5. Comunicación de los datos utilizando MQTT

¿Que es MQTT?

MQTT (Message Queuing Telemetry Transport) es un protocolo de comunicación ligero y de publicación-suscripción que se utiliza comúnmente en aplicaciones de Internet de las cosas (IoT).

MQTT se basa en un modelo de mensajería de publicación/suscripción, en el que los clientes pueden publicar mensajes en un tema (topic) y otros clientes pueden suscribirse a ese tema para recibir los mensajes. Los temas son cadenas de caracteres que representan un canal de comunicación en particular. Los clientes pueden suscribirse a un tema específico y recibir los mensajes que se publican en ese tema.

MQTT se utiliza comúnmente en aplicaciones IoT debido a su baja sobrecarga y su capacidad para funcionar en redes con ancho de banda limitado y alta latencia. También es compatible con la seguridad a nivel de transporte mediante TLS/SSL, lo que lo hace adecuado para entornos empresariales y de seguridad crítica.

¿Cómo hago un primer ejemplo?

Este código es un ejemplo básico de cómo conectar un dispositivo ESP8266 a un broker MQTT usando la biblioteca PubSubClient de Arduino.

En la función `setup()`, se inicializan los pines del dispositivo y se configura la conexión Wi-Fi. A continuación, se establece la conexión con el servidor MQTT mediante el método `setServer()` de la biblioteca PubSubClient. También se establece la función `callback()` para manejar los mensajes entrantes.

En la función `loop()`, se verifica si el cliente MQTT está conectado. Si no lo está, se intenta una reconexión llamando a la función `reconnect()`. Luego, se llama a la función `loop()` del cliente para procesar los mensajes entrantes y salientes.

Finalmente, en la función `loop()`, se envía un mensaje al broker MQTT cada dos segundos mediante el método `publish()` del cliente, y se incrementa el valor de la variable `value` para que cada mensaje tenga un número de secuencia único.

En resumen, este código muestra cómo establecer una conexión básica entre un dispositivo ESP8266 y un broker MQTT y cómo enviar mensajes a través de la red MQTT.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Update these with values suitable for your network.
const char* ssid = "wifi_name";
const char* password = "wifi_password";
const char* mqtt_server = "mqtt.makersupv.com";
#define USER "makersupv"
#define PASS "makersupv"
String TEAM = "t00";
String OUT_TOPIC = "taller/" + TEAM + "/outTopic";
String IN_TOPIC = "taller/" + TEAM + "/inTopic";

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (64)
char msg[MSG_BUFFER_SIZE];
int value = 0;

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the voltage level
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
  } else {
    digitalWrite(BUILTIN_LED, HIGH);  // Turn the LED off by making the voltage HIGH
  }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String client_id = "";
    // Attempt to connect
    if (client.connect(client_id.c_str(), USER, PASS)) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish(OUT_TOPIC.c_str(), "hello world");
      // ... and resubscribe
```

```
      client.subscribe(IN_TOPIC.c_str());
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void setup() {
  pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an output
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    ++value;
    snprintf (msg, MSG_BUFFER_SIZE, "hello world #%ld", value);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(OUT_TOPIC, msg);
  }
}
```

Para poder verificar que estaís recibiendo información podeís descargaros una aplicación en vuestro movil como MyMQTT esta disponible para iOS y Android. Con esta aplicación podréis debuggear lo que está suciendo, si recibis mensajes, también si podéis actuar sobre los diferentes componentes.

Solución

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN  D7     // Pin digital conectado al sensor DHT
#define DHTTYPE DHT11  // DHT 11
DHT_Unified dht(DHTPIN, DHTTYPE);

#define LED_ROJO D5
#define LED_VERDE D6

#define LDR_PIN A0
int sensorLDRValue = 0;  // variable para almacenar el valor proveniente del sensor LDR



// Update these with values suitable for your network.
const char* ssid = "makersupv";
const char* password = "makersupv";
const char* mqtt_server = "mqtt.makersupv.com";
#define USER "makersupv"
#define PASS "makersupv"
String TEAM = "TEAM_MAKERS";
String OUT_TOPIC_TEMP = "taller/" + TEAM + "/sens/DHT11/temp";
String OUT_TOPIC_HUM = "taller/" + TEAM + "/sens/DHT11/hum";
String OUT_TOPIC_LDR = "taller/" + TEAM + "/sens/LDR";
String IN_TOPIC_LED_VERDE = "taller/" + TEAM + "/leds/verde";
String IN_TOPIC_LED_ROJO = "taller/" + TEAM + "/leds/rojo";

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE  (64)
char msg[MSG_BUFFER_SIZE];
int value = 0;
```

```
void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  if(topic == IN_TOPIC_LED_VERDE.c_str()){
    // Switch on the LED if an 1 was received as first character
    if ((char)payload[0] == '0') {
      digitalWrite(LED_VERDE, LOW);   // Turn the LED on (Note that LOW is the voltage
level
    } else if ((char)payload[0] == '1'){
```

```
      digitalWrite(LED_VERDE, HIGH);  // Turn the LED off by making the voltage HIGH
    }
  }

  if(topic == IN_TOPIC_LED_ROJO.c_str()) {
    // Switch on the LED if an 1 was received as first character
    if ((char)payload[0] == '0') {
      digitalWrite(LED_ROJO, LOW);   // Turn the LED on (Note that LOW is the voltage
level
    } else if ((char)payload[0] == '1'){
      digitalWrite(LED_ROJO, HIGH);  // Turn the LED off by making the voltage HIGH
    }
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String client_id = "";
    // Attempt to connect
    if (client.connect(client_id.c_str(), USER, PASS)) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      //client.publish(OUT_TOPIC.c_str(), "hello world");
      // ... and resubscribe
      client.subscribe(IN_TOPIC_LED_VERDE.c_str());
      client.subscribe(IN_TOPIC_LED_ROJO.c_str());
      //client.subscribe(OUT_TOPIC_HUM.c_str());

    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

