
子아茹♡

IOT

IOT refers to network of interconnected physical devices that can collect, share and exchange data over the internet without human intervention.

IOT means connecting everyday objects to the internet so they can send and recieve data.

Characteristics of IOT :-

1. Connectivity :- Device connect through the internet and local networks.

2. Automation :- IOT systems can operate automatically without human help.

3. Sensing :- Devices sense or collect real world data like temprature, motion etc.

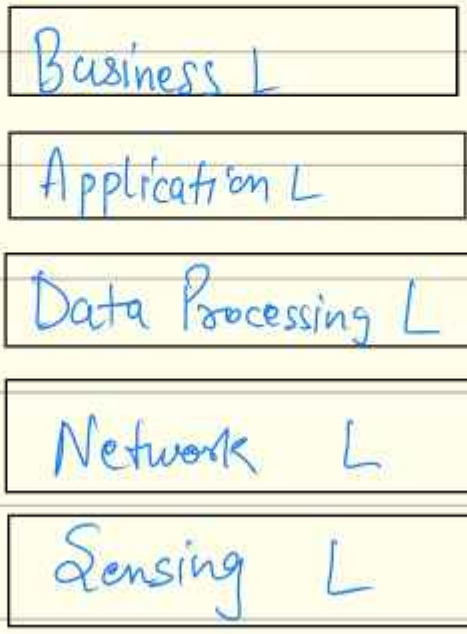
4. Cloud Computing Integration :- Data is processed and stored in cloud servers for scalability.

5. Real time operations :- IOT devices provide real time monitoring and response.

6 Data Analytics :- System analyze data to make smart descisions.

7. Energy efficient :- Runs on battery consumes less power.

Architecture of IOT :-



Components of IOT :-

① **Sensors / Devices** :- These collect data from the environment such as temperature, humidity, motion etc.

eg:- Temperature sensor, motion detector, camera

② **Connectivity / Network** :- Transmits data from sensors to the cloud or server.

eg :- Wifi, Bluetooth, ZigBee, Cellular (4G/5G).

③ **Data processing / Cloud** :- Data collected by devices is processed analyzed, and stored here.

eg :- Cloud platform like AWS IOT, Azure IOT.

④ **User Interface (UI)** :- Allow users to monitor or control devices.

eg :- Mobile app. web dashboard, voice assistant

(5) Actuators :- Perform actions based on processed data.
eg:- Turning on lights, adjusting AC temperature, locking doors.

IOT Ecosystem :-

The IOT ecosystem is a complete network or system that includes all the components (devices, connectivity, platforms, software, users, etc.) which work together to collect transmit, analyze, and act on data from IOT devices.

Sensor / Devices
↓
Connectivity / Communication Network
↓
IOT Gateway
↓
Cloud / Data Storage Platform
↓
Data processing
↓
UI / Application

IOT / Devices / Sensors / Actuators :-

Device Collect data from env. (sensors) and perform actions (actuators)

Connectivity / Communication Network :- Medium that connects IOT devices to cloud or other devices.

IOT Gateway :- Acts as bridge btw IOT devices and cloud - it filters, preprocesses and securely transmit data.

Cloud / Data Storage Platform :- Stores, process and analyze large amount of data.

Data Processing / Analytics :- Converts raw data into useful insights and trigger action automatically.

UI / Applications. :- Provide a way for user to monitor, control, and make desions.

IOT Communication APIs →

Application programming interface is software that allows to web application to communicate with each other on the Internet.

a) REST Based API :-

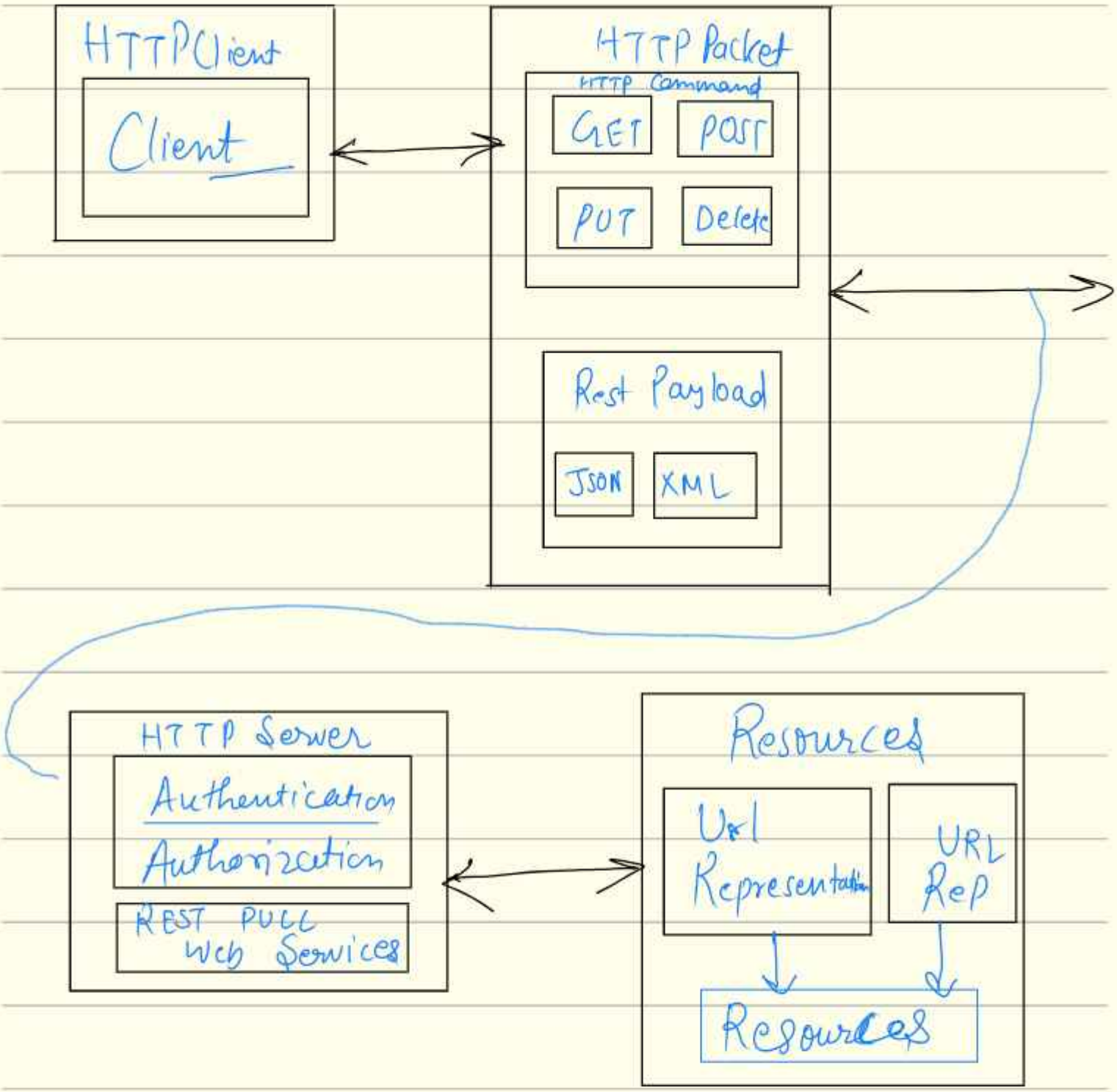
Representational state transfer. It creates an object and there after sent the values of object in response to client request.

Uses web protcols like HTTP / HTTPS

Data format → JSON or XML

Uses Client server model.

