Design, Prototyping, and Deployment of a Wireless Sensor Network for Pollution Monitoring

A Project Report submitted
for evaluation of the course
EE 396 : DESIGN LAB

by

Dalisha Nagpal
(220108019)

Priyanshu Maurya
(220108046)

under the guidance of

Prof. Prabir Barooah

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

***Abstract*—Air pollution, particularly fine particulate matter, poses a significant threat to public health and environmental sustainability, especially in regions with poor air quality. Effective monitoring of personal and environmental exposure to pollutants is critical for understanding their impact and informing health policies. This project aims to design and deploy a low-cost, portable air quality monitoring system to measure particulate matter (PM1.0, PM2.5, PM10), temperature, and humidity in diverse settings, including remote, industrial, and campus areas. By integrating affordable sensors and wireless communication, the system captures real-time data, enabling detailed analysis of air quality trends. Encased in a weatherproof enclosure with solar power, the device is field-deployable, offering a scalable solution for continuous monitoring. The collected data supports health exposure studies by providing accurate, location-specific insights into daily pollution levels, contributing to better environmental management and public health outcomes.**

I. Background

Air pollution poses a significant threat to public health and environmental sustainability, particularly in India, where it is a leading cause of premature mortality and morbidity. According to the State of Global Air 2024 report, based on the Global Burden of Disease (GBD) Study 2021, air pollution accounted for approximately **2.1 million deaths in India in 2021**, with ambient particulate matter (**PM2.5**) being the primary contributor. This makes air pollution the second-leading risk factor for death in the country, surpassed only by high blood pressure. The majority of these deaths are linked to non-communicable diseases such as heart disease, stroke, lung cancer, and chronic obstructive pulmonary disease (COPD), highlighting the severe health consequences of prolonged exposure to polluted air.

This is compounded by traditional air quality monitoring systems being often costly and limited in their deployment, particularly in remote or underserved areas. These systems typically rely on stationary, high-cost instruments that provide data from centralized locations, which may not capture the spatial and temporal variability of pollution exposure across diverse microenvironments—such as homes, workplaces, or remote areas—where individuals spend most of their time. This limitation hinders the ability to assess personal exposure to pollutants, a critical factor in understanding the health impacts of air pollution and informing targeted interventions.

The sources of air pollution in India are diverse, including industrial emissions, vehicular exhaust, construction dust, crop burning, and household use of solid fuels for cooking and heating. In 2021, industrial pollution accounted for 51% of **PM2.5** emissions, vehicles 27%, crop burning 17%, and other sources 5% (Air Pollution in India). This variability underscores the need for monitoring systems that can operate in urban, industrial, and rural settings to provide a comprehensive picture of pollution exposure. Moreover, children under age five are particularly vulnerable, with 169,400 deaths linked to air pollution in India in 2021, the highest globally for this age group (State of Global Air 2024).

To address these challenges, there is a growing need for **low-cost, portable, and scalable air quality monitoring solutions** that can provide real-time data across a wide range of settings. Such systems can empower individuals, communities, and policymakers with actionable insights to mitigate health risks and improve environmental management. This project aims to develop a **wireless sensor network** for real-time air quality monitoring, leveraging affordable sensors and Internet of Things (IoT) technologies for wireless connectivity to measure key environmental parameters, including **particulate matter (PM1.0, PM2.5, PM10)**, **temperature**, and **humidity**.

The proposed system consists of a data collection module, i.e., a DFRobot Air Quality Monitor sensor, which uses laser scattering for accurate PM measurements, and a 7Semi EC200U LTE 4G GNSS IoT Smart Modem with inbuilt Arduino for data processing and transmission. These modules are designed to be portable and autonomous, powered by a 10W 12V solar panel and a 3P 3.7 V, 2200 mAh Li-ion battery configuration managed by a Waveshare Solar Power Manager. This setup ensures that the system can operate independently in remote, industrial, and campus environments, where access to power and connectivity may be limited. The use of 4G LTE communication overcomes limitations of Wi-Fi availability and SMS restrictions in India, enabling reliable data uploads to a cloud platform like ThingSpeak.

