IOT SEM ANSWERS

- **READ AT YOUR OWN RISK**
- **REFER DIGITAL NOTES AND PPT**

UNIT-1

1)Define IOT. Explain Characteristics and components of IOT.

A)
Definition of IoT A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network, often communicate data associated with users and their environments.

The Internet of Things refers to the rapidly growing network of connected objects that can collect and exchange data in real-time using embedded sensors. Cars, lights, refrigerators, and appliances can all be connected to the IoT.

The Internet of things is a connecting bridge between the physical world and the cyber world and Machine to Machine communication Any physical object can be transformed into an IoT device if it can be connected to the internet to be controlled or communicate information.

A lightbulb that can be switched on using a smartphone app is an IoT device, as is a motion sensor

Characteristics of the Internet of Things:

Connectivity — Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connect, this should be guaranteed at all times. For example, the connection between people through internet devices like mobile phones, and other gadgets, also a connection between Internet devices such as routers, gateways, sensors, etc.

Intelligence and Identity — The extraction of knowledge from the generated data is very important. For example, a sensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.

Scalability — The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as an outcome is enormous, and it should be handled appropriately.

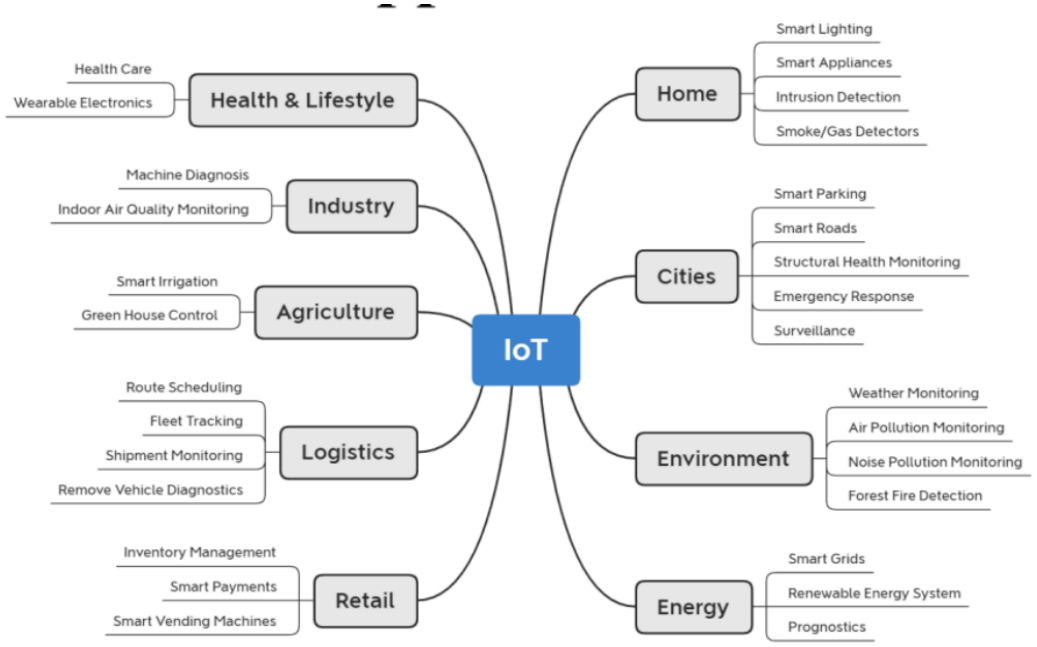
Dynamic and Self-Adapting (Complexity) –IoT devices should dynamically adapt themselves to changing contexts and scenarios. Assume a camera meant for surveillance. It should be adaptable to work in different conditions and different light situations (morning, afternoon, and night).

Architecture — IoT architecture cannot be homogeneous in nature. It should be hybrid, supporting different manufacturers ‘ products to function in the IoT network. IoT is not owned by anyone engineering branch. IoT is a reality when multiple domains come together.

Safety — There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also be at the risk. Therefore, equipment safety is also critical.

2)Discuss various application areas of IoT

A)



Home Automation

- **Smart Lighting**
- Control lighting by remotely (mobile or web applications)

• **Smart Appliances**
• Provide status information to the users remotely
• **Intrusion Detection**
• Use security cameras and sensors (PIR sensors and door sensors)
• Detect intrusions and raise alerts
• The alerts form: an SMS or an email sent to the user
• **Smoke/Gas Detectors**
• Use optical detection, ionization, or air sampling techniques to detect the smoke
• Gas detectors can detect harmful gases
    • Carbon monoxide (CO)
    • Liquid petroleum gas (LPG)
• Raise alerts to the user or local fire safety department

Cities

• **Smart Parking**
• Detect the number of empty parking slots
• Send the information over the internet and accessed by smartphones
• **Smart Roads**
• Provide information on driving conditions, traffic congestions, accidents
• Alert for poor driving conditions
• **Structural Health Monitoring**
• Monitor the vibration levels in the structures (bridges and buildings)
• Advance warning for imminent failure of the structure
• **Surveillance**
• Use the large number of distributed and internet connected video surveillance cameras
• Aggregate the video in cloud-based scalable storage solutions
• **Emergency Response**
• Used for critical infrastructure monitoring
• Detect adverse events

Environment

• **Weather Monitoring**
• Collect data from several sensors (temperature, humidity, pressure, etc.)
• Send the data to cloud-based applications and storage back-ends
• **Air Pollution Monitoring**
• Monitor emission of harmful gases (CO2, CO, NO, NO2 , etc.)
• Factories and automobiles use gaseous and meteorological sensors
• Integration with a single-chip microcontroller, several air pollution sensors, GPRSmodem, and a GPS module

• **Noise Pollution Monitoring**
• Use a number of noise monitoring stations
• Generate noise maps from data collected
• **Forest Fire Detection**
• Use a number of monitoring nodes deployed at different locations in a forests
• Use temperature, humidity, light levels, etc.
• Provide early warning of potential forest fire
• Estimates the scale and intensity
• **River Floods Detection**
• Monitoring the water level (using ultrasonic sensors) and flow rate (using the flow velocity sensors)
• Raise alerts when rapid increase in water level and flow rate is detected.

Energy

•**Smart Grids**
• Collect data regarding electricity generation, consumption, storage (conversion of energy into other forms), distribution, equipment health data
• Control the consumption of electricity
• Remotely switch off supply
• **Renewable Energy Systems**
• Measure the electrical variables
• Measure how much the power is fed into the grid
• **Prognostics**
• Predict performance of machines or energy systems
• By collect and analyze the data from sensors

Retail

• **Inventory Management**
• Monitoring the inventory by the RFID readers
• Tracking the products
• **Smart Payments**
• Use the NFC
• Customers store the credit card information in their NFC-enabled
• **Smart Vending Machines**
• Allow remote monitoring of inventory levels
• Elastic pricing of products
• Contact-less payment using NFC
• Send the data to the cloud for predictive maintenance
• The information of inventory levels
• The information of the nearest machine in case a product goes out of stock in a machine

Logistics

- **Route Generation & Scheduling**
- Generate end-to-end routes using combination of route patterns
- Provide route generation queries
- Can be scale up to serve a large transportation network
- **Fleet Tracking**
- Track the locations of the vehicles in real-time
- Generate alerts for deviations in planned routes
- **Shipment monitoring**
- Monitoring the conditions inside containers
- Using sensors (temperature, pressure, humidity)
- Detecting food spoilage
- **Remote Vehicle Diagnostics**
- Detect faults in the vehicle
- Warn of impending faults
- IoT collects the data on vehicle (speed, engine RPM, coolant temperature)
- Generate alerts and suggest remedial actions

Agriculture

- **Smart Irrigation**
- Use sensors to determine the amount of moisture in the soil
- Release the flow of water
- Using predefined moisture levels
- Water Scheduling
- **Green House Control**
- Automatically control the climatological conditions inside a green house
- Using several sensors to monitor
- Using actuation devices to control
- Valves for releasing water and switches for controlling fans
- Maintenance of agricultural production

Industry

- **Machine Diagnosis**
- Sensors in machine monitor the operating conditions
- For example: temperature & vibration levels
- Collecting and analyzing massive scale machine sensor data
- For reliability analysis and fault prediction in machines
- **Indoor Air Quality Monitoring**
- Use various gas sensors
- To monitor the harmful and toxic gases (CO, NO, NO2 , etc.)

• Measure the environmental parameters to determine the indoor air quality
• Temperature, humidity, gaseous pollutants, aerosol

Health & Lifestyle
• Collect the health-care data
• Using some sensors: body temperature, heart rate, movement (with accelerometers), etc.
• Various forms: belts and wrist-bands
• Wearable electronic
• Assists the daily activities
•Smart watch
• Smart shoes
• Smart wristband

3)Explain in detail about Physical design of IOT with neat suitable diagram.

A)

Physical Design of IoT

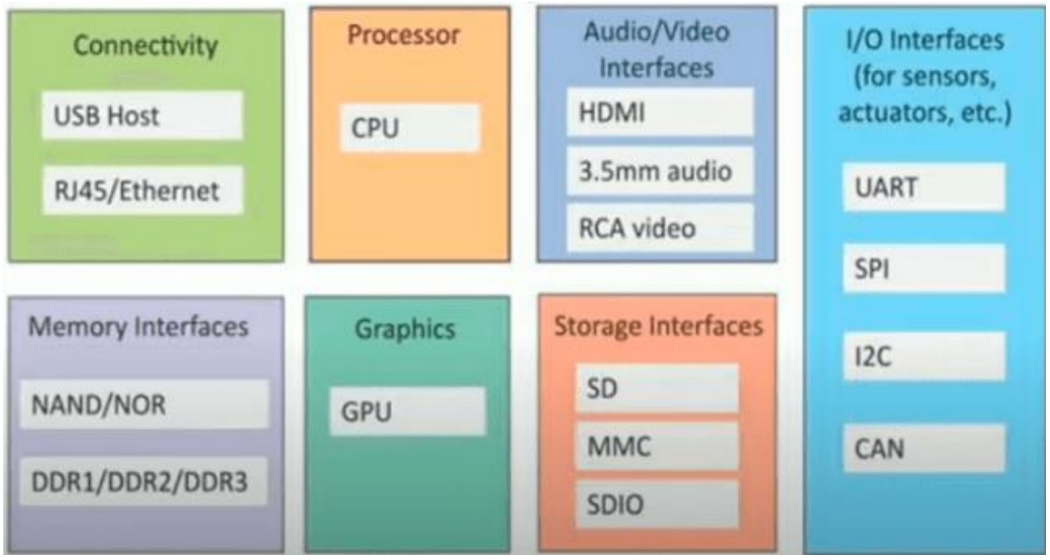
A physical design of an IoT system refers to the individual node devices and their protocols that are utilized to create a functional IoT ecosystem. Each node device can perform tasks such as remote sensing, actuating, monitoring, etc., by relying on physically connected devices.
It may also be capable of transmitting information through different types of wireless or wired connections. The things/devices in the IoT system are used for:

- Building connections
- Data processing
- Providing storage
- Providing interfaces
- Providing graphical interfaces

The devices generate data, and the data is used to perform analysis and do operations for improving the system. For instance, a moisture sensor is used to obtain the moisture data from a location, and the system analyses it to give an output.

Things/Devices

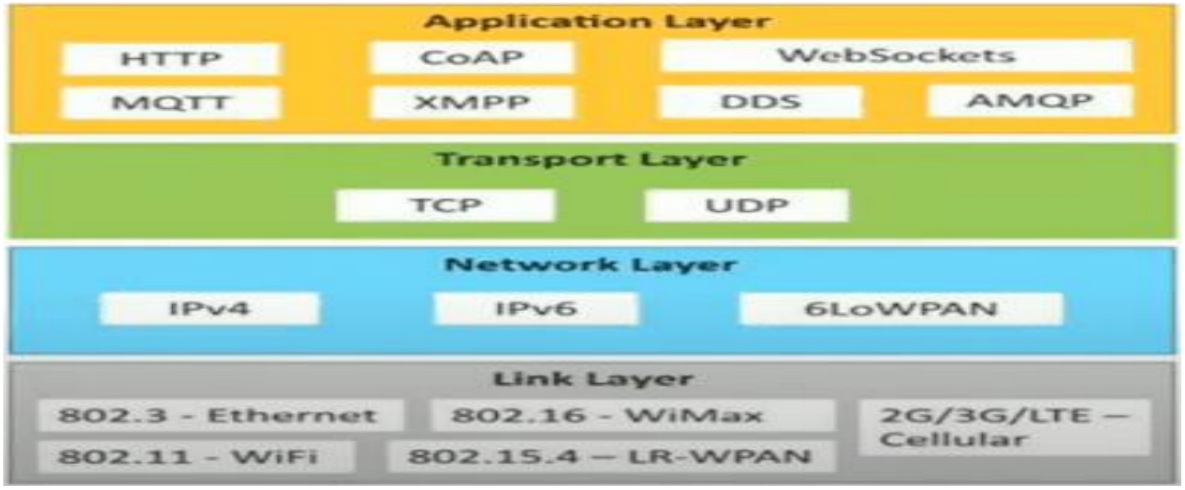
Things/Devices are used to build a connection, process data, provide interfaces, provide storage, and provide graphics interfaces in an IoT system. all these generate data in a form that can be analyzed by an analytical system and program to perform operations and used to improve the system. for example temperature sensor that is used to analyze the temperature generates the data from a location and is then determined by algorithms.



Connectivity:
Devices like USB hosts and ETHERNET are used for connectivity between the devices and the server.
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.
Audio/Video Interfaces: An interface like HDMI and RCA devices is used to record audio and videos in a system.
Input/Output interface: To give input and output signals to sensors, and actuators we use things like UART, SPI, CAN, etc.
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

IoT Protocols

These protocols are used to establish communication between a node device and a server over the internet. it helps to send commands to an IoT device and receive data from an IoT device over the internet. we use different types of protocols that are present on both the server and client side and these protocols are managed by network layers like application, transport, network, and link layer.



Application Layer protocol

In this layer, protocols define how the data can be sent over the network with the lower layer protocols using the application interface. these protocols include HTTP, WebSocket, XMPP, MQTT, DDS, and AMQP protocols.

HTTP

Hypertext transfer protocol is a protocol that presents in an application layer for transmitting media documents. it is used to communicate between web browsers and servers. it makes a request to a server and then waits till it receives a response and in between the request server does not keep any data between the two requests.

WebSocket

This protocol enables two-way communication between a client and a host that can be run on an untrusted code in a controlled environment. This protocol is commonly used by web browsers.

MQTT

It is a machine-to-machine connectivity protocol that was designed as a publish/subscribe messaging transport. and it is used for remote locations where a small code footprint is required.

Transport Layer

This layer is used to control the flow of data segments and handle error control. also, these layer protocols provide end-to-end message transfer capability independent of the underlying network.

TCP

The transmission control protocol is a protocol that defines how to establish and maintain a network that can exchange data in a proper manner using the internet protocol.

UDP

a user datagram protocol is a part of an internet protocol called the connectionless protocol. this protocol is not required to establish the connection to transfer data.

Network Layer

This layer is used to send datagrams from the source network to the destination network. we use IPv4 and IPv6 protocols as host identification that transfers data in packets.

IPv4

This is a protocol address that is a unique and numerical label assigned to each device connected to the network. an IP address performs two main functions host and location addressing. IPv4 is an IP address that is 32-bit long.

IPv6

It is a successor of IPv4 that uses 128 bits for an IP address. it is developed by the IETF task force to deal with long-anticipated problems.

Link Layer

Link-layer protocols are used to send data over the network's physical layer. it also determines how the packets are coded and signaled by the devices.

Ethernet

It is a set of technologies and protocols that are used primarily in LANs. it defines the physical layer and the medium access control for wired ethernet networks.

WiFi

It is a set of LAN protocols and specifies the set of media access control and physical layer protocols for implementing wireless local area networks.

4)Define sensor and mention different types of sensors available in the market.

A)

SENSORS

A sensor detects (senses) changes in the ambient conditions or in the state of another device or a system, and forwards or processes this information in a certain manner

They perform some input functions by sensing or feeling the physical changes in characteristics of a system in response to a stimuli.

For example heat is converted to electrical signals in a temperature sensor, or atmospheric pressure is converted to electrical signals in a barometer Transducers Transducers convert or transduce energy of one kind into another.

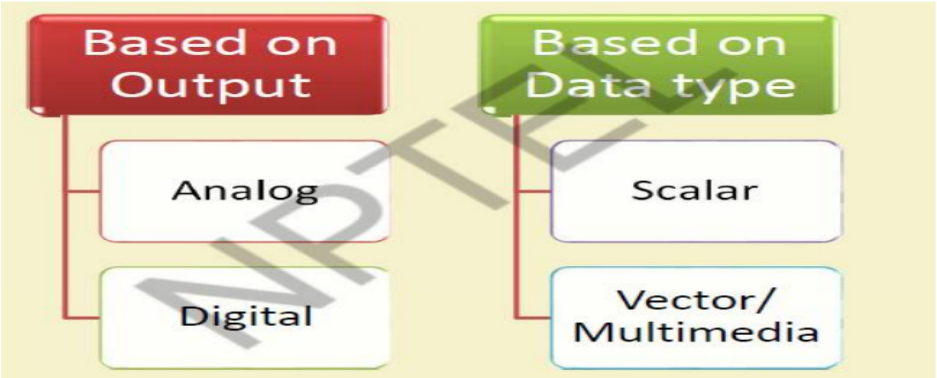
For example, in a sound system, a microphone (input device) converts sound waves into electrical signals for an amplifier to amplify (a process), and a loudspeaker (output device) converts these electrical signals back into sound waves.

Sensor Features

It is only sensitive to the measured property (e.g., A temperature sensor senses the ambient temperature of a room.)

It is insensitive to any other property likely to be encountered in its application (e.g., A temperature sensor does not bother about light or pressure while sensing the temperature.)

It does not influence the measured property (e.g., measuring the temperature does not reduce or increase the temperature).



Analog Sensors

Analog Sensors produce a continuous output signal or voltage which is generally proportional to the quantity being measured.

Physical quantities such as Temperature, Speed, Pressure, Displacement, Strain etc. are all analog quantities as they tend to be continuous in nature.

For example, the temperature of a liquid can be measured using a thermometer or thermocouple (e.g. in geysers) which continuously responds to temperature changes as the liquid is heated up or cooled down.

Digital Sensors

Digital Sensors produce discrete digital output signals or voltages that are a digital representation of the quantity being measured.
Digital sensors produce a binary output signal in the form of a logic "1" or a logic "0", ("ON" or "OFF").
Digital signal only produces discrete (non-continuous) values, which may be output as a single "bit" (serial transmission), or by combining the bits to produce a single "byte" output (parallel transmission).

Scalar Sensors

Scalar Sensors produce output signal or voltage which is generally proportional to the magnitude of the quantity being measured.

Physical quantities such as temperature, color, pressure, strain, etc. are all scalar quantities as only their magnitude is sufficient to convey an information.

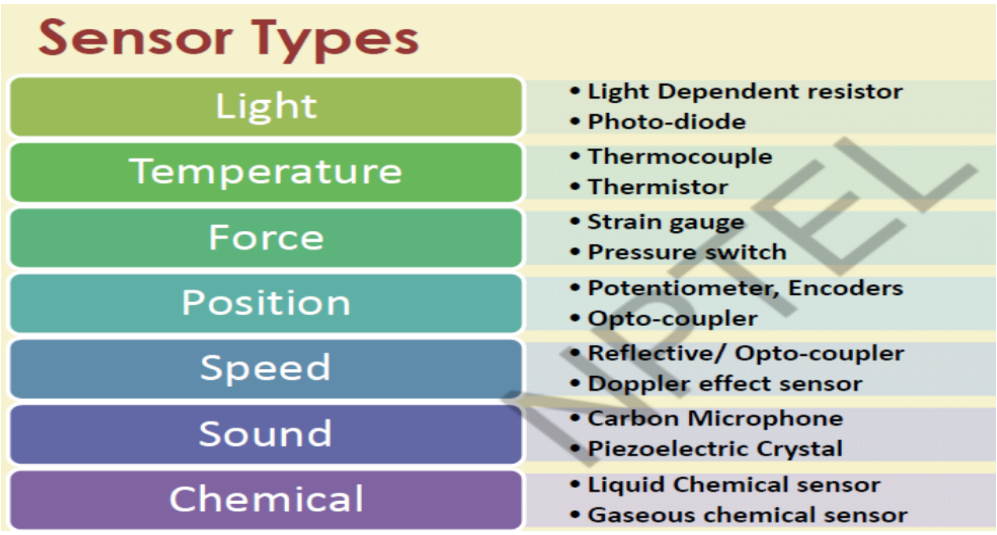
For example, the temperature of a room can be measured using a thermometer or thermocouple, which responds to temperature changes irrespective of the orientation of the sensor or its direction.

Vector Sensors

Vector Sensors produce output signal or voltage which is generally proportional to the magnitude, direction, as well as the orientation of the quantity being measured.

Physical quantities such as sound, image, velocity, acceleration, orientation, etc. are all vector quantities, as only their magnitude is not sufficient to convey the complete information.

For example, the acceleration of a body can be measured using an accelerometer, which gives the components of acceleration of the body with respect to the x,y,z coordinate axes.



5)Discuss IoT levels and deployment templates.

A)

An IoT system comprises of the following components:

• **Device:** An IoT device allows identification, remote sensing, actuating and remote monitoring capabilities. You learned about various examples of IoT devices in section.

• **Resource:** Resources are software components on the IoT device for accessing, processing, and storing sensor information, or controlling actuators connected to the device. Resources also include the software components that enable network access for the device.

• **Controller Service:** Controller service is a native service that runs on the device and interacts with the web services. Controller service sends data from the device to the web service and receives commands from the application (via web services) for controlling the device.

• **Database:** Database can be either local or in the cloud and stores the data generated by the IoT device.

• **Web Service:** Web services serve as a link between the IoT device, application, database, and analysis components. Web service can be either implemented using HTTP and REST principles (REST service) or using WebSocket protocol (WebSocket service).

• **Analysis Component:** The Analysis Component is responsible for analyzing the IoT data and generate results in a form which are easy for the user to understand.

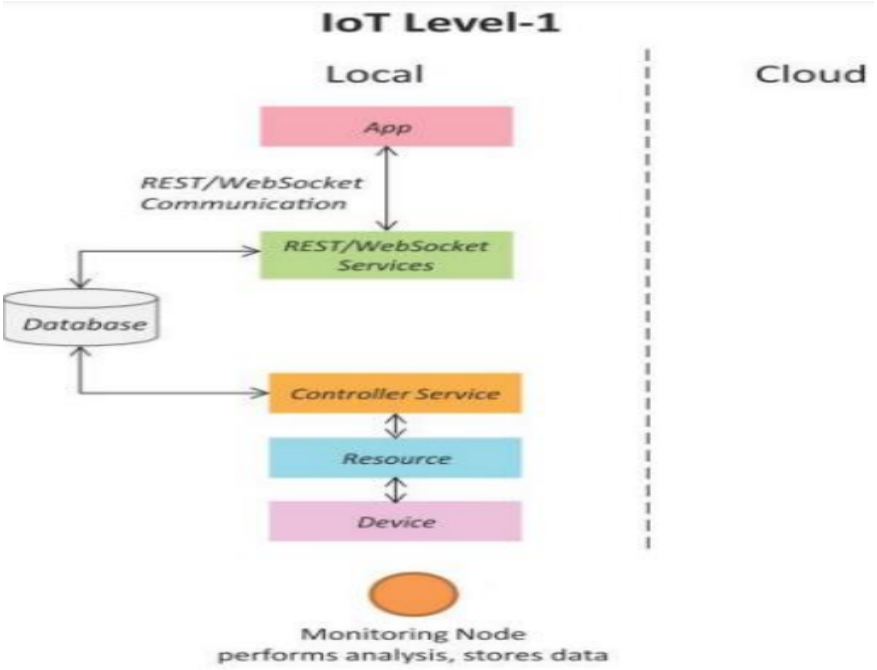
• **Application:** IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data.

