disks or SSDs, Raspberry Pi stores its operating system and user files on a MicroSD card.

The MicroSD card contains:

- Raspberry Pi OS
- User programs
- Configuration files
- Application software

4. GPIO Header

The General Purpose Input Output (GPIO) header enables Raspberry Pi to communicate with external electronic devices.

Applications include interfacing:

- LEDs
- Relays
- Temperature sensors
- Ultrasonic sensors
- Servo motors
- LCD displays

SGOU

5. USB Ports

USB ports connect external devices such as:

- Keyboard
- Mouse
- Flash drive
- Camera
- Wi-Fi adapter (older models)

6. HDMI Port

Used for connecting:

- Monitor
- Television
- Projector

7. Ethernet Port

Provides wired Internet connectivity.

Useful for:

- Industrial automation
- Network servers
- IoT gateways

Solved Illustration

A student wants to build a Raspberry Pi–based home automation system that requires internet connectivity, sensor interfacing, and a graphical display. Identify the Raspberry Pi components required for these functions.

- Ethernet Port or Wi-Fi Module: Internet connectivity.
- GPIO Header: Sensor interfacing.
- HDMI Port: Connecting a monitor or display.

Together, these components enable networking, hardware interaction, and user visualization for the home automation system.

6.4.2.2 Operating System Installation

The Raspberry Pi requires an operating system to execute programs and manage hardware resources. Raspberry Pi OS, based on Linux, is commonly used.

SGOU

The general installation process involves:

1. Preparing a microSD card.
2. Installing the Raspberry Pi operating system on the card.
3. Inserting the card into the Raspberry Pi.
4. Connecting the power supply.
5. Starting the Raspberry Pi.
6. Completing the initial configuration.
7. Connecting to the network.
8. Updating the operating system and required packages.

After installation, the Raspberry Pi provides a working environment in which Python programs and other IoT software can be executed.

Raspberry Pi Operating System (Raspberry Pi OS)

The Raspberry Pi Operating System, formerly known as Raspbian, is a Linux-based operating system specially designed for Raspberry Pi hardware.

It provides a complete computing environment with:

- Desktop interface
- Command-line interface
- Programming tools
- Internet access
- Multimedia applications
- File management
- Device drivers

Features of Raspberry Pi OS

- Free and open source
- Linux-based
- Secure environment
- Multi-user support
- Multitasking
- Python pre-installed
- Supports programming and IoT development

SGOU

Linux Desktop Environment

The Raspberry Pi desktop includes:

- Desktop icons
- Start menu
- Taskbar
- Terminal
- File Manager
- Web Browser
- Programming applications

Terminal

The Linux Terminal enables users to execute text-based commands for:

- File management
- Program execution
- Software installation
- Network configuration
- System administration

Why Does Raspberry Pi Use Linux?

Linux is highly stable, secure, open source, and customizable. It supports networking, multitasking, programming, and a vast ecosystem of software, making it an ideal operating system for IoT and embedded computing.

Installing Raspberry Pi OS

The installation process typically involves:

1. Download Raspberry Pi OS.
2. Insert a MicroSD card into a computer.
3. Use the Raspberry Pi Imager to write the operating system to the card.
4. Insert the MicroSD card into the Raspberry Pi.
5. Connect the keyboard, mouse, monitor, and power supply.
6. Complete the initial configuration wizard.
7. Update the operating system.

SGOU

Solved Illustration

A student powers on a Raspberry Pi, but the display remains blank. Identify possible reasons.

Possible causes include:

- The MicroSD card does not contain Raspberry Pi OS.
- The HDMI cable is not connected properly.
- The power supply is inadequate.
- The operating system image is corrupted.
- The monitor is set to the wrong input source.

Checking each of these systematically usually resolves the issue.

6.4.2.3 Software Setup

Python is commonly used for Raspberry Pi IoT applications because it is relatively easy to learn and provides libraries for hardware and network programming.

Software may be required for:

- GPIO control
- Sensor communication
- Data processing
- Network communication
- Database connectivity
- Cloud communication
- MQTT communication
- HTTP requests

For example, a Python program can be used to read a sensor value and send the value to a remote server. One of the major strengths of Raspberry Pi is its excellent support for Python, a high-level programming language known for its simplicity, readability, and extensive libraries. Python is the preferred language for Raspberry Pi because it enables developers to create IoT applications quickly without dealing with complex programming syntax. Most Raspberry Pi operating system installations include Python pre-installed, allowing users to begin programming immediately.

Python IDEs on Raspberry Pi

Several Integrated Development Environments (IDEs) are available for writing Python programs.

Common options include:

SGOU

- Thonny Python IDE
- Geany
- Visual Studio Code
- IDLE

Among these, Thonny is the default IDE in Raspberry Pi OS and is recommended for beginners.

Features of Thonny IDE

- User-friendly interface
- Built-in debugger
- Syntax highlighting
- Error messages
- Automatic code completion
- Easy program execution

Although large-scale software projects often use Visual Studio Code or PyCharm, many educational institutions and IoT laboratories prefer Thonny because of its simplicity and seamless integration with Raspberry Pi OS.

