A close up of a logo

Description generated with very high confidence

**Smog Track: Real-Time Air Quality Monitoring**

By

**Dhanush V (2140148)**

**Rahul Kalaburgi (2140149)**

**Ramya R (2140171)**

Under the guidance of

**Prof. MUKUND N**

Department of Physics and Electronics

**A project report submitted in partial fulfillment for the award of degree of B.Sc. (Electronics) of CHRIST (Deemed to be University), Bengaluru.**

**www.christuniversity.in**

**April 2024**

A close up of a logo

Description generated with very high confidence

**Certificate**

This is to certify that, the project titled “**SmogTrack: Real-Time Air Quality Monitoring”** is a bonafide record of the work done by **“Ramya R”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Science (Electronics) of CHRIST (Deemed to be University), Bengaluru during the year 2023 - 24.

**Head (Electronics) Project Guide**

(Prof. Benny Sebastian) (Prof. Mukund K N)

Examination Date: **Name of the Student:** Ramya R

Examination Centre: Christ University **Class:** VI Semester CME

**Register No.:** 2140171

**Examiners**

1. 2.

# ACKNOWLEDGMENT

We extend our deepest gratitude and appreciation to the esteemed Vice Chancellor of Christ University, whose unwavering support and guidance have been instrumental in shaping our academic journey and fostering an environment of excellence.

we express our sincere gratitude to Dr. BENNY SEBASTAIN. HOD, Department of Physics and Electronics, Christ University, Bengaluru for his stimulating guidance, continuous encouragement, and motivation throughout the present work.

We also wish to extend my thanks to our Project Co-Ordinator Dr. MUKUND K N Professor, Department of Physics and Electronics, Christ University, Bengaluru for his unstilted support, valuable guidance, and help throughout the seminar-related work.

We also wish to thank my Guide Dr. MUKUND K N, Professor, Department of Physics and Electronics, Christ University, Bengaluru for his critical, insightful comments, guidance, and constructive suggestions and support to improve the quality of this seminar work.

We extend our heartfelt gratitude to our esteemed friends whose unwavering encouragement, camaraderie, and shared experiences have profoundly enriched our journey at Christ University. Their steadfast support, through the peaks and valleys of academic life, has served as a beacon of strength and inspiration. We are indebted to their enduring companionship, which has truly enhanced our collegiate experience

We would like to thank all the faculty members and non-teaching staff of the Department of Physics and Electronics, for their constant support. And I would like to thank our parents and friends for their constant moral and financial support.

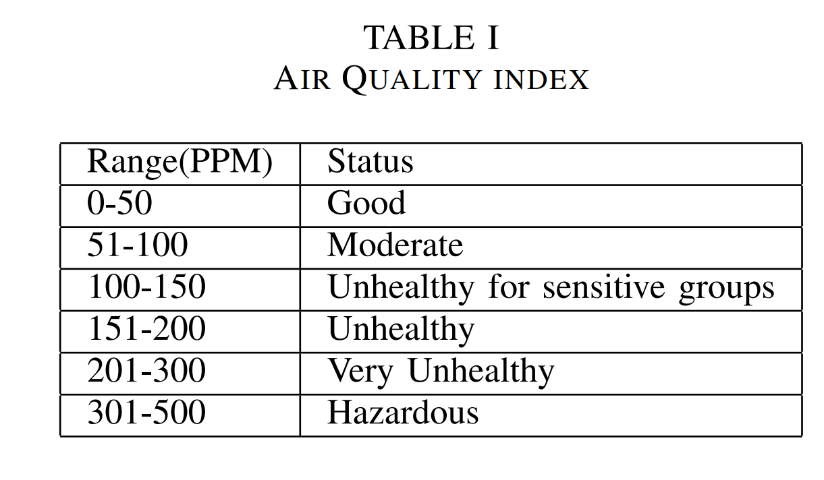
# INDEX

|  |  |  |
| --- | --- | --- |
| **S. No** | **Chapter** | **Page. No** |
| **1.** | **Introduction** | **5** |
| **2.** | **Project Description** | **6-10** |
| **3.** | **Details of Circuit** | **10-14** |
| **4.** | **Working and Operating Procedure** | **15-16** |
| **5.** | **Software tools and programming used** | **17-20** |
| **6.** | **Conclusion and Future Scope** | **21-22** |
| 7. | **Reference** | **23** |

