**AIR POLLUTION MONITORING SYSTEM**

**USING ARDUINO**

*An Mini Project Report Submitted*

In partial fulfillment of the requirement for the award of the degree of

## Bachelor of Technology

**in**

**Computer Science and Engineering (Internet of Things)**

**by**

## ANANTHULA MANOJ 20N31A6907

## ANANTHANENI AMRUTHA 20N31A6906

## ALLTHOLA HARIKA 20N31A6905

***Under the Guidance of***

**Mrs. V. Bala**

**Assistant Professor**

**Department of Emerging Technologies**

**MRCET (Autonomous Institution, UGC Govt. of India)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-DATA SCIENCE**

**(EMERGING TECHNOLOGIES)**

**MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous Institution – UGC, Govt. of India)**

**(Affiliated to JNTU, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC – ‘A’ Grade, ISO 9001:2015 Certified)**

Maisammaguda (v), Near Dullapally, Via: Kompally, Hyderabad – 500 100, Telangana State, India

2023-2024

**DECLARATION**

We hereby declare that the project entitled “**AIR POLLUTION MONITORING SYSTEM USING ARDUINO**” submitted to **Malla Reddy College of Engineering and Technology,** affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) as part of IV Year B. Tech – I Semester and for the partial fulfilment of the requirement for the award of **Bachelor of Technology** in **Computer Science and Engineering (Internet of Things)** is a result of original research work done by us.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

## 1.ANANTHULA MANOJ 20N31A6907

## 2.ANANTHANENI AMRUTHA 20N31A6906

## 3.ALLTHOLA HARIKA 20N31A6905



CERTIFICATE

This is to certify that this is the bonafide record of the project titled “**AIR POLLUTION MONITORING SYSTEM USING ARDUINO”**, submitted by **ANANTHULA MANOJ**(20N31A6907), **ANANTHANENI AMRUTHA (**20N31A6906) and **ALLTHOLA HARIKA** (20N31A6905) of **B. Tech IV YEAR – I Semester** in the partial fulfilment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering (Internet of Things)**, Dept. of CSE (Emerging Technologies) during the year 2022-2023. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Mrs. V. Bala Mrs. V. Bala

**Project Guide**  **Project Coordinator**

**Department of CSE (ET) Department of CSE (ET)**

Dr. P. Dileep  Dr. M. V. Kamal

**OVERALL COORDINATOR EXTERNAL HEAD OF THE DEPARTMENT**

**EXAMINER**

**Date of Viva-Voce Examination held on:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## ACKNOWLEDGEMENTS

We feel ourself honored and privileged to place our warm salutation to our college “Malla Reddy College of Engineering and Technology (Autonomous Institution – UGC Govt. of India) and our Principal **Dr. S Srinivasa Rao,** Professor who gave us the opportunity to do the Mini Project during our IV Year B. Tech and profound the technical skills.

We express our heartiest thanks to our Director **Dr. V S K Reddy,** Professor for encouraging us in every aspect of our project and helping us realize our full potential.

We also thankful to our Head of the Department **Dr. M V Kamal,** Professor for providing training and guidance, excellent infrastructure, and a nice atmosphere for completing this project successfully.

We would like to express our sincere gratitude and indebtedness to our project supervisor **Mrs. V. Bala,** Assistant Professor for her valuable suggestions and interest throughout the course of this project.

We convey our heartfelt thanks to our Project Coordinator **Dr. P Dileep,** Professor for allowing for their regular guidance and constant encouragement during our dissertation work.

We would like to thank all our supporting **staff** of the Department of CSE (Emerging Technologies) and even all other department who have been helpful directly and in-directly in making our project a success.

Finally, we would like to take this opportunity to thank our **families** for their support and blessings for completion of our project that gave us the strength to do our project.

**ANANTHULA MANOJ 20N31A6907**

**ANANTHANENI AMRUTHA 20N31A6906**

**ALLTHOLA HARIKA 20N31A6905**

**ABSTRACT**

In this project we are going to make an IoT Based Air Pollution Monitoring System in which we will monitor the Air Quality and it will trigger an alarm when the air quality goes down beyond a certain level. This means, when there are a sufficient number of harmful gases present in the air like CO2, smoke, alcohol, benzene, and NH3 it will show the air quality (in PPM) on the LCD. So that we can monitor it very easily. It consists of a micro controller, gas sensors, a mobile unit, a temporary memory buffer with internet connectivity which collects data from different locations along with coordinating information at a certain time of the day. The readings for a particular location are averaged in a closed time and space. We have used the MQ135 sensor as the air quality sensor which is the best choice for monitoring Air Quality as it can detect most harmful gases and can measure their amount accurately. This information can then be applicable in many ways. Analysis of monitoring data allows us to assess how bad air pollution are from day to day.

#### TABLE OF CONTENTS

**S.No. Topic Page No.**

CHAPTER 1: INTRODUCTION --------------------------------------------------------------- 1

1.1: Problem Definition

1.2:Existing System

1.3:Proposed System

CHAPTER 2:SYSTEM REQUIREMENTS -------------------------------------------------- 4

2.1:S/W Requirements

2.2:H/W Requirements

2.3:Languages used in Arduino Uno Development

2.4:Modules Used In Arduino

CHAPTER 3: SYSTEM DESIGN --------------------------------------------------------------- 8

3.1:Software

3.2:UML Diagrams

3.3:System Architecture/Class Diagram/Sequential Diagram

CHAPTER 4: SOFTWARE DEVELOPMENT LIFE CYCLE ----------------------------- 15

4.1:SDLC

4.2:Phases of SDLC

CHAPTER 5: ARDUINO COMPONENTS----------------------------------------------------- 18

5.1:Arduino

5.2:Components in Arduino

5.3:Components used in Air Pollution Monitoring System

CHAPTER 6: ARDUINO CIRCUIT ----------------------------------------------------------- 22

6.1: Arduino

6.2: Languages used in Arduino Uno Development

6.3: Modules Used In Arduino

CHAPTER 7: IMPLEMENTATION ---------------------------------------------------------- 27

CHAPTER 8: TESTING ------------------------------------------------------------------------- 34

CHAPTER 9: CONCLUSION AND FUTURE SCOPE ------------------------------------ 41

9.1: Conclusion

9.2: Future Scope

CHAPTER 10: REFERENCES ----------------------------------------------------------------- 42

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Figure No. | Figure Title | Page No. |
| 1 | Dataflow Diagram | 10 |
| 2 | Use Case Diagram | 11 |
| 3 | System Architecture | 12 |
| 4 | Class Diagram | 13 |
| 5 | Sequential Diagram | 14 |
| 6 | Phases of SDLC | 15 |
| 7 | Arduino board Representation | 23 |
| 8 | Circuit Diagram | 26 |
| 9 | Circuit Model | 34 |
| 10 | Test Case for Detecting Fresh Air | 35 |
| 11 | Test case for Detecting Poor Air | 36 |
| 12 | Test Case for Detecting Very Poor Air | 37 |
| 13 | Test Case for Sending Data to Cloud | 38 |
| 14 | Test Case for Showing Connectivity of Wi-Fi Module | 39 |
| 15 | Overall Test Cases | 40 |

## CHAPTER 1 INTRODUCTION

Air pollution is a growing concern that poses significant risks to the environment and human health worldwide. The presence of harmful gases such as carbon dioxide (CO2), smoke, alcohol, benzene, and ammonia (NH3) in the air can have detrimental effects, ranging from respiratory problems to environmental degradation. In response to this critical issue, we have embarked on a project to develop an IoT-Based Air Pollution Monitoring System.

Our project's primary objective is to create a comprehensive system capable of continuously monitoring air quality and providing timely alerts when pollution levels deteriorate beyond specified thresholds. By doing so, we aim to empower individuals and communities to take proactive measures in safeguarding their well-being and the environment.

The core components of our Air Pollution Monitoring System include a microcontroller, gas sensors, a mobile unit, a temporary memory buffer, and an internet-connected interface. These components work together seamlessly to collect, process, and disseminate real-time air quality data from various locations. This data is crucial for tracking and analyzing pollution levels, making informed decisions, and implementing pollution control measures effectively.

Central to our system's functionality is the use of the MQ135 sensor, renowned for its ability to detect and measure a wide range of harmful gases with remarkable accuracy. The sensor provides precise measurements in parts per million (PPM), which are then displayed on a Liquid Crystal Display (LCD) for easy monitoring. Additionally, the data is made accessible through a web-based interface, ensuring convenient remote monitoring and access.

The collected air quality data serves as a valuable resource for various stakeholders, including researchers, environmentalists, policymakers, and concerned citizens. It enables them to gain insights into air pollution trends, assess the effectiveness of mitigation efforts, and make data-driven decisions to combat air pollution effectively.

In recent years, the issue of air pollution has gained significant attention due to its adverse effects on public health and the environment. The rapid industrialization, urbanization, and increasing vehicular emissions have amplified the concentration of harmful gases and particulate matter in the atmosphere, posing a substantial threat to the well-being of communities worldwide. As a result, the need for effective air quality monitoring systems has become increasingly paramount in the global discourse on sustainability and public health.

Our IoT-Based Air Pollution Monitoring System represents a cutting-edge solution that leverages the power of modern technology to address the complexities of monitoring and managing air quality. By harnessing the capabilities of the Internet of Things (IoT), our system enables real-time data collection, analysis, and dissemination, facilitating proactive measures to mitigate the impact of air pollution.

At the heart of our system lies a sophisticated microcontroller, engineered to process data from a network of advanced gas sensors strategically placed in key monitoring locations. These sensors, including the MQ135 and other specialized detectors, have been meticulously calibrated to detect a wide spectrum of pollutants accurately. They provide continuous measurements of harmful gases, particulate matter, and volatile organic compounds (VOCs), enabling comprehensive insights into the air quality parameters.

To ensure seamless accessibility and usability, our system incorporates a user-friendly mobile unit equipped with intuitive interfaces for monitoring air quality data remotely. This feature enables users to receive real-time alerts and notifications, empowering them to make informed decisions promptly and take necessary precautions during periods of heightened pollution levels.

Furthermore, our system integrates a temporary memory buffer mechanism that guarantees the continuous and uninterrupted collection of data, even in the event of network connectivity issues. This redundancy ensures that no critical data points are lost and that the system maintains its reliability and accuracy at all times.

By consolidating the data into a centralized and internet-connected interface, our solution facilitates the seamless sharing of information with relevant stakeholders, including environmental agencies, local authorities, and the general public. This open and transparent approach fosters community engagement and collaboration, encouraging collective efforts towards implementing sustainable environmental practices and fostering a cleaner and healthier living environment for all.

## PROBLEM DEFINITION

The problem at hand is the growing menace of air pollution, characterized by the presence of hazardous gases such as CO2, smoke, alcohol, benzene, and NH3 in the atmosphere. This poses substantial health risks and environmental degradation. The current challenge lies in the absence of real-time and accessible air quality monitoring systems, which hampers the timely detection of pollution and the issuance of alerts when air quality deteriorates beyond safe levels. Additionally, there is a lack of efficient data collection and analysis methods, limited public awareness, and underutilization of air quality data for informed decision-making. Addressing these issues is vital to mitigating the adverse effects of air pollution and safeguarding both human well-being and the environment.

