Internet of Things

Dr. Bhaskar Mondal

September 8, 2025

Contents

1 Introduction **7**
- 1.0.1 The Three Cs of IoT . . . 7
- 1.1 Modern Applications of IoT . . . 10
  - 1.1.1 Smart Homes . . . 10
  - 1.1.2 Healthcare . . . 10
  - 1.1.3 Agriculture . . . 11
  - 1.1.4 Smart Cities . . . 11
  - 1.1.5 Industrial IoT (IIoT) . . . 11
  - 1.1.6 Other Applications . . . 11
  - 1.1.7 Breaking Down the "Anything, Anywhere, Anytime" Paradigm: 12
  - 1.1.8 Types of IoT Connections . . . 13
  - 1.1.9 IoT Enabling Technologies . . . 14

2 Physical Design of IoT **17**
- 2.0.1 How IoT works . . . 18
- 2.0.2 IoT Protocol Layers . . . 22
- 2.0.3 Connectivity Layers . . . 27

3 Logical Design **29**
- 3.1 IoT functional blocks . . . 29
- 3.2 IoT Communication Models . . . 30
- 3.3 Communication Models . . . 31
- 3.4 Request–Response Communication Model . . . 31
  - 3.4.1 How It Works . . . 31
  - 3.4.2 Example in IoT . . . 32
  - 3.4.3 Communication Technologies Using This Model . . . 32
  - 3.4.4 Advantages . . . 32
  - 3.4.5 Limitations . . . 33
  - 3.4.6 When to Use in IoT . . . 33
  - 3.4.7 Hypertext Transfer Protocol (HTTP) . . . 33
- 3.5 Constrained Application Protocol (CoAP) . . . 35
  - 3.5.1 Why Not HTTP? The Motivation for CoAP . . . 35
  - 3.5.2 What is CoAP? . . . 36
  - 3.5.3 CoAP Architecture . . . 36
  - 3.5.4 CoAP Message Format . . . 37
  - 3.5.5 Security in CoAP . . . 37
  - 3.5.6 CoAP vs HTTP . . . 37
  - 3.5.7 Advantages of CoAP . . . 38
  - 3.5.8 Limitations of CoAP . . . 38
  - 3.5.9 Use Cases of CoAP . . . 38

3.5.10 Example Scenario . . . 38
3.5.11 Real-World Example . . . 39
3.6 Publish–Subscribe (Pub/Sub) communication model . . . 39
3.6.1 Core Concepts . . . 40
3.6.2 How It Works . . . 40
3.6.3 Advantages . . . 40
3.7 MQTT Protocol . . . 40
3.7.1 Why MQTT for IoT? . . . 41
3.7.2 MQTT Architecture . . . 41
3.7.3 MQTT Topics . . . 42
3.7.4 Quality of Service (QoS) . . . 42
3.7.5 Key Features of MQTT . . . 42
3.7.6 Real-World Applications . . . 42
3.7.7 Comparison with HTTP . . . 43
3.8 Push–Pull Communication Model . . . 43
3.8.1 Push Communication . . . 43
3.8.2 Example . . . 43
3.8.3 Advantages . . . 43
3.8.4 Disadvantages . . . 44
3.8.5 Pull Communication . . . 44
3.8.6 Example . . . 44
3.8.7 Advantages . . . 44
3.8.8 Disadvantages . . . 44
3.8.9 Push–Pull Hybrid Approach . . . 44
3.8.10 Comparison Table . . . 45
3.8.11 Applications in IoT . . . 45
3.9 Exclusive Pair Communication Model . . . 45
3.9.1 Working Principle . . . 46
3.9.2 Advantages . . . 46
3.9.3 Limitations . . . 46
3.9.4 IoT Applications . . . 47
3.10 REST-based Communication APIs . . . 47
3.10.1 REST in IoT Communication . . . 48
3.10.2 HTTP Methods in REST APIs . . . 48
3.10.3 Advantages of REST for IoT . . . 48
3.10.4 Limitations . . . 48
3.10.5 REST vs. Other Communication Protocols . . . 49
3.11 REST vs HTTP . . . 49
3.12 WebSocket-based Communication APIs . . . 50
3.12.1 Basics of WebSockets . . . 50
3.12.2 Key Features . . . 50
3.12.3 How WebSocket APIs Work . . . 51
3.12.4 Message Exchange Model . . . 51
3.12.5 Closing the Connection . . . 52
3.12.6 Advantages for IoT . . . 52
3.12.7 Implementation in IoT Systems . . . 52
3.12.8 Security Considerations . . . 53
3.13 Levels and Deployment Templates . . . 54
3.13.1 Levels of IoT Architecture . . . 54
3.13.2 Deployment Templates in IoT . . . 55

3.13.3 Example Mapping: Levels to Deployment Templates 57
3.14 IoT Communication Models and Their Relation to Communication Patterns 57
3.14.1 Device-to-Device (D2D) 57
3.14.2 Device-to-Gateway (D2G) 58
3.14.3 Device-to-Cloud (D2C) 58
3.14.4 Backend-to-Backend (B2B) 58
3.15 Topologies 59
3.15.1 IoT Network Topologies 59
3.15.2 Bus Topology 60
3.15.3 Hybrid Topology 60
3.15.4 Mapping Topologies to Communication Models 60

4 Sensors and Actuator **63**
4.1 Transducer 64
4.2 Types of Transducers 64
4.2.1 Based on Energy Conversion 64
4.2.2 Based on Output Signal 65
4.2.3 Examples of Transducers 65
4.3 Working Principle 65
4.4 Applications 66
4.5 Advantages and Limitations 66
4.5.1 Advantages 66
4.5.2 Limitations 66
4.6 Sensor 66
4.6.1 Role of Sensors 66
4.7 Good Features of Sensors 68
4.8 Sensor Resolution 73
4.9 Types of Sensors in IoT 74
4.10 Sensor Types based on Application 75
4.10.1 Temperature Sensors 75
4.10.2 Motion Sensors 75
4.10.3 Humidity Sensors 75
4.10.4 Gas Sensors 76
4.10.5 Other Sensors 76
4.11 Applications of Sensors in IoT 76
4.11.1 Smart Cities 76
4.11.2 Healthcare 76
4.11.3 Agriculture 76
4.11.4 Industrial IoT 76
4.12 Sensor Outputs 76
4.12.1 Voltage output 76
4.12.2 Current Output 77
4.12.3 Variable-Resistance Sensors 78
4.12.4 Frequency based Sensors 79
4.13 Working principals 80
4.14 Challenges and Future Trends 80
4.14.1 Challenges 80
4.14.2 Future Trends 81
4.15 Actuators in IoT 86

4.16 What is an Actuator? . . . 86
4.16.1 Definition . . . 86
4.16.2 Actuators in IoT Context . . . 87
4.17 Classification of Actuators . . . 87
4.17.1 By Energy Source . . . 87
4.17.2 By Motion . . . 87
4.17.3 By Control . . . 88
4.17.4 By Application Domain . . . 88
4.18 Detailed Types of Actuators in IoT . . . 89
4.18.1 DC Motors . . . 89
4.18.2 Stepper Motors . . . 90
4.18.3 Servo Motors . . . 91
4.18.4 Solenoids . . . 91
4.18.5 Piezoelectric Actuators . . . 92
4.18.6 Shape Memory Alloys (SMA) . . . 93
4.18.7 Hydraulic and Pneumatic Actuators . . . 94
4.18.8 MEMS Actuators . . . 95
4.19 IoT Case Studies Involving Actuators . . . 95
4.19.1 Smart Home HVAC (Heating, ventilation, and air conditioning) 95
4.19.2 Agriculture: Smart Irrigation . . . 95
4.19.3 Healthcare: Insulin Pumps . . . 95
4.19.4 Autonomous Vehicles . . . 95
4.20 Comparison of Major Actuators . . . 96
4.21 Future of Actuators in IoT . . . 96

6 Basics of IoT Networking 97
6.0.1 Convergence of Domains . . . 97
6.0.2 IoT Components . . . 97
6.0.3 IoT Interdependence . . . 99
6.0.4 IoT Service Oriented Architecture . . . 99
6.0.5 IoT Categories . . . 99
6.0.6 IoT Gateways . . . 100
6.0.7 Key Technologies for IoT . . . 100
6.0.8 Technical Deviations from Regular Web . . . 102
6.0.9 IoT Challenges . . . 102
6.0.10 Considerations . . . 103
6.0.11 IoT World Forum Standardized Architecture . . . 104
6.0.12 IoT World Forum Reference Model . . . 104

Chapter 1

Introduction

What is Internet? The word Internet comes from the combination of Interconnection and Network. The Internet of "Net" (Network of Networks) is the largest computer network in the world that connect billion of computer users. Where 'Network' is a collection of computers and devices connected through a communication channel and transmission media allowing one to share resources (hardware, software, data, etc.).)

Defining Internet of Things:

Internet technology connecting devices, machines, and tools to the internet by means of wireless technologies. Unification of technologies such as low power embedded systems, cloud computing, big data, machine learning, and networking.

In the 2000s, we are heading into a new era of ubiquity, where the "users" of the Internet will be counted in billions and where humans may become the minority as generators and receivers of traffic. Instead, most of the traffic will flow between devices and all kinds of "things", thereby creating a much wider and more complex Internet of Things. – ("The Internet of Things", ITU Internet Report 2005)

The Internet of Things (IoT) is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment[1]

IoT can also be defined as "The acceptable definition in the industry and essentially how ACG defines it is, IoT: Connectivity of devices not designed for direct human interaction, connectivity or control," says Dennis Ward, Internet of Things analyst at ACG. "It includes appliances, vehicles, machinery, sensors, monitors, devices – e.g. all verticals/horizontals - healthcare, financial, wearables, utilities, etc."

Why do we need IoT

- Continuous monitoring and surveillance
- Alarm in any unexpected or irregular events
- Enables Quick actions

1.0.1 The Three Cs of IoT

The three Cs of IoT are crucial elements that underpin the functionality of IoT systems:

[1]Gartner Research: http://www.gartner.com/it-glossary/internet-of-things/

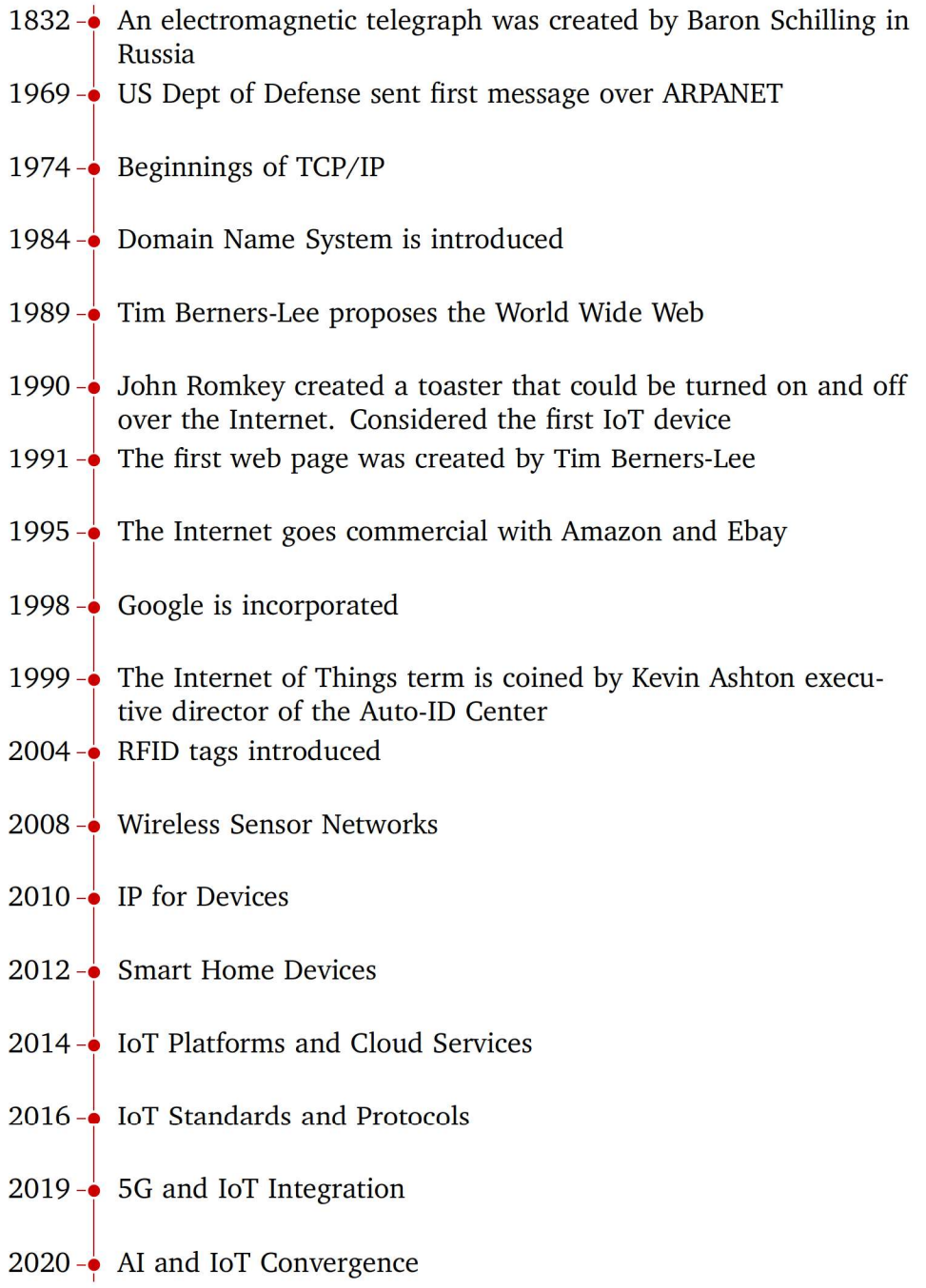


Figure 1.1: Milestones in the history of IoT

Connect

- Establishing Connectivity: This involves connecting various devices and sensors to the internet, enabling them to communicate with each other and with central systems.
- Communication Protocols: Different communication protocols, such as Wi-Fi, Bluetooth, Zigbee, LoRaWAN, and cellular networks, are used to facilitate data transmission between devices and the internet.

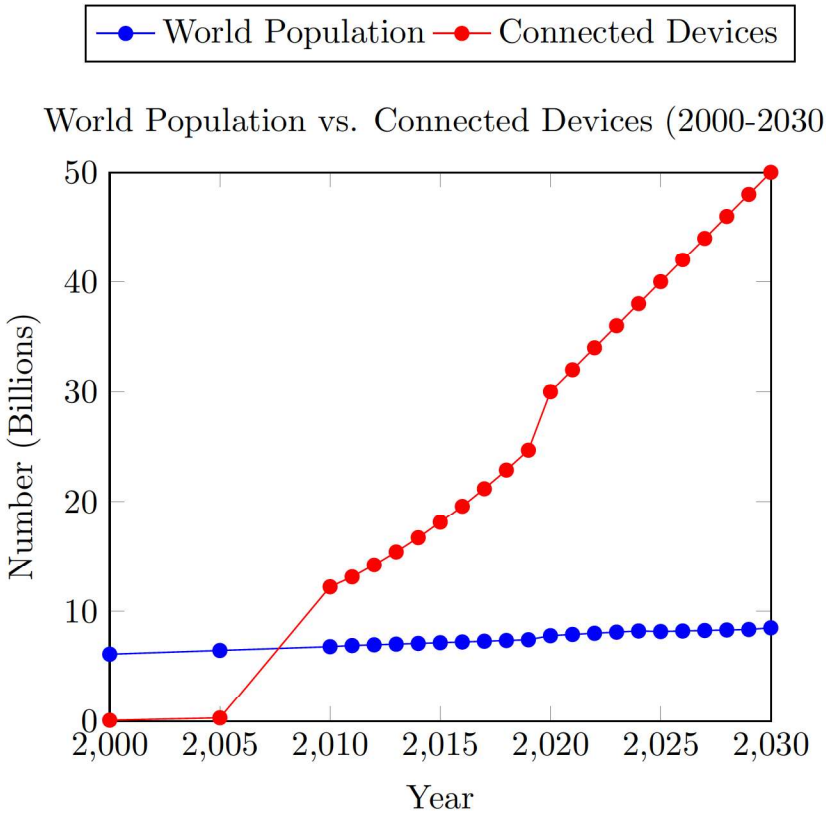


Figure 1.2: World Population vs. Connected Devices (2000-2030), Data source United Nations World Population Prospects 2022, and Statista

- Network Infrastructure: A robust network infrastructure, including gateways and routers, is essential to ensure seamless communication and data flow.

Control

- Remote Control: IoT devices can be controlled remotely through various interfaces, such as smartphone apps, web portals, or voice assistants.
- Automation: IoT devices can automate tasks and processes, reduces human intervention and improve efficiency.
- Decision-Making: By analyzing data collected from sensors, IoT systems can make informed decisions and trigger actions, such as adjusting temperature, activating alarms, or sending notifications.

Communicate

- Data Exchange: IoT devices exchange data with each other and with central systems, enabling them to share information and coordinate actions.
- Data Analysis: Collected data is analyzed to extract valuable insights and trends, which can be used to optimize processes and make data-driven decisions.

- User Interaction: IoT systems can interact with users through various channels, such as mobile apps, web portals, and voice assistants, providing information and enabling control.

By effectively combining these three Cs, IoT systems can revolutionize various industries and improve our daily lives.

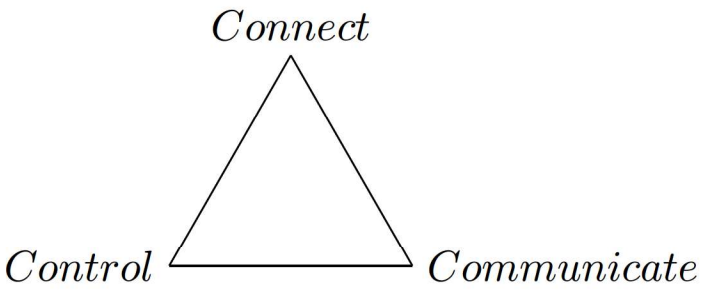


Figure 1.3: 3Cs of IoT

1.1 Modern Applications of IoT

The Internet of Things (IoT) has revolutionized various industries, bringing about significant advancements and transforming the way we live. By connecting everyday objects to the internet, IoT enables seamless communication and data exchange, leading to innovative applications across diverse sectors.

Here are some of the most prominent modern applications of IoT:

1.1.1 Smart Homes

- Automated Control: IoT devices like smart thermostats, lighting systems, and security cameras can be controlled remotely through smartphones or voice assistants.
- Energy Efficiency: IoT devices can optimize energy consumption by adjusting temperature, lighting, and appliance usage based on occupancy and real-time data.
- Enhanced Security: Smart security systems with IoT-enabled sensors can detect intrusions, monitor activity, and alert homeowners in case of emergencies.

1.1.2 Healthcare

- Remote Patient Monitoring: Wearable devices and IoT sensors can track vital signs, physical activity, and sleep patterns, providing valuable insights for healthcare professionals.
- Telemedicine: IoT enables remote consultations between patients and doctors, improving accessibility to healthcare services, especially in rural areas.
- Smart Medical Devices: IoT-enabled medical devices can provide real-time data on patient conditions, aiding in accurate diagnosis and treatment.

1.1.3 Agriculture

- Precision Agriculture: IoT sensors can monitor soil moisture, temperature, and other environmental factors, enabling farmers to optimize irrigation, fertilization, and pest control.
- Automated Farming: IoT-powered devices can automate tasks like irrigation, harvesting, and livestock monitoring, increasing efficiency and reducing labor costs.
- Predictive Maintenance: IoT can predict equipment failures, allowing farmers to schedule maintenance proactively and minimize downtime.

1.1.4 Smart Cities

- Smart Traffic Management: IoT-enabled traffic sensors can optimize traffic flow, reduce congestion, and improve air quality.
- Smart Grids: IoT can enhance energy distribution and consumption, leading to reduced energy costs and a more sustainable environment.
- Public Safety: IoT-based public safety systems can monitor crime rates, detect emergencies, and improve response times.

1.1.5 Industrial IoT (IIoT)

- Predictive Maintenance: IoT sensors can monitor equipment performance, predict maintenance needs, and optimize production processes.
- Supply Chain Optimization: IoT can track the movement of goods, optimize inventory management, and improve supply chain visibility.
- Quality Control: IoT-enabled sensors can monitor product quality and identify defects in real-time.

1.1.6 Other Applications

- Smart Retail: IoT can enhance the shopping experience through personalized recommendations, inventory management, and automated checkout.
- Logistics and Supply Chain: IoT can optimize logistics operations, improve delivery efficiency, and ensure product quality.
- Wearable Technology: Smartwatches, fitness trackers, and other wearable devices leverage IoT to monitor health, fitness, and lifestyle.

As IoT technology continues to evolve, we can expect to see even more innovative and impactful applications in the future.

The phrase "anything, anywhere, anytime" encapsulates the core promise of the Internet of Things (IoT). It signifies the potential to connect and control virtually any device or object, from any location, at any moment. This concept is driving innovation and transforming various industries.

1.1.7 Breaking Down the "Anything, Anywhere, Anytime" Paradigm:

1. Anything
   - Diverse Devices: IoT encompasses a vast array of devices, from simple sensors to complex machines.
   - Data-Driven Insights: These devices collect and transmit data, enabling data-driven decision-making and automation.
2. Anywhere
   - Global Connectivity: IoT devices can be deployed anywhere in the world, enabling remote monitoring and control.
   - Ubiquitous Access: With the proliferation of wireless networks and cellular connectivity, IoT devices can access the internet from almost any location.
3. Anytime
   - Real-time Monitoring: IoT devices can provide real-time data and insights, enabling timely responses to events and changes.
   - Continuous Operation: Many IoT devices are designed to operate continuously, ensuring uninterrupted data collection and analysis.

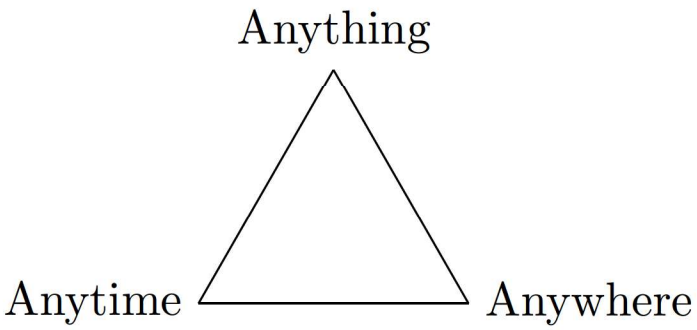


Figure 1.4: 3As of IoT

The Internet of Things (IoT) can be categorized into four main types:

1. Consumer IoT: Includes devices like smart home gadgets, wearable technology, and personal health trackers. The focus is on convenience for individual customers.
2. Commercial IoT: Includes tools and systems used outside of the home, such as connected devices in retail and healthcare.
3. Industrial IoT (IIoT): Includes interconnected devices in the industrial sector, such as manufacturing machinery and energy management devices. The focus is on improving efficiency, security, and output of operations.
4. Infrastructure IoT: Includes technologies like smart grids and traffic management systems that support urban infrastructure, such as smart cities and transportation systems.

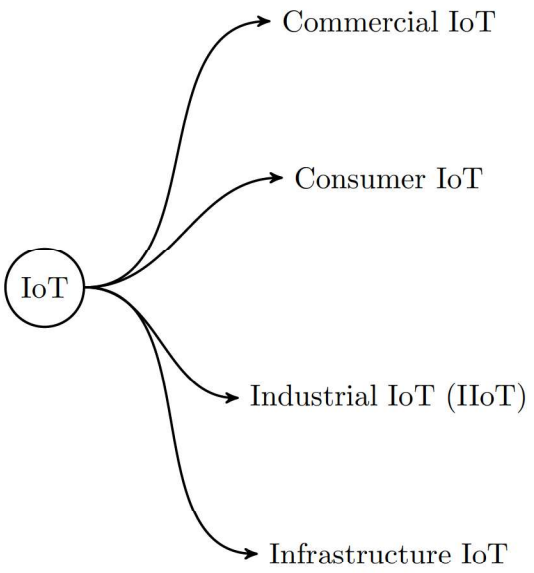


Figure 1.5: Types of IoT

1.1.8 Types of IoT Connections

In the Internet of Things (IoT), there are two main types of connections:

People Connected to Things

- This type of connection allows people to interact with and control devices directly. It involves a human interface (e.g., a smartphone app, wearable device, or web interface) through which people can communicate with IoT devices to monitor and manage their functionality.
- **Examples**:
  - **Smart Home**: Users can control lights, thermostats, and security cameras via apps, enabling personalized settings and remote control.
  - **Wearables**: Fitness trackers, smartwatches, and other wearables connect to mobile apps, allowing users to track health metrics, like heart rate, steps, and sleep patterns.
  - **Healthcare**: Patients and healthcare providers can monitor vital signs in real-time through connected medical devices, improving health outcomes and enabling remote care.

Things Connected to Things (Machine-to-Machine, or M2M)

- In this connection type, devices communicate directly with each other without human intervention. This is also known as Machine-to-Machine (M2M) communication, where devices exchange data autonomously to make decisions or trigger actions.
- **Examples**:
  - **Industrial IoT (IIoT)**: In manufacturing, machines on an assembly line can coordinate to adjust production speed, detect defects, and optimize energy use without human oversight.

- **Smart Grid**: Devices in an electrical grid, such as smart meters and transformers, communicate to balance supply and demand, reducing energy waste and improving grid reliability.
- **Autonomous Vehicles**: Vehicles can communicate with other vehicles (Vehicle-to-Vehicle, V2V) and infrastructure (Vehicle-to-Infrastructure, V2I) to enhance traffic flow, avoid collisions, and support autonomous driving.

These two connection types—People-to-Things and Things-to-Things—are foundational in IoT, enabling a range of applications from enhanced user control to fully automated systems.

1.1.9 IoT Enabling Technologies

The Internet of Things (IoT) relies on a set of enabling technologies that make it possible to connect, communicate, and process data between devices and systems. These technologies support IoT's core functions: sensing, connectivity, data processing, and user interaction. Here are the primary enabling technologies in IoT:

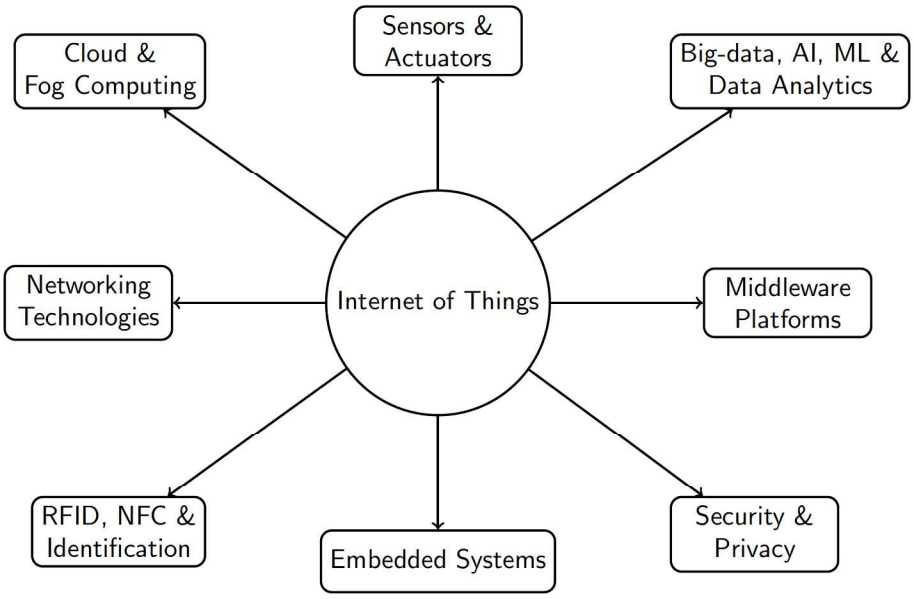


Figure 1.6: Types of IoT

Connectivity Technologies

- **Wi-Fi**: A widely used technology for connecting devices in home and office environments. Suitable for high-bandwidth applications.
- **Bluetooth and BLE (Bluetooth Low Energy)**: Ideal for short-range communication between devices; BLE consumes less power, making it suitable for wearable and low-power devices.

- **Cellular (4G, 5G)**: Provides wide-area network coverage, supporting mobile IoT applications in real-time or near-real-time. 5G, in particular, enables higher bandwidth and low latency, critical for real-time IoT applications like autonomous vehicles.

- **LPWAN (Low Power Wide Area Network)**: Technologies like LoRa, NB-IoT, and Sigfox are optimized for long-range, low-power, and low-data-rate applications, ideal for rural and large-scale deployments like smart agriculture.

- **Zigbee and Z-Wave**: Wireless communication protocols mainly used in home automation, offering low power consumption for battery-operated devices.

