**Mini Project Report**

# A Real – Time Remote Ambient Gas Monitoring Approach

*Mini Project Report Submitted in partial fulfilment of the requirements for the degree of Bachelor of Technology from Maulana Abul Kalam Azad University of Technology, West Bengal*

*(Formerly known as West Bengal University of Technology)*

***Submitted By***


1. **SK. AFROZ AHAMED**
   (Roll No.: **34900321023**, Registration No.: **213490100310028** of 2021– 2022)


*Under the Guidance of*

Project Supervisor: **Prof. (Mr.) Rajib Das,** Assistant Professor.
Technical Facilitator: **Mr. Abhijit Sarma**, Lab Assistant.


Department of Electronics and Communication Engineering,

Cooch Behar Government Engineering College, Cooch Behar, West Bengal




Department of Electronics and Communication Engineering
**Cooch Behar Government Engineering College,**
Village: Harinchawra, P.O.: Ghughumari,
District: Cooch Behar, West Bengal

June, 2024

# Acknowledgement

It is our pleasure to acknowledge our supervisor **Prof. (Mr.) Rajib Das, Assistant Professor, Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, Cooch Behar, West Bengal**, for being not only the source of encouragement, but also great resource of knowledge and information. We shall remain indebted to him for the immense help we have received from him.

Additionally, we would like to express our special thanks to our technical facilitator **Mr. Abhijit Sarma**, **Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, Cooch Behar, West Bengal**, for his time and efforts he provided in this project. Your useful advice and suggestions were helpful to us during the project's completion. In this aspect, we are eternally grateful to you.

We would also like to thank all the faculty members, technical assistants and staffs of the Department of Electronics and Communication Engineering, Cooch Behar Government Engineering College, for their kind and friendly cooperation extended to us.

We have no appropriate words to express sincere thanks to our family members for their continuous support and encouragement.

**Sk. AFROZ AHAMED**
**(Roll no: 349000321023)**

# Abstract

Air quality monitoring is crucial for ensuring public health and environmental protection. Traditional centralized air quality monitoring systems face challenges such as high latency, network dependency, and limited scalability. This paper presents a novel approach to real-time air quality assessment using edge computing and the IoT platform ThingSpeak. The proposed system integrates ambient gas sensors with edge computing devices to collect, process, and analyze air quality data locally before transmitting the summarized information to a cloud platform for further analysis and visualization.

The architecture leverages the computational power of edge devices to perform initial data processing, reducing the amount of data transmitted to the cloud, thereby minimizing latency and bandwidth usage. This approach ensures continuous monitoring even in areas with intermittent internet connectivity. ThingSpeak provides an efficient platform for storing, visualizing, and analyzing the processed data in real-time, enabling stakeholders to make informed decisions swiftly.

Experimental results demonstrate the system's capability to monitor various gas concentrations, such as CO2, NO2, and O3, in different environments. The edge computing-enabled setup shows significant improvements in response time and data transmission efficiency compared to traditional cloud-only systems. This study underscores the potential of combining edge computing with IoT platforms for scalable, real-time air quality monitoring, paving the way for enhanced environmental management and public health protection.

# TABLE OF CONTENTS


| Chapter No | Name of the Chapter | Page No |
|---|---|---|
| 1 | **INTRODUCTION**<br>1.1 Overview<br>1.2 Objectives<br>1.3 Background and Motivation | 1-2 |
| 2 | **METHODOLOGY**<br>2.1 Equipment's Description<br>2.2 MEMS gas sensor Specification<br>2.3 Overview of Software | 3-7 |
| 3 | **SYSTEM DESIGN**<br>3.1 Flow Chart<br>3.2 Wire Diagram | 8-9 |
| 4 | **SYSTEM IMPLEMENTATION**<br>4.1 Hardware Connection | 10-11 |
| 5 | **WORKING PRINCIPLE**<br>5.1 Initialization<br>5.2 Sensor Reading<br>5.3 Gas Sensor Concentration<br>5.4 Data Processing<br>5.5 Continuous Monitoring | 12-14 |
| 6 | **RESULT ANALYSIS**<br>6.1 Data Collection & Integration<br>6.2 Data Processing<br>6.3 Remote Monitoring | 15-18 |
| 7 | **CONCLUSION**<br>7.1 Key Achievement<br>7.2 Future Works | 19-20 |
| **LIST OF FIGURES** | | 21 |
| **LIST OF ABBREVIATIONS** | | 21 |
| **APPENDIX –A**<br>**APPENDIX –B**<br>**APPENDIX –C** | | 22-26 |
| **REFERENCE** | | 27-28 |

## CHAPTER 1

# INTRODUCTION

Air high-quality is an important environmental thing that appreciably influences public fitness and the surroundings. With urbanization and business activities at the rise, the need for non-stop and correct air first-rate tracking has turn out to be imperative. Traditional air exceptional monitoring systems, which rely upon centralized statistics processing and storage, regularly be afflicted by latency, excessive prices, and shortage of scalability. To cope with those challenges, this paper introduces a novel approach for real-time air quality assessment the use of side computing and ThingSpeak, focusing on far off ambient gas monitoring.

## 1.1 OVERVIEW:

This system uses the power of edge computing to process data at the source of data generation, reducing latency and bandwidth consumption. Combined with ThingSpeak, a cloud-based IoT analytics platform, the proposed solution enables efficient data discovery, storage and remote access. This approach ensures timely and reliable air quality assessments, providing valuable insights for environmental monitoring and public health interventions

## 1.2 OBJECTIVE:

### 1.2.1 Real-time data collection and processing:

- To continuously collect and process real-time air quality data through edge computing devices.
- Edge devices (such as sensors) installed at sites collect pollution and air data, which are processed on site to reduce downtime and enable immediate analysis

### 1.2.2 Effective data transmission:

- To efficiently transfer processed data to a central cloud platform (ThingSpeak) for storage, visualization, and further analysis.
- Using edge computing reduces the amount of raw data transmitted over the network, reduces bandwidth consumption, and ensures that only relevant data is transmitted.

### 1.2.3 Scalability and Flexibility:

- To develop a flexible and scalable system that can quickly integrate new sensors and cover an extended geographical area.
- ThingSpeak can be used with modular edge computing units to improve system efficiency and adapt to new needs with minimal maintenance.

### 1.2.4 Enhanced Decision-Making:

- To provide a practical approach and enable better decision making for environmental organizations and the public.
- By providing real-time data visualization and historical data analysis with ThingSpeak, stakeholders can make informed decisions about air quality policy and if pollution levels exceed safe limits a, immediate action has been taken.

### 1.2.5 Cost-Effectiveness and Energy Efficiency:

- To develop a cost-effective and energy-efficient air quality monitoring system.
- Edge computing reduces the need to repeatedly transfer data to the cloud, reducing operational costs and energy consumption. The use of low-power sensors and optimized processing algorithms further improves system efficiency.

## 1.3 MOTIVATION AND BACKGROUND

### 1.3.1 Background

The increasing concern over air quality and its impact on public health and the environment necessitates innovative approaches to monitor and manage pollution levels. Traditional air quality monitoring systems often rely on centralized data collection and processing, which can result in delays and gaps in information dissemination.

Advancements in edge computing and the Internet of Things (IoT) offer new opportunities for enhancing air quality monitoring systems. Edge computing brings computational power closer to the data source, reducing latency and enabling real-time data analysis.

### 1.3.2 Motivation

The motivation for utilizing edge computing and platforms like ThingSpeak in air quality monitoring stems from the need for a more efficient, real-time, and scalable solution. Traditional methods are often hindered by their dependence on centralized servers, which can create bottlenecks and delays in data transmission and processing.

ThingSpeak, an IoT analytics platform, complements this approach by providing a robust framework for collecting, analysing, and visualizing sensor data. It enables seamless data integration and real-time updates, making it an ideal tool for remote ambient gas monitoring

In summary, leveraging edge computing and ThingSpeak for real-time air quality assessment addresses the limitations of conventional systems. This approach not only enhances the timeliness and accuracy of air quality data but also supports proactive environmental management and public

## CHAPTER 2

# METHODOLOGY

## 2.1 EQUIPMENT DESCRIPTION:

**2.1.1 ESP 32:** ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Chinese company based in Shanghai, and is manufactured by TSMC using their 40 nm process.[15]

### Specification:

- **Microcontroller:** Espressif ESP32
- **Core: Dual-core** Tensilica LX6 microprocessor
- **Clock Speed:** Up to 240 MHz
- **Memory:** 520 KB SRAM, 4 MB flash
- **Connectivity:** Wi-Fi 802.11 b/g/n, Bluetooth 4.2 and BLE
- **GPIO Pins:** 34 digital GPIO pins, 12 Analog input pins
- **Operating:** Voltage: 3.3V

