CONEXIÓN MQTT, BROKER – CLIENTE INFORME N°3

Cachay,William. Castellanos,Paula.Laverde,Miguel
williamcachay@unisangil.edu.co
paulacastellanos@unisangil.edu.co
miguellaverde@unisangil.edu.co

Resumen - **Este informe es acerca de la práctica de laboratorio N°3 de la asignatura Internet de las cosas (IOT), el cual consta de un sistema domótico de apertura de una puerta, control de temperatura y un mensaje por medio de correo, se propone una implementación de MQTT (Message Queue Telemetry Transport) , el cual es un protocolo de comunicación M2M (machine-to-machine) de tipo message queue que se le da una serie de instrucciones, este nos permite conectarnos como clientes a un servidor central denominado bróker, un cliente puede publicar mensajes en un determinado topic y los clientes que estén suscritos al topic el bróker les enviará este mensaje. Esta implementación se desarrolló en montaje en protoboard y un microcontrolador arduino el cual se programó para llevar a cabo el MQTT.**

Términos relevantes **– MQTT, Internet, Temperatura, Puertas, M2M, Broker.**

Abstract- This report is about laboratory practice N ° 3 of the subject Internet of things (IOT), which consists of a home automation system for opening a door, temperature control and a message by mail, it is proposes an implementation of MQTT (Message Queue Telemetry Transport), which is a communication protocol M2M (machine-to-machine) of type message queue that is given a series of instructions, this allows us to connect as clients to a central server called a broker, a client can post messages on a certain topic and the clients that are subscribed to the topic will be sent this message by the broker. This implementation was developed in a breadboard mount and an arduino microcontroller which was programmed to carry out the MQTT.

Relevant terms - MQTT, Internet, Temperature, M2M, broker.

I. INTRODUCCIÓN

MQTT es un protocolo de mensajería estándar de OASIS para Internet de las cosas (IoT). Está diseñado como un transporte de mensajería de publicación / suscripción extremadamente liviana que es ideal para conectar dispositivos remotos con una huella de código pequeña y un ancho de banda de red mínimo. Hoy en día, MQTT se utiliza en una amplia variedad de industrias, como la automotriz, la fabricación, las telecomunicaciones, el petróleo y el gas, etc [1].

Está basado en la pila TCP/IP como base para la comunicación. En el caso de MQTT cada conexión se mantiene abierta y se "reutiliza" en cada comunicación. Es una diferencia, por ejemplo, a una petición HTTP 1.0 donde cada transmisión se realiza a través de conexión. MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech) en 1999 como un mecanismo para conectar dispositivos empleados en la industria petrolera. Aunque inicialmente era un formato propietario, en 2010 fue liberado y pasó a ser un estándar en 2014 según la OASIS (Organization for the Advancement of Structured Information Standards). El funcionamiento del MQTT es un servicio de mensajería push con patrón publicador/suscriptor (pub-sub). Como vimos en la entrada anterior, en este tipo de infraestructuras los clientes se conectan con un servidor central denominado broker. Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes suscritos [2].

Los clientes inician una conexión TCP/IP con el broker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS. Para ello el cliente envía un mensaje CONNECT que contiene información necesaria (nombre de usuario, contraseña, client-id…). El broker responde con un mensaje CONNACK, que contiene el resultado de la conexión (aceptada, rechazada, etc). Para enviar los mensajes el cliente emplea mensajes PUBLISH, que contienen el topic y el payload. Para suscribirse y desuscribirse se emplean mensajes SUBSCRIBE y UNSUSCRIBE, que el servidor responde con SUBACK y UNSUBACK. Por otro lado, para asegurar que la conexión está activa los clientes mandan periódicamente un mensaje PINGREQ que es respondido por el servidor con un PINGRESP. Finalmente, el cliente se desconecta enviando un mensaje de DISCONNECT [2].

II. OBJETIVOS

- Realizar una conexión MQTT entre el Broker y el Microcontrolador.

III. MATERIALES

- Multímetro Digital.
- Fuente de alimentación.
- Protoboard.

