Unit II

SENSORS, ACTUATOR & PROTOCOLS :IoT Sensors: Introduction, sensor types, features, basic components, working principles of different sensors , Actuator and it's types, Microcontroller for IoT **IoT protocols:** MQTT, COAP, SOAP, REST, HTTP, XMPP, WEBSOCKET

SENSORS

- Sensor is a device which can be able to detect environment changes.
- The basic functionality of Sensors is to sense and convert physical parameters into electrical signals.
- It is able to measure physical quantities like temperature, moisture, pressure etc and convert into electric signals.

  Properties of sensor:
- Range: a range(maximum or minimum) which is used to measure by sensors.
- Sensitivity: detect changes in output signal
- Resolution: changes in sensor detection

Sensor Classification/ Types:

- **Active Sensor :** An active sensor is a sensing device that requires an external source of power to operate; It is one which transmits a signal into the environment and then measures the response that comes back.
- One example is an ultrasonic system:
- **passive sensors** measure reflected sunlight emitted from the sun. When the sun shines, passive sensors measure this energy.

Difference Between Active and Passive Sensors

Active Sensors	Passive Sensors
Active sensor produce electric voltage or current in response to the environmental change/stimulation.	Passive sensor generates a change in quantity of electrical energy, e.g. resistance, inductance, or capacitance as a result of environmental changes.
Active sensors provide their own energy source for illumination.	Passive sensors generate energy when the natural energy (like sunlight) is available.
Active sensors are able to obtain measurements anytime (day & night)	Passive sensors can obtain measurements only in the Day time

DIFFERENCE TYPES OF SENSORS :

- **Temperature sensor:**

1) A temperature sensor is a device used to measure temperature. This can be air temperature, liquid temperature or the temperature of solid matter.
2) A temperature sensor is a device, typically, a thermocouple or resistance temperature detector, that provides temperature measurement in a readable form through an electrical signal.
3) Different Temperature Sensor:
4) Thermostats: A thermostat is a contact type temperature sensor consisting of a bi-metallic strip made up of two dissimilar metals such as aluminium, copper, nickel, or tungsten.

- **Thermistors:** Thermistors or thermally sensitive resistors are the ones that change their physical appearance when subjected to change in the temperature.

- **Thermocouples:** A thermocouple usually consists of two junctions of dissimilar metals, such as copper and constantan that are welded or crimped together. One of these junctions, known as the Cold junction, is kept at a specific temperature while the other one is the measuring junction, known as the Hot junction.

- **Negative Temperature Coefficient (NTC) Thermistor:** A thermistor is basically a sensitive temperature sensor that reacts precisely to even the minute temperature changes. It provides a huge resistance at very low temperatures.

Moisture / Humidity Sensor:

- It can detect humidity or moisture of air.
- DHT Humidity

Pressure Sensor:

- It measure the pressure or force applied on any object.
- BMP sensor

Motion Sensor

- It is used to detect movement of any object.
- PIR sensor

Level Sensor:

- It translate the level of liquid into electric signal.

Image Sensor:

- It is used to capture the image and convert it into digital format for data processing.

Proximity sensors:

- Proximity sensors detects the presence or absence of a nearby object without getting in contact with them. Proximity sensors are largely used for detection of presence of people in specific area and thereby controlling on/off of lights and AC to save energy. It is also been used in elevator doors to detect entry/exit of person before closing the doors
- It can be implemented using different techniques like ultrasonic, infrared, laser

Gas Sensor:
- It is used to detect any type of gases.

Smoke Sensor:
- It is used to detect the conditions of smoke.

IR (Infrared) Sensor:
- It is use to detect infrared radiation emitted by object.

Acceleration Sensor:
- It is use to detect the rate of change of object's velocity.

Gyroscopic Sensor:
- It is use to measure the rotation of an object.

Optical Sensor:
- It is use to detect light reflection of any object

Chemical Sensor:
- It is use to detect the chemical compounds in a object.

Water Quality Sensor:
It is use to detect the quality of water like ph level, oxygen levels, chemical presence

ACTUATOR

- Actuator is a physical device which provide the support to other devices.
- It is convert the energy into motion.
- Actuator is a mechanical device that take energy that is created by air, electricity, liquid etc and convert it into some kind of motions.
- It is mainly used in manufacturing or industrial applications.
- For example:- motor, pump, valves

Types of Actuators

1. Electrical Actuators
2. Hydraulic Actuator
3. Pneumatic Actuator
4. Mechanical Actuator

1. **Electrical Actuator:** It is a electromechanical device that converts electrical energy into mechanical energy.

Advantages:
- It has many applications in industries or manufacturing sector, where automatic task should be performed.
- It produce less noise
- It provide easy control

Disadvantages:
- It is expensive.
- It needs more safety conditions.
- It depends on lot of environmental conditions.

2. **Hydraulic Actuator:**
- Hydraulic actuator uses hydraulic power to perform mechanical operations.
- They actuated by fluid, gases , cylinder etc
- They are mostly used in industrial , manufacturing, construction applications

Advantage:

- Reduce Man power
- High speed or more power
- Use in Industrial loads

Disadvantage:

- It is more expensive.
- Leakage can reduce it's efficiency.

3. **Pneumatic Actuator:**

It uses energy formed by compressed air or vacuum at higher pressure.
Pressure converts into linear or rotatary motion.

Advantage:

- Less expensive
- Mostly used in Civil Engineering work (Manufacturing or Construction).
- Low maintenance

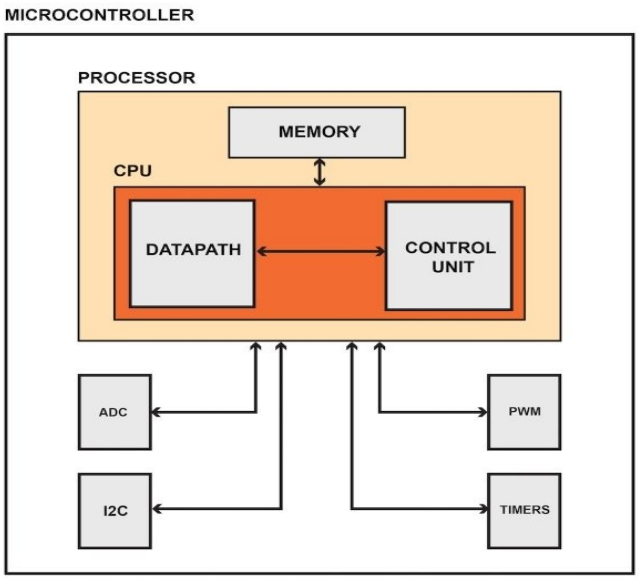
Disadvantage:

- Loss pressure reduce efficiency
- It needs compressed air.