By deploying this sensor network, the project seeks to provide detailed, location-specific air quality data that can support health exposure studies, inform pollution control strategies, and enhance environmental awareness. The system's affordability and scalability make it accessible to underserved areas, where air quality monitoring is often absent. The data collected can also contribute to predictive models and early warning systems, enabling proactive measures to mitigate pollution spikes.

The project aligns with national efforts like India's National Clean Air Programme (NCAP), which aims to reduce PM concentrations in urban areas, and global ini-

tiatives such as the United Nations' Sustainable Development Goals, particularly Goal 3 (Good Health and Well-Being) and Goal 11 (Sustainable Cities and Communities). By providing a sustainable monitoring solution, the system supports these initiatives and contributes to long-term environmental and public health goals.

In summary, this project addresses the urgent need for accessible and effective air quality monitoring in India by developing a **low-cost, portable, and wireless sensor network**. By providing real-time data on key pollutants, the system aims to contribute to a better understanding of air pollution dynamics and support efforts to improve public health and environmental quality.

II. System Architecture

The architecture diagram illustrates the seamless integration of hardware components to achieve continuous air quality monitoring. The DFRobot Air Quality Monitor collects environmental data, which is processed by the EC200U modem's embedded ATMEGA328P-AU microcontroller (Arduino Uno compatible). The modem transmits the data via TCP/IP over 4G LTE to ThingSpeak, where it is stored and visualized. The solar power system, comprising a 10W 12V solar panel, Waveshare Solar Power Manager, and 3P Li-ion battery pack, ensures energy autonomy. A weatherproof enclosure protects the system while maintaining proper sensor airflow. This architecture provides a scalable, efficient solution for real-time air quality monitoring with remote access capabilities.

A. *Block Diagram:*

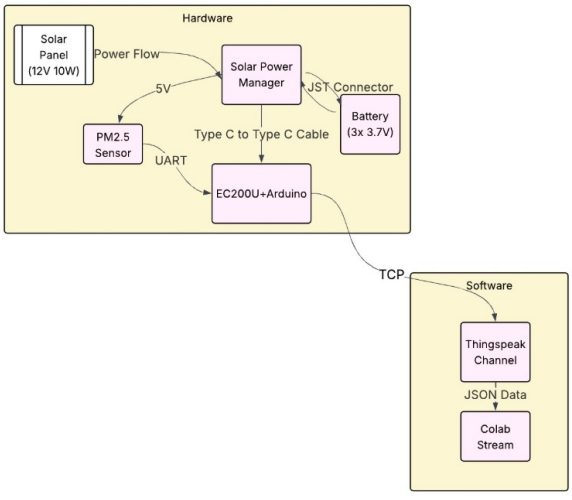

Fig. 1. Block Diagram of the prototype's structure

III. Hardware Design

A. *Component selection*

*1) **7Semi EC200U LTE 4G GNSS IoT Smart Modem with inbuilt Arduino:**:* The 7Semi EC200U LTE 4G IoT Smart Modem is the heart of our air quality monitoring system, handling data collection, processing, and sending it to the internet. It's like a tiny computer and phone combined, using a built-in Arduino chip to run our code and a 4G connection to send sensor data to a website called ThingSpeak. It can be powered by a USB cable or a battery, which is great for our solar-powered setup. The modem talks to the air quality sensor through special pins and uses a SIM card to connect to the internet, just like a smartphone. We picked it because it's easy to program with Arduino software, saves power, and works almost anywhere, making it perfect for monitoring air quality in far-off places.

Role in the System:

- Collects data from the DFRobot Air Quality Monitor via UART.
- Processes data using the embedded Arduino Uno.
- Transmits data to the ThingSpeak server via TCP/IP over 4G LTE.
- Manages power states to optimize energy efficiency.

