PALESTINE POLYTECHNIC UNIVERSITY

College of Information Technology and Computer Engineering

Department of Computer Engineering

Smart and Automated Air Quality Monitoring System for Industrial Facilities Using Machine Learning and Pollution Prediction

Project Team:

Qassam Yaser Awawdeh 211039

Zidan nidal talahmeh 211047

Laith awni Alfaqeh 211038

Supervisor:

Dr. Elayan Abu Gharbieh

January 26

Certification and Anti-Plagiarism Declaration

This is to certify that the introduction to graduation project titled Smart and Automated Air Quality Monitoring System for Industrial Facilities Using Machine Learning and Pollution Prediction' was prepared by the student(s) listed below under the supervision of Dr. Elayan Abu Gharbieh in partial fulfillment of the requirements for the Bachelor's degree in Engineering in Computer Systems Engineering. We affirm that no part of this work has been reproduced illegally or constitutes plagiarism, including but not limited to direct copying or unacknowledged use of others' work. All referenced materials have been properly cited and used solely to support and substantiate the project's ideas. By signing this declaration, we confirm our commitment to upholding academic integrity and certify that we have not and will not, engage in plagiarism, cheating, or any other violations of academic standards. We acknowledge that we bear full responsibility and accept any consequences should this declaration be found to have been violated.

Date:

Introduction To Graduation Project Group:

Qassam Yaser Awawdeh Signature:

Zidan nidal talahmeh Signature:

Laith awni Alfaqeh Signature:

ABSTRACT

This project presents the design and development of a smart and autonomous system for air quality monitoring and prediction in industrial environments. It aims to address the environmental and health risks resulting from industrial emissions and poor ventilation conditions by integrating sensor-based data acquisition and smart analytical models.

It gathers real-time environmental information and transmits this to a central processing unit wirelessly, where pollution trends are predicted and processed using machine learning algorithms. It operates air purification and ventilation automatically, according to predictions, to provide safe and sustainable working conditions. A dashboard offers simple monitoring, remote control, and instant alarms for more-than-safe air quality.

The prototype created demonstrates the potential of combining smart sensing and data-driven prediction to aid in enhanced air quality control, energy conservation, and occupational safety in factory plants.

ACKNOWLEDGEMENTS

In the name of Allah, we are incredibly grateful to our supervisor, Dr. Elayan Abu Gharbyeh, whose guidance, encouragement, and thoughtful feedback have been essential throughout the course of our graduation project, *"Smart Air Quality Monitoring System for Industrial Facilities Using Machine Learning and Pollution Prediction."* His support helped us navigate challenges, refine our ideas, and stay focused on our goals.

Our sincere thanks also go to Dr. Ayman Wazwaz and Dr. Amal Al-Dweik for their valuable advice, time, and support, which greatly enriched our project and learning experience.

We would also like to extend our appreciation to the faculty and staff of the Computer Systems Engineering Department at the College of Information Technology and Computer Engineering, Palestine Polytechnic University. Their continuous support and the resources they provided were vital in bringing this project to life.

To our families and friends — thank you from the bottom of our hearts. Your patience, motivation, and belief in us made a world of difference and kept us going when things got tough.

Lastly, a big thank you to everyone who helped us along the way — whether through advice, encouragement, or a helping hand. We are truly grateful.

Contents

List of Tables

List of Figures

Abbreviations

AI	Artificial Intelligence
IoT	Internet of Things
PM2.5	Particulate Matter 2.5 micrometers
PM10	Particulate Matter 10 micrometers
VOC	Volatile Organic Compounds
NH_4	Ammonia
HVAC	Heating, Ventilation, and Air Conditioning
SMS	Short Message Service
GDPR	General Data Protection Regulation
NO_x	Nitrogen Oxides
SO_x	Sulfur Oxides
CO	Carbon Monoxide
IDE	Integrated Development Environment
API	Application Programming Interface
REST	Representational State Transfer
FCM	Firebase Cloud Messaging
SVM	Support Vector Machine
LCD	Liquid Crystal Display
DHT11	Temperature and Humidity Sensor
MQ135	Air Quality Sensor
SRAM	Static Random Access Memory
GPIO	General Purpose Input/Output
UART	Universal Asynchronous Receiver/Transmitter
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
BLE	Bluetooth Low Energy
HEPA	High-Efficiency Particulate Air filter
MQTT	Message Queuing Telemetry Transport
Wi-Fi	Wireless Fidelity
MySQL	Structured Query Language Database
UPS	Uninterruptible Power Supply
LoRa	Long Range Communication
LPWAN	Low Power Wide Area Network
Li-ion	Lithium-ion Battery

1 Chapter 1: Introduction

1.1 Preface

Industrial air pollution represents a significant environmental issue as well as a major health concern. The current traditional air quality monitoring systems function independently without real-time predictive functions which restricts their ability to implement proactive pollution control measures. The research introduces a Smart Air Quality Monitoring System for Industrial Facilities Using Machine Learning and Pollution Prediction which combines IoT-based sensor inputs with artificial intelligence-driven analytics. The system seeks to improve air pollution monitoring and forecasting capabilities and management strategies to reduce environmental and health risks.

1.2 Problem Statement

Plastic recycling serves as an essential method to decrease environmental pollution yet it generates critical health dangers within recycling facilities. The shredding and washing and melting operations at recycling facilities produce dangerous pollutants which include PM2.5, VOCs, NO_2, CO_2, NH_3 and other substances that enter the indoor air. The health and safety of workers will suffer from these pollutants unless proper monitoring measures are implemented. The current industrial monitoring systems present high costs and deployment challenges for small and medium factories. A smart real-time air quality monitoring system needs to be developed because it should be affordable to ensure safe working conditions in plastic recycling environments

1.3 Project Aims and Objectives

The main goal of this project is to develop and deploy a smart air quality monitoring system that combines machine learning and real-time pollution prediction to enhance air quality management in industrial environments. The key objectives include: Deploying IoT-based air quality sensors for real-time environmental data collection.

a) The system will provide accurate forecasts of air quality conditions to enable proactive pollution management.
b) The system will provide an interactive dashboard with real-time monitoring and alerts to enhance environmental awareness.
c) The system will autonomously monitor pollution levels and take appropriate actions to maintain air quality.
d) The system will activate ventilation and filtration systems to reduce pollution exposure
e) The system will trigger warning notifications when pollution measurements surpass established safety limits.

f) The system provides analytical reports to industrial administrators and environmental regulatory bodies through its data-driven decision-making capabilities.
g) The system promotes sustainable industrial practices through its ability to support proactive pollution regulation and mitigation strategies.

The proposed system will connect conventional air quality monitoring to contemporary intelligent pollution management solutions to create a safer healthier industrial environment through its target objectives.

1.4 Project Requirements

To specify the system requirements, the following functional and non-functional requirements can be considered:

1.4.1 Functional requirement

1. Real-time Data Collection:
   - The system requires sensors to monitor air quality parameters including $PM_{2.5}$, PM_{10}, CO_2, NO_2, VOCs, temperature, and humidity in real time.
   - The sensors need to send data to a central processing unit according to scheduled time periods.
2. Data Processing and Storage:
   - The system requires sensor data to undergo filtering and cleaning operations before storing it in a structured database for historical analysis.
   - The system needs to perform real-time data aggregation alongside pre-processing functions for machine learning models.
   - Pollution Prediction:
   - The system needs to run machine learning models which predict upcoming air quality patterns.
   - The system needs to generate both short-term predictions that span one hour and long-term predictions that span one day.
3. Automated Decision-Making:
   - The system should use predefined thresholds to classify pollution levels into four categories: safe, moderate, high, critical.
   - The system should activate automatic responses when threshold values are surpassed.
4. Alert and Notification System:
   - The system needs to deliver immediate alerts through email and mobile applications to facility managers along with other relevant parties.

- The system displays warnings through digital dashboards and sets audible alarms for personnel working at the site.

5. User Dashboard and Reporting:
   - A mobile dashboard shows real-time and historical air quality data to users.
   - The system needs to create automated reports which regulatory agencies can use for compliance purposes.

1.4.2 Non-Functional requirement

1. Performance:
   a) The system must process the data received from the sensors within a period not exceeding five seconds.
   b) The system must be able to control at least 10 nodes for the system to be effective.
2. Reliability:
   a) The system efficiency must be efficient
   b) The system must provide a way to store backup data in the event of a malfunction, such as a power outage, either locally or on the network.
3. Scalability:
   c) The system must be scalable, allowing gradual expansion and integration of additional sensors efficiently without redesigning the entire system.

1.5 System Description

It is an AI-powered certified air quality system that integrates air quality monitoring, data analysis, and future prediction of values, taking necessary actions, such as impacts. The system uses a set of sensors, control devices, and a CPU for this system.

Core System Components:

a) Sensors: Real-time air quality data is gathered by a network of sensors which operates across the facility.
b) The data processing function of a small computer system using Raspberry Pi technology prepares the information and transmits it to servers.
c) The system implements predictive models to evaluate trends and predict pollution levels which enables preventive actions.
d) The system controls automated ventilation and purifier systems through its decisions while sending alerts to appropriate personnel.
e) The dashboard interface allows managers to monitor trends and generate reports while confirming that all operations comply with established rules.

The system focuses on worker safety, following regulations, and sustainability by using smart, data-driven actions instead of just reacting to problems.

1.6 Project Limitations/constraints

1. Technical Constraints:
   a) Limited to specific pollutants (PM2.5, Carbon Dioxide, Nitrogen Oxides, etc.) due to sensor availability.
   b) The need for historical data to predict and anticipate future outcomes by artificial intelligence may not exist in the early stages of operation.
2. Operational Constraints:
   a) Power dependence: The need for stable power units and in case of emergency the need for another electrical source such as batteries or electric generators
   b) Network dependence: The system needs a network to transfer data from sensors to the processing unit or between other processing units, regardless of the type of network.
3. Scope Constraints:
   a) The system does not directly control industrial machinery (only interfaces with HVAC/purifier systems).
   b) Geographical limitations: Calibration is needed for different industrial environments (e.g., chemical plants vs. textile factories).
   c) Regulatory Constraints:
   a) Compliance with local data privacy laws (e.g., GDPR for EU deployments).
4. Budget/Resource Constraints:
   a) High-precision sensors and machine learning infrastructure may incur significant costs.
   b) Requires periodic maintenance (sensor calibration, model retraining).

1.7 Project Schedule

Table 1. 1:Project schedule in the first semester

	The first semester				
➢ Week	1 - 5	5 - 6	6 - 7	7 - 8	8 - 15
➢ Selection of project Idea					
➢ Describe idea					
➢ Idea Analyses					
➢ Literature review					
➢ Design					
➢ Documentation					

Table 1. 2: Project schedule in the Second semester

	The Second semester				
➢ Week	1 - 5	5 - 6	6 - 7	7 - 8	8 - 15
➢ System Implementation					
➢ System testing					
➢ system operation					
➢ Documentation					

1.8 Report Outline

This report consists of the following: Chapter One is an introduction and overview of the project. We discussed the project's limitations, objectives, and when it would work in its ideal form. Chapter Two provides reviews of the theories used, as it uses air quality monitoring sensors and microcontrollers. Chapter Three will discuss the system components, the various design options proposed, and the software needed to design the system. We will also discuss the parts needed to design the system, such as sensors, processing units, interaction units, and output units. Chapter Four will be the appropriate conclusion that will be drawn after designing the experiment from Chapter One to Chapter Four, and the actual goal of applying this experiment on the ground.

2 Chapter 2: Background

2.1 Preface

In this chapter, we will discuss the necessary theoretical techniques and basic concepts needed to understand the project's working principle and explain them in general. We will discuss air quality concepts such as types, types of industrial pollutants, and how the Internet of Things system works as a technology for collecting and communicating data. The chapter also discusses the algorithms used in data analysis to enable real-time decision-making and the possibilities of machine learning. The decision-making method, the important procedures that are followed, and the methods of data processing and storage will also be discussed. Then, the project will be compared to previous projects, the similarities and differences between them will be clarified, and whether it was successful or not.

2.2 Theoretical background

Creating a smart system for monitoring and forecasting air quality requires integrating knowledge and technologies from multiple, distinct, and interconnected fields. Key concepts include:

2.2.1 Air Quality and Industrial Pollutants

Air pollution is pollution of the indoor or outdoor environment by any chemical, physical or biological agent that modifies the natural characteristics of the atmosphere [1]. Industrial operations generate multiple air contaminants which result in poor air quality while creating hazards. The system tracks the following important air pollutants:

- **Particulate Matter (PM2.5 and PM10):** The particles exist as inhalable matter with size ranges below 2.5 micrometers (PM2.5) and below 10 micrometers (PM10). The particles derive from combustion operations together with industrial production and dust emissions. The main health consequences affect the respiratory system and cardiovascular system [2]. Figure 2. 1 show the Size comparisons for PM particles

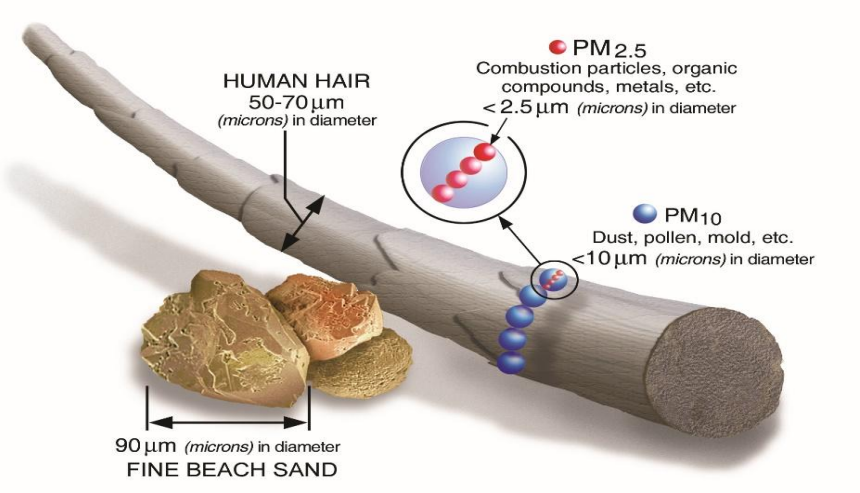


Figure 2. 1 : Size comparisons for PM particles

- **Carbon Dioxide (CO2):** The atmospheric gas carbon dioxide (CO2) exists naturally in concentrations ranging from 300 to 700 ppm [3]. CO2 exists as an asphyxiant gas in the atmosphere yet its indoor presence becomes dangerous when industrial areas or other poorly ventilated spaces exist. The presence of elevated CO2 concentrations in poorly ventilated areas results a negative health effect on workers. The gas-phase concentration of CO2 at larger levels results in multiple symptoms like faster breathing rate and lassitude and sleepiness and headache and convulsions and dyspnea and sweating and dizziness and narcosis [3].

- **Nitrogen Oxides (NOx):** (NOx) Refer to a group of nitrogen oxides such as Nitrogen Dioxide (NO2). NO2 functions as a gas indicator for nitrogen oxides (NOx) and represents these pollutants in the atmosphere. The air receives NO2 primarily from fuel combustion activities in vehicles and power plants and equipment operations. High NO2 concentrations in the air causes respiratory irritation which makes asthma worse and results in breathing difficulties and wheezing symptoms that require hospital visits. The human body develops asthma more easily and becomes more prone to respiratory infections when exposed to NO2 for extended periods. The most susceptible groups to NO2 exposure include people with asthma together with children and elderly individuals. NO2 in the atmosphere reacts with other elements to create acid rain that causes damage to sensitive ecosystems including lakes and forests [4].

- **Carbon Monoxide (CO):** The colorless odorless tasteless gas carbon monoxide emerges mainly from carbon-based fuel combustion that is incomplete such as gasoline diesel wood

and natural gas [42]. The gas CO builds up swiftly in industrial areas with limited ventilation especially when fuel-burning equipment operates in enclosed spaces. The gas poses a significant threat indoors because it binds strongly to hemoglobin to form carboxyhemoglobin which reduces blood oxygen delivery. The presence of elevated CO levels in the environment produces severe physical symptoms which include headaches and dizziness and confusion and nausea and unconsciousness or death in extreme cases. The risk of CO exposure increases for plastic recycling plant workers who operate machinery or experience accidental combustion during melting processes. The absence of detectable senses for CO requires continuous gas sensor monitoring to stop poisoning incidents and maintain workplace safety [42].

- **Environmental Parameters (Temperature and Humidity):** The measurements of temperature and humidity affect both pollutant formation and spread as well as sensor readings but they do not qualify as pollutants. The monitoring of these parameters ensures accurate data analysis and model functioning.

2.2.2 Cross-Platform Mobile Application (Flutter)

A mobile application is a software program designed to operate on portable devices such as smartphones and tablets. Cross-platform development allows a single codebase to run on multiple operating systems.

2.2.3 Flutter Framework

Flutter is an open-source framework created by Google for developing natively compiled applications from a single codebase. It includes a rich set of widgets and a powerful rendering engine that delivers consistent user interfaces across Android, iOS, desktop, and web platforms.

2.2.4 Data Programming Language

Dart is an object-oriented programming language developed by Google. It is designed for fast, modern application development and supports asynchronous programming. Dart compiles into native machine code, enabling smooth performance and quick execution in mobile and desktop environments.

2.2.5 RESTful API Communication

RESTful API works by enabling an application to interact with a server via HTTP methods like GET, POST, PUT, DELETE. Representational State Transfer (REST) is a style of architectural design employed in network communication. It is popular for backend processing with mobile applications. The rest command is useful when the server is interacted with because it does not consume a lot of resources and is very simple.

2.2.6 Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging (FCM) is a cloud-based platform that supports sending notifications and updates even when the application is not running. It is widely used to deliver alerts, reminders, and system updates efficiently.

2.2.7 Fl Chart

This library is open-source and that allow developers to display real-time data interactively and clearly.

