1. **List the IoT protocol stack commonly used in the following ISO/OSI reference model layers.**

    a. **Physical layer and Data link layer**

    b. **Network layer**

    c. **Transport layer**

    d. **Application layer**

OSI Layer	IoT Protocols Commonly Used	Description / Function
a. Physical Layer & Data Link Layer	- IEEE 802.15.4 - Bluetooth Low Energy (BLE) - Wi-Fi (IEEE 802.11) - LoRa / LoRaWAN (Physical + MAC) - NFC - Zigbee (uses 802.15.4 PHY/MAC) - Ethernet (for gateways)	These protocols handle data transmission over physical media and link management (MAC addressing, framing, error detection, etc.). They provide low-power wireless connectivity suitable for IoT devices.
b. Network Layer	- IPv6 (Internet Protocol v6) - 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) - RPL (Routing Protocol for Low-Power and Lossy Networks) - NAT64 (in some gateway use cases)	Responsible for addressing, routing, and packet forwarding between IoT nodes and gateways. 6LoWPAN enables IPv6 over constrained networks.
c. Transport Layer	- UDP (User Datagram Protocol) - TCP (Transmission Control Protocol, less common in constrained devices) - DTLS (Datagram Transport Layer Security) - TLS (Transport Layer Security)	Ensures end-to-end communication, data segmentation, reliability, and security. UDP is preferred in IoT for its low overhead.
d. Application Layer	- CoAP (Constrained Application Protocol) - MQTT (Message Queuing Telemetry Transport) - AMQP (Advanced Message Queuing Protocol) - HTTP/HTTPS (HyperText Transfer Protocol) - XMPP (Extensible Messaging and Presence Protocol) - DDS (Data Distribution Service)	Application protocols define how IoT devices exchange data, report sensor readings, and receive commands. CoAP and MQTT are lightweight and optimized for constrained environments.

2. Compare IoT protocol stack vs Internet protocol stack for all ISO/OSI reference model layers.

ISO/OSI Layer	Internet Protocol Stack (Traditional Internet)	IoT Protocol Stack (for constrained devices)
Application Layer (Layer 7)	HTTP, HTTPS, FTP, SMTP, DNS	CoAP, MQTT, AMQP, XMPP, DDS, HTTP (sometimes)
Transport Layer (Layer 4)	TCP, UDP	UDP (mainly), TCP (sometimes), DTLS/TLS for security
Network Layer (Layer 3)	IPv4, IPv6, ICMP, OSPF, BGP	IPv6, 6LoWPAN, RPL
Data Link Layer (Layer 2)	Ethernet (IEEE 802.3), Wi-Fi (IEEE 802.11)	IEEE 802.15.4, BLE, Zigbee, LoRaWAN, Wi-Fi, NFC
Physical Layer (Layer 1)	Wired (copper, fiber optics), Wireless (Wi-Fi, LTE)	IEEE 802.15.4 PHY, BLE, LoRa, Sub-GHz ISM bands

3. Develop a home automation system that includes security and measures ambient temperature using CoAP protocol. Within the home network, different IoT devices are connected and communication is established among them using the CoAP protocol. A CoAP server helps in updating data to cloud and accessing over local network. The data is updated on to the cloud for further analysis and status of devices at home is retrieved. =>

1. Home Automation Security and Ambient Temperature Measurement

In a home automation system, **security** ensures the protection of the home and its devices from unauthorized access or control. Security in such systems involves the use of smart locks, motion sensors, door sensors, and surveillance modules. These components help monitor and safeguard the home environment.

Each device is connected to the home network and communicates securely with the central controller. For example, when a door is opened or motion is detected, a CoAP message is sent to the home server, which triggers an alert or activates other security devices such as alarms or cameras.

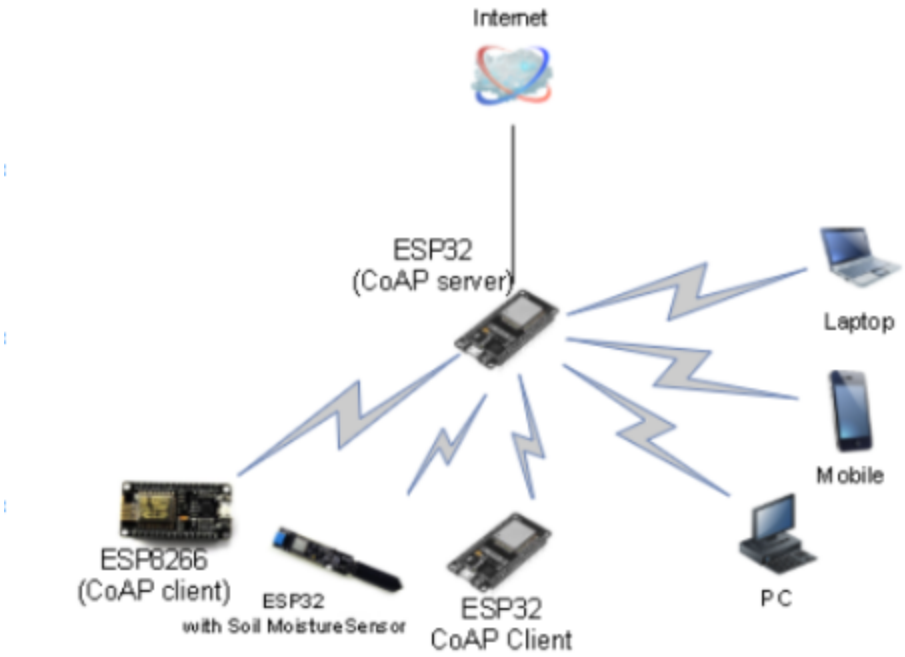
Along with security, the system also **measures the ambient temperature** using IoT sensors like **DHT11**, **DHT22**, or **BME280**. These sensors send real-time temperature readings to the CoAP server at fixed intervals. The data helps in maintaining comfort by automatically controlling air conditioners or fans based on temperature thresholds.

