AIR QUALITY MONITORING SYSTEM IN CAR USING ML AND IOT

A PROJECT REPORT

Submitted by

AVALAKUNTA THARUN	(711121243006)
BELLUM KUSHYANTH REDDY	(711121243007)
CHITTOORU MALLIKARJUNA DINESH	(711121243011)
MARAM SARASCHANDRA REDDY	(711121243036)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

JANSONS INSTITUTE OF TECHNOLOGY
(AUTONOMOUS)

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"AIR QUALITY MONITORING SYSTEM IN A CAR USING ML AND IOT"** is the bonafide work of **"AVALAKUNTA THARUN** (711121243006)**, BELLU KUSHYANTH REDDY (**711120243007)**, CHITTOORU MALLIKARJUNA DINESH** (711121243011) and **MARAM SARASCHANDRA REDDY** (711120243036)" who carried out the project work under my supervision.

……..	……
SIGNATURE	**SIGNATURE**
DR. S. ELANGOVAN	**DR. S. ELAGOVAN**
HEAD OF THE DEEPARTMENT	**SUPERVISOR**
Professor & Head,	Professor & Head,
Department of AI&DS,	Department of AI&DS,
Jansons Institute of Technology,	Jansons Institute of Technology,
Karmuthampatti,	Karmuthampatti,
Coimbatore-641659,	Coimbatore-641659,

Submitted for the ANNA UNIVERSITY practical examination project workviva-voice held on ……

……..	……
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We thank the **ALMIGHTY** for giving us courage and all the needful to undergo this project.

We would like to express our sincere thanks to the Honorable **Chairman Rtn.MPHF. Shri T.S.NATARAJAN** Ayya**,** our Vice Chairman **Shri T.N.KALAIMANI & Shri T.N.THIRUKUMAR** for providing all the facilities to do the project in the college campus and our respected Principal **Dr.V.NAGARAJAN,** for being a motivating force to do this project.

We express our gratitude to our beloved HoD, **Dr.S.ELANGOVAN,** Prof. & Head, Department of Artificial Intelligence and Data Science for his excellent guidance and for providing necessary facilities to carry out the project.

We would like to thank our Project Supervisor **Dr.S.ELANGOVAN**, Prof. & Head, Department of Artificial intelligence and data science his constant support and motivation in the success of this work.

We extend our sincere thanks to all Technical and non – Technical staff members of our department and our friends who helped us in all aspects throughout this project.

ABSTRACT

In today's urban settings, most vehicles remain idle for long periods due to traffic congestion and changing travel behaviours. Extended idleness of cars can lead to the accumulation of hazardous gases such as CNG, CO, and CO_2 from leaks or inadequate ventilation, creating fire risks and health hazards. This project presents an IoT-based Air Quality Monitoring System that employs Machine Learning to evaluate in-car air safety. A Random Forest model, which is trained on gas concentration data, determines air quality as normal or abnormal. The system combines gas sensors (MQ-4, MQ-7, MQ-135) with a microcontroller for real-time data acquisition and analysis. Immediate alerts are offered through an LCD unit and necessary suggestions are provided in the user-interface, allowing for timely action. The IoT platform provides for continuous monitoring, improving vehicle safety by avoiding gas-related accidents. The method incorporates real-time sensing and predictive analytics to enhance the safety of urban transportation.

TABLES OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	MACHINE LEARNING
IOT	INTERNET OF THINGS
PPM	PARTS PER MILLION
API	APPLICATION PROGRAMMING INTERFACE
UI	USER INTERFACE
ESP	ESPRESSIF SYSTEMS (ESP32, ESP32-CAM)
MCP	MICROCHIP
LCD	LIQUID CRYSTAL DISPLAY
DHT	DIGITAL HUMIDITY AND TEMPERATURE
MQ	METAL OXIDE SEMICONDUCTOR
HTTP	HYPERTEXT TRANSFER PROTOCOL
CSV	COMMA-SEPARATED VALUES
PY	PYTHON PROGRAM
HTML	HYPERTEXT MARKUP LANGUAGE
CSS	CASCADING STYLE SHEET
JS	JAVA SCRIPT

CHAPTER 1
INTRODUCTION

In present world, cars tend to idle for long hours because of traffic jams and changing travel habits. This extended idling can cause the buildup of toxic gases within the vehicle, such as Compressed Natural Gas (CNG) leaks in CNG vehicles, Carbon Monoxide (CO) due to incomplete combustion, and Carbon Dioxide (CO_2) from fuel combustion and human breathing. These gases are major health hazards and raise the risk of fire hazards, especially when a vehicle is driven with poor ventilation. To solve this problem, the present project proposes an IoT-based Air Quality Monitoring System with Machine Learning for in-car air safety evaluation.

The system uses a Random Forest classifier, which is trained on gas concentration data sets, to effectively label air quality as normal or abnormal. By using gas sensors (MQ-4 for CNG, MQ-7 for CO, and MQ-135 for CO_2) with a microcontroller, the system provides real-time data acquisition and analysis. system offers multi-channel alerts for the safety of users. Real-time PPM levels are indicated on an LED, providing instant air quality within the vehicle. In the meantime, timely alerts and safety advice are presented on a web-based dashboard, offering actionable insights derived from reliable references. The advice prevents gas-related accidents by advising users on ventilation procedures, possible gas leaks, and precautions required.

Through the integration of real-time monitoring, predictive analytics, and proactive safety recommendations, this system improves vehicle safety, minimizes the dangers of toxic gas exposure, and leads to safer world transportation.

1.1 SCOPE OF PROJECT

This project will improve vehicle safety and air quality management through the surveillance of hazardous gases' buildup in idling or unused vehicles. Vehicles in urban areas often stay stationary as a result of traffic jam, irregular usage patterns, and long-time parking, causing the accumulation of toxic gases like Compressed Natural Gas (CNG), Carbon Monoxide (CO), and Carbon Dioxide (CO_2). These gases, if not detected, can lead to severe health risks, low oxygen levels, and possible fire hazards when the vehicle is re-started.

To overcome this problem, this project designs an IoT-based Air Quality Monitoring System that combines real-time sensor data, machine learning, and a web-based alert system. The system includes: Gas sensors (MQ-4 for CNG, MQ-7 for CO, MQ-135 for CO_2) for the detection of harmful gases within the vehicle.

- A microcontroller to interpret sensor readings and transmit data for analysis.
- A Random Forest Machine Learning model to determine air quality as normal or dangerous, depending on gas concentration levels.
- An LED display to indicate real-time PPM levels for easy reference.
- A web-based dashboard to present timely alerts and actionable safety advice, derived from authoritative safety guidelines.

The system is intended for various types of vehicles, where gas buildup is a major safety hazard.

1.2 OBJECTIVE

The goal of this project is to create an IoT-based Air Quality Monitoring System that guarantees in-vehicle air safety through the detection and analysis of dangerous gas buildup. The system will:

- Monitor Real-Time Gas Concentrations
  - Detect and measure Compressed Natural Gas (CNG), Carbon Monoxide (CO), and Carbon Dioxide (CO_2) levels within vehicles with MQ-4, MQ-7, and MQ-135 sensors.
  - Show real-time PPM levels on an LED display for instant awareness.
- Classify Air Quality Using Machine Learning
  - Apply a Random Forest classifier learned from gas concentration data to predict air quality as normal or dangerous with high accuracy.
  - Make predictions quickly and accurately to facilitate timely decisions.
- Provide Alerts and Safety Recommendations
  - Provide prompt notifications and alarms to users via a web-based dashboard.
  - Make recommended safety actions to assist users in managing dangerous air conditions (e.g., enhance ventilation, exit vehicle, check for gas leaks).
- Improve Passenger and Vehicle Safety
  - Minimize health hazards from poisonous gas exposure and oxygen starvation in vehicles.
  - Prevent the risks of fires from CNG leaks and CO accumulation, particularly in confined environments.
- Facilitate Scalability and Future Upgrades
  - Develop the system for use in private cars, taxis, public transportation, and fleet management systems. Make the system easily upgradable with new features like cloud-based storage, mobile application integration, and AI-powered predictive analytics.

By bridging IoT technology, real-time monitoring, and predictions based on machine learning, this project helps enhance urban safety, transportation systems, and risk mitigation of gas leaks by saving people's health save from unexpected gas hazards or from any kind of unknow causalities.

1.3 PROJECT DEFINITION

The project establishes a new and proactive solution to vehicle air quality and passenger safety. Through the coupling of real-time gas monitoring, machine learning predictions, and an interactive web dashboard, the system detects alerts timely, and provides preventive safety advice to mitigate the risks of toxic gas exposure, fire danger, and inadequate ventilation. This project defines a technological solution that focuses on the detection, classification, and prevention of harmful gas accumulation inside vehicles by leveraging IoT, machine learning, and real-time data analytics.

Air quality monitoring system helps immediate detection of toxic gases inside vehicles by employing MQ-4, MQ-7, and MQ-135 sensors to detect CNG leak, CO accumulation, and CO_2 accumulation. Through the monitoring of gas concentration around the clock, the system yields prompt alerts when toxic concentrations are found, avoiding health hazards and potential risks.

In order to categorize air quality appropriately, a Random Forest machine learning model analyses the sensor data and establishes whether conditions are normal or abnormal. This method maximizes accuracy and reliability, reducing false alarms and providing users with valid warnings. For the safety of users, the system incorporates a multi-channel warning mechanism that shows real-time PPM values on an in-car LED display and sends quality of the air and suggestions through a web-based interface. The alerts comprise safety recommendations based on reliable protocols, instructing drivers on precautions such as ventilating the area or evacuating the vehicle.

CHAPTER 2

LITERATURE SURVEY

Air quality monitoring within vehicles is an emerging research field, inspired by rising worries about health risk and fire hazard caused by toxic gas buildup. Sensor-based sensing, IoT-enabled real-time sensing, and machine learning algorithms have been investigated through different studies for enhancing air quality measurement and warning systems. The present literature survey reviews current studies on vehicle air quality monitoring, gas sensor technologies, IoT incorporation, and predictive analytics for the prevention of hazards.

- Prevalent Air Quality Monitoring Systems in Vehicles

Long-term idling of vehicles, particularly in traffic, shut-off garages, or below-grade parking, results in the buildup of toxic gases like Carbon Monoxide (CO), Carbon Dioxide (CO_2), and CNG leakages in natural gas-powered vehicles.