IoT Levels

IOT – Level 1

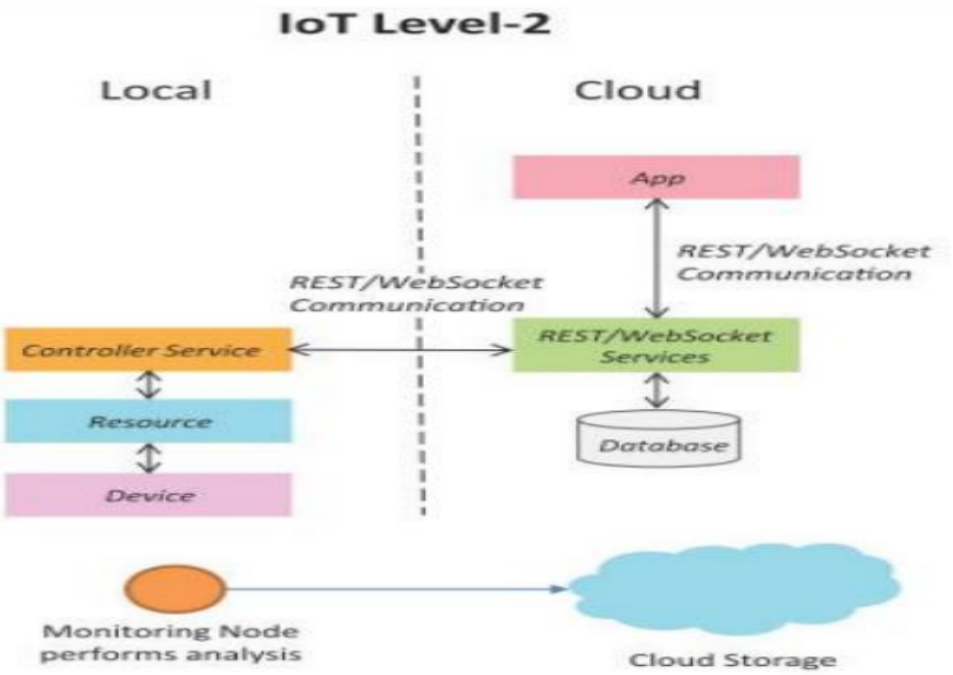
• A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application.

• Level-1 IoT systems are suitable for modelling low- cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive



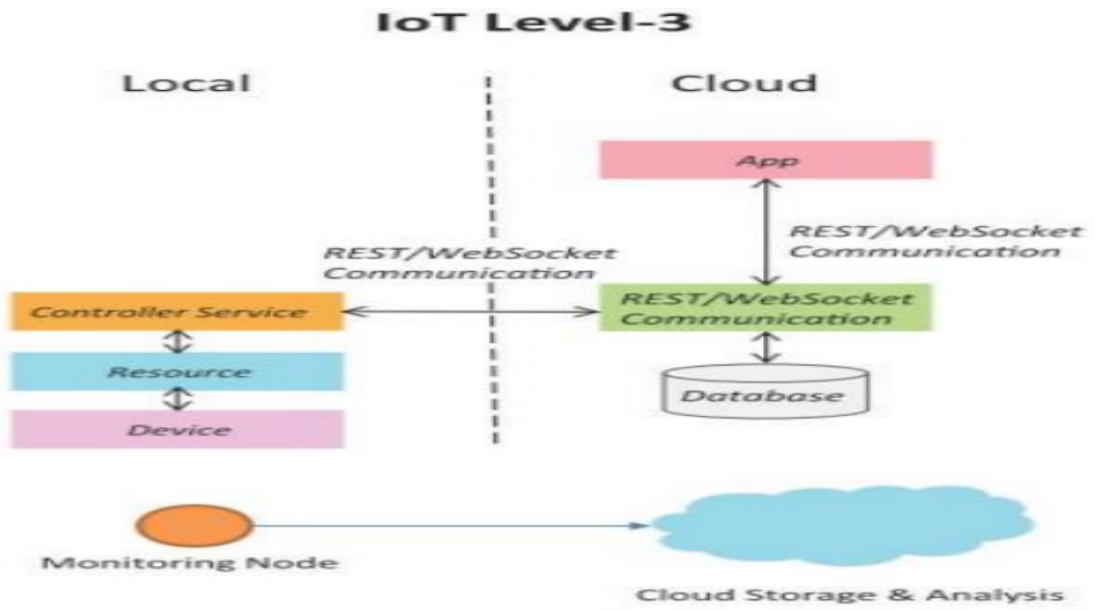
IOT Level-2

• 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.
• Level-2 IoT systems are sutable for solutions where the data involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself.



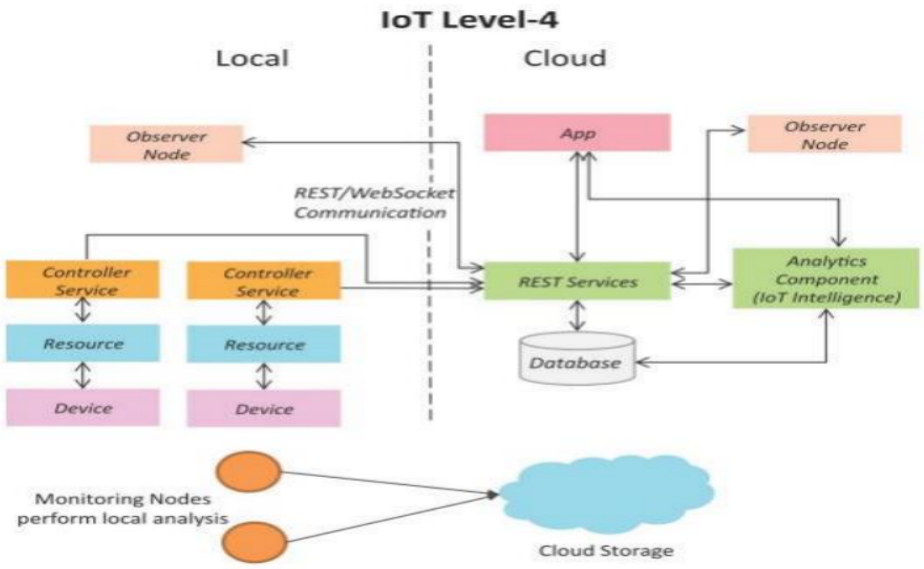
IOT Level-3

• A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and application is cloud- based.
• Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive.



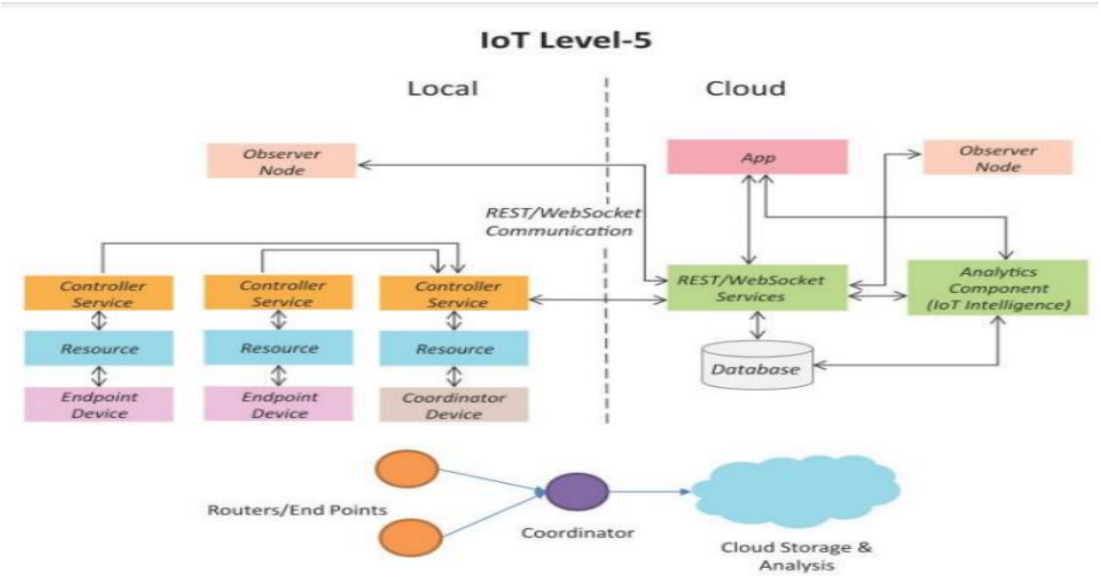
IOT Level-4

• A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud-based.
• Level-4 contains local and cloud- based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices.
• Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.



IOT Level -5

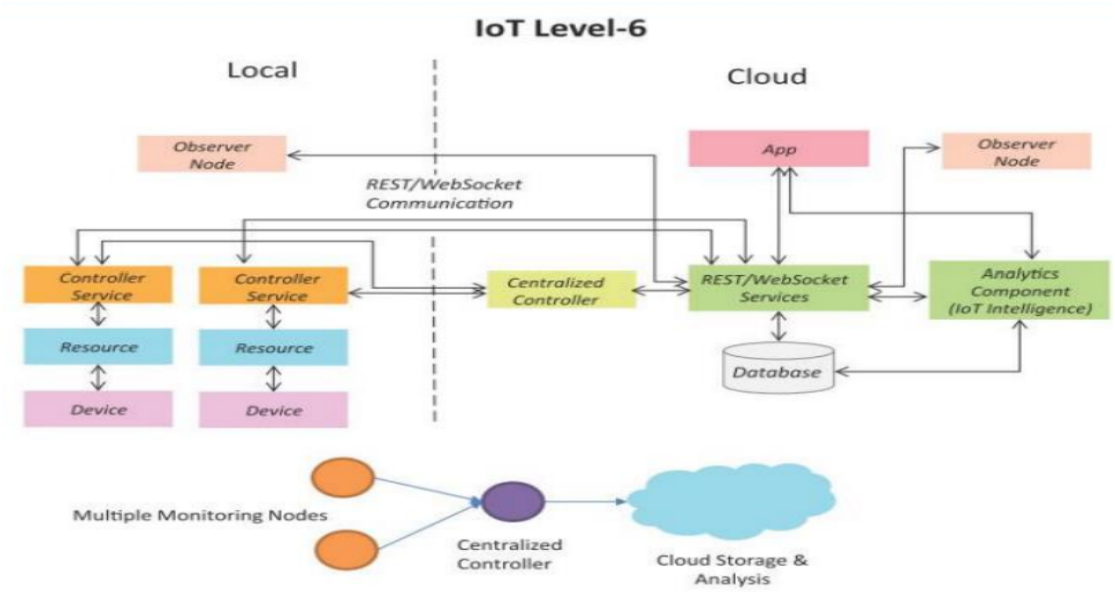
• A level-5 IoT system has multiple end nodes and one coordinator node.
• The end nodes that perform sensing and/or actuation.
• Coordinator node collects data from the end nodes and sends to the cloud.
• Data is stored and analyzed in the cloud and application is cloud-based.
• Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive.



IOT Level-6

• 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 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.



6)Explain in detail about Logical design of IOT with a neat suitable diagram.

A)

Logical Design of IoT

A logical design for an IoT system is the actual design of how its components (computers, sensors, and actuators) should be arranged to complete a particular function. It doesn't go into the depth of describing how each component will be built with low-level programming specifics.

IoT logical design includes:
1. IoT functional blocks
2. IoT communications models
3. IoT communication APIs

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. These functional blocks consist of devices that handle the communication between the server and the host, enable monitoring control functions, manage the data transfer, secure the IoT system using authentication and different functions, and provide an interface for controlling and monitoring various terms. The Functional blocks are:

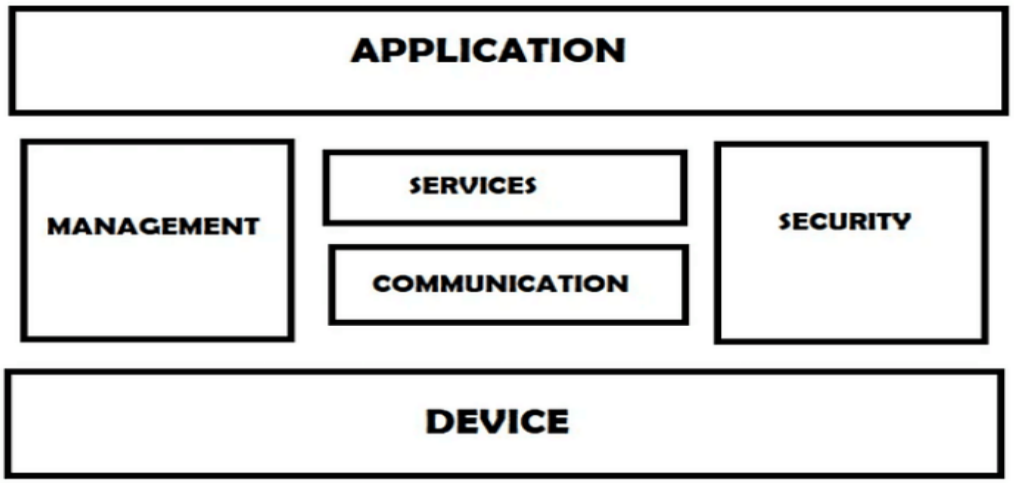
Device: An IoT system comprises of devices that provide sensing, actuation, monitoring, and control functions.

Communication: Handles the communication for the IoT system. Services: services for device monitoring, device control service, data publishing services, and services for device discovery.

Management: this block provides various functions to govern the IoT system.

Security: This block secures the IoT system and by providing functions such as authentication, authorization, message and content integrity, and data security.

Application: This is an interface that the users can use to control and monitor various aspects of the IoT system. The application also allows users to view the system status and view or analyze the processed data.



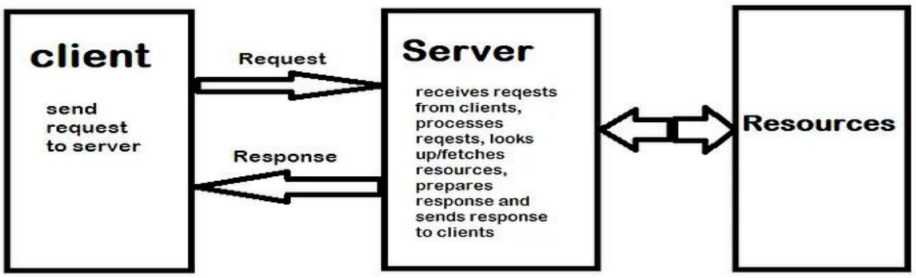
2. IoT Communication Models

There are multiple kinds of models available in an Internet of Things system that is used for communicating between the system and server,
such as:
Request-Response Model

Request-response model 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 representation, prepares the response, and then sends the response to the client. Request-response is a stateless communication model and each request-response pair is independent of the others.

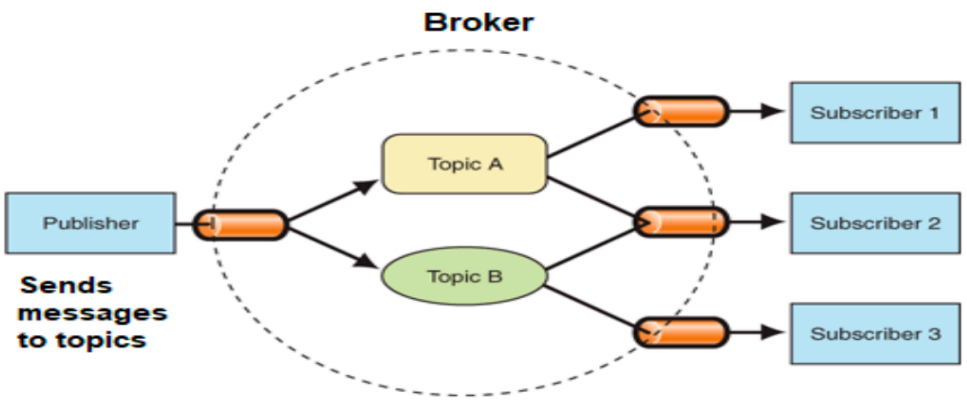
HTTP works as a request-response protocol between a client and a server. A web browser may be the client, and an application on a computer that hosts a website may be the server.

Example: A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.



Request-Response Communication Model

Publisher-Subscriber Model —
This model comprises three entities:
Publishers, Brokers, and Consumers.
Publishers are the source of data.
It sends the data to the topic which is managed by the broker. They are not aware of consumers.
Consumers subscribe to the topics which are managed by the broker. Brokers' responsibility is to accept data from publishers and send it to the appropriate consumers.
The broker only has the information regarding the consumer to which a particular topic belongs which the publisher is unaware.



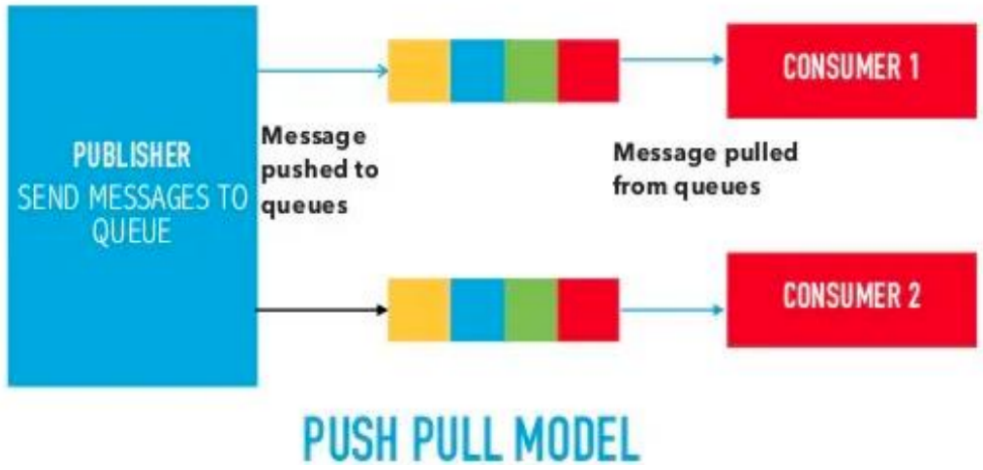
Push-Pull Model —

The push-pull model constitutes data publishers, data consumers, and data queues.

Publishers and Consumers are not aware of each other.

Publishers publish the message/data and push it into the queue. The consumers, present on the other side, pull the data out of the queue. Thus, the queue acts as the buffer for the message when the difference occurs in the rate of push or pull of data on the side of a publisher and consumer.

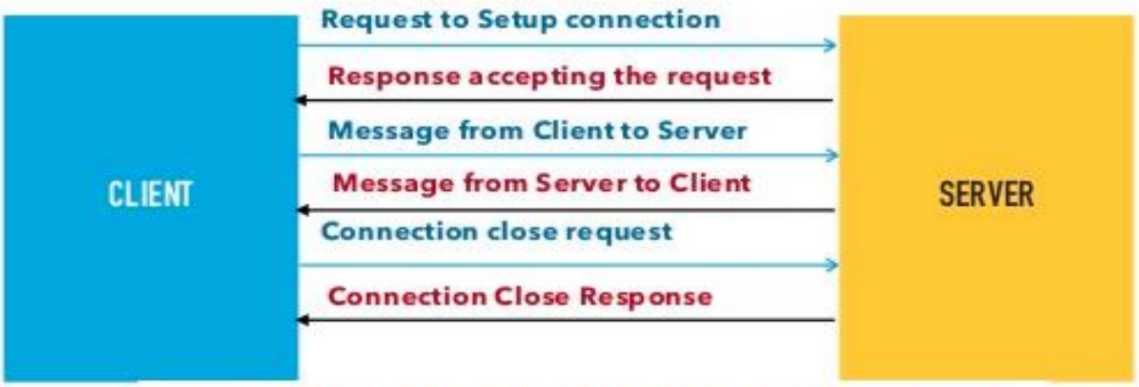
Queues help in decoupling the messaging between the producer and consumer. Queues also act as a buffer which helps in situations where there is a mismatch between the rate at which the producers push the data and consumers pull the data.



Exclusive Pair –

Exclusive Pair is the bi-directional model, including full-duplex communication between client and server. The connection is constant and remains open till the client sends a request to close the connection.

The Server has the record of all the connections which has been opened. This is a state-full connection model and the server is aware of all open connections. WebSocket-based communication API is fully based on this model.



EXCLUSIVE PAIR COMMUNICATION MODEL

3. IoT communication API

In IoT, there are 2 communication APIs –

REST — based Communication APIs

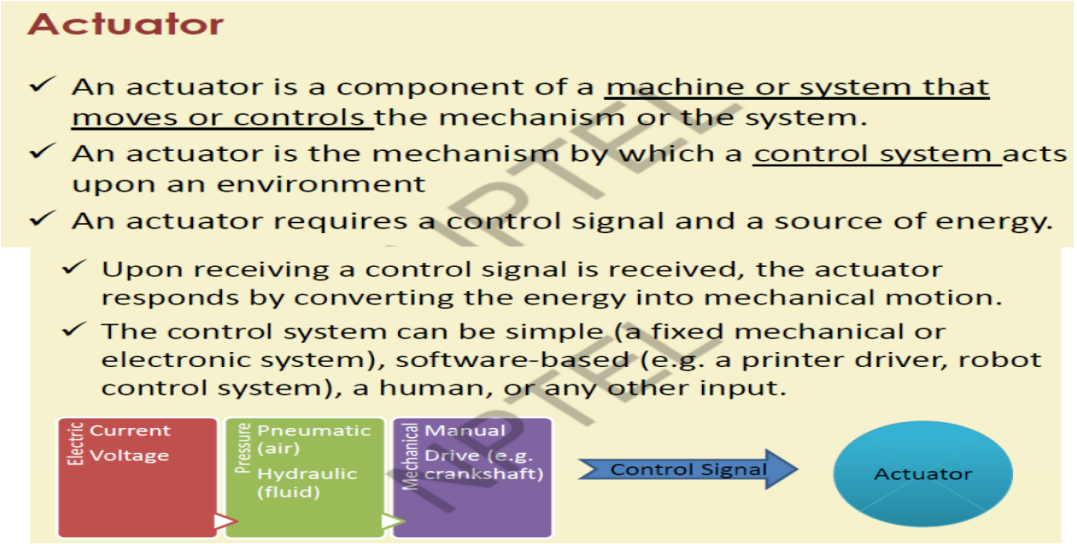
Web Socket — based Communication APIs Web service can either be implemented using REST principles or using Web Socket Protocol –

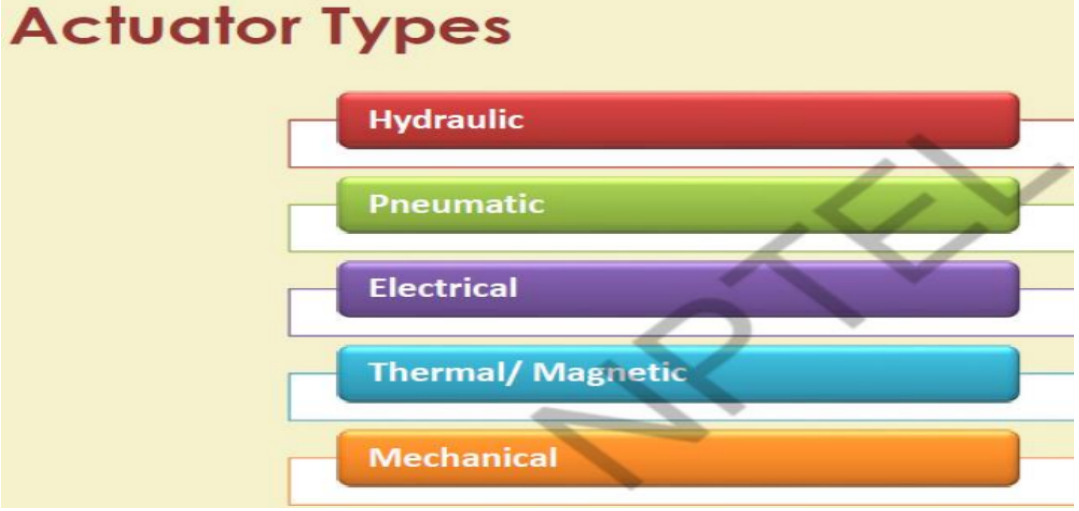
1. REST-Based Communication API: REpresentational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred. REST APIs follow the request-response communication model. The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.