Sensor Technologies

- **Environmental Sensors**: Measure temperature, humidity, air quality, and other environmental conditions.

- **Motion and Position Sensors**: Include accelerometers, gyroscopes, and GPS to detect movement and location, often used in mobile and wearable devices.

- **Proximity and Occupancy Sensors**: Used in applications such as smart lighting, asset tracking, and security to detect presence and movement.

- **Optical and Imaging Sensors**: Used in surveillance, facial recognition, and autonomous vehicles to gather visual information.

- **Biosensors**: Track physiological data, commonly used in healthcare and wearable applications.

Cloud and Edge Computing

- **Cloud Computing**: Provides a scalable platform for data storage, processing, and analytics. Cloud services support IoT with data management, machine learning, and large-scale processing.

- **Edge Computing**: Processes data closer to the data source (edge devices) to reduce latency, conserve bandwidth, and enable real-time decision-making. Common in applications where immediate action is required, like autonomous systems and industrial automation.

Big Data Analytics and Machine Learning

- **Data Analytics**: Extracts actionable insights from vast amounts of IoT data, supporting applications from predictive maintenance in industrial settings to trend analysis in retail.

- **Machine Learning and AI**: Enhances decision-making by enabling predictive models, anomaly detection, and automation. Essential for applications like smart cities and healthcare monitoring.

Artificial Intelligence (AI) and Machine Learning (ML)

- **AI**: Facilitates advanced decision-making processes, such as identifying patterns in sensor data, optimizing operations, and enabling autonomous systems.
- **ML**: Processes IoT data to enable applications such as predictive maintenance, behavior analysis, and personalized recommendations.

Security Technologies

- **Encryption**: Protects data both in transit and at rest, ensuring that sensitive IoT data remains secure.
- **Authentication and Authorization**: Verifies the identity of devices and controls access to data and resources.
- **Blockchain**: Offers decentralized, secure methods for IoT data management, particularly useful for ensuring data integrity in complex systems like supply chains.

Data Storage Technologies

- **Databases**: Both relational (SQL) and non-relational (NoSQL) databases are used to store and manage the large volume of data generated by IoT devices.
- **Data Lakes**: Used to store massive amounts of raw IoT data, allowing for further processing and analysis when needed.

IoT Platforms and Middleware

- **IoT Platforms**: Provide a unified environment to manage IoT devices, process data, and develop applications. Examples include AWS IoT, Microsoft Azure IoT, and Google Cloud IoT.
- **Middleware**: Acts as a bridge, enabling interoperability between diverse IoT systems and devices.

Power Management Technologies

- **Energy Harvesting**: Collects energy from environmental sources (e.g., solar, thermal, vibration) to power IoT devices in remote locations.
- **Low-Power Design**: IoT devices are often designed for low power consumption to extend battery life, especially important for wearable and remote sensors.

These enabling technologies work together to support the seamless functionality of IoT systems across various domains, from smart homes and cities to industrial automation and healthcare.

What is not IoT Anything that requires human interaction or control is not considered an element of the IoT/E ecosystem, according to ACG Research and IDC. In some cases IP-enabled devices they do not consider elements of the Internet of Things/Everything. Non-IoT devises include computers, laptops, tables, cell phones, etc.

Chapter 2

Physical Design of IoT

Definition and Characteristics of IoT

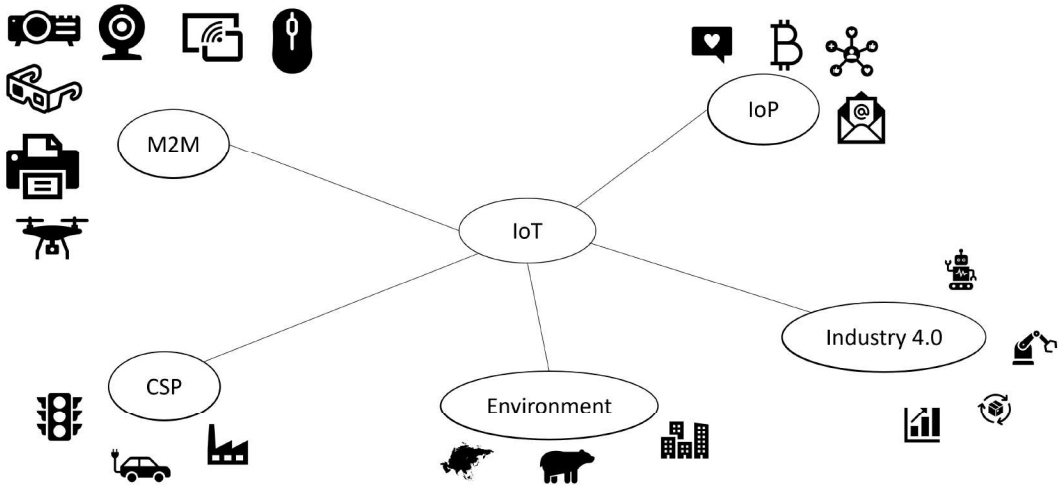


Figure 2.1: Terminological Interdependence

IoP Internet of People (IoP) refers to the network of people interconnected through digital platforms, wearables, mobile devices, social media, and smart services, enabling seamless human-to-human and human-to-machine communication. In short, IoP extends IoT by embedding humans as active nodes in the network, making IoT not just machine-to-machine (M2M), but people-to-people (P2P) and people-to-machine (P2M), driving more intelligent, personalized, and user-centric digital ecosystems.

M2M Machine-to-Machine (M2M) refers to direct communication between devices or systems without human intervention. In the context of the Internet of Things (IoT), M2M is the foundation that enables devices—such as sensors, actuators, smart meters, and industrial machines—to exchange data over wired or wireless networks. M2M is the communication backbone of IoT, allowing billions of devices to interact seamlessly and power applications across industries.

Devices collect data (e.g., temperature, pressure, location) through sensors, transmit it via communication protocols (such as Wi-Fi, ZigBee, LTE, or 5G), and trigger automated actions in other devices or systems.

Environment a process that uses Internet of Things (IoT) technology to collect data about the environment, such as air quality, temperature, and humidity levels, etc.

CPS Cyber-Physical Systems (CPS) are integrations of computation, networking, and physical processes. They tightly couple the cyber world (software, algorithms, communication) with the physical world (sensors, actuators, machines, and environments).

In the IoT context, CPS plays a crucial role by enabling real-time monitoring, control, and automation of physical systems through intelligent decision-making. IoT provides the sensing, communication, and data-sharing backbone, while CPS ensures that the collected data is processed and translated into timely actions in the physical world.

CPS in IoT enables a seamless fusion of the physical and digital worlds, making systems more efficient, autonomous, and intelligent. It forms the backbone of modern smart infrastructures and future innovations like smart cities and autonomous systems.

Industry 4.0 Industry 4.0 refers to the fourth industrial revolution, characterized by the integration of advanced digital technologies into manufacturing and industrial processes. At its core, it emphasizes smart factories, where machines, devices, and systems communicate and cooperate autonomously to optimize production.

The Internet of Things (IoT) plays a central role in enabling Industry 4.0 by connecting physical assets (sensors, machines, robots, production lines) with digital platforms. IoT enables real-time data collection, monitoring, and analysis, which drives intelligent decision-making and automation.

2.0.1 How IoT works

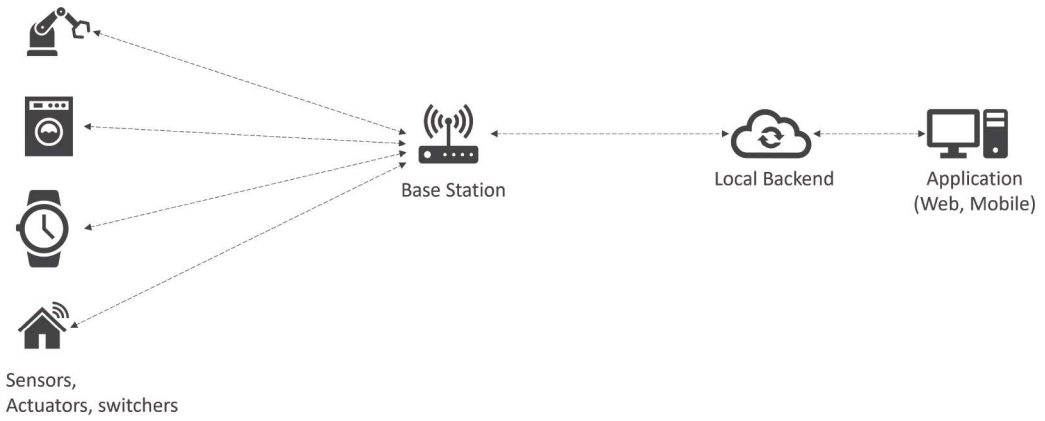


Figure 2.2: How IoT works

There are four fundamental components of an IoT system:

Sensors/Devices Sensors or devices are key component that helps to collect live data from the surrounding environmen t. All this data may have various levels of complexities. It could be a simple temperature monitoring sensor, or it may be in the form of the video feed.

Connectivity All the collected data is sent to a cloud infrastructure. The sensors should be connected to the cloud using various like mobile or satellite networks, Bluetooth, WI-FI, WAN, etc.

Data Processing This process can be just checking the temperature, reading on devices like AC or heaters. sometimes it can also be complex like identifying objects, using computer vision on video

User Interface triggering alarms on user's phones or sending them notification through email or text message. An interface can actively check IoT system. For example, the user has a camera installed in his home. He wants to access video recording and all the feeds with the help of a web server.

The physical design of IoT refers to the hardware components and physical devices that make up an IoT system. These components include sensors, actuators, connectivity devices, and embedded systems, all of which work together to collect, transmit, and process data. The physical design is crucial as it directly affects the functionality, efficiency, and reliability of IoT applications. Here's a breakdown of the primary components involved in the physical design of IoT:

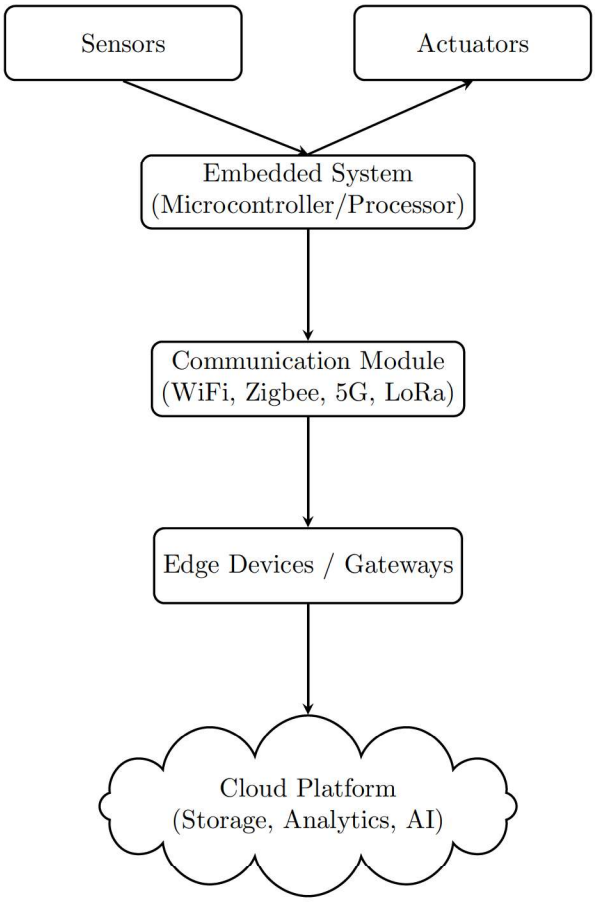


Figure 2.3: Physical Design

Sensors

- **Function**: Sensors are the input devices of an IoT system, responsible for gathering data from the physical environment. They detect changes in conditions like temperature, humidity, light, motion, and pressure.
- **Examples**:
  - Temperature sensors for monitoring climate conditions.
  - Proximity sensors for detecting object movement.
  - Optical sensors for capturing images and video.
- **Types**: Environmental sensors, motion sensors, chemical sensors, biosensors, and more, each designed for specific measurements in their respective applications.

Actuators

- **Function**: Actuators are output devices that act on the environment based on the commands they receive, usually in response to data collected by sensors.
- **Examples**:
  - Motors that adjust the position of solar panels.
  - Valves that control water flow in irrigation systems.
  - Speakers that produce sounds for alarms or notifications.
- **Types**: Linear actuators, rotary actuators, electric actuators, hydraulic actuators, etc.

Embedded Systems

- **Function**: These are small, dedicated computers or microcontrollers embedded within devices to perform specific tasks. Embedded systems often integrate sensors, actuators, and communication modules to process and control the data flow within IoT devices.
- **Examples**:
  - Microcontrollers like Arduino, Raspberry Pi, and ESP8266 that control smart home devices.
  - Specialized chips and modules for processing in wearable devices.
- **Design Considerations**: Power efficiency, processing capability, size, and compatibility with other components are essential to ensure optimal performance.

Communication Modules

- **Function**: These modules enable data transmission between IoT devices and the cloud or local network. They are essential for connecting IoT devices to each other and to central servers.
- **Examples**:
  - Wi-Fi modules for home automation.
  - Cellular modules for mobile IoT applications.
  - LoRa and Sigfox modules for low-power, wide-area applications.
- **Types**: Wi-Fi, Bluetooth, Zigbee, LoRa, NB-IoT, Sigfox, and cellular modules (2G/3G/4G/5G).

Power Sources

- **Function**: Powering IoT devices is a critical part of physical design, especially for devices in remote locations or where battery replacement is challenging.
- **Examples**:
  - Batteries for wearables and remote sensors.
  - Solar cells for IoT devices in outdoor, off-grid locations.
- **Considerations**: Low power consumption, energy harvesting, and battery life optimization are vital for the long-term sustainability of IoT devices.

Edge Devices and Gateways

- **Function**: Edge devices and gateways serve as intermediaries between IoT devices and the cloud. They collect data from sensors, process it at the edge, and then transmit the data to a cloud server or central system if needed.
- **Examples**:
  - Local processing units for real-time data analysis in industrial IoT.
  - Gateways that connect low-power devices in smart homes to Wi-Fi or cellular networks.
- **Benefits**: By processing data closer to the source, edge devices reduce latency, bandwidth usage, and dependency on cloud infrastructure.

User Interface Devices

- **Function**: While not strictly part of the physical "embedded" design, user interface devices allow users to interact with IoT systems by monitoring, managing, and controlling devices.
- **Examples**:
  - Smartphones and tablets with apps to control smart home devices.
  - Wearables like smartwatches for health tracking.

– Dashboard screens in vehicles for monitoring sensor data.

- **Design Elements**: User-friendliness, accessibility, and real-time feedback are critical to enhancing the usability of IoT interfaces.

The physical design of IoT is a blend of various interconnected components working in unison to collect, process, transmit, and act upon data in real time. The effectiveness of an IoT system depends on the careful selection, design, and integration of these physical components, keeping in mind the specific requirements of the application.

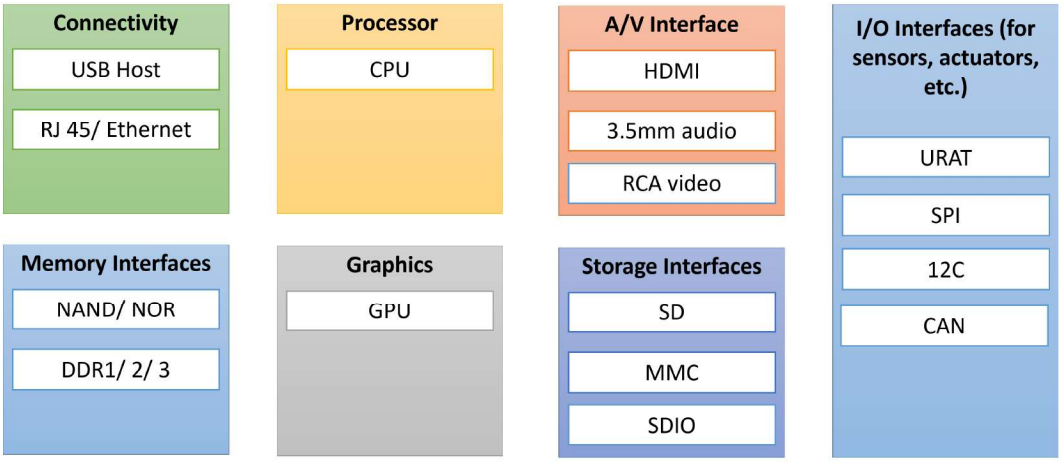


Figure 2.4: Physical Design of IoT Devices

An IoT device may consist of several interfaces for connections to other devices, both wired and wireless.

- I/O interfaces for sensors (UART: Universal Asynchronous Receiver/Transmitter; SPI: Serial Peripheral Interface; I2C: stands for Inter-Integrated Circuit; CAN: Controller Area Network protocol)
- Interfaces for internet connectivity
- Memory and storage interfaces (MMC: Multi Media Card)
- Audio/video interfaces

2.0.2 IoT Protocol Layers

Physical and MAC Layers

This layer covers how a device is physically connected to a network via wired or wireless mechanisms, as well as how devices are uniquely identified by a MAC address (or potentially another method) for physical addressing. Most standards combine the physical and MAC layer protocols; these protocols are essential in establishing communication channels. For IoT, considerations when designing at this level include devices that need to operate with a long battery life, require low power consumption, and have less processing capabilities. Other points to consider are lower bandwidth availability and the need to scale in terms of connecting and operating many more devices in a single environment.

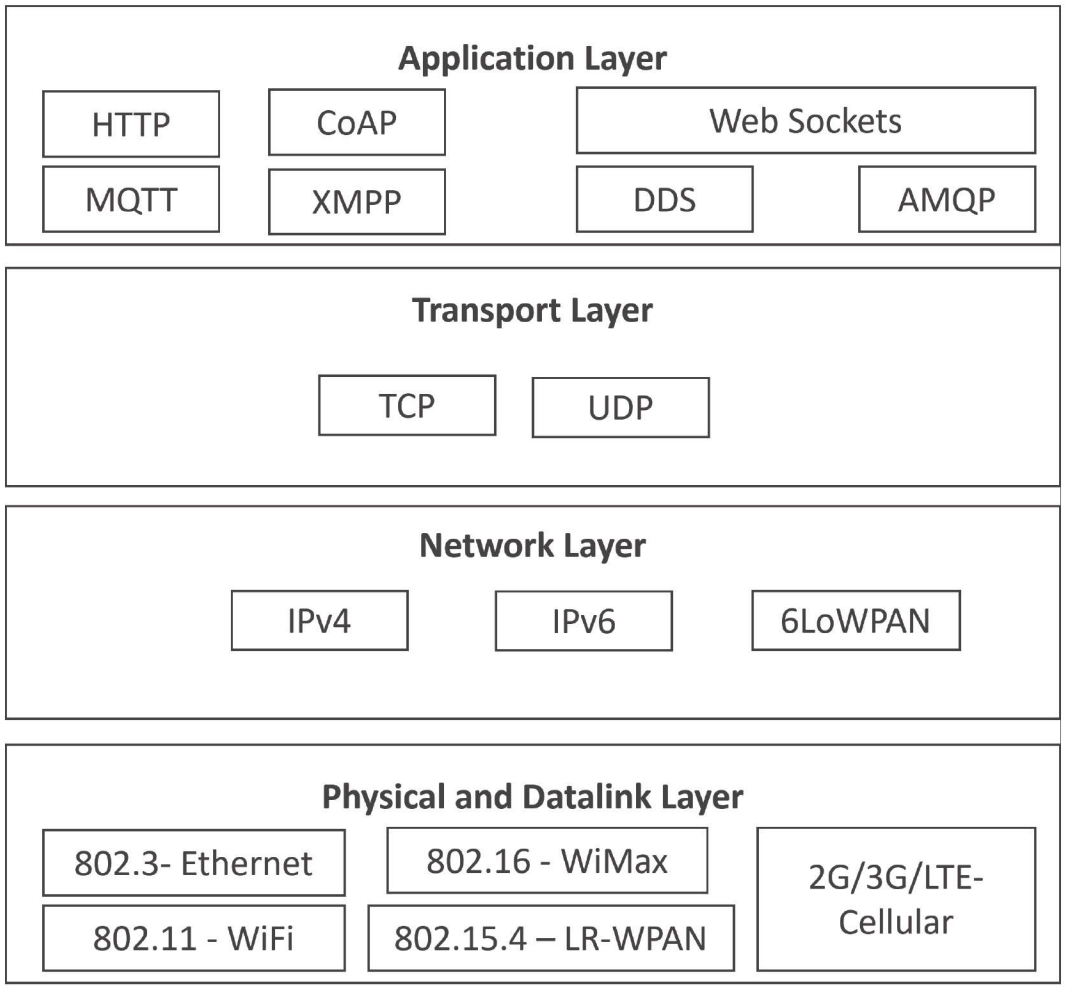


Figure 2.5: Layers of IoT communication and Protocols

In IoT, wired Ethernet 802.3 and Wi-Fi 802.11 a/b/g/n standards are often leveraged, depending on the environment. Smart cities and manufacturing plant floors are good examples with dense coverage. Other technologies in use include 802.15.4 (802.15.4e, 802.15.4g, WirelessHART, ISA100.11a), cellular (2G, 3G, 4G, CDMA, LTE), Low Power Wide Area Network LPWAN (Long Range Radio LoRa, SigFox, Narrow Band IoT NB-IoT), 802.16 WiMax, RFID, NFC, Bluetooth (including Bluetooth Low Energy BLE), and Zigbee.

Network Layer

This layer focuses on logical addressing and how to deliver packets of information between source and destination endpoints, particularly between different networks. Routing and encapsulation protocols need to be lightweight (constrained devices) and highly scalable (potentially millions of endpoints).

The Internet Protocol (IP) is an essential element of IoT. This includes both IPv4 and IPv6; the latter is essential to address scale. IPv4 provides around 4.3 billion addresses in total, which can create a challenge as we move toward the predicted 20–50 billion endpoints by 2020. IPv6 provides around 340 billion billion billion billion addresses, meaning that the scalability challenge is negated. However, not all IoT devices need a unique or a public address; many will be deployed on

IEEE Standard	Medium	Data Rate	Max Segment Length
IEEE 802.3	Coaxial cable	10 Mbps	500 meters
IEEE 802.3i (10BASE-T)	Twisted pair (Cat 3)	10 Mbps	100 meters
IEEE 802.3u (Fast Ethernet)	Twisted pair (Cat 5)	100 Mbps	100 meters
IEEE 802.3z (1000BASE-X)	Fiber optic	1 Gbps	550 meters (multi-mode fiber)
IEEE 802.3ab (1000BASE-T)	Twisted pair (Cat 5e)	1 Gbps	100 meters
IEEE 802.3ae (10GBASE)	Fiber optic	10 Gbps	10 kilometers (single-mode fiber)
IEEE 802.3an (10GBASE-T)	Twisted pair (Cat 6a)	10 Gbps	100 meters
IEEE 802.3ba (40GBASE, 100GBASE)	Fiber optic	40 Gbps, 100 Gbps	40 kilometers (single-mode fiber)
IEEE 802.3bs (200GBASE, 400GBASE)	Fiber optic	200 Gbps, 400 Gbps	10 kilometers (single-mode fiber)

Table 2.1: IEEE Ethernet Standards and Properties

IEEE Standard	Frequency Band	Max Data Rate	Range (Indoor)	Medium
IEEE 802.11	2.4 GHz	2 Mbps	20 m	DSSS/FHSS
IEEE 802.11b	2.4 GHz	11 Mbps	35 m	DSSS
IEEE 802.11a	5 GHz	54 Mbps	30 m	OFDM
IEEE 802.11g	2.4 GHz	54 Mbps	35 m	OFDM/DSSS
IEEE 802.11n	2.4/5 GHz	600 Mbps	70 m	OFDM/MIMO
IEEE 802.11ac	5 GHz	6.93 Gbps	35 m	OFDM/MU-MIMO
IEEE 802.11ax (Wi-Fi 6)	2.4/5 GHz	9.6 Gbps	50 m	OFDMA/MU-MIMO
IEEE 802.11ad (WiGig)	60 GHz	7 Gbps	10 m	OFDM

Table 2.2: IEEE WiFi Standards Overview

private networks that will continue to use private address ranges or will be hidden behind gateways at the edge and fog layers of the network. The use of IP not only provides interoperability benefits, but also helps with longevity and future-proofing of solutions. With the speed of change of IoT devices and technologies, the physical and data link layers evolve every few years. Using IP provides support for a smooth evolution of technologies, without changing core architectures, affecting the stability of deployments, or introducing new use cases. Even if the endpoints do not support IP, gateways can be deployed at the edge or fog levels to provide connectivity and transport, as well as to support multiple physical and data link layer types.

Table 2.3: IEEE WiMax Standards and Properties

Standard	**Frequency Band**	**Medium**	**Description/ Properties**
IEEE 802.16-2004	2 - 66 GHz	Wireless (NLOS[1])	Original fixed WiMax standard
IEEE 802.16e-2005	2 - 6 GHz	Wireless (NLOS)	Adds support for mobility (mobile WiMax)
IEEE 802.16j-2009	2 - 6 GHz	Wireless (NLOS)	Adds support for relay stations
IEEE 802.16m-2011	Sub-6 GHz	Wireless (NLOS)	Enhanced mobile WiMax with better throughput
IEEE 802.16h-2010	2 - 11 GHz	Wireless (NLOS)	Adds support for interference mitigation

[1] Non-Line of Sight (NLOS)

Generation	**Standard**	**Medium (Frequency)**	**Data Rate**	**Main Use Cases**
2G	GSM, CDMA (IS-95)	850 MHz, 900 MHz, 1800 MHz	Up to 64 kbps	Voice calls, SMS
3G	UMTS, WCDMA, CDMA2000	850 MHz, 900 MHz, 2100 MHz	384 kbps - 2 Mbps	Voice, video calls, basic internet
4G	LTE, WiMAX	600 MHz, 700 MHz, 2.6 GHz	Up to 1 Gbps (download)	HD video streaming, gaming, IoT
5G	NR (New Radio)	Sub-6 GHz, mmWave (24–52 GHz)	Up to 20 Gbps	Enhanced mobile broadband, IoT, autonomous vehicles
6G	TBD	THz range (100 GHz–1 THz)	Projected: Up to 1 Tbps	Holographic communications, AI integration, ultra-fast IoT

Table 2.4: IEEE Standards for Mobile Communication (2G/3G/4G/5G/6G)

Many last-mile communication options can be unreliable and unpredictable, so a new routing protocol was created to address routing for constrained devices such as those in wireless sensor networks. The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) routes IPv6 traffic over low-power networks and lossy networks (LLN). LLNs are a class of network in which both the devices and their communication mechanisms are constrained. LLN devices are typically constrained by processing power, memory, and battery; their communications are characterized by high loss rates, low data rates, and instability. LLNs can scale from a few dozen up to thousands of devices.

In areas with low-power radio communication, the IPv6 Low Power Wireless

Property	IPv4	IPv6	6LoWPAN
Address Length	32 bits	128 bits	Based on IPv6 (compressed)
Address Notation	Dotted Decimal (e.g., 192.168.0.1)	Hexadecimal (e.g., 2001:0db8::1)	IPv6-based compressed
Address Space	4.3 billion	˜340 undecillion	Limited to local network scope
Header Size	20 bytes	40 bytes	Compressed to fit low-power networks
Security	IPsec (optional)	IPsec (mandatory)	IPsec over IEEE 802.15.4
Network Size	Limited, lacks scalability	Large, supports scalability	Optimized for low-power networks
Use Case	Traditional networks	Future internet, mobile devices	IoT, sensor networks, low-power WPAN
Fragmentation	Required for certain links	Not necessary	Adapted for low-power link layers

Table 2.5: Comparison of IPv4, IPv6, and 6LoWPAN Properties

Personal Area Network (6LoWPAN) can be leveraged. It was designed with very constrained devices in mind and allows IPv6 to be used over 802.15.4 wireless networks. 6LoWPAN optimizes the transmission of IPv6 packets over LLNs such as IEEE 802.15.4 through header compression.

Transport Layer

This layer addresses secure end-to-end communication, including reliability, bandwidth, and congestion management, as well as sequencing and session maintenance. Operating in constrained and highly geographically dispersed environments, as well as leveraging physical media that is less reliable, makes User Datagram Protocol (UDP) the protocol of choice in place of the more heavyweight Transmission Control Protocol (TCP). Transport Layer Security (TLS) and Datagram TLS (DTLS) are typically leveraged for secure transport.

Application Layer

This layer covers application-level messaging and provides the interface between the user and the desired IoT application. Hypertext Transfer Protocol (HTTP) and Secure HTTP (HTTPS) continue to be leveraged in IoT. In addition, the Constrained Application Protocol (CoAP) is often leveraged as a lightweight alternative to HTTP as a specialized web transfer protocol for use with constrained nodes and constrained networks. It is often used in combination with 6LoWPAN.