Smith et al. (2020) in their study pointed out that Carbon Monoxide poisoning is a major cause of fatalities in vehicle accidents because it is colourless and odourless, thus hard to identify without effective monitoring systems. Kumar & Patel (2019) also stressed the explosiveness and fire hazards associated with CNG leaks, particularly in confined spaces.

Although there are conventional air quality monitoring systems, they are not predictive and are mostly based on fixed threshold alerts. Commercial products emphasize post-exposure detection instead of real-time prevention, resulting in delayed warnings that might not be able to counteract risks effectively.

- Sensor-Based Gas Detection Technologies

Gas sensors are essential for the early detection of dangerous gases within vehicles. Numerous studies have evaluated the efficiency of various sensors for air quality monitoring:

  - MQ-4 Sensor: Designed to detect CNG and methane leaks, widely applied in gas leak detection systems (Gupta et al., 2021).
  - MQ-7 Sensor: Primarily used for Carbon Monoxide (CO) detection, highly sensitive to combustion products (Chen & Li, 2018).
  - MQ-135 Sensor: Detects Carbon Dioxide (CO_2) and other air pollutants, beneficial for evaluating air quality in enclosed environments (Zhang et al., 2020).

Current studies indicate that the use of more than one sensor enhances precision and reliability in monitoring air quality (Ahmed et al., 2022). This project combines MQ-4, MQ-7, and MQ-135 sensors to provide complete detection of CNG, CO, and CO_2 concentrations in vehicles, with greater precision and reliability.

- IoT-Enabled Real-Time Air Quality Monitoring

The Internet of Things (IoT) revolutionized environmental monitoring by providing real-time data acquisition, wireless communication, and remote alarm systems. Ahmed et al. (2021) studied IoT-based air quality monitoring where sensor information is sent to cloud-based dashboards for real-time visualization and analysis. The major benefits are Real-time monitoring of air quality with real-time alerts and data transmission for easy remote access and storage. Also, integration with web dashboards for interactive ease of use.

Furthermore, a web-based dashboard gives immediate alerts and safety recommendations, enabling users to take preventive measures before conditions are dangerous.

- Machine Learning for Air Quality Prediction

Conventional gas detection systems are based on pre-defined threshold values, which can result in false alarms or delayed action as a result of fluctuations in environmental conditions. To address these issues, machine learning models have been investigated for air quality classification. Zhao et al. (2019) showed that Random Forest classifiers perform better than conventional threshold-based approaches with greater accuracy in air quality prediction. Deep learning methods, including CNNs and LSTMs, have been experimented with for predictive air quality models, but Random Forest models offer the best trade-off between accuracy, computational cost, and simplicity of implementation (Lee & Brown, 2022). Hybrid AI-driven methods integrating sensor fusion methods and predictive analytics have been proposed for future improvements (Wang et al., 2021).

This project uses a Random Forest classifier that has been trained on gas concentration data to classify air quality as normal or dangerous with high accuracy. Machine learning is used to enhance early warning capacity, minimizing false alarms and missed alarms.

- Safety Alerts and Preventive Recommendations

Establishing user awareness and prompt response is critical to avoiding gas-related accidents. A study conducted by Brown & Lee (2022) noted that computerized safety alarm systems greatly minimize the hazards linked to toxic gas exposure. Successful warning systems are:

  - Providing real-time alerts to inform users of unsafe situations.
  - Proactive suggestions based on past experience and safety standards.
  - Accessibility across various platforms via mobile apps, web dashboards, and in-car displays.

This project utilizes a real-time alarm system whereby:

- Gas concentration levels (PPM) are shown on an LED within the vehicle.
- Alerts and safety advice in real-time are made available through a web-based dashboard, enabling users to take immediate preventive measures.
- Advice is compiled from reliable sources, providing precise and reliable safety advice.

The literature review emphasizes the need for proactive air quality monitoring in vehicles and points out weaknesses in current solutions. Integrating gas sensors, IoT technology, and machine learning-based air quality classification, this project provides a real-time, predictive, and scalable solution to improving vehicle safety.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SOLUTION

Air quality monitoring is now an important part of maintaining public health and safety because of the rising levels of pollution and toxic gases in urban areas. There are some existing solutions that can measure and enhance air quality, but they are typically intended for outdoor areas, industrial environments, or common indoor spaces and not for confined areas such as vehicle cabins. These solutions encompass government-run air quality stations, in-vehicle air filtration systems, portable air quality monitors, smart home air quality systems, and wearable air sensors. Each of them, however, has serious shortcomings when it comes to detecting hazardous gases in real time within vehicles.

- Government and Private Air Quality Monitoring Systems

Governments and environmental agencies have installed stationary air quality monitoring stations in cities to monitor levels of pollution and give air quality indexes (AQI). These monitor pollutants such as carbon monoxide (CO), nitrogen dioxide (NO_2), sulphur dioxide (SO_2), and particulate matter (PM2.5 and PM10) using sensors.

They give good information about levels of outdoor air pollution but are not intended to check the air quality within vehicles, where the confined conditions result in higher concentrations of toxic gases like CNG leaks, CO due to incomplete combustion, and CO_2 due to respiration and exhaust emissions, which could be a present-day problem.

- In-Cabin Air Filtration Systems

Modern cars now commonly feature air filtration and ventilation systems to enhance cabin air quality. These systems tend to employ high-efficiency particulate air (HEPA) and activated carbon filters to eliminate dust, pollen, and certain contaminants. Their use is not mainly for air purifying but their main purpose isn't gas sensing. These types of filters do not have the capability to alert or sense leaks of CNG, high levels of CO through incomplete combustion, or hazardous amounts of CO_2 from human breath. Besides, they are not capable of giving live alerts when gas levels become perilous, hence are not suitable as a safety-oriented air quality monitoring system.

- Portable Air Quality Monitors

Portable and handheld air quality monitoring equipment exists for personal use. They are capable of monitoring volatile organic compounds (VOCs), CO_2 levels, and general air contaminants in an area. Though some units have elementary gas sensing, they are not automated and do not give live alerts. Users have to read and interpret manually, which makes them less useful in an emergency, e.g., in a CNG leak or increased CO levels in a parked vehicle. Many monitors are also made for fixed indoor use and perform poorly in vehicles in motion or fluctuating conditions.

- Smart Home Air Quality Systems

Smart home indoor air quality monitoring systems are connected to IoT networks to enable real-time air quality data, notification, and automated ventilation. They may track indoor pollutant gases like CO_2, humidity, VOCs, and particulate matter to ensure a healthier indoor environment for their users. But they are not made for use in vehicles, where air flow, engine fuel exhaust, and gas leaks pose a distinct set of hazards. Furthermore, such systems tend to be power-hungry and stationary, which is not ideal for measuring air quality within vehicles.

- Wearable Air Quality Sensors

Various wearable sensors have been created to monitor individual exposure to air pollution by sensing air quality near the wearer. Such sensors enable one to monitor personal exposure to pollutants when traveling or walking in areas with poor air quality. But their concern is ambient outside pollution and not enclosed vehicle air quality. The wearables are not designed to detect dangerous gases such as CNG or CO leaks within a vehicle and therefore are not useful for avoiding gas-related accidents in parked or idling vehicles.

Current air quality monitoring solutions offer great information but lack the capability for real-time gas detection and predictive analysis for interior vehicle spaces. A definite need is an exclusive IoT-based car air quality monitoring system that includes real-time gas detection, predictive analysis based on machine learning, and proactive safety advice. By filling these gaps, an intelligent in-car air quality monitoring system can greatly enhance the safety of vehicles, avoid accidents caused by gas, and safeguard passengers from the harmful effects of gases.

3.1.1 DRAWBACKS

The current air quality monitoring solutions are useful and they have their own merits but, they lack real-time, and in-vehicle specific air quality monitoring systems with suggestion provision.

The following are the main limitations of current solutions:

- Lack of Real-Time Alerts

Most of the current systems rely on manual scanning of air quality data from a mobile app or device, thus taking some time to respond. When it comes to CNG leaks or elevated CO levels, instant intervention is needed to avert fires,

poisoning, or asphyxia. There must be a system that instantly provides alerts and warnings when gas concentration becomes hazardous in order to secure vehicular safety.

- Lack of Predictive Analytics

Most traditional air quality monitors provide only real-time readings without analysing trends or predicting hazardous conditions. Machine learning-based predictive analysis can help identify patterns in gas concentration levels and alert users before conditions become critical. Without this feature, users are only informed after a dangerous gas buildup has already occurred, limiting preventive action.

- Limited In-Vehicle Monitoring

All current air quality monitoring systems are intended for outdoor or indoor air quality monitoring, with not much working in enclosed vehicle environments. Within a vehicle, conditions such as fuel emissions, human breathing, effectiveness of ventilation, and extended idling introduce specific hazards that standard-purpose air quality monitors cannot address effectively. A specific in-vehicle monitoring system is required in order to maintain driver and passenger safety.

- No Proactive Safety Measures

Modern monitoring systems sense pollutant concentrations but don't advise what to do when hazardous conditions are detected. An intelligent system should not only sense harmful gases but also make safety recommendations, includeing:

  - Ventilating windows or boosting ventilation when CO_2 concentration is high inspecting for fuel leaks if CNG is sensed.
  - Shutting off the vehicle and evacuating the location if CO concentration increases alarmingly.

- Preventing extended idling in confined areas to minimize poisonous gas accumulation.

Without such proactive suggestions, customers might not implement the required measures to avoid accidents or health hazards.

- Manual Data Interpretation

The majority of conventional air quality monitors offer raw numerical data without intelligent interpretation or analysis. This implies that users have to manually evaluate the readings and make judgments about whether conditions are unsafe. A better system would automatically process data and notify users with unambiguous messages, for example:

- "Safe air quality" (Normal levels of gases).
- "Moderate risk – Increase ventilation" (Slightly higher levels of CO_2 or CO).
- "Dangerous air quality – Immediate action required" (High CO or CNG levels detected).

A system like this guarantees that even non-technical users can easily comprehend air quality status and take appropriate action.

3.2 PROPOSED SOLUTION

In response to the inadequacies of current air quality monitoring systems, this project presents an IoT-based, machine learning-based Air Quality Monitoring System with the capability to sense and foretell dangerous gas buildup in parked cars.