2. Web Socket-Based Communication APIs: Web Socket APIs allow bi-directional, full-duplex communication between clients and servers. It follows the exclusive pair communication model. This Communication API does not require a new connection to be set up for each message to be sent between clients and servers. Once the connection is set up the messages can be

sent and received continuously without any interruption. WebSocket APIs are suitable for IoT Applications with low latency or high throughput requirements.

7) Define actuator and mention different types of actuators available in the market.





Hydraulic Actuators

- ✓ A hydraulic actuator consists of a cylinder or fluid motor that uses hydraulic power to facilitate mechanical operation.
- ✓ The mechanical motion is converted to linear, rotary or oscillatory motion.
- ✓ Since liquids are nearly impossible to compress, a hydraulic actuator exerts considerable force.
- ✓ The actuator's limited acceleration restricts its usage.

Pneumatic Actuators

- ✓ A pneumatic actuator converts energy formed by vacuum or compressed air at high pressure into either linear or rotary motion.
- ✓ Pneumatic rack and pinion actuators are used for valve controls of water pipes.
- ✓ Pneumatic energy quickly responds to starting and stopping signals.
- ✓ The power source does not need to be stored in reserve for operation.
- ✓ Pneumatic actuators enable large forces to be produced from relatively small pressure changes (e.g., Pneumatic brakes can are very responsive to small changes in pressure applied by the driver).
- ✓ It is responsible for converting pressure into force.

NPTEL

Electric Actuators

- ✓ An electric actuator is generally powered by a motor that converts electrical energy into mechanical torque.
- ✓ The electrical energy is used to actuate equipment such as solenoid valves which control the flow of water in pipes in response to electrical signals.
- ✓ Considered as one of the cheapest, cleanest and speedy actuator types available.

Thermal or Magnetic Actuators

- ✓ These can be actuated by applying thermal or magnetic energy.
- ✓ They tend to be compact, lightweight, economical and with high power density.
- ✓ These actuators use shape memory materials (SMMs), such as shape memory alloys (SMAs) or magnetic shape-memory alloys (MSMAs).
- ✓ Some popular manufacturers of these devices are *Finnish Modti Inc.* and *American Dynalloy*.

Soft Actuators

- ✓ Soft actuators (e.g. polymer based) are designed to handle fragile objects like fruit harvesting in agriculture or manipulating the internal organs in biomedicine.
- ✓ They typically address challenging tasks in robotics.
- ✓ Soft actuators produce flexible motion due to the integration of microscopic changes at the molecular level into a macroscopic deformation of the actuator materials.

8)Explain the various communication models used in IOT.

9)Explain the various levels will be used in deployment of IOT.

5&6 ANSWERS FOR BOTH 8,9 QUESTIONS.

10)Discuss the general block diagram of an IOT device.

A) **REFER 6 TH ANSWER**

UNIT-2

1)Explain the steps to interface any actuator to Arduino board?

Actuators

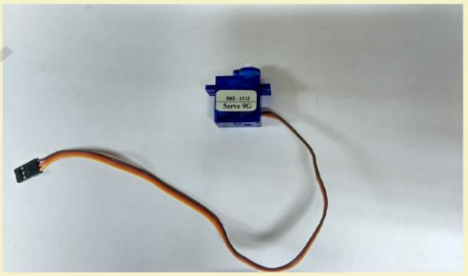
- Mechanical/Electro-mechanical device
- Converts energy into motion
- Mainly used to provide controlled motion to other components

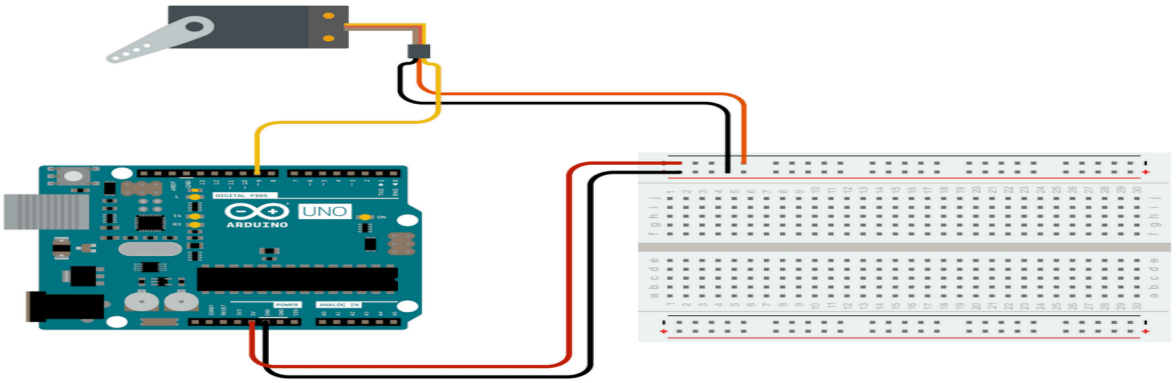
Basic Working Principle

Uses different combination of various mechanical structures like screws, ball bearings, gears to produce motion.

Servo Motor

- High precision motor
- Provides rotary motion 0 to 180 degree
- 3 wires in the Servo motor
  - Black or the darkest one is Ground
  - Red is for power supply
  - Yellow for signal pin





Servo Library on Arduino

- Arduino provides different library- SERVO to operate the servo motor
- Create an instance of servo to use it in the sketch

```
Servo myservo;
```

NPTEL

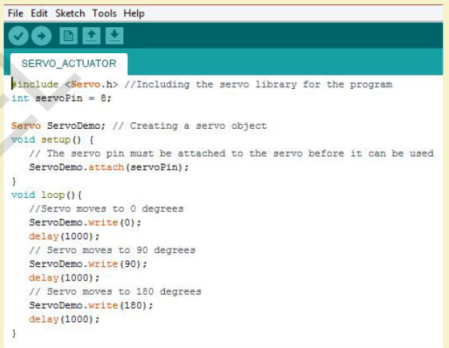
Sketch: SERVO_ACTUATOR

```
#include <Servo.h>
//Including the servo library for the program
int servoPin = 12;

Servo ServoDemo; // Creating a servo object
void setup() {
  // The servo pin must be attached to the servo
before it can be used
  ServoDemo.attach(servoPin);
}
```

```
void loop(){
  //Servo moves to 0 degrees
   ServoDemo.write(0);
   delay(1000);

   // Servo moves to 90 degrees
   ServoDemo.write(90);
   delay(1000);

  // Servo moves to 180 degrees
   ServoDemo.write(180);
   delay(1000);
}
```

Sketch: SERVO_ACTUATOR (contd..)

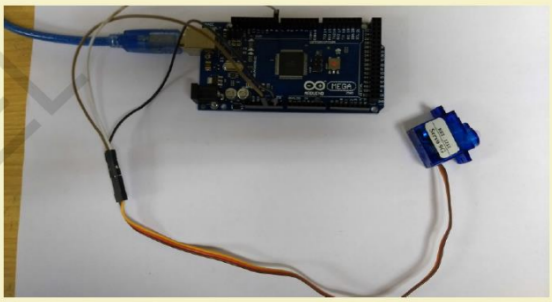
- Create an instance of Servo
- The instance must be attached to the pin before being used in the code
- Write() function takes the degree value and rotates the motor accordingly



NPTEL

Connection

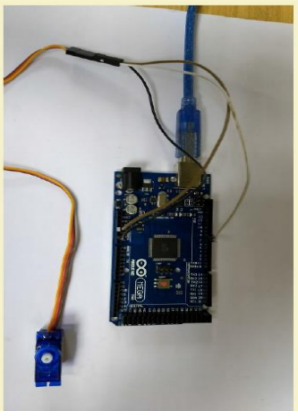
- Connect the Ground of the servo to the ground of the Arduino board.
- Connect the power supply wire to the 5V pin of the board.
- Connect the signal wire to any digital output pin (we have used pin 8).



NPTEL

Output

The motor turns 0, 90 and 180 degrees with a delay of 1 second each.



NPTEL

2) Explain the functions of digital I/O, analog I/O & advanced I/O in Arduino programming, with an example.

A)

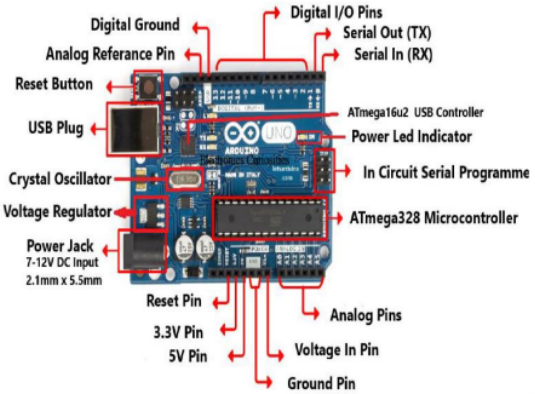
Interfacing with analog and digital devices

Digital I/O

Digital I/O (input/output) pins on an Arduino board can be used to read and write digital signals. Digital signals are either HIGH (5V) or LOW (0V). Digital I/O pins can be used to connect to a variety of devices, such as switches, buttons, LEDs, and motors.

Example:

```
// Set pin 13 to output mode
pinMode(13, OUTPUT);
// Turn on the LED
digitalWrite(13, HIGH);
// Wait for 1 second
delay(1000);
// Turn off the LED
digitalWrite(13, LOW);
```



Interfacing with analog and digital devices (Contd.)

Analog I/O

Analog I/O pins on an Arduino board can be used to read and write analog signals. Analog signals are continuous voltages that can have any value between 0V and 5V. Analog I/O pins can be used to connect to a variety of devices, such as temperature sensors, light sensors, and potentiometers.

Example:

```
// Set pin A0 to input mode
pinMode(A0, INPUT);
// Read the value of the potentiometer
 int potentiometerValue = analogRead(A0);
// Print the potentiometer value to the serial monitor
Serial.println(potentiometerValue);
```

Interfacing with analog and digital devices (Contd.)

Advanced I/O

Some Arduino boards also have advanced I/O features, such as pulse-width modulation (PWM), serial communication, and I2C communication.

•PWM: PWM is a technique for generating variable voltage signals. PWM signals can be used to control the speed and brightness of motors and LEDs.

•Serial communication: Serial communication is a way to send and receive data between devices over a serial interface. Serial communication can be used to connect to a variety of devices, such as computers, GPS modules, and Bluetooth modules.

•I2C communication: I2C communication is a two-wire serial communication protocol. I2C communication can be used to connect to a variety of devices, such as sensors, actuators, and displays.

Interfacing with analog and digital devices (Contd.)

```
// Set pin 10 to PWM output mode
pinMode(10, OUTPUT);
// Set the PWM frequency to 50Hz
analogWriteFrequency(10, 50);
// Set the PWM duty cycle to 50%
analogWrite(10, 128);
```

3) Explain the steps to interface a digital sensor to Arduino board with an example?

A)

working with sensors and actuators

Sensors

- Electronic elements
- Converts physical quantity/ measurements into electrical signals
- Can be analog or digital

Types of Sensors

Some commonly used sensors:

- Temperature
- Humidity
- Compass
- Light
- Sound
- Accelerometer

NPTEL

working with sensors and actuators (Contd.)

Sensor Interface with Arduino

- Digital Humidity and Temperature Sensor (DHT)
- PIN 1, 2, 3, 4 (from left to right)
  - PIN 1- 3.3V-5V Power supply
  - PIN 2- Data
  - PIN 3- Null
  - PIN 4- Ground

NPTEL

working with sensors and actuators (Contd.)

DHT Sensor Library

- Arduino supports a special library for the DHT11 and DHT22 sensors
- Provides function to read the temperature and humidity values from the data pin

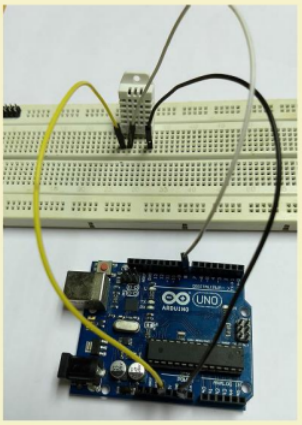
```
dht.readHumidity()
dht.readTemperature()
```

NPTEL

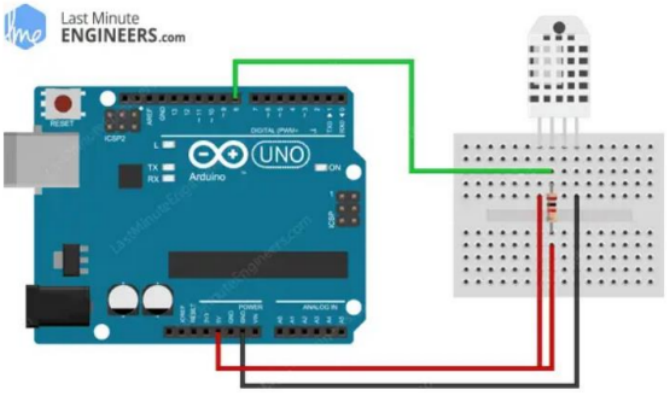
working with sensors and actuators (Contd.)

Connection

- Connect pin 1 of the DHT to the 3.3 V supply pin in the board
- Data pin (pin 2) can be connected to any digital pin, here 12
- Connect pin 4 to the ground (GND) pin of the board



NPTEL



Wiring DHT22 to Arduino UNO

working with sensors and actuators (Contd.)

Sketch: DHT_SENSOR (contd..)

```
#include <DHT.h>;
DHT dht(8, DHT22);     //Initialize DHT sensor
float humidity;        //Stores humidity value
float temperature;     //Stores temperature
value
void setup()
{
 Serial.begin(9600);
 dht.begin();
}
void loop()
{
  //Read data from the sensor and store it to variables
humidity and temperature
  humidity = dht.readHumidity();
  temperature= dht.readTemperature();
  //Print temperature and humidity values to serial
monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("%, Temperature: ");
  Serial.print(temperature);
  Serial.println(" Celsius");
  delay(2000); //Delay of 2 seconds
}
```

NPTEL

working with sensors and actuators (Contd.)

Output

The readings are printed at a delay of 2 seconds as specified by the delay() function

```
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
Humidity: 65.80%, Temperature: 26.00 Celsius
```

NPTEL

4)Explain about the Arduino programming control structures with an example.

A)

Arduino programming utilizes several control structures to manage the flow of code execution. These include:

Conditional Statements (if, else if, else): These structures allow you to make decisions based on certain conditions.

```
int sensorValue = analogRead(A0); // Reading analog sensor connected to pin A0


if (sensorValue < 500) {
    // Do something if sensor value is less than 500
    digitalWrite(LED_BUILTIN, HIGH); // Turn on the built-in LED
} else if (sensorValue >= 500 && sensorValue < 800) {
    // Do something else if sensor value is between 500 and 800
    digitalWrite(LED_BUILTIN, LOW); // Turn off the built-in LED
} else {
    // Do something if the above conditions are not met
    // ...
}
```

2)Loops (for, while, do-while): These structures enable repetitive execution of a block of code.

```
// Example of a for loop to blink an LED 5 times
for (int i = 0; i < 5; i++) {
    digitalWrite(LED_BUILTIN, HIGH); // Turn on the LED
    delay(500); // Wait for 500 milliseconds
    digitalWrite(LED_BUILTIN, LOW); // Turn off the LED
    delay(500); // Wait for another 500 milliseconds
}
```

3) Switch Case Statements: These are used to select one of many code blocks to be executed.

```
int option = 2; // Assuming this value comes from some condition

switch (option) {
    case 1:
        // Code to execute if option is 1
        break;
    case 2:
        // Code to execute if option is 2
        break;
    case 3:
        // Code to execute if option is 3
        break;
    default:
        // Code to execute if option doesn't match any case
        break;
}
```

5)Write a program to implement push button controlled Led on Aruduino Uno Board.

A)

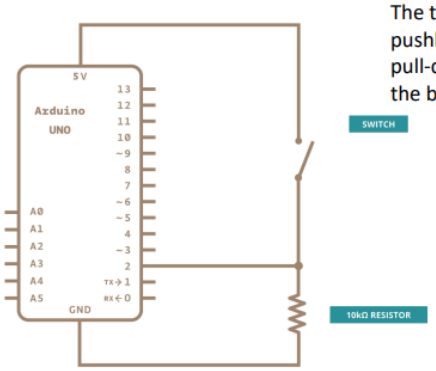
PUSH BUTTON CONTROLLED LED

Pushbuttons or switches connect two points in a circuit when you press them. This example turns on the built-in LED on pin 13 when you press the button.

Hardware

•Arduino Board
•Momentary button or Switch
•10K ohm resistor
•hook-up wires
•breadboard

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to 5 volts, so that we read a HIGH.



Connect three wires to the board. The first two, red and black, connect to the two long vertical rows on the side of the breadboard to provide access to the 5 volt supply and ground. The third wire goes from digital pin 2 to one leg of the pushbutton. That same leg of the button connects through a pull-down resistor (here 10K ohm) to ground. The other leg of the button connects to the 5 volt supply.

You can also wire this circuit the opposite way, with a pullup resistor keeping the input HIGH, and going LOW when the button is pressed. If so, the behavior of the sketch will be reversed, with the LED normally on and turning off when you press the button.

If you disconnect the digital I/O pin from everything, the LED may blink erratically. This is because the input is "floating" - that is, it will randomly return either HIGH or LOW. That's why you need a pull-up or pull-down resistor in the circuit.

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;  // the number of the pushbutton pin
const int ledPin = 13;    // the number of the LED pin

// variables will change:
int buttonState = 0;  // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

6) Explain about the comparison operators with an example.

Comparison operators in programming (including Arduino) are used to compare two values and determine the relationship between them. Here are some common comparison operators:

1)Equal to (==): Checks if two values are equal.

int a = 5;

int b = 10;

```
if (a == b) {
    // This condition won't be true, as a is not equal to b
    // Code here won't execute
} else {
    // Code in this block will execute
    // Since a is not equal to b
}
```

2)Not equal to (!=): Checks if two values are not equal.

```
int x = 3;
int y = 3;

if (x != y) {
    // This condition won't be true, as x is equal to y
    // Code here won't execute
} else {
    // Code in this block will execute
    // Since x is equal to y
}
```

3)Greater than (>): Checks if one value is greater than another.

```
int m = 8;
int n = 5;
if (m > n) {
```

```
    // This condition is true, as m is greater than n
    // Code here will execute
} else {
    // Code in this block won't execute
}
```

4)Less than (<): Checks if one value is less than another.

```
int p = 10;
int q = 15;

if (p < q) {
    // This condition is true, as p is less than q
    // Code here will execute
} else {
    // Code in this block won't execute
}
```

5)Greater than or equal to (>=) and Less than or equal to (<=): Check if one value is greater than or equal to, or less than or equal to, another value, respectively.

```
int num1 = 7;
int num2 = 7;

if (num1 >= num2) {
    // This condition is true, as num1 is greater than or equal to num2
    // Code here will execute
```

```
} else {
    // Code in this block won't execute
}
```

7)Discuss the steps involved in the process of interfacing a Bluetooth module to Arduino Uno.

Interfacing a Bluetooth module to Arduino Uno allows for wireless communication between the Arduino board and a Bluetooth-enabled device, such as a smartphone or computer. This enables real-time data exchange and remote control capabilities for Arduino projects. The process typically involves connecting the necessary hardware components, uploading an Arduino sketch, and establishing a Bluetooth connection.

1. Hardware Components:
    - Arduino Uno board
    - HC-05 Bluetooth module
    - Jumper wires
    - Breadboard (optional)

2. Hardware Connections:
    - Connect the VCC pin of the HC-05 module to the 5V pin of the Arduino Uno.
    - Connect the GND pin of the HC-05 module to the GND pin of the Arduino Uno.
    - Connect the RX pin of the HC-05 module to any available digital pin on the Arduino Uno (e.g., pin 10).
    - Connect the TX pin of the HC-05 module to another available digital pin on the Arduino Uno (e.g., pin 11).

3. Arduino Sketch:

    - Open the Arduino IDE software.
    - Create a new sketch.
    - Include the SoftwareSerial library to handle software-based serial communication:
    - C++

```
#include <SoftwareSerial.h>
```

    -
    -
    - Define the pins connected to the Bluetooth module:
    - C++

```
#define RX_PIN 10
#define TX_PIN 11
```

    -
    -
    - Create a SoftwareSerial object for communication with the Bluetooth module:
    - C++

```
SoftwareSerial bluetoothSerial(RX_PIN, TX_PIN);
```

    -
    -
    - Initialize the Bluetooth module in slave mode:
    - C++

```
void setup() {
  bluetoothSerial.begin(9600);
  // Set pins as input or output
}
```

    -
    -
    - Read incoming data from the Bluetooth module:

- C++

```
void loop() {
  if (bluetoothSerial.available()) {
    char incomingData = bluetoothSerial.read();
    // Process incoming data
  }
}
```

-
-
- Send data to the Bluetooth module:

C++

```
    bluetoothSerial.println("Hello from Arduino!");
```

4 .Upload the sketch to the Arduino Uno board.

5 .Bluetooth Connection:

- Pair your Bluetooth device with the HC-05 module using the default passcode "1234".
- Open a Bluetooth terminal application on your device and connect to the HC-05 module.
- Send and receive data through the Bluetooth terminal to verify communication with the Arduino Uno.

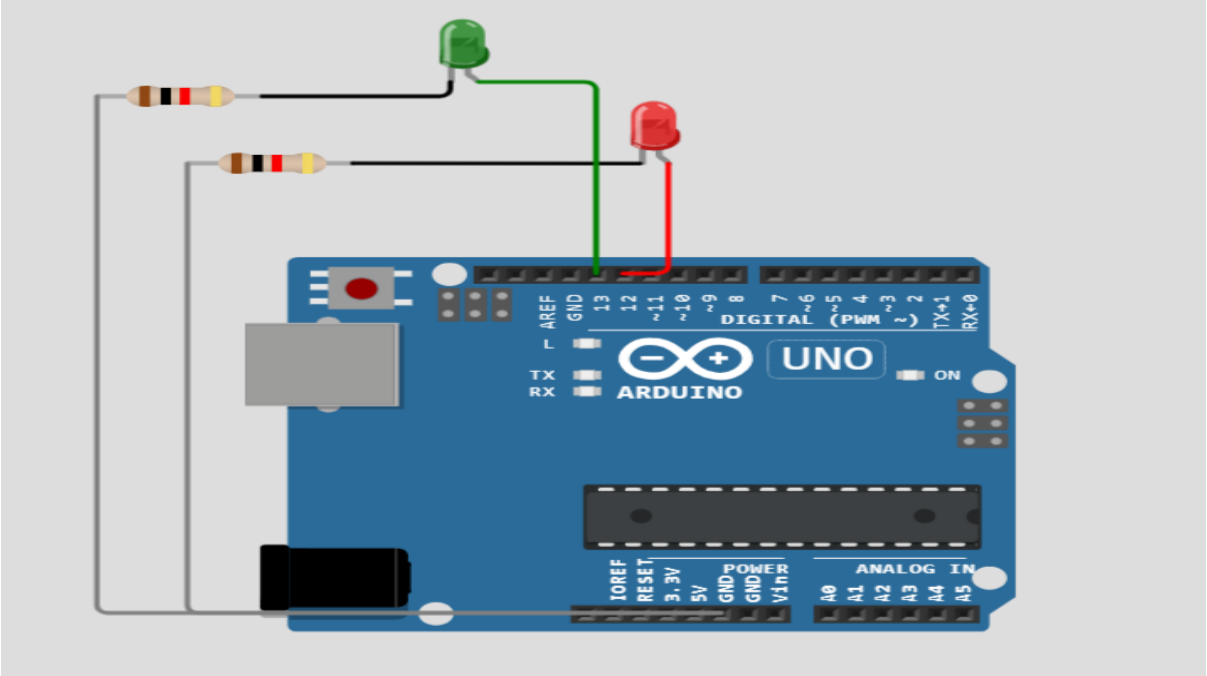
8)Write a program to interface a Humidity sensor to Aruduino Uno Board with a neat circuit diagram.