## EXISTING SYSTEM

* Traditional air pollution monitoring systems typically consist of fixed, government-operated stations strategically placed across urban areas.
* These stations utilize expensive and sophisticated equipment to measure various air pollutants such as particulate matter, gases, and volatile organic compounds.
* The data collected from these stations is sent to central databases and used to calculate air quality indices that provide an overall picture of the air quality in a region.
* It involves the use of specialized equipment, which can be costly and require technical expertise for setup and maintenance.
* These systems often consist of stationary monitoring stations placed at specific locations. They use sophisticated sensors to measure various pollutants and transmit data to central servers for analysis.
* There are some limitations in this existing system cost, uneven coverage, Accessibility, and scalability.

## PROPOSED SYSTEM

* The proposed Air Pollution Monitoring System using Arduino is designed to provide a comprehensive and cost-effective solution to monitor air quality in real-time.
* By leveraging the versatility of Arduino boards and sensors, this system aims to bridge the gap between traditional monitoring methods and modern technology, enabling accurate data collection, analysis, and dissemination for informed decision-making.
* It aims to measure various pollutants in the air and provide real-time data for analysis.
* The system will use sensors to detect pollutants such as particulate matter (PM2.5 and PM10), carbon monoxide (CO), sulfur dioxide (SO2), and nitrogen dioxide (NO2).
* The Arduino board will process the sensor data and transmit it to a display unit or a remote server for visualization.
* This project can contribute to environmental awareness and help people make informed decisions regarding air quality.

**CHAPTER2**

**SYSTEM REQUIREMENTS**

# 2.1 SOFTWARE REQUIREMENTS:

* **Arduino IDE**

The Arduino Integrated Development Environment (IDE) is the primary software tool used for writing, compiling, and uploading code to Arduino boards. It provides a user-friendly interface for creating and managing Arduino projects, making it accessible for beginners and experienced developers alike.

* **Thingspeak.com**

ThingSpeak.com is an IoT analytics platform that allows users to collect, analyze, and visualize data from their IoT devices. It provides an easy-to-use interface for storing, retrieving, and analyzing data in the cloud. Here are some key features and functionalities of ThingSpeak.com:

* + Data Collection: ThingSpeak enables users to collect data from various IoT devices, sensors, and applications, making it ideal for projects involving environmental monitoring, smart agriculture, industrial automation, and more.
  + Data Analysis and Visualization: The platform offers powerful data analysis tools, including MATLAB analytics, for processing and visualizing data in real-time. Users can create custom plots, graphs, and charts to gain insights into their data trends and patterns.
  + Cloud Storage: ThingSpeak provides cloud-based data storage, ensuring that users can securely store and access their data from anywhere at any time. This feature allows for seamless data management and retrieval, facilitating easy integration with other cloud-based services and applications.
  + Integration with IoT Devices: ThingSpeak supports integration with a wide range of IoT devices, including Arduino, Raspberry Pi, ESP8266, and other popular microcontroller platforms. It provides APIs and libraries that simplify the process of sending data from IoT devices to the cloud.
  + Real-Time Alerts and Notifications: Users can set up alerts and notifications based on predefined thresholds or conditions in their data. This feature enables users to receive real-time alerts via email, SMS, or other communication channels, ensuring prompt responses to critical events or anomalies.
  + Open API: ThingSpeak offers an open API that allows users to access and manipulate their data programmatically. This feature facilitates seamless integration with third-party applications and services, enabling users to customize their IoT workflows and applications according to their specific requirements.
  + Community and Collaboration: ThingSpeak has a vibrant community of developers, researchers, and enthusiasts who share projects, insights, and best practices related to IoT and data analytics. The platform encourages collaboration and knowledge sharing, fostering innovation and creativity in the IoT ecosystem.

ThingSpeak.com serves as an invaluable tool for IoT developers, researchers, and hobbyists, empowering them to harness the power of data and analytics to create innovative and impactful IoT solutions for various domains and applications.

# 2.2 HARDWARE REQUIREMENTS:

* Arduino Uno R3
* Gas Sensors (MQ135)
* LCD Display
* Potentiometer
* Resistor
* Wi-fi Module (ESP8266)
* Bread Board
* Buzzer

#### Programming Language:

* Embedded C Programming

**2.3** **LANGUAGES USED IN ARDUINO UNO DEVELOPMENT :**

When working with the Arduino Uno, you can use several programming languages and environments to develop applications for the board. The primary language used for Arduino development is a simplified version of C++ that is supported by the Arduino IDE (Integrated Development Environment). Here are the main languages and tools used for Arduino Uno development:

Arduino Programming Language (based on C++): The Arduino programming language is a simplified version of C and C++. It provides easy-to-use functions and libraries to in with the hardware components, making it accessible for beginners and experienced developers alike.

Arduino IDE: The Arduino Integrated Development Environment is the primary software tool used for writing and uploading code to the Arduino board. It provides a code editor, compiler, and a simple interface for managing libraries and board settings.

C and C++: Since the Arduino programming language is based on C++, developers familiar with C or C++ can also directly program the Arduino Uno using these languages. They can access the full power of C++ to develop more complex and advanced applications.

MicroPython: Some variations of Arduino boards, such as the Arduino Nano 33 IoT and Arduino MKR series, support MicroPython, which is a subset of the Python 3 programming language optimized to run on microcontrollers and constrained systems.

JavaScript (Node.js): For IoT projects, it's also possible to use JavaScript and Node.js to communicate with the Arduino Uno through a serial connection or using a compatible board like the Arduino Nano 33 IoT, which supports JavaScript.

While the primary language for Arduino development is a simplified version of C++, the board's flexibility and compatibility with various programming languages make it a versatile platform for a wide range of applications and projects.

**2.4 MODULES USED IN ARDUINO:**

Arduino is compatible with a wide range of modules and sensors, allowing users to easily expand the capabilities of their projects. These modules enable interaction with various types of input and output devices, such as sensors, displays, communication modules, motors, and more. Some of the commonly used modules in Arduino projects include:

Sensors:

Temperature and humidity sensor (DHT11, DHT22)

Ultrasonic sensor (HC-SR04)

Light sensor (LDR, photo resistor)

Motion sensor (PIR sensor)

Gas sensor (MQ series)

Accelerometer and gyroscope (MPU-6050)

Actuators:

Servo motor

DC motor

Stepper motor

Relay module

Communication Modules:

Wi-Fi modules (ESP8266, ESP32)

Bluetooth modules (HC-05, HC-06)

RFID modules

GSM modules (SIM900, SIM800)

Displays:

LCD display (16x2, 20x4)

OLED display

Seven-segment display

TFT display

Wireless Communication:

RF transmitters and receivers

LoRa modules

Zigbee modules

Infrared (IR) communication modules

Input Devices:

Keypad module

Touchscreen module

Potentiometer

Other Modules:

Real-time clock (RTC) modules

GPS modules

SD card modules

Audio modules

Motor driver modules

These modules can be easily integrated into Arduino projects using the appropriate libraries and code examples available in the Arduino IDE. They significantly expand the capabilities of the Arduino platform, allowing users to create a wide variety of projects in fields such as home automation, robotics, IoT, and more.

**CHAPTER 3**

**SYSTEM DESIGN**

* 1. **SOFTWARE**

### Arduino IDE (Integrated Development Environment) is a crucial software tool for developing the IoT-Based Air Pollution Monitoring System described in your abstract. It plays a central role in programming the microcontroller, which is the core of the system.

### Arduino IDE serves as the primary programming environment for the microcontroller used in the air pollution monitoring system. It allows you to write, compile, and upload code to the microcontroller effortlessly. In your project, the microcontroller is responsible for interfacing with gas sensors, collecting data, and controlling the LCD display. Arduino IDE simplifies the coding process by providing a user-friendly interface and libraries specifically designed for Arduino boards.

### **PROGRAMMING LANGUAGE:**

Arduino IDE uses a simplified version of the C/C++ programming language, making it accessible to both beginners and experienced developers. This language is well-suited for embedded systems and microcontroller programming. It allows you to define the behavior of your air pollution monitoring system, read sensor data, control outputs (like the LCD display), and establish communication with the internet-connected components.

#### KEY FEATURES AND BENEFITS:

Arduino IDE offers a rich library ecosystem that includes libraries for interfacing with various sensors, Arduino is an open-source electronics platform that is widely used for creating interactive projects. It consists of both hardware and software components and is popular among hobbyists, educators, and professionals for its versatility and ease of use. Here are some of the key features and benefits of Arduino:

Key Features:

* Open-source: Arduino's hardware designs and software are open-source, allowing users to understand, modify, and distribute the technology as they see fit.
* User-friendly: Arduino is known for its user-friendly development environment, which simplifies the process of writing code and uploading it to the Arduino board.
* Versatility: It supports a wide range of sensors, modules, and components, enabling the creation of various projects ranging from simple LED displays to complex robotics and automation systems.
* Expandability: Arduino boards can be easily expanded and integrated with various shields, which are add-on boards that provide additional functionalities such as wireless communication, motor control, and display capabilities.
* Cross-platform compatibility: Arduino software is compatible with multiple operating systems, including Windows, Mac OS X, and Linux.
* Large community support: Arduino has a large and active community of users and developers who share knowledge, provide support, and contribute to the development of new libraries, projects, and tutorials.

Key Benefits:

* Learning platform: Arduino serves as an excellent educational tool for teaching electronics and programming concepts, making it popular in schools, universities, and workshops.
* Cost-effective: Arduino boards are relatively inexpensive compared to other development platforms, making them accessible to a wide range of users, including students, hobbyists, and professionals.
* Rapid prototyping: With its simplified programming environment and a wide range of compatible components, Arduino facilitates rapid prototyping, enabling users to quickly test and validate their ideas and concepts.
* Customization: Arduino allows users to customize and tailor their projects according to their specific needs, enabling them to create unique and personalized solutions.
* Integration with other technologies: Arduino can easily interface with other technologies and platforms, making it suitable for integration into larger and more complex systems and projects.
* Overall, the key features and benefits of Arduino make it a popular choice for individuals and organizations looking to develop interactive and innovative electronics projects, prototypes, and products.