```
// la función de setup() se ejecuta una vez cuando presionas el botón de reset o enciendes la placa
void setup() {

  Serial.begin(115200);        // inicializa el puerto serie (USB - COM) en una velocidad de comunicación de 115200 bauds
  Serial.println("\n\n");

  pinMode(LED_ROJO, OUTPUT);   // inicializa el pin digital LED_ROJO como salida.
  pinMode(LED_VERDE, OUTPUT);  // inicializa el pin digital LED_VERDE como salida.

  dht.begin(); // Inicializar dispositivo DHT
  Serial.println(F("Sensor DHT11"));
// Imprime los detalles del sensor de temperatura.
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  Serial.println(F("------------------------------------"));
  Serial.println(F("Sensor de Temperatura"));
  Serial.print  (F("Valor Max:   ")); Serial.print(sensor.max_value);
Serial.println(F("°C"));
  Serial.print  (F("Valor Min:   ")); Serial.print(sensor.min_value);
Serial.println(F("°C"));
  Serial.print  (F("Resolucion:  ")); Serial.print(sensor.resolution);
Serial.println(F("°C"));
// Imprime los detalles del sensor de humedad.
  dht.humidity().getSensor(&sensor);
  Serial.println(F("Sensor de Humedad"));
  Serial.print  (F("Valor Max:   ")); Serial.print(sensor.max_value);
Serial.println(F("%"));
  Serial.print  (F("Valor Min:   ")); Serial.print(sensor.min_value);
Serial.println(F("%"));
  Serial.print  (F("Resolucion:  ")); Serial.print(sensor.resolution);
Serial.println(F("%"));
  Serial.println(F("------------------------------------"));

  pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an output
```

```
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

// la función de loop() se ejecuta una y otra vez para siempre
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;

    // read the value from the sensor:
    sensorLDRValue = analogRead(LDR_PIN);
    Serial.print(F("LDR: "));
    Serial.print(sensorLDRValue);
    snprintf (msg, MSG_BUFFER_SIZE, "%ld", sensorLDRValue);
    client.publish(OUT_TOPIC_LDR.c_str(), msg);

    mostrar_datos_sensor_DHT11();

    Serial.println();

  }
}

void mostrar_datos_sensor_DHT11(){
  // Obtenga si hay evento de temperatura e imprima su valor.
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature)) {
    Serial.print(F("\t"));
    //Serial.println(F("¡Error al leer la temperatura!"));
  }
  else {
```

```
    Serial.print(F("\t"));
    Serial.print(F("Temperatura: "));
    Serial.print(event.temperature);
    snprintf (msg, MSG_BUFFER_SIZE, "%ld", (int) event.temperature);
    client.publish(OUT_TOPIC_TEMP.c_str(), msg);
    Serial.print(F("°C"));
  }
  // Obtenga si hay evento de humedad e imprima su valor.
  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity)) {
    Serial.print(F("\t"));
    //Serial.println(F("¡Error al leer la humedad!"));
  }
  else {
    Serial.print(F("\t"));
    Serial.print(F("Humedad: "));
    Serial.print(event.relative_humidity);
    snprintf (msg, MSG_BUFFER_SIZE, "%ld", (int) event.relative_humidity);
    client.publish(OUT_TOPIC_HUM.c_str(), msg);
    Serial.print(F("%"));
  }
}
```

Ahora te toca a ti, coge el ejemplo del tutorial 4 y combinalo con el ejemplo anterior para poder enviar y recibir datos de los sensores.

Estos son los temas (topics) que debes usar:

Tipo	**Descripción**	**Unidades**	**Topic MQTT**
Sensor	Lectura sensor temperatura DHT11	ºC	taller/**tXX** /sens/DHT11/temp
Sensor	Lectura sensor humedad relativa DHT11	%rh	taller/**tXX**/sens/DHT11/hum
Sensor	Lectura sensor luminosidad LDR	Volts	taller/**tXX**/sens/LDR
Actuador LED	Indicador LED Verde		taller/**tXX**/leds/verde

Actuador LED	Indicador LED Rojo		taller/**tXX**/leds/rojo

> Recuerda cambiar **t00** por el número de tu equipo

Revision #13
Created 28 April 2023 19:48:27 by Rafa
Updated 11 May 2023 18:33:03 by Rafa