A) REFER 3rd QUESTION

9)Write a program to interface an LED to Arduino board and it should be ON for 1Sec and OFF for 2Sec continuously, draw a neat circuit diagram.

A)

```
int ledPin = 12; // change this value to 13 to blink green led

// setup() function is for initialization
// it only runs once
void setup() {
  // pinMode() function set a certain pin to INPUT or OUTPUT
  pinMode(ledPin, OUTPUT); // set pin #8 as OUTPUT pin
}

// loop() function, like its name, keep running over and over (looping)
void loop() {
  // digitalWrite() function outputs a binary signal (HIGH or LOW) to certain pin
  digitalWrite(ledPin, HIGH); // write a digital signal HIGH to pin 8
  // This will cause light ON

  // delay() function causes CPU to pause processing (do nothing)
  // it delays in millisecond
  // 1 second = 1000 millisecond
  delay(1000); // CPU pauses for 1000 millisecond (1 second)
  // Light will remain ON for 1 second

  digitalWrite(ledPin, LOW); // write a digital signal LOW to pin 8
  delay(1000); // CPU pauses for 1000 millisecond (1 second)
}
```

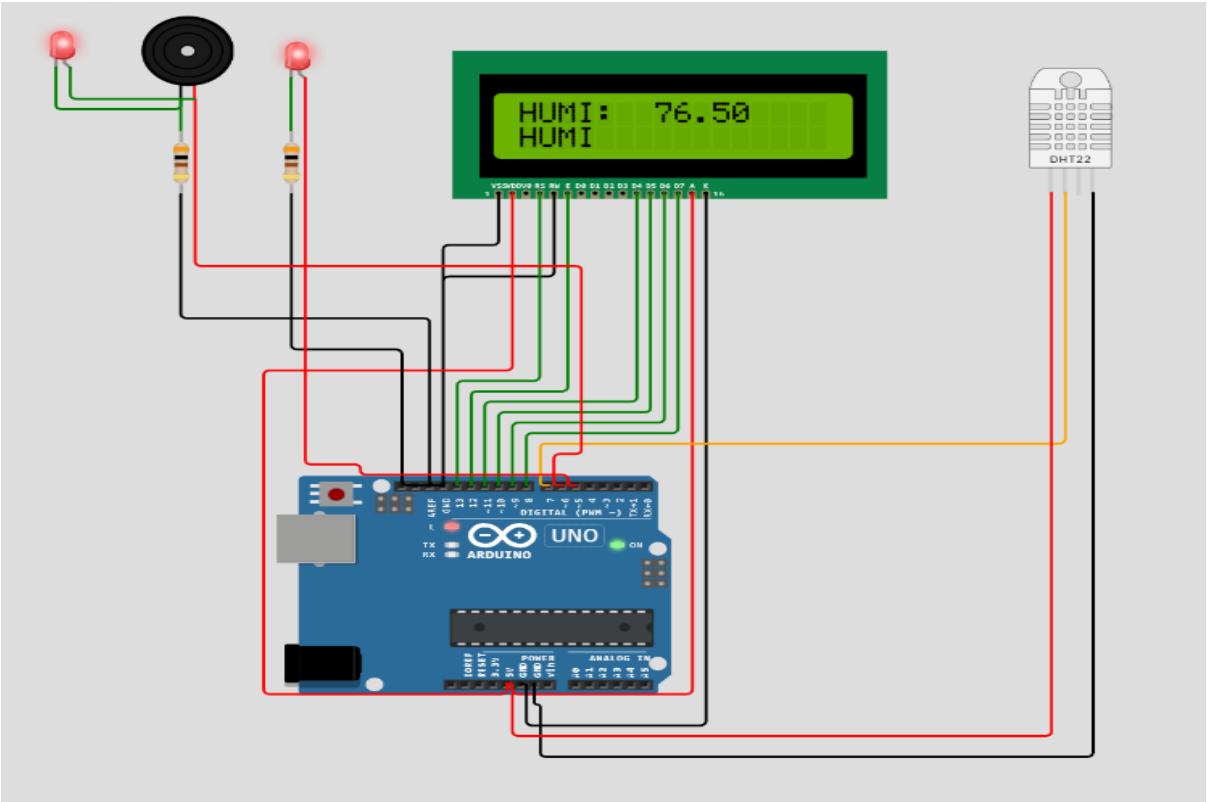


10) Write a program to interface a Humidity sensor and its values will be displayed on LCD, with a neat circuit diagram.

A)
```
#include <LiquidCrystal.h>
#include <dht.h>


int RS = 13;
int EN = 12;
```

```
int D4 = 11;
int D5 = 10;
int D6 = 9;
int D7 = 8;
int buzzer = 6;
int led = 5;
int datapin = 7;
dht DHT;
LiquidCrystal lcd (RS, EN, D4, D5, D6, D7);
int count = 0;




void setup() {

  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("WELCOME TO");
  delay(1000);

  lcd.setCursor(0, 0);
  lcd.print("Bn Abdulsalam.");
  delay(100);

  lcd.setCursor(0, 1);
  lcd.print("ELECT ELECT.");
  delay(100);
  lcd.clear ();


}

void loop () {
  int readData = DHT.read22 (datapin);
  float TEMP = DHT.temperature;
  float HUMI = DHT.humidity;


  if (TEMP > 30) {
    {digitalWrite(buzzer,HIGH);}
    lcd.setCursor(0, 0);
```

```
  lcd.print ("TEMP:  ");
  lcd.print (TEMP);
  lcd.setCursor(0,1);
  lcd.print ("TEMP TOO HIGH");
  lcd.clear ();
 if (TEMP<30){digitalWrite(buzzer, LOW);}

 }


if (HUMI>50){{digitalWrite(led,HIGH);}
 lcd.setCursor(0, 0);
 lcd.print ("HUMI:  ");
 lcd.print (HUMI);
 lcd.setCursor(0, 1);
 lcd.print ("HUMI TOO HIGH");

 }



}
```



UNIT-3

1)Explain RFID and NFC protocols in detail and mention its applications?

A)

Introduction

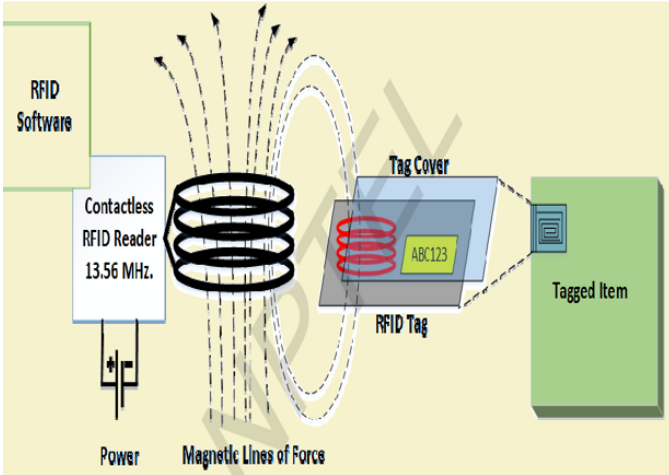
- ✓ RFID is an acronym for "radio-frequency identification"
- ✓ Data digitally encoded in RFID tags, which can be read by a reader.
- ✓ Somewhat similar to barcodes.
- ✓ Data read from tags are stored in a database by the reader.
- ✓ As compared to traditional barcodes and QR codes, RFID tag data can be read outside the line-of-sight.

RFID Features

- ✓ RFID tag consists of an integrated circuit and an antenna.
- ✓ The tag is covered by a protective material which also acts as a shield against various environmental effects.
- ✓ Tags may be passive or active.
- ✓ Passive RFID tags are the most widely used.
- ✓ Passive tags have to be powered by a reader inductively before they can transmit information, whereas active tags have their own power supply.

Working Principle

- ✓ Derived from Automatic Identification and Data Capture (AIDC) technology.
- ✓ AIDC performs object identification, object data collection and mapping of the collected data to computer systems with little or no human intervention.
- ✓ AIDC uses wired communication
- ✓ RFID uses radio waves to perform AIDC functions.
- ✓ The main components of an RFID system include an RFID tag or smart label, an RFID reader, and an antenna.

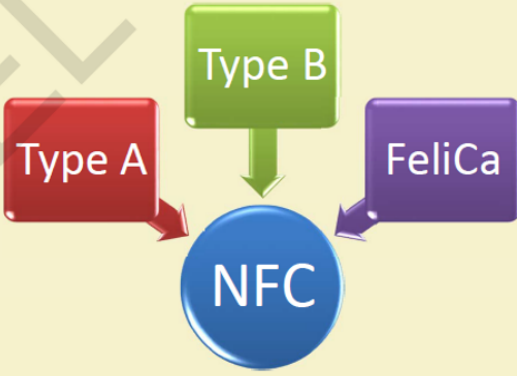


Applications

- ✓ Inventory management
- ✓ Asset tracking
- ✓ Personnel tracking
- ✓ Controlling access to restricted areas
- ✓ ID badging
- ✓ Supply chain management
- ✓ Counterfeit prevention (e.g. in the pharmaceutical industry)

Introduction

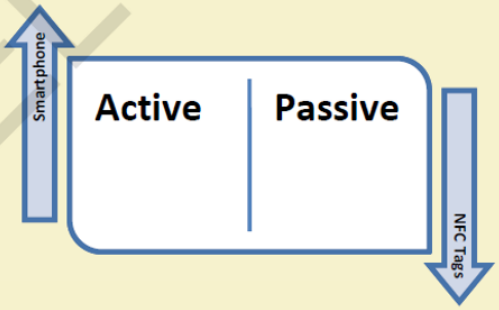
- ✓ **Near field communication**, or NFC for short, is an offshoot of radio-frequency identification (RFID).
- ✓ NFC is designed for use by devices within **close proximity** to each other.
- ✓ All NFC types are similar but communicate in slightly different ways.
- ✓ FeliCa is commonly found in Japan.



NPTEL

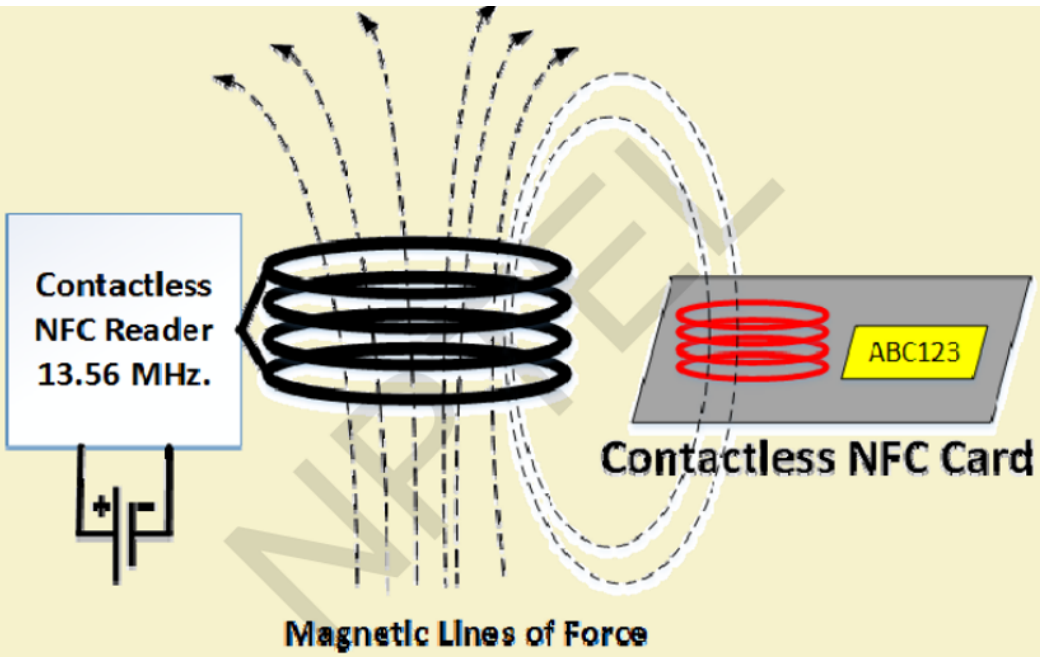
NFC Types

- ✓ **Passive devices** contain information which is readable by other devices, however it cannot read information itself.
- ✓ NFC tags found in supermarket products are examples of passive NFC.
- ✓ **Active devices** are able to collect as well as transmit information.
- ✓ Smartphones are a good example of active devices.



Working Principle

- ✓ Works on the principle of **magnetic induction**.
- ✓ A reader emits a small electric current which creates a magnetic field that in turn bridges the physical space between the devices.
- ✓ The generated field is received by a similar coil in the client device where it is turned back into electrical impulses to communicate data such as identification number status information or any other information.
- ✓ 'Passive' NFC tags use the energy from the reader to encode their response while 'active' or 'peer-to-peer' tags have their own power source.



NPTEL

NFC Specifications

- ✓ NFC's data-transmission frequency is 13.56MHz.
- ✓ NFC can transmit data at a rate of either 106, 212 or 424 Kbps (kilobits per second).
- ✓ Tags typically store between 96 and 512 bytes of data.
- ✓ Communication range is less than 20cms.

Modes of Operation

Peer-to-peer — Lets two smartphones swap data

Read/Write — One active device picks up info from a passive one

Card emulation — NFC device can be used like a contactless credit card

NFC Applications

- ✓ Smartphone based payments.
- ✓ Parcel tracking.
- ✓ Information tags in posters and advertisements.
- ✓ Computer game synchronized toys.
- ✓ Low-power home automation systems.

2)Explain Bluetooth protocol in detail and mention its applications?

A)

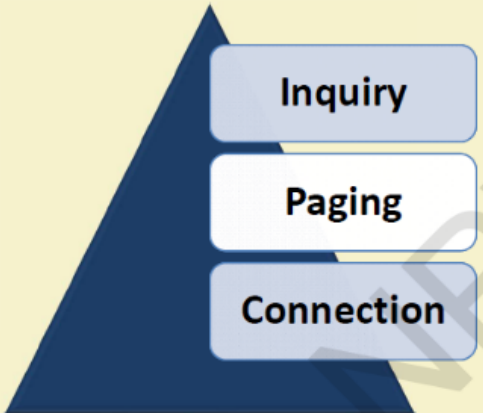
Bluetooth Low Energy (BLE)

Features

- ✓ Bluetooth technology operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHZ.
- ✓ Uses spread spectrum hopping, full-duplex signal at a nominal rate of 1600 hops/sec.
- ✓ Bluetooth supports 1Mbps data rate for version 1.2 and 3Mbps data rate for Version 2.0 combined with Error Data Rate.

- ✓ Bluetooth operating range depends on the device:
  - **Class 3** radios have a range of up to 1 meter or 3 feet
  - **Class 2** radios are most commonly found in mobile devices have a range of 10 meters or 30 feet
  - **Class 1** radios are used primarily in industrial use cases have a range of 100 meters or 300 feet.

Connection Establishment



Inquiry run by one Bluetooth device to try to **discover other devices near it.**

Process of **forming a connection** between two Bluetooth devices.

A device either actively **participates** in the network or enters a low-power sleep mode.

NPTEL

Modes

Actively transmitting or receiving data.

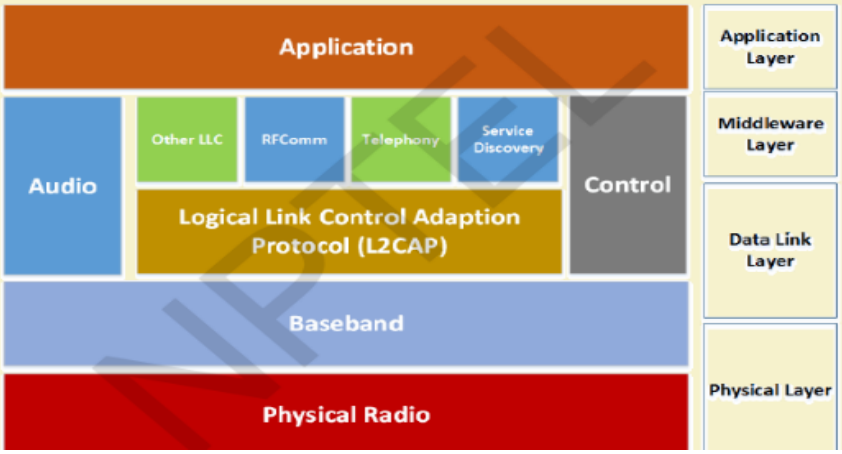
Sleeps and only listens for transmissions at a set interval .

Power-saving mode where a device sleeps for a defined period and then returns back to active mode .

Slave will become inactive until the master tells it to wake back up.

ce: "Bluetooth Basics", Tutorials, Sparkfun.com (Online)

Protocol Stack



Baseband

- ✓ Physical layer of the Bluetooth.
- ✓ Manages physical channels and links.
- ✓ Other services include:
  - Error correction
  - Data whitening
  - Hop selection
  - Bluetooth security
- ✓ Manages asynchronous and synchronous links.
- ✓ Handles packets, paging and inquiry.

NPTEL

L2CAP

- ✓ The Logical Link Control and Adaptation Protocol (L2CAP).
- ✓ Layered over the Baseband Protocol and resides in the data link layer.
- ✓ Used to multiplex multiple logical connections between two devices.
- ✓ Provides connection-oriented and connectionless data services to upper layer protocols.
- ✓ Provides:
  - Protocol multiplexing capability
  - Segmentation and reassembly operation
  - Group abstractions

RFComm

- ✓ Radio Frequency Communications (RFCOMM).
- ✓ It is a cable replacement protocol used for generating a virtual serial data stream.
- ✓ RFCOMM provides for binary data transport .
- ✓ Emulates EIA-232 (formerly RS-232) control signals over the Bluetooth baseband layer, i.e. it is a serial port emulation.
- ✓ RFCOMM provides a simple reliable data stream to the user, similar to TCP.
- ✓ Supports up to 60 simultaneous connections between two BT devices.

Service Discovery Protocol (SDP)

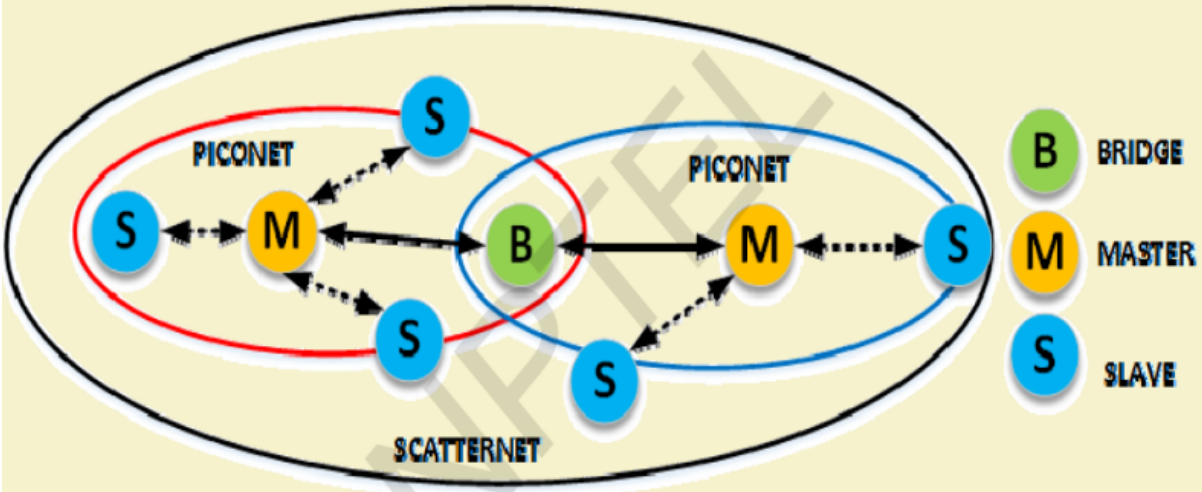
- ✓ Enables applications to discover available services and their features.
- ✓ Addresses the unique characteristics of the Bluetooth environment such as, dynamic changes in the quality of services in RF proximity of devices in motion.
- ✓ Can function over a reliable packet transfer protocol.
- ✓ Uses a request/response model.

NPTEL

Piconets

- ✓ Bluetooth enabled electronic devices connect and communicate wirelessly through short range networks known as **Piconets**.
- ✓ Bluetooth devices exist in small ad-hoc configurations with the ability to act either as master or slave.
- ✓ Provisions are in place, which allow for a **master** and a **slave** to switch their roles.
- ✓ The simplest configuration is a point to point configuration with one master and one slave.

- ✓ When more than two Bluetooth devices communicate with one another, it is called a **PICONET**.
- ✓ A Piconet can contain up to seven slaves clustered around a single master.
- ✓ The device that initializes establishment of the Piconet becomes the **master**.
- ✓ The master is responsible for transmission control by dividing the network into a series of time slots amongst the network members, as a part of **time division multiplexing** scheme.



NPTEL

Features of Piconet

- ✓ Within a Piconet, the clock and unique **48-bit address** of master determines the timing of various devices and the frequency hopping sequence of individual devices.
- ✓ Each Piconet device supports 7 simultaneous connections to other devices.
- ✓ Each device can communicate with several piconets simultaneously.
- ✓ Piconets are established dynamically and automatically as Bluetooth enabled devices enter and leave piconets.

- ✓ There is no direct connection between the slaves.
- ✓ All connections are either master-to-slave or slave-to-master.
- ✓ Slaves are allowed to transmit once these have been polled by the master.
- ✓ Transmission starts in the slave-to-master time slot immediately following a polling packet from the master.
- ✓ A device can be a member of two or more Piconets.

- ✓ A device can be a slave in one Piconet and master in another. It however cannot be a master in more than once Piconets.
- ✓ Devices in adjacent Piconets provide a bridge to support inner-Piconet connections, allowing assemblies of linked Piconets to form a physically extensible communication infrastructure known as **Scatternet**.