MICRO-CONTROLLER

- A microcontroller is an integrated circuit (IC) device used for controlling other portions of an electronic system, usually via a microprocessor unit (MPU), memory, and some peripherals.
- These devices are optimized for embedded applications that require both processing functionality and agile, responsive interaction with digital, analog, or electromechanical components.
- "Microcontroller" is a well-chosen name because it emphasizes defining characteristics of this product category. The prefix "micro" implies smallness and the term "controller" here implies an enhanced ability to perform control functions.

A **microcontroller** is a small and low-cost microcomputer, which is designed to perform the specific tasks of embedded systems like displaying microwave's information, receiving remote signals, etc.



Microprocessor vs Microcontroller

Microcontroller	Microprocessor
Microcontrollers are used to execute a single task within an application.	Microprocessors are used for big applications.
Its designing and hardware cost is low.	Its designing and hardware cost is high.
Easy to replace.	Not so easy to replace.
It is built with CMOS technology, which requires less power to operate.	Its power consumption is high because it has to control the entire system.
It consists of CPU, RAM, ROM, I/O ports.	It doesn't consist of RAM, ROM, I/O ports. It uses its pins to interface to peripheral devices.

The Elements of a Microcontroller

A microcontroller consists of a central processing unit (CPU), nonvolatile memory, volatile memory, peripherals, and support circuitry.

1. The Central Processing Unit

- The CPU performs arithmetic operations, manages data flow, and generates control signals in accordance with the sequence of instructions created by the programmer. The extremely complex circuitry required for CPU functionality is not visible to the designer. Integrated development environments and high-level languages such as C is used to writing code for microcontrollers.

2. Memory

- Nonvolatile memory is used to store the microcontroller's program—i.e., the (often very long) list of machine-language instructions that tell the CPU exactly what to do. The word "Flash" (which refers to a specific form of nonvolatile data storage) instead of "nonvolatile memory."
- Volatile memory (i.e., RAM) is used for temporary data storage. This data is lost when the microcontroller loses power. Internal registers also provide temporary data storage, but we don't think of these as a separate functional block because they are integrated into the CPU.

3. Peripherals

- We use the word "peripheral" to describe the hardware modules that help a microcontroller to interact with the external system. The following bullet points identify the various categories of peripherals and provide examples.
- **Clock generation:** internal oscillator, crystal-drive circuitry, phase-locked loop
- **Timing:** general-purpose timer, real-time clock, external-event counter, pulse-with modulation
- **Analog signal processing:** operational amplifier, analog comparator
- **Input/output:** general-purpose digital input and output circuitry, parallel memory interface
- **Serial communication:** UART, SPI, I2C, USB
- **Data converters:** analog-to-digital converter, digital-to-analog converter, reference-voltage generator

4. Support Circuitry

- Microcontrollers incorporate a variety of functional blocks that cannot be classified as peripherals because their primary purpose is not to control, monitor, or communicate with external components. they support the internal operation of the device, simplify implementation, and improve the development process.
- **Debug circuitry** allows the designer to carefully monitor the microcontroller as it is executing instructions. This is an important, and sometimes indispensable, method of tracking down bugs and optimizing firmware performance.
- **Interrupts** are an extremely valuable aspect of microcontroller functionality. Interrupts are generated by external or internal hardware-based events, and they cause the processor to immediately respond to these events by executing a specific group of instructions.

Types of Microcontrollers

- Microcontrollers are divided into various categories based on memory, architecture, bits and instruction sets. Following is the list of their types −

(A) Bit: Based on bit configuration, the microcontroller is further divided into three categories.

- **8-bit microcontroller** − This type of microcontroller is used to execute arithmetic and logical operations like addition, subtraction, multiplication division, etc. For example, Intel 8031 and 8051 are 8 bits microcontroller.
- **16-bit microcontroller** − This type of microcontroller is used to perform arithmetic and logical operations where higher accuracy and performance is required. For example, Intel 8096 is a 16-bit microcontroller.
- **32-bit microcontroller** − This type of microcontroller is generally used in automatically controlled appliances like automatic operational machines, medical appliances, etc.

(B) Memory: Based on the memory configuration, the microcontroller is further divided into two categories.

- **External memory microcontroller** − This type of microcontroller is designed in such a way that they do not have a program memory on the chip. Hence, it is named as external memory microcontroller. For example: Intel 8031 microcontroller.
- **Embedded memory microcontroller** − This type of microcontroller is designed in such a way that the microcontroller has all programs and data memory, counters and timers, interrupts, I/O ports are embedded on the chip. For example: Intel 8051 microcontroller.

(C) Instruction Set: Based on the instruction set configuration, the microcontroller is further divided into two categories.

- **CISC** − CISC stands for complex instruction set computer. It allows the user to insert a single instruction as an alternative to many simple instructions.
- **RISC** − RISC stands for Reduced Instruction Set Computers. It reduces the operational time by shortening the clock cycle per instruction.

(D) Memory Architecture:

- Harvard Memory Architecture Microcontroller
- Princeton Memory Architecture Microcontroller
- The majority of MCUs use one of the following architecture:
- ARM
- MIPS
- X86

- **8051 microcontroller:** The most universally employed set of microcontrollers come from the 8051 family. 8051 Microcontrollers persist to be an ideal choice for a huge group of hobbyists and experts.

- **PIC Microcontroller:** Peripheral Interface Controller (PIC) provided by Micro-chip Technology to categorize its solitary chip microcontrollers. These appliances have been extremely successful in 8 bit micro-controllers.

- **AVR Microcontroller:** AVR also known as Advanced Virtual RISC, is a customized Harvard architecture 8 bit RISC solitary chip micro controller. It was invented in the year 1966 by Atmel. Harvard architecture signifies that program & data are amassed in different spaces and are used simultaneously.