To allow data exchange and facilitate control of the data pipeline, a messaging service is often leveraged within IoT deployments. Messaging protocols such as Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), and Extensible Messaging and Presence Protocol (XMPP) have been leveraged for some time. More recently, feature-rich message services such as the Cisco Edge Fog Fabric (EFF) have been introduced, providing detailed topologies,

strong QoS mechanisms, and real-time analytics capabilities as part of the IoT data/content pipeline management.

Industrial IoT environments and specific markets continue to use more industry-specific protocols that have been designed over many years to address certain vertical or market needs. IEC 61850 Sampled Values (SV), Generic Object Oriented Substation Event (GOOSE), and Manufacturing Message Specification (MMS), IEC 60870, Modbus, Distributed Network Protocol (DNP3), and OLE for Process Control (OPC), provide the core communication mechanisms for industrial environments such as power utilities, manufacturing, oil and gas, and transportation.

Many IoT environments, particularly industrials, have a requirement to connect legacy devices and sensors. This means that, in addition to IP- and Ethernet-based protocols, serial-based protocols must be connected. This not only adds integration complexity, but it introduces security considerations.

In summary, as well as in practice, different IoT standards use many of these protocols. Choosing the right protocol often comes down to the vendor, the environment, the network topology, the bandwidth available, and the vertical or market in which the use case will be deployed. Other considerations can include power constraints and usage, reliability requirements, and, of course, security. Some of the options listed, such as IEEE 802.15.4, have security mechanisms built in, such as access control, message integrity, replay protection, and message confidentiality.

2.0.3 Connectivity Layers

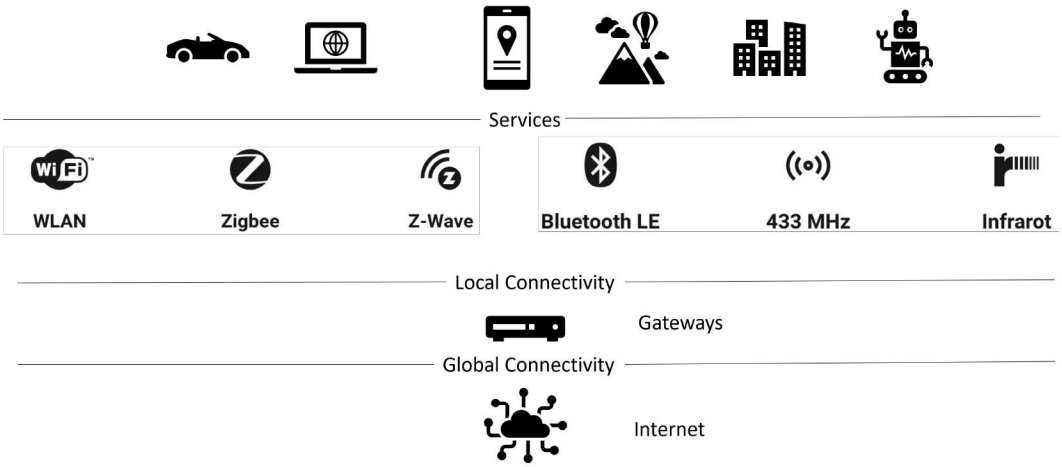


Figure 2.6: Connectivity Layers

Local Connectivity

Mainly Short-Range wireless Technologies are used

- Wi-Fi: Offers high data rates but requires significant power consumption.
- Bluetooth: Provides reliable, low-power communication for short distances.
- Zigbee: Energy-efficient mesh networking for home automation and industrial applications.
- Z-Wave: Secure, low-power wireless protocol for home automation.

Global Connectivity

- Cellular Networks:
  - 4G/5G: High-speed connectivity for mobile devices and IoT applications requiring high bandwidth.
- LPWAN Technologies:
  - LoRaWAN: Long-range, low-power communication for various IoT applications, including smart cities and agriculture.
  - NB-IoT: Narrowband IoT, optimized for low-power, low-bandwidth devices like sensors and trackers.
  - Sigfox: Low-power, wide-area network technology for simple IoT devices.

Key Considerations for IoT Connectivity

- Security: Strong encryption and authentication mechanisms are essential to protect sensitive data.
- Power Efficiency: Low-power technologies and energy-saving techniques are crucial for battery-powered devices.
- Scalability: The infrastructure should be able to handle a growing number of devices and data.
- Reliability: Reliable connectivity is essential for critical IoT applications.
- Cost-Effectiveness: The choice of technologies should balance performance and cost.

Chapter 3

Logical Design

3.1 IoT functional blocks

Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation.

An IoT system comprises a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication and management.

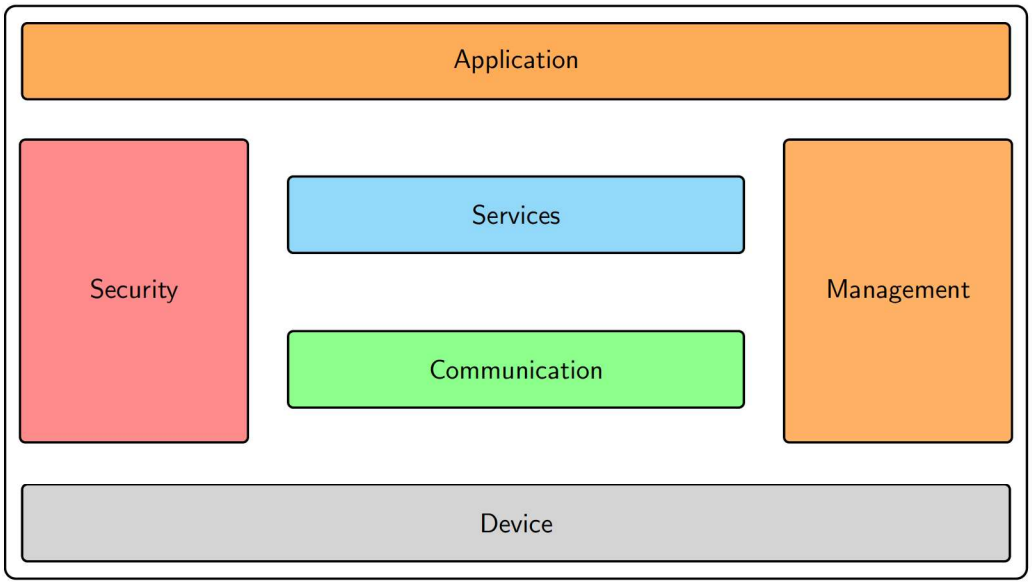


Figure 3.1: IoT functional blocks

- **Device** : Devices such as sensing, actuation, monitoring and control functions.
- **Communication** : IoT Protocols Services like device monitoring, device control services, data publishing services and device discovery
- **Management** : Functions to govern the system
- **Security** : Functions as authentication, authorization, message and content integrity, and data security Applications

3.2 IoT Communication Models

The Internet of Things (IoT) uses different communication models depending on how devices, gateways, and cloud systems interact. Understanding these models is essential for IoT engineers, as each model has its advantages, limitations, and application scenarios.

1. Device-to-Device (D2D) Communication

In this model, IoT devices communicate directly with each other without involving any intermediate gateway or cloud server. It is useful for short-range, low-power communication, such as smart home appliances, wearable devices, or industrial sensors that need to exchange data locally.

Examples: Bluetooth Low Energy (BLE), ZigBee, Z-Wave, Near Field Communication (NFC), Thread.

2. Device-to-Gateway (D2G) Communication

Here, IoT devices connect to an intermediate gateway (also called an edge device) which aggregates, processes, and forwards data to the cloud. The gateway often provides protocol translation, security, and connectivity to the Internet. This model is common in smart homes, healthcare monitoring systems, and industrial IoT.

Examples: Wi-Fi, ZigBee, Bluetooth, 6LoWPAN, LoRaWAN (end devices to gateway).

3. Device-to-Cloud (D2C) Communication

In this model, devices communicate directly with a cloud service using Internet-based protocols. This allows remote monitoring, large-scale analytics, and integration with web/mobile applications. However, it requires reliable Internet connectivity.

Examples: MQTT, CoAP, HTTP/HTTPS, WebSockets, AMQP.

4. Backend-to-Backend (B2B) Communication

Once data reaches the cloud, different cloud platforms or backend systems often need to communicate with each other. This model ensures interoperability between different organizations, applications, or services. It supports big data analytics, enterprise integration, and cross-platform IoT solutions.

Examples: RESTful APIs, MQTT broker-to-broker bridging, AMQP (between servers), XMPP, Data Distribution Service (DDS).

Communication Model	**Example Protocols**
Device-to-Device (D2D)	Bluetooth Low Energy (BLE), ZigBee, Z-Wave, NFC, Thread
Device-to-Gateway (D2G)	Wi-Fi, ZigBee, Bluetooth, 6LoWPAN, LoRaWAN
Device-to-Cloud (D2C)	MQTT, CoAP, HTTP/HTTPS, WebSockets, AMQP
Backend-to-Backend (B2B)	RESTful APIs, MQTT bridging, AMQP, XMPP, DDS

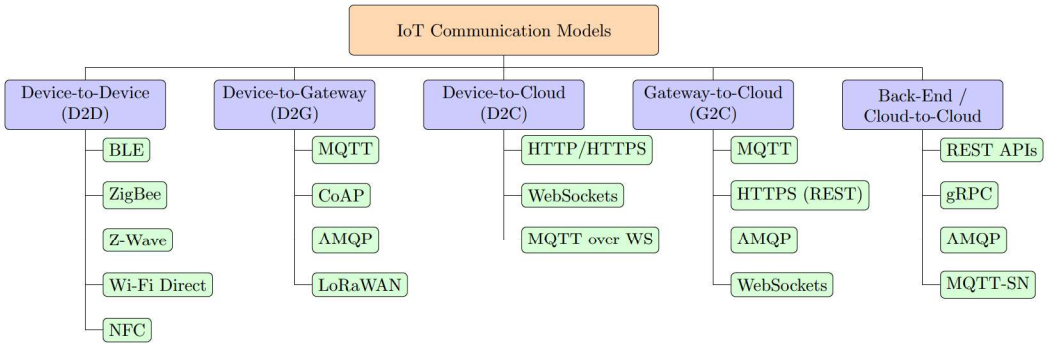


Figure 3.2: IoT Communication Models

3.3 Communication Models

- Request–Response Communication Model
- Publish–Subscribe Communication Model
- Push–Pull Communication Model
- Exclusive Pair Communication Model
- REST-based Communication APIs
- WebSocket-based Communication APIs
- IoT communication API

3.4 Request–Response Communication Model

The **Request–Response Model** is a communication pattern where one device (the *client*) sends a **request** to another device or server (the *server*), and the server sends back a **response**.
Analogy: Think of it like ordering food in a restaurant:

- You (the client) tell the waiter (the server) what you want — **Request**.
- The waiter brings your food — **Response**.

In IoT terms:

- A **sensor node** or IoT application may request data from another device or cloud service.
- The **server** (device, gateway, or cloud API) processes the request and returns the required data or action result.

3.4.1 How It Works

The basic flow is:

1. **Client Sends Request** — Contains details about what data or action is needed. Example: "Give me the temperature reading."

2. **Server Processes Request** — The server checks the request, retrieves or processes the data, and prepares a response.
3. **Server Sends Response** — Contains the result of the request (data, confirmation, or error message).
4. **Client Receives Response** — The client processes the data and may display it or trigger other actions.

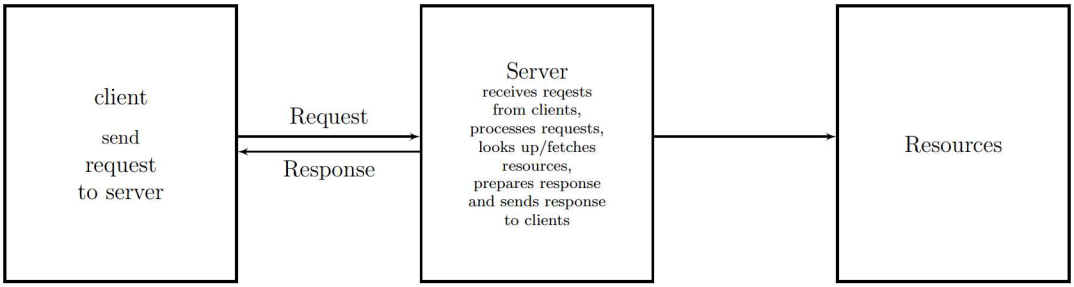


Figure 3.3: Request–Response Communication Model

3.4.2 Example in IoT

Consider a smart home temperature monitoring system:

- **Client:** A smartphone app.
- **Server:** A Raspberry Pi with a connected temperature sensor.

Process:

1. The app sends a request: `GET /temperature`
2. The Raspberry Pi reads the sensor data and replies: `Temperature: 25 °C`

3.4.3 Communication Technologies Using This Model

Several IoT communication protocols follow the request–response pattern:

- HTTP/HTTPS (Common in RESTful APIs)
- CoAP (Constrained Application Protocol for resource-limited IoT devices)

3.4.4 Advantages

- **Simplicity:** Easy to implement and understand.
- **Predictable Behavior:** Every request has a matching response.
- **Compatibility:** Works well with existing web and network infrastructure.
- **Error Handling:** Clients know if the request failed based on the response code.

3.4.5 Limitations

- **Latency:** The client must wait for the response before proceeding.
- **Inefficiency for Continuous Data:** Not ideal for streaming real-time sensor data.
- **Dependency on Availability:** If the server is offline, the request fails.

3.4.6 When to Use in IoT

The request–response model is best suited for:

- On-demand data retrieval (e.g., fetching temperature, humidity, device status).
- Configuration changes (e.g., updating device settings).
- Remote command execution (e.g., turning on a light).

The Request–Response Communication Model forms the backbone of many IoT systems because of its simplicity and reliability. However, in designing IoT architectures, engineers should also consider other patterns like **publish–subscribe** for continuous or large-scale data streaming. A balanced combination of models ensures that IoT systems remain **efficient**, **scalable**, and **responsive**.

3.4.7 Hypertext Transfer Protocol (HTTP)

When we talk about the Internet, one of the most fundamental technologies that makes it work is **HTTP (Hypertext Transfer Protocol)**. As Computer Science undergraduates, you may already know HTTP as the protocol behind websites. But in the world of the Internet of Things (IoT), HTTP plays an equally important role. Let us explore how.

What is HTTP?

HTTP is an **application-layer protocol** used for transferring data between a client and a server. A client (such as a browser or an IoT device) sends an HTTP request, and the server responds with the required information, often in the form of text, JSON, or multimedia data. It works on top of **TCP/IP**, ensuring reliable communication.

For example, when you open a website, your browser sends an HTTP request such as:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

The server then responds with the HTML page requested.

Why HTTP Matters for IoT

IoT devices—such as sensors, smart appliances, and wearables—often need to communicate with cloud servers or applications. HTTP provides a simple and widely understood way for these devices to **send data** (e.g., temperature readings) and **receive commands** (e.g., turn on the light).

Some reasons HTTP is popular in IoT:

- **Ubiquity:** HTTP is universally supported. Almost all platforms and languages have HTTP libraries.
- **Human-readability:** Requests and responses are text-based, making them easy to debug.
- **Integration with Web Services:** IoT systems often interact with REST APIs that rely on HTTP.

Structure of HTTP Communication

HTTP follows a **request–response model**:

1. **Request:** An IoT device (client) sends a request to the server. Example: A temperature sensor posts data.

```
POST /data HTTP/1.1
Content-Type: application/json

{ "sensor_id": "temp01", "value": 27.3 }
```

2. **Response:** The server acknowledges and may send instructions.

```
HTTP/1.1 200 OK
Content-Type: application/json

{ "status": "received" }
```

HTTP Methods in IoT

The main HTTP methods relevant for IoT are:

- **GET:** Retrieve information (e.g., check device status).
- **POST:** Send data (e.g., sensor readings).
- **PUT:** Update resources (e.g., update device configuration).
- **DELETE:** Remove resources (e.g., unregister a device).

These methods align with the RESTful architecture commonly used in IoT platforms.

Challenges of HTTP in IoT

While HTTP is useful, it is not always the best choice for every IoT scenario:

- **Overhead:** HTTP headers are relatively large, consuming bandwidth and power.

- **Latency:** It is not optimized for real-time, low-latency communication.
- **Resource Requirements:** Some constrained devices may not have enough memory or processing power to handle HTTP efficiently.

For such cases, lightweight protocols like **MQTT** or **CoAP** are preferred. Still, HTTP remains the go-to for IoT systems that interact directly with web applications or APIs.

HTTP is more than just the protocol behind websites—it is also a foundational technology in IoT communication. By enabling devices to exchange data with servers and web applications, HTTP simplifies the integration of IoT into the broader Internet. However, as engineers, you must evaluate whether HTTP is suitable for a particular IoT use case, or whether alternatives like MQTT or CoAP provide better efficiency. Understanding HTTP gives you a solid base for exploring advanced IoT communication models.

3.5 Constrained Application Protocol (CoAP)

The **Internet of Things (IoT)** is reshaping how devices interact with each other and with humans. From smart homes and wearable devices to industrial automation and healthcare monitoring, billions of "things" are being connected to the internet. These devices, however, are often **resource-constrained**: they have limited power, processing capability, and memory.

For such devices, using traditional web communication protocols like **HTTP** is inefficient. HTTP is text-based, verbose, and optimized for desktops and servers rather than tiny sensors running on battery. This is where the **Constrained Application Protocol (CoAP)** comes in.

CoAP is a specialized web transfer protocol designed to enable **lightweight communication** between constrained devices in **low-power, lossy networks (LLNs)**. Standardized by the IETF in **RFC 7252 (2014)**, CoAP has become one of the foundational protocols in the IoT ecosystem.

This article explains CoAP in detail—its architecture, features, working principles, advantages, and use cases—so you can understand why it is often referred to as the "HTTP for IoT."

3.5.1 Why Not HTTP? The Motivation for CoAP

Before diving into CoAP, let us understand why HTTP is not ideal for IoT:

- **Overhead:** HTTP uses verbose text headers, which can consume hundreds of bytes per request. For a sensor with only a few kilobytes of RAM, this is wasteful.
- **Transport Layer:** HTTP typically runs over TCP, which is reliable but heavyweight for constrained devices. TCP's handshake, acknowledgments, and congestion control consume bandwidth and power.
- **Battery Drain:** IoT devices often run on batteries. TCP+HTTP communication consumes more energy compared to lightweight alternatives.
- **Broadcast/Multicast Support:** HTTP is designed for one-to-one communication, not efficient for scenarios where one server needs to notify multiple clients simultaneously.

CoAP solves these issues by being **binary, lightweight, UDP-based, and multicast-friendly**, while still being compatible with the web architecture (REST).

3.5.2 What is CoAP?

CoAP stands for **Constrained Application Protocol**. It is a **RESTful application-layer protocol** that allows constrained devices to communicate using the same principles as the web. Like HTTP, CoAP uses methods such as **GET, POST, PUT, and DELETE** to interact with resources.

The main differences are:

- CoAP runs over **UDP** instead of TCP, making it faster and lighter.
- CoAP uses a **compact binary message format**, reducing overhead.
- CoAP supports **asynchronous communication** and **multicast**.
- CoAP messages fit within a single **datagram (often 64–128 bytes)**, suitable for small packets in wireless sensor networks.

Thus, CoAP enables even tiny IoT sensors to participate in the "web of things."

3.5.3 CoAP Architecture

RESTful Design

Like HTTP, CoAP is based on REST (Representational State Transfer). Devices expose their data as **resources**, identified by URIs (Uniform Resource Identifiers). For example:

```
coap://sensor1.local/temperature
coap://lightbulb.local/state
```

Message Layer

At the core of CoAP is a **message layer** that ensures reliability despite using UDP. Each message has a **Message ID** and a **type**:

- **Confirmable (CON):** Requires acknowledgment (like TCP reliability).
- **Non-confirmable (NON):** No acknowledgment needed (like UDP fire-and-forget).
- **Acknowledgment (ACK):** Confirms receipt of a CON message.
- **Reset (RST):** Indicates a message was received but could not be processed.

Request/Response Model

On top of the message layer sits the **request/response layer**, which defines methods similar to HTTP:

- **GET:** Retrieve a resource.
- **POST:** Create or update a resource.
- **PUT:** Update a resource.
- **DELETE:** Remove a resource.

Asynchronous Notifications (Observe Option)

One of CoAP's powerful features is the **Observe mechanism**. Instead of clients repeatedly polling a sensor, they can "observe" a resource. Whenever the resource changes (e.g., temperature rises), the server **pushes updates** to all subscribed clients.

Resource Discovery

CoAP defines a special resource `/.well-known/core` where devices can publish their available resources. Clients can query this to discover what a device offers.

3.5.4 CoAP Message Format

CoAP messages are compact and designed to fit in constrained networks. Each message consists of:

- **Header (4 bytes):** Includes version, type, token length, code, and message ID.
- **Token (0–8 bytes):** Used to match requests with responses.
- **Options (variable):** Key-value pairs for additional parameters (e.g., URI path, content format).
- **Payload (optional):** Actual data (e.g., temperature value).

For example, a GET request to `/temperature` might only take a few dozen bytes.

3.5.5 Security in CoAP

Security is essential in IoT. CoAP relies on **DTLS (Datagram Transport Layer Security)** to provide:

- Authentication
- Encryption
- Integrity

DTLS is essentially "TLS for UDP." However, DTLS adds overhead, so in extremely constrained networks, lightweight alternatives like **OSCORE (Object Security for Constrained RESTful Environments)** are used.

3.5.6 CoAP vs HTTP

Feature	HTTP	CoAP
Transport Layer	TCP	UDP
Message Format	Text (verbose)	Binary (compact)
Overhead	High	Low
Reliability	Built-in (TCP)	Optional (CON messages)
Multicast Support	No	Yes
Designed For	Web applications	IoT constrained devices

3.5.7 Advantages of CoAP

- **Lightweight and efficient:** Small messages, ideal for constrained devices.
- **RESTful compatibility:** Easy to integrate with web services.
- **Supports multicast:** Efficient for group communication.
- **Asynchronous communication:** Observe mechanism avoids polling.
- **Interoperability:** Proxies can translate CoAP to HTTP, enabling integration with existing web systems.

3.5.8 Limitations of CoAP

- **UDP dependency:** Some networks block UDP traffic.
- **Security overhead:** DTLS can still be heavy for ultra-low-power devices.
- **Scalability challenges:** Not as mature as HTTP for very large deployments.
- **Reliability trade-offs:** Requires careful design to avoid message loss.

3.5.9 Use Cases of CoAP

- **Smart Homes:** Light switches, thermostats, and appliances controlled via CoAP.
- **Healthcare IoT:** Wearable sensors transmitting patient vitals.
- **Industrial IoT:** Machine monitoring and predictive maintenance.
- **Environmental Monitoring:** Sensors reporting temperature, humidity, or pollution levels.
- **Smart Cities:** Streetlights, parking meters, and waste bins communicating using CoAP.

3.5.10 Example Scenario

Imagine a **smart agriculture system**. A soil moisture sensor exposes its data at:

```
coap://farm-sensor1.local/moisture
```

A client application sends a **GET** request to fetch the moisture level. If the farmer wants real-time updates, the app can register an **Observe** option. The sensor will then automatically send notifications whenever moisture levels drop below a threshold.

If irrigation is needed, the client can send a **PUT** request to:

```
coap://farm-actuator1.local/pump
```

with payload `ON` to start the water pump.

Future of CoAP

With the growth of **IPv6, 6LoWPAN, and low-power wireless technologies** like Zigbee, Thread, and NB-IoT, CoAP's relevance is increasing. Its ability to integrate with **HTTP via proxies** ensures that IoT devices can interact seamlessly with traditional web services. Moreover, the adoption of **OSCORE** for end-to-end security is addressing security concerns in constrained networks.

As billions of IoT devices continue to join the internet, protocols like CoAP will be critical in ensuring **scalable, secure, and efficient communication**.

The Constrained Application Protocol (CoAP) is a **lightweight, UDP-based, RESTful protocol** designed specifically for the challenges of IoT. By combining the familiarity of HTTP's REST principles with optimizations for constrained environments, CoAP bridges the gap between tiny sensors and the global web.

For computer science students, understanding CoAP is essential, as it not only highlights how traditional web technologies are adapted for IoT but also demonstrates the importance of designing **efficient, secure, and scalable systems** in a resource-constrained world.

In short, CoAP is the "language" that lets tiny IoT devices speak the web's language—efficiently and effectively.

3.5.11 Real-World Example

Consider a smart home system:

- A **temperature sensor** publishes readings to topic `/home/livingroom/temperature`.
- A **humidity sensor** publishes readings to topic `/home/livingroom/humidity`.
- A **dashboard application** subscribes to both topics for display.
- A **mobile app** subscribes only to temperature data.
- An **alert system** subscribes to humidity data to trigger dehumidifiers.

The Publish–Subscribe model is a cornerstone of modern IoT communication. It provides the scalability, flexibility, and reliability necessary for distributed and dynamic IoT environments. Protocols like **MQTT** and **AMQP** are built on this model, making it a must-know concept for IoT engineers.

3.6 Publish–Subscribe (Pub/Sub) communication model

In Internet of Things (IoT) systems, devices must exchange data efficiently, reliably, and in a scalable manner. One of the most widely used messaging patterns to achieve this is the **Publish–Subscribe (Pub/Sub) communication model**. Unlike the traditional **client–server** model, Pub/Sub decouples the sender (publisher) and receiver (subscriber) in terms of space, time, and synchronization. This makes it especially suitable for IoT networks where devices may be intermittently connected.

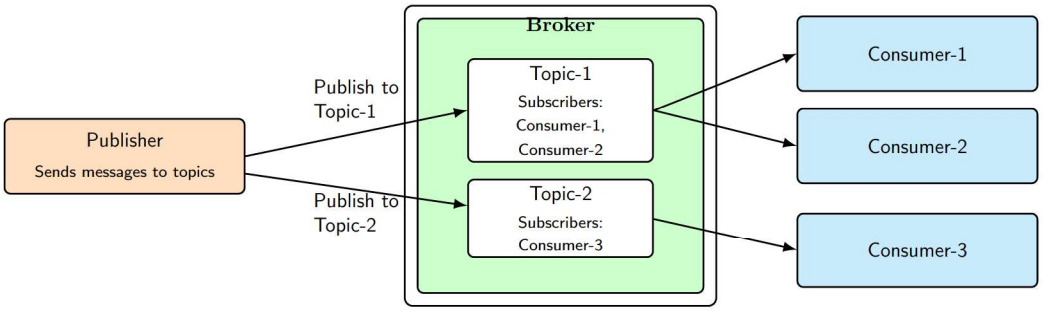


Figure 3.4: Request–Response Communication Model

3.6.1 Core Concepts

In a Publish–Subscribe architecture:

- **Publisher:** A device or application that sends messages (e.g., temperature readings, sensor status).
- **Subscriber:** A device or application that expresses interest in receiving specific messages.
- **Broker:** An intermediary entity that receives all messages from publishers and routes them to appropriate subscribers based on topics.
- **Topic:** A logical channel or subject name used to categorize messages (e.g., `/home/livingroom/temperature`).

3.6.2 How It Works

1. Publishers send messages to the broker with an associated topic.
2. The broker maintains a list of subscribers for each topic.
3. When a message is received, the broker forwards it only to the subscribers interested in that topic.

3.6.3 Advantages

- **Decoupling:** Publishers and subscribers are independent of each other.
- **Scalability:** The broker can handle many publishers and subscribers efficiently.
- **Flexibility:** Subscribers can dynamically subscribe or unsubscribe from topics.

3.7 MQTT Protocol

The Internet of Things (IoT) connects billions of devices — from smart bulbs and fitness trackers to industrial sensors and autonomous vehicles. One of the biggest challenges in IoT systems is enabling these devices to **communicate efficiently**, especially since many of them have limited computing power, memory, or battery

life. This is where **MQTT (Message Queuing Telemetry Transport)** comes in.

MQTT is a **lightweight, publish–subscribe messaging protocol** designed specifically for low-bandwidth, high-latency, or unreliable networks — exactly the kind of environment many IoT devices operate in.

It was originally developed by IBM in the late 1990s for monitoring oil pipelines via satellite links. Today, it has become one of the most popular protocols in IoT ecosystems.

3.7.1 Why MQTT for IoT?

IoT devices often:

- Run on small batteries.
- Use low-power processors.
- Connect via unstable wireless networks (Wi-Fi, Zigbee, 4G/5G, LoRaWAN, etc.).

Traditional communication protocols like HTTP are too heavy. MQTT solves this by being:

- **Lightweight**: Minimal header size (as low as 2 bytes).
- **Efficient**: Works well even with limited bandwidth.
- **Reliable**: Offers different Quality of Service (QoS) levels.
- **Scalable**: Can handle thousands of devices.