Applications

- ✓ Audio players
- ✓ Home automation
- ✓ Smartphones
- ✓ Toys
- ✓ Hands free headphones

3)Describe about Application Layer Protocols: (a) CoAP (b) MQTT

A)

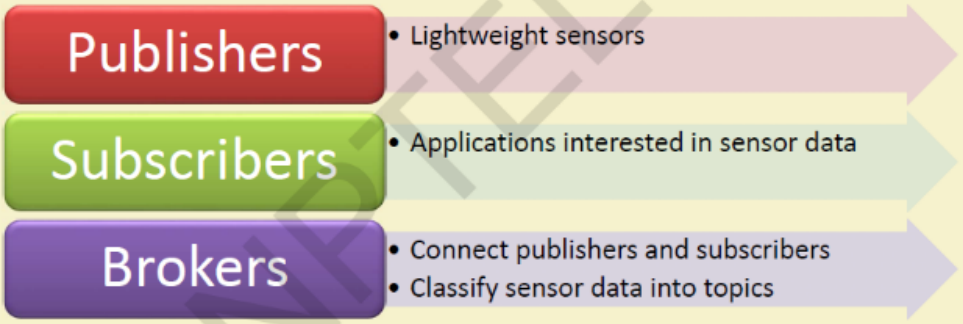
MQTT

Introduction

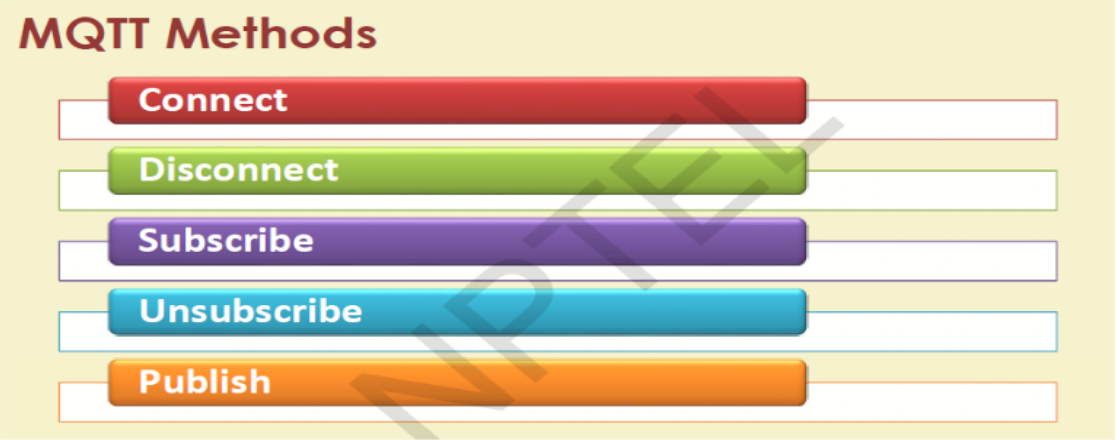
- ✓ **Message Queue Telemetry Transport.**
- ✓ ISO standard (ISO/IEC PRF 20922).
- ✓ It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
- ✓ MQTT was introduced by IBM in 1999 and standardized by OASIS in 2013.
- ✓ Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and networks and communications on the other side.
- ✓ A message broker controls the publish-subscribe messaging pattern.
- ✓ A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker.
- ✓ Designed for:
  - Remote connections
  - Limited bandwidth
  - Small-code footprint

MQTT (Contd.)

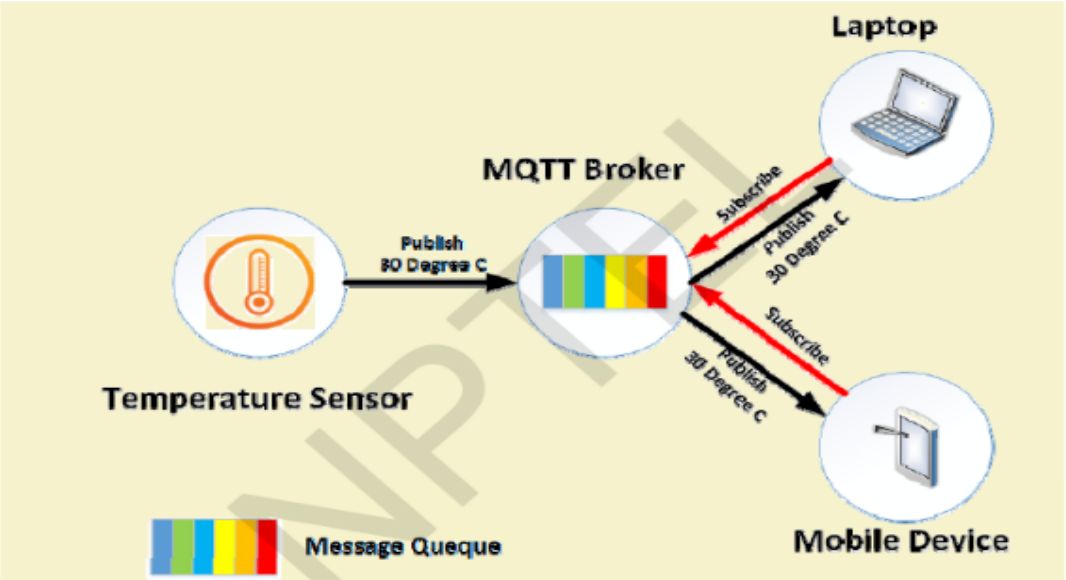
MQTT Components



MQTT (Contd.)



MQTT (Contd.)



Communication

- ✓ The protocol uses a **publish/subscribe** architecture (HTTP uses a request/response paradigm).
- ✓ Publish/subscribe is **event-driven** and enables messages to be pushed to clients.
- ✓ The central **communication point is the MQTT broker**, which is in charge of dispatching all messages between the senders and the rightful receivers.
- ✓ Each client that publishes a message to the broker, includes a **topic** into the message. The **topic is the routing information for the broker.**
- ✓ Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client.
- ✓ Therefore the clients don't have to know each other. They only communicate over the topic.
- ✓ This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

MQTT Topics

- ✓ A topic is a **simple string** that can have more hierarchy leve which are separated by a slash.
- ✓ A sample topic for sending temperature data of the living room could be *house/living-room/temperature*.
- ✓ On one hand the client (e.g. mobile device) can subscribe t the exact topic or on the other hand, it can use a **wildcard**.
- ✓ The subscription to *house/+/temperature* would result in all messages sent to the previously mentioned topic *house/living-room/temperature,* as well as any topic with an arbitrary value i the place of living room, such as *house/kitchen/temperature*.
- ✓ The plus sign is a **single level wild card** and only allows arbitrary values for one hierarchy.
- ✓ If more than one level needs to be subscribed, such as, the enti sub-tree, there is also a **multilevel wildcard** (#).
- ✓ It allows to subscribe to all underlying hierarchy levels.

Applications

- ✓ **Facebook Messenger** uses MQTT for online chat.
- ✓ **Amazon Web Services** use Amazon IoT with MQTT.
- ✓ **Microsoft Azure** IoT Hub uses MQTT as its main protocol for telemetry messages.
- ✓ The **EVRYTHNG IoT platform** uses MQTT as an M2M protocol for millions of connected products.
- ✓ **Adafruit** launched a free MQTT cloud service for IoT experimenters called Adafruit IO.

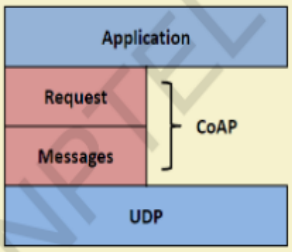
CoAP

Introduction

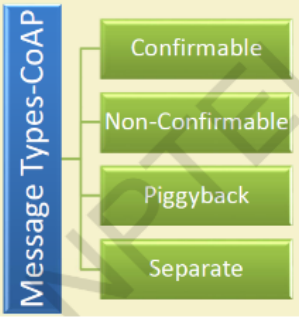
- ✓ CoAP – **Constrained Application Protocol**.
- ✓ **Web transfer protocol** for use with constrained nodes and networks.
- ✓ **Designed for Machine to Machine** (M2M) applications such as smart energy and building automation.
- ✓ Based on **Request-Response model** between end-points
- ✓ Client-Server interaction is **asynchronous over a datagram oriented transport protocol** such as UDP
- ✓ The Constrained Application Protocol (CoAP) is a session layer protocol designed by IETF Constrained RESTful Environment (CoRE) working group to provide lightweight RESTful (HTTP) interface.
- ✓ Representational State Transfer (REST) is the standard interface between HTTP client and servers.
- ✓ Lightweight applications such as those in IoT, could result in significant overhead and power consumption by REST.
- ✓ CoAP is designed to enable low-power sensors to use RESTful services while meeting their power constraints.

- ✓ Built over UDP, instead of TCP (which is commonly used with HTTP) and has a light mechanism to provide reliability.
- ✓ CoAP architecture is divided into two main sub-layers:
  - Messaging
  - Request/response.
- ✓ The messaging sub-layer is responsible for reliability and duplication of messages, while the request/response sub-layer is responsible for communication.
- ✓ CoAP has four messaging modes:
  - Confirmable
  - Non-confirmable
  - Piggyback
  - Separate

CoAP Position

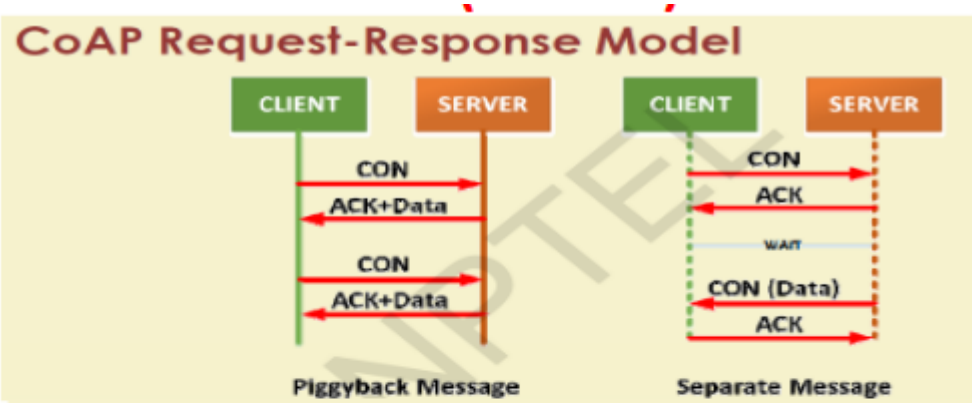


CoAP Message Types



NPTEL

- ✓ Confirmable and non-confirmable modes represent the reliable and unreliable transmissions, respectively, while the other modes are used for request/response.
- ✓ Piggyback is used for client/server direct communication where the server sends its response directly after receiving the message, i.e., within the acknowledgment message.
- ✓ On the other hand, the separate mode is used when the server response comes in a message separate from the acknowledgment, and may take some time to be sent by the server.
- ✓ Similar to HTTP, CoAP utilizes GET, PUT, PUSH, DELETE messages requests to retrieve, create, update, and delete, respectively



Features

- ✓ Reduced overheads and parsing complexity.
- ✓ URL and content-type support.
- ✓ Support for the discovery of resources provided by known CoAP services.
- ✓ Simple subscription for a resource, and resulting push notifications.
- ✓ Simple caching based on maximum message age.

4)Explain Zigbee protocol in detail and mention the advantages over IEEE 802.15.4 ?

ZigBee

Features of ZigBee

- ✓ Most widely deployed enhancement of IEEE 802.15.4.
- ✓ The ZigBee protocol is defined by **layer 3 and above**. It works with the 802.15.4 layers 1 and 2.
- ✓ The standard uses layers 3 and 4 to define additional communication enhancements.
- ✓ These enhancements include authentication with valid nodes, encryption for security, and a data routing and forwarding capability that enables mesh networking.
- ✓ The most popular use of ZigBee is wireless sensor networks using the mesh topology.

ZigBee (Contd.)

ZigBee (Contd.)

Important Components

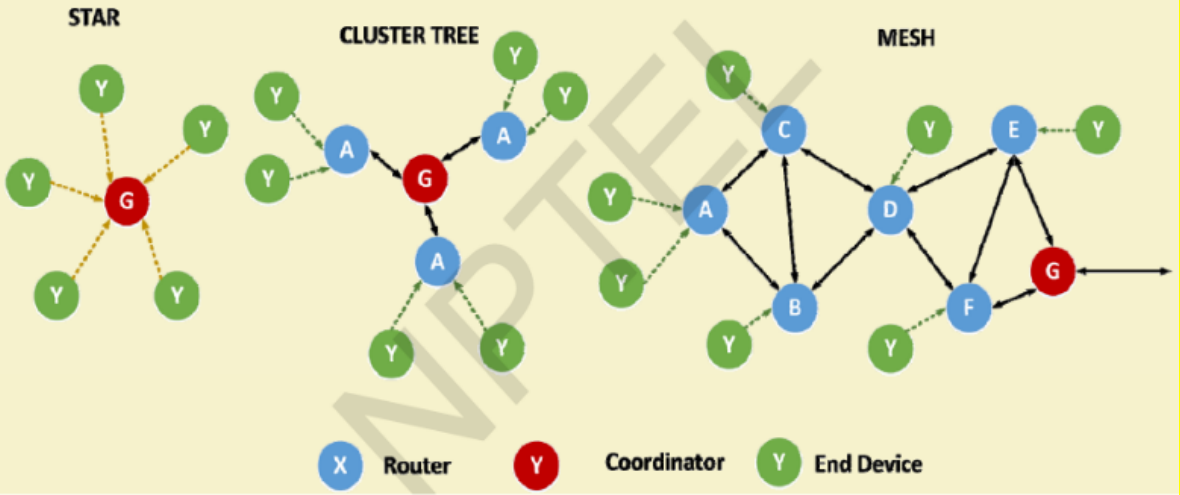
ZDO

- ZigBee Device Object
  (Device management, Security, Policies)

APS

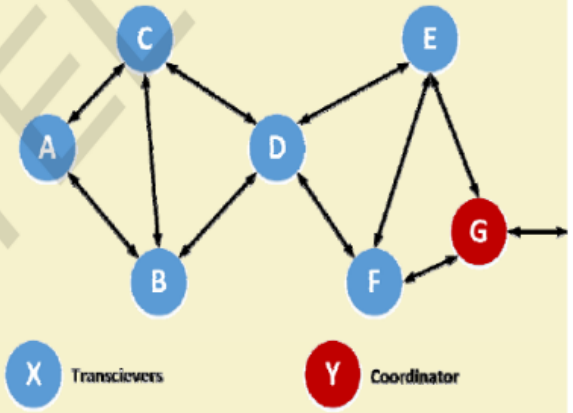
- Application Support Sub-layer
  (Interfacing and control services, bridge between network and other layers)

ZigBee Topologies



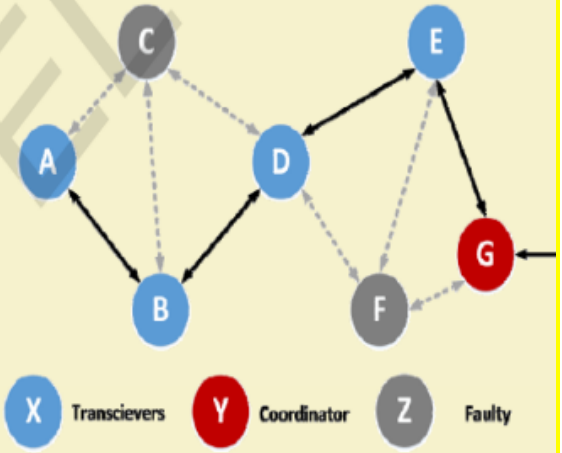
ZigBee Mesh

- ✓ In a mesh, any node can communicate with any other node within its range.
- ✓ If nodes are not in range, messages are relayed through intermediate nodes.
- ✓ This allows the network deployment over large areas.



ZigBee Mesh (Contd.)

- ✓ Meshes have increased network reliability.
- ✓ For example, if nodes C and F are down, the message packets from A can still be relayed to G via B and E.
- ✓ ZigBee mesh networks are self-



ZigBee Types

- ✓ *ZigBee Coordinator (ZC)*:
  - The Coordinator forms the root of the ZigBee network tree and might act as a bridge between networks.
  - There is a single ZigBee Coordinator in each network, which originally initiates the network.
  - It stores information about the network under it and outside it.
  - It acts as a Trust Center & repository for security keys.
- ✓ *ZigBee Router (ZR)*:
  - Capable of running applications, as well as relaying information between nodes connected to it.
- ✓ *ZigBee End Device (ZED)*:
  - It contains just enough functionality to talk to the parent node, and it cannot relay data from other devices.
  - This allows the node to be asleep a significant amount of the time thereby enhancing battery life.
  - Memory requirements and cost of ZEDs are quite low, as compared to ZR or ZC.

ırces:

ZigBee Network Layer

- ✓ The network layer uses Ad Hoc On-Demand Distance Vector (AODV) routing.
- ✓ To find the final destination, the AODV broadcasts a route request to all its immediate neighbors.
- ✓ The neighbors relay the same information to their neighbors, eventually spreading the request throughout the network.
- ✓ Upon discovery of the destination, a low-cost path is calculated and

Applications

- ✓ Building automation
- ✓ Remote control (RF4CE or RF for consumer electronics)
- ✓ Smart energy for home energy monitoring
- ✓ Health care for medical and fitness monitoring
- ✓ Home automation for control of smart homes
- ✓ Light Link for control of LED lighting
- ✓ Telecom services

NPTEL

5) Describe the features of 6LowPAN and mention routing considerations in it?

6LowPAN

Introduction

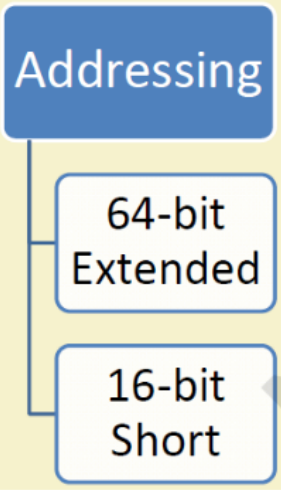
- ✓ Low-power Wireless Personal Area Networks over IPv6.
- ✓ Allows for the smallest devices with limited processing ability to transmit information wirelessly using an Internet protocol.
- ✓ Allows low-power devices to connect to the Internet.
- ✓ Created by the Internet Engineering Task Force (IETF) - RFC 5933 and RFC 4919.

Features of 6LoWPANs

- ✓ Allows IEEE 802.15.4 radios to carry 128-bit addresses of Internet Protocol version 6 (IPv6).
- ✓ Header compression and address translation techniques allow the IEEE 802.15.4 radios to access the Internet.
- ✓ IPv6 packets compressed and reformatted to fit the IEEE 802.15.4 packet format.
- ✓ Uses include IoT, Smart grid, and M2M applications.

6LowPAN (Contd.)

Addressing in 6LoWPAN

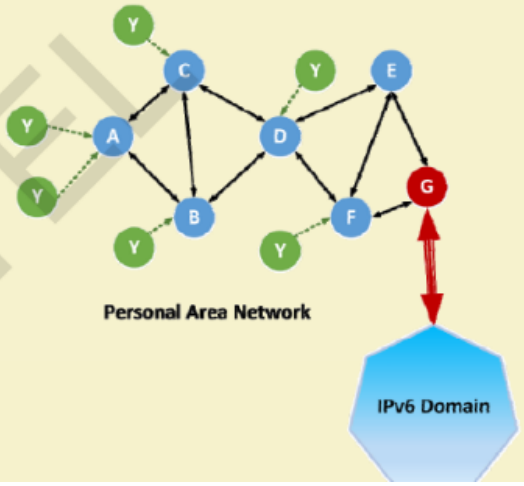


- 64-bit addresses: globally unique
- 16 bit addresses: PAN specific; assigned by PAN coordinator
- IPv6 multicast not supported by 802.15.4
- IPv6 packets carried as link layer broadcast frames

6LoWPAN (Contd.)

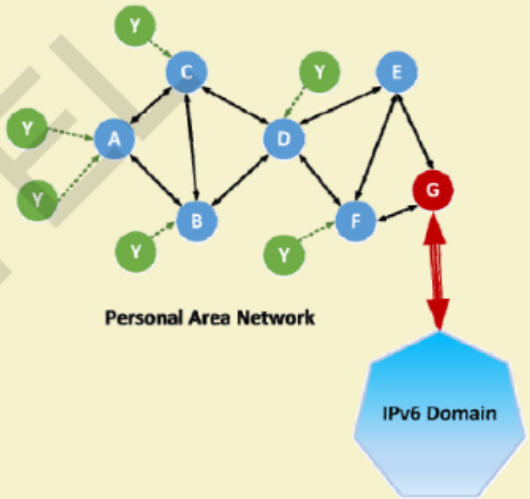
6LoWPAN Routing Considerations

- ✓ Mesh routing within the PAN space.
- ✓ Routing between IPv6 and the PAN domain
- ✓ Routing protocols in use:
  - **LOADng**
  - **RPL**



6LoWPAN Routing Considerations

- ✓ Mesh routing within the PAN space.
- ✓ Routing between IPv6 and the PAN domain
- ✓ Routing protocols in use:
  - **LOADng**
  - **RPL**



LOADng Routing

- ✓ Derived from AODV and extended for use in IoT.
- ✓ Basic operations of LOADng include:
  - Generation of **Route Requests (RREQs)** by a LOADng Router (originator) for discovering a route to a destination,
  - **Forwarding of such RREQs** until they reach the destination LOADng Router,
  - Generation of **Route Replies (RREPs)** upon receipt of an RREQ by the indicated destination, and unicast hop-by-hop forwarding of these RREPs towards the originator.

NPTEL

- If a route is detected to be broken, a **Route Error (RERR)** message is returned to the originator of that data packet to inform the originator about the route breakage.
- **Optimized flooding** is supported, reducing the overhead incurred by RREQ generation and flooding.
- Only the destination is permitted to respond to an RREQ.
- Intermediate LOADng Routers are explicitly prohibited from responding to RREQs, even if they may have active routes to the sought destination.
- RREQ/RREP messages generated by a given LOADng Router share a single unique, monotonically increasing sequence number.

RPL Routing

- ✓ Distance Vector IPv6 **routing protocol for lossy and low power networks.**
- ✓ Maintains routing topology using low rate beaconing.
- ✓ Beaconing rate increases on detecting inconsistencies (e.g. node/link in a route is down).
- ✓ Routing information included in the datagram itself.
- ✓ **Proactive**: Maintaining routing topology.
- ✓ **Reactive**: Resolving routing inconsistencies.

NPTEL

- ✓ RPL separates packet processing and forwarding from the routing optimization objective, which helps in Low power Lossy Networks (LLN).
- ✓ RPL supports message confidentiality and integrity.
- ✓ Supports Data-Path Validation and Loop Detection
- ✓ Routing optimization objectives include
  - minimizing energy
  - minimizing latency
  - satisfying constraints (w.r.t node power, bandwidth, etc.)

- ✓ RPL operations require bidirectional links.
- ✓ In some LLN scenarios, those links may exhibit asymmetric properties.
- ✓ It is required that the reachability of a router be verified before the router can be used as a parent.

6)Explain in detail about Message Queue Telemetry Transport Protocol.

A)

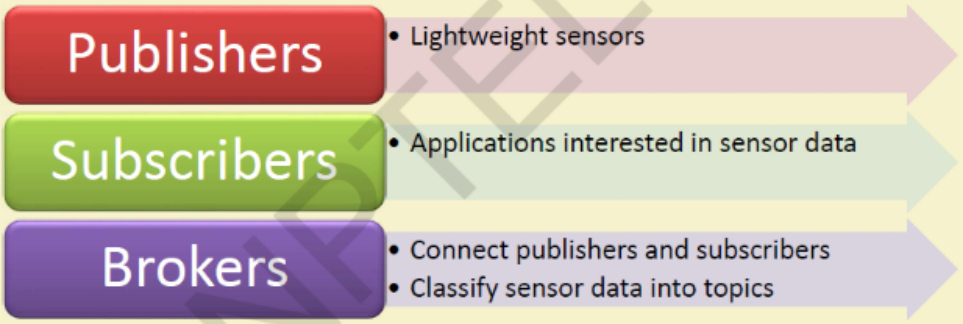
MQTT

Introduction

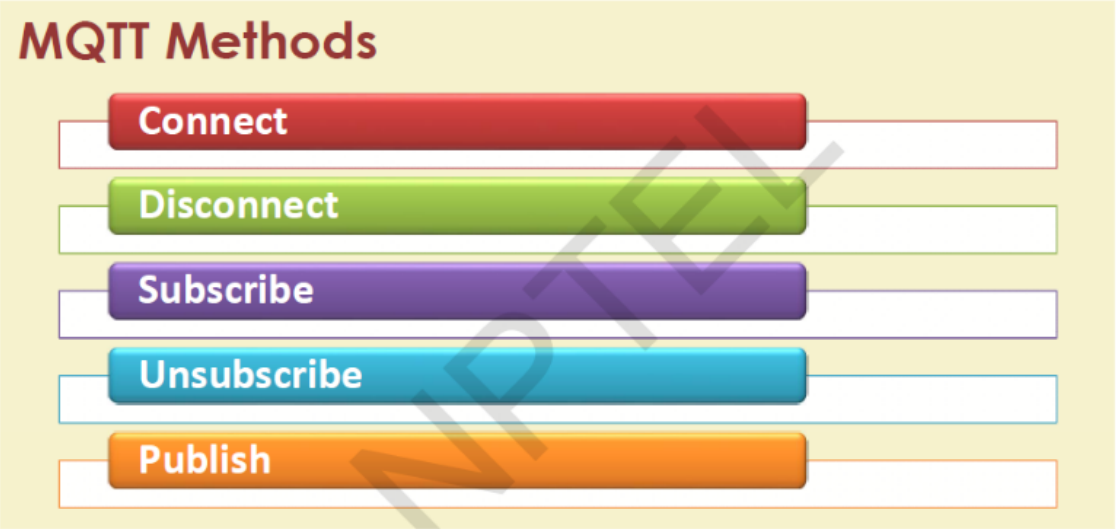
- ✓ **Message Queue Telemetry Transport.**
- ✓ ISO standard (ISO/IEC PRF 20922).
- ✓ It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
- ✓ MQTT was introduced by IBM in 1999 and standardized by OASIS in 2013.
- ✓ Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and networks and communications on the other side.
- ✓ A message broker controls the publish-subscribe messaging pattern.
- ✓ A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker.
- ✓ Designed for:
  - Remote connections
  - Limited bandwidth
  - Small-code footprint

MQTT (Contd.)

