# Lab Assignment No: 02

**AIM:** Implementing the Protocol MQTT with real life applications (Home Automation).

**OBJECTIVES:**

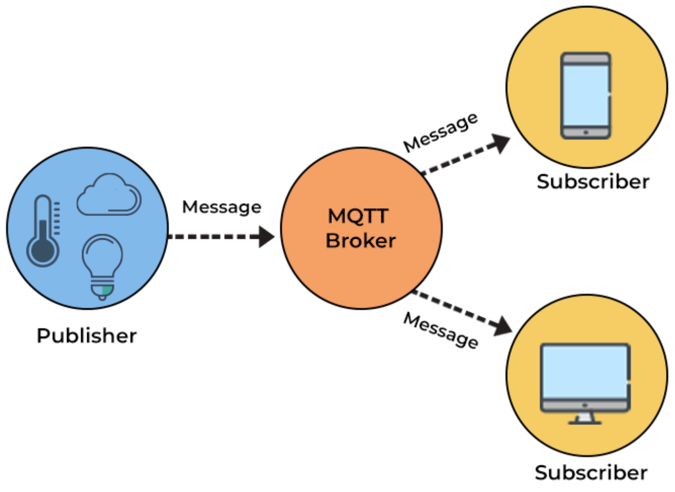
- To understand and implement the MQTT communication protocol.
- To design and develop a basic home automation system using IoT devices.
- To enable real-time monitoring and control of home appliances over the internet

**Components Required:**

- ESP8266 or ESP32 (or Raspberry Pi if preferred)
- DHT11/DHT22 Temperature & Humidity Sensor
- Wi-Fi Connection
- MQTT Broker (e.g., Mosquitto or public brokers like broker.hivemq.com)
- MQTT Client App (e.g., MQTT Explorer or mobile app like MyMQTT)
- Arduino IDE or Python (MicroPython) for coding

**THEORY:**

Message queuing telemetry support (MQTT) is defined as a low bandwidth consumption machine-to-machine protocol that helps IoT devices communicate with each other, with minimal code requirements and network footprint.



Message Queuing Telemetry Transport is suitable for machine-to-machine (M2M) communications because of its optimization for low-bandwidth and high-latency environments. MQTT is a publisher/subscriber protocol the main broker manages. This signifies that there is no direct link between the transmitter and the recipient.

Even though MQTT is frequently written as message queuing telemetry transport, MQTT communication does not use message queuing. Since all receivers with interest in specific messages ("marked by the topic") are enlisted as users, the sources of information publish their information, and all recipients with a desire for specific messages ("marked by the topic") access this information. MQTT is utilized for everything from IoT and IIoT to linking cloud environments in IoT and IIoT.

**How Does MQTT Work?**

MQTT (Message Queuing Telemetry Transport) is a lightweight **publish/subscribe (pub/sub)** communication protocol ideal for IoT applications like home automation. Unlike traditional models where clients talk directly, MQTT uses a **broker** to manage all communication.

**Core Concepts:**

- **Publisher**: Sends data (e.g., a sensor reporting temperature).
- **Subscriber**: Receives data (e.g., an app showing room temperature).
- **Broker**: Acts as a middleman, receiving messages from publishers and sending them to the right subscribers.

Publishers and subscribers don't need to know each other—communication happens only through the broker using **topics** (e.g., home/livingroom/temp).

**MQTT Message Structure**

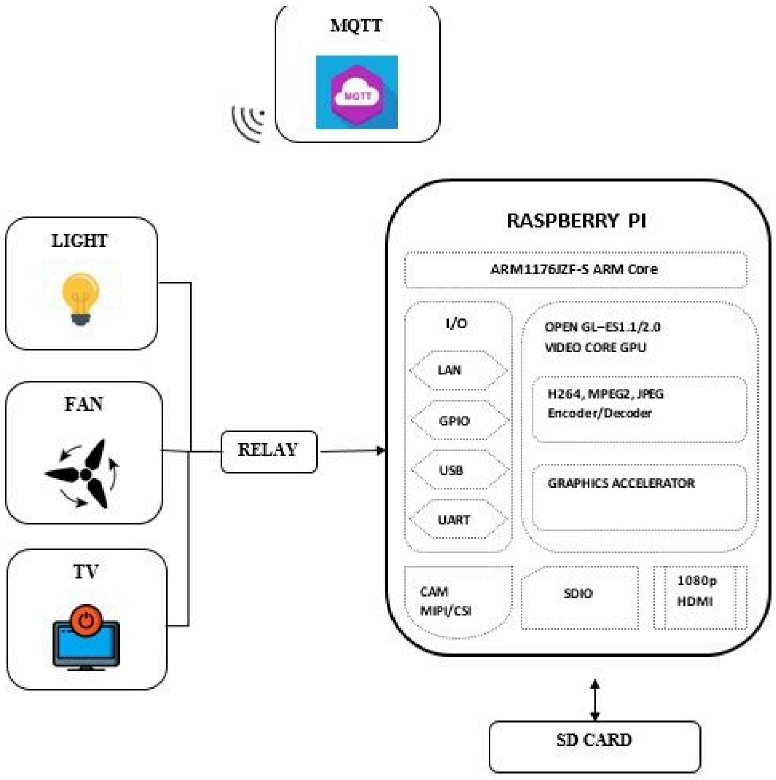
- Very compact: uses a 2-byte header for efficiency.
- Payloads can be up to 256 MB.
- Supports **three QoS (Quality of Service)** levels for reliability:
    - **QoS 0 (At most once)** – No guarantee of delivery. Lightweight and fast.
    - **QoS 1 (At least once)** – Guarantees delivery but may result in duplicates.
    - **QoS 2 (Exactly once)** – Ensures the message is delivered once without duplicates. Most reliable.