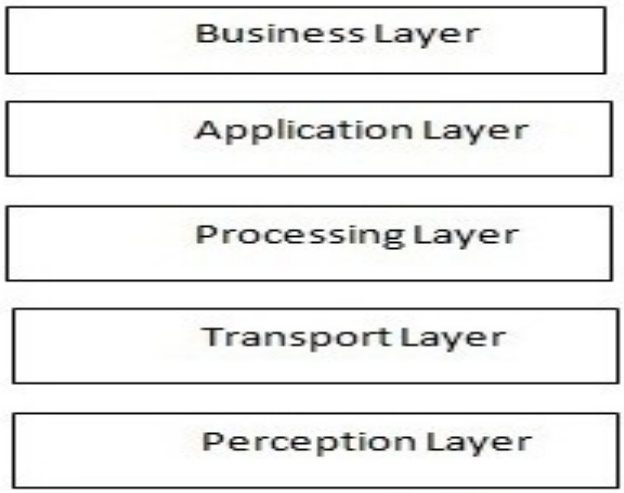
- **AMR Microcontroller:** AMR is the name of a company that designs micro processors architecture. It is also engaged in licensing them to the producers who fabricate genuine chips. In actuality AMR is a 32 bit genuine RISC architecture. It was initially developed in the year 1980 by Acorn Computers Ltd.

Applications of Microcontrollers:

- Light sensing & controlling devices
- Temperature sensing and controlling devices
- Fire detection & safety devices
- Industrial instrumentation devices
- Process control devices

OVERVIEW OF IOT PROTOCOLS

- **IOT PROTOCOLS:** IoT protocols are the standards that developers use to connect devices into one network and enable them to exchange data. Apart from sending and receiving data packages, these protocols ensure the network's security and device compatibility.



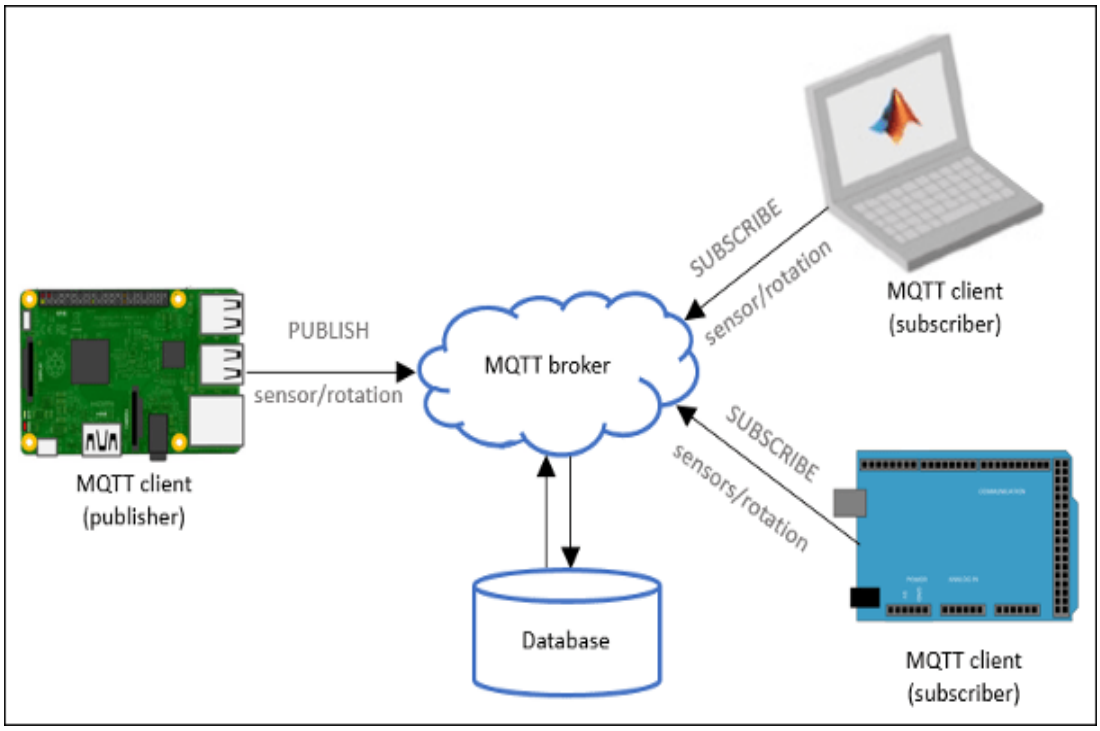
MQTT [Message Queue Telemetry Transport]:

- MQTT (Message Queue Telemetry Transport) is an open protocol for asynchronous data exchange between physically scattered devices that works at the application layer.
- MQTT is a lightweight **publish/subscribe** messaging protocol designed for M2M (machine to machine) telemetry in low bandwidth environments.
- It was designed by Andy Stanford-Clark (IBM) and Arlen Nipper in 1999 for connecting Oil Pipeline telemetry systems over satellite.
  [word "telemetry" in english dictionary, means measuring and sending values]

- The protocol is intended for use on wireless and low-bandwidth networks. A mobile application that uses MQTT sends and receives messages by calling an MQTT library. The messages are exchanged through an MQTT messaging server.
- This protocol has been specially designed for sensor networks and wireless sensor networks. MQTT allows devices to send or publish data information on a given topic to a server.
- The protocol uses port 1883 by default (or port 8883 if an SSL connection is established). It's not suitable for transferring voice or video data.
- MQTT is used when there's a need to minimize the data packet size and when there are restrictions on channel bandwidth. This is why MQTT is often used as the basis for industrial IoT solutions and M2M systems.

MQTT ARCHITECTURE

- There are three parts of MQTT architecture –
- **MQTT Client –** Any third party client who wishes to subscribe to data published by API, is considered as an MQTT Client.
- **MQTT Server –** The API acts as an MQTT server. The MQTT server will be responsible for publishing the data to the clients.
- **MQTT Broker –** All messages passed from the client to the server should be sent via the broker.
  The MQTT Client and the MQTT Server need to connect to the Broker in order to publish or subscribe messages.



