UNIVERSIDAD DE COLIMA

Universidad de Colima

Facultad de Telemática

Tecnología de Redes Emergentes

Académico:

Amezcua Valdovinos Ismael

Alumno

Valdez Gutierrez Aldo Eduardo

Publicación de mensajes con MQTT

Colima, Col; a 16 de abril de 2024

Universidad de Colima
Facultad de Teleática
Publicación de mensajes con MQTT

Para implementar hacer publicaciones en Wokwi usaremos la librería de PubSubClient la cual nos sirve para implementar MQTT en Arduino en si y al Wokwi ser una simulación de Arduino esta librería nos sirve para implementar en Wokwi el conectarnos, publicar, suscribirnos a MQTT se puede encontrar más información sobre esto en https://github.com/knolleary/pubsubclient

A su vez para la lectura de temperaturas usaremos la librería de DHT la cual nos permite usar los sensores de DHT la cual nos permite leer obtener, procesar y leer temperaturas:

```
#include <DHT.h>
```

Y finalmente usamos la librería de WiFI para conectarnos a internet en esta ocupamos configurar la conexión WiFi en WokWi con los siguientes datos:

const char* ssid = "Wokwi-GUEST";
const char* password = "";

Y la importación de la librería:

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

Primeramente, ya con el código de medición de temperatura hecho empezaremos a implementar el PubSubClient, empezaremos incluyendo la librería en nuestro código:

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

Luego pondremos todos los datos que necesitamos para el funcionamiento donde se incluye la dirección del bróker, el cliente, el tema y el puerto donde va funcionar:

```
const char* mqtt_broker = "broker.hivemq.com";
const char* mqtt_topic = "ucol/iot/aldo";
const char* mqtt_client_id = "cliente_aldo";
const int mqtt_port = 1883;
```

Luego haremos una función para reconectarnos al bróker en caso de que se pierda la conexión con este, la cual va intentar conectarnos de nuevo cuando se pierda la conexión y también cuando se inicie la ejecución quedando así:

```
void reconnect() {
  while (!mqttClient.connected()) {
    Serial.println("Conectando con broker...");

    if (mqttClient.connect(mqtt_client_id)) {
      Serial.println("Conectado a broker");
    } else {
      Serial.print("Error: ");
      Serial.println(mqttClient.state());
      Serial.println("Reintento en 5 sec...");
      delay(5000);
   }
  }
```

```
}
```

Luego en la función principal de Setup vamos crear el servidor de MQTT donde nos vamos a conectar dando de parámetros la dirección principal y los puertos quedando estos así:

```
mqttClient.setServer(mqtt_broker, mqtt_port);
```

Luego en la función loop empezaremos viendo si existe una conexión de MQTT si no existe esta se manda a llamar la función de reconnect para conectarnos al bróker:

```
  if (!mqttClient.connected()){
    reconnect();
  }
  mqttClient.loop();
```

Finalmente hacemos un char el cual se publicará en el tema:

```
  char json[200];
  sprintf(json, "{\"temperature\": %f, \"humidity\": %f}", t, h);
  Serial.println(json);
  mqttClient.publish(mqtt_topic, json);
```

Usaremos el mqttClient.publish y este llevara de parámetros el tema y el nombre del char que en este caso es json.

Ahora para recibir los mensajes cree un código de JavaScript junto con node.js el cual va usar la librería de MQTT para crear la conexión y la suscripción al tema para recibir los mensajes, a su vez para recibir un mensaje de error si no se logro conectar o un mensaje de desconexión cuando se cierre esta quedando así:

```
const mqtt = require('mqtt');

const mqtt_broker = 'mqtt://broker.hivemq.com';
const mqtt_topic = 'ucol/iot/aldo';
const mqtt_client_id = 'cliente_aldo_suscriptor';

const client = mqtt.connect(mqtt_broker, {
    clientId: mqtt_client_id
});

client.on('connect', function () {
    console.log('Conectado al broker MQTT');

    client.subscribe(mqtt_topic, function (err) {
        if (!err) {
            console.log('Suscripción exitosa al tema:', mqtt_topic);
        } else {
            console.error('Error al suscribirse al tema:', err);
```

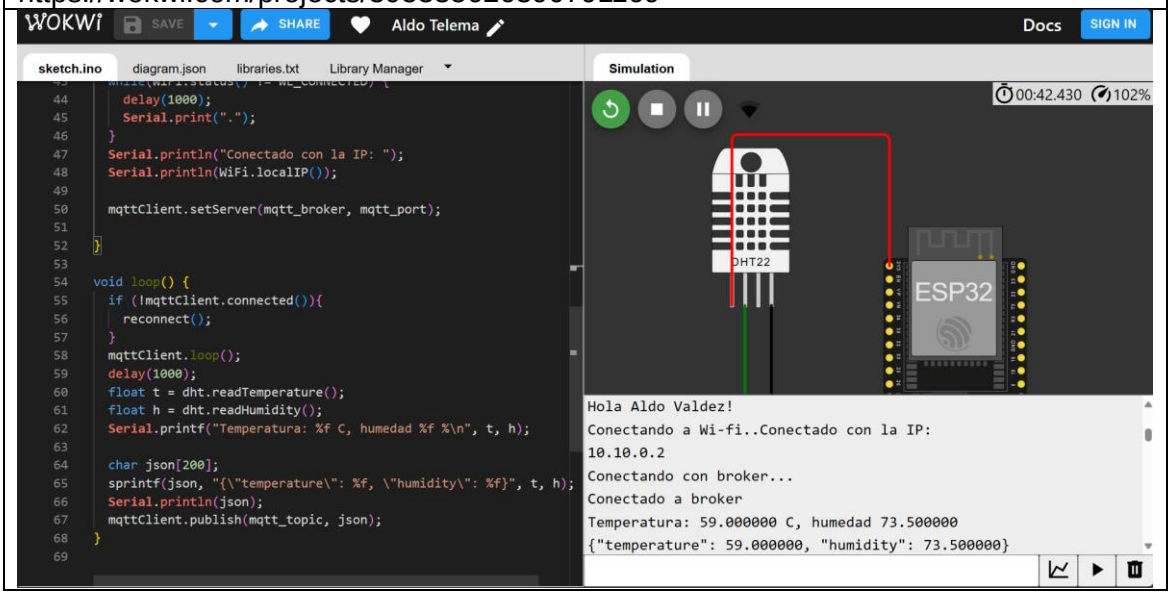
```
        }
    });
});

client.on('message', function (topic, message) {
    console.log('Mensaje recibido en el tema:', topic,
message.toString());
});

client.on('error', function (error) {
    console.error('Error de conexión:', error);
});

client.on('close', function () {
    console.log('Conexión cerrada');
});
```

Ejecutamos esto en nuestra terminal usando node (nombre archivo en mi caso client.js viendo los resultados de esta:

```
pwsh  0ms                                   79%  16,17:50
» node '.\Documents\MQTT ESP32\client.js'
Conectado al broker MQTT
Suscripción exitosa al tema: ucol/iot/aldo
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
```

Evidencias:

https://wokwi.com/projects/395335626390791169



```
Conectado al broker MQTT
Suscripción exitosa al tema: ucol/iot/aldo
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
Mensaje recibido en el tema: ucol/iot/aldo {"temperature": 59.000000, "humidity": 73.500000}
```