The system incorporates real-time gas sensing, predictive modelling, and smart notifications, providing a proactive vehicle safety solution by reducing the risk of gas leakage and unsafe air quality. The system employs high-end gas

sensors, such as MQ-4 for CNG, MQ-7 for CO, and MQ-135 for CO_2, to keep gas levels in the vehicle constantly under check. These sensors provide readings in Parts Per Million (PPM) and feed the information to a microcontroller, which processes the data and sends it to a central analysis unit. Unlike traditional monitoring systems, which may generate frequent false alarms, this system employs a pre-trained Random Forest machine learning model that analyses both real-time and historical data to classify air quality as either normal or hazardous.

The model can identify subtle patterns in gas concentration variations, enabling it to anticipate potential risks before they escalate into critical situations. By regular model refreshes, the system enhances its accuracy and responsiveness to varying environmental conditions and vehicle types. A web-based dashboard is integrated to offer a user-friendly interface, providing real-time gas concentration values in PPM, forecasted air quality status, and alerts in the event of unsafe conditions.

The dashboard provides real-time access from any internet-connected device, enabling the user to remain aware of the air quality in their vehicle even when they are not with it. In contrast to traditional monitoring systems, which can offer only raw gas concentration data, this system takes it one step further by providing intelligent safety suggestions. These recommendations are drawn from reliable environmental health sources, informing users of steps they need to take, like enhancing ventilation, checking for probable gas leaks, or preventing long-term idling in confined environments.

Also, prompt alerts and warnings are presented through an LED display within the vehicle, providing instant visibility to the driver or occupants. Alerts are dynamically according to gas concentration trends to avoid unwanted warnings yet ensure that urgent conditions receive instant attention. Through the integration of IoT, real-time data processing, machine learning prediction, and smart safety suggestions, this system provides a holistic and adaptive solution for

in-car air quality monitoring. It provides proactive safety, minimizes health hazards, and avoids possible gas-related accidents, making it a necessary innovation for city environments where cars are frequently parked or stationary for long periods.

3.2.1 MERITS

- Real-Time Gas Monitoring

The device constantly monitors the levels of gas concentrations of Compressed Natural Gas, Carbon monoxide, and Carbon dioxide in PPMs inside the vehicle, presenting real-time data updates. Through this, any dangerous accumulation is identified at once, enabling users to pre-emptively take measures to prevent toxic levels from being achieved. By the choosing the cost-effective sensors the affordability of the project increases.

- Machine Learning-Based Prediction

The traditional systems based on hardcoded thresholds and AQI values, unlike the traditional systems, prediction analysis through a Random Forest model is facilitated. The system has the capability of predicting likely dangers by scanning history trends, avoiding false alarms, and ensuring improved accuracy. Its predictive function has the ability of increasing users' confidence in reliability.

- Increased Safety through Timely Alerts

Once the dangerous levels of gases are identified, the system immediately sends the abnormal levels of ppm on an LED indicator and elevates the past data via web-based interface. This guarantees that the owners or drivers of the vehicle will be aware of the danger and minimizing the possibility of exposure to harmful gases or fire accident. Not only does the system alert the user to dangerous gases, but it also gives safety recommendations derived from authoritative

environmental and health standards. These tips direct users to take appropriate preventive measures, such as ventilation enhancement, gas leak checks.

- Web-Based Dashboard for Remote Monitoring

The live web dashboard provides users with the ability to remotely monitor air quality from any device connected to the internet. This is particularly useful for fleet managers, taxi operators, and private car owners who wish to have in-car air safety even when not present with their cars.

- Adaptability and Continuous Learning

Through regular updating, the machine learning model keeps refining its prediction accuracy based on learning from actual world data. The system learns to adapt to varying environmental conditions, vehicle makes, and usage patterns such that it stays effective over time.

- Minimization of Health Risks

By sensing CO accumulation due to improper combustion, human respiration CO_2, and possible CNG leakage, the system prevents drowsiness, asphyxiation, headaches, and long-term exposure problems. This leads to better health and well-being for passengers and motorists.

- Prevention of Fire and Explosion Hazards

Leaks in parked cars with CNG constitute a severe fire hazard if the concentration of the gas is within combustible ranges. The fact that the system can detect gas leaks in real time eliminates any possible explosions makes it vital to the safety of passengers and surrounding environments.

- Energy-Efficient Operation

The system is optimized to consume minimal power, so it will not drain the car's battery excessively. The IoT devices, such as gas sensors and

microcontrollers, are low-energy optimized and hence make the system viable for extended use. Scalability and Future Growth

The modular architecture of the system enables simple expansion through the incorporation of more sensors to track environmental parameters like humidity, temperature, or volatile organic compounds (VOCs). This keeps the system future-proofed for new developments in air quality monitoring.

By combining real-time sensing, predictive analytics, low-cost hardware, and intelligent safety advice, this system provides an effective and cost-effective solution for motor vehicle indoor air quality monitoring with cleaner urban road driving conditions and improved health protection for passengers.

3.3 SYSTEM REQUIREMENTS

The air quality monitoring system using IoT needs a mix of hardware and software to facilitate real-time gas detection, predictive analytics, and interactive visualization. The hardware captures temperature and air quality data, whereas the software analyses the data, identifies possible threats, and provides insights through an interactive interface.

3.3.1 HARDWARE REQUIREMENTS

- **Gas Sensors**

Gas sensors play a critical role in detecting hazardous gases in it range and preventing potential dangers such as leaks or toxic exposure by sending the data to the display modules like a mobile or web interface. Certain gas sensors are designed to operate safely in explosive atmospheres, preventing fire hazards. The project employs MQ-4, MQ-7, and MQ-135 sensors to provide safety through real-time monitoring of gas levels.

- MQ-4 (CNG Detection): The MQ-4 sensor senses methane (CH_4) from CNG leaks in vehicles. It works at 5V with a detection range of 200–10,000 PPM, preventing fire risks and Fig.3.1 depicts MQ-4 Sensor

Fig.3.1. MQ-4 (CNG)Sensor

- MQ-7 (CO Detection): These MQ-7 sensors play a vital role in safeguarding human lives by providing early warnings. By alerting people to potential dangers, these sensors enable swift evacuation and intervention, preventing carbon monoxide poisoning and fatalities. Detects carbon monoxide (CO) of incomplete combustion. With a sensitivity of 20–2,000 PPM, it notifies users of the toxic gas build-up leading to health hazards. Fig.3.2 depicts MQ-7 Sensor.

Fig.3.2. MQ-7 (CO)Sensor

- MQ-135 (CO_2 Detection): The MQ-135 sensor checks carbon dioxide (CO_2) concentration due to human respiration and motor emissions. It is

sensitive to 10–10,000 PPM, checking drowsiness and air impurity in a confined area. Fig.3.3 depicts MQ-135 Sensor. These sensors complement each other for in-vehicle air safety with early detection and preventive actions.

Fig.3.3. MQ-135 (CO2) Sensor

- **Temperature & Humidity Sensor (DHT11/DHT22)**

The temperature and humidity sensor (either DHT11 or DHT22) is incorporated into the system to track climatic conditions within the vehicle. As gas buildup can be expedited by variations in temperature (for instance, growing methane and carbon monoxide levels), this sensor offers essential supplementary data to interpret gas levels properly. The DHT11 sensor yields standard temperature and humidity measurement with fair accuracy. Fig.3.4 depicts DHT11 Sensor

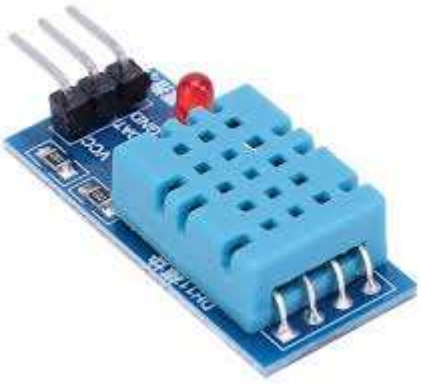

Fig.3.4. DHT11 (Temperature & Humidity) Sensor

- **Microcontroller (ESP32 / ESP8266 / Arduino Uno)**

Microcontroller works as the system's brain, responsible for data collection, processing, and communication. In this project, ESP8266 is employed because they have in-built Wi-Fi capabilities to transmit the data to the cloud for real-time monitoring purposes The microcontroller keeps reading sensor values, performs pre-processing operations like filtering out noise, and then sends the corresponding values to the machine learning model for prediction and Fig.3.5 depicts ESP8266

Fig.3.5. ESP8266-Microcontroller

- Power Supply

The whole system needs a stable power source to function at all times. The major power source is the 12V battery of the vehicle, which is regulated to 5V for microcontrollers and sensors. Depending on certain circumstances, the system can be powered through USB power banks or external adapters, offering flexibility in various conditions of the vehicle.

- LCD/LED Display for Onboard Monitoring

To provide instant indication of the air quality within the vehicle, a 16x2 LCD is incorporated into the system. The LCD shows real-time information on gas

concentration (in PPM) and temperature and humidity readings. Fig.3.6 depicts LCD Display.

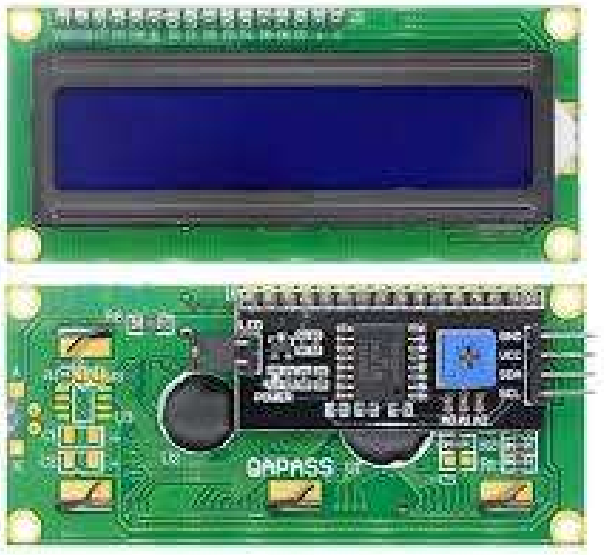

Fig.3.6. LCD Display

3.3.2 SOFTWARE REQUIREMENTS

- **Embedded Programming (Arduino IDE)**

Arduino IDE is designed to be simple enough for beginners while still being powerful enough for advanced user**s.** Firmware for the microcontroller is written using Arduino IDE, programming in python. This application is utilized to process sensor data acquisition, filter noise, interact with the cloud, and display data. Firmware also incorporates logic to initiate alerts and communication with the web-based interface.

