PROJECT REPORT
ON
IoT Based Air Quality Monitoring using CAN Protocol

Carried Out at

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING
ELECTRONIC CITY, BANGALORE.

UNDER THE SUPERVISION OF

Mr. Pavan Vilas Jadhav
Project Engineer
C-DAC Bangalore

Submitted By
Akhilesh Manoj Lad (248500130004)
Ayushi Shukla (248500130015)
Gundeti Sai Kiran (240850130019)
Samala Lokesh (248500130027)
Shivtej Eknath Patil (248500130028)

PG DIPLOMA IN EMBEDDED SYSTEMS & DESIGN
C-DAC, BANGALORE

Candidate's Declaration

We hereby certify that the work being presented in the report entitled **"IoT Based Air Quality Monitoring System using CAN protocol"**, in the partial fulfillment of the requirements for the degree of Post Graduation Diploma and submitted in the department of Embedded Systems and Design of the C-DAC Bangalore, is an authentic record of our work carried out during the period 25th Nov 2024 – 12th Feb 2025 under the supervision of "**Mr. Pavan Vilas Jadhav**", C-DAC Bangalore.

The matter presented in the report has not been submitted by me for the award of any degree of this or any other Institute/University.

(Names of Candidates)

Akhilesh Manoj Lad (248500130004)

Ayushi Shukla (248500130015)

Gundeti Sai Kiran (240850130019)

Samala Lokesh (248500130027)

Shivtej Eknath Patil (248500130028)

Counter Signed by

Mr. Pavan Vilas Jadhav

Project Engineer

ACKNOWLEDGMENT

I take this opportunity to express my gratitude to all those people who have been directly and indirectly with me during the completion of this project.

I pay thank to "**Mr. Pavan Vilas Jadhav**" who has given guidance and a light to me during this major project. His versatile knowledge about "**IoT Based Air Quality Monitoring System using CAN protocol**" has eased us in the critical times during the span of this Final Project.

I acknowledge here out debt to those who contributed significantly to one or more steps. I take full responsibility for any remaining sins of omission and commission.

Akhilesh Manoj Lad (248500130004)

Ayushi Shukla (248500130015)

Gundeti Sai Kiran (240850130019)

Samala Lokesh (248500130027)

Shivtej Eknath Patil (248500130028)

ABSTRACT

Air pollution has become a significant concern affecting human health and the environment. To address this issue, we propose an **IoT-based Air Quality Monitoring System** utilizing the **Controller Area Network (CAN) protocol** for reliable and efficient data transmission. The system comprises **two ESP32 microcontrollers**, where the first ESP32 collects real-time air quality data using multiple sensors, including the **Nova PM SDS011** (for particulate matter), **DHT22** (for temperature and humidity), and **MQ135** (for detecting harmful gases). The collected data is displayed on a **0.96-inch OLED screen** and transmitted via the **MCP2515 CAN module** to the second ESP32.

The second ESP32 receives the data over CAN communication, processes it, and sends it to **ThingSpeak**, a cloud-based IoT analytics platform, for remote monitoring and analysis. This system ensures **low-latency, robust, and interference-free communication** using the CAN protocol, making it suitable for **industrial and environmental applications**. The proposed system provides **real-time monitoring, remote accessibility, and improved data reliability**, making it a cost-effective solution for air quality assessment.

Table of Contents

Table of Figures

Chapter -1: Project Overview

1.1 Introduction

Air pollution is one of the most pressing environmental challenges affecting human health, ecosystems, and climate stability. With rapid urbanization, industrialization, and an increase in vehicular emissions, air quality has deteriorated significantly, leading to severe health issues such as respiratory diseases, cardiovascular problems, and allergies. Harmful pollutants such as particulate matter (PM2.5 and PM10), carbon dioxide (CO_2), ammonia (NH_3), and volatile organic compounds (VOCs) contribute to air pollution, making it essential to monitor air quality in real-time. Traditional air quality monitoring systems are often expensive, complex, and limited in accessibility, making them unsuitable for widespread use. Hence, the need for a cost-effective, reliable, and scalable air quality monitoring system has become more crucial than ever.

This project, **"IoT-Based Air Quality Monitoring System using CAN Protocol",** is designed to address this issue by leveraging modern embedded systems, sensor technology, and IoT. The system consists of two ESP32 microcontrollers, where one acts as a Sensor Node and the other as a Receiver Node. The Sensor Node integrates multiple air quality sensors Nova PM Sensor SDS011, DHT22, and MQ135 to measure critical environmental parameters such as particulate matter levels (PM2.5, PM10), temperature, humidity, and gas concentrations. The collected data is displayed on a 0.96-inch OLED screen for immediate local monitoring and is simultaneously transmitted to the Receiver Node using the Controller Area Network (CAN) protocol. The Receiver Node receives and processes the data before sending it to ThingSpeak, a cloud-based IoT analytics platform, for remote monitoring.

The CAN protocol is chosen for data communication due to its reliability, robustness, and resistance to electrical noise, making it ideal for real-time industrial applications. Unlike traditional wireless communication methods such as Wi-Fi or Bluetooth, CAN is designed for efficient, error-free, and high-speed data transmission, which is crucial for applications requiring accuracy and real-time monitoring. This makes the system highly suitable for smart cities, industrial environments, hospitals, and residential air quality monitoring applications.

One of the key features of this project is its scalability and modularity. Additional sensor nodes can be integrated into the system to monitor air quality across different locations within an area, creating a distributed sensor network. This approach allows for broader environmental monitoring and analysis, which can help authorities take data-driven decisions to implement pollution control measures.

Furthermore, by integrating IoT capabilities with ThingSpeak, users can access real-time air quality data from anywhere using a web browser or mobile device. This enables individuals, researchers, and government agencies to monitor air quality trends over time, set threshold alerts, and take necessary actions to mitigate pollution.

The system is designed to be energy-efficient and cost-effective, making it suitable for continuous operation in both indoor and outdoor environments. By utilizing the ESP32 microcontroller, which has built-in Wi-Fi and low-power operation modes, the system minimizes power consumption while ensuring efficient data processing and transmission.

1.2 Objective

The primary objective of this project, **"IoT-Based Air Quality Monitoring System using CAN Protocol",** is to design and implement a real-time air quality monitoring system that efficiently measures, transmits, and visualizes environmental parameters. This system aims to provide accurate and continuous monitoring of air pollution levels using embedded systems, sensor technology, and IoT-based cloud integration.

Specific Objectives:

1. **Air Quality Data Collection:**
    - Integrate **Nova PM SDS011**, **DHT22**, and **MQ135** sensors with an ESP32 microcontroller.
    - Measure key environmental parameters such as PM2.5, PM10, temperature, humidity, and harmful gas concentrations (CO_2, NH_3, VOCs, etc.).
2. **Local Data Display and Processing:**
    - Display real-time sensor readings on a **0.96-inch OLED screen** for immediate visualization.
    - Process sensor data efficiently using the ESP32 microcontroller.
3. **Data Transmission Using CAN Protocol:**
    - Implement Controller Area Network (CAN) communication using **MCP2515 CAN modules** to transmit data from the Sensor Node to the Receiver Node.

- Ensure reliable, noise-resistant, and efficient communication between nodes.

4. **Cloud-Based Remote Monitoring:**
    - Connect the Receiver Node to **ThingSpeak** via Wi-Fi to upload sensor data for remote access.
    - Enable real-time monitoring through a web dashboard accessible from any device.
5. **Scalability and Adaptability:**
    - Develop a system that supports **multiple sensor nodes** for broader air quality monitoring.
    - Ensure low-power operation and cost-effectiveness for widespread deployment.

By achieving these objectives, this project provides an **effective and scalable solution** for air quality monitoring, helping individuals, industries, and governments make data-driven decisions for pollution control and environmental protection.

1.3 System Architecture & Components

The **IoT-Based Air Quality Monitoring System using CAN Protocol** is designed with a distributed two-node architecture to efficiently collect, transmit, and display real-time air quality data. The system consists of two ESP32 microcontroller-based nodes:

1. **Sensor Node** – Collects air quality data from various sensors and transmits it using the CAN protocol.
2. **Receiver Node** – Receives the sensor data, processes it, and uploads it to ThingSpeak for remote monitoring.

This architecture ensures efficient, reliable, and noise-resistant communication while allowing scalability for future expansion.

➢ **<u>Components of the System Architecture</u>**

✓ ***Sensor Node (ESP32 + Sensors + CAN Transmitter)***

The Sensor Node is responsible for collecting air quality parameters and transmitting the data using the MCP2515 CAN module. It consists of:

- **ESP32 Microcontroller** – Manages sensor data processing and CAN communication.
- **Nova PM Sensor SDS011** – Measures PM2.5 and PM10 particulate matter concentrations.
- **DHT22 Sensor** – Captures temperature and humidity data.

- **MQ135 Gas Sensor** – Detects harmful gases such as CO_2, NH_3, benzene, and VOCs.
- **0.96-inch OLED Display** – Displays real-time air quality readings for local monitoring.
- **MCP2515 CAN Controller**– Converts sensor data into CAN messages and transmits them to the Receiver Node.

✓ **Receiver Node (ESP32 + CAN Receiver + IoT Integration)**

The Receiver Node is responsible for receiving data via CAN, processing it, and sending it to the cloud. It consists of:

- **ESP32 Microcontroller** – Decodes received CAN messages and forwards the data to the cloud.
- **MCP2515 CAN Controller** – Receives sensor data from the Sensor Node.
- **Wi-Fi Module (Built-in ESP32)** – Connects to the internet and uploads data to ThingSpeak.

