UNIT - 1

1) Define the Internet of Things.(PYQ)

A) "The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction."

OR

The Internet of Things (IoT) describes the network of physical objects—"things"—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.

2) What are the characteristics of IoT ? (IMP) (PYQ)

A) Characteristics of IOT:

1. Connectivity – It is an essential feature of IoT. IoT lets you connect mobile phones, laptops, and other internet devices. Any person can get information about anything at any time and place. Also remote connections are possible.

2. Dynamic and Self-Adapting – IoT devices and systems have capability to dynamically adapt with the changing contexts and take actions based on their operating conditions, user's context or sensed environment.
Eg:- Surveillance system

3. Self Configuring – IoT devices have self-configuring capability, allowing a large number of devices to work together to provide certain functionality. (Weather Station)
These devices have the ability to configure themselves.

4. Integrated into the information network – IoT devices are usually integrated into the information network that allows them to communicate and exchange data with other devices and systems.

5. Interoperable communication model – IoT devices can communicate with other devices that are present in the same network or different network.

6. Unique Identity – Each IoT device has a unique identity and unique identifier (IP Address).
IoT devices interfaces allow users to query the devices, monitor their status, control them remotely in association with the control, configuration and management infrastructure.

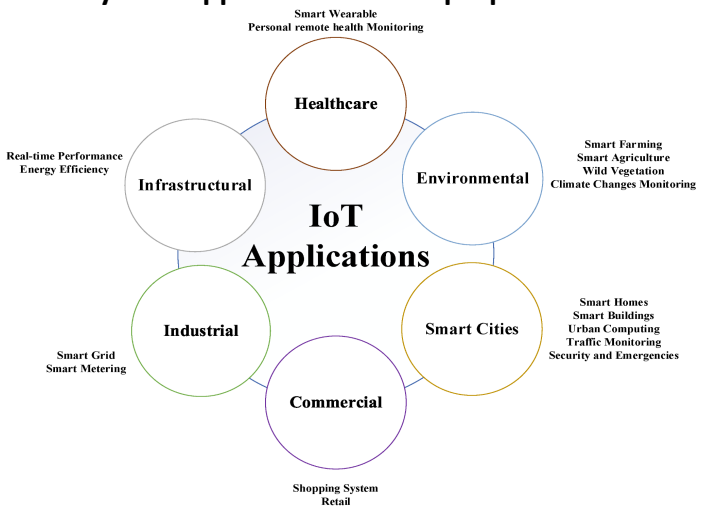
7. Intelligence – The intelligence of IoT devices is the intelligence of smart sensors and devices to sense data, interact with each other and collect a massive amount of data for analysis.
Eg:- Alexa, Siri, Google Assistant etc.

3) List the applications of IoT in various fields. Explain any one application with a proper example. (IMP)

OR

Discuss Application of Internet of things in detail.

1. **Home Automation**
2. **Environment Monitoring**
3. **Energy Management**
4. **Agriculture**
5. **Industry**



A)

1. Home Automation:

- Home automation is constructing automation for a domestic, mentioned as a smart house. In the IoT home automation ecosystem, you can control your devices like light, fan, TV, etc.
- A domestic automation system can monitor and/or manage home attributes adore lighting, climate, enjoyment systems, and appliances.
- Smart lighting helps in saving energy by adapting life to the ambient condition.

2. Environment Monitoring: The applications of IoT in environmental monitoring are broad – environmental protection, extreme weather monitoring, water safety, endangered species protection, commercial farming, and more.

- **Air and Water Pollution:** Current monitoring technology for air and water safety primarily uses manual labor along with advanced instruments, and lab processing. IoT improves on this technology by reducing the need for human labor.
- **Commercial Farming:** IoT systems detect changes to crops, soil, environment, and more. They optimize standard processes through analysis of large, rich data collections and also prevent health hazards (e.g., e. coli) from happening and allow better control.

3. Energy Management: IoT-based energy management systems provide a wide range of benefits for every part of the electricity supply chain, including electric utilities, distributors, & consumers. The benefits of using IoT technology for energy management and conservation are:

a) Minimize carbon emission: Companies increasingly integrate IoT energy consumption and management software and other solutions to their operations to decrease their carbon footprint; optimizing the use of resources.

b) Integrate green energy: Using energy monitoring sensors, performance and power consumption data, utilities, for instance, better understand how to maximize the use of renewables in their services

c) Automate processes: Using IoT-based monitoring systems, for example, producers automate costly on-site asset management and improve maintenance operations.

4. Agriculture: In order to meet the current and future farming needs, farmers should use smart methods and techniques. IoT offers many such smart techniques like its automation feature that helps the farmers to fertilize their plants at regular intervals, keep a check on the usage of water, and be aware of the right time to harvest. analyze the soil's texture, nutrients, and also its ability to yield. In order to predict the weather before/during/after a yield, one can use AIIMETOE / Pynco, the **2 best IoT farming devices.**

5. Industry:

- IoT applications deal with developing the industry and it's working methods with the help of software used for data analysis, sensors, tracking devices, and machines that are effective and masterful.
- These help a firm to have accurate, enhanced, and transparent functioning. One can not only improve things but can also identify the damaged spots for an accurate cure.
- Other industrial applications are: Quality control, Supply chain optimization, Plant safety improvement and many others.

4) Give the difference between IoT and M2M. (IMP) (PYQ)

M2M	IOT
Abbreviation for machine to machine	Abbreviation for internet of things
It is about direct machine to machine communication	It is about sensor automation and internet platform
It support point to point communication	It support cloud based communication
Device not necessary relay on internet	Device necessary relay on internet
It mostly based on hardware	It based on both hardware and software
Machine normally communicates with single machine at a time	Many user can access at a time over internet
It uses either proprietary or non IP based protocols	It uses IP based protocols
Its for only B2B business type	Its for B2B and B2C business type
Limited number of devices can be connected at a time	More number of devices can be connected at a time
It does not support open API's	It supports open API's
It is less scalable	It is more scalable

5) Draw and Explain IoT Stack. (IMP)

A) IoT Stack:

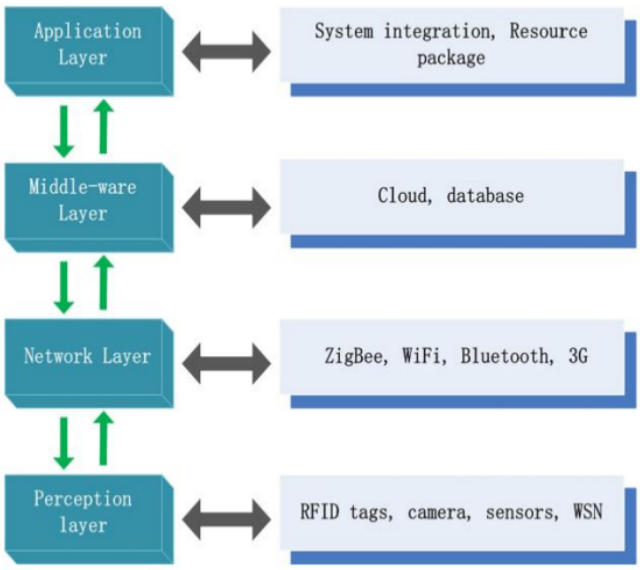
1) Application Layer: This is the topmost layer of the IoT stack, it depends on the specifics of the different implemented IoT applications, such as smart industry, building, city, and health applications.

2) Middleware Layer: The middleware layer obtains data from the network layer, links the system to the cloud and database, and performs data processing and storage.

3) Network Layer: This layer provides communication infrastructure for the IoT devices. It includes protocols such as Wi-Fi, Bluetooth, Zigbee, and cellular networks.

4) Perception Layer: The perception layer is where the IoT devices sense and collect data from the physical world. It includes sensors, actuators, and other devices that interact with the environment.

5) Physical Layer: This is the lowest layer of the IoT stack, which includes the **physical components** of the IoT device. It includes hardware such as **microcontrollers, processors**, and power sources.



6) Write a short note on Physical Design of IoT Application

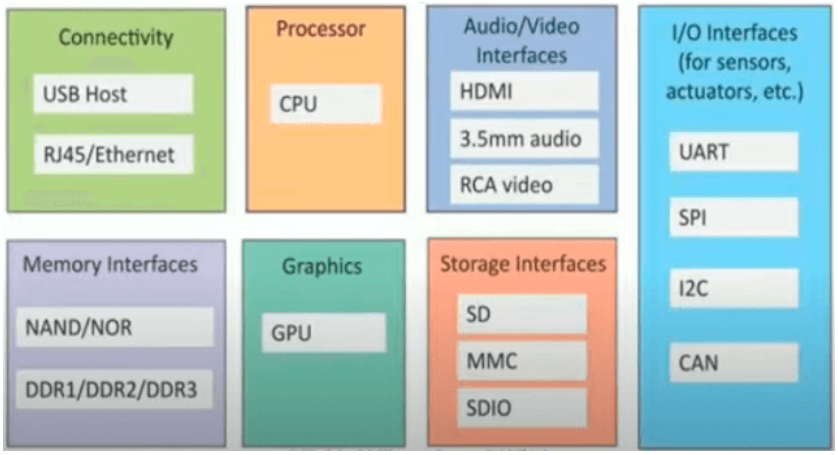
OR

List and Explain Component of IoT Application. [CPAIDG]

A) A physical design of an IoT system refers to the **individual node devices** and their **protocols** that are utilized to **create a functional IoT ecosystem.**

The "Things" in IoT usually refers to **IoT devices** (Node Devices) **having unique identities** and can **perform remote sensing, actuating and monitoring capabilities.**

Ex - Temperature sensor that is used to analyze the temperature generates the data from a location and is then determined by algorithms.



1) Connectivity - Devices like USB hosts and ETHERNET are used for connectivity between the devices and the server.

2) Processor - A processor like a CPU and other units are used to process the data. these data are further used to improve the decision quality of an IoT system.

3) Audio/Video Interfaces - An interface like HDMI and RCA devices is used to record audio and videos in a system.

4) Input/Output interface - To give input and output signals to sensors, and actuators we use things like UART, SPI, CAN, etc.

5) Storage Interfaces - Things like SD, MMC, and SDIO are used to store the data generated from an IoT device.

Other things like **DDR and GPU** are used to control the activity of an IoT system.

7) Write a short note on Logical Design of IoT Application. (PYQ)

A) The logical design of an IoT system refers to an **abstract representation of entities and processes** without going into the low-level specifications of implementation. it uses Functional Blocks, Communication Models, and Communication APIs to implement a system.

IoT logical design includes:

1) IoT functional blocks:

IoT systems include several functional blocks such as Devices, communication, security, services, and application. The functional blocks provide sensing, identification, actuation, management, and communication capability.

2) IoT communications models:

There are multiple kinds of models available in an IOT system that is used for communicating between the system and server, such as:

- Request-response model
- Push-pull model
- Publish-subscribe model
- Exclusive pair model

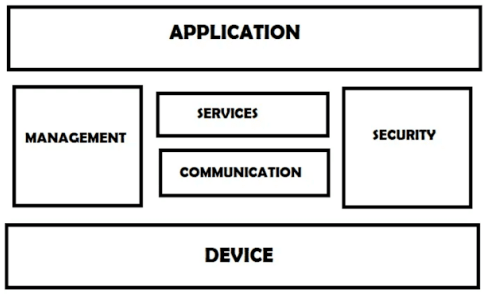
3) IoT communication APIs:

APIs are used to communicate between the server and system in IoT. Some API's include:

- REST-based communication APIs
- Client-server
- Stateless
- Cacheable
- Websocket based communication API

8) Which are the functional blocks in IoT ?

A) IoT systems include several functional blocks such as Devices, communication, security, services, and application. The functional blocks provide sensing, identification, actuation, management, and communication capability.