### Pin Diagram:

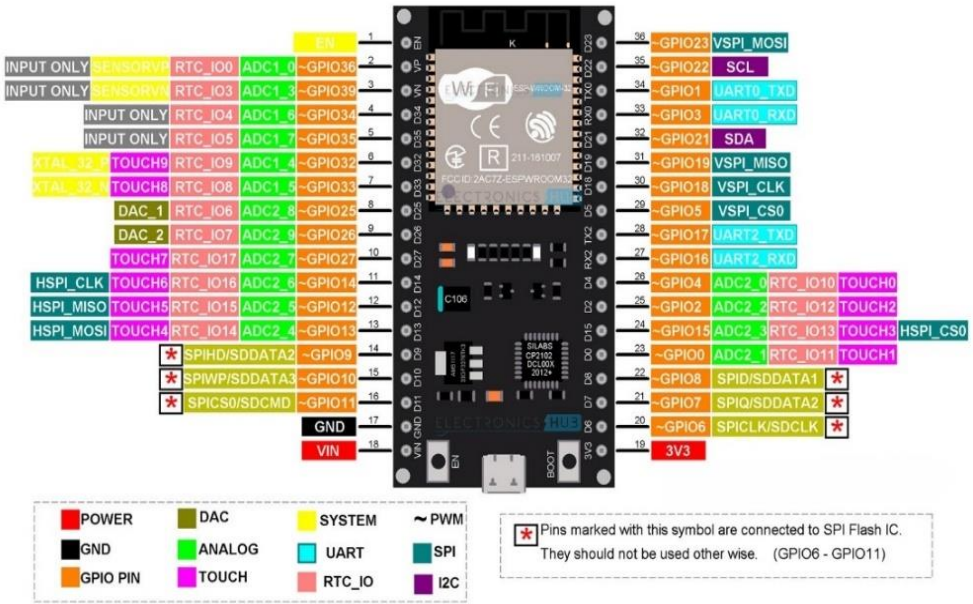



Figure-2.1 ESP32

**2.1.2 DHT 11:** The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It's fairly simple to use, but requires careful timing to grab data. You can get new data from it once every 2 seconds, so when using the library from Adafruit, sensor readings can be up to 2 seconds old.

Comes with a 4.7K or 10K resistor, which you will want to use as a pullup from the data pin to VCC.[22][23]

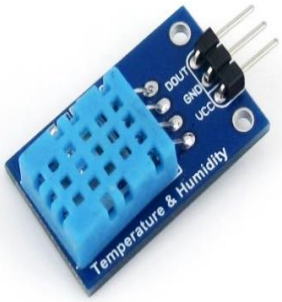



Figure-2.2 DHT11

## Specification:

- **Sensor Type:** Digital temperature and humidity sensor
- **Operating Voltage:** 3.3V - 5V
- **TemperatureRange:**0°Cto50°C
- **Humidity Range:** 20% to 90% RH
- **Temperature:** ±2°C
- **Humidity:** ±5% RH

**2.1.3 ADS1115:** ADS1115 is an ultra-small, low-power, 16-bit precision AD converter (Analog to Digital Converter) with an internal reference voltage. Mainly used in high-precision instrumentation, automotive electronics, battery voltage collection and other high-precision collection occasions. Its general function is to amplify, improve accuracy, and convert analog to digital for data analysis.[24]

## Specification: -

**Operating Voltage (VDC):** 2.0V to 5.5V

**Data Rate:** 8SPS to 860SPS

## Pin Diagram:

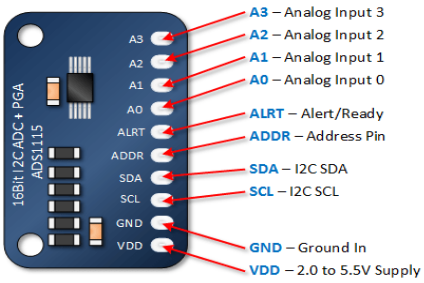



Figure -2.3 ADS1115

**2.1.4 CO gas sensor:** A carbon monoxide detector or CO detector is a device that detects the presence of the carbon monoxide (CO) gas to prevent carbon monoxide poisoning. In the late 1990s Underwriters Laboratories changed the definition of a single station CO detector with a sound device to carbon monoxide (CO) alarm.[16]

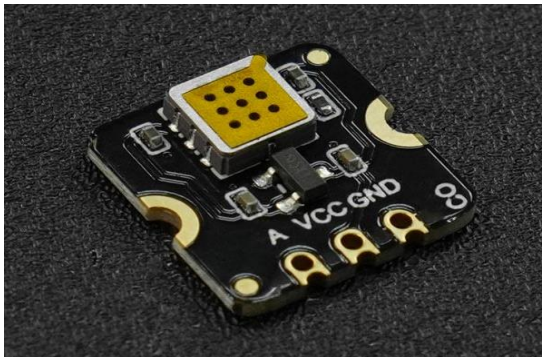



Figure-2.4 MEMS GAS SENSOR(CO)

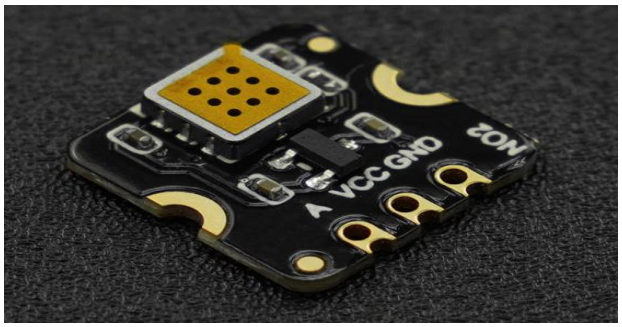



Figure-2.5 MEMS GAS SENSOR (NO2)

**2.1.5 NO2 gas sensor:** Nitrogen dioxide (NO2) gas detectors are devices used to monitor and detect the presence of NO2 gas in various commercial and industrial settings. NO2 is a toxic gas that is generated from the combustion of fossil fuels, such as in vehicle emissions and industrial processes.[17]

**2.1.6 H2S gas sensor:** Hydrogen Sulphides ($H_2S$) is a colourless toxic gas with an odour like rotten eggs. This gas naturally occurs during the chemical breakdown of organic compounds in the absence of Oxygen. $H_2S$ is highly toxic and can cause adverse effects on the human body as low as 2ppm and can be fatal at levels as low as 100ppm with prolonged exposure. $H_2S$ sensors detect dangerous levels of hydrogen sulfide gas to ensure a safe working environment. Typical areas where gas sensing, detection, and monitoring are critical. [18]

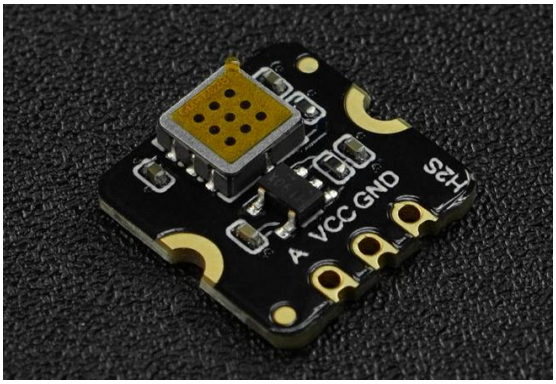
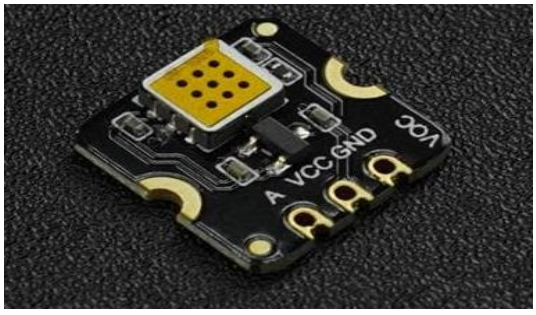



Figure-2.6 MEMS GAS SENSOR (H2S)




**2.1.7 VOC Sensor:** Volatile Organic Compounds (VOCs) are a group of carbon-based chemicals that easily evaporate at room temperature, entering the air as gases. These compounds can originate from various sources, including household products, paints, solvents, cleaning agents, fuels, and industrial processes. While some VOCs are harmless, others can be toxic and harmful to both human health and the environment. [19]

Figure-2.7 MEMS GAS SENSOR (VOC)

**2.1.8 Breadboard and Jumper Wires:** The breadboard is a white rectangular board with small embedded holes to insert electronic components. It is commonly used in electronics projects. We can also say that breadboard is a prototype that acts as a construction base of electronics.

A breadboard is derived from two words bread and board. The word breadboard was initially used to slice the bread pieces. But it was further named as a breadboard for its use in electronics around the 1970s. Hence, the term breadboard refers to these boards only and provides a quick electrical connection. Jumper wires are electrical wires with connector pins at each end.

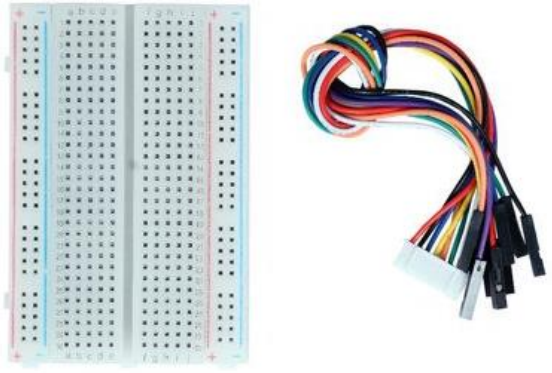


Figure-2.8 Breadboard & Jumper Wires

**2.1.9 Power supply:** The system uses a 5V linear-regulated DC power supply provided by a standard mobile adapter. This setup offers high stability, minimal EMI, and galvanic isolation, ensuring the optimal functioning of the ESP32 and connected sensors.