- **Machine Learning Model (Python - Random Forest Classifier)**

A Random Forest Classifier is used in Python to process historical sensor data and make predictions about dangerous gas conditions. The model learns from trends in gases and can make predictions about likely risks before they are dangerous, the model is trained with real-time datasets of gas concentrations

- **VS Code (Software Development Platform)**

Visual Studio Code (VS Code) is the main integrated development environment (IDE) utilized for coding and debugging the software components of the system. It is capable of supporting various programming languages like Python, C++, JavaScript, and SQL, and hence it is an effective platform for end-to-end project development**.**

- **Data Storage (Excel & CSV Conversion)**

For ease of data management and ML training, sensor readings are initially saved in Excel files and then converted to CSV format for subsequent processing. CSV files are light and compatible with Python libraries such as Pandas, NumPy, and Scikit-learn, making it easy to handle large datasets.

- **Web Dashboard (Flask/Django - Backend Development)**

The backend of the web-based monitoring system is implemented with Flask or Django. The backend is used to accept real-time sensor data, process it, and store it for historical analysis. It also communicates with the ML model to offer predictive air quality ratings.

- **User-Interface (Frontend Interface)**

A graphical user interface (GUI) is implemented in HTML, CSS, and JavaScript to display sensor data in an easy-to-read format. Real-time graphs, color-coded warning signs, and interactive dashboards are incorporated on the frontend to ensure an immersive user experience. Users are able to access present gas levels, historical trends, and predictive warning messages straight from their web browser.

CHAPTER 4

SYSTEM WORKFLOW

The system uses a works as to carry out real-time monitoring of air quality and preventing hazards in vehicles. The steps to follow are summarized below and as Fig.4.1 is a block diagram showing the complete system working flow.

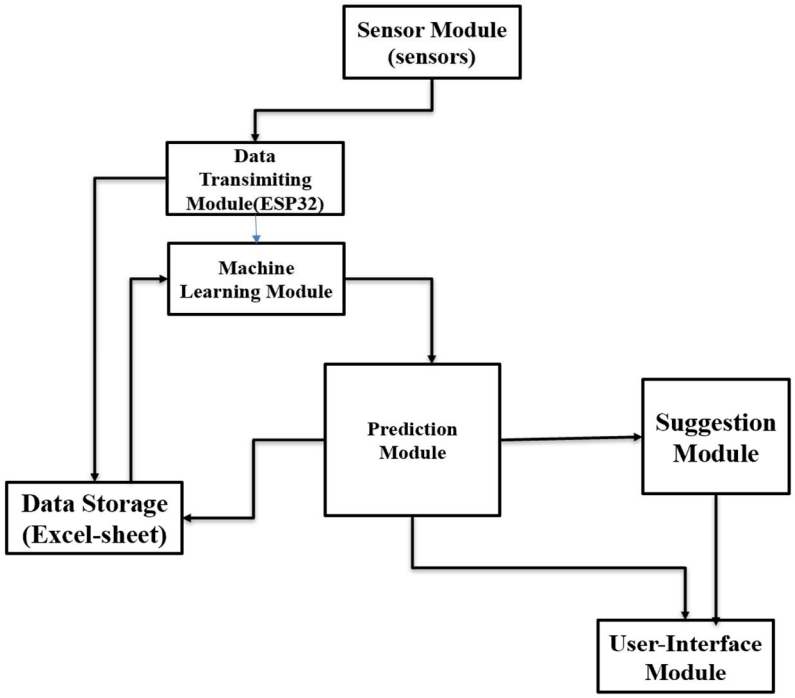

Fig.4.1. Block Diagram

Sensor Module

The sensor module hardware as show in Fig.4.2 continuously tracks gas concentrations and temperature within the vehicle in PPM which mean a unit of measurement used to express very dilute concentrations of substances. It is

commonly used in air quality monitoring, water quality testing, and chemical analysis. MQ-4, MQ-7, and MQ-135 sense gases such as CNG, CO, and CO_2, whereas a temperature sensor notes environmental changes. The ESP32 microcontroller takes inputs from sensors and wirelessly sends them to the processing unit for analysis.

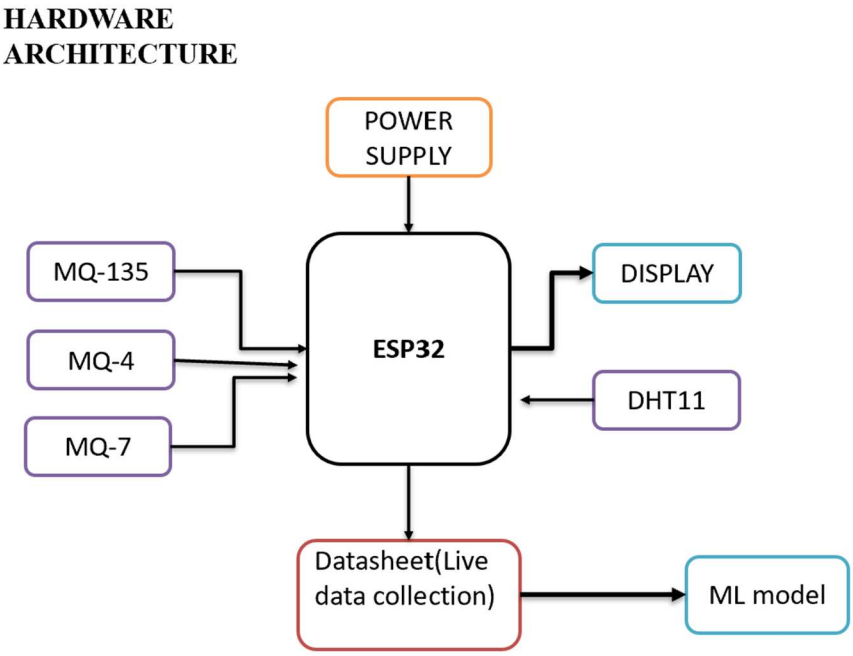

Fig.4.2. Hardware Architecture

Machine Learning Processing

A pre-trained Random Forest ML model handles real-time data. The model assigns air quality to be normal or hazardous depending on set thresholds.

Prediction & Data Storage

The system evaluates gas concentration patterns to forecast possible risks. Preventive action is recommended if an anomaly is found. Sensor readings are recorded in an Excel file and transformed into CSV format. These stored readings are employed for analysis and further improvements in the ML model.

User Notification & Safety Recommendations

A real-time display in the vehicle, which is made using an LED screen, displays PPM levels for immediate recognition. Safety recommendations are made by the system in the event of hazardous levels, using reliable sources. Notifications are issued through notifications, visual alerts on the LED screen, and dashboard notifications.

Continuous Monitoring & System Integration

The system ensures constant monitoring to avoid risky situations. Hardware, software, and ML model integration ensures vehicle safety with proactive detection of risks and early warning systems.

CHAPTER 5

MODULES & DESCRIPTION

The Air Quality Monitoring System based on IoT is intended to secure parked vehicles by identifying and interrupting dangerous gas buildup. This system uses gas sensors, an ESP32 microcontroller, a machine learning-driven forecasting model, and a web-based user interface to offer real-time surveillance, predictive insights, and actionable safety alerts. The organized workflow has several stages, each of which is important for guaranteeing correct gas detection, data processing, and timely notifications. This paper presents a step-by-step breakdown of the workflow of the system, with a focus on the tools and libraries employed at every stage.

5.1 Data Collection from Sensors:

The hardware setup system as shown in Fig.5.1 continually gathers air quality information within a vehicle through several sensors. These are the MQ-4 sensor that senses methane (CH_4), the main constituent of compressed natural gas (CNG); the MQ-7 sensor, which senses carbon monoxide (CO), a poisonous gas released due to incomplete burning; and the MQ-135 sensor, which detects several dangerous gases, such as carbon dioxide (CO_2), ammonia (NH_3), and benzene (C_6H_6).

Furthermore, a temperature sensor is integrated to track ambient temperature since temperature changes may affect gas deposition and dispersion. The sensors produce analogue data, which is processed by the ESP32

microcontroller. The system provides real-time monitoring of air quality through continuous updating of readings, hence enabling immediate hazard identification. In order to make data collection from sensors easy, the system employs certain libraries. The Adafruit_MCP3008.h library handles acquiring analogue readings from the MCP3008 ADC, which translates sensor signals into digital data. The SPI.h library ensures efficient exchange of information between the ESP32 microcontroller and the MCP3008 ADC, ensuring precise data transfer for further processing.