6.4.3 Connectivity

Connectivity is one of the most important characteristics of an IoT system. The main purpose of IoT is to allow physical devices to communicate with other devices, servers, cloud platforms, and applications.

The Raspberry Pi provides several methods for establishing connectivity.

6.4.3.1 Wi-Fi Connectivity

Wi-Fi allows the Raspberry Pi to connect to a wireless network without requiring a physical network cable.

This is particularly useful for IoT applications installed in locations where Ethernet cables are inconvenient.

For example, a Raspberry Pi installed in a greenhouse can connect to the local Wi-Fi network and transmit temperature and humidity readings to a cloud platform.

The basic communication path is:

SGOU

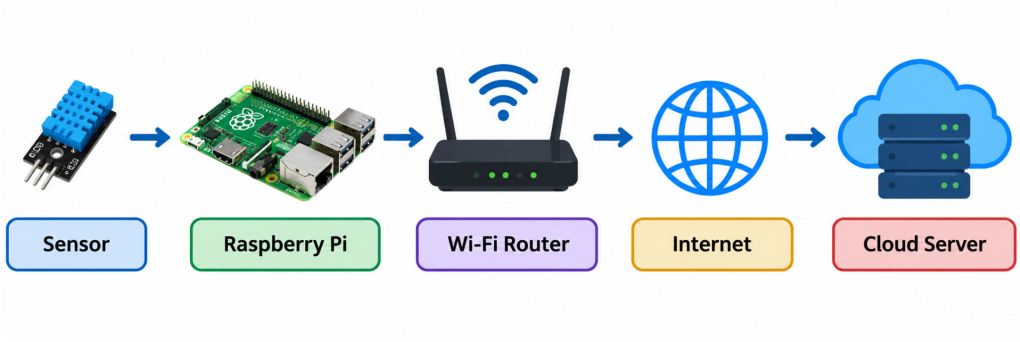


Fig. 6.4.4 Communication path

Once connected to Wi-Fi, the Raspberry Pi can communicate with remote systems using protocols such as HTTP and MQTT.

6.4.3.2 Ethernet Connectivity

Ethernet provides a wired network connection.

It is useful when a stable and reliable network connection is required. For example, Raspberry Pi can be connected through Ethernet in an industrial monitoring system where reliable communication is important.

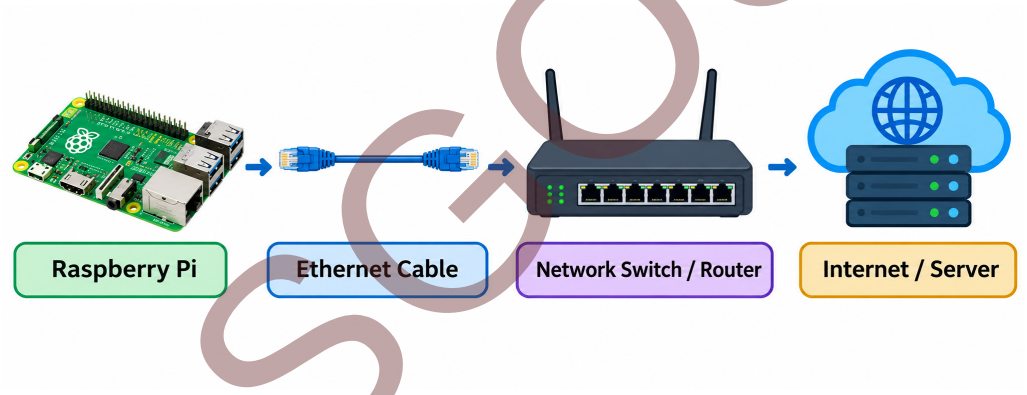


Fig. 6.4.5 Raspberry Pi Ethernet-Based Network Communication

Compared with wireless communication, Ethernet does not depend on wireless signal strength in the same way, although it requires physical cabling.

6.4.3.3 Bluetooth Connectivity

Bluetooth can be used for short range communication between the Raspberry Pi and compatible devices.

For example, the Raspberry Pi can communicate with a Bluetooth enabled sensor or another nearby device.

Bluetooth is useful when:

- The devices are located close to each other.

SGOU

- Low power communication is required.
- A direct short range connection is sufficient.

6.4.3.4 IP Address

When the Raspberry Pi connects to a network, it normally receives an IP address. The IP address identifies the Raspberry Pi on the network.

Other devices on the same network can use this address to communicate with the Raspberry Pi, depending on the services running on it.

Understanding IP addresses is important when implementing network based IoT applications.

6.4.4 Sensor Integration

Sensor integration is the process of connecting sensors to a Raspberry Pi so that it can collect information from the physical environment. Sensors detect physical parameters such as temperature, humidity, light, motion, distance, gas concentration, and soil moisture and provide the corresponding data to the Raspberry Pi.

The Raspberry Pi can connect sensors through GPIO pins or communication interfaces such as I2C, SPI, and UART. Digital sensors can provide data directly through suitable interfaces, while analog sensors may require an Analog to Digital Converter (ADC) because Raspberry Pi does not have general purpose built-in analog input.