## 2.2 MEMS GAS SENSOR SPECIFICATION:

| Detection Gas | Range (PPM) | Operating Voltage (V) | Max Operating Current(mA) | Output Signal | Operating Temperature (°C) | Operating Humidity | Lifespan |
|---|---|---|---|---|---|---|---|
| **CO** | 5-5000 | 3.3-5 | 20 | Analog Voltage | -10-50°C | 15-90% RH | >=5 Years (Air) |
| **NO2** | 0.1-10 | 3.3-5 | 20 | Analog Voltage | -10-50°C | 15-90% RH | >=5 Years (Air) |
| **H2S** | 0.5-50 | 3.3-5 | 20 | Analog Voltage | -10-50°C | 15-90% RH | >=5 Years (Air) |
| **VOC** | 1-500 | 3.3-5 | 20 | Analog Voltage | -10-50°C | 15-90% RH | >=5 Years (Air) |

## 2.3 OVERVIEW OF SOFTWARE:

The software component of the Real-Time Air Quality Assessment Using Edge Computing and ThingSpeak project integrates various sensors, processes collected data, and presents it through a user interface. Utilizing the ESP32 microcontroller, programmed via the Arduino IDE, it manages data from multiple sensors including CO, NO2, H2S, and VOC. Here's an overview of the software aspects:

### 2.3.1. Arduino IDE Platform:

The Arduino IDE is a user-friendly platform for developing and deploying projects on Arduino boards, featuring a code editor, libraries, and tools like the Serial Monitor for debugging.

### 2.3.2. Libraries:

- **DHT Library:** Simplifies interfacing with DHT11 temperature and humidity sensors.
- **Wire Library:** Facilitates I2C communication for data exchange with I2C devices.
- **Adafruit_ADS1X15 Library:** Allows easy interfacing with ADS1015/ADS1115 ADCs for analog-to-digital conversion.
- **Wi-Fi Library:** Enables Arduino boards to connect to Wi-Fi networks for IoT applications.
- **ThingSpeak Library:** Allows data logging and retrieval from ThingSpeak's IoT platform.

### 2.3.3. Sensor Interfacing:

- **Initialization:** Setting up configurations for sensor communication.
- **Configuration:** Pin mapping, communication protocol, data rate, power settings.
- **Data Retrieval:** Extracting sensor information for analysis or control.

### 2.3.4. Data Processing:

- **Data Acquisition:** Collecting ambient air quality data from sensors.
- **Data Transmission:** Sending data to ThingSpeak for processing.
- **Data Storage and Sharing:** Storing data for historical analysis and sharing with stakeholders via APIs, apps, or web portals.

### 2.3.5. User Interface:

- **Serial Monitor**: Using Arduino IDE's Serial Monitor for debugging and real-time data viewing.
- **ThingSpeak Cloud:** Utilizing ThingSpeak for data visualization.
- **Mobile Application:** Platforms like Thingview Free for real-time monitoring on mobile devices.

### 2.3.6. Communication and Data Transmission:

- **Remote Monitoring:** Enhancing environmental awareness with minimal latency.
- **IoT Integration:** Integrating real-time air quality assessment into IoT for efficient monitoring and analysis.
- **Data Logging:** Ensuring timely and efficient environmental insights.

## CHAPTER 3:

# DIAGRAM

## 3.1 PROPOSED APPROACH

In this invention, environment is monitored based on IoT by detecting the level of gas. The proposed work can provide the physical interaction such that amount of gas is detected in the environment. These levels of gases are read accurately by the wireless sensors i.e. temperature sensor, humidity sensor, VOC sensor and gas sensor and hence no human intervention is there in the system in the sensing process. Thingspeak web server shows the graphical representation of environmental parameters such as temperature, humidity, VOC and gases such as Carbon dioxide, Carbin monoxide and Nitrogen dioxide. Monitoring team can detect and monitor the gas level without even getting exposed to the gas as the monitoring team is situated remotely and able to access the accurate data of gas level. As several environmental parameters are involved in this system, so general monitoring of the environment is also possible as in smart cities where there is continuous monitoring of environment for the well- being of the habitats. IoT has reduced the cost of the system to a greater extent, making it more flexible and scalable based on the application implemented. Sensor node is flexible to be extended along with detectors that can also be extended without so much issue.

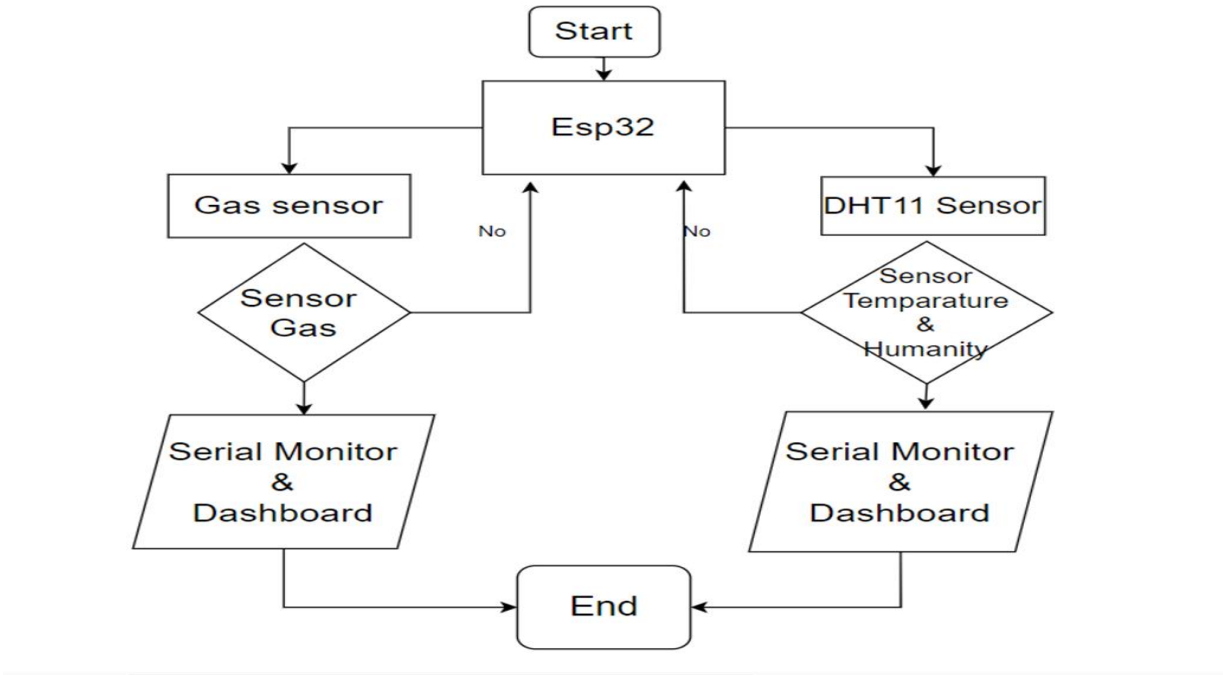



Figure-3.1 Flowchart

In this work, an IoT based environmental monitoring is focused using Thingspeak web server, where the environmental parameters are monitored by the wireless sensor network without any human intervention. In the surroundings of big factories, harmful gases leakage occurs that increases the presence of harmful gases in the environment. These gases undergo breakdown leading to the increase of temperature and decrease of humidity by which health hazard occurs. Sensor module is incorporated for sensing the variation of atmospheric parameters which is displayed using Thingspeak web server.

All the operators and experts can view the number of gases in the environment so that necessary precautions can be carried out. Five sensors are utilized for monitoring the environmental features such as humidity, temperature, nitrogen dioxide, hydrogen sulphide, carbon dioxide and carbon monoxide level . Sensor module involves temperature and humidity sensor, VOC sensor, hydrogen sulphide sensor, carbon dioxide sensor, nitrogen dioxide sensor and carbon monoxide sensor, for sensing the various parameters of the environment. Data acquisition is done using the sensor network which is then sent to the interface where controlling and manipulation of sensor data are done to provide sensor output for further processing. Microcontroller unit where the microcontroller board controls the communication process with the local network and does the whole processing of measured data provided by the interfacing sensor unit so that the user can access the collected data from remote location by using Thingspeak web server. Thingspeak web server can display the data collected from the sensors which is available free of cost. Leakage of harmful gases increase the temperature of the environment and decrease the humidity of the environment in a drastic way. Large amount of harmful gases leakage results in presence of smoke in the environment which is sensed by gas sensor. Environmental parameters change rapidly whenever level of harmful gases increase in number leading to health issues.[1]

## 3.2 CIRCUIT/ WIRENG DIAGRAM

In this Circuit Diagram DHT11 connect with ESP32 with D13 pin, and ADS1115 ADC connect with ESP32. Our Gas Sensors are also connected with ADS1115 ADC for gain voltage. The Circuit Connection will follow below.

