✅ Student Final Guide: Project 3: Real-Time EEG Signal Monitoring and Logging to Google Sheets via ESP32

🧠 Overview

This project enables you to:

- Collect EEG data from an ESP32 microcontroller.
- Stream the data over serial communication.
- Use Python and Flask to read and serve the data.
- Expose the local server to the internet using ngrok.
- Log the data into Google Sheets via Google Colab.The Linux CodeTekonto+5LinkedIn+5The Linux Code+5

📁 Step 1: Create a Google Cloud Project

1. Navigate to Google Cloud Console.
2. Click on the project dropdown (top bar) → click “New Project”.
3. Set:
    - **Name**: `esp32-eeg-data-project`
4. Click **Create**.
5. Ensure the newly created project is selected as the active project.

🧩 Step 2: Enable Required APIs

Enable both APIs to allow Google Sheets access via Python:

- **Google Sheets API**: Enable Here
- **Google Drive API**: Enable Here

⏳ *Wait 1–2 minutes after enabling for APIs to propagate before proceeding.*

👤 Step 3: Create a Service Account

1. Go to Service Accounts.
2. Click "+ CREATE SERVICE ACCOUNT".
3. Set:
    - **Name**: `eeg-logger`
4. Click "Create and Continue".
5. Assign Role:
    - **Basic → Editor**
6. Click **Continue** → then **Done**.Microcontrollers Lab+2Sabbirz | Tech blog+2DEV Community+2

🔑 Step 4: Create and Download JSON Key

1. In the Service Accounts list, click on the one you just created.
2. Navigate to the **Keys** tab → Click "Add Key" → "Create new key".
3. Select **JSON** → Click **Create**.
4. This downloads a file like:
    - `esp32-eeg-data-project-xxxxxxxxxxxx.json`

🔒 *Keep this JSON file safe — you'll upload this in Google Colab or your Python project.*

📄 Step 5: Share Your Google Sheet with the Service Account

1. Open your target Google Sheet in a browser.
2. Open your downloaded JSON file and copy the `client_email` field.
    - It looks like:
        - `esp32-eeg-data-project@esp32-eeg-data-project.iam.gserviceaccount.com`
3. In the Google Sheet, click “Share” and paste the email, giving it **Editor** access.Hackster

✅ *This authorizes the Python script to write to the sheet.*

🧠 Step 6: ESP32 Arduino Code for EEG Data Acquisition

cpp

CopyEdit

```
#define SAMPLE_RATE 256             // Sampling rate in Hz

#define BAUD_RATE 9600              // Serial speed

#define INPUT_PIN 34                // Use ADC1 channel on GPIO34

#define ADC_RESOLUTION 12           // ESP32 max ADC resolution
(12-bit)

#define V_REF 3.3                   // ADC reference voltage in volts
```

```
#define ADC_MID 2048                // Midpoint of 12-bit ADC (virtual
0V)


void setup() {
  Serial.begin(BAUD_RATE);
  analogReadResolution(ADC_RESOLUTION);  // Set ADC to 12-bit (0–4095)
}


void loop() {
  static unsigned long lastMicros = 0;
  unsigned long now = micros();
  unsigned long interval = 1000000UL / SAMPLE_RATE;


  if (now - lastMicros >= interval) {
    lastMicros = now;


    int raw_adc = analogRead(INPUT_PIN);                    // Read
from ADC
    float signal = raw_adc - ADC_MID;                       // Remove
1.65V bias (simulate 0V-centered signal)
    float filtered = EEGFilter(signal);                     // Apply
digital filter
    Serial.println(filtered, 2);                            // Output
filtered signal
  }
```

```
}


// -------------------- Digital IIR Bandpass Filter
--------------------
// Designed in cascaded biquad form, cutoff ~1–50 Hz, fs = 500 Hz
float EEGFilter(float input) {
  float output = input;

  {
    static float z1, z2;
    float x = output - -0.95391350 * z1 - 0.25311356 * z2;
    output = 0.00735282 * x + 0.01470564 * z1 + 0.00735282 * z2;
    z2 = z1;
    z1 = x;
  }
  {
    static float z1, z2;
    float x = output - -1.20596630 * z1 - 0.60558332 * z2;
    output = 1.00000000 * x + 2.00000000 * z1 + 1.00000000 * z2;
    z2 = z1;
    z1 = x;
  }
  {
```

```
    static float z1, z2;
    float x = output - -1.97690645 * z1 - 0.97706395 * z2;
    output = 1.00000000 * x + -2.00000000 * z1 + 1.00000000 * z2;
    z2 = z1;
    z1 = x;
  }
  {
    static float z1, z2;
    float x = output - -1.99071687 * z1 - 0.99086813 * z2;
    output = 1.00000000 * x + -2.00000000 * z1 + 1.00000000 * z2;
    z2 = z1;
    z1 = x;
  }

  return output;
}
```