3.7.2 MQTT Architecture

At the heart of MQTT is the **broker-based architecture**.

- **Broker**: The central server that receives and distributes messages (e.g., Mosquitto, HiveMQ, EMQX).
- **Clients**: Devices or applications that connect to the broker. A client can **publish** messages or **subscribe** to receive them.

This is known as the **publish–subscribe model**.

Example

Imagine a smart home system:

- A temperature sensor publishes data `temperature = 28°C` to the topic: `home/livingroom/tem`
- A mobile app subscribes to the same topic.
- The MQTT broker ensures that whenever the sensor sends data, the mobile app receives it instantly.

3.7.3 MQTT Topics

Messages in MQTT are organized using **topics**, which work like folder paths.

- Example: `factory/machine1/temperature`

Clients can subscribe to specific topics or use wildcards (`+`, `#`) to receive multiple topics.

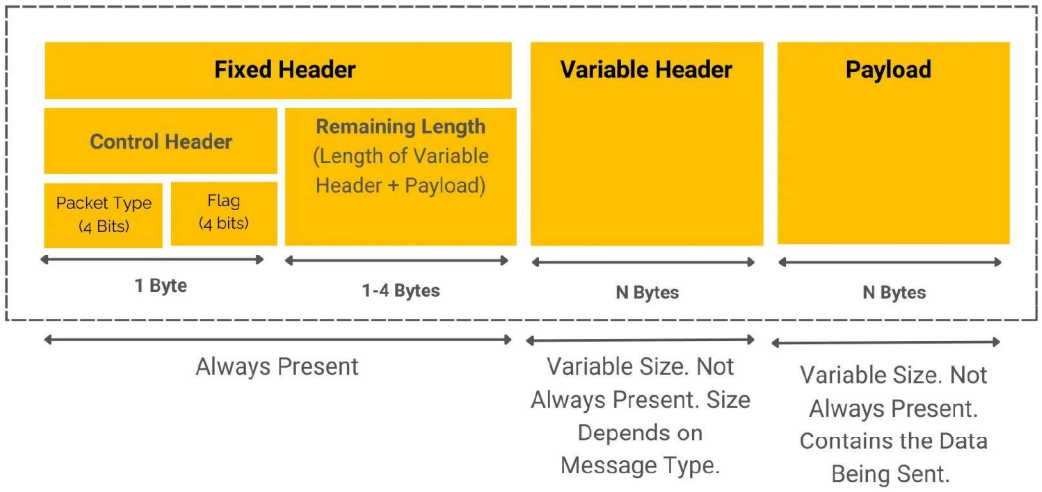


Figure 3.5: MQTT frame format

3.7.4 Quality of Service (QoS)

MQTT offers three QoS levels:

- **QoS 0 (At most once)**: "Fire and forget" — no guarantee of delivery.
- **QoS 1 (At least once)**: Message is delivered at least once (may be duplicated).
- **QoS 2 (Exactly once)**: Guaranteed single delivery (more overhead).

3.7.5 Key Features of MQTT

- **Retained Messages**: Broker stores last message on a topic for new subscribers.
- **Last Will and Testament (LWT)**: If a device disconnects unexpectedly, the broker notifies others.
- **Secure Communication**: Supports TLS/SSL encryption.

3.7.6 Real-World Applications

- Smart Homes: Controlling lights, ACs, or appliances.
- Healthcare: Patient monitoring devices.
- Industry 4.0: Machines reporting status for predictive maintenance.
- Automobiles: Vehicle-to-cloud communication.

3.7.7 Comparison with HTTP

Feature	HTTP	MQTT
Communication Model	Request–Response	Publish–Subscribe
Overhead	High (200–400 bytes headers)	Very low (2 bytes minimum)
Best for	Web Applications	IoT, M2M Communication
Scalability	Limited	High

MQTT has become the **default standard for IoT messaging** because of its efficiency, scalability, and reliability in constrained environments.

For students, the key takeaways are:

- Understand **publish–subscribe architecture**.
- Appreciate why **lightweight protocols** are needed in IoT.
- Experiment with an MQTT broker (like Mosquitto) and devices (Raspberry Pi, ESP32).

3.8 Push–Pull Communication Model

In IoT systems, devices and services need to exchange information efficiently. The **Push–Pull Communication Model** defines two complementary strategies for data transfer:

- **Push:** The sender initiates the data transmission to the receiver without an explicit request.
- **Pull:** The receiver requests data from the sender when needed.

Choosing between these strategies depends on application requirements such as latency, bandwidth, and energy consumption.

3.8.1 Push Communication

Definition In the **Push** model, the *producer* of data actively sends updates to the *consumer* whenever new data is available.

3.8.2 Example

A temperature sensor that sends readings to a cloud server every 10 seconds, regardless of whether the server has requested the data.

3.8.3 Advantages

- Low latency for time-sensitive updates.
- Efficient for continuous monitoring applications.

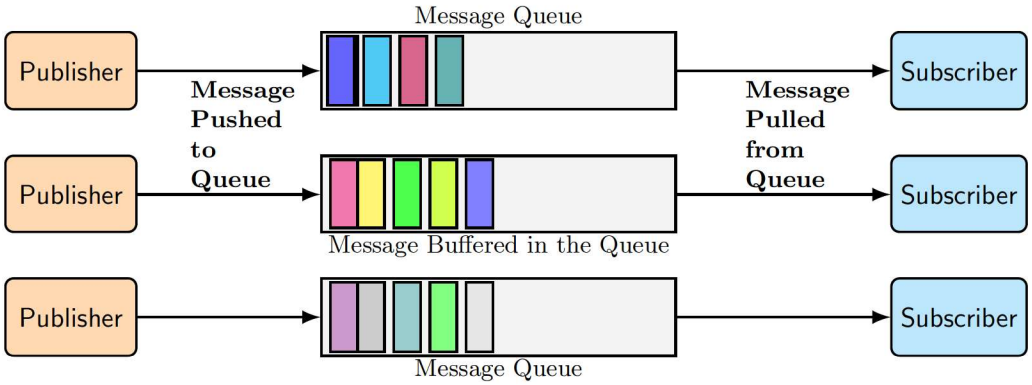


Figure 3.6: Push–Pull Communication Model

3.8.4 Disadvantages

- Can waste bandwidth if updates are sent more frequently than needed.
- May drain battery in energy-constrained devices.

3.8.5 Pull Communication

Definition In the **Pull** model, the *consumer* requests data from the *producer* when required.

3.8.6 Example

A mobile application querying a smart lock's status only when the user opens the app.

3.8.7 Advantages

- Bandwidth-efficient for infrequent data needs.
- Reduced unnecessary data transfers.

3.8.8 Disadvantages

- Increased latency, as data is retrieved only after a request.
- Higher request overhead for large-scale systems.

3.8.9 Push–Pull Hybrid Approach

Many IoT applications adopt a hybrid model, where critical updates are **pushed** in real-time, and less urgent data is **pulled** on demand.

- Example: In a smart agriculture system, soil moisture alerts are pushed immediately, while historical data is pulled during analysis.

3.8.10 Comparison Table

Feature	Push Model	Pull Model
Initiator	Sender	Receiver
Latency	Low	Higher
Bandwidth Usage	Potentially high	On-demand, lower
Energy Efficiency	Lower for constrained devices	Higher for constrained devices
Best Suited For	Real-time monitoring, alerts	On-demand queries, infrequent updates

3.8.11 Applications in IoT

- **Push:** Security alarms, live video streaming, industrial equipment monitoring.
- **Pull:** Periodic status checks, historical data retrieval, user-initiated control.

Understanding the Push–Pull communication model is crucial for designing efficient IoT systems. By selecting the right model—or a combination of both—engineers can optimize system performance, resource usage, and user experience.

3.9 Exclusive Pair Communication Model

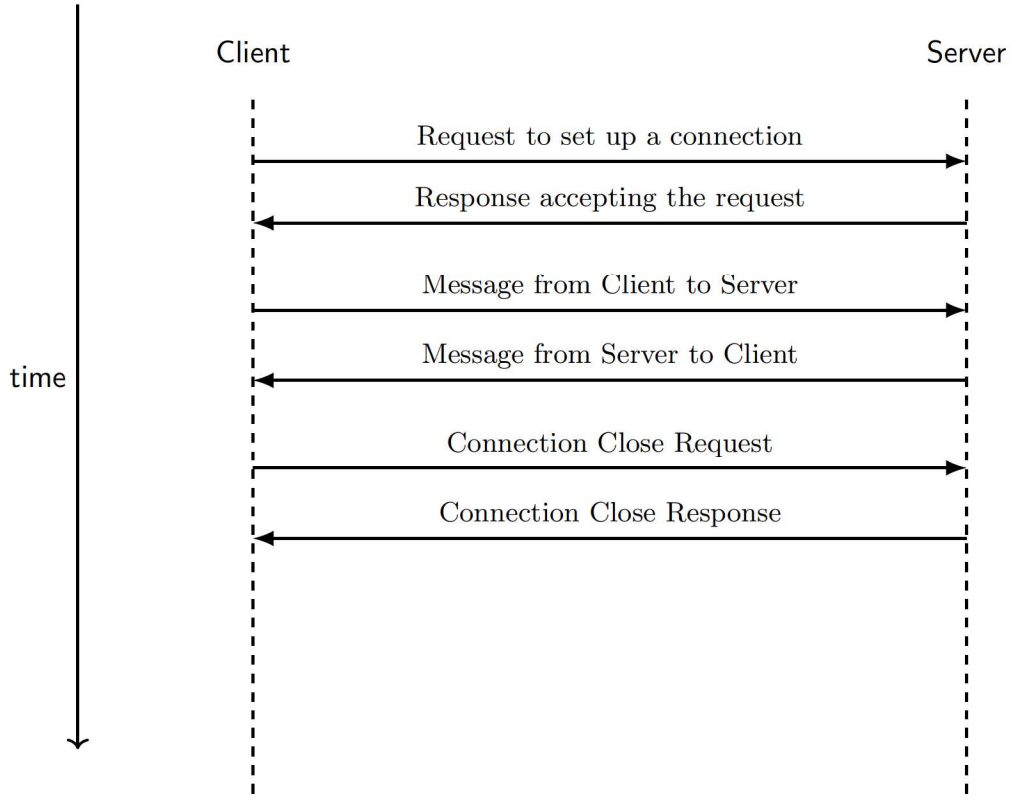


Figure 3.7: Exclusive Pair Communication Model

In Internet of Things (IoT) networks, various devices communicate with each other to exchange data for monitoring, control, or automation. Communication models define how data flows between devices. The **Exclusive Pair Communication Model** is one such approach where two devices communicate *only* with each other, without routing traffic through other nodes or sharing the channel with others.

Definition An **Exclusive Pair** refers to two nodes in a network that:

- Maintain a **dedicated** communication link.
- Exchange data directly (peer-to-peer).
- Restrict access so that no third device can intercept or join the link.

This approach ensures privacy, low latency, and predictable performance.

3.9.1 Working Principle

In this model:

1. **Pair Formation:** Two devices discover each other (via Bluetooth pairing, Zigbee bonding, or direct Wi-Fi connection).
2. **Link Establishment:** A dedicated communication channel is created using a secure protocol (e.g., AES encryption).
3. **Data Exchange:** The devices transmit and receive messages directly, without an intermediary server or router.
4. **Exclusive Maintenance:** Only the paired devices know the session keys and communication parameters.

3.9.2 Advantages

- **Security:** Reduced risk of interception since only two nodes are involved.
- **Low Latency:** Direct communication minimizes delay.
- **Predictable Bandwidth:** No contention with other devices.

3.9.3 Limitations

- **Scalability Issues:** Not suitable for large-scale many-to-many communication.
- **Pair Management Overhead:** Requires setup and maintenance of each pair.
- **Underutilization of Network Resources:** Dedicated links may remain idle.

3.9.4 IoT Applications

The exclusive pair model is often used in:

- **Wearable Devices:** Smartwatch paired exclusively with a smartphone.
- **Industrial Sensors:** Sensor–controller pairs in factory automation.
- **Medical Devices:** Glucose monitor paired with an insulin pump.

The Exclusive Pair Communication Model is valuable for applications requiring secure, direct, and reliable connections. While it offers strong performance and privacy benefits, engineers must weigh its scalability and resource usage limitations when designing IoT systems.

3.10 REST-based Communication APIs

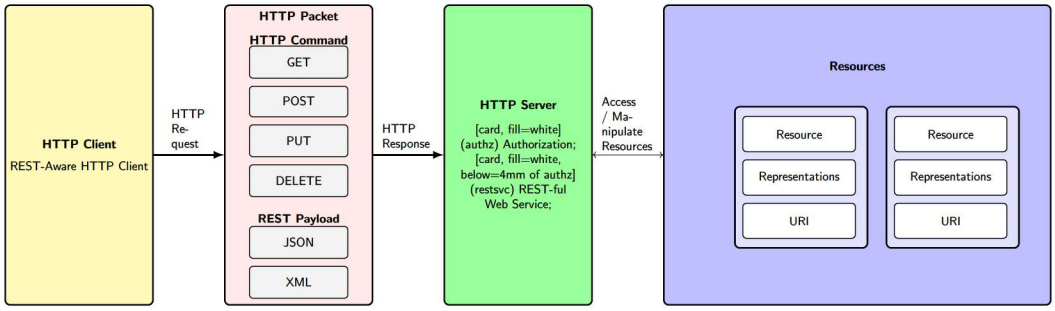


Figure 3.8: REST-based Communication APIs

The **Internet of Things (IoT)** connects billions of physical devices to the internet, enabling them to send, receive, and process data. To achieve seamless communication between these devices and backend services, *Application Programming Interfaces* (APIs) are required. Among different API architectures, REST has emerged as the most widely adopted due to its lightweight and stateless nature.

What is REST? REST stands for **Representational State Transfer**. It is an architectural style, not a protocol, for designing networked applications. REST is built on top of *HTTP* (Hypertext Transfer Protocol) and leverages its methods for communication.

The core principles of REST are:

1. **Statelessness:** Each request from a client to a server must contain all the necessary information; the server does not store client context.
2. **Uniform Interface:** A standard set of rules for interaction between clients and servers.
3. **Resource-based:** Information is represented as **resources** identified by URLs.
4. **Client-Server Separation:** Clients and servers are independent and evolve separately.
5. **Cacheable Responses:** Responses can be cached to improve performance.

3.10.1 REST in IoT Communication

In IoT, REST APIs are used for:

- Sending sensor data from devices to the cloud.
- Controlling actuators remotely.
- Querying device status.
- Integrating with third-party services.

For example, a temperature sensor can send readings to a cloud endpoint:

```
POST /temperature HTTP/1.1
Host: api.iotcloud.com
Content-Type: application/json

{
  "device_id": "sensor_101",
  "value": 27.3,
  "unit": "C"
}
```

3.10.2 HTTP Methods in REST APIs

REST uses standard HTTP methods:

- `GET` - Retrieve resource data.
- `POST` - Create new resource.
- `PUT` - Update an existing resource.
- `DELETE` - Remove a resource.
- `PATCH` - Partially update a resource.

3.10.3 Advantages of REST for IoT

1. **Lightweight:** Suitable for low-power devices.
2. **Scalable:** Supports large-scale deployments.
3. **Interoperable:** Works with different platforms and programming languages.
4. **Human-readable:** Data is usually exchanged in JSON or XML.

3.10.4 Limitations

- **Overhead:** HTTP headers add extra data for small payloads.
- **Latency:** Not always ideal for real-time communication.
- **Security:** Requires HTTPS and proper authentication.

3.10.5 REST vs. Other Communication Protocols

While REST is widely used, IoT systems may also use:

- **MQTT** - Lightweight publish/subscribe protocol for constrained devices.
- **CoAP** - Constrained Application Protocol optimized for low-power devices.
- **WebSockets** - Full-duplex communication for real-time updates.

REST is best suited for request-response interactions where devices communicate intermittently.

REST-based APIs provide a robust, scalable, and easy-to-implement communication mechanism for IoT systems. They enable developers to integrate devices, cloud platforms, and applications seamlessly using standard web technologies. However, for highly constrained or real-time applications, other protocols like MQTT or CoAP might be more suitable.

3.11 REST vs HTTP

Many students confuse REST and HTTP, but they are not the same. REST is an **architectural style**, whereas HTTP is a **protocol**. REST can use HTTP as its transport, but it is not limited to it (for example, CoAP in IoT is RESTful but does not use HTTP).

What is REST?

REST (Representational State Transfer) is an architectural style for designing distributed systems, introduced by Roy Fielding. It is based on resources, identified by URIs, and uses stateless communication. REST defines constraints such as:

- Client-server separation
- Statelessness
- Cacheability
- Layered system
- Uniform interface (methods like GET, POST, PUT, DELETE)

What is HTTP?

HTTP (Hypertext Transfer Protocol) is a communication protocol used for client-server communication. It defines how requests and responses are formatted and transmitted. HTTP provides methods such as GET, POST, PUT, and DELETE, and typically runs over TCP/IP.

Comparison

Feature	REST	HTTP
Type	Architectural style	Communication protocol
Defined by	Roy Fielding (2000 thesis)	IETF RFCs (e.g., RFC 2616)
Purpose	Guide to design scalable services	Define request/response communication
Methods	Suggests uniform interface (GET, POST, etc.)	Provides actual methods (GET, POST, etc.)
Transport Layer	Independent (can use HTTP, CoAP, etc.)	Runs over TCP/IP
Example	"Expose resources as URIs, stateless requests"	"Send a GET request to fetch a page"

Relationship

- REST is protocol-agnostic: it can use HTTP, CoAP, or other protocols.
- Most RESTful APIs today use HTTP as the transport layer.
- Therefore, REST and HTTP are related but not identical: REST is the design philosophy, while HTTP is the concrete implementation.

3.12 WebSocket-based Communication APIs

In Internet of Things (IoT) applications, efficient and real-time communication between devices, gateways, and servers is critical. Traditional HTTP-based APIs, while simple and widely supported, follow a *request–response* model that is not optimal for continuous, low-latency data exchange.

WebSocket-based Communication APIs overcome these limitations by establishing a persistent, bidirectional communication channel between clients and servers over a single TCP connection. This capability makes them ideal for IoT scenarios involving *live sensor data*, *real-time control commands*, and *continuous device monitoring*.

3.12.1 Basics of WebSockets

WebSockets are a protocol standardized by the IETF as RFC 6455. Unlike HTTP, which closes the connection after each request–response cycle, WebSockets keep the connection open, allowing both ends to send data at any time.

3.12.2 Key Features

- **Full-duplex Communication:** Data can flow simultaneously in both directions.
- **Low Latency:** Eliminates repeated HTTP handshakes and reduces overhead.
- **Persistent Connection:** Maintains an open channel for ongoing data exchange.

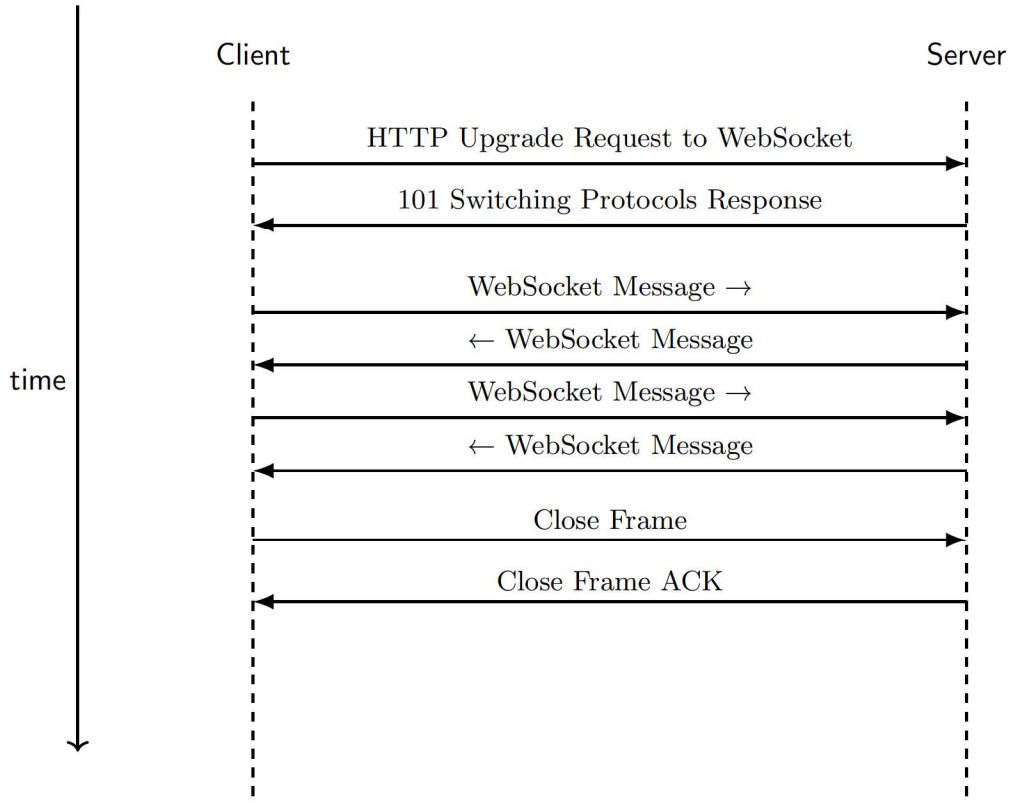


Figure 3.9: WebSocket-based Communication APIs

- **Binary and Text Support:** Supports both text (UTF-8) and binary data formats.

3.12.3 How WebSocket APIs Work

Connection Upgrade

A WebSocket connection typically starts as an HTTP request and then upgrades to the WebSocket protocol using the `Upgrade` and `Connection` headers:

```
GET /ws-endpoint HTTP/1.1
Host: example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

3.12.4 Message Exchange Model

Once established, both the server and client can:

1. Send messages without waiting for the other to request them.
2. Receive messages asynchronously.

3.12.5 Closing the Connection

Either side can send a *close frame* to gracefully terminate the connection.

3.12.6 Advantages for IoT

In IoT systems, WebSocket APIs offer several benefits:

- **Real-time Monitoring:** Sensor data (e.g., temperature, motion) can be pushed to the server instantly.
- **Efficient Control:** Commands (e.g., "turn on light") reach devices without delay.
- **Reduced Bandwidth Usage:** Avoids repetitive HTTP headers in frequent updates.
- **Scalability:** Suitable for systems with thousands of concurrent connections.

3.12.7 Implementation in IoT Systems

Client Side (Device or Web Application)

A JavaScript client can connect to a WebSocket API as follows:

```
const socket = new WebSocket('wss://iot-server.example/ws');

socket.onopen = () => {
    console.log('Connected to server');
    socket.send('Hello from IoT device');
};

socket.onmessage = (event) => {
    console.log('Message from server:', event.data);
};
```

Server Side

Servers can be implemented in Node.js, Python, or other languages. For example, in Python using `websockets`:

```
import asyncio
import websockets

async def handler(websocket):
    async for message in websocket:
        print(f"Received: {message}")
        await websocket.send("Acknowledged")