- Jumpers Macho a Macho a Hembra (o un metro de cable utp).
- Modulo ESP 01 (ESP8266).
- Arduino UNO (o ESP 32 u otro microcontrolador).
- Demás materiales necesarios.

IV. MARCO TEÓRICO

- **MQTT**

MQTT significa MQ Telemetry Transport, pero anteriormente se conocía como Message Queuing Telemetry Transport. MQTT se está convirtiendo rápidamente en uno de los principales protocolos para despliegues de IOT (Internet de las cosas). Sin embargo, hay una serie de factores importantes que competan su indudable funcionalidad, es muy interesante protocolo que se puede usar con nuestros arduinos o módulos para comunicarse con distintos sensores, uno de los campos donde mejores resultados se pueden dar es en la domótica.[3]

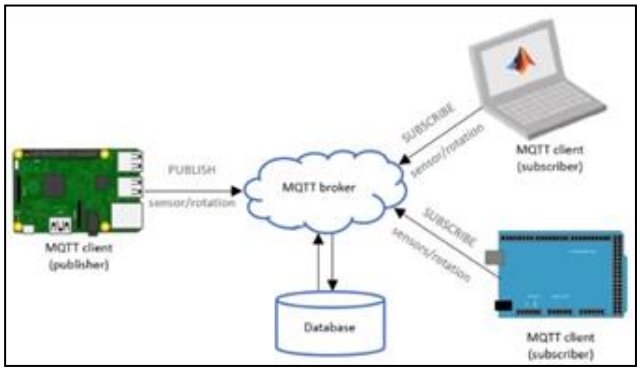

Imagen 1. MQTT.

- **BROKER**

Un bróker es una entidad financiera autorizada y regulada que ejecuta las ordenes de los traders en los mercados. También es responsable de la seguridad de los fondos que le pertenece a cada cliente. El bróker le proporciona al trader la plataforma de trading y una gran colección de herramientas de análisis, varios tipos de gráficos, etc.[4]

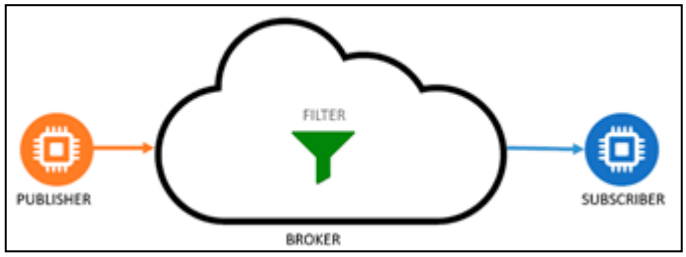

Imagen 2. Broker.

- **CONTROL DE TEMPERATURA Y HUMEDAD**

Control de la temperatura ambiente dependiendo de la temperatura exterior y de los hábitos de calefacción de los residentes, las necesidades de calefacción de habitaciones, apartamentos y edificios enteros cambian constantemente. Los termostatos de radiador y los sistemas de control de la temperatura compensan estas fluctuaciones, manteniendo en cada habitación la temperatura en el nivel fijado específicamente por el residente.[5]

Imagen 3. Control de temperatura y humedad.

V. DESARROLLO DEL LABORATORIO

A. Procedimiento

1. Diseñe e implemente un sistema domótico de apertura de una puerta, control de temperatura y notificación por medio de correo, la siguiente figura; expone el sistema en general.

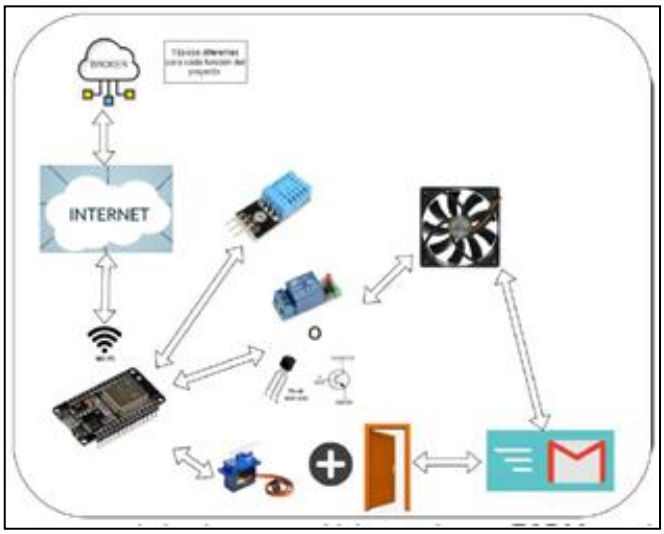

Ilustración 1. Sistema Domótico.