## [Air Quality measurement](https://github.com/manjiyani/Air-Quality-Monitoring-System/tree/master" \l "air-quality-measurement)

1. Converting air pollutant concentration
2. Converting Micro grams per cubic meter to PPM ppmv = mg/m^3 x (0.08205 x T) / M

3.Converting PPM to Micro grams per cubic meter mg/m^3 = ppmv x M /(0.08205 x T)

Where,

mg/m^3 = microgram of pollutant per cubic meter of air

ppmv = air pollutant concentration, in parts per million by volume

T = ambient temperature in kelvin

0.08205 = Universal gas constant

M = Molecular weight of air pollutant

#### Data Flow Diagrams / UML Diagrams

**3.2.1 Data Flow Diagram**

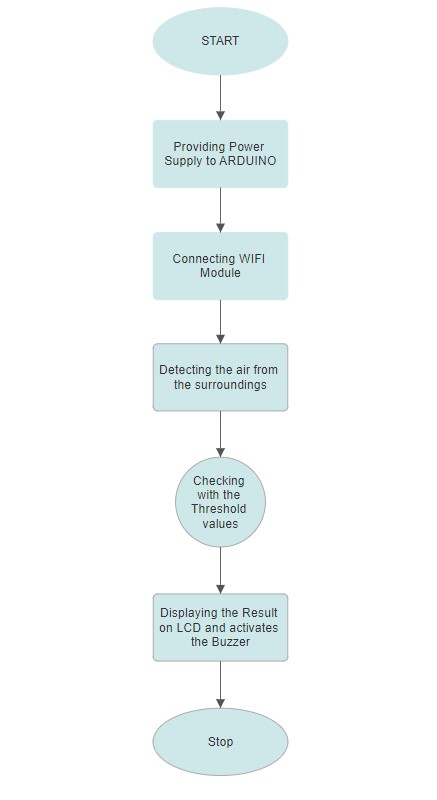
****

Fig 1: Data Flow Diagram

#### 3.2.2 Use Case Diagram:

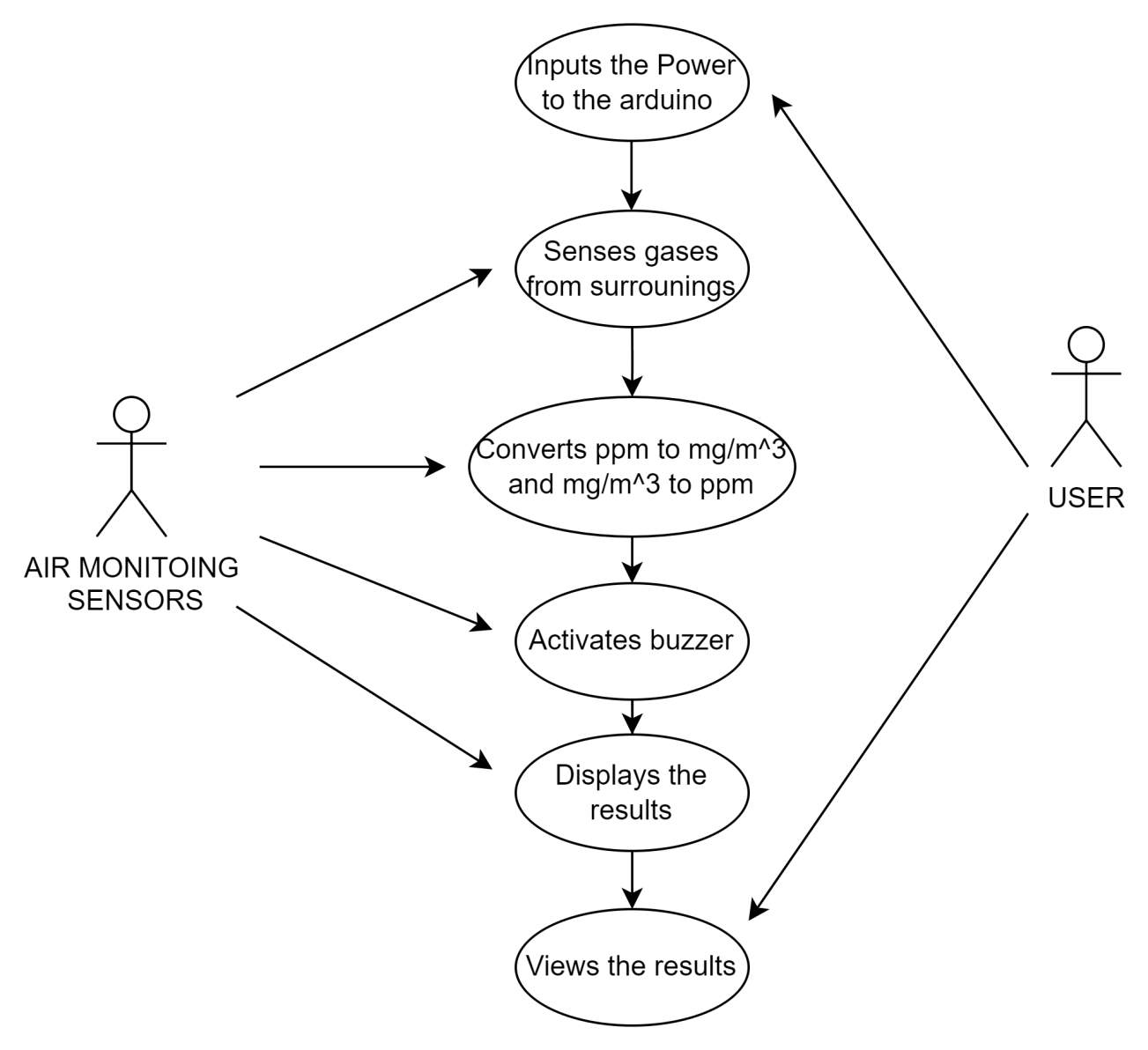
****

Fig 2: Use Case Diagram

#### System Architecture/ Class Diagram/ Sequential Diagram:

**3.3.1 System Architecture**

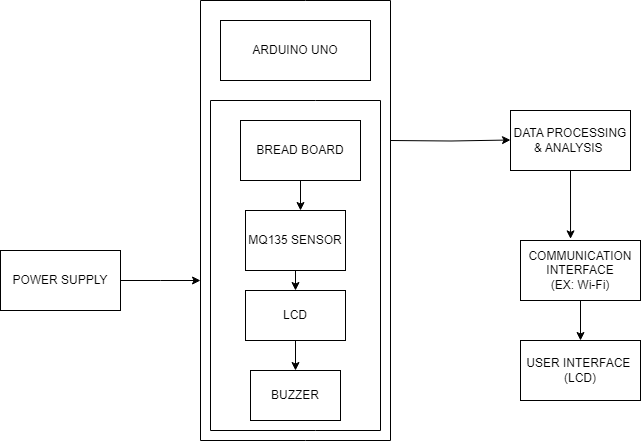
****

Fig 3: System Architecture

#### 3.3.2 Class Diagram:

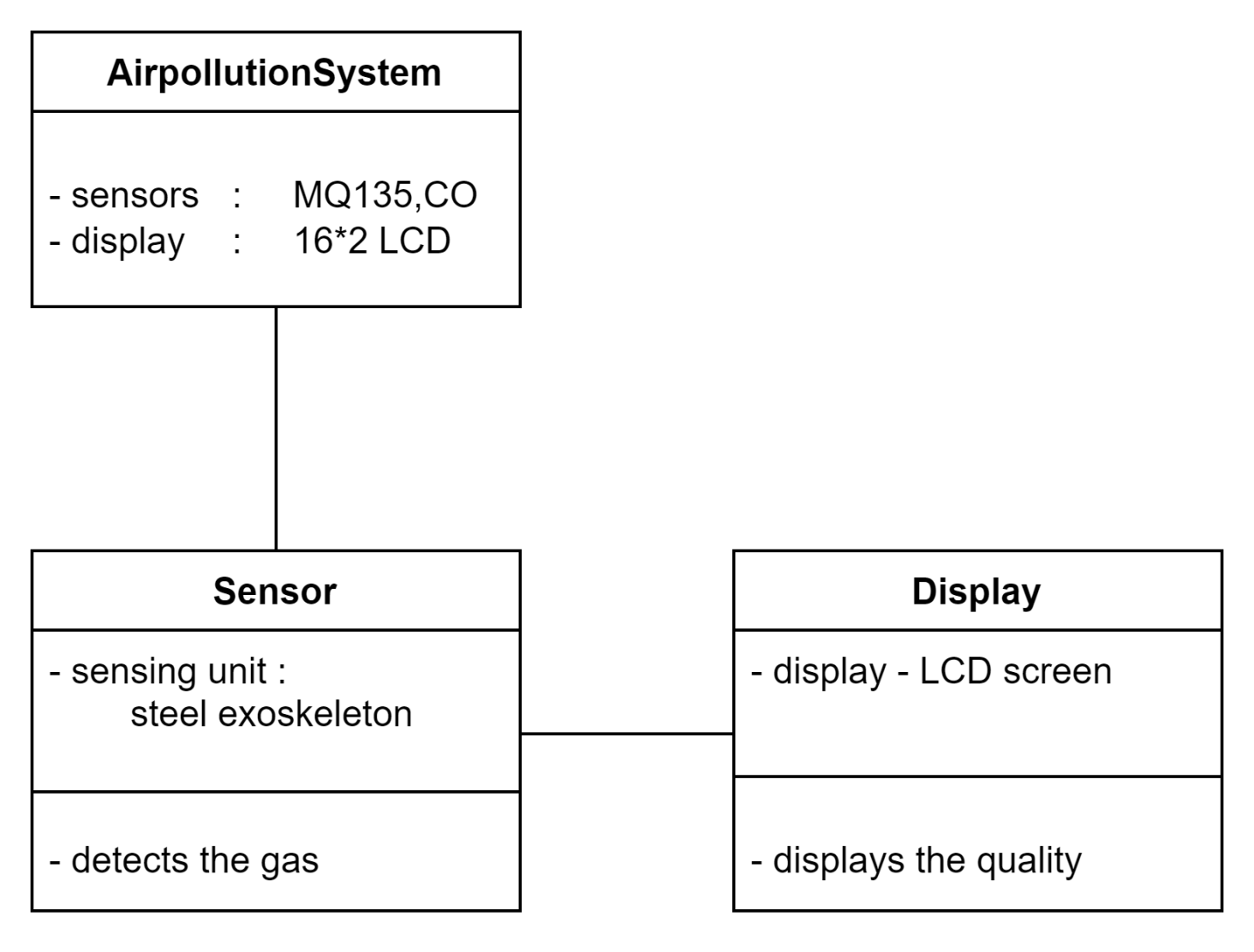
****

Fig 4:Class Diagram

#### 3.3.3 Sequential Diagram:

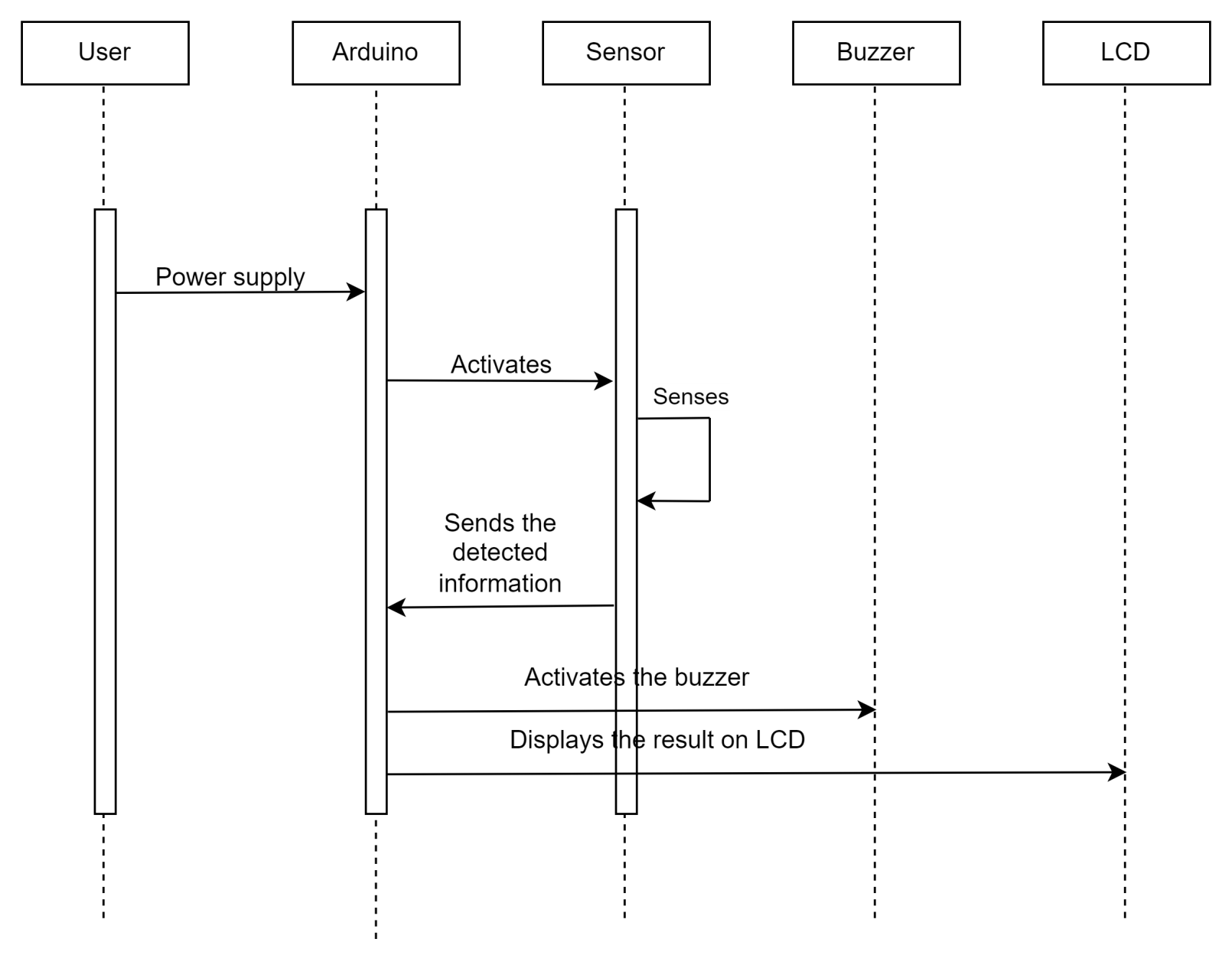
****

Fig 5: Sequential Diagram

# CHAPTER 4

**SOFTWARE DEVELOPMENT LIFE CYCLE**

## 4.1 SDLC:

## SDLC stands for Software Development Life Cycle, which is a process used by software development teams to design, develop, test, and deploy software applications. The SDLC consists of several stages, including planning, analysis, design, implementation, testing, and maintenance.

## 4.2 PHASES OF SDLC:

A system development life cycle or SDLC is essentially a project management model. It defines different stages that are necessary to bring a project from its initial idea or conception all the way to deployment and later maintenance.

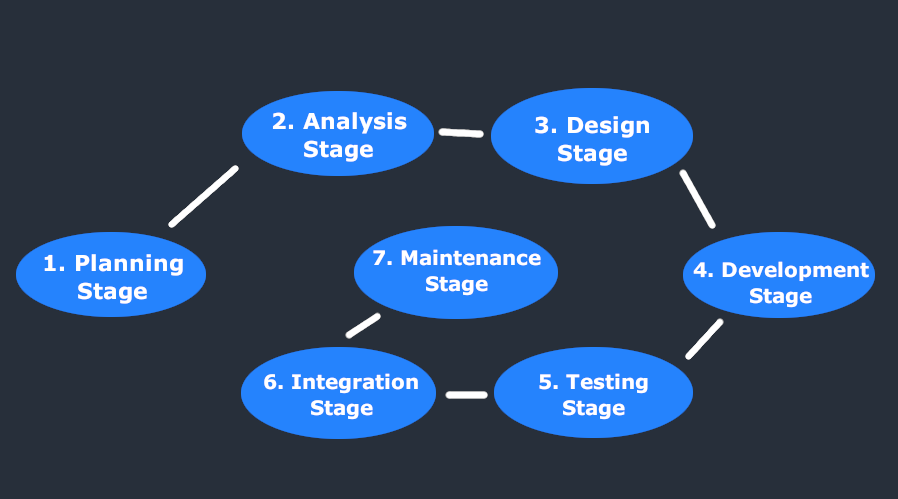


Fig 6: Phases of SDLC

### **7 Stages of the System Development Life Cycle:**

### There are seven primary stages of the modern system development life cycle. Here’s a brief breakdown:

* Planning Stage
* Feasibility or Requirements of Analysis Stage
* Design and Prototyping Stage
* Software Development Stage
* Software Testing Stage
* Implementation and Integration
* Operations and Maintenance Stage

### Planning Stage:

Before we even begin with the planning stage, the best tip we can give you is to take time and acquire proper [understanding of app development life cycle.](https://clouddefense.ai/blog/understanding-app-development-life-cycle) The planning stage (also called the feasibility stage) is exactly what it sounds like: the phase in which developers will plan for the upcoming project. It helps to define the problem and scope of any existing systems, as well as determine the objectives for their new systems. By developing an effective outline for the upcoming development cycle, they'll theoretically catch problems before they affect development. And help to secure the funding and resources they need to make their plan happen. Perhaps most importantly, the planning stage sets the project schedule, which can be of key importance if development is for a commercial product that must be sent to market by a certain time.

### Analysis Stage:

The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

Developers may:

* Define any prototype system requirements
* Evaluate alternatives to existing prototypes
* Perform research and analysis to determine the needs of end-users

Furthermore, developers will often create a software requirement specification or SRS document. This includes all the specifications for software, hardware, and network requirements for the system they plan to build. This will prevent them from overdrawing funding or resources when working at the same place as other development teams.

### Design Stage:

The design stage is a necessary precursor to the main developer stage. Developers will first

outline the details for the overall application, alongside specific aspects, such as its:

* User interfaces
* System interfaces
* Network and network requirements
* Databases

They’ll typically turn the SRS document they created into a more logical structure that can later be implemented in a programming language. Operation, training, and maintenance plans will all be drawn up so that developers know what they need to do throughout every stage of the cycle moving forward. Once complete, development managers will prepare a design document to be referenced throughout the next phases of the SDLC.

### Development Stage:

The development stage is the part where developers actually write code and build the application according to the earlier design documents and outlined specifications. This is where [Static Application Security Testing](https://clouddefense.ai/sast-static-application-security-testing) or SAST tools come into play. Product program code is built per the design document specifications. In theory, all of the prior planning and outlined should make the actual development phase relatively straightforward. Developers will follow any coding guidelines as defined by the organization and utilize different tools such as compilers, debuggers, and interpreters. Programming languages can include staples such as C++, PHP, and more. Developers will choose the right programming code to use based on the project specifications and requirements.