**CHAPTER 1: INTRODUCTION**

Air is getting polluted because of the release of toxic gases by industries, vehicle emissions, and increased concentration of harmful gases and particulate matter in the atmosphere. The level of pollution is increasing rapidly due to factors like industries, urbanization, increase in population, and vehicle use which can affect human health. Particulate matter is one of the most important parameters having a significant contribution to the increase in air pollution. This creates a need for measurement and analysis of real-time Air Quality Monitoring so that appropriate decisions can be taken in a timely period. This paper presents a real-time standalone air quality monitoring. The Internet of Things is nowadays finding profound use in every sector and plays a key role in our SmogTrack: Real-Time Air Quality Monitoring too. The setup will show the air quality in PPM on the webpage so that we can monitor it very easily. In this IoT project, you can monitor the pollution level from anywhere using your computer or mobile. The setup will show the air quality in PPM on the webpage so that we can monitor it very easily. In this IoT project, you can monitor the pollution level from anywhere using your computer or mobile. The air condition is very polluted. In recent years, car emissions, chemicals from factories, smoke, and dust have been everywhere. That is the reason why now the air condition is much polluted. The effect of air pollution is very bad for our health, especially for places where the air in our body is taken for breathing. In our lungs may cause some diseases, such as asthma, cough, and lung disorders. The air pollution cannot be detected by human feelings. Air pollution may contain a lot of dangerous substances, such as LPG gas, carbon monoxide, and methane. Substances in the polluted air are very dangerous. For example, if the carbon monoxide is above 100 ppm, it makes humans feel dizzy and nauseous, and within minutes they could die.

As shown in Table 1 cited at, it explains the Air Quality Index ranges. Firstly, 0-50 PPM can be considered completely safe. 51-100 PPM can be considered as Moderate where this could be usually observed in traffic areas. 100- 150 PPM can be considered Unhealthy but only for sensitive groups.



**CHAPTER 2: PROJECT DESCRIPTION**

The Air Quality Monitoring System, an innovative amalgamation of Arduino and IoT technologies, revolutionizes the way we perceive and manage indoor air quality. This chapter delves into the intricate design, components, and capabilities of the Air Quality Monitoring System, showcasing its advanced features and the underlying technology driving its operation.

**2.1 System Architecture**

At its core, the Air Quality Monitoring System boasts a robust architecture comprising Arduino microcontrollers, a diverse array of sensors, actuators, and IoT modules. The central processing unit, an Arduino board, orchestrates the collection, analysis, and dissemination of air quality data. Interfaced with the Arduino board are an assortment of sensors including gas sensors, particulate matter sensors, and environmental sensors, tasked with continuously monitoring critical parameters influencing air quality.

Augmenting the sensor array, the Air Quality Monitoring System integrates actuators such as air purifiers, exhaust fans, and HVAC systems, enabling dynamic control based on real-time sensor data. This seamless interaction between sensors and actuators forms the bedrock of the system's functionality, facilitating swift adjustments to optimize indoor air quality.

The IoT component of the Air Quality Monitoring System facilitates seamless connectivity to the internet, enabling remote monitoring and control via a dedicated web or mobile application. Leveraging Wi-Fi or other wireless protocols, users gain instant access to comprehensive insights into indoor air quality metrics from anywhere, at any time.

**2.2 Key Features**

The Air Quality Monitoring System encompasses a host of advanced features aimed at improving indoor air quality and enhancing user convenience:

Real-time Monitoring: Users gain access to real-time data on pollutant levels, humidity, temperature, and other environmental parameters via an intuitive web or mobile application.

Remote Control: The system empowers users to remotely control connected devices such as air purifiers, fans, and ventilation systems, allowing for on-demand adjustments to optimize air quality.

Data Analytics: Robust data logging capabilities enable users to analyze historical trends and patterns, facilitating informed decisions for proactive air quality management.

Alert Mechanisms: The Air Guard System is equipped with alert mechanisms to notify users of abnormal conditions or hazardous pollutant levels, ensuring prompt intervention to safeguard occupants' health.

**2.3 Implementation Details**

Implementing the Air Quality Monitoring System entails meticulous assembly and programming of hardware components, coupled with the development of user-friendly software applications. Arduino sketches are tailored to facilitate sensor data acquisition, actuator control, and seamless communication protocols. Concurrently, web or mobile applications are meticulously crafted to provide users with an intuitive interface for data visualization and system control.