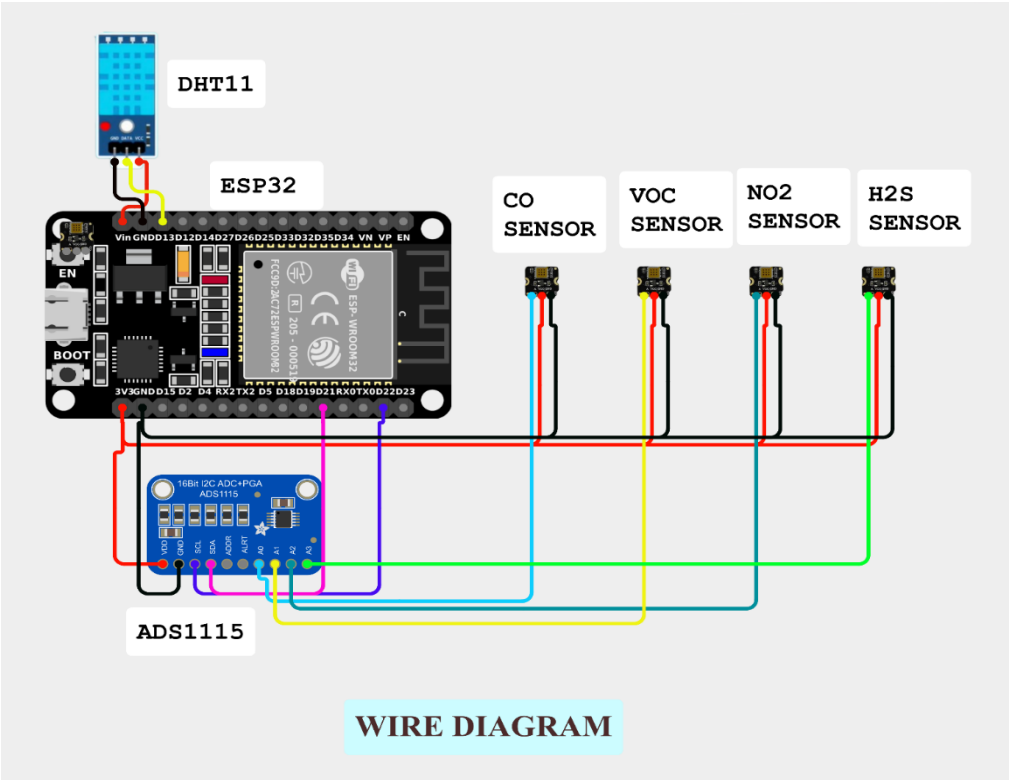



Figure-3.2 Wire Diagram

# CHAPTER 4:

## 4.1. HARDWARE CONNECTIONS

### 4.1.1 DHT11 Sensor to ESP32:

| DHT11 pin | ESP32 pin |
|---|---|
| VCC | Vin |
| GND | GND |
| DATA | D13 |

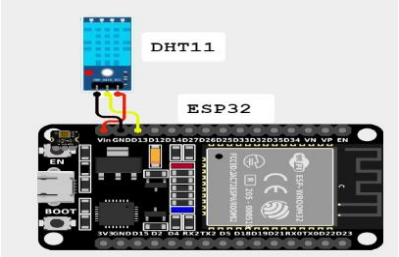



Figure-4.1 DHT11&ESP32

### 4.1.2 ADS1115 ADC to ESP32:

| ADS1115 pin | ESP32 pin |
|---|---|
| VDD | 3V3 |
| GND | GND |
| SCL | D22(GPIO22) |
| SDA | D21(GPIO21) |

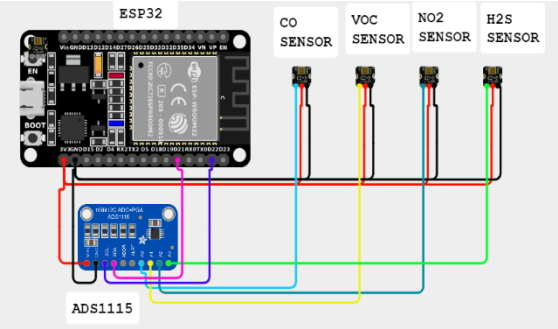



Figure – 4.2 ADS1115&ESP32

&MEMS Gas Sensor

### 4.1.3 MEMS Gas Sensors to ADS1115:

| MEMS Gas Sensor | ADS1115 pin |
|---|---|
| CO | A0 |
| VOC | A1 |
| NO2 | A2 |
| H2S | A3 |

*** All Gas Sensor VCC pin connect with 3V3 on ESP32
& GND pin Connect with GND on ESP32

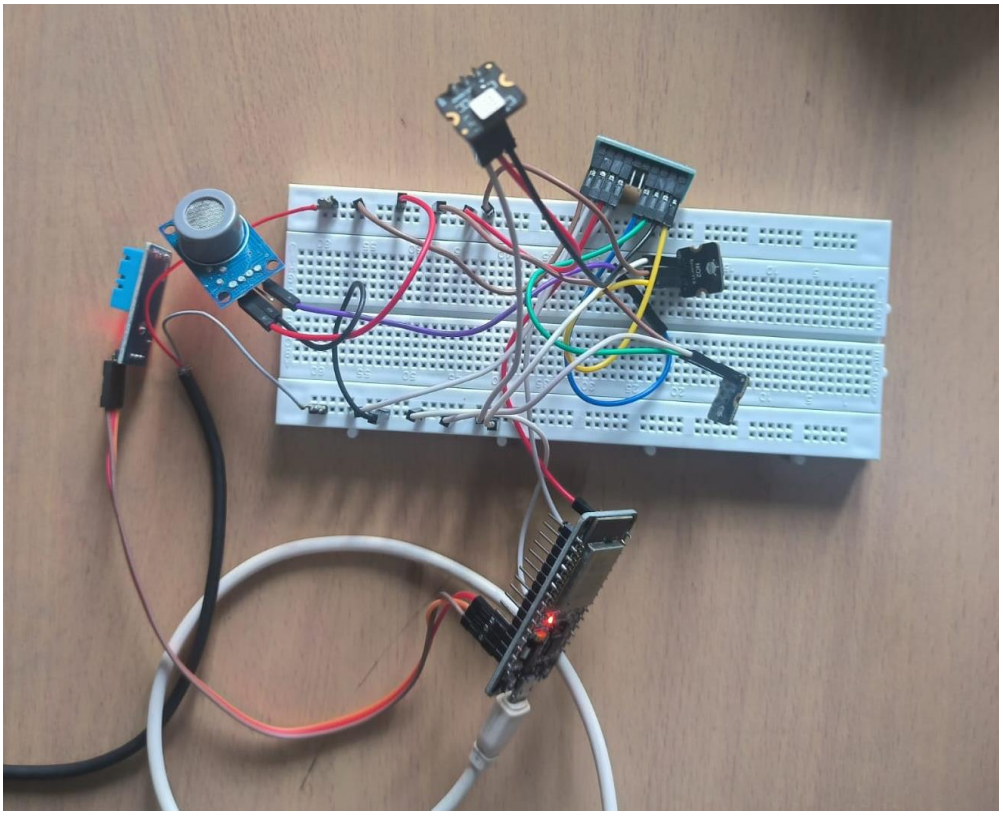


Figure 4.3 Breadboard setup

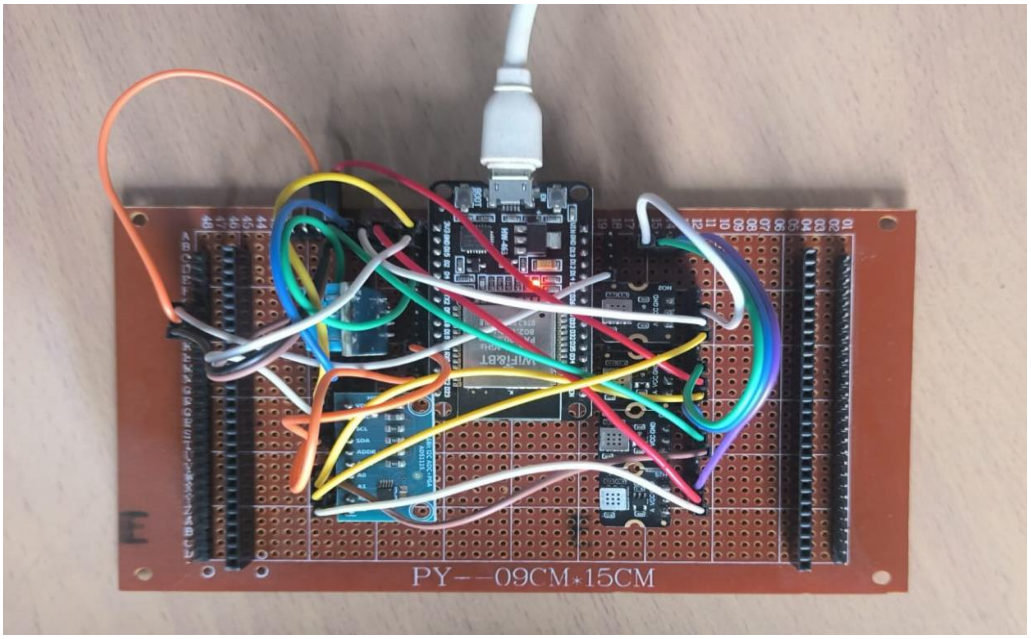



Figure-4.4 Hardware setup

## Chapter 5:

# WORKING PRINCIPLE

The real-time air quality assessment system using edge computing and ThingSpeak operates through a well-defined process involving initialization, sensor readings, gas concentration measurement, data processing, and continuous monitoring. The core components include the ESP32 microcontroller, DHT11 sensor, and ADS1115 ADC, which work together to ensure accurate and efficient environmental monitoring.

## 5.1. INITIALIZATION

The initialization phase involves setting up the ESP32 microcontroller and configuring the sensors to ensure proper communication and data handling.

- **Configuring GPIO Pins:**
  - All sensors are correctly connected to the ESP32's GPIO pins. This step is crucial to ensure that each sensor can send its data to the microcontroller without interference.
  - Specific pins are designated for each sensor based on the microcontroller's pin layout and the sensors' requirements.

- **Library Initialization:**
  - The required libraries for sensor communication and data handling are initialized. These libraries provide predefined functions and methods to simplify interaction with the sensors.
  - For instance, the DHT library facilitates easy communication with the DHT11 sensor, while the Wire library enables I2C communication for the ADS1115 ADC.