El sistema debe tener las siguientes características:

2. Debe tener un bróker MQTT para centralizar la información, implementa el bróker como se explicó en clase (https://myqtthub.com/).

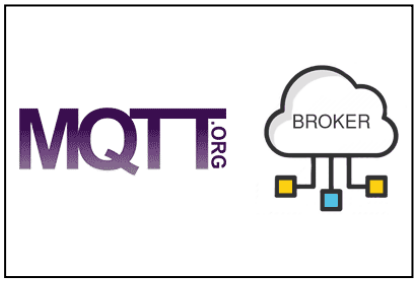

Ilustración 2. Representación de MQTT y un Broker.

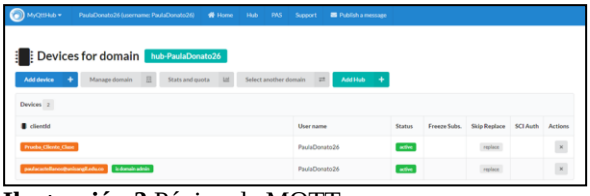

Ilustración 3.Página de MQTT.

3. Se debe disponer de los tópicos:

- ✓ Topic_Raiz: Debe informar toda la información del sistema temperatura, humedad y si la puerta está abierta o cerrada.

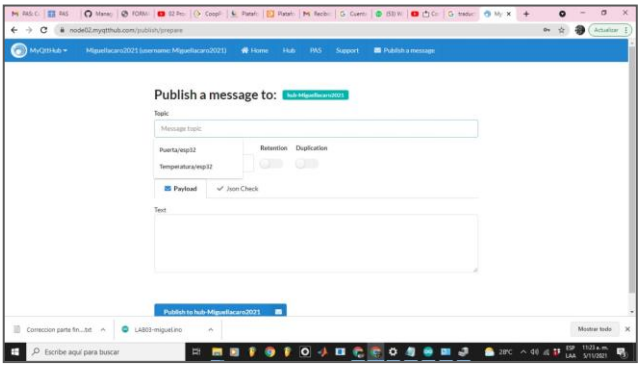

Ilustración 4. Topic_Raiz.

- ✓ Temperatura: Información de la temperatura actual actualizada cada 25 segundos, cuando se publique desde el bróker a este topic se debe enviar la temperatura para configurar el sistema de control de temperatura.

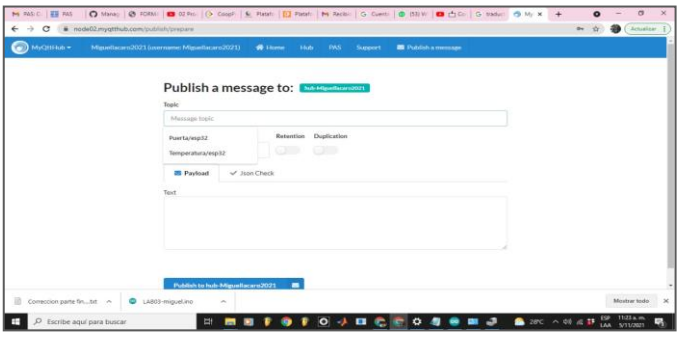

Ilustración 5. Comando para poner un set de temperatura menor al ambiente para encender el ventilador.

- ✓ Humedad: Información de la humedad actual actualizada cada 25 segundos.

