# 7.4 Efficient IoT Messaging with MQTT

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed for constrained devices and unreliable networks, making it perfect for IoT. Instead of the request-response model, MQTT uses a publish/subscribe (pub/sub) model.

## MQTT Components

- **Broker**: A central server that acts as a intermediary, receiving messages from publishers and distributing them to interested subscribers
- **Client (Publisher)**: A device (like an ESP32 with a sensor) that publishes messages (e.g., temperature readings) to a specific "topic" on the broker. It doesn't know or care who reads the message.
- **Client (Subscriber)**: Another device or application (like a mobile app or a server) that subscribes to that same topic. The broker automatically forwards any message published to that topic to all subscribers.

## MQTT Concepts

- **Topic**: A hierarchical string that acts as a channel for messages (e.g., home/livingroom/lamp).
- **Quality of Service (QoS)**:
  - **QoS 0 (At most once)**: The message is sent once ("fire and forget"). It's fast but offers no delivery confirmation.
  - **QoS 1 (At least once)**: The message is guaranteed to be delivered, but it might arrive more than once.
  - **QoS 2 (Exactly once)**: The most reliable level, guaranteeing the message is delivered exactly one time. It is the slowest due to a more complex handshake.

## MQTTS

Just like HTTPS, MQTTS is MQTT secured with TLS encryption to protect data confidentiality and integrity.

## Example

```
#include <WiFi.h>
#include <PubSubClient.h>

// --- Replace with your network credentials ---
const char* ssid = "YOUR_WIFI_SSID";
```

```
const char* password = "YOUR_WIFI_PASSWORD";
// ---------------------------------------

// --- MQTT Broker Configuration ---
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 1883;

// --- Topics ---
// Topic to publish messages to
const char* publish_topic = "Digilab/Modul7/status";
// Topic to subscribe to for incoming messages
const char* subscribe_topic = "Digilab/Modul7/command";

WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);

// This function is called whenever a message arrives on a subscribed topic
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String message;
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
  Serial.println(message);

  // Example: Turn on a built-in LED if the message is "ON"
  if (message == "ON") {
    Serial.println("Turning on lamp");
    digitalWrite(LED_BUILTIN, HIGH);
  } else if (message == "OFF") {
    Serial.println("Turning off lamp");
    digitalWrite(LED_BUILTIN, LOW);
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!mqttClient.connected()) {
    Serial.print("Attempting MQTT connection...");
```

```
    // Create a random client ID
    String clientId = "ESP32Client-";
    clientId += String(random(0xffff), HEX);

    // Attempt to connect
    if (mqttClient.connect(clientId.c_str())) {
      Serial.println("connected");
      // Subscribe to the command topic upon connection
      mqttClient.subscribe(subscribe_topic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(mqttClient.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}


void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected!");

  // Configure MQTT client
  mqttClient.setServer(mqtt_server, mqtt_port);
  mqttClient.setCallback(callback);
}

void loop() {
  // Ensure the MQTT client is connected
  if (!mqttClient.connected()) {
    reconnect();
```

```
  }
  mqttClient.loop(); // This allows the client to process incoming messages

  // --- Publish a message every 10 seconds ---
  static unsigned long lastMsg = 0;
  unsigned long now = millis();
  if (now - lastMsg > 10000) {
    lastMsg = now;

    char msg[50];
    snprintf(msg, 50, "Uptime: %lu seconds", millis() / 1000);

    Serial.printf("Publishing message to %s: %s\n", publish_topic, msg);
    mqttClient.publish(publish_topic, msg);
  }
}
```

---

Revision #2
Created 2025-08-28 16:59:39 UTC by NS
Updated 2025-08-29 13:52:11 UTC by NS