asyncio.run(websockets.serve(handler, "localhost", 8765))
```

3.12.8 Security Considerations

- Use `wss://` (WebSocket Secure) for encrypted communication.
- Authenticate clients before upgrading to a WebSocket connection.
- Implement message validation to prevent malicious commands.

WebSocket-based communication APIs are an essential tool for modern IoT applications requiring real-time, bidirectional communication. They improve performance, scalability, and user experience compared to traditional polling or long-polling HTTP methods.

As IoT systems grow in complexity and scale, mastery of WebSocket protocols and API design will be a key skill for engineers developing responsive, efficient, and secure device networks.

Communication Model	**Common Protocol(s)**	**Typical Usage in IoT**	**Connection Style**	**Advantages**
Request–Response (REST)	HTTP/ HTTPS	Device configuration, data retrieval from sensors, firmware updates	Stateless, short-lived connections	Simple, widely adopted, easy integration with web services, human-readable formats (JSON/XML)
Publish–Subscribe	MQTT, AMQP, DDS	Real-time telemetry from many sensors to multiple subscribers	Persistent broker-mediated connections	Scalable, decouples producers and consumers, efficient for low-bandwidth devices
WebSocket-based	WebSocket over TCP	Live dashboards, IoT chat applications, control systems needing instant feedback	Persistent, full-duplex connection	Low latency, bidirectional, efficient for continuous updates
CoAP (Constrained Application Protocol)	CoAP over UDP	Resource-constrained devices, low-power networks (e.g., smart meters, LPWAN)	Stateless or stateful, lightweight messaging	Very lightweight, supports multicast, designed for constrained environments
Exclusive Pair (TCP Socket)	TCP/IP	Direct device-to-device communication without intermediaries	Persistent connection between two endpoints	Predictable performance, reliable delivery, suitable for streaming

Table 3.1: Comparison of Common IoT Communication Models and Their Characteristics

3.13 Levels and Deployment Templates

The Internet of Things (IoT) is a transformative technology that connects physical devices to the digital world. For engineering students, understanding the *Levels of IoT architecture* and the corresponding *Deployment Templates* is crucial for designing, implementing, and scaling IoT solutions effectively. This article explains these concepts in detail, providing a clear bridge between theoretical models and practical deployment scenarios.

3.13.1 Levels of IoT Architecture

The **levels** in IoT architecture can be thought of as functional layers, where each level focuses on a particular set of responsibilities.

Perception Level

The *Perception Level*, also called the **Sensing Layer**, is responsible for collecting data from the physical world. This includes:

- Sensors (temperature, humidity, motion, etc.)
- Actuators (motors, relays, valves)
- RFID tags and readers

The main goal is accurate and reliable data acquisition.

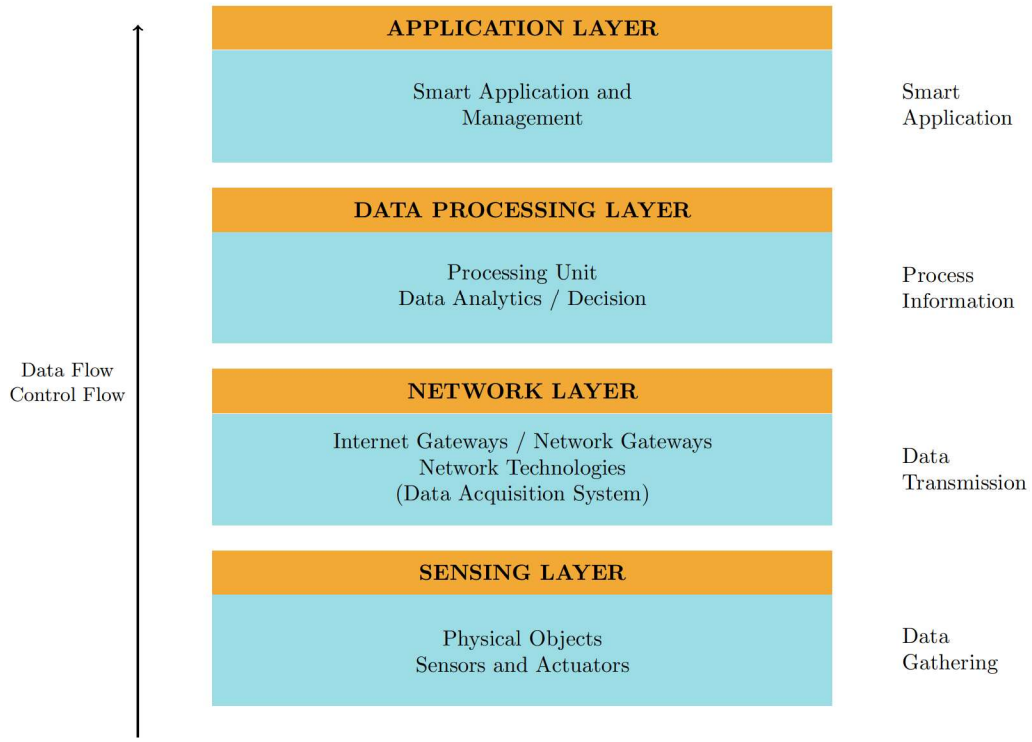


Figure 3.10: REST-based Communication APIs

Network Level

The *Network Level* transmits data from perception devices to processing units. It involves:

- Communication protocols: Wi-Fi, Zigbee, LoRaWAN, Bluetooth
- Network infrastructure: gateways, routers, base stations

This layer ensures **secure and efficient** data transfer.

Edge Level

The *Edge Level* introduces **edge computing**, where some processing happens close to the data source. This reduces latency, network congestion, and improves real-time decision-making.

Application Level

The *Application Level* provides services to end-users. Examples:

- Mobile or web dashboards
- Control systems for smart homes
- Predictive maintenance platforms

Business Level

The *Business Level* focuses on **data-driven decision-making**. It turns processed IoT data into actionable insights that improve efficiency, safety, or revenue.

3.13.2 Deployment Templates in IoT

A **Deployment Template** is a predefined architecture or configuration plan for deploying IoT solutions. These templates act like "recipes" to speed up development and maintain best practices.

Single-Tier Deployment

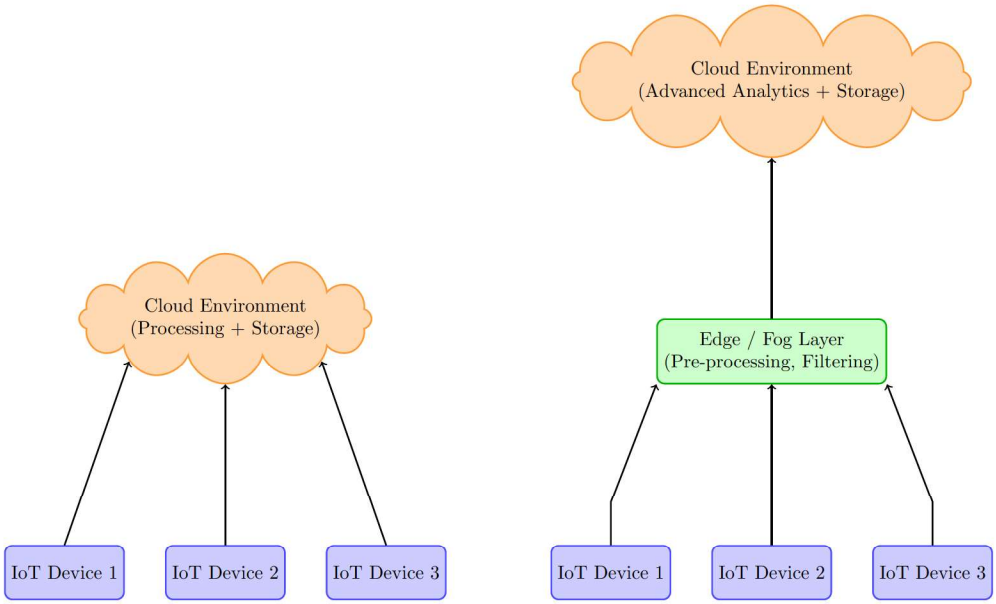
In a *single-tier* template:

- All processing occurs in a single environment, typically the cloud.
- Devices send raw data directly to the cloud.
- Suitable for small-scale prototypes or academic projects.

Two-Tier Deployment

In a *two-tier* template:

- Edge devices pre-process data.
- The cloud performs advanced analytics.
- Balances latency reduction and computational power.



(a) Single-Tier Deployment (b) Two-Tier Deployment

Figure 3.11: Single and Two-Tier Deployment

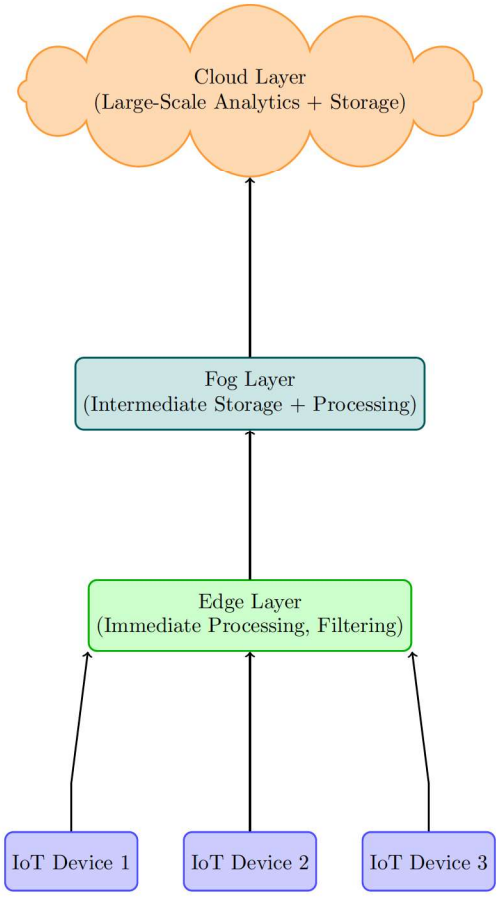


Figure 3.12: Three-Tier Deployment

Three-Tier Deployment

The *three-tier* template includes:

1. Edge layer for immediate processing.
2. Fog layer for intermediate storage and processing.
3. Cloud layer for large-scale analytics and storage.

This is widely used in industrial IoT (IIoT) deployments.

3.13.3 Example Mapping: Levels to Deployment Templates

The following mapping shows how architecture levels align with deployment templates:

- **Perception Level** → Device layer in all templates.
- **Network Level** → Connectivity backbone in all templates.
- **Edge Level** → Included in two-tier and three-tier deployments.
- **Application Level** → Cloud applications or dashboards.
- **Business Level** → Data analytics and visualization tools in the cloud.

For engineering students, understanding the *levels* of IoT architecture helps in systematically breaking down system design, while *deployment templates* provide practical guidance for implementation. The synergy between these two concepts ensures that IoT solutions are not only technically sound but also scalable, secure, and cost-effective.

3.14 IoT Communication Models and Their Relation to Communication Patterns

IoT communication does not rely on a single mechanism. Different application needs are fulfilled using different communication patterns such as *request response*, *publish–subscribe*, *push–pull*, and *exclusive pair communication*. Modern IoT platforms also expose *REST-based* and *WebSocket-based APIs* to integrate devices with cloud applications.

Below we explain each IoT communication model with examples of how these general patterns apply.

3.14.1 Device-to-Device (D2D)

In D2D communication, devices talk directly with each other using short-range protocols.

- **Exclusive Pair Communication:** Bluetooth Classic and NFC often use exclusive pair communication, where two devices form a direct link.
- **Push–Pull Model:** A sensor may push data to an actuator (e.g., ZigBee-based light switch and bulb).

3.14.2 Device-to-Gateway (D2G)

Devices communicate with a gateway, which aggregates and forwards data.

- **Request–Response:** Wi-Fi and 6LoWPAN devices may use HTTP to request services from the gateway.
- **Publish–Subscribe:** LoRaWAN end devices publish sensor data to the gateway, which then relays to the cloud.
- **Push–Pull:** The gateway can periodically pull data from devices or devices may push alerts to the gateway.

3.14.3 Device-to-Cloud (D2C)

Here, devices connect directly to cloud services over the Internet.

- **Request–Response:** REST APIs (HTTP/HTTPS) allow devices to send requests and get responses from the cloud.
- **Publish–Subscribe:** MQTT is the most common protocol, where devices publish data to a broker and subscribers receive updates.
- **WebSocket APIs:** Provide full-duplex, low-latency communication between device and cloud, useful in real-time apps.

3.14.4 Backend-to-Backend (B2B)

Cloud services and enterprise systems need to exchange IoT data.

- **Request–Response:** RESTful APIs are used to fetch or update IoT data across platforms.
- **Publish–Subscribe:** AMQP, DDS, or broker-to-broker MQTT connections distribute events across services.
- **Push–Pull:** Data warehouses may pull sensor data streams from IoT platforms for analytics.
- **IoT Communication APIs:** Many cloud platforms (AWS IoT, Azure IoT Hub, Google Cloud IoT) expose standard APIs for integration.

Mapping

IoT Model	Communication Pattern	Examples
Device-to-Device (D2D)	Exclusive Pair, Push–Pull	Bluetooth pairing, ZigBee sensor-to-actuator
Device-to-Gateway (D2G)	Request–Response, Publish–Subscribe, Push–Pull	HTTP over Wi-Fi, LoRaWAN uplink, gateway polling
Device-to-Cloud (D2C)	Request–Response, Publish–Subscribe, WebSockets	REST APIs, MQTT broker, WebSocket-based real-time apps
Backend-to-Backend (B2B)	Request–Response, Publish–Subscribe, Push–Pull, IoT APIs	RESTful APIs, MQTT bridging, DDS, AWS/Azure IoT APIs

3.15 Topologies

The Internet of Things (IoT) is not only about devices and sensors but also about how these devices *communicate* with each other and the cloud. A critical aspect of IoT is the **network topology**; the arrangement of nodes and their interconnections. Understanding topologies helps engineers design efficient, scalable, and fault-tolerant IoT systems.

This article explains different IoT network topologies, relates them to practical use cases, and justifies which communication protocols are most suitable for each scenario.

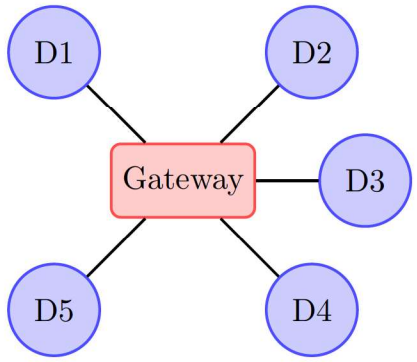
3.15.1 IoT Network Topologies

Star Topology

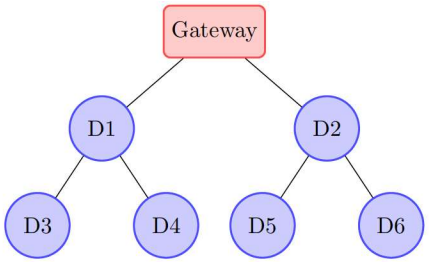
In a star topology, all devices (sensors or nodes) connect directly to a central hub or gateway. **Advantages:** Easy management, simple to deploy, low latency. **Disadvantages:** Single point of failure at the hub.

Use Case: Smart homes (e.g., sensors connected to a home router or hub).

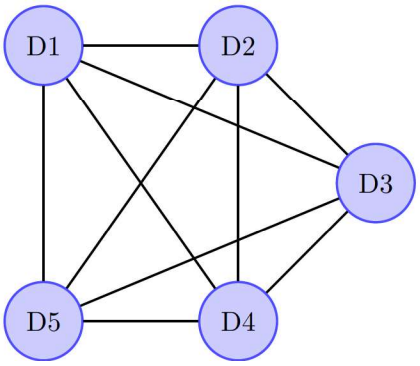
Protocols: ZigBee, Z-Wave, Wi-Fi. These are chosen because they support low-power operation (ZigBee/Z-Wave) or high throughput (Wi-Fi) for consumer IoT devices.



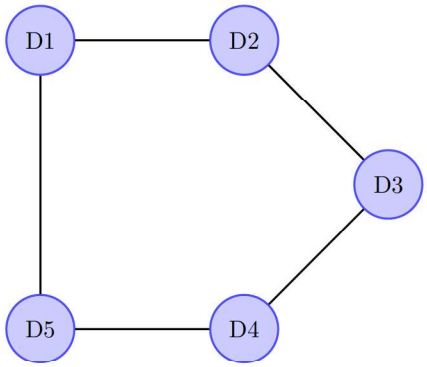
(a) Star Topology



(b) Tree Topology



(c) Mesh Topology



(d) Ring Topology

Figure 3.13: IoT Network Topologies

Mesh Topology

In a mesh topology, each node can communicate with multiple neighboring nodes, creating multiple redundant paths. **Advantages:** High reliability, self-healing, scalable. **Disadvantages:** Higher complexity and power consumption.

Use Case: Industrial IoT and Smart Cities, where reliability and coverage are critical.

Protocols: ZigBee, Thread, Bluetooth Mesh. These allow devices to forward data across the network, ensuring robust communication even if some nodes fail.

Tree (Clustered) Topology

This topology extends the star by connecting multiple stars in a hierarchical manner (clusters connected via parent nodes). **Advantages:** Scalable, structured routing. **Disadvantages:** Failure of higher-level nodes affects the subtree.

Use Case: Precision agriculture, where multiple clusters of soil sensors send data to regional gateways.

Protocols: 6LoWPAN with RPL (Routing Protocol for Low-power and Lossy Networks). This protocol supports hierarchical routing and IPv6 integration.

3.15.2 Bus Topology

All devices share a single communication line (medium). **Advantages:** Simple, low cost. **Disadvantages:** Collisions, difficult fault isolation.

Use Case: Legacy systems in factory automation.

Protocols: CAN (Controller Area Network), Modbus. These are used in industrial IoT for reliable, deterministic communication over a shared bus.

3.15.3 Hybrid Topology

Hybrid topologies combine two or more structures, such as star-mesh or star-bus. **Advantages:** Flexibility, optimized for mixed environments. **Disadvantages:** Complexity and higher cost.

Use Case: Smart grids, where consumer devices form star networks but substations are interconnected using mesh or tree.

Protocols: LoRaWAN (star-of-stars), cellular IoT (NB-IoT, LTE-M). These balance scalability with wide-area coverage.

3.15.4 Mapping Topologies to Communication Models

Different communication models in IoT (Device-to-Device, Device-to-Gateway, Device-to-Cloud, and Backend-to-Backend) align with specific topologies:

- **Device-to-Device (D2D):** Works best in *mesh* (Bluetooth Mesh, ZigBee) for local peer-to-peer communication.
- **Device-to-Gateway (D2G):** Common in *star* topologies, where devices send data to a local hub (e.g., Wi-Fi smart home appliances).
- **Device-to-Cloud (D2C):** Often seen in *hybrid topologies*, where gateways forward data to the cloud via MQTT, CoAP, or HTTPS.

- **Backend-to-Backend (B2B):** Usually implemented in the cloud layer, independent of topology, using REST APIs, AMQP, or MQTT bridging.

For IoT engineers, selecting the right topology and communication protocol is a design decision influenced by power, range, scalability, and reliability requirements.

- *Star* is ideal for simple, low-power home applications.
- *Mesh* suits critical industrial and city-scale systems.
- *Tree* balances scalability in clustered environments.
- *Bus* remains relevant in industrial legacy systems.
- *Hybrid* topologies enable large-scale, heterogeneous deployments like smart grids.

By mapping protocols to these topologies and use cases, engineers can build efficient, reliable, and scalable IoT systems.

Chapter 4

Sensors and Actuator

The Internet of Things (IoT) connects physical devices to the internet, enabling data collection and communication. Sensors play a crucial role in IoT systems, providing the raw data necessary for applications across industries such as healthcare, agriculture, and smart cities. This report explores the types of sensors used in IoT, their applications, and future trends.

The Internet of Things (IoT) represents a network of interconnected devices that communicate and share data. Sensors form the backbone of IoT systems, enabling real-time monitoring, data collection, and control.

Sensor

A sensor is a device that detects changes in physical conditions, such as temperature, pressure, or light intensity, and produces an output corresponding to the change.

Examples:

- A thermistor detects temperature changes.
- A photodiode senses light intensity.

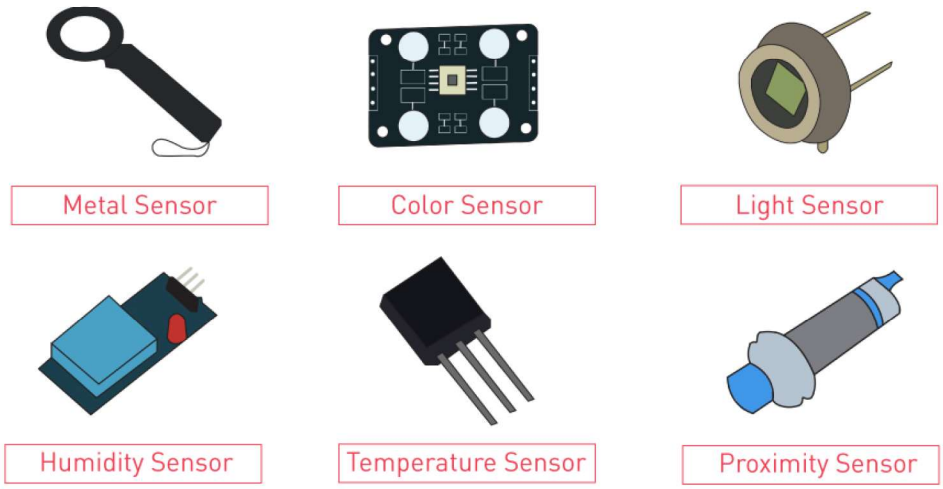


Figure 4.1: Sensors in IoT

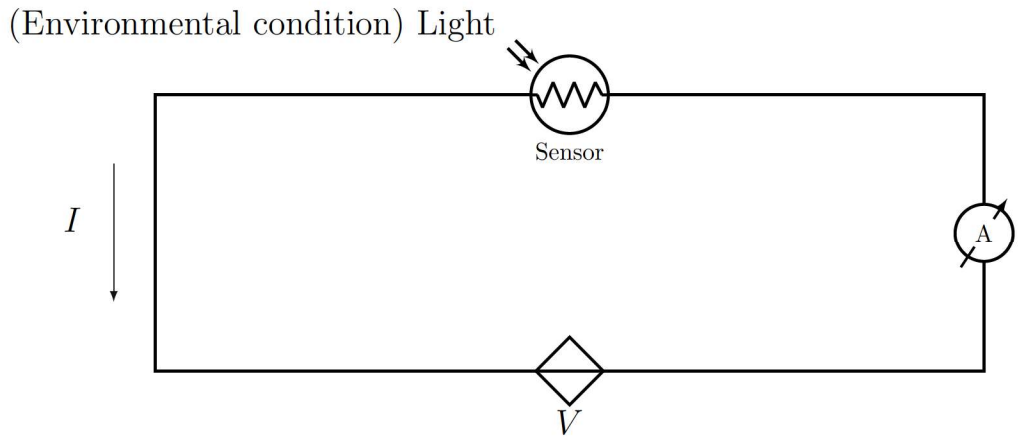


Figure 4.2: Sensors measuring environment condition light, where V is voltage bias, A is measurement of current

4.1 Transducer

A transducer is a device that converts one form of energy into another. It plays a vital role in sensing, measuring, and control systems by enabling the transfer and transformation of energy. This document provides an overview of transducers, their types, working principles, and applications in various fields.

In a wider term, any device that transforms energy from one form to another is referred to as a transducer. Any kind of energy, including thermal, electrical, mechanical, and so on, may be converted in this way. As an example of a transducer, consider a loudspeaker that transforms electrical energy into sound energy.

A transducer is an essential device used in converting physical quantities such as temperature, pressure, or light into electrical signals or vice versa. It is widely utilized in fields such as instrumentation, automation, and communication.

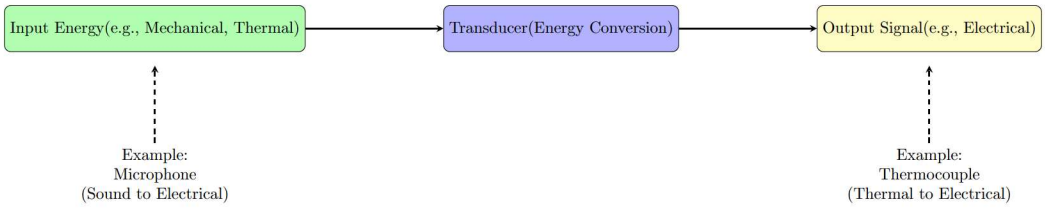


Figure 4.3: Transducers

4.2 Types of Transducers

Transducers can be classified based on various criteria:

4.2.1 Based on Energy Conversion

- **Active Transducers**: Require no external power source (e.g., piezoelectric sensors).

- **Passive Transducers**: Require external power for operation (e.g., thermistors).

4.2.2 Based on Output Signal

- **Analog Transducers**: Provide continuous output (e.g., strain gauges).
- **Digital Transducers**: Provide discrete output (e.g., encoders).

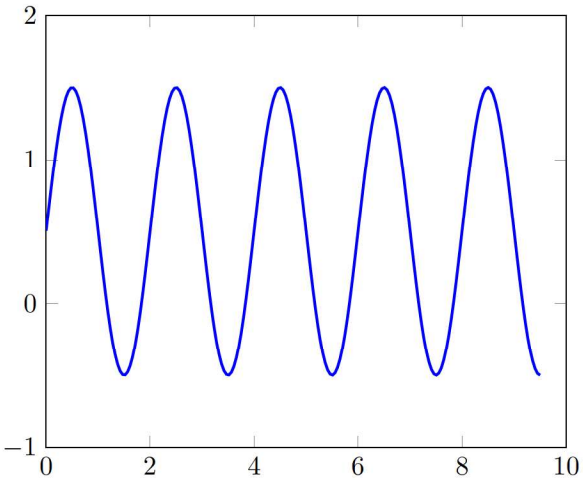


Figure 4.4: Analog Signal

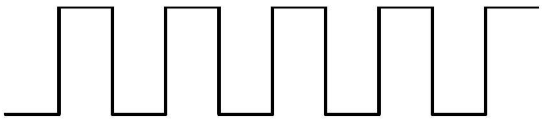


Figure 4.5: Digital Signal

4.2.3 Examples of Transducers

- **Temperature Transducers**: Thermocouples, thermistors
- **Pressure Transducers**: Bourdon tubes, diaphragms
- **Displacement Transducers**: LVDTs, potentiometers

4.3 Working Principle

The operation of a transducer involves the following steps:

1. A physical quantity is detected by the transducer.
2. The transducer converts the quantity into an intermediate signal, such as an electrical or mechanical signal.
3. The signal is amplified, processed, or directly utilized by the system for measurement, control, or display.

4.4 Applications

Transducers are used in various fields, including:

- **Medical Applications**: Ultrasound transducers, ECG sensors
- **Industrial Automation**: Pressure and flow measurement in pipelines
- **Consumer Electronics**: Microphones and speakers
- **Aerospace**: Altimeters and navigation systems

4.5 Advantages and Limitations

4.5.1 Advantages

- High accuracy and sensitivity
- Versatility in converting different energy forms
- Wide range of applications

4.5.2 Limitations

- May require additional signal conditioning