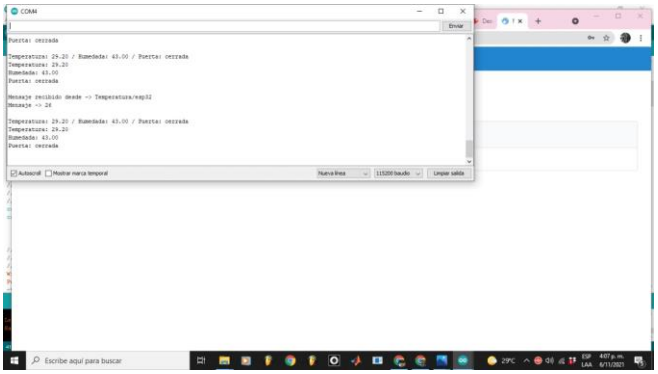

Ilustración 6. Información actual.

- ✓ Puerta: Información del estado de puerta cerrada o abierta, cuando se publique desde el bróker a este topic (Cerrar o Abrir) se debe accionar el motor (Servomotor) y simular (Maqueta) la apertura o cerrar la puerta.

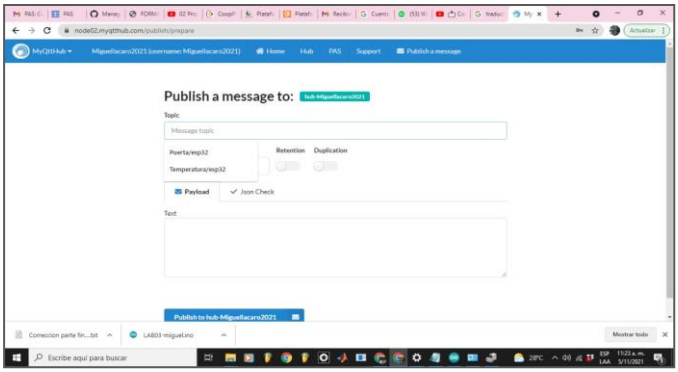

Ilustración 7. Comando para abrir la puerta.

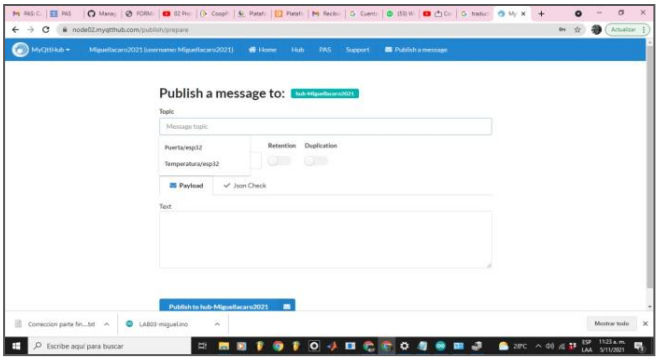

Ilustración 8. Comando para cerrar la puerta.

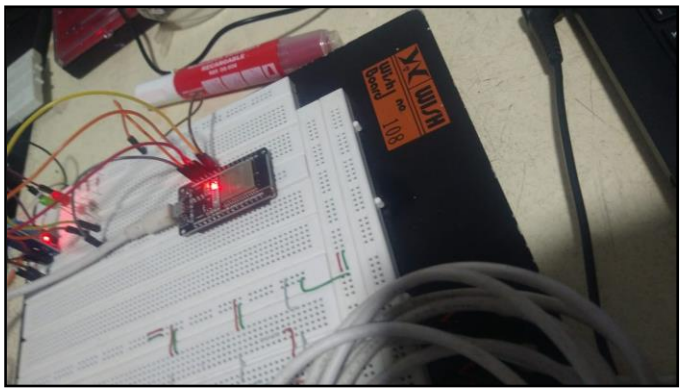

Ilustración 9. Montaje en Protoboard Encendido.