Each request involves setting up a new TCP connection



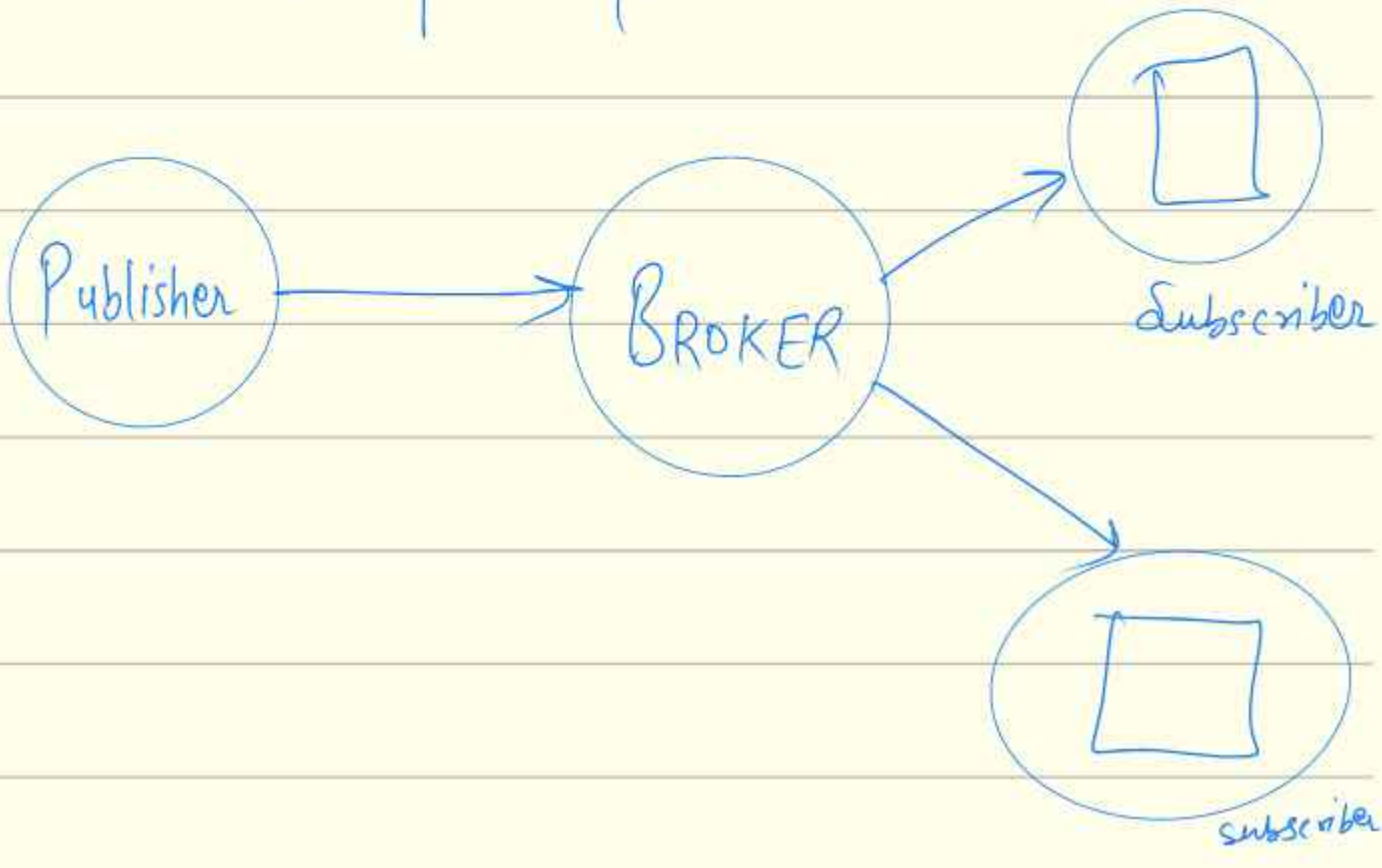
2 Web Sockets :-

- Provides two way (bi-directional) communication.
- Used in real time application.

- Single tcp Connection.
- No header overhead
- full duplex.

Websockets

MQTT → Message Queuing Telemetry Support
User publish /suscriber mode



M2M and IOT :-

Both M2M and IOT are technologies that enables communication btw devices without human intervention. However M2M is limited to direct device-to-device communication while IOT is an advanced form that connects devices through the internet, enabling intelligent data analysis and cloud-based control.

M2M	IOT
1. Machine to Machine	1. Internet of Things.
2. It refers to direct communication between machines or devices using wired or wireless networks.	2. It is a network of interconnected physical devices that communicate via Internet to collect and share data.
3. Point to point communication btw devices	3. Internet based communication through IP networks and cloud.

4. Uses cellular, wired, or short range wireless connections.	4. Uses internet, cloud computing and multiple wireless technologies (Wifi, Bluetooth, Zigbee etc.
5. Limited to specific applications like telemetry and remote monitoring.	5. Boarder scope including smart homes, cities healthcare, etc.
6. Data is usually transmitted to a single server or central system of monitoring.	6. Data is collected, stored, and processed in the cloud for analytics and descision making.
7. Minimal or no human interaction required once deployed.	7. Provide interfaces (apps, dashboards) for human interaction & control.

8. Mostly reactive - performs pre-defined tasks.	8. Smart and adaptive - supports automation & AI-based deasion-making.
9. Less scalable: limited no. of devices can be connected.	9. Highly scalable millions of device can connected globally.
10. eg: Smart electricity meters communicating with a control center.	10. eg: Smart Home systems controlled by smartphone and cloud.
11. Technised: GSM, RFID.	11. Cloud Computing, Big Data, AI

Reason of shifting from M2M to IOT :-

1. Limited Connectivity in M2M :- M2M systems mostly use point to point connections, restricting scalability and global access.

2. Lack of Data Intelligence :-

M2M only transmits data; it does not analyze or provide insights for decision-making.

2. Internet Integration in IOT :-

IOT connects devices via the internet, enabling remote control and cloud-based management.

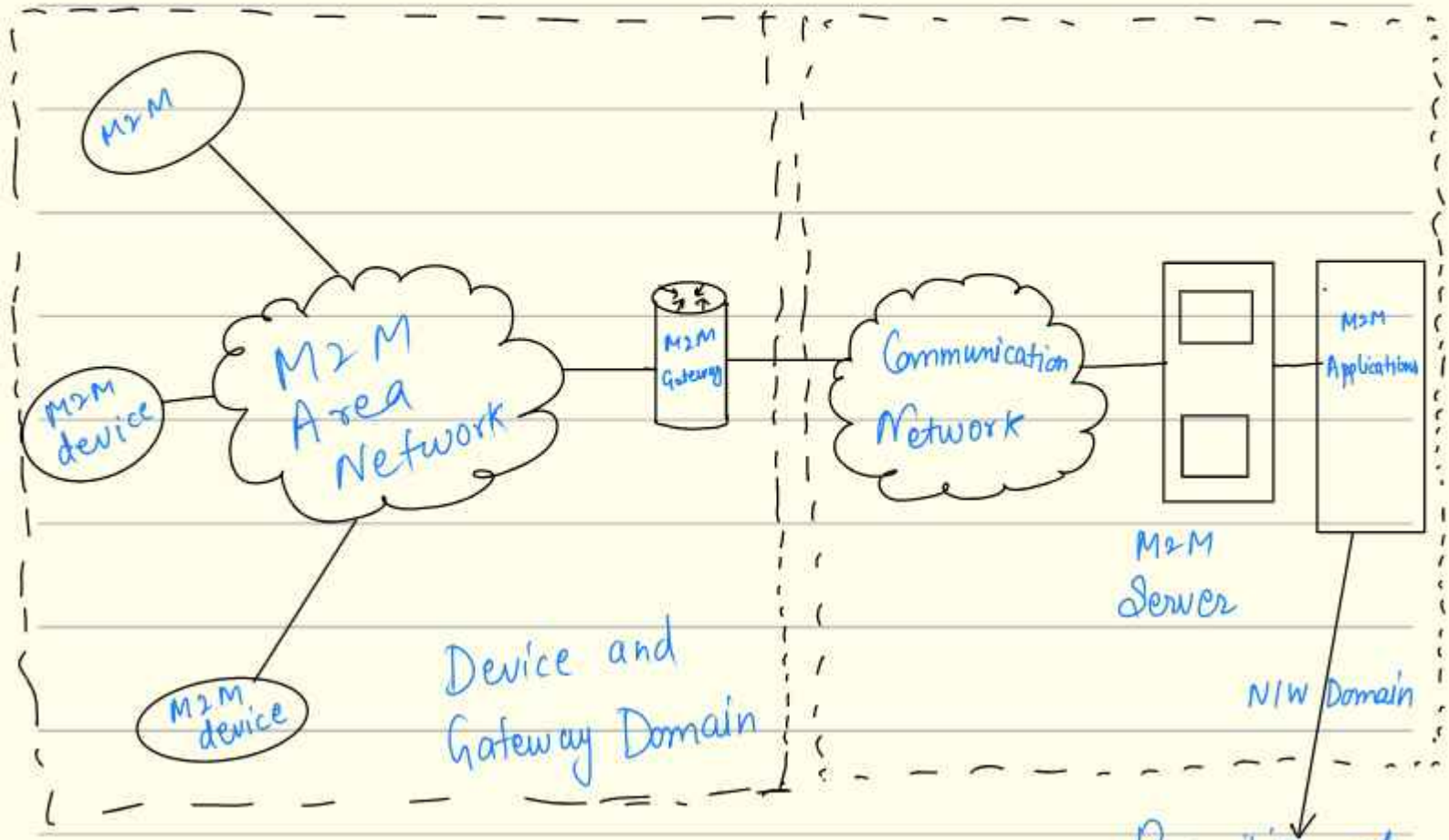
4. Scalability and Flexibility :-

IOT can connect millions of heterogeneous devices, unlike M2M, which is limited to specific machines.

