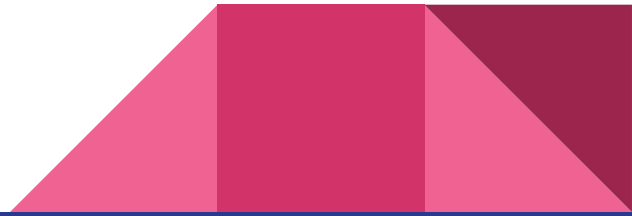
Introduction to IoT

RFID, ZigBee, Edge, Gateways, Platforms, Applications

Internet of Things (IoT)

The Internet of Things refers to a network of physical objects equipped with sensors, actuators, software, and connectivity that enables them to collect, exchange, and act on data. IoT integrates wireless communication, micro-electromechanical systems, and the Internet. Each device is uniquely identifiable and interoperates across existing network infrastructures. Due to the vast number of connected objects, IPv6 is essential for addressing.

IoT supports sensing and actuation, enabling remote monitoring and control without human intervention. It spans consumer, industrial, medical, and smart city environments.



IoT Architecture

IoT architecture consists of four layers:

Sensing Layer

Includes devices that gather data from the environment or specific domains. Examples include temperature, humidity, light, air quality, motion, ECG monitors, soil moisture sensors, GPS modules, industrial pressure or gas sensors, wearable heart-rate monitors, and smart home devices such as thermostats and door locks.

Network Layer

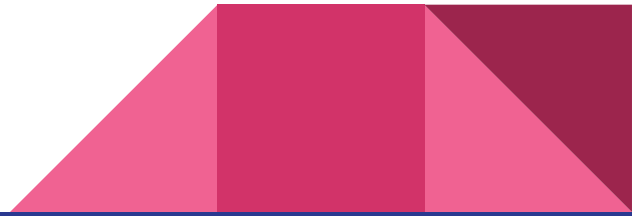
Transfers data between devices and cloud systems using gateways. Responsibilities include protocol configuration, encryption, authentication, malware protection, and routing through technologies such as Wi-Fi, Bluetooth, ZigBee, cellular networks, and MQTT/HTTP-based communication.

Data Processing Layer

Performs data aggregation, filtering, analytics, and decision-making. This processing may occur at gateways or in cloud services. It supports real-time analytics, event detection, and contextual decision logic.

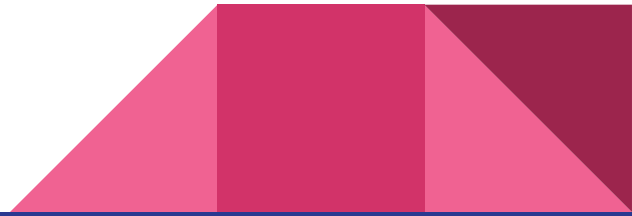
Application Layer

Provides user interfaces and domain-specific applications. Examples include fitness dashboards, pollution indexes, smart parking interfaces, and home automation controllers.



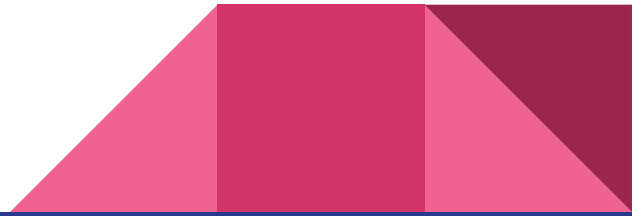
Edge Computing

Edge computing processes data near the source rather than sending all data to cloud systems. It places compute and storage resources close to IoT devices, reducing latency and bandwidth usage. It supports real-time responses, longer battery life, improved resilience, and localised analytics. Multi-Access Edge Computing (MEC) extends these capabilities to telecom environments.



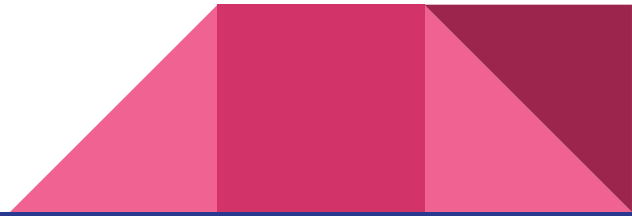
IoT Gateways

Gateways connect IoT devices to backend systems. Their core functions include data aggregation, protocol translation (e.g., MQTT, CoAP, HTTP, Bluetooth), preliminary data processing, security management, connectivity control, and support for local analytics. Gateways enable bidirectional communication between devices and cloud applications.



RFID Technology

RFID uses electromagnetic fields to identify and track objects through tags containing stored information. Passive tags draw energy from the reader’s field, while active tags contain their own power source and operate at greater distances. RFID does not require line-of-sight, unlike barcodes. Applications include manufacturing tracking, pharmaceutical authentication, inventory management, and animal identification.



ZigBee Technology

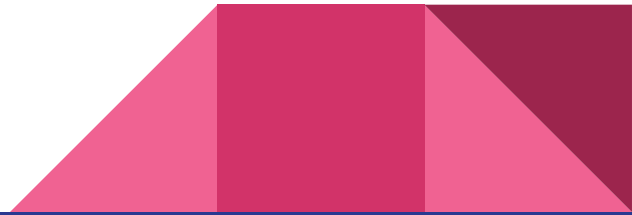
ZigBee is a low-power, low-data-rate wireless communication standard based on IEEE 802.15.4. It supports mesh networking and is suitable for battery-powered control and monitoring applications. ZigBee operates in several ISM bands, including 2.4 GHz, 784 MHz, 868 MHz, and 915 MHz, with data rates between 20 kbit/s and 250 kbit/s.