- Some transducers are sensitive to environmental factors

A sensor and a transducer are critical components in measurement and control systems. While they are often used interchangeably, they serve different purposes in the system. This document highlights their distinctions and similarities.

Comparison Table

The following table summarizes the key differences between a sensor and a transducer:

4.6 Sensor

A sensor is a device that detects changes in physical conditions, such as temperature, pressure, or light intensity, and produces an output corresponding to the change.

Examples:

- A thermistor detects temperature changes.
- A photodiode senses light intensity.

4.6.1 Role of Sensors

Sensors are devices that detect and measure physical or environmental changes, converting them into signals for interpretation.

Aspect	Sensor	Transducer
Definition	A device that detects physical or environmental changes.	A device that converts one form of energy into another.
Primary Role	Measures a physical quantity.	Converts the measured quantity into an electrical signal or another energy form.
Output Type	Non-electrical (e.g., displacement, temperature).	Electrical or another energy form.
Examples	Thermistor, photodiode, pressure gauge.	Thermocouple, LVDT, microphone.
Functionality	Part of a transducer system in some cases.	Can include sensors as a component.
Applications	Used for detecting physical changes.	Used in systems requiring signal processing or energy conversion.

Table 4.1: Comparison between a Sensor and a Transducer.

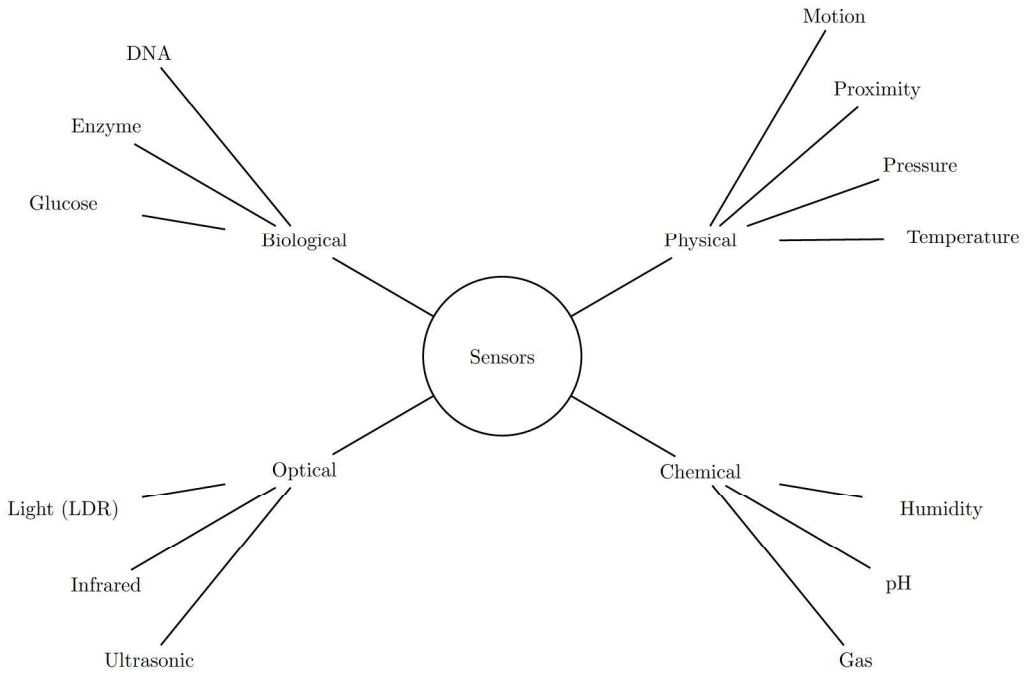


Figure 4.6: Types of Sensors in IoT

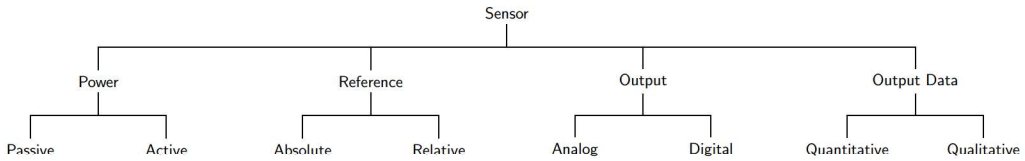


Figure 4.7: Types of Sensors based on different property

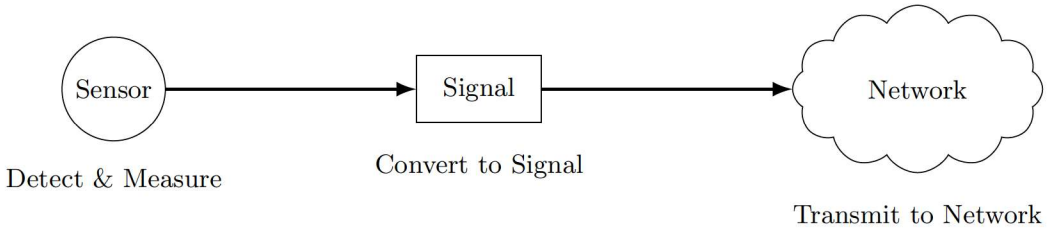


Figure 4.8: Sensors are devices that detect and measure physical or environmental changes, converting them into signals for interpretation.

4.7 Good Features of Sensors

1. **Accuracy and Precision** The degree to which a measured value resembles the actual or accepted reference value is referred to as accuracy. What matters is "hitting the target." Measurements from an accurate sensor are generally quite near to the true value.

   Contrarily, precision is associated with measurement repeatability. It explains the variance that appears while measuring the same component or parameter more than once. Even when a precise sensor is far from the true value, it can reliably produce the same measurement, hence it may not always be accurate.

   It is impossible to exaggerate the significance of these concepts, particularly in engineering applications. Measurements taken with a sensor that is accurate but not exact can have a wide range of accuracy. On the other hand, a highly accurate yet imprecise sensor consistently produces measurements that are off. Both precision and accuracy are desired qualities in many applications to guarantee accurate and consistent measurements.

   - Ability to measure values close to the true or standard value.
   - Low measurement error.

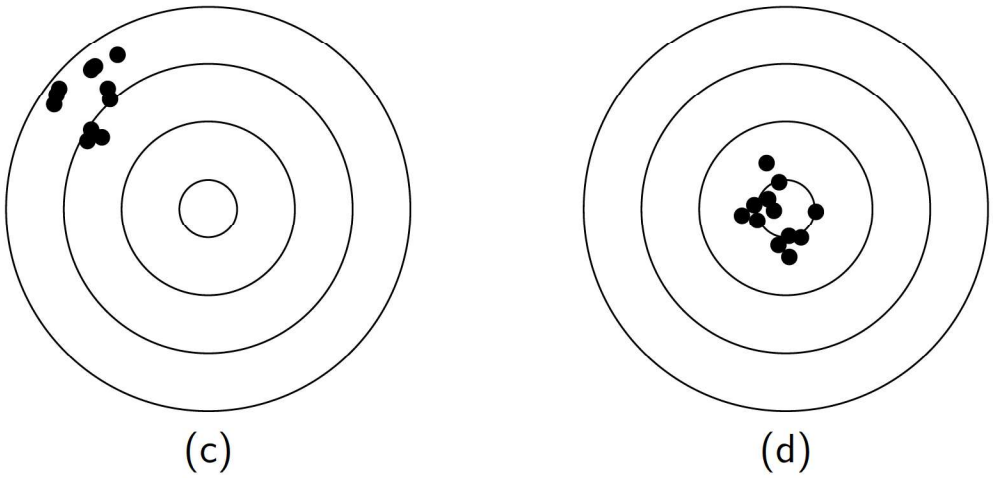


Figure 4.9: Precision vs Accuracy

The following variables may affect a sensor's accuracy and precision:

- Calibration
- Environmental Influences
- Wear and Tear
- Noise
- Quality of Components

2. **Sensitivity** One of the essential properties of sensors is sensitivity, which is the capacity to identify even the smallest variations in the measurand, which is the stimulus or the amount, item, or quality that is meant to be measured. High sensitivity sensors are extremely useful in situations where accuracy and detail are crucial since they can detect even minute deviations. Think of medical imaging systems, for example, where a diagnosis can be changed by a fraction of a degree difference. On the other hand, in systems like a home thermostat, where only large changes are concerning, a low-sensitivity sensor could be sufficient.

   A sensor's sensitivity can be affected by several things:

   - Material Properties
   - Sensor Geometry
   - External Interference
   - System Calibration
   - Amplification
   - Linearity and Nonlinearity

3. **Resolution** Resolution, to put it simply, is the smallest change in the measured quantity that a sensor can consistently identify and show. For example, a thermometer with a resolution of 0.1°C may detect temperature changes as little as 0.1°C.

   Comprehending the resolution is crucial for several reasons:

   - Clarity of Information
   - Enhanced Control
   - Distinguishing Noise from Data

   Despite being a standalone characteristic, a sensor's resolution is closely related to the system's overall performance:

   - Informed Decision-Making
   - Data Interpretation
   - System Response
   - Error Minimization

4. **Range** In sensor technology, range is a spectrum of values of a physical characteristic that a sensor can measure with reliability, from minimum to maximum. It basically defines the limits of a sensor's functioning range. A temperature sensor, for example, may have a range of -50°C to 150°C, meaning that it can measure and report temperature differences within these limits.

   There are multiple factors that affect a sensor's range:

- Material Characteristics
- Design and Construction
- Calibration
- Environmental Conditions
- Electrical Characteristics

5. **Linearity and Nonlinearity** If the output of a sensor varies proportionately with a change in the measured amount within its designated range, the sensor is said to be linear. When the sensor's output is plotted against the measured quantity, the resultant graphic would be a straight line. The sensor's sensitivity should ideally increase with the steepness of this line (slope).

   Nonlinearity shows how a sensor's actual response deviates from its predicted linear response. It represents the maximum deviation between the actual and linear responses of the sensor over its operating range and is commonly stated as a percentage of full scale (%FS).

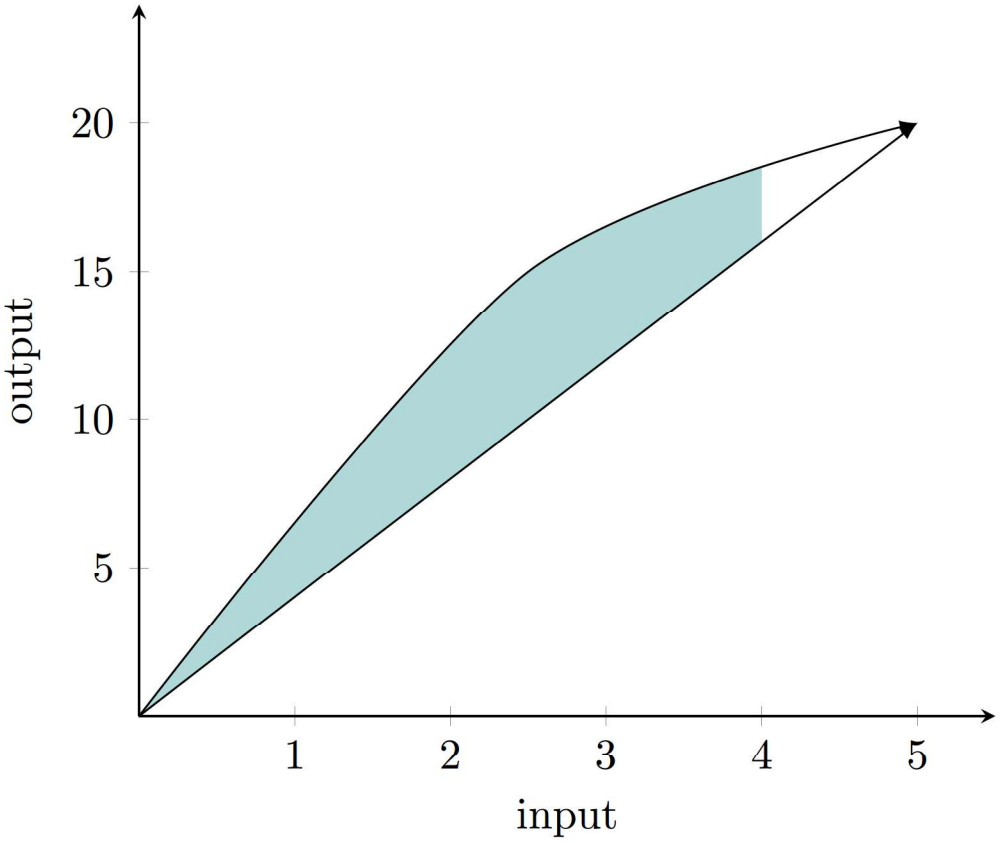


Figure 4.10: Precision vs Accuracy

- Consistent output proportional to the input across its range.

6. **Stability, Drift and Repeatability** It is crucial to guarantee steady and reliable results in the ever-changing field of sensor technology. Two essential ideas related to long-term sensor performance measurements are stability and drift.

Stability: Consistent performance over time and environmental changes.

Drift describes an unwanted shift in sensor output when the input doesn't change. Instead of occurring suddenly, this variation is usually noticed over long periods of time and might be attributed to age, environmental variables, or intrinsic properties of the sensor materials.

Sensor drift can be caused by various factors, such as:

- Aging of Components
- Thermal Effects
- Mechanical Stress
- Chemical Contamination
- Electronic Noise

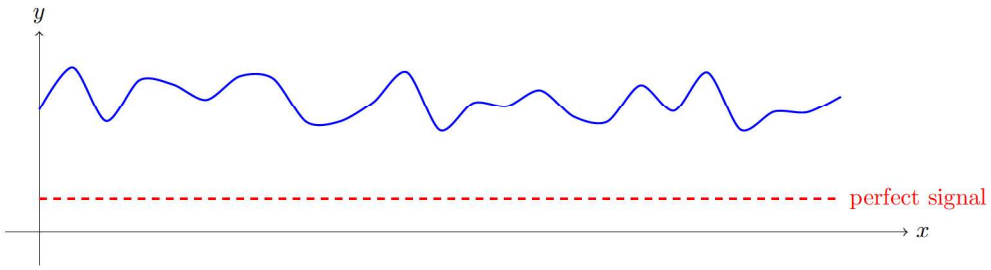


Figure 4.11: Precision vs Accuracy

Methods to Minimize and Compensate for Drift

The first step in reducing and offsetting drift is identifying its underlying cause. The following are some tactics:

- Calibration
- Temperature Compensation
- Material Choice
- Differential Measurement
- Feedback Systems
- Environmental Control
- Advanced Signal Processing
- Shielding and Protection

Repeatability: Ability to produce the same output for repeated measurements under identical conditions.

7. **Response Time** Response time describes how long it takes a sensor to respond to a change in the parameter it is tracking and generate an output in line with that change. The time interval that passes between applying an input stimulus (change) and waiting for the sensor's output to settle at a suitable value that reflects the change is what it basically is. A common criterion for

many sensors would be the amount of time it takes for the output to change by one step in input before reaching, say, 90

There are several instances that demonstrate the importance of reaction time:

- Safety
- Process Control
- Consumer Electronics

The reaction time of a sensor can be affected by several variables:

- Sensor Design
- Material Properties
- External Environment
- Signal Processing

8. **Hysteresis** When a particular input level is approached, hysteresis in the context of sensors is the difference in output that occurs from first increasing the input from a lower value and then decreasing it from a higher one. Consider the following situation: if you gradually increase pressure on a pressure sensor from 0 to 100 units, the reading may vary when you decrease pressure back from 100 units to the same intermediate value. Hysteresis is the term used to describe this difference or lag in response. A number of things can cause

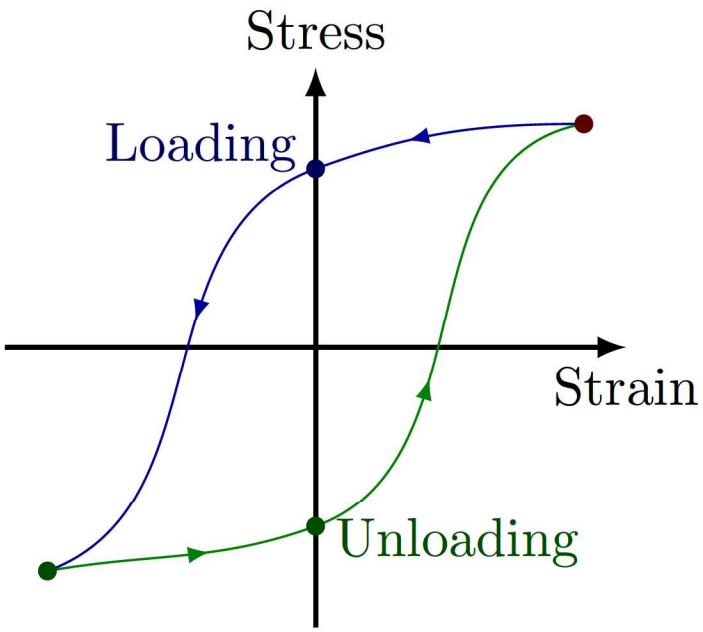


Figure 4.12: Hysteresis

sensors to hysteresis:

- Material Memory
- Mechanical Friction or Stiction
- Thermal Effects

9. **Robustness and Durability**
    - Withstand harsh environments like extreme temperatures, pressure, humidity, or vibrations.
10. **Low Power Consumption**
    - Energy-efficient, especially for battery-operated and IoT applications.
11. **Compact Size and Lightweight**
    - Facilitates integration into various systems without increasing the system's bulk.
12. **Ease of Integration**
    - Compatibility with other components and systems, such as microcontrollers or networks.
13. **Low Cost and Availability**
    - Affordable and readily available in the market.
14. **Noise Resistance**
    - Ability to filter out interference and provide clear, accurate data.
15. **Calibration Simplicity**
    - Easy to calibrate for maintaining accuracy over time.

    Importance of System Calibration and Signal Interpretation
    - Calibration Simplicity
    - Predictable Response
    - Accuracy
    - Signal Processing
    - System Feedback
16. **Signal Quality**
    - Generates strong, high-quality signals that are easy to process.
17. **Application-Specific Features**
    - Depending on the application, a sensor may need specialized features, such as:
        - Wireless communication.
        - Data encryption for security.
        - Ability to detect multiple parameters simultaneously.

4.8 Sensor Resolution

Sensor Resolution refers to the smallest measurable change in the input parameter that a sensor can reliably detect and represent in its output. It determines the sensor's ability to provide detailed measurements.

Key Aspects of Sensor Resolution

1. **Smallest Increment:** The resolution is often expressed as the smallest increment of change in the measured variable that the sensor can detect. Example: A temperature sensor with a resolution of 0.1°C can detect temperature changes as small as 0.1°C.

2. **Measurement Units:** The resolution is typically specified in the units of the measured parameter (e.g., volts, meters, degrees Celsius).

3. **Relation to Accuracy:** A high-resolution sensor is not necessarily accurate. Accuracy measures the closeness to the true value, while resolution measures the sensor's ability to distinguish between small changes.

4. **Digital and Analog Resolution:**

   - **Digital Sensors:** Resolution is often determined by the bit depth of the analog-to-digital converter (ADC). For example, a 12-bit ADC divides the sensor's range into $2^{12} = 4096$ discrete levels.
   - **Analog Sensors:** Resolution is determined by the sensor's design and inherent noise.

5. **Impact of Noise:** Noise in the sensor's signal can limit its effective resolution, as small changes may be obscured.

Importance of Sensor Resolution

- **High Resolution:** Essential for applications requiring precise measurements, such as medical devices, scientific research, and robotics. Example: A blood pressure monitor must have fine resolution to detect slight variations in pressure.
- **Low Resolution:** Sufficient for applications where fine detail is unnecessary, such as detecting room occupancy.

Example of Resolution Calculation

If a temperature sensor has a range of 0°C to 100°C and uses a 10-bit ADC:

$$\text{Resolution} = \frac{\text{Range}}{\text{Number of Levels}} = \frac{100}{2^{10}} = \frac{100}{1024} \approx 0.098\,^{\circ}\text{C}$$

4.9 Types of Sensors in IoT

The sensors can also be classified based on the following criteria: Primary Input quantity (Measurand), Transduction principles, (Using physical and chemical effects), Material and Technology, Property, and Application.

1. Based on Activity

- **Passive Sensors:**
  - Do not require an external power source to operate.

  - Rely on the energy from the physical or environmental changes they measure.
  - **Example:** Thermocouples, Photodiodes (in photovoltaic mode).

- **Active Sensors:**
  - Require an external power source to operate.
  - Use the supplied energy to measure the parameter and generate a signal.
  - **Example:** Ultrasonic Sensors, Infrared Sensors.

2. Based on Output

- **Analog Sensors:**
  - Produce a continuous signal proportional to the measured quantity.
  - **Example:** Thermistors, Light-Dependent Resistors (LDRs).

- **Digital Sensors:**
  - Generate discrete signals (binary or digital values).
  - **Example:** Proximity Sensors, Digital Thermometers.

3. Based on Data Type

- **Scalar Sensors:**
  - Measure a single value or magnitude of a physical parameter.
  - **Example:** Temperature Sensors, Pressure Sensors.

- **Vector/Multimedia Sensors:**
  - Measure data that has direction or involves multiple dimensions.
  - **Example:** Accelerometers (vector), Cameras (multimedia).

4.10 Sensor Types based on Application

4.10.1 Temperature Sensors

Temperature sensors are used to measure and monitor temperature in applications like smart homes, agriculture, and industrial processes.

4.10.2 Motion Sensors

Motion sensors detect movement and are widely used in security systems and smart lighting.

4.10.3 Humidity Sensors

Humidity sensors measure the moisture content in the air and find applications in weather monitoring and HVAC systems.

4.10.4 Gas Sensors

Gas sensors detect the presence of specific gases and are critical for air quality monitoring and safety systems.

4.10.5 Other Sensors

- Light Sensors - Pressure Sensors - Proximity Sensors

4.11 Applications of Sensors in IoT

4.11.1 Smart Cities

Sensors enable traffic monitoring, waste management, and energy-efficient lighting in smart cities.

4.11.2 Healthcare

Wearable sensors monitor patient vitals, providing real-time health data for better care.

4.11.3 Agriculture

IoT sensors monitor soil moisture, temperature, and crop health to optimize farming practices.

4.11.4 Industrial IoT

Sensors in factories enable predictive maintenance, automation, and monitoring of equipment.

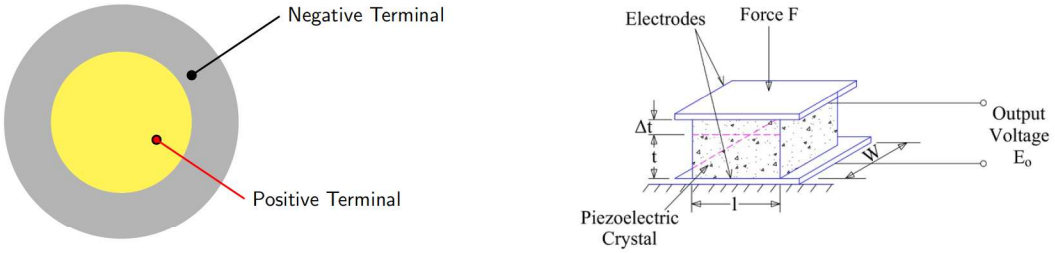
4.12 Sensor Outputs

Voltage output is one of the most widely used methods for sensor information sharing. Because it often reacts proportionately to changes in the measurand, or the quantity being measured, this type of output is adaptable to a broad range of sensors.

4.12.1 Voltage output

Piezoelectric Sensors

These work on the basis of the piezoelectric phenomenon, which states that when a material experiences stress, it will generate a voltage. Piezoelectric materials, such as quartz or certain ceramics, produce a voltage in reaction to mechanical strain and are frequently employed in accelerometers, microphones, and specific kinds of pressure sensors. Their ability to sense dynamic changes, particularly fast movements or vibrations, is what makes them perfect.



(a) Piezoelectric Sensor (b) Piezoelectric Sensor transducer.

Figure 4.13: Piezoelectric Sensor

Photovoltaic Sensors

Voltage is produced by these sensors from light energy. When photovoltaic or solar cells are exposed to light, they produce a voltage. These cells are commonly built of semiconductor materials like silicon. The light intensity determines how much voltage is generated. When there is sufficient light, they are commonly utilized as power sources or in light intensity measurement.

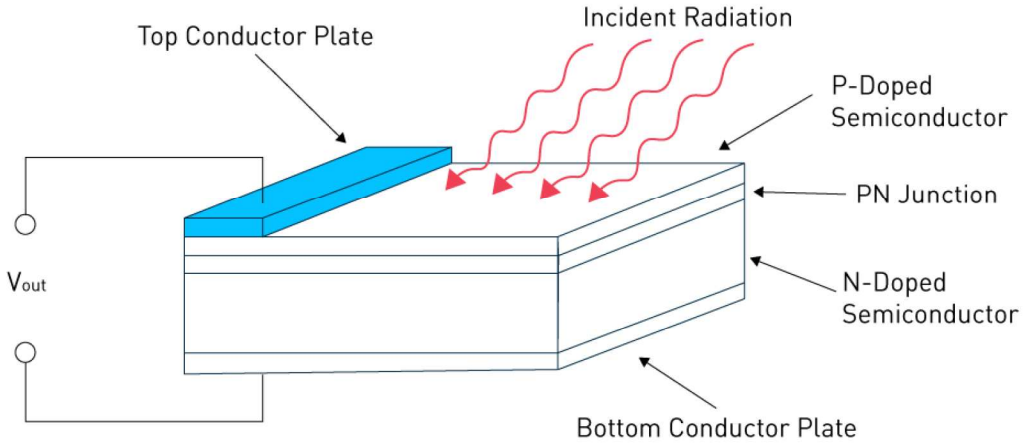


Figure 4.14: Photovoltaic Sensor

Other voltage-based sensors exist, each with a different basis of operation; examples include thermocouples, which produce voltage in response to temperature variations, and Hall effect sensors, which provide voltage outputs when magnetic fields are present.

4.12.2 Current Output

Photodiode in Photoconductive Mode

Semiconductor devices called photodiodes, generate electricity in response to light as depicted in Figure. 4.2. A reverse-biased photodiode is one that receives a voltage provided in the opposite direction of the forward-bias direction of the diode when it is operating in the photoconductive mode. In these circumstances, electron-hole pairs are produced when photons strike the diode. These carriers are accelerated by

the reverse bias, producing a photocurrent that is proportionate to the amount of light. The photoconductive mode can respond more quickly than the photovoltaic mode, but it may also produce more noise.