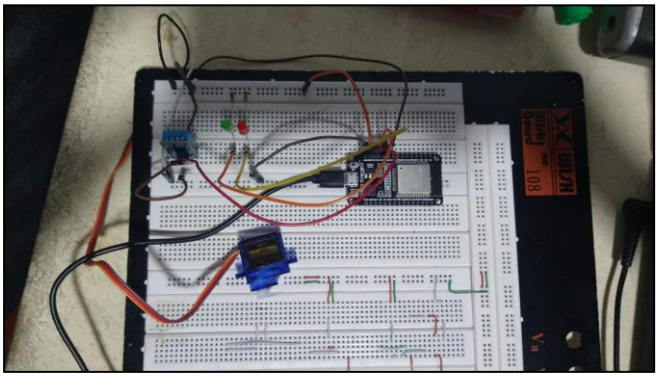

Ilustración 10. Montaje en Protoboarb Off.

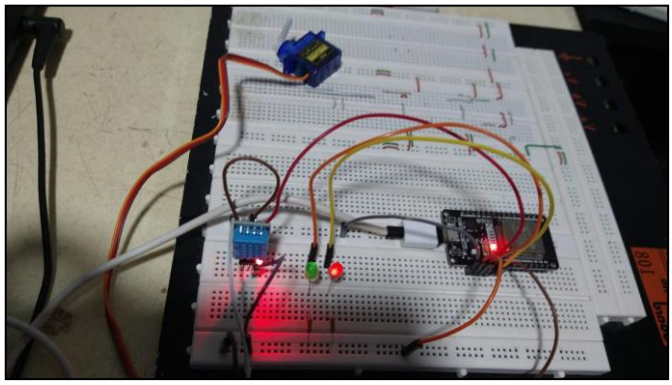

Ilustración 11. Montaje completo.

TABLA INDICACIONES LEDS CIRCUITO

Led Verde ON	Abrió puerta
Led Rojo ON	Alta Temperatura
Led Verde OFF	Puerta Cerrada
Led Rojo ON	Temperatura por debajo del seting

4. Se debe implementar una librería de correo (Email) para notificar en dos casos (utilizar de guía el siguiente tutorial https://www.youtube.com/watch?v=cRw3DtHVPAc&t=505s):
   - ✓ Enviar un correo cuando se realice la acción de abrir la puerta (se envia un correo a una cuenta personal, se recomienda cambiar la clave temporalmente o crear un correo de prueba).
   - ✓ Enviar un correo cuando se realice la acción de encendido de ventilador por temperatura (corre de prueba electivaiotunisangil@gmail.com contraseña: ElectivaIOT20).

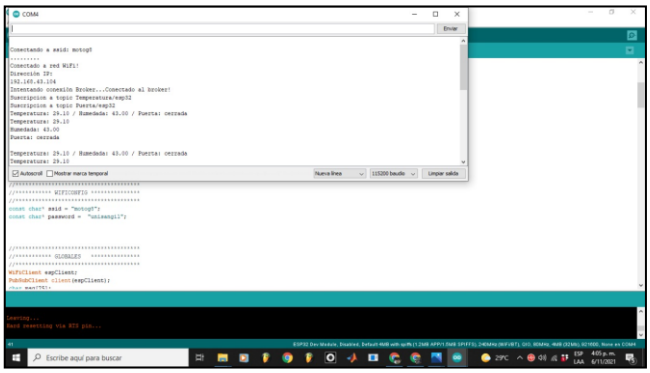

Ilustración 12. Monitor serie conectado al bróker el esp.

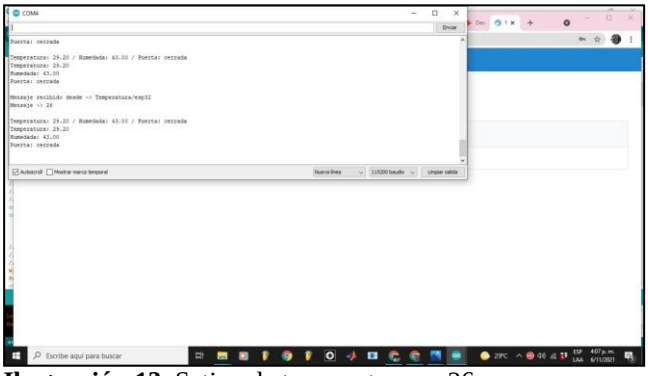

Ilustración 13. Seting de temperature en 26.

Ilustración 14. Confirmación de correo.