These functional blocks consist of devices that provide monitoring , control functions, handle communication between host and server, manage the transfer of data, secure the system using authentication and other functions, and interface to control and monitor various terms.

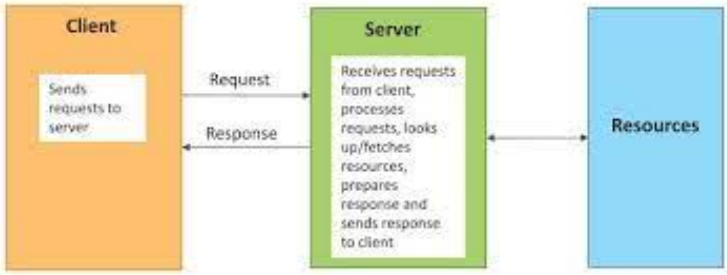
1. Application - It is an interface that provides a control system that is used by users to view the status and analyze the system.
2. Management - This functional block provides various functions that are used to manage an IoT system.
3. Services - This functional block provides some services like monitoring and controlling a device and publishing and deleting the data and restoring the system.

4. Communication - This block handles the communication between the client and the cloud-based server and sends/receives the data using some protocols
5. Security - This block is used to secure an IoT system using some functions like authorization, data security, authentication, 2-step verification, etc.
6. Device - These devices are used to provide sensing and monitoring control functions that collect data from the outer environment.

9) List and Explain Communication Models in IoT. (IMP)

A) Communication Models in IoT:

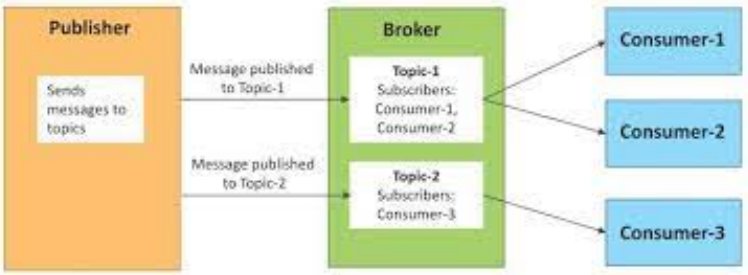
i. Request-Response Communication Model



It is a communication model in which the client sends requests to the server and the server responds to the requests.

When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client

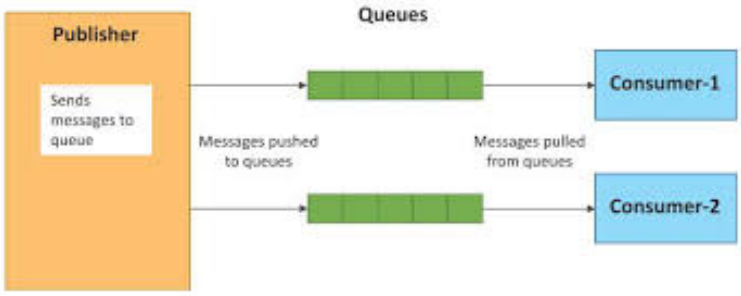
ii. Publish-Subscribe Communication Model



In this communication model, we have a broker between publisher and consumer. Here publishers are the source of data but they are not aware of consumers.

Publishers send the data managed by the brokers and when a consumer subscribes to a topic that is managed by the broker and when the broker receives data from the publisher it sends the data to all the subscribed consumers.

iii. Push-Pull Communication Model



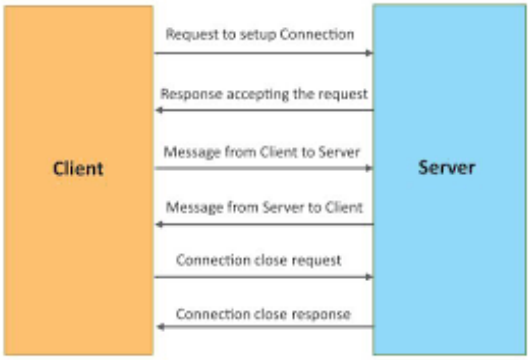
It is a communication model in which the data is pushed by the producers in a queue and the consumers pull the data from the queues. There also producers are not aware of the consumers.

Queues **help in decoupling the messages** between the producers and consumers.

Queues **also act as a buffer** which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumers pull data.

iv. Exclusive Pair Communication Model

It is a bidirectional, full duplex communication model that uses a persistent connection between the client and server.
Once the connection is set up it remains open until the client sends a request to close the connection.
Client and server can send messages to each other after connection setup.



10) List and Explain Connectivity Models in IoT. (PYQ)

A) Device-to-Device (D2D): This model involves direct communication between IoT devices without the need for an intermediary device. It is used in scenarios where low-latency communication is required, such as in industrial automation or smart homes.

Device-to-Cloud (D2C): This model involves direct connection between IoT devices and cloud services for data storage, processing, and analysis. It is used in scenarios where devices need to send data to the cloud for analysis or monitoring, such as in healthcare or agriculture.

Device-to-Gateway (D2G): This model involves communication through an intermediary device (gateway) that collects data from the IoT devices and sends it to the cloud for processing and storage. It is commonly used in smart city applications, where a large number of devices need to be connected.

Backend Data Sharing: This model involves real-time data exchange between backend systems, such as databases or cloud services. It allows multiple systems to share data and work together, enabling seamless integration and collaboration between different applications and services. It is commonly used in enterprise applications, where different systems need to share data for better decision-making.

11) Define the terms :

a) Node in IOT - A node in IoT refers to a physical device or sensor that is connected to the internet and has the ability to communicate with other devices or systems. These devices can be simple or complex, and they can gather data from the environment, process it, and send it to other devices or the cloud for further analysis.

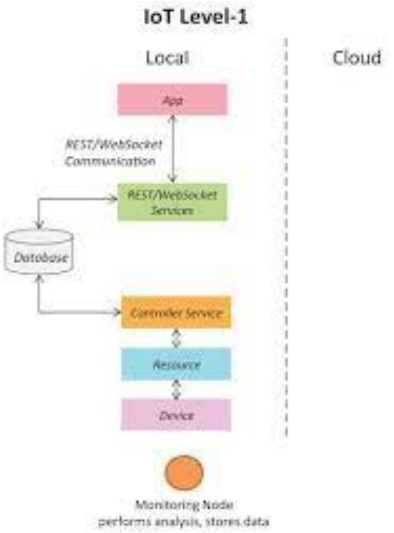
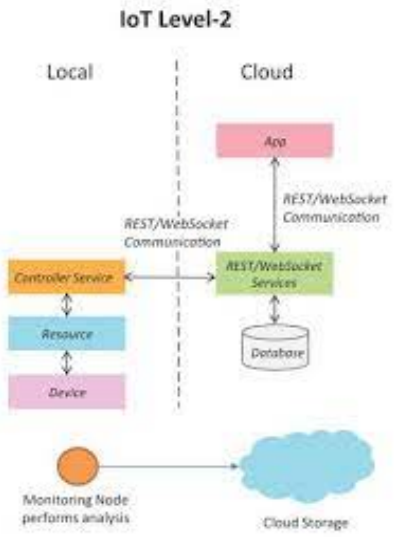
b) Gateway - A gateway in IoT is a device that acts as an intermediary between IoT devices and the cloud. It is responsible for collecting data from IoT devices, processing it, and sending it to the cloud for storage and analysis.

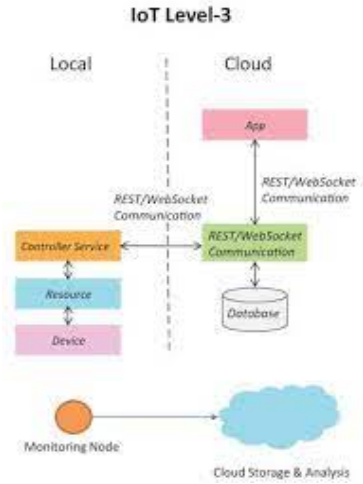
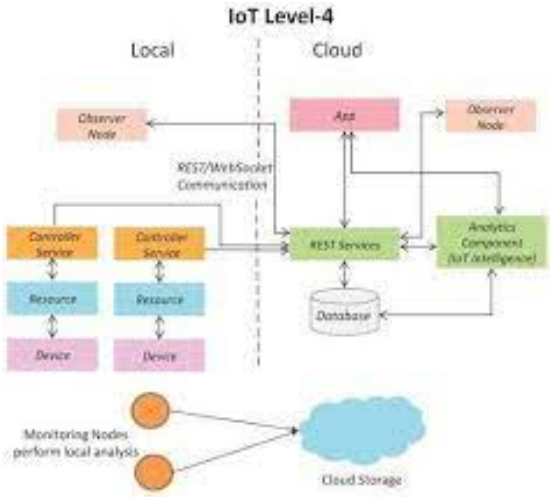
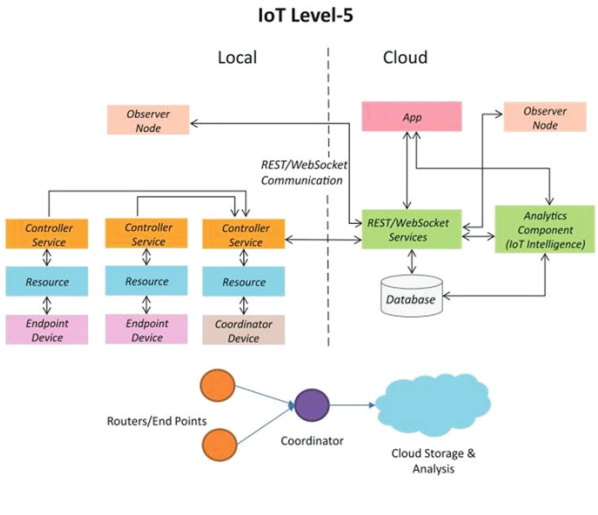
c) WSN - Wireless Sensor Network (WSN) is a network of interconnected sensors and actuators that communicate with each other wirelessly and used to monitor the system, physical or

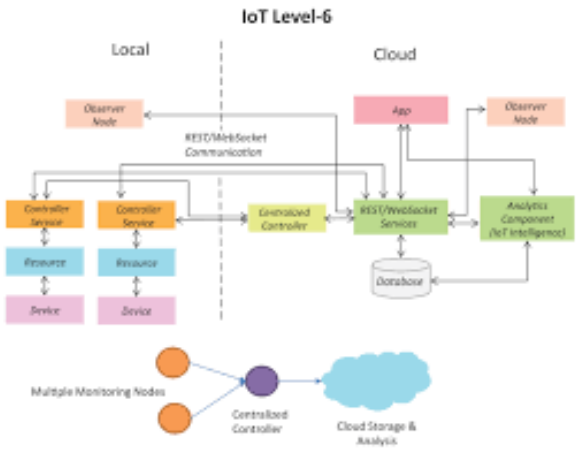
environmental conditions. WSNs are commonly used in applications such as smart agriculture, smart cities, and industrial automation

12) List and Explain Levels of IoT Applications.

A) Levels of IoT Applications:

Diagram	Description
	• A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application. • These systems are suitable for modeling low-cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.
	• A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis. • Data is stored in the cloud and application is usually cloud- based. • These systems are suitable for solutions where the data involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself.

	• A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and the application is cloud- based. • These systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive.
	• A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and the application is cloud-based. • It contains local and cloud-based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices. • These are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.
	• A level-5 IoT system has multiple end nodes and one coordinator node. • The end nodes that perform sensing and/or actuation. • Coordinator node collects data from the end nodes and sends it to the cloud. • Data is stored and analyzed in the cloud and the application is cloud-based. • These systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive

<table>
<tr>
<td> 
</td>
<td>

A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud.
Data is stored in the cloud and the application is cloud-based.
The analytics component analyzes the data and stores the results in the cloud database.
The results are visualized with the cloud-based application. The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.