MQTT Basic Concepts:

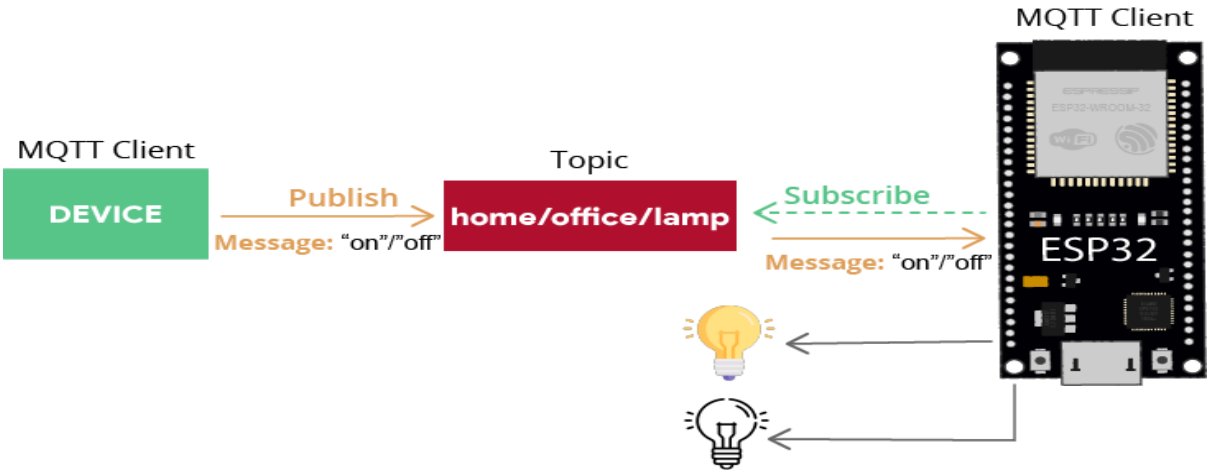
- In MQTT there are a few basic
- Publish/Subscribe
- Messages
- Topics
- Broker

MQTT – Publish/Subscribe:

- The first concept is the *publish and subscribe* system. In a publish and subscribe system, a device can publish a message on a topic, or it can be subscribed to a particular topic to receive messages.

For example **Device 1** publishes on a topic.
Device 2 is subscribed to the same topic that **device 1** is publishing in.
So, **device 2** receives the message.

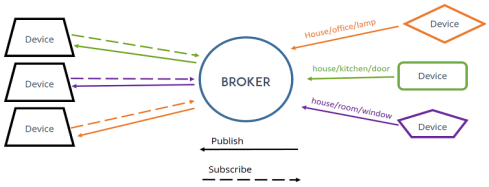
- **MQTT – Messages**
- Messages are the information that you want to exchange between your devices. It can be a message like a command or data like sensor readings, for example.
- **MQTT – Topics**
- A topic is a string(UTF-8). Using this string, Broker filters messages for all connected clients. One topic may consist of one or more topic levels. Forward slash(topic level separator) is used for separating each topic level. Topics are the way to how you specify where you want to publish the message.
- Topics are represented with strings separated by a forward slash. Each forward slash indicates a topic level. Here's an example of how you would create a topic for a lamp in your home office:



- A device publishes "on" and "off" messages on the **home/office/lamp** topic.
- You have a device that controls a lamp (it can be an ESP32, ESP8266, or any other board or device). The ESP32 that controls your lamp, is subscribed to that same topic: **home/office/lamp**.
- So, when a new message is published on that topic, the ESP32 receives the "on" or "off" messages and turns the lamp on or off.

MQTT – Broker:

- The MQTT broker is responsible for receiving all messages, filtering the messages, deciding who is interested in them, and then publishing the message to all subscribed clients.
- Main responsibilities of a Broker are-
- Receive all messages
- Filter messages
- Decide which are interested clients
- Publish messages to all the subscribed clients



- There are several brokers you can use. In home automation projects, we use the Mosquitto Broker installed on a Raspberry Pi. You can also install the Mosquitto broker on your PC (which is not as convenient as using a Raspberry Pi board, because you have to keep your computer running all the time to keep the MQTT connection between your devices).

Features supported by MQTT:

1. Authentication:

- MQTT provides authentication of every user who intends to publish or subscribe to particular data. The user id and password is stored in the API database, into a separate collection called *'mqtt*
- While connecting to MQTT broker, we provide the username name and password, and the MQTT Broker will validate the credentials based on the values present in the database.

2. Access Control:

- MQTT determines which user is allowed to access which topics. This information is stored in MongoDB under the table *'mqtt_acl'*
- By default, all users are allowed to access all topics by specifying '#' as the allowed topic to publish and subscribe for all users.

3. QoS:

- The Quality of Service (QoS) level is the Quality transfer of messages which ensures the delivery of messages between sending body & receiving body. There are 3 QoS levels in MQTT:
- ***At most once***(0) –The message is delivered at most once, or it is not delivered at all.
- ***At least once***(1) – The message is always delivered at least once.
- ***Exactly once***(2) – The message is always delivered exactly once.

4. Last Will Message:

- MQTT uses the Last Will & Testament(LWT) mechanism to notify ungraceful disconnection of a client to other clients. In this mechanism, when a client is connecting to a broker, each client specifies its last will message which is a normal MQTT message with QoS, topic, retained flag & payload. This message is stored by the Broker until it it detects that the client has disconnected ungracefully.

5. Retain Message:

- MQTT also has a feature of Message Retention. It is done by setting TRUE to retain the flag. It then retained the last message & QoS for the topic. When a client subscribes to a topic, the broker matches the topic with a retained message. Clients will receive messages immediately if the topic and the retained message are matched. Brokers only store one retained message for each topic.

6. Duplicate Message:

- If a publisher doesn't receive the acknowledgement of the published packet, it will resend the packet with DUP flag set to true. A duplicate message contains the same Message ID as the original message.

7. Session:

- In general, when a client connects with a broker for the first time, the client needs to create subscriptions for all topics for which they are willing to receive data/messages from the broker. Suppose a session is not maintained, or there is no persistent session, or the client lost a connection with the broker, then users have to resubscribe to all the topics after reconnecting to the broker.
- For the clients with limited resources, it would be very tedious to subscribe to all topics again. So brokers use a persistent session mechanism, in which it saves all information relevant to the client. 'clientId' provided by client is used as 'session identifier', when the client establishes a connection with the broker.

Features not-supported by MQTT:

1. Not RESTful:

MQTT does not allow a client to expose RESTful API endpoints. The only way to communicate is through the publish /subscribe mechanism.

2. Obtaining Subscription List:

The MQTT Broker doesn't have the Client IDs and the subscribed topics by the clients. Hence, the API needs to publish all data to all possible combinations of topics. This would lead to a problem of network congestion in case of large data.