2. IoT Connections and Communication using CoAP

In this system, multiple IoT devices such as temperature sensors, door locks, lights, and motion detectors are interconnected through a local Wi-Fi network. Communication among them is established using the **Constrained Application Protocol (CoAP)**.

CoAP is a lightweight, REST-based protocol designed for constrained devices and networks. It uses simple **GET, POST, PUT, and DELETE** methods similar to HTTP but works over **UDP**, reducing data overhead.

For example:



- A temperature sensor **POSTs** its data to the CoAP server.
- A mobile app **GETs** data from the server to display current readings.
- The CoAP **Observe** feature allows real-time updates so that devices automatically respond when conditions change.

This efficient communication ensures low power consumption and fast responses among IoT devices within the home network.

3. Updating Data to Cloud and Local Network Access

The **CoAP server** inside the home network acts as a **gateway** between local devices and the cloud. It collects sensor data and device status, then **updates it to the cloud** using secure Internet protocols such as HTTP or MQTT.

Fig. 2. CoAP architecture.

This allows users to **access and monitor their home remotely** through cloud dashboards or mobile apps. At the same time, users inside the house can access the same data **locally** through the CoAP server, even if the Internet connection is temporarily unavailable.

Thus, the system provides both **local control** (for quick responses) and **remote access** (for convenience and monitoring).

4. Cloud Data Analysis and Device Status Retrieval

Once data reaches the cloud, it is stored and analyzed for better understanding of home conditions. Cloud platforms perform **data analysis** such as temperature trend tracking, energy-usage optimization, and anomaly detection (for example, detecting sudden temperature rises that may indicate fire hazards).
Users can **retrieve the current status of devices**—like whether doors are locked, lights are on, or room temperature—through a mobile or web interface connected to the cloud.
If any abnormal activity is detected (e.g., intrusion or temperature beyond limit), the cloud service or local server can immediately notify the user via alerts or messages.

4. Case study on:

a. Airtel IoT's SuperTracker device for tracking shipments

The Challenge

A leading supply-chain logistics company operating across 29 states and 156 hubs struggled to track high-value consignments effectively.

- Lack of real-time visibility when using third-party fleets.
- Dependence on manual or vendor-provided data led to billing errors.
- Communication gaps between vendors and company managers caused delays in responding to incidents.
- Increased risk of cargo theft, misplacement, and delayed deliveries.

These limitations made it essential to adopt a modern IoT-based tracking system.

3. The Solution – Airtel IoT SuperTracker

Airtel's IoT SuperTracker is a compact, **portable tracking device** powered by Airtel's IoT connectivity platform. Unlike fixed GPS trackers, it is **independent of vehicles**, meaning it can be attached directly to shipments.

Key features:

- Independent Shipment Tracking.
- Long-lasting battery
- Real-time visibility
- Geo-fencing security
- Accurate billing

4. Impact and Benefits

Implementation of the Airtel SuperTracker produced measurable improvements:

- **Enhanced operational efficiency** through optimized routes and reduced downtime.
- **Lower support and maintenance costs** because of the device's reliability.
- **High return on investment (ROI)** from faster deliveries and reduced losses even when a few devices were not recovered.
- **Greater security and customer trust** owing to reliable real-time monitoring of high-value goods.

B. Data collection, storage and computing using Xively platform

=> 1. Data Collection

In the proposed system, smart sensors such as:

- DHT11 – for temperature and humidity,
- MQ-2 – for detecting gases or smoke,
- ACS712 – for measuring current and power consumption,
  collect real-world parameters from the environment.

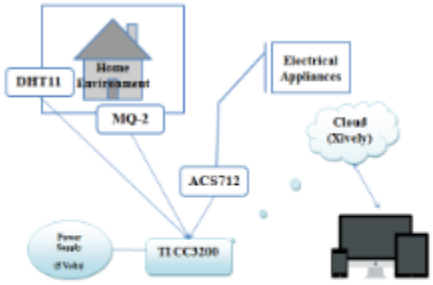


Figure 1. Block Diagram of the System

The raw sensor readings are calibrated using Arduino Due and TI CC3200 microcontrollers. The TI CC3200's built-in Wi-Fi module sends HTTP POST requests to the Xively cloud server, transmitting real-time sensor data.

Thus, data collection occurs through continuous sensing, local calibration, and secure uploading to Xively.

2. Data Storage on the Xively Cloud

Once transmitted, the data is stored on Xively using a unique Feed ID and API Key generated for each device.
Within Xively, data is organized into channels (e.g., Temperature, Humidity, Power Consumption), where each channel logs values over time.
The platform also allows:

- Historical data storage,
- Data visualization in graphical format, and
- Real-time updates through continuous streams.

This enables users to monitor multiple environmental parameters remotely using smartphones, tablets, or computers.

3. Computing and Visualization

Xively supports data processing and analytics through its cloud interface.
Users can:

- View trends and graphs for temperature, humidity, and energy consumption.
- Analyze power usage efficiency (e.g., comparing LED vs. incandescent bulbs).
- Receive alerts or logs when threshold values (like gas concentration) exceed safe limits.

The platform also logs each HTTP request in a request log, providing transparency and tracking for all sensor communications.

4. Advantages

- Real-time remote monitoring of environment and power usage.
- Cloud-based computation with minimal hardware processing load.
- Ease of deployment using API and channel-based architecture.