The basic sensor integration process is:

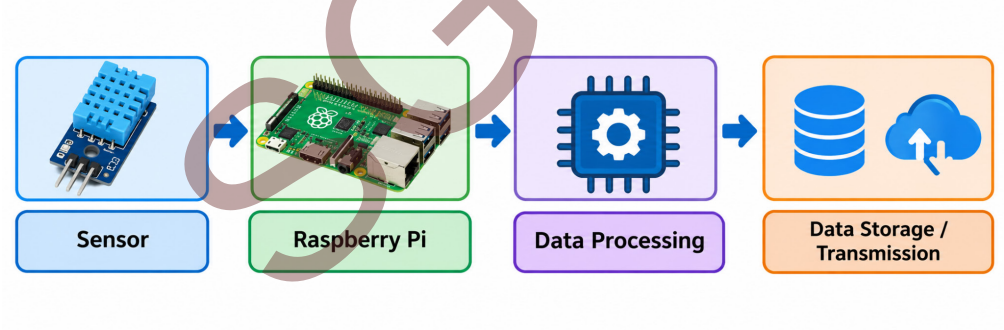


Fig. 6.4.6 Basic sensor integration process.

For example, a temperature sensor connected to Raspberry Pi can measure room temperature. The Raspberry Pi reads the sensor value, processes it, and can display it locally or transmit it through Wi-Fi to a cloud server for remote monitoring.

Sensor integration is therefore an important part of IoT because it enables the Raspberry Pi to sense the physical environment, collect data, process it, and communicate the information to other systems.

6.4.4.1 GPIO (General Purpose Input Output) Pins

One of the most important features of Raspberry Pi is its GPIO header, which enables interaction with the external physical world.

SGOU

GPIO pins allow Raspberry Pi to:

- Read sensor data
- Control actuators
- Communicate with electronic modules
- Interface displays
- Connect communication devices

GPIO Functions

GPIO pins support multiple functions.

Digital Input

Reads digital signals from devices such as:

- Push buttons
- PIR sensors
- Limit switches

Digital Output

Controls external devices.

Examples:

- LEDs
- Relays
- Buzzers

Communication Interfaces

GPIO also supports:

- UART
- SPI
- I^2C
- PWM

These interfaces enable communication with advanced electronic modules.

SGOU

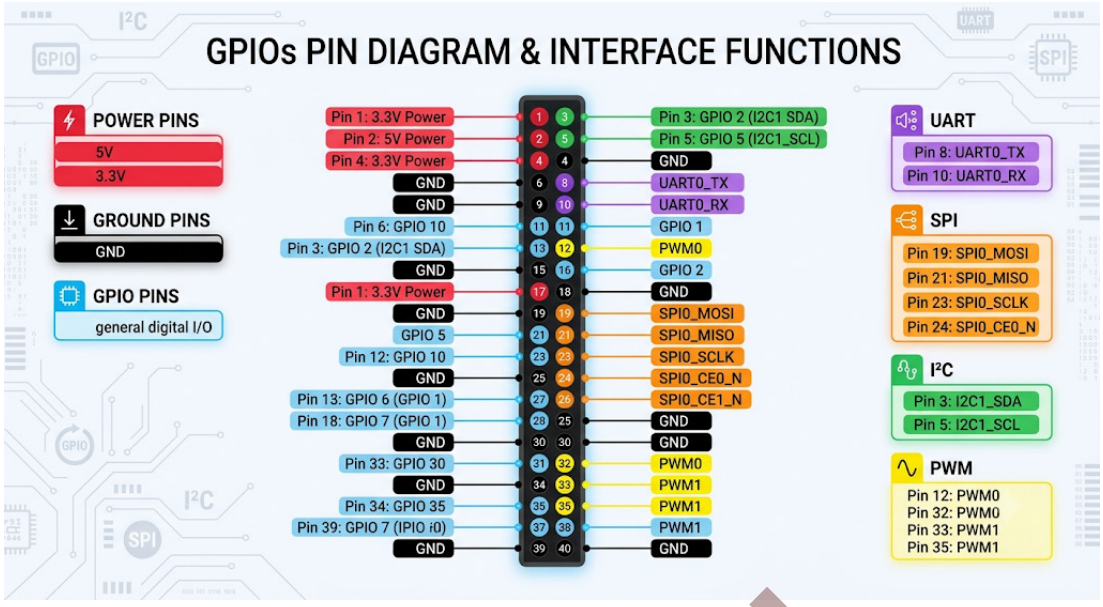


Fig 6.4.7 GPIO header of a Raspberry Pi board.

GPIO Safety

Unlike Arduino, Raspberry Pi GPIO pins operate at 3.3 V.

Applying 5 V directly to a GPIO pin may permanently damage the board.

Therefore:

- Use proper voltage levels.
- Use level shifters when necessary.
- Verify wiring before powering the system.

Many industrial automation projects use Raspberry Pi together with external interface circuits such as opto-isolators, relay modules, and voltage converters to protect the GPIO pins from electrical faults and high-voltage equipment.