MQTT Versions

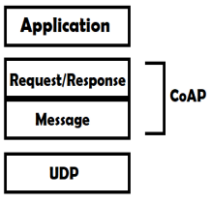
- There are two different variants of MQTT and several versions.
- MQTT v3.1.0 –
- MQTT v3.1.1 – In Common Use
- MQTT v5 – Currently Limited use

CoAP [Constrained Application Protocol]

- CoAP stands for **Constrained Application Protocol**, and it is defined in RFC 7252. CoAP is a simple protocol with **low overhead specifically designed for constrained devices** (such as microcontrollers) and constrained networks. This protocol is used in M2M data exchange and is **very similar to HTTP.**

The main features of CoAP protocols are:

- Web protocol used in M2M with constrained requirements
-Asynchronous message exchange
-Low overhead and very simple to parse
-URI and content-type support
-Proxy and caching capabilities



When to use

- Some of the specific cases in which CoAP are useful are:
- **Your hardware cannot run HTTP or TLS:** If this is the case then running CoAP and DTLS can practically do the same as HTTP. If one is an expert on HTTP APIs, then the migration will be simple. You receive GET for reading and POST, PUT and DELETE for mutations and the security runs on DTLS.

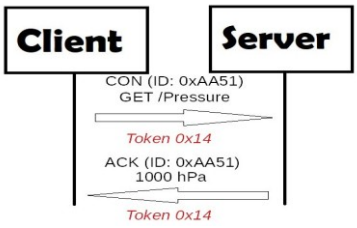
- **Your hardware uses battery:** If this is ones problem then running CoAP will improve the battery performance when compared with HTTP over TCP/IP. UDP saves some bandwidth and makes the protocol more efficient.
- **A subscription is necessary:** If one cannot run MQTT and HTTP polling is impossible then CoAP is a solution

CoAP Protocol Messages:

- CoAP Protocol supports four different message types:
- **Confirmable:**
  -A CoAP confirmable message is a reliable message. During exchanging messages between two endpoints, these messages can be reliable.
  - In the CoAP protocol, a reliable message is obtained using a Confirmable message (CON). Using this kind of message, the client can be sure that the message will arrive at the server.
  -A *CoAP Confirmable message* is sent again and again until the other party sends an acknowledge message (ACK). The ACK message contains the same ID of the confirmable message (CON).
- **Non-confirmable (NON) messages** that don't require an Acknowledge by the server. These mesages are unreliable messages means do not contain critical information that must be delivered to the server. To this category belongs messages that contain values read from sensors. Even if these messages are unreliable, they have a unique ID.
- **Acknowledgment**
- **Reset**

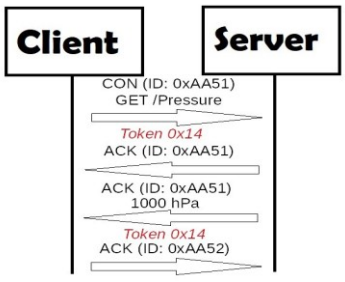
CoAP Request/Response Model

- This is Second layer in the abstraction layer. Here request is sent using a Confirmable (CON) or Non-Confirmable (NON) message. There are several scenarios depending on if the server can answer immediately to the client request or the answer if not available:
- If the server can answer immediately to the client request then if the request is carried using a Confirmable message (CON) then the server sends back to the client an Acknowledge message containing the response or the error code:



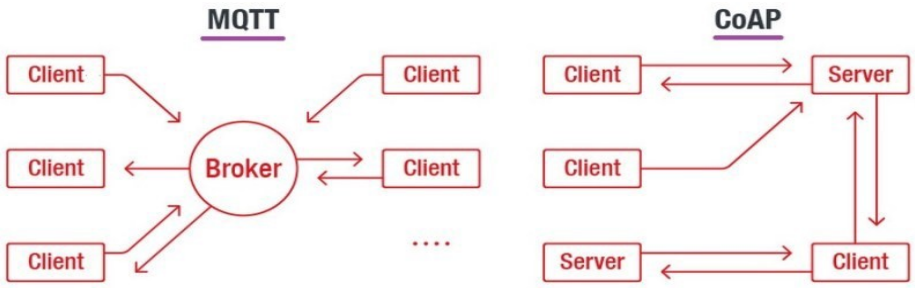
Here Token is different from the Message ID and it is used to match the request and the response.

- If the server can't answer to the request, then server sends an Acknowledge with an empty response. As soon as the response is available then the server sends a new Confirmable message to the client containing the response. At this point the client sends back an Acknowledge message:



If the request coming from the client is carried using a NON-confirmable message then the server answer using a NON-confirmable message.

COAP vs MQTT



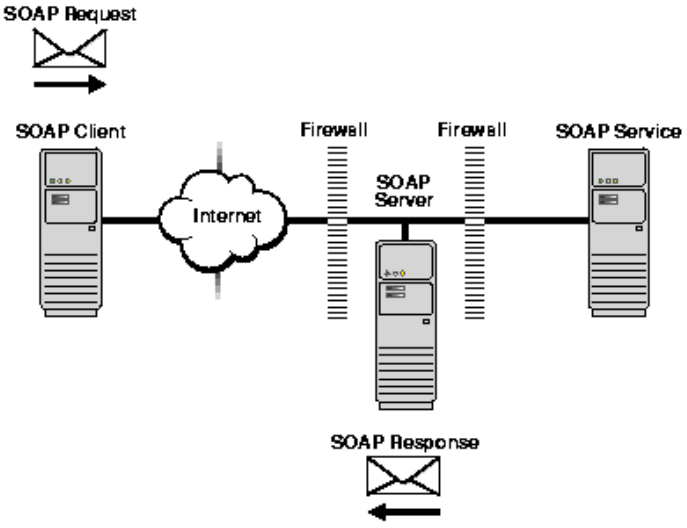
Difference between MQTT & CoAP

MQTT	CoAP
This model has publishers and subscribers as main participants	Uses requests and responses
Central broker handles message dispatching, following the optimal publisher to client path.	Message dispatching happens on a unicasting basis (one-to-one). The process is same as HTTP.
Event-oriented operations	Viable for state transfer
Establishing a continual and long-lasting TCP connection with the broker is essential for the client.	Involved parties use UDP packets (async) for message passing and communication.
No message labeling but have to use diverse messages for different purposes.	It defines messages properly and makes its discovery easy.

SOAP [Simple Object Access Protocol]

- SOAP (Simple Object Access Protocol) is one more widely used protocol for exchanging data in the form of structured XML messages within a distributed computing system.
- SOAP is based on the XML language and extends some protocols working at the application layer, such as HTTP, FTP, and SMTP.
- In other way we can say, SOAP is an XML-based protocol for accessing web services over HTTP.
- In-built error detection is a key part of this protocol.

SOAP ARCHITECTURE



- In general, a SOAP service remote procedure call (RPC) request/response sequence includes the following steps:
- A ***SOAP client*** formulates a request for a service. This involves creating a conforming XML document, either explicitly or using Oracle SOAP client API.
- A SOAP client sends the XML document to a ***SOAP server.***
- This **SOAP request** is posted using HTTP or HTTPS to a SOAP Request Handler running as a servlet on a Web server. SOAP message, an XML document, that represents a SOAP request for a service that provides an address from an address book.
- The Web server receives the SOAP message, an XML document, using the SOAP Request Handler Servlet. The server then dispatches the message as a service invocation to an appropriate server-side application providing the requested service.
- A response from the service is returned to the SOAP Request Handler Servlet and then to the caller using the standard SOAP XML payload format

SOAP message structure

- a simple SOAP Message has the following elements –
- The Envelope element
- The header element and
- The body element
- The Fault element (Optional)

Envelope is the core and essential element of every message, which begins and concludes messages with its tags, enveloping it, hence the name.
Header (optional) determines the specifics, extra requirements for the message, e.g. authentication.
Body includes the request or response.
Fault (optional) shows all data about any errors that could emerge throughout the API request and response.

Advantages and disadvantages of SOAP

- **Advantages of SOAP**
  1. Simplicity
  2. Portability
  3. Firewall friendliness
  4. Use of open standards
  5. Universal acceptance.
  6. Protocol independence

7. Language independence
8. Platform and operating system independence

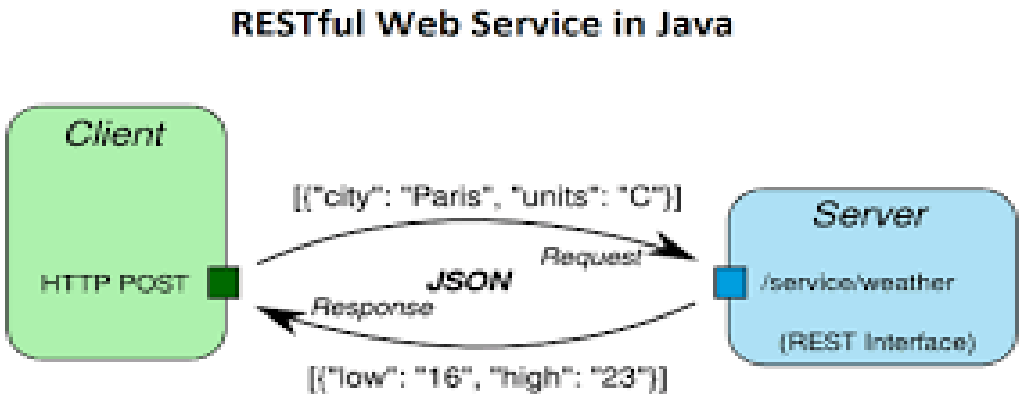
- **Disadvantages of SOAP are:**
  1. Too much reliance on HTTP
  2. Statelessness
  3. Serialization by value and not by reference.

REST [REPRESENTATIONAL STATE]

- REST stands for Representational State Transfer.
- The concept of REST was introduced in 2000 by Roy Fielding, a noted computer scientist who has influenced the development of many WWW standards.
- It's an architectural style for developing web services.
- A lot of people believe that there is a REST protocol in IoT. However, REST itself is a concept, not an IoT protocol.
- REST is the basis for the most widely used form of API and is designed to be used over any protocol. However, it typically uses HTTP or COAP to work with components in a particular IoT device.
- Web services are defined on the principles of REST and can be defined as a RESTful web service.
- RESTful web services can use normal POST, DELETE, PUT, and HTTP verbs of GET for working the components listed above.
- Web services which follow the REST architectural style are known as RESTful web services. It allows requesting systems to access and manipulate web resources by using a uniform and predefined set of rules.

What's the Difference Between REST and RESTful?

- A REST web service is a Representational State Transfer and an architectural pattern for creating web services.
- On the other hand, the RESTful service is one that implements that pattern.



METHODS OF REST API

- **A. HTTP Get :** Get request are wont to retrieve resource information only and to not modify it that's it don't change the state of the resource which is why it's called to be safe method also get also GET API should be idempotent ,which means that it should give the identical result for multiple identical request.

- **B. HTTP Post :** It is used for creating the new sub ordinate resources. The request body consists if the data send to the server. If invoked two identical post request then it will generate two different response.

- **C. HTTP Put :** It is primarily used to update existing resources .If the resource does not exists then the API decides whether to create a new resource or not .If a new resource has been created by PUT API then the server should be informed about

- **D. HTTP Delete:** It is used to delete resources.

- **E. HTTP Patch:** It is used to modify capabilities.

- **F. HTTP Head:** The HEAD method is similar to GET, except without the response body. If GET /users return a list of users, then HEAD /users will make the same request but the list of users is not returned back

Advantages of REST

- Resource-based. A primary benefit of using REST, from both a client and server perspective,. Employing a resource-based approach, REST defines how developers interact with web services.
- Communication. REST-based interactions communicate their status through numerical HTTP status codes. REST APIs use these HTTP status codes to detect errors and ease the API monitoring process.

They include the following:

- 404 error indicates that a requested resource wasn't found;
- 401 status response code is triggered by an unauthorized request;
- 200 status response code indicates that a request was successful; and
- 500 error signals an unrecoverable application fault on the server.
- Familiarity. Most developers are already familiar with key elements of the REST architecture, such as Secure Sockets Layer (SSL) encryption and Transport Layer Security (TLS).
- Language-independent. When creating RESTful APIs or web services, developers can employ any language that uses HTTP to make web-based requests. This capability makes it easy for programmers to choose the technologies they prefer to work with and that best suit their needs.
- Pervasive. The popularity of REST is due to its widespread use in both server- and client-side implementations.

For example, on the server side, developers can employ REST-based frameworks, including Restlet and Apache CXF. On the client side, developers can employ a variety of frameworks (i.e., jQuery, Node.js, Angular, EmberJS, etc.) and invoke RESTful web services using standard libraries built into their APIs.

- Web APIs. When it comes to caching, RESTful services employ effective HTTP mechanisms. For example, by providing many endpoints, a REST API makes it easier for developers to create complex queries that can meet specific deployment needs.

