– A12: MQTT Publisher - Temperature and Brightness –

Objetivo:

La dirección de la ESI ha contactado con nosotros con el fin de monitorizar la temperatura y luminosidad de las diferentes aulas del edificio Fermín Caballero con el fin de estudiar el gasto energético de acuerdo con esas dos variables. Para tal fin deberemos desplegar varios nodos con comunicación WiFi (ESP32) y dos sensores capaces de medir dichas variables. A su vez, la comunicación se realizará mediante el protocolo MQTT y cuya arquitectura constará de un broker y al menos un subscriptor que se ejecutarán en un pc, mientras que los publicadores corresponderán a los nodos ESP32.

A partir de aquí se pide:

- Desarrollo del nodo, que será replicado tantas veces como integrantes contenga el grupo de trabajo.
- La temperatura será expresada en grados centígrados, mientras que la luminosidad en porcentaje.
- Definición del mensaje transmitido (JSON, protobuf, texto, ….).
- Definición de los topics donde publicará cada nodo. Deberá ser distinto para cada nodo.
- Configuración del broker MQTT distinta a la por defecto
- Topic del subscriptor, que deberá recibir todos los mensajes de los diferentes nodos

En nuestro caso, ya que no vivimos en la misma ciudad y para realizar la práctica lo idóneo sería estar conectados a la misma red Wi-Fi, no podremos realizar una demostración con los tres nodos funcionando de forma simultánea. Aun así, el código desarrollado para cada nodo es idéntico en todos ellos salvo el topic en el que deben publicar. Por tanto, la única línea que cambiará en cada nodo será la 17 que tendrá el siguiente formato:

#define TOPIC "A12/NODO<Letra>".

Solución:

A través de los puertos GPIO se puede realizar una conversión analógico-digital, por lo tanto, es posible tomar un valor de voltaje entre 0 y 3,3v y mapearlo con una resolución de 12 bits para convertirlo en un valor entre 0 y 4095.

Definimos las siguientes variables y pines:

```
#define LIGHT_SENSOR_PIN 32 // ESP32 pin GIOP32
#define TEMPERATURE_SENSOR_PIN 33 //ESP32 pin GPIO33

const int B = 4275;               // B value of the thermistor
const int R0 = 100000;            // R0 = 100k

const char* ssid = "AndroidApp";
const char* password = "12345678";

#define TOPIC "A12/NODOA"
#define BROKER_IP "192.168.43.174" //direccion IP del broker
#define BROKER_PORT 2883 //puerto en el que escucha el broker

WiFiClient espClient;
PubSubClient client(espClient);
```

La solución consta de dos métodos principales: uno para conectarse al Wi-Fi y otro para realizar la conexión mqtt. La lógica principal del programa se realizará en el void loop.

En el primero de los métodos, nos conectaremos a una red Wi-Fi con el ssid y contraseñas declaradas anteriormente. Mediante un bucle while esperaremos hasta que esto ocurra y mostraremos la IP por pantalla.

```
void wifiConnect()
{
  WiFi.begin(ssid, password);

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

  Serial.println("Connected to the WiFi network");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}
```

En el segundo método, establecemos la conexión mqtt mediante la IP y el puerto del bróker. El resto del código se encarga del control de posibles errores.

```
void mqttConnect() {
  client.setServer(BROKER_IP, BROKER_PORT);
  while (!client.connected()) {
    Serial.print("MQTT connecting ...");

    if (client.connect("ESP32Client1")) {
      Serial.println("connected");
    } else {
      Serial.print("failed, status code =");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");

      delay(5000);  //* Wait 5 seconds before retrying
    }
  }
}
```

En el void setup inicializamos los dos métodos anteriores, así como la comunicación serial en 115200 bits por segundo.

```
void setup() {
  // initialize serial communication at 115200 bits per second:
  Serial.begin(115200);
  wifiConnect();
  mqttConnect();
}
```