6.4.5 Data Transmission

Data transmission is the process of sending the data collected by sensors from the Raspberry Pi to another device, server, database, or cloud platform through a communication network. In an IoT system, the Raspberry Pi processes the sensor data and transmits it using Wi-Fi, Ethernet, or other communication technologies.

Common communication protocols used for IoT data transmission include HTTP, HTTPS, and MQTT. MQTT is widely used because it is lightweight and suitable for IoT devices with limited resources.

SGOU

The basic process is:

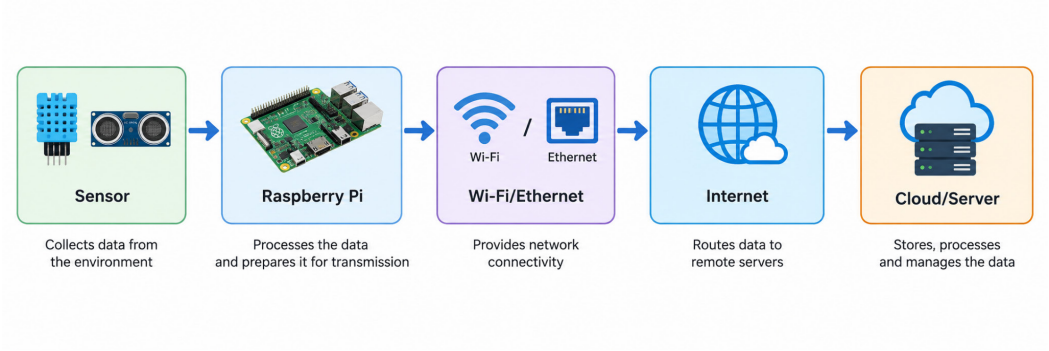


Fig 6.4.8 Data transmission process

For example, a temperature sensor connected to a Raspberry Pi may measure 30°C. The Raspberry Pi reads and processes this value and sends it to a cloud server using MQTT or HTTP. The data can then be stored in a database and displayed through a web or mobile application.

Thus, data transmission enables remote monitoring, data storage, analysis, and communication between IoT devices and applications.

6.4.6 Raspberry Pi Boot Process

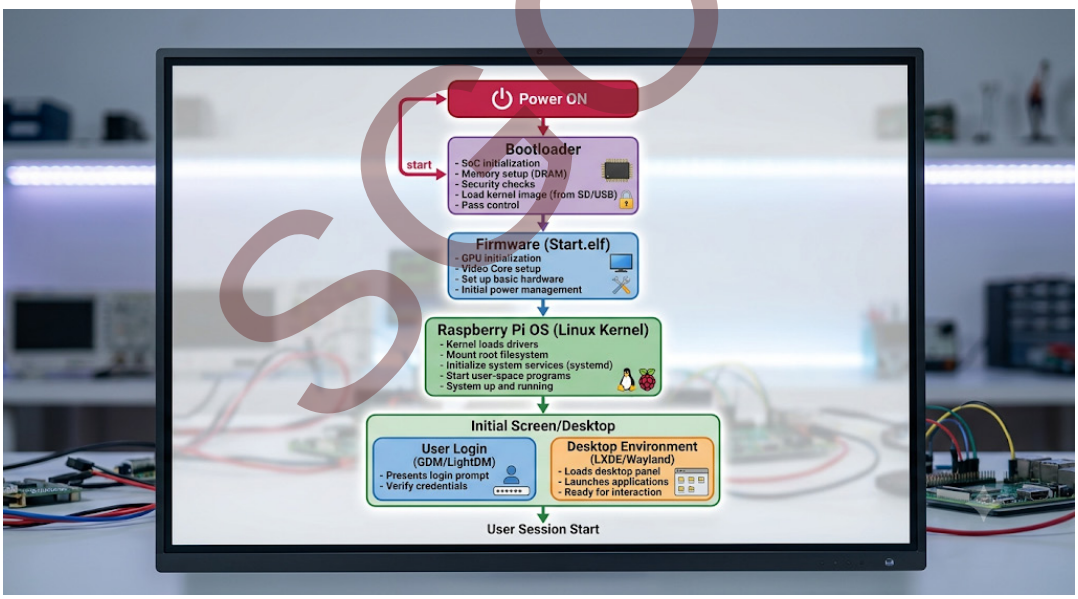


Fig. 6.4.9 Simplified Raspberry Pi boot process.

Unlike a microcontroller, Raspberry Pi follows a complete computer boot sequence.

The boot process includes:

1. Power ON
2. Bootloader initialization

SGOU

3. Loading firmware
4. Loading the Raspberry Pi Operating System from the MicroSD card
5. Starting Linux services
6. Displaying the desktop or command-line interface

6.4.7 Raspberry Pi-Based IoT Applications

The Raspberry Pi has become one of the most widely adopted platforms for developing Internet of Things (IoT) systems because it combines computing power, networking capabilities, and hardware interfacing on a single board. It can collect data from sensors, process the information locally, communicate with cloud platforms, and control actuators in real time.