### Testing Stage:

Building software is not the end. Now it must be tested to make sure that there aren’t any bugs and that the end-user experience will not negatively be affected at any point. During the testing stage, developers will go over their software with a fine-tooth comb, noting any bugs or defects that need to be tracked, fixed, and later retested. It’s important that the software overall ends up meeting the quality standards that were previously defined in the SRS document. Depending on the skill of the developers, the complexity of the software, and the requirements for the end-user, testing can either be an extremely short phase or take a very long time.

### Implementation and Integration Stage:

After testing, the overall design for the software will come together. Different modules or designs will be integrated into the primary source code through developer efforts, usually by leveraging training environments to detect further errors or defects. The information system will be integrated into its environment and eventually installed. After passing this stage, the software is theoretically ready for market and may be provided to any end-users.

Maintenance Stage:

The SDLC doesn’t end when software reaches the market. Developers must now move into a maintenance mode and begin practicing any activities required to handle issues reported by end-users. Furthermore, developers are responsible for implementing any changes that the software might need after deployment.

# CHAPTER 5

**ARDUINO COMPONENTS**

## 5.1 ARDUINO

## Arduino is an open-source electronics platform that consists of both hardware and software components. It is designed for creating interactive electronic projects and prototypes by individuals, hobbyists, makers, and professionals. Arduino provides an accessible and user-friendly way to develop embedded systems and micro controllerbased applications, making it widely popular in the world of electronics and DIY projects.

## Here are key components of the Arduino platform:

## Hardware: Arduino boards are the physical devices that form the core of the platform. These boards come in various models, with the most common being the Arduino Uno. Arduino boards are built around microcontrollers, such as the Atmel AVR family, and include input/output pins for connecting sensors, actuators, and other electronic components.

## Software: Arduino IDE (Integrated Development Environment) is a software development environment that allows users to write, compile, and upload code to Arduino boards. It uses a simplified version of the C/C++ programming language and includes a set of libraries for interacting with hardware components.

## Programming Language: Arduino uses a simplified version of C/C++ to create code for controlling the hardware. This makes it accessible to both beginners and experienced programmers.

## 

## 5.2 COMPONENTS IN ARDUINO :

Arduino boards consist of various components that collectively make up the hardware platform for building electronic projects and prototypes. Here are the main components commonly found on an Arduino board:

Microcontroller:

The Arduino Uno R3 is powered by the ATmega328P microcontroller, which is at the heart of the board. This microcontroller has 32KB of flash memory for storing your program, 2KB of SRAM for data storage, and 1KB of EEPROM for non-volatile data storage. It operates at a clock speed of 16 MHz.

Digital I/O Pins:

The board features 14 digital input/output pins (labeled D0 to D13) that can be used to connect to various external components like sensors, LEDs, and other digital devices. These pins can be configured as either input or output.

Analog Input Pins:

The Arduino Uno R3 has 6 analog input pins (labeled A0 to A5). These pins allow you to read analog voltage values, making it suitable for connecting analog sensors and other devices.

Power Supply Options:

USB: You can power the Arduino Uno via the USB port, which also provides a communication interface to your computer for programming and serial communication.

DC Power: The board can be powered through the DC barrel jack. It accepts an input voltage in the range of 7V to 12V. An onboard voltage regulator ensures that the board operates at 5V even when powered with higher voltages.

Voltage Regulator:

The Arduino Uno R3 includes a voltage regulator that provides a stable 5V output from the input voltage. This ensures a consistent power supply for your projects.

Reset Button:

The reset button allows you to restart your program or reset the board without disconnecting the power source. It's a handy feature during development and debugging.

LEDs:

Power LED: This LED indicates when the board is powered.

Pin 13 LED (L LED): Connected to digital pin 13, this LED is often used for basic debugging and testing digital output.