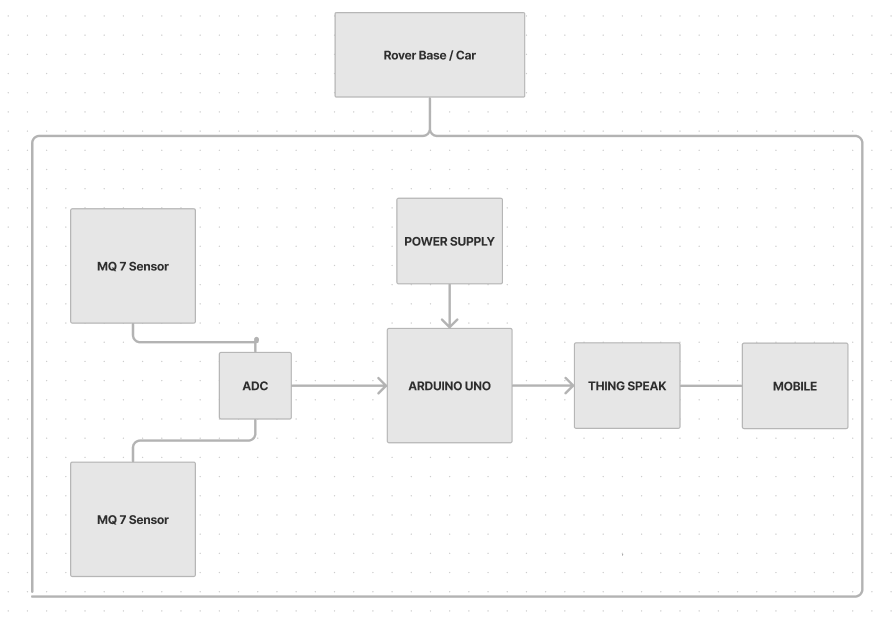
Adherence to best practices in power management, safety protocols, and scalability is imperative throughout the implementation phase to ensure the reliability and effectiveness of the Air Quality Monitoring System.

**2.4 Future Directions**

The Air Quality Monitoring System presents boundless opportunities for future enhancements and innovations. Prospective avenues for advancement include the integration of advanced sensor technologies for detecting specific pollutants, the incorporation of machine learning algorithms for predictive analytics, and the development of adaptive algorithms for autonomous system optimization.

In summary, the Air Quality Monitoring System epitomizes a paradigm shift in indoor air quality management, harnessing the potential of Arduino and IoT technologies to deliver a comprehensive solution for monitoring and enhancing indoor environments. This chapter has provided a comprehensive overview of the system's architecture, features, implementation intricacies, and prospects, setting the stage for a deeper exploration in subsequent chapters.

**BLOCK DIAGRAM**

****

**Block Diagram Description**:

MQ7 Sensor: This component encompasses a variety of gas sensors strategically placed within the indoor environment to detect and measure concentrations of various pollutants such as carbon monoxide (CO), nitrogen dioxide (NO2), volatile organic compounds (VOCs), and other harmful gases.

MQ135 Sensor: Positioned to monitor airborne particulate matter, including dust, smoke, pollen, and other microscopic particles, this sensor provides insights into air quality by measuring particle concentration and size distribution.

Arduino Microcontroller: Serving as the central processing unit, the Arduino microcontroller facilitates data acquisition from sensors, controls actuators based on predefined logic or user commands, and manages communication with the IoT module for remote monitoring and control.

ESP32 Microcontroller: The ESP32 module serves as a powerful microcontroller unit with integrated Wi-Fi and Bluetooth capabilities, providing a versatile platform for IoT applications. Equipped with a dual-core processor and a rich set of peripherals, the ESP32 offers high computational performance and flexibility for managing sensor data, communication protocols, and device control.

IoT Module: This module enables connectivity to the internet, allowing users to remotely monitor indoor air quality and control connected devices via a web or mobile application. Leveraging Wi-Fi or other wireless protocols, the IoT module facilitates seamless communication between the AirGuard System and user interfaces.

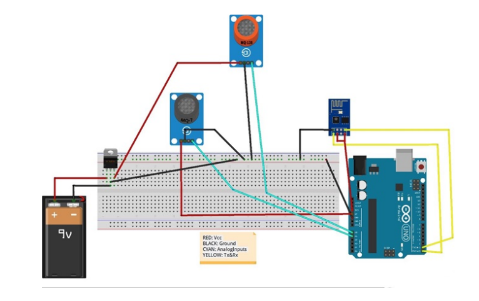
Rover or Car Base: The rover or car base serves as the mobile platform for deploying sensors and collecting data in various locations. Additionally, it features rugged construction and advanced navigation capabilities to traverse diverse terrain and reach remote areas, ensuring comprehensive coverage for air quality monitoring. By incorporating the rover or car base into the project, the system gains mobility and flexibility, allowing for dynamic data collection and analysis across different environments, and contributing to a more comprehensive understanding of air quality dynamics.

User Interface (Web/Mobile Application): The user interface provides a user-friendly platform for accessing real-time air quality data, receiving alerts, and controlling connected devices remotely. Through intuitive web or mobile applications, users can visualize indoor air quality metrics, set preferences, and take proactive measures to maintain a healthy indoor environment.

Power Supply: A stable and reliable power supply is essential for the uninterrupted operation of the AirGuard System. This component ensures adequate power distribution to all system elements, including sensors, actuators, microcontrollers, and IoT modules, while incorporating safeguards for overcurrent and overvoltage protection.