</td>
</tr>
</table>

13) List and Explain Security Challenges, Design and Development Challenges in IoT.

A) Security Challenges –

- **Lack of visibility**
  Users often deploy IoT devices without the knowledge of IT departments, which makes it impossible to have an accurate inventory of what needs to be protected and monitored.
- **Limited security integration**
  Because of the variety and scale of IoT devices, integrating them into security systems ranges from challenging to impossible.
- **Open-source code vulnerabilities**
  Firmware developed for IoT devices often includes open-source software, which is prone to bugs and vulnerabilities.
- **Poor testing**
  Because most IoT developers do not prioritize security, they fail to perform effective vulnerability testing to identify weaknesses in IoT systems.
- **Weak passwords**
  IoT devices are commonly shipped with default passwords that many users fail to change, giving cyber criminals easy access. In other cases, users create weak passwords that can be guessed.

DESIGN & DEVELOPMENT CHALLENGES –

- **Scalability:** IoT solutions must be scalable to accommodate a large number of connected devices and data streams. The IoT solutions should ensure that the solution can accommodate growth without sacrificing performance.
- **Security:** IoT solutions must be secure to protect devices, data, and users from unauthorized access and cyber attacks. Security must be maintained throughout the life cycle of the solution.

- **Power consumption:** IoT devices are often battery-powered and must conserve power to prolong their battery life. Designers must balance the need for functionality with the need to conserve power to ensure that IoT devices can operate for an extended period.
- **Data management:** IoT solutions generate a vast amount of data that must be managed, processed, and analyzed. The design and development of IoT solutions should take into account data management and ensure that the solution can handle large data volumes.
- **Cost:** The cost of IoT devices and solutions must be reasonable to ensure their widespread adoption. The design and development of IoT solutions must balance functionality with cost-effectiveness to ensure that the solution is affordable for users.

14) Which are the emerging technologies that go hand in hand with IoT ?

A) There are several emerging technologies that are closely related to IoT (Internet of Things) and are likely to have a significant impact on its development and growth. Here are some of the most important ones:

- **Artificial Intelligence (AI):** The vast amount of data generated by IoT devices can be analyzed and used to train AI algorithms to recognize patterns and make predictions. This can help optimize IoT systems, reduce downtime, and improve efficiency.
- **5G Networks:** The high bandwidth and low latency offered by 5G networks can enable IoT devices to communicate with each other and with cloud services faster and more reliably, making real-time applications and services possible.
- **Blockchain:** The decentralized and secure nature of blockchain technology can help secure IoT devices and data, prevent tampering, and enable new business models based on trusted data exchange.
- **Augmented Reality (AR):** AR technology can be used to visualize data and information from IoT devices, allowing users to interact with data in real-time and gain insights that would otherwise be difficult to obtain. Ex- Metaverse
- **Quantum Computing:** It is expected to significantly improve data processing speed and efficiency, which could be particularly beneficial for IoT applications that generate large amounts of data.

These emerging technologies are likely to play a crucial role in shaping the future of IoT, enabling new applications and use cases, and improving the efficiency and effectiveness of existing ones.

15) What are the differences between Machines in M2M and Things in IOT ?

A) Machines in M2M refers to the direct communication between two machines without human intervention. In M2M, machines or devices are connected to each other, exchanging data and commands to perform tasks or processes. This communication can be wired or wireless, and can occur over short or long distances. It refers to industrial machines, equipment, or devices that communicate with each other to automate processes and optimize operations. M2M systems are often closed systems, where devices are connected in a private network. M2M systems are typically used in industrial automation, logistics, and supply chain management.

On the other hand, Things in IoT refers to the interconnection of physical devices, vehicles, buildings, and other items embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data. IoT devices can communicate with each other, as well as with humans, to perform a variety of tasks such as automation, monitoring, and control. It refers to everyday objects or devices that are connected to the internet and have the ability to collect and share data. IoT systems are often open systems, where devices are connected to the internet. IoT systems are used in a variety of applications such as smart homes, wearable devices, and healthcare monitoring.

In short, the main difference between M2M and IoT is that M2M is a direct communication between machines, while IoT involves a network of devices communicating with each other, as well as with humans. The term "things" in IoT refers to physical devices, while in M2M, the term "machines" is used to describe devices or equipment.

16) How do data collection and analysis approaches differ in M2M and IOT ?

A) M2M data is collected in point solutions and often in on-premises storage infrastructure.
In contrast to M2M, the data in IoT is collected in the cloud (can be public, private or hybrid cloud).

OR

Machine-to-Machine (M2M) and Internet of Things (IoT) are two distinct concepts, even though they are often used interchangeably. M2M refers to direct communication between machines, while IoT refers to a network of devices, sensors, and software applications that communicate with each other.

The data collection and analysis approaches in M2M and IoT differ in several ways:

1. **Scale:** IoT devices are typically distributed on a larger scale than M2M devices. This means that IoT generates larger volumes of data, which requires specialized tools and techniques for collection and analysis.
2. **Data types:** M2M devices tend to generate structured data, such as sensor readings or device status updates, while IoT devices generate a mix of structured and unstructured data, such as text, images, and video.

3. **Data transmission:** M2M devices often use proprietary protocols for data transmission, whereas IoT devices use standardized protocols such as MQTT, CoAP, or HTTP. This makes it easier to collect and analyze data from IoT devices.
4. **Data analysis:** The analysis of M2M data is often focused on identifying anomalies or trends in sensor readings or other device parameters. In contrast, IoT data analysis can involve a **wide range of techniques, such as machine learning, natural language processing, and image recognition.**
5. **Use cases:** M2M is typically used in industrial applications, such as monitoring and controlling machinery on a factory floor. IoT is used in a broader range of applications, including home automation, healthcare, transportation, and agriculture.

Overall, the data collection and analysis approaches in M2M and IoT differ based on the scale of deployment, the types of data generated, the protocols used for data transmission, the methods used for data analysis, and the specific use cases.

17) What is the function of a centralized network controller in SDN ?

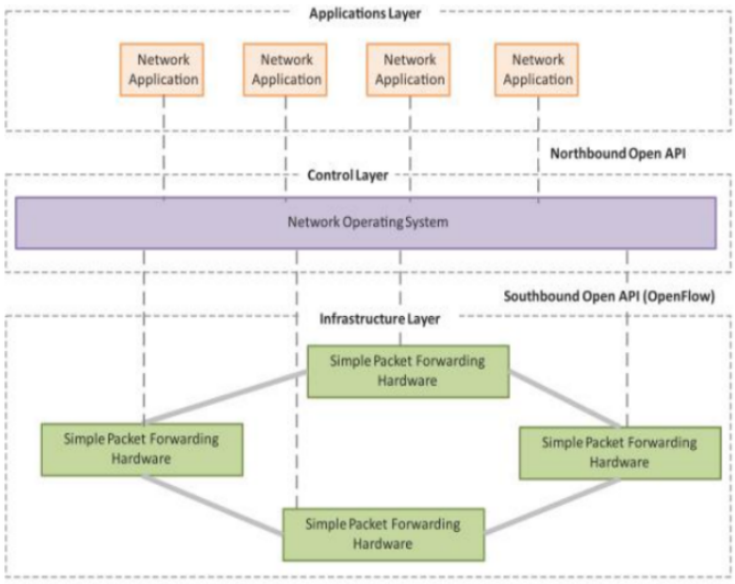
A) In a Software-Defined Networking (SDN) architecture, the network control plane is separated from the data plane and centralizes the network controller.

The centralized network controller is a critical component of an SDN, and its primary function is to control the behavior of the network devices by dynamically programming the flow tables of the switches.

The functions of a centralized network controller in an SDN include:

1. **Configuration:** The controller configures the network devices by programming the flow tables of the switches. This allows the network administrator to control the behavior of the network devices, such as defining policies for traffic management, Quality of Service (QoS), and security.
2. **Traffic engineering:** The controller monitors network traffic and optimizes the flow of traffic by selecting the best paths through the network. This enables the network to adapt to changes in traffic patterns and network topology, and helps to prevent network congestion.
3. **Fault management:** The controller detects and responds to network faults, such as link failures or switch failures, by reconfiguring the network topology and redirecting traffic along alternate paths.
4. **Security:** The controller enforces network security policies, such as access control, firewall rules, and intrusion detection, by configuring the network devices to enforce these policies.

18) Draw and explain SDN architecture.



A) Software-Defined Networking (SDN) is a networking architecture that separates the control plane from the data plane and centralizes the network controller.

SDN Architecture consists of different components. The architecture is composed of three main components: the control plane, the data plane, and the management plane.

1) Control Plane: This is the brain of the SDN architecture, where **the network policies and rules are defined and implemented**. The control plane consists of a **centralized controller that manages and controls the network devices** such as switches and routers.

In the SDN architecture, All the decision-making is done in the control plane.

2) Data Plane: This is where the **network traffic flows** and the network devices such as switches and routers forward packets according to the rules and policies defined in the control plane.

3) Management Plane: This is where network administrators can configure and manage the SDN infrastructure. The management plane **provides a user interface or APIs** for administrators to interact with the SDN infrastructure and perform tasks such as creating network policies, adding new devices, and monitoring network performance.

<u>Key elements of SDN are : -</u>

1) **Centralized Network Controller** - With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network.

2) **Programmable Open APIs -** SDN architecture supports programmable open APIs for interface between the SDN application and control layers

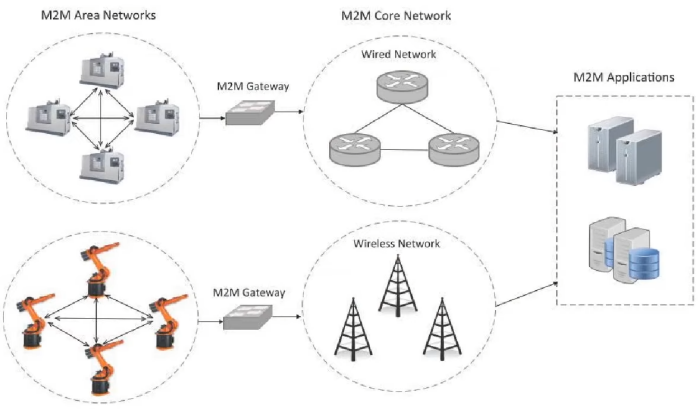
3) **Standard Communication Interface (OpenFlow) -** SDN architecture uses a standard communication interface between the control and infrastructure layers OpenFlow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocol for the Southbound interface

19) Draw and explain M2M system architecture and M2M gateway.

A)

M2M System Architecture :

To develop and deploy M2M Technology, we must follow current standard of M2M like ETSI, ANSI C12, etc



M2M architecture consists of three domains:

1) M2M Applications Domain provides application for M2M technology, such as server applications and end-user applications.

2) M2M Device Domain contains **devices that can** connect to the M2M Network domain. M2M Device Domain can be called as M2M Area Network. Diverse technologies can be used to support various applications. There are two types of devices for M2M Device Domain:

a) Devices that one capable to directly connect to the network and

b) Devices that cannot directly connect to the network domain, requires an M2M gateway in order to connect to the network.

3) M2M Network Domain provides communication network between M2M Application Domain and M2M Device Domain

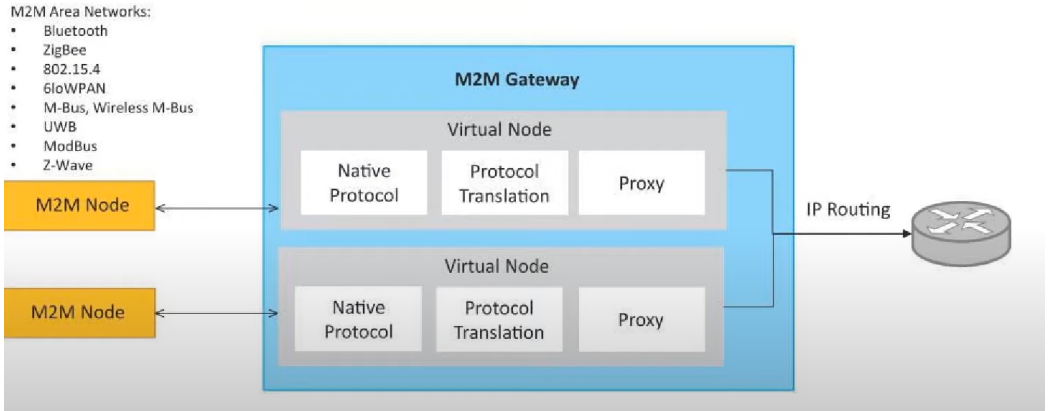
M2M System Architecture Overview :

- An M2M area network comprises machines (or M2M nodes) which have embedded hardware modules for sensing, actuation and communication.
- The communication network provides connectivity to remote M2M area networks.
- The communication network can use either wired or wireless networks (IP- based)

M2M Gateway :

- Since non-IP based protocols are used within M2M area networks, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M area networks, M2M gateways are used.

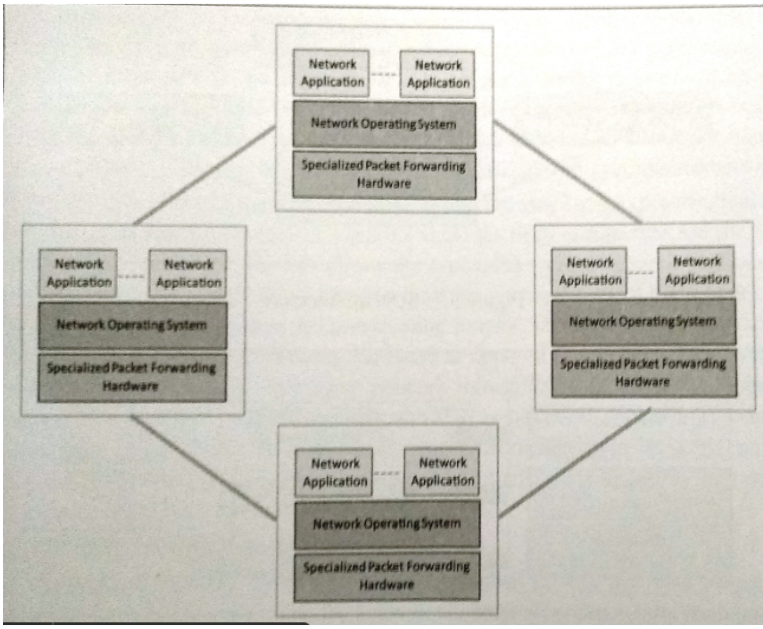
M2M Gateway Block Diagram :



20) Write a short note on Conventional network architecture and explain its limitations

A) A conventional network architecture also known as traditional network architecture, is a hierarchical network design that is commonly used in enterprise networks.

It consists of several layers, with each layer serving a specific function in the network. Conventional n/w is built with switches/routers. The three main layers in a conventional network architecture are the access layer, the distribution layer, and the core layer.



1) **Access Layer:** The access layer is the first layer in the network architecture and it is responsible for connecting end-user devices such as computers, printers to the network. It provides network access & security features to ensure that only authorized devices can access the network.

2) Distribution Layer:

The distribution layer is responsible for providing connectivity between the access layer and the core layer. It aggregates the traffic from the access layer and forwards it to the appropriate destination in the core layer.

3) Core Layer: The network layer is the core of conventional IoT architecture. The network layer acts as a bridge or gateway; it collects information and delivers it to the perception layer. It is responsible for providing high-speed connectivity between different parts of the network. It is designed to be highly available and fault-tolerant, with redundant links and devices to ensure that the network can continue to operate even in the event of a failure.

<u>Here are some of the limitations of conventional network architecture</u>:

1) **Scalability:** Conventional network architecture may struggle to grow as an organization expands, resulting in increased complexity and decreased efficiency.

2) **Cost:** The hierarchical design of conventional network architecture can result in higher costs due to the need for additional hardware and software to support the network structure.

3) **Complexity:** The hierarchical design can make it challenging to troubleshoot and resolve network issues, leading to increased complexity and downtime.

4) **Management Overhead:** Multiple network devices, Upgradation of network devices are difficult for conventional network architecture .

21) Define Sensor & Actuator and uses of both. (IMP) (PYQ)

A) Sensor: A device that provides a usable output in response to a specified measurement.

The sensor attains a physical parameter and converts it into a signal suitable for processing (e.g. electrical, mechanical, optical) the characteristics of any device or material to detect the presence of a particular physical quantity. *Sensors* are used for sensing things and devices etc.

Uses of Sensors:

- The main function of these sensors is to gather information from the surroundings.
- **Temperature sensor** is used to detect the heat energy which is produced from an object otherwise nearby. The main role of these sensors in manufacturing is for temperature monitoring of machines.
- **Smoke sensors** have been used in various applications like homes, industries, etc. Also, by adding a wireless connection to smoke detectors
- **Motion sensors are** used for security reasons. These are also used in hand dryers, energy management systems, automatic parking systems, automatic door controls, etc.

Actuator: An actuator is a device that converts energy into motion. The main purpose of an actuator is to control the movements within the machines. In an IoT system, the actuator can act on data collected by sensors to create an outcome as determined by the chosen settings of the user. A good example of an actuator is a shutoff valve.

Uses of Actuators:

Actuators are used within IoT systems to perform routine actions on equipment to improve process efficiencies. For example The actuators used in sliding doors obtain their input from the motion sensors. When the sensor detects a movement, the actuator gets activated and causes the panels of the sliding doors to move sideways, thereby controlling the movement of the door.

Unit - 2

1) List out the steps involved in IOT system design methodology.

A) The steps involved are:

Step-1–Purpose and Requirements Specification: In this step, the purpose of the IoT system is defined, and the requirements are identified. This includes identifying the stakeholders, their needs, and the expected outcomes.

Step-2–Process Specification: The processes and workflows involved in the IoT system are defined in this step. This includes identifying the data flows, the sequence of operations, and the interaction between different components.

Step-3–Domain Model Specification: The domain model specifies the entities and their relationships involved in the IoT system. This includes defining the physical and logical components, the sensors, and the actuators.

Step-4–Information Model Specification: The information model defines the data elements and their attributes. This includes identifying the data types, units of measurement, and the data format.

Step-5–Service Specifications: In this step, the services provided by the IoT system are defined. This includes identifying the service requirements, the interfaces, and the protocols used for communication between different components.

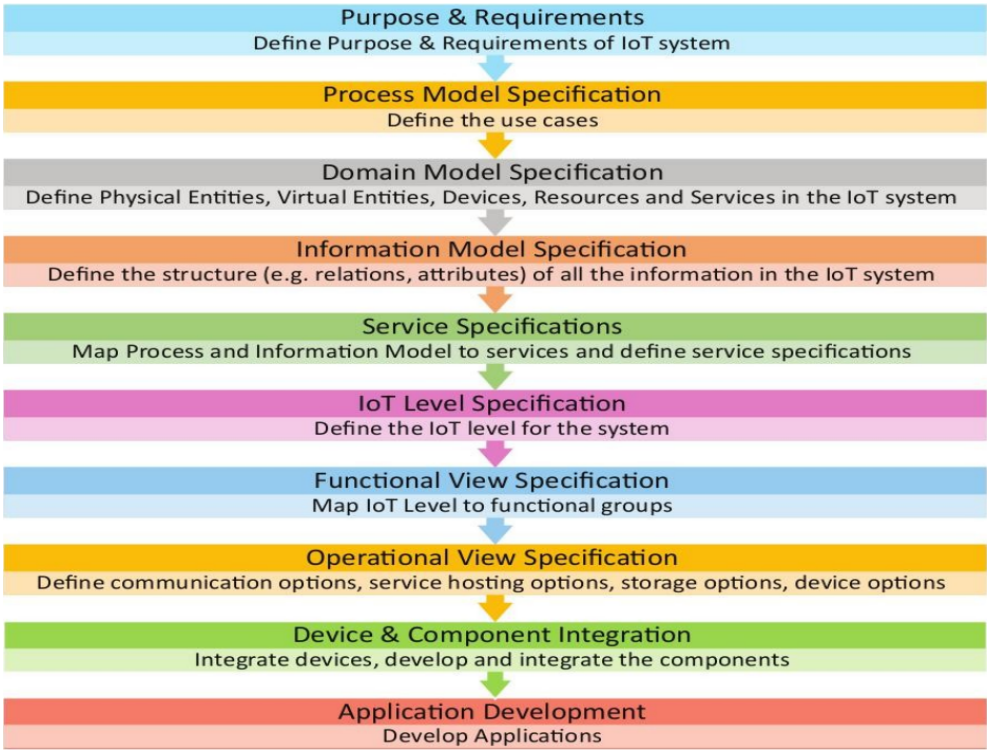
Step-6–IoT Level Specification: In this step, the overall architecture and system level requirements are defined. This includes identifying the communication protocols, data formats, security protocols, and cloud platforms to be used in the system.

Step-7–Functional View Specification: The functional view specifies the functions and services provided by the IoT system. This includes identifying the data flows, interfaces, and interaction between different components in the system.

Step-8–Operational View Specification: The operational view specifies the operational and performance characteristics of the IoT system. This includes identifying the system's response time, throughput, and availability, and defining the performance metrics to be used to evaluate the system's performance.

Step-9–Device & Component Integration: In this step, the devices and components required for the IoT system are selected and integrated. This includes configuring the devices, connecting them to the network, and integrating them with other components in the system.

Step-10–Application Development: Once the hardware and software components are integrated, applications are developed to enable users to interact with the IoT system. This includes developing user interfaces, data visualization tools, and other applications that enable users to interact with the system and make use of the data collected by the system.



2) What is the difference between a physical and virtual entity ?

A)

Physical Entity	Virtual Entity
Has a physical presence	Has no physical presence
Can be touched, seen, and felt	Cannot be touched, seen, or felt
Exists in the physical world	Exists in a digital or virtual world
Subject to the laws of physics	Subject to the rules and constraints of the digital or virtual environment
Examples: people, animals, buildings, furniture, machines	Examples: computer software, digital images, online accounts, virtual reality objects

3) Explain the purpose and requirements specification of IOT system design methodology.

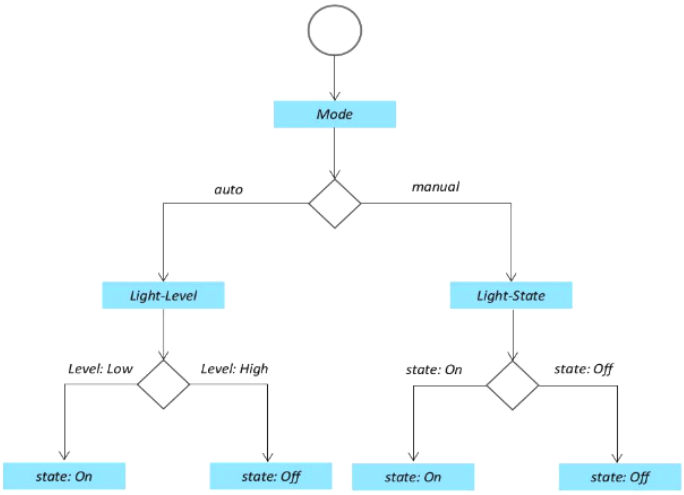
A) The purpose of an Internet of Things (IoT) system design methodology is to provide a structured approach for developing IoT solutions that meet the needs of stakeholders and users. The methodology should guide the design team through the various stages of development, from defining the problem and requirements to testing and deployment.