5. Advanced Applications. :-

IOT supports smart cities, healthcare, agriculture, and AI-driven systems, going beyond simple device to device communication of M2M.

M2M Architecture



Responsibility of Applications.

→ Maintenance → Services

→ Upgrading

→ Client Applicating

Technologies of M2M Communication Protocols :-

1. Message Queuing Telemetry Transport (MQTT)
2. Constrained Application Protocol (COAP)

3. Advanced Message Queuing Protocol (AMQP)
4. Lightweight Machine to Machine (LWM2M)

2. COAP :- COAP is designed to allow simple devices to join IOT through the low bandwidth restricted network.

[Bandwidth → frequency of data
eg → 2G → low bandwidth]

COAP is designed for M2M and IOT applications. COAP is an application layer protocol that follows the request response model.

COAP Layer :-

Application Layer
UDP Layer

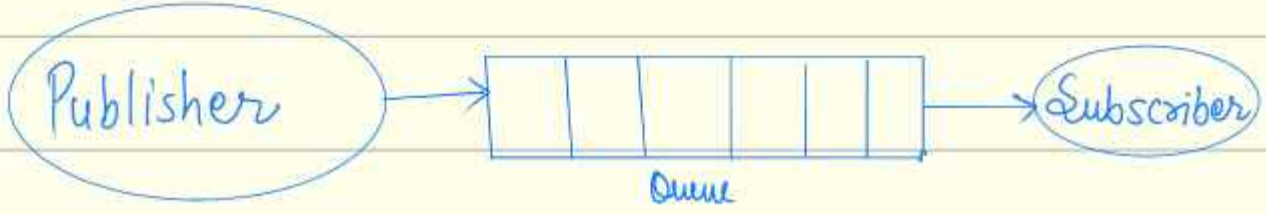
Application Layer :- It is designed for communication method based on the request response model.

UDP Layer :- It is designed to deal with UDP and asynchronous messages.

2. AMQP :- Based on Queue (Pull - Push Model).

It is an open standard application layer protocol used for message-oriented communication between systems. It enables reliable, secure, and asynchronous message exchange between between different applications or devices — even if they are built in different languages or run on different platforms.

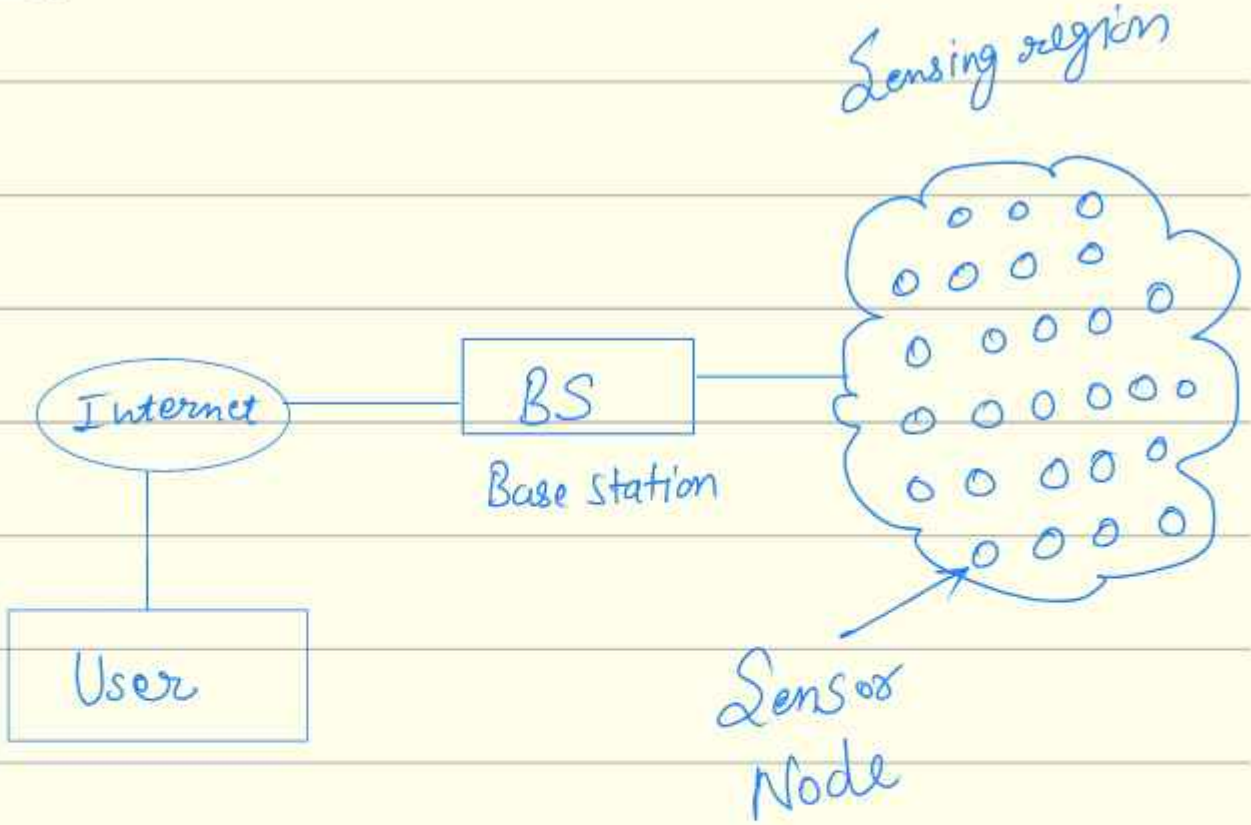
AMQP is designed to provide message delivery guarantee, queuing and routing between sender and reciever in distributed systems.



4. LWM2M :- Developed for managing and communicating with IOT and M2M device efficiently especially with limited power, memory and processing capability.

It is designed to provide device management, service enablement, and data communication in resource Constraint IOT env.

WSN :-



WSN are highly distributed lightweight nodes, deployed in large no. to monitor the environment or system and transmit the collected data wirelessly to central location.

Components of WSN :

1. **Sensor Nodes** : Small devices that sense data from the environment.

2. **Base Station** : Collect data from multiple sensor nodes and forwards it to the main server or cloud.

3. **Gateway / Server** :

Acts as an interface between the WSN and external network

4. Power Unit.

5. **Communication module**

Provides wireless connectivity using protocols like ZigBee, Bluetooth or WiFi.

Privacy and Security Issues in IOT :-

- Data collection without consent
- Data ownership
- Data sharing and sale.
- Tracking and Serveillance.
- Indequate Data Deletion.
- Weak Authentication and Authorization.
- lack of encryption.
- Malware and botnets
- Physical Security threats.
- DOS Attack.

BACnet :- (Building Automation and Control Network)
Developed for building automation and control systems.

Zigbee ÷ Wireless communication protocol based on the IEEE 802.15.4 standard.
Designed for low rate, low-data-rate, and short-range communication.

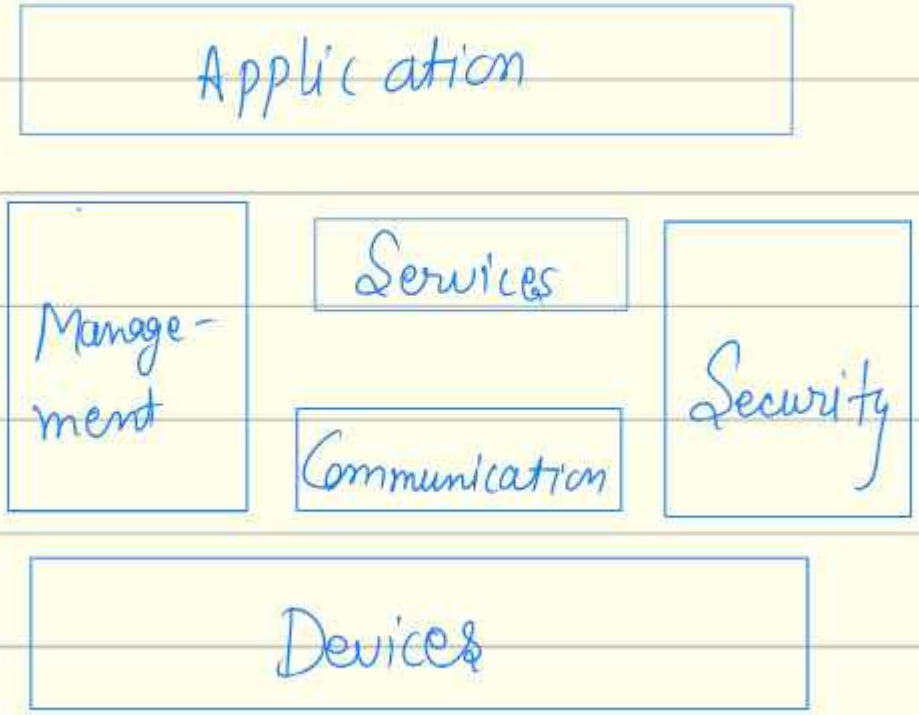
Modbus ÷ Modbus is a serial communication protocol developed by Modicon for connecting industrial electronic devices. It is one of the oldest and most widely used industrial protocols.