2.2.8 Machine Learning (ML)

Machine learning (ML) is a branch of artificial intelligence (AI) focused on enabling computers and machines to mimic the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data. There are several methods of machine learning (ML), but in our project we will use Supervised learning method.

2.2.9 Joblib and PKL Files

Joblib is a Python library used for efficient serialization of machine learning models and numerical objects. It is used to save trained models and data preprocessors with minimal memory usage.

PKL is a binary file used for storing Python objects. It is used for storing trained models without having to retrain them. In this system, the AI model and scaler are stored as PKL files using the Joblib library. These files are then loaded at server startup.

- **Supervised learning**

Supervised learning, also known as supervised machine learning, is defined by its use of labeled datasets to train algorithms to classify data or predict outcomes accurately. As input data is fed into the model, the model adjusts its weights until it has been fitted appropriately. This occurs as part of the cross-validation process to ensure that the model avoids overfitting or underfitting. Some

methods used in supervised learning include neural networks, Naïve Bayes, linear regression, logistic regression, random forest, and support vector machine (SVM) [6]. Figure *2. 2* show Supervised learning model.

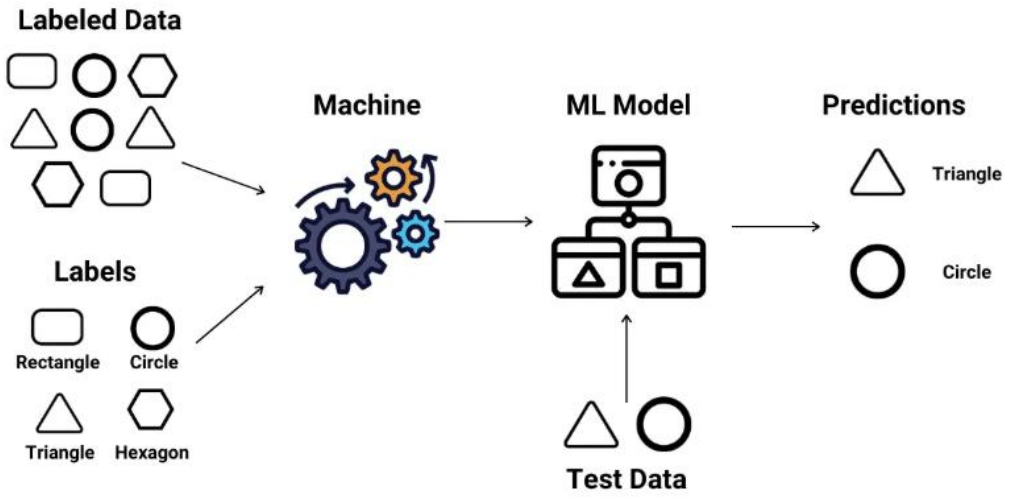


Figure 2. 2 : Supervised learning [7]

2.2.9 Data Processing and Management

Data management is the process of collecting, organizing, managing, and storing data in databases to support productivity, efficiency, decision-making, and future access as needed.

- **Database**

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

2.2.10 Automated Decision-Making and Control

Automated decision-making is the process of making a decision by automated means without any human involvement. In our project, the system will take actions without human actions based on the ML model that was trained before, and the threshold (Predefined limits for pollutant concentrations, based on occupational safety or environmental standards).

2.2.11 Alerting and Notification System

Alert notification systems allow for real-time dissemination of information and intelligence via equipment such as cellular phones, pagers, personal digital assistants, computers, etc. Their primary function is to quickly alert responders to potential threats or emergency situations and to provide direction on how to respond to alerts [8].

2.3 Literature Review

2.3.1 Developing a Method of Treating Air Pollution and Monitoring Air Quality

This project focuses on building a basic yet functional air quality monitoring system using an MQ135 gas sensor and DHT11 temperature and humidity sensor, connected to an Arduino Uno. The air quality readings are displayed on an LCD screen, and if pollution exceeds a preset threshold, the system activates a DC fan to help reduce pollutant concentration. The project prioritizes simplicity and affordability, aiming to provide a low-cost air purification mechanism suitable for domestic or small-space applications. Unlike the proposed system, it does not include machine learning or advanced data analytics. There is also no online connectivity or data logging [9].

- **Conclusion**: after these comparisons and what appeared in the **Table 2.** ***1*** , it became clear that there is a fundamental difference between the aforementioned project and our traffic project, its application in terms of the project's ability to explore gases, process data, predict future trends, methods of reducing pollution, and methods of dealing with data. Therefore, this project does not even cover 30% of our project.

Table 2. 1 Summary of comparison between our project and other projects

Feature	Our Project	Developing a Method of Treating Air Pollution and Monitoring Air Quality	Smart Air Quality Monitoring IoT-Based Infrastructure for Industrial Environments	Sensor-Based Machine Learning Approach for Indoor Air Quality Monitoring in an Automobile Manufacturing
Sensor	PM2.5, PM10, CO_2, NOx, VOCs, Temp, Humidity	MQ135, DHT11	Sensors for harmful gases and compound	CO_2, Fine Dust (PM), Temperature, Air Velocity
Microcontroller	raspberry pi / ESP	Arduino Uno	Arduino Uno	raspberry pi
Display	mobile dashboard	16x2 LCD screen	Local web server interface via Wi-Fi (browser-based, no LCD)	Not specified – likely internal data logs used for training ML models
Prediction	Yes - Machine Learning	No	No – Threshold-based alerts only	Yes – Machine learning models to predict pollutant concentrations
Automation	Yes – Intelligent action based on ML analysis	Yes – Fan turns on based on gas threshold	No – Threshold-based alerts only	No automation mentioned
Data Logging	Cloud + Historical Data	No	Cloud + Real-time only	Centralized data logging for training + evaluation
Target Area	Industrial Facilities	Indoor/domestic	Urban/Indoor spaces (general purpose)	Welding area in
Connectivity	Wi-Fi Enabled	No Internet Connectivity	Wi-Fi via ESP8266 module	No

2.3.2 Smart Air Quality Monitoring IoT-Based Infrastructure for Industrial Environments

This project utilized a network of IoT sensors (MQ135, DHT11) connected to an Arduino and ESP8266 Wi-Fi module to monitor real-time pollutants such as CO, NO2, and particulate matter in urban areas. Data was sent to a cloud server and displayed via a web dashboard. This system differs from our project in that it is unable to take measures to reduce the problem of rising pollution. It is a system that only captures and processes data. There are no actual measures to reduce pollution, nor are there any control systems [10].

- **Conclusion** In conclusion, as this **Table 2. *1*** shows, there is a clear difference between the two projects. The proposed project covered approximately the data acquisition and analysis part of the project, while the process of reducing and standardizing it and the orders that limit it by controlling certain matters that will be mentioned later were not addressed by this project. Therefore, the similarity rate is approximately 50%.

2.3.3 Sensor-Based Machine Learning Approach for Indoor Air Quality Monitoring in an Automobile Manufacturing

This study focuses on optimizing HVAC systems in an automobile manufacturing setting by utilizing machine learning algorithms to monitor indoor air quality. Data were collected from a welding area over two production weeks, measuring parameters such as CO_2 levels, fine dust (PM), temperature, and air velocity. Various machine learning models were trained and tested to predict pollutant concentrations, with the Long Short-Term Memory (LSTM) model demonstrating the highest accuracy ($R^2 = 0.821$). The study concluded that integrating machine learning with sensor data can effectively reduce energy consumption in HVAC systems without compromising air quality, after taking a look at this project, it became clear that the goal of the project is to save energy in heating and cooling systems based on readings and data of air quality monitoring in terms of temperature, humidity and pollution. This is what our project shares with the percentage of data collection, but differs in the basic goal and the problem that is sought to be solved [11].

- **Conclusion** In this research, as shown above and in the **Table 2. *1*** , it became clear to us that there is a fundamental difference in the goal of conducting the two projects. In the proposed project, the goal was to improve electricity consumption in cooling and air conditioning systems by collecting pollution data and using it to analyze the data and control the operating methods of cooling and heating systems. As for our project, it is a safety system to reduce pollution and maintain the safety of workers in industrial areas, not only to provide electricity in cooling and heating systems, but also to fully control ventilation and purification systems and even industrial machines.

2.3.4 Summary of the difference

After studying these three projects, it became clear that the percentage of difference between them and our proposed system ranges between 50% and 70%, depending on the focus and depth of each study.

The first project demonstrated that the process of reading air pollutants is relatively simple and incomplete, with pollution control being limited to the activation of a single fan, without any data processing or prediction mechanisms.

The second project presented a more advanced approach, where air quality monitoring, data processing, and predictive analysis are performed. However, it lacks real-time interventions to mitigate pollution. The system is limited to issuing alerts, unlike our project, which takes multiple practical actions based on real-time data.

As for the third project, it diverges from the core objective of our system. While it utilizes air quality data and machine learning to optimize energy consumption in HVAC systems, it does not offer direct methods for pollution reduction. The project's main focus is on reducing electricity usage by turning heating or cooling systems on/off based on air quality data.

In contrast, our project not only features accurate environmental data readings and smart electricity usage, but it also includes multiple mechanisms to maintain air quality, reduce pollution, and ensure the safety of individuals in industrial environments

2.4 Summary

In this chapter, we began by discussing the contents of this section and what will be covered in it. We then explained the Theoretical that will be used in implementing this project, citing the sources of this information and referring to them. After that, we made a comparison between our project and three other projects from both theoretical and practical perspectives, highlighting the similarities and differences. We also showed the percentage of conformity between each project and the materials of our project in advance, assessing whether they address the problem of our project or not. It became clear to us that the three proposed projects do not address the problem of our project, as mentioned above.

3 Chapter 3: System Design

3.1 Preface

This chapter provides an examination of the key hardware and software elements necessary for our project. Each major subsystem will be discussed including air pollution detection mechanisms, data processing algorithms, mobile monitoring interfaces, and the deployment of cloud and local databases. Additionally, the machine learning approaches used for predictive analytics are detailed.

3.2 Hardware Components

This section outlines the hardware components options for the system. All components were chosen to optimize reliability, performance, cost-efficiency, and suitability for industrial environments

3.2.1 Controller

A. Node Controller

Table 3. 1: Comparison between Node Controllers

Feature	**Selected: ESP32 Dev Board**  **Figure 3. 1: ESP32**	**Evaluated but not selected: Pico**  **Figure 3. 2: Pico**
Type	Microcontroller	Microcontroller
Processor	Dual-core 240 MHz	Dual-core ARM Cortex-M0+ (RP2040) running up to 133 MHz
Memory	520 KB SRAM	264 kB on-chip SRAM + 2 MB on-board flash
Connectivity	Built-in Wi-Fi and Bluetooth	No built-in Wi-Fi or Bluetooth
GPIO Pins	34	26
Power Efficiency	Supports deep sleep	low-power sleep
Interface	Digital, Analog, SPI, I2C, UART	SPI, I²C, UART, ADC, PWM, PIO
Key Features	Wireless communication, high processing speed	Basic microcontroller for simple tasks
Final Decision	Powerful and integrated wireless features ideal for industrial applications	Recommended for embedded control tasks

[12]

➢ **Why ESP32?**

As we show in *Table 3. 1* we have these benefits in ESP32 more than Arduino uno.

- Built-in Wi-Fi and Bluetooth reduce the need for external modules.
- Powerful dual-core processor (240 MHz) suitable for pre-processing sensor data.
- Supports multiple analog and digital sensors.
- Energy-efficient, supports deep sleep mode for power optimization.

B. Central Controller

Table 3.2: Comparison between Central Controllers

Feature	**Selected: Raspberry Pi 5 (8GB)**  **Figure 3. 3: Raspberry Pi 5**	**Evaluated but not selected: NVIDIA Jetson Nano (4GB)**  **Figure 3. 4: NVIDIA Jetson Nano**
Type	Single-board Computer	Single-board Computer
CPU	Quad-core 64-bit ARM Cortex-A76 processor @ 2.4 GHz	Quad-core ARM Cortex-A57 @ 1.43 GHz
GPU	Broadcom Video Core VII graphics processor, supports OpenGL ES 3.1 and Vulkan 1.2	128-core Maxwell GPU (superior GPU)
RAM	8 GB LPDDR4	4 GB LPDDR4
Storage	microSD + USB 3.0 SSD support optional PCIe 2.0 ×1 for NVMe or high-speed HAT storage	microSD + USB 3.0 SSD support
Connectivity	Ethernet, Wi-Fi, Bluetooth (via USB dongle)	Ethernet only (no built-in Wi-Fi)
Power Supply	5V/5A USB-C	5V/4A barrel jack
HDMI Output	Dual micro-HDMI ports	HDMI + DisplayPort

[13], [14]

➢ **Why Raspberry Pi 5?**

As we show in *Table 3.2* we have these benefits in: Raspberry Pi more than NVIDIA Jetson Nano.

- Quad-core CPU supports Linux-based OS and can run Python-based ML models.
- Built-in Wi-Fi + Ethernet for flexible connectivity.
- Supports data logging, dashboard hosting, and communication with multiple nodes.
- HDMI output for diagnostics and setup.
- Equipped with IUNIKER ABS Case including cooling fan and heatsinks for thermal management.

3.2.2 Air Quality Sensors

A. Air Quality (VOCS) Sensor

Table 3. 3: Comparison between Air Quality Sensor: MQ-135 and MQ-7

Feature	**Selected: MQ-135**  **Figure 3. 5: MQ-135**	**Evaluated but not selected: MQ-7**  **Figure 3. 6 : MQ-7**
Gas Detection	CO_2, NH_3, benzene, alcohol, smoke	Primarily Carbon Monoxide (CO)
Sensitivity	Broad range of gases	Only CO detection
Output	Analog	Analog
Power Supply	5V	5V
Key Features	Indoor air quality monitoring for multiple gases	CO-specific sensor only
Final Decision	MQ-135 covers more pollutants essential for air quality analysis	MQ-7 is too specific (CO only)

[16], [29]

➢ **Why MQ-135?**

As we show in **Table 3.** ***3*** we have these benefits in: MQ-135 more than MQ-7

- Detects CO_2, NH_3, benzene, alcohol, smoke, and other harmful gases.
- Widely used in indoor air quality monitoring.
- Cost-effective and easy to interface with analog inputs.

B. Smoke and Gas Sensor

Table 3. 4: Comparison between Air Quality Sensor: MQ-2 and MQ-5

Feature	**Selected: MQ-2**  **Figure 3. 7: MQ-2**	**Evaluated but not selected: MQ-5**  **Figure 3. 8: MQ-5**
Gas Detection	Smoke, LPG, methane, hydrogen	LPG, natural gas, coal gas
Sensitivity	Broad, sensitive to smoke and gases	Focused on LPG and natural gas
Output	Analog	Analog
Power Supply	5V	5V
Key Features	Fire detection and gas leak detection	Mainly gas leakage detection
Final Decision	MQ-2 provides better smoke detection capabilities important for early warning	MQ-5 less sensitive to smoke

[19], [30]

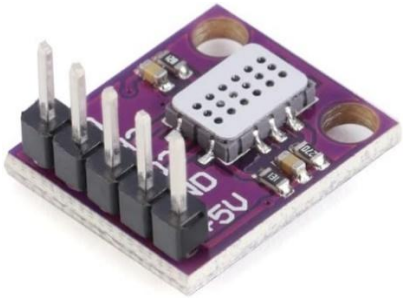
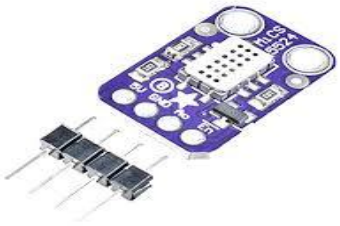
➢ **Why MQ-2?**

As we show in **Table 3.** ***4*** we have these benefits in: MQ-2 more than MQ-5

- Sensitive to LPG, methane, hydrogen, smoke, and other gases.
- Provides early fire detection or gas leakage detection capability.
- Simple analog output and low cost.

C. Multi-Gas Sensor

Table 3. 5: Comparison between Air Quality Sensor: MiCS-6814 and: MiCS-5524

Feature	**Selected: MiCS-6814**  **Figure 3. 9: MiCS-6814**	**Evaluated but not selected: MiCS-5524**  **Figure 3. 10: MiCS-5524**
Gas Detection	NO_2, CO, NH_3	CO, CH_4, LPG
Channels	3 independent channels	2 channels
Output	Analog	Analog
Power Supply	5V	5V
Size	Small	Small
Key Features	Multi-gas detection for industrial gases	Limited to combustible gases
Final Decision	MiCS-6814 detects critical industrial pollutants (NO_2, NH_3)	MiCS-5524 is more for general safety systems

[18], [31]

- **Why MiCS-6814?**

  As we show in **Table 3. *5*** we have these benefits in: MiCS-6814more than MiCS-5524

  - Detects NO_2, CO, and NH_3 with three internal sensing elements.
  - Small size, accurate detection, and industrial-grade performance.
  - Low power consumption and suitable for continuous monitoring.

D. Particulate Matter Sensor

Table 3. 6: Comparison between Air Quality Sensor PMS5003 and DSM501A

Feature	**Selected: PMS5003**  **Figure 3. 11: PMS5003**	**Evaluated but not selected: DSM501A**  **Figure 3. 12: PDSM501A**
PM Measurement	PM1.0, PM2.5, PM10	PM2.5
Technology	Laser scattering	Infrared LED scattering
Output	UART	PWM
Accuracy	±10%	Rough estimation only
Key Features	High precision, calibrated output	Low-cost, rough estimation
Final Decision	PMS5003 offers high accuracy essential for industrial monitoring	DSM501A less reliable and less accurate

[17], [32]

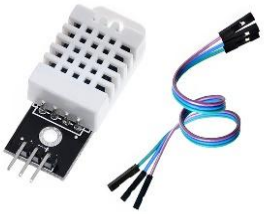
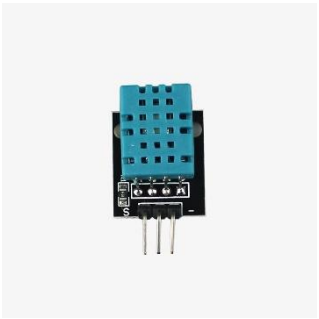
- **Why PMS5003**

  As we show in **Table 3. *6*** we have these benefits in: PMS5003 more than DSM501A

  - Laser-based sensor with high precision in detecting fine particles (PM2.5 and PM10).
  - Long lifespan and factory-calibrated output.
  - Ideal for detecting dust and particulate pollution in industrial environments.