**Common MQTT Message Types**

1. **CONNECT** – Client connects to the broker (may use a username/password).
2. **PUBLISH** – Sends data to a topic (e.g., home/kitchen/lightStatus).
3. **SUBSCRIBE** – Listens to topics to receive updates from the broker.

**Why Use MQTT in Home Automation?**

- Efficient and lightweight – ideal for low-power IoT devices.
- Real-time control and monitoring of devices (lights, fans, temperature).
- Scalable – easily manages multiple devices with minimal bandwidth use.

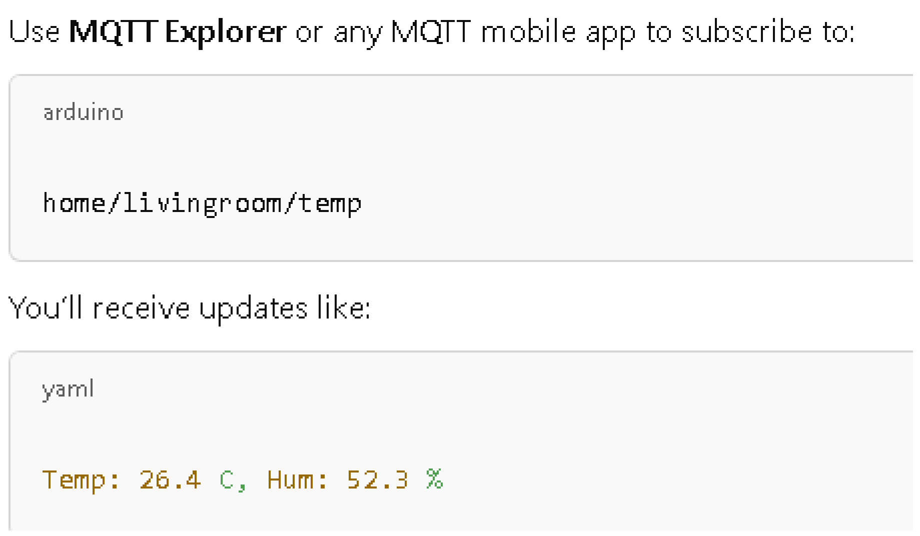


DHT11 Sensor --> ESP32/ESP8266 --> MQTT Broker --> Subscriber (Mobile/Desktop App)

```
#include <WiFi.h>

#include <PubSubClient.h>

#include "DHT.h"


#define DHTPIN 4
```

```
#define DHTTYPE DHT11


const char* ssid = "YourWiFiSSID";

const char* password = "YourWiFiPassword";

const char* mqttServer = "broker.hivemq.com";

const int mqttPort = 1883;


WiFiClient espClient;

PubSubClient client(espClient);

DHT dht(DHTPIN, DHTTYPE);


void setup() {

  Serial.begin(115200);

  dht.begin();


  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

  }


  client.setServer(mqttServer, mqttPort);

  while (!client.connected()) {

    if (client.connect("ESP32Client")) {
      Serial.println("Connected to MQTT Broker");
    } else {
      delay(500);
    }
  }
}

void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();

  if (!isnan(temp) && !isnan(hum)) {
    String payload = "Temp: " + String(temp) + " C, Hum: " + String(hum) + " %";
    client.publish("home/livingroom/temp", payload.c_str());
    Serial.println("Data sent: " + payload);
  }

  delay(5000); // Send every 5 seconds
}
```

**Output:**



Use **MQTT Explorer** or any MQTT mobile app to subscribe to:

```arduino
home/livingroom/temp
```

You'll receive updates like:

```yaml
Temp: 26.4 C, Hum: 52.3 %
```

**CONCLUSION:**