EVIDENCIAS PRÁCTICA FINAL

Integrantes: JOSE DAVID FLOREZ RAMOS - 2174241
DIEGO ANDRES GARCIA DIAZ - 2195533
Fecha: 28/11/2024

Tarea 1: CREACIÓN DE PUBLICADOR-ESP32

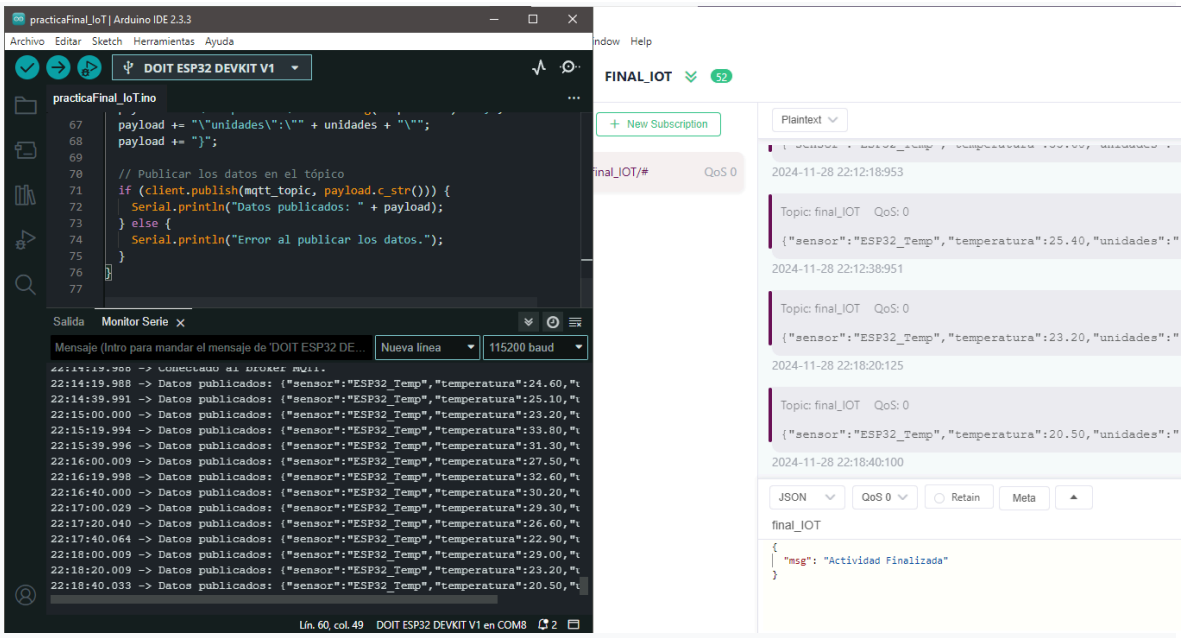
- Configure una ESP32 para que envíe datos de temperatura aleatorios.
- Cree un Topic en EMQX para que reciba los datos a través del protocolo MQTT.