Data Logging and Analytics: Integrated data logging capabilities enable the storage of historical air quality data for analysis and trend identification. Advanced analytics algorithms may be employed to derive insights, predict trends, and optimize system performance for long-term air quality management.

**CHAPTER 3: DETAILS OF CIRCUIT**

****

1. **Microcontroller Unit (MCU):**
   * Arduino: A popular choice for prototyping and DIY projects due to its ease of use and extensive community support. Arduino boards come in various models with different capabilities, such as Arduino Uno, Nano, or Mega.
   * ESP32: Offers Wi-Fi and Bluetooth connectivity, making it suitable for IoT applications. It provides more processing power and memory compared to Arduino boards, allowing for more complex data processing and communication tasks.
2. **Air Quality Sensors:**
   * Particulate Matter (PM) Sensor: Measures the concentration of particles in the air, typically categorized into PM2.5 and PM10 sizes.
   * Gas Sensors: Detect gases such as carbon monoxide (CO), nitrogen dioxide (NO2), ozone (O3), and volatile organic compounds (VOCs). Common gas sensor modules include MQ series sensors for detecting specific gases.
3. **Data Analytics:**
   * CSV Module: Allows data logging to an SD card for storing sensor readings over time. Data can be saved in text or CSV format for later analysis.
4. **Display Interface:**
   * Things speak: Displays real-time sensor readings, air quality indices, and other relevant information in a user-friendly format.
5. **Communication Module:**
   * Wi-Fi Module: Enables wireless connectivity to local networks or the internet, allowing the system to transmit air quality data to remote servers, cloud platforms, or smartphone applications.
6. **Power Supply:**
   * DC Power Supply: Supplies stable DC voltage to the circuit components. It can be provided through batteries, AC adapters, or external power sources.
   * Voltage Regulator: Regulates the input voltage to ensure consistent power supply to the microcontroller and other components, protecting them from voltage fluctuations or overvoltage conditions.
7. **Rover or Car Base:**

* Enclosure: The circuitry is housed within a protective enclosure to shield it from environmental factors and ensure durability. The enclosure design allows for proper ventilation to prevent heat buildup and maintain optimal operating conditions for the components.
* Movement: it features rugged construction and advanced navigation capabilities to traverse diverse terrain and reach remote areas, ensuring comprehensive coverage for air quality monitoring. By incorporating the rover or car base into the project, the system gains mobility and flexibility, allowing for dynamic data collection and analysis across different environments, and contributing to a more comprehensive understanding of air quality dynamics.

1. **Calibration Mechanism:**
   * Calibration: Contains known concentrations of calibration gases used to calibrate gas sensors and verify their accuracy.
   * Calibration Values: Initiates the calibration process, allowing the system to compare sensor readings against the reference values and adjust sensor outputs accordingly.

**Hardware Used**

## Arduino Uno:



The Arduino Uno, featuring ATmega328P MCU, orchestrates sensor data acquisition and actuator control, pivotal for the air quality monitoring system. Its compatibility with Arduino IDE and diverse connectivity options make it an ideal choice for IoT applications.

## MQ 7 Sensor:

It is highly sensitive towards carbon monoxide so it is also known as a carbon monoxide sensor. It provides fast response time. It is having low cost. It is stable and long-life sensor. The working range of this gas sensor is 20 to 2000 PPM

## MQ 135 Sensor:

## The MQ135 gas sensor detects various air contaminants, including ammonia, benzene, and CO2, providing crucial data for air quality assessment. Its analog output interface makes it compatible with microcontrollers like Arduino for real-time monitoring applications.

## LM7805 Regulator:

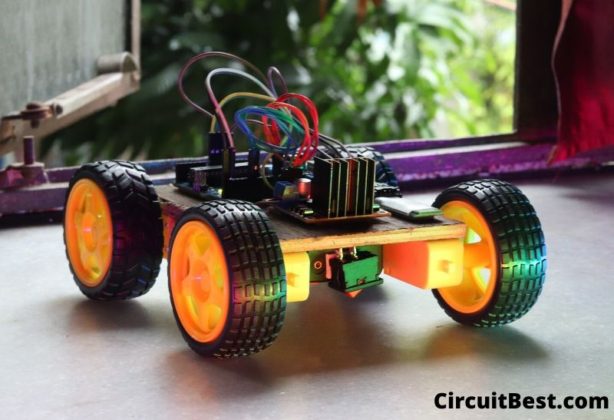
## The LM7805 is a voltage regulator IC that converts an input voltage (up to 35V) to a stable 5V output, making it ideal for powering low-voltage electronic circuits. With its built-in overcurrent and thermal shutdown protection, it ensures reliable performance in diverse applications, including powering microcontrollers, sensors, and other digital components.