TCP → Transmission Control protocol
UDP → User Datagram Protocol

Logical Design ÷ logical design of IOT defines the abstract architecture - how system functions, how data flows, how software components communicate.

1. IOT fundamental block
2. IOT communication models
3. IOT Communication API's

1. IOT functional Blocks :



Physical Design of IOT :

Physical design of IOT refers to the actual physical components (hardware) that make up an IOT system the devices, sensors, actuators, and how they are connected.

→ IOT Devices | Things : Physical objects embedded with sensors and actuators.

→ Sensors : Collect data from env.

→ Actuators : Perform action based on commands.

→ Connectivity / Communication ÷ Wifi, Bluetooth, Zigbee.

→ IOT Gateway ÷ Acts as a bridge beth. IOT devices and the cloud.

→ Cloud / Server ÷ Stores, processes, and analyzes data recieved from devices.

→ UI ÷ Allows user to interact with IOT systems (mobile apps, dashboards, etc.)

Protocol Standardization for IOT ⇒

It is the process of defining and adopting common rules, formats, and procedures for communication between IOT devices and networks.

Objective of Standardization ⇒

a) Interoperability ÷ Different vendor's device can work together.

(b) Scalability :- Million of devices can connect without system failure.

(c) Reliability :- Ensure accurate and timely data delivery.

(d) Security :- Protect data from hackers and unauthorized access.

(e) Cost Reduction :- Less development efforts and testing cost.

Importance of Protocol Standardization :-

1. Device Compatibility :- One light bulb should work with any smart hub.
2. Efficient Communication :- Optimized data transfer model.
3. Better Innovation :- Developers can focus on features instead of compatibility.
4. Global Acceptance :- Device work worldwide.
5. Strong Security Framework :- Standard security algorithms improve trust.

Key Areas where Standardization is Required

1. Communication Standards :- Define how devices talk.

Examples :

- IEEE 802.15.4
- Bluetooth Low Energy.
- Wi-fi

2. Data Formats :- Define how data is represented.

eg → JSON, XML, SenML

3. Application layer Protocols :- Control messaging and data exchange.

eg :- HTTP, MQTT, CoAP, AMQP

4. Security Standards :-

Define
- Encryption
- Authentication
- Access Control

eg :- TLS, DTLS, AES, OAuth.

Major Standardization Bodies :-

- IEEE, IETF, ISO/IEC, OneM2M, ITU, W3C

SCADA ⇒ SCADA (Supervisory Control And Data Acquisition) is a system used to monitor and control industrial processes such as

- Power plants
- Water Systems
- Manufacturing
- Oil & gas
- Traffic Control

SCADA collects data from sensors and machines and sends it to a central system for monitoring and control.

Common SCADA Protocols :-

(1) **Modbus** :- • Most widely used SCADA protocol
• Master - Slave Communication.

Types : · Modbus RTU (Serial Communication)
· Modbus TCP (Ethernet)

Used in PLC Communication

2. DNP3 (Distributed Network Protocol) :-

· Used in electric grids & utilities
· Works over long distances.
Supports : · Time stamped data.
Event reporting.

4) IEC 60870

· Used in power systems.
· For transmission of telemetry data.
· Reliable communication protocol.

(5) BACnet :-

· Mainly used in Building Automation.
· Controls : ° HVAC , Lighting , Fire Alarm Systems

RFID Protocols ⇒ RFID (Radio Frequency Indentification) uses radio waves to identify objects automatically.

It consists of :

- RFID Tag
- RFID Reader
- Backend System

Used in : • Inventory, ID Cards, Toll collection, Asset tracking.

Examples :

1. EPC Global Protocols :

Used in supply chain systems :

- EPC Class -1 Gen-2 standard.
- Used by UHF RFID tags.
- Manages tag - reader communication.

2. ISO RFID Standards :

Standard	Type
ISO 14443	Contactless smart cards
ISO 15693	Vicinity cards
ISO 18000	RFID air interface standards.

3. NFC Protocol (Near Field Communication):

- Short range.
- Used in:
  - Debit cards
  - Mobile payments
- Based on ISO/IEC 14443

4) Reader Communication Protocols

- LLRP (Low Level Reader Protocol).
- ALE (Application Level Events)

These handle:

- Communication between reader and server.
- Data processing.

Issues with IOT Standardization :-

(1) Lack of Universal Standards :-

- Different organizations make different Standards.
- No single protocol is accepted worldwide.
- Causes compatability problems between devices.

(2) Interoperability Problems :-

- Devices from different manufactures cannot communicate easily.
- Different protocols and formats create integration issues.

(3) Security and Privacy Issues :-

- Weak security in many IoT protocols.
- Data can be stolen or hacked.
- Lack of standard security guidelines.

(4) Scalability Issue :-

- Some standards are not suitable for large networks.
- Hard to manage millions of devices.

(5) High Implementation Cost :-

Supporting multiple standards increase hardware and software cost.

Unified Data Standards :-

Unified data standards define how data is

- Collected
- Stored
- Shared
- Understood

in a common format across all IOT system.

Purpose :-

To make data exchange easy between different IOT platforms.

Key features :-

(a) Common Data Format :-

eg. • JSON, XML, CBOR, BSON

(b) Data naming rules :-

Defines : • Standard names
- Units of measurement
- Field Structure

© Semantic Standards :-

eg :-

- Sensor data must have
  - Type
  - Timestamp
  - location

(d) Open APIs :-

So applications can read and write data easily.

(e) Data models :-

eg:

- Device
- User
- Sensor
- Event

Have fixed schema.

⇒ IEEE 802.15.4

IEEE 802.15.4 is a low power wireless communication standard used for short-range communication in IOT networks.

Features :-

- low data rate (20–250 kbps)
- low power consumption
- Short range (10–100m)
- Supports star and mesh topology.

Used by Zigbee, 6LoWPAN, Thread, WirelessHART.

BACnet Protocol ⇒

BACnet (Building Automation and Control Network) is used for building automation systems like:

- HVAC
- Fire Alarm.
- Lighting
- Security systems.

Features :-

- Open and vendor - independent
- Supports Ethernet, IP, MS/TP
- Object-based communication.

⇒ Modbus Protocol ⇒

Modbus is an industrial communication protocol used in SCADA and PLCs.

Types:

- Modbus RTU (Serial)
- Modbus ASCII
- Modbus TCP (Ethernet)

Communication Type → Master-Slave (Client-Server).

KNX Protocol :-

KNX is an international standards for home and building automation.

Controls

- Lighting
- HVAC
- Blinds
- Security

⇒ Zigbee Protocol ⇒

Zigbee is a low-power wireless protocol based on IEEE802.15.4.

Features :-

- Mesh networking
- Very low power
- Secure Communication
- Short range.

⇒ Network Layer (Zigbee) :-

Role: The network layer handles:

- Routing
- Addressing
- Network formation
- Device joining / leaving.

Functions → route discovery, packet forwarding, Network security.

⇒ APS layer (Application Support layer) :-

Role:

APS layer acts as a bridge between

- Network layer
- Application layer

Functions: —

- Data binding
- Group management
- frame formatting
- Security management.