🐍 Step 7: Python Flask Server to Read Serial Data

python

CopyEdit

```
from flask import Flask, jsonify
```

```
import serial

import threading


# Flask app setup

app = Flask(__name__)


# Arduino serial port configuration

serial_port = 'COM9'  # Replace with your port

baud_rate = 9600       # Baud rate for Arduino communication


# Attempt to connect to the serial port

ser = None

try:

    ser = serial.Serial(serial_port, baud_rate, timeout=1)

    print(f"Connected to {serial_port}")

except Exception as e:

    print(f"Error connecting to {serial_port}: {e}")


# Shared variable to store the latest sensor data (as a list of
floats)

latest_data = [0.0]  # Initialize with one value (most recent sensor
reading)
```

```
# Home route
@app.route('/', methods=['GET'])
def home():
    return "Server is running. Use /data to fetch sensor data."

# Data route to fetch the latest sensor data as JSON
@app.route('/data', methods=['GET'])
def get_data():
    return jsonify(latest_data)  # Return the latest data as JSON

# Function to read data from the Arduino
def read_serial():
    global latest_data
    if ser is None:
        print("Serial port not available. Exiting serial reader.")
        return

    while True:
        try:
            line = ser.readline().decode('utf-8').strip()
            if line:
                print("Raw data from Arduino:", line)  # Debugging
output
```

```
                # Convert the value to a float
                try:
                    value = float(line)
                    latest_data[0] = value  # Update with the new
value
                except ValueError:
                    print(f"Invalid data received: {line}")
        except serial.SerialException as e:
            print("Serial disconnected:", e)
            break  # Exit the loop if there's a serial exception
        except Exception as e:
            print("Error reading from serial:", e)

# Run Flask app and serial reader in parallel using threading
if __name__ == '__main__':
    # Start the serial reading in a separate thread
    threading.Thread(target=read_serial, daemon=True).start()
    # Run Flask server
    app.run(host='0.0.0.0', port=8080)  # Expose on all network
interfaces
```

🌐 Step 8: Install and Use ngrok https://youtu.be/aFwrNSfthxU?si=Fb9eZXgu2TkLjsnf

1. Download ngrok from ngrok.com.
2. Unzip and install it.
3. Authenticate ngrok:
    - Sign up and get your auth token from the ngrok dashboard.

Run:

bash
CopyEdit
```
ngrok authtoken YOUR_AUTH_TOKEN
```

    -

Expose your local server:

bash
CopyEdit
```
ngrok http 8080
```

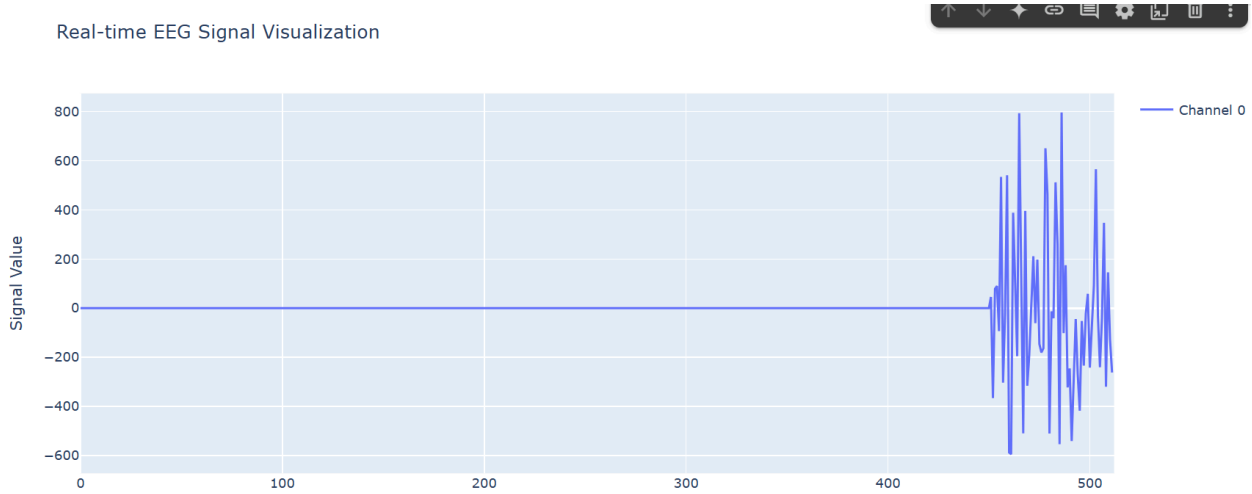
4.
5. Note the generated public URL (e.g., `https://abcd1234.ngrok.io`).Tekontohelp.rocketadmin.com+4The Linux Code+4Topaz+4IoT Design Pro+2GeeksforGeeks+2Hackster+2Codreline+5fxis.ai+5LinkedIn+5The Linux Code+1LinkedIn+1

    🔗 *This URL will be used in Google Colab to access your local Flask server.*

📓 Step 9: Google Colab Script to Log Data to Google Sheets

1. Open Google Colab.
2. Upload your JSON key file.

3. Visualise the Data in the graph as output and same is stored in google sheet



ESP332toGOOGLESHEET

File Edit View Insert Format Data Tools Extensions Help

100% $ % .0 .00 123 Defaul... 10 B

A1 | Time

	A	B	C	D	E	F
4	2025-05-12 11:5	78.25				
5	2025-05-12 11:5	89.7				
6	2025-05-12 11:5	-93.34				
7	2025-05-12 11:5	534.41				
8	2025-05-12 11:5	-303.44				
9	2025-05-12 11:5	-25.4				
10	2025-05-12 11:5	540.59				
11	2025-05-12 11:5	-588.23				
12	2025-05-12 11:5	-595.16				
13	2025-05-12 11:5	388.03				
14	2025-05-12 11:5	101.22				
15	2025-05-12 11:5	-195.81				
16	2025-05-12 11:5	793.55				
17	2025-05-12 11:5	122.75				
18	2025-05-12 11:5	-510.19				
19	2025-05-12 11:5	395.69				
20	2025-05-12 11:5	-316.28				

4.