Some common Raspberry Pi-based IoT applications are discussed below.

1. Smart Home Automation

A Raspberry Pi can act as the central controller of a smart home by connecting sensors, relays, lights, fans, cameras, and smart appliances.

Functions

- Automatic lighting
- Remote appliance control
- Security monitoring
- Smart door locking
- Energy management

Benefits

- Convenience
- Energy efficiency
- Enhanced security
- Remote access through mobile applications

2. Smart Agriculture

Agriculture increasingly relies on IoT technologies to improve productivity and conserve resources.

SGOU

Components

- Soil Moisture Sensor
- Temperature Sensor
- Humidity Sensor
- Raspberry Pi
- Relay Module
- Water Pump

Working

The Raspberry Pi continuously collects data from environmental sensors. Based on predefined conditions, it automatically controls irrigation systems, ventilation, or lighting.

Benefits

- Water conservation
- Increased crop yield
- Reduced labour
- Real-time monitoring

3. Industrial Monitoring

Industries use Raspberry Pi for monitoring machine conditions and environmental parameters.

Typical applications include:

- Temperature monitoring
- Vibration analysis
- Equipment health monitoring
- Predictive maintenance
- Production monitoring

4. Healthcare Monitoring

Raspberry Pi-based systems can monitor patients using various biomedical sensors.

Applications include:

SGOU

- Heart rate monitoring
- Body temperature monitoring
- Oxygen saturation monitoring
- Remote patient observation

5. Smart Surveillance

By integrating a camera module with Raspberry Pi, intelligent surveillance systems can be developed.

Features include:

- Motion detection
- Image capture
- Video recording
- Face recognition
- Remote monitoring

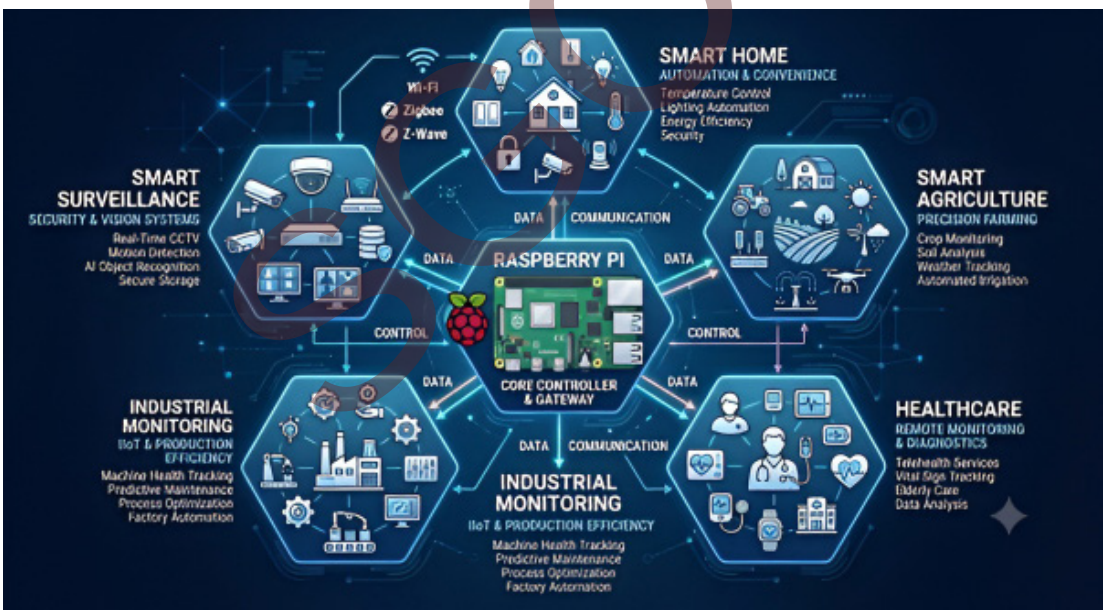


Fig 6.4.10 Diverse Internet of Things applications developed using Raspberry Pi.

6.4.8 Raspberry Pi vs Arduino

Although Raspberry Pi and Arduino are both widely used in IoT projects, they are fundamentally different platforms designed for different purposes.

Table 6.4.3 Comparison of Raspberry Pi and Arduino

Feature	Raspberry Pi	Arduino
Device Type	Single Board Computer	Microcontroller Board
Operating System	Raspberry Pi OS (Linux)	No Operating System
Processor	ARM Processor	ATmega328P (Uno)
Multitasking	Yes	No
Programming Languages	Python, C, C++, Java	Arduino C/C++
Storage	MicroSD Card	Flash Memory
Networking	Built-in Wi-Fi/Ethernet (model-dependent)	External modules generally required
Best Suited For	IoT, AI, Edge Computing, Multimedia	Embedded Control, Sensors, Actuators
Boot Time	Requires boot process	Starts immediately
Power Consumption	Higher	Lower

Choosing Between Raspberry Pi and Arduino

Choose Arduino when:

- Real-time control is required.
- Simple sensor-based projects are being developed.
- Low power consumption is important.
- Cost must be minimized.