⇒ Security in IOT (Zigbee / Protocol Level) :-

Security methods :-

i) **Encryption** :-

- AES-128 encryption used
- Protects data from hackers.

ii) **Authentication** :-

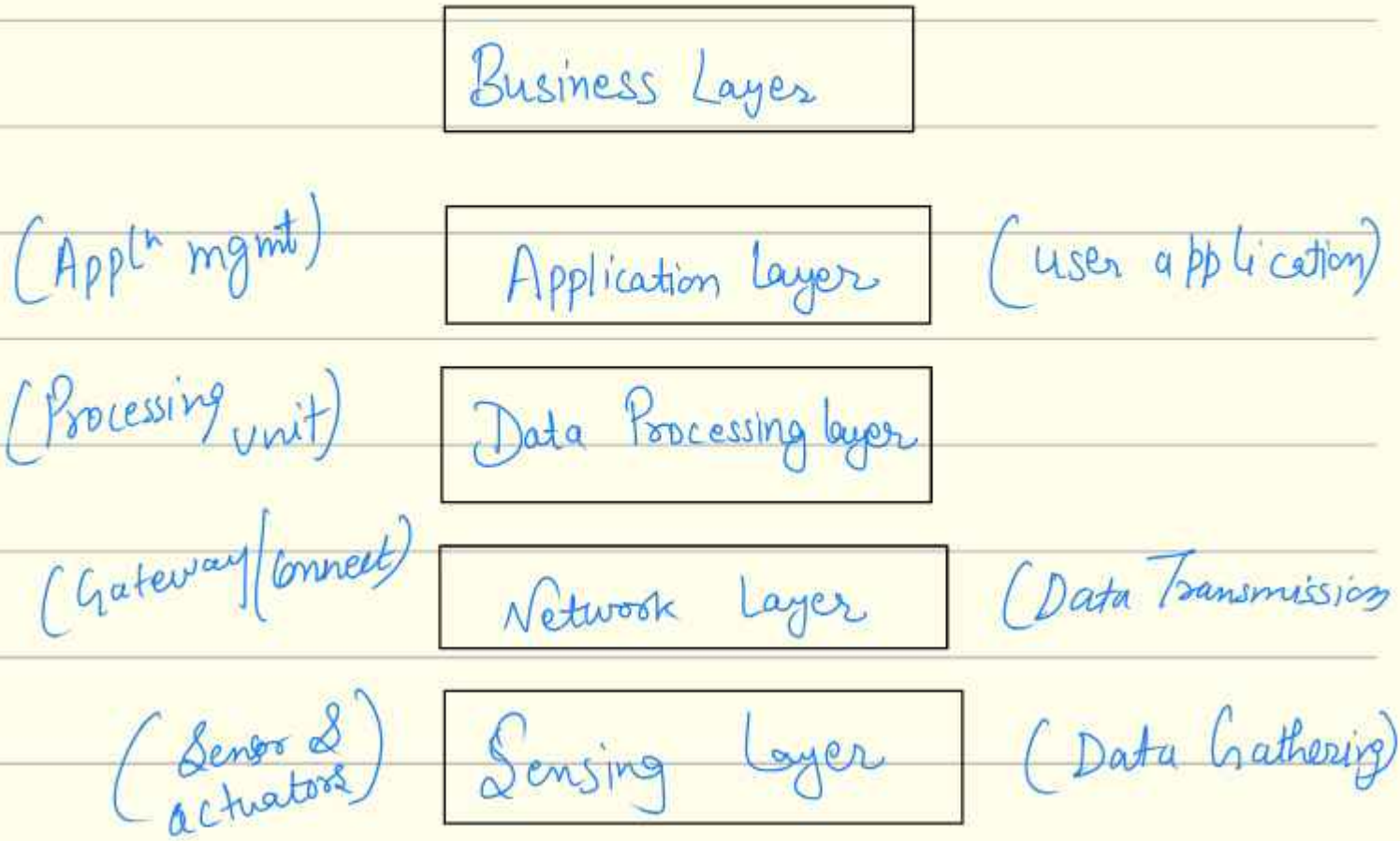
- Verifies device identity

iii) **Key management** :-

- Network layer
- Link key
- Master key.

iv) **Data Integrity** :- Prevents modification during transfer.

Architecture of IOT ⇒



Layers of IOT Architecture ⇒

(I) Device Layer (Perception Layer) ⇒

- Sensor and Actuators
- Collect real - world data.

(II) Network Layer ⇒

- Transfers data from devices to server.
- Uses protocols : Wi-Fi, Zigbee, MQTT, CoAP

(III) Processing Layer :-

- Stores data
- Performs Analytics
- Data management
- Decision making

(IV) Application Layer :-

- User Interface.
- Control Dashboard
- Mobile / web apps

(V) Business Layer (Optional) :-

- Reports
- User management
- Automation rules.

IOT Open Source Architecture :-

IOT open source architecture is a design framework for IOT systems where software protocols, and platforms are freely available to use, modify and distribute.

It allows developers to build IOT systems using open standards and community developed tools instead of vendor-locked platforms.

Purpose :-

- Enable interoperability
- Reduce cost
- Avoid vendor lock-in
- Promote innovation.

Layers of IOT Open Sourse Architecture

(I) Device Layer :- Includes : • Sensors, • Actuators, • Microcontrollers (ESP32, Arduino).

Purpose - Collect real-world data, Perform

(II) Connectivity layer → Provides network communication

MQTT, CoAP, HTTP, WebSockets, Bluetooth, Zigbee, Wifi

(III) Middleware / Platform Layer :-

Acts as a bridge between devices & applications

Examples:

- ThingsBoard, IOTivity, Node-RED, Kaa IOT, OpenHAP

Functions:

- Device management
- Data collection
- Protocol translation
- Rule processing

(IV) Data Management Layer

Handles:

- Data Storage
- Processing
- Analytics

Tools:
- InfluxDB
- MongoDB
- Apache Kafka
- MySQL

⑤ Application Layer :-

Frontend and user interface:

- Web apps
- Mobile apps
- Dashboards.

• OIC Architecture and Design Principles :-

(OIC = Open Interconnect Consortium, now called OCF - Open Connectivity Foundation)

OIC Architecture :- OIC architecture provides a standard framework to allow different IOT devices and platforms to communicate & work together.

• Main Component of OIC Architecture :-

① IOT Devices (Things)

- Sensors, actuators, appliances
- Expose their features as resources.

Example:

- Bulb → / light
- Sensor → / temp

(II) Resource Layer :-

Each device exposes data as a resource :

- URI
- Resource type
- Attributes
- Interfaces.

(III) Communication Layer :-

OIC uses :

- RESTful communication.
- Protocols : CoAP, HTTP, MQTT
- Methods : GET, POST, PUT, DELETE.

(IV) Security Layer :- Provides

- Authentication
- Authorization
- Encryption
- Secure-onboarding.

(V) Cloud Interface :- Connect devices to Cloud services for :

- Storage
- Analytics.
- Remote access.

(VI) Application layer :-

User apps control devices using standard APIs.

OIC Design Principles :-

(I) Interperability :- Devices from different brands must work together.

(II) Resource-Based Model :- Everything is treated as a resources.

(III) RESTful Architecture :-

Uses web-style access for devices :

- GET → Read data.
- POST → Create
- PUT → Update
- DELETE → Remove.

(IV) Security By Design :-

Built in : • Encryption
• Access Control List (ACL)
• Device authentication

(V) Device Discovery :-

Devices automatically discover each other on the network.

(VI) Platform Independence :-

Works on : • Linux, • Android, • Windows
• Embedded Systems.

(VII) Scalability :-

Supports small networks to millions of devices.

(VIII) Data Abstraction :-

Hardware is hidden behind software models

⇒ IOT Devices ⇒ IOT devices are physical objects that collect data, communicate over a network, and performs actions.

Types of IOT Devices :-

i) Sensors

Collect data such as:

- Temprature
- Humidity
- Motion
- Light
- Pressure

ii) Actuators :- Perform actions:

- Turn ON/OFF devices.
- Control motors.
- Open / close valves.

iii) Smart Devices :-

Built-in processing + communication.

- Smart bulbs
- Smart AC
- Smart wearables.

iv) Gateways :

Bridge between :

- Devices and internet / cloud
- Convert protocols

eg → · Raspberry Pi
· Routers
· Edge devices.

Deployment models in IOT ⇒ describes how IOT devices send data, where data is processed and how systems are connected.

(I) Device – to – Device (D2D) Model

Devices communicate directly without internet or cloud.

Both devices are in the same network.

Use case:

- Bluetooth speakers.
- Smart watch controlling light.

Advantages:
- No internet required
- Fast communication
- Simple setup

(II) Device to Gateway model :

Devices sends data to a central gateway.

The gateway :

- Collects data
- Converts protocol
- Sends data to the cloud.

- Multiple devices connected
- Less load on cloud.
- Gateway can filter data.

(III) Device to Cloud model :-

Devices connect directly to cloud servers without gateway

Device → Internet → Cloud → App

Advantages :

- Easy access from anywhere.
- Data stored safely.
- Easy management.

eg :-

- Smart camera upload videos.
- Fitness band sends data online.
- Smart fridge.

(IV) Backend Data Sharing Model :-

Data is shared between multiple cloud systems.

Device → CloudA ↔ CloudB → Application.

Example :

- Traffic system sharing data with weather app
- Smart city database linking.

Advantages :

- Data reuse.
- Better decisions.
- large scale applications

IOT Stack ⇒ An IOT stack is a set of software layers that works together to:

- Connect devices
- process data
- store data.
- Show info to user.

Open Sourc IOT stack →

An open source IOT stack is a collection of IOT software components whose source code is freely available to use, modify, share.

