Real-Time Environmental Data Acquisition Using the Bosch BME280 Sensor and ESP32-S3

Abstract:

This paper presents a system for real-time environmental data acquisition and transmission using the Bosch BME280 sensor, an ESP32 microcontroller, and Google Sheets as a cloud-based data storage platform. The BME280 sensor, known for its high precision, measures temperature, humidity, and atmospheric pressure, making it ideal for various environmental monitoring applications. The ESP32 microcontroller interfaces with the sensor, processes the data in real time, and wirelessly uploads the processed data to a Google Sheet using HTTP POST requests. This setup allows for seamless integration with Google Sheets, enabling remote access to historical environmental readings from any internet-enabled device. The system is designed with a focus on low power consumption, scalability, and ease of deployment, making it suitable for applications such as smart agriculture, weather stations, and indoor climate monitoring. This paper details the hardware and software implementation, including data handling techniques, system architecture, and the performance evaluation of the proposed system in real-world environments.

Introduction:

This paper presents a real-time environmental monitoring system that leverages the Bosch BME280 sensor and ESP32 microcontroller to capture and upload environmental data to Google Sheets via HTTP POST requests. The BME280 sensor provides accurate measurements of temperature, humidity, and atmospheric pressure, making it ideal for applications such as smart agriculture, weather stations, and indoor climate monitoring. The ESP32 microcontroller, known for its versatility and low power consumption, interfaces with the sensor to collect data in real time. The data is then transmitted to a Google Sheet, providing a cloud-based solution for remote access and data storage. This approach offers users the convenience of accessing environmental data from any internet-enabled device, with the added benefits of long-term storage and the ability to analyse trends over time. The system is designed for ease of deployment, scalability, and low power consumption, making it well-suited for both large-scale environmental applications and localized monitoring projects. This paper outlines the hardware and software components of the system, the data handling and transmission methods, and an evaluation of its performance in various real-world scenarios.

Hardware Setup:

ESP32: A microcontroller with built-in Wi-Fi that connects to your Wi-Fi network and gathers data from the sensor.
BME280 Sensor: A sensor that reads environmental data, such as temperature, humidity, and pressure. This sensor communicates with the ESP32 via I2C.
I2C Configuration: Custom I2C pins (SDA˙PIN 8 and SCL˙PIN 18) are used to connect the ESP32 and BME280.

Wi-Fi Connection:
The ESP32 connects to the specified Wi-Fi network using the credentials (SSID and password).
Once connected, it can send HTTP requests over the internet.
Sensor Data Collection:
The ESP32 uses the **Adafruit BME280** library to read temperature, humidity, and pressure data from the sensor.
It then packages this data into a JSON object using the **Arduino Json** library for structured transmission over HTTP.
HTTP Communication:
The ESP32 sends an HTTP POST request to a Google Apps Script Web App URL (scriptURL) which handles the data.
In the HTTP POST request, sensor data (temperature, humidity, pressure) is sent in JSON format.
If the request is successful, the ESP32 receives a response from the server and prints it to the Serial Monitor.

Google Apps Script (Server-side):

POST Request Handling: The script receives the sensor data in JSON format, extracts the values, and appends them to a Google Sheet. Each row contains a timestamp and the sensor data.
GET Request Handling: For debugging or data retrieval, the Apps Script can respond to GET requests, returning the last row of data from the sheet in JSON format.
Error Handling: If the script encounters any errors (e.g., no data is received), it logs the error and responds with an appropriate message.

Key Technical Aspects:

I2C Protocol: The Inter-Integrated Circuit (I2C) protocol is a widely used communication standard designed for low-speed, short-distance, serial data exchange between devices, primarily in embedded systems. Developed by Philips (now NXP) in the early 1980s, I2C enables multiple slave devices to communicate with one or more master devices over just two lines: a serial data (SDA) line and a serial clock (SCL) line. The protocol supports multi-master and multi-slave configurations and uses 7-bit or 10-bit addressing to uniquely identify each device on the bus. It operates in half-duplex mode and supports multiple speed modes, including standard mode (up to 100 kbit/s), fast mode (up to 400 kbit/s), and high-speed mode (up to 3.4 Mbit/s).

HTTP POST: An **HTTP POST** is one of the main request methods used in the HTTP (Hypertext Transfer Protocol), typically used to send data from a client (like a web browser or app) to a server. Unlike the GET request, which appends data to the URL and retrieves data, a POST request submits data in the body of the request.

ESP32 Features:
Wi-Fi connectivity to send HTTP requests.
Ability to read data from the BME280 sensor using I2C.
JSON Data Structure: Sensor data is structured in JSON format for transmission.
Google Apps Script: This acts as the server, processing incoming POST requests and managing data in Google Sheets.
Data Logging: The project ensures that sensor data is logged with a timestamp for future analysis.

Methodology:

Data Acquisition:

Sensor and Microcontroller Setup: The ESP32 microcontroller was connected to a BME280 sensor via the I2C protocol to collect temperature, humidity, and pressure readings.
Google Sheets Integration: A Google Apps Script was developed to receive the sensor data from the ESP32 and append it to a Google Sheet. The script's doPost function handled incoming data, while the doGet function allowed for real-time data retrieval for analysis and debugging.