Fig. 2. 7Semi EC200U LTE 4G GNSS IoT Smart Modem with inbuilt Arduino

*2) **DFRobot PM2.5 Air Quality Monitor**:* The DFRobot Air Quality Monitor is the key sensor in our air quality monitoring system, designed to measure tiny particles in the air, **temperature**, and **humidity** with great accuracy. It uses a laser to detect particles like **PM1.0**, **PM2.5**, and **PM10**, which can affect health, and it also tracks **temperature** and **humidity** to give a complete picture of the environment. Powered by a simple 5V supply, it sends data quickly through a connection called UART, making it easy to work with our modem. We chose this sensor because it's small, reliable, and gives fast, precise readings, which is perfect for checking air quality in

places like cities or rural areas where pollution can be a problem.

Fig. 3. DFRobot Air Quality Monitor

*3) **Solar Power System**:* The system is designed to be autonomous, relying on solar power for continuous operation in remote locations.

- Our air quality monitoring system uses a 10W 12V polycrystalline solar panel to keep it running without needing a plug, which is perfect for remote places like fields or forests. This panel captures sunlight and turns it into electricity to charge the battery and power the sensor and modem. It's designed to work efficiently, producing enough energy even on cloudy days to keep everything going. We chose this solar panel because it's affordable, durable, and just the right size to meet our system's needs, ensuring our air quality monitor can work continuously wherever it's placed.

Fig. 4. The chosen solar panel

- The Waveshare Solar Power Manager and 3P Li-ion battery pack work together to keep our air quality monitoring system powered up using sunlight, making it perfect for remote areas without electricity. The Waveshare controller takes energy from the solar panel, supports a wide range of inputs, and delivers a steady 5V output to run the sensor and modem. It also safely charges three Li-ion batteries in a 3P configuration, each with a 2200 mAh capacity and together, the battery pack stores enough energy to keep the system running for up to 15 hours when it's dark or cloudy. We chose these components because they're reliable, easy to connect, and ensure our system stays on without needing constant maintenance, allowing us to monitor air quality continuously in far-off places.

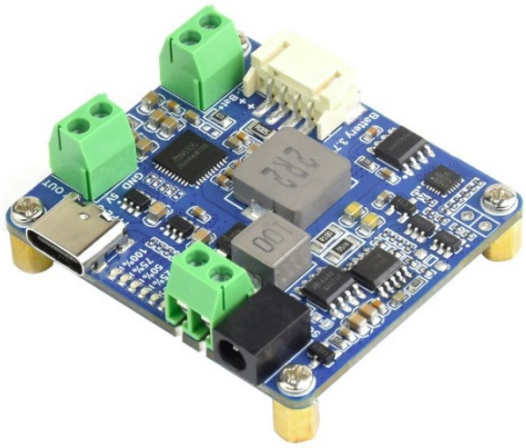

Fig. 5. Solar Power Manager

*4) **Perf Board Design and Support:**:*

- The 15 x 20 cm perf board is the backbone of our air quality monitoring system, acting like a sturdy platform that holds and connects all the key components neatly inside the waterproof box. Made from strong FR4 material, this double-sided board is designed with tiny holes spaced closely together, making it easy to solder parts like the 3P battery holder at the top center, the Waveshare Solar Power Manager in the bottom left corner, the EC200U modem in the middle bottom, and the PM2.5 Module Adapter in the bottom right. The battery holder powers the system, the solar manager handles energy from the solar panel, the modem processes and sends data, and the adapter links to the air quality sensor using a 15 cm JST cable. We chose this board because its size fits perfectly in the

box, leaving room for other parts, and its reliable design ensures all connections stay secure, even in tough environments, making our system simple to build and maintain.

- Support: Double-sided foam tape (3 mm thick, EVA material) applied to the perf board's underside, securing it to the box's base. The foam tape minimizes mechanical stress from vibrations or impacts during transport or deployment, enhancing component longevity and connection stability.