Layers of an Open Source IOT Stack →

(I) Device Layer →

Includes: • Sensors, • Actuators • Microcontrollers (ESP32, Arduino)

Role: • Collect data from environment
• Perform actions.

(III) Connectivity Layer :-

Enable communication.

Protocols: MQTT, CoAP, HTTPS, WebSockets, ZigBee.

(III) Gateway / Edge Layer :-

Acts as middleman.

Functions:

- Data aggregation
- Protocol translation
- Local processing.

Tools: • Node-RED
• Edge X Foundry.

(IV) Middleware / Platform Layer :-

Manages devices and data.

Open source platforms:

- IOTivity
- Things Board
- OpenHAB
- Kaa IOT

Functions:

- Device management.
- Data Routing
- Rule engine.

(V) Data Management Layer :-

- Stores and analyzes data.

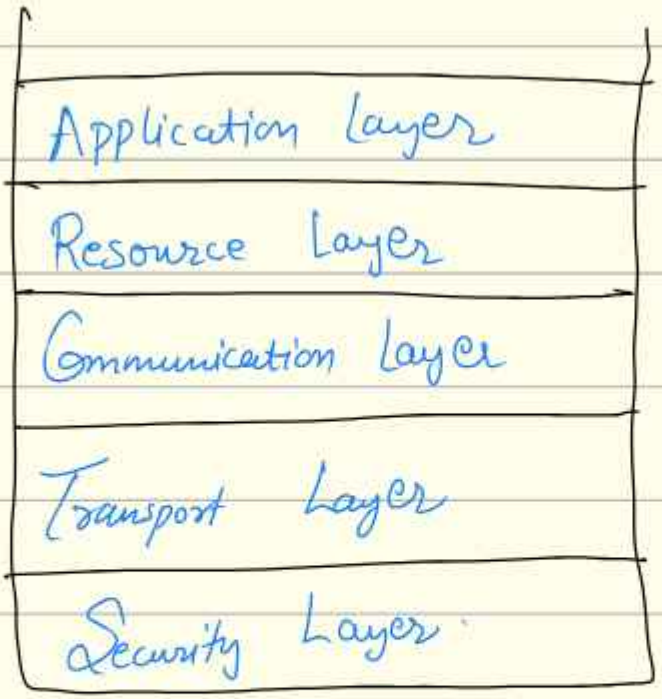
eg: InfluxDB, MongoDB, MySQL.

(VI) Application Layer :-

User Interfaces.

eg:
- Web dashboards.
- Mobile apps.

Examples : IoTivity, ThingsBoard, Node-RED, Edgex Foundry.



(I) Application Layer :- This layer contains the user applications that control and monitor IOT devices.

Functions :-

- Sends commands
- Display sensor data.
- Uses IOTivity APIs
- Controls devices through REST calls.

Eg :- Mobile app to control lights and AC.

(II) Resource Layer :-

Represents everything as a resource (light, fan, tempSensor).

Functions:

- Creates and manages resources.
- Assigns URI and attributes.
- Handle GET, POST, PUT, DELETE requests.
- Resource discovery

(III) Communication Layer :-

Handles message exchange between devices.

Functions:

- REST - based Communication

- Request / response handeling
- Message formatting
- Device discovery.

Protocols used:
- CoAP
- HTTP.

(IV) Transport Layer ⇒ Handles actual data transfer on network.

Functions:
- Packet transmission
- Network interface support.

Technologies ÷ • TCP/IP, • UDP, • Wi-Fi, • Bluetooth
• Ethernet

(V) Security Layer ÷
Protects data and devices.

Functions:
- Authentication
- Encryption
- Authorization.
- Secure Onboarding
- Key management

Security tools:
- DTLS/TLS
- Access Control Lists (ACLs)
- Certificates.

Resource model and Abstraction :-

Resource Model :-

Resource :-

In IOT resource is any functional unit exposed by a device.

It could be

- A sensor reading (like temp or humidity)
- A control action (like turning a light on/off)

Resources are the interface between the physical world and digital world.

Resources are accessed via std interfaces often RESTful.

Resource Model Explanation :-

The resource model defines how these resources are represented and accessed.

Structur of a Resource :-

Each resource has

Part	Meaning
URI	Unique address of resource
Resource type	Type of device
Interface	How resource is accessed
Properties	Data values.

Operations on Resources :- Uses REST methods

Method	Action
GET	Read
POST	Create
PUT	Update
DELETE	Remove

Advantages of Resource model :-

- Standard way to access devices
- Easier integration.
- Interoperability between vendors
- Simple programming.

Resource Abstraction :-

Resource abstraction hides hardware details and allow all devices to be accessed in the same

Purpose :-

- Single interface for all devices.
- No need to know internal structure
- Easy to program.
- Platform independent.

How it works :-

Diff devices expose resources in same format.

eg:

Bulb from Company A → /light
Bulb from Company B → /light

Both accessed in the same way, even if internally different

Benefits of Resource Abstaction :-

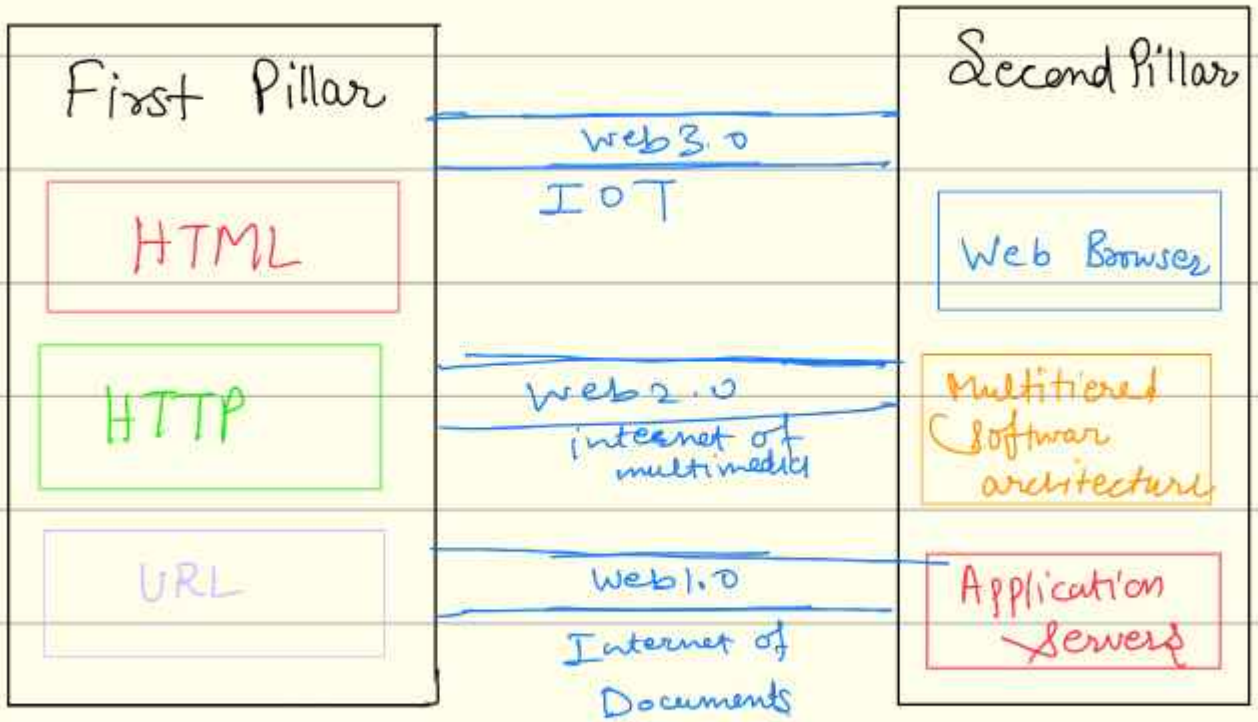
- Code is reusable
- Device replacement is easy.
- Faster development
- Reduced Complexity.