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

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);
  }
}
```

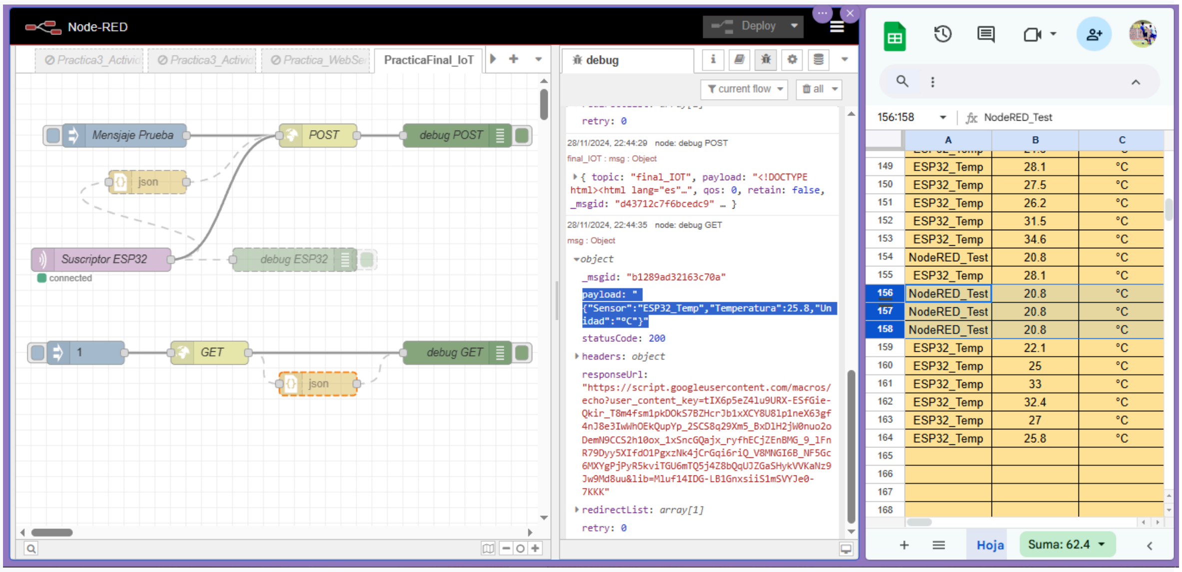
Tarea 3: Configuración en Node-RED

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

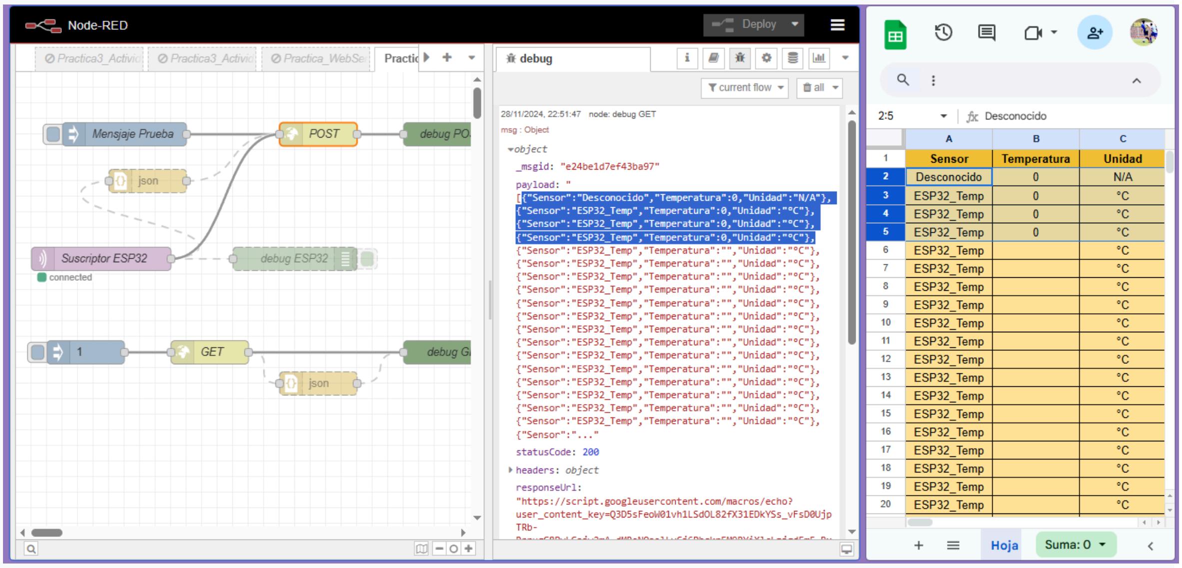
- Obtener la lista de temperaturas y mostrarla en el panel de debug.
- Permitir registrar una nueva temperatura desde un nodo de entrada.
- Instrucciones: Describe el flujo, los nodos utilizados y cómo se conectan entre sí. Incluye capturas de pantalla del flujo.