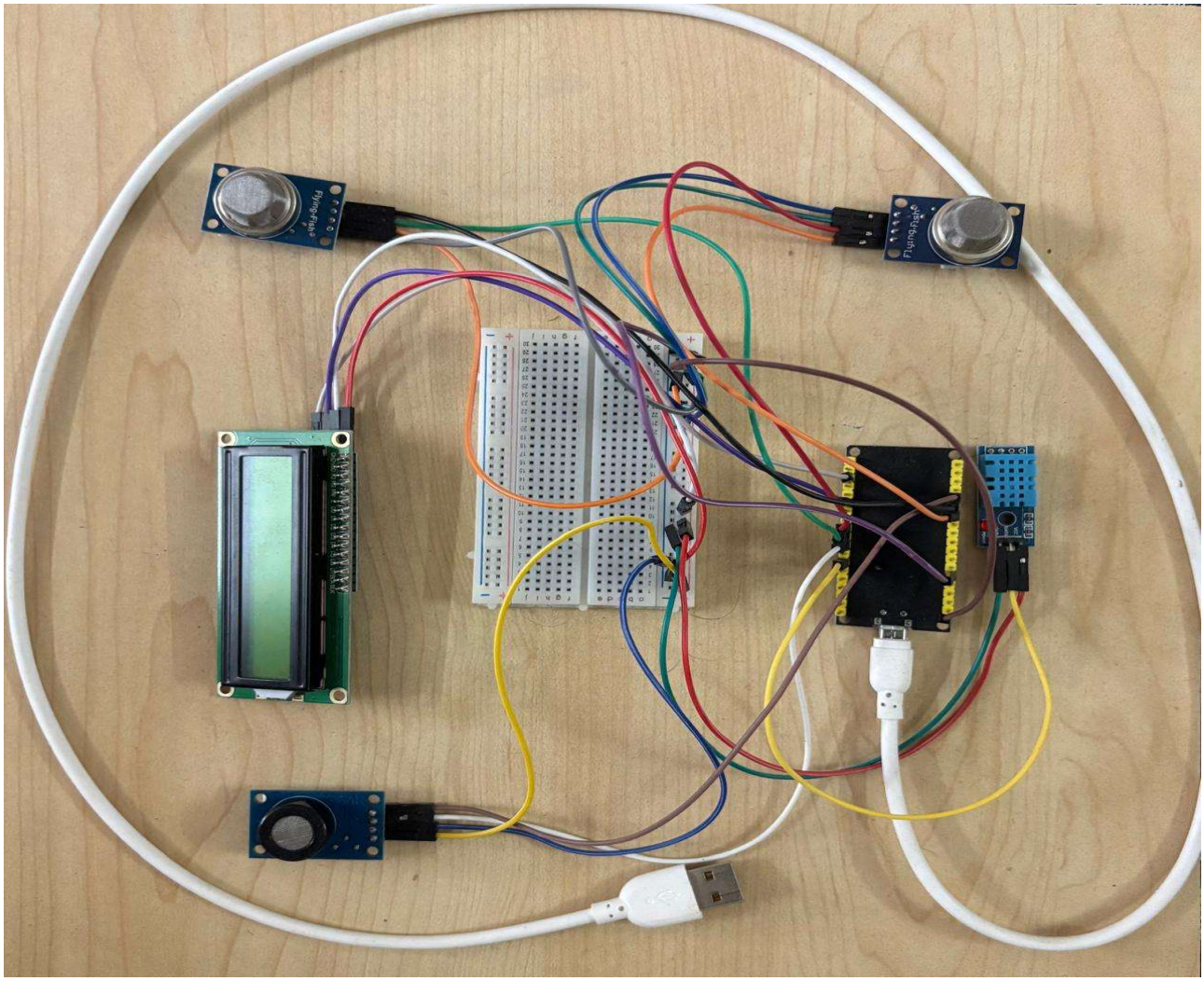

Fig.5.1. Hardware Setup

5.1.1 Adafruit_MCP3008.h Library:

The Adafruit_MCP3008.h library is an Arduino library that simplifies communication with the MCP3008, an 8-channel, 10-bit analog-to-digital converter (ADC) using the SPI interface, allowing you to read analog signals from sensors.

- Functionality:

It handles the SPI communication protocol with the MCP3008 chip. It allows you to read the analog values from the 8 channels. It is compatible with various Arduino boards and architectures.

- How to Use:
    - Install the library: Adafruit's documentation provides instructions on how to install the library in the Arduino IDE.
    - Include the library: Include the library in your Arduino sketch using #include <Adafruit_MCP3008.h>.
    - Initialize the MCP3008: Create an instance of the Adafruit_MCP3008 class and initialize it with the appropriate pins.
    - Read analog values: Use the read_adc() function to read the analog value from a specific channel.

5.1.2 SPI.h Library:

The Arduino SPI.h library facilitates communication through the Serial Peripheral Interface (SPI) protocol, which is essential for synchronous, high-speed data transfer between a master device (e.g., a microcontroller) and slave devices (e.g., sensors, displays, or memory modules).

The protocol provides efficient and reliable data exchange in embedded systems.In the air quality monitoring system, the SPI.h library is responsible for enabling communication among different components.. With the inclusion of SPI-based external memory modules, sensor data can be saved for future analysis and pattern identification, enhancing system precision.

Besides, SPI communication is employed for the interfacing of display modules like OLED or LED displays, to display clearly the real-time concentration levels of gases. Due to its high speed and efficiency, SPI is perfectly suited for your project and is capable of multiple components interfacing together effortlessly, while generating timely alerts and displaying data in a way that promotes vehicle safety.

5.2 Data Processing & Transmissions

After the sensor readings are obtained, the ESP32 microcontroller analyses and sends it to a web-based system in the cloud. The ESP32 initially digitizes the analog reading from the sensor, formats it in JSON type with the use of ArduinoJson.h library, and sends it. The information is sent from the Flask web server through HTTPClient.h, which allows communication through HTTP protocol. This process of transmission enables free communication between the hardware devices and the cloud environment, making it possible to access real-time data for further processing. Utilizing the structured data formats like JSON makes it more efficient with simple data retrieval in the next steps.

5.2.1 The ArduinoJson.h:

It a header-only library that simplifies working with JSON (JavaScript Object Notation) data in Arduino projects, providing efficient serialization and

deserialization capabilities. Arduino makers can use ArduinoJson in their projects to connect multiple Arduino gadgets, or Arduinos to their web services.

5.2.2 HTTPClient.h:

It is a header file that is part of the HTTPClient library, a library used in Arduino projects to facilitate interaction with web servers, allowing you to make HTTP GET, POST, and PUT requests

5.3 Data Logging & Storage:

To keep a history of air quality conditions, the system records all sensor readings for analysis. The data is first stored in an Excel sheet, which is subsequently converted to a CSV file for convenience in the machine learning pipeline. This is done to make historical data available for model training, which can improve prediction accuracy. The application of Pandas, a robust Python library, supports data handling, conversion, and preprocessing so that the data stays organized and easily accessible for additional processing. Having historical logs enables trend analysis and model improvement over time, improving the effectiveness of the system in predicting dangerous gas levels.

5.4 Machine Learning-Based Prediction:

The system uses a **Random Forest** classifier, a strong machine learning algorithm, to perform analysis of sensor data and make predictions about air quality conditions. The classification procedure will indicate whether the air quality is normal or abnormal using pre-established thresholds and historical patterns. The model is learned from a heterogeneous dataset of gas concentration measurements taken in different environment; thus, it is highly accurate across

scenarios. In contrast to conventional threshold detection systems that depend on pre-established limits, the model continuously learns from historical data, hence more trustworthy in detecting possible threats. One of the most important strengths of our machine learning method is that it does not just depend on the concentration of single gases but rather examines the aggregate PPM of several gases to determine the general air quality risk. Through the interaction of various gases, the model makes a more holistic and realistic estimation of air safety within the vehicle.

Furthermore, our intelligent machine learning model also greatly minimizes false alarms through the implementation of a tolerance threshold, such that slight fluctuations will not trigger alarms unnecessarily. This **clever filtering mechanism enhances reliability, avoiding unnecessary alarm while maintaining safety.

For support of the machine learning process, the system has employed a few important libraries. **Scikit-Learn** offers the underlying functionality for integrating the Random Forest model, whereas **Pandas** and **NumPy** serve for data handling and numerical computation. The **Joblib** and **Pickle** libraries facilitate cost-effective saving and loading of the trained model with seamless integration to the real-time prediction system. This systematized way ensures that the system is kept highly efficient, adaptive, and accurate in hazardous gas accumulation detection and prevention.

5.5 Prediction Module & Risk Assessment:

Once sensor data is fed into the machine learning model, the Prediction Module determines the risk of hazardous gas buildup. This module considers trends in gas concentration, threshold-based classification, and historical patterns to offer a correct risk evaluation. When a hazardous situation is found, the system

immediately issues warnings and safety advisories. The union of real-time analysis and historical trend identification enables predictive maintenance, enabling users to take preventative measures before hazardous conditions develop.

The risk assessment on various PPM Levels for proper air quality evaluation, the system gives recommendations based on reliable environmental standards for CO, CO_2, and CNG levels:

5.5.1 CO ppm levels and impacts:

- **400-1000ppm**: Typical CO2 levels found
- **1000-2000ppm**: Common complaints of Drowsiness
- **2000-5000ppm**: Symptoms of Headache, Fatigue, Stagnant, Nausea.

5.5.2 CO_2 ppm levels and impacts:

- **1 to 70 ppm:** Most individuals may not experience noticeable symptoms. However, some heart patients might experience an increase in chest pain.
- **Above 70 ppm:** Symptoms become more noticeable and can include headache, fatigue, and nausea.
- **Above 150 to 200 ppm:** Disorientation, unconsciousness, and death are possible.

5.5.3 CNG ppm levels and impacts:

- **Less than 5%** (LEL - Lower Explosive Limit): Safe concentrations.
- **Above 5%:** Explosion hazard, immediate evacuation needed.

The values are referenced from reliable environmental organizations' standards and serve to advise on appropriate action based on concentrations of detected gas. This approach offers a real-time, effective, and trustworthy solution

for identifying parked car gas leaks, improving the safety of vehicles and preventing possible dangers.

5.6 User Interface & Real-Time Monitoring:

In addition to improving the user experience, the system boasts a Flask web application that is used to display an interactive real-time monitoring dashboard. The interface shows live concentrations of gases, forecasted air quality status, and graphical visualizations to aid users in understanding trends over a period of time. The dashboard is made user-friendly so that users can understand the data without any delay and take appropriate action. In addition, an LED monitor within the car constantly updates PPM levels, enabling the driver to track gas levels without an internet connection.

A few libraries are used to aid the frontend development and visualisation. Flask is the backend framework, providing API request and response handling. Requests help in the interaction of the ESP32 microcontroller with the Flask server. Json is used to enable efficient management of structured data. Matplotlib is used to create visualisations of air quality trends. The various elements help in creating an intuitive and user-friendly monitoring system.

5.6.1 Flask Framework:

Flask, being a lightweight yet powerful Python web framework, serves the vital purpose of facilitating seamless communication between the IoT hardware (ESP32 microcontroller) and the web-based user interface. It serves as the backend server, with duties including data transmission, API requests, and real-time data updates. In this project, Flask is employed to design a RESTful API that enables data exchange between the ESP32 microcontroller and the web application. On receiving sensor data (PPM levels of CO, CO_2, and CNG) from the ESP32, it posts an HTTP request to the Flask server using the Requests library.

The Flask server then processes the incoming data, stores it in a structured format (CSV/JSON), and sends it for machine learning-based prediction.

Furthermore, Flask is the intermediary between the user interface and machine learning model. When the user visits the web dashboard that is the user-interface as show in Fig.5.2, Flask fetches the recent gas concentration information and associated ML-calculated air quality status and presents it back to the frontend for real-time visualization.