Disadvantages of REST

- **Architecture.** Developers working with REST frequently encounter limitations due to its architecture design. These include multiplexing multiple requests over a single TCP connection, having different resource requests for each resource file, server request uploads, and long HTTP request headers, which cause delays in webpage loading.
- **Stateless applications :** Since HTTP does not store state-based information between request-response cycles, the client must perform state management tasks. This makes

it difficult for programmers to implement server updates without the use of client-side polling or other types of webhooks that send data and executable commands from one app to another.

- **Definition.** Developers generally disagree over defining REST-based designs. As an architectural style, REST lacks a clear reference implementation or a definitive standard that designates whether a specific design can be defined as RESTful. This also leads to uncertainty over whether a given web API conforms to REST-based principles.
- **Data overfetching/underfetching.** RESTful services frequently return large amounts of unusable data combined with relevant information, typically the result of multiple server queries. These inefficiencies also increase the time it takes for a client to return all the required data.

DIFFERENCE B/W REST & SOAP

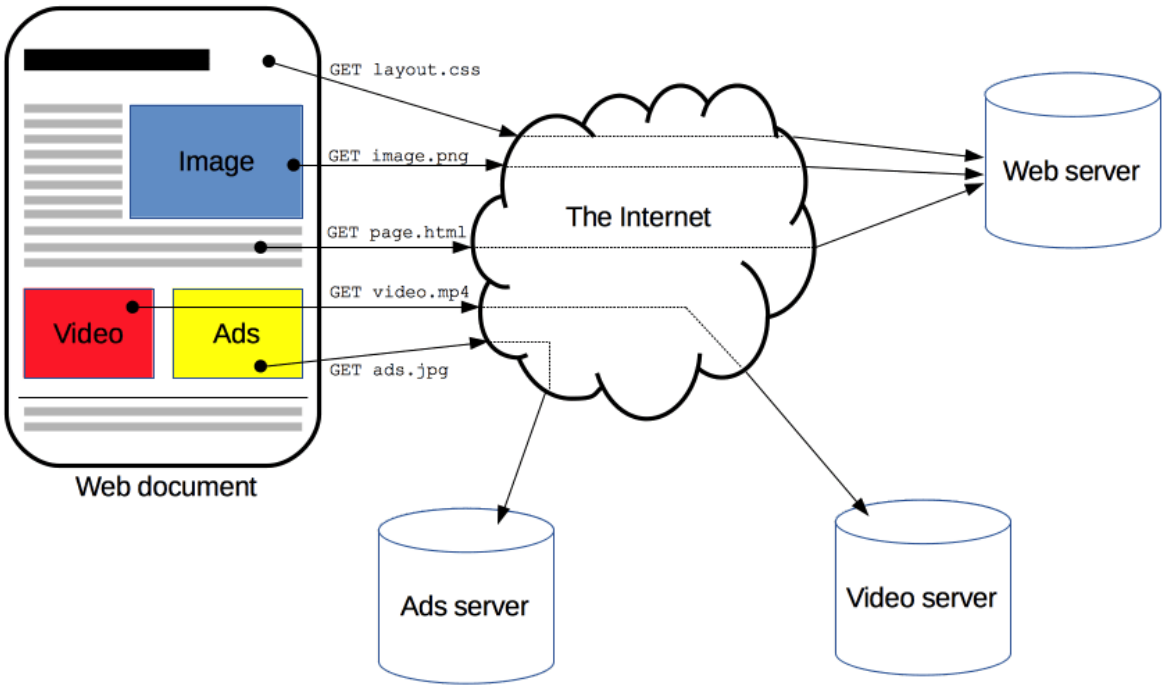
	REST	SOAP
What is it?	An architectual style	A protocol
Stands for?	REpresentional state transfer	Simple object access protocol
Can use the other?	Can use SOAP and any other protocol e.g. HTTP	It can't - because it's a protocol
Security	RESTful inherits security from the transport layer	Defines its own security
Bandwidth	Requires less	Requires more bandwidth
Data Formats Compatible	Plain Text, HTML, XML, JSON etc.	XML

HTTP PROTOCOL

- HTTP stands for **HyperText Transfer Protocol**.
- It is a protocol used to access the data on the World Wide Web (www).
- The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, and so on.
- This protocol is known as HyperText Transfer Protocol because of its efficiency that allows us to use in a hypertext environment where there are rapid jumps from one document to another document.
- HTTP was invented alongside HTML to create the first interactive, text-based web browser: the original World Wide Web. Today, the protocol remains one of the primary means of using the Internet.

HTTP Working

- As a request-response protocol, HTTP gives users a way to interact with web resources such as HTML files by transmitting hypertext messages between clients and servers. HTTP clients generally use Transmission Control Protocol(TCP) connections to communicate with servers.
- HTTP utilizes specific request methods in order to perform various tasks. All HTTP servers use the GET and HEAD methods, but not all support the rest of these request methods:
- GET requests a specific resource in its entirety
- HEAD requests a specific resource without the body content
- POST adds content, messages, or data to a new page under an existing web resource
- PUT directly modifies an existing web resource or creates a new URI if need be
- DELETE gets rid of a specified resource



Features of HTTP :

- **HTTP is simple :** HTTP is generally designed to be simple and human readable, even with the added complexity introduced in HTTP/2 by encapsulating HTTP messages into frames.

- **HTTP is extensible :** Introduced in HTTP/1.0, HTTP headers make this protocol easy to extend and experiment with. New functionality can even be introduced by a simple agreement between a client and a server about a new header's semantics.

- **HTTP is stateless, but not sessionless :** HTTP is stateless: there is no link between two requests being successively carried out on the same connection. This immediately has the prospect of being problematic for users attempting to interact with certain pages coherently, for example, using e-commerce shopping baskets. But while the core of HTTP itself is stateless, HTTP cookies allow the use of stateful sessions.

- **Media independent:** HTTP protocol is a media independent as data can be sent as long as both the client and server know how to handle the data content. It is required for both the client and server to specify the content type in MIME-type header.

HTTP Messages :

- HTTP messages are of two types: request and response. Both the message types follow the same message format.
- **Request Message:** The request message is sent by the client that consists of a request line, headers, and sometimes a body.
- **Response Message:** The response message is sent by the server to the client that consists of a status line, headers, and sometimes a body.