Choose **Raspberry Pi** when:

- A full operating system is needed.
- Networking and Internet connectivity are essential.
- Image processing or multimedia applications are required.
- Artificial Intelligence or Machine Learning applications are involved.

Can Raspberry Pi and Arduino Work Together?

Yes. Many advanced IoT systems use Arduino to read sensor data and perform real-time control while Raspberry Pi handles networking, cloud communication, data storage, and user interfaces. This combination leverages the strengths of both platforms.

SGOU

6.4.9 Best Practices for Raspberry Pi Projects

Developing reliable Raspberry Pi applications requires adherence to good hardware and software practices.

Hardware Best Practices

- Use a high-quality power supply.
- Avoid applying incorrect voltages to GPIO pins.
- Ensure adequate ventilation for prolonged operation.
- Use protective enclosures in industrial environments.
- Disconnect power before modifying hardware connections.

Software Best Practices

- Keep the operating system updated.
- Use strong passwords for remote access.
- Regularly back up important data.
- Write modular and well-documented Python programs.
- Test individual components before full system integration.

SGOU

Recap

- Raspberry Pi is a low cost Single Board Computer (SBC) that can run a Linux based operating system.
- It supports Python programming and provides GPIO, networking, multimedia, and hardware interfacing capabilities.
- The GPIO pins allow Raspberry Pi to connect with sensors, actuators, LEDs, motors, and other external devices.
- Raspberry Pi OS provides a complete computing environment for developing and running IoT applications.
- Raspberry Pi can communicate with other devices using protocols such as UART, SPI, I²C, HTTP, and MQTT.
- The MicroSD card is used to store the operating system, applications, and user data.
- Raspberry Pi can work as an IoT gateway by collecting, processing, and forwarding data between IoT devices and cloud platforms.

- Raspberry Pi is widely used in smart homes, healthcare, agriculture, surveillance, robotics, and industrial automation.
- Raspberry Pi and Arduino can be used together, where Arduino can handle real time hardware control and Raspberry Pi can provide higher level computing and networking.
- Raspberry Pi is suitable for Edge Computing because it can process data locally before sending it to a remote server or cloud.
- Emerging technologies such as Edge AI, TinyML, and Industrial IoT are expanding the applications of Raspberry Pi.

Objective Type Questions

1. What is Raspberry Pi?
2. What does SBC stand for?
3. Which operating system is commonly used with Raspberry Pi?
4. Which programming language is commonly used with Raspberry Pi?
5. What is used to interface sensors with Raspberry Pi?
6. Which connector is used to connect a camera module?
7. Which storage device is commonly used in Raspberry Pi?
8. Name a lightweight IoT messaging protocol supported by Raspberry Pi.
9. Which protocol is commonly used for web communication?
10. Which technology enables local data processing?
11. What type of device is Arduino?
12. What acts as an interface between IoT devices and the cloud?
13. Which technology combines AI with edge devices?
14. Which technology enables machine learning on small devices?
15. What technology connects industrial devices and systems?

SGOU

Answers

1. SBC
2. Single Board Computer
3. Raspberry Pi OS
4. Python
5. GPIO
6. CSI
7. MicroSD
8. MQTT
9. HTTP
10. Edge Computing
11. Microcontroller
12. Gateway
13. Edge AI
14. TinyML
15. Industrial IoT

Assignments

1. Explain the architecture and major hardware components of Raspberry Pi with a neat diagram.
2. Describe Raspberry Pi OS and explain the basic boot process of Raspberry Pi.
3. Explain GPIO programming on Raspberry Pi using Python with a suitable example.
4. Compare Raspberry Pi and Arduino based on architecture, processing capability, programming, communication, and applications.
5. Discuss the role of Raspberry Pi in Internet of Things applications with suitable examples.
6. Explain the working of a Raspberry Pi based smart home automation system with a neat block diagram.

SGOU

7. Describe the applications of Raspberry Pi in healthcare, agriculture, surveillance, robotics, and industrial automation.
8. Discuss the role of Raspberry Pi in Edge Computing, Edge AI, TinyML, and Industrial IoT.
9. Explain how Raspberry Pi can be used as an IoT Gateway for collecting and transmitting sensor data.
10. Describe the process of sensor integration and data transmission using Raspberry Pi in an IoT system.

Suggested Reading

1. Halfacree, Gareth. *Raspberry Pi Cookbook*. O'Reilly Media, 2023.
2. Monk, Simon. *Raspberry Pi for Dummies*. Wiley, 2022.
3. Richardson, Matt. *Getting Started with Raspberry Pi Pico*. Maker Media, 2022.
4. Blum, Jeremy. *Exploring Raspberry Pi*. Wiley, 2021.
5. Raspberry Pi Foundation. *The Official Raspberry Pi Beginner's Guide*. Raspberry Pi Press, 2023.