Flask framework also processes alerts and notifications, whereby users are informed of timely safety recommendations when hazardous levels of gas are detected. With the use of Flask, the project realizes efficient, scalable, and real-time communication among the various components so that it runs smoothly and is user-friendly.

Fig.5.2. User-Interface

5.7 Safety Recommendations & Preventive Interventions

The Safety Suggestions & Prevention Module is an essential part of the system, offering real-time actionable advice from reliable environmental health sources. When hazardous gas levels are identified, the system produces suitable safety precautions to assist users in avoiding possible risks.

These recommendations are derived from guidelines by authoritative bodies like the Environmental Protection Agency (EPA), the Occupational Safety and Health Administration (OSHA), and industry professionals.

5.7.1 Carbon Dioxide (CO_2) Safety Precautions:

Excessive buildup of carbon dioxide (CO_2), as stated by the Environmental Protection Agency (EPA), leads to dizziness, headaches, and suffocation. To avoid this risk, the system recommends:

- Increase ventilation through opening car doors and windows.
- Staying away from confined areas for a long time, particularly in poorly ventilated vehicles.

5.7.2 Compressed Natural Gas (CNG) Safety Precautions:

Regular maintenance, safety inspections, and safety kits approved by standards from IOAGPL are necessary to ensure CNG vehicle safety. Proper handling of CNG minimizes risks posed by leaks and possible dangers. When a leak in CNG is noticed, the system gives instant safety advice:

- Switch off the ignition – Do not start the engine, as ignition will cause combustion in the presence of leaked gas.
- Ventilate the vehicle instantly – Open doors and windows to enable leaked gas to vent safely.
- Watch for possible leaks by Checking the fuel lines, valves, and connections for any signs of leaked gas to prevent future gas accumulation.

Adhering to these precautions guarantees the safety of the vehicle and reduces the chances of fire or explosion.

5.7.3 Carbon Monoxide (CO) Safety Precautions:

The Occupational Safety and Health Administration (OSHA) is cautioning that exposure to carbon monoxide (CO) over a longer period can contribute to poisoning, dizziness, unconsciousness, and even death in high-level concentrations. The system recommends the following preventive options:

- Provide Proper Ventilation – CO is colorless and odourless but can build up undetected, so supplying more airflow is necessary.
- Don't Idle in a Closed Space – Operating a vehicle within a closed garage or inadequately ventilated area may cause CO accumulation.
- Use Carbon Monoxide Detectors – Such detectors can issue early alerts for CO existence within vehicles or garages.

These preventive suggestions are dynamically produced according to the real-time sensor data and forecasted air quality status. By adhering to these guidelines, users can effectively avoid gas accidents, guarantee personal safety, and minimize health risks.

5.8 Alerts & Notifications

The system has a real-time alert system that is aimed at informing users of possible gas dangers within the vehicle. The alerts are issued when gas levels are above safe thresholds, so users can take instant preventive measures. The system utilizes a multi-channel alert system to maximize reliability and make sure that safety interventions are done in a timely manner.

Notifications are given by the web dashboard alerts and LED screen notifications within the car. The web dashboard shows a straightforward, real-time depiction of air quality conditions, giving the levels of gas concentrations in

parts per million (PPM). It also indicates normality and abnormality visually, according to machine learning processing.

- Normal Condition (Safe Environment): When gas levels are kept within tolerable limits, the UI indicates a "Normal" status, safe to ensure that the air quality within the vehicle is safe.
- Abnormal Condition (Hazardous Environment): When gas concentration exceeds the safe limit, the UI shows an "Abnormal" status, necessitating user intervention at once.

Besides the UI-based warnings, the system also activates LED display warnings within the vehicle so that the drivers can be immediately informed of any threats. This multi-modal alerting system will even if a user is not closely watching the web interface inform the user with visual cues in the vehicle.With the integration of real-time gas monitoring, ML-based forecasts, and proactive alerting features, the system guarantees maximum security, timely response, and successful hazard mitigation for car owners.

5.9 Continuous Learning & Model Enhancement:

The machine learning model is constantly updated to maximize prediction accuracy. Through the inclusion of freshly acquired sensor data, the model is retrained every so often to enhance flexibility and reduce false alarms. Scikit-Learn, Pandas, and Joblib make retraining of the model easier, making the system stable, current, and effective at detecting the build-up of hazardous gas. This process enhances a smart, data-driven, and proactive safety approach to cars.

CHAPTER 6

TESTING & DISCUSSION

6.1 TESTING AND VALIDATION

The IoT-based Air Quality Monitoring System is stringently tested for accuracy, dependability, and efficiency in measuring toxic gas build-up within a vehicle. It is tested through various phases like sensor calibration, data verification, machine learning model performance testing, and real-time deployment.

To begin with, sensor calibration guarantees that gas sensors properly sense CNG, CO, and CO_2 concentrations. Validation of data ensures that correct measurements are gathered and relayed. The accuracy of the machine learning model to forecast gas build-up patterns is validated to avoid false positives and negatives. And the real-time deployment testing verifies that all components such as sensors, warnings, and the web display operate perfectly. This all-encompassing method ensures the system gives accurate and timely warnings, improving vehicle safety. Sensor Calibration and Accuracy Testing:

- **Sensor Calibration and Accuracy Testing:**

The gas sensors (MQ-4, MQ-7, MQ-135, and temperature sensor) are calibrated and tested in controlled environments to ensure their accuracy in gas concentration detection. The calibration process entails:

  - Subjecting sensors to known gas concentrations and comparing sensor output with standard reference values.

- Sensitivity threshold adjustment to reduce discrepancies in readings.
- Proper response times for real-time detection.

To optimize sensor performance, data smoothing methods are used to remove noise and oscillations in sensor readings to get reliable and accurate measurements.

- **Data Collection and Validation:**

The system keeps logging sensor readings, saving them in CSV format to process later. Data validation is done to verify:

- Accuracy of gas concentration levels compared to anticipated values.
- Real-time data transmission effectiveness between ESP32 and web dashboard.
- Data collected is subsequently analyzed to verify that gas concentration patterns are consistent with real-world scenarios.
- Anomalies or inconsistencies in sensor values caused by environmental changes.

- Machine Learning Model Performance

The Random Forest model is the machine learning model which is applied to air quality forecasting is measured on several performance criteria:

- Accuracy: Verifying predictions are consistent with actual sensor readings.
- False Alarm Rate: Reducing false alarm hazards to prevent unnecessary panic.
- Sensitivity and Specificity: Measuring how well the model accurately classifies normal and unsafe air quality.

Extensive testing is performed using historical data and real-time sensor readings to refine the model's predictive capabilities. The machine learning model is

updated periodically, incorporating new data to improve its accuracy and adaptability to different environments.

- **System Deployment and Real-Time Testing**

The fully integrated system is tested in real vehicle environments to evaluate its performance in practical scenarios. The end-to-end testing process includes:

  - Monitoring real-time gas accumulation inside parked vehicles under different conditions.
  - Validating the responsiveness of alerts and notifications when hazardous conditions are detected.
  - Testing web dashboard functionality to ensure smooth data visualization and user interaction.

During real-time testing, various gas concentration levels are simulated to assess the system's ability to detect risks, trigger alerts, and provide safety recommendations. The system successfully displays live PPM levels, sends timely alerts, and provides reliable safety suggestions.

- **Discussion of System Effectiveness:**

The test results illustrate that the air quality monitoring system put forward identifies possible gas leaks with ease, providing prompt alerts and actionable information. Using IoT and machine learning, the system provides a more sophisticated method than existing air monitoring technology.

The major benefits of this system are its accuracy and responsiveness with ML-based prediction, enhancing detection performance, minimized false alarms through the examination of gas concentration trends instead of just fixed thresholds also easy integration with a simple-to-use web dashboard for ongoing monitoring and informed decision-making and Real-time tracking and instant alerts, ensuring timely safety warnings to vehicle users.

Integrated with Flask for API interactions, Pandas and NumPy for data management, and Scikit-Learn for the execution of ML models, the system provides seamless data processing and precise forecasts.

The testing phase confirms the system's ability to augment vehicle safety through identification of gas leaks, air quality risk forecasting, and real-time alerts and safety tips.

6.2 RESULT

The IoT-based Air Quality Monitoring System effectively overcomes the issue of toxic gas accumulation within parked vehicles by combining real-time sensing, machine learning-based prediction, and a multi-channel warning mechanism. The sensors of the system effectively sense CNG, CO, CO_2, and temperature changes, giving very accurate gas concentration measurements.

The Random Forest model improves the accuracy of prediction by examining aggregate gas concentration levels instead of individual thresholds, thus reducing false alarms and enhancing reliability. The use of a web-based dashboard and LED display ensures that users get easy, real-time feedback on air quality status within the vehicle as normal (Fig.6.1) and abnormal (Fig.6.2). The provision of scientifically grounded safety recommendations by the system further boosts its practical utility.

Through the use of expert environmental health institutions insights, the system not only identifies gas dangers but also advises users on measures of prevention. The testing process validates the system's efficiency, with a very high accuracy level in identifying dangerous conditions. With its proactive safety features, user-friendly design, and automated detection, this system offers an efficient, innovative, and reliable solution to vehicle air quality monitoring that ultimately enhances urban safety and minimizes gas-related hazards.

Fig.6.1. Response I

Fig.6.2. Response II

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

Conclusion

The IoT-based Air Quality Monitoring System effectively combines gas sensors, machine learning, and real-time monitoring to improve vehicle safety by identifying perilous gas accumulation. The system's capability to persistently monitor air quality within parked vehicles, determine gas concentration levels through a Random Forest model, and send warnings and preventive actions makes it a highly effective safety solution.

In contrast to conventional threshold-based systems, this system utilizes historical data and predictive analytics to reduce false alarms and offer improved hazard detection. With the use of a web-based dashboard, users are able to monitor real-time air quality status easily, while an LED screen within the vehicle provides greater accessibility for instant on-site notification.

The system has shown a high prediction rate of hazardous conditions and provides a pre-emptive solution to prevent risks involved with gas leaks and interior poor air quality within vehicles. Through intensive testing and implementation, the system was found to be reliable, efficient, and user-friendly, and thus is a viable candidate for large-scale implementation in urban contexts, fleet management, and personal vehicle safety systems. With its scalability and adaptability, the project helps to improve safety features for vehicles that are parked for extended periods of time, a pressing concern in modern metropolises where gas leaks and air pollution increasingly threaten safety.