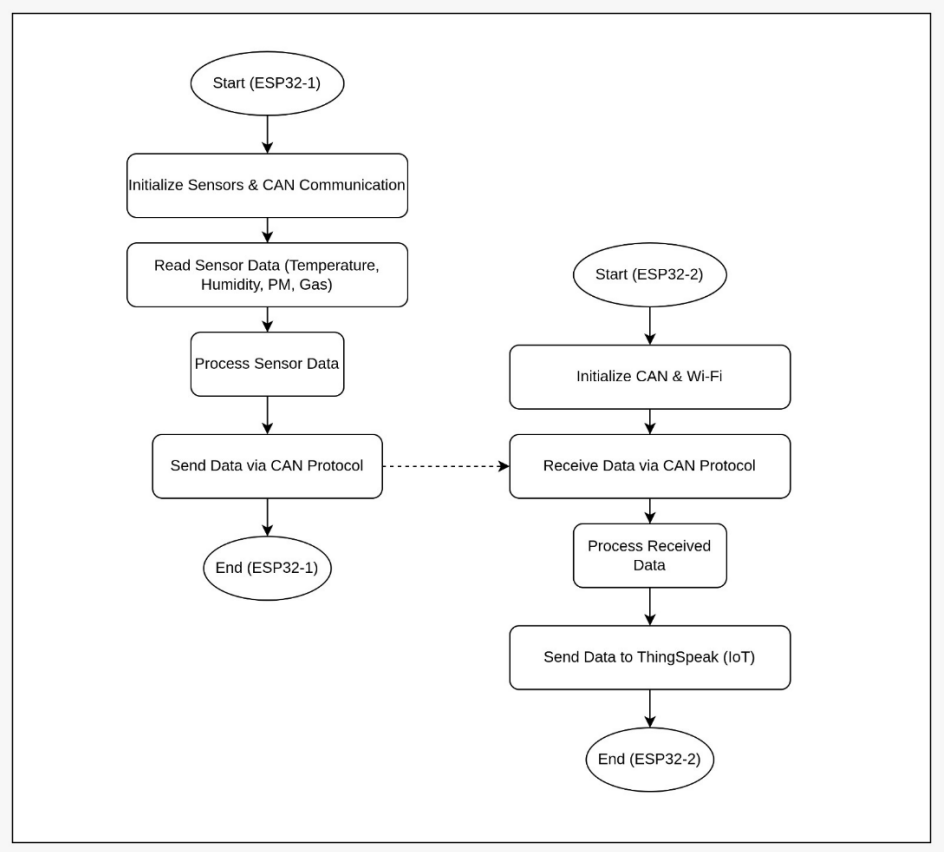

Figure 1:System Architecture

➢ Block Diagram

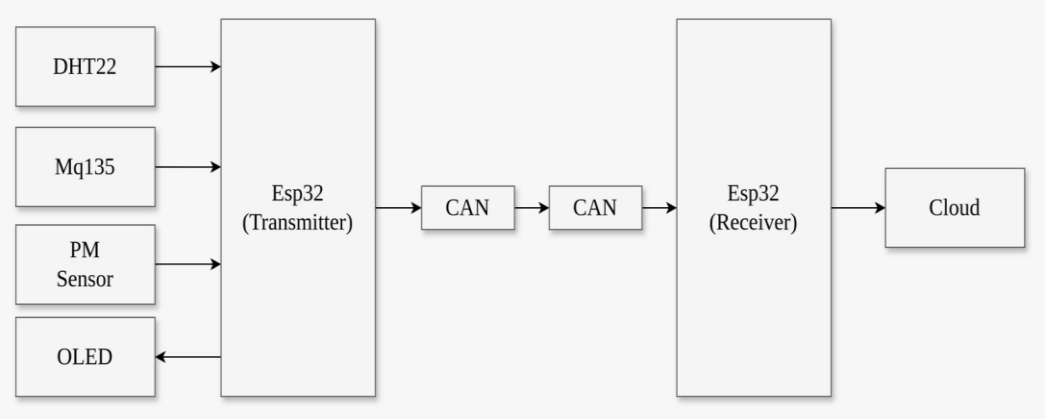

Figure 2: Block Diagram

➢ Circuit Diagram

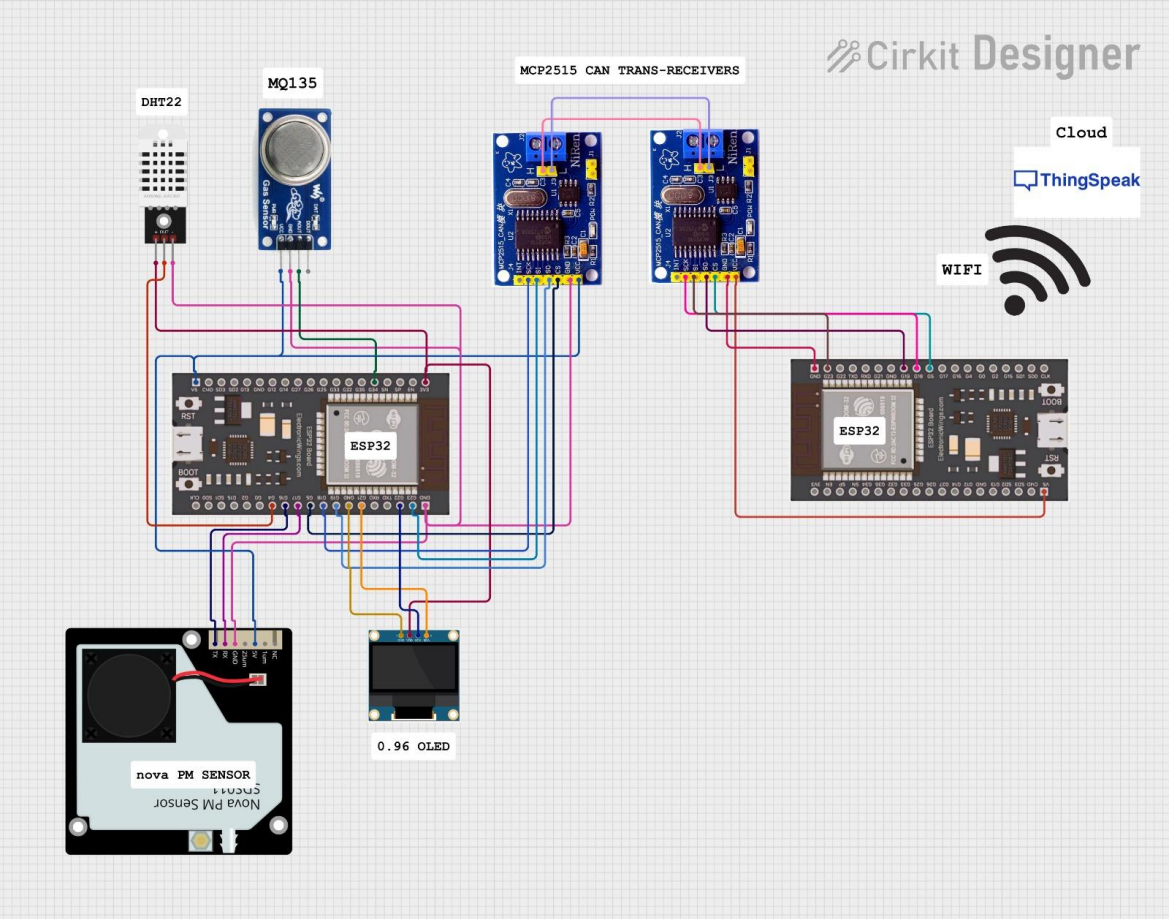

Figure 3: Circuit Diagram

➢ Data Communication Flow

The data flow in the system follows these steps:

- ✓ **Data Acquisition:**
  - The Sensor Node reads air quality parameters from the SDS011, DHT22, and MQ135 sensors.
  - The ESP32 processes the sensor readings and formats them into a structured message.
- ✓ **Data Transmission via CAN Protocol:**
  - The ESP32 sends sensor data to the MCP2515 CAN module, which converts the data into CAN messages.
- ✓ **Data Reception at Receiver Node:**
  - The Receiver Node, equipped with another MCP2515 CAN module, receives the transmitted data.
  - The ESP32 extracts sensor values from the received CAN message and processes them.
- ✓ **Cloud Upload for IoT Monitoring:**
  - The Receiver Node connects to the ThingSpeak cloud platform via Wi-Fi.
  - The processed sensor data is uploaded to ThingSpeak, enabling remote monitoring via web dashboards.

1.4 Working Principle

The **IoT-Based Air Quality Monitoring System using CAN Protocol** operates through a structured data collection, processing, transmission, and cloud integration approach. The system consists of two main nodes Sensor Node and Receiver Node which communicate using the Controller Area Network (CAN) protocol. The Sensor Node collects environmental data from various sensors, transmits it via CAN, and the Receiver Node processes this data before sending it to the ThingSpeak cloud platform for remote monitoring.

- ➢ **Sensor Node Operation**

- ✓ **Sensor Data Collection:**
  - The Nova PM SDS011 sensor measures PM2.5 and PM10 particulate matter concentrations in the air.
  - The DHT22 sensor records temperature and humidity levels.

- The MQ135 gas sensor detects air pollutants such as CO_2, NH_3, benzene, and VOCs.

✓ **Data Processing:**

- The ESP32 reads the analog and digital outputs from the sensors.
- The sensor values are converted into readable digital data.
- The formatted sensor readings are displayed on a 0.96-inch OLED screen for local monitoring.

✓ **Data Transmission via CAN Protocol:**

- The ESP32 sends the collected data to the MCP2515 CAN Controller, which converts the sensor readings into CAN messages.
- The MCP2515 CAN transceiver transmits these messages over the CAN bus to the Receiver Node.
- Each CAN message consists of a unique identifier, sensor values, and error-checking bits to ensure reliable transmission.

➢ **<u>Receiver Node Operation</u>**

✓ **Receiving CAN Data:**

- The Receiver Node consists of another ESP32 microcontroller and an MCP2515 CAN module.
- The MCP2515 transceiver receives the transmitted CAN messages from the Sensor Node.
- The ESP32 extracts the sensor values from the CAN frame and processes them.

✓ **Data Processing & Validation:**

- The ESP32 checks for error bits and message integrity to ensure accurate data reception.
- If errors are detected, the system requests retransmission using built-in CAN error-handling mechanisms.

✓ **Wi-Fi-Based Cloud Transmission:**

- The Receiver Node connects to the internet via Wi-Fi (built-in ESP32 module).
- Sensor values are formatted into HTTP requests and sent to ThingSpeak.
- The ThingSpeak platform processes and displays the real-time air quality data in graphical form, accessible from any web browser or mobile device.

1.5 Key Features

- ✓ **Multi-Sensor Integration:**
  - Uses Nova PM SDS011, DHT22, and MQ135 sensors to measure PM2.5, PM10, temperature, humidity, and gas concentrations (CO_2, NH_3, VOCs, etc.).
- ✓ **Real-Time Data Monitoring:**
  - Displays sensor readings on a 0.96-inch OLED screen for instant visualization.
  - Sends data to ThingSpeak cloud for remote monitoring.
- ✓ **Reliable CAN Communication:**
  - Implements Controller Area Network (CAN) protocol for noise-resistant, efficient, and real-time data transmission.
  - Uses MCP2515 CAN controller & TJA1050 transceiver for robust connectivity.
- ✓ **IoT & Cloud Integration:**
  - ESP32 uploads air quality data to ThingSpeak, allowing remote access via a web dashboard.
  - Users can analyse historical trends and receive alerts for air pollution levels.
- ✓ **Scalability & Modularity:**
  - Additional sensor nodes can be integrated to cover larger monitoring areas.
  - Supports expansion for smart city & industrial applications

1.6 Advantages

- **High Reliability:** CAN communication ensures error-free data transmission even in noisy environments.
- **Low Power Consumption:** ESP32 operates efficiently, making the system suitable for continuous monitoring**.**
- **Cost-Effective Solution:** Uses affordable components, making it budget-friendly for deployment in homes, industries, and cities**.**
- **Remote Accessibility:** Cloud integration allows users to monitor air quality from anywhere**.**
- **Environmental Benefits:** Helps authorities track pollution levels and implement better control measures.

1.7 Applications

- ✓ **Smart City Air Quality Monitoring**
  - **Urban Pollution Control:** City authorities can deploy multiple sensor nodes across different locations to monitor air pollution levels and take preventive actions.
  - **Traffic-Related Pollution Analysis:** Helps track vehicle emissions by placing monitoring units near roads and highways.
  - **Public Awareness:** Real-time air quality data can be displayed on public screens or mobile applications to inform citizens about pollution levels.
- ✓ **Industrial Air Quality Monitoring**
  - **Workplace Safety:** Industries that emit harmful gases (e.g., chemical plants, factories, mining sites) can use this system to ensure air quality remains within safe limits.
  - **Regulatory Compliance:** Helps industries comply with government environmental regulations by providing continuous air quality reports.
  - **Hazardous Gas Detection:** Detects harmful gases like NH_3, CO_2, VOCs, preventing workplace accidents.

- ✓ **Healthcare & Indoor Air Quality Monitoring**
  - **Hospitals & Clinics:** Ensures clean air for patients, especially those with respiratory diseases such as asthma.
  - **Homes & Offices:** Monitors indoor air quality in smart homes and workplaces, alerting users to pollutants.
  - **Allergy Prevention:** Helps people with allergies and respiratory conditions avoid exposure to harmful airborne particles.
- ✓ **Agricultural & Greenhouse Monitoring**
  - **Crop Protection:** Ensures the air quality in greenhouses and farms remains optimal for plant growth.
  - **Livestock Health:** Monitors air pollution in livestock farms to reduce the impact of toxic gases (NH_3, CO_2) on animal health.
- ✓ **IoT-Based Research & Environmental Studies**

- **Climate Change Research:** Scientists can analyse long-term air quality trends to study the effects of pollution on climate change.
- **University & Educational Use:** Used in engineering and environmental science programs for research projects and hands-on learning.
- **Forest Fire Detection:** Helps detect sudden changes in air quality due to smoke and CO_2 emissions from wildfires.

✓ **Smart Transportation Systems**

- **Public Transport Safety:** Monitors air quality inside buses, trains, and airports, ensuring passenger comfort.
- **Tunnels & Underground Systems:** Tracks air pollution in subways and tunnels, where ventilation is limited.

✓ **Real-Time Disaster Monitoring & Early Warning Systems**

- **Gas Leak Detection:** Helps detect gas leaks in oil refineries, factories, and residential areas to prevent explosions.
- **Forest & Industrial Fire Monitoring:** Identifies hazardous air conditions caused by industrial fires or forest fires, sending alerts in real time.

1.8 Advantages of Using CAN Protocol in IoT

- **High Reliability**: Handles noisy environments effectively.
- **Error Detection**: Built-in error correction ensures accurate data transmission.
- **Long-Distance Communication**: CAN can communicate over several meters without interference.
- **Multi-Node Communication**: Allows multiple sensors/nodes to be integrated into the system.

1.9 Conclusion

This IoT-based Air Quality Monitoring System provides an efficient, reliable, and real-time solution for monitoring air pollution. By integrating CAN communication, the system ensures stable and interference-free data transmission, making it suitable for industrial, urban, and indoor air quality applications. The cloud-based monitoring via ThingSpeak enables remote access and data analysis, helping users track pollution levels and make informed decisions.

Chapter -2: Literature Survey

2.1 Introduction to Air Quality Monitoring

Air quality monitoring is crucial for assessing pollution levels and ensuring a healthy environment. With increasing urbanization, industrial activities, and indoor pollution sources, monitoring air quality has become essential to mitigate health risks and environmental damage.

Importance of Air Quality Monitoring:

- Urban Environments:

In cities, vehicular emissions, construction activities, and industrial processes release pollutants such as particulate matter (PM), nitrogen oxides (NOx), sulfur dioxide (SO_2), and carbon monoxide (CO). Monitoring air quality in urban areas helps in:

- Reducing health risks from exposure to pollutants.
- Implementing effective traffic and industrial emission control policies.
- Ensuring compliance with environmental regulations.

- Industrial Environments:

Factories and power plants emit toxic gases and particulate matter, leading to occupational health hazards and environmental degradation. Air quality monitoring in industrial areas is essential for:

- Protecting workers from exposure to hazardous substances.
- Ensuring industries adhere to emission norms.
- Preventing long-term environmental contamination.

- Indoor Environments:

Indoor air pollution, often overlooked, can be caused by poor ventilation, household chemicals, and combustion sources. Common pollutants include volatile organic compounds (VOCs), CO, and PM. Monitoring indoor air quality helps in:

- Reducing respiratory diseases like asthma.
- Enhancing indoor comfort and productivity.
- Identifying sources of pollution for mitigation.

Impact of Air Pollution

- Health Effects:

Air pollution is a leading cause of respiratory diseases, cardiovascular conditions, and even neurological disorders. Fine particulate matter (PM2.5) can penetrate deep into the lungs and enter the bloodstream, causing:

- Asthma and lung infections.
- Increased risk of heart attacks and strokes.
- Adverse effects on brain function and cognitive development in children.

- Environmental Effects:

Air pollution leads to acid rain, soil degradation, and reduced agricultural yields. High levels of ozone (O_3) damage vegetation, while PM and heavy metals contaminate water sources, affecting ecosystems.

- Climate Change:

Greenhouse gases (GHGs) such as CO_2 and methane (CH_4) contribute to global warming, while black carbon accelerates glacier melting. Monitoring air pollutants helps in tracking emissions and formulating climate mitigation policies.

Air Quality Standards

Various organizations have established air quality guidelines to define safe exposure levels:

- World Health Organization (WHO): Sets global air quality guidelines for PM2.5, PM10, NO_2, SO_2, and CO.
- Environmental Protection Agency (EPA - USA): Defines National Ambient Air Quality Standards (NAAQS) for six major pollutants.
- Central Pollution Control Board (CPCB - India): Sets national air quality standards and monitors pollution levels in Indian cities.

Need for Real-Time Air Quality Monitoring

[1]Traditional air monitoring methods rely on manual sampling and laboratory analysis, which are time-consuming and provide delayed results. Real-time monitoring using IoT-based sensors offers:

- Instant data on pollution levels.
- Early warnings for hazardous conditions.
- Better policy implementation through data-driven decisions.

2.2 Existing Types of Air Quality Monitoring Systems

Air quality monitoring systems have evolved over time, from traditional methods requiring manual sampling and laboratory analysis to modern IoT-based solutions that provide real-time data. Each approach has its advantages and limitations, influencing its effectiveness in various applications.

➢ **Traditional Air Quality Monitoring Systems**

[2]Traditional air quality monitoring systems rely on fixed monitoring stations and laboratory analysis. These methods have been the foundation of regulatory air quality assessments for decades.

❖ **Conventional Methods**

1. **Manual Air Sampling**
    - Air samples are collected using filters, adsorption tubes, or impingers.
    - The samples are analyzed in laboratories to determine pollutant concentrations.
    - Common methods include gravimetric analysis for particulate matter (PM), gas chromatography for VOCs, and spectrophotometry for gaseous pollutants.
2. **Government-Run Monitoring Stations**
    - Many countries operate continuous ambient air quality monitoring stations **(CAAQMS)** managed by environmental agencies.
    - In India, the Central Pollution Control Board **(CPCB)** runs a nationwide network of monitoring stations that measure pollutants like **PM2.5, PM10, NO_2, SO_2, CO, O_3, and NH_3**.
    - The U.S. Environmental Protection Agency **(EPA)** operates the **Air Quality System (AQS)** to monitor and regulate pollution levels.

Limitations of Traditional Systems

- **High Cost**: Setting up and maintaining monitoring stations requires expensive equipment and skilled personnel.
- **Limited Mobility**: Fixed stations cannot provide localized air quality data for specific locations.
- **Lack of Real-Time Data**: Manual sampling and lab analysis cause delays in reporting pollution levels.
- **Sparse Coverage**: Monitoring stations are often placed far apart, missing localized pollution sources.

- **IoT-Based Air Quality Monitoring Systems**

[3]IoT-based air quality monitoring systems utilize smart sensors, wireless communication, and cloud platforms to collect and analyze real-time air quality data.

Advantages of IoT-Based Systems

1. **Real-Time Data Collection**: Sensors continuously measure air pollutants and transmit data without delays.
2. **Low-Cost and Scalable**: Portable and low-power sensor nodes can be deployed in multiple locations.
3. **Remote Monitoring**: Data can be accessed through cloud platforms from anywhere.
4. **Automated Alerts**: IoT systems can trigger alerts when pollution levels exceed safe limits.

Examples of IoT-Based Air Quality Monitoring Projects

1. **Wi-Fi-Based Systems**: [4]Devices like ESP32 and Raspberry Pi connect air quality sensors to cloud platforms via Wi-Fi.
2. **LoRa-Based Systems**: [5]Long-range, low-power communication is used for monitoring air quality in large areas (e.g., smart cities).
3. **GSM-Based Systems**: [6]Cellular networks enable real-time data transmission in remote locations where Wi-Fi is unavailable.

Cloud Platforms for Air Quality Monitoring

- **ThingSpeak**: Open-source IoT analytics platform used for storing and visualizing sensor data.
- **Blynk**: Mobile app-based IoT platform that enables real-time monitoring and control.
- **AWS IoT**: Cloud-based solution by Amazon for large-scale IoT deployments with AI-driven analytics.

Challenges in IoT-Based Air Quality Monitoring

- **High Power Consumption**: Continuous data transmission and sensor operation drain battery-powered devices quickly.
- **Network Limitations**: IoT devices depend on stable internet connections (Wi-Fi, LoRa, or GSM), which may not always be available.
- **Data Reliability**: Low-cost sensors may have accuracy issues compared to reference-grade monitoring systems.

2.3 Existing System

Traditional air quality monitoring systems rely on manual air sampling and laboratory analysis, which require specialized equipment, trained personnel, and significant time**,** making them impractical for real-time monitoring. Government-operated air quality monitoring stations, such as those managed by the Central Pollution Control Board (CPCB) in India**,** provide accurate but highly localized data, limiting their accessibility for broader population groups. In recent years, IoT-based air quality monitoring systems have been introduced using communication technologies like Wi-Fi, GSM, LoRa, and Bluetooth, allowing remote data access and real-time updates. However, Wi-Fi-based systems suffer from range limitations and network instability**,** GSM-based systems require SIM cards and incur high operational costs**,** LoRa-based solutions offer long-range communication but are expensive, and Bluetooth-based systems are short-range and unsuitable for cloud integration. These existing systems also face scalability challenges, high power consumption, and potential data transmission failures in harsh environmental conditions, making them less reliable for large-scale deployment.

2.4 Proposed System

To overcome these limitations, our proposed IoT-based Air Quality Monitoring System using CAN protocol provides a cost-effective, reliable, and power-efficient solution for real-time air pollution monitoring. The system consists of two ESP32 microcontrollers**,** where the first node (sensor node) collects air quality data using sensors such as Nova PM SDS011 (PM2.5 & PM10), MQ135 (CO_2, NH_3, Benzene), and DHT22 (Temperature & Humidity). The acquired data is displayed on a 0.96-inch OLED screen for local monitoring and transmitted via MCP2515 CAN module to a second ESP32 (gateway node). This gateway node processes the received data and uploads it to ThingSpeak**,** enabling remote access and real-time visualization**.** The CAN protocol ensures interference-free, collision-free, and reliable multi-node communication, making it superior to Wi-Fi and GSM-based solutions by reducing power consumption and improving stability. The system is scalable and adaptable, allowing additional sensor nodes to be incorporated into the CAN network without affecting performance, making it suitable for smart city applications, industrial monitoring, and environmental research**.**

2.5 Future Scope

The proposed system can be further enhanced by integrating additional gas sensors such as MQ7 (Carbon Monoxide), NO_2, SO_2, and O_3 sensors to provide more comprehensive air quality data, making it applicable for industrial emissions monitoring and environmental impact assessments. To improve predictive capabilities, AI and Machine Learning algorithms can be employed to analyze historical air quality data, detect pollution trends, and forecast hazardous air conditions, enabling proactive pollution control measures. A mobile application can be developed for real-time data access, location-based air quality alerts, and push notifications to notify users when pollution levels exceed safe limits. Additionally, power optimization techniques, such as solar panel integration and battery-powered operation, can enhance system sustainability and portability, making it ideal for remote and off-grid deployments. In the long term, this system can be expanded into a large-scale smart city network, where multiple sensor nodes are strategically placed across urban areas to create a real-time air quality mapping system, assisting government authorities in pollution regulation, traffic management, and public health safety initiatives.

2.6 Bibliography

[7]The rapid rise in air and noise pollution has become a critical issue, necessitating real-time monitoring and control. This project presents an IoT-based solution using Raspberry Pi to measure the Air Quality Index and Noise Intensity of a region. The system comprises four key modules: Air Quality Index Monitoring, Sound Intensity Detection, Cloud-Based Monitoring, and Anomaly Notification. Sensors detect pollutants and noise levels, while the Wi-Fi-enabled Raspberry Pi processes and uploads data to the cloud for periodic analysis. In case of hazardous conditions, the system triggers alerts, ensuring timely intervention.

[8]IoT-based air quality monitoring system using an MQ2 sensor and ESP32 microcontroller to detect pollutant levels. It provides real-time data via an LCD display and ThingSpeak cloud, triggering alerts if pollution exceeds safe limits. The system also integrates a DHT11 sensor for temperature and humidity monitoring, ensuring reliable and efficient environmental tracking.

[9]India faces severe air pollution, with over 10 cities ranking among the worst. The Air Quality Index (AQI) monitors pollution levels using various contaminants. This study develops an IoT-based air quality monitoring system that detects CO, smoke, and PM levels, providing real-time alerts via a buzzer for public safety.

[10]Urban pollution poses a significant health risk. This paper presents a low-cost, IoT-based air quality monitoring system. Using an ESP32 module, Wi-Fi, and two sensors, it measures CO2, temperature, and humidity. Data is accessible via smartphone or web, allowing remote monitoring and resets. The system provides real-time readings and alerts, enabling informed responses.

[11]IoT-based system for real-time air quality monitoring via AQI. Using an Arduino UNO and sensors, it measures NO2, CO, and PM2.5. Data is transmitted via ESP8266 Wi-Fi to the ThingSpeak IoT platform. An Android app retrieves data from ThingSpeak, displaying PPM and AQI levels, making the system suitable for real-world applications.

[12]Overcrowding and shrinking open spaces worsen air quality, impacting health. This study proposes an IoT-based air quality monitoring system using LoRaWAN. It uses WisBlock Kit 4 and BME680 sensors to measure AQI, temperature, humidity, and pressure. Evaluation shows stable LoRa coverage and fast website response. This WiFi-independent system is suitable for various settlements and informs air quality-conscious urban planning.

[13]Real-time, self-powered AQMS uses IoT and cloud computing for sustainable air quality monitoring. Based on an AVR Microcontroller and GSM modem, it's scalable for deployment across urban areas. A suction pump draws air through sensors measuring various pollutants, temperature, humidity, and noise. Data is collected hourly (adaptable) and processed in the cloud using AI for insights and identifying critical emission locations. The system is designed for Karachi, aiming to contribute to its smart city transformation.

[14]Air pollution causes millions of deaths annually and harms ecosystems. Governments use air quality monitoring systems to regulate emissions. This research explores air quality measurement and prediction using IoT sensors. By deploying stationary and portable

sensors, and applying machine learning to real-world data, the study demonstrates effective air quality detection and forecasting, benefiting smart cities and businesses.

[15]Global air pollution, driven by factors like population growth and industrialization, negatively impacts human health. This study develops an IoT-based pollution monitoring system to track levels of harmful substances like CO2, benzene, NH3, and NO2. Addressing limitations of current systems, the proposed three-phase method displays air quality in PPM on an LCD and mobile app, enabling real-time monitoring from anywhere. Advantages include real-time analysis, while limitations include coverage area and interpretation challenges.

[16]Urban air pollution is a major health concern. This article presents a low-power air quality monitoring system using a Raspberry Pi4 and various sensors (CO2, temperature, humidity, etc.). Data is collected, processed, and transmitted to the cloud for continuous monitoring and storage, addressing the need for accessible air quality information.

[17]IoT for energy-saving streetlights and air quality monitoring. Streetlights automatically turn off at dawn and on at dusk using LDR sensors, reducing energy waste. The system also monitors temperature, humidity, and gas leaks (smoke, alcohol, LPG). Data is accessible online via smartphones/computers, and a GSM module alerts authorities to dangerous gas levels. LDR and air quality data are uploaded to ThingSpeak.

[18]IoT-based Air Quality Monitoring (AQM) system aims to raise pollution awareness. It provides real-time warnings and data logging, displaying pollution levels on an LCD screen. Located in high-pollution areas, the AQM measures harmful and harmless particles, displaying the Air Quality Index digitally and graphically. An accompanying Android app, using GPS, allows users to access and compare global pollution data.

[19]IoT-based system for real-time air and noise pollution monitoring. Using air sensors, it detects harmful gases like NH3, benzene, smoke, and CO2. The system also monitors noise levels, triggering a buzzer when a threshold is exceeded. The goal is to track air and noise pollution in various urban locations for a smarter environment.

[20]Urban air quality monitoring often overlooks agricultural areas, despite the impact of pollutants on crops and farmers' health. Current monitoring stations have limited coverage. This work proposes a low-power, IoT-based network for air quality monitoring specifically in agricultural fields, addressing the need for broader, more efficient assessment.

[21]Conventional air monitoring systems, while precise, are bulky, expensive, and limited in data types, hindering widespread use. This paper proposes a real-time air pollution monitoring and forecasting system using IoT. This approach drastically reduces hardware costs, enabling large-scale deployment of a sensor network. Beyond traditional monitoring, the system uses neural networks to analyze sensor data and forecast pollution trends, enabling targeted interventions to minimize negative impacts.

[22]Indoor air pollution significantly harms human health. This paper presents a Bolt-based IoT system for real-time monitoring of indoor pollutants like carbon dioxide, carbon monoxide, and particulate matter within university premises. The system provides pollution data directly to smart devices. The paper also suggests measures to improve indoor air quality, aiming to positively impact student health and academic performance.

[23]Limited data resolution and accuracy pose challenges for air quality research in Malaysia due to the sparse distribution of monitoring stations. This paper proposes a Distributed Air Quality Monitoring System (DAQMS) using IoT for improved data collection coverage. A low-cost mobile device, using an ESP32 and multiple sensors, streams data to a cloud server accessible via a web dashboard. Future plans focus on improving data accuracy and user experience. DAQMS addresses data granularity issues and raises public awareness of air pollution.

[24]Researchers have developed various air quality monitoring systems. This report proposes an IoT-enabled industrial air quality monitoring device using an MQ-135 gas sensor to detect contaminants like alcohol. A Node MCU ESP 8266 Wi-Fi module transmits real-time data to smart devices via an IoT platform.

[25]IoT's transformative impact on air quality monitoring. It reviews IoT-based systems for real-time assessment across various environments, highlighting the integration of technologies like 5G, AI, and blockchain. The paper discusses tailored IoT platforms, improvement strategies, and data utilization for policy and engagement. Future work suggests combining satellite/drone and ground-level monitoring for enhanced air quality management.

[26]IoT-based indoor air quality monitoring system to improve thermal comfort and health. Deploying a network of sensors, it collects real-time data on pollutants like PM, VOCs, CO, and NO2. Data is wirelessly transmitted to a central computer and accessible via an online server, with customizable alerts for specific thresholds. The goal is to enhance indoor air quality and reduce respiratory problems, especially for children and allergy sufferers.

[27]Air pollution poses a significant health risk, particularly to vulnerable populations. Current navigation systems lack air quality awareness. This research proposes an IoT-based navigation system, similar to Google Maps, that prioritizes air quality. Kit boxes deployed at intervals monitor AQI, temperature, and humidity, transmitting data to a central database. A web application displays real-time air quality on a map, enabling users, especially vulnerable groups, to choose less polluted routes.

Chapter -3: Hardware Description

3.1 ESP32-WROOM-32

- **Description**

ESP32-WROOM-32 is a powerful Wi-Fi + Bluetooth + Bluetooth LE MCU module, with two complementary PCB antennas in different directions. This module has the same layout of pins as ESP32-WROOM-32E except some pins are not led out, facilitating quick and easy migration between these two modules. With two unique antennas design on one single module, ESP32-WROOM-DA can be used to develop IoT applications that need stable connectivity over a broad spectrum, or to deploy Wi-Fi in challenging and hazardous environments, or to overcome communication problems in Wi-Fi-dead spots. This module is an ideal choice for indoor and outdoor devices for smart home, industrial control, consumer electronics, etc.

- **Features**

✓ ***CPU and On-Chip Memory***

• ESP32-D0WD-V3 embedded, Xtensa® dual-core 32-bit LX6 microprocessor, up to 240 MHz

• 448 KB ROM for booting and core functions

• 520 KB SRAM for data and instructions

• 16 KB SRAM in RTC

✓ ***Wi-Fi***

• 802.11b/g/n

• Bit rate: 802.11n up to 150 Mbps

• A-MPDU and A-MSDU aggregation

• 0.4 μs guard interval support

• Center frequency range of operating channel: 2412 ~ 2484 MHz

✓ ***Bluetooth***

• Bluetooth V4.2 BR/EDR and Bluetooth LE specification

• Class-1, class-2 and class-3 transmitter

• AFH

• CVSD and SBC

✓ ***Peripherals***

• SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, TWAI® (compatible with ISO 11898-1, i.e. CAN Specification 2.0)

✓ ***Integrated Components on Module***

• 40 MHz crystal oscillator

• 8 MB SPI flash

✓ ***Antenna Options***

• On-board dual PCB antennas

✓ ***Operating Conditions***

• Operating voltage/Power supply: 3.0 ~ 3.6 V

• Operating ambient temperature: –40 ~ 85 °C

➢ **Applications**

• Home Automation

• Smart Building

• Industrial Automation

• Smart Agriculture

• Audio Applications

• Health Care Applications

• Wi-Fi-enabled Toys

• Wearable Electronics

• Retail & Catering Application

➢ **Block Diagram**

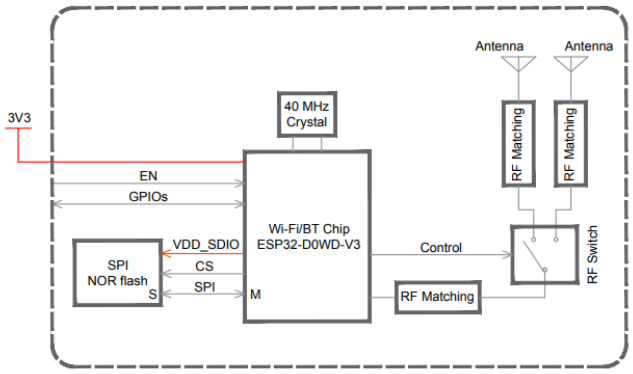

Figure 4: ESP32-WROOM-32 Block Diagram

➢ Pin Layout

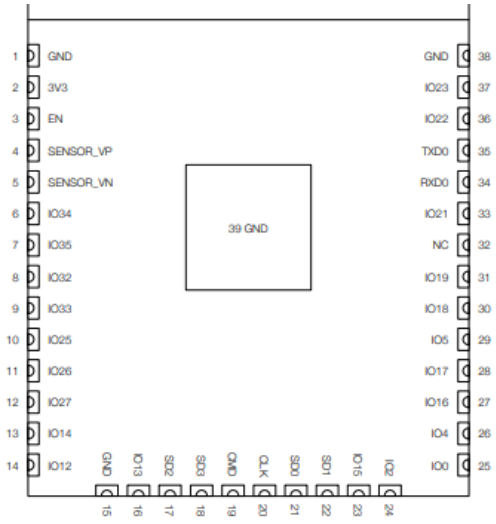

Figure 5: Pin Layout (Top View)

➢ Programming

- **Arduino IDE**: The ESP32 is compatible with the **Arduino IDE** through the ESP32 core, making it easy to program with familiar tools.
- **MicroPython**: Support for programming the ESP32 with **MicroPython**, a Python-based interpreter.
- **ESP-IDF (Espressif IoT Development Framework)**: The official framework for ESP32 development using C or C++.

➢ Peripheral Schematics

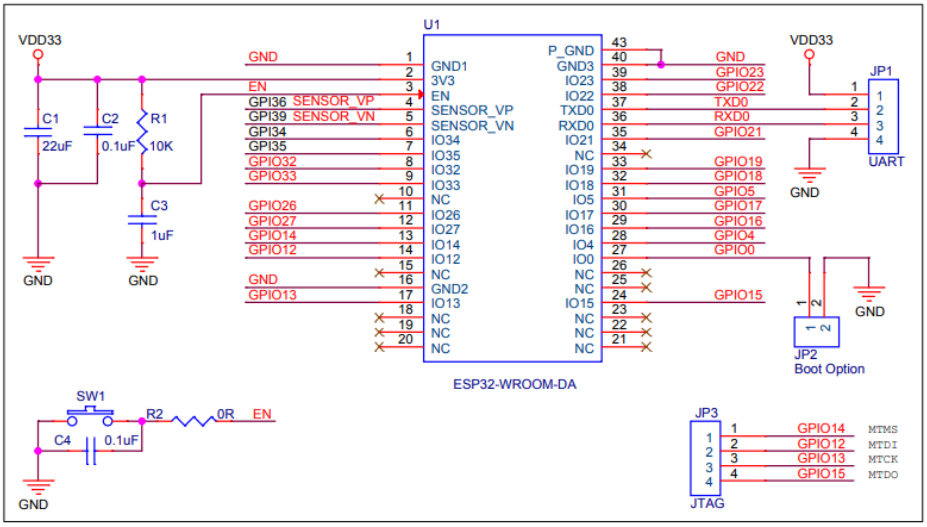

Figure 6: Peripheral Schematics

3.2 Nova PM sensor (SDS011)

- **Overview**

  The SDS011 using principle of laser scattering, can get the particle concentration between 0.3 to 10μm in the air. It with digital output and built-in fan is stable and reliable.

Figure 7: Nova PM sensor

- **Characteristics**

  1. Accurate and Reliable: laser detection, stable, good consistency;
  2. Quick response: response time is less than 10 seconds when the scene changes;
  3. Easy integration: UART output (or IO output can be customized), fan built-in;
  4. High resolution: resolution of 0.3μ g/m3;

- **Working principle**

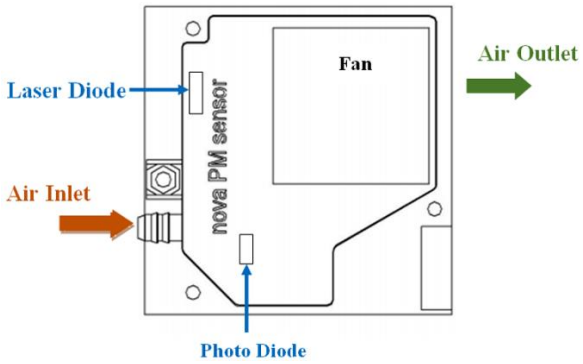

Figure 8: Working of PM sensor

Using laser scattering principle: Light scattering can be induced when particles go through the detecting area. The scattered light is transformed into electrical signals and these signals

will be amplified and processed. The number and diameter of particles can be obtained by analysis because the signal waveform has certain relations with the particles diameter.

- **Technical Parameters**

S.No	Item	Parameter
1	Measurement parameters	PM2.5,PM10
2	Range	0.0-999.9 μg /m3
3	Rated voltage	5V
4	Rated current	70mA±10mA
5	Temperature range	Storage environment: -20 ~ +60℃
6	Frequency	1Hz
7	Minimum resolution of particle	0.3 μm
8	Product size	71x70x23mm

Technical Parameters of Nova PM sensor

- **Connection of PM sensor to ESP32**

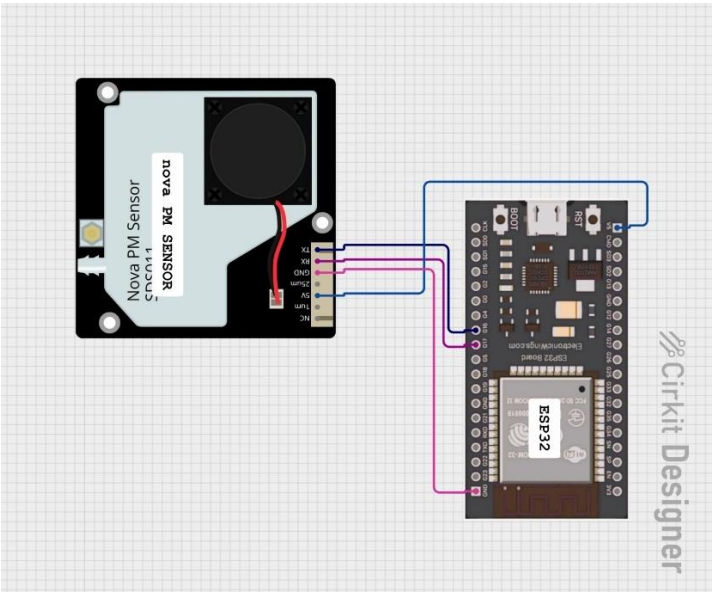

Figure 9: Connection of PM sensor to ESP32

- **Pin Layout**

No	Name	Comment
1	NC	Not Connect
2	1μm	PM2.5: 0-999μg/m³;PWM Output
3	5V	5V Input
4	2.5μm	PM10: 0-999 μg/m³;PWM Output
5	GND	Ground
6	Rx	RX of UART（TTL）@3.3V
7	Tx	TX of UART（TTL）@3.3V

Pin Layout of Nova PM sensor

3.3 DHT22

- **Description**

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip computer. Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory. Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

Figure 10: DHT22

- **Features**

  * Full range temperature compensated

  * Relative humidity and temperature measurement

  * Calibrated digital signal

  *Outstanding long-term stability

  *Extra components not needed

  * Long transmission distance

  * Low power consumption

  *4 pins packaged and fully interchangeable

- **Connection of DHT22 to ESP32**

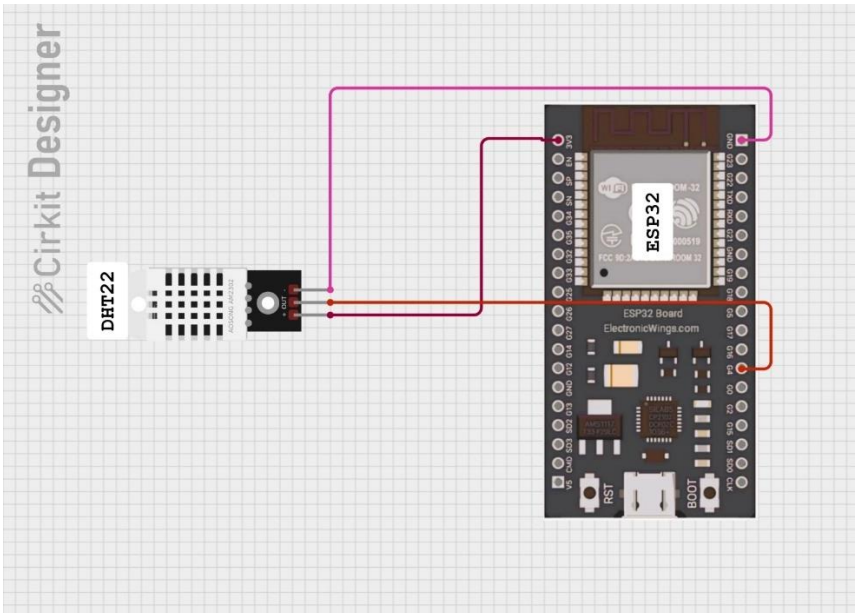

Figure 11: Connection of DHT22 to ESP32

- **Technical Specification**

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-

	0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

Technical Specifications of DHT22

3.4 MQ135

- **Description**

  Sensitive material of MQ135 gas sensor is SnO2, which with lower conductivity in clean air. When target pollution gas exists, the sensor's conductivity gets higher along with the gas concentration rising. Users can convert the change of conductivity to correspond output signal of gas concentration through a simple circuit. MQ135 gas sensor has high sensitivity to ammonia gas, sulfide, benzene series steam, also can monitor smoke and other toxic gases well. It can detect kinds of toxic gases and is a kind of low-cost sensor for kinds of applications.

Figure 12: MQ135 Gas Sensor

- **Features**
  - **Gas Detection:** Can detect ammonia (NH_3), sulfur (SO_2), benzene (C_6H_6), carbon dioxide (CO_2), and smoke.
  - **High Sensitivity:** Responds effectively to a wide range of harmful gases.
  - **Fast Response Time:** Provides quick readings after detecting gas concentration.

- **Analog and Digital Output:** Supports both analog (voltage variation) and digital output (threshold-based).
- **Adjustable Sensitivity:** A potentiometer is available to set the gas detection threshold.
- **Low Power Consumption:** Suitable for battery-powered applications.
- **Operating Voltage: 5V DC**.
- **Preheat Time Required:** Typically **over 24 hours** for stable readings.
- **Long Lifespan:** Usually lasts for several years in proper conditions.
- **Compact Size:** Can be integrated into small devices for real-time air monitoring.
- **Affordable and Easy to Use:** Widely used in DIY and industrial applications.

➢ **Applications**

- Indoor and outdoor air quality monitoring.
- Gas leakage detection in homes and industries.
- Environmental pollution monitoring.
- Smart IoT-based air monitoring systems.
- Automated ventilation systems.

➢ **Technical Parameters**

Model			MQ135
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite, Metal cap
Target Gas			ammonia gas, sulfide, benzene series steam
Detection range			10 ~ 1000ppm(ammonia gas, toluene, hydrogen, smoke)
Standard Circuit Conditions	Loop Voltage	V_c	≤24V DC
	Heater Voltage	V_H	5.0V±0.1V AC or DC
	Load Resistance	R_L	Adjustable
Sensor character under standard test conditions	Heater Resistance	R_H	29Ω±3Ω (room tem.)
	Heater consumption	P_H	≤950mW
	Sensitivity	S	Rs(in air)/Rs(in 400ppm H_2)≥5
	Output Voltage	Vs	2.0V ~ 4.0V（ in 400ppm H_2 ）
	Concentration Slope	α	≤0.6(R_{400ppm}/R_{100ppm} H_2)
	Tem. Humidity		20℃±2℃；55%±5%RH

Standard test conditions	Standard test circuit	Vc:5.0V±0.1V； V_H: 5.0V±0.1V
	Preheat time	Over 48 hours

Technical parameters of MQ135 gas sensor

- **Connection of MQ135 to ESP32**

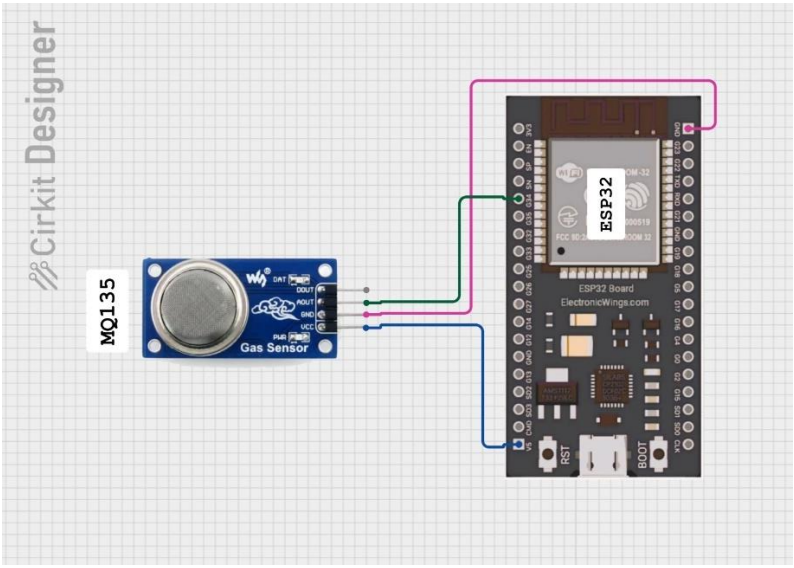

Figure 13: Connection of MQ135 to ESP32

- **Pin Layout**

Pin	Name	Description
1	VCC	Power supply (3.3V to 5V)
2	GND	Ground
3	DO (Digital Output)	Digital output for gas detection (use with threshold)
4	AO (Analog Output)	Analog output for gas concentration (use with ADC)

Pin Layout of MQ135 Gas Sensor

3.5 0.96-inch OLED

- **Description**

  The display is made from hundreds of LEDs, each representing one pixel in a 128x64 grid. Thus, it doesn't need any backlight, and it needs less power than the classic LCD screens. That's 8192 pixels all individually lit at your fingertips! The design is 5V ready with an onboard regulator, so is compatible with any 3.3V board. The breakout board uses only about 20 mA, depending on active pixels. It is easy to control over the I2C interface. It is even easier to connect with the easy system - no soldering required.

Figure 14: OLED Display

- **Features**
  - Screen diagonal: 0.96" wide
  - Current consumption: ~20 mA (depending on active pixels)
  - Logic voltage level: 5V (on I2C header)
  - Operating voltage: 5V (onboard regulator for 3.3V)
  - Communication: I2C (address: 0x3C)
  - Connectors: easyC x2 Optional white pixel color
- **Connection of OLED to ESP32**

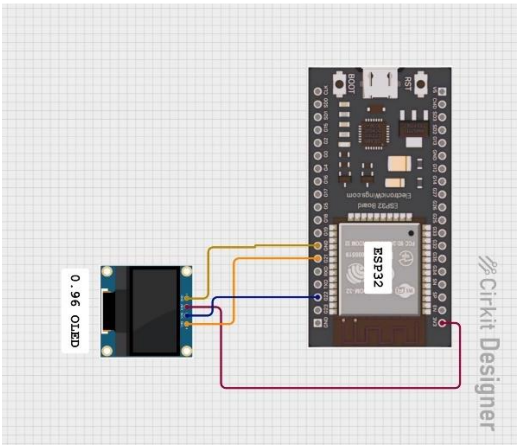

Figure 15: Connection of OLED to ESP32

➢ **Pin Layout**

OLED Pin	Esp32 Pin
VCC	3.3 V
GND	GND
SDA	GPIO 21
SCK	GPIO 22

Pin Layout of OLED display

3.6 MCP2515 CAN Controller

➢ **Description**

Microchip Technology's MCP2515 is a stand-alone Controller Area Network (CAN) controller that implements the CAN specification, Version 2.0B. It is capable of transmitting and receiving both standard and extended data and remote frames. The MCP2515 has two acceptance masks and six acceptance filters that are used to filter out unwanted messages, thereby reducing the host MCU's overhead. The MCP2515 interfaces with microcontrollers (MCUs) via an industry standard Serial Peripheral Interface (SPI).

Figure 16: CAN Module

➢ **Features**

• Implements CAN V2.0B at 1 Mb/s:

- 0 to 8-byte length in the data field
- Standard and extended data and remote frames

• Receive Buffers, Masks and Filters:

- Two receive buffers with prioritized message storage

- Six 29-bit filters
- Two 29-bit masks

• Data Byte Filtering on the First Two Data Bytes (applies to standard data frames)

• Three Transmit Buffers with Prioritization and Abort Features

• High-Speed SPI Interface (10 MHz):

- SPI modes 0,0 and 1,1

• One-Shot mode Ensures Message Transmission is Attempted Only One Time

• Clock Out Pin with Programmable Prescaler:

- Can be used as a clock source for other device(s)

• Start-of-Frame (SOF) Signal is Available for Monitoring the SOF Signal:

- Can be used for time slot-based protocols and/or bus diagnostics to detect early bus degradation

• Interrupt Output Pin with Selectable Enables

• Buffer Full Output Pins Configurable as:

- Interrupt output for each receive buffer

- General purpose output

• Request-to-Send (RTS) Input Pins Individually Configurable as:

- Control pins to request transmission for each transmit buffer

- General purpose inputs

➢ **Pin Layout**

MCP2515 Pin	Description	ESP32 Connection
VCC	Power (5V or 3.3V)	**3.3V** (or 5V if using level shifter)

MCP2515 Pin	Description	ESP32 Connection
GND	Ground	**GND**
CS	SPI Chip Select	**GPIO 5**
SO	SPI MISO	**GPIO 19**
SI	SPI MOSI	**GPIO 23**
SCK	SPI Clock	**GPIO 18**
INT	Interrupt (active low)	**GPIO 2**

Pin Layout of MCP2515 CAN

- **Connection of CAN to ESP32**

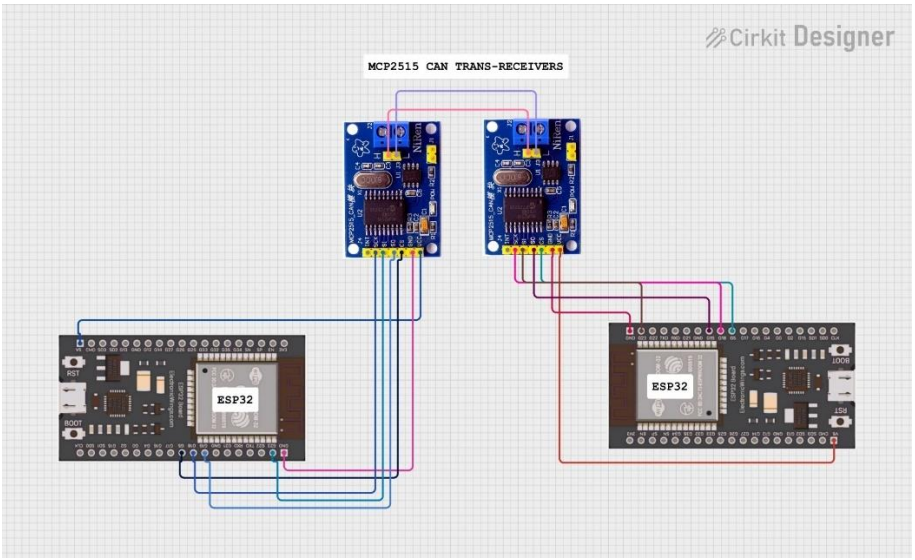

Figure 17: Connection of CAN to ESP32

- **Benefits**
  - **Simplified CAN Implementation:** Handles the complex CAN protocol, simplifying the design of CAN-based systems.
  - **Reduced Microcontroller Load:** Message filtering and buffering reduce the processing required from the microcontroller.
  - **Flexible Configuration:** Offers various configuration options to tailor the CAN communication to specific application needs.

- **Cost-Effective:** A relatively low-cost solution for adding CAN connectivity to a system.

➢ **Applications**

The MCP2515 is widely used in various applications, including:

- Automotive electronics
- Industrial automation
- Embedded systems
- CAN-based networks

Chapter -4: Software Description

4.1 Embedded C language

Embedded C is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems. Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations.

The C programming language is perhaps the most popular programming language for programming embedded systems. We mentioned other popular programming languages).

Most C programmers are spoiled because they program in environments where not only there is a standard library implementation, but there are frequently a number of other libraries available for use. The cold fact is, that in embedded systems, there rarely are many of the libraries that programmers have grown used to, but occasionally an embedded system might not have a complete standard library, if there is a standard library at all. Few embedded systems have capability for dynamic linking, so if standard library functions are to be available at all, they often need to be directly linked into the executable. Oftentimes, because of space concerns, it is not possible to link in an entire library file, and programmers are often forced to "brew their own" standard c library implementations if they want to use them at all. While some libraries are bulky and not well suited for use on microcontrollers, many development systems still include the standard libraries which are the most common for C programmers.

C remains a very popular language for micro-controller developers due to the code efficiency and reduced overhead and development time. C offers low-level control and is considered more readable than assembly. Many free C compilers are available for a wide variety of development platforms. The compilers are part of an IDEs with ICD support, breakpoints, single-stepping and an assembly window. The performance of C compilers has improved considerably in recent years, and they are claimed to be more or less as good as assembly, depending on who you ask. Most tools now offer options for customizing the compiler optimization. Additionally, using C increases portability, since C code can be compiled for different types of processors.

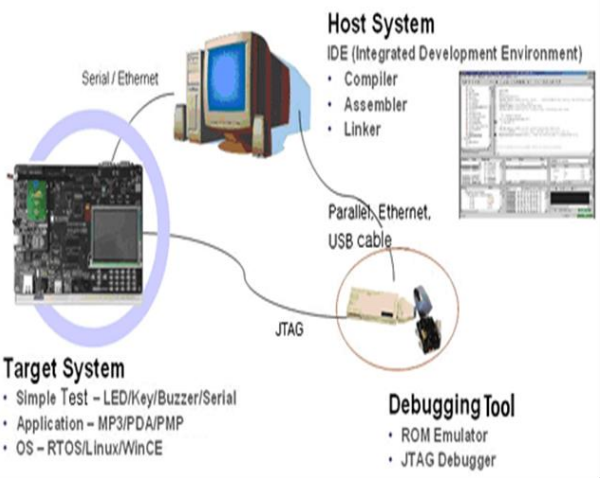

Figure 18: Embedded System Development Environment

4.2 Arduino IDE compiler

Arduino is an open-deliver electronics platform based mostly on smooth-to-use hardware and software utility. Arduino boards can observe inputs - slight on a sensor, a finger on a button, or a Twitter message - and flip it into an output - activating a motor, turning on an LED, publishing a few components online. You could tell your board what to do by sending a hard and fast of commands to the microcontroller at the board. To do so that you use the Arduino programming language (based totally mostly on Wiring), and the Arduino software (IDE), based on Processing.

Over the years Arduino has been the brain of lots of obligations, from regular gadgets to complex medical gadgets. A worldwide community of makers - college students, hobbyists, artists, programmers, and specialists - has collected spherical this open-deliver platform, their contributions have brought as much as a terrific amount of available know-how that can be of terrific assist to novices and experts alike.

Arduino has become born on the Ivrea interaction format Institute as a clean tool for instant prototyping, geared towards university college students without a historic past in electronics and programming. As quickly as it reached a miles wider community, the Arduino board started converting to conform to new dreams and traumatic situations, differentiating its provide from smooth eight-bit boards to merchandise for IoT

Programs, wearable, three-D printing, and embedded environments. All Arduino boards are without a doubt open-deliver, empowering clients to assemble them independently and ultimately adapt them to their unique dreams. The software program, too, is open-supply, and its miles growing thru the contributions of customers globally.

The advantages of the Arduino IDE utility are

1. much less steeply-priced
2. The clean smooth programming surroundings
3. Extensible software program application utility and hardware

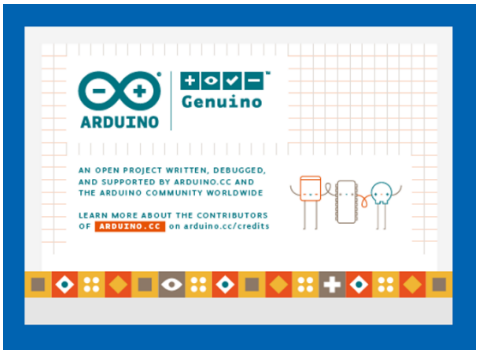

Figure 19: Arduino IDE

The Arduino venture gives the Arduino blanketed development surroundings (IDE), it really is a go-platform software program software developed in the programming language Java. It is developed to introduce programming application with software improvement.

The Arduino venture gives the Arduino blanketed development surroundings (IDE), it really is a go-platform software program software developed in the programming language Java. It is developed to introduce programming application with software improvement.

It includes a code editor with features in conjunction with syntax highlighting, brace matching, and automatic indentation, and offers a simple one-click mechanism to collect and load packages to an Arduino board. A software program written with the IDE for Arduino is known as a "cool lively film".

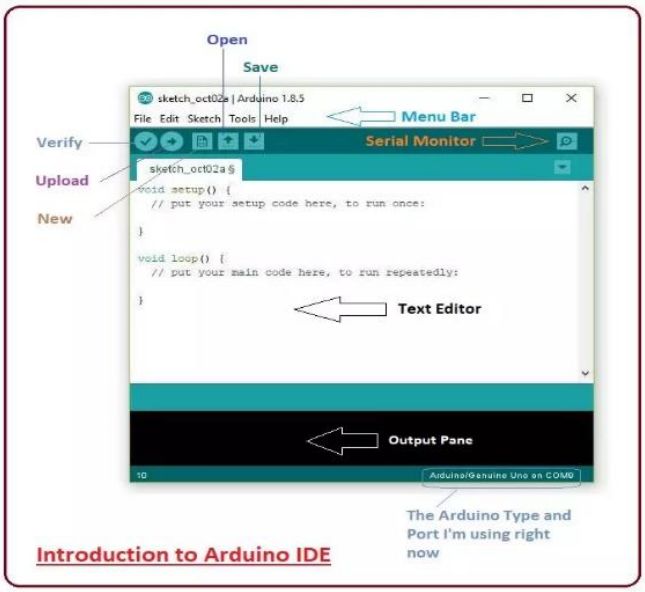

Figure 20: Using Arduino IDE app

Arduino IDE permits the languages C and C++ the use of special hints to set up code. The Arduino IDE materials a software program software library called Wiring from the Wiring task, which offers many, not unusual enter and output techniques. a massive Arduino C/C++ cool animated film embodies abilities that might be compiled and related with a utility stub vital() into an executable cyclic government software:

- Setup (): a feature that runs as fast as on the start of software and which can initialize settings.
- loop (): a characteristic called time and again until the board powers off.
- Writing Sketches
- Report
- Edit
- Caricature
- Equipment
- Help
- Sketchbook
- Tabs, more than one documents, and Compilation
- Importing
- Libraries
- Hardware

- Serial screen
- Possibilities
- Language assist

The Arduino Software (IDE) - carries a text editor for writing code, a message location, a text console, a toolbar with buttons for all functions, and a group of menus. It joins to the Arduino and Genuino hardware to add packages and talk with them.

4.3 Serial Monitor

Indicates serial facts being despatched from the Arduino UNO or Genuino board (USB or serial board). To ship data to the board, enter textual content and click on at the "supply" button 36 2336 or press enter. Choose out the baud price from the drop-down that fits to Serial. 12 Begin in your comic strip. Be conscious that on domestic home windows, Mac or Linux, the Arduino UNO or Genuino board will reset whilst you join with the serial display.

You could also interface to the board from Processing, Flash, MaxMSP, and so on (see the 12-interfacing net page for information). Possibilities: A few picks can be set inside the options dialog (determined under the Arduino menu on the 36 Mac, or document on home windows and Linux). The relaxation can get in the options file, 12 whose region is confirmed in the choice conversation.

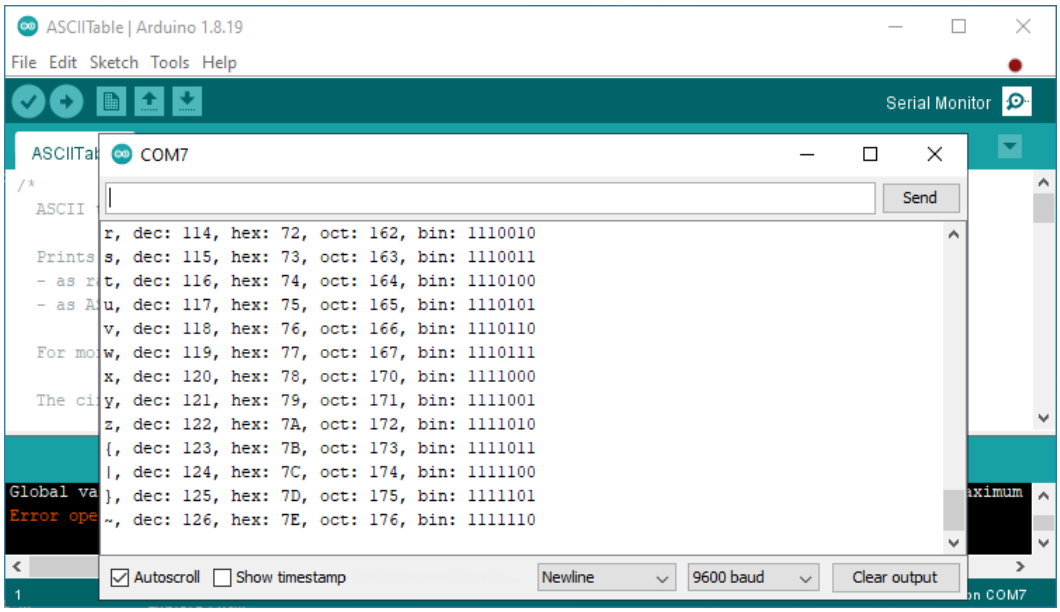

Figure 21: Serial Monitor

Arduino software (IDE) consists of the built-in to befit for the forums in the following listing, all primarily based on the AVR mcu. The Boards manager included inner the fashionable set up allows to feature assist for the growing variety of new boards based totally on special cores like Arduino UNO, Arduino 0, Edison, Galileo and so on.

4.4 Steps to dump

- Connect your Arduino device to your computer using a USB cable.
- Open the Arduino Integrated Development Environment (IDE) on your computer.
- Open the code you have written in the Arduino IDE editor.
- Choose the correct board type and port under the "Tools" menu. The board type should match the type of Arduino board you have, such as Uno, Mega, or Nano. The port should correspond to the serial port to which the Arduino is connected.
- Verify your code by clicking the "Verify" button (the checkmark icon) in the top left corner of the IDE window. This will compile your code and check for errors.
- If there are no errors, click the "Upload" button (the right-facing arrow icon) to upload the compiled code to the Arduino.
- Wait for the upload to complete and the status message "Done uploading" to appear in the bottom status bar of the IDE.
- The code is now on your Arduino and running. You can interact with it by sending commands via the serial monitor in the Arduino IDE, or by using other means, such as physical inputs or outputs on the board.

Note: If you encounter any problems during the upload process, make sure that your Arduino is connected properly, and try again. If the issue persists, consult the Arduino documentation or online forums for troubleshooting guidance.

4.5 Working Procedure

The working procedure of Arduino can be broken down into the following steps:

Connecting the hardware: Connect the Arduino board to your computer using a USB cable. Make sure that you have all the required components for your project, such as sensors, actuators, and other components.

Writing the code: Open the Arduino Integrated Development Environment (IDE) on your computer and write the code for your project. The code should define the functionality of the project and control the behaviour of the components connected to the Arduino.

Uploading the code: Verify the code for syntax errors and upload it to the Arduino board by clicking the "Upload" button in the Arduino IDE.

Running the code: Once the code is uploaded, the Arduino board will start executing it. Depending on the code, the board may interact with the components connected to it a perform various tasks, such as reading input data from sensors, controlling actuators, and displaying information on an output device.

Monitoring the performance: You can monitor the performance of the Arduino by using the serial monitor in the Arduino IDE. The serial monitor allows you to view the output generated by the code and send commands to the Arduino board.

4.6 ThingSpeak

ThingSpeak is a popular, open-source Internet of Things (IoT) platform that allows you to collect, visualize, and analyse live data streams in the cloud. Think of it as a bridge between your physical devices and the digital world, enabling you to make sense of the information they gather.

➢ **Key Features:**

- **Data Collection:** You can send data from any internet-connected device directly to ThingSpeak using various methods like REST APIs or MQTT.
- **Data Visualization:** ThingSpeak provides tools to create instant visualizations of your data, such as charts and graphs, making it easy to understand trends and patterns.
- **Data Analysis:** With built-in MATLAB analytics, you can perform advanced analysis on your data, including filtering, transformations, and statistical calculations.
- **Data Storage:** ThingSpeak stores your data securely in the cloud, so you can access it anytime, anywhere.
- **Alerts and Actions:** You can set up alerts to notify you when certain conditions are met, and even trigger actions based on your data.
- **Community and Resources:** ThingSpeak has a large community of users and provides extensive documentation and tutorials to help you get started.

➢ **How it Works:**

1. **Connect Devices:** Your IoT devices (sensors, microcontrollers, etc.) send data to ThingSpeak over the internet.
2. **Store Data:** ThingSpeak stores this data in channels, which are like containers for your data streams.
3. **Visualize and Analyse:** You can use ThingSpeak's tools or integrate with MATLAB to visualize and analyse your data.
4. **Take Action:** Based on your analysis, you can set up alerts, trigger actions, or even control your devices remotely.

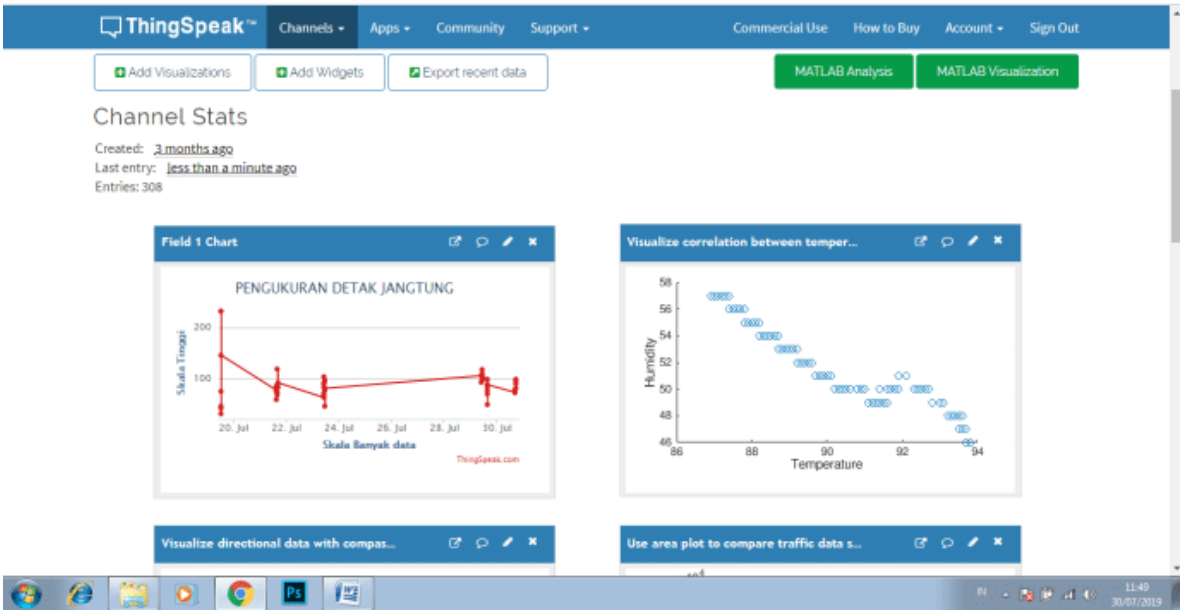

Figure 22: ThingSpeak Application

➢ **Benefits of Using ThingSpeak:**

- **Ease of Use:** ThingSpeak is designed to be user-friendly, even for those with limited programming experience.
- **Rapid Prototyping:** It's ideal for quickly building and testing IoT projects.
- **Data Analysis:** The integration with MATLAB provides powerful analytical capabilities.
- **Scalability:** ThingSpeak can handle large amounts of data and scale with your needs.
- **Open Source:** Being open-source, it offers flexibility and customization options.

➢ **Common Use Cases:**

- **Environmental Monitoring:** Tracking temperature, humidity, pollution levels, etc.
- **Smart Agriculture:** Monitoring soil conditions, weather data, and crop health.
- **Home Automation:** Controlling lights, appliances, and security systems.
- **Industrial Monitoring:** Tracking equipment performance and predicting maintenance needs.

Chapter -5: Testing and Results

5.1 Testing

- **Sensor Performance**: Ensure the sensors for air quality parameters (e.g., CO2, NO2, PM2.5, PM10, temperature, humidity) are accurate and responsive.
- **CAN Protocol Communication**: Test the CAN protocol communication between microcontrollers and other IoT devices.
- **Data Transmission**: Verify that the air quality data collected by the sensors is transmitted correctly via CAN to the IoT device and uploaded to the cloud or server.
- **System Scalability**: Check if the system supports multiple devices or can scale to handle a larger number of sensors in different areas.

5.2 Test Setup

• **Hardware used**: Microcontrollers, sensors, CAN transceivers, IoT module, power supply.

• **Software tools**: Programming platforms (Arduino IDE, PlatformIO, etc.), CAN protocol libraries, IoT cloud platforms (ThingSpeak, Blynk, etc.), and data visualization tools.

• **Test environment**: Indoor or outdoor air quality, controlled lab environment for initial tests, or real-world deployment for long-term testing.

5.3 Test Cases

✓ **Sensor Accuracy Test**:

- *Objective*: Verify that the air quality sensors provide accurate readings for various pollutants.
- *Method*: Compare sensor readings with reference values from certified air quality monitoring devices.
- *Expected Outcome*: The readings from the sensors should closely match the reference values.

✓ **CAN Communication Test**:

- *Objective*: Test the data transmission using the CAN protocol between multiple devices.

- *Method*: Send data from the sensors to a microcontroller using CAN protocol and ensure it's received correctly.
- *Expected Outcome*: Successful transmission of data with minimal errors or data loss.

✓ **Data Integrity Test**:

- *Objective*: Ensure that the air quality data is accurately transmitted from the sensors via CAN and correctly uploaded to the IoT cloud.
- *Method*: Monitor data consistency at each stage (sensor to microcontroller, microcontroller to IoT platform).
- *Expected Outcome*: No loss of data or incorrect readings during transmission.

✓ **Real-Time Monitoring Test**:

- *Objective*: Verify that the system updates the air quality data in real time on the IoT platform.
- *Method*: Use an IoT dashboard (e.g., ThingSpeak) to check if air quality data is updated at the correct intervals.
- *Expected Outcome*: Real-time updates every X seconds or minutes, as per the system's configuration.

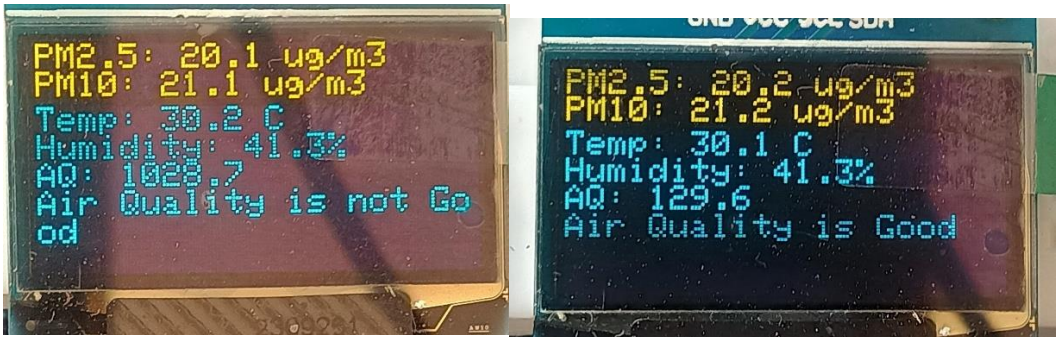

Figure 23: Real-time Outputs

✓ **Power Consumption Test**:

- *Objective*: Test the power consumption of the system when running continuously.
- *Method*: Measure the current draw of the system during operation.
- *Expected Outcome*: Ensure the power consumption is within acceptable limits for prolonged use (especially for IoT devices powered by batteries).

5.4 Results

✓ **Sensor Accuracy Results:**

- **CO2 Sensor**: The sensor reading was within ±5% of the reference value in controlled indoor conditions.
- **PM2.5 Sensor**: Accuracy improved with calibration; readings matched with reference device to within ±10 µg/m³.
- **Temperature and Humidity Sensors**: The sensors showed accuracy within ±0.5°C for temperature and ±2% for humidity compared to a calibrated reference.

✓ **CAN Protocol Communication Results:**

- **Message Transmission**: The CAN protocol successfully transmitted air quality data from sensor nodes to the central microcontroller with no packet loss.
- **Bus Load**: During tests with multiple devices (up to 4 nodes), the CAN bus maintained stable communication without significant delays or errors.

✓ **Wi-Fi and CAN Set-up**

```
.....
Wi-Fi Connected
Entering Configuration Mode Successful!
Setting Baudrate Successful!
CAN Module Initialized Successfully!
Received: PM2.5=10.0, PM10=10.0, Temp=28.0, Humidity=39.0, AQ=626.0
Received: PM2.5=10.0, PM10=10.0, Temp=28.0, Humidity=39.0, AQ=605.0
Received: PM2.5=10.0, PM10=10.0, Temp=28.0, Humidity=39.0, AQ=623.0
Received: PM2.5=10.0, PM10=10.0, Temp=28.0, Humidity=39.0, AQ=619.0
Received: PM2.5=10.0, PM10=11.0, Temp=28.0, Humidity=39.0, AQ=627.0
Received: PM2.5=10.0, PM10=11.0, Temp=28.0, Humidity=39.0, AQ=595.0
```

Figure 24 : Wi-Fi and CAN Set-up

✓ **Data Integrity Results:**

- **Cloud Upload**: The data was correctly uploaded to the IoT platform without any discrepancies. The air quality data was continuously recorded in real-time.
- **Error Rate**: Less than 0.5% error rate in data transmission over CAN protocol during the test period.

✓ **Real-Time Data collection**

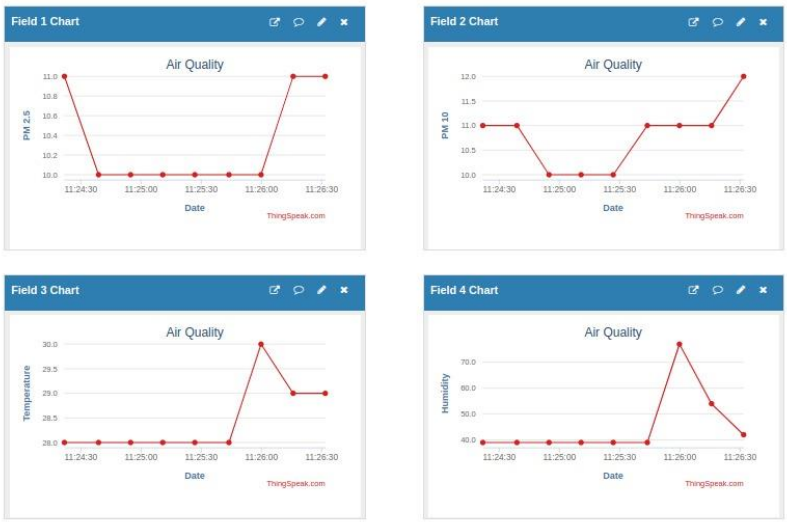

Figure 25: Real-Time Data collection

✓ **Real-Time Monitoring:**

- **Dashboard Updates**: The system updated the air quality data every 10 seconds on the ThingSpeak platform with no lag or delay.
- **Visualization**: The IoT platform successfully displayed air quality trends with graphs and gauges for CO2 levels, PM concentrations, and other metrics.

✓ **Power Consumption:**

- **Current Draw**: The system consumed around 80mA during active operation and dropped to 30mA in sleep mode (for low-power configurations).
- **Battery Life**: With a 2000mAh battery, the system could run for approximately 15–20 hours before needing a recharge.

5.5 Discussion of Results

- **Performance**: Overall, the system performed well in terms of accuracy, reliability, and real-time monitoring. The CAN protocol was an effective choice for local communication, providing stable data transmission.
- **Challenges Encountered**: The primary challenge was sensor calibration for accurate measurements in different environmental conditions. Calibration was critical for the reliability of data, especially for particulate matter (PM) sensors.

- **System Limitations**: In extreme environmental conditions (high humidity, dust), the accuracy of certain sensors like PM2.5 and CO2 fluctuated slightly. The power consumption for continuous operation could be optimized further, especially for battery-powered systems.

5.6 Conclusion

- The system was able to effectively monitor air quality in real-time, transmit data via CAN protocol, and display information on the IoT platform.
- Future improvements could include enhancing sensor calibration, reducing power consumption, and adding more advanced features like predictive analytics or alerts based on predefined thresholds.

Chapter -6: Conclusion

The IoT-based air quality monitoring system utilizing the CAN protocol successfully demonstrates the ability to monitor and transmit air quality data in real-time. Through the integration of various sensors, such as CO2, PM2.5, and temperature/humidity sensors, the system was able to capture accurate environmental data, which was then communicated effectively via the CAN protocol to an IoT platform. The real-time monitoring provided actionable insights into air quality levels, ensuring that the data was consistently updated and accessible for analysis.

The system showed reliable performance, with accurate sensor readings and minimal data loss during transmission. However, challenges such as sensor calibration in varying environmental conditions and optimizing power consumption for long-term use were identified. While the system operated efficiently in controlled conditions, further work is required to enhance battery life and sensor stability in more extreme environments.

In conclusion, this project highlights the potential of IoT and CAN protocol for smart air quality monitoring solutions, offering valuable real-time data for improving public health and environmental awareness. With future optimizations, the system could be implemented in smart cities, industrial zones, or healthcare applications, making it a promising tool for sustainable environmental monitoring and management.

References

[1] T. S. G. S M Abiduzzaman, “Real-Time Outdoor Air Quality Monitoring System,” in *researchgate*, Malasiya, 2021.

[2] M. B. M. G. T. Boyanka Marinova Nikolova, “AIR QUALITY MONITORING SYSTEM,” in *researchgate*, Technical University of Sofia, Studenstki Grad, 2006.

[3] P. Y. T. ,. V. P. K. R. Anand Jayakumar, “IoT Based Air Pollution Monitoring System,” in *IRJET*, Sri Ramakrishna Institute of Technology, India , 2021.

[4] V. S. B. P. P. a. R. P. T Dineshkumar, “ Air Quality Monitoring System Based on IoT,” in *researchgate*, Thottiam, Tamil Nadu, India , 2020.

[5] E. twahirwa, “A LoRa enabled IoT-based Air Quality Monitoring System for Smart City,” in *IEEE*, Kigali, Rwanda, 2018.

[6] M. I. A. S. &. N. H. M. Yunus1, “Development of Blynk IoT-Based Air Quality Monitoring System,” in *Journal of Engineering Technology*, Universiti Kuala Lumpur British Malaysian Institute, 2021.

[7] A. K. Saha, A Raspberry Pi Controlled Cloud Based Air and Sound Pollution Monitoring System with Temperature and Humidity Sensing, Institute of Engineering & Management, Kolkata, India: IEEE, 2018.

[8] Kalaivani, “Air Monitoring with Cloud and IoT,” in *IEEE*, Coimbatore, India , 2023.

[9] A. Kumar, “Design and Analysis of IoT based Air Quality Monitoring System,” in *IEEE*, Mathura, India , 2020.

[10] K. H. Yusof, “Design and Development of Real Time Indoor and Outdoor Air Quality Monitoring System Based onIoT Technology,” in *IEEE*, Shah Alam, Selangor, Malaysia , 2022.

[11] P. William, “Design and Implementation of IoT based Framework for Air Quality Sensing and Monitoring,” in *IEEE*, Dehradun, Uttarakhand, India , 2022.

[12] T. H. Rochadiani, “Design of Air Quality Monitoring Using LoRaWAN In Human Settlement,” in *IEEE*, Tangerang, Indonesia , 2022.

[13] M. Hussain, “Design of Low Cost, Energy Efficient, IoT Enabled, Air Quality Monitoring System with Cloud Based Data Logging, Analytics and AI,” in *IEEE*, Karachi, Pakistan , 2020.

[14] A. Samal, “Integrated IoT-Based Air Quality Monitoring and Prediction System: A Hybrid Approach,” in *IEEE*, NIT Rourkela, India , 2023.

[15] D. S. Ganesh, “ Internet of Things Enabled Air Quality Monitoring System,” in *IEEE*, Coimbatore, India , 2023.

[16] S. Faiazuddin, “IoT based Indoor Air Quality Monitoring system using Raspberry Pi4,” in *IEEE*, Rayalaseema University, Kurnool, A.P.India , 2020.

[17] M. N. S. Prasanth, “IOT based Energy Efficient Smart Street Lighting Technique with Air Quality Monitoring,” in *IEEE*, CVR College of Engineering ,Hyderabad, India, 2024.

[18] Manikandan, “IoT Based Air Quality Monitoring System with Email Notification,” in *IEEE*, Krishnankoil, India , 2021.

[19] D. T. Manglani, “IoT based Air and Sound Pollution Monitoring System for Smart Environment,” in *IEEE*, Jaipur, Rajasthan, India , 2022.

[20] Y. Gamal, “IOT-based air quality monitoring system for agriculture,” in *IEEE*, Egyptian Atomic Energy Authority, Cairo 11787, Egypt, 2022.

[21] C. Xiaojun, “IOT- Based Air Pollution Monitoring and Forecasting System,” in *IEEE*, Nantong, Jiangsu Prov, China , 2015.

[22] P. Asthana, “IoT Enabled Real Time Bolt based Indoor Air Quality Monitoring System,” in *IEEE*, Lucknow Campus, 2018.

[23] M. A. M. A. S. E. O. Chee Hoo Kok, “IoT based Low Cost Distributed Air Quality Monitoring System for Big Data Collection,” in *IEEE*, Penang, Malaysia , 2020.

[24] S. Das, “IoT Based Industrial Air Quality Monitoring System,” in *IEEE*, Kolkata, India, 2020.

[25] S. K. Pendekanti, “Recent Developments in IoT-Based Air Quality Monitoring and Control,” in *IEEE*, Manipal, India - 576104 , 2024.

[26] F. A. Ali, “IoT-based Real-time Monitoring System for Indoor Air Quality for Human Health,” in *IEEE*, Bhubaneswar, India, 2023.

[27] P. S. E. a. R. S. S. G. Pandiarajan S, “IoT-Based Air Quality Navigation System for Vulnerable Populations,” in *IEEE*, Department of CSE, KIT - Kalaignarkarunanidhi Institute of Technology, Coimbatore, Tamil Nadu, India, 2024.