

# EVIDENCIAS PRÁCTICA FINAL

Integrantes: JOSE DAVID FLOREZ RAMOS - 2174241

DIEGO ANDRES GARCIA DIAZ - 2195533

Fecha: 28/11/2024

## Tarea 1: CREACIÓN DE PUBLICADOR-ESP32

- Configure una ESP32 para que envíe datos de temperatura aleatorios.
- Cree un Topic en EMQX para que reciba los datos a través del protocolo MQTT.

Código cargado en la ESP32:

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

// Configuración WiFi
const char* ssid = "PISO_2"; // Reemplaza con el nombre de
tu red WiFi
const char* password = "Planetaland2"; // Reemplaza con la
contraseña de tu red WiFi

// Configuración del broker MQTT
const char* mqtt_broker = "broker.emqx.io"; // Cambia si usas otro
broker
const int mqtt_port = 1883;
const char* mqtt_topic = "final_IOT"; // Tópico donde se
publicarán los datos

WiFiClient espClient;
PubSubClient client(espClient);

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

  // Conexión a WiFi
  Serial.print("Conectando a WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nWiFi conectado.");

  // Configuración del cliente MQTT
  client.setServer(mqtt_broker, mqtt_port);
  conectarBrokerMQTT();
}

void loop() {
  if (!client.connected()) {
    conectarBrokerMQTT(); // Reconectar si se pierde la conexión al
broker
  }
}
```

```

// Simular y publicar datos del sensor de temperatura
publicarDatos();

// Esperar 5 minutos
delay(20000);
}

void conectarBrokerMQTT() {
    Serial.print("Conectando al broker MQTT...");
    while (!client.connected()) {
        if (client.connect("ESP32_Client")) { // Nombre único para la
ESP32
            Serial.println("\nConectado al broker MQTT.");
        } else {
            Serial.print(".");
            delay(1000);
        }
    }
}

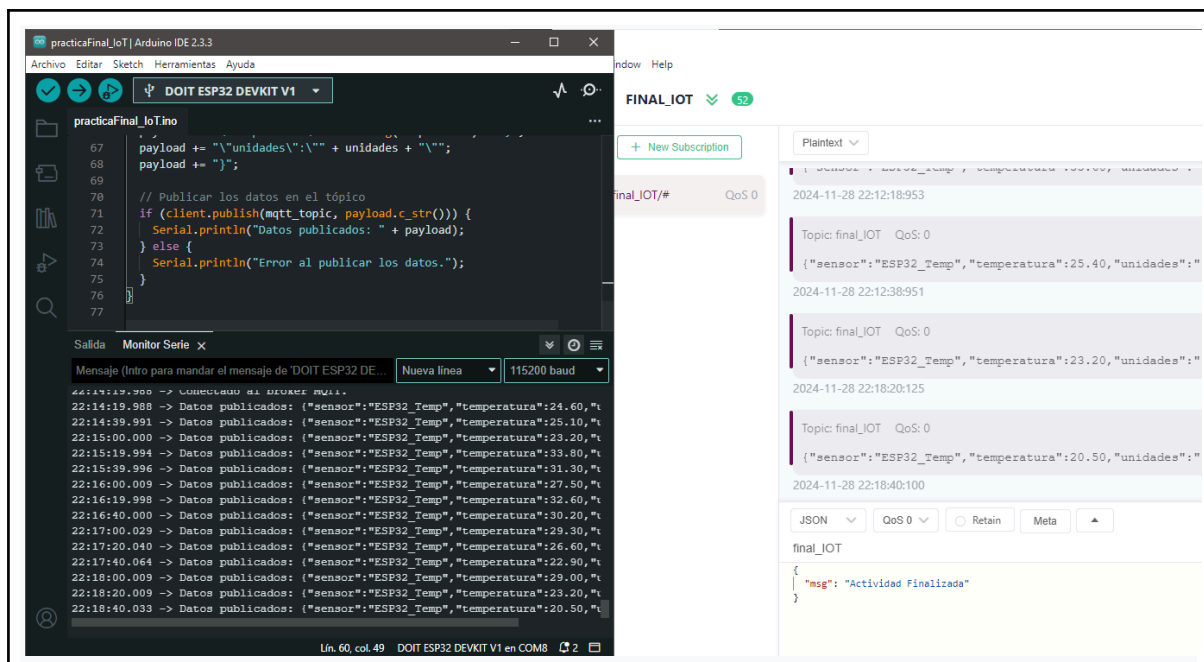
void publicarDatos() {
    // Generar datos simulados del sensor
    String sensor = "ESP32_Temp";
    float temperatura = random(200, 350) / 10.0; // Valores entre 20.0
y 35.0
    String unidades = "°C";

    // Formatear los datos en formato JSON
    String payload = "{";
    payload += "\"sensor\": \"" + sensor + "\", ";
    payload += "\"temperatura\": " + String(temperatura) + ", ";
    payload += "\"unidades\": \"" + unidades + "\"";
    payload += "}";

    // Publicar los datos en el tópico
    if (client.publish(mqtt_topic, payload.c_str())) {
        Serial.println("Datos publicados: " + payload);
    } else {
        Serial.println("Error al publicar los datos.");
    }
}

```

Acá se observan los datos en EMQX:



## Tarea 2: Configuración Web Service

Desarrolla un Webservice en Google Apps Script que permita gestionar un registro de temperaturas. El script debe incluir las funciones para:

- Obtener todas las temperaturas (GET).
- Registrar una nueva temperatura (POST).
- El webservice debe mandar los datos a una hoja de cálculo de google sheet previamente creada y configurada para la recepción de datos.

### URL Google Sheets:

[https://docs.google.com/spreadsheets/d/1z5piyHX22xoT\\_f2SywR\\_ajxuYoA2CUd5GzHCKeaQhXo/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1z5piyHX22xoT_f2SywR_ajxuYoA2CUd5GzHCKeaQhXo/edit?usp=sharing)

Se muestra el código usado en Google Apps Script para realizar las solicitudes GET y POST:

```
// ID del Google Sheets
const SHEET_ID = "1z5piyHX22xoT_f2SywR_ajxuYoA2CUd5GzHCKeaQhXo";
const SHEET_NAME = "Hoja 1";

//
//-----

// Función para manejar las solicitudes GET (OBTIENE EL ÚLTIMO DATO REGISTRADO EN EL GOOGLE SHEETS)
// function doGet(e) {
//   try {
//     // Obtener el último registro válido de la hoja de cálculo
//     const ultimoRegistro = obtenerUltimoRegistro();

//     // Enviar los datos en formato JSON
//     return
ContentService.createTextOutput(JSON.stringify(ultimoRegistro))
```

```

//                                     .setMimeType(ContentService.MimeType.JSON);
//   } catch (error) {
//     // En caso de error, devolver el mensaje en JSON
//     const errorMessage = {
//       status: "error",
//       message: error.message
//     };
//   }
//                                     return
ContentService.createTextOutput(JSON.stringify(errorMessage))
//                                     .setMimeType(ContentService.MimeType.JSON);
//   }
// }

// // Función para obtener el último registro válido de la hoja de
// cálculo
// function obtenerUltimoRegistro() {
//   const ss = SpreadsheetApp.openById(SHEET_ID);
//   const sheet = ss.getSheetByName(SHEET_NAME);

//   if (!sheet) {
//     throw new Error("No se encontró la hoja especificada.");
//   }

//   const lastRow = sheet.getLastRow();

//   if (lastRow < 2) {
//     // Si no hay datos más allá del encabezado
//     throw new Error("No hay datos en la hoja.");
//   }

//   // Leer los encabezados y el último registro válido
//   const headers = sheet.getRange(1, 1, 1,
sheet.getLastColumn()).getValues()[0];
//   const lastData = sheet.getRange(lastRow, 1, 1,
sheet.getLastColumn()).getValues()[0];

//   // Validar que el último registro tenga datos válidos
//   if (lastData.every(cell => cell === "")) {
//     throw new Error("El último registro está vacío o es inválido.");
//   }