The requirements specification for an IoT system design methodology should include the following:

- **Problem definition:** Identify the problem that the IoT system is intended to solve, such as improving operational efficiency, reducing costs, or enhancing user experiences.
- **User requirements:** Define the needs and expectations of users for the IoT system, including the user interface, features, and functionality.
- **Technical requirements:** Specify the technical requirements of the IoT system, such as the hardware and software components, communication protocols, and data storage.
- **Performance requirements:** Define the performance metrics that the IoT system must meet, such as response time, throughput, and reliability.
- **Security requirements:** Specify the security measures that the IoT system must incorporate to protect against unauthorized access, data breaches, and other threats.

4) Discuss process specifications with a neat diagram for IOT system design methodology with an example.

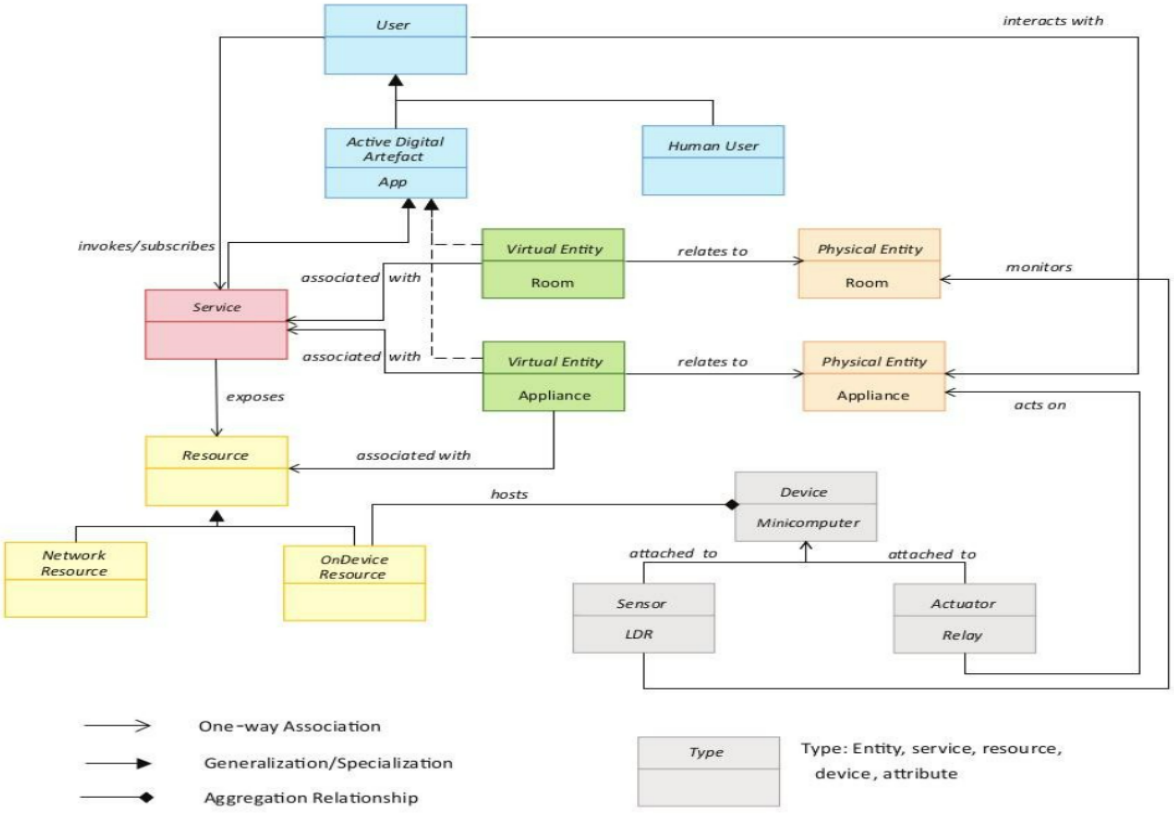
A) In this step, the use cases of the IoT system are formally described based on and derived from the purpose and requirement specifications. The process specifications for designing an IoT system include:



- **Requirements Gathering:** Gather system requirements from stakeholders.
- **Architecture Design:** Design the overall system architecture.
- **Technology Selection:** Select appropriate hardware, software, and communication protocols.
- **Development and Testing:** Develop and test the system to ensure it meets the requirements.
- **Deployment and Maintenance:** Deploy and maintain the system in the production environment.

Example: For a smart home IoT system that controls lighting and temperature, gather homeowner requirements, design the system architecture, select appropriate sensors and control systems, develop and test the system, and deploy and maintain it.

5) Explain the domain model specifications in detail with a neat diagram for IOT system design methodology.



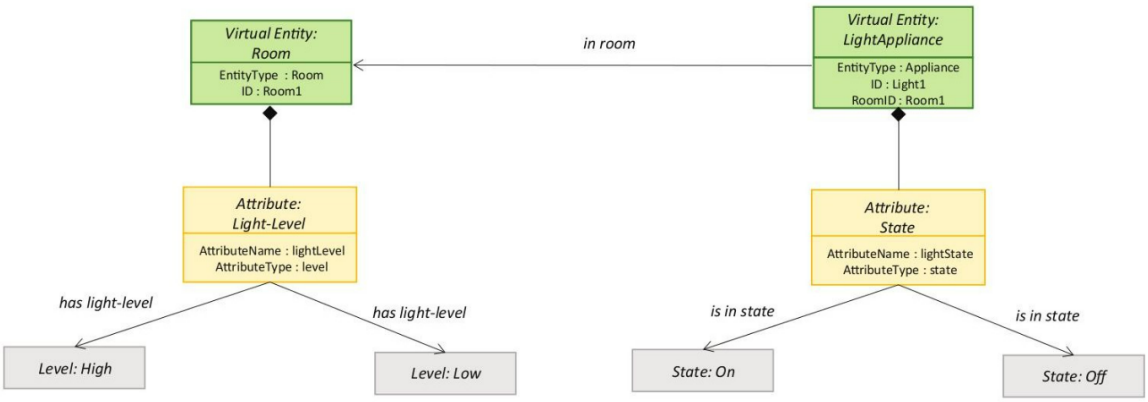
The domain model describes the main concepts, entities and objects in the domain of the IoT system to be designed.

- Domain model provides an abstract representation of the concepts, objects and entities in the IoT domain, independent of any specific technology or platform.
- With the domain model, the IoT system designers can get an understanding of the IoT domain for which the system is to be designed.

<u>Domain modeling in IoT system design involves</u>:

- ➢ Identifying the entities involved, such as devices, software, and users.
- ➢ Defining the attributes of each entity, including data and properties.
- ➢ Establishing the relationships between the entities.
- ➢ Modeling the behavior and interactions between entities.
- ➢ Refining the model based on feedback and testing.

6) Explain the information model specification for IOT system design methodology with diagrams.



- The Information Model defines the structure of all the information in the IoT system, for example, attributes of Virtual Entities, relations, etc. Information model does not describe the specifics of how the information is represented or stored.
- To define the information model, we first list the Virtual Entities defined in the Domain Model. Information model adds more details to the Virtual Entities by defining their attributes and relations

Information modeling in IoT system design involves:

- Identifying data requirements, such as sensor readings and user inputs.
- Defining data flow between system components, such as sensors, gateways, and cloud servers.
- Specifying data storage structures, databases, and retention policies.
- Specifying data processing requirements, such as analytics and machine learning.
- Defining data security requirements, including encryption and access controls.

7) What are the service specifications in designing IOT systems ?

A) Service specifications define the services in the IoT system, service types, service inputs/output, service endpoints, service schedules, service preconditions and service effects. In IoT system design, service specifications refer to the functionality and capabilities that the system provides to its users. The following are the service specifications for designing IoT systems:

1. **User Requirements**: Identify the needs and requirements of the users of the system.
2. **Service Architecture**: Design the service architecture, including the services provided and interfaces between them.

3. **Service Level Agreements**: Define the SLAs that specify the quality of service that the system will provide.
4. **Service Delivery**: Define how services will be deployed, installed, configured, and maintained.
5. **Service Management**: Define the processes and tools used to manage services, including monitoring, reporting, and troubleshooting

8) Discuss the various functional groups involved in functional view specifications in detail.

A)

1. **Input/Output Functions:** In IoT systems, these functions manage the flow of data between sensors, actuators, and other devices. They may include data acquisition, data validation, and data transmission mechanisms.

2. **Processing Functions:** These functions are responsible for processing the data collected from IoT devices, often in real-time. They may include data filtering, aggregation, and analysis, as well as decision-making processes based on predefined rules or machine learning algorithms.

3. **Storage Functions:** In IoT systems, these functions manage the storage and retrieval of data generated by the devices. They may include cloud-based storage solutions, edge storage, and data caching mechanisms.

4. **Control Functions:** These functions are responsible for managing the overall operation of the IoT system, ensuring its reliability and stability. They may include error handling, system monitoring, and performance optimization mechanisms, as well as device management and firmware updates.

5. **Communication Functions:** In IoT systems, these functions facilitate communication between devices, as well as between devices and external entities, such as cloud services or other IoT systems. They may include various communication protocols, such as MQTT, CoAP, or HTTP, and data exchange formats, such as JSON or XML.

6. **User Interface Functions:** These functions provide a means for users to interact with the IoT system, monitor its status, and control its operation. They may include web-based interfaces, mobile applications, and voice-activated interfaces.

9) Explain the operational view specifications in designing IOT systems.

A)

1. **System Components**: Specifies the physical components of the system, including hardware and software components. It defines the sensors, actuators, microcontrollers, communication protocols, and other components needed to implement the system.
2. **System Architecture:** Defines the overall architecture of the system, including how the various components are connected, how they communicate with each other, and how data flows through the system. It describes the network topology, protocols used, and the communication interfaces between the various components.
3. **User Interface:** Specifies the user interface used to interact with the system, including the display screens, buttons, switches, and other elements used by end-users to interact with the system.It describes how users can configure, monitor, and control the system.
4. **Functional Requirements:** Specifies the specific capabilities and features the system must provide, defining the use cases and scenarios that the system must support.
5. **Performance Requirements:** Specifies the performance requirements of the system, including its response time, throughput, and availability.
6. **Environmental Requirements:** Specifies the physical environment in which the system will operate, including the operating temperature range, humidity levels, and other environmental factors that the system must be able to withstand.

10) Explain the importance of device ,component integration and application development in IOT system design.

A) The importance of device/component integration and application development in IoT system design:

- Device/component integration is the process of integrating various sensors, devices, and components to collect, store, and analyze data.
- It is essential to ensure that the components are compatible and can communicate with each other seamlessly to avoid data loss, security breaches, and system failure.
- Application development involves developing software applications that allow users to interact with the system and make sense of the collected data.
- Applications can range from simple dashboards that provide real-time data visualization to complex analytics tools that enable users to perform advanced data analysis.
- Device/component integration and application development can lead to improved data collection and management, increased efficiency and automation, enhanced user experience, and improved system security.
- Proper integration and development enable automated rules that can trigger specific actions based on data collected, ensuring increased efficiency, and automation.

- Users can enjoy real-time data visualization, alert notifications, and other features that enhance the user experience.
- To ensure system security, data should be encrypted during transmission and stored securely, access controls should be in place, and firmware should be updated regularly.

11) What is the need for controller service ?