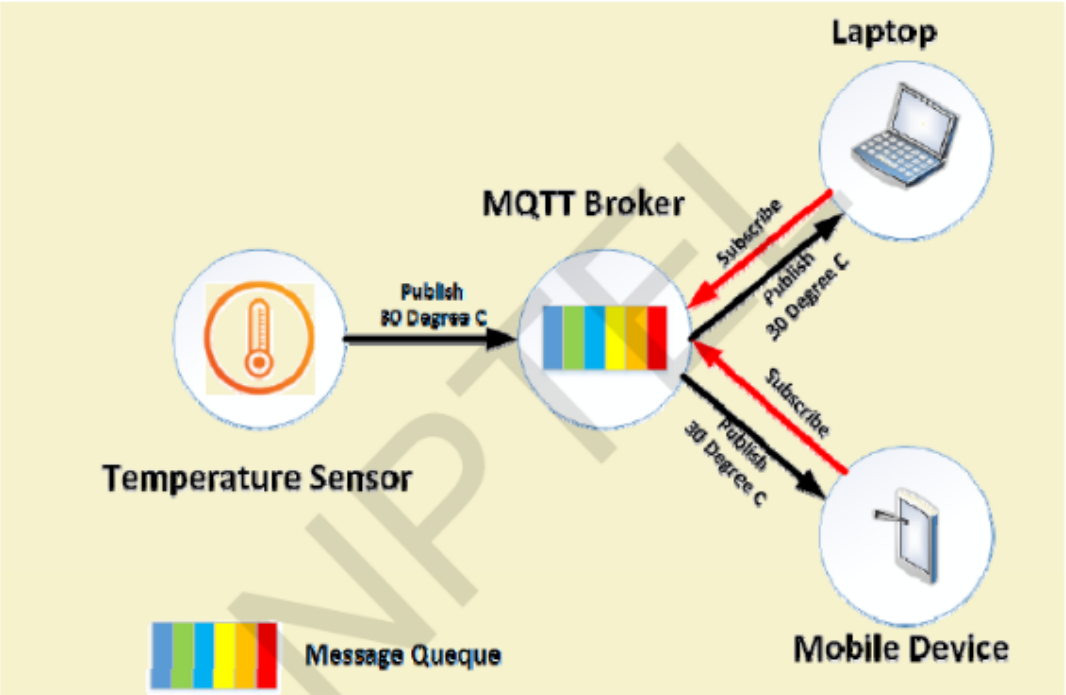
MQTT Components



MQTT (Contd.)



MQTT (Contd.)



Communication

- ✓ The protocol uses a **publish/subscribe** architecture (HTTP uses a request/response paradigm).
- ✓ Publish/subscribe is **event-driven** and enables messages to be pushed to clients.
- ✓ The central **communication point is the MQTT broker**, which is in charge of dispatching all messages between the senders and the rightful receivers.
- ✓ Each client that publishes a message to the broker, includes a **topic** into the message. The **topic is the routing information for the broker.**
- ✓ Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client.
- ✓ Therefore the clients don't have to know each other. They only communicate over the topic.
- ✓ This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

MQTT Topics

- ✓ A topic is a **simple string** that can have more hierarchy leve which are separated by a slash.
- ✓ A sample topic for sending temperature data of the living room could be *house/living-room/temperature*.
- ✓ On one hand the client (e.g. mobile device) can subscribe t the exact topic or on the other hand, it can use a **wildcard**.
- ✓ The subscription to *house/+/temperature* would result in all messages sent to the previously mentioned topic *house/living-room/temperature,* as well as any topic with an arbitrary value i the place of living room, such as *house/kitchen/temperature*.
- ✓ The plus sign is a **single level wild card** and only allows arbitrary values for one hierarchy.
- ✓ If more than one level needs to be subscribed, such as, the enti sub-tree, there is also a **multilevel wildcard** (#).
- ✓ It allows to subscribe to all underlying hierarchy levels.

Applications

- ✓ **Facebook Messenger** uses MQTT for online chat.
- ✓ **Amazon Web Services** use Amazon IoT with MQTT.
- ✓ **Microsoft Azure** IoT Hub uses MQTT as its main protocol for telemetry messages.
- ✓ The **EVRYTHNG IoT platform** uses MQTT as an M2M protocol for millions of connected products.
- ✓ **Adafruit** launched a free MQTT cloud service for IoT experimenters called Adafruit IO.

7)Discuss in detail about XMPP protocol with a neat suitable diagram.

XMPP: Extensible Messaging and Presence Protocol

XMPP (Extensible Messaging and Presence Protocol), formerly known as Jabber, is an open communication protocol for real-time messaging and presence information. It is a decentralized protocol, meaning that there is no single controlling entity, and anyone can run an XMPP server. This makes XMPP a highly flexible and extensible protocol that can be used for a wide variety of applications.

XMPP Architecture

XMPP is based on a client-server architecture. Clients connect to servers using TCP/IP sockets. Once connected, clients can exchange messages, presence information, and other data with the server. The server then routes this data to other clients or servers as needed.

XMPP Messages

XMPP messages are XML-based. This means that messages are represented as XML documents. XML documents are made up of tags, which are enclosed in angle brackets (< >). Tags can have attributes, which are name-value pairs. The value of an attribute can be enclosed in single or double quotes.

For example, the following XML document represents an XMPP message from the user "alice" to the user "bob":

```
XML
<message to="bob@example.com">
  <body>Hello, Bob!</body>
</message>
```

XMPP Presence

XMPP presence information indicates whether a user is available for communication. XMPP presence information can be one of the following:

- Available: The user is available for communication.
- Unavailable: The user is not available for communication.
- Away: The user is away from their computer but may still be available for communication.
- DND (Do Not Disturb): The user does not want to be disturbed.

XMPP Extensions

XMPP is a highly extensible protocol. This means that new features can be added to the protocol by defining new XML tags and elements. There are hundreds of XMPP extensions, which provide a wide variety of features, such as file transfer, audio and video conferencing, and group chat.

XMPP Server-to-Server Communication

XMPP servers can communicate with each other using a variety of mechanisms, including TCP, UDP, and S/MIME. This allows XMPP servers to federate, which means that users on different servers can communicate with each other.

XMPP Use Cases

XMPP is used for a wide variety of applications, including:

- Instant messaging (IM)
- Presence information
- Group chat
- File transfer
- Audio and video conferencing
- Social networking
- Real-time collaboration
- IoT (Internet of Things)
- WebRTC

XMPP Diagram

The following diagram illustrates the XMPP architecture:

```
                  Client                          Client
                    |                               |
                    |                               |
                    V                               V
                  TCP/IP                          TCP/IP
    +----------------------+              +----------------------+
    | XMPP Server          |              | XMPP Server          |
    +----------------------+              +----------------------+
        |                             |
        |                             |
        |                             |
   Federated XMPP Network
```

Conclusion

XMPP is a powerful and versatile protocol that can be used for a wide variety of applications. It is an open, decentralized, and extensible protocol, which makes it a good choice for organizations that need a flexible and scalable communication solution.

8)Compare Zigbee and Zwave protocols and mention its applications

A) READ ZIGBEE FROM 4th QUESTION

Z-Wave

Introduction

- ✓ Zwave (or Z wave or Z-wave) is a protocol for communication among devices used for home automation.
- ✓ It uses RF for signaling and control.
- ✓ Operating frequency is 908.42 MHz in the US & 868.42 MHz in Europe.
- ✓ Mesh network topology is the main mode of operation, and can support 232 nodes in a network.

- ✓ Zwave utilizes GFSK modulation and Manchester channel encoding.
- ✓ A central network controller device sets-up and manages a Zwave network.
- ✓ Each logical Zwave network has 1 Home (Network) ID and multiple node IDs for the devices in it.
- ✓ Nodes with different Home IDs cannot communicate with each other.
- ✓ Network ID length=4 Bytes, Node ID length=1 Byte.

GFSK

- ✓ Gaussian Frequency Shift Keying.
- ✓ Baseband pulses are passed through a Gaussian filter prior to modulation.
- ✓ Filtering operation smoothens the pulses consisting of streams of -1 and 1, and is known as **Pulse shaping**.
- ✓ Pulse shaping limits the modulated spectrum width.

- ✓ Uses source routed network mesh topology using 1 primary controller.
- ✓ Devices communicate with one another when in range.
- ✓ When devices are not in range, messages are routed though different nodes to bypass obstructions created by household appliances or layout.
- ✓ This process of bypassing radio dead-spots is done using a message called **Healing.**
- ✓ As Zwave uses a source routed static network, mobile devices are excluded from the network and only static devices are considered.

9)Differentiate HTTP and MQTT protocols used in IOT.

A)

HTTP and MQTT are both widely used protocols for communication in the Internet of Things (IoT). However, they have different strengths and weaknesses that make them suitable for different use cases.

HTTP

HTTP (Hypertext Transfer Protocol) is a stateless, request-response protocol. This means that each HTTP request is independent of any previous requests, and the server does not maintain any state between requests. This makes HTTP a simple and efficient protocol for retrieving web pages and other resources.

However, HTTP is not well-suited for real-time communication, such as sending sensor data or controlling devices. This is because HTTP is designed for one-way communication, from the client to the server. The server cannot send unsolicited messages to the client, and the client must constantly poll the server for new data.

MQTT

MQTT (Message Queuing Telemetry Transport) is a publish-subscribe protocol. This means that clients can subscribe to topics, and any messages published to those topics will be delivered to all subscribed clients. MQTT is a lightweight protocol that is

designed for resource-constrained devices, and it is very efficient for sending small messages.

MQTT is well-suited for real-time communication because it is a bidirectional protocol. The server can send messages to clients, and clients can send messages to the server. This makes MQTT a good choice for applications that require real-time updates, such as sensor monitoring or device control.

Here is a table summarizing the key differences between HTTP and MQTT:

Feature	HTTP	MQTT
Protocol type	Request-response	Publish-subscribe
Communication direction	One-way (client to server)	Bidirectional
Message size	Large	Small
Complexity	Simple	More complex
Resource requirements	High	Low
Suitability for real-time communication	Poor	Excellent

Use cases for HTTP in IoT:

- Retrieving sensor data at regular intervals
- Configuring devices
- Sending commands to devices

Use cases for MQTT in IoT:

- Real-time sensor monitoring
- Device control
- Real-time data streaming
- IoT applications that require bidirectional communication

In general, HTTP is a good choice for applications that require large, infrequent data transfers, while MQTT is a good choice for applications that require small, frequent data transfers with real-time updates.

10)Discuss in detail about two protocols used in low power applications.

A)

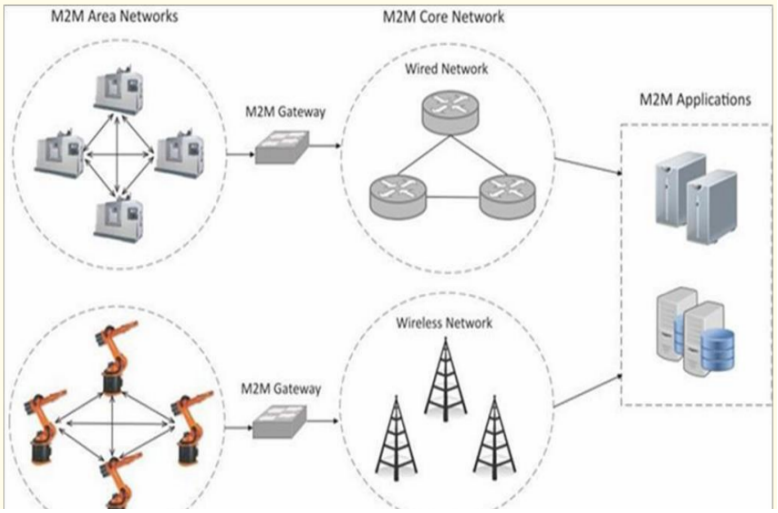
WRITE ABOUT ZIGBEE AND BLUETOOTH LOW ENERGY

UNIT-4

1) Define Machine-to- Machine and explain the working of M2M Gateway unit?

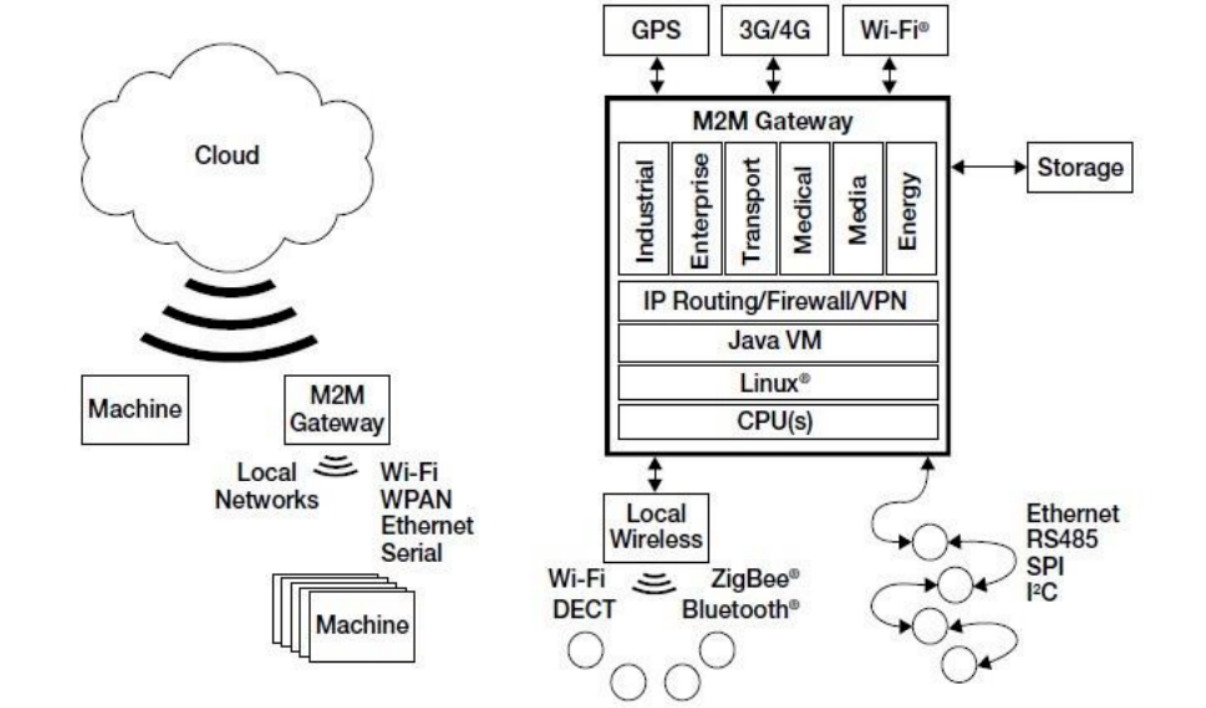
M2M:

- Machine-to-Machine (M2M) refers to networking of machines (or devices) for the purpose of remote monitoring and control and data exchange.
- Term which is often synonymous with IoT is Machine-to-Machine (M2M).
- IoT and M2M are often used interchangeably.



- An M2M area network comprises of machines(or M2M nodes) which have embedded network modules for sensing, actuation and communicating various communication protocols can be used for M2M LAN such as ZigBee, Bluetooth, M-bus, Wireless M-Bus etc., These protocols provide connectivity between M2M nodes within an M2M area network.
- The communication network provides connectivity to remote M2M area networks. The communication network provides connectivity to remote M2M area network. The communication network can use either wired or wireless network(IP based).
- While the M2M are networks use either proprietary or non-IP based communication protocols, the communication network uses IP-based network. Since non-IP based protocols are used within M2M area network, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M are network, M2M gateways are used.

Figure 2: Example of an M2M Gateway



M2M Gateway

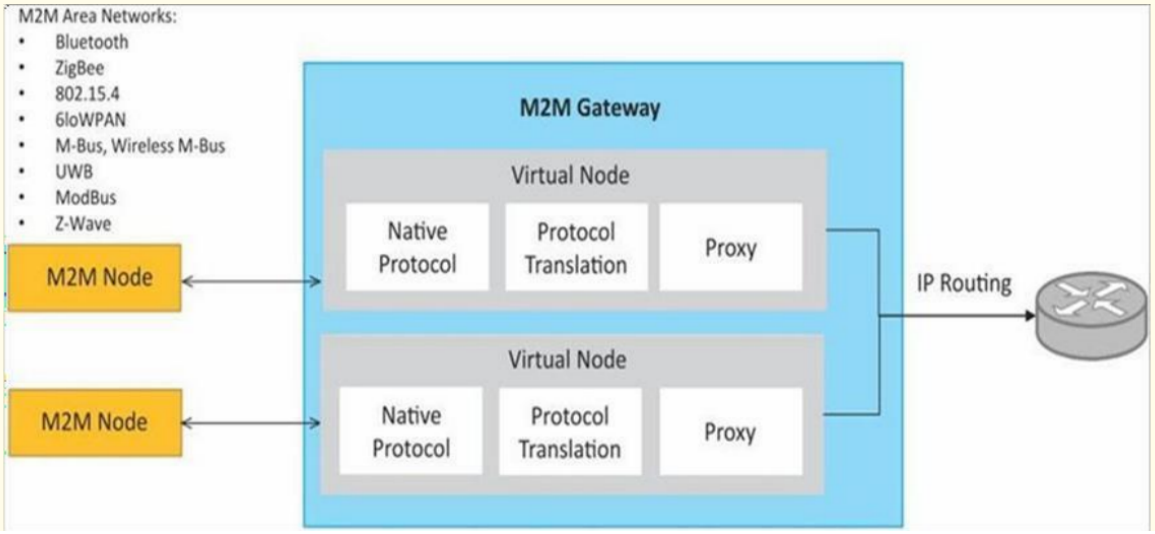
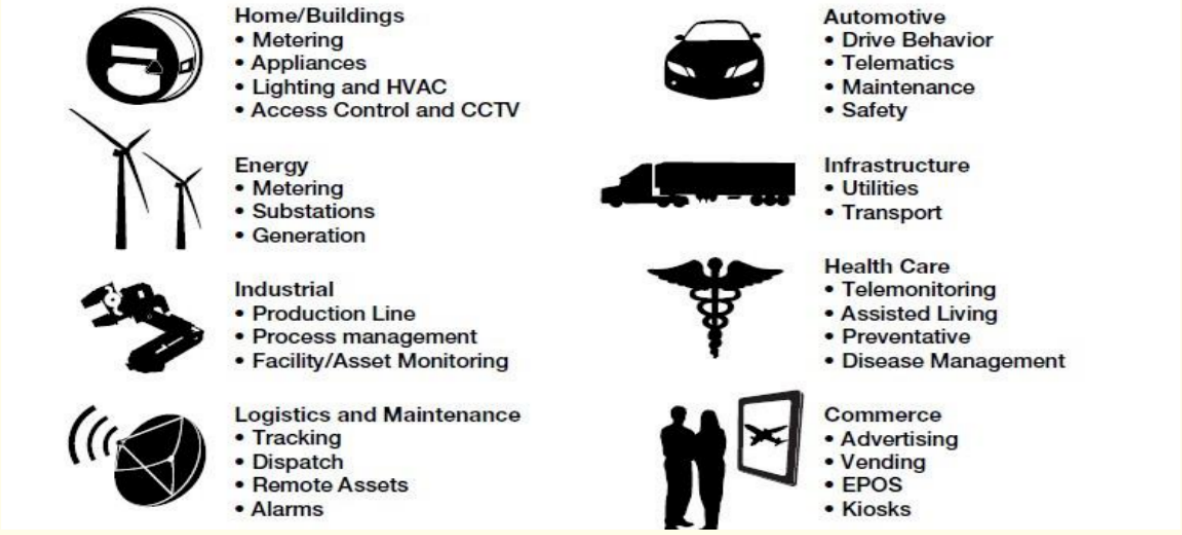


Figure 1: M2M Gateway Applications



M2M gateways offer the following key advantages:

- Can be applied and scaled to multiple settings—home, industrial, office, public, retail
- Avoids duplication of equipment
- A local intelligent node turns raw data into useful information
- A secure node bridges broadband WAN and local wireless connections

2) Differentiate M2M and IOT, explain the communication in IOT and M2M?

Differences between IoT and M2M

- **Communication Protocols:**
  - Commonly uses M2M protocols include ZigBee, Bluetooth, ModBus, M-Bus, WirelessM-Bustec.,
  - In IoT uses HTTP, CoAP, WebSocket, MQTT,XMPP,DDS,AMQPetc.,
- **Machines in M2M Vs Things inIoT:**
  - Machines in M2M will be homogenous whereas Things in IoT will be heterogeneous.
- **Hardware Vs SoftwareEmphasis:**
  - the emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software.
- **Data Collection &Analysis**
  - M2M data is collected in point solutions and often in on-premises storage infrastructure.
  - The data in IoT is collected in the cloud (can be public, private or hybrid cloud).
- **Applications**
  - M2M data is collected in point solutions and can be accessed by on-premises applications such as diagnosis applications, service management applications, and on- premisis enterprise applications.
  - IoT data is collected in the cloud and can be accessed by cloud applications such as analytics applications, enterprise applications, remote diagnosis and management applications,etc.

M2M vs IoT

M2M	IOT
Point-to-point connectivity frequently includes customer-site gear.	A network consists of both remote and local devices that transmit information via IP.
Many devices connect to networks via cellular or cable connections.	The data delivered is through a cloud-based intermediary layer.
Designed for small scale projects	Can usually be scaled for large projects
M2M devices do not always need to be connected to the internet	In most circumstances, devices need continuous internet access.
Because devices must adhere to the same communication protocols, integration choices are limited.	There are endless integration possibilities, but you'll need a platform that can handle all of your communications.

3)Discuss Software Defined Networking and Network Function Virtualization?

A)

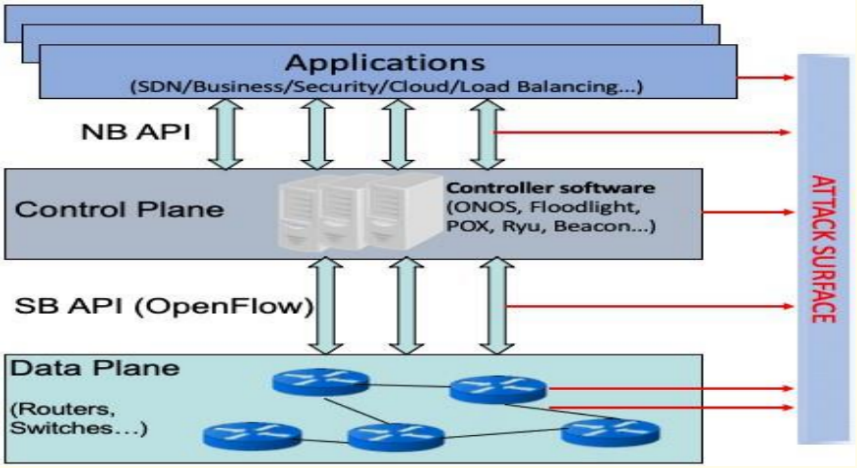
Software Defined Networking(SDN):

- **Software-defined networking** (**SDN**) technology is an approach to network management that enables dynamic, programmatically efficient network configuration to improve network performance and monitoring.
- **Software-Defined Networking (SDN)** is an approach to networking that uses software-based controllers or application programming interfaces (APIs) to communicate with underlying hardware infrastructure and direct traffic on a network.
- This model differs from that of traditional networks, which use dedicated hardware devices (i.e., routers and switches) to control network traffic. SDN can create and control a virtual network – or control a traditional hardware – via software.
- SDN is meant to address the static architecture of traditional networks and may be employed to centralize network intelligence in one network component by disassociating the forwarding process of network packets (data plane) from the routing process (control plane).