## Esp 01 Wifi Module:

## The ESP-01 is a low-cost, compact Wi-Fi module based on the ESP8266 chipset, offering reliable wireless connectivity for IoT applications. It features a TCP/IP stack and supports Wi-Fi protocols, enabling seamless integration with Arduino and other microcontrollers. With its small form factor and ease of use, the ESP-01 is widely used for remote monitoring, home automation, and IoT projects.

## Rover or Car Base:

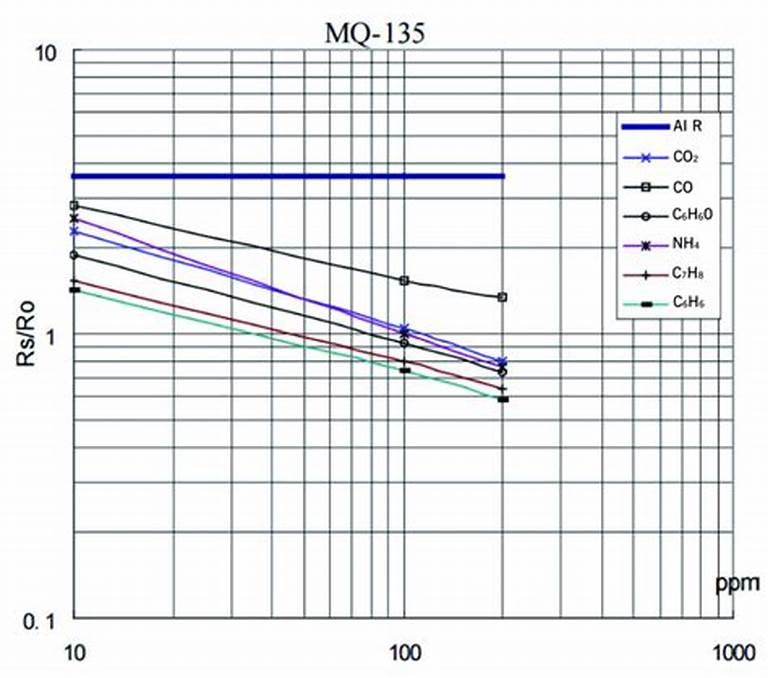
## The rover or car base forms the fundamental structure of the vehicle, providing stability and support for the entire system. It typically consists of a sturdy chassis designed to withstand rugged terrains, equipped with motor mounts and suspension systems to ensure smooth movement over uneven surfaces. The base configuration can vary, offering options for differential or skid-steer drive systems, adjustable wheelbases, and customizable layouts to accommodate diverse applications and payloads.



Additionally, the base serves as the platform for integrating essential components such as sensors, actuators, and control modules, facilitating seamless interaction between hardware components for efficient operation. Its versatility and adaptability make it a crucial component in building agile and reliable robotic vehicles for various tasks ranging from exploration and surveillance to logistics and research**.**

**CHAPTER 4: WORKING AND OPERATING PROCEDURE**

We have used an Arduino Uno Development kit that comes with an ATMega328P microcontroller. To provide Wi-Fi Support for it, we have used a cost-effective ESP-01 Wi-Fi module which helps us to connect to the Thing Speak Platform. Figure 1 represents the connections between the components used like Arduino Uno, MQ135, MQ7, ESP-01 Wi-fi Module, 9Volts Battery, and LM7805 Regulator. ESP-01 is connected to a 3.3Volts pin of Arduino Uno. MQ135 is connected to the 5Volts pin of Arduino Uno. As power won't be sufficient to drive one more sensor, MQ7 is connected to a 9Volts Battery via a 5Volts LM7805 Regulator. ESP-01 is connected to the Local Hotspot by giving the corresponding SSID and Password. The reason for using the LM7805 Regulator is that the 9Volts supply should not be directly given to the MQ7 sensor where it needs only 5Volts input at maximum, so the regulator does the job of stepping 9Volts to 5Volts. The most important step is to calibrate the sensor in fresh air and then draw an equation that converts the sensor output voltage value into our convenient units PPM (parts per million).



MQ135 Datasheet-Change in Resistance vs change in PPM

Using equations, we will be able to convert the sensor output values into PPM (Parts per Million). Now we developed the Code and flashed it into the Arduino Uno giving proper connections as mentioned.

After connecting the ESP-01 successfully to the hotspot, it gets established with the Thing speak website, and the account API Key is written in Arduino Code which helps to save the data only to our account bearing the given API key. Thing speak needs 15 seconds of refresh interval to push to the data. It shows the field charts of MQ135 and MQ7 sensor values for the location where the experiment is conducted in PPM (Parts per million). Also, it shows the visualization charts for corresponding sensors. The below figure shows the graphical analysis of the values collected with timestamping on the X axis and Air Quality PPM on the Y axis.



