ESP8266 Sensor Logger using Arduino IDE & Flask

# Objective

The goal of this project is to build a complete system using an ESP8266 or ESP32 microcontroller that reads sensor data (such as temperature, acceleration, gyroscope, magnetometer, and angle data), formats it into JSON, and sends it to a Flask-based Python server over WiFi. The server receives this data, stores it in a CSV file, and provides a live-updating web interface to view the logged data in real-time.

# ARDUINO CODE EXPLANATION

## 1. Libraries Used

## Wire.h

Used for I2C communication. It connects the ESP8266/ESP32 with devices like the DS3231 RTC.

## HWT9053.h

Custom library created to interface with the HWT9053-485 9-DOF IMU sensor. Provides access to angle, acceleration, gyroscope, magnetometer, and temperature data.

## SoftwareSerial.h

Enables additional serial ports using digital pins, necessary for RS485 to TTL data reception from the HWT9053 sensor.

## SPI.h

Allows SPI communication between the ESP board and external peripherals like the SD card module.

## SD.h

Handles interaction with the SD card module. It helps save sensor data locally in a .csv format.

## RTClib.h

Communicates with DS3231 RTC to get current date and time. This ensures proper timestamping of sensor readings even without internet.

## ESP8266WiFi.h

Manages WiFi connections for ESP8266, allowing it to connect to a network and send data online.

## ESP8266HTTPClient.h

Handles HTTP client functionality. Used to send HTTP POST requests with sensor data to the Flask server.

## WiFiUdp.h

Supports sending/receiving UDP packets over WiFi. Used in combination with NTPClient to get time from internet-based NTP servers.

## NTPClient.h

Fetches current time from NTP servers using WiFi + UDP. Acts as an alternative or supplement to DS3231 for timestamping.

## 2. Key Functions in Code

- setup (): Initializes serial monitor, WiFi connection, and sensor modules.  
- loop (): Continuously reads sensor values, prepares a JSON payload, and sends it to the Flask server.

## 3. Output/Working of Arduino Code

The Arduino continuously reads data from sensors (e.g., angle, acc, gyro, mag, temp), forms a JSON string using ArduinoJson, and sends it to the Flask server endpoint `/sensor/upload` using an HTTP POST request. If successful, the server responds with a confirmation message.

# PYTHON FLASK CODE EXPLANATION

## 1. Libraries Used

### Flask

Web server framework to handle HTTP requests and serve HTML pages.

### request

Part of Flask, used to access incoming JSON data from the client (ESP device).

### jsonify

Converts Python dictionaries into JSON responses.

### render\_template\_string

Renders HTML code directly from a Python string using Jinja2 templating.

### csv

Used to write incoming data to a CSV file for logging and later viewing.

### os

Checks whether the CSV file already exists to prevent overwriting headers.

### datetime

Generates timestamp if ESP device doesn't send one.

## 2. Flask Code Structure & Function Explanation

- `@app.route('/sensor/upload', methods=['POST'])`: Accepts data sent from Arduino. The `request.get\_json()` function fetches the JSON, parses it, and appends the values to `sensor\_data.csv`.  
- `@app.route('/view')`: Serves an HTML page that auto-refreshes every second and displays the data from the CSV in a table. The table scrolls to the newest data automatically using JavaScript.

## 3. Working/Output of Flask Code

When data is received from the ESP, the server writes a new row to `sensor\_data.csv`. When a user visits `http://<server\_ip>:5000/view`, they see a live-updating HTML table with all logged values (timestamp, angleX/Y/Z, acceleration, gyroscope, magnetometer, and temperature). Each second, the table refreshes to show the latest data.

# ADDITIONAL HARDWARE COMPONENTS AND INTEGRATION:

The setup is expanded with the inclusion of an HWT9053-485 sensor, an RS485 to TTL converter, an SD card module, and a DS3231 Real-Time Clock (RTC) module. These components work together to log sensor data with accurate timestamps, even when disconnected from the Flask server or WiFi.

## 1. HWT9053-485 Sensor with RS485 to TTL

The HWT9053-485 is a 9-axis IMU sensor providing data such as angle, acceleration, gyroscope, magnetometer, and temperature via RS485 serial communication. Since the ESP8266 does not support RS485 directly, an RS485 to TTL converter is used to adapt the signal for UART reading. The ESP reads this sensor data periodically in the main loop.

## 2. SD Card Logging with DS3231 RTC Timestamping

An SD card module is used to store the sensor data in `.csv` format for offline access. The SD module communicates with the ESP8266 via the SPI protocol. To ensure each entry is accurately timestamped even without internet access, a DS3231 RTC module is connected to the SD card system.  
  
The DS3231 provides precise time information that is used as the timestamp when logging the sensor data to the SD card. This timestamp is appended alongside the sensor readings, making the logged data consistent and traceable.

## 3. Combined Working Summary with RTC and SD Card

1.HWT9053-485 sensor sends IMU data via RS485.

2. RS485 to TTL converter adapts signal to UART for ESP8266.

3. ESP8266 reads data, fetches current timestamp from DS3231 RTC.

4. ESP8266 writes the data + timestamp to SD card over SPI.

5. If WiFi is available, data is also sent to Flask server for live view and CSV logging.