A) A controller service is a type of service in a software system that provides centralized management and control of resources or components. The need for a controller service arises in complex software systems where there are multiple components or resources that need to be managed and controlled in a consistent and coordinated manner. Some of the key reasons why a controller service is needed are:

- **Simplify system management:** A controller service simplifies system management by providing a single point of control for managing and coordinating multiple components or resources. This makes it easier to manage the system as a whole, reducing complexity and increasing efficiency.
- **Increase system reliability:** A controller service can improve system reliability by providing a centralized point of control for monitoring and managing components or resources. This enables the service to detect and respond to failures or errors quickly, minimizing the impact on the overall system.
- **Enable system scalability:** A controller service can facilitate system scalability by providing a centralized point of control for adding or removing components or resources as needed. This enables the service to scale the system up or down as required, without requiring significant changes to the underlying system architecture.
- **Enhance system security:** A controller service can enhance system security by providing a centralized point of control for enforcing security policies and access controls across multiple components or resources. This makes it easier to ensure that security policies are consistently applied throughout the system.

12) What is cloud computing.What are Basic Cloud Characteristics ?

A) Cloud computing refers to the delivery of computing resources, such as computing power, storage, and applications, over the internet on a pay-as-you-go basis. In cloud computing, users can access and use computing resources on demand, without the need for local infrastructure or capital expenditure. The basic characteristics of cloud computing are:

- **On-demand self-service:** Users can provision computing resources, such as virtual machines or storage, as needed without requiring human interaction with service providers.
- **Broad network access:** Cloud services can be accessed over the internet through a variety of devices, such as desktops, laptops, tablets, and smartphones.
- **Resource pooling:** Cloud providers can pool computing resources, such as storage and processing power, to serve multiple users or customers, enabling greater efficiency and flexibility.
- **Rapid elasticity:** Cloud resources can be scaled up or down quickly to meet changing user demands. This enables users to rapidly and easily adjust their computing resources to match their needs.
- **Measured service:** Cloud service providers can monitor and measure resource usage, allowing users to pay only for the resources they use.

These basic cloud characteristics enable cloud computing to provide several benefits to users, including reduced costs, greater scalability, increased flexibility, improved reliability, and faster time-to-market for applications and services.

13) Explain SaaS, PaaS,IaaS in detail.

A) SaaS, PaaS, and IaaS are three common models for delivering cloud computing services. Each model provides a different level of abstraction and control over the underlying infrastructure, allowing users to choose the level of control that best fits their needs.

SaaS (Software-as-a-Service): SaaS is a cloud computing model in which a provider delivers software applications over the internet, allowing users to access the applications through a web browser or mobile app. Users pay for the service on a subscription basis and can typically access the application from anywhere with an internet connection. Examples of SaaS include Google Workspace, Salesforce, and Microsoft Office 365.

PaaS (Platform-as-a-Service): PaaS is a cloud computing model in which a provider delivers a platform for developing and deploying applications over the internet. The provider manages the infrastructure and provides a range of tools and services for building and deploying applications. Users pay for the service on a usage basis. Examples of PaaS include Google App Engine, Heroku, and Microsoft Azure.

IaaS (Infrastructure-as-a-Service): IaaS is a cloud computing model in which a provider delivers virtualized computing resources, such as virtual machines, storage, and networking, over the internet. The provider manages the physical infrastructure, while users are responsible for

managing the virtualized resources. Users pay for the service on a usage basis. Examples of IaaS include Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform.

14) What are the advantages and disadvantages of cloud computing ?

A) <u>Advantages of cloud computing:</u>

- **Cost savings:** Pay for computing resources on a usage basis and eliminate the need for upfront capital investment in hardware and software.
- **Scalability:** Quickly and easily scale up or down computing resources based on business needs.
- **Accessibility:** Work from anywhere with an internet connection and collaborate more easily.
- **Reliability:** High levels of uptime and availability through service level agreements (SLAs) ensure reliable and continuous service.
- **Security:** Cloud computing providers often have robust security measures in place to protect against cyber threats and data breaches.

<u>Disadvantages of cloud computing:</u>

- **Dependence on internet connectivity:** Cloud computing relies on a stable internet connection, and slow or unreliable connectivity can affect the user experience and productivity.
- **Limited control over infrastructure:** Users have limited control over the underlying infrastructure, which can be a disadvantage for users who require fine-grained control over their computing resources.
- **Data privacy concerns:** Storing sensitive data in the cloud raises concerns about data privacy and security.
- **Vendor lock-in:** Migrating from one cloud provider to another can be difficult, and some providers make it challenging to move data or applications to another provider.
- **Potential for downtime:** While cloud providers offer high levels of uptime, there is always the risk of downtime, which can cause disruptions in business operations and productivity.

Unit - 3

1) Discuss various security and privacy issues in the Internet of things.

A) Some most common threats to the security and privacy of IoT devices.

i. Weak Credentials – Generally, large manufactures ship their products with a username of "admin" and with the password "0000" or "1234" and the consumers of these devices don't change them until they are forced to by security executives.

ii. Complex Structure of IoT Devices – IoT devices have a very complex structure that makes it difficult to find the fault in devices. Even if a device is hacked the owner of that device will be unaware of that fact. Hackers can force the device to join any malicious botnets or the device may get infected by any virus.

iii. Insecure Data Transfer – It is very difficult to transmit data securely in such a large amount as there are billions of IoT enabled devices. There is always a risk of data leaking or getting infected or corrupted.

iv. Small Scale Attacks – IoT devices are attacked on a very small scale. Manufacturing companies are trying to secure their devices for large scale attacks but no company is paying attention to small attacks.

v. Smart Objects – Smart objects are the main building block of any device. These smart objects should be able to communicate with another object or device or a sensor in any infrastructure securely. Even while these devices or objects are not aware of each other's network status

2) Discuss the regulatory issues for IOT.

A) Here are some of the regulatory issues for IoT:

i. Security: One of the biggest concerns with IoT devices is security. Many of these devices are connected to the internet, which makes them vulnerable to cyberattacks. There have been instances where hackers have taken control of IoT devices, causing harm or disruption. To address this issue, governments may need to implement regulations that require IoT device manufacturers to meet specific security standards.

ii. Privacy: IoT devices often collect personal data, such as location, health information, and usage patterns. This data can be used for targeted advertising or sold to third party companies. Governments may need to implement regulations that require companies to be transparent about how they collect and use data and provide individuals with the right to opt-out of data collection.

iii. Interoperability: With so many different IoT devices available, it can be challenging to ensure that they are compatible with each other. Governments may need to implement regulations

that encourage or mandate interoperability standards to make it easier for devices from different manufacturers to work together.

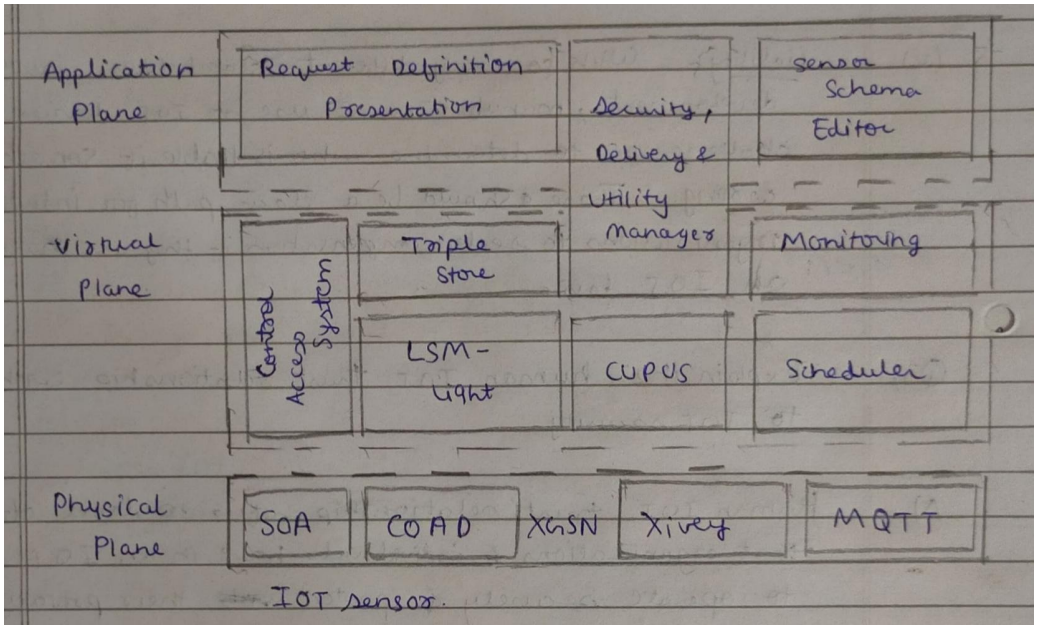
iv. Liability: With so many different stakeholders involved in the development, manufacture, and use of IoT devices, it can be challenging to determine who is liable if something goes wrong. Governments may need to implement regulations that clarify liability and ensure that there is a clear path for individuals or organizations to seek compensation if they are harmed by an IoT device.

v. Spectrum allocation: IoT devices often use radio frequencies to communicate with each other and with the internet. Governments may need to regulate the allocation of these frequencies to prevent interference and ensure that there is enough bandwidth available to support the growing number of IoT device

3) Explain the human IOT trust relationship with respect to IOT security.

- The human IoT trust relationship refers to the level of trust that individuals and organizations have in IoT devices to operate securely and protect their privacy.
- Trust is critical to the success of IoT systems because if users do not trust the devices, they are less likely to use them or share data with them.
- Establishing trust requires addressing security concerns such as vulnerability to cyberattacks and data breaches.
- IoT devices should be designed with privacy in mind, and manufacturers should provide clear information about data collection and usage practices.
- Users should have control over their data, such as opting out of data collection or deleting their data.
- Education is crucial to building trust in IoT devices by informing users about the risks and benefits of using IoT devices and best practices for security.

4) Explain open IOT architecture.



- Open IoT architecture refers to an approach to designing IoT systems that emphasizes interoperability, scalability, and modularity. It is a framework for building IoT solutions that are flexible, adaptable, and able to integrate with a wide range of devices and services.
- It is based on open standards and protocols, which means that devices and services from different vendors can work together seamlessly. This is important because it allows organizations to build complex IoT systems that can grow and evolve over time without being locked into a single vendor or technology.
- One of the key principles of open IoT architecture is the separation of concerns. This means that different components of the IoT system are designed to perform specific functions, such as data collection, data processing, and data visualization. By separating these functions, organizations can build more flexible and scalable systems that can be easily modified or replaced as needs change.
- Another important aspect of open IoT architecture is the use of APIs (Application Programming Interfaces) to enable communication between different components of the system. APIs provide a standardized way for devices and services to exchange data and commands, which makes it easier to build and integrate different components of the IoT system.
- It also emphasizes the use of open-source software and hardware, which allows organizations to leverage the work of the larger community of developers and contributors. This can help to accelerate development, reduce costs, and improve the quality of the final product.

5) Explain open IOT platform capabilities.

A) Open IoT platforms have several capabilities, including:

- **Device management:** Open IoT platforms enable the management of large numbers of devices from different manufacturers, with varying capabilities and protocols.
- **Data collection and processing:** Open IoT platforms can collect and process data from different sources, such as sensors, wearables, and other IoT devices. They can also transform raw data into meaningful insights for decision-making.
- **Connectivity:** Open IoT platforms support various connectivity options such as Wi-Fi, Bluetooth, cellular, and Ethernet to ensure that devices can communicate and share data seamlessly.
- **Security and privacy:** Open IoT platforms incorporate security features such as data encryption, access control, and device authentication to protect data and devices from cyber threats.
- **Customization and scalability:** Open IoT platforms are highly customizable and scalable, allowing users to tailor the platform to their specific needs and accommodate changes in their IoT infrastructure.

Unit - 4

1) What is an Arduino? List the different types of Arduino boards available. (IMP) (PYQ)

A) Arduino is an open source platform and has a variety of controllers and microprocessors. The Arduino is a single circuit board, which consists of different interfaces or parts. The board consists of the set of digital and analog pins that are used to connect various devices and components, which we want to use for the functioning of the electronic devices.