For analysis, we use Jupyter Notebook hosted on Google Co-Laboratory. A GPU was used to train the model and reduce computation time. The data was collected by a combination of four sensors kept in and around the University for a few days. Continuous data was collected before saving it to a storage device over the cloud. For any dataset, pre-processing is the most important step. The first step was to parse the timestamp generated from the device and make it into a format usable by the model. This was done using the following code. The second step was to remove outliers from the data and fill in missing values. This was done by replacing the missing value with a value from the same column which would ensure that the values did not get skewed completely. The third was to remove outliers. From data visualization, it was seen that there was a recurring value of -999 which could be the sensor output when the sensor was switched off and these redundant values were removed

In short, The air quality monitoring system operates by collecting data from sensors such as the MQ135 gas sensor, which detects various pollutants like carbon dioxide, ammonia, and benzene in the air. The LM7805 regulator ensures a stable power supply to the sensors and microcontroller. The Arduino Uno processes the sensor data and sends it to the ESP8266 Wi-Fi module for IoT connectivity. This enables remote monitoring and control via the internet, allowing users to access real-time air quality information through a dedicated application or web interface.

**CHAPTER 5: SOFTWARE TOOLS AND PROGRAMMING USED**

* Integrated Development Environment (IDE)

The Arduino Integrated Development Environment (IDE) serves as the primary software tool for programming the Arduino Uno microcontroller. It provides a user-friendly interface for writing, compiling, and uploading code to the Arduino board. Key features of the Arduino IDE include:

* Code Editor: A text editor with syntax highlighting and auto-completion features, making it easier to write and edit code.
* Compiler: A built-in compiler that translates the written code into machine-readable instructions for the Arduino board.
* Serial Monitor: A tool for debugging and monitoring serial communication between the Arduino board and connected devices.
* Library Manager: A library manager that allows users to easily add, remove, and update libraries for extended functionality.
* Thing Speak Platform

Thing Speak is a popular IoT platform that enables the collection, analysis, and visualization of sensor data in real-time. It provides cloud-based storage for data and offers various tools for data analysis and visualization, including graphs and charts. Thing Speak also supports integration with other platforms and services, making it a versatile solution for IoT projects.

* Program Code Implementation

#include <SoftwareSerial.h>

SoftwareSerial espSerial(2, 3); // RX, TX

String apiKey = "001A32KQE6LCMXA7";  //Change this key to your "Write API key"

int gas\_sensor = A0; //Sensor pin

float m = -0.3376; //Slope

float b = 0.7165; //Y-Intercept

float R0 = 2.82; //Sensor Resistance in fresh air from previous code

int CO\_sensor = A1; //Sensor pin

float m1 = -0.6527; //Slope

float b1 = 1.30; //Y-Intercept

float R01 = 7.22; //Sensor Resistance

void setup() {

  Serial.begin(9600);      // PC to Arduino Serial Monitor

  espSerial.begin(115200); // Arduino to ESP01 Communication

  pinMode(gas\_sensor, INPUT);

  pinMode(CO\_sensor, INPUT);

}

void loop() {

  // Gas sensor calculations

  float sensor\_volt; // Define variable for sensor voltage

  float RS\_gas; // Define variable for sensor resistance

  float ratio; // Define variable for ratio

  float sensorValue = analogRead(gas\_sensor); // Read analog values of gas sensor

  sensor\_volt = sensorValue \* (5.0 / 1023.0); // Convert analog values to voltage

  RS\_gas = ((5.0 \* 10.0) / sensor\_volt) - 10.0; // Get value of RS in a gas

  ratio = RS\_gas / R0;  // Get ratio RS\_gas/RS\_air

  double ppm\_log = (log10(ratio) - b) / m; // Get ppm value in linear scale according to the ratio value

  double ppm = pow(10, ppm\_log); // Convert ppm value to log scale

  // CO sensor calculations

  float sensor\_volt1; // Define variable for sensor voltage

  float RS\_gas1; // Define variable for sensor resistance

  float ratio1; // Define variable for ratio

  float sensorValue1 = analogRead(CO\_sensor); // Read analog values of CO sensor

  sensor\_volt1 = sensorValue1 \* (5.0 / 1023.0); // Convert analog values to voltage