Obtener datos (GET):

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



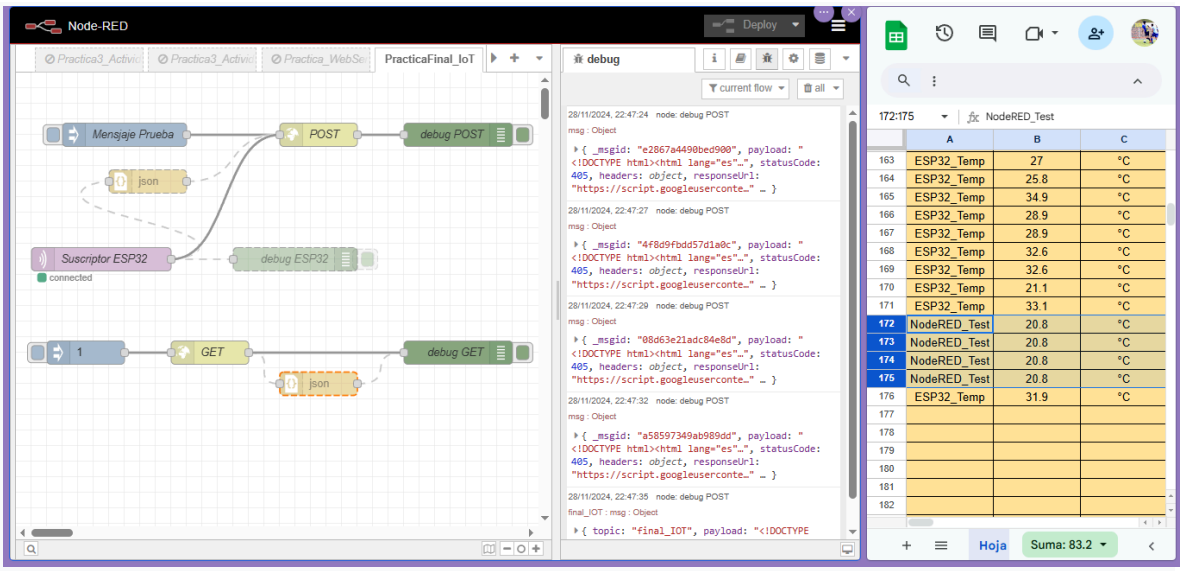
Ahora se obtienen los todos los datos que se han cargado en el Google Sheets:



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:



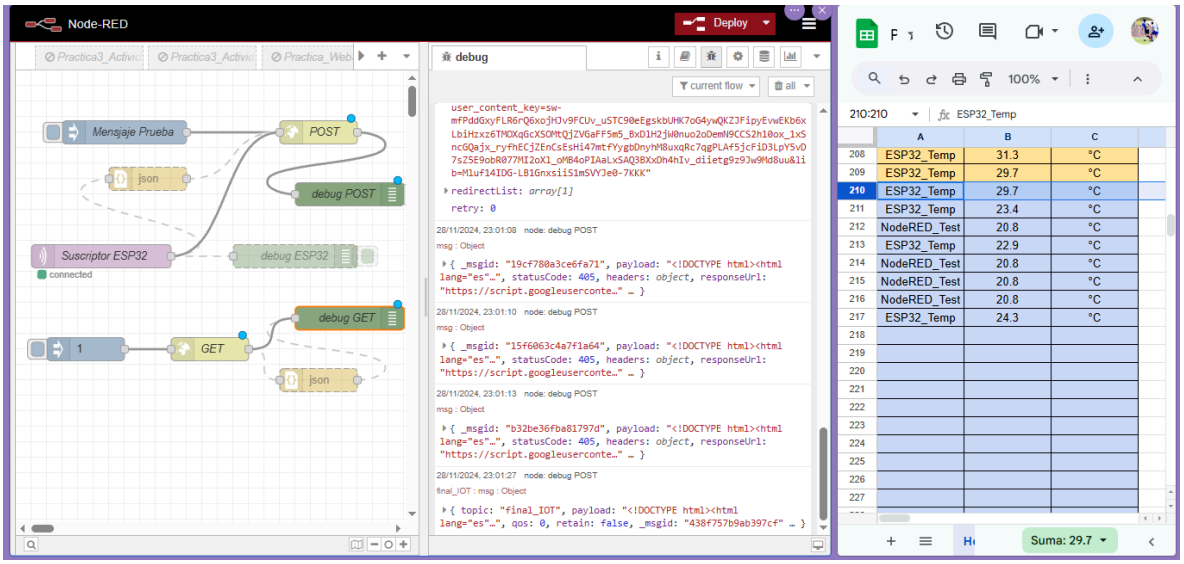
Tarea 4: Simulación de Solicitudes

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.

