

EVIDENCIAS PRACTICA 1

PRIMER MÓDULO MQTT

(PARTE 2)

Grupo:

Integrantes: David Flores y Diego García.

Fecha: 01/10/2024

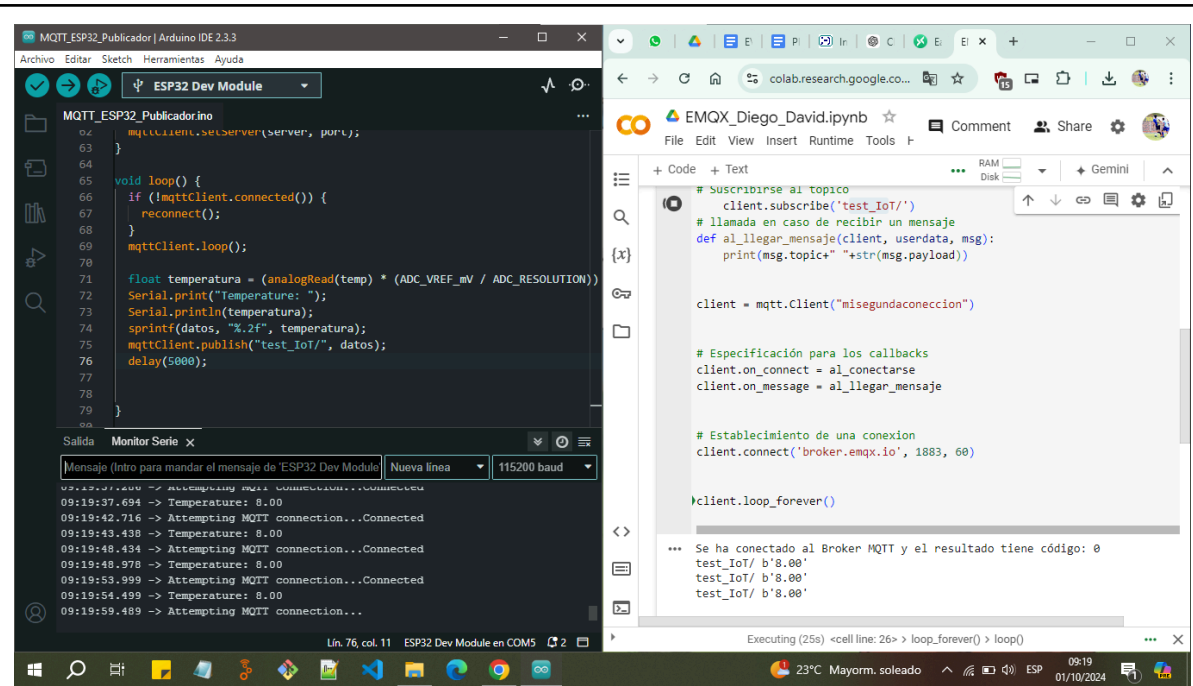
Actividad 4: ESP32 como publicador

The screenshot displays the Arduino IDE environment with a sketch titled 'MQTT_ESP32_Publicador.ino'. The code is written in C++ and implements an MQTT publisher for an ESP32. It includes the necessary libraries, sets the server and port, and defines a loop function that reads a temperature sensor, prints the value to the serial monitor, and publishes it to the 'test_IoT/' topic. The serial monitor shows the device successfully connecting to the MQTT broker and publishing temperature readings of 8.00.