- **Communication Protocols:**
  - Setting up the I2C protocol for communication between the ESP32 and the ADS1115 ADC is essential. The I2C protocol allows multiple devices to communicate with the ESP32 using just two wires (SDA and SCL).
  - This setup ensures efficient and reliable data transmission from the analog sensors to the microcontroller.

## 5.2. SENSOR READINGS

The DHT11 sensor is responsible for measuring temperature and humidity.

- **DHT11 Measurement:**
  - The sensor reads the ambient temperature and humidity at regular intervals. It uses its built-in sensor elements to measure these parameters accurately.
  - The data is then prepared for transmission to the ESP32.

- **Data Transmission:**
  - The DHT11 sensor transmits the data to the ESP32 as a digital signal. The ESP32 reads this signal using a designated GPIO pin and stores the values in variables for further processing.
  - The DHT library functions are utilized to simplify the reading process and ensure accurate data retrieval.

## 5.3. GAS SENSOR CONCENTRATION

Various gas sensors, such as those for CO, NO2, H2S, and VOCs, detect the presence and concentration of different gases in the environment. These sensors produce an analog voltage that is proportional to the gas concentration.

- **Analog Signal from Sensors:**
  - Each gas sensor provides an analog output corresponding to the concentration of the target gas. The output voltage varies linearly with the gas concentration.
  - This analog signal needs to be converted into a digital format that the ESP32 can process.

- **Analog-to-Digital Conversion:**
  - The ADS1115 ADC reads these analog signals and converts them into digital values. This conversion is essential because the ESP32 can only process digital data.
  - The ADS1115 provides high-resolution digital readings, which are then sent to the ESP32 over the I2C protocol. The high resolution of the ADS1115 (up to 16 bits) ensures that even small changes in gas concentration are detected and accurately represented in the digital data.
  - The ADS1115 also offers programmable gain, allowing the system to adjust the sensor input range dynamically. This feature is particularly useful for enhancing the precision of measurements across different gas concentration levels.
  - Additionally, the ADS1115's multiplexer can switch between multiple analog inputs, enabling the system to monitor several gas sensors using a single ADC. This capability simplifies the hardware design and reduces costs.

## 5.4. DATA PROCESSING

The ESP32 processes the sensor readings and gas concentration values to convert them into meaningful units that can be easily interpreted.

- **Calibration:**
  - Calibration formulas or algorithms are applied to the raw sensor data to obtain accurate measurements. This step compensates for any sensor inaccuracies or environmental factors.
  - Calibration ensures that the data reflects the true environmental conditions.

- **Data Conversion:**
  - The analog readings are converted into standardized units such as parts per million (ppm) for gas concentration. This conversion allows for consistent and comparable measurements.
  - Standardized units facilitate easier interpretation and comparison of the data.

- **Filtering and Smoothing:**
  - Techniques are implemented to improve data accuracy and stability, such as filtering out noise and smoothing out fluctuations. This step ensures that the data is reliable and reflects true environmental conditions.
  - Filtering algorithms remove spurious data points, while smoothing algorithms reduce short-term variations.

## 5. CONTINUOUS MONITORING

The system continuously monitors and updates the sensor readings in real-time. This ongoing process allows for the timely detection of changes in environmental conditions.

- **Real-time Data Acquisition:**
  - Sensors are regularly polled to obtain the latest readings. The frequency of polling can be adjusted based on the specific requirements of the application.
  - Real-time data acquisition ensures that the system can quickly respond to changes in environmental conditions.

- **Live Data Updates:**
  - The current values are continuously updated and displayed. This real-time display allows users to monitor environmental conditions as they change.
  - Live data updates provide immediate insights into air quality, enabling quick decision-making.

- **Data Upload to ThingSpeak:**
  - The processed data is transmitted to the ThingSpeak cloud platform for remote access and monitoring. ThingSpeak provides tools for data visualization, analysis, and storage.
  - Users can access the data from anywhere, facilitating remote monitoring and management of environmental conditions.

The working principles of this temperature, humidity, and gas detection system using the ESP32 microcontroller, DHT11 sensor, and ADS1115 ADC provide a comprehensive solution for environmental monitoring. The system effectively measures and processes data, ensuring accurate and continuous tracking of temperature, humidity, and gas concentration levels. Real-time monitoring through ThingSpeak allows for remote access and ongoing assessment, making the system highly valuable for various applications in environmental monitoring and safety systems. This setup ensures that users can maintain optimal conditions and ensure safety in diverse environments, leveraging advanced edge computing and IoT capabilities for efficient and effective monitoring.

# CHAPTER 6

# RESULT ANALYSIS

## 6.1. DATA COLLECTION AND INTEGRATION

The integration of the ESP32 microcontroller with the DHT11 sensor and ADS1115 ADC allowed for successful acquisition of environmental data. The collected data includes:

• **Temperature and Humidity:** Measured by the DHT11 sensor.

• **Gas Concentrations:** Measured by the gas sensors (CO, H2S, VOC, and NO2) interfaced through the ADS1115 ADC.

These data points were periodically uploaded to the ThingSpeak cloud platform, enabling remote monitoring and analysis.

### 6.1.1. Temperature and Humidity

The DHT11 sensor provided consistent readings for temperature and humidity. The data was recorded over several days, and the following observations were made:

• **Temperature Variations**: The temperature readings showed expected daily fluctuations corresponding to environmental changes, demonstrating the sensor's reliability in capturing ambient temperature.

• **Humidity Trends:** Humidity levels also displayed typical variations, influenced by factors such as weather conditions and time of day.

The data was visualized on ThingSpeak, revealing clear patterns and trends. For instance, temperature peaks were observed during the afternoon, while humidity levels tended to rise at night.

### 6.1.2. Gas Concentrations

The gas sensors connected through the ADS1115 ADC provided valuable insights into the presence and concentration of various gases:

• **CO Sensor:** Detected carbon monoxide levels with accuracy. Elevated CO levels were observed in environments with poor ventilation or during combustion activities.

• **H2S Sensor:** Showed responsiveness to hydrogen sulphide, particularly in areas with decaying organic matter or industrial emissions.

• **VOC Sensor:** Detected volatile organic compounds effectively, with higher readings in areas with significant human activity or chemical usage.

• **NO2 Sensor:** Monitored nitrogen dioxide levels, capturing variations linked to traffic emissions and industrial activities.

Each sensor's output was analysed by comparing the analog readings from the ADS1115 with known concentration levels. Calibration ensured the accuracy of these measurements.

## 6.2. DATA PROCESSING AND CALIBRATION

Calibration was crucial in converting raw analog signals into meaningful gas concentration values. The calibration process involved:

• **Baseline Measurement:** Establishing a baseline for sensor readings in a clean air environment.

• **Calibration Curves:** Using known gas concentrations to create calibration curves, mapping analog values to actual gas concentrations.

• **Data Smoothing:** Applying filtering techniques to reduce noise and improve the reliability of the readings.

This data processing ensured that the environmental measurements were accurate and could be reliably used for further analysis.

## 6.3. REMOTE MONITORING

Uploading the data to ThingSpeak facilitated remote monitoring, offering several advantages:

• **Real-time Access:** Users could access current sensor readings from anywhere, enhancing the system's practicality for remote applications.

• **Historical Data:** ThingSpeak's data storage allowed for historical analysis, enabling the identification of long-term trends and patterns.

