Module Code & Module Title

CC5068NI– Cloud Computing & IoT

Assessment Weightage & Type

50% Group Coursework

Year and Semester

2022-23 Spring

Group Members

London Met ID	Student Name
21040608	Kishor Bamjan Tamang
21040612	Mingmar Lama
21040631	Stuti Pangeni
21040605	Binit Kunwar
22068713	Shimon Sharma

Coursework Due Date: 4th May 2023

Coursework Submission Date: 4th May 2023

Word Count: 3291

Acknowledgement

We would like to express our sincere gratitude to Mr. Sugat Man Singh, who has been an invaluable source of support and guidance throughout the research and writing of this paper and project. His expertise and insight into the latest internet technologies have been invaluable in helping us understand the complexities of various topics integrated and used in this project.

We would also like to thank Islington College for providing us with the resources and support needed to conduct this research. Without their assistance, this project would not have been possibly completed.

Finally, we would like to acknowledge the hard work and dedication of all the individuals who have helped us with the development, research and ideas in this project. Their efforts have added more value to this project of ours.

Thank you all for your invaluable contributions.

Sincerely,
Team 1 - N1

Abstract

This report outlines the development of Air monitoring and alerting system that provides real-time tracking of air quality to help prevent respiratory diseases caused by exposure to pollutants. After discussing the current scenario and problems associated with air pollution and outlines how this system aims to address these issues.

The system utilizes an Arduino UNO R3 micro-controller and two integrated sensors (i.e., MQ-135 and DHT-11) to detect air pollutants and temperature. The system also includes a Wi-Fi module i.e., ESP-32 that allows it to connect to ThingSpeak, a cloud platform used for storing, analysing, and visualizing the air quality data collected by the system and is designed to be cost-effective, easy to use, and effective in alerting users of poor air quality.

The report includes various test cases of the system's performance and accuracy and also provides a detailed overview of the hardware architecture, flowchart, and circuit diagram used in the system, as well as the software and hardware development process.

The report suggests additional features that can be added to improve the system in the future, such as integrating more sensors or implementing machine learning algorithms for more advanced data analysis. The air quality monitoring and warning system provides a simple and effective solution to the problem of air pollution and its effects on human health, with the added benefit of being able to remotely access the air quality data via ThingSpeak.

Table of Abbreviation

DHT-11	Digital Temperature and Humidity Sensor
ESP32	Espressif Systems Platform 32
IDE	Integrated Development Environment
I/O	Input/Output
IOT	Internet of Things
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MQ-135	Metal Oxide Semiconductor Gas Sensor
MCU	Micro-Controller Unit
UNO R3	Arduino UNO Revision 3
Wi-Fi	Wireless Fidelity

Table of Contents

Table of Figures

List of Tables

1. Introduction

1.1. Current Scenario

Air pollution is a critical issue that has gained global attention due to its significant impact on people's quality of life. Various studies have shown the mortality rate of individuals due to air pollution as shown in the figure below.

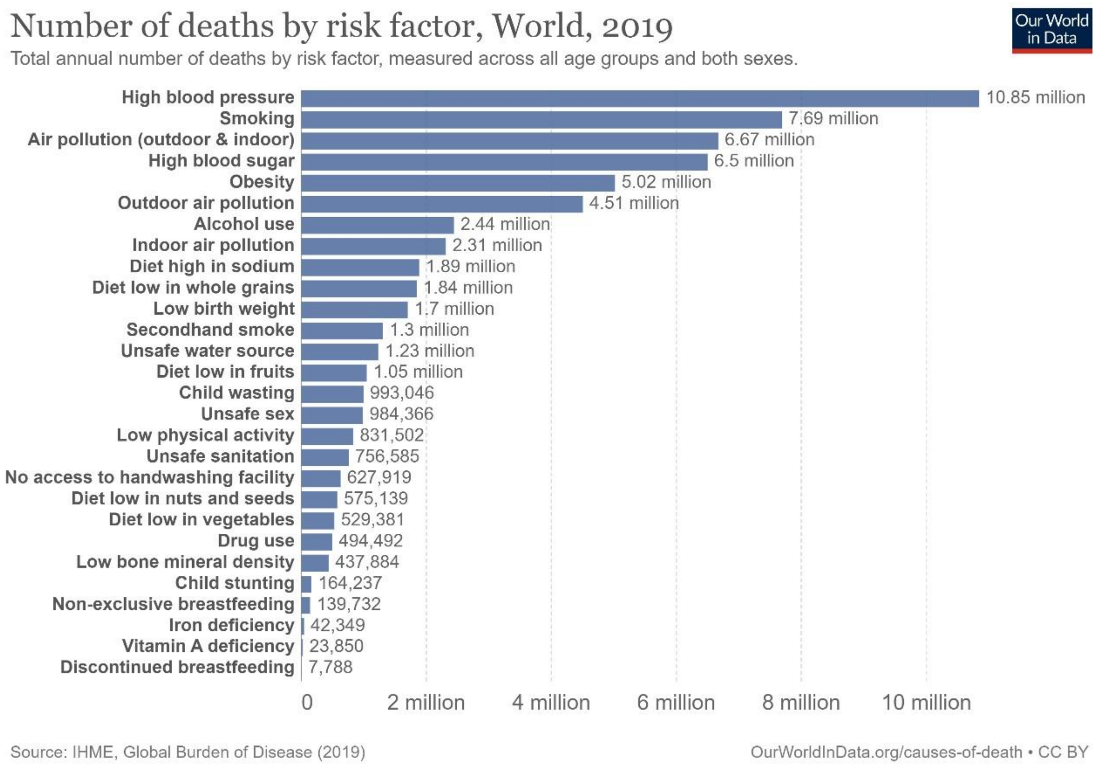


Figure 1 Number of deaths by various risk factor (Hannah Ritchie, 2021).

Air pollution is a critical issue that has garnered global attention in the recent times. Various research shows that 9 out of 10 individuals breathe air with high levels of pollutants, which significantly impacts their quality of life. Both indoor and outdoor air pollution are responsible for the deaths of approximately 7 million people worldwide each year, according to studies. Recent research in 2019 revealed that deaths resulting from air pollution, including both indoor and outdoor pollution, accounted for approximately 8.8 million deaths worldwide, 1.8 million more than reported by WHO in 2016. All these statistics and surveys demonstrate that air pollution is a significant issue in the present times.

1.2. Problem Statement and Project as a Solution

As mentioned earlier, air pollution has become a critical environmental concern that the world cannot ignore in recent years. The health of all living beings has been affected by air pollution, causing the emergence of new respiratory disorders, and adversely affecting the health of countless individuals. Moreover, traditional air monitoring stations that detect and measure air pollution are costly and challenging to install.

This project of air quality monitoring and alerting system offers the following solutions to the problems caused by air pollution:

- Real-time monitoring and detection of air quality, humidity and temperature, enabling timely action to be taken to address pollution.
- Identifying priority pollutants of air pollution.
- Providing a simple and economical but effective alternative to old air quality monitoring stations.
- Small, Portable and Cost-effective system.
- Visualizing the data in a chart in the cloud.

1.3. Aim and Objectives

The main aim of this project is to develop a system that provides real-time data on air quality, humidity and temperature that notifies users when the air quality gets worse and sends the data in the cloud, thereby addressing the issues mentioned earlier.

The primary objective of this project is to design a real-time air quality monitoring and alerting system that addresses the issues stated earlier. To achieve this aim, the following objectives have been set:

- Conduct research on the present state of air pollution, its negative effects, and potential solutions to mitigate the problem.
- Detect major air pollutants using MQ-135 gas sensor.
- Measure temperature and humidity using the DHT-11 sensor.
- Send data over to cloud for visualization.
- Test the air quality monitoring and alerting system using various test case scenarios.
- Analyse the outcomes of the system and identify future enhancements that can be added to it.

2. Background

2.1. System Overview

The system created in this project is an air quality monitoring and alerting system that utilizes the MQ-135 gas sensor to detect major air pollutants such as CO2, NO2, NH3, and other gases as well. DHT-11 sensor is also used to measure temperature and humidity.

The readings from both sensors are processed by an Arduino UNO R3 microcontroller, which sends the data to an ESP32 Wi-Fi module for wireless internet communication. The ESP32 module is programmed to upload the collected data to ThingSpeak, an IoT platform for storing, analysing, and visualizing data. When the air quality is found to be critical, the system triggers an alert through a buzzer and an LED. Our system can be categorized into four main components:

- Sensors (MQ-135 and DHT-11)
- Actuator (LCD, Green and Red LED, and Buzzer)
- Micro-Controller (UNO R3 and ESP32 Wi-Fi module)
- Cloud (ThingSpeak)

The system is based on an Arduino UNO R3 microcontroller that performs all logical, arithmetic, and control operations on the sensor inputs and produces output. The system also uses an ESP32 module and ThingSpeak cloud platform to upload the real-time air quality data. In addition to the LCD 16 x 2 screen that displays the temperature, humidity, and air quality condition, the system includes green and red LED lights. The green LED indicates good air quality, while the red LED warns the users of critical air quality conditions.

The system also includes various actuators. The buzzer produces a buzzing sound to alert users in addition to the visual notification provided by the LED lights, ensuring that users are notified when the air quality is detected to be critical.

2.2. Design Diagrams

2.2.1. Hardware Architecture

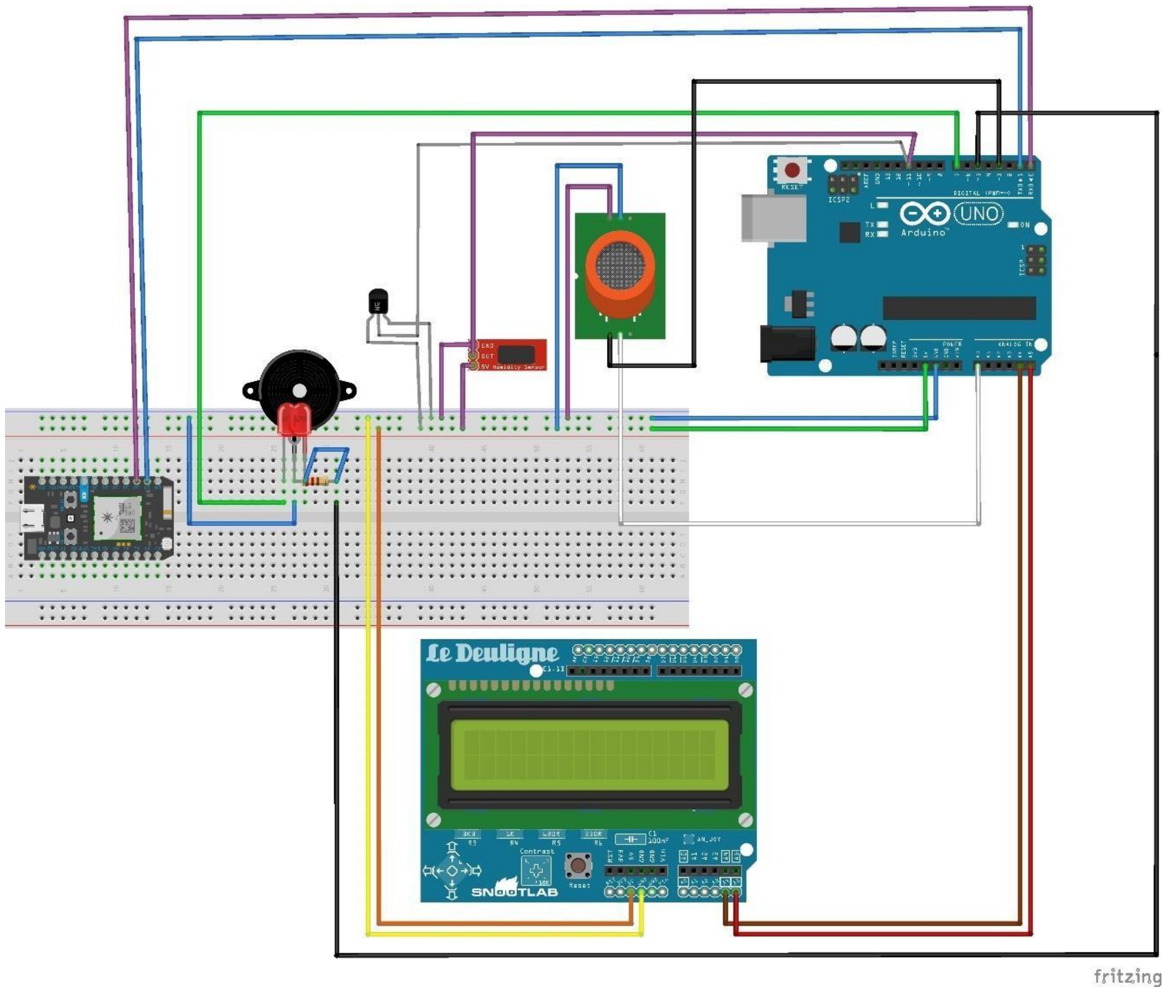


Figure 2 Hardware Architecture of our System.

2.2.2. Flowchart

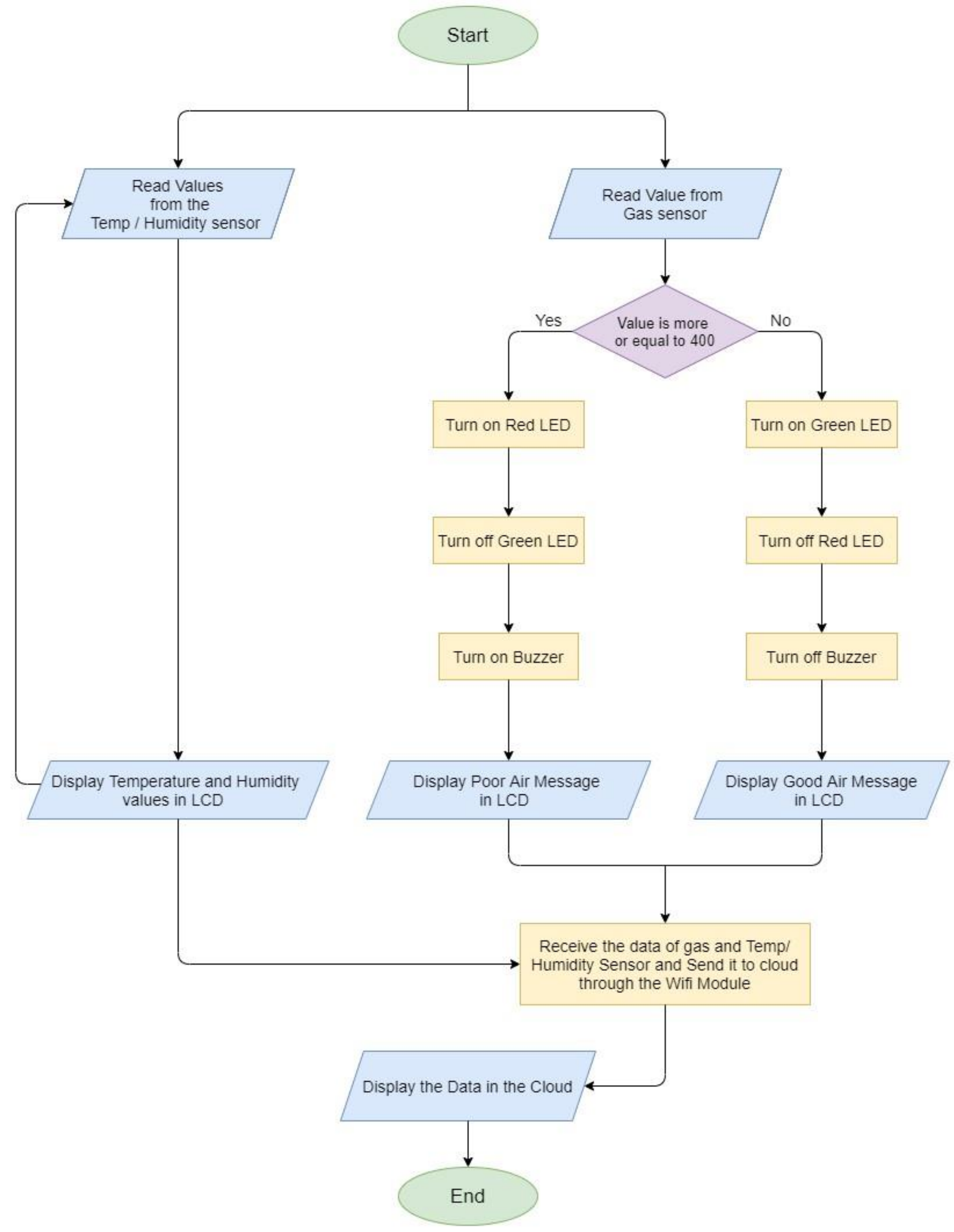

Start
Read Values from the Temp / Humidity sensor
Read Value from Gas sensor
Value is more or equal to 400
Yes
No
Turn on Red LED
Turn off Green LED
Turn on Buzzer
Turn on Green LED
Turn off Red LED
Turn off Buzzer
Display Temperature and Humidity values in LCD
Display Poor Air Message in LCD
Display Good Air Message in LCD
Receive the data of gas and Temp/ Humidity Sensor and Send it to cloud through the Wifi Module
Display the Data in the Cloud
End