Reference

1. Richardson, Matt, and Shawn Wallace. *Getting Started with Raspberry Pi*. 4th ed., Maker Media, 2021.
2. Monk, Simon. *Programming Raspberry Pi: Getting Started with Python*. McGraw-Hill Education, 2022.
3. Upton, Eben, and Gareth Halfacree. *Raspberry Pi User Guide*. 5th ed., Wiley, 2021.
4. Raspberry Pi Foundation. *Raspberry Pi Documentation*. Raspberry Pi Foundation, https://www.raspberrypi.com/documentation/.
5. Barrett, Steven F. *Raspberry Pi Projects for the Evil Genius*. McGraw-Hill Education, 2020.

SGOU

MODEL QUESTION PAPER SETS

SGOU

SREENARAYANAGURU OPEN UNIVERSITY

QP CODE: ……

Reg. No. : .....

**Name: ……..... **

BACHELOR OF COMPUTER APPLICATION, SIXTH SEMESTER

END SEMESTER EXAMINATION

Discipline Specific Elective Course

B21CA24DE - Internet of Things

2023-2024 - Admission Onwards (CBCS - UG)

MODEL QUESTION PAPER- SET A

Time: 3 Hour Max Marks: 70

SECTION A

Answer any **ten** *questions of the following. Each question carries* ***one*** *mark.*

(10 × 1 = 10 Marks)

1. Which layer of the IoT architecture delivers dashboards and alerts to end users?
2. What is the central system that unifies HVAC, lighting and security in a smart building called?
3. What term describes data processing carried out close to the sensor rather than in the cloud?
4. What is latency?
5. Which communication model uses a gateway between devices and the cloud?
6. Which communication technology is suitable for wireless remote control systems?
7. Name the two major versions of the Internet Protocol.
8. Name one transport-layer protocol used in IoT communication
9. Which application-layer protocol is commonly associated with constrained IoT devices and can operate over UDP?
10. Define throughput.
11. What is the purpose of sensors?

SGOU

12. What are alarms and events?
13. Which stage of the lifecycle checks whether sensor records are plausible and consistent?
14. What is hadoop?
15. Define a vulnerability in OT security

SECTION B

Answer any **five** *questions of the following. Each question carries* **two** *marks.*

(5×2 =10 Marks)

16. What are the challenges of IoT?
17. What are the key differences between IT systems and OT systems?
18. Define an actuator in IoT and give one suitable example.
19. List the four layers of a Wireless Sensor Network (WSN) and state the major function of each layer.
20. What is 6LoWPAN? State its purpose
21. What is RPL? Mention its purpose in IoT networks.
22. Compare TCP and UDP.
23. Explain Edge, Fog and Cloud
24. State any two functions of a Raspberry Pi in IoT implementation.
25. Explain the role of sensors in collecting data in a Raspberry Pi-based IoT system.

SECTION C

Answer any **five** *questions of the following. Each question carries* **four** *marks.*

(5×4= 20 Marks)

26. Explain the simplified IoT architecture with a neat diagram.
27. Compare Wi-Fi, RFID, Bluetooth, RF, and Satellite communication.
28. Define Quality of Service (QoS) and list any four important QoS parameters.
29. Discuss the challenges involved in adopting IP for resource-constrained IoT devices
30. Discuss how latency, bandwidth, and packet overhead affect IoT communication
31. Explain any four functions of SCADA.
32. Explain SCADA Architecture.
33. Explain the common types of streaming windows.

SGOU

34. Explain the FAIR framework and how it supports information security risk analysis.
35. Describe the process of implementing an IoT system using Raspberry Pi, covering setup, connectivity, sensor integration, and data transmission.

SECTION D

Answer any **two** *questions of the following. Each question carries* **fifteen** *marks.*

(2×15 = 30Marks)

36. Explain the building blocks of an IoT system and describe the layered architecture of IoT with neat diagrams.
37. Explain the different IoT communication models in detail with suitable examples.
38. Discuss the advantages of IP in IoT with reference to interoperability, scalability, end-to-end coMnnectivity, and vendor independence
39. Explain how topology and dependency analysis help diagnose IoT network failures.

SGOU

SREENARAYANAGURU OPEN UNIVERSITY

QP CODE: ……

Reg. No. : .....

**Name: ……..... **

BACHELOR OF COMPUTER APPLICATION, SIXTH SEMESTER

END SEMESTER EXAMINATION

Discipline Specific Elective Course

B21CA24DE - Internet of Things

2023-2024 - Admission Onwards (CBCS - UG)

MODEL QUESTION PAPER- SET B

Time: 3 Hour | Max Marks: 70

SECTION A

Answer any **ten** *questions of the following. Each question carries* ***one*** *mark.*

(10 × 1 = 10 Marks)

1. Which driver relates to devices with limited battery, memory and processing power?
2. How many layers does the oneM2M architecture broadly divide an IoT system into?
3. What technology reduced the physical size and cost of modern sensors?
4. Which node collects information from all sensor nodes in a WSN?
5. Name any two common security threats in IoT communication.
6. Which IEEE standard defines Wi-Fi communication?
7. What is meant by network scalability?
8. State one factor that affects communication efficiency in IoT
9. What is IP optimization?
10. Write any one use of digital certificates in IoT.
11. What is the purpose of Actuators?
12. What are Polling and Long Polling?
13. Why can an extreme sensor value be misleading?