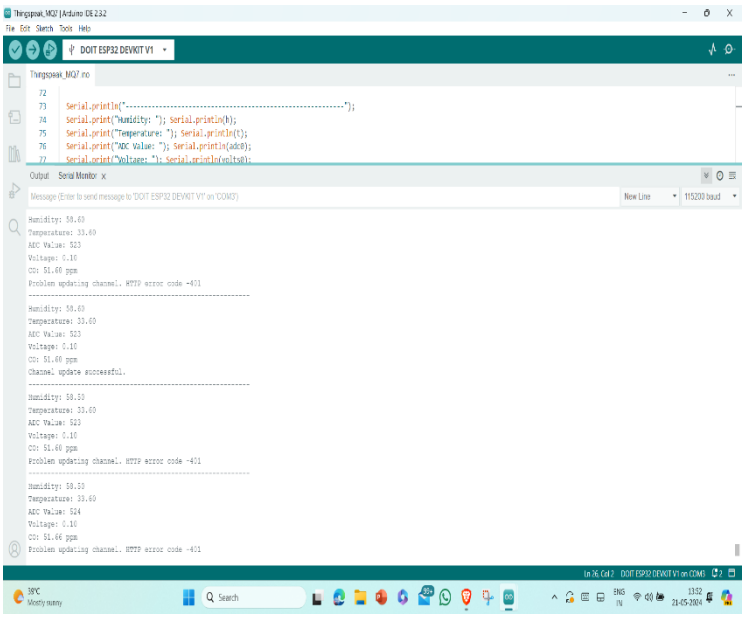


Figure 5.1 Serial Monitor

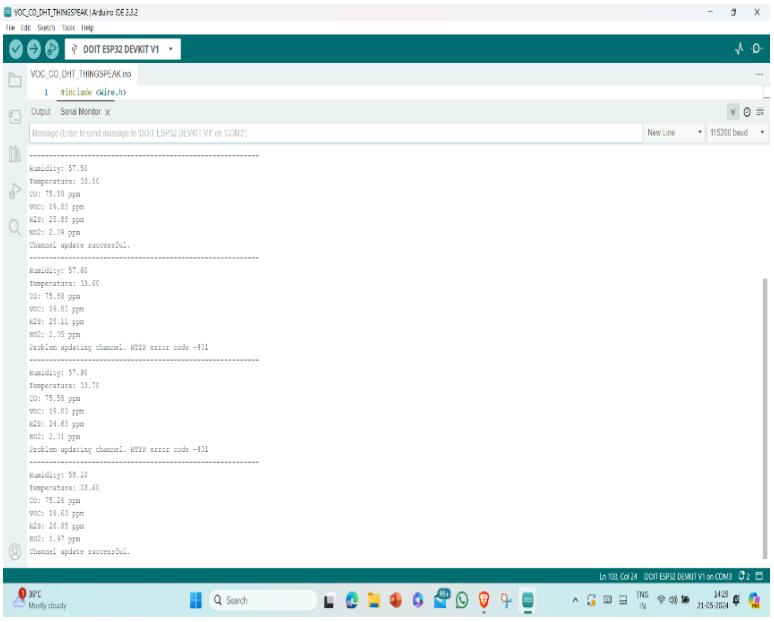


Figure 5.1 Serial Monitor

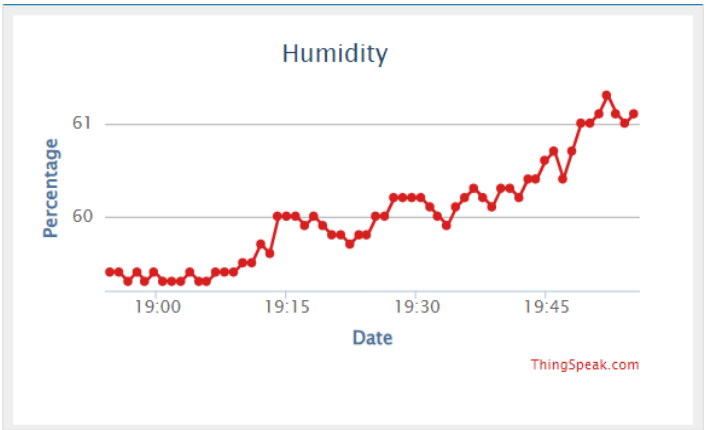



Figure 5.3 Thingspeak - Humidity

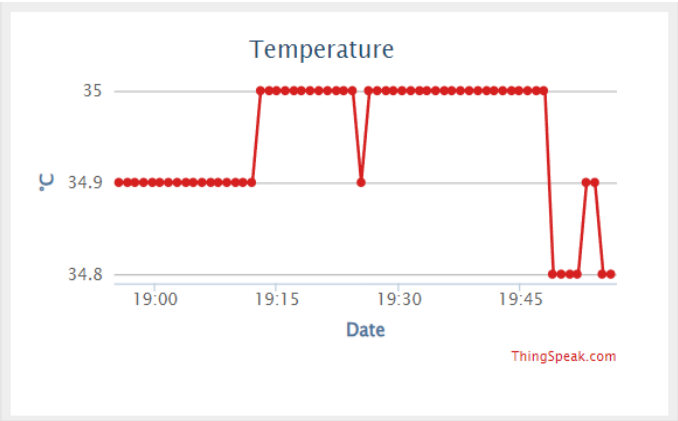



Figure 5.4 Thingspeak - Temperature

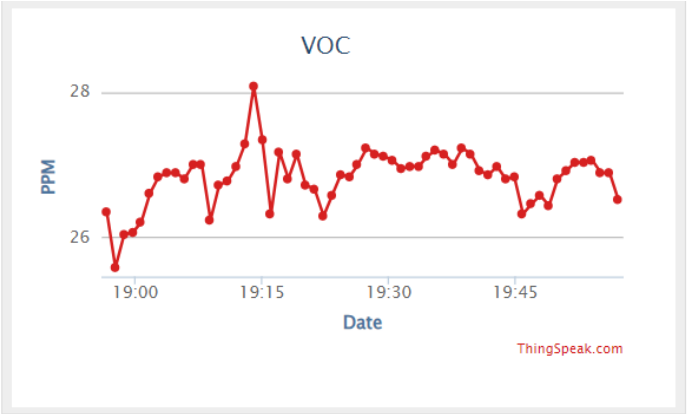



Figure 5.5 Thingspeak - VOC

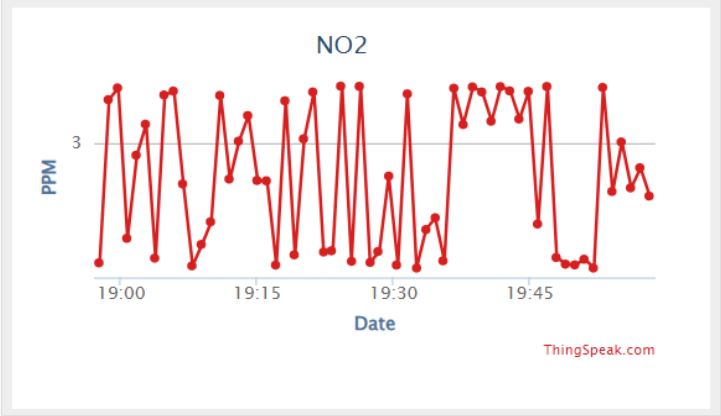



Figure 5.6 Thingspeak – NO2

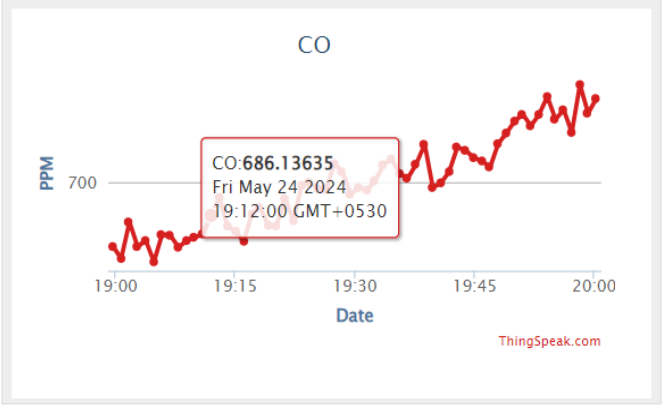



Figure 5.7 Thingspeak - CO

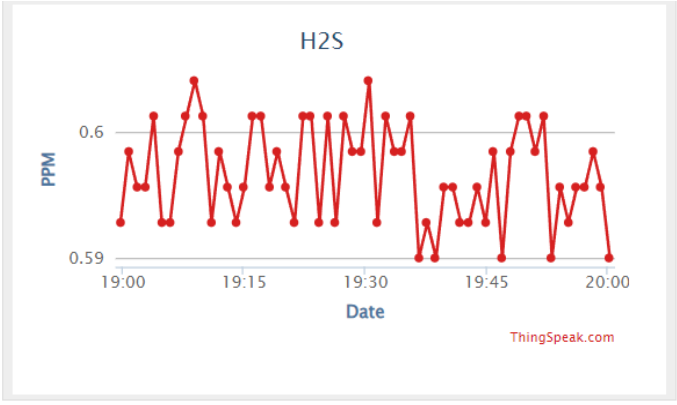



Figure 5.8 Thingspeak – H2S

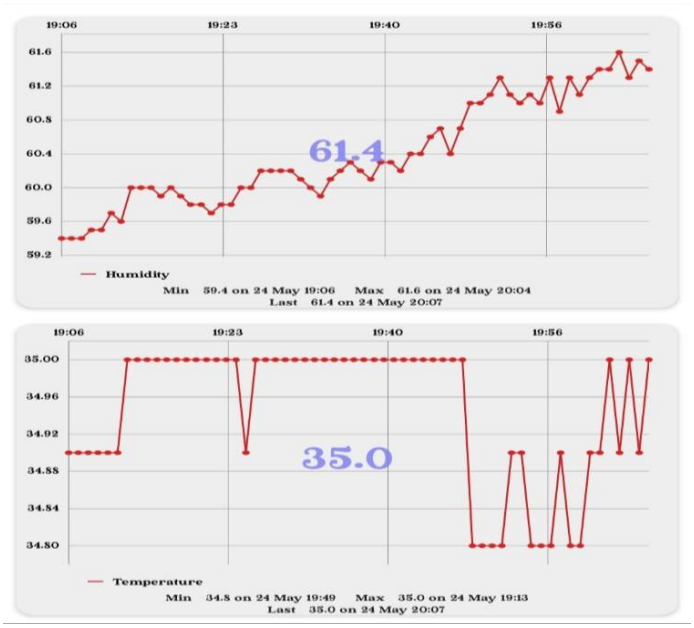



Figure 5.9 Thinkview-Temperature and Humidity

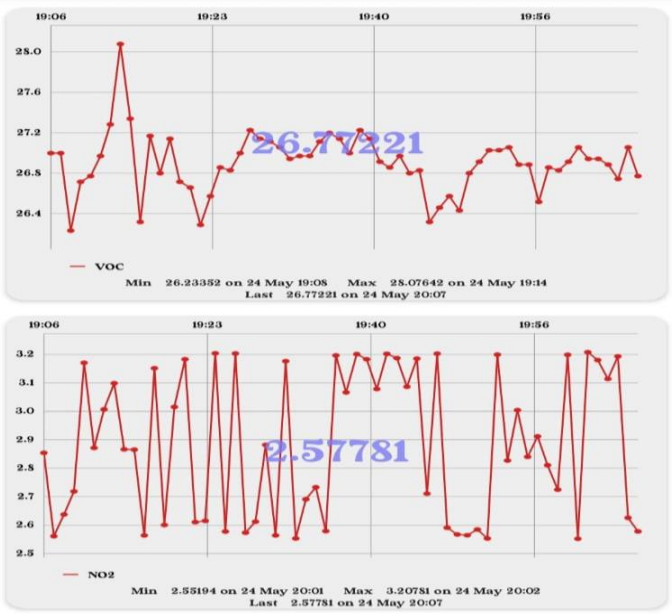



Figure 5.10 Thinkview- VOC and NO2

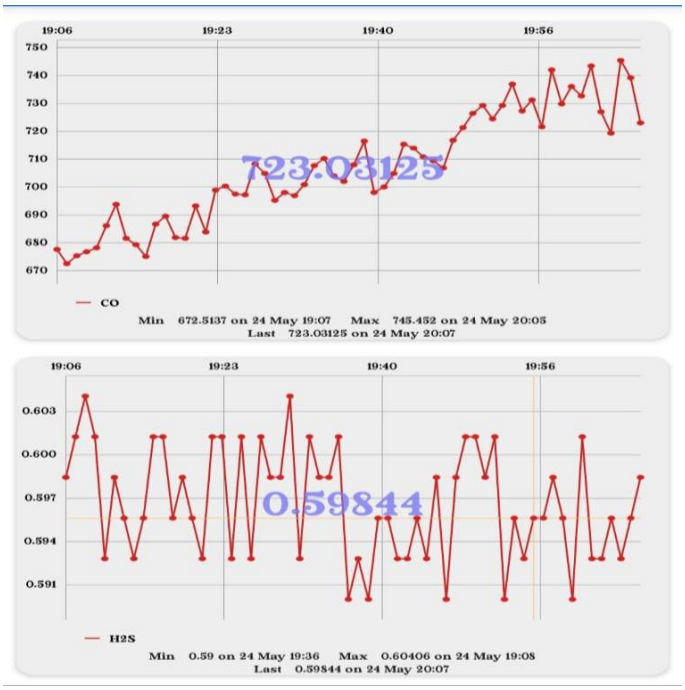



Figure 5.11 Thinkview-CO and H2S

## CHAPTER 7:

# CONCLUSION

In conclusion, the implementation of the temperature, humidity, and gas detection system using ESP32, DHT11, CO gas sensor, H2S gas sensor, NO2 gas sensor, VOC sensor in Arduino IDE provides valuable insights into the environmental conditions and gas presence. Through the integration of these sensors and the development of the software code, the system can measure and monitor temperature, humidity, and gas concentration levels.

## 7.1 KEY ACHIEVEMENTS:

- **Accurate and Real-Time Data Collection:**

The integration of DHT11 for temperature and humidity measurements, alongside the ADS1115 ADC interfacing with CO, H2S, VOC, and NO2 sensors, enabled comprehensive environmental monitoring.

- **Reliable Sensor Performance:**

Real-time data acquisition was consistently accurate, with sensors responding reliably to environmental changes and gas concentrations.

- **Effective Data Processing and Calibration:**

Calibration processes ensured that sensor readings were converted into meaningful and precise values.

- **Enhanced Data Accuracy:**

Data smoothing and filtering techniques further enhanced the accuracy and reliability of the collected data, addressing potential noise and inconsistencies.

- **Seamless Remote Monitoring with ThingSpeak:**

ThingSpeak provided a powerful platform for remote data visualization, storage, and analysis.

- **Global Data Accessibility:**

The capability to access real-time and historical data from anywhere facilitated continuous monitoring and assessment of air quality.

- **Safety and Responsiveness:**

The potential for implementing alerts and notifications based on specific conditions (e.g., high levels of CO) adds a critical layer of safety and responsiveness.

- **Edge Computing Advantages:**

Utilizing the ESP32 for edge computing allowed for local data processing, reducing latency and ensuring faster response times.

- **Efficient System Performance:**

This approach minimized the need for constant cloud communication, conserving bandwidth and making the system more efficient.

## 7.2 FUTURE SCOPE:

- **Air Pollution Monitoring:**

Enhanced systems can be developed for urban and industrial areas to monitor air quality, identify pollution sources, and provide data for regulatory compliance and public health safety.

- **Poultry Farm:**

In poultry farms, ammonia levels can be harmful to the health of chickens. Real-time remote ambient gas monitoring can be used to track ammonia levels in poultry farms and ensure that they stay within safe limits.

- **Coal Mine:**

Coal mines can be dangerous places due to the presence of methane gas. Real-time remote ambient gas monitoring can be used to detect methane leaks and prevent explosions.

- **Advanced Environmental Monitoring System:**

Real-time remote ambient gas monitoring can be used to create advanced environmental monitoring systems that can track a variety of pollutants and environmental conditions. This information can be used to assess the overall health of an environment and identify potential problems.

- **Alerts & Pollution Control:**

Real-time remote ambient gas monitoring can be used to improve pollution control efforts. By tracking pollution levels in real time, authorities can take steps to reduce pollution sources.

- **Smart City Integration:**

Real-time remote ambient gas monitoring can be integrated into smart city initiatives. This information can be used to improve air quality, public health, and safety in cities.

- **Agricultural Applications:**

Real-time remote ambient gas monitoring can be used in agricultural applications to track levels of gases such as ammonia and methane. This information can be used to improve manure management practices and reduce greenhouse gas emissions.

- **Public Health and Awareness:**