Diff between IOT and WOT :-

IOT	WOT
1. (Internet of Things)	1. (Web of Things)
2. IOT is the concept of connecting physical devices to the internet	2. WOT is the concept of connecting IoT devices to the web.
3. IOT focuses on collecting data from sensors & devices.	3. WOT focuses on accessing and controlling devices through web technologies

4. It uses protocols like MQTT, CoAP and Zigbee for communication.	4. It uses web protocols like HTTP and REST for communication.
5. IOT systems usually require special apps or software for control.	5. WoT systems can be accessed easily using web browsers and web applications.
6. IOT mainly deals with device to device and device to cloud communication	6. WOT mainly deals with device to device and web-to-device communication.
7. IOT is more hardware focused and system oriented.	7. WOT is more software-focused and web oriented.
8. IOT is used in industrial automation, smart homes and healthcare systems.	8. WoT is used to make IOT devices usable through normal web platforms
9. eg:- More Complex. Eg: Smart meter	9. Simple for web dev. Eg: Web API to control meter

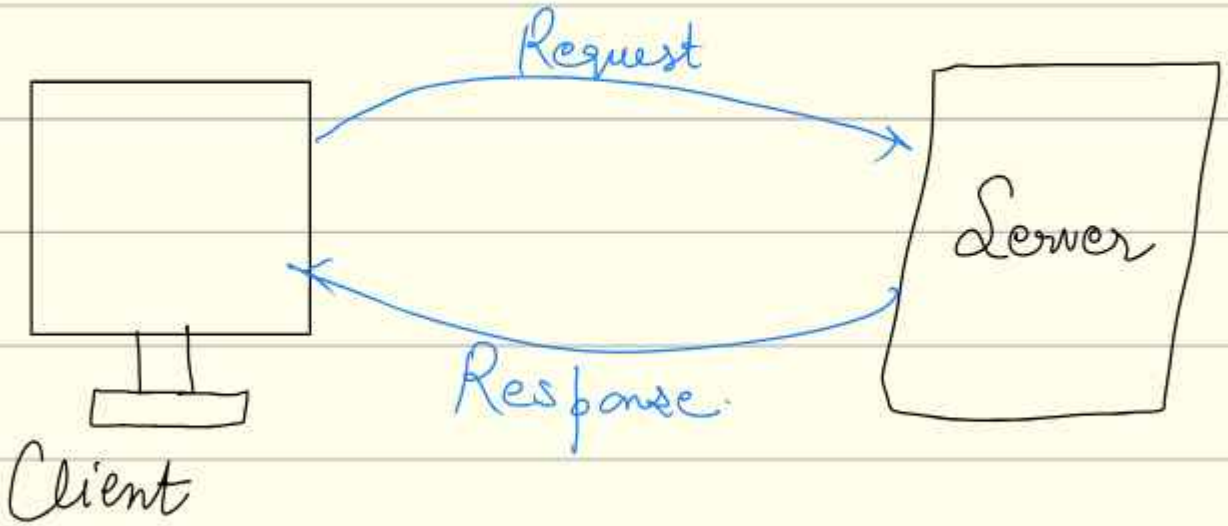
Two Pillars of Web

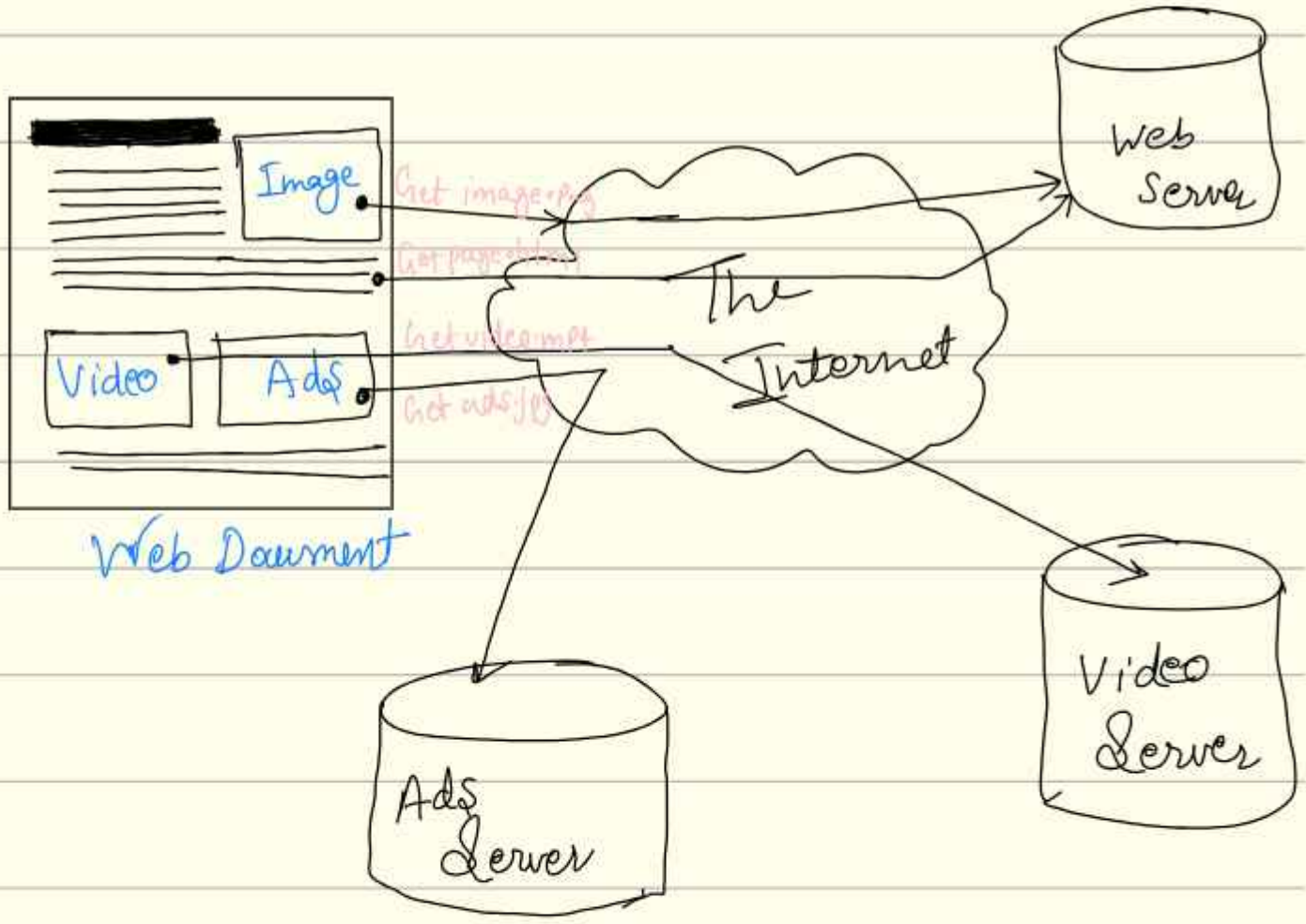


- First pillar focuses on Describing service, how to locate and how to response.
  - HTML (Hypertext Markup Language) → Create Resources
  - HTTP (Hypertext transfer protocol) → Define the format to transmit message
  - URL (Universal Resource Locator) → locate Resources.

- Second pillar focuses on way to access resource how to design web application and how to run them.
  - Web Browsers → way to access resource using URL
  - Multitiered Architecture → how to design web applications.

- Application Servers → How to run web-applications i.e database connectivity.





HTTP is a protocol which allows the fetching of resources, such as HTML documents.

It is the foundation of any data exchange on the web and a client server protocol, which means requests are initiated by the recipient, usually the web browser.

A complete document is reconstructed from the different sub-documents fetched, for instance text, layout description, images, videos, scripts & more.

Architecture Standardization for WoT ⇒

⊙ Flexibility ⇒

- There is a wide variety of physical devices configuration for WoT implementations.
- Functional WoT architecture should be able to to be mapped to and cover all of the variations.

⊙ Compatibility ⇒

- Functional WoT architecture should provide bridge between IoT solutions and Web technology based on WoT Concepts.

- It should guarantee to be upper compatible to IoT solutions and current standards.

⊙ Security and Privacy ⇒

- Functional WoT architecture should have the room for providing security & privacy functionalities.
- In the IoT solutions, once cyber security barrier is hacked, it is more easily led to safety and/or privacy issues than conventional web solutions.

⊙ Scalability ⇒ Scalability means the WoT system can grow easily when more devices are added

Explanation :

Architecture standardization allows millions of devices to be added without changing the whole system. New devices can join easily and work with existing applications.

⊙ Interoperability ⇒ Interoperability means different systems and platforms can communicate smoothly

Explanation ÷ WoT uses common web standards, so devices, services, and platforms can exchange dat easily even if they use different technologies.

Platform Middleware for WoT ⇒

Platform middleware for WoT is the software layer that works between IoT devices and web applications.

It helps connect devices to the web and makes different technologies work together easily.

In simple words:

Middleware is the bridge between devices & web applications.

Main Functions of WoT Middleware ÷

(I) Device Discovery and Registration ÷ It finds new devices automatically and registers them on the platform.

(II) Data management :- It collects, stores, and forwards data to applications.

(III) Protocol Translation :- It converts IoT protocols (MQTT, CoAP) into web protocols (HTTP, WebSocket).

(IV) Security Services :- It provides authentication, encryption, and access control for safe communication.