Como mencionamos anteriormente, la lógica del programa la encontramos en el void loop:

```
void loop() {
  // reads the input on analog pin (value between 0 and 4095)

  int lightValue = analogRead(LIGHT_SENSOR_PIN);
  int a = analogRead(TEMPERATURE_SENSOR_PIN);

  float R = 4095/a-1.0;
  R = R0*R;
  float temperature = 1.0/(log(R/R0)/B+1/298.15)-273.15;

  lightValue=(lightValue*100)/4095;

  char chtemp[8];
  dtostrf(temperature,4,2,chtemp);

  char chlight[4];
  dtostrf(lightValue,3,0,chlight);

  char message[100] = "Temperature value = ";
  strcat(message,chtemp);
  strcat(message," C - Light percentage(%) = ");
  strcat(message, chlight);

  client.publish(TOPIC,message);

  delay(500);

}
```

Leemos los datos de los sensores de luz y temperatura y los almacenamos en distintas variables.

Calculamos la temperatura a partir del valor obtenido por el sensor.

Mediante la función dtostrf de Arduino realizamos las conversión de valores numéricos a cadenas de texto.

Concatenamos las cadenas al mensaje que subirá el publicador.

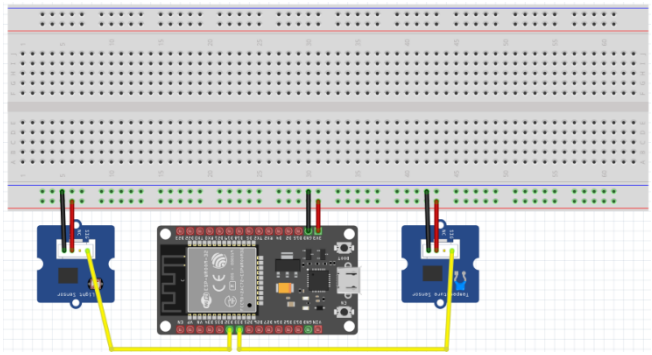
Por último, el nodo publica el mensaje en el topic definido al inicio del código (ver primera imagen).

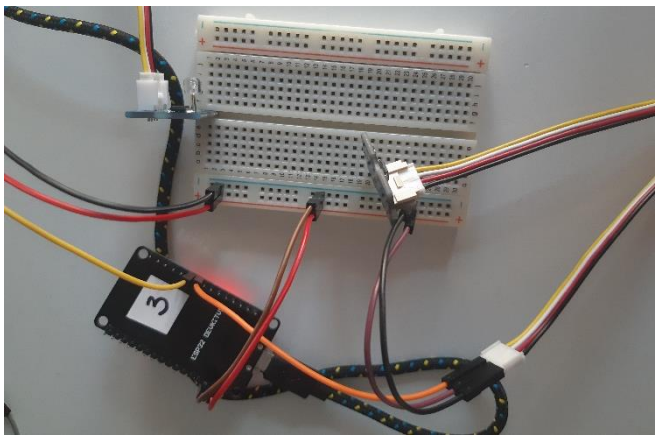
Configuración del broker(broker.config):

La configuración del broker que utilizaremos para crearlo se encuentra en el archivo broker.config situado en el directorio principal. La creación del mismo será lo primero que deberemos hacer para poder ejecutar el programa ya que si no esta iniciado, los nodos no podrán realizar la conexión mqtt y el programa quedará bloqueado en el método mqttConnect. La direccion ip utilizada en este archivo dependerá de la red WI-FI utilizada para la ejecución del programa.

```
listener 2883 192.168.43.174
max_connections 4
allow_anonymous true
connection_messages true
```

Circuito diseñado:





Modo de ejecución(Windows):

Para la ejecución del programa primero deberemos iniciar el broker. Para ello utilizaremos el comando:

"C:\Program Files\mosquitto\mosquitto.exe" -c broker.conf

A continuación iniciaremos el suscriptor que será quien reciba todos los mensajes enviados por los nodos publicadores. Para ello el comando utilizado será:

"C:\Program Files\mosquitto\mosquitto_sub.exe" -t A12/# -h 192.168.43.174 -p 2883

Como podemos ver; el topic utilizado para el suscriptor es A12/#. Esto le permitirá recibir los mensajes publicados por todos los nodos.

La salida del programa en el suscriptor deberá ser algo así: