IoT Communication - MQTT

Welcome to Day 3 of our IoT Workshop Series! Building upon the foundation laid on Day 2, where you delved into the intricacies of sensors and their protocols, today is all about application. Get ready to harness that knowledge and dive into the world of data transmission as we explore how to gather and send sensor data using MQTT.

1. MQTT in detail

2. ESP8266 and WiFi

Next, we'll take a closer look at connecting the ESP8266 to WiFi. This step is crucial for enabling wireless communication, allowing your devices to seamlessly interact with the digital world.

In this example code, you will first of all connect to a WiFi network and then try to access a predefined API. This API is the root of our public weather station API. You can find out more about the project right here: HSRW Weather Station at Campus Kamp-Lintfort.

wifi-http-api-example.ino

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

const uint16_t HTTP_PORT = 443; // Use 443 for HTTPS and 80 for HTTP

// Replace with your WiFi credentials
const char* SSID = "iotlab";
const char* PASSWORD = "iotlab18";

// Replace with your API details
const char* API_HOST = "weather.eolab.de";
const char* API_ENDPOINT = "/api";

const uint8_t PERIOD_MINUTES = 1;

// Create an instance of WiFiClientSecure
WiFiClientSecure client;

void setup() {
  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.begin(SSID, PASSWORD);
```

```
  Serial.print("\n\nConnecting to ");
  Serial.print(SSID);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("\nConnected to WiFi");

  // Set the client to verify the server's certificate
  client.setInsecure();

  // Give the client a chance to perform the handshake
  delay(1000);
}

void loop() {
  if (WiFi.status() == WL_CONNECTED) {

    client.connect(API_HOST, HTTP_PORT);
    // Make an HTTPS GET request
    client.println("GET " + String(API_ENDPOINT) + " HTTP/1.1");
    client.println("Host: " + String(API_HOST));
    client.println("Connection: close");
    client.println();

    // Wait for the response
    while (client.connected()) {
      if (client.available()) {
        // read an incoming byte from the server and print it to serial
monitor:
        char c = client.read();
        Serial.print(c);
      }
    }

    client.stop();
  }

  // Wait for a while before making the next request
  delay(PERIOD_MINUTES * 60 * 1000); // convert from min to sec (60)
and from sec to ms (1000)
}
```

3. Our first MQTT Publish

Stepping ahead, you'll have the chance to implement your first MQTT publish from the ESP8266. While the following example provides a static demonstration, we encourage you to take it a step further by incorporating one of the sensors available to you.

To start, search and install the library “PubSubClient ” by Nick O'Leary in Arduino IDE or download and install it manually from source Github PubSubClient by knolleary.

mqtt-publish.ino

```
/*
 Basic ESP8266 MQTT example
 This sketch demonstrates the capabilities of the pubsub library in
combination
 with the ESP8266 board/library.
 It connects to an MQTT server then:
  - publishes "hello world" to the topic defined as outTopic every two
seconds
 It will reconnect to the server if the connection is lost using a
blocking
 reconnect function. See the 'mqtt_reconnect_nonblocking' example for
how to
 achieve the same result without blocking the main loop.
*/

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

#define MSG_BUFFER_SIZE   (50)

// Update these with values suitable for your network and mqtt broker

const char* ssid = "";
const char* password = "";
const char* mqtt_server = "broker.hivemq.com";

const char* outTopic = "unicaes/workshop/2023/jan";
const char* inTopic = "unicaes/workshop/2023/to-jan";

// Set up of some needed variables

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
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 reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish(outTopic, "hello world");
    } 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);
}

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(outTopic, msg);
  }
}
```

4. Subscribe

Now, let's move on to an essential aspect of our workshop. We'll guide you through the process of subscribing to an MQTT topic. This step is a practical gateway to interactive IoT applications. You'll learn how to exchange MQTT messages between devices, potentially triggering actions like LED reactions or serial outputs. While we'll provide initial guidance, feel free to explore further and experiment with this dynamic feature.

mqtt-subscribe-example.ino

```
/*
 Basic ESP8266 MQTT example
 This sketch demonstrates the capabilities of the pubsub library in
combination
 with the ESP8266 board/library.
 It connects to an MQTT server then:
  - publishes "hello world" to the topic defined as outTopic every two
seconds
  - subscribes to the topic defined as inTopic, printing out any
messages
    it receives. NB - it assumes the received payloads are strings not
binary
  - If the first character of the topic defined as inTopic is an 1,
switch ON the ESP Led,
    else switch it off
 It will reconnect to the server if the connection is lost using a
blocking
```

```
 reconnect function. See the 'mqtt_reconnect_nonblocking' example for
how to
 achieve the same result without blocking the main loop.
*/

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

// Update these with values suitable for your network and mqtt broker

const char* ssid = "";
const char* password = "";
const char* mqtt_server = "broker.hivemq.com";

const char* outTopic = "unicaes/workshop/2023/jan";
const char* inTopic = "unicaes/workshop/2023/to-jan";

// Set up of some needed variables

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE   (50)
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...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish(outTopic, "hello world");
      // ... and resubscribe
      client.subscribe(inTopic);
    } 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(outTopic, msg);
  }
}
```

Recording

From:
https://wiki.eolab.de/ - **HSRW EOLab Wiki**

Permanent link:
https://wiki.eolab.de/doku.php?id=latinet:unicaes:workshops:communication-23&rev=1725993283

Last update: **2024/09/10 20:34**