Figure 3 Flowchart of our System

2.2.3. Circuit Diagram

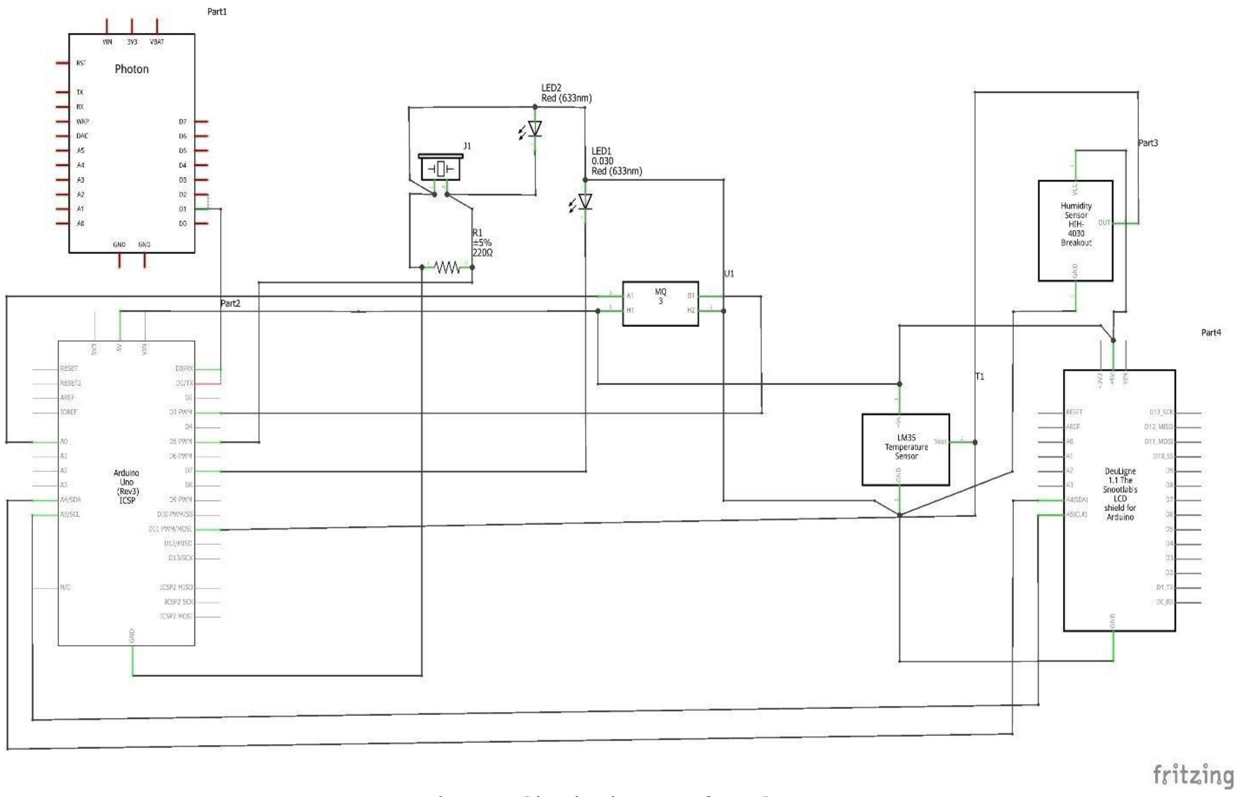


Figure 4 Circuit Diagram of our System

2.3. Requirement Analysis

The following is a brief description of the software and hardware resources used in the development of the system.

2.3.1. Software

A. Arduino IDE

The sketches for the system were developed using the Arduino IDE and uploaded to the Arduino UNO R3 board for this project after checking and debugging various coding errors in the code.

B. Draw.IO

Draw.io was used to create a diagram of the system, which included the different components such as sensors, microcontrollers, output devices, and the cloud platform. The diagram helped in visualizing the system architecture and the flow of data between the different components.

C. Fritzing

Most of the diagram used in this project were made by using fritzing and draw.io such as circuit diagram, hardware architecture and flow chart.

D. ThingSpeak

ThingSpeak was used to visualize the data collected by the system. The data collected by the system was sent to ThingSpeak's cloud platform for data visualization.

2.3.2. Hardware

A. Arduino UNO R3

Arduino UNO was used as the main controller to process the data from the sensors and to control the output devices, such as the buzzer and LED in our system. It played a crucial role in the system's functionality, allowing us to read the sensor data and process it into meaningful information for the user.

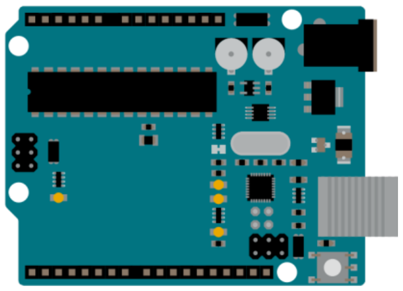


Figure 5 Visual Diagram of Arduino UNO R3 (Arduino, 2023).

B. Bread Board

The breadboard was used to assemble and test the electronic components such as sensors, actuators, and other devices of our system. It was mainly used for extended connectivity for our hardware.

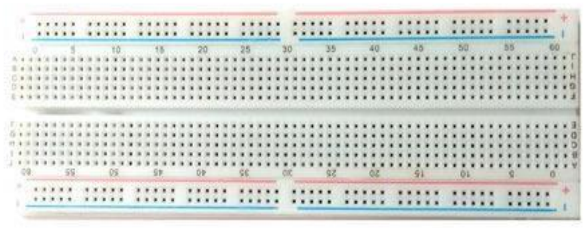


Figure 6 Visual Image of a Bread Board (Components 101, 2018).

C. Buzzer

The buzzer was used as an output device to alert users when the air quality is poor or toxic gas gets detected.

Figure 7 Visual Image of a Buzzer (Elprocus, 2023).

D. DHT-11

In our project, the DHT11 sensor was used as one of the input devices for measuring the quality of the air. It was connected to the Arduino board, which processed the data and provided the output through various output devices according to the quality of the air.

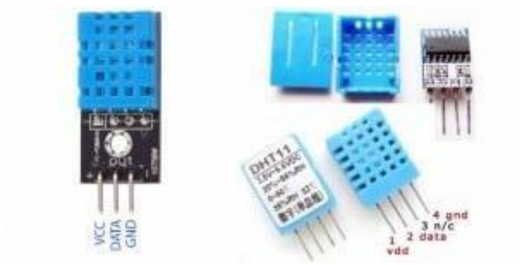


Figure 8 Visual Image of a DHT-11 Sensor (Newton, 2022).

E. ESP32 NodeMCU

NodeMCU was used as the main controller to gather data from the sensors and send it to the ThingSpeak's channel for visualization in the cloud via Wi-Fi. We have programmed it using the Arduino IDE and integrated it with our sensors and actuators to achieve the functionality of internet connectivity.

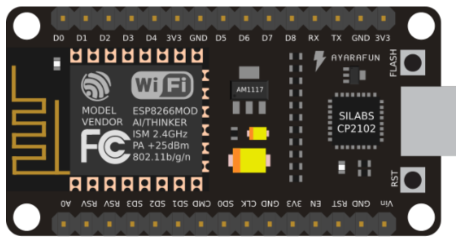

D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
MODEL ESP8266MOD
VENDOR AI/THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
AM1117
SILABS CP2102
FLASH
RST
A0 RSV RSV SD3 SD2 SD1 CMD SD0 CLK GND 3V3 EN RST GND Vin

Figure 9 Visual Diagram of ESP32 NodeMCU (The ThingsBoard Authors, 2023).

F. Jumper Wires

Jumper wires were used to connect the sensors, LEDs, buzzer, and LCD display to the Arduino board, allowing for the creation of a functional system. There are various types of jumper wires which were used in our system such as male-to-male, male-to-female, and female-to-female. The image shown above is a male-to-male jumper wire.

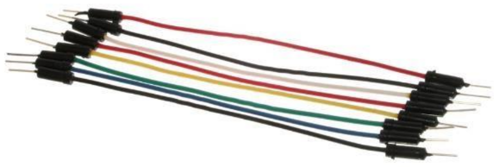


Figure 10 Visual Image of Jumper Wires (Hemmings, 2018).

G. LCD

The 16 x 2 LCD has been used in our project to display the temperature and humidity in real-time also shows the air quality message via the detection done by the gas sensor and temp / humidity sensor used for the system.

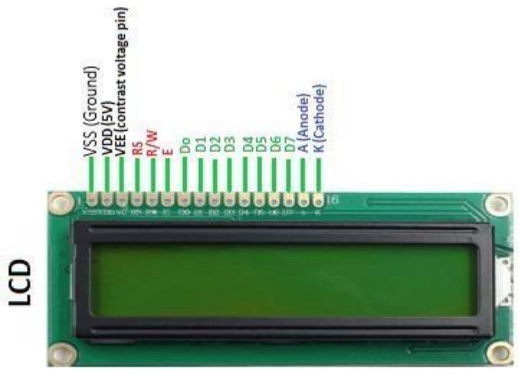


Figure 11 Visual Image of LCD Screen (Javatpoint, 2023).