//   // Crear un objeto JSON basado en los encabezados y el último
// registro
//   const data = {};
//   headers.forEach((header, index) => {
//     data[header] = lastData[index];
//   });

//   return data;
// }

//

```

```

-----
-----

// Función para manejar las solicitudes GET (OBTIENE TODOS LOS DATOS
REGISTRADOS EN EL GOOGLE SHEETS)
function doGet() {
  const ss = SpreadsheetApp.openById(SHEET_ID); // Reemplaza SHEET_ID con
el ID de tu hoja de cálculo
  const sheet = ss.getSheetByName(SHEET_NAME); // Reemplaza SHEET_NAME
con el nombre de tu hoja de cálculo

  if (!sheet) {
    return ContentService.createTextOutput(
      JSON.stringify({ error: "No se encontró la hoja especificada." })
    ).setMimeType(ContentService.MimeType.JSON);
  }

  const data = sheet.getDataRange().getValues(); // Obtiene todos los
datos de la hoja
  if (data.length <= 1) {
    // Verifica si solo hay encabezados o está vacía
    return ContentService.createTextOutput(
      JSON.stringify({ error: "No hay datos en la hoja de cálculo." })
    ).setMimeType(ContentService.MimeType.JSON);
  }

  // Extrae los encabezados de la primera fila
  const headers = data[0];

  // Convierte las filas restantes en objetos basados en los encabezados
  const jsonData = data.slice(1).map(row => {
    const rowData = {};
    headers.forEach((header, index) => {
      rowData[header] = row[index];
    });
    return rowData;
  });

  // Devuelve los datos como JSON
  return ContentService.createTextOutput(JSON.stringify(jsonData))
    .setMimeType(ContentService.MimeType.JSON);
}

//
-----
-----

// Función para manejar las solicitudes POST (REGISTRA DATOS EN EL GOOGLE
SHEETS)
function doPost(e) {
  const sheet =
SpreadsheetApp.openById(SHEET_ID).getSheetByName(SHEET_NAME);

```

```

if (!e.postData || !e.postData.contents) {
    return ContentService.createTextOutput("No se enviaron datos.")
        .setMimeType(ContentService.MimeType.TEXT);
}

try {
    // Procesar el contenido del cuerpo en JSON
    const data = JSON.parse(e.postData.contents);

    // Extraer los valores del JSON
    const sensor = data.sensor || "ESP32_Temp";
    const temperatura = data.temperatura || 0; // Default a 0 si no se
envía
    const unidades = data.unidades || "°C";

    // Agregar datos a la hoja de Google Sheets
    sheet.appendRow([sensor, temperatura, unidades]);

    // Respuesta de confirmación
    return ContentService.createTextOutput("Datos guardados
correctamente.")
        .setMimeType(ContentService.MimeType.TEXT);