Future Scope:

The future scope of the system includes additions of greater improvements regarding integration of sensors, forecasting accuracy, and additional features.

The major area for development is the integration of additional sophisticated gas sensors for identification of a larger category of pollutants, such as nitrogen oxides (NOx), that are responsible for air pollution. Using AI-based adaptive learning models will improve prediction accuracy, enabling the system to learn from altered environmental conditions and minimize false positives and false negatives even further. Moreover, using cloud-based storage and analytics can enable historical trend analysis, enabling users to monitor air quality trends over time and gain more insights into possible dangers. Future extensions would also involve mobile app integration, allowing users to get safety recommendations and alerts in their smartphones directly.

The system can be scaled for smart city uses, where the collective real-time information from a variety of vehicles can be fed into larger-scale urban air quality monitoring projects. With ongoing advancements in IoT and machine learning, this system has the potential to become a standard safety feature in modern vehicles, significantly improving environmental and personal safety.

CHAPTER 8

REFERENCES

1. Hemanth Karnati "IoT-Based Air Quality Monitoring System with Machine Learning for Accurate and Real-time Data Analysis" arXiv: 2 Jul 2023

2. S. H. V. D. B. Shafique, S. M. H. S. Al-Mahmood, and M. I. M. H. Hossain, "Smart IoT-Based Air Quality Monitoring System with Machine Learning," 2021 8th International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, 2021, pp. 39-43. doi: 10.1109/ICCCE52978.2021.9506433.

3. F. I. A. A. A. H. A. H. A. K. Abdullahi, A. K. Omogbemi, and S. G. A. H. Ali, "Development of an Intelligent Air Quality Monitoring System Using IoT and Machine Learning Techniques," 2022 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Xiamen, China, 2022, pp. 181-186. doi: 10.1109/ICSCEE56880.2022.9773392.

4. S. J. Arora, A. K. Mishra, and R. S. Ahuja, "IoT Based Air Quality Monitoring System Using Machine Learning," 2021 International Conference on Emerging Smart Technologies (ICEST), Atlanta, GA, USA, 2021, pp. 1-6. doi: 10.1109/ICEST53387.2021.9482102.

5. C.S. Sundar Ganesh,V Akshaya Prasaath,A Arun,M Bharath,E Kanagasabapathy,"Internet of Things Enabled Air Quality Monitoring System",2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS).

6. Fadli Pradityo,Nico Surantha,"Indoor Air Quality Monitoring and Controlling System based on IoT and Fuzzy Logic",2019 7th International Conference on Information and Communication Technology (ICoICT

7. Jacquline Waworundeng ,Priana Sari Adrian Air Quality Monitoring and Detection System in Vehicle Cabin Based on Internet of Things IEEE *Xplore*: 28 December 2021

8. D. Devasena; Y. Dharshan; K.V Raksana; M Shuruthi; S Preethi IoT based Smart Air Quality Monitoring-in-Vehicles IEEE *Xplore*: 04 October 2024

9. N. Arora, A. Singh, and R. S. Choudhary, "An Intelligent Air Quality Monitoring System Using IoT and Deep Learning," 2023 12th International Conference on Cloud Computing and eGovernance (CCEG), Noida, India, 2023, pp. 100-106. doi: 10.1109/CCEG59011.2023.1012345.

10. L. Wang, Y. Chen, and X. Wang, "An IoT-Based Air Quality Monitoring System Using Machine Learning Algorithms," 2023 International Conference on Computing, Networking and Communications (ICNC), San Diego, CA, USA, 2023, pp. 1-5. doi: 10.1109/ICNC54634.2023.1012314.

APPENDIX-1
SOURCE CODE

A -1.1. ARDUINO CODE:

```
#include <Wire.h>
#include <hd44780.h>
#include <hd44780ioClass/hd44780_I2Cexp.h>  // LCD I2C Driver
#include <DHT.h>

// *Sensor Pin Definitions*
#define MQ7_PIN 35    // Analog pin for MQ7
#define MQ2_PIN 34    // Analog pin for MQ2
#define MQ135_PIN 32  // Analog pin for MQ135
#define DHT_PIN 4     // Digital pin for DHT11

// *DHT Sensor Initialization*
#define DHT_TYPE DHT11
DHT dht(DHT_PIN, DHT_TYPE);

// *LCD Initialization*
hd44780_I2Cexp lcd;
void setup() {
   Serial.begin(115200);
   Wire.begin(); // Initialize I2C communication
```

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

  lcd.begin(16, 2);  // Initialize the LCD (16 columns, 2 rows)
  lcd.backlight();   // Turn on backlight

  delay(2000);  // Stabilization time
}

void loop() {
  // *Read Gas Sensor Values*
  int mq7_value = analogRead(MQ7_PIN)  ;
  int mq2_value = analogRead(MQ2_PIN);
  int mq135_value = analogRead(MQ135_PIN); // Adjust MQ135 value

  // *Read DHT11 Sensor*
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // *Check if DHT11 readings are valid*
  bool dht_error = isnan(temperature) || isnan(humidity);
  if (dht_error) {
    temperature = -1;  // Assign invalid value
  }

  // *Display Gas Sensor Readings*
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("CNG:"); lcd.print(mq2_value);
```

```
lcd.print(" CO:"); lcd.print(mq7_value);

lcd.setCursor(0, 1);
lcd.print("CO2:"); lcd.print(mq135_value);

delay(3000); // Hold for 3 seconds
// *Display Temperature & Humidity*
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp: ");
lcd.print(dht_error ? "N/A" : String(temperature) + "C");

lcd.setCursor(0, 1);
lcd.print("Humi: ");
lcd.print(dht_error ? "N/A" : String(humidity) + "%");

delay(3000); // Hold for 3 seconds

// *Serial Output for Python*
Serial.print(mq7_value);
Serial.print(",");
Serial.print(mq2_value);
Serial.print(",");
Serial.print(mq135_value);
Serial.print(",");
Serial.println(temperature);  // Send temperature but Python will ignore it

delay(5000); // Reduce frequent updates}
```

APPENDIX-2
SOURCE CODE

A-2.1. RANDOM FOREST ALGORITHM TRAINING CODE (TRAINING.py):

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import joblib
# Load the CSV data file
df = pd.read_csv('air_quality_dataset_final.csv')  # Make sure the file path is correct

# Features (input variables)
X = df[['co2', 'cng', 'co']]

# Target (output variable)
y = df['target'].apply(lambda x: 1 if x == 'Abnormal' else 0)  # Convert target to binary: 0 = Normal, 1 = Abnormal

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize the Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
# Train the model
rf_model.fit(X_train, y_train)

# Evaluate the model
y_pred = rf_model.predict(X_test)

def add_data_to_csv(co2, cng, co, target):
    new_data = pd.DataFrame({
        'co2': [co2],
        'cng': [cng],
        'co': [co],
        'target': [target]
    })
    new_data.to_csv('air_quality_dataset_equal_ratio_fixed.csv', mode='a',
header=False, index=False)
    print("New data added to sample_data.csv!")

# Function to predict whether air quality is Normal or Abnormal
def predict_air_quality(co2, cng, co):

    # Make prediction
    prediction = rf_model.predict([[co2, cng, co]])  # New input data

    # Convert prediction back to label
    return 'Abnormal' if prediction[0] == 1 else 'Normal'
```

```
# Example: predict for new sensor data
new_data = {'co2': 1001, 'cng': 300, 'co': 30}

prediction = predict_air_quality(new_data['co2'], new_data['cng'],
new_data['co'])

joblib.dump(rf_model, 'air_quality_model.pkl')
print("✅ Model trained and saved as 'air_quality_model.pkl' successfully!")
# Print prediction
print(f"Predicted air quality: {prediction}")
add_data_to_csv(new_data['co2'], new_data['cng'],new_data['co'],prediction)
```

A-2.2. SAVEING SENSOR DATA CODE (Save sensor data.py):

```
import serial
import pandas as pd
import os
import re  # Import regular expressions to clean data

# Configure the serial port
SERIAL_PORT = "COM12"  # Change this to match your ESP32 port
BAUD_RATE = 115200
ser = serial.Serial(SERIAL_PORT, BAUD_RATE)
data_list = []
# Specify the Excel file path
file_path = "sensor_data.xlsx"
```

```
# Check if the file already exists. If it does, delete it to create a new one.
if os.path.exists(file_path):
    os.remove(file_path)
try:
    while True:
        try:
            line = ser.readline().decode('utf-8', errors='ignore').strip()  # Ignore non-UTF-8 bytes
            # Skip empty lines or ESP32 reset messages
            if not line or "rst:" in line or "ets" in line or "Err" in line:
                print(f"Skipping system message: {line}")
                continue
            # Clean the data: Extract numeric values from the string using regex
            values = re.findall(r'-?\d+\.?\d*', line)  # Handles negative and decimal values
            # Check if we got exactly 4 values (MQ7, MQ2, MQ135, Temperature)
            if len(values) == 4:
                try:
                    # Convert values to appropriate types
                    mq7_value = int(values[0])
                    mq2_value = int(values[1])
                    mq135_value = int(values[2])
                    temperature = float(values[3])
                    data_list.append([mq7_value, mq2_value, mq135_value,temperature])

                    # Save data to an Excel file
```

```
                df = pd.DataFrame(data_list, columns=["MQ7", "MQ2",
"MQ135","TEMP"])
                df.to_excel(file_path, index=False)


                print(f"Saved: MQ7 = {mq7_value}, MQ2 = {mq2_value}, MQ135
= {mq135_value}, Temp = {temperature}")
            except ValueError:
                print(f"Skipping invalid data: {values}")


        except UnicodeDecodeError:
            print("Received invalid characters, skipping...")


except KeyboardInterrupt:
    print("\nData collection stopped.")
    ser.close()
```