E. Temperature and Humidity Sensor

Table 3. 7: Comparison between Air Quality Sensor DHT-22 and DHT-11

Feature	**Selected: DHT-22**  **Figure 3. 13: DHT-22**	**Evaluated but not selected: DHT-11**  **Figure 3. 14: DHT-11**
Temperature Range	-40°C to 80°C	0°C to 50°C
Humidity Range	0–100% RH	20–90% RH
Accuracy (Temp)	±0.5°C	±2°C
Accuracy (Humidity)	±2% RH	±5% RH
Key Features	Wide range, high accuracy	Basic, low-accuracy sensor
Final Decision	DHT-22 suitable for industrial variability	DHT-11 too limited for harsh environments

[15], [33]

- **Why DHT-22?**

As we show in **Table 3.** *7* we have these benefits in DHT-22 more than DHT-11

- High accuracy (±0.5°C temperature, ±2% humidity).
- Wide measurement range (–40°C to 80°C, 0–100% RH).
- Durable and suitable for industrial environments.

F. Carbon Dioxide Sensor Module

Table 3. 8: Comparison between Air Quality Sensor NDIR CO_2 Sensor and MH-Z14A

Feature	**Selected: NDIR CO_2 Sensor**  **Figure 3. 15: NDIR CO_2 Sensor**	**Evaluated but not selected: MH-Z14A (Electrochemical CO_2 Sensor)**  **Figure 3. 16: MH-Z14A**
Technology	Non-dispersive infrared (NDIR)	Electrochemical
CO_2 Range	400–5000 ppm	0–10000 ppm
Accuracy	±(50 ppm + 5% of reading)	±50 ppm
Output	UART, PWM	UART, Analog
Key Features	High stability, not affected by humidity	Limited lifespan, chemical drift
Final Decision	NDIR sensors are more reliable long-term	Electrochemical sensors degrade over time

[20], [28], [34]

- **Why NDIR CO_2 Sensor?**

As we show in **Table 3.** ***8*** we have these benefits in NDIR CO_2 Sensor more than MH-Z14A

- High precision CO_2 concentration measurement using non-dispersive infrared (NDIR) technology.
- Immune to temperature and humidity variations compared to chemical sensors.
- Suitable for accurate indoor air quality analysis

Why These Sensors?

- They collectively cover critical air quality parameters: CO_2, VOCs, smoke, NO_2, NH_3, PM2.5, PM10, temperature, and humidity.
- Chosen for high accuracy, industrial reliability, low power requirements, and compatibility with the ESP32 and Raspberry Pi platforms

3.2.3 Air Reduction Methods

The system controls air reduction components based on pollution thresholds:

- **Ventilation Fan (AC 220V or DC 12V):**
  Activates to exhaust polluted air when PM levels are moderate to high.
- **Industrial Air Purifier (HEPA + Activated Carbon):**
  Reduces fine dust (PM2.5/PM10) and VOCs. Automatically triggered when PM/VOC levels exceed thresholds.
- **HVAC System Integration (Smart Relay Controlled):**
  The Raspberry Pi controls existing HVAC systems to improve ventilation and air recirculation.
- **Alarms, Speakers, and Lights**

  In high-risk situations, these devices provide visual and audible warnings to those present to evacuate.
- **LCD display Screen**

  The severity level is displayed on the screen and is constantly updated

3.2.4 Connection Method (Node ↔ Centralization)

- **Why Wi-Fi?**

As we see in … below we get these benefits in WIFI

- Available in most industrial facilities.
- High data rate for real-time monitoring.
- Easy integration with Raspberry Pi using MQTT or HTTP.

- **Rejected Options:**

- Bluetooth: Short range, poor scalability.
- Wired UART: Inflexible for scalable, distributed systems.

Table 3. 9: Comparison between Connection Method: Wi-Fi and LoRa

Feature	**Selected: Wi-Fi (via ESP32)**	**Evaluated but not selected: LoRa (RFM95 Module)**
Type	Wireless Local Area Network (WLAN)	Low Power Wide Area Network (LPWAN)
Range	~50–100 meters indoors, ~300 meters outdoors	Up to 10 km (line of sight)
Data Rate	High (~100 Mbps)	Low (~0.3–50 kbps)
Latency	Low	High
Power Consumption	Moderate	Very Low
Cost	Low (Wi-Fi modules are cheap)	Moderate (LoRa modules are more expensive)
Infrastructure	Needs Wi-Fi router or Access Point	No infrastructure needed if peer-to-peer
Complexity	Easy to configure	More complex setup (frequencies, duty cycle limits)
Key Features	Fast, real-time communication, easy internet integration	Long-distance, low-power communication
Final Decision	Wi-Fi was selected because it supports high-speed real-time sensor data transmission inside industrial facilities with existing Wi-Fi infrastructure	LoRa was rejected because although it offers long range, it has low data rates and would slow real-time monitoring

[22], [23]

3.2.5 Power Supply

Table 3. 10: Power Supply Specification

Feature	**5V DC Adapter / Li-ion Battery / Solar Backup**
Voltage	5V DC regulated
Power Stability	Very stable (via adapters and battery regulation)
Backup Power	Easy (Li-ion batteries, solar charging)
Safety	High (low voltage)
Power Consumption	Low (optimized for low-power devices like ESP32)
Complexity	Simple (plug-and-play)
Cost	Low (adapters, batteries are cheap)
Key Features	Safe, reliable, portable, green energy option with solar
Final Decision	Selected 5V system ensures safety, scalability, and easy green energy integration

A result for last Table 3. 10 we get that as result

- **Node Power Supply:**
  - 5V 2A adapter or Li-ion battery + step-down converter. [24]
  - Optional: Solar + battery for green energy + backup.[25]
- **Central Unit:**
  - Raspberry Pi official 5V 3A USB-C adapter.
- **Backup Power:**
  - UPS unit for Raspberry Pi to prevent data loss.
  - Battery bank with charging circuit for ESP32 nodes.

3.3 Software Components

In this section, we will introduce the software & Algorithm we're going to use.

3.3.1 Pollution Level Detection

Pollution level detection is the process to detect the air quality levels for each sensor. In this section we will discuss the pollution levels for each sensor. *Table 3. 11* show these levels. [35], [35], [36], [14], [38], [39], [40], [41],[43]

Table 3. 11: Pollution levels

Sensor	Pollutant	Safe	Moderate	High	Critical
PMS5003	PM2.5 (µg/m³)	0-12.5	13-35	35.5-55	More than 55
PMS5003	PM10 (µg/m³)	0–54	55–154	155–254	More than 254
MQ-135	CO2 (ppm)	400–800	801-1200	1201-2000	More than 2000
MiCS-6814	NO2 (ppb)	0-50	51-100	101-200	More than 200
MQ-135	VOCs (ppb)	0-0.5	0.51-1	1.1-3	More than 3
DHT22	Temperature (°C)	20-26	27-30	31-35	More than 35
DHT22	Humidity (%)	30-60	20-29 or 61-70	10-19 or 71-80	Less than 10 or More than 80
MiCS-6814	Co (ppm)	0-35	36-100	101-200	More than 200

3.3.2 Automated Process Algorithm

In this subsection, we are discussing how the system analyzes pollution data as it comes in, decides on the air quality level, and then automatically kicks off the right response – all without needing a person to step in. The whole point of automation, really, is using technology or systems to get results without much human interaction. Our smart air quality setup relies heavily on this. The automatic system constantly checks the live pollution feed, makes a call on the air quality, and then takes action by itself. Why is this important? It means industrial sites can stay safe around the clock, even if staff aren't available. It's always monitoring, making smart choices, and controlling things like fans, speakers or alerts based on the pollution limits we've set.

3.3.3 Structure of the Algorithm

The Automated Process Algorithm operates through four main stages:

A. Input Acquisition

- Pollution data is collected from sensors measuring PM2.5, PM10, CO_2, NO_2, VOCs, temperature, and humidity.
- The sensor node collects readings every 30 seconds to capture short-term environmental fluctuations; however, the processed and aggregated data is transmitted to the Raspberry Pi every five minutes to optimize network usage and ensure stable real-time monitoring.

B. Data Classification and Decision-Making

Each pollutant reading is compared against pre-established safety thresholds as shown in Table 3.1 Based on concentration readings, the system rates the air quality into four levels: Safe, Moderate, High, and Critical.

Dependent on the classified status, the response is dictated by the decision-making algorithm:

- Safe: No response.
- Moderate: A notification is sent to facility managers via the mobile application, email, or SMS
- High: Automated mitigation response is initiated.
- Critical: Emergency alarms are triggered and warnings are sent to environmental authorities.

C. Action Execution

The decision is translated into real-world actions through:

- T General-Purpose Input/Output (GPIO) signals to relays controlling fans, air purifiers, alarms.
- Network commands to control HVAC and emergency protocols.
- Dashboard updates and alert notifications.

3.3.4 Internet of Things (IoT)

In this subsection we talk about the Internet of Things (IoT) communication system enables real-time data collection, transmission, device control, and remote monitoring by connecting multiple sensor nodes to a centralized server using wireless networking technologies.[44]

- **IoT System Architecture**

Table 3. 12: IoT System Architecture

Layer	Function	How it Works
Sensing Layer	Collect environmental data from multiple points.	1. Sensors (PM2.5, PM10, CO_2, NO_2, VOCs, Temperature, Humidity) connected to ESP32 read the surrounding environment. 2. Sensor readings are captured every 30 seconds. 3. Data is preprocessed by ESP32 (filtering noise, averaging values).
Network Layer	Transfer data wirelessly to the server.	4. ESP32 connects to Wi-Fi network. 5. ESP32 publishes the formatted sensor data to MQTT topic. 6. Raspberry Pi runs Mosquito broker and subscribes to the topics to receive sensor messages. 7. The network also allows Raspberry Pi to receive remote control commands from users via MQTT
Processing Layer	Analyze incoming data and trigger system actions.	8. Raspberry Pi stores incoming messages into a local database (MySQL). 9. Raspberry Pi runs Python scripts that classify pollution levels (Safe, Moderate, High). 10. Based on classification, Raspberry Pi controls output devices (fans, purifiers, alarms) via GPIO pins. 11. If a user sends a remote-control command (e.g., Turn ON fan), Raspberry Pi processes it immediately and updates the database. Note: In critical pollution cases, the ESP32 sensing node immediately sends the data to the Raspberry Pi, which triggers the ESP32 action node and sends an instant mobile alert to the manager.
Application Layer	Visualize data, send alerts to users and remote control.	12. The server updates a mobile dashboard with real-time air quality data. 13. Users can monitor air quality, receive email alerts based on pollution events. 14. Users can also send remote control commands (start/stop fans, alarms) via dashboard buttons, with actions reflected immediately on the system.

3.3.5 Cross-Platform Mobile Application

A mobile application was designed to act as a user-friendly tool for an air quality monitoring system. Users may control the application from any location with internet access. The mobile application provides fully real-time visualization of the sensor readings collected by the ESP32 nodes. To achieve this, the Raspberry Pi pushes the latest processed values to a cloud backend (Firebase), which allows the Flutter application to display continuously updated measurements

without relying on the database refresh interval. This ensures truly live monitoring, even when the Raspberry Pi stores aggregated readings every few minutes for machine learning and historical analytics.

The application also supports:

- Live dashboard with instant updates (Safe/Moderate/High/Critical).
- Interactive charts with real-time and predicted values.
- Alerts and notifications through Firebase Cloud Messaging.
- Remote control of fans, purifiers, alarms, and warning lights.
- Auto/Manual modes with logged user commands.
- Daily compliance reports and PDF export.
- Multi-user access with role-based permissions (Admin / Operator).

The mobile application visualizes both the real-time sensor values and the predicted pollution levels generated by the machine learning model running on the Raspberry Pi. This allows factory managers to see upcoming air quality risks before they occur and take preventive actions.

3.3.6 Machine Learning Implementation

To enhance the system's intelligence and predictive capabilities, a supervised machine learning model is integrated into the central unit (Raspberry Pi). The primary goal of the model is to analyze historical and real-time pollution data and predict future air quality levels within a specific time window (e.g., one hour ahead).

The machine learning workflow consists of the following main stages:

- **Data Collection:**

Sensor readings such as PM2.5, PM10, CO_2, NO_2, VOCs, temperature, and humidity are collected continuously by the ESP32 nodes and transmitted to the Raspberry Pi. All readings are timestamped and stored in a structured MySQL database.

- **Data Preprocessing:**

The stored data undergoes cleaning and normalization to remove outliers, handle missing values, and scale features for consistent input into the ML model. This step ensures accurate and stable learning behavior.

- **Model Training:**

A supervised learning approach is utilized with the Stochastic Gradient Descent Regressor (SGDRegressor) algorithm. The model is further trained on labelled past sensor data where

concentrations of airborne pollutants ($PM_{2.5}$, PM_{10}, CO_2, NO_2, VOCs, CO) and corresponding environmental factors (temperature, humidity) are known.

Offline training is performed using Python-based packages such as scikit-learn to establish a baseline predictive model that can make estimates of pollution levels 60 minutes in advance. After

deployment, the model operates on the Raspberry Pi, with it continually updating incrementally (online learning) with fresh sensor readings. This adaptive training process supports the model in learning adaptively from existing data dynamically, which increases prediction accuracy and maintains uniform forecasting of future air quality conditions.

- **Real-Time Prediction:**

Once the model is trained and validated, it is saved and deployed on the Raspberry Pi. During system operation, the Pi feeds new live sensor data into the model to generate predictions of upcoming pollution levels (e.g., one hour into the future). These predictions are then used to trigger proactive decisions such as early activation of fans or alert notifications.

- **Continuous Learning:**

The system can be extended in the future to support periodic retraining using newly collected data, allowing the model to adapt to changes in environmental behavior over time.

- **AQI Estimation Using a Normalized Weighted Method**

In this system, the Air Quality Index computation is performed using a weighted normalization method instead of the traditional breakpoint-based Air Quality Index calculation. This computation aims to offer a computationally friendly and interpretable air quality metric.

This is done by normalizing every concentration of pollutants with respect to a predefined limiting value as presented in Table 3.1:

$n_i = C_i / L_i$, where C_i represents the measured concentration of pollutant i, and L_i denotes the corresponding reference limit.

The normalized pollutant values are combined using a weighted linear aggregation reflecting their relative health impacts:

$$AQI_{weighted} = (0.40 \cdot n_{PM2.5} + 0.25 \cdot n_{CO2} + 0.15 \cdot n_{VOC} + 0.10 \cdot n_{NO2} + 0.10 \cdot n_{CO}) \times 300$$

The highest weight for $PM_{2.5}$ is chosen due to the fact that it has very bad health effects. In comparison, CO_2, VOCs, NO_2, and CO have proportionally reduced weights. The calculated AQI value is limited to a range of 0-500 to ensure further compatibility with widely used AQI scales and to ease threshold-based decision logic.

This will be the weighted AQI value and should work as the target output of the supervised machine learning model to predict the levels of AQI in the future with much accuracy and proactive air quality management decisions.[45],[46],[47]

3.3.7 Cloud and Local Database

In this section, we will talk about how we manage and store the air quality data that the system collects. To do this, we decided to use both a local database on the Raspberry Pi and cloud database for backup and remote access. We will explain how the system saves sensor readings, how it handles remote control commands from users, and how the local and cloud databases work together to make sure that no data is lost and everything stays organized.

- **Purpose of Database**

  - Store all sensor readings automatically as they happen.
  - Keep a history of air quality data to understand pollution trends.
  - Record every action the system takes whether it's automatic (like turning on a fan) or manual (when a user sends a remote command).
  - Back up the information safely in the cloud.
  - Allow users to monitor and control the system from anywhere.

- **How the Databases Work Together**

Table 3. 13: How the databases work together

Type of Storage	What It Does	How It Works
Local Database (on Raspberry Pi)	Saves sensor readings, pollution levels, system alerts, and user control commands immediately.	- Every time the Raspberry Pi receives new sensor data or a user control command, it writes the information into tables inside the local database using MySQL. - This database supports real-time operations and live dashboard updates.
Cloud Database	Creates a backup and allows remote access.	- The Raspberry Pi can push copies of the data to a cloud server at regular intervals for example, every hour. - This means even if the Raspberry Pi crashes or loses power, the data is still safe and can be accessed from anywhere. - If cloud server does not receive data at the predefined interval, it sends notification to user via email or SMS

- **Data in Database**

The database stores these data:

1. User Name.
2. Sensor Reading.
3. System Action.
4. User Commands.
5. Alert Sent.

So, the tables we need is:

1. Users: Store the name of user that has the authorization for access system.
2. SensorData: Store all sensor reading.
3. SystemActions: Store Automatic system operations.
4. UserCommands: Store Manual controls by users.
5. Alerts: Store Sent alert messages.