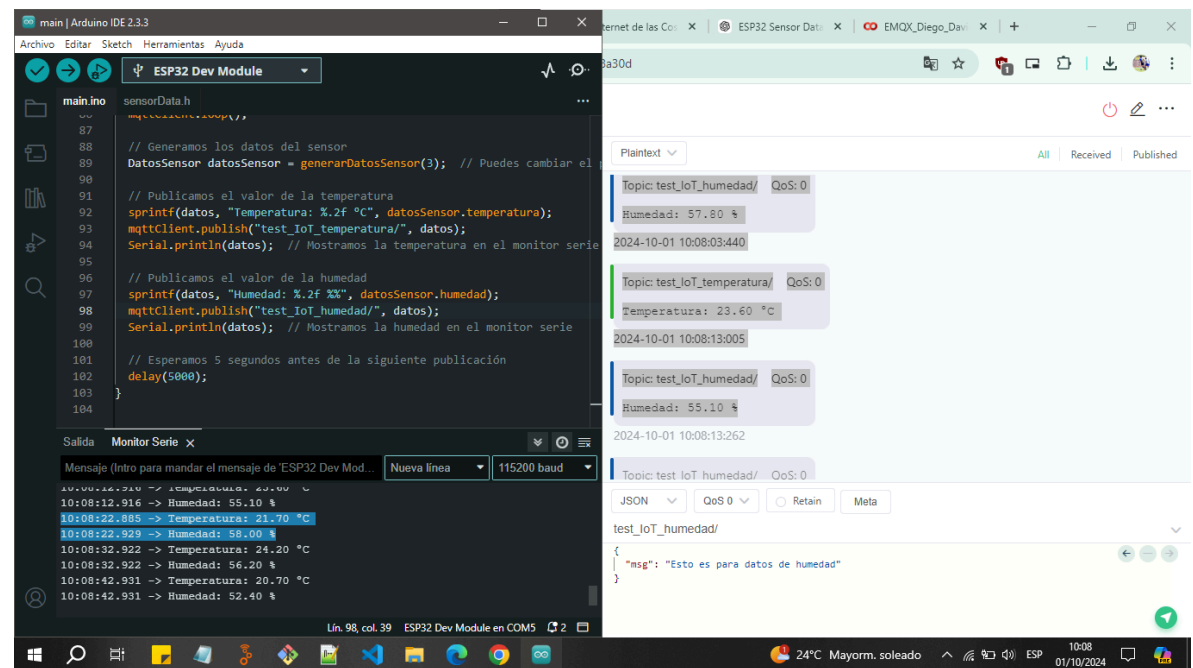
```
MQTT_ESP32_Publicador.ino
62 mqttClient.setServer(server, port);
63 }
64
65 void loop() {
66   if (!mqttClient.connected()) {
67     reconnect();
68   }
69   mqttClient.loop();
70
71   float temperatura = (analogRead(temp) * (ADC_VREF_mV / ADC_RESOLUTION));
72   Serial.print("Temperature: ");
73   Serial.println(temperatura);
74   sprintf(datos, "%.2f", temperatura);
75   mqttClient.publish("test_IoT/", datos);
76   delay(5000);
77 }
78
79 }
```

The MQTT client interface on the right shows the 'test_IoT/' topic with three received messages, all containing the value '8.00'. The interface also shows the 'test_IoT/' topic selected in the dropdown menu, and the 'msg' field contains the JSON payload: {"msg": "Hola, soy Diego"}.

Actividad 4: ESP32 y Python



Actividad de cierre: Evidencias etapa 1



Actividad de cierre: Evidencias etapa 2

EMQX_Diego_David.ipynb · Colab

colab.research.google.com

EMQX_Diego_David.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
client.on_connect = al_conectarse
client.on_message = al_llegar_mensaje