A-2.3. PREDICTION AND SUGGESTION CODE (APP.py):

```
from flask import Flask, jsonify, render_template
import pandas as pd
import random
import time
from joblib import load
from threading import Thread, Lock
app = Flask(__name__)
# Load trained model
try:
```

```
    model = load('air_quality_model.pkl')
except Exception as e:
    print(f"Error loading model: {e}")
    model = None

latest_data = {}
data_lock = Lock()  # To ensure thread safety

def read_latest_excel_data():
    try:
        df = pd.read_excel("sensor_data.xlsx")
        latest_row = df.iloc[-1]
        return int(latest_row['MQ2']), int(latest_row['MQ7'])
,int(latest_row['MQ135']),int(latest_row['TEMP']) # Convert to int to avoid
int64 issue
    except Exception as e:
        print(f"Error reading Excel: {e}")
        return None, None

def generate_random_co2():
    return random.randint(300, 1500)

def get_reasons_and_suggestions(target, co2, cng, co):
    reasons, suggestions = [], []
    if target == "Abnormal":
        reasons.append("Air quality is poor")
        suggestions.append("Increase ventilation, check surroundings")
```

```
    # CO₂ Levels Analysis
    if co2 > 1100:
        reasons.append("Dangerously high CO2 levels detected!")
        suggestions.append("Open windows, use an air purifier, improve
ventilation immediately.")
    elif co2 > 1000:
        reasons.append("CO2 levels slightly above normal.")
        suggestions.append("Consider opening windows or using AC ventilation.")

    # CNG Levels Analysis
    if cng > 1100:
        reasons.append("CNG levels are dangerously high!")
        suggestions.append("Check for leaks and evacuate if necessary.")
    elif cng > 1000:
        reasons.append("CNG levels slightly above normal.")
        suggestions.append("Monitor the gas source and ensure proper airflow.")

    # CO Levels Analysis
    if co > 80:
        reasons.append("Severe Carbon Monoxide (CO) detected!")
        suggestions.append("Leave the area immediately and check for leaks.")
    elif co > 70:
        reasons.append("CO levels slightly above normal.")
        suggestions.append("Ensure air circulation and avoid prolonged
exposure.")

    # Combined Cases
```

```
    if (co2 > 1000 and cng > 1000) or (co2 > 1000 and co > 70) or (cng > 1000 and co > 70):
        reasons.append("Multiple gases are above safe levels!")
        suggestions.append("Take immediate action: ventilate the area, check for leaks, and avoid prolonged exposure.")

    # If everything is normal
    if not reasons:
        reasons.append("Everything looks fine.")
        suggestions.append("No actions needed, air quality is good.")

    return reasons, suggestions

def predict_air_quality(co2, cng, co):
    if model:
        prediction = model.predict([[co2, cng, co]])
        return "Abnormal" if prediction[0] == 1 else "Normal"

    return "Unknown"

def update_data():
    global latest_data
    while True:
        cng, co ,co2,temp= read_latest_excel_data()
        if cng is not None and co is not None and co2 is not None:

            predicted_target = predict_air_quality(co2, cng, co)
```

```
            # Fix: Now correctly passing all arguments
            reasons, suggestions = get_reasons_and_suggestions(predicted_target,
co2, cng, co)

            with data_lock:
                latest_data = {
                    "co2": co2,
                    "cng": cng,
                    "co": co,
                    "temp":temp,
                    "target": predicted_target,
                    "reasons": reasons,
                    "suggestions": suggestions
                }
            print(f"Updated Data: {latest_data}")
        time.sleep(5)

@app.route('/')
def index():
    return render_template("index.html")

@app.route('/get_data', methods=['GET'])
def get_latest_data():
    with data_lock:
        return jsonify(latest_data)
```

```
@app.route('/check_model', methods=['GET'])
def check_model():
    return jsonify({"model_loaded": model is not None})


if __name__ == "__main__":
    Thread(target=update_data, daemon=True).start()
    app.run(debug=True)
```

APPENDIX-3

SOURCE CODE

A-3.1. CASCADING STYLE SHEETS CODE (STYLE.css):

```
/* Google Fonts */
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;600&family=
Orbitron:wght@500&display=swap');


/* Global Styles */
body {
    font-family: 'Poppins', sans-serif;
    background: radial-gradient(circle, #091236, #1b2a49, #2c3e50);
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    color: white;
}
/* Container */
.container {
    display: flex;
    width: 95%;
```

```
    max-width: 1400px;
    gap: 20px;
}
/* Graph Panel (Fixed Size & Left-Aligned) */
.graph-panel {
    flex: 0.7;
    background: rgba(255, 255, 255, 0.1);
    padding: 15px;
    border-radius: 10px;
    text-align: left;
    backdrop-filter: blur(10px);
    transition: transform 0.5s ease-in-out;
    height: fit-content;
}
.graph-panel:hover {
    transform: translateY(-5px);
}
.graph-panel canvas {
    width: 300px !important;
    height: 200px !important;
}
/* Dashboard */
.dashboard {
    flex: 2;
    padding: 20px;
    animation: fadeIn 1.5s ease-in-out;
```

```
}
/* Sensor Data */
.sensor-data p {
    font-size: 18px;
    margin: 5px 0;
}
/*  Temperature Styling */
#temperature-value {
    font-weight: bold;
    color: #ffcc00;
}
/* Status */
.status {
    font-size: 24px;
    font-weight: bold;
    padding: 12px;
    border-radius: 8px;
    transition: transform 0.5s ease-in-out;
}
/* History Panel (Fixed to Right) */
.history-panel {
    flex: 1;
    background: rgba(255, 255, 255, 0.1);
    padding: 15px;
    border-left: 2px solid rgba(255, 255, 255, 0.3);
    overflow-y: auto;
```

```
    max-height: 90vh;
    width: 250px;
}
```

A-3.2. HYPERTEXT MARKUP LANGUAGE CODE (INDEX.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Futuristic Air Quality Dashboard</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;600&famil
y=Orbitron:wght@500&display=swap"
        rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script> <!-- ✅ Chart.js
for Graphs -->
</head>
<body>
    <div class="container">
        <!-- Left: Graph Section (Compact & Fixed) -->
        <div class="graph-panel">
            <h2>📊 Current Sensor Data</h2>
            <canvas id="sensorChart"></canvas>
```

```
    </div>

    <!-- Center: Dashboard -->
    <div class="dashboard">
      <header>
        <h1>🌎 Air Quality Monitoring System</h1>
        <p>Real-time environmental tracking</p>
      </header>

      <div class="grid-container">
        <div class="card sensor-card">
          <h2> 📡 Live Sensor Data</h2>
          <div class="sensor-data">
            <p>CO₂: <span id="co2-value">--</span> </p>
            <p>CNG: <span id="cng-value">--</span> </p>
            <p>CO: <span id="co-value">--</span> </p>
            <p> 🌡️ Temperature: <span id="temp-value">--</span> °C</p>
<!-- ✅ Added Temperature -->
          </div>
        </div>

        <div class="card status-card">
          <h2> 📊 Air Quality Status</h2>
          <p class="status" id="air-status">Checking...</p>
        </div>
```

```
            <div class="card issues-card">
                <h2>⚠ Detected Issues</h2>
                <ul id="reasons"></ul>
            </div>

            <div class="card suggestions-card">
                <h2> 💡 Suggested Actions</h2>
                <ul id="suggestions"></ul>
            </div>
        </div>
    </div>

    <!-- Right: Full History Panel -->
    <aside class="history-panel">
        <h2> 📜 Previous Readings (<span id="history-count">0</span>)</h2>
        <ul id="history-list"></ul>
    </aside>
</div>

<script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
```

A-3.3. DYNAMIC CHANGE OF CONTENT CODE - JAVA SCRIPT (SCRIPT.js):

```
document.addEventListener("DOMContentLoaded", function () {
    const co2Element = document.getElementById("co2-value");
    const cngElement = document.getElementById("cng-value");
    const coElement = document.getElementById("co-value");
    const temperatureElement = document.getElementById("temp-value");
    const airStatusElement = document.getElementById("air-status");
    const reasonsList = document.getElementById("reasons");
    const suggestionsList = document.getElementById("suggestions");
    const historyList = document.getElementById("history-list");
    const historyCount = document.getElementById("history-count");
    let sensorChart;
    let sensorData = {
        labels: [],
        datasets: [{
            label: "CO2",
            data: [],
            borderColor: "rgb(255, 99, 132)",
            fill: false
        },
        {
            label: "CNG",
            data: [],
            borderColor: "rgb(54, 162, 235)",
            fill: false
```

```
        },
        {
            label: "CO",
            data: [],
            borderColor: "rgb(255, 206, 86)",
            fill: false
        },
        {
            label: "Temperature",
            data: [],
            borderColor: "rgb(255, 165, 0)",  // ✅ Orange for Temperature
            fill: false
        }]
    };
    function fetchData() {
        fetch("/get_data")
            .then(response => response.json())
            .then(data => {
                console.log("Fetched data:", data);

                // ✅ Update Live Values
                co2Element.innerText = data.co2;
                cngElement.innerText = data.cng;
                coElement.innerText = data.co;
                temperatureElement.innerText = data.temp; // ✅ Update
Temperature
```

```
            airStatusElement.innerText = data.target;
            airStatusElement.style.color = data.target === "Abnormal" ? "red" :
"green";
            // ✅ Update Detected Issues
            reasonsList.innerHTML = "";
            data.reasons.forEach(reason => {
                let li = document.createElement("li");
                li.innerText = reason;
                reasonsList.appendChild(li);
            });
            // ✅ Update Suggested Actions
            suggestionsList.innerHTML = "";
            data.suggestions.forEach(suggestion => {
                let li = document.createElement("li");
                li.innerText = suggestion;
                suggestionsList.appendChild(li);
            });
            // ✅ Update History Panel
            let historyEntry = document.createElement("li");
            historyEntry.innerText = `CO2: ${data.co2}, CNG: ${data.cng}, CO:
${data.co}, Temp: ${data.temp}°C → ${data.target}`;
            historyList.prepend(historyEntry);
            historyCount.innerText = historyList.children.length;
            // ✅ Update Graph
            updateGraph(data);
        })
```

```
            .catch(error => console.error("Error fetching data:", error));
    }
    function updateGraph(data) {
        let timestamp = new Date().toLocaleTimeString();

        if (sensorData.labels.length > 10) {
            sensorData.labels.shift();
            sensorData.datasets.forEach(dataset => dataset.data.shift());
        }

        sensorData.labels.push(timestamp);
        sensorData.datasets[3].data.push(data.temp);

        sensorChart.update();
    }
    const ctx = document.getElementById("sensorChart").getContext("2d");
    sensorChart = new Chart(ctx, { type: "line", data: sensorData });

    setInterval(fetchData, 5000);
    fetchData();
});
```