Here are some of the different **types of Arduino boards** available:

- **Arduino Uno:** This is the most popular and widely used board. It has 14 digital input/output pins, 6 analog inputs, a USB connection, a power jack and a reset button.
- **Arduino Pro Mini:** This is a smaller version of the Uno board with fewer input/output pins. It is designed for projects that require a small size and low power consumption.
- **Arduino Mega:** This board is similar to the Uno but has more input/output pins and more memory. It is ideal for projects that require more processing power.
- **Arduino Nano:** This is a smaller version of the Uno board with similar features. It is designed for projects that require a compact size.
- **Arduino Leonardo:** This board has a built-in micro USB connection, which allows it to act as a mouse or keyboard. It has 20 digital input/output pins and 12 analog inputs.

2) Explain the Arduino program structure.

A) The Arduino program structure is a basic framework used to create programs that run on Arduino microcontrollers. It consists of two mandatory functions :-

```
void setup ( ) {

}
void Loop ( ) {

}
```

1) **setup()** – The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

2) **loop()** – After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

3) Explain the use of the following functions. (IMP)

A. **Serial.begin()** - It sets the baud rate for serial data communication. The baud rate signifies the data rate in bits per second.
Syntax : *Serial.begin(speed)*
Where, **speed** is the specified baud rate

B. **pinMode()** - It is used to configure a specific pin to behave either as an input or an output.
Syntax : *pinMode(pin, mode)*

pin	Number of pin
mode	INPUT, OUTPUT

C. **digitalRead()** - It is used to read a HIGH (5V) or a LOW (0V) value from a digital pin.
Syntax : *digitalRead(pin, value)*

pin	Number of pin
value	HIGH or LOW

D. **digitalWrite()** - It is used to write a HIGH (5V) or a LOW (0V) value to a digital pin.
Syntax : *digitalWrite(pin, value)*

pin	Number of pins
value	HIGH or LOW

E. **analogRead()** - It reads the value from the specified analog pin present on the particular Arduino board.
Syntax : *analogRead(pin)*
where, **pin** is the number of pin

F. **analogWrite()** - It generates a digital pulse to the chosen PWM pin and is used to write the PWM value. **(PWM pins are 3, 5, 6, 9, 10 and 11) [Analog pulse → Digital Pulse]**
Syntax : *analogWrite(pin, value)*
where, **pin** is the number of pins
value is the duty cycle (0-255)

G. **Serial.read() -** It reads the incoming serial data in the Arduino. The int data type is used here.
Syntax : *Serial.read()*
where, **serial** signifies the serial port object.

H. **Serial.println() -** It prints the data to the serial port in ASCII format which is human-readable.
Syntax : *Serial.println(value)*
Where, **value** is data type value

I. **pulseIn() -** Reads an incoming pulse on a pin and measures the duration of the pulse (in ms).
Syntax : *pulseIn(pin, value)*
where, **pin** is the number of pins
value is the level of pulse (HIGH or LOW)

J. **delay() -** It is used to stop the program for a specific time interval before executing the next line.
Syntax : *delay(ms)*
where, **ms** is the time in milliseconds
Other delay functions
1. delayMicroseconds() — Here Delay will be in micro- seconds.
2. micros() - Returns the number of microseconds since the Arduino board began running the current program.(unsigned long)
3. millis() - Number of milliseconds passed since the program started. (unsigned long)

K. **String Related Functions -**
1) charAt(): This function returns the character located at a specific index within a string.
Syntax : strName.charAt(index)
2) concat(): This function appends one string to the end of another string.
Syntax : strName.concat(parameter)
3) equals(): Compare two Strings for equality. The comparison is case-sensitive.
Syntax : str1.equals(str2)
4) length(): Returns the length of the string.
Syntax : myString.length()
5) substring(): Get a substring of a String.
Syntax : myString.substring(from, to)

4) Simple programs using an arduino board. **[NOTE - Figures are for reference only]**

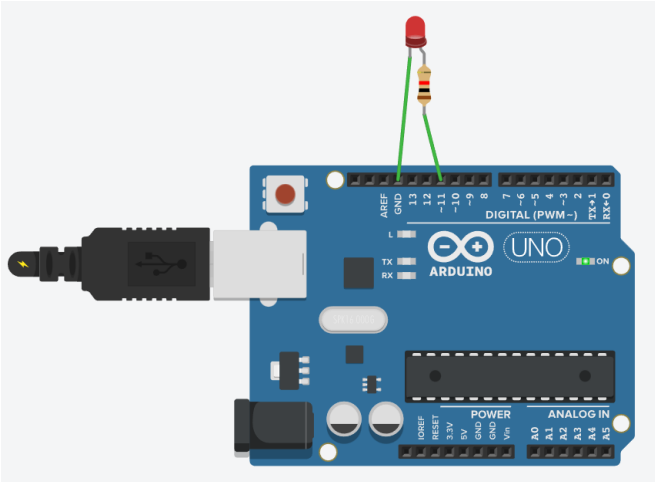
A. LED Interfacing

Code -

```
int pin=11;

void setup()
{
 pinMode(pin, OUTPUT);
}

void loop()
{
 digitalWrite(pin, HIGH);
 delay(2000);
 digitalWrite(pin, LOW);
 delay(2000);
}
```



B. SWITCH Interfacing

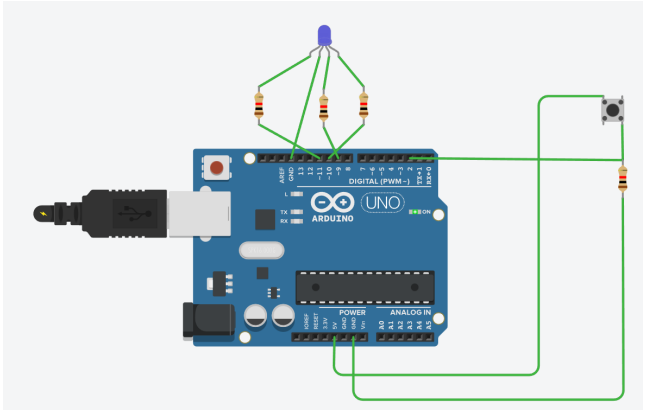
Code -

```
 int switchPin = 2;
 int redPin = 11;
 int greenPin = 10;
 int bluePin = 9;

 int switchState = 0;
 int ledState = 0;

void setup() {
 pinMode(switchPin, INPUT);
 pinMode(redPin, OUTPUT);
 pinMode(greenPin, OUTPUT);
 pinMode(bluePin, OUTPUT);
}

void loop() {
 switchState = digitalRead(switchPin);
```



```
 if (switchState == HIGH) {

  ledState++;
  if (ledState > 3) {
   ledState = 1;
  }
  switch (ledState) {
   case 1:
    digitalWrite(redPin, HIGH);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, LOW);
    break;

   case 2:
    digitalWrite(redPin, LOW);
    digitalWrite(greenPin, HIGH);
    digitalWrite(bluePin, LOW);
    break;

   case 3:
    digitalWrite(redPin,LOW);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, HIGH);
  }
  delay(2000);
 } else {
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(bluePin, LOW);
 }
}
```

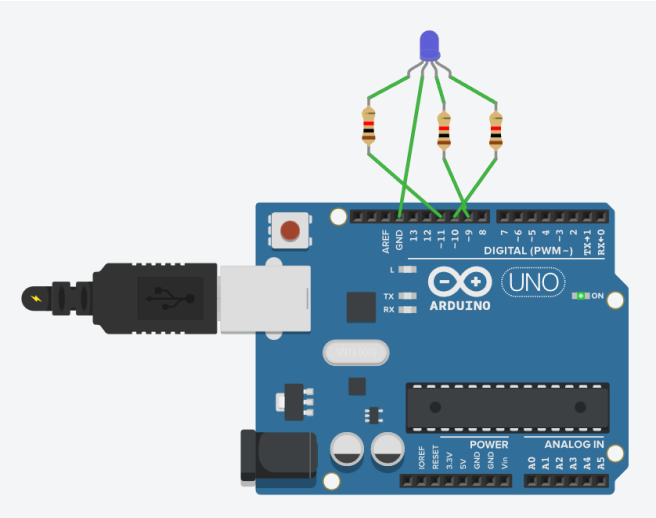
C. RGB Led Interfacing

<u>Code</u> -

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;
```

```
void setup()
{
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop()
{
  digitalWrite(redPin, HIGH);
  delay(2000);
  digitalWrite(redPin, LOW);
  delay(2000);
  digitalWrite(greenPin, HIGH);
  delay(2000);
  digitalWrite(greenPin, LOW);
  delay(2000);
  digitalWrite(bluePin, HIGH);
  delay(2000);
  digitalWrite(bluePin, LOW);
  delay(2000);
}
```

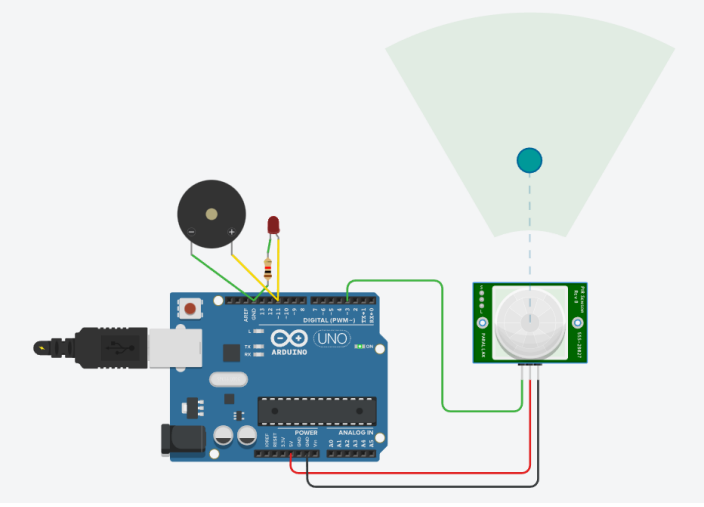


D. MOTION Sensor interfacing (PYQ)

Code -

```
int buttonstate = 0;
void setup()
{
  Serial.begin(9600);
  pinMode(3, INPUT);          // Motion Sensor
  pinMode(11, OUTPUT);        // LED & buzzer
}

void loop()
{
  buttonstate = digitalRead(3);
  if(buttonstate == HIGH) {
    digitalWrite(11, HIGH);
```



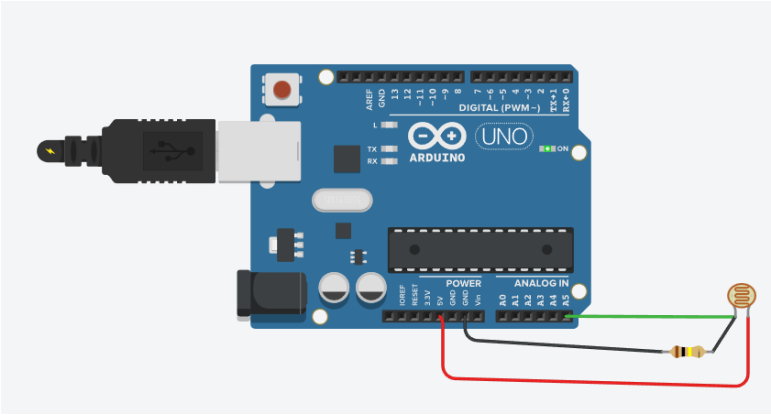
```
  Serial.print("Object Detected \n");
 }
 else {
  digitalWrite(11, LOW);
  Serial.print("---------- \n");
 }
 delay(2000);
}
```

E. LDR Interfacing (PYQ)

Code -