The following is the structured of our tables:

Table 3. 14: Users Table

Table Name	Field Name	Data Type	Key Type	Description
Users	UserID	INT	PK	Unique ID for each user
	Username	VARCHAR(50)		User's login name
	Email	VARCHAR(100)		User's email address
	PasswordHash	VARCHAR(255)		Encrypted password (hash)
	Role	VARCHAR(20)		Type of user (Admin, Operator)
	CreatedAt	DATETIME		Account creation time

Table 3. 15: SensorData Table

Table Name	Field Name	Data Type	Key Type	Description
SensorData	ReadingID	INT	PK	Unique ID for each sensor reading
	Timestamp	DATETIME		Date and time of the reading
	SensorType	VARCHAR(50)		Type of sensor
	Value	Float		Recorded value

Table 3. 16: SystemActions Table

Table Name	Field Name	Data Type	Key Type	Description
SystemActions	ActionID	INT	PK	Unique ID for each action
	Timestamp	DATETIME		Date and time of the action
	ActionType	VARCHAR(50)		Action type (e.g., Fan ON)
	TriggerLevel	VARCHAR(20)		Pollution level

Table 3. 17: UserCommands Table

Table Name	Field Name	Data Type	Key Type	Description
UserCommands	CommandID	INT	PK	Unique ID for each command
	Timestamp	DATETIME		Date and time of the command
	CommandType	VARCHAR(50)		Type of manual command
	UserID	INT	FK → Users(UserID)	User who is sued the command

Table 3. 18: Alerts Table

Table Name	Field Name	Data Type	Key Type	Description
Alerts	AlertID	INT	PK	Unique ID for each alert
	Timestamp	DATETIME		Date and time of the alert
	AlertType	VARCHAR(50)		Type of alert (e.g., Warning)
	RecipientInfo	VARCHAR(100)		Email/SMS target
	AlertMessage	VARCHAR(255)		Short textual description of the alert

3.4 Conceptual System Design

The Smart and Automated Air Quality Monitoring System for Industrial Facilities consists of several interconnected modules working together to ensure a healthy and safe environment. The system relies on a network of high-precision sensors to measure key air pollutants such as PM2.5, PM10, CO_2, NO_2, volatile organic compounds (VOCs), along with temperature and humidity. The collected data is transmitted to an edge processing unit (such as a Raspberry Pi), where initial filtering operations are performed. Subsequently, the data is sent to a central processing unit for

deeper analysis using machine learning models to predict future pollution levels based on historical patterns and real-time fluctuations. Based on the predictions and predefined safety thresholds, the system makes automated decisions such as activating ventilation systems, air purifiers, or triggering emergency alarms when necessary. Additionally, the system provides an interactive mobile dashboard for real-time and historical data visualization, sends notifications via SMS and email, and generates periodic analytical reports to support industrial management and ensure compliance with environmental standards. The system is designed to be scalable, data-secure, and compliant with international air quality and environmental regulations These things are clear in the project block diagram Figure 3. 17.

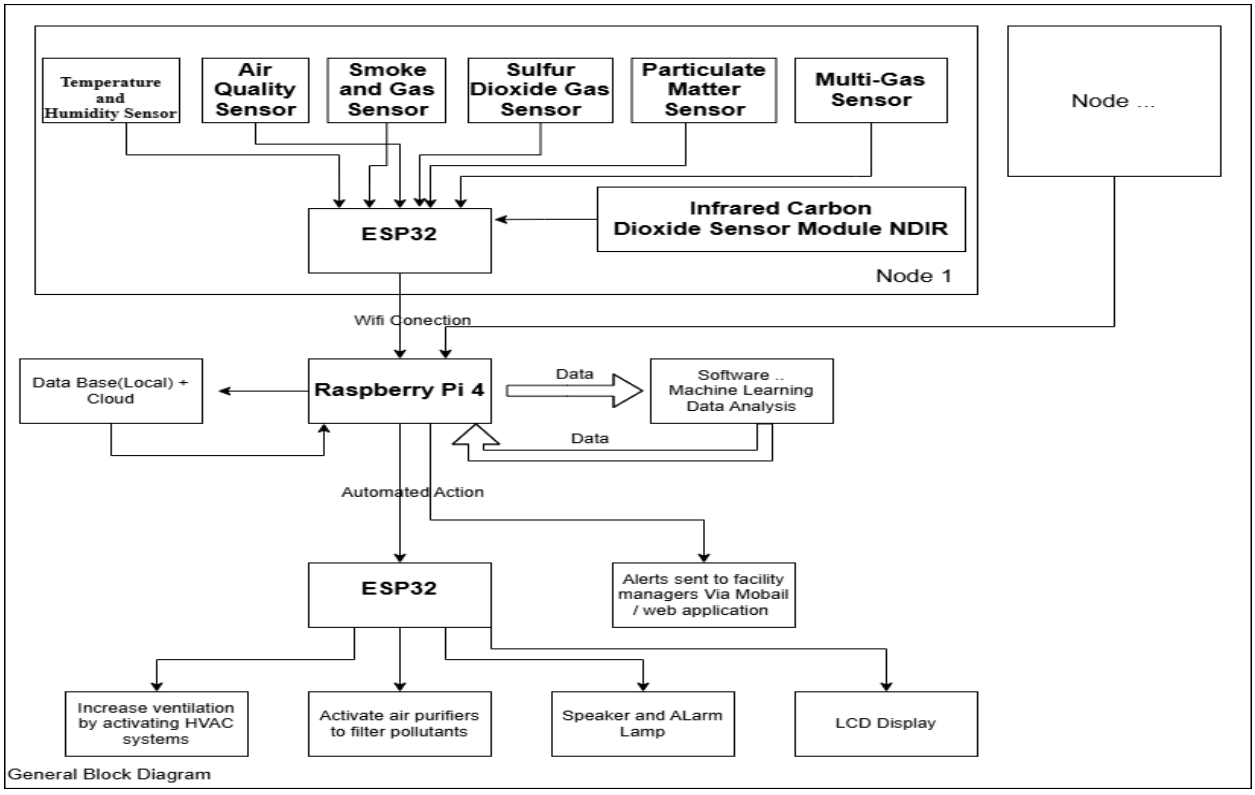


Figure 3. 17: General block diagram

3.5 Algorithms and methodologies

This section presents the algorithms and methodologies employed in the design and operation of the smart air quality monitoring system. It explains the machine learning models used for pollution prediction, data processing techniques, and the automated decision-making strategies. Each

approach is selected to ensure real-time performance, accuracy, and reliability in managing air quality in industrial environments.

3.5.1 Flowchart

The Figure 3.18 represents a flowchart for an air quality monitoring process using sensors and a central microcomputer (like a Raspberry Pi). The process starts by reading data from sensors, then sending it to the central system for analysis. Based on the analysis, pollution levels are classified as Safe, Moderate, High, or Critical. Each pollution level triggers specific actions, such as sending alerts, activating automated responses, or notifying environmental authorities. If the data doesn't match any category, it is labeled as a "Wrong Process."

3.5.2 sequence diagrams

The Figure 3.19 represents a sequence diagram for an air quality monitoring system (SAQM). It shows the interaction between different components: the manager, input ESP32, Raspberry Pi 4, output ESP32, and database/cloud. The process starts when the input ESP32 sends a sensor reading to the Raspberry Pi, which analyzes the data and sends a decision flag to the output ESP32. The output ESP32 then sends the processed data to the database or cloud, receives a response, and notifications are sent back to the manager. This sequence ensures real-time monitoring and response.

3.5.3 Activity Diagram

The Figure 3.20 represents an activity diagram for the SAQM system. It starts by reading air quality data from various sensors, which the ESP32 sends to a Raspberry Pi via Wi-Fi. The Raspberry Pi stores the data locally and, in the cloud, then uses machine learning to analyze it. Based on the pollution level, different actions are taken: if the air is safe, the system waits; if pollution is moderate, alerts are sent to managers; if high, mitigation systems are activated; and if critical, emergency alarms are triggered and authorities are notified. ESP32 devices then perform actions like activating HVAC systems, air purifiers, and updating displays.

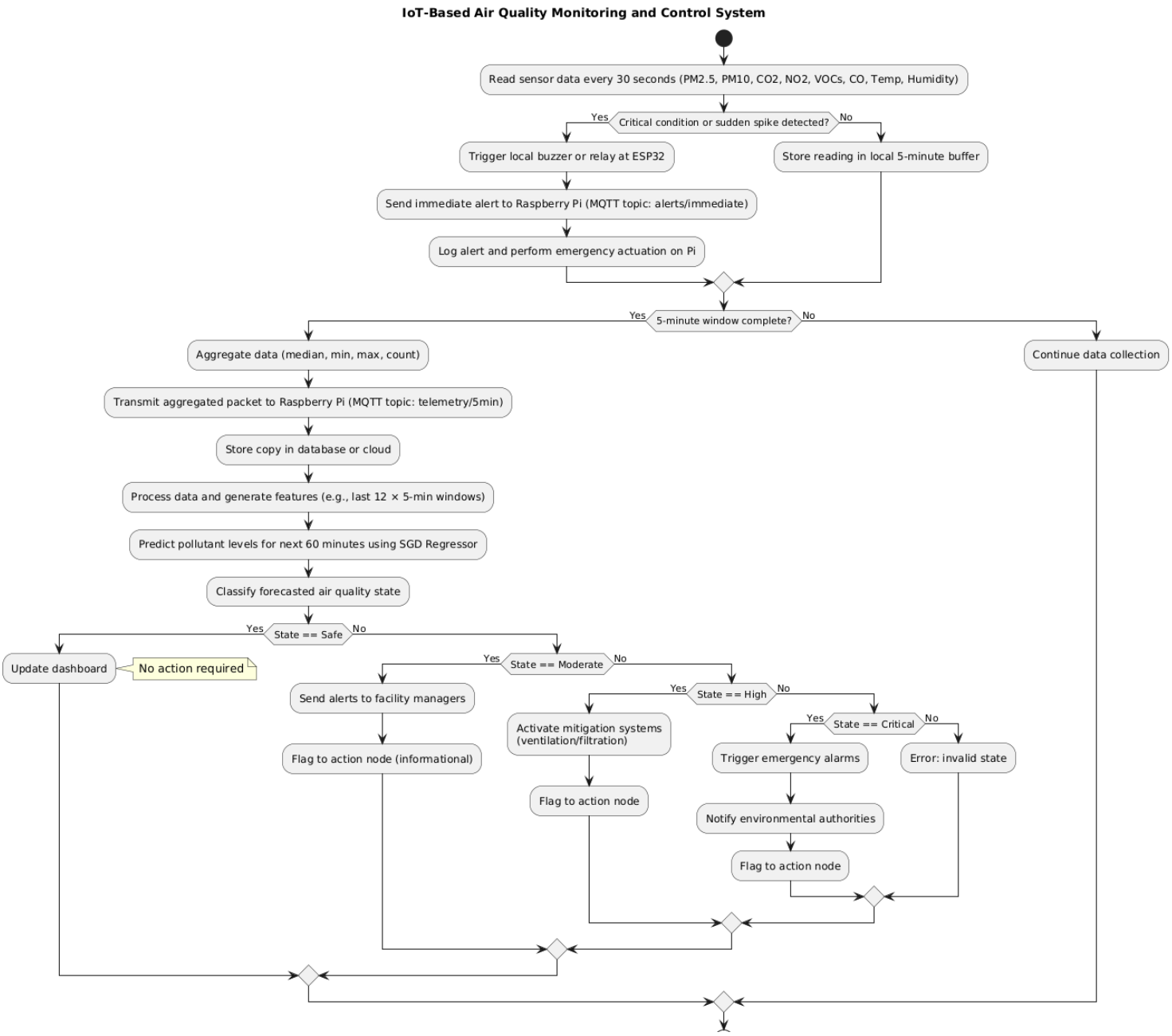


Figure 3. 18:Flow Chart Diagram

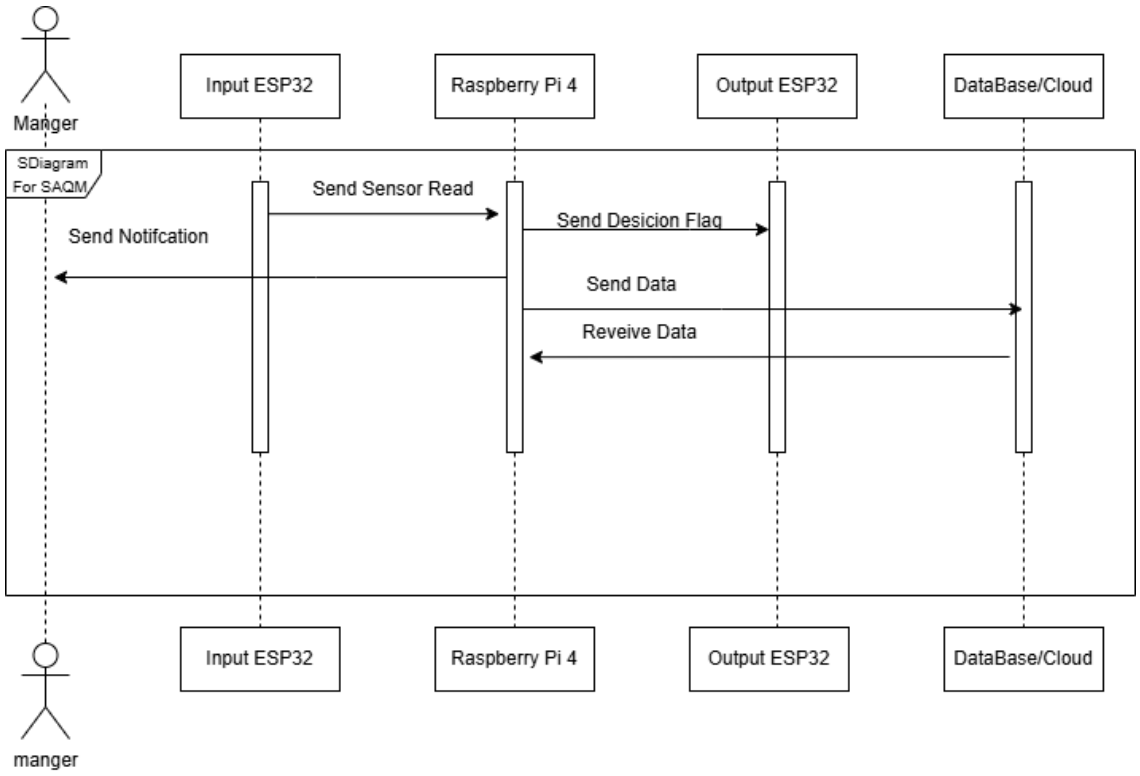


Figure 3. 19: Sequence Diagram

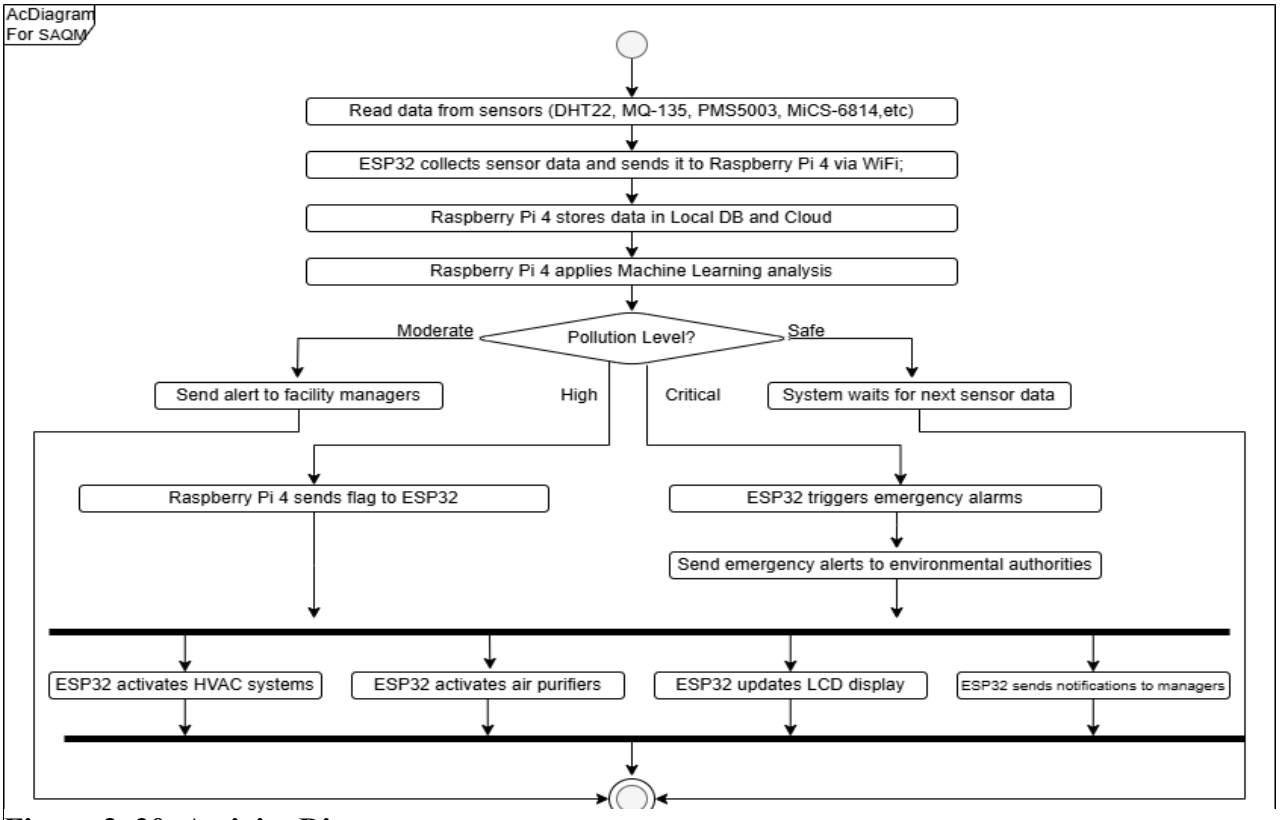


Figure 3. 20: Activity Diagram

3.6 Schematic diagrams

3.6.1 Central Schematic diagrams

Figure 3.21 is the system's central processing unit, Raspberry Pi. The Raspberry Pi takes environment data acquired by the ESP32 sensor nodes through Wi-Fi. The data is processed and predicted using the AI module and sent as corresponding control signals to the ESP32 controllers through Wi-Fi for actuation. Different microcontrollers have been utilized in this implementation for isolating the control, data acquisition, and processing functionalities. ESP32 controllers are utilized for periodic sensing (30 seconds), local aggregation of reading (every five minutes). However, when a threshold level is reached particularly during a critical pollution event the ESP32 sensing node immediately transmits the data to the Raspberry Pi, which performs decision-making and sends actuation commands to the ESP32 action node. The Raspberry Pi, however, undertakes higher-level tasks like data aggregation, machine learning–driven forecasting, decision-making, and communication with the mobile dashboard and alert system. This distributed architecture improves system scalability, reliability, and efficiency. It ensures critical actions are still performed locally on each ESP32 even when the central processor (Raspberry Pi) is suffering from network latency or processing burden, but allows the Raspberry Pi to focus on more complex analysis and forecast tasks that require additional computational power.

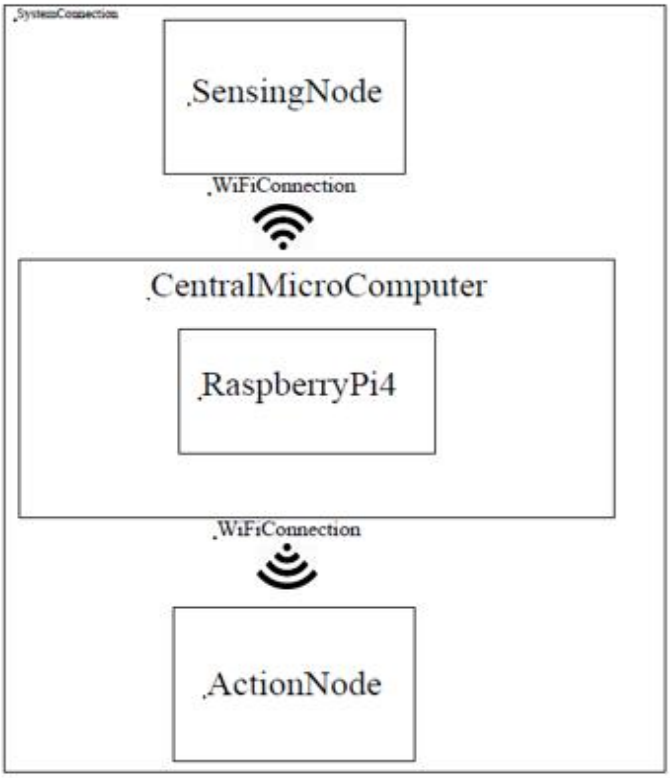


Figure 3. 21: Central Schematic diagrams

3.6.2 Sensing Node Schematic diagrams

The Figure 3. 22 shows the circuit schematic for the sensing node in the SAQM system. It connects various sensors (like DHT22, MiCS-6814, PMS5003, MH-Z19c, MQ-135A, and MQ-2) to an ESP32-C3-WROOM-02-N4 microcontroller. The sensors only collect and send raw data to the ESP32 without performing any data processing or decision-making. The ESP32 then transmits the collected data via Wi-Fi to a central system (Raspberry Pi) where the analysis and decision-making happen. In critical conditions, the ESP32 node bypasses the periodic schedule and immediately pushes the data to ensure fast reaction by the central controller.

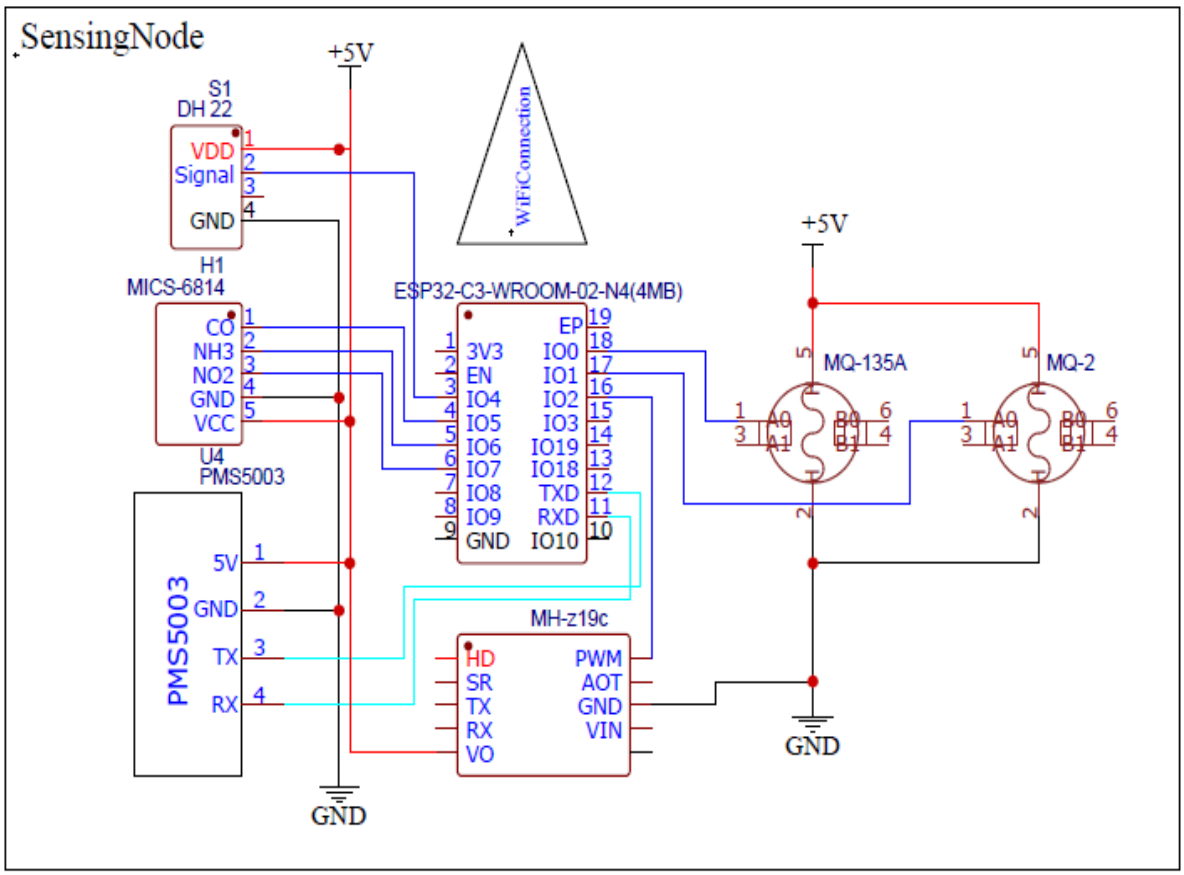


Figure 3. 22: Sensing Node Schematic diagrams

3.6.2 Action Node Schematic diagrams

The Figure 3. 23 shows ESP 32 the controller responsible for implementation, as it takes the data processed completely from the central processing unit (Raspberry Pi), as this controller controls the LCD screen and the alarm speaker with the alarm light. It also controls the HAVCS devices and controls the pollution filter, as it needs a relay, and this relay is sent to a contactor to control the HAVCS and pollution filters.

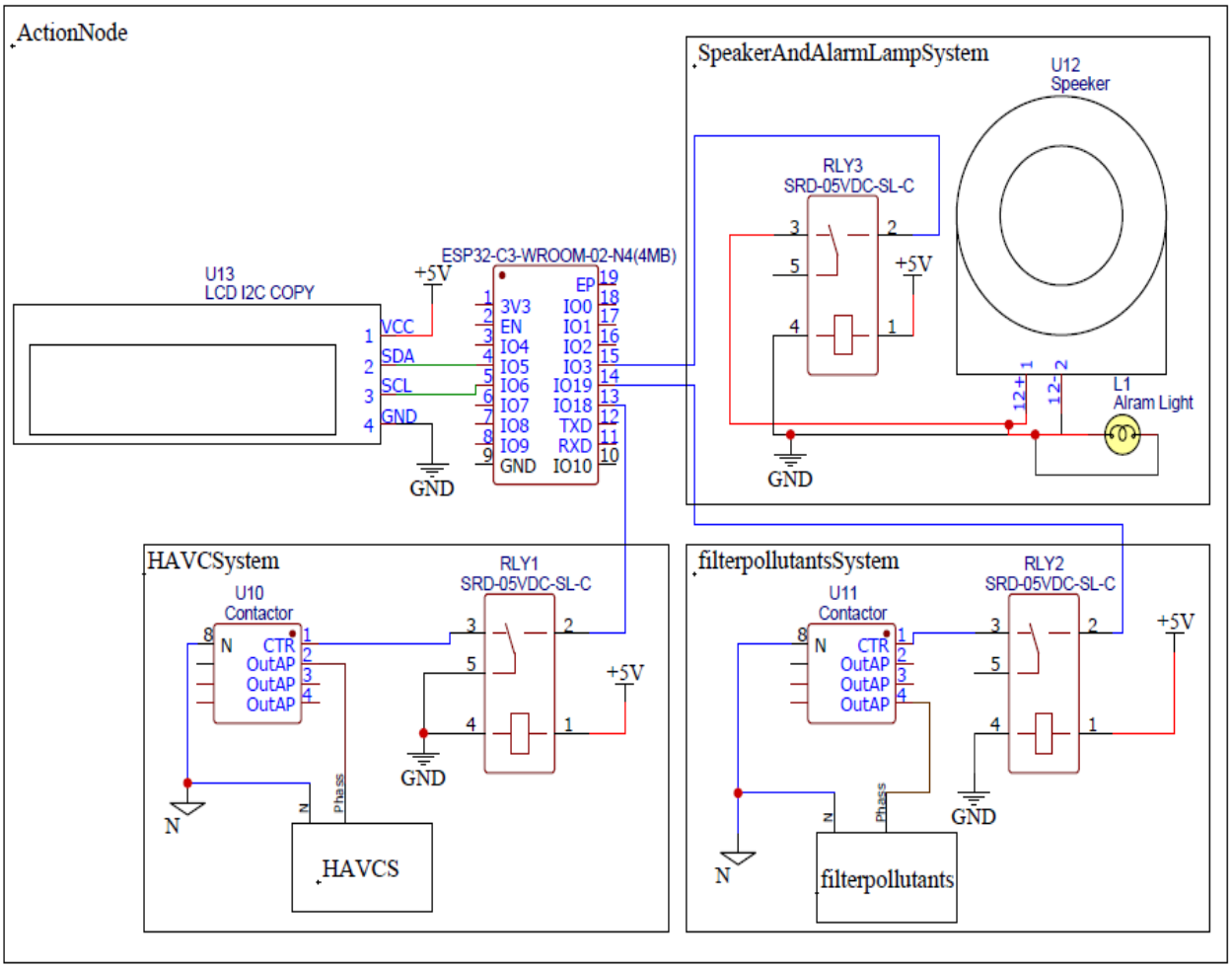


Figure 3. 23: Action Node Schematic diagrams

3.6 Summary

In this chapter, we discussed the overall design of the project, where we explained the hardware components needed to build the project, compared these components with other available options, and selected the best ones that suited our requirements. We then discussed and explained the software techniques needed. After that, we presented the algorithms and the complete structure of the project using the Block diagram and the conceptual diagram. We also illustrated the interaction between different parts of the project using the Sequence Diagram. Finally, we constructed the project using the Schematic method, demonstrating how all parts are connected to the controllers, how the non-controller parts are interconnected, and how the actuators, such as pollution filters, are connected to their designated controllers.

4 Chapter 4: Implantation and Testing

4.1 Preface

This chapter presents the practical implementation of the Smart and Automated Air Quality Monitoring System for Industrial Facilities Using Machine Learning. It describes the hardware integration, software implementation, communication architecture, and the newly developed Flutter mobile application used for remote monitoring and control. The chapter also outlines the machine learning deployment and details the testing procedures used to validate the performance, stability, and responsiveness of the entire system.

4.2 Hardware Implementation

The final prototype integrates sensing, processing, actuation, and cloud-connected monitoring components into a unified architecture. The system consists of three main hardware units: the Raspberry Pi 4 as the central controller, the ESP32 sensing node for environmental data collection, and the ESP32 action node for physical responses such as ventilation and alarms. Together, these components form a distributed air-quality monitoring platform capable of real-time detection and automated intervention.

- **Central Processing Unit: Raspberry Pi 4**

The Raspberry Pi 4 serves as the core processing unit of the system. It receives continuous sensor readings from the sensing node, processes them using Python-based software, and executes the trained SGDRegressor machine learning model to predict air-quality levels. When hazardous conditions are detected, the Pi sends activation commands to the ESP32 action node to trigger alarms or ventilation. Additionally, the Raspberry Pi functions as the backend server for the Flutter mobile application, providing live data streaming, historical storage, and communication endpoints. Its multi-core processor enables fast inference and ensures that alert mechanisms operate with minimal delay.

- **ESP32 Sensing Node**

The sensing node is built using an ESP32 microcontroller connected to several air-quality and environmental sensors. These include the MQ-2, MQ-7, and MQ-135 gas sensors, the PMS particulate matter sensor, and the DHT22 temperature and humidity sensor. The ESP32 collects sensor readings, applies basic preprocessing, and sends the data wirelessly to the Raspberry Pi using a lightweight protocol. The node operates continuously and supports quick recovery if Wi-Fi connectivity is interrupted. The use of the DHT22 improves gas-sensor stability by providing humidity compensation. as illustrated in Figure 4. 1

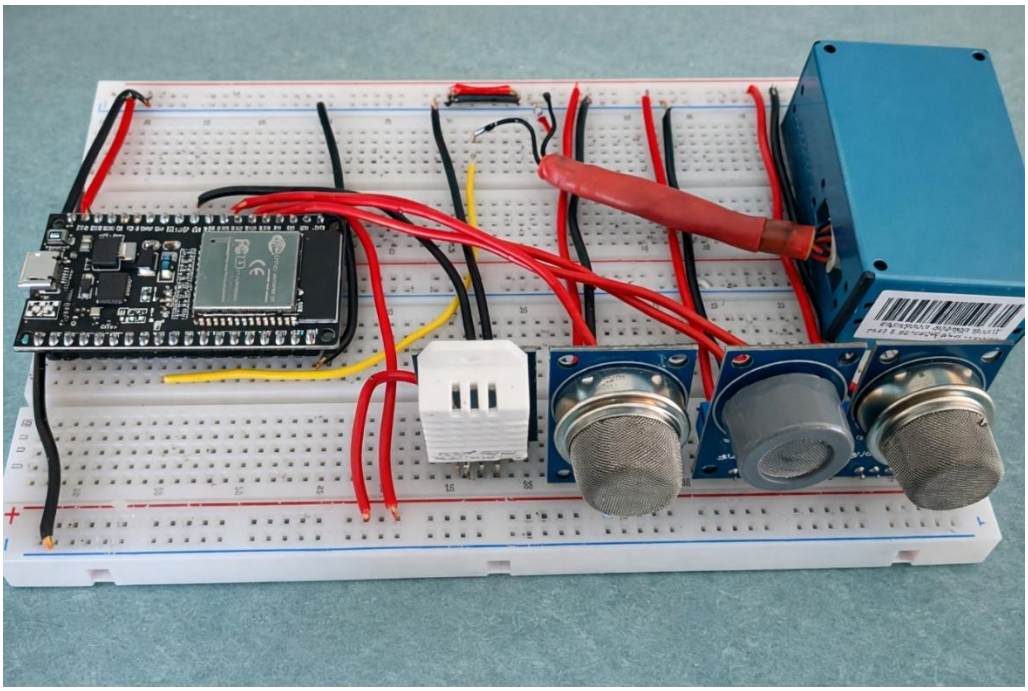


Figure 4. 1: sensing node

- **ESP32 Action Node**

The action node consists of an ESP32 connected to a relay module, cooling fan, buzzer, LED indicators, and an OLED display. It executes commands received from the Raspberry Pi, enabling fully automated or remotely controlled responses. In Auto Mode, the node reacts solely based on instructions generated by the Raspberry Pi. In Manual Mode, the Flutter app allows the user to directly control the fan, ventilation system, and alarm features. The node also includes a large green button, which represents a prototype of an air filtration unit; since a physical filter was not available during development, the button was used to simulate the presence and activation of the filter mechanism. The structured wiring, relay isolation, and OLED status messages ensure safe and reliable operation. as illustrated in Figure 4.2

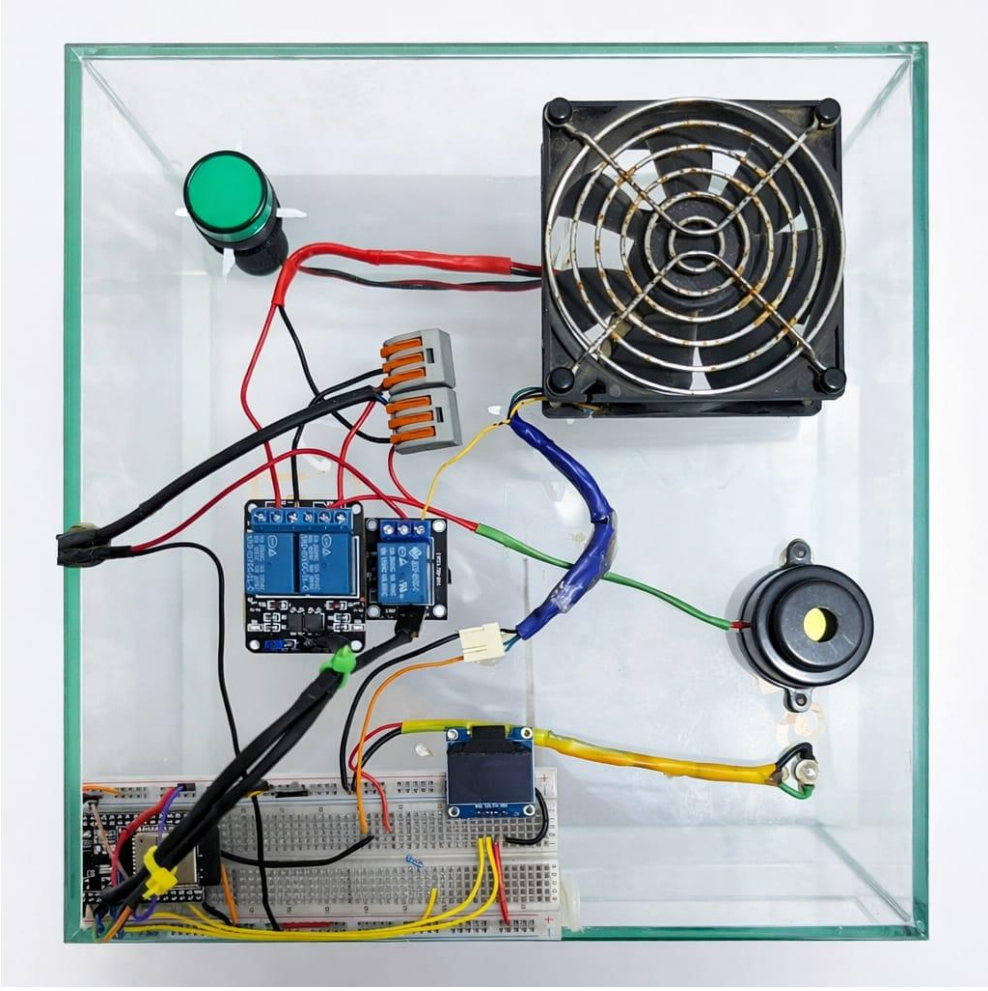


Figure 4. 2: action node

4.3 Software Implementation

4.3.1 Central Node Software

This section reviews the Raspberry Pi's central node implementation program, focusing on data exchange, real-time artificial intelligence, and data requirements management. The central node's functionalities were previously covered in Section 3.6.1.