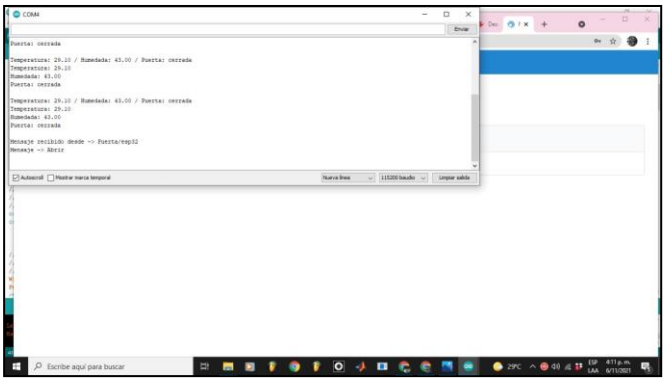

Ilustración 15. Puerta Abierta.

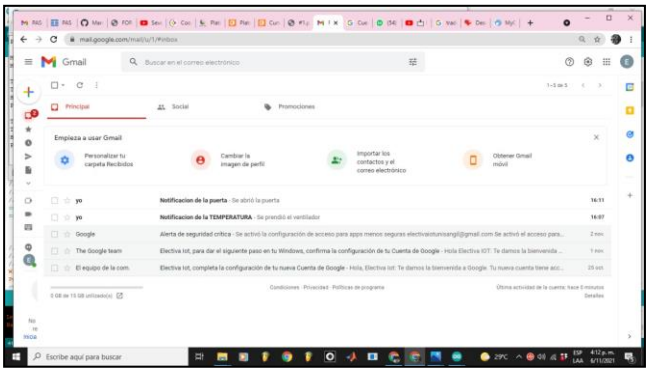

Ilustración 16. Notificación correo Puerta.

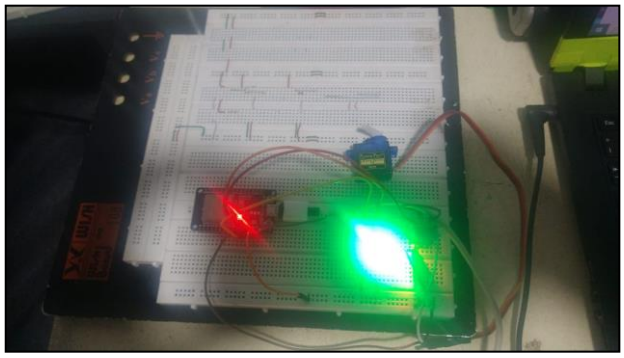

Ilustración 17. Led Verde Puerta abierta.

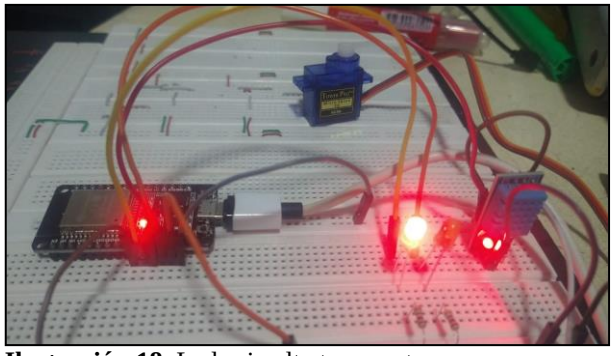

Ilustración 18. Led rojo alta temperatura.

5. Crear un algoritmo para que cuando llegue una publicación al tópico de temperatura al cliente (microntrolador), se configure la temperatura a controlar, dentro del micro se debe crear una función que si detecta que se supero la temperatura programa se debe prender el ventilador con ayuda del transistor o un relé y se debe notificar por correo que se encendio el sistema de control.

6. Crear un algoritmo para que cuando llegue una publicación al topico de puerta al cliente (microcontrolador), dentro del micro se debe crear una función que si se envía Abrir se accione el motor o servo motor y abra la puerta, si se envía Cerrar se efectúa la acción de cerrar la puerta, se debe notificar por correo que se abrió la puerta.