# Establecimiento de una conexion
client.connect('broker.emqx.io', 1883, 60)

client.loop_forever()
```

... Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
test_IoT_temperatura/ b'Temperatura: 23.60 \xc2\xbcC'
test_IoT_humedad/ b'Humedad: 50.40 %'
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
test_IoT_temperatura/ b'Temperatura: 23.70 \xc2\xbcC'
test_IoT_humedad/ b'Humedad: 51.50 %'
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
test_IoT_temperatura/ b'Temperatura: 23.70 \xc2\xbcC'
test_IoT_humedad/ b'Humedad: 51.50 %'
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
test_IoT_temperatura/ b'Temperatura: 23.70 \xc2\xbcC'
test_IoT_humedad/ b'Humedad: 51.50 %'
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
Se ha conectado al Broker MQTT y el resultado tiene código: 0
test_IoT_temperatura/ b'Temperatura: 23.70 \xc2\xbcC'
test_IoT_humedad/ b'Humedad: 51.50 %'

Executing (38s) <cell line: 27> > loop_forever() > _reconnect_wait()

Internet de las Cosas: EVI

ESP32 Sensor Data Integ

ba30d

Plaintext

2024-10-01 10:10:33443

Topic: test_IoT_humedad/ QoS: 0

Humedad: 51.50 %

2024-10-01 10:10:13464

Topic: test_IoT_humedad/ QoS: 0

Humedad: 52.70 %

2024-10-01 10:10:23333

Topic: test_IoT_humedad/ QoS: 0

Humedad: 52.70 %

2024-10-01 10:10:33362

JSON QoS 0 Retain Meta

test_IoT_humedad/

```
{
  "msg": "Esto es para datos de humedad"
}
```

24°C Mayorm. soleado 10:10 01/10/2024

Actividad de cierre: Evidencias etapa 3

main | Arduino IDE 2.3.3

Archivo Editar Sketch Herramientas Ayuda

ESP32 Dev Module

main.ino sensorData.h