(V) Device management :- It checks device status, update software, and controls settings.

(VI) API Management :- It provides APIs so developers can easily create web applications.

Unified Multitier WoT Architecture ⇒

The Unified multitier WoT Architecture is a layered model that integrates IoT systems with web technologies to create a standardized, intelligent and user-friendly environment.

It ensures the sensors, devices, applications, and users can all communicate through a framework.

Why Unified Architecture ⇒

Different IoT systems often use different protocols, data formats, and communication methods.

This causes interoperability problems - devices from different brands can't talk easily.

A unified multitier WoT Architecture provides a common framework that brings together:

- Various IOT protocols (like MQTT, CoAP, HTTP)
- Different data models.
- And Cloud or web platforms.

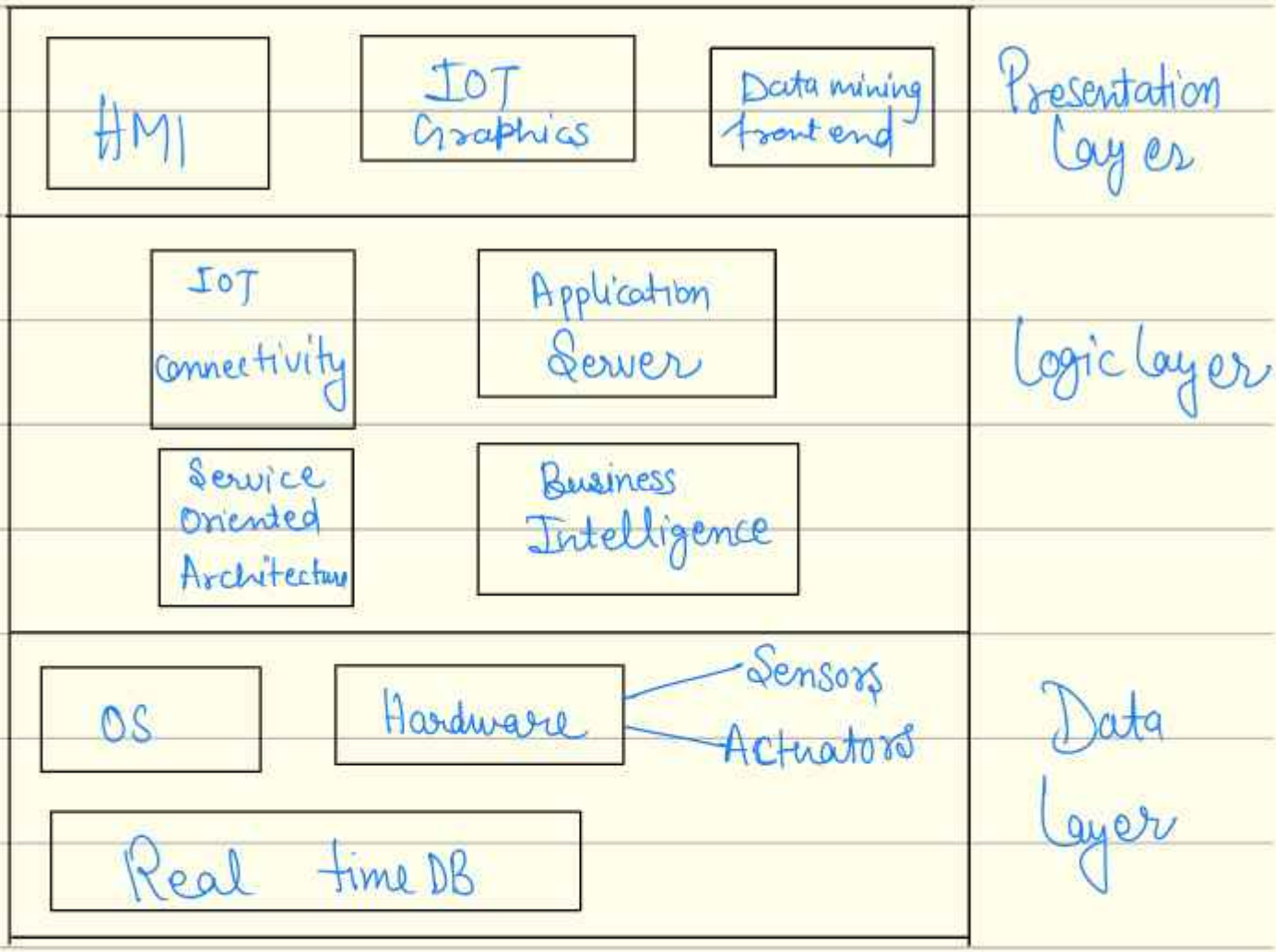
So all devices and services can work together smoothly

Structure :-

The architecture is divided into three major layers

1. Data layer.
2. Logic layer.
3. Presentation layer.

Each layer performs a specific fun- starting from data collection to user presentation



1. Data Layer (Foundation layer) :-

Purpose :- To collect and store real-time data from physical world.

Components :

Sensor & Actuators :- Measure and control environmental parameters (eg, temp, humidity, motion).

eg: A temp sensor reads the room temp, while an actuator turns on a fan.

Hardware :- Microcontrollers or embedded boards (eg. Arduino, ESP32, Raspberry Pi) that process raw signals from sensors.

Operating System (OS) :- Controls device-level functions and communication between hardware & software.

Real-Time DB :- Stores sensor data and system status instantly for a quick access and analysis.

eg :- The temp sensor measures 30°C → microcontroller sends it stored in real-time db.

2. Logic Layer (Middle Layer) :-

Purpose:-

To process, analyze, and manage data from database.
It acts as the intelligent brain of the system.

Components :

- IOT connectivity :- Handles network communication and data transmission.

Supports multiple protocols like HTTP, MQTT, CoAP and WebSockets to connect IoT devices with servers.

- Service - Oriented Architecture (SOA) :- Defines how diff. services interact using standard interfaces. It allows different devices or systems to communicate regardless of manufacturer or protocol.
- Application Server :-

Runs backend services, executes application logic, and connects data from sensors to applications.
For eg. It can decide "if temp > 30°C → Turn on the fan".

- Business Intelligence (BI) :-

Performs data analysis, visualization, and decision-making. It uses AI or analytics to identify patterns and generate insights.

eg:- The system detects high temp → BI analyzes patterns → send command to turn on fan.

3. Presentation Layer (Top Layer) :-

Purpose :-

To present data and insights to users in an understandable form.

This layer provides the interface for user interaction.

Components :-

- HMI (Human-Machine Interface) :- Displays system data and allows control through dashboards, screens, or apps.

eg:- A mobile app showing live temperature and control buttons.

- IoT Graphics :-

Represents IoT data visually using graphs meters, and charts for easier understanding.

- Data mining frontend :-

Provides advanced analytics, trends, and predictive analysis through machine learning and data mining tools

eg :- The farmer checks a dashboard that displays temp trends and predictions for irrigation timing.

WoT Portals and Business Intelligence

WoT Portals ⇒

WoT portals are web-based dashboards that allow users to :

- Monitor devices.
- Control appliances
- View Sensor data
- Manage users

Example Use :- Smart home portal, factory dashboard, city monitoring website.

Features of WoT Portals :-

- Login system
- Real time data view
- Control buttons
- Alert System
- Reports

Business Intelligence (BI) in IOT →

Business Intelligence means analyzing IOT data to help companies make better decisions.

What BI does :

- Converts raw sensor data into charts.
- Show trends.

- Predicts failures.
- Improves performance.

Uses:

- Predict machine failure
- Reduce cost.
- Increase production
- Improve quality.

IOT Applications for Industry :-

a) Future factory concept (Smart factory) :-

Future factories use IoT to make machines:

- Smart, Automatic, Self-monitoring, Efficient

Features :-

- Real-time monitoring
- Automated control
- Predictive maintainance
- Energy management.

(b) Brownfield IoT :-

Brownfield IOT means installing IoT systems on old factories and machines.

Benefits :-

- No need to replace machines.
- Save money.
- Improves monitoring.

(c) Smart Objects :-

Smart Objects are daily use objects with:

- Sensors
- Processing power
- Internet

Example :- Smart machines, RFID tags, trackers.

(d) Smart Applications :-

Smart applications use IoT data to:

- Take decisions
- Show insights.
- Automatically control systems.

eg :-
Smart energy management
production control apps.

Study of Existing IoT Platforms/Middlewares :-

(a) IoT-A :-

IOT-A is a research project that designed a reference architecture for IOT.

Focus:
- Standard models
- Interoperability
- Architecture design.

(b) Hydra :-

Hydra is an open-source middleware plat.

Provides :-

- Device management
- Data management
- API services
- Cloud connectivity.