- **Data Sending and Receiving:** The central node (Raspberry Pi) acts as the main communication point in the system, configuring and running a Flask-based web server. This server provides a REST API for data exchange between the central node and the sensor and execution nodes. This communication relies

```
(venv) pi4@raspberrypi:~/esp_project $
Started AI trainer: /home/pi4/esp_projct/aitrainer_mean.py
* Serving Flask app 'server7'
* Debug mode: off
WARNING: This iss a development server. Do not use it in a
production deployment.
* Running on all addresse (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.0.124 : 5000
```

Figure 4. 3: Flask server

on the HTTP protocol with JSON formatting for data transfer. This sup-section illustrates the mechanism for receiving data from the sensor nodes and sending control commands to the execution nodes, as shown in Figure 4. 3

1. **Receiving data from the sensor node**: The sensor nodes periodically send readings of environmental pollutants to the central node via POST HTTP requests , as shown in Figure 4. 4

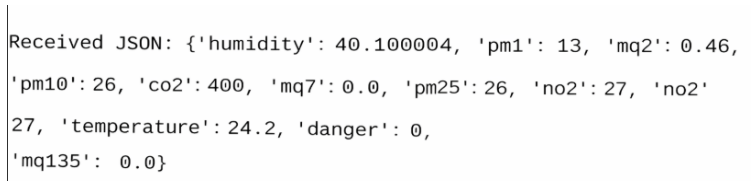


Figure 4. 4: receiving data from sensors

2. **Sending data and control commands to the execution node** :The central node, through the Flask server, sends control commands resulting from the decision logic to the execution nodes via HTTP requests. These commands include fan speed, filter status, alarm status, and the overall status of pollutants and operating mode, as shown in Figure 4. 5

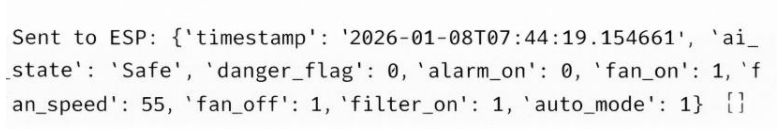


Figure 4. 5: sending data to action node

- **Real-Time Artificial Intelligence Processing and Training**: The central node includes a real-time AI module responsible for analyzing incoming environmental data and assessing air quality. Sensor data is passed directly to the AI model as soon as it is received, allowing the model to infer the current state of the system. The model supports incremental learning, enabling it to be updated gradually without requiring a complete retraining process. This allows the system to adapt to continuous environmental changes. Details of the AI model, its training mechanism, and performance evaluation are explained in more detail in Section 4.5.3.

- **Firebase and Database Management**: Data management at the central node relies on a hybrid architecture that combines local storage with a real-time cloud database.

1. **Local Database**: A local MySQL database is used to store historical sensor readings accompanied by a timestamp, ensuring that the data is saved and can be accessed later, in addition to the system continuing to operate even if the network connection is interrupted, as shown in Figure 4. 6

ID	Timestamp	mq2	mq7	mq135	pm1	pm25	co2	tempeat	humidity
34018	2026-01-08 07:51:34	0.47	0	0	11	22	400	24.5	39.4
34017	2026-01-08 07:51:26	0.48	0	0	12	23	400	24.5	39.4
34016	2026-01-08 07:51:19	0.48	0	0	12	28	400	24.5	39.4
34015	2026-01-08 07:51:11	0.48	0.1	0	12	27	400	24.5	39.4
34014	2026-01-08 07:51:03	0.47	0	0	10	28	400	24.5	39.4

Figure 4. 6: MySQL database

2. **Cloud (Firebase)**: The system's cloud architecture relies on the use of two different Firebase services, with tasks divided between Firebase Realtime Database and Firebase Firestore according to the nature of the data and system requirements. as shown in Figure 4. 7

```
▲Firebase reading pushed: 200 {"name":"-OiThMaVRAvstuIV__06V"}

▲Firebase factory state updated: 200 {"co2":400,"pm25":26,"timestamp":"2026-
  01-08T07:44:12.048178","pm1":14,"mq2":0.47,"danger_flag":0,"danger":0}

▲Firestore upload successful
```

Figure 4. 7: Firebase upload data

- Firebase Realtime Database is used to store and manage the real-time state of the system, such as air quality status, as shown in ***Figure 4. 8***

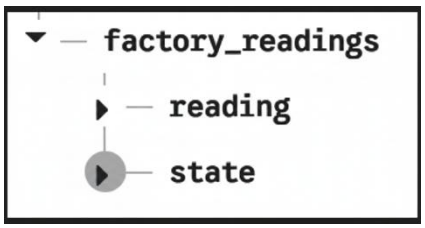


Figure 4. 8: firebase state

- Firebase Firestore stores historical data of environmental readings received from sensor nodes. This data includes measured values accompanied by a timestamp, as shown in Figure 4. 9

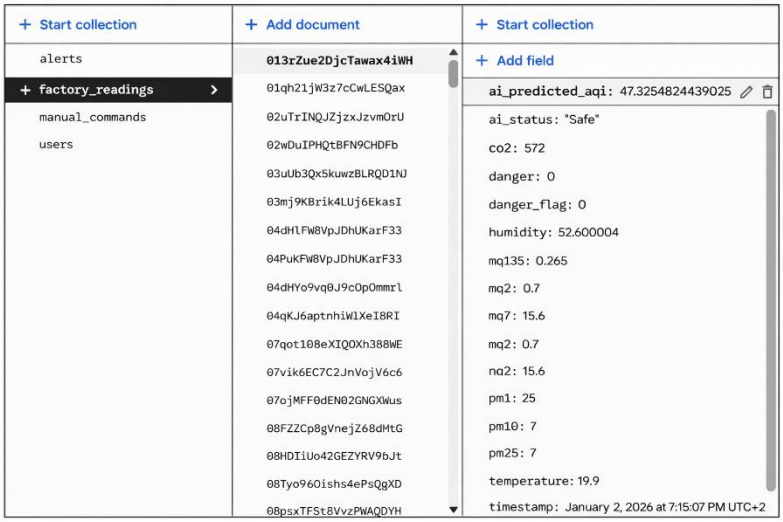


Figure 4. 9: firebase historical data

4.3.2 ESP32 Software Implementation

The ESP32 module is responsible for collecting sensor data and controlling actuators. Its role is to read data in real time, send it to the server for artificial intelligence analysis, and execute control commands received from the server. This section explains software implementation on the ESP32 without re-explaining the algorithms themselves.

- Sensing Node Esp: The ESP32 reads a range of environmental variables, such as PM2.5, CO_2, VOC, NO_2, and CO_2, at specified time intervals. The raw data is simply processed on the ESP32 to ensure transmission accuracy before being sent to the server.
  - Estimating the CO2: Since a dedicated NDIR sensor for measuring CO_2 was unavailable, an estimated method was adopted using the MQ-135 sensor, which shows a notable response to volatile organic gases, including changes associated with CO_2 concentration. As the equation shown in Figure 4. 10

```
# ---------- CO2 synthetic from MQ-135 ----------
co2 = 400 + mq135_norm * 2600  # 400–3000 ppm
```

Figure 4. 10: CO2 equation from mq-135

- Estimating the NO2: The concentration of NO_2 gas was estimated using a hybrid synthetic model that incorporates several environmental indicators instead of a direct NO_2 sensor, in order to simulate the actual behavior of the gas in industrial environments.

- Check Danger flag :The Danger Flag indicator in this system is designed to function as an early warning system based on a comprehensive, real-time assessment of multiple pollutants and environmental parameters. This indicator aims to simplify the air quality situation and translate it into a logical (safe/dangerous) decision, as illustrated in Figure 4. 11 enabling a faster response in emergencies where prediction is not required.

```
# ---------- Danger flag ----------
danger_flag = 0
if pm25 is not None and pm25 >= 55.5:
    danger_flag = 1
if pm10 is not None and pm10 >= 254:
    danger_flag = 1
if co2 >= 2000:
    danger_flag = 1
if no2_ppb >= 200:
    danger_flag = 1
if vocs_ppb > 3:
    danger_flag = 1
if co_ppm > 200:
    danger_flag = 1
if temp is not None and temp >= 31:
    danger_flag = 1
if hum is not None and (hum <= 19 or hum >= 71):
    danger_flag = 1
if mq2_index > 3:
    danger_flag = 1
```

Figure 4. 11: danger flag values

- Action Node Esp: The Action Node operates in two modes: Auto Mode, where the fan, filter, and alarm are controlled automatically based on the AI status and Danger Flag, and Manual Mode, where operating commands are executed directly by the user without the intervention of the intelligent system. When the system is powered on, the initial state is to turn off all triggers (Fan, Filter, Alarm) until the first control commands are received from the server.as Shown in Figure 4. 12

```
# ===== منطق Auto / Manual =====
if auto_mode:
    # ===== Auto Mode =====
else:
    # ===== Manual Mode =====
```

Figure 4. 12: action node modes

4.3.3 Mobile Application

The mobile application was developed using Flutter, adopting a provider-based state management architecture to ensure scalability, maintainability, and clear separation of concerns. The application serves as the primary user interface for monitoring industrial air quality, receiving alerts, controlling mitigation devices, and generating analytical reports.

- The app integrates with Firebase services, including:

  - Firebase Authentication for secure user access.
  - Firebase Realtime Database for real-time sensor readings and control commands.
  - Cloud Firestore for historical data, reports, alerts, and audit logs.

- Project Structure and Architectural Design: The lib folder contains all the project's source files, organized according to their main functions: core for core functions, models for data models, providers for data and services, screens for screens and interfaces, services for auxiliary services, widgets for reusable elements, as well as the firebase_options.dart

configuration file and the start file main.dart. This organization enhances code clarity, ease of maintenance, and reusability within the application.as showing in Figure 4. 13

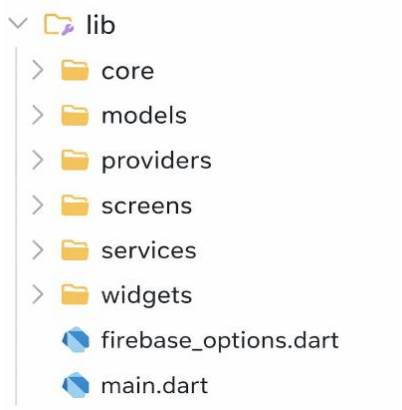


Figure 4. 13: Flutter file structure

- The core/ directory defines global application logic and constants
  - Alert Threshold Evaluation: The file alert_thresholds. dart implements all pollutant classification logic using threshold-based evaluation methods. Each pollutant is mapped to four severity levels: safe, moderate, high, and critical. This logic is reused consistently across: Alert generation, Live AirQulityIndex calculation, and Notification triggering, as showing in Figure 4. 14

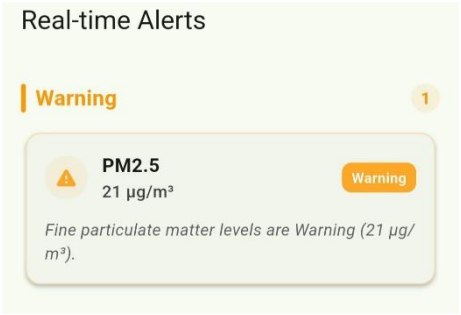


Figure 4. 14: Real-time alert

- Data Models Implementation: The models/ directory encapsulates all domain entities exchanged between Firebase, providers, and UI components
  - Live Sensor Reading Model the LiveReading model represents a complete snapshot of environmental conditions received from the Realtime Database, as shown in Figure 4. 15
  - It includes:
    - Raw sensor values (PM2.5, PM10, CO_2, VOCs, NO_2, CO, temperature, humidity)
    - AI-generated prediction fields
    - Calculated AQI and aggregated air quality status
- State Management Using Providers: The application uses Provider as the state management solution, each provider

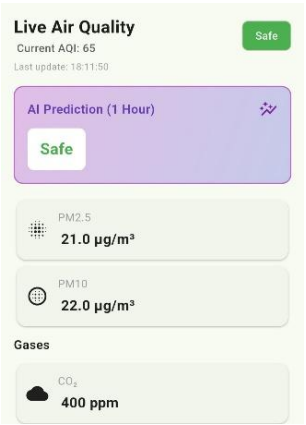


Figure 4. 15: live air quality

controls a single functional domain, which improves modularity.

- Live Data Provider: LiveDataProvider listens continuously to factory_readings/state in Firebase Realtime Database.Its responsibilities include: parsing incoming sensor data, triggering notifications for High and Critical conditions, computing AQI updates in real time
- Control Provider
  - ControlProvider manages all actuator-related commands, including: automatic/manual mode switching, fan power and speed, air purifier state, and alarm activation, as showing in Figure 4. 16
  - Every control action updates Realtime Database (for hardware devices), and logs the command in Firestore.

Figure 4. 16: Operation mode

- Authentication and User Management: Authentication is implemented using Firebase Authentication (Email & Password).
  - The AuthProvider: Tracks authentication state changes,Loads user profiles from Firestore,Enforces email verification
  - User roles (Admin, Operator) are stored in Firestore and can be extended later for role-based access control.
  - This is illustrated in the following Figure 4. 17

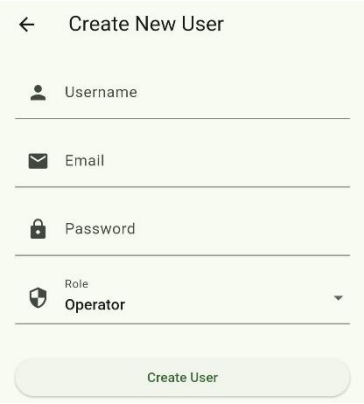


Figure 4. 17: user create

- Alerts and Notification System : The alert system is fully reactive and derived from live sensor readings. AlertsProvider generates pollutant-specific alerts dynamically,NotificationService send local notifications for real-time alerts and global AQI notifications.

- Historical Data Visualization: The HistoryProvider retrieves historical readings from Firestore based on:
    - Selected pollutant
    - Selected time range (Last Hour, 24h, Week, Month)
    - This is illustrated in the following Figure 4. 18

Figure 4. 18: historical data

- Report Generation Module: The reporting subsystem includes: Historical pollutant statistics, and Critical and high alerts, and Manual control commands, ReportProvider produces a structured ReportData object, which is then exported using ReportPdfService. The generated report includes: Environmental summary, Alerts timeline, User intervention log, and Generation timestamp,as showing in Figure 4. 19

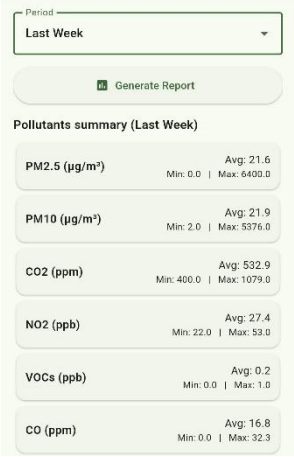


Figure 4. 19: report generation

- User Interface Implementation: Each screen in the screens/ directory corresponds to a functional module
    - Dashboard: live monitoring
    - History: trend analysis
    - Alerts: active alerts
    - Levels: air quality classifications
    - Control: actuator management
    - Reports: PDF generation
    - Settings: preferences and notifications
    - This is illustrated in the following Figure 4. 20

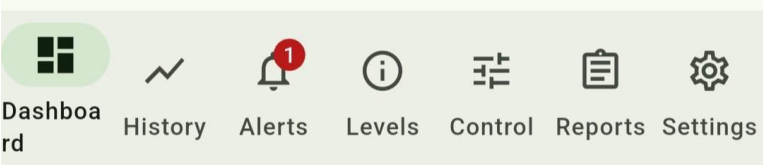


Figure 4. 20: Navigation Bar

4.3.4 Machine Learning Model Deployment

The AI was implemented within a distributed architecture, where the AI model does not run directly on the microcontroller's control bus. Instead, a central processing unit is created and runs on the Raspberry Pi. This approach allows for more challenging machine learning models while reducing the resource consumption of the ESP32 module. Figure 4. 22 illustrates the complete system response and the data flow from sensor acquisition to AI-based predictions and actuator control.