```
41 int led = 2;
42 char datos[50]; // Ajustamos el tamaño para incluir ambos datos
43
44 void wifi() {
45   Serial.print("Conectando a ");
46   Serial.println(ssid);
47 }
```

Salida Monitor Serie x

Mensaje (Intro para mandar el mensaje de 'ESP32 Dev Module') Nueva línea 115200 baud

16:40:59.959 -> Humedad: 53.90 %
16:41:09.959 -> Temperatura: 24.80 °C
16:41:10.077 -> Humedad: 55.90 %
16:41:19.968 -> Temperatura: 22.60 °C
16:41:20.266 -> Humedad: 57.00 %
16:41:29.992 -> Temperatura: 24.80 °C
16:41:29.992 -> Humedad: 57.90 %
16:41:39.962 -> Temperatura: 23.10 °C
16:41:40.082 -> Humedad: 56.80 %
16:41:50.295 -> Temperatura: 22.60 °C
16:41:50.295 -> Humedad: 58.20 %
16:42:00.252 -> Temperatura: 24.80 °C
16:42:00.253 -> Humedad: 54.40 %
16:42:10.025 -> Temperatura: 21.50 °C
16:42:10.025 -> Humedad: 55.30 %
16:42:19.988 -> Temperatura: 24.80 °C
16:42:19.988 -> Humedad: 51.30 %
16:42:30.048 -> Temperatura: 23.40 °C
16:42:30.048 -> Humedad: 51.90 %
16:42:39.996 -> Temperatura: 22.70 °C
16:42:39.996 -> Humedad: 53.70 %

Lín. 29, col. 24 ESP32 Dev Module en COM5 2

temperatura.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

Temperatura: 21.30 °C
Temperatura: 23.80 °C
Temperatura: 24.80 °C
Temperatura: 22.40 °C
Temperatura: 21.30 °C
Temperatura: 21.60 °C
Temperatura: 24.80 °C
Temperatura: 22.60 °C
Temperatura: 24.80 °C
Temperatura: 23.10 °C
Temperatura: 22.60 °C
Temperatura: 24.80 °C

Linea 1, columna 1 100% Windows (CRLF) ANSI

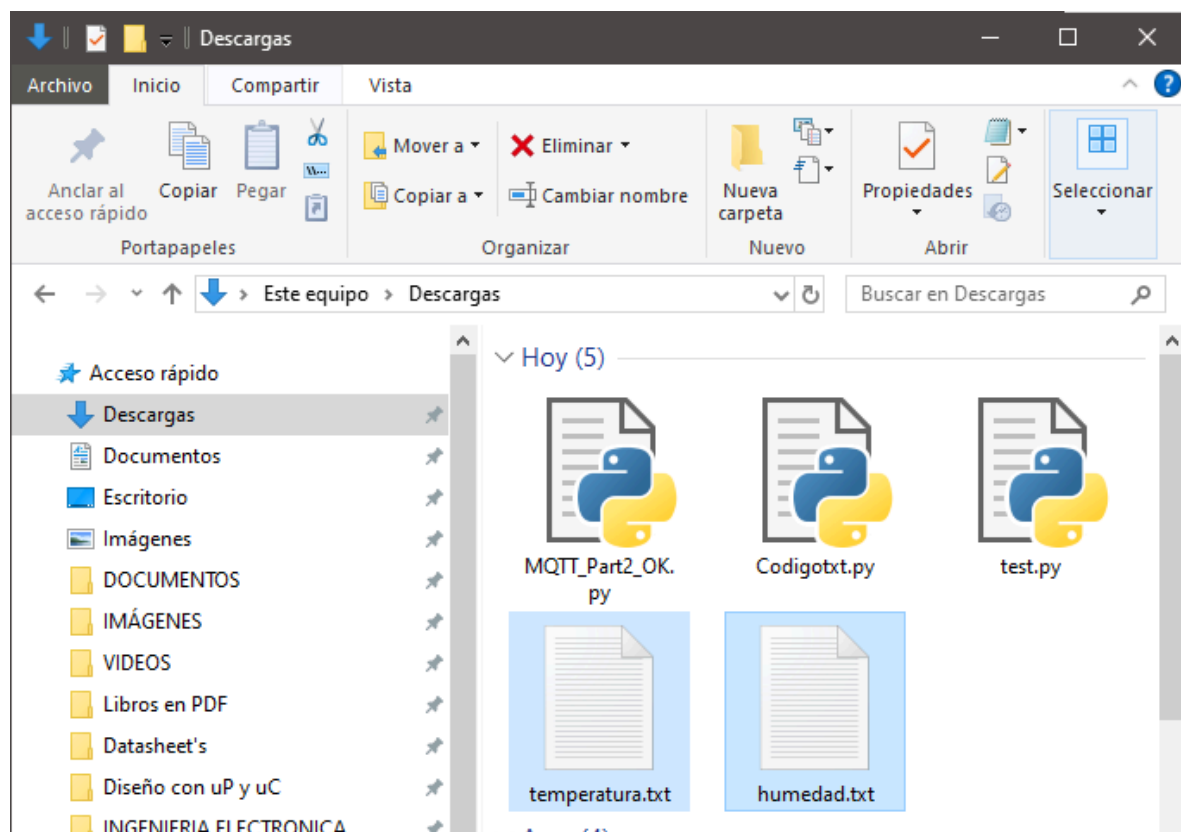
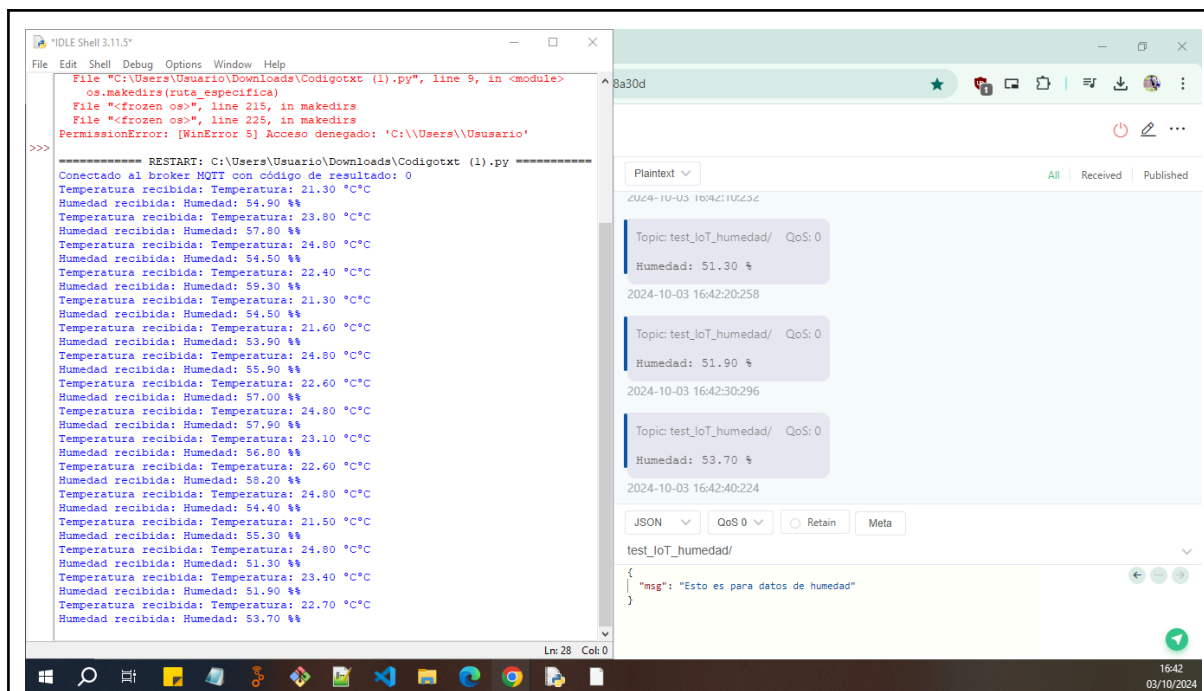
humedad.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

Humedad: 54.90 %
Humedad: 57.80 %
Humedad: 54.50 %
Humedad: 59.30 %
Humedad: 54.50 %
Humedad: 53.90 %
Humedad: 55.90 %
Humedad: 57.00 %
Humedad: 56.80 %
Humedad: 58.20 %
Humedad: 54.40 %

Linea 1, columna 1 100% Windows (CRLF) UTF-8

27°C Parc. soleado 16:42 03/10/2024



Finalmente el código que se usó fue el siguiente:

```
import paho.mqtt.client as mqtt
import os

# Especificar la ruta completa donde se guardarán los archivos
ruta_especifica = "C:\\Users\\Usuario\\Downloads" # Cambia esta ruta
```

```

por la que necesites

# Verificar si la carpeta existe, si no, crearla
if not os.path.exists(ruta_especifica):
    os.makedirs(ruta_especifica)

# Variables para almacenar las rutas completas de los archivos
temperatura_file = os.path.join(ruta_especifica, "temperatura.txt")
humedad_file = os.path.join(ruta_especifica, "humedad.txt")

# Función que maneja los mensajes recibidos de los tópicos MQTT
def on_message(client, userdata, message):
    topic = message.topic
    payload = message.payload.decode("utf-8")

    # Verificar el tópico y escribir en el archivo correspondiente
    if topic == "test_IoT_temperatura/":
        with open(temperatura_file, "a") as f:
            f.write(f"{payload}\n") # Guardar cada dato en una nueva
línea
            print(f"Temperatura recibida: {payload}°C")

        elif topic == "test_IoT_humedad/":
            with open(humedad_file, "a") as f:
                f.write(f"{payload}\n") # Guardar cada dato en una nueva
línea
                print(f"Humedad recibida: {payload}%")

# Función que se ejecuta cuando la conexión al broker se realiza con
éxito
def on_connect(client, userdata, flags, rc):
    print(f"Conectado al broker MQTT con código de resultado:
"+str(rc))

    # Suscribirse a los tópicos de temperatura y humedad
    client.subscribe("test_IoT_temperatura/")
    client.subscribe("test_IoT_humedad/")

# Configuración del cliente MQTT
broker = "broker.emqx.io"
port = 1883

client = mqtt.Client("PythonSubscriber") # Crear un nuevo cliente
MQTT
client.on_connect = on_connect # Establecer la función de conexión
client.on_message = on_message # Establecer la función para manejar
los mensajes

# Conectarse al broker MQTT
client.connect(broker, port)

# Mantener la conexión activa y esperar los mensajes
client.loop_forever()

```