} catch (error) {
    // Manejo de errores
    return ContentService.createTextOutput("Error: " + error.message)
        .setMimeType(ContentService.MimeType.TEXT);
}
}

```

### Tarea 3: Configuración en Node-RED

Crea un flujo en Node-RED que interactúe con el web service que has desarrollado. El flujo debe:

- Obtener la lista de temperaturas y mostrarla en el panel de debug.
- Permitir registrar una nueva temperatura desde un nodo de entrada.
- Instrucciones: Describe el flujo, los nodos utilizados y cómo se conectan entre sí.

Incluye capturas de pantalla del flujo.

#### Obtener datos (GET):

Primero se obtienen el último dato cargado en el Google Sheets:

The Node-RED interface shows a workflow with a POST node, a debug node, and a GET node. The debug console shows a successful POST request and a GET response. A Google Sheets spreadsheet is visible on the right, showing temperature data for ESP32\_Temp and NodeRED\_Test.

	A	B	C
149	ESP32_Temp	28.1	°C
150	ESP32_Temp	27.5	°C
151	ESP32_Temp	26.2	°C
152	ESP32_Temp	31.5	°C
153	ESP32_Temp	34.6	°C
154	NodeRED_Test	20.8	°C
155	ESP32_Temp	28.1	°C
156	NodeRED_Test	20.8	°C
157	NodeRED_Test	20.8	°C
158	NodeRED_Test	20.8	°C
159	ESP32_Temp	22.1	°C
160	ESP32_Temp	25	°C
161	ESP32_Temp	33	°C
162	ESP32_Temp	32.4	°C
163	ESP32_Temp	27	°C
164	ESP32_Temp	25.8	°C
165			
166			
167			
168			

Ahora se obtienen los todos los datos que se han cargado en el Google Sheets:

The Node-RED interface shows a workflow with a POST node, a debug node, and a GET node. The debug console shows a successful POST request and a GET response. A Google Sheets spreadsheet is visible on the right, showing temperature data for ESP32\_Temp and NodeRED\_Test.

	A	B	C
1	Sensor	Temperatura	Unidad
2	Desconocido	0	N/A
3	ESP32_Temp	0	°C
4	ESP32_Temp	0	°C
5	ESP32_Temp	0	°C
6	ESP32_Temp		°C
7	ESP32_Temp		°C
8	ESP32_Temp		°C
9	ESP32_Temp		°C
10	ESP32_Temp		°C
11	ESP32_Temp		°C
12	ESP32_Temp		°C
13	ESP32_Temp		°C
14	ESP32_Temp		°C
15	ESP32_Temp		°C
16	ESP32_Temp		°C
17	ESP32_Temp		°C
18	ESP32_Temp		°C
19	ESP32_Temp		°C
20	ESP32_Temp		°C

En conclusión, se puede obtener el último dato, como todos los datos registrados en el Google Sheets, esto depende del código proporcionado para la función **doGet()** en Google Apps Script.

### Registro de datos (POST):

Usando el bloque inject en Node-RED se registra un dato y también se registran los datos provenientes de la ESP32, los datos seleccionados en el Google Sheets, corresponden a los datos registrados gracias al bloque inject en Node-RED (**NodeRED\_Test**) y lo que tiene el nombre de **ESP32\_Temp** corresponde a los datos provenientes de la ESP32:

The screenshot shows the Node-RED interface with a flow that includes a POST node, a GET node, and a debug node. The debug console displays the output of the POST request, including the payload and headers. The right panel shows a Google Sheet with temperature data.

	A	B	C
163	ESP32_Temp	27	°C
164	ESP32_Temp	25.8	°C
165	ESP32_Temp	34.9	°C
166	ESP32_Temp	28.9	°C
167	ESP32_Temp	28.9	°C
168	ESP32_Temp	32.6	°C
169	ESP32_Temp	32.6	°C
170	ESP32_Temp	21.1	°C
171	ESP32_Temp	33.1	°C
172	NodeRED_Test	20.8	°C
173	NodeRED_Test	20.8	°C
174	NodeRED_Test	20.8	°C
175	NodeRED_Test	20.8	°C
176	ESP32_Temp	31.9	°C
177			
178			
179			
180			
181			
182			

#### Tarea 4: Simulación de Solicitudes

Simula una serie de solicitudes (GET, POST) desde Node-RED y documenta las respuestas del web service.

- Instrucciones: Proporciona ejemplos de las solicitudes realizadas y las respuestas recibidas. Explica cualquier error que haya ocurrido y cómo se resolvió.

#### Solicitud POST:

Se observa en la tabla de color amarillo los datos registrados previamente, ahora, en color azul se observan las nuevas pruebas, los datos registrados provenientes de la ESP32, se registran bajo el nombre de **ESP32\_Temp**, ahora si quiero registrar un dato "manualmente", uso el bloque **inject** y se registrará un dato en específico que se puede reconocer bajo el nombre de **NodeRED\_Test**, de este modo se logra evidenciar que la SOLICITUD POST se realizó exitosamente:

The screenshot shows the Node-RED interface with a flow that includes a POST node, a GET node, and a debug node. The debug console displays the output of the POST request, including the payload and headers. The right panel shows a Google Sheet with temperature data.

	A	B	C
208	ESP32_Temp	31.3	°C
209	ESP32_Temp	29.7	°C
210	ESP32_Temp	29.7	°C
211	ESP32_Temp	23.4	°C
212	NodeRED_Test	20.8	°C
213	ESP32_Temp	22.9	°C
214	NodeRED_Test	20.8	°C
215	NodeRED_Test	20.8	°C
216	NodeRED_Test	20.8	°C
217	ESP32_Temp	24.3	°C
218			
219			
220			
221			
222			
223			
224			
225			
226			
227			

#### Solicitud GET:

Primero se observa gracias a la SOLICITUD GET, todos los datos registrados en el Google Sheets:



Node-RED interface showing a flow for ESP32 temperature data. The flow includes a 'Mensaje Prueba' input, a 'POST' node, a 'json' node, a 'debug POST' node, a 'Suscriptor ESP32' node, a 'debug ESP32' node, a 'GET' node, a 'debug GET' node, and a 'json' node. The debug console shows a POST request with a topic 'final\_IOT' and a payload containing HTML. The debug console also shows a GET request with a message object containing a \_msgid and a payload array of sensor data. The right panel shows a table with columns A, B, and C, containing temperature data for ESP32\_Temp.

	A	B	C
218	ESP32_Temp	26.1	°C
219	ESP32_Temp	20.2	°C
220	ESP32_Temp	25.8	°C
221	ESP32_Temp	28.2	°C
222	ESP32_Temp	25.6	°C
223	ESP32_Temp	31.8	°C
224	ESP32_Temp	29.4	°C
225	ESP32_Temp	24.2	°C
226	ESP32_Temp	23.5	°C
227	ESP32_Temp	33.2	°C
228	ESP32_Temp	34.6	°C
229	ESP32_Temp	29.8	°C
230			
231			
232			
233			
234			
235			
236			
237			

Suma: 29.7

La URL de la implementación en Google Apps Script para este caso:

[https://script.googleusercontent.com/macros/echo?user\\_content\\_key=VAZzRTedWe2qGzcu70MoT-oA-5n7gw-KKu-vf4R\\_cQDg4WEkmFjqi8rK6YSFR0YS4nCimCtSIC6LNLGA9-4P5bZjiyq2DicGm5\\_BxDIH2jW0nuo2oDemN9CCS2h10ox\\_1xSncGQajx\\_ryfhECjZEnCsEsHi47mtfYygbDnyhM8uxqRc7qgPLAf5jcFiD3LpY5vD7sZ5E9obR077MI2oXI\\_oMB4oPIAaLxSAQ3BXxDh4hlv\\_diietg9z9Jw9Md8uu&lib=MLuf14IDG-LB1GnxsiS1mSVYJe0-7KKK](https://script.googleusercontent.com/macros/echo?user_content_key=VAZzRTedWe2qGzcu70MoT-oA-5n7gw-KKu-vf4R_cQDg4WEkmFjqi8rK6YSFR0YS4nCimCtSIC6LNLGA9-4P5bZjiyq2DicGm5_BxDIH2jW0nuo2oDemN9CCS2h10ox_1xSncGQajx_ryfhECjZEnCsEsHi47mtfYygbDnyhM8uxqRc7qgPLAf5jcFiD3LpY5vD7sZ5E9obR077MI2oXI_oMB4oPIAaLxSAQ3BXxDh4hlv_diietg9z9Jw9Md8uu&lib=MLuf14IDG-LB1GnxsiS1mSVYJe0-7KKK)