```
int serialReading=0;
void setup()
{
 Serial.begin(9600);
}

void loop()
{
 serialReading =analogRead(A5);
 Serial.println(serialReading);
 delay(2000);
}
```



SERIAL READINGS:

Maximum - 1018

Minimum - 366

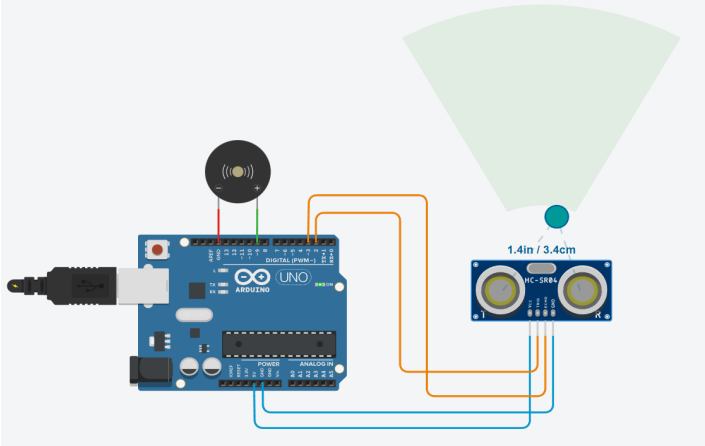
F. Ultrasonic Sensor Interfacing

Code -

```
int echo_pin = 3;         // Transmits ultrasonic burst and receives an echo
int trigger_pin = 2;      // Triggers ultrasonic sound pulses & initiates ultrasonic burst
int buzzer_pin = 9;

void setup()
{
 Serial.begin(9600);
 pinMode(echo_pin, INPUT);
```

```
  pinMode(trigger_pin, OUTPUT);
  pinMode(buzzer_pin, OUTPUT);
}
void loop()
{
  digitalWrite(trigger_pin, HIGH);
  digitalWrite(trigger_pin, LOW);
  int time = pulseIn(echo_pin, HIGH);
  float distance = (time * 0.034) / 2;
// Speed = 340m/s
```

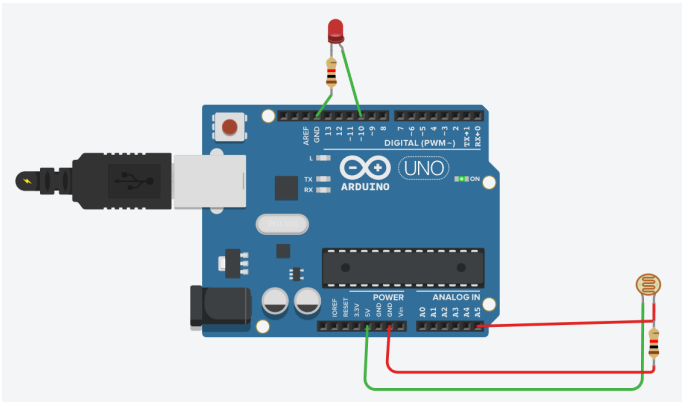



```
  if(distance < 5) {
    Serial.println((String)"Buzzer On  [Distance: " + distance + "]");
    analogWrite(buzzer_pin, 210);
  }

  else if(distance > 5 && distance < 10){
    Serial.println((String)"Buzzer On [Distance: " + distance + "]");
    analogWrite(buzzer_pin, 125);
  }
  else{
    Serial.println((String)"Buzzer Off [Distance: " + distance + "]");
    analogWrite(buzzer_pin, 0);
  }
  delay(1000);
}
```

G. LED brightness control using PWM

Code -

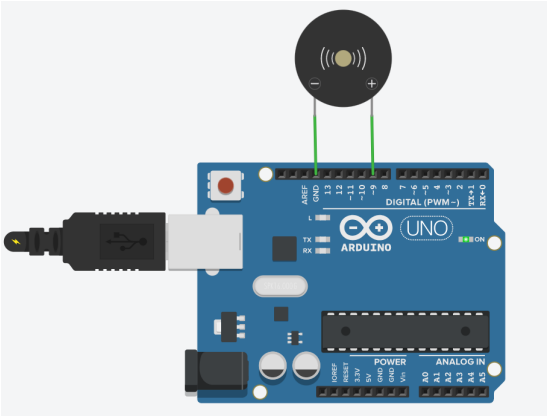


```
int serialReading = 0;
void setup()
{
  Serial.begin(9600);
  pinMode(10, OUTPUT);
}
void loop()
{
  serialReading = analogRead(A5);
```

```
  Serial.println((String)"Intensity value: " + serialReading);
  analogWrite(10, analogRead(A5)/4);
  delay(1000);
}
```

H. Generate Different tone using PWM

Code -



```
int sound = 9;
void setup()
{
  Serial.begin(9600);
  pinMode(sound, OUTPUT);
}

void loop()
{
  for (int pwmVal = 0; pwmVal < 255; pwmVal++) {
    analogWrite(sound, pwmVal);
    Serial.println((String) "PWM value: " + pwmVal);
    delay(1000);
  }
}
```

5) Give the difference between Arduino and Raspberry pi. (IMP)

Arduino	Raspberry pi
Based on a microcontroller	Based on a microprocessor
Needs an OS	Doesn't require OS
Best for repetitive small tasks	Best for general computing
Cheaper (Open-source)	Expensive (Closed-source)
Logic level is 5V	Logic level is 3V
Simple hardware design	Complex hardware design
Programmed in C/C++	Programmed in C/C++, Python, Ruby etc.
No Bluetooth or Wifi support	Supports Bluetooth and Wifi

Low power consumption	High power consumption
Eg:- Traffic control light controller etc.	Eg:- Robot Controllers, Stop motion cameras

6) List the key features of ESP8266 (Node MCU). (IMP)

A)

- **Integrated Wi-Fi:** The ESP8266 has built-in Wi-Fi connectivity, making it easy to connect to wireless networks and communicate with other devices.
- **Small size**: The ESP8266 is a compact microcontroller, making it easy to incorporate into projects with limited space.
- **Low power consumption:** The ESP8266 is designed to operate on low power, making it ideal for battery-powered projects.
- **GPIO pins:** The ESP8266 has a number of General Purpose Input/Output (GPIO) pins that can be used to control external devices or read sensor data.
- **Programming options:** The ESP8266 can be programmed using a variety of languages, including Lua, Arduino, and C++, making it accessible to a wide range of developers.
- **Integrated USB-to-serial converter:** The ESP8266 includes a built-in USB-to-serial converter, making it easy to connect to a computer for programming and debugging.
- **High processing power:** Despite its small size, the ESP8266 has a powerful processor, allowing it to perform complex tasks quickly and efficiently.
- **Low cost:** The ESP8266 is one of the most affordable microcontrollers available, making it a popular choice for hobbyists and DIY enthusiasts.

7) List the key features of Raspberry Pi. (IMP)

A)

1. **Compact Size:** The Raspberry Pi is small and lightweight, making it easy to transport and fit into tight spaces.
2. **GPIO Pins:** The Raspberry Pi has GPIO (General Purpose Input/Output) pins, which allow users to connect external devices such as sensors, lights, and motors.
3. **Operating System:** The Raspberry Pi can run various operating systems, including Linux and Windows 10 IoT Core.
4. **High-performance:** Despite its small size, the Raspberry Pi has impressive processing power, capable of handling a variety of tasks.
5. **Connectivity:** The Raspberry Pi has built-in Ethernet, Wi-Fi, and Bluetooth connectivity, allowing users to connect to the internet and other devices.
6. **Camera:** The Raspberry Pi has a camera module that can be used for various projects, such as video streaming and image recognition.

7. **Audio:** The Raspberry Pi has a 3.5mm audio jack and can be used to play audio or connect to speakers.
8. **Expandability:** The Raspberry Pi can be expanded using various accessories such as display screens, cases, and HATs (Hardware Attached on Top).
9. **Low-cost:** The Raspberry Pi is affordable, making it accessible to a wide range of people but not as low as Arduino.

8) Explain the use of the following functions of ESP8266 : WiFi.mode(), WiFi.begin() and WiFi.config()

A)

1. **WiFi.mode(mode):** Used to set the WiFi mode of the ESP8266 chip, such as
   - ➢ connecting to an existing network as a client
   - ➢ creating a new network as an access point.
2. **WiFi.begin(ssid, password)**: Used to **connect the ESP8266 to an existing WiFi network as a client by providing the SSID and password.**
3. **WiFi.config(local_ip, gateway, subnet):** Used to **configure the IP address, gateway, and subnet mask of the ESP8266** when connected to a WiFi network by providing the relevant IP addresses.

9) Describe GPIO available in raspberry pi.

A)

- Raspberry Pi has 40 GPIO pins that can be used for various tasks. These pins are arranged in two rows of 20 pins each, with each pin having a specific function.
- The GPIO pins are controlled by software using the RPi. GPIO uses the Python library or other programming languages such as C/C++.
- Each GPIO pin can be set as an input or output pin by configuring it as a digital input or output.
- The GPIO pins are also used for connecting various sensors, actuators, and other external devices, such as LED lights, buzzers, motors, and temperature sensors.
- It's important to note that GPIO pins can only handle a limited amount of current, so it's necessary to use external components such as resistors, capacitors, and transistors to protect the Raspberry Pi and connected devices.

GPIO Functions :-

1) RPi.GPIO, is a Python module to control the GPIO interface on the Raspberry Pi.

2) GPIO.setmode(mode): This function sets the numbering mode used for the GPIO pins. There are two modes available:

- GPIO.BCM - The BCM mode uses a consistent numbering scheme across different Raspberry Pi

- GPIO.BOARD - The BOARD mode uses the physical pin numbers on the header of the Raspberry Pi board.

3) GPIO.setup(channel, direction): This function sets up a GPIO pin for input or output.
Where channel is the channel number based on the numbering system you specified when you called setmode.

4) GPIO.input(channel,state): where channel is the channel number as used in setup.

- It will return a value of 0, GPIO.LOW, or False if was at a low level
- It will return 1, GPIO.HIGH, or True if it was at a high level.

5) GPIO.output(channel, state):

- where channel is the channel number
- state is the desired output level: GPIO.LOW for a low value or GPIO.HIGH for a high level.

6) GPIO.cleanup(): When you are done with the library, it is good practice to free up any resources used and return all channels back to the safe default of being inputs.

10) ESP8266 node MCU Operating modes - Station mode and Access point mode with program (IMP)

A) The ESP8266 NodeMCU board can operate in two modes - Station mode and Access point mode.

- In Station mode, the board connects to an existing Wi-Fi network and can communicate with other devices on that network.
- In Access point mode, the board acts as a Wi-Fi hotspot and other devices can connect to it.

Example that demonstrates both Station mode and Access point mode on the ESP8266 NodeMCU board:

```
#include <ESP8266WiFi.h>

// Network Credentials
const char* ssid = "id";
const char* password = "123";
// Set up access point credentials
const char* ap_ssid = "id";
const char* ap_password = "456";

// Status Check
void setup() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED);
```

```
  WiFi.softAP(ap_ssid, ap_password);
}

void loop() {
// Station mode: print the IP address every 10 seconds
  Serial.print("Station mode IP address: ");
  Serial.println(WiFi.localIP());
// Access point mode: print the number of clients every 10 seconds
  Serial.print("Access point mode clients: ");
  Serial.println(WiFi.softAPgetStationNum());
  delay(10000);
}
```

Explanation - In the setup() function, we first set up Station mode by connecting to the Wi-Fi network. We wait for the connection to be established and then print the board's IP address. Next, we set up Access point mode by creating a Wi-Fi hotspot with the given credentials.
In the loop() function, we demonstrate both modes. In Station mode, we print the IP address every 10 seconds. In Access point mode, we print the number of connected clients every 10 seconds.