USB-to-Serial Converter: The board uses a USB-to-Serial converter chip (usually an ATmega16U2 or ATmega8U2) to establish a serial communication link between your computer and the board. This chip handles USB communication, making it easier to upload sketches and communicate with your computer.

ICSP (In-Circuit Serial Programming) Header:

This header allows you to program the ATmega328P microcontroller with an external programmer, which can be useful for more advanced users.

Analog Reference (ARef):

The ARef pin allows you to set an external reference voltage for the analog inputs, which can be useful for more precise analog measurements.

EEPROM:

The ATmega328P microcontroller includes a small amount of EEPROM (Electrically Erasable Programmable Read-Only Memory) for data storage, which can be useful for saving configuration settings or other data that needs to be retained between power cycles.

**5.3 COMPONENTS USED IN AIR POLLUTION MONITORING SYSTEM ARE:**

* Arduino Uno R3
* Gas Sensors (MQ135)
* LCD Display
* Potentiometer
* Resistor
* Wi-fi Module (ESP8266)
* Bread Board
* Buzzer

**Arduino Uno R3:**

* Description: The Arduino Uno R3 is a popular microcontroller board based on the ATmega328P microcontroller. It features digital and analog input/output pins, a USB connection for programming and power, and is widely used for various electronic projects.
* Use: It serves as the central controller for your project, responsible for reading sensor data, displaying information on the LCD, and controlling other components.

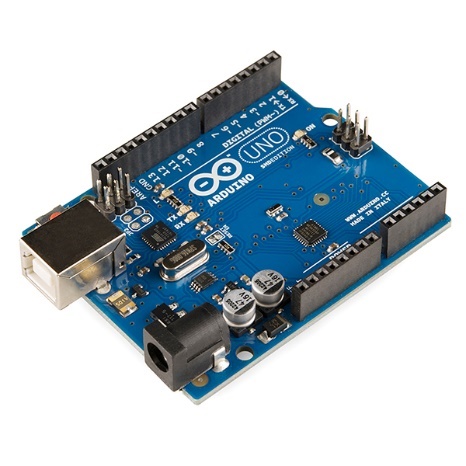


Fig: Arduino board

**Gas Sensors (MQ135):**

* Description: The MQ135 gas sensor is a versatile air quality sensor capable of detecting various gases, including carbon dioxide (CO2), smoke, alcohol, benzene, and ammonia (NH3).
* Use: The MQ135 sensor is used to measure air quality by detecting the concentration of harmful gases. It provides data to assess air pollution levels.



Fig: MQ135 Gas sensor

**LCD Display:**

* Description: An LCD (Liquid Crystal Display) is a visual output component that can display text and numbers. Common LCD displays used with Arduino are 16x2 or 20x4 character LCDs.
* Use: The LCD display is used to show air quality data, making it accessible and readable to users.



Fig: 16\*2 LCD display

**Potentiometer:**

* Description: A potentiometer (pot) is a variable resistor with a knob or dial that can be turned to adjust its resistance.
* Use: In your project, a potentiometer can be used for user input or calibration purposes. For example, it can adjust the contrast or brightness of the LCD display.



Fig: Potentiometer

**Resistor:**

* Description: Resistors are passive electronic components that restrict the flow of electrical current.
* Use: Resistors are used in electronic circuits for various purposes, such as current limiting, voltage division, and pull-up/pull-down configurations to ensure proper operation of components like sensors and LEDs.

****

Fig: Resistors

**Wi-Fi Module (ESP8266):**

* Description: The ESP8266 is a Wi-Fi module that provides wireless internet connectivity to microcontroller projects. It features built-in Wi-Fi capabilities and can be programmed using Arduino IDE.
* Use: The ESP8266 module enables your project to connect to the internet and transmit air quality data to a web server for remote monitoring and alerts.

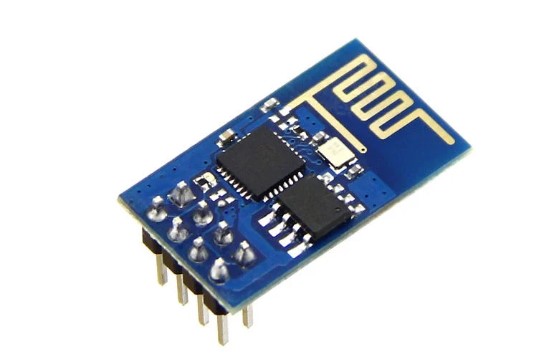
****

Fig: ESP8266 WIFI-module

**Breadboard:**

* Description: A breadboard is a prototyping board with a grid of holes and metal connectors that allow you to build and test electronic circuits without soldering.
* Use: The breadboard is used for assembling and testing your project's electronic components and connections.

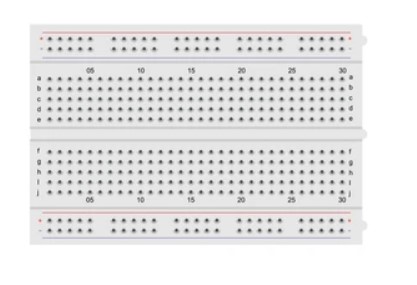
****

Fig: Bread Board

**Buzzer:**

* Description: A buzzer is an audio output component that produces sound when an electrical current is applied.
* Use: The buzzer can be used to generate audible alarms or alerts when air quality levels surpass safe thresholds, providing a warning to users.



Fig: Buzzer

**CONNECTIONS REQUIRED:**

* LCD RS pin to digital pin 12
* LCD Enable pin to digital pin 11
* LCD D4 pin to digital pin 5
* LCD D5 pin to digital pin 4
* LCD D6 pin to digital pin 3
* LCD D7 pin to digital pin 2
* LCD R/W pin to ground
* LCD VSS pin to ground
* LCD VCC pin to 5V
* 10K resistor:
* Ends to +5V and ground
* Wiper to LCD VO pin (pin 3)

# CHAPTER 6

**ARDUINO CIRCUIT**

**6.1 ARDUINO:**

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It consists of a programmable circuit board often referred to as a microcontroller and a development environment, or IDE (Integrated Development Environment), used to write software for the board.

Here are some key points about Arduino:

Open-source: The hardware and software specifications of Arduino are open-source, meaning that the designs and codes are freely available for the public to study, modify, and distribute.

Micro controller: The core of the Arduino platform is a micro controller — typically, the Atmel AVR series, such as the ATmega328, ATmega2560, etc. Other variants, including those with ARM processors, have also been introduced.

Simplified programming environment: Arduino provides a simple and easy-to-use programming environment that allows you to write code and upload it to the board. The Arduino IDE is used for writing and uploading the code to the Arduino board.

Versatile use cases: Arduino can be used to develop interactive objects, taking inputs from various sensors and controlling lights, motors, and other physical outputs. It's widely used in creating a variety of projects such as robotics, home automation, environmental monitoring, and much more.

Expansion capabilities: Arduino is expandable through its various shields and modules, which allow for the addition of functionalities such as WiFi, Bluetooth, motor control, and more.

Community support: Arduino has a large and active community that contributes to the platform by creating libraries, sharing projects, and providing support for beginners and experienced users alike.

Overall, Arduino is popular due to its simplicity and versatility, making it an excellent platform for beginners and professionals to create interactive and innovative electronic projects.

**Arduino Board Representation:**

The Arduino Uno is one of the most popular Arduino boards, equipped with a variety of pins that serve different functions. Here's a description of the pins found on the Arduino Uno board:

Digital Pins (0-13): These pins can be used for both digital input and digital output. They can read signals or provide a digital output of either HIGH (5 volts) or LOW (0 volts).

Analog Pins (A0-A5): The Arduino Uno has six analog input pins that can also be used as digital pins. These

pins can read analog signals from sensors or provide digital outputs similar to the digital pins.

Power Pins:

5V and 3.3V Pins: These pins provide regulated power output at 5 volts and 3.3 volts, respectively.

Vin: This pin can be used to provide an external power supply to the board.

GND (Ground) Pins: These pins are used as the ground reference for the power supply and signals.

PWM (Pulse Width Modulation) Pins (3, 5, 6, 9, 10, and 11): These pins can provide a simulated analog output by using a technique called PWM. They are capable of simulating an analog output with varying voltage levels.

Reset Pin: This pin is used to restart the program running on the Arduino board.

Communication Pins:

TX/RX Pins (0 and 1): These pins are used for serial communication with other devices.

I2C Pins (A4, A5): These pins are used for I2C communication, which allows the Arduino to communicate with multiple devices using only two wires.

SPI Pins (10, 11, 12, and 13): These pins are used for Serial Peripheral Interface (SPI) communication, allowing the Arduino to communicate with multiple devices using a master-slave architecture.

Understanding the functions of these pins is crucial for successfully interfacing the Arduino Uno with various sensors, actuators, and other electronic components.

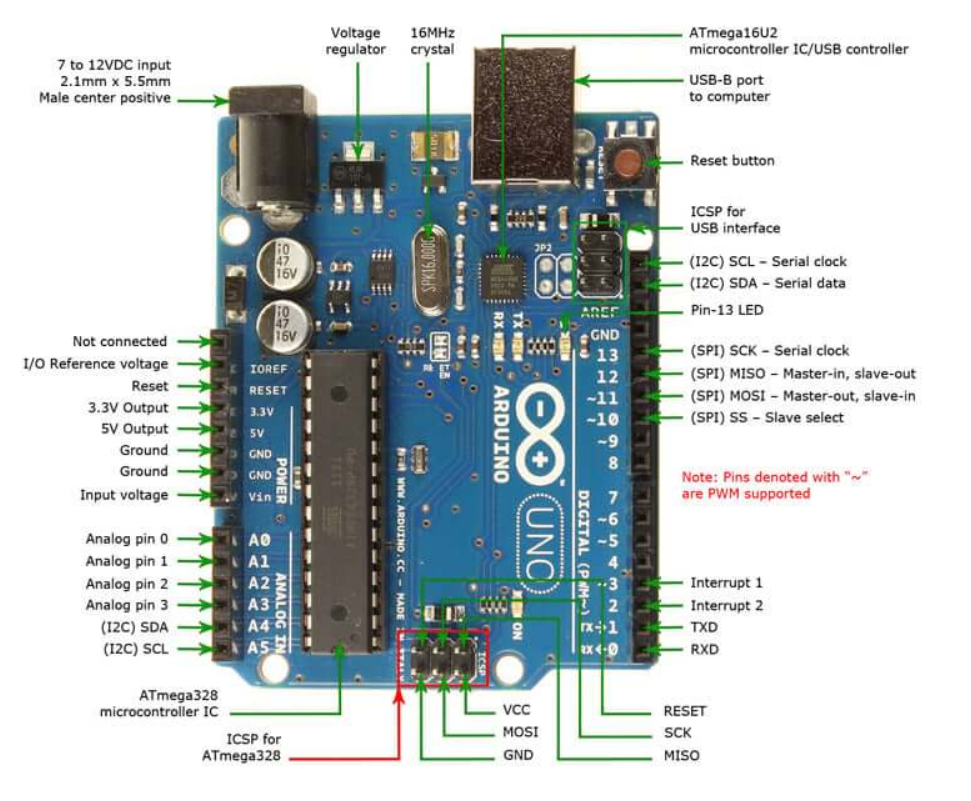


Fig 7: Arduino Board Representation

**6.2** **LANGUAGES USED IN ARDUINO UNO DEVELOPMENT :**

When working with the Arduino Uno, you can use several programming languages and environments to develop applications for the board. The primary language used for Arduino development is a simplified version of C++ that is supported by the Arduino IDE (Integrated Development Environment). Here are the main languages and tools used for Arduino Uno development:

Arduino Programming Language (based on C++): The Arduino programming language is a simplified version of C and C++. It provides easy-to-use functions and libraries to interface with the hardware components, making it accessible for beginners and experienced developers alike.