  RS\_gas1 = ((5.0 \* 10.0) / sensor\_volt1) - 10.0; // Get value of RS in a gas

  ratio1 = RS\_gas1 / R01;  // Get ratio RS\_gas/RS\_air

  double ppm\_log1 = (log10(ratio1) - b1) / m1; // Get ppm value in linear scale according to the ratio value

  double ppm1 = pow(10, ppm\_log1); // Convert ppm value to log scale

  // ESP01 communication

  espSerial.println("AT+CIPMUX=0\r\n"); // To Set MUX = 0

  delay(2000); // Wait for 2 sec

  // TCP connection

  String cmd = "AT+CIPSTART=\"TCP\",\""; // TCP connection with https://thingspeak.com server

  cmd += "184.106.153.149"; // IP addr of api.thingspeak.com

  cmd += "\",80\r\n\r\n"; // Port No. = 80

  espSerial.println(cmd); // Display above Command on PC

  Serial.println(cmd); // Send above command to ESP01

  delay(1000);

  if (espSerial.find("ERROR")) { // If returns error in TCP connection

    Serial.println("AT+CIPSTART error"); // Display error msg to PC

    //return;

  }

  // Prepare GET string

  String getStr = "GET /update?api\_key=";

  getStr += apiKey;

  getStr +="&field1=";

  getStr += ppm;

  getStr +="&field2=";

  getStr += ppm1;

  getStr += "\r\n\r\n";

  Serial.println(getStr); // Display GET String on PC

  cmd = "AT+CIPSEND="; // Send data length

  cmd += String(getStr.length());

  cmd += "\r\n";

  Serial.println(cmd); // Display Data length on PC

  espSerial.println(cmd); // Send Data length command to ESP01

  if (espSerial.find(">")) { // If prompt opens //verify connection with cloud

    Serial.println("Pushed whole data TO CLOUD"); // Display confirmation msg to PC

    espSerial.print(getStr); // Send GET String to ESP01

  }

  else {

    espSerial.println("AT+CIPCLOSE\r\n"); // Send Close Connection command to ESP01

    Serial.println("AT+CIPCLOSE"); // Display Connection closed command on PC

  }

  // thingspeak free version needs 15-20 sec delay between every push

  delay(15000); // wait for 16sec

}

BRIEF EXPLANATION OF THE CODE

1. Include Libraries: The code includes the Software Serial library for communication with the ESP8266 module.
2. Define Variables: The code defines the API key for Thing Speak, pin assignments for gas and CO sensors, calibration parameters, and serial communication with the ESP8266 module.
3. Setup Function: In the setup function, the serial communication is initialized for both the Arduino and the ESP8266 module, and pin modes are set for the gas and CO sensors.
4. Loop Function:

* Gas Sensor Reading: Analog readings are taken from the gas sensor, and the voltage, resistance, and ppm (parts per million) values are calculated using calibration parameters.
* CO Sensor Reading: Analog readings are taken from the CO sensor, and ppm values are calculated similarly to the gas sensor.
* ESP8266 Communication: Commands are sent to the ESP8266 module to establish a TCP connection with the Thing Speak server.
* Thing Speak Data Transmission: The data, including gas ppm, CO ppm, and API key, is formatted into a GET request and sent to Thing Speak for updating the channel.
* Delay: A delay is added to comply with Thing Speak's free version requirements.

1. Explanation: The code first initializes communication and pin modes in the setup function. In the loop function, it reads sensor values, calculates ppm values, establishes a connection with Thing Speak using the ESP8266 module, sends data to Thing Speak, and then adds a delay before repeating the process.

This loop allows continuous monitoring and updating of air quality data on Thing Speak for remote access and analysis.

**CHAPTER 6: CONCLUSION AND FUTURE SCOPE**

From all the above information provided, we can calculate Air Quality in PPM. The problem with the MQ135 sensor is that specifically, it can’t tell the Carbon Monoxide or Carbon Dioxide level in the atmosphere, but the pros of MQ135 are that it can detect smoke, CO, CO2, and NH4 as mentioned. So, just to tell the individual gas level particularly, we have used a CO (Carbon Monoxide) MQ7 sensor. This project also corrects the PPM calculations mentioned. This project can be used both indoors as well as outdoors. For indoor, we can make this kit as a compact device such that if every home starts using the device, we can monitor the indoor air quality of a particular targeted area. Due to increasing air pollution, there is a necessity to keep an eye on Indoor air quality too. But for outdoor purposes, certainly, one sensor is not sufficient because one sensor has a sensitivity range of around 1 meter, so a network of sensors has to be deployed to monitor the outdoor air quality. Enough care is taken while calibrating the sensors. This project also targets the Machine Learning analysis on the dataset collected.

In conclusion, this Arduino-based air quality monitoring system offers a cost-effective and efficient solution for real-time air quality monitoring and data logging. By utilizing MQ135 and CO sensors, the system can accurately measure the concentration of various gases in the environment. The integration with Thing Speak enables remote monitoring and data visualization, allowing users to access air quality information from anywhere with an internet connection.