```

1 [
2   {
3     "Sensor": "Desconocido",
4     "Temperatura": 0,
5     "Unidad": "N/A",
6   },
7   {
8     "Sensor": "ESP32_Temp",
9     "Temperatura": 0,
10    "Unidad": "°C",
11  },
12  {
13    "Sensor": "ESP32_Temp",
14    "Temperatura": 0,
15    "Unidad": "°C",
16  },
17  {
18    "Sensor": "ESP32_Temp",
19    "Temperatura": 0,
20    "Unidad": "°C",
21  },
22  {
23    "Sensor": "ESP32_Temp",
24    "Temperatura": "",
25    "Unidad": "°C",
26  },
27  {
28    "Sensor": "ESP32_Temp",
29    "Temperatura": "",
30    "Unidad": "°C",
31  },
32  {
33    "Sensor": "ESP32_Temp",
34    "Temperatura": "",
35    "Unidad": "°C",
36  },
37  {
38    "Sensor": "ESP32_Temp",
39    "Temperatura": "",
40    "Unidad": "°C",
41  },
42  {
43    "Sensor": "ESP32_Temp",
44    "Temperatura": "",
45    "Unidad": "°C",
46  },
47  {
48    "Sensor": "ESP32_Temp",
49    "Temperatura": "",

```

Finalmente, se observa el último dato registrado en el Google Sheets:

The screenshot displays a Node-RED workflow on the left and a Google Sheets spreadsheet on the right. The Node-RED interface shows a flow starting with a 'Mensaje Prueba' input, followed by a 'POST' node, a 'json' node, and a 'debug POST' node. Below this, there is a 'Suscriptor ESP32' node (labeled 'connected'), followed by a 'debug ESP32' node, a 'GET' node, and another 'json' node. The 'debug GET' node shows the following data:

```

{
  "Sensor": "ESP32_Temp",
  "Temperatura": 29.3,
  "Unidad": "°C"
}

```

The Google Sheets spreadsheet on the right shows a table with columns A, B, and C. The data in the table is as follows:

	A	B	C
232	ESP32_Temp	34.7	°C
233	ESP32_Temp	22.1	°C
234	ESP32_Temp	20.8	°C
235	ESP32_Temp	32.2	°C
236	ESP32_Temp	31.5	°C
237	ESP32_Temp	20.5	°C
238	ESP32_Temp	22.1	°C
239	ESP32_Temp	22.9	°C
240	ESP32_Temp	23.3	°C
241	ESP32_Temp	27.1	°C
242	ESP32_Temp	23.5	°C
243	ESP32_Temp	29.3	°C
244	ESP32_Temp	32	°C
245			
246			
247			
248			
249			
250			
251			

The bottom of the spreadsheet shows a summary row with 'Suma: 32'.

La URL de la implementación en Google Apps Script para este caso:

[https://script.googleusercontent.com/macros/echo?user\\_content\\_key=3c8SLsUNlaKhpgc3oyEGxwUMpnCXXhFOTO\\_DFGjxsvYcMnfn1TVNYM7pQUVhvYSdC5KdyuZsOMCLNLGA9-4P5e0gwGBE\\_9A7m5\\_BxDIH2jW0nuo2oDemN9CCS2h10ox\\_1xSncGQaix\\_ryfHECjZEnJCpOjMvCTGsvArUCL4z701yOPPFarix1mO7ARf1qtmgk7m7FLirnoDpJW3DN0YEDa5Wu3R5Ct73ykHgm9G00D8JPryU2RmsiB9z9Jw9Md8uu&lib=MLuf14IDG-LB1GnxsiS1mSVYJ\\_e0-7KKK](https://script.googleusercontent.com/macros/echo?user_content_key=3c8SLsUNlaKhpgc3oyEGxwUMpnCXXhFOTO_DFGjxsvYcMnfn1TVNYM7pQUVhvYSdC5KdyuZsOMCLNLGA9-4P5e0gwGBE_9A7m5_BxDIH2jW0nuo2oDemN9CCS2h10ox_1xSncGQaix_ryfHECjZEnJCpOjMvCTGsvArUCL4z701yOPPFarix1mO7ARf1qtmgk7m7FLirnoDpJW3DN0YEDa5Wu3R5Ct73ykHgm9G00D8JPryU2RmsiB9z9Jw9Md8uu&lib=MLuf14IDG-LB1GnxsiS1mSVYJ_e0-7KKK)

The screenshot shows a Google Apps Script code editor with the following code:

```

1
2
3  "Sensor": "ESP32_Temp",
4  "Temperatura": 22.1,
5  "Unidad": "°C"

```

**Aclaración:** Para observar el último dato registrado o todos los datos registrados, se debe modificar el la función **doGet()** en el código de Google Apps Script, por último en las URL para cada caso realizado para la función **doGet()** se pueden observar los datos que se obtuvieron en ese momento.