ZigBee devices use small packets (maximum 128 bytes) and support both 64-bit IEEE addresses and 16-bit short addresses, enabling networks with more than 65,000 nodes. Its applications include home automation, lighting controls, smart meters, industrial monitoring, security systems, and healthcare devices.

IoT Platforms

IoT platforms provide middleware that enables device provisioning, messaging, identity management, data collection, analytics, visualization, and rule-based automation. Typical stacks include gateways, communication modules, messaging systems, OTA update mechanisms, configuration tools, and analytics engines.

Examples include OpenRemote (open-source), AWS IoT (commercial), and ThingsBoard (open-source). Platforms support workflows such as event rules ("When-Then") and flow-based orchestration.



IoT Application Domains

Consumer and Healthcare Applications

- Infant monitoring systems providing real-time breathing, temperature, and activity information.
- Medication adherence tools using smart caps and ingestible sensors.
- Wearable health monitors capturing ECG, heart rate, respiration, and activity.
- Elderly monitoring systems that detect anomalies in daily routines.

Environmental and Smart City Applications

- Air pollution monitoring using distributed sensors that detect particulate matter and gases.
- Smart parking systems that monitor occupancy and compute utilisation levels.

Agriculture and Home Automation

- Smart irrigation and hydroponics systems based on real-time soil and environment data.
- Smart lighting systems with remote control and automated energy optimisation.
- Smart thermostats controlling HVAC systems based on environmental data.
- Smart locks supporting remote access, monitoring, and status notifications.
- Wireless outlets with energy monitoring and remote switching.

Industrial and Energy Applications

- Smart meters measuring real-time electricity consumption.
- ZigBee-based home energy management systems integrating sensors, outlets, and controllers.
- Parking navigation systems using ZigBee-enabled wireless sensor networks.

Q. A smart agriculture company deploys soil moisture sensors, humidity sensors, and environmental monitors across a 20-acre farm. The system frequently suffers from delayed alerts and excessive cloud bandwidth consumption.

Using the four-layer IoT architecture (Sensing, Network, Data Processing, Application), describe how you would systematically redesign the system to improve latency, efficiency, and reliability. Clearly state the key actions in each layer.

Redesigning IoT Agriculture Using 4-Layer Architecture

1. Sensing Layer

- Ensure the soil moisture, humidity, and environmental sensors are calibrated and capable of local buffering.
- Add basic filtering at device level (remove duplicate or excessively frequent readings).
- Configure sensors to send data periodically rather than continuously to reduce overload.

2. Network Layer

- Use an IoT gateway for protocol translation and secure transmission instead of directly sending all data to the cloud.
- Introduce light-weight protocols such as MQTT to reduce communication overhead.
- Apply encryption and authentication to secure data in transit.

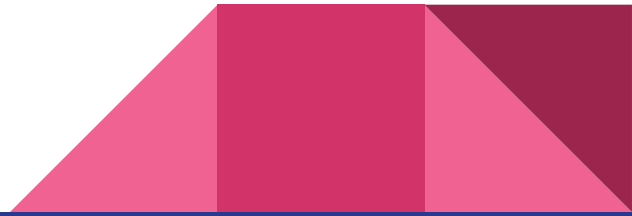
3. Data Processing Layer

- Perform **edge analytics** at the gateway:
  - Calculate derived metrics (e.g., moisture trend, humidity stability).
  - Trigger local alerts for irrigation when soil moisture drops below threshold.
- Forward only **summarised** or **event-based** data to the cloud to reduce bandwidth.

4. Application Layer

- Provide dashboards showing field condition trends, irrigation decisions, and alerts.
- Allow farmers to configure thresholds or rules (e.g., when moisture < 20%, start irrigation).

Overall: This redesign reduces latency (decisions happen at the edge), minimises cloud load, improves reliability, and ensures timely responses.



Q. You are designing a Smart Parking Management solution using the OpenRemote IoT Platform. Your goal is to display real-time occupancy of three parking facilities and trigger alerts when a site reaches 90% capacity.

a. Describe two different architectural approaches using OpenRemote components (Agents, Rules, Services, Manager UI). For each approach, specify the data flow.
b. State any assumptions you may need to make for device connectivity and protocol support.

Approaches Using OpenRemote IoT Platform

a. Two Architectural Approaches

Approach 1: Device → Agent → Rules → Manager UI

1. Parking sensors send data (MQTT/HTTP).
2. **Agent** in OpenRemote receives sensor values and updates asset attributes (Spaces Occupied, Spaces Total).
3. **Rules engine** applies a “When–Then” rule:
    - WHEN occupancy ≥ 90%
    - THEN trigger alert and update asset state.
4. **Manager UI** displays occupancy and alerts in real time

Approach 2: Third-party Gateway → OpenRemote Services → Flow Rules → Dashboard

1. External parking system aggregates raw sensor data.
2. Data ingested into OpenRemote through a REST/MQTT service.
3. **Flow-based rules** compute total occupancy percentage across multiple sites.
4. Insights appear on a **dashboard** similar to the SmartCity Parking example.

b. Assumptions

- Sensors support MQTT/HTTP or are connected through a third-party gateway.
- Parking asset attributes exist: SpacesTotal, SpacesOccupied, Status.
- The OpenRemote deployment has agents configured for data ingestion.
- The network layer is stable and secured.