Real-time remote ambient gas monitoring can be used to raise public awareness of air quality issues. By making this information available to the public, people can make informed decisions about their health and safety.

# List of Figures

Figure 2.1: ESP 32 …….....Page 3
Figure 2.2: DHT 11……...…….....Page 4
Figure 2.3 ADS1115……..……..Page 4
Figure 2.4 MEMS gas sensor (CO)…….…….……..Page 5
Figure 2.5 MEMS gas sensor (NO2) ……..…. .. Page 5
Figure 2.6 MEMS gas sensor (H2S) …….…….Page 5
Figure 2.7 MEMS gas sensor (VOC)……..…Page 5
Figure 2.8 Breadboard and Jumper wires……..…..Page 6
Figure 3.1 Flow Chart…….Page 8
Figure 3.2 Wire Diagram……..…..….Page 9
Figure 4.1 DHT11&ESP32 ……...Page 10
Figure 4.2 ADS1115&ESP32 &MEMS Gas Sensors ……...Page 10
Figure 4.3 Breadboard setup……...Page 11
Figure 4.4 Hardware setup……..Page 11
Figure 5.1 Serial Monitor…….…….....Page 16
Figure 5.2 Serial Monitor……...Page 16
Figure 5.3 Thinkspeak - Humidity……. Page 17
Figure 5.4 Thinkspeak – Temperature……..…...Page 17
Figure 5.5 Thinkspeak – VOC……...… Page 17
Figure 5.6 Thinkspeak – NO2…….. …. Page 17
Figure 5.7 Thinkspeak – CO……..…… Page 17
Figure 5.8 Thinkspeak – H2S…….…… Page 17
Figure 5.9 Thinkview-Temperature and Humidity…..…….……. Page 18
Figure 5.10 Thinkview- VOC & NO2.……Page 18
Figure 5.11 Thinkview- CO & H2S…… ……. Page 18

## List of Abbreviation

**MEMS** - Microelectromechanical system

**DHT** - Digital Humidity and Temperature

**H2S** - Hydrogen sulphide

**NO2** - Nitrogen dioxide

**CO** - Carbon Monoxide

**VOC** - Volatile organic compounds

**APPENDIX A**

**Code for MEMS_DHT_THINGSPEAK_HARDWARE**

```
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <WiFi.h>
#include <ThingSpeak.h>
#include "DHT.h"

// Replace with your network credentials
char* ssid ="CGEC"; // enter SSID
char* password ="cgec@9830"; // enter password

// ThingSpeak settings
unsigned long myChannelNumber = 8;
const char* myWriteAPIKey = "CV57KJXKBVJ1EBPB";

Adafruit_ADS1115 ads;  /* Use this for the 16-bit version */

WiFiClient client;

#define DHTPIN 4 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT
DHT dht(DHTPIN, DHTTYPE);

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

void setup() {
  Serial.begin(115200);
  Serial.println("Initializing...");

  // Initialize the DHT sensor
  dht.begin();

  // Initialize the ADS1115
  if (!ads.begin()) {
    Serial.println("Failed to initialize ADS.");
    while (1);
  }
  Serial.println("ADS1115 initialized.");

  // Connect to Wi-Fi
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("\nWiFi connected");

  // Initialize ThingSpeak
  ThingSpeak.begin(client);
  Serial.println("ThingSpeak initialized");
}

void loop() {
  // Read humidity and temperature from DHT11 sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);

  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    delay(2000); // Delay before trying again
    return; // Skip the rest of the loop if DHT reading fails
  }

  // CO
  int16_t adc0 = ads.readADC_SingleEnded(0);
  float volts0 = ads.computeVolts(adc0); // Convert to voltage (assuming default gain)
  float co_ppm = mapfloat(volts0, 0.0, 3.3, 5.0, 5000.0);

  //VOC
  int16_t adc1 = ads.readADC_SingleEnded(1);
  float volts1 = ads.computeVolts(adc1); // Convert to voltage (assuming default gain)
  float voc_ppm = mapfloat(volts1, 0.0, 3.3, 1.0, 500.0);

  //NO2
  int16_t adc2 = ads.readADC_SingleEnded(2);
  float volts2 = ads.computeVolts(adc2); // Convert to voltage (assuming default gain)
  float no2_ppm = mapfloat(volts2, 0.0, 3.3, 0.1, 10.0);

  //H2S
  int16_t adc3 = ads.readADC_SingleEnded(3);
  float volts3 = ads.computeVolts(adc3); // Convert to voltage (assuming default gain)
  float h2s_ppm = mapfloat(volts3, 0.0, 3.3, 0.5, 50.0);

  Serial.println("-------------------------------------------------------");
  Serial.print("Humidity: "); Serial.println(h);
  Serial.print("Temperature: "); Serial.println(t);
  //Serial.print("ADC Value: "); Serial.println(adc0);
  //Serial.print("Voltage: "); Serial.println(volts0);
  Serial.print("CO: "); Serial.print(co_ppm); Serial.println(" ppm");
  Serial.print("VOC: "); Serial.print(voc_ppm); Serial.println(" ppm");
  Serial.print("H2S: "); Serial.print(h2s_ppm); Serial.println(" ppm");
  Serial.print("NO2: "); Serial.print(no2_ppm); Serial.println(" ppm");

  // Send data to ThingSpeak
```

```
 ThingSpeak.setField(1, h);
 ThingSpeak.setField(2, t);
 ThingSpeak.setField(6, co_ppm);
 ThingSpeak.setField(3, voc_ppm);
 ThingSpeak.setField(7, h2s_ppm);
 ThingSpeak.setField(5, no2_ppm);
 int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

 if (x == 200) {
  Serial.println("Channel update successful.");
 } else {
  Serial.println("Problem updating channel. HTTP error code " + String(x));
 }

 // Wait 1 miniutes to update again
 delay(60000);
}
```

**APPENDIX B**

**Code for ADS**

```
#include <Adafruit_ADS1X15.h>

Adafruit_ADS1115 ads;  /* Use this for the 16-bit version */

//Adafruit_ADS1015 ads;     /* Use this for the 12-bit version */

// Function to map the value from one range to another

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max) {

  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;}

void setup(void)

{ Serial.begin(115200);

  Serial.println("Getting single-ended readings from AIN0..3");

  Serial.println("ADC Range: +/- 6.144V (1 bit = 3mV/ADS1015, 0.1875mV/ADS1115)");

  if (!ads.begin())

  {  Serial.println("Failed to initialize ADS.");

 while (1)}}
```

```
void loop(void)
{int16_t adc0, adc1, adc2, adc3;
  float volts0, volts1, volts2, volts3;
  adc0 = ads.readADC_SingleEnded(0);
  adc1 = ads.readADC_SingleEnded(1);
  adc2 = ads.readADC_SingleEnded(2);
  adc3 = ads.readADC_SingleEnded(3);

  volts0 = ads.computeVolts(adc0);
  volts1 = ads.computeVolts(adc1);
  volts2 = ads.computeVolts(adc2);
  volts3 = ads.computeVolts(adc3);

  // Map ADC values to desired ranges
  float co_ppm = mapfloat(volts0, 0.0, 6.144, 5.0, 5000.0);
  float no2_ppm = mapfloat(volts1, 0.0, 6.144, 0.1, 10.0);
  float h2s_ppm = mapfloat(volts2, 0.0, 6.144, 0.5, 50.0);
  float voc_ppm = mapfloat(volts3, 0.0, 6.144, 1.0, 500.0);

  Serial.println("-----------------------------------------------------------");
  Serial.print("CO: "); Serial.print(co_ppm); Serial.println(" ppm");
  Serial.print("NO2: "); Serial.print(no2_ppm); Serial.println(" ppm");
  Serial.print("H2S: "); Serial.print(h2s_ppm); Serial.println(" ppm");
  Serial.print("VOC: "); Serial.print(voc_ppm); Serial.println(" ppm");
delay(5000);}
```

**APPENDIX C**

**Code for Gas Sensor**

```
int sensorPin_NO2 = 32; //D32 pin of ESP32
int sensorValue_NO2 = 0;

int sensorPin_CO = 33; //D33 pin of ESP32
int sensorValue_CO = 0;

int sensorPin_H2S = 27; //D27 pin of ESP32
int sensorValue_H2S = 0;

int sensorPin_VOC = 4; //D4 pin of ESP32
int sensorValue_VOC = 0;
void setup() {
  // put your setup code here, to run once:
 Serial.begin(115200);
}

void loop() {
  //Read VOC Sensor Data
  sensorValue_NO2 =analogRead(sensorPin_NO2); //Range 0.1-10 ppm
  sensorValue_CO =analogRead(sensorPin_CO); //Range 5-5000 ppm
  sensorValue_H2S =analogRead(sensorPin_H2S); //Range 0.5-50 ppm
  sensorValue_VOC =analogRead(sensorPin_VOC); //Range 1-500 ppm
  +
  Serial.print( "    NO2 : ");
  Serial.print( sensorValue_NO2);
  Serial.print("  PPM");

  Serial.print( "    CO : ");
  Serial.print( sensorValue_CO);
  Serial.print("  PPM");

  Serial.print( "    H2S : ");
  Serial.print( sensorValue_H2S);
  Serial.print("  PPM");

  Serial.print( "    VOC : ");
  Serial.print( sensorValue_VOC);
  Serial.println("  PPM");

  delay(5000);}
```

# REFERENCE


1. Sindhwani, N., Anand, R., Vashisth, R., Chauhan, S., Talukdar, V., & Dhabliya, D. (2022, November). Thingspeak-based environmental monitoring system using IoT. In *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 675-680). IEEE.

2. Gayathri, K. (2019, March). Implementation of environment parameters monitoring in a manufacturing industry using IoT. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)* (pp. 858-862). IEEE.

3. Lobur, M., Korpyljov, D., Jaworski, N., Iwaniec, M., & Marikutsa, U. (2020, April). Arduino based ambient air pollution sensing system. In *2020 IEEE XVIth International conference on the perspective technologies and methods in MEMS design (MEMSTECH)* (pp. 32-35). IEEE.

4. Phala, K. S. E., Kumar, A., & Hancke, G. P. (2016). Air quality monitoring system based on ISO/IEC/IEEE 21451 standards. *IEEE Sensors Journal*, *16*(12), 5037-5045.

5. Shum, L. V., Rajalakshmi, P., Afonja, A., McPhillips, G., Binions, R., Cheng, L., & Hailes, S. (2011, April). On the Development of a Sensor Module for real-time Pollution Monitoring. In *2011 International Conference on Information Science and Applications* (pp. 1-9). IEEE.

6. Agnihotri, P., Tiwari, S., & Mohan, D. (2020, February). Design of Air pollution monitoring system using wireless sensor network. In *2020 International Conference on Electrical and Electronics Engineering (ICE3)* (pp. 33-38). IEEE.

7. Kadri, A., Yaacoub, E., Mushtaha, M., & Abu-Dayya, A. (2013, February). Wireless sensor network for real-time air pollution monitoring. In *2013 1st international conference on communications, signal processing, and their applications (ICCSPA)* (pp. 1-5). IEEE.

8. Kim, S. H., Jeong, J. M., Hwang, M. T., & Kang, C. S. (2017, October). Development of an IoT-based atmospheric environment monitoring system. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 861-863). IEEE.

9. Mokrani, H., Lounas, R., Bennai, M. T., Salhi, D. E., & Djerbi, R. (2019, August). Air quality monitoring using iot: A survey. In *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)* (pp. 127-134). IEEE.

10. Kanál, A. K., & Tamás, K. (2020, May). Assessment of indoor air quality of educational facilities using an IoT solution for a healthy learning environment. In *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)* (pp. 1-6). IEEE.

11. Aamer, H., Mumtaz, R., Anwar, H., & Poslad, S. (2018, October). A very low cost, open, wireless, internet of things (iot) air quality monitoring platform. In *2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT)* (pp. 102-106). IEEE.

12. Azirani, M. A., Kuhnke, M., Werle, P., & Sorgatz, W. (2018, September). Online Fault Gas Monitoring System for Hermetically Sealed Power Transformers. In *2018 Condition Monitoring and Diagnosis (CMD)* (pp. 1-5). IEEE.

13. Engel, L., Tarantik, K. R., Benito-Altamirano, I., Pannek, C., Prades, J. D., & Wöllenstein, J. (2019, July). Screen-Printable Colorimetric Sensors for the Monitoring of Toxic Gases in Ambient Air. In *2019 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)* (pp. 1-3). IEEE.

14. Parida, D., Behera, A., Naik, J. K., Pattanaik, S., & Nanda, R. S. (2019, May). Real-time environment monitoring system using ESP8266 and thingspeak on internet of things platform. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)* (pp. 225-229). IEEE.

15. https://www.espressif.com/en/products/socs/esp32

16. https://wiki.dfrobot.com/SKU_SEN0564_Fermion_MEMS_CO_Sensor_breakout

17. https://wiki.dfrobot.com/SKU_SEN0574_Fermion_MEMS_NO2_Sensor_breakout

18. https://wiki.dfrobot.com/SKU_SEN0568_Fermion_MEMS_H2S_Sensor_breakout

19. https://wiki.dfrobot.com/SKU_SEN0566_Fermion_MEMS_VOC_Sensor_breakout

20. https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

21. https://randomnerdtutorials.com/esp32-thingspeak-publish-arduino/

22. https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-web-server-arduino-ide/

23. https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/

24. https://how2electronics.com/how-to-use-ads1115-16-bit-adc-module-with-esp32/

25. https://www.electronicshub.org/esp32-pinout/