Arduino IDE: The Arduino Integrated Development Environment is the primary software tool used for writing and uploading code to the Arduino board. It provides a code editor, compiler, and a simple interface for managing libraries and board settings.

C and C++: Since the Arduino programming language is based on C++, developers familiar with C or C++ can also directly program the Arduino Uno using these languages. They can access the full power of C++ to develop more complex and advanced applications.

MicroPython: Some variations of Arduino boards, such as the Arduino Nano 33 IoT and Arduino MKR series, support MicroPython, which is a subset of the Python 3 programming language optimized to run on microcontrollers and constrained systems.

JavaScript (Node.js): For IoT projects, it's also possible to use JavaScript and Node.js to communicate with the Arduino Uno through a serial connection or using a compatible board like the Arduino Nano 33 IoT, which supports JavaScript.

While the primary language for Arduino development is a simplified version of C++, the board's flexibility and compatibility with various programming languages make it a versatile platform for a wide range of applications and projects.

**6.3** **Modules Used In Arduino:**

Arduino is compatible with a wide range of modules and sensors, allowing users to easily expand the capabilities of their projects. These modules enable interaction with various types of input and output devices, such as sensors, displays, communication modules, motors, and more. Some of the commonly used modules in Arduino projects include:

Sensors:

Temperature and humidity sensor (DHT11, DHT22)

Ultrasonic sensor (HC-SR04)

Light sensor (LDR, photo resistor)

Motion sensor (PIR sensor)

Gas sensor (MQ series)

Accelerometer and gyroscope (MPU-6050)

Actuators:

Servo motor

DC motor

Stepper motor

Relay module

Communication Modules:

Wi-Fi modules (ESP8266, ESP32)

Bluetooth modules (HC-05, HC-06)

RFID modules

GSM modules (SIM900, SIM800)

Displays:

LCD display (16x2, 20x4)

OLED display

Seven-segment display

TFT display

Wireless Communication:

RF transmitters and receivers

LoRa modules

Zigbee modules

Infrared (IR) communication modules

Input Devices:

Keypad module

Touchscreen module

Potentiometer

Other Modules:

Real-time clock (RTC) modules

GPS modules

SD card modules

Audio modules

Motor driver modules

These modules can be easily integrated into Arduino projects using the appropriate libraries and code examples available in the Arduino IDE. They significantly expand the capabilities of the Arduino platform, allowing users to create a wide variety of projects in fields such as home automation, robotics, IoT, and more.

**CIRCUIT DIAGRAM:**

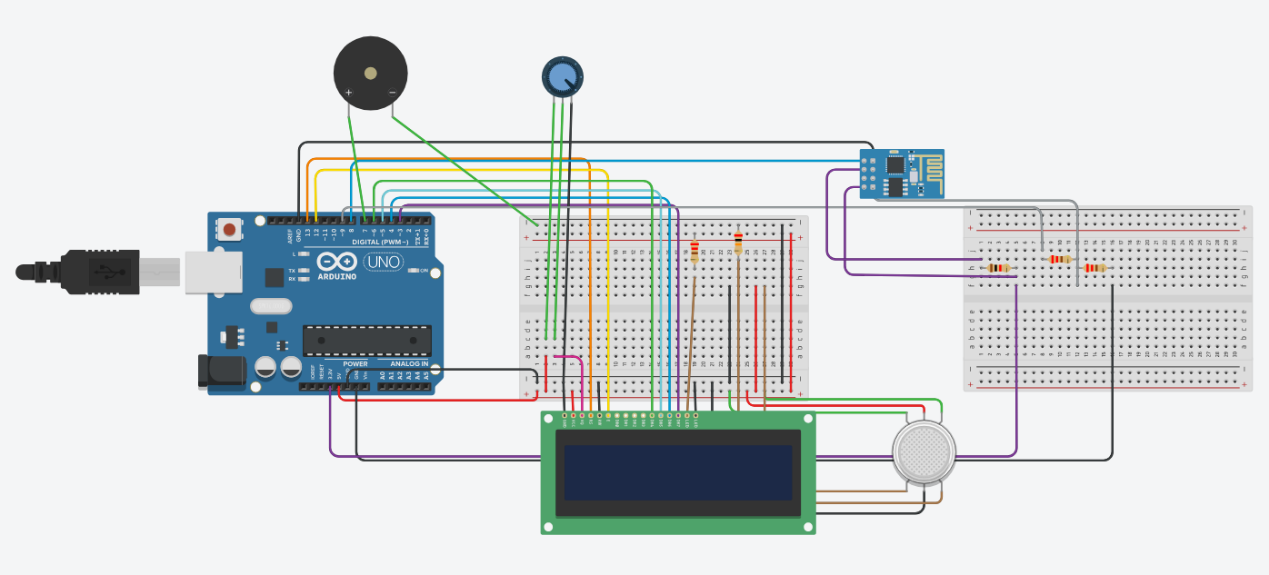


Fig 8: Circuit Diagram

## CHAPTER 7

## IMPLEMENTATION

**STEPS TO FOLLOW FOR IMPLEMENTING CODE IN ARDUINO IDE:**

Running code in the Arduino IDE involves a few simple steps. Here's a step-by-step guide on how to run code using the Arduino IDE:

Install Arduino IDE:

If you haven't already, download and install the Arduino IDE from the official Arduino website (https://www.arduino.cc/en/software) for your operating system (Windows, macOS, or Linux).

Connect Your Arduino Board:

Plug your Arduino board (e.g., Arduino Uno) into your computer using a USB cable. Ensure that the board is recognized by your computer, and the necessary drivers are installed (often automatic).

Open Arduino IDE:

Launch the Arduino IDE that you installed in Step 1.

Select Your Board:

Go to the "Tools" menu and select "Board." Choose the appropriate Arduino board you're using. For example, if you're using an Arduino Uno, select "Arduino Uno."

Select the COM Port:

In the same "Tools" menu, go to "Port" and select the COM port that your Arduino is connected to. On Windows, it will look like "COMX" (e.g., COM3); on macOS and Linux, it may appear as "/dev/ttyUSBX" or something similar.

Write or Open Your Code:

You can either write your code directly in the Arduino IDE or open an existing sketch (Arduino project) by going to "File" > "Open." If you're writing code from scratch, type it into the Arduino IDE's code editor.

Verify/Compile Your Code:

Click the checkmark icon (or "Verify" under "Sketch") to compile your code. This step checks for any syntax errors in your code. If there are errors, they will be displayed in the message area at the bottom of the IDE. Correct any errors before proceeding.

Upload Your Code:

Click the right arrow icon (or "Upload" under "Sketch") to upload your code to the Arduino board. This step compiles your code once again and then uploads it to the connected Arduino. You should see a progress bar during the upload process.

Monitor the Serial Monitor (Optional):

If your code uses the Serial Monitor for debugging or displaying information, you can open it by clicking the

magnifying glass icon (or "Serial Monitor" under "Tools"). This allows you to view serial output from your Arduino.

Observe the Arduino Board:

Your code will run on the Arduino board, and you should see any expected behavior based on your code's logic. For example, if you have programmed it to blink an LED, you'll see the LED blinking.

Debug and Refine (If Necessary):

If your code doesn't work as expected, use the Serial Monitor and other debugging techniques to identify and address any issues. Make changes to your code as needed and repeat the upload process until your project behaves as intended.

Disconnect and Reconnect (If Necessary):

If you need to upload new code or make changes to your project, you may need to disconnect and reconnect your Arduino board to the computer before repeating the steps. The IDE provides a user-friendly interface for writing, compiling, and uploading code to your microcontroller, making it accessible for both beginners and experienced developers.

## The major code for implementation:

## 

## #include <LiquidCrystal.h>

## #include <SoftwareSerial.h>

## // initialize the library with the numbers of the interface pins

## LiquidCrystal lcd(13, 12, 6, 5, 4, 3); // LCD connections

## /\*

## D13 ==> RS

## GND ==> R/W

## D12 ==> Enable

## D6 ==> DB4

## D5 ==> DB5

## D4 ==> DB6

## D3 ==> DB7

## 

## \*/

## float t=0;

## char data = 0;

## String apiKey = "XBQDVORXXGAROWDW"; // Write API key

## // connect 8 to TX of ESP

## // connect 9 to RX of ESP

## SoftwareSerial ser(8,9); // RX, TX

## // Buzzer Pin

## const int buzzerPin = 7;

## void setup()

## {

## // enable debug serial

## Serial.begin(9600); // serial data transmission at Baudrate of 9600

## // enable software serial

## ser.begin(9600);

## lcd.begin(16, 2); // to initialize LCD

## // Set up the buzzer pin

## pinMode(buzzerPin, OUTPUT);

## lcd.setCursor(0,0);

## lcd.print(" Welcome");

## lcd.setCursor(0,1);

## lcd.print(" To ");

## delay(3000);

## lcd.clear();

## lcd.setCursor(0,0);

## lcd.print(" AIR");

## lcd.setCursor(0,1);

## lcd.print("QUALITY MONITOR");

## delay(3000);

## ser.println("AT"); // Attenuation

## delay(1000);

## ser.println("AT+GMR"); // To view version info for ESP-01 output: 00160901 and ESP-12 output: 0018000902-AI03

## delay(1000);

## ser.println("AT+CWMODE=3"); // To determine WiFi mode

## /\*

## 1 = Station mode (client)

## 2 = AP mode (host)

## 3 = AP + Station mode (ESP8266 has a dual mode)

## \*/

## delay(1000);

## ser.println("AT+RST"); // To restart the module

## delay(5000);

## ser.println("AT+CIPMUX=1"); // Enable multiple connections

## /\*

## 0: Single connection

## 1: Multiple connections (MAX 4)

## \*/

## delay(1000);

## String cmd="AT+CWJAP=\"SSID\",\"PASSWORD\""; // connect to Wi-Fi

## ser.println(cmd);

## delay(1000);

## ser.println("AT+CIFSR"); // Return or get the local IP address

## delay(1000);

## lcd.clear();

## lcd.setCursor(0,0);

## lcd.print(" WIFI");

## lcd.setCursor(0,1);

## lcd.print(" CONNECTED");

## }

## 

## void loop()

## {

## delay(1000);

## t = analogRead(A0); // Read sensor value and store in variable t

## Serial.print("Airquality = ");

## Serial.println(t);

## lcd.clear();

## lcd.setCursor (0, 0);

## lcd.print ("Air Qual: ");

## lcd.print (t);

## lcd.print (" PPM ");

## 

## lcd.setCursor (0,1);

## if (t <= 500) {

## lcd.print("Fresh Air");

## Serial.print("Fresh Air ");

## }

## else if (t >= 500 && t <= 1000) {

## lcd.print("Poor Air");

## Serial.print("Poor Air");

## }

## else if (t >= 1000) {

## lcd.print("Very Poor");

## Serial.print("Very Poor");

## 

## // Activate the buzzer for high air pollution

## tone(buzzerPin, 1000, 1000); // Add your desired tone and duration

## delay(1000); // Wait for a second

## noTone(buzzerPin); // Stop the buzzer

## }

## 

## delay(10000);

## lcd.clear();

## lcd.setCursor(0,0);

## lcd.print(" SENDING DATA");

## lcd.setCursor(0,1);

## lcd.print(" TO CLOUD");

## esp\_8266();

## }

## void esp\_8266()

## {

## // TCP connection AT+CIPSTART=4,"TCP","184.106.153.149",80

## String cmd = "\nAT+CIPSTART=4,\"TCP\",\""; // Establish TCP connection

## cmd += "184.106.153.149"; // api.thingspeak.com

## cmd += "\",80";

## ser.println(cmd);

## Serial.println(cmd);

## if(ser.find("Error")) {

## Serial.println("AT+CIPSTART error");

## return;

## }

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

## getStr += apiKey;

## getStr +="&field1=";

## getStr +=String(t);

## getStr += "\r\n\r\n";

## // send data length

## cmd = "AT+CIPSEND="; // Send data AT+CIPSEND=id,length

## cmd += String(getStr.length());

## ser.println(cmd);

## Serial.println(cmd);

## delay(1000);

## ser.print(getStr);

## Serial.println(getStr);

## // thingspeak needs 16 sec delay between updates

## delay(17000);

## }

## USAGE OF SERIAL MONITOR:

## The Serial Monitor is a valuable tool in the Arduino Integrated Development Environment (IDE) that allows you to communicate with your Arduino board. It facilitates both the input and output of data, enabling you to easily debug and monitor your Arduino projects. Here are some common use cases of the Serial Monitor:

## Debugging: You can use the Serial Monitor to print out debug messages or variable values from your Arduino code. By using the Serial.print() or Serial.println() functions in your code, you can send important information to the Serial Monitor for real-time monitoring and debugging.

## Data Visualization: The Serial Monitor can display sensor readings, output values, or any other data in real-time. It enables you to observe the behavior of your Arduino project and the changes in the values of variables as the code executes.

## Interaction: You can use the Serial Monitor to interact with your Arduino board. For instance, you can send commands from the Serial Monitor to control specific actions or parameters of your project. This is particularly useful for projects involving user input or configuration settings.

## Troubleshooting: If your project encounters issues, the Serial Monitor can help you identify and troubleshoot problems. By printing relevant data and status information, you can analyze the behavior of your project and pinpoint potential errors in your code or hardware connections.

## Data Logging: The Serial Monitor can be used to log data from your Arduino project onto your computer. By logging sensor readings or other crucial data, you can analyze the performance of your project over time and make informed decisions based on the collected data.

## To open the Serial Monitor in the Arduino IDE, click on the magnifying glass icon in the top right corner or navigate to Tools > Serial Monitor. Ensure that the baud rate in the Serial Monitor matches the baud rate specified in your Arduino code to ensure proper communication between your Arduino board and the Serial Monitor.

**DISPLAY OF SERIAL MONITOR CODE:**

The Serial Monitor in the Arduino IDE displays data transmitted from the Arduino board to the computer via the Serial.print(), Serial.println(), or Serial.write() functions in your Arduino sketch. It can also accept input from the user, allowing you to interact with the Arduino board. Below is an example of how the Serial Monitor might appear, displaying information sent from the Arduino:

**CODE FOR SERIAL MONITOR:**

lcd.setCursor (0,1);

if (t <= 500) {

lcd.print("Fresh Air");

Serial.println("Fresh Air");

}

else if (t >= 500 && t <= 1000) {

lcd.print("Poor Air");

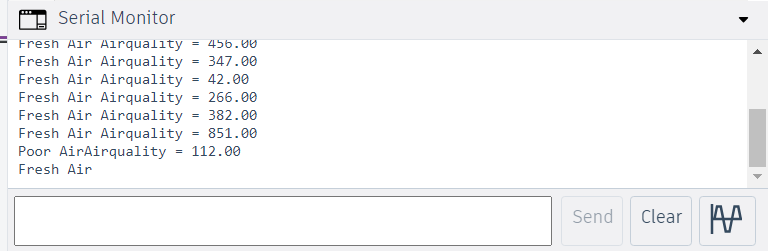
Serial.println("Poor Air");

}

else if (t >= 1000) {

lcd.print("Very Poor");