H. LED

LEDs were used as output devices in our system to indicate the level of air quality detected by the sensors where green LED represents good air and red led represents poor air.

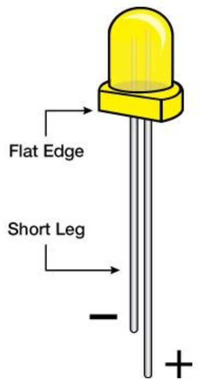


Figure 12 Visual Diagram of a LED (IDEW, 2018).

I. MQ-135

The MQ-135 gas sensor was integrated in our system to detect the presence of harmful gases in the air.

Figure 13 Visual Image of MQ-135 Sensor (Newton, 2022).

J. Resistors

Resistor was used for the LEDs by adjusting their voltage or current levels in our project.

Figure 14 Visual Image of a Resistor (Aaron Guzman, 2023).

3. Development

The development of the system was carried in couple of steps. Those steps of the development procedures are briefly overview as follows:

3.1. Planning and Design

The initial step of this project was to conduct thorough research on the topic. The aim was to acquire a better understanding of the importance and current state of the system. Various reliable sources were referred to during the research process. Additionally, research was conducted to familiarize the group with the required hardware and software needed to construct the system.

After researching the hardware and software, a list of required resources was generated. The expected final budget was calculated using this list, and then the necessary budget was planned to be collected by equally dividing the cost among all the members of the group.

The development work and all the tasks of the project was then distributed among all the members of the group. The work was divided in a way that ensured that every member of the group was responsible for every aspect of the system's development.

3.2. Collecting artifacts and resources

The group collected the expected budget calculated during the planning phase from all members as planned. With the list of necessary hardware created during the planning phase, all the required hardware was purchased online and most of the resource were provided by the college resource department. Additionally, all the necessary software needed to complete the project was installed on the devices of group members who were assigned for research and development part. This includes downloading all the required libraries that would be needed later to write Arduino sketches for the system. The group ensured that they had all the necessary resources to begin building the system after completing every task.

3.3. System Development

In this step, the focus was on ensuring the functionality of all the essential hardware, software, and libraries required for the project. A thorough check was performed to ensure that all the necessary items were available and ready for use. Next, the purchased hardware was tested to ensure that they were in good working condition. This was necessary to avoid any inconvenience or delay during the assembly process. Once the hardware was verified to be functional, the assembly process was initiated. The team began assembling the different components of the system.

During the assembly process, the previously created blueprint of the system was consulted. Based on the blueprint, the hardware components were physically assembled. The Arduino board was powered through a USB cable connected to a laptop. Two sensors, MQ-135 and DHT-11, were connected to the Arduino board as analog and digital input sources, respectively, using jumper wires. In addition, green and red LEDs and a buzzer were connected to the digital pins of the Arduino board via jumper wires to provide digital output. An LCD display was connected to the Arduino board using jumper wires in analog pins to display the output on the screen. All hardware components connected to the Arduino board were powered through the +5V pin. Resistors were connected to the positive ends of the LEDs to prevent the overflow of current that could damage the LEDs.

After the hardware assembly was completed, the Wi-Fi module ESP32 was used to implement the ThingSpeak which is a cloud platform for data visualization. The ESP32 module was connected to the Arduino board, and the system was programmed to transmit the data collected from the sensors to the ThingSpeak platform through the ESP32 module. This allowed the air quality data to be stored in the cloud and accessed remotely.

Once all the hardware resources were assembled correctly, the sketch writing for the air quality monitoring and alerting system began. Sketches were written to establish the logic of reading sensor values and processing them to measure air quality and provide output through various output devices based on the air quality.

The sketches were then debugged to ensure they were error-free and worked as intended. Furthermore, the team integrated the ESP32 Wi-Fi module with the Arduino board to upload the sensor data to ThingSpeak. This cloud platform allowed us to visualize and analyse the data in real-time and set up alerts to notify them if the air quality reached unsafe levels. Once the testing was completed successfully, the air quality monitoring and alerting system was ready for deployment.

3.4. Testing

Upon the completion of the system development, various test cases were conducted to ensure its proper functioning. To validate the system's performance, the MQ-135 sensor was exposed to CO2 and smoke, while the DHT sensor was subjected to heat. The system was able to accurately measure the air quality and provided output through different output devices based on the air quality. The system's response to the various test cases was observed to be in accordance with its intended functionality, thus ensuring the successful completion of the development process.

4. Results and Findings

4.1. Project Outcomes

The system monitors air quality and shows users when harmful pollutants like CO2, NO2, and smoke are detected by printing alerting message in the LCD. It uses a red LED and buzzing sound to notify users of poor air pollutants or gases, while the LCD screen displays the air quality status, temperature, and humidity. The system also helps to calibrate the air quality sensor by displaying temperature and humidity readings.

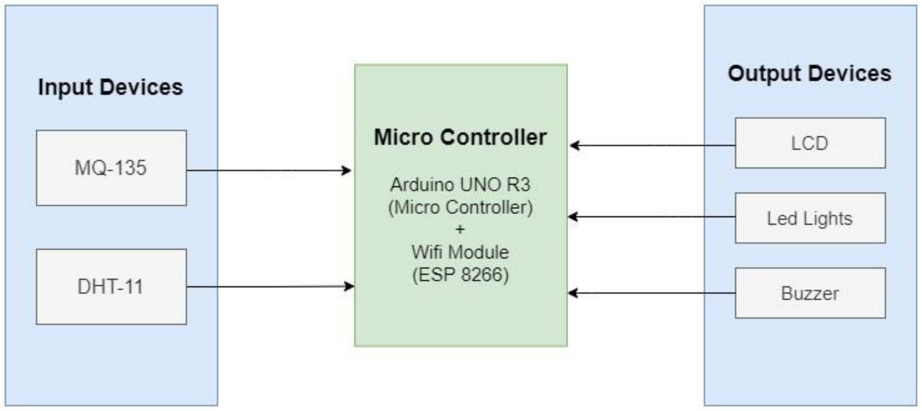


Figure 15 Block Diagram of our System

Our main aim was to assemble all the necessary hardware and make the system work properly. The block diagram above was what we had expected our project outcome would be like after the completion. And, although the system had intended outcomes and met the expectations of this project. But still it had limitations. The limitations of the project are as follows:

- It is designed to monitor and alert for only a few specific pollutants. It may not be able to detect other harmful gases that may be present in the air.
- It requires a stable and reliable internet connection to send data to a remote server for analysis and storage. Any interruption in the connection may lead to data loss.
- It does not consider the weather conditions, which can have a significant impact on the air quality.
- It may not be suitable for outdoor use as it is not waterproof or weatherproof.