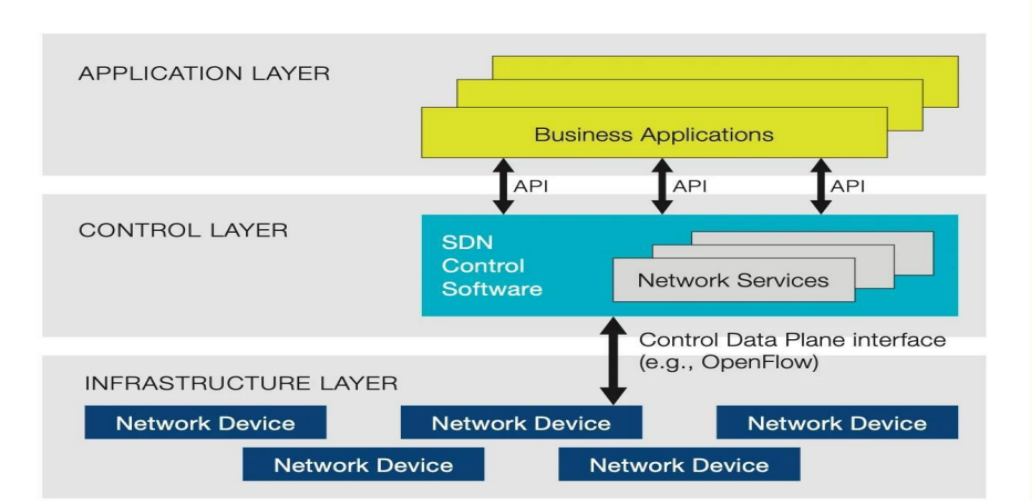
Architecture

- The three layers in an SDN architecture are:
- Application: the applications and services running on the network
- Control: the SDN controller or "brains" of the network
- Infrastructure: switches and routers, and the supporting physical hardware
- To communicate between these layers, SDN uses northbound and southbound application program interfaces (APIs) where the northbound API communicates between the application and the control layers and the southbound API communicates between the infrastructure and control layers.

SDN Architecture



SDN Architecture



OpenFlow and Open Source's Role in SDN Architecture

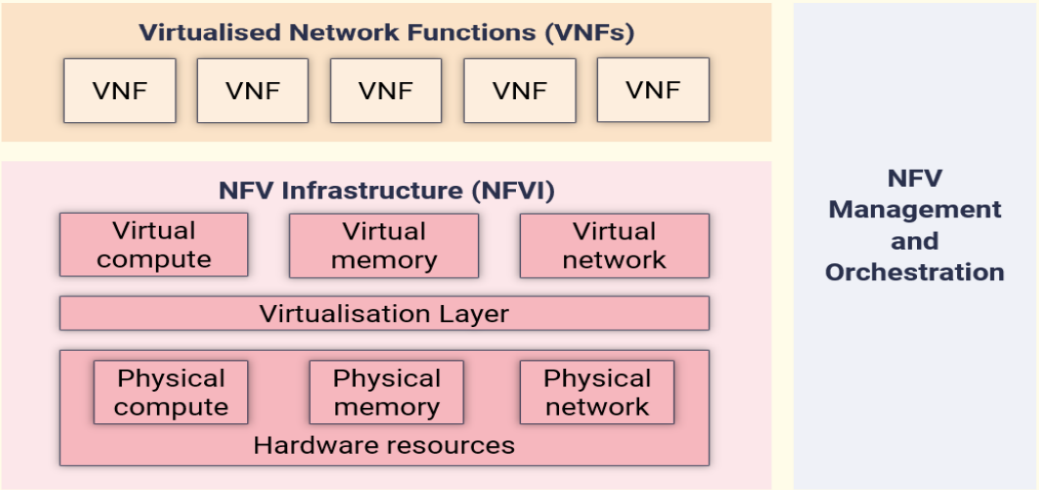
- Traditionally, both the SDN control plane and data plane elements of a networking architecture were packaged in proprietary, integrated code distributed by one or more proprietary vendors.
- The OpenFlow open-source standard, created in 2008, was recognized as the first SDN architecture that defined how the control and data plane elements would be separated and communicate with each other using the OpenFlow protocol.
- The Open Network Foundation (ONF) is the body in charge of managing OpenFlow standards.

SDN Key Elements

- **SDN Applications :**SDN applications are programs that communicate behaviors and needed resources with the SDN controller via application programming interfaces (APIs). In addition, the applications can build an abstracted view of the network by collecting information from the controller for decision-making purposes. These applications could include networking management, analytics, or business applications used to run large data centers.
- **SDN Controller:**The SDN controller is a logical entity that receives instructions or requirements from the SDN application layer and relays them to the networking components. The controller also extracts information about the network from the hardware devices and communicates back to the SDN applications with an abstract view of the network.
- **SDN Networking Devices :**
- The SDN networking devices control the forwarding and data processing capabilities for the network. This includes the forwarding and processing of the data path.
- The SDN architecture APIs are often referred to as northbound and southbound interfaces, defining the communication between the applications, controllers, and networking systems. A northbound interface is defined as the connection between the controller and applications, whereas the southbound interface is the connection between the controller and the physical networking hardware.

NFV(Network Functions Virtualization)

- Network Functions Virtualization (NFV) is the decoupling of network functions from proprietary hardware appliances and running them as software in virtual machines (VMs).
- The different functions — such as firewalls, traffic control, and virtual routing — are called virtual network functions (VNFs).
- NFV uses virtualized networking components to support an infrastructure totally independent of hardware. The standard resources of compute, storage, and network functions can all be virtualized and placed on commercial off-the-shelf (COTS) hardware like x86 servers. Having virtualized resources means that VMs can be given portions of the resources available on the x86 server. That way, multiple VMs can run on a single server and scale to consume the remaining free resources.
- This also means that resources are less often sitting idle and data centers with virtualized infrastructure can be more effectively used. Within the data center and the outside networks, the data plane and control plane can also be virtualized with NFV.



Key Elements of NFV

1.Virtualized Network Function(VNF):
VNF is a software implementation of a network function which is capable of running over the NFV Infrastructure (NFVI).

2.NFV Infrastructure(NFVI):
NFVI includes compute, network and storage resources that are virtualized.

3.NFV Management and Orchestration:
NFV Management and Orchestration focuses on all virtualization-specific management tasks and covers the orchestration and life- cycle management of physical and/or software resources that support the infrastructure virtualization, and the life-cycle management of VNFs.

SDN vs NFV

Category	SDN	NFV
Reason for Being	Separation of control and data, centralization of control and programmability of network	Relocation of network functions from dedicated appliances to generic servers
Target Location	Campus, data center / cloud	Service provider network
Target Devices	Commodity servers and switches	Commodity servers and switches
Initial Applications	Cloud orchestration and networking	Routers, firewalls, gateways, CDN, WAN accelerators, SLA assurance
New Protocols	OpenFlow	None yet
Formalization	Open Networking Forum (ONF)	ETSI NFV Working Group

4)Explain the need the IOT System Management and mention the Characteristics of SNMP?

A)

Need for IoT Systems Management

- IoT systems have complex software, hardware (sensors, actuators), network resources, data collection, analysis services, communication protocols, and user interfaces.
- The need for managing IoT systems are:

1.Automating Configuration:

- System management interfaces provide predicate and easy-to-use management capability to automation system configuration when a system consists of multiple devices or nodes.
- Ensures all devices have the same configuration and variations or errors due to manual configurations are avoided.

2. Monitoring Operational & Statistical Data:

- Operational data:- the system's operating parameters that are collected by the system at runtime.
- Statistical data:- system performance (e.g. CPU and memory usage) data for fault diagnosis or prognosis (forecasting).

Need for IoT Systems Management Contd..

3. Improved Reliability:

- By validating the system configurations before use.

4. System-Wide Configuration:

- IoT systems consist of multiple devices or nodes, which have wide system configurations for the correct functioning.
- Each device is configured separately (either manual or automated).
- Used in system faults or undesirable outcomes.
- Ensures that the configuration changes are either applied to all devices or to none.
- In the failure, the configuration changes are rolled back.

5. Multiple System Configurations:

- Some systems have multiple valid configurations according to different times or in certain conditions.

Need for IoT Systems Management Contd..

6. Retrieving & Reusing Configurations:

- Help in reusing the configurations for other devices of the same type.
- Ensure that when a new device is added, the same configuration is applied.
- The management system can retrieve the current configuration from a device and apply the same to the new devices.

SNMP (SIMPLE NETWORK MANAGEMENT PROTOCOL)

- Simple Network Management Protocol (SNMP) is an Internet Standard protocol for collecting and organizing information about managed devices on IP networks and for modifying that information to change device behavior.
- Devices that typically support SNMP include cable modems, routers, switches, servers, workstations, printers, and more.
- SNMP is widely used in network management for network monitoring. SNMP exposes management data in the form of variables on the managed systems organized in a management information base (MIB), which describes the system status and configuration. These variables can then be remotely queried (and, in some circumstances, manipulated) by managing applications.

SNMP Components

Simple Network Management Protocol (SNMP) –

- SNMP is an application layer protocol that uses UDP port number 161/162.SNMP is used to monitor the network, detect network faults, and sometimes even used to configure remote devices.

SNMP components –

There are 3 components of SNMP:

1. SNMP Manager :

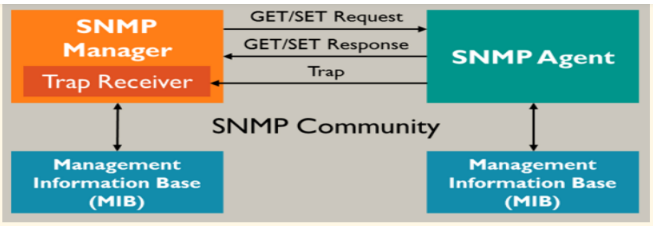
- It is a centralized system used to monitor network. It is also known as Network Management Station (NMS)

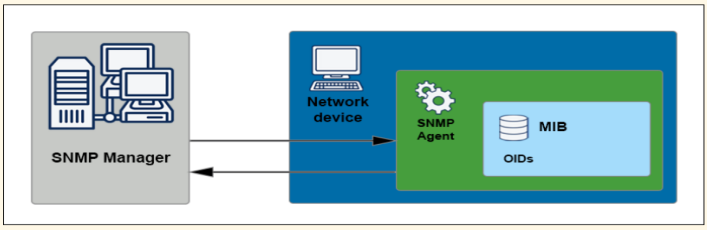
2. SNMP agent :

- It is a software management software module installed on a managed device. Managed devices can be network devices like PC, routers, switches, servers, etc.

3. Management Information Base :

- MIB consists of information on resources that are to be managed. This information is organized hierarchically. It consists of objects instances which are essentially variables.





Limitations of SNMP

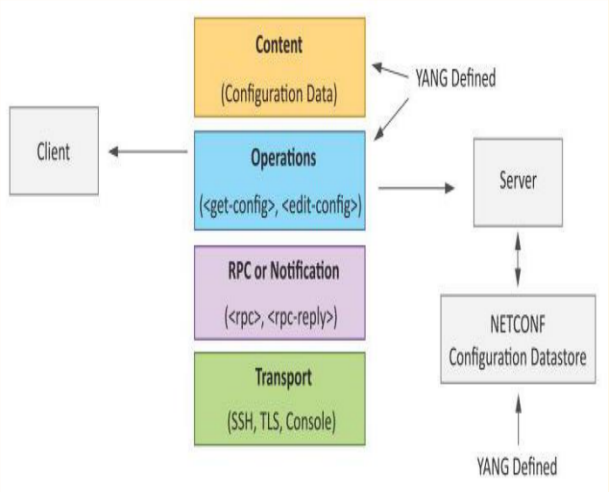
- SNMP is stateless in nature and each SNMP request contains all the information to process the request. The application needs to be intelligent to manage the device.
- SNMP is a connectionless protocol which uses UDP as the transport protocol, making it unreliable as there was no support for acknowledgement of requests.
- MIBs often lack writable objects without which device configuration is not possible using SNMP.
- It is difficult to differentiate between configuration and state data in MIBs.
- Retrieving the current configuration from a device can be difficult with SNMP.
- Earlier versions of SNMP did not have strong security features.

5)Describe IOT Systems Management with NETCONF-YANG?

NETCONF (Network Configuration Protocol)

- The Network Configuration Protocol (NETCONF) is a network management protocol allowing a network management system (NMS) to deliver, modify, and delete configurations of network devices. Standard application programming interfaces (APIs) are available on network devices for the NMS to manage the devices using NETCONF.
- NETCONF uses Extensible Markup Language (XML)-based data encoding for the configuration data and protocol messages like YANG, and uses a simple remote procedure call (RPC) mechanism to implement communication between a client and a server. A client can be a script or an application running on an NMS. A server is typically a network device.

NETCONF (Network Configuration Protocol)



A client provides the following functions:

- Manages network devices using NETCONF.
- Sends RPC requests to a NETCONF server to query or modify one or more parameter values.
- Learns the status of a managed device based on the alarms and events sent by the NETCONF server of the managed device.

A server maintains information about managed devices and responds to the client-initiated requests.

- When receiving a request from a NETCONF client, the NETCONF server parses the request and sends a reply to the client.
- If a fault or another type of event occurs on a managed device, the NETCONF server reports an alarm or event to the client through the notification mechanism. This allows the client to learn the status of the managed device.

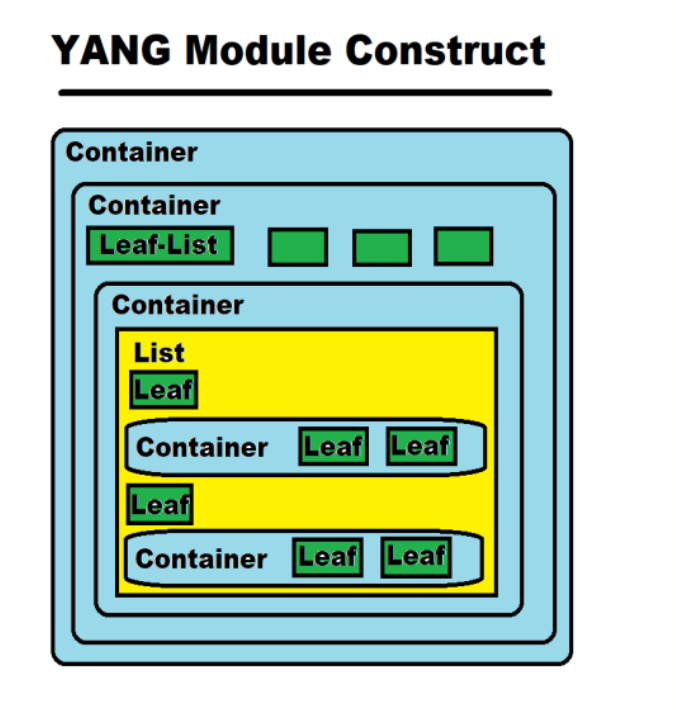
NETCONF

- NETCONF works on SSH transport protocol.
- Transport layer provides end-to-end connectivity and ensure reliable delivery of messages.
- NETCONF uses XML-encoded Remote Procedure Calls (RPCs) for framing request and response messages.
- The RPC layer provides mechanism for encoding of RPC calls and notifications.
- NETCONF provides various operations to retrieve and edit configuration data from network devices.
- The Content Layer consists of configuration and state data which is XML-encoded.
- The schema of the configuration and state data is defined in a data modeling language called YANG.
- NETCONF provides a clear separation of the configuration and state data.
- NETCONF provides security mechanisms such as authentication and authorization to ensure message transmission security.
- The configuration data resides within a NETCONF configuration data store on the server.

YANG

- Yang is a data modeling language for NETCONF modeling.
- YANG modules contain the definitions of the configuration data, state data, RPC calls that can be issued and the format of the notifications.
- YANG modules defines the data exchanged between the NETCONF client and server.
- A module comprises of a number of 'leaf' nodes which are organized into a hierarchical tree structure.
- The 'leaf' nodes are specified using the 'leaf' or 'leaf-list' constructs.
- Leaf nodes are organized using 'container' or 'list' constructs.
- A YANG module can import definitions from other modules.
- Constraints can be defined on the data nodes, e.g. allowed values.
- YANG can model both configuration data and state data using the 'config' statement.
- YANG generates YANG models (also called YANG files) by describing data structures, data integrity constraints, and data operations.

YANG MODULE

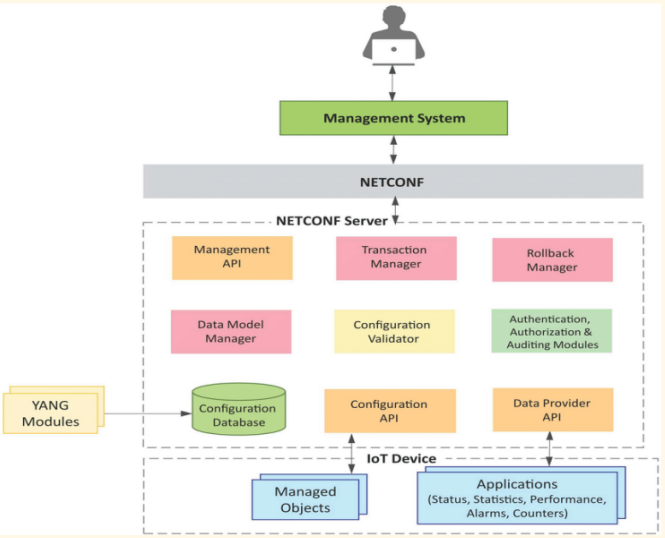


This YANG module is a YANG version of the toaster MIB

- The toaster YANG module begins with the header information followed by identity declarations which define various bread types.
- The leaf nodes (‘toasterManufacturer’, ‘toasterModelNumber’ and toasterStatus’) are defined in the ‘toaster’ container.
- Each leaf node definition has a type and optionally a description and default value.
- The module has two RPC definitions (‘make-toast’ and ‘cancel-toast’).

IoT Systems Management with NETCONF - YANG

- Management System
- Management API
- Transaction Manager
- Rollback Manager
- Data Model Manager
- Configuration Validator
- Configuration Database
- Configuration API
- Data Provider API



6) Explain SDN in detail with a suitable block diagram.

A) REFER 3rd QUESTION

7) Discuss any use case of Network Function Visualization.

A) REFER 3rd QUESTION

8)Explain in detail about session based network management protocol with suitable diagram.

A) REFER 4th QUESTION

9) Explain the pros and cons of Machine to Machine and Internet of Things.

A) REFER 2nd QUESTION

10)Explain Simple Network Management Protocol in detail with a suitable block diagram.

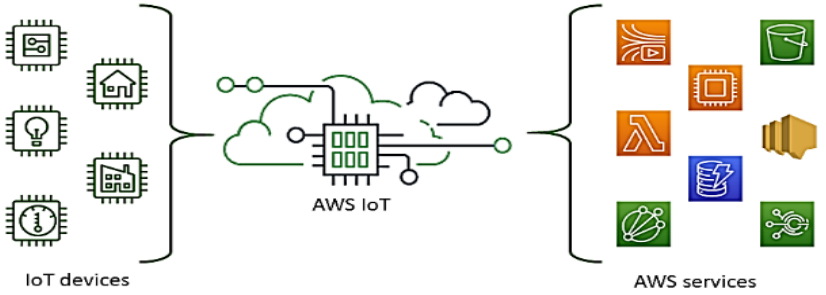
A) REFER 4th QUESTION

UNIT-5

1)Explain different IoT Platforms available in the market?

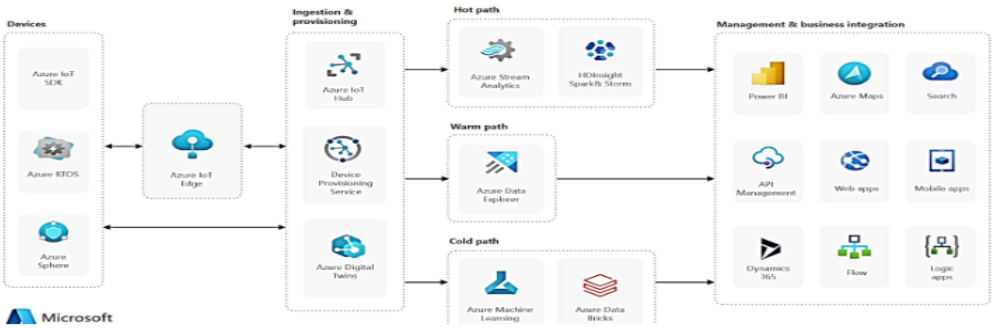
Different IoT Platforms

- **Amazon Web Services (AWS) IoT Core:** AWS IoT Core is a managed cloud service that lets you connect your IoT devices securely, manage them over the air, and collect data from them. It provides a wide range of features for device management, data management, and application enablement.



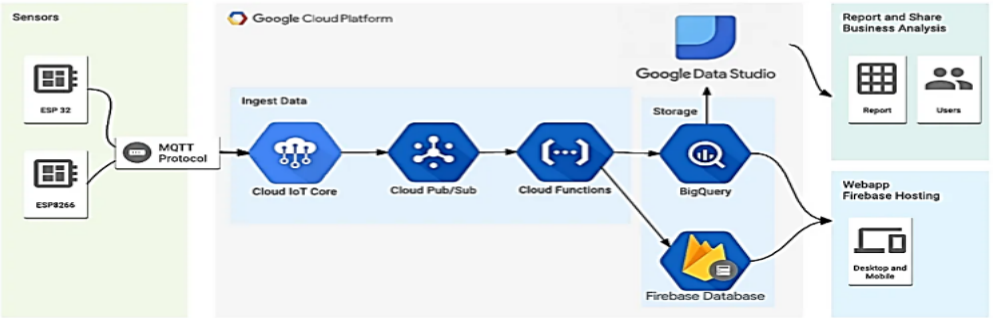
Different IoT Platforms (Contd.)

- **Microsoft Azure IoT:** Microsoft Azure IoT is another popular IoT platform that offers a wide range of features for device management, data management, and application enablement. It is also well-integrated with other Microsoft Azure services, such as Azure Machine Learning and Azure Cognitive Services.