- **Model Deployment:** After training the AI model outside the operating environment on data set showing in Figure AP II. 1, the model was saved as a (.pkl) file using the (Joblib) library. When the server starts, the model and scaler are loaded into memory simultaneously to ensure fast response times during actual operation.
- **Real-Time Data Flow:** The received values are passed from the server to the AI unit directly without prior storage during the passing process.
- **AI Execution Pipeline :**Upon data arrival at the server, the following steps are performed:
  - Data conversion to a numerical matrix.
  - Scaling operation applied using the same parameters used during training.
  - Passing the data to the AI model for prediction.
  - Extracting the prediction output (AQI and the state's classification value).

  This process is executed within fractions of a second, as shown in Figure 4. 21, thereby meeting real-time operational requirements

Figure 4. 21: AQI Result

- **Continuous Learning:** To ensure that the continuous learning process does not lead to a decline in system performance or loss of prior knowledge, a set of safeguards was adopted during the AI model update:
  - **Controlled Incremental Updates:** The model is not updated directly with each new reading, but rather new data is periodically collected within specific time windows, which reduces the impact of outliers and noise.
  - **Balanced Training Data :**When the update is performed, the new data is combined with selected samples of historical data, in order to preserve previous model knowledge and prevent the phenomenon of Catastrophic Forgetting.
  - **Performance Monitoring** :After each update, the model's performance is evaluated using key performance indicators (KPIs) (such as prediction accuracy), and if a decline in performance is observed, the update is ignored and the previous version of the model is retained.

- **Versioning and Rollback Mechanism** :Each copy of the model is saved with a different version number, allowing a return to a previous stable model in case of any unexpected performance degradation.

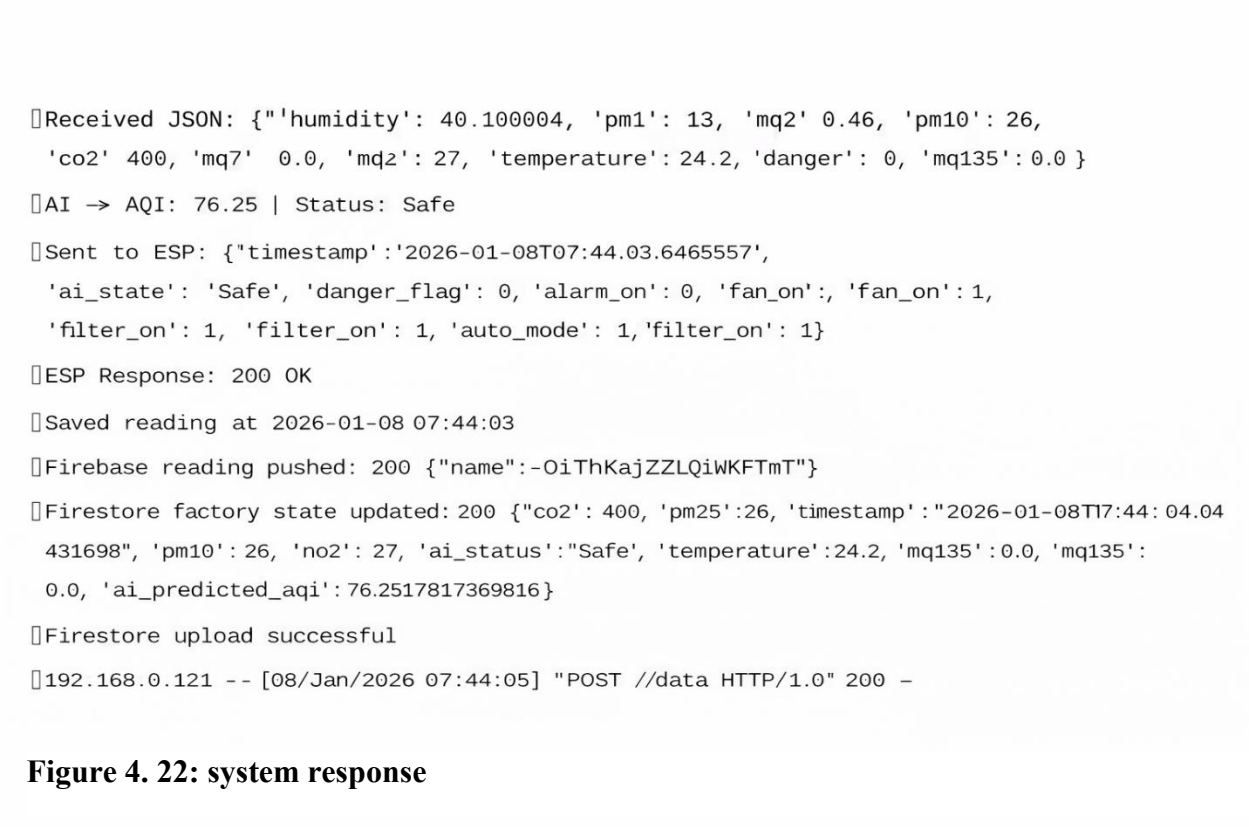


Figure 4. 22: system response

4.4 Validation and Testing

Validation and testing were conducted to ensure that the proposed smart air quality monitoring system operates correctly, reliably, and in accordance with the specified functional and non-functional requirements. Testing focuses on verifying the correct operation of individual system components, while validation ensures that the collected data, communication processes, decision-making mechanisms, and system actions meet real-world industrial requirements. The evaluation process was carried out at multiple levels, including individual nodes, integrated subsystems, and the complete system.

4.4.1 Testing

Testing was performed to evaluate the functionality, stability, and responsiveness of the system components under different operating conditions. The testing methodology was divided into three main stages as follows:

A. Sensing Node Testing

Each sensing node was tested independently to ensure correct sensor readings and stable

operation. This included verifying sensor initialization, accuracy of pollutant measurements, periodic data acquisition, and correct data transmission to the central unit via Wi-Fi. Sensor behavior was observed under normal and elevated pollution levels to confirm reliable performance.

B. Action Node Testing
The action node was tested to validate its ability to receive commands from the central controller and execute corresponding actions. This included testing relays, ventilation fans, air purifiers, alarms, LEDs, and display outputs. Manual and automatic command execution were both verified to ensure correct system response.

C. Whole System Testing
In this phase, all system components were connected and tested together as an integrated system. End-to-end testing was performed to ensure seamless interaction between sensing nodes, the central processing unit, action nodes, cloud services, and the mobile application. This stage confirmed real-time monitoring, automated decision-making, and synchronized system responses.

4.4.2 Validation

Validation was carried out to confirm that the system meets design objectives and performs effectively in real operational scenarios. The validation process included the following aspects:

A. Sensor Readings Validation
Sensor outputs were compared against expected environmental ranges and reference values to verify accuracy and consistency over time.

B. Communication Validation
The reliability of data transmission between sensing nodes, the central unit, cloud services, and the mobile application was validated, ensuring minimal data loss and acceptable latency.

C. AI Module Validation
The machine learning module was validated by evaluating prediction accuracy, model responsiveness, and consistency between predicted and actual pollution levels.

D. Storage, Database, and Cloud Validation
Data storage mechanisms were validated to ensure secure, accurate, and continuous logging of sensor data, system actions, and alerts both locally and on the cloud.

E. Action Elements Validation
Automated responses such as fan activation, air purification, alarms, and notifications were validated to ensure they are triggered correctly based on pollution thresholds and AI predictions.

F. Mobile Application Monitoring and Control Validation
The mobile application was validated for real-time data visualization, alert reception, and remote-control functionality, ensuring usability and reliable system interaction for end users.

4.5 Summary

This chapter described the complete hardware and software implementation of the system, including the mobile application, communication framework, and machine learning components. The testing procedures verified that the system performs reliably under various environmental conditions and meets all functional requirements.

5 Chapter 5: Result and Discussion

5.1 Preface

This chapter presents the results obtained from testing the complete prototype. It includes analysis of sensor data, system response, machine learning predictions, and mobile-application performance. The chapter also discusses the reliability, accuracy, and usability of the system.

5.2 Sensor Data Results

The sensors provided clear and consistent data patterns. The MQ-7 sensor produced noticeable increases when exposed to carbon monoxide sources, while the MQ-2 sensor responded strongly to smoke, LPG, and other flammable substances. The MQ-135 sensor detected the presence of organic pollutants when tested near industrial cleaning chemicals. The PMS sensor delivered detailed particulate matter readings and showed rapid change during dusty conditions. The DHT22 sensor maintained stable temperature and humidity values, enabling improved gas-sensor compensation. All sensors produced repeatable results during multiple testing sessions. As shown in Figure 4. 4

5.3 Machine Learning Model Results

The experimental evaluation confirmed that the deployed SGDRegressor model performed reliably during real-time operation. The model consistently predicted air-quality levels from live sensor inputs with stable behavior under continuous system operation. The generated predictions aligned well with expected environmental patterns and successfully captured long-term pollution trends, while remaining resistant to sudden fluctuations caused by noisy or invalid sensor data. The model demonstrated good generalization capability and operated with negligible latency, making it suitable for real-time industrial monitoring. Although explicit numerical accuracy metrics were not calculated, practical testing showed that the prediction performance was sufficiently accurate for industrial decision-making, while maintaining high system stability and dependable operation over extended periods.as shown in Figure 5. 1

Date & Time	Status	Current MAE	New MAE	Best MAE	Improvement Ratio
2025-12-08 12:03:27	STATUS-IMPROVED	1.6551486139750278	0.41325471814711	0.413254718141	+0.7503230423073717
2025-12-08 13:43:28	STATUS-NO IMPROVEMENT REVERTED	0.396397329599988	0.79136908829115	0.413254718141	-0.9147918295486
2025-12-08 15:23:19	STATUS-NO IMPROVEMENT REVERTED	0.3954920231061167	1.58344666807643	0.413254718141	-2.831422136440934
2025-12-08 17:09:33	STATUS-IMPROVED	0.39517152676343715	0.395771930793075	0.3957719307935	+0.042297589816066
2025-12-08 20:43:30	STATUS-NO IMPROVEMENT REVERTED	0.3953397211462726	0.35584907809962	0.3957719307935	-0.0005930738074111
2025-12-08 21:33:21	STATUS-IMPROVED	0.3953397211462763	0.35584907809962	0.3957719307935	-0.0005930758074111
2025-12-08 21:33:21	STATUS-NO IMPROVEMENT REVERTED	0.38549547195172017	0.385498028099963	0.3854980280916	-0.025900955491154
2025-12-10 01:08:37	STATUS-IMPROVED	0.7515792352616252	5.19413694502564	0.751579235261	-5.91096385929029
2025-12-10	STATUS-NO IMPROVEMENT	0.7515792352616252	5.19413694502564	0.751579235261	-5.91096385299029

Figure 5. 1:AI decision

5.4 System Response Evaluation

The SGDRegressor model performed reliably when predicting air-quality levels from live sensor inputs. Predictions matched expected environmental patterns and were stable during continuous operation. The model demonstrated good generalization and operated with negligible latency, making it suitable for real-time monitoring. Although numerical accuracy metrics were not computed, practical testing showed the model to be sufficiently accurate for industrial decision-making. As shown in Figure 4. 22

5.5 Discussion

The results demonstrate that the system successfully combines hardware sensing, machine-learning prediction, mobile application interaction, and automatic control into a reliable industrial solution. Compared to traditional gas detectors, the inclusion of machine learning provides earlier warning capability and better interpretation of environmental trends. The mobile app significantly enhances usability by enabling remote monitoring, historical analysis, and manual control. The distributed architecture improves safety and reliability by separating sensing and actuation nodes. Overall, the system meets the intended objectives and performs effectively in real-world scenarios.

5.6 Summary

This chapter analyzed the results of the implemented system, confirming that all components—including sensors, machine learning model, action controls, and mobile application—function as expected. The system is capable of real-time monitoring, instant alerting, and user interaction, demonstrating strong potential for industrial deployment.

6 Chapter 6: Conclusion and Future Work

6.1 Concluding Remarks

This project presented the design and implementation of a Smart and Automated Air Quality Monitoring System for Industrial Facilities Using Machine Learning. The system successfully combined distributed sensing, real-time data analysis, wireless communication, automated response mechanisms, and mobile application monitoring into one integrated platform. By utilizing a Raspberry Pi 4 as the central processing unit and two ESP32 nodes as sensing and action controllers, the system demonstrated efficient performance and high reliability during real-world testing scenarios.

The sensing node consistently delivered accurate readings from the MQ gas sensors, PMS particulate matter sensor, and DHT22 environmental sensor. These readings were processed by the Raspberry Pi, which used the trained SGDRegressor model to estimate air-quality levels and to determine when ventilation or alarms should be activated. The machine learning model produced stable and meaningful predictions aligned with expected environmental behavior.

The action node executed control commands rapidly, activating the fan, buzzer, and status indicators within less than a second. This immediate response confirmed the effectiveness of the communication architecture and the suitability of the system for industrial environments where rapid intervention is necessary. The system also proved stable during long-duration testing, maintaining wireless connectivity and consistent performance without errors or interruptions.

A major strength of the system lies in the Flutter-based mobile application, which provides live monitoring, warning notifications, critical alerts, interactive historical graphs, PDF report generation, and manual control functionality. The app greatly enhances usability by allowing supervisors and safety engineers to access system data remotely and take control whenever required. The inclusion of Auto Mode and Manual Mode gives the user flexibility to either rely on machine-learning-driven decisions or operate the system manually.

Overall, the project achieved its objectives and produced a functional, robust, and efficient air-quality monitoring system capable of making automated decisions and offering comprehensive insights into environmental conditions. The combination of hardware reliability, intelligent prediction models, and user-friendly software proves that such a system can significantly improve safety standards in industrial facilities.

6.2 Future Work

Although the implemented system performed effectively, several opportunities for enhancement and future development exist.

One potential improvement is the integration of cloud storage and remote servers to support long-term data retention beyond the current two-week period. Cloud connectivity would enable advanced analytics, centralized dashboarding, and multi-facility monitoring from a single platform. Additional machine learning models could be explored, such as deep-learning methods or ensemble techniques, to further improve prediction accuracy and detect more complex environmental patterns.

Hardware improvements may also include expanding the sensing capabilities by adding sensors for additional pollutants such as ozone (O_3), sulfur dioxide (SO_2), and nitrogen dioxide (NO_2). An industrial-grade enclosure could be developed to enhance durability and protect the sensors from harsh environments.

The mobile application can be extended by incorporating AI-driven alert explanations, suggesting causes of pollution spikes, or recommending ventilation strategies. Support for automatic calibration routines or maintenance reminders would also improve long-term reliability. Introducing multi-user access roles and secure authentication layers would increase safety and allow broader deployment in large industrial sites.

In future implementations, integrating LoRaWAN or 5G communication could enable the sensing node to operate over long distances or across large facilities without relying solely on Wi-Fi. A web-based dashboard could complement the mobile application and offer browser-based monitoring for control rooms and supervisory offices.

Overall, there are many possibilities for enhancing the system's intelligence, connectivity, and usability. These improvements could make the platform even more powerful and versatile, increasing its value in industrial safety applications and opening opportunities for commercial-scale deployment.

REFERENCES

[1]. World Health Organization (WHO). *Air Pollution*. https://www.who.int/health-topics/air-pollution (Accessed: 03 April 2025).

[2]. U.S. Environmental Protection Agency (EPA). *Particulate Matter (PM) Basics*. https://www.epa.gov/pm-pollution/particulate-matter-pm-basics (Accessed: 03 April 2025).

[3]. Cee R. *Carbon dioxide in workplace atmospheres*. Salt Lake, UT: OSHA Technical Center; 1990. p. 1–26.

[4]. U.S. Environmental Protection Agency (EPA). *Basic information about NO_2*. https://www.epa.gov/no2-pollution/basic-information-about-no2 (Accessed: 03 April 2025).

[5]. Wikipedia. *Sulfur dioxide*. https://en.wikipedia.org/wiki/Sulfur_dioxide (Accessed: 03 April 2025).

[6]. IBM. *What is machine learning (ML)?*. https://www.ibm.com/think/topics/machine-learning (Accessed: 04 April 2025).

[7]. Solanki D. *Supervised machine learning: A beginner's guide*. Medium. https://medium.com/@dhara732002/supervised-machine-learning-a-beginners-guide-9ac0b07eccbb (Accessed: 04 April 2025).

[8]. U.S. Department of Homeland Security. *Alert notification systems: Homeland security*. https://www.dhs.gov/publication/alert-notification-systems (Accessed: 04 April 2025).

[9].Amor, M., Arar, H. and Almkhamrah, A. (2008) *Developing a method of treat air pollution and monitoring air quality, DSpace Home*. Available at: https://scholar.ppu.edu/handle/123456789/7015 (Accessed: 05 April 2025).

[10]. Aqib, M. (2016) *IOT based air pollution monitoring system using Arduino, IoT Based Air Pollution Monitoring System using Arduino & MQ135 Sensor*. Available at: https://circuitdigest.com/microcontroller-projects/iot-air-pollution-monitoring-using-arduino (Accessed: 10 December 2016).

[11].Wendy, Y. *et al.* (2021) *Sensor-based machine learning approach for indoor air quality monitoring in an automobile manufacturing, MDPI*. Available at: https://www.mdpi.com/1996-1073/14/21/7271 (Accessed: 05 April 2025).

[12] Systems, E. (no date) *ESP32*, *ESP32 Resources | Espressif Systems*. Available at: https://www.espressif.com/en/products/socs/esp32/resources (Accessed: 26 April 2025).

[13] Raspberry Pi Foundation, "Raspberry Pi 4 Model B Datasheet," accessed on Apr. 26, 2025. [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[14] NVIDIA, "Jetson Nano Developer Kit User Guide," accessed on Apr. 26, 2025. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[15] Aosong Electronics, “DHT22 (AM2302) Temperature and Humidity Sensor Datasheet,” accessed on Apr. 26, 2025. [Online]. Available: https://www.aosong.com/en/products-61.html

[16] Winsen Electronics, “MQ-135 Gas Sensor Datasheet,” accessed on Apr. 26, 2025. [Online]. Available: https://www.winsen-sensor.com/d/files/PDF/MQ-135.pdf

[17] Plantower, “PMS5003 Particulate Matter Sensor Datasheet,” accessed on Apr. 26, 2025. [Online]. Available: http://www.plantower.com/en/content/?108.html

[18] SGX Sensortech, “MiCS-6814 Multi-Gas Sensor Datasheet,” accessed on Apr. 26, 2025. [Online]. Available: https://www.sgxsensortech.com/content/uploads/2014/08/0289_Datasheet-MiCS-6814-rev-16.pdf

[19] Hanwei Electronics, “MQ-2 Flammable Gas Sensor Datasheet,” accessed on Apr. 26, 2025. [Online]. Available: https://www.winsen-sensor.com/d/files/PDF/MQ-2.pdf

[20] Cubic Sensor and Instrument Co., “CM1106SL-NS NDIR CO_2 Sensor Datasheet,” accessed on Apr. 26, 2025. [Online]. Available: https://en.gassensor.com.cn/Products/NDIRgas-sensor/CM1106SL-NS.html

[21] SPEC Sensors, “3SP-SO2-20 Sulfur Dioxide Gas Sensor Datasheet,” accessed on Apr. 26, 2025. [Online]. Available: https://www.sgsensors.com/products/3sp-so2-20

[22] IEEE Standards Association, “IEEE 802.11 Wireless LAN Standards,” accessed on Apr. 26, 2025. [Online]. Available: https://standards.ieee.org/standard/802_11-2016.html

[23] Semtech Corporation, “LoRa Modulation Basics,” accessed on Apr. 26, 2025. [Online]. Available: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-modulation-basics/

[24] Texas Instruments, “Designing 5V Systems with Li-Ion Batteries,” accessed on Apr. 26, 2025. [Online]. Available: https://www.ti.com/lit/an/slva940/slva940.pdf

[25] Victron Energy, “Solar Charge Controllers Product Overview,” accessed on Apr. 26, 2025. [Online]. Available: https://www.victronenergy.com/solar-charge-controllers

[26] Arduino, "Arduino Uno Rev3," [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3. [Accessed: 25-Apr-2025].

[27] Bluetooth Special Interest Group (SIG), "Bluetooth Core Specification Version 5.2," [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/. [Accessed: 25-Apr-2025].

[28] Electronics Tutorials, "UART - Universal Asynchronous Receiver Transmitter," [Online]. Available: https://www.electronics-tutorials.ws/data/uarts.html. [Accessed: 25-Apr-2025].

[29] Hanwei Electronics, "MQ-7 Carbon Monoxide Gas Sensor Datasheet," [Online]. Available: https://www.winsen-sensor.com/d/files/PDF/MQ-7.pdf. [Accessed: 26-Apr-2025].

[30] Hanwei Electronics, "MQ-5 Gas Sensor Datasheet," [Online]. Available: https://www.winsen-sensor.com/d/files/PDF/MQ-5.pdf. [Accessed: 26-Apr-2025].

[31] SGX Sensortech, "MiCS-5524 Gas Sensor Datasheet," [Online]. Available: https://www.sgxsensortech.com/content/uploads/2014/08/0289_Datasheet-MiCS-5524-rev-16.pdf. [Accessed: 26-Apr-2025].

[32] Samyoung Electronics, "DSM501A Dust Sensor Datasheet," [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/gp2y1010au.pdf. [Accessed: 26-Apr-2025].

[33] Aosong Electronics, "DHT-11 Temperature and Humidity Sensor Datasheet," [Online]. Available: https://www.aosong.com/en/products-91.html. [Accessed: 26-Apr-2025].

[34] Winsen Electronics, "MH-Z14A CO_2 Sensor Datasheet," [Online]. Available: https://www.winsen-sensor.com/d/files/PDF/MH-Z14A.pdf. [Accessed: 26-Apr-2025].

[35] U.S. Environmental Protection Agency (EPA), "Particulate Matter (PM) Basics," [Online]. Available: https://www.epa.gov/pm-pollution/particulate-matter-pm-basics. [Accessed: 26-Apr-2025].

[36] U.S. Environmental Protection Agency (EPA), "Carbon Dioxide Concentrations," [Online]. Available: https://www.epa.gov/indoor-air-quality-iaq/what-carbon-dioxide-co2. [Accessed: 26-Apr-2025].

[37] U.S. Environmental Protection Agency (EPA), "Basic Information about NO_2," [Online]. Available: https://www.epa.gov/no2-pollution/basic-information-about-no2. [Accessed: 26-Apr-2025].

[38] U.S. Environmental Protection Agency (EPA), "Sulfur Dioxide (SO_2) Pollution," [Online]. Available: https://www.epa.gov/so2-pollution. [Accessed: 26-Apr-2025].

[39] U.S. Environmental Protection Agency (EPA), "Volatile Organic Compounds' Impact on Indoor Air Quality," [Online]. Available: https://www.epa.gov/indoor-air-quality-iaq/volatile-organic-compounds-impact-indoor-air-quality. [Accessed: 26-Apr-2025].

[40] World Health Organization (WHO), "WHO Housing and Health Guidelines: Temperature," [Online]. Available: https://www.who.int/publications/i/item/9789241550376. [Accessed: 26-Apr-2025].

[41] U.S. Environmental Protection Agency (EPA), "Indoor Air Quality: Humidity," [Online]. Available: https://www.epa.gov/indoor-air-quality-iaq/indoor-relative-humidity. [Accessed: 26-Apr-2025].

[42] J. Smith and R. Allen, “Occupational Exposure to Carbon Monoxide in Industrial Workplaces,” International Journal of Environmental Health, vol. 27, no. 2, pp. 84–91, 2020.

[43] Occupational Safety and Health Administration (OSHA), “Carbon Monoxide Poisoning,” OSHA Fact Sheet, 2023. [Online]. Available: https://www.osha.gov/sites/default/files/publications/CarbonMonoxide-Factsheet.pdf. Accessed: May 3, 2025.

[44] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010

[45] Y. Zhou, Y. Zheng, and X. Li, “Simplified air quality index models for IoT-based environmental monitoring systems,” *Sensors*, vol. 21, no. 9, pp. 1–18, 2021.

[46] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[47] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

AI APPENDICE I

نموذج إقرار وتوثيق استخدام أدوات الذكاء الاصطناعي في مشروع التخرج

اسم الطالب / أسماء الطلبة: قسام ياسر عواودة / ليث عوني الفقيه / زيدان نضال التلاحمة	عنوان المشروع: نظام ذكي وآلي لمراقبة جودة الهواء للمنشآت الصناعية باستخدام التعلم الآلي والتنبؤ بالتلوث
الرقم / الأرقام الجامعية: 211039 / 211038 /211047	اسم المشرف الأكاديمي: د. عليان أبو غربية
الكلية / الدائرة: تكنولوجيا المعلومات هندسة الحاسوب	السنة/الفصل الدراسي:2025

الغرض من الاستخدام	اسم الأداة	أماكن الاستخدام في التقرير (فصول / صفحات)	الأمر الأساسي المستخدم (**Prompt**)	هل تم تعديل الناتج ؟	نسبة الاستخدام التقديرية	ملاحظات
توليد نصوص	☐ ChatGPT				0%	لم يستخدم
	☐ Gemini				0%	لم يستخدم
	☐ Claude				0%	لم يستخدم
	☐ Microsoft Copilot				0%	لم يستخدم
	☐ other: ____				0%	لم يستخدم
تدقيق لغوي	☐ Grammarly		—		0%	لم يستخدم
	☐ Quillbot				0%	لم يستخدم
	☐ Word Editor AI				0%	لم يستخدم
	☐ ChatGPT	غالبية التقرير	أعطني التحسينات اللغوية للنص التالي	نعم	15%	لم ينشئ نص كامل انما حدد

		الفصل 1 و 2 خصيصا				جزئية وأعطى نصيحة تحسينية
تلخيص محتوى	☐ ChatGPT	تلخيص بعض المراجع	اقتبس النصوص الخاصة ب "" من هذا المرجع	نعم	20%	قراءة سريعة لأجزاء من مراجع ضخمة
	☐ SMMRY				0%	لم يستخدم
	☐ Notion AI				0%	لم يستخدم
	☐ Scholarcy				0%	لم يستخدم
	☐ other: ____				0%	لم يستخدم
تحليل بيانات	☐ Excel Copilot				0%	لا يوجد بيانات
	☐ Python AI Assistants				0%	لا يوجد بيانات
	☐ Tableau AI				0%	لا يوجد بيانات
	☐ other: ____				0%	لا يوجد بيانات
رسم مخططات / أشكال	☐ ChatGPT	**Conceptual diagram**	نصائح لتطوير الرسم وهل من أخطاء بها	نعم	8%	لام يتم اخذ أي عمل جاهز انما تم رسمها بشكل كامل
	☐ Canva AI				0%	لم يستخدم
	☐ Visio AI				0%	لم يستخدم
	☐ Lucidchart AI				0%	لم يستخدم
	☐ other: ____				0%	لم يستخدم
كتابة أكواد برمجية	☐ GitHub Copilot				0%	لم يستخدم

	☐ Replit AI				0%	لم يستخدم
	☐ chatgpt	حل مشاكل	إعطاء الكود و نص الإرور		10%	تم الاستخدام
	☐ other: ____				0%	لم يستخدم
توثيق مراجع	☐ EndNote				0%	لم يستخدم
	☐ Mendeley				0%	لم يستخدم
	☐ Zotero				0%	لم يستخدم
	☐ ChatGPT	بعض المراجع	اعطني مرجع موثوق عن "عنوان الموضوع"		20%	غالبية البحث كانت تتم من قوقل
	☐ other: ____				0%	لم يستخدم
توليد داتا	☐ ChatGPT	إعطاء داتا سيت	تم تحديد المراجع و الوقت و القيم والزمن وتم توليد الداتا لتحاكي بيئة واقعية			الداتا ست منه و التدقيق اثناء التدريب

إقرار فريق مشروع التخرج:

نقر بأن استخدام أدوات الذكاء الاصطناعي تم بشكل مسؤول وبما يتوافق مع السياسات الجامعية، وقد راجعنا وحررنا كافة المخرجات بما يعكس فهمنا الشخصي.

توقيع الطالب .....التاريخ:.....

توقيع الطالب ..... التاريخ:.....

توقيع الطالب .....التاريخ:.....

APPENDICE II

Timestamp	Day	PM2.5	PM10	CO2	NO2	VOCs	CO	Temperatu	Humidity	AQI	AQI_plus_
1/4/2025 7:00	Saturday	26.32	78.62	671.9	0.0195	0.177	0.461	25.65	52.5	108.26	183.58
1/4/2025 7:05	Saturday	52.99	42.62	654.5	0.0299	0.171	0.644	25.37	51.4	132.69	176.98
1/4/2025 7:10	Saturday	69.63	10	670.1	0.0299	0.171	0.795	24.09	48.6	150.89	161.95
1/4/2025 7:15	Saturday	30.27	123.08	818.9	0.0307	0.354	0.679	25.7	60.3	141.66	125.09
1/4/2025 7:20	Saturday	42.72	121.33	563.6	0.0437	0.285	0.939	29.15	44.6	122.83	190.15
1/4/2025 7:25	Saturday	24.77	167.19	694.8	0.0234	0.392	1.008	26.33	54.4	126.92	175.9
1/4/2025 7:30	Saturday	51.16	76.35	628	0.0324	0.306	0.367	29.71	59.9	139.46	175.61
1/4/2025 7:35	Saturday	92.74	26.85	775.4	0.0349	0.364	1.201	27.53	57	200.61	184.3
1/4/2025 7:40	Saturday	123.72	150.64	575.7	0.0334	0.329	0.504	29.3	57.1	208.71	177.73
1/4/2025 7:45	Saturday	150.52	115.94	560.4	0.0442	0.243	0.444	25.16	52.2	226.81	186.78
1/4/2025 7:50	Saturday	69.38	62.25	564.5	0.0369	0.417	0.59	31.83	61.6	160.58	207.03
1/4/2025 7:55	Saturday	77.52	137.22	516.1	0.0343	0.15	0.512	25.91	47.8	141.63	236.94
1/4/2025 8:00	Saturday	73.37	99.17	652.9	0.0208	0.539	1.106	29.47	47.4	183.58	169.34
1/4/2025 8:05	Saturday	99.93	109.68	653	0.0218	0.141	1.069	30.63	45.6	176.98	158.07
1/4/2025 8:10	Saturday	68.4	87.13	585.5	0.0384	0.42	0.794	28.1	54.5	161.95	224.32
1/4/2025 8:15	Saturday	48.63	113.84	626.3	0.0451	0.166	1.1	28.6	61	125.09	127.75
1/4/2025 8:20	Saturday	73.16	117.13	731.6	0.0415	0.526	0.948	27.35	45.2	190.15	191.19
1/4/2025 8:25	Saturday	71.57	141.13	838.3	0.0214	0.246	1.141	27.41	56.5	175.9	229.5
1/4/2025 8:30	Saturday	59.86	136.22	864.2	0.0323	0.352	1.103	27.96	47.3	175.61	205.1
1/4/2025 8:35	Saturday	91.9	132.76	659	0.0441	0.318	0.659	30.15	57.8	184.3	187.95
1/4/2025 8:40	Saturday	79.76	71.95	712.2	0.0312	0.321	1.106	27.36	45.4	177.73	239.3
1/4/2025 8:45	Saturday	93.37	99.33	709.1	0.0377	0.27	1.172	27.59	48.3	186.78	267.62
1/4/2025 8:50	Saturday	81.27	153.18	870.9	0.037	0.464	0.987	29.27	53.2	207.03	215.53
1/4/2025 8:55	Saturday	123.58	93.34	740.3	0.0421	0.472	0.458	27.34	59.2	236.94	227.54
1/4/2025 9:00	Saturday	98.73	135.13	650.3	0.0386	0.067	0.748	25.4	58.8	169.34	228.51
1/4/2025 9:05	Saturday	54.74	57.54	828.3	0.0623	0.246	0.981	31.64	51.1	158.07	206.83
1/4/2025 9:10	Saturday	119.38	142.37	755.2	0.0406	0.353	0.987	30.63	37.6	224.32	241.26
1/4/2025 9:15	Saturday	39.32	158.02	650.1	0.0513	0.281	2	30.23	51.7	127.75	209.78
1/4/2025 9:20	Saturday	100.88	137.83	749.8	0.039	0.184	1.128	29.93	51.8	191.19	246.57
1/4/2025 9:25	Saturday	103.08	155.95	893.4	0.0308	0.445	0.998	28.13	47.5	229.5	237.48
1/4/2025 9:30	Saturday	100.52	60.91	734.1	0.0452	0.374	1.181	29.81	53.9	205.1	299.25
1/4/2025 9:35	Saturday	76.15	155.92	790.5	0.0468	0.393	1.289	28.19	49.4	187.95	266.49
1/4/2025 9:40	Saturday	126.38	96.52	897.5	0.0434	0.278	0.975	31.05	54.8	239.3	208.45
1/4/2025 9:45	Saturday	144.95	245.03	937.5	0.048	0.347	1.075	31.9	46.9	267.62	236.39
1/4/2025 9:50	Saturday	103.76	167.47	928.5	0.0441	0.227	1.07	28.97	44.3	215.53	218.48
1/4/2025 9:55	Saturday	111.02	172.01	892.7	0.037	0.327	1.292	30.42	37.5	227.54	196.9
1/4/2025 10:00	Saturday	90.61	61.15	990.7	0.0515	0.466	1.185	31.47	52.6	228.51	215.46
1/4/2025 10:05	Saturday	112.02	63.14	736.4	0.0487	0.254	1.62	28.6	46	206.83	273.69
1/4/2025 10:10	Saturday	118.66	196.77	762.7	0.0372	0.556	0.587	29.37	48.5	241.26	239.15
1/4/2025 10:15	Saturday	114.12	143.95	579.9	0.0684	0.452	1.367	31.43	41	209.78	330.62
1/4/2025 10:20	Saturday	126.84	138.81	684.8	0.0377	0.615	1.52	28.43	48.4	246.57	252.82
1/4/2025 10:25	Saturday	100.62	147.41	901.1	0.0504	0.561	1.353	28.48	55.3	237.48	220.08
1/4/2025 10:30	Saturday	159.12	199.57	968.8	0.0587	0.519	0.906	30.43	56.7	299.25	289.04
1/4/2025 10:35	Saturday	132.51	140.34	811.5	0.0371	0.634	0.937	31.66	45.6	266.49	264.79
1/4/2025 10:40	Saturday	79.98	205.66	716.4	0.044	0.682	0.964	28.9	51.9	208.45	249.51
1/4/2025 10:45	Saturday	116.69	135.9	903.7	0.0347	0.352	1.202	30.15	44.8	236.39	255.57
1/4/2025 10:50	Saturday	94.04	186.64	784.4	0.053	0.552	1.3	33.17	38.6	218.48	235.38
1/4/2025 10:55	Saturday	57.84	127.92	954.8	0.0413	0.523	1.249	30.67	42.8	196.9	256.83
1/4/2025 11:00	Saturday	114.91	196.44	756.3	0.0339	0.299	0.674	31.61	40	215.46	230.49

Figure AP II. 1: Sample from Dataset