}



## CHAPTER 8

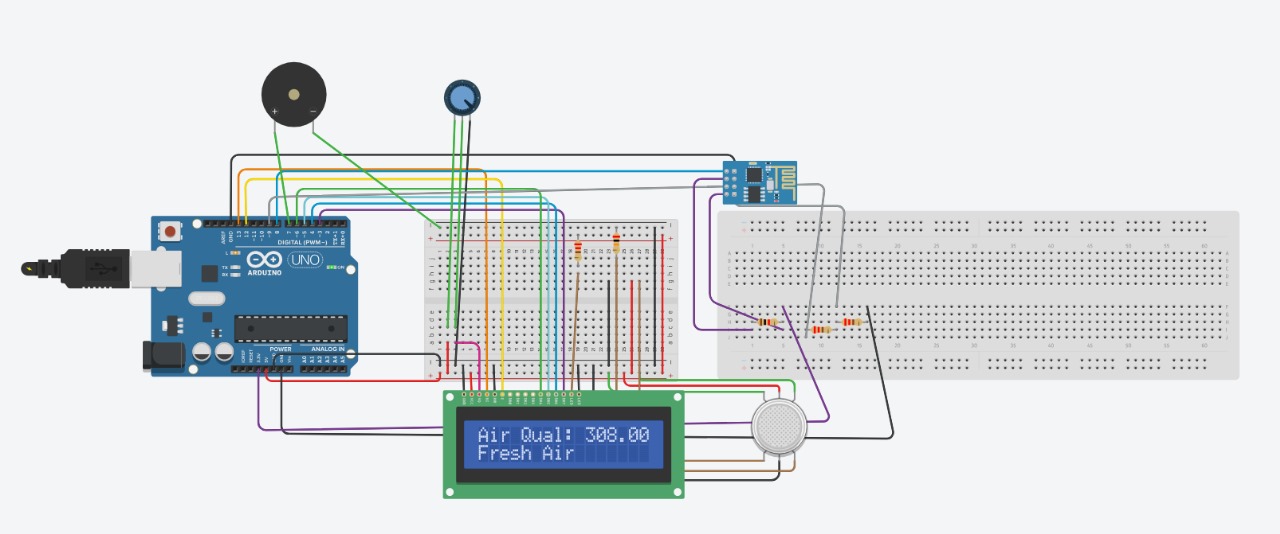
## TESTING

## CIRCUIT MODEL:

## 

Fig 9: Circuit model

#### 8.1 TEST CASE FOR DETECTING FRESH AIR:

Fig 10: Test case for detecting fresh air

**8.2 TEST CASE FOR DETECTING POOR AIR:**

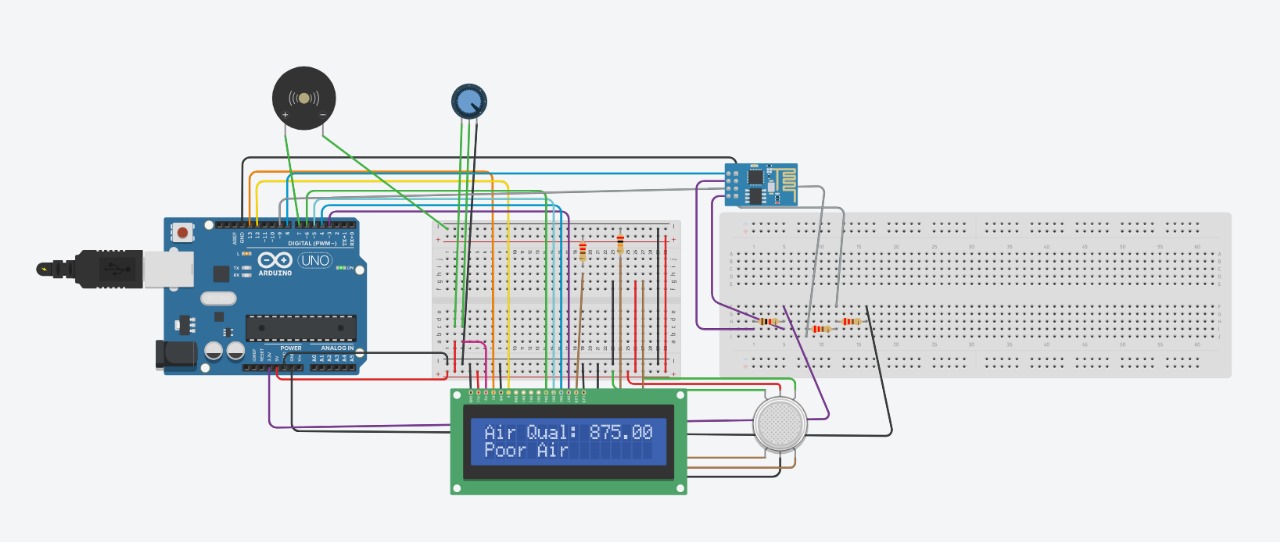
****

Fig 11: Test case for detecting poor air

#### 8.3 TEST CASE FOR DETECTING VERY POOR AIR:

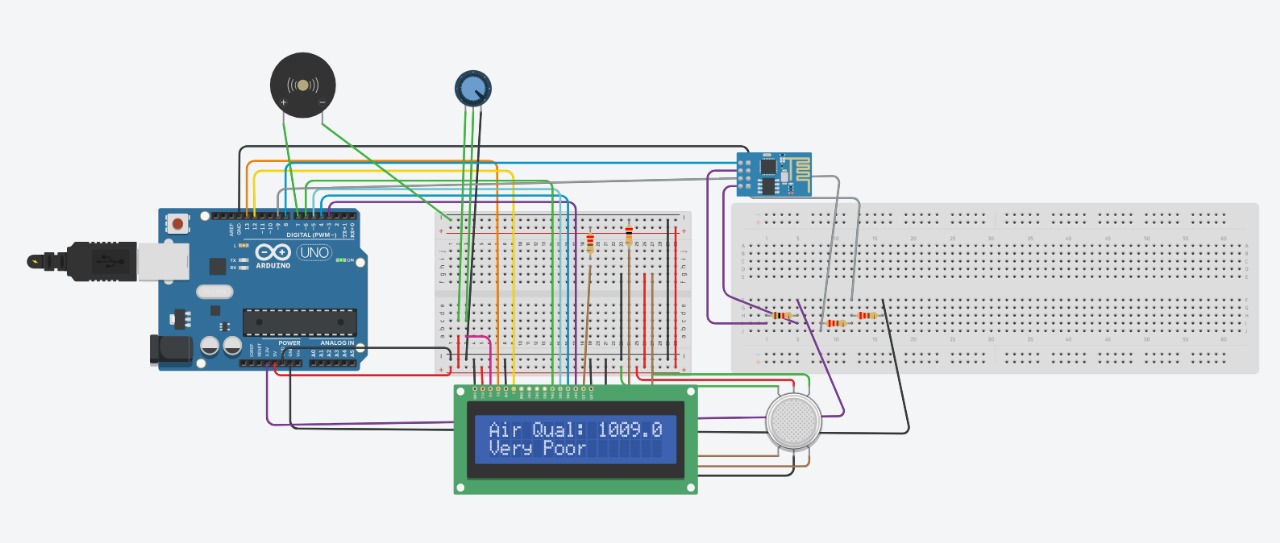


Fig 12: Test case for detecting very poor air

**8.4 TEST CASE FOR SENDING DATA TO CLOUD:**

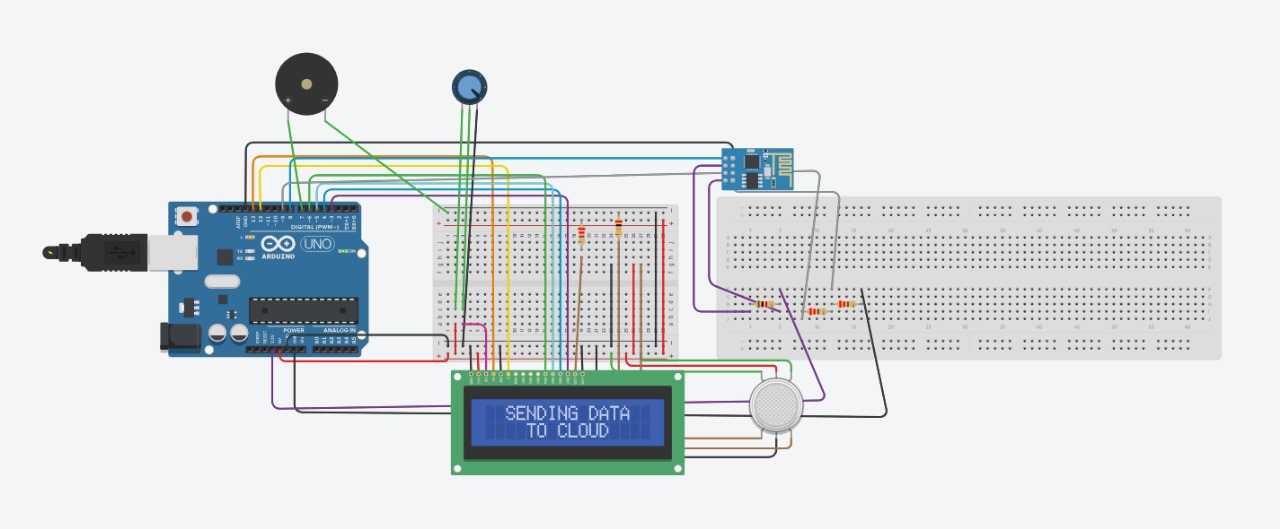
****

Fig 13: Test case for sending data to cloud

**8.5 TEST CASE FOR SHOWING CONNECTIVITY OF WI-FI MODULE:**

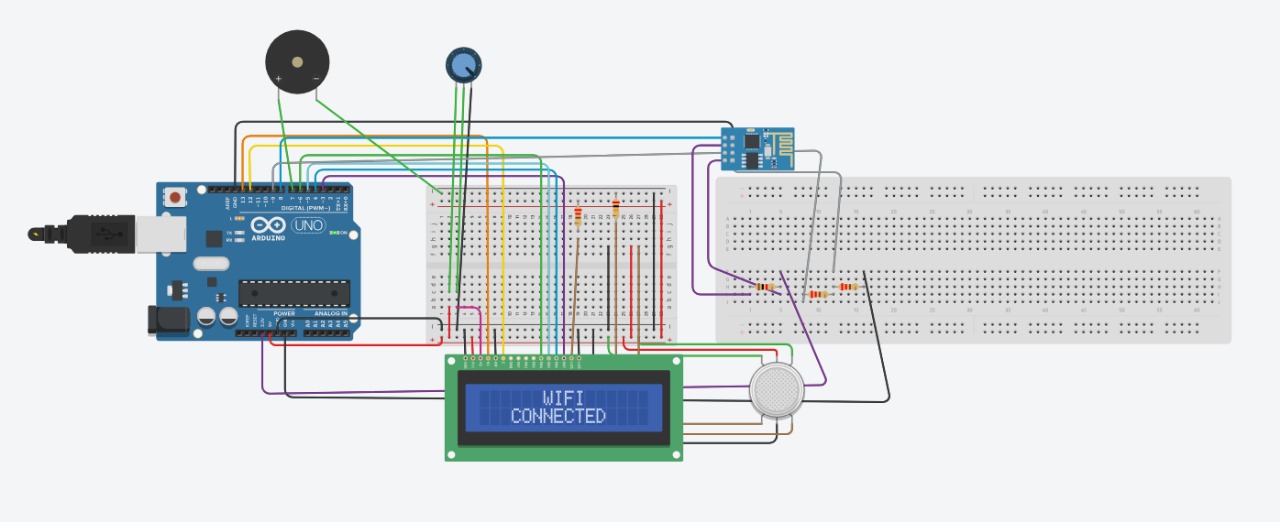


Fig 14: Test case for showing connectivity of wi-fi module

**8.6 OVERALL TEST CASES:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **FUNCTIONAL TEST CASE** | **POSITIVE/NEGATIVE** |
| **1** | **The buzzer activates while power supply is given to Arduino** | **Negative** |
| **2** | **All the system is connected to wi-fi module** | **Positive** |
| **3** | **Buzzer beeps for longer time when air quality is low** | **Negative** |
| **4** | **After every detection the data is sent to cloud** | **Positive** |
| **5** | **Buzzer beeps continuously, if the air is very poor** | **Positive** |
| **6** | **Buzzer doesn’t beep, if the air is fresh** | **Negative** |

Fig 15: Overall test cases

# CHAPTER 9

**CONCLUSION AND FUTURE SCOPE**

## 9.1 CONCLUSION

In this the IoT-Based Air Pollution Monitoring System introduced in this abstract is a vital and timely endeavor in the face of escalating air pollution concerns. By employing cutting-edge technology and the versatile MQ135 sensor, this project endeavors to create a robust system capable of real-time air quality monitoring and immediate alarms when pollution levels exceed safety thresholds. This comprehensive solution has the potential to revolutionize how we perceive and manage air quality.

This system holds promise for significant future applications. The data collected by this system has the potential to shape environmental policies, guide pollution control measures, and enhance global collaboration in addressing air quality challenges. In a world increasingly impacted by air pollution, this project is a beacon of innovation and hope, offering tangible steps towards safeguarding our planet's health and the well-being of future generations.

## In conclusion, the implementation of an air pollution monitoring system using Arduino represents a promising and cost-effective approach to addressing the growing concern of air quality worldwide. The Arduino platform, with its user-friendly programming environment and extensive sensor compatibility, offers a robust foundation for collecting, analyzing, and visualizing air quality data. This system empowers individuals and communities to actively monitor their local environments, enabling better-informed decisions and potentially leading to improved air quality management.

## Furthermore, the scalability of Arduino-based air pollution monitoring systems is a key advantage. Users can easily expand the system by integrating additional sensors, communication modules, and data storage options to meet specific project requirements. This adaptability makes it suitable for a wide range of applications, from personal monitoring devices to city-wide sensor networks.

## Lastly, the open-source nature of the Arduino platform fosters collaboration and knowledge sharing among developers and communities. This not only accelerates innovation but also ensures that air pollution monitoring solutions remain accessible and affordable to a broad audience. With Arduino, we are not only monitoring air quality; we are contributing to a collective effort to create a cleaner and healthier environment for everyone.

## 9.2 FUTURE SCOPE

* Enhanced Sensor Technology: Incorporating advanced sensor technology and artificial intelligence for more accurate and real-time detection of a wider range of air pollutants, enabling a deeper understanding of air quality.
* Integration with Smart Cities: Expanding the project's implementation to smart city initiatives, allowing for more extensive and interconnected air quality monitoring networks to support urban planning and environmental management.
* Mobile Applications: Developing mobile applications to provide users with convenient access to air quality data, personalized alerts, and data visualization, enhancing public awareness and engagement.
* Integration with IoT (Internet of Things): The convergence of Arduino with IoT technology will enable real-time remote monitoring and data transmission. This means that data from sensors can be collected, analyzed, and accessed from anywhere with an internet connection, allowing for more extensive and efficient monitoring.
* Mobile Applications: The development of mobile applications that connect to Arduino-based monitoring systems can provide users with easy access to air quality data on their smartphones. These apps could also send alerts or recommendations based on real-time data.
* Advanced Sensors: As sensor technology continues to advance, we can expect more accurate and sensitive sensors to become available. This will enable the monitoring of a broader range of pollutants and improved precision in measurements.
* Machine Learning and AI Integration: Machine learning and artificial intelligence can be used to analyze and predict air quality patterns based on historical data. This could lead to more accurate forecasting and better decision-making for pollution control.
* Smart Cities and Urban Planning: The data collected from these monitoring systems can inform urban planners and policymakers about pollution hotspots and trends. This information can be used to design cleaner and more sustainable cities.
* Environmental Activism and Advocacy: Arduino-based monitoring systems have the potential to empower communities to become actively involved in advocating for cleaner air. This technology can be a catalyst for environmental awareness and activism.
* Miniaturization and Wearable Devices: Future developments might lead to smaller, more portable air quality monitoring devices that individuals can wear or carry with them. This would enable personal exposure monitoring and facilitate data collection in various environments.
* Integration with Smart Home Systems: Arduino-based monitoring systems can be integrated with smart home systems to automatically control air purification devices or adjust ventilation based on real-time air quality data.
* Open Data Platforms: Establishing open data platforms for sharing air quality information can foster collaboration among researchers, policymakers, and citizens, leading to a more comprehensive understanding of air pollution and potential solutions.
* Environmental Regulation Compliance: Industries and businesses could implement Arduino-based systems to monitor and ensure compliance with environmental regulations, leading to more responsible industrial practices.

# CHAPTER 10

**REFERENCES**

* Al-Dahoud, Ali, et al. "Monitoring Metropolitan City Air-quality Using Wireless Sensor Nodes based on ARDUINO and XBEE."
* Rajkumar, M. Newlin, M. S. Sruthi, and V. Venkatesa Kumar. "IOT Based Smart System for Controlling Co2 Emission." (2017).
* Air\_pollution:https://en.wikipedia.org/wiki/Air\_pollution
* Air\_pollution\_detector:https://en.wikipedia.org/wiki/
* Air\_pollution\_sensor Potentiometer:https://en.wikipedia.org/wiki/Potentiometer.