B. CÓDIGO

```
#include <ESP32Servo.h>
#include <ESP32_MailClient.h>
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"


SMTPData datosSMTP;
#define DHTPIN 13
#define DHTTYPE DHT11
DHT Sensor(DHTPIN, DHTTYPE);
Servo motor;
#define servopin 27
int Led_verde =12;
int Led_rojo =14;



//*************************************
//********** MQTT CONFIG **************
//*************************************
const char *mqtt_server = "node02.myqtthub.com";
const int mqtt_port = 1883;
const char *mqtt_user = "esp_32";
const char *mqtt_pass = "esp_32";
const char *root_topic_subscribe1 = "Temperatura/esp32";
const char *root_topic_subscribe2 = "Puerta/esp32";
const char *root_topic_publish = "Topic_raiz/public_esp32";
const char *root_topic_publish1 =
"Temperatura/public_esp32";
const char *root_topic_publish2 = "Humedad/public_esp32";
const char *root_topic_publish3 = "Puerta/public_esp32";


//*************************************
//********** WIFICONFIG ***************
//*************************************
const char* ssid = "Tenda_8973A8";
const char* password =  "KeTjeJD9";



//*************************************
//********** GLOBALES   ***************
```

```
//**************************************
WiFiClient espClient;
PubSubClient client(espClient);
char msg[75];
char msg1[25];
char msg2[25];
char msg3[25];
long alta_t=35;
String puerta = "Cerrar";
String estado = "cerrada";
String tema="";
float t;


//*************************
//** F U N C I O N E S ***
//*************************
void callback(char* topic, byte* payload, unsigned int length);
void reconnect();
void setup_wifi();

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  Sensor.begin();
  pinMode(Led_verde, OUTPUT);
  pinMode(Led_rojo, OUTPUT);
  motor.attach(servopin);
  Cerrar();
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  if (client.connected()){
    float h= Sensor.readHumidity();
    float t= Sensor.readTemperature();
    if (t>alta_t) {
      digitalWrite(14, HIGH);
    }else {
      digitalWrite(14, LOW);
    }

    if (puerta=="Cerrar"){
      estado="cerrada";
      digitalWrite(12, LOW);
    }else if (puerta=="Abrir"){
      estado="abierta";
      digitalWrite(12, HIGH);
    }

    String str = "Temperatura: " + String(t) + " / Humedada: " +
String(h) + " / Puerta: " + (estado);
    String str1 = "Temperatura: " + String(t);
    String str2 = "Humedada: " + String(h);
    String str3 = "Puerta: " + (estado);
    str.toCharArray(msg,75);
    str1.toCharArray(msg1,25);
    str2.toCharArray(msg2,25);
    str3.toCharArray(msg3,25);
    client.publish(root_topic_publish,msg);
    client.publish(root_topic_publish1,msg1);
    client.publish(root_topic_publish2,msg2);
    client.publish(root_topic_publish3,msg3);

    Serial.println(msg);
    Serial.println(msg1);
    Serial.println(msg2);
    Serial.println(msg3);
    Serial.println("");
    delay(10000);
  }
  client.loop();
}


//*******************************
//***    CONEXION WIFI      ***
//*******************************
void setup_wifi(){
  delay(5000);
  // Nos conectamos a nuestra red Wifi
  Serial.println();
  Serial.print("Conectando a ssid: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

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

  Serial.println("");
  Serial.println("Conectado a red WiFi!");
  Serial.println("Dirección IP: ");
  Serial.println(WiFi.localIP());
}


//*******************************
//***    CONEXION MQTT      ***
//*******************************

void reconnect() {

  while (!client.connected()) {
    Serial.print("Intentando conexión Broker...");
    // Creamos un cliente ID
    String clientId = "Modulo_esp32";

    // Intentamos conectar
    if (client.connect(clientId.c_str(),mqtt_user,mqtt_pass)) {
      Serial.println("Conectado al broker!");
      // Nos suscribimos
      if(client.subscribe(root_topic_subscribe1)){
```

```
      Serial.println("Suscripcion a topic "+
String(root_topic_subscribe1));
    }else{
      Serial.println("fallo Suscripciión a topic "+
String(root_topic_subscribe1));
    }
    if(client.subscribe(root_topic_subscribe2)){
      Serial.println("Suscripcion a topic "+
String(root_topic_subscribe2));
    }else{
      Serial.println("fallo Suscripciión a topic "+
String(root_topic_subscribe2));
    }
   } else {
    Serial.print("falló conexión broker:( con error -> ");
    Serial.print(client.state());
    Serial.println(" Intentamos de nuevo en 5 segundos");
    delay(5000);
   }
  }
}


//*****************************
//***      CALLBACK       ***
//*****************************

void callback(char* topic, byte* payload, unsigned int
length){
  String incoming = "";
  Serial.print("Mensaje recibido desde -> ");
  Serial.print(topic);
  Serial.println("");
  for (int i = 0; i < length; i++) {
    incoming += (char)payload[i];
  }
  incoming.trim();
  tema=topic;
  Serial.println("Mensaje -> " + incoming);
  Serial.println("");

  if (tema=="Puerta/esp32"){
    if (incoming=="Cerrar"){
      puerta=incoming;
      Cerrar();
    }else if (incoming=="Abrir"){
      puerta=incoming;
      Abrir();

datosSMTP.setLogin("smtp.gmail.com",465,"electivaiotunisa
ngil@gmail.com","ElectivaIOT20");

datosSMTP.setSender("MODULO_ESP32","electivaiotunisan
gil@gmail.com");
      datosSMTP.setPriority("High");
      datosSMTP.setSubject("Notificacion de la puerta");
      datosSMTP.setMessage("Se abrió la puerta",false);

datosSMTP.addRecipient("electivaiotunisangil@gmail.com");
      if (!MailClient.sendMail(datosSMTP)){
        Serial.println("Error enciando correo, " +
MailClient.smtpErrorReason());
      }
      datosSMTP.empty();
    }
  }

  if (tema=="Temperatura/esp32"){
    alta_t=incoming.toInt();
    t= Sensor.readTemperature();
    if (t>alta_t){

datosSMTP.setLogin("smtp.gmail.com",465,"electivaiotunisa
ngil@gmail.com","ElectivaIOT20");

datosSMTP.setSender("MODULO_ESP32","electivaiotunisan
gil@gmail.com");
      datosSMTP.setPriority("High");
      datosSMTP.setSubject("Notificacion de la
TEMPERATURA");
      datosSMTP.setMessage("Se prendió el ventilador",false);

datosSMTP.addRecipient("electivaiotunisangil@gmail.com");
      if (!MailClient.sendMail(datosSMTP)){
        Serial.println("Error enciando correo, " +
MailClient.smtpErrorReason());
      }
      datosSMTP.empty();
    }
  }
}


void Abrir(){
  motor.write(0);
  delay(500);
}
void Cerrar(){
  motor.write(130);
  delay(500);
}
```

7. VÍDEO DE EXPLICACIÓN

https://www.youtube.com/watch?v=V3hJ91NT6Tg

IX. CONCLUSIÓN

- Las conexiones del circuito deben de ser estables ya que una mala conexión nos estropeaba la entrada y salida de los comandos.
- Que como estamos trabajando en una plataforma de Internet la conexión a internet debe de ser estable ya que de lo contrario se presentarán fallas.
- Se presentaron inconvenientes con la plataforma en

una ocasión ya que no estaba funcionando y no permitía el envío de comandos generando error.

- Se aplicaron los conocimientos aprendidos en clases y se cumplió con la práctica.

REFERENCES

[1] MQTT: The Standard for IoT Messaging. https://mqtt.org/

[2] 17 de abril, 2019, Luis Llamas; ingeniería, informática y diseño. https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[3] https://descubrearduino.com/mqtt-que-es-como-se-puede-usar-y-como-funciona/

[4] https://admiralmarkets.com/es/education/articles/forex-basics/broker-que-es

[5] Guía Laboratorio N°2.

Autores

William Cachay
Paula Castellanos
Miguel Laverde