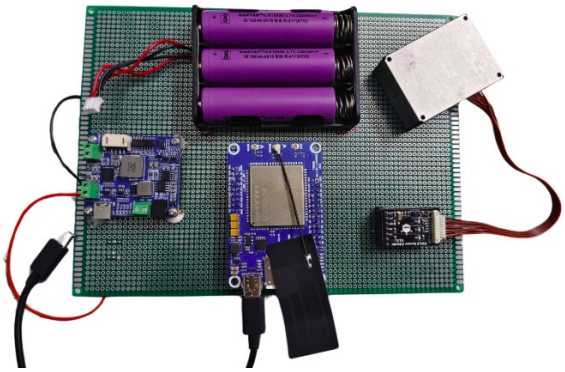

Fig. 6. Front view of perf board

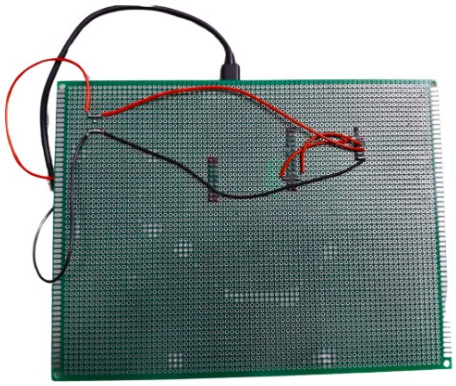

Fig. 7. Back view of perf board showcasing the soldered ends and connections.

*5) **Enclosure and Mounting**:*

- The ABS Junction Box from GLOBOMOTIVE, measuring 300 x 250 x 120 mm, is the protective home for our air quality monitoring system, keeping all components safe from rain, dust, and extreme temperatures. Made from tough ABS plastic, it's waterproof and dustproof, perfect for harsh outdoor settings. We placed the DFRobot Air Quality Monitor at the bottom center of the box and drilled three holes—two on the sides and one in the middle of the bottom—to let air flow in and out for accurate sensor readings. The holes prevent air from mixing, ensuring the sensor works properly. We also used foam tape to secure the perf board, which holds the modem and other parts, to the box's base, reducing shakes and bumps during transport. We chose this box because it's strong, roomy enough for all our components, and keeps everything safe and stable, making it ideal for monitoring air quality in remote or tough locations.
- Challenges and Solutions: Airflow Optimization: Initially planned for two holes, but added a third hole on the bottom length to meet nominal airflow requirements, ensuring sufficient ventilation without compromising IP65/67 protection.

Fig. 8. Holes were made on the waterproof box, allowing airflow.

- Heat Sink: Manages heat dissipation for the PM2.5 sensor. It is attached to the sensor to maintain optimal temperatures.

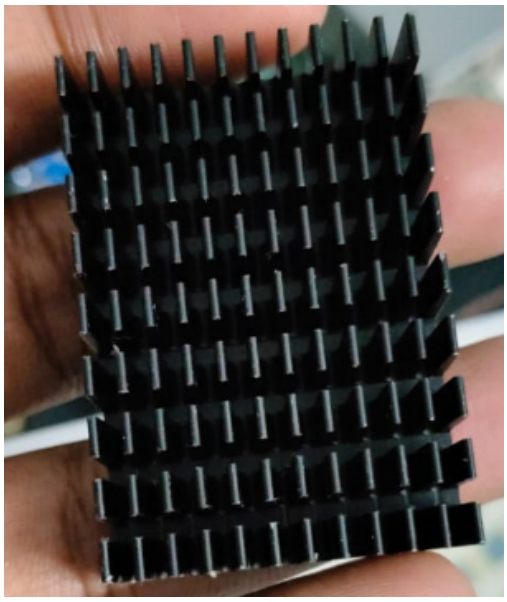

Fig. 9. Heat sink

IV. Software Design

A. Code Explanation

The code for our air quality monitoring system is designed to collect data from a sensor, process it, and send it to a website called ThingSpeak so we can see the results online. It starts by setting up two communication channels: one to talk to the DFRobot Air Quality Monitor, which measures tiny particles in the air (**PM1.0**, **PM2.5**, **PM10**), **temperature**, and **humidity**, and another to talk to the EC200U modem, which sends data over the internet using 4G. Every 5 minutes, the system reads a packet of data from the sensor, checks if it's correct, and pulls out the important measurements. If the data is good, it formats these measurements into a message and uses the modem to send them to ThingSpeak, where they can be viewed as charts or graphs. The code also handles tasks like initializing the modem, managing connections, and waiting for responses to ensure everything runs smoothly. This setup enables real-time air quality monitoring with user-friendly commands.