XMPP PROTOCOL

- The **extensible messaging and presence protocol (XMPP)** is now widely used as a communication protocol. Based on extensible markup language (XML), XMPP enables fast, near-real-time exchange of data between multiple entities on a network.
- The xmpp protocol is based on the typical client server architecture, in which the xmpp client uses the xmpp server with the tcp socket.
- Xmpp provides a general framework for messaging across a network, offering a multitude of applications beyond traditional instant messaging (im) and the distribution of presence data.
- It enables the discovery of services residing locally or across a network, as well as finding out about the availability of these services.

XMPP Functions :

- Sending and receiving direct messages between users
- Checking and communicating users' status (presence) information
- Managing contact lists and subscriptions to other users (in other words, adding friends/connections to chat with)
- Blocking individual users

XMPP WORKING :

- **Client-Server Architecture**
- XMPP works by passing small, structured chunks of XML data between endpoints (clients) via intermediary servers.
- In other words, if you send a message to your friend using XMPP, that message, as part of an XML document, first travels to a server instead of traveling directly to your friend's device.
- Each client has a unique name, similar to an email address, that the server uses to identify and route messages.
- XMPP provides a uniform way for each client to contact the server, keeping expectations consistent between machines.

XMPP ADVANTAGES:

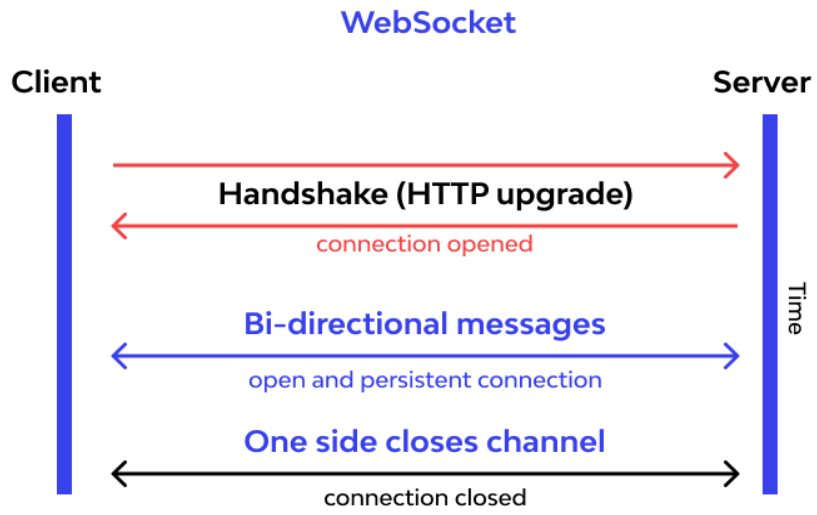
- Addressing scheme to recognize devices on the network
- Client-server architecture
- Decentralized
- Flexible
- Open standards and formalized

XMPP LIMITATIONS:

- Text-based messaging and no provision for end-to-end encryption
- No provision for quality of service
- The data flow is usually more than 70 per cent of the XMPP protocol server, of which nearly 60 per cent is repeated; the protocol has a large overhead of data to multiple recipients
- Absence of binary data
- Limited scope for stability

WEBSOCKET

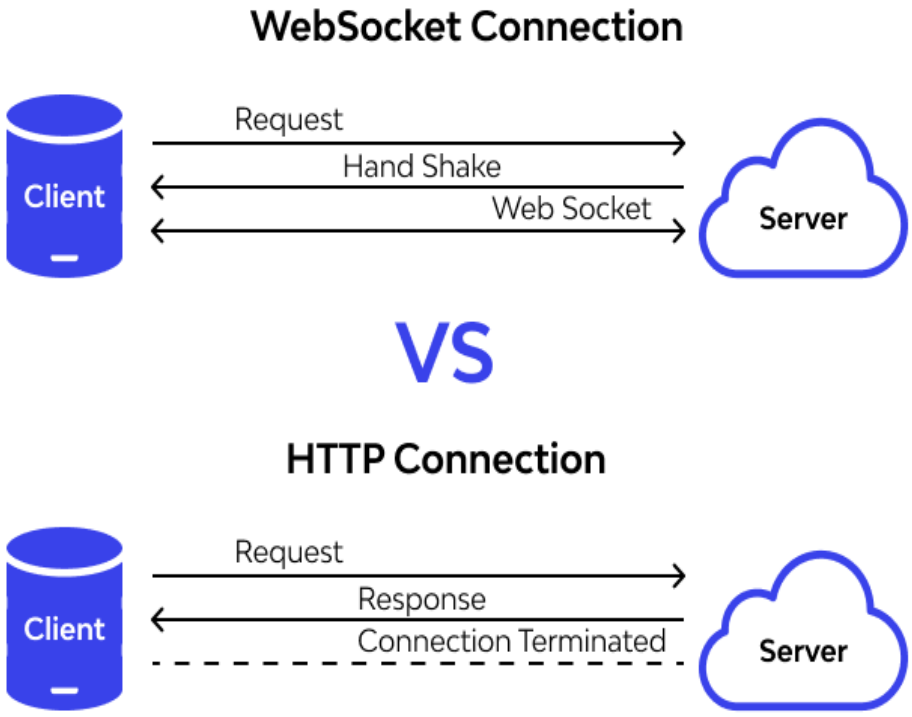
- A WebSocket is a persistent connection between a client and server. WebSockets provide a bidirectional, full-duplex communications channel that operates over HTTP through a single TCP/IP socket connection.



- It is used for real-time data updates and synchronization, live text chat, video conferencing, VOIP, IoT control and monitoring.
- These capabilities enable apps in gaming, social networks, logistics, finance and home, vehicle and industrial automation, to name a few.
- WebSocket is a duplex protocol used mainly in the client-server communication channel.
- The connection, developed using the WebSocket, lasts as long as any of the participating parties lays it off. Once one party breaks the connection, the second party won't be able to communicate as the connection breaks automatically at its front.
- **WebSocket need support from HTTP to initiate the connection.**

Why WebSocket?

- The idea of WebSockets was borne out of the limitations of HTTP-based technology.
- With HTTP, a client requests a resource, and the server responds with the requested data.
- HTTP is a strictly unidirectional protocol — any data sent from the server to the client must be first requested by the client.
- With long-polling, a client makes an HTTP request with a long timeout period, and the server uses that long timeout to push data to the client.
- WebSockets, on the other hand, allow for sending message-based data, similar to UDP, but with the reliability of TCP.
- WebSocket uses HTTP as the initial transport mechanism, but keeps the TCP connection alive after the HTTP response is received so that it can be used for sending messages between client and server.



WebSocket Features :

- Developing real-time web application
- Creating a chat application
- Working up on gaming application