Transistor-Based Sensors

These make use of transistors' intrinsic characteristics. A transistor's current can vary under specific circumstances, and this variation might be related to the amount being measured.

Electrochemical Sensors

These sensors, which are frequently used to detect gases, generate electricity when a particular chemical reaction occurs at an electrode.

Consideration of load

A sensor's current output is usually meant to pass through a load resistance that is known already, over which a voltage corresponding to the current will grow. The sensor's performance can be greatly impacted by the type of load.

- Load Resistance: The selection of load resistance is essential. Non-linearity, a narrower signal range, or saturation might result from a mismatch. For example, a voltage range of 1V to 5V will be produced by a 4mA to 20 mA current loop sensor intended to function across a 250-ohm load. This voltage range will vary if a different load resistance is used.
- Grounding and Shielding: In order to prevent ground loops, which might inject undesired noise into the current signal, proper grounding is necessary. Shielding may also be required to shield the transmission from outside electromagnetic interference.
- Cable Length: One benefit of current-output sensors, such as the 4-20 mA system, is that the signal resists voltage changes across the cable over extended cable lengths. On the other hand, very long cables may introduce capacitance, which may reduce the bandwidth of the signal.

4.12.3 Variable-Resistance Sensors

Thermistors

Thermistors are semiconductor-based, temperature-sensitive resistors. The temperature greatly affects their resistance. Positive temperature coefficient (PTC) and negative temperature coefficient (NTC) are the two main types of thermistors. As the temperature rises, PTC thermistors exhibit an increase in resistance, whereas NTC thermistors exhibit a drop in resistance. Due to its great sensitivity, thermistors are frequently employed in applications that call for accurate temperature readings within a constrained temperature range.

Resistance Temperature Detectors (RTDs)

Precision temperature sensors, or RTDs, are composed of a metallic element. The element's metal of choice—platinum, nickel, or copper—usually depends on the needs

of the application, and more especially, on the sensor input of the instrument. RTDs are highly accurate and predictable over a broad temperature range since their resistance rises linearly with temperature. Since long-term stability is essential in industrial and scientific applications, resistor-thermistors (RTDs) are utilized instead of thermistors because of their almost linear resistance-temperature connection. Ad-

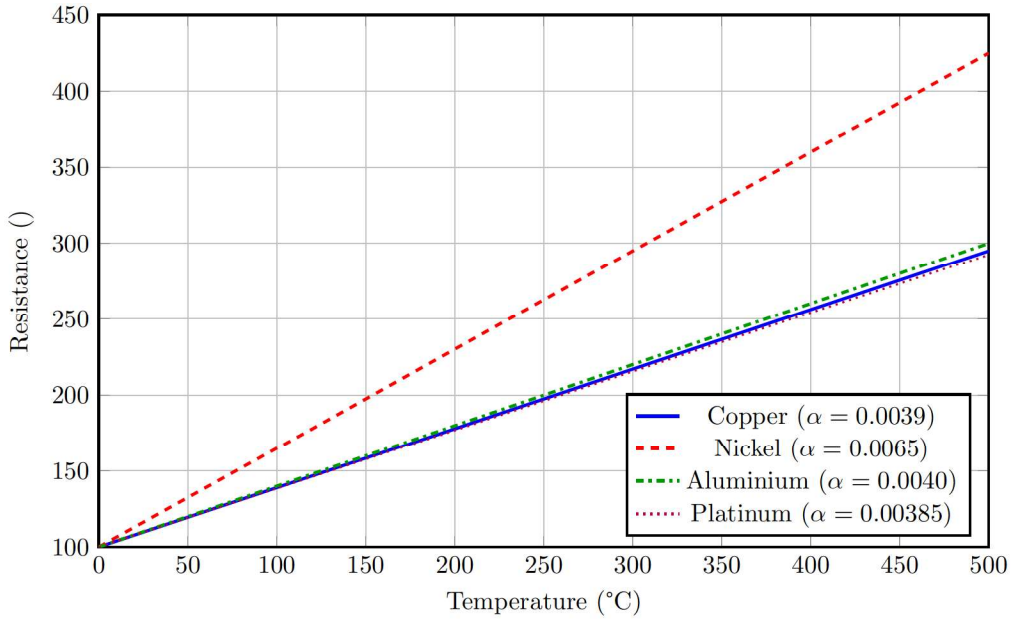


Figure 4.15: Resistance curves for Copper, Nickel, Aluminium, Platinum. linear approximation using the temperature coefficient of resistance (TCR)

ditional sensors with changing resistance are photoresistors, whose resistance varies with light intensity, and strain gauges, whose resistance changes under mechanical force.

Impact of Excitation Voltage/Current depends on

- Self-Heating
- Signal-to-Noise Ratio (SNR)
- Linearity
- Bridge Circuits

4.12.4 Frequency based Sensors

Oscillatory Sensors

Either an external input can cause these sensors to oscillate, or they can oscillate naturally. One example of an external influence that can alter the natural resonant frequency is temperature or pressure in quartz crystal resonators. The amount of the physical change can be estimated by measuring the frequency shift brought about by the external stimulus.

Frequency Modulation Techniques

It's possible that sensors don't always produce a frequency output natively. In these situations, the primary output of the sensor (such as voltage or resistance) can be modulated in a variety of ways to produce a frequency signal. A popular technique is frequency modulation (FM), which modifies a carrier wave's frequency based on the output of the sensor. Electronic circuits often employed for this purpose include voltage-to-frequency converters and voltage controller oscillators (VCOs) in phase-locked loops (PLLs).

Frequency-Based Measurement Advantages

- High Immunity to Offset and Drift
- Robustness to Noise
- High Resolution and Precision

4.13 Working principals

Table 4.2: Overview of Mechanical Sensors (from MPS Scholar)

Sensor Type	Working Principle	Materials	Applications
Accelerometers	Inertia-based displacement detection using a seismic mass; measured via capacitive or piezoresistive MEMS structures	Silicon (MEMS), polysilicon resistors, micromachined proof masses	Mobile devices (orientation), vehicle dynamics (airbags, stability), seismic sensing
Piezoelectric Sensors	Direct conversion of mechanical stress to voltage	Quartz, Lead Zirconate Titanate (PZT), Gallium Orthophosphate	Vibration monitoring, microphones, ultrasonic sensors
Load Cells / Strain Gauges	Measure deformation-induced resistance change (piezoresistive effect) in strain gauges bonded to structures	Constantan (Cu–Ni alloy), silicon strain gauges, foil resistors	Force/load measurement in industrial machinery, weighing scales, structural monitoring

4.14 Challenges and Future Trends

4.14.1 Challenges

- Power consumption - Security and privacy - Interoperability

4.14.2 Future Trends

- Integration with AI and Machine Learning - Development of self-powered sensors - Expansion of 5G networks for better connectivity

Sensors are integral to the success of IoT systems, providing the data required for real-time decision-making and automation. As technology advances, sensors will become smarter, more efficient, and more ubiquitous.

Sensor Type	Working Principle	Materials	Advantages / Limitations	Applications
Thermocouples	Voltage generated via Seebeck effect between two dissimilar metals	Common types: Type J (Iron-Constantan), Type K (Chromel-Alumel), Type T (Copper-Constantan)	*Advantages:* Wide temperature range, fast response, low cost. *Limitations:* Lower accuracy, drift over time, needs cold-junction compensation	Industrial process control, HVAC, automotive exhaust, scientific research, power plants
RTDs (Resistance Temperature Detectors)	Resistance changes nearly linearly with temperature	Pure platinum (e.g., Pt100)	*Advantages:* High accuracy, excellent linearity, long-term stability. *Limitations:* Slower response compared to thermocouples or thermistors	HVAC, automotive, chemical, aerospace, pharmaceutical, food & beverage, industrial process control
Thermistors	Resistance varies (exponentially) with temperature; NTC decreases resistance, PTC increases	Metal-oxide ceramics (e.g., manganese, nickel, cobalt, copper oxides)	*Advantages:* Very high sensitivity, low cost, fast response. *Limitations:* Nonlinear response, limited temperature range	Consumer electronics, medical devices, battery packs, thermostats, over-temperature protection
Infrared (IR) Sensors / Pyrometers	Detect infrared radiation emitted from object surface and convert to temperature based on emissivity	Thermopiles or pyroelectric detectors	*Advantages:* Non-contact measurement. *Limitations:* Affected by emissivity, requires clear line of sight	Night vision, thermal imaging, industrial temperature monitoring, body temperature screening, predictive maintenance

Table 4.3: Overview of Temperature Sensor Types

Sensor Type	Key Features	Working Principle	Material	Applications / Remarks
PN Photodiode	Basic p–n junction; moderate sensitivity and speed.	Incident photons create electron–hole pairs in depletion region; photocurrent proportional to light intensity.	Silicon (Si) most common; sometimes InGaAs for IR.	Used in low-light detection, simple optical receivers.
PIN Photodiode	Intrinsic layer reduces capacitance, improves speed.	Wide intrinsic region improves carrier separation efficiency and response time.	Silicon (Si) for visible; InGaAs for near-IR (0.9–1.7 µm).	High-frequency communication, fast response sensors.
Avalanche Photodiode (APD)	Very high sensitivity due to avalanche gain.	High reverse bias causes impact ionization; carriers generate secondary carriers → internal amplification.	Silicon for visible; InGaAs for telecom wavelengths.	Long-distance fiber communication, LIDAR, weak-signal detection.
Intensity-Modulated Fiber-Optic Sensor	Robust, simple, insensitive to EMI.	Light intensity modulated by bending, microbending, or reflection losses in optical fiber.	Glass or plastic optical fibers.	Strain, displacement, vibration, pressure measurement.
Phase-Modulated Fiber-Optic Sensor	Extremely precise and sensitive.	Environmental changes alter refractive index → phase shift in transmitted light is detected.	Single-mode silica optical fibers.	High-resolution sensing of strain, temperature, seismic activity.

Table 4.4: Types of Optical Sensors: Features, Principles, and Materials (from MPS-Scholar)

Sensor Type	Working Principle	Material	Applications
Glass Electrode (pH Sensor)	Electrochemical potential across a glass membrane varies with hydrogen-ion activity, measured against a reference electrode (via Nernst equation)	Specialized pH-sensitive glass	Medical diagnostics, food & beverage manufacturing, environmental monitoring
ISFET (Ion-Sensitive FET pH Sensor)	Ion-sensitive membrane replaces the MOSFET gate, converting ion concentration changes into variations in source–drain current	MOSFET-based semiconductor with ion-sensitive gate oxide	Compact and rugged pH sensing for diagnostics, process control, and environmental monitoring
Catalytic Gas Sensor (Pellistor)	Combustible gas burns on a heated sensing element, changing temperature and resistance in proportion to gas concentration	Catalytic bead (metal oxide on a heated bead)	Detection of flammable gases (e.g., methane, propane) in industrial safety applications
Infrared Gas Sensor	Gas absorbs specific IR wavelengths; concentration determined by measuring attenuation of transmitted IR light	IR light source and detector, optical filters or absorption cells	Detection of gases like CO_2 and hydrocarbons for environmental monitoring and safety systems
Semiconductor Gas Sensor	Adsorbed gas molecules alter conductivity (resistance) of semiconductor surface material	Metal oxide semiconductors (e.g., SnO_2, ZnO)	VOC detection, air quality monitoring, industrial gas sensing
Electrochemical Gas Sensor (e.g., CO/O_2)	Target gas undergoes electrochemical reaction on electrode, generating current proportional to concentration	Electrochemical cell (working, counter, and reference electrodes in electrolyte)	Measuring gases like CO and O_2 in medical, environmental, and safety applications
hline **Enzyme-Based Biosensor**	Enzyme reacts with analyte; biochemical change generates an electrical signal (e.g., via amperometry)	Immobilized enzymes (e.g., glucose oxidase) on electrode surface	Glucose monitoring, clinical diagnostics, biochemical assays
Antibody-Based Biosensor	Specific antigen–antibody binding yields a detectable signal (optical or electrochemical), often using tagged antibodies	Antibodies immobilized on transducer surface (e.g., electrochemical, optical)	Detection of biomarkers, pathogens, and environmental toxins in diagnostics and environmental monitoring

Table 4.5: Chemical and Biological Sensors: Principles, Materials, and Applications (from MPScholar)

Sensor Type	Working Principle	Materials	Applications
Piezoresistive Pressure Sensor	Pressure deforms a diaphragm, causing stress in piezoresistors; change in resistance is measured.	Metal or semiconductor strain gauge; often implemented using MEMS technology on silicon.	Industrial process control and monitoring; medical equipment (e.g. blood pressure measurement), tire pressure monitoring.
Capacitive Pressure Sensor	Pressure alters diaphragm displacement, changing capacitance between plates; the change is measured.	Capacitor structures with dielectric materials, typically MEMS-fabricated diaphragms.	Medical devices, industrial process control, atmospheric pressure monitoring (due to high sensitivity and stability).
Resonant Pressure Sensor	Pressure shifts the resonant frequency of a mechanical resonator (e.g. silicon); frequency change is detected.	Mechanical resonators—often silicon-based MEMS structures that may incorporate piezoelectric or capacitive elements.	High-precision applications: barometric altitude measurements, aerospace and aviation systems (e.g. cabin/fuel/hydraulic pressure), critical industrial process control, research laboratories.

Table 4.6: Types of Pressure Sensors: Principles, Materials, and Applications

4.15 Actuators in IoT

The Internet of Things (IoT) is an ecosystem of interconnected devices that sense, compute, and act upon data from the physical world. While sensors capture environmental information, actuators are responsible for executing physical actions based on control signals. Actuators are the bridge between the digital and physical realms of IoT, allowing systems to manipulate their environment intelligently. This article presents a comprehensive exploration of actuators in IoT targeted at undergraduate Computer Science engineering students. Covering definitions, classifications, working principles, and real-world applications, the discussion spans traditional actuators such as motors and solenoids to modern micro-electromechanical systems (MEMS) and shape-memory alloys (SMAs). Mathematical models, equations, and case studies are included to provide a rigorous understanding. By the end, students will have a thorough foundation in how actuators empower IoT applications in domains like smart homes, healthcare, agriculture, industry, and smart cities.

The Internet of Things (IoT) is often described as a network of "smart" devices capable of sensing, computing, and acting. A typical IoT system can be visualized as a three-part pipeline:

1. **Sensing** — Data collection from the environment using sensors.
2. **Processing** — Analyzing the data and making decisions using microcontrollers, embedded systems, or cloud computing.
3. **Actuation** — Executing physical actions in the real world through actuators.

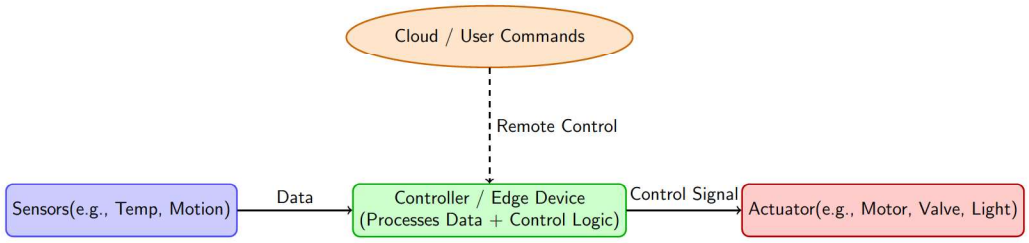


Figure 4.16: A typical IoT system

Among these, actuation is arguably the most critical step that gives IoT its *impact*. For instance, a smart thermostat would be useless if it could only measure room temperature but not regulate heating or cooling. Actuators provide this "muscle" to IoT systems, ensuring they can act, not just observe.

For undergraduate students in Computer Science, understanding actuators may seem outside the traditional curriculum. However, as IoT becomes integral to areas like smart infrastructure, healthcare technology, and autonomous systems, even software engineers must develop an appreciation of the hardware layer. This chapter aims to bridge that gap by offering a detailed yet accessible discussion.

4.16 What is an Actuator?

4.16.1 Definition

An actuator is a transducer that converts energy into mechanical or physical output. Formally, an actuator can be modeled as a function:

$$A : (u, E) \mapsto y$$

where:

- u = control input (voltage, current, pneumatic pressure, etc.)
- E = source of energy (electrical, hydraulic, pneumatic, thermal)
- y = output displacement, velocity, force, torque, or other physical effect

4.16.2 Actuators in IoT Context

In an IoT system, actuators:

- Receive digital commands from microcontrollers.
- Convert those commands into analog signals (e.g., PWM signals for motors).
- Drive mechanical movement or other forms of action.

For example, in a smart irrigation system:

- Moisture sensors detect dryness in soil.
- The controller decides irrigation is needed.
- An actuator opens the irrigation valve.

This loop completes the cyber-physical integration that defines IoT.

4.17 Classification of Actuators

4.17.1 By Energy Source

1. **Electrical Actuators**: most common, driven by current or voltage.
2. **Hydraulic Actuators**: use incompressible fluids for very high force.
3. **Pneumatic Actuators**: use compressed air for rapid, medium-force actuation.
4. **Thermal Actuators**: rely on heat-induced expansion.
5. **Magnetic Actuators**: employ magnetic fields to generate displacement.

4.17.2 By Motion

Actuator types by motion are broadly classified into three categories:

- **Linear**: Linear Actuators, which move objects in a straight line. Linear actuators are used for pushing, pulling, and lifting in applications like robotics and manufacturing.
- **Rotary**: Rotary Actuators, which produce circular or rotational movement. Rotary actuators are used for turning components, controlling valves, and rotating machinery.

- **Oscillatory**: a device that produces back-and-forth or repetitive linear motion, unlike a rotary actuator that creates spinning motion or a linear actuator that produces straight-line motion. These actuators are widely used in consumer products and entertainment systems to generate vibrations or repetitive movements in devices like electric toothbrushes, mobile phone vibrators, and vibrating screens. Used in electric toothbrushes, linear-motor shavers, and mobile phone vibrators.

4.17.3 By Control

- **Open-loop**
  - *How it Works*: The actuator receives a control signal and performs its function without verifying the actual output.
  - *Example:* A simple electric motor that, when turned on, rotates at a certain speed, but the system doesn't check if the desired speed was actually reached.
  - *Characteristics:* Simpler and less complex, but less accurate, as they do not account for external factors that may affect performance.
- **Closed-loop**
  - *How it Works:* The system includes a feedback mechanism, such as a sensor, that monitors the actuator's output and compares it to the desired input. The actuator then adjusts its output to correct any discrepancies.
  - *Example:* A thermostat in a home uses feedback from a temperature sensor to turn the heater on or off, ensuring the desired room temperature is maintained.
  - *Characteristics:* More accurate and reliable, capable of self-correction, and can compensate for disturbances.
- **Smart actuators**
  - *How it Works:* These actuators incorporate built-in intelligence, such as microprocessors and sensors, to autonomously manage their operation. They often execute closed-loop control functions and may have advanced features for diagnostics, communication, and self-optimization.
  - *Example:* An intelligent valve actuator that can automatically adjust its position based on real-time data from the process, communicating its status and allowing for remote adjustments.
  - *Characteristics:* High precision, advanced functionality, and can operate independently of a central controller, simplifying system design and improving efficiency.

4.17.4 By Application Domain

- Industrial IoT
- Smart homes
- Medical IoT
- Robotics

4.18 Detailed Types of Actuators in IoT

This section expands into detail on the major categories of actuators relevant to IoT.

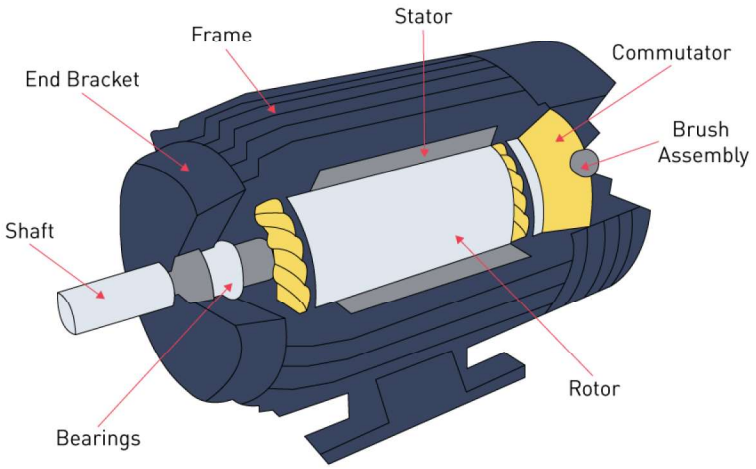


Figure 4.17: DC Motor

4.18.1 DC Motors

Working Principle: Based on Lorentz force. When a conductor carrying current I is placed in a magnetic field B, it experiences a force:

$$F = B \cdot I \cdot L \cdot \sin(\theta)$$

where L is conductor length and θ the angle between conductor and field.

In a motor, this results in torque:

$$T = k \cdot \Phi \cdot I_a$$

where Φ is flux, I_a is armature current, and k a motor constant.

Applications in IoT:

- Fans in smart HVAC systems
- Wheels in IoT-enabled robots
- Pumps in water management systems

Torque per Unit Volume

A crucial parameter in the design and selection of electric motors, is torque density, or torque per unit volume. It measures how efficiently a motor generates torque relative to its size. Applications where space and weight are limited, such as in aerospace, robotics, and portable electronic devices, place particular importance on this concept.

Dividing the torque produced by a motor by its volume determines torque density. Typically, it's denoted in Newton-meters per cubic meter (Nm/m^3). A high torque density implies that a motor can generate considerable torque relative to its compact size. In design scenarios where space efficiency is paramount, motors with high torque density are often favored.

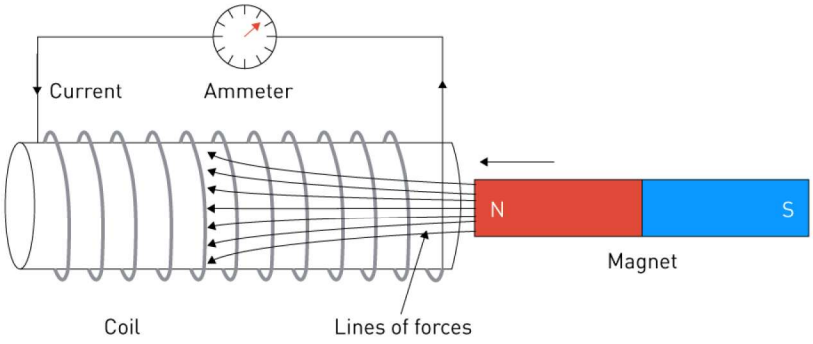


Figure 4.18: Faraday's Law

4.18.2 Stepper Motors

Principle: Rotate in discrete steps based on electromagnetic coils energized sequentially. Step angle θ is given by:

$$\theta = \frac{360^\circ}{N_s}$$

where N_s is number of steps per revolution.

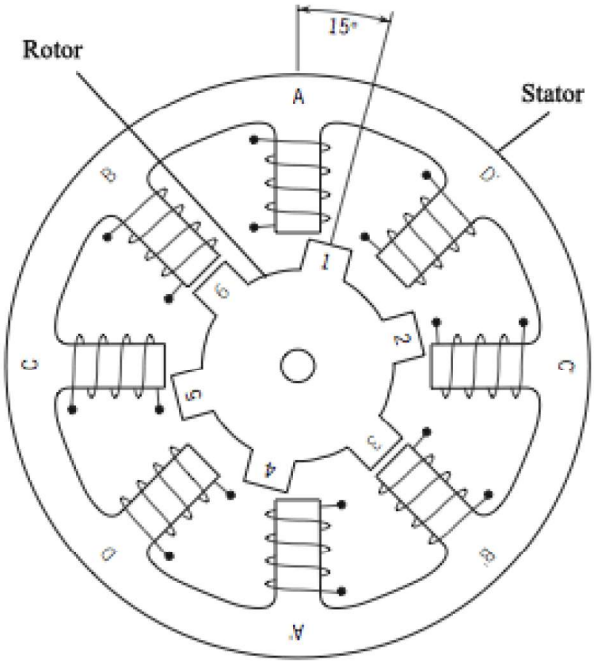


Figure 4.19: Stepper Motor

Applications:

- 3D printers
- CNC machines
- Camera rotation in smart surveillance systems

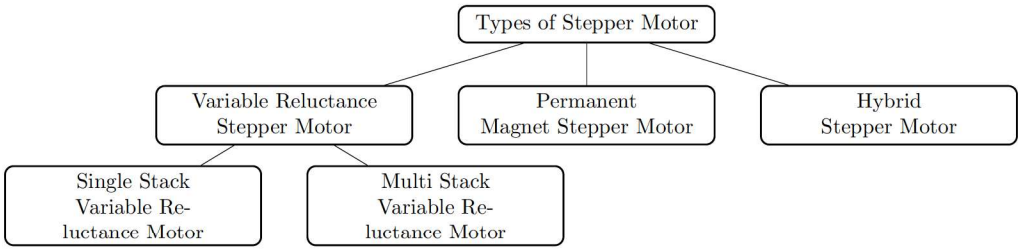


Figure 4.20: Types of Stepper Motors

4.18.3 Servo Motors

A servo motor is essentially a DC motor with feedback (usually via a potentiometer or encoder). Position is controlled by PWM signals. **Equation:**

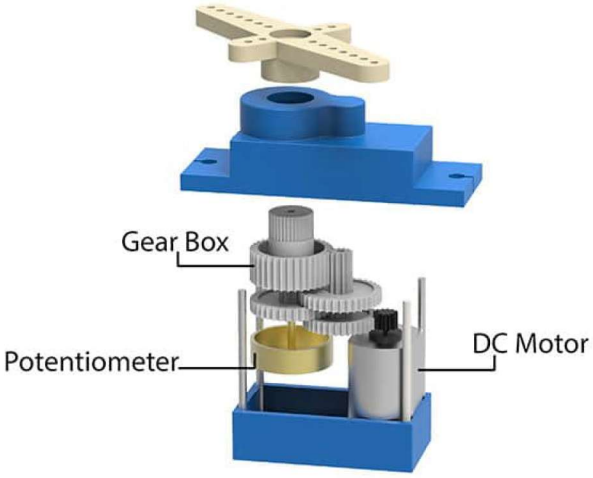


Figure 4.21: Servo-Motor

$$\theta(t) = K_p \cdot e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

where $e(t)$ is error, representing closed-loop PID control.

Applications:

- Robotic joints
- Antenna positioning
- Drone control surfaces

4.18.4 Solenoids

A solenoid produces linear motion using an electromagnetic coil. **Force Equation:**

$$F = \frac{N^2 I^2 \mu_0 A}{2g^2}$$

where N = turns, I = current, A = area, g = air gap.

Applications:

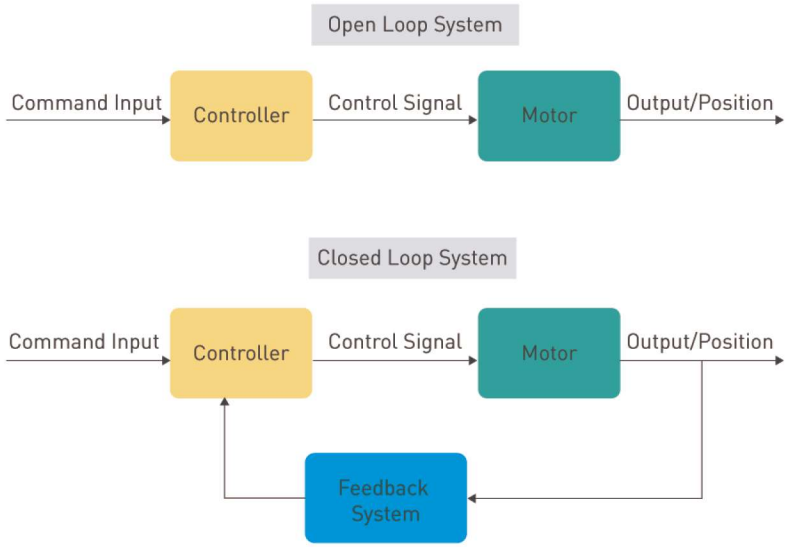
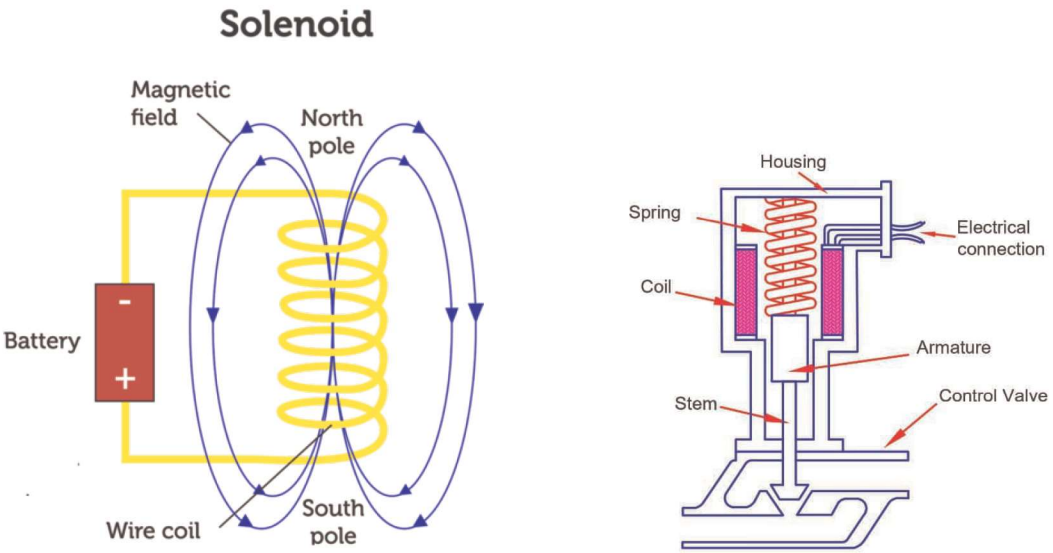


Figure 4.22: Open-Loop and Closed-Loop Control of Motors



(a) Solenoid current (b) electric acutaated solenoid valve

Figure 4.23: Solenoids Sensor

- Smart door locks
- ATM and vending machines
- IoT-enabled water valves

4.18.5 Piezoelectric Actuators

Principle: Certain crystals generate displacement when voltage is applied.
Equation:

$$x = d_{33} \cdot V$$

where d_{33} = piezoelectric coefficient, V = applied voltage.

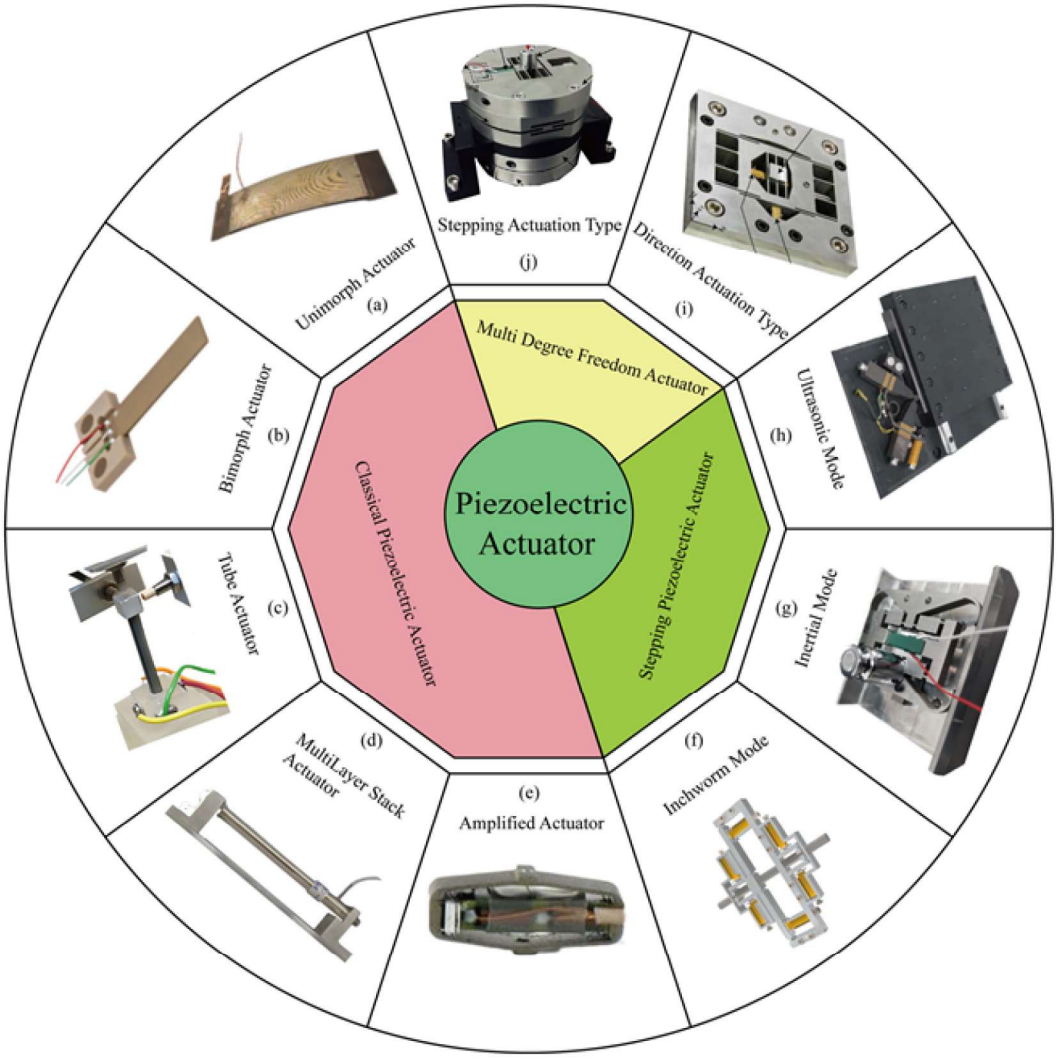


Figure 4.24: Piezoelectric Actuators examples

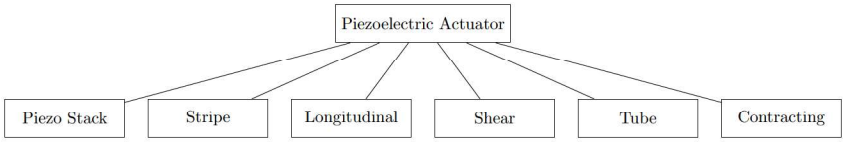


Figure 4.25: Piezoelectric Actuators types

Applications:

- Inkjet printers
- Precision micro-positioning
- Medical ultrasound devices

4.18.6 Shape Memory Alloys (SMA)

SMAs such as Nitinol change shape when heated above a transition temperature.

Applications:

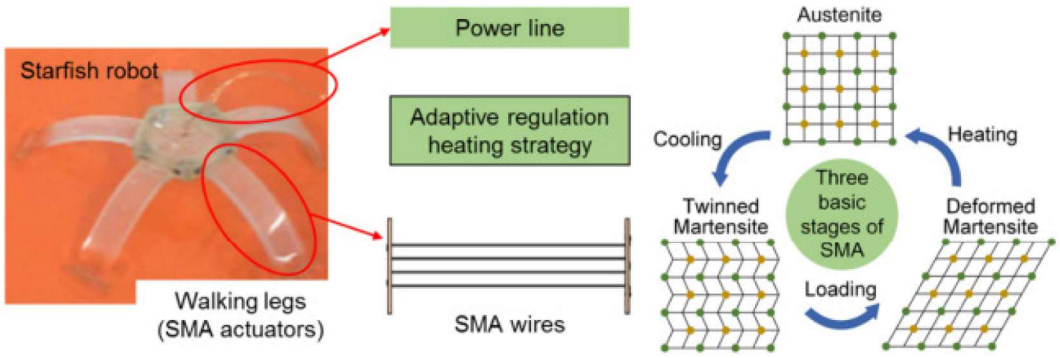


Figure 4.26: SMA Actuators

- Medical stents
- Micro-actuators for robotics
- Adaptive smart glasses

4.18.7 Hydraulic and Pneumatic Actuators

Equation (Hydraulic):

$$F = P \cdot A$$

where P is pressure and A area of piston.

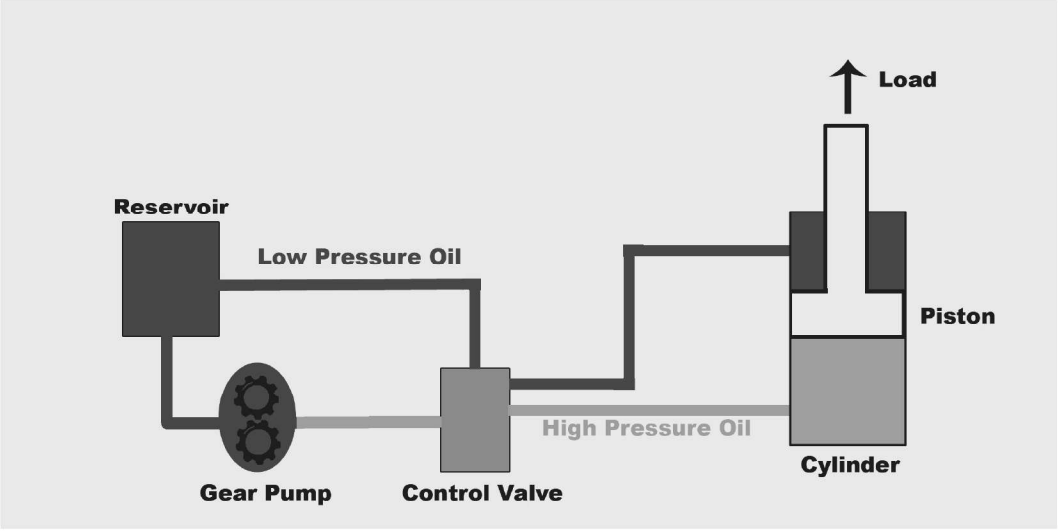


Figure 4.27: Hydraulic Actuator

Applications:

- Industrial robotic arms
- Smart heavy machinery

4.18.8 MEMS Actuators

Micro-Electro-Mechanical Systems actuators are miniaturized devices integrated on chips. They operate using electrostatic, piezoelectric, or thermal principles.

Applications:

- Smartphone vibration motors
- Microfluidic pumps
- Optical MEMS for communication

Figure 4.28: 1400RPM Speed 3V Mobile Phone Micro DC Coin Vibration Motor

4.19 IoT Case Studies Involving Actuators

4.19.1 Smart Home HVAC (Heating, ventilation, and air conditioning)

Sensors detect temperature and actuators regulate fans and dampers.

4.19.2 Agriculture: Smart Irrigation

Soil sensors → microcontroller → solenoid valve → irrigation.

4.19.3 Healthcare: Insulin Pumps

Glucose sensor data drives a micro-pump actuator to release insulin.

4.19.4 Autonomous Vehicles

Actuators control steering, braking, and acceleration based on sensor fusion.

Type	Motion	Precision	Force	IoT Applications
DC Motor	Rotary	Low	Medium	Fans, wheels
Stepper	Rotary	High	Medium	3D printers, CNC
Servo	Rotary	Very high	Medium	Robotics, drones
Solenoid	Linear	Medium	High	Locks, valves
Piezo	Linear	Ultra-high	Low	Medical, printing
SMA	Linear	Medium	Low	Biomedical, wearables

Table 4.7: Comparison of Common Actuator Types

4.20 Comparison of Major Actuators

4.21 Future of Actuators in IoT

- MEMS/NEMS scaling for miniaturized IoT.
- Energy-efficient soft robotics.
- AI-integrated adaptive actuators.
- Bio-inspired designs.

Actuators transform IoT systems from passive sensing devices into active participants in the physical world. Their classification, principles, and diverse applications demonstrate their indispensable role across industries. For computer science students, understanding actuators provides a holistic perspective, bridging software and hardware in the age of IoT.

Chapter 6

Basics of IoT Networking

6.0.1 Convergence of Domains

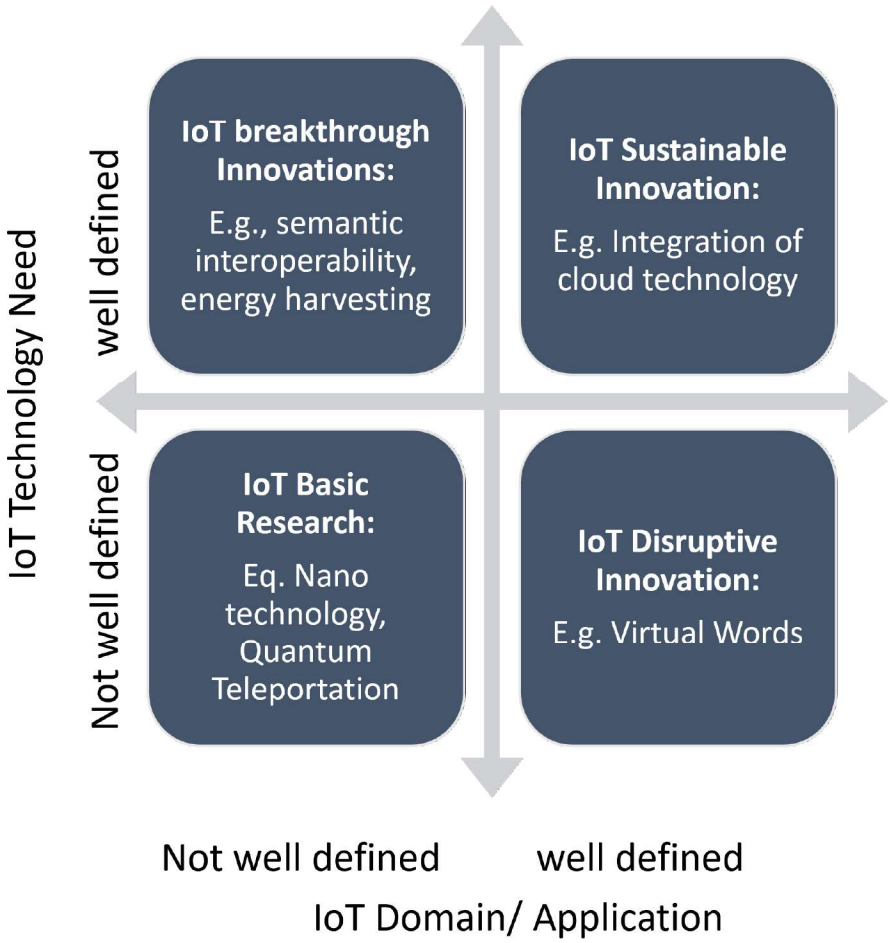


Figure 6.1: Innovation Matrix of IERC — Internet of Things European Research Cluster.

6.0.2 IoT Components

- Device (The Thing)

- Local Network
- Internet
- Backend Services
- Applications

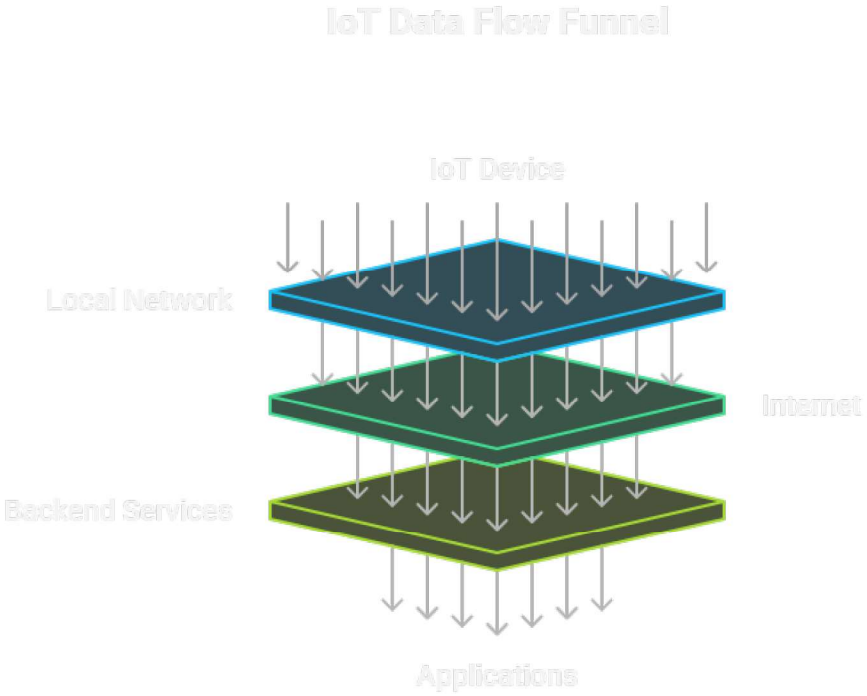


Figure 6.2: IoT Components

Functional Components of IoT

- Component for interaction and communication with other IoT devices
- Component for processing and analysis of operations
- Component for Internet interaction
- Component for handling Web services of applications
- Component to integrate application services
- User interface to access IoT

An Example IoT Implementation

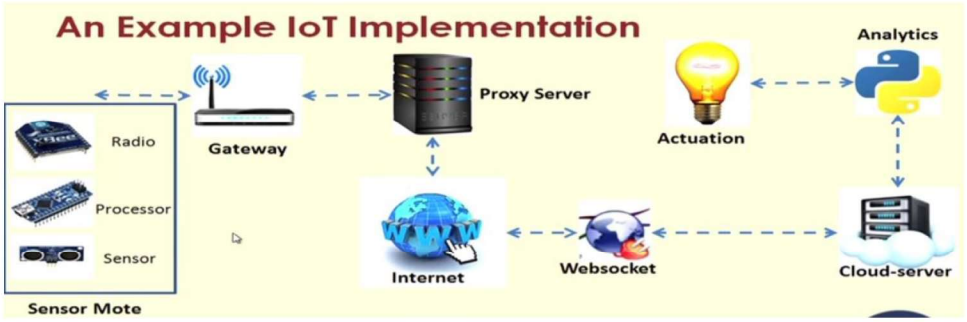


Figure 6.3: An Example IoT Implementation

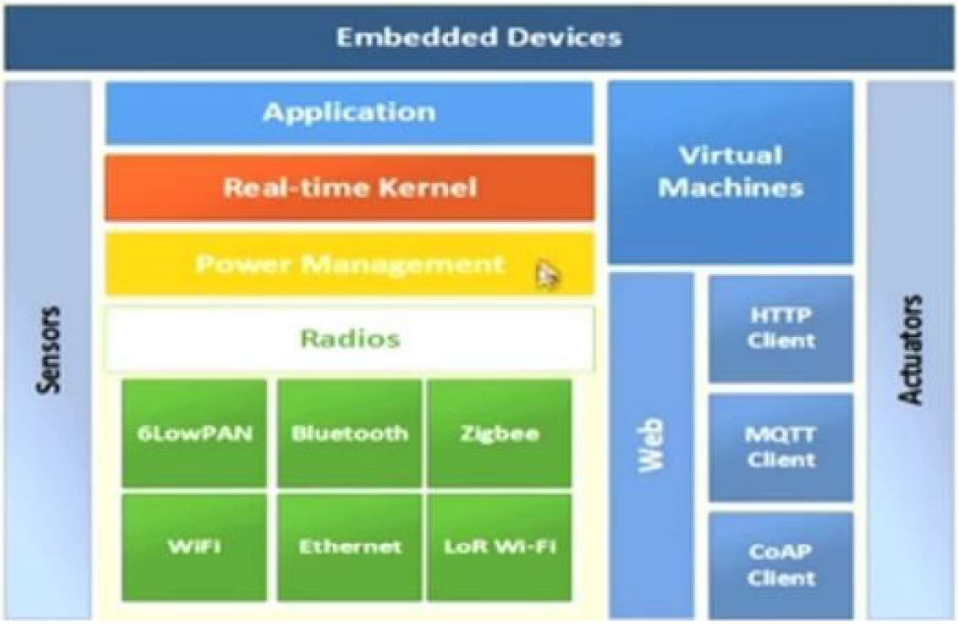


Figure 6.4: IoT Interdependence

6.0.3 IoT Interdependence

6.0.4 IoT Service Oriented Architecture

6.0.5 IoT Categories

Industrial IoT

- IoT device connects to an IP network and the global Internet.
- Communication between the nodes done using regular as well as industry specific technologies.

Consumer IoT

- IoT device communicates within the locally networked devices.
- Local communication is done mainly via Bluetooth, Zigbee or WiFi.
- Generally limited to local communication by a Gateway.

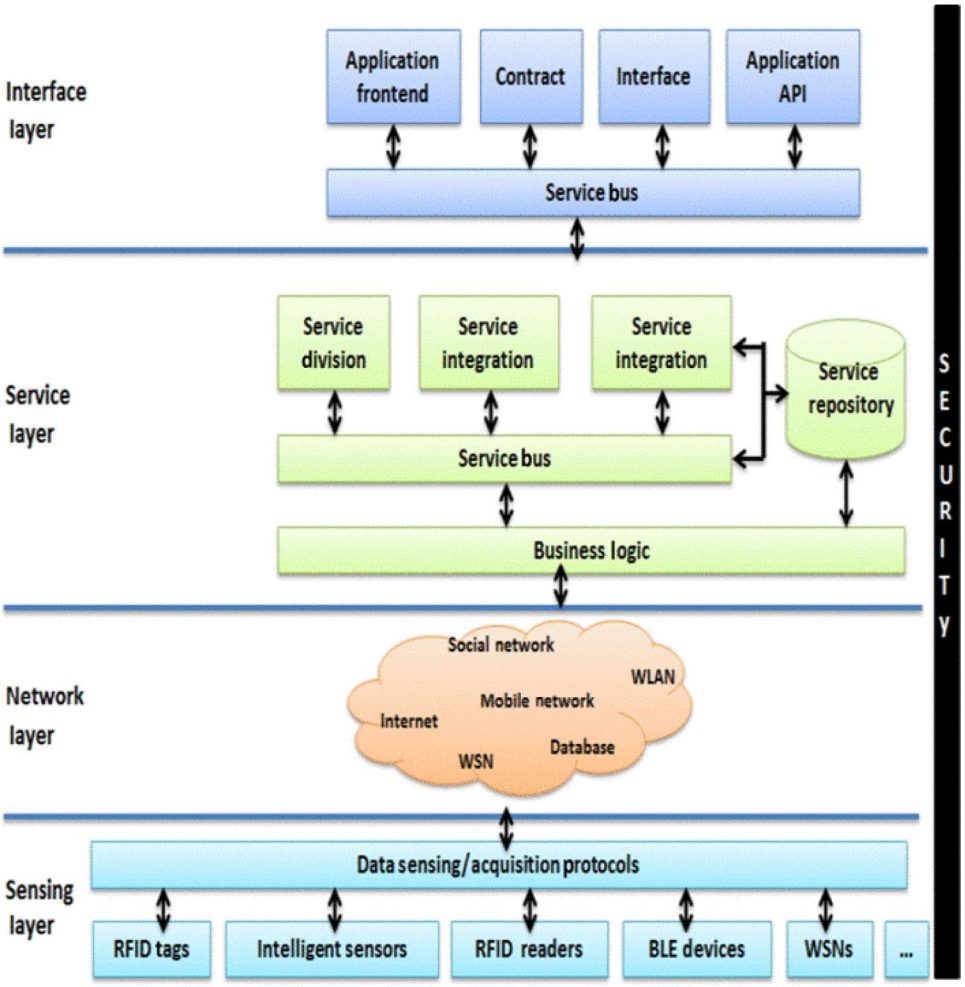


Figure 6.5: IoT Service Oriented Architecture

6.0.6 IoT Gateways

6.0.7 Key Technologies for IoT

- Device Intelligence
  - Objects should be able to intelligently sense and interact with the environment
  - Possibly store some passive or acquired data
  - Communicate with the world around them
  - Object-to-gateway device communication or even direct object-to-object communication is desirable
- Communication Capabilities
  - IP is considered to be key capability for IoT objects
  - Self-configuring capabilities, especially how an IoT device can establish its connectivity automatically without human intervention, are also of interest
  - IPv6 auto-configuration & multihoming features are useful, particularly scope-based IPv6 addressing features

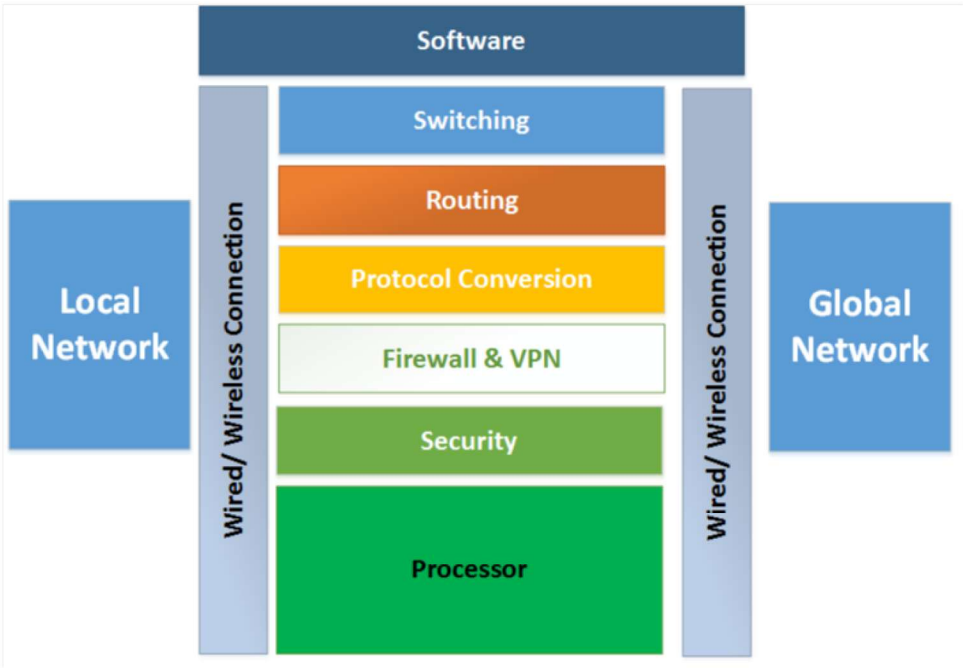


Figure 6.6: IoT Gateways

- Mobility Support
  - Some objects move independently, while others will move as one of group
  - Therefore, according to the moving feature, different tracking methods are required.
  - Mobile IPv6 (MIPv6) offers several capabilities that can address this requirement.
- Device Power
  - Devices have ultra-low-power capabilities
  - Devices must be of low cost
  - Devices must have small physical size & light in weight
  - Duty cycle—continuous or intermittent
- Sensor Technology
  - A sensor network is an infrastructure comprising sensing (measuring), computing, communication, data collection, monitoring, surveillance, and medical telemetry.
- RFID Technology
  - RFIDs are electronic devices associated with objects ("things") that transmit their identity (usually a serial number) via radio links.
- Satellite Technology
  - Ability to support mobility in all geographical environments (including Antarctica)

6.0.8 Technical Deviations from Regular Web

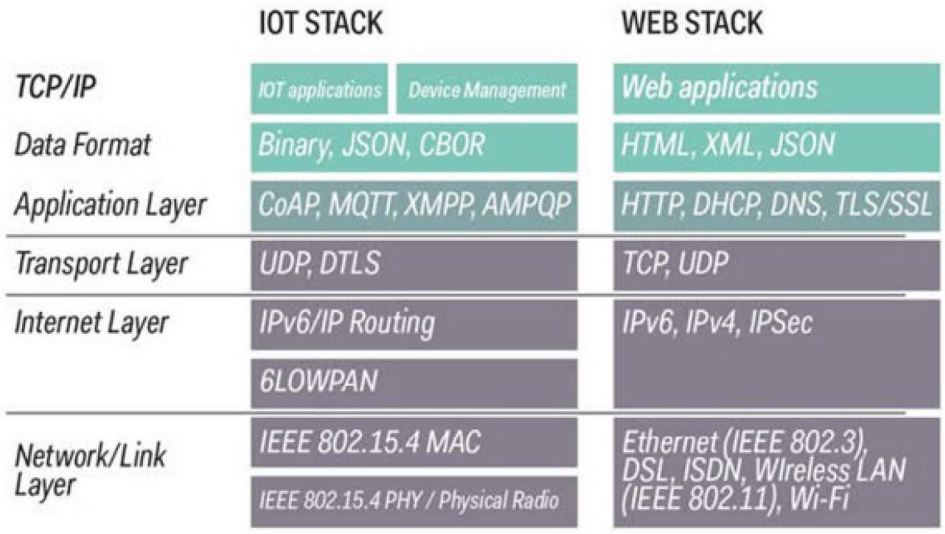


Figure 6.7: Technical Deviations from Regular Web

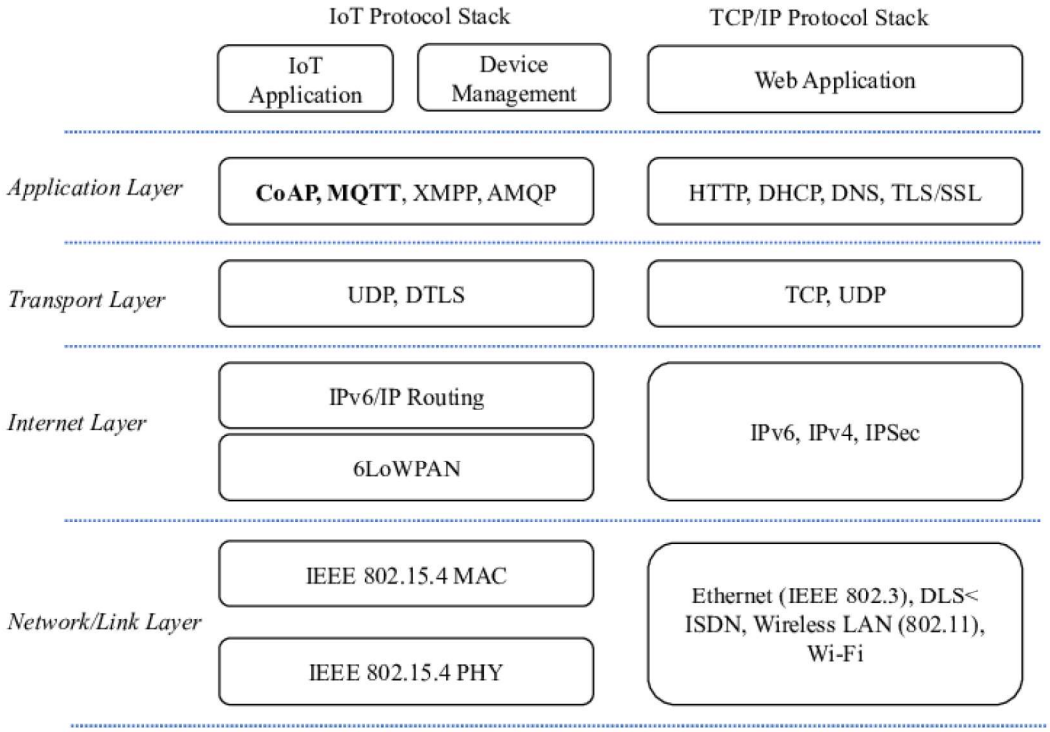


Figure 6.8: Technical Deviations from Regular Web

6.0.9 IoT Challenges

- Security
- Scalability

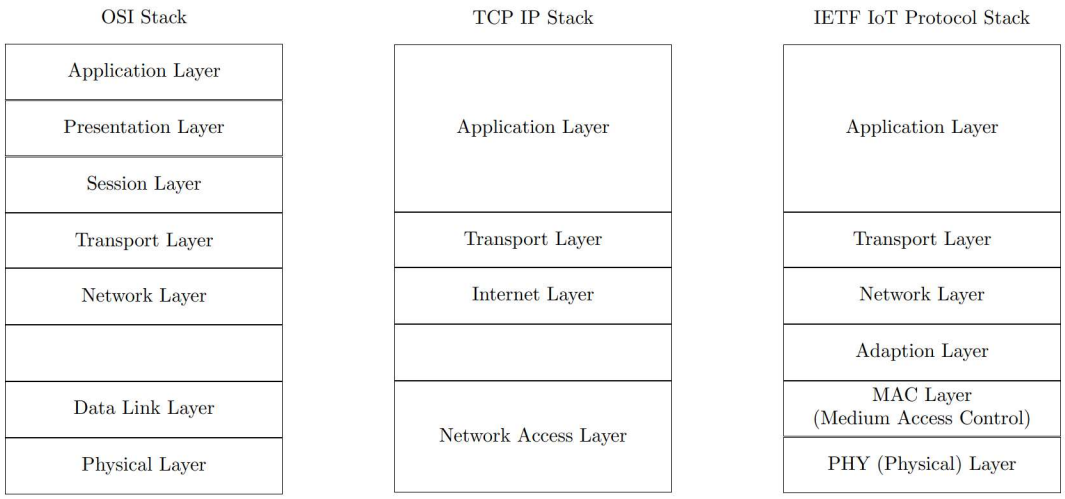


Figure 6.9: Technical Deviations from Regular Web

- Energy efficiency
- Bandwidth management
- Modeling and Analysis
- Interfacing
- Interoperability
- Data storage
- Data Analytics
- Complexity management (e.g., SDN)

6.0.10 Considerations

- Communication between the IoT devices(s) and the outside world dictates the network architecture.
- Choice of communication technology dictates the IoT devices hardware requirements and costs.
- Due to the presence of numerous applications of IoT enabled devices, a single networking paradigm not sufficient to address all the needs of the consumer or the IoT device.

Complexity of Networks

- Growth of networks
- Interference among devices
- Network management
- Heterogeneity in networks
- Protocol standardization within networks

6.0.11 IoT World Forum Standardized Architecture

- In an IoT System
  - Data is generated by multiple kind of devices
  - Processed in different ways
  - Transmitted to different locations
  - Acted upon by application
- The purposed IoT Reference model is comprised of seven levels.
- Each level is defined with terminologies that can be standardized to create a globally accepted frame of reference.

The model describes how tasks in each level should be handled to maintain simplicity, allow high scalability, and ensure supportability. Finally, the model defines the functions required for an IoT system to be complete. Figure illustrates the IoT Reference Model and its levels.

- Collaboration and Processes
- Application Layer
- Data Abstraction
- Data Accumulation
- Edge Computing
- Connectivity
- Physical Devices and Controllers

6.0.12 IoT World Forum Reference Model

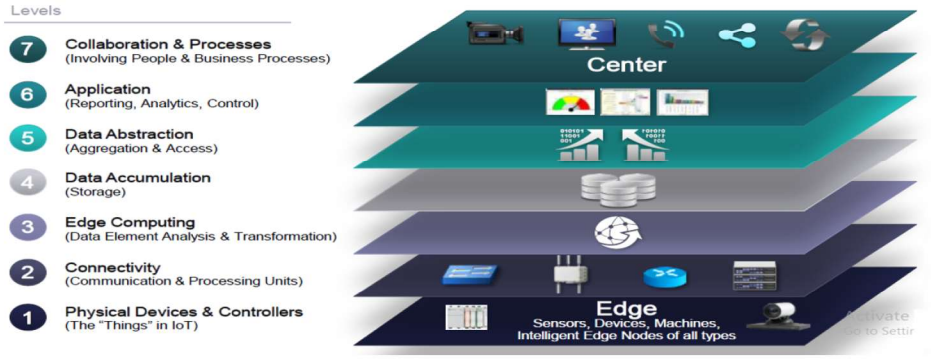


Figure 6.10: IoT World Forum Reference Model

1. Level 1: Physical Devices and Controllers
   - that might control multiple devices.

- These are the "Things" in IoT, and they include a wide range of endpoint devices that send and receive information.
- Dozens or hundreds of equipment manufacturers will produce IoT devices.
- To simplify compatibility and support manufacturability the IoT reference model generally describes the level of processing needed from Level 1 Devices

2. Level 2 : Connectivity

   - The most important function of Level 2 is reliable timely information transmission. This include transmissions:
     - Between devices (Level 1) and network
     - Across networks (East-West)
     - Between the network (Level 2) and low-level information processing occurring at Level 3.
   - One objective of the IoT Reference Model is communications and processing to be executed by existing networks. The IoT reference model does not require or indicate creation of a different network. It relies on existing network.

   However, some legacy devices are not IP enabled, which will require introducing communication gateways. Other devices will require proprietary controllers to serve the communication function. As Level 1 devices increases rapidly in number, the ways in which they interact with Level 2 connectivity equipment may change. Regardless of the details , Level 1 devices communicate through the IoT System by interacting with Level 2 connectivity equipments.

3. Level 3 : Edge (Fog) Computing

   - The functions of Level 3 are driven by the need to convert network data flows into information that is suitable for storage and higher level processing at Level 4 (data accumulation)
   - This means that Level 3 activities focus on high level data analysis and transformation
   - A basic principle of IoT Reference model is that *the most intelligent system initiates information processing as early and as close to the edge of the network* as possible. This is sometimes referred to as fog computing.

4. Level 4 : Data Accumulation

   - At level 4, Data Accumulation, Data in motion is converted to Data at rest.
   - As Level 4 captures data and puts it at rest, it is now usable by applications on a non real time basis.
   - In short, Level 4 converts event-based data to query-based processing. This is a crucial step in bridging the differences between the real time networking world and non real time application world.

5. Level 5 : Data Abstraction

- The Data Abstraction functions of Level 5 are focused on rendering data and its storage in ways that enable developing simpler, performance enhanced applications.
- It reconciles multiple data formats and ensures consistently semantics from various sources.
- Confirms that the data set is complete and consolidates data into one place or multiple data stores using virtualization.

6. Level 6 : Application

- Level 6 is the application Level where information interpretation occurs.
- Applications vary based on vertical markets, the nature of device data and business needs.
- Monitoring and control applications represent many different application models, programming patterns and software stacks , mobility, application servers, hypervisors, multi-threading , multi-tenancy etc.
- Suffice it to say that application complexity will vary widely.

7. Level 7 : Collaboration and Processes

- Consumes and shares application information
- Applications execute business logic to empower people. People use applications and associate data for their specific needs.
- Application (Level 6) give business people the right data, at the right time, so they can do the right thing.
- But frequently, the action needed requires more than one person. People must be able to communicate and collaborate, sometimes using the traditional internet, to make the IoT useful.
- Communication and collaboration often requires multiple steps.
- This is why Level 7 as shown in next figure represents a higher level than a single application.

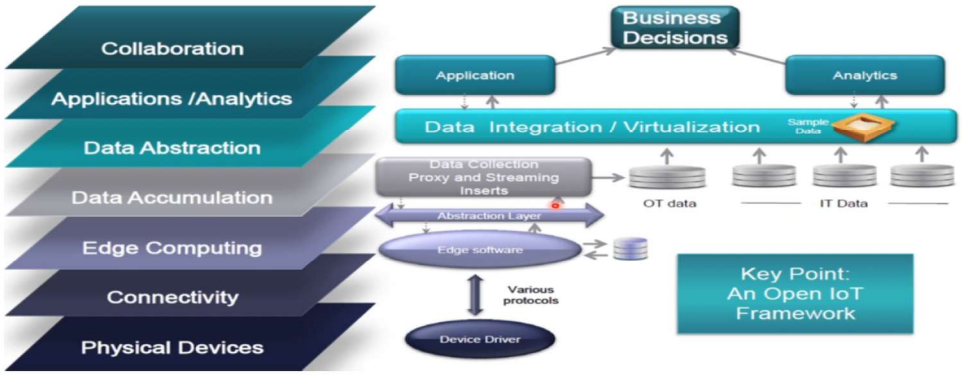


Figure 6.11: An Open IoT Framework