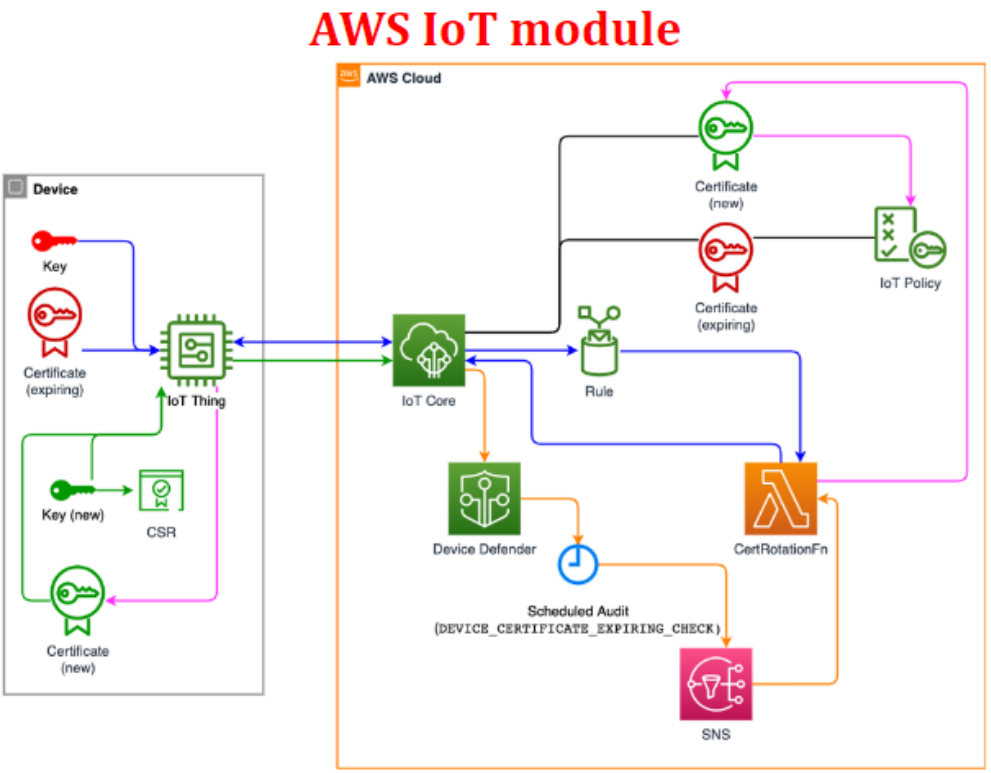
Different IoT Platforms (Contd.)

- **Google Cloud IoT:** Google Cloud IoT is a managed cloud service that lets you connect your IoT devices securely, manage them over the air, and collect data from them. It is a fully managed service, so you don't need to worry about managing the inf



2)Explain in brief about AWS IoT module.

A)



The diagram shows the following components:

- **AWS IoT ExpressLink Module:** This is the hardware module that is certified to work with AWS IoT Core. It includes the necessary software and firmware to connect to the cloud and securely manage your IoT devices.
- **Microcontroller:** This is the main processor on the IoT device. It runs the application code that collects and processes data from the device's sensors and actuators.
- **Sensors:** These devices measure physical or environmental conditions, such as temperature, humidity, or pressure.
- **Actuators:** These devices perform actions, such as controlling lights, motors, or valves.
- **Network Interface:** This connects the IoT device to a network, such as Wi-Fi, Ethernet, or cellular.
- **AWS IoT Core:** This is the managed cloud service that lets you connect

your IoT devices securely, manage them over the air, and collect data from them.

Here is a brief explanation of how the block diagram works:

1. The microcontroller collects data from the sensors.
2. The microcontroller processes the data and sends it to the AWS IoT ExpressLink module.
3. The AWS IoT ExpressLink module encrypts the data and sends it to AWS IoT Core.
4. AWS IoT Core stores the data in a secure database.
5. AWS IoT Core can be used to manage the IoT devices, such as updating the firmware or sending commands to the devices.
6. AWS IoT Core can also be used to analyze the data from the devices and trigger actions based on the analysis.

Key features of AWS IoT modules include:

1. **Pre-certified security:** AWS IoT modules incorporate AWS-mandated security requirements, ensuring that your devices are protected from remote attacks and can establish a secure connection to the AWS cloud.
2. **Simplified device connectivity:** AWS IoT modules provide easy-to-use software libraries and APIs that simplify the process of connecting your devices to AWS IoT Core.
3. **Reduced development time:** AWS IoT modules eliminate the need to develop your own device connectivity firmware, saving you time and effort.
4. **Scalability:** AWS IoT modules are designed to support a wide range of device types and network protocols, making them a scalable solution for growing IoT deployments.
5. **Cost-effectiveness:** AWS IoT modules are available from a variety of hardware partners, offering a range of price points to fit your budget.

3)Compare AWS IoT and Micorsoft Azure IoT platforms.

A) **Device Management**

Both AWS IoT and Microsoft Azure IoT offer comprehensive device management features, including device registration, provisioning, monitoring, and over-the-air (OTA) updates. AWS IoT also offers support for multiple device protocols, while Microsoft Azure IoT has a stronger focus on industrial IoT applications.

Data Management

AWS IoT and Microsoft Azure IoT both offer scalable data storage and processing capabilities. AWS IoT is integrated with Amazon Kinesis, which makes it easy to stream and analyze IoT data. Microsoft Azure IoT is integrated

with Azure Data Lake Storage, which is a petabyte-scale data lake for storing and analyzing IoT data.

Application Enablement

Both AWS IoT and Microsoft Azure IoT provide a wide range of services for developing and deploying IoT applications. AWS IoT is integrated with AWS services such as AWS Lambda, Amazon Machine Learning, and Amazon DynamoDB. Microsoft Azure IoT is integrated with Azure services such as Azure Functions, Azure Machine Learning, and Azure Cosmos DB.

Security

AWS IoT and Microsoft Azure IoT both offer a multi-layered security approach to protect IoT devices and data. AWS IoT uses AWS Identity and Access Management (IAM) to control access to devices and data. Microsoft Azure IoT uses Azure Active Directory (AAD) to control access to devices and data.

Pricing

AWS IoT and Microsoft Azure IoT both offer a pay-as-you-go pricing model, so you only pay for the resources that you use. AWS IoT's pricing is based on the number of devices, messages, and rules that you use. Microsoft Azure IoT's pricing is based on the number of devices, messages, and ingress and egress data.

Ease of Use

Both AWS IoT and Microsoft Azure IoT offer an easy-to-use web console and APIs for managing and developing IoT applications. AWS IoT's web console is slightly more intuitive, while Microsoft Azure IoT's APIs are more comprehensive.

Overall

AWS IoT and Microsoft Azure IoT are both powerful and versatile IoT platforms that can be used to support a wide range of IoT applications. The best platform for you will depend on your specific needs and requirements.

4)Enlist challenges faced by IoT platform in the implementation of real time applications.

1. **Latency:** Latency, or the delay between when data is generated and when it is processed and acted upon, is a critical factor for real-time applications. High latency can make it difficult to respond to events in a timely manner, which can have negative consequences for the user experience and the effectiveness of the application.
2. **Scalability:** Real-time IoT applications can generate a large amount of data, which can quickly overwhelm traditional data processing and storage systems. The IoT platform must be able to scale to handle this volume of data without compromising performance or reliability.
3. **Security:** Real-time IoT applications are often used to collect and process sensitive data, which makes them a target for cyberattacks. The IoT

platform must be secure and compliant with all relevant data privacy regulations.

4. **Reliability:** Real-time IoT applications must be highly reliable, as downtime can have serious consequences. The IoT platform must be able to withstand outages and recover quickly from failures.
5. **Interoperability:** Real-time IoT applications often need to connect with a variety of devices and systems, which can make it difficult to ensure interoperability. The IoT platform must support a wide range of protocols and standards to enable seamless communication between devices.

5)Explain the steps involved in the process of interfacing any sensor to ThingSpeak Cloud.

1. Set up a ThingSpeak account:

- Go to https://thingspeak.com/ and create an account if you don't have one.
- Sign in to your account and create a new channel by clicking on the New Channel button.
- Enter a name for your channel and a description of what it will be used for.
- Select the fields that you want to use to store your sensor data. Click on Save Channel.

2. Gather your materials:

- You will need the following materials to interface your sensor to ThingSpeak Cloud:
- A microcontroller board, such as an Arduino or a Raspberry Pi
- A sensor, such as a temperature sensor or a humidity sensor
- Jumper wires
- A USB cable to connect your microcontroller board to your computer

3. Write the code:

- You will need to write some code to read data from your sensor and send it to ThingSpeak Cloud. The specific code you write will depend on the type of sensor you are using and the microcontroller board you are using.
- For example, here is an example of code that will read data from a temperature sensor and send it to ThingSpeak Cloud using an Arduino Uno.

4. Upload the code to your microcontroller board:

- Once you have written the code, you need to upload it to your microcontroller board.
- The specific process for uploading code will depend on the type of microcontroller board you are using.
- For example, here are the instructions for uploading code to an Arduino Uno:

- Connect the Arduino Uno to your computer using a USB cable.
- Open the Arduino IDE.
- Select the Arduino Uno from the Tools > Board menu.
- Select the COM port that your Arduino Uno is connected to from the Tools > Port menu.
- Click on the Upload button.

5. Test your code:

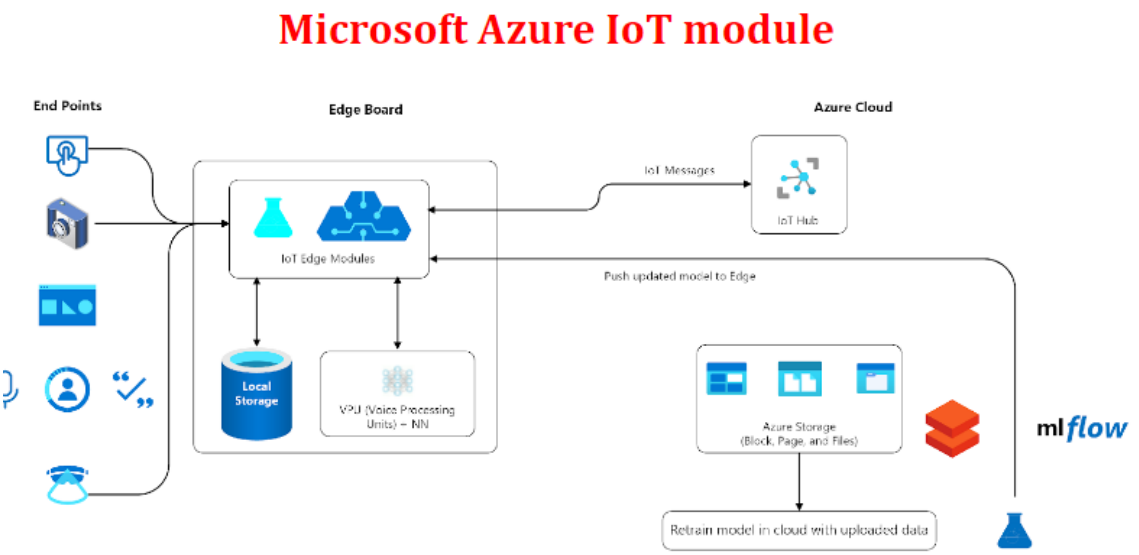
- Once you have uploaded the code to your microcontroller board, you can test it to make sure that it is working correctly.
- To do this, you can use the ThingSpeak Channels app or the ThingSpeak API to view your sensor data.

6. Deploy your application:

- Once you have tested your code and verified that it is working correctly, you can deploy your application.
- This may involve connecting your microcontroller board to the internet or using a battery-powered device to run the application.

6)Explain in brief about Microsoft Azure IoT module.

A)



The diagram shows the following components:

- **Azure IoT Edge Module:** This is the hardware module that is certified to work with Azure IoT Edge. It includes the necessary software and firmware to connect to the cloud and securely manage your IoT devices.
- **Microcontroller:** This is the main processor on the IoT device. It runs the

application code that collects and processes data from the device's sensors and actuators.

- **Sensors:** These devices measure physical or environmental conditions, such as temperature, humidity, or pressure.
- **Actuators:** These devices perform actions, such as controlling lights, motors, or valves.
- **Network Interface:** This connects the IoT device to a network, such as Wi-Fi, Ethernet, or cellular.
- **Azure IoT Edge runtime:** This is the software that runs on the IoT device and manages the Azure IoT Edge modules. It is responsible for installing, configuring, and running the modules.
- **Azure IoT Hub:** This is the managed cloud service that lets you connect your IoT devices securely, manage them over the air, and collect data from them. It can also be used to analyze the data from the devices and trigger actions based on the analysis.

Here is a brief explanation of how the block diagram works:

1. The microcontroller collects data from the sensors.
2. The microcontroller processes the data and sends it to the Azure IoT Edge module.
3. The Azure IoT Edge module encrypts the data and sends it to Azure IoT Hub.
4. Azure IoT Hub stores the data in a secure database.
5. Azure IoT Hub can be used to manage the IoT devices, such as updating the firmware or sending commands to the devices.
6. Azure IoT Hub can also be used to analyze the data from the devices and trigger actions based on the analysis.

Microsoft Azure IoT modules are the smallest unit of computation managed by Azure IoT Edge. They are containers that can host Azure services, third-party services, or your own code. Azure IoT modules can perform a wide range of tasks, such as:

- **Pre-processing device data:** Azure IoT modules can be used to pre-process device data before it is sent to the cloud. This can help to reduce the amount of data that is transmitted and make it easier to analyze the data in the cloud.
- **Running analytics on device data:** Azure IoT modules can be used to run analytics on device data in real time. This can be used to identify trends, anomalies, and patterns in the data.

- **Taking actions based on device data:** Azure IoT modules can be used to take actions based on device data. This can include sending alerts, controlling devices, or triggering workflows.
- **Extending the capabilities of Azure IoT Edge:** Azure IoT modules can be used to extend the capabilities of Azure IoT Edge. This can be done by adding new features or functionality to the platform.

Here are some of the benefits of using Azure IoT modules:

- **Modular design:** Azure IoT modules are designed to be modular, so they can be easily reused and combined to create complex applications.
- **Scalability:** Azure IoT modules can be scaled to support a wide range of workloads.
- **Security:** Azure IoT modules are sandboxed, so they cannot access the underlying operating system. This helps to protect the device from security threats.

7)Discuss the steps involved in the process of retrieving the data from ThingSpeak Cloud.

A)
Steps involved in the process of retrieving the data from ThingSpeak Cloud.

Retrieving data from ThingSpeak Cloud involves several steps, including:

1. Create a ThingSpeak Channel:

- Sign up for a ThingSpeak account or sign in to your existing account.
- Click the "Channels" tab and then click the "Create New Channel" button.
- Enter a name for your channel and select the appropriate data types (fields) for your data.
- Click the "Create Channel" button.

2. Generate an API Key:

- Click the "My Channels" tab and locate your newly created channel.

- Click the "API Keys" button and then click the "Generate New API Key" button.
- Enter a label for your API key and click the "Generate API Key" button.
- Copy the generated API key as you will need it later.

3. Retrieve Data Using API or Channel Feed:

- ThingSpeak provides two main methods for retrieving data:

  a) Using the ThingSpeak API:

- Use the ThingSpeak API to make HTTP requests to ThingSpeak servers and retrieve data.
- You can use various programming languages and tools to make API calls and process the JSON or CSV data returned by ThingSpeak.

- b) Using Channel Feeds:

- Access the channel feed URL directly in your web browser or use a feed reader to subscribe to the channel's data updates.
- Channel feeds provide a real-time stream of data in JSON or CSV format.

4. Parse and Process Data:

- Once you have retrieved the data, you will need to parse it into a usable format depending on your programming language or tool.
- For JSON data, use a JSON parser to extract the relevant data fields.
- For CSV data, use a CSV parser to extract the data into tables or arrays.

5. Analyze and Utilize Data:

- Analyze the retrieved data to gain insights, identify trends, or make informed decisions.
- Visualize the data using graphs, charts, or other data visualization techniques.
- Utilize the data for various purposes, such as monitoring sensor data, controlling devices, or performing data analysis.

Remember that the specific steps and tools you use will depend on your programming language, toolset, and the specific data you want to retrieve.

8) Compare Google Cloud IoT and Micorsoft Azure IoT platforms.

Device Management
Both Google Cloud IoT and Microsoft Azure IoT offer comprehensive device management features, including device registration, provisioning, monitoring, and over-the-air (OTA) updates. Google Cloud IoT also offers support for multiple device protocols, while Microsoft Azure IoT has a stronger focus on industrial IoT applications.
Data Management
Google Cloud IoT and Microsoft Azure IoT both offer scalable data storage and processing capabilities. Google Cloud IoT is integrated with Google Pub/Sub and Google Cloud Dataflow, which makes it easy to stream and analyze IoT data. Microsoft Azure IoT is integrated with Azure Event Hubs and Azure Data Lake Storage, which is a petabyte-scale data lake for storing and analyzing IoT data.
Application Enablement
Both Google Cloud IoT and Microsoft Azure IoT provide a wide range of services for developing and deploying IoT applications. Google Cloud IoT is integrated with Google Cloud services such as Cloud Functions, Cloud Machine Learning, and Cloud Storage. Microsoft Azure IoT is integrated with Azure services such as Azure Functions, Azure Machine Learning, and Azure Cosmos DB.

Security
Google Cloud IoT and Microsoft Azure IoT both offer a multi-layered security approach to protect IoT devices and data. Google Cloud IoT uses Google Cloud IAM to control access to devices and data. Microsoft Azure IoT uses Azure Active Directory (AAD) to control access to devices and data.
Pricing
Google Cloud IoT and Microsoft Azure IoT both offer a pay-as-you-go pricing model, so you only pay for the resources that you use. Google Cloud IoT's pricing is based on the number of devices, messages, and rules that you use. Microsoft Azure IoT's pricing is based on the number of devices, messages, and ingress and egress data.
Ease of Use
Both Google Cloud IoT and Microsoft Azure IoT offer an easy-to-use web console and APIs for managing and developing IoT applications. Google Cloud IoT's web console is slightly more intuitive, while Microsoft Azure IoT's APIs are more comprehensive.

Overall
Google Cloud IoT and Microsoft Azure IoT are both powerful and versatile IoT platforms that can be used to support a wide range of IoT applications. The best platform for you will depend on your specific needs and requirements.

9)Explain the pros and cons of ThingSpeak Cloud for IOT.

Pros of ThingSpeak Cloud for IoT:

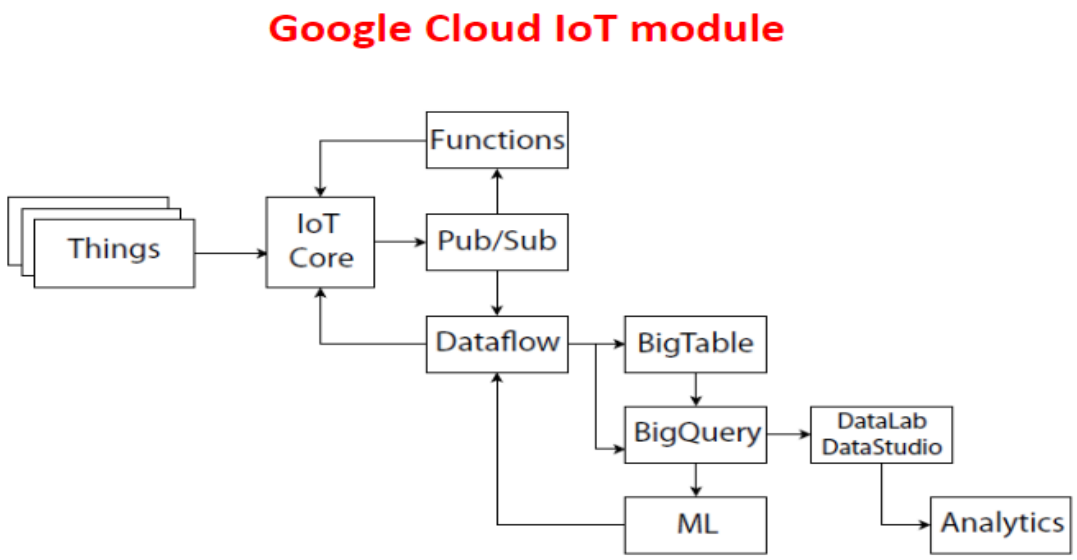
- **Ease of use:** ThingSpeak Cloud is a user-friendly platform that is easy to get started with. It has a drag-and-drop interface and a wide range of pre-built channels and widgets that make it easy to connect your devices and visualize your data.
- **Free tier:** ThingSpeak Cloud has a generous free tier that allows you to connect up to four devices and store up to 300 data points per channel per day. This makes it a great option for prototyping and testing your IoT applications.
- **Scalability:** ThingSpeak Cloud can be scaled to support a wide range of IoT applications, from small hobby projects to large enterprise deployments.
- **Security:** ThingSpeak Cloud uses industry-standard security protocols to protect your data.
- Wide range of features: ThingSpeak Cloud offers a wide range of features for connecting, managing, and analyzing your IoT data. These features include:
  - **Device management:** ThingSpeak Cloud allows you to manage your IoT devices, including registering, provisioning, monitoring, and updating them.
  - **Data storage and processing:** ThingSpeak Cloud stores your IoT data and provides a variety of tools for processing and analyzing the data.
  - **Data visualization:** ThingSpeak Cloud provides a variety of tools for visualizing your IoT data, including charts, graphs, and maps.
  - **Notifications:** ThingSpeak Cloud allows you to set up notifications to alert you to events, such as when a sensor value exceeds a certain threshold.
  - **Integrations:** ThingSpeak Cloud integrates with a variety of third-party services, such as IFTTT and Zapier.

Cons of ThingSpeak Cloud for IoT:

- **Limited bandwidth:** The free tier of ThingSpeak Cloud has limited bandwidth. This means that you may need to upgrade to a paid plan if you are generating a lot of data.
- **Limited storage:** The free tier of ThingSpeak Cloud has limited storage. This means that you may need to upgrade to a paid plan if you need to store more data.
- **Limited customizability:** ThingSpeak Cloud is a relatively closed platform. This means that there is limited opportunity to customize the platform to your specific needs.

10) Explain in brief about Google Cloud IoT module.

A)



The diagram shows the following components:

- **Google Cloud IoT Core:** This is the managed cloud service that lets you connect your IoT devices securely, manage them over the air, and collect data from them.
- **Google Cloud IoT ExpressLink Module:** This is the hardware module that is certified to work with Google Cloud IoT Core. It includes the necessary software and firmware to connect to the cloud and securely manage your IoT devices.

- **Microcontroller:** This is the main processor on the IoT device. It runs the application code that collects and processes data from the device's sensors and actuators.
- **Sensors:** These devices measure physical or environmental conditions, such as temperature, humidity, or pressure.
- **Actuators:** These devices perform actions, such as controlling lights, motors, or valves.
- **Network Interface:** This connects the IoT device to a network, such as Wi-Fi, Ethernet, or cellular.

Here is a brief explanation of how the block diagram works:

1. The microcontroller collects data from the sensors.
2. The microcontroller processes the data and sends it to the Google Cloud IoT ExpressLink Module.
3. The Google Cloud IoT ExpressLink Module encrypts the data and sends it to Google Cloud IoT Core.
4. Google Cloud IoT Core stores the data in a secure database.
5. Google Cloud IoT Core can be used to manage the IoT devices, such as updating the firmware or sending commands to the devices.
6. Google Cloud IoT Core can also be used to analyze the data from the devices and trigger actions based on the analysis.

A Google Cloud IoT module, also known as a Google Cloud IoT ExpressLink module, is a pre-certified hardware module that integrates seamlessly with Google Cloud IoT Core. It provides a secure and reliable way to connect your IoT devices to the cloud, enabling you to collect, store, and analyze device data. These modules are designed to work with a variety of microcontrollers and network interfaces, making them a versatile solution for a wide range of IoT applications.

Key features of Google Cloud IoT modules include:

1. **Pre-certified security:** Google Cloud IoT modules incorporate Google-mandated security requirements, ensuring that your devices are protected from remote attacks and can establish a secure connection to the Google cloud.
2. **Simplified device connectivity:** Google Cloud IoT modules provide easy-to-use software libraries and APIs that simplify the process of connecting your devices to Google Cloud IoT Core.
3. **Reduced development time:** Google Cloud IoT modules eliminate the need to develop your own device connectivity firmware, saving you time and effort.

4. **Scalability:** Google Cloud IoT modules are designed to support a wide range of device types and network protocols, making them a scalable solution for growing IoT deployments.
5. **Cost-effectiveness:** Google Cloud IoT modules are available from a variety of hardware partners, offering a range of price points to fit your budget.