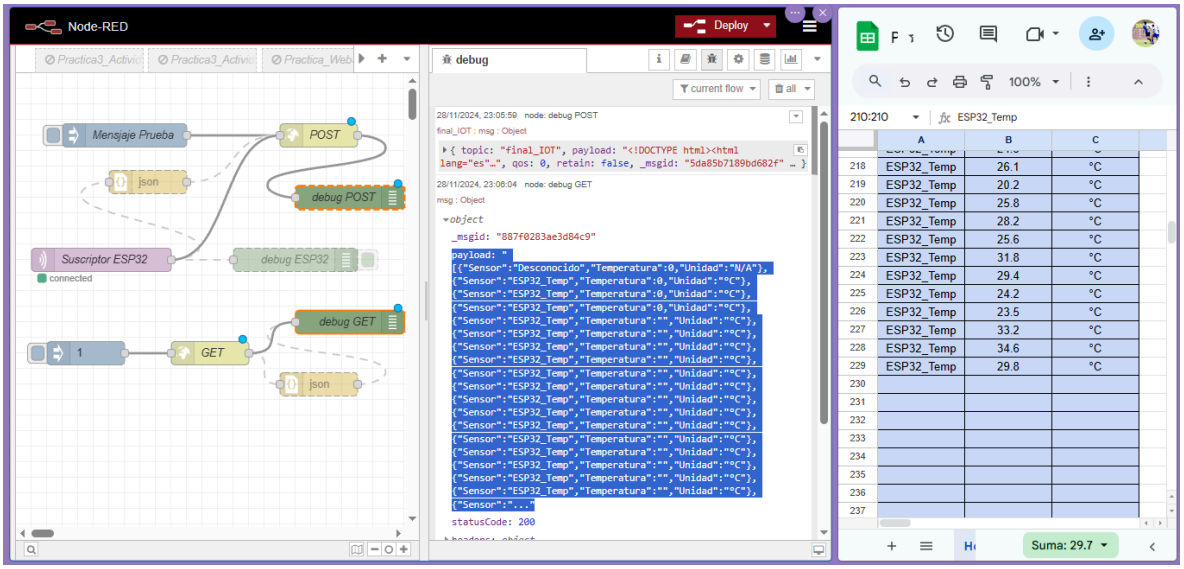
Solicitud POST:

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:



Solicitud GET:

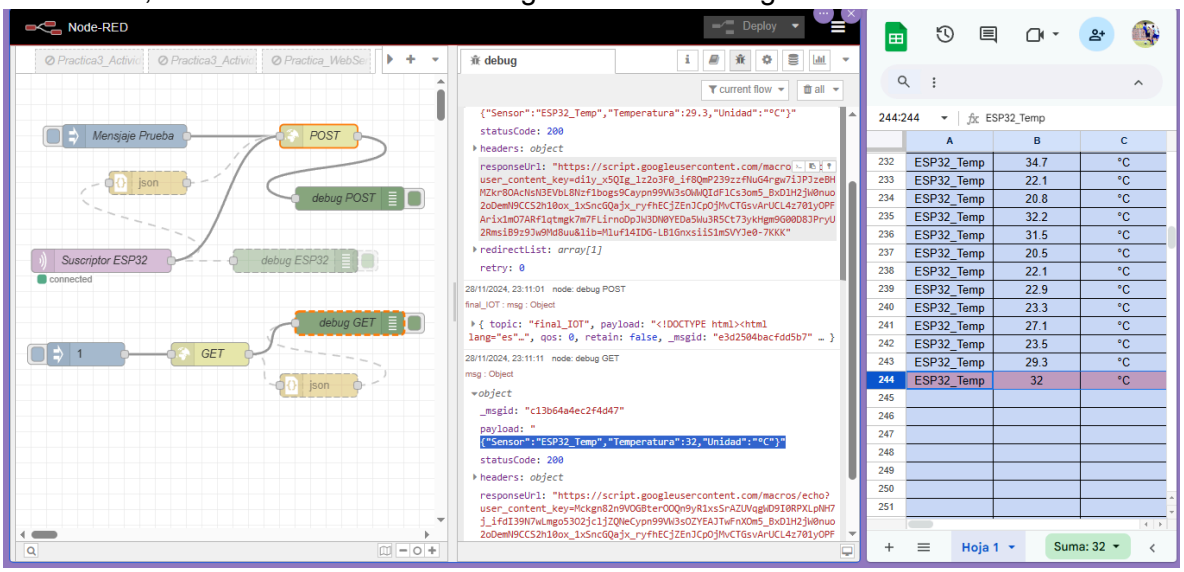
Primero se observa gracias a la SOLICITUD GET, todos los datos registrados en el Google Sheets:



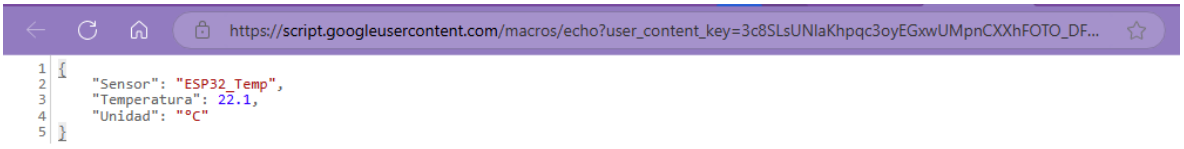
La URL de la implementación en Google Apps Script para este caso:
https://script.googleusercontent.com/macros/echo?user_content_key=VAZzRTedWe2qGzcu70MoT-oA-5n7gw-KKu-vf4R_cQDg4WEkmFjqi8rK6YSFR0YS4nCiMCtSlC6LNLGA9-4P5bZjiyq2DicGm5_BxDlH2jW0nuo2oDemN9CCS2h10ox_1xSncGQajx_ryfhECjZEnCsEsHi47mtfYygbDnyhM8uxqRc7qgPLAf5jcFiD3LpY5vD7sZ5E9obR077MI2oXl_oMB4oPIAaLxSAQ3BXxDh4hIv_diietg9z9Jw9Md8uu&lib=Mluf14IDG-LB1GnxsiS1mSVYJe0-7KKK


```
[
    {
        "Sensor": "Desconocido",
        "Temperatura": 0,
        "Unidad": "N/A"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": 0,
        "Unidad": "°C"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": 0,
        "Unidad": "°C"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": 0,
        "Unidad": "°C"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": "",
        "Unidad": "°C"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": "",
        "Unidad": "°C"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": "",
        "Unidad": "°C"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": "",
        "Unidad": "°C"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": "",
        "Unidad": "°C"
    },
    {
        "Sensor": "ESP32_Temp",
        "Temperatura": "",
```

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



La URL de la implementación en Google Apps Script para este caso:
https://script.googleusercontent.com/macros/echo?user_content_key=3c8SLsUNIaKhpqc3oyEGxwUMpnCXXhFOTO_DFGjxsvYcMNfn1TVNYM7pQUVhvYSdC5KdyuZsOMCLNLGA9-4P5e0gwGBe_9A7m5_BxDlH2jW0nuo2oDemN9CCS2h10ox_1xSncGQajx_ryfhECjZEnJCpOjMvCTGsvArUCL4z701yOPFArix1mO7ARf1qtmgk7m7FLirnoDpJW3DN0YEDa5Wu3R5Ct73ykHgm9G00D8JPryU2RmsiB9z9Jw9Md8uu&lib=Mluf14IDG-LB1GnxsiiS1mSVYJe0-7KKK



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.