APPLICATIONS

* Indoor Air Quality Monitoring: It can be deployed in homes, offices, and industrial settings to monitor indoor air quality and ensure a healthy living or working environment.
* Environmental Monitoring: The system can be used for outdoor air quality monitoring in urban areas, construction sites, or industrial zones to assess pollution levels and implement necessary interventions.
* Healthcare: It can be employed in hospitals or healthcare facilities to monitor air quality in patient rooms, operating theaters, or critical care units to prevent the spread of airborne diseases and ensure patient safety.
* Research and Education: The system can serve as an educational tool for students and researchers studying air pollution, atmospheric science, or environmental engineering, enabling hands-on learning and experimentation.
* Smart Buildings: Integrated into smart building systems, the data collected by the air quality monitoring system can be used to automate ventilation, heating, and air conditioning systems for energy efficiency and occupant comfort.
* Urban Planning: Municipalities and city planners can use the data gathered by these systems for urban planning purposes, such as identifying pollution hotspots, implementing air quality improvement measures, and assessing the effectiveness of environmental policies.

FUTURE WORK

We can use one more sensor that tells the ozone layer status, but it costs very high. Also, we can use a PM2.5 laser dust sensor helpful exclusively for vehicle and factory emissions sensing. Thing Speak has a limitation in that it requires 15-20 seconds for every push of the values which is not reliable. We plan to use another IoT platform my devices Cayenne which is very fast in showing the values from the Arduino that helps us to collect more values in the dataset. Cayenne also comes with a ready Android/iOS application. But it doesn’t work with Arduino Uno rather works with only Node MCU or Raspberry Pi. If we use Node MCU, even the cost becomes less than the current setup. But the limitation in Node MCU is, it has only one analog input pin, so we will use ADS1115 I2C 16Bit ADC as an analog extender for Node MCU or a simple CD4051 8 to 1 Analog Multiplexer could easily overcome the problem of having only one analog input pin of NodeMCU(ESP8266). CD4051 Multiplexer is highly recommended as it is very cheap to purchase and easy to handle many sensors for NodeMCU. NodeMCU (ES8266) has inbuilt Wi-fi support (ESP-12E) and a microcontroller. We can link this to the Facebook API using IFTTT, Webhooks, and Adafruit platform collectively, such that users can request the air quality via the Facebook messenger chat application and get the output on the screen using a chatbot. Machine learning can also be implemented on the dataset such that we can predict the harmfulness of the air quality on people if the same bad air quality continues.

**REFERENCES**

[1] Tragos, E. Z., Angelakis, V., Fragkiadakis, A., Gundlegard, D., Nechifor, C. S., Oikonomou, G., ... & Gavras, A. (2014, March). Enabling reliable and secure IoT-based smart city applications. In 2014 IEEE International Conference on Pervasive Computing and Communication Workshops(PERCOM WORKSHOPS) (pp. 111-116). IEEE.

[2] Shah, J., & Mishra, B. (2016, January). IoT-enabled environmental monitoring system for smart cities. In 2016 International Conference on Internet of Things and Applications (IOTA) (pp. 383-388). IEEE.

[3] Pasha, S. (2016). ThingSpeak based sensing and monitoring system for IoT with Matlab Analysis. International Journal of New Technology and Research, 2(6).

[4] Khan, R., Khan, S. U., Zaheer, R., & Khan, S. (2012, December). Future internet: The Internet of things architecture, possible applications and key challenges. In 2012 10th International Conference on frontiers of Information Technology (pp. 257-260). IEEE.

[5] ”Malaya Ranjan, Rai Kumar, ”Understanding Parts per million in real time air quality index”, Journal of Mathematics and advanced sciences, pp. 23-29, September 2009.

[6] Kumar, N. S., Vuayalakshmi, B., Prarthana, R. J., & Shankar, A. (2016, November). IOT based smart garbage alert system using Arduino UNO. In 2016 IEEE Region 10 Conference (TENCON) (pp. 1028-1034). IEEE.

[7] Kumar, S., & Jasuja, A. (2017, May). Air quality monitoring system based on IoT using Raspberry Pi. In 2017 International Conference on Computing, Communication and Automation (ICCCA) (pp. 1341-1346). IEEE.

[8] Talari, S., Shafie-Khah, M., Siano, P., Loia, V., Tommasetti, A., & Catalao, J. (2017). A review of smart cities based on the internet of ˜things concept. Energies, 10(4), 421.

[9] Ma, Y., Yang, S., Huang, Z., Hou, Y., Cui, L., & Yang, D. (2014, December). Hierarchical air quality monitoring system design. In 2014 International Symposium on Integrated Circuits (ISIC) (pp. 284-287).

[10] Ahlgren, B., Hidell, M., & Ngai, E. C. H. (2016). Internet of things for smart cities: Interoperability and open data. IEEE Internet Computing, 20(6), 52-56.

[11] Cho, R. (2018, June 26). What you should know about air quality alerts. Retrieved from https://phys.org/news/2018-06-air-quality.html