SGOU

14. What is Apache Spark?

15. What is Arduino

SECTION B

Answer any **five** *questions of the following. Each question carries* **two** *marks.*

(5×2 =10 Marks)

16. What are the key features of an IoT system?

17. What are the benefits of IT–OT convergence?

18. Differentiate between any two IoT communication models based on Internet requirement, cloud usage, and applications.

19. Explain the major challenges associated with IoT communication with suitable examples

20. What is interoperability? Why is it important in IoT?

21. What is packet overhead? How can it affect IoT communication?

22. Explain SCADA Communication.

23. Write about the IoT data lifecycle.

24. Differentiate between a threat and a vulnerability in OT security.

25. Explain how a vulnerability in an OT system can be exploited by an attacker

SECTION C

Answer any **five** *questions of the following. Each question carries* **four** *marks.*

(5×4= 20 Marks)

26. Compare the different IoT architectural models.

27. Explain the important characteristics of a good IoT communication protocol.

28. Explain any four types of sensors used in IoT with their applications and suitable examples

29. Compare IPv4 and IPv6 with respect to their addressing capabilities and suitability for IoT

30. Describe the effects of network congestion and packet loss on IoT communication

31. Explain the concept of Web Technologies and IoT.

32. Explain TCP reliability and flow control.

33. Explain topology and dependency analysis in an IoT deployment.

SGOU

34. Discuss four common threats and vulnerabilities affecting operational technology systems.

35. Compare the OCTAVE and FAIR frameworks in terms of their approaches to information security risk analysis.

SECTION D

Answer any **two** *questions of the following. Each question carries* **fifteen** *marks.*

(2×15 = 30Marks)

36. Explain the various applications of the Internet of Things (IoT) with suitable examples.

37. Explain Radio Frequency Identification (RFID) technology in detail. Describe the components and working principle of an RFID system. Also explain the advantages and limitations of RFID.

38. Explain the major IoT application layer protocols and discuss their features and applications.

39. Explain the common challenges in Operational Technology (OT) security, discussing major threats, vulnerabilities, and the importance of risk analysis.

SGOU

സർവ്വകലാശാലാഗീതം

വിദ്യയാൽ സ്വതന്ത്രരാകണം
വിശ്വപൗരരായി മാറണം
ഗ്രഹപ്രസാദമായ് വിളങ്ങണം
ഗുരുപ്രകാശമേ നയിക്കണേ

കൂരിരുട്ടിൽ നിന്നു ഞങ്ങളെ
സൂര്യവീഥിയിൽ തെളിക്കണം
സ്നേഹദീപ്തിയായ്വിളങ്ങണം
നീതിവൈജയന്തി പാറണം

ശാസ്ത്രവ്യാപ്തിയെന്നുമേകണം
ജാതിഭേദമാകെ മാറണം
ബോധരശ്മിയിൽ തിളങ്ങുവാൻ
ജ്ഞാനകേന്ദ്രമേ ജ്വലിക്കണേ

കുരീപ്പുഴ ശ്രീകുമാർ

SGOU

SREENARAYANAGURU OPEN UNIVERSITY

Regional Centres

Kollam
Sreenarayanaguru Open University
Headquarters Building
BSNL Bhavan, Vellayittambalam,
Kollam, Kerala, Pin : 691012
Ph: 9188912554
email:rckollam@sgou.ac.in

Tripunithura
Govt. College
Tirpunithura, Ernakulam
Kerala, Pin : 682301
Ph: 04842927436
email:rcedirector@sgou.ac.in

Thalassery
Govt. Brennen College
Dharmadam, Thalassery
Kannur, Pin : 670106
Ph: 04952990494
email:rctdirector@sgou.ac.in

Pattambi
Sree Neelakanta Govt. Sanskrit College
Pattambi, Palakkad
Kerala, Pin : 679303
Ph: 04662912009
email:rcpdirector@sgou.ac.in

Kozhikode
Govt. Arts and Science College
Meenchantha, Kozhikode,
Kerala, Pin : 673002
Ph: 04952920228
email:rckdirector@sgou.ac.in

SAY NO TO RAGGING

BEFORE YOU EVEN THINK OF RAGGING



Humiliation

Blacklisting

Ruined Career

Download

ANTI RAGGING

App

Don't just stand and watch. Stop Ragging! Show Character

Visit UGC Website i.e. www.ugc.ac.in & www.antiraging.in to UGC Anti Ragging regulations.

Ministry of Education
Government of India

Are You Being Ragged ?

विश्वविद्यालय अनुदान आयोग
University Grants Commission

SSGOU

DON'T LET IT BE TOO LATE

SAY NO TO DRUGS

LOVE YOURSELF AND ALWAYS BE HEALTHY



SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

Internet of Things

COURSE CODE: B21CA24DE

SGOU

Sreenarayanaguru Open University
Kollam, Kerala Pin- 691601, email: info@sgou.ac.in, www.sgou.ac.in Ph: +91 474 2966841

ISBN 978-81-903304-6-6