*1) **Software Serial**:* In our air quality monitoring system, we used **SoftwareSerial** in the code because the 7Semi EC200U modem, which has an embedded ATMEGA328P-AU microcontroller (like an Arduino Uno), has only one hardware serial port for communication. This port is typically used to connect to a computer for debugging or displaying information on the Serial Monitor. However, our system needs to communicate with two devices at the same time: the DFRobot Air Quality Monitor (to read sensor data) and the EC200U's modem module (to send data to ThingSpeak over 4G). Since we can't use the single hardware serial port for both, SoftwareSerial creates virtual serial ports on other digital pins (pins 2 and 3 for the modem, pins 8 and 9 for the sensor). This lets us talk to both devices without needing extra hardware, making the setup simpler and cheaper for a beginner-friendly project.

We needed to **switch between SoftwareSerial ports in the loop** (using `mySerial.listen()` for the sensor and `gsm.listen()` for the modem) because SoftwareSerial can only listen to one virtual port at a time. In the code, we first switch to the sensor's port to read the 32-byte data packet containing **PM1.0**, **PM2.5**, **PM10**, **temperature**, and **humidity**. Once we've collected and processed this data, we switch to the modem's port to send the data to ThingSpeak via 4G. Switching is necessary to avoid missing data from either device, as trying to listen to both simultaneously would cause conflicts and errors. This approach ensures the system reads sensor data accurately and sends it to the cloud reliably, which is critical for real-time air quality monitoring in places like remote or industrial areas.

B. AT Commands

*1) **Introduction to AT Commands**:* AT commands, or "Attention" commands, are the primary method for communication between the Quectel EC200U-CN 4G LTE module (integrated in the 7Semi EC200U-CN LTE 4G GPS GNSS Mini Modem) and its host controller. These commands allow hardware developers to instruct the modem to perform various functions, such as:

- Establishing network connectivity
- Sending data
- Retrieving status information

In this system, AT commands enable the module to:

- Transmit air quality sensor data (e.g., **PM2.5**, CO2, **temperature**) to a cloud server (e.g., ThingSpeak) via TCP
- Support GNSS for location tracking

*2) **How AT Commands Are Communicated**:* AT commands are single-line instructions sent to the EC200U-CN module in sequential order, typically requiring a response before the next command. Communication can occur via:

- UART (Universal Asynchronous Receiver/Transmitter):
  - A two-wire, half-duplex protocol commonly used for serial communication between the modem and a host microcontroller (e.g., ESP32, STM32).
  - Default baud rate: 115200 bps, configurable (e.g., 9600 bps).
- USB
  - Modern modems like the EC200U-CN support AT command transmission via USB for simplified integration.
- Network (Remote Diagnostics)
  - AT commands can be sent wirelessly over the network for remote configuration or diagnostics (less common in IoT setups).

In this system, the inbuilt Arduino sends AT commands via UART to the EC200U-CN to:

- Configure the module
- Establish a TCP connection
- Transmit sensor data
- The module responds with status messages (e.g., OK, ERROR) or data.

*3) **AT Command Syntax**:* AT commands are used to control the modem in the air quality monitoring system. The general syntax is expressed as:

`AT<COMMAND><SUFFIX><DATA>`

where the components are defined as follows:

- AT: The command prefix that initiates communication with the modem.
- <COMMAND>: A specific instruction, such as QIACT or QISEND, directing the modem to perform a particular task.
- <SUFFIX>: Indicates the type of command:
  - ?: Read command, used to retrieve current settings.
  - =?: Test command, used to check supported parameters.
  - =: Set command, used to configure parameters.
  - None: Execution command, used to perform an action directly.
- <DATA>: Optional parameters or data required for the command, such as values or addresses.