4.2. Test Cases

4.2.1. Test 1: MQ-135 and DHT-11 in normal state

Objective	To check the output in normal air, temperature and humidity
Action	• Power was supplied to the micro-controllers. • Initial Codes for the micro-controllers were uploaded via Arduino IDE.
Expected Result	• Good air message should be displayed on the LCD. • Normal Temperature and Humidity should be displayed on the LCD. • Green LED should turn on.
Actual Result	• Good air message was displayed. • Normal Temperature and Humidity was displayed. • Green LED was turned on.
Conclusion	The test was successful.

Table 1 Test: MQ-135 and DHT-11 in normal state

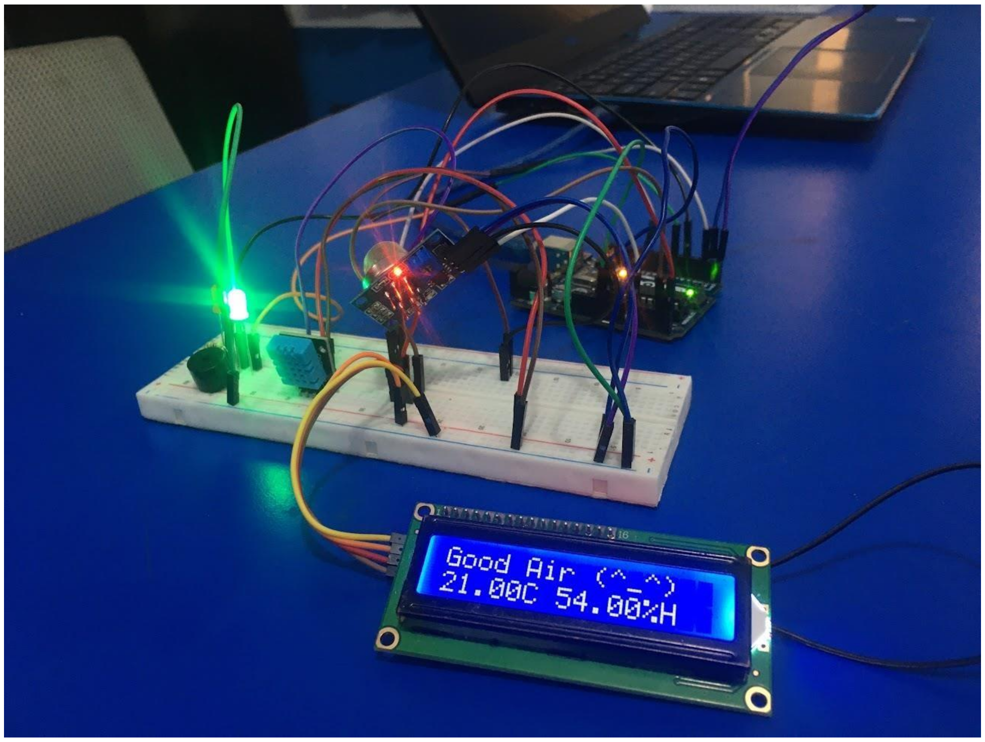


Figure 16 Normal test message shown in LCD

4.2.2. Test 2: MQ-135 exposed to toxic gas

Objective	To check the output on LCD when MQ-135 is exposed to lighter's gas.
Action	• Lighted gas using the lighter. • MQ-135 was placed to detect the poor gas.
Expected Result	• Poor air message should be displayed on the LCD. • Normal Temperature and Humidity should be displayed on the LCD. • Red LED should turn on. • Buzzer should turn on.
Actual Result	• Poor air message was displayed. • Normal Temperature and Humidity was displayed. • Red LED was turned on. • Buzzer was turned on.
Conclusion	The test was successful.

Table 2 Test: MQ-135 exposed to toxic gas

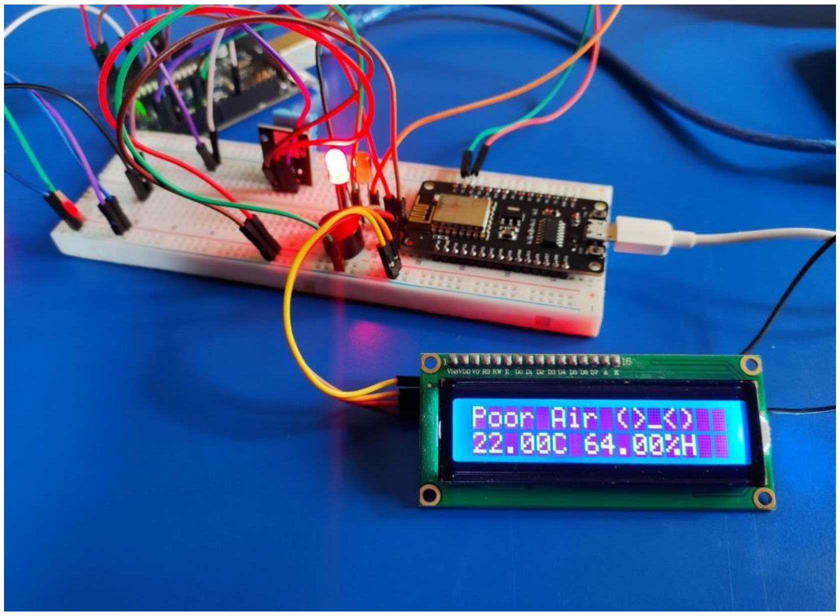


Figure 17 MQ-135 Detected Poor Gas and Displayed Alert Message

4.2.3. Test 3: DHT-11 exposed to fire

Objective	To check the output in normal air but, high temperature and low humidity
Action	• Lit fire and place the DHT-11 near it.
Expected Result	• Good air message should be displayed on the LCD. • High Temperature and Lower Humidity should be displayed on the LCD.
Actual Result	• Good air message was displayed. • High Temperature and Lower Humidity was displayed.
Conclusion	The test was successful.