*4) **Categories of AT Commands**:* AT commands are categorized based on their functionality:

- **Test Commands** (AT+<COMMAND>=?): Check supported parameters. Example: AT+QIACT=?.
- **Read Commands** (AT+<COMMAND>?): Retrieve current settings. Example: AT+QIACT?.
- **Set Commands** (AT+<COMMAND>=<value>): Configure parameters. Example: AT+IPR=9600.
- **Execution Commands** (AT+<COMMAND>): Execute an action. Example: AT.

*5) **AT Commands Used in the Air Quality Monitoring System**:* The following AT commands are utilized to manage communication, internet connectivity, data transmission, and GNSS functionality in the system:

AT (EXECUTION COMMAND)

- **Purpose**: Tests communication with the modem.
- **Syntax**: AT
- **Response**:
  - OK: Module is operational.
  - No response/ERROR: Check baud rate or connection.
- **Usage**: Sent at startup to verify modem functionality.

AT+IPR=9600 (Set Command)

- **Purpose**: Sets the UART baud rate to 9600 bps.
- **Syntax**: AT+IPR=<rate>
- **Example**: AT+IPR=9600
- **Response**:
  - OK: Baud rate set.
  - ERROR: Invalid rate.
- **Usage**: Adjusts baud rate for compatibility; host switches to 9600 bps after.

AT+QIACT? (Read Command)

- **Purpose**: Checks PDP context activation (internet connection).
- **Syntax**: AT+QIACT?
- **Response**:
  - +QIACT: <cid>,<context_state>,<context_type>,
  - OK
- **Usage**: Ensures active internet connection before data transmission.

AT+QIOPEN=1,0,"TCP","api.thingspeak.com",80,0,0 (Set Command)

- **Purpose**: Opens a TCP connection to ThingSpeak.
- **Syntax**:
- AT+QIOPEN=<contextID>,<connectID>,<service_type
- **Example**:
- AT+QIOPEN=1,0,"TCP","api.thingspeak.com",80,0,0
- **Response**:
  - OK
  - +QIOPEN: 0,0
- **Usage**: Initiates connection to ThingSpeak for data upload.

AT+QISEND=0,70 (Set Command)

- **Purpose**: Prepares to send 70 bytes over TCP.
- **Syntax**: AT+QISEND=<connectID>,<length>
- **Example**: AT+QISEND=0,70
- **Response**:
  - >: Ready for data.
  - ERROR: Invalid ID or connection.
- **Usage**: Sets up data transmission to ThingSpeak.

Data Sent

- **Purpose**: Sends a GET request after the > prompt from AT+QISEND.
- **Example Format**: GET /update?api_key=XYZ
- **Response**:
  - SEND OK: Data sent successfully.
  - SEND FAIL: Check network or formatting.
- **Usage**: Ensures ThingSpeak API receives the GET request correctly.

AT+QIRD=0,100 (Set Command)

- **Purpose**: Reads up to 100 bytes from the response buffer.
- **Syntax**: AT+QIRD=<connectID>,<length>
- **Example**: AT+QIRD=0,100
- **Response**:
  - +QIRD: <length>
  - <data>
  - OK
- **Usage**: Confirms successful data upload to ThingSpeak.

AT+QICLOSE=0 (Set Command)

- **Purpose**: Closes the TCP connection.
- **Syntax**: AT+QICLOSE=<connectID>
- **Example**: AT+QICLOSE=0
- **Response**:
  - OK: Connection closed.

  - `ERROR`: Invalid ID.
- **Usage**: Frees resources after data transmission.

AT+QGPS=1 (Set Command)

- **Purpose**: Starts the GNSS engine.
- **Syntax**: `AT+QGPS=1`
- **Response**:
  - `OK`: GNSS started.
  - `ERROR`: Already running or hardware issue.
- **Usage**: Enables GPS tracking for location data.

AT+QGPSLOC=1 (Execution Command)

- **Purpose**: Retrieves current GPS location.
- **Syntax**: `AT+QGPSLOC=1`
- **Response**:
  - `+QGPSLOC: <latitude>,<longitude>, <HDOP>,<altitude>,<fix>,<cog>, <spkm>,<spkn>,<date>,<utc>`
  - `OK`
- **Description**: Fetches coordinates, altitude, speed, and time after GNSS activation.
- **Usage**: Tags air quality data with precise location.

AT+QGPSEND (Execution Command)

- **Purpose**: Stops the GNSS engine to save power.
- **Syntax**: `AT+QGPSEND`
- **Response**:
  - `OK`: GNSS stopped.
  - `ERROR`: GNSS not running.
- **Usage**: Turns off GNSS after location retrieval to extend battery life.

C. Filtering and Visualizing ThingSpeak Data in Google Colab

*1) **System Overview**:* To monitor and analyze air quality data from our sensors, we imported live readings from our ThingSpeak channel into Google Colab. This setup allowed us to process, clean, and visualize the data effectively using Python's powerful data handling and plotting libraries, making it accessible for real-time environmental monitoring.

*2) **Data Retrieval**:* Sensor data is accessed from ThingSpeak via its Read API, which delivers the latest channel entries in JSON format. The data includes five key parameters: **temperature**, **relative humidity**, **PM1**, **PM2.5**, and **PM10**, each mapped to a specific field (field1 to field5) in the ThingSpeak channel.

Fig. 10. Data received on thingspeak.com

*3) **Handling Inconsistent Sensor Readings**:* During testing, sensors occasionally produced zero values due to signal losses or initialization delays, which distorted graph scaling and misled visualization. To address this, we replaced zero values with the most recent valid reading or the next valid value if the previous was invalid. This cleaning approach ensured continuous, realistic data trends without altering the actual environmental patterns.

*4) **Visualization**:* After cleaning, we used an interactive plotting library to create live, real-time graphs for all five parameters, each displayed in a separate subplot aligned on a shared time axis. This visualization enabled us to monitor environmental changes, verify data transmission and sensor stability, and identify trends or fluctuations. The plots are refreshed regularly, acting like a live dashboard for up-to-date monitoring.

Fig. 11. Data after filtering in Google Colab

*5) **Further Capabilities**:* The system's integration with Python's ecosystem allows for future enhancements. Statistical and machine learning libraries like scikit-learn or prophet can enable predictive analysis, while anomaly detection and automated alerts can provide real-

time responses to pollution spikes or system failures, enhancing the system's utility.

V. Testing and Validation

A. *Hardware Testing*

The EC200U-CNAA modem was tested for SIM connectivity and TCP/IP transmission using AT commands, with loose Type-C connectors re-soldered for reliability. The DFRobot Air Quality Monitor's data output was verified via serial monitor and cross-checked with a calibrated monitor. The power system, including the 12V solar panel and Waveshare Solar Power Manager, was confirmed compatible, with the battery providing 15 hours of autonomy without sunlight.

B. *Validation*

Data accuracy was validated at $\pm 10\%$ for **PM2.5** and $\pm 5\%$ for **temperature** and **humidity**. Transmission reliability reached 95% success with optimized 5-minute intervals. The power system sustained operation with adequate solar recharging, ensuring stability.

VI. Results

The system was successfully deployed and tested locally, recording data in CSV format for offline analysis and streaming it live to ThingSpeak, visualized in Google Colab. It operated stably for 15 hours without sunlight, demonstrating effective battery backup. With 5–6 hours of daily sunlight, the solar-powered system functioned reliably day and night. Over a week, it consistently captured and transmitted data without interruption, validating the robustness and reliability of both the hardware and communication pipeline under real-world environmental conditions. The GPS module accurately reported the node's location throughout testing.

created_at	entry_id	Temp	Rh	PM1	PM2.5	PM10	latitude	longitude
2025-04-08T21:37:40+05:30	1	27.66	50.99	71	121	137	26.19	91.7
2025-04-08T21:38:47+05:30	2	27.66	50.99	69	110	127	26.19	91.7
2025-04-08T21:39:56+05:30	3	27.66	50.99	66	113	125	26.19	91.7
2025-04-08T21:40:28+05:30	4	27.66	50.99	73	125	137	26.19	91.7
2025-04-08T21:41:01+05:30	5	27.66	50.99	68	114	121	26.19	91.7
2025-04-08T21:42:39+05:30	6	27.66	50.88	69	117	129	26.19	91.7
2025-04-08T21:44:31+05:30	7	27.66	50.88	70	117	130	26.19	91.7
2025-04-08T21:44:57+05:30	8	27.66	50.88	67	113	127	26.19	91.7
2025-04-08T21:45:30+05:30	9	27.66	50.88	69	123	136	26.19	91.7
2025-04-08T21:46:02+05:30	10	27.66	50.77	69	113	127	26.19	91.7
2025-04-08T21:46:34+05:30	11	0	0	0	0	0	26.19	91.7
2025-04-08T21:47:07+05:30	12	27.55	50.88	71	121	135	26.19	91.7
2025-04-08T21:47:40+05:30	13	27.55	50.77	69	115	127	26.19	91.7
2025-04-08T21:48:12+05:30	14	27.55	50.77	68	115	131	26.19	91.7
2025-04-08T21:48:46+05:30	15	27.55	50.88	67	117	129	26.19	91.7
2025-04-08T21:49:18+05:30	16	27.55	50.77	65	115	127	26.19	91.7
2025-04-08T21:51:03+05:30	17	27.55	50.66	67	117	125	26.19	91.7
2025-04-08T21:51:36+05:30	18	27.55	50.66	69	119	131	26.19	91.7
2025-04-08T21:52:08+05:30	19	27.55	50.66	67	113	124	26.19	91.7
2025-04-08T21:52:41+05:30	20	27.55	50.66	69	122	137	26.19	91.7
2025-04-08T21:53:12+05:30	21	27.55	50.66	69	117	132	26.19	91.7
2025-04-08T21:53:48+05:30	22	27.55	50.55	71	121	135	26.19	91.7
2025-04-08T21:54:17+05:30	23	27.55	50.55	70	119	134	26.19	91.7
2025-04-08T21:55:24+05:30	24	27.55	50.44	73	120	129	26.19	91.7
2025-04-08T21:55:58+05:30	25	27.55	50.33	69	121	135	26.19	91.7
2025-04-08T22:00:43+05:30	26	27.66	50	68	114	123	26.19	91.7

Fig. 12. Air quality data recorded in CSV format.

VII. Future Development

To enhance the system, we plan to scale up by deploying more nodes and designing a custom PCB. Edge computing and machine learning will be implemented for predictive analysis. Power efficiency will be improved with better batteries and panels, and additional sensors for VOC/CO2 will be integrated. A mobile app and enhanced ThingSpeak dashboards will improve user interaction.

References

[1] Y. Zhang et al., "Air pollution and health impacts in India," *Environ. Sci. Technol.*, vol. 52, no. 5, pp. 2456–2464, 2018.

[2] R. Kumar et al., "IoT-based air quality monitoring systems," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1890–1900, 2020.

[3] X. Li et al., "Solar-powered IoT systems for environmental monitoring," *Renew. Energy*, vol. 134, pp. 567–575, 2019.

[4] S. Patel et al., "Low-cost sensor networks for air quality," *J. IoT Appl.*, vol. 3, no. 2, pp. 45–53, 2021.

[5] Z. Wang et al., "Advances in air quality sensors," *Sensors*, vol. 22, no. 4, p. 1234, 2022.

[6] *Quectel EC200U Series Hardware Design Manual*, Quectel Wireless Solutions, 2023.

[7] *Waveshare Solar Power Manager User Manual*, Waveshare Electronics, 2024.

[8] "An introduction to cellular AT commands," Cavli Wirelesshttps://www.cavliwireless.com/blog/nerdiest-of-things/an-introduction-to-cellular-at-commands, 2023.

[9] "Air Pollution in India: Sources and Impacts," Environ. Res. Inst., 2021.

[10] *State of Global Air 2024*, Health Effects Institute, 2024.