Table 3 Test: DHT-11 exposed to fire

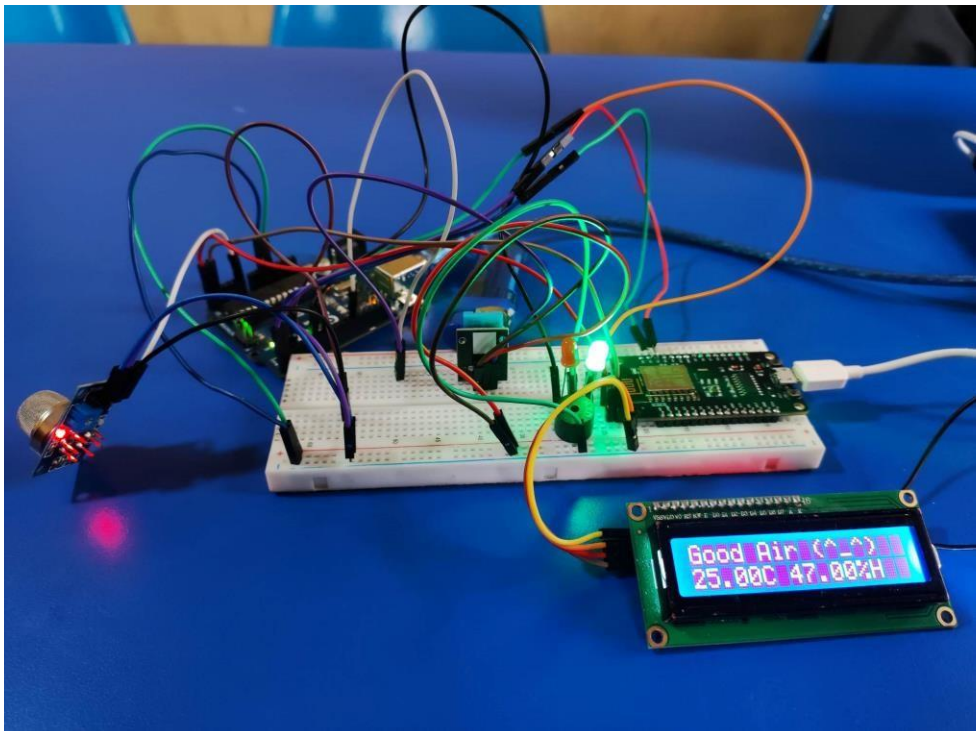


Figure 18 Lighted Fire in the DHT-11 and temperature increase message displayed

4.2.4. Test 4: DHT-11 detached from the system

Objective	To check the output after the detachment of DHT-11
Action	• DHT-11 sensor was removed from the breadboard.
Expected Result	• Good air message should be displayed on the LCD. • No temperature and humidity should be displayed on the LCD.
Actual Result	• Good air message was displayed. • No temperature and humidity were displayed.
Conclusion	The test was successful.

Table 4 Test: DHT-11 detached from the system

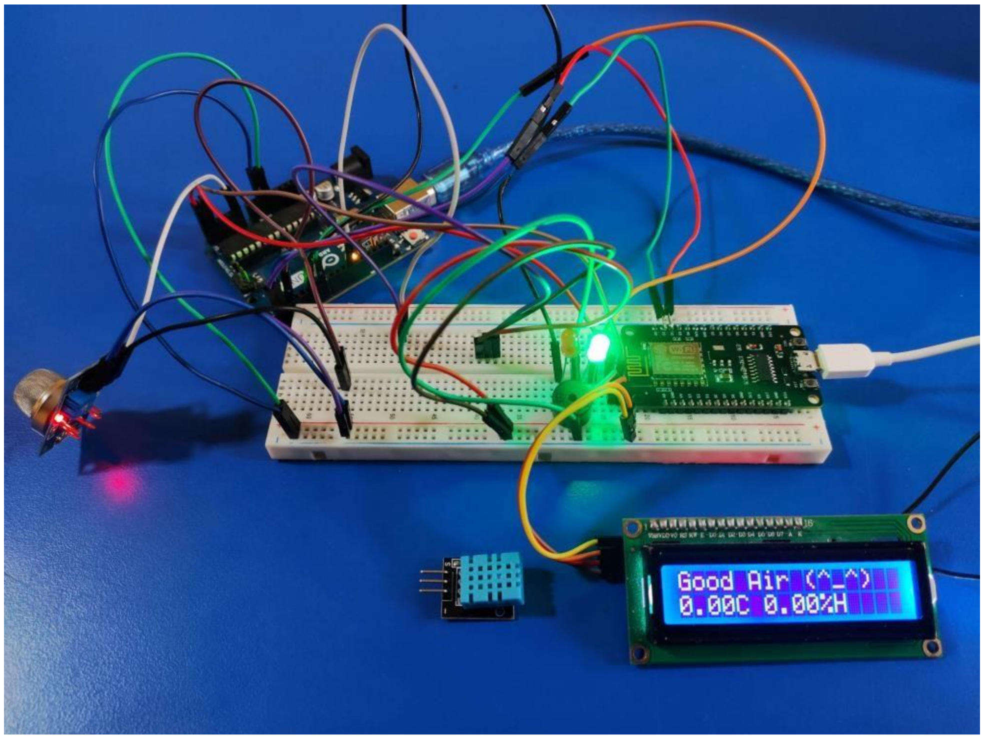
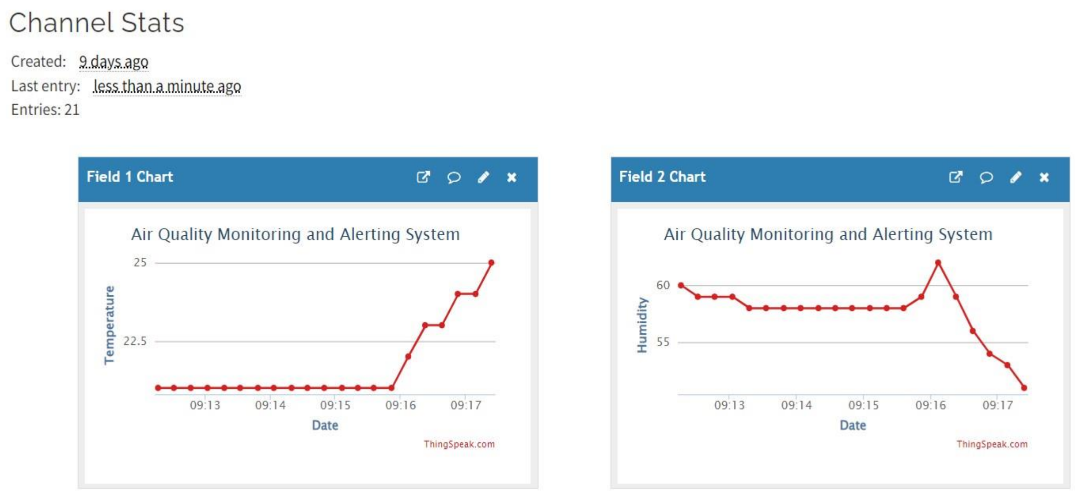


Figure 19 DHT-11 removed from the system and no temp / humidity were displayed

4.2.5. Test 5: Data Visualization in ThingSpeak

Objective	To check the data visualization chart in ThingSpeak
Action	• DHT-11 sensor was installed again in the breadboard. • NodeMCU was given access to the ThingSpeak's channel and internet connection.
Expected Result	• Temperature and Humidity data should be sent to the cloud and displayed in chart.
Actual Result	• Temperature and Humidity data was sent to the cloud and displayed in chart.
Conclusion	The test was successful.

Table 5 Test: Data Visualization in ThingSpeak


Channel Stats

Created: 9 days ago

Last entry: less than a minute ago

Entries: 21

Figure 20 Temperature and Humidity data visualization in ThingSpeak

5. Future Works

The system created during this project is currently just a prototype and needs further improvements to become a more advanced final product. Therefore, the following features may be incorporated into the system that may help make it more polished final product.

- System processing can be migrated to the cloud services such as AWS, Azure to improve security, integrity, and availability.
- MQ-135 sensor used in the system can be replaced with a more sensitive and sophisticated sensor that does not require calibration.
- Integration with a mobile app to access real-time air quality data and alerts from smartphones.
- Integration with smart home systems like Google Home or Amazon Alexa to receive air quality alerts via voice assistants and trigger actions like turning on air purifiers.

6. Conclusion

Air pollution has become a significant concern globally due to its harmful effects on human health. However, traditional air quality monitoring stations are costly and difficult to establish. To address this problem, the project proposes a portable and cost-effective solution for real-time air quality monitoring and detection. The system in summary uses an air quality sensor, an Arduino, and various output devices to alert users, can provide real-time pollution alerts and collect air pollution data is sent to the cloud through Wi-Fi module for data visualization.

The prototype system has limitations and can be improved with additional features. Logging through a server can be implemented to maintain a data log and display and analyse it in the future. A more sensitive and sophisticated sensor can replace the current air quality sensor to detect specific gases.

The project proposes a portable and cost-effective solution for real-time air quality monitoring and detection, which can provide real-time pollution alerts and collect air pollution data. The system can be improved by adding features such as moving the Arduino's system processing to the cloud which was done for data visualization in ThingSpeak but could have been migrated to cloud services for proper and reliable logging backups. Another addition could be using a more sensitive sophisticated air quality sensor for better detection and accuracy.

7. References

The MathWorks, Inc. (2023). Learn More About ThingSpeak. Retrieved from ThingSpeak: https://thingspeak.com/pages/learn_more

Aaron Guzman, A. G. (2023). Resistor - IoT Penetration Testing Cookbook [Book]. Retrieved from Oreilly: https://www.oreilly.com/library/view/iot-penetrationtesting/9781787280571/5c8f9772-0de5-432c-ae7b-c5700c178acf.xhtml

Arduino. (2021, September 15). About Arduino. Retrieved from https://www.arduino.cc/en/about

Arduino. (2023). UNO R3 | Arduino Documentation. Retrieved from https://docs.arduino.cc/hardware/uno-rev3

Barragán, H. (2023). Breadboard. Retrieved from Wiring: http://wiring.org.co/learning/tutorials/breadboard/

Components 101. (2018, June 8). Breadboard Connections, Features, Circuit Examples & Datasheet. Retrieved from https://components101.com/misc/breadboard-connectionsuses-guide

Elprocus. (2023). What is a Buzzer : Working & Its Applications. Retrieved from https://www.elprocus.com/buzzer-working-applications/

Fritzing. (2021). Retrieved from https://fritzing.org/

Hannah Ritchie, M. R. (2021, January). Air Pollution. Retrieved from ourworldindata.org: https://ourworldindata.org/air-pollution

Hemmings, M. (2018, January 30). What is a Jumper Wire? Retrieved from SparkFun Education: https://blog.sparkfuneducation.com/what-is-jumper-wire

IDEW. (2018, May 29). LED Light - Internet of Things Project. Retrieved from docs.idew.org: https://docs.idew.org/internet-of-things-project/references-for-wiring-andcoding/led-light

Javatpoint. (2023). Arduino LCD Display. Retrieved from https://www.javatpoint.com/arduino-lcd-display

Newton, A. (2022, August 22). DHT11 Humidity Temperature Monitor on ThingSpeak with NodeMCU. Retrieved from How to Electronics: DHT11 Humidity Temperature Monitor on ThingSpeak with NodeMCU

Newton, A. (2022, August 22). IoT Based Air Quality Index Monitoring with ESP8266 & MQ135. Retrieved from How To Electronics: https://how2electronics.com/iot-air-qualityindex-monitoring-esp8266/

NodeMCU. (2023, February 9). NodeMCU Documentation. Retrieved from https://nodemcu.readthedocs.io/en/dev-esp32/

The ThingsBoard Authors. (2023). NodeMCU overview. Retrieved from thingsboard.io: https://thingsboard.io/docs/samples/nodemcu

8. Appendix

8.1. Appendix A: Hardware and Software Definitions

- **Arduino IDE**

The Arduino IDE is a software tool that provides a user-friendly interface for writing and uploading sketches to any Arduino board. This tool includes a text editor for writing code, as well as a message box, a text terminal, and toolbars with basic functions (Arduino, 2021). Sketches are programs written in the C++ programming language.

- **Draw.io**

Draw.io is a tool available online that lets users create diverse types of diagrams like flowcharts, mind maps, and network diagrams. It is a straightforward and userfriendly tool that offers an extensive collection of shapes and symbols to illustrate different elements in a diagram.

- **Fritzing**

Fritzing is an open-source software tool designed to help users create and document electronic projects. It aims to make electronics more accessible to everyone by providing a user-friendly interface for designing and sharing circuit diagrams, as well as creating printed circuit board (PCB) layouts (Fritzing, 2021).

- **ThingSpeak**

ThingSpeak is a free online platform used to develop and prototype IoT (Internet of Things) systems that provides a simple and user-friendly interface for developing, storing, and visualizing data from connected devices (The MathWorks, Inc., 2023).

- **Arduino UNO**

Arduino UNO is a microcontroller board which features 14 digital input/output pins, with 6 being capable of pulse width modulation (PWM) outputs, and 6 analog inputs. The board has a 16 MHz ceramic resonator, a USB connection, an ICSP header, a reset button, and a power jack (Arduino, 2023). It has everything required to support the microcontroller, making it easy to get started by simply connecting it to a computer via USB or powering it with an AC-to-DC adapter or battery.

- **Breadboard**

A breadboard is a tool used for creating temporary prototypes and testing circuit designs without the need for soldering. It features a board with holes in which electronic components can be inserted and connected through wires (Barragán, 2023).

- **Buzzer**

A buzzer is a device that produces an audible signal, usually in the form of a loud and continuous sound, by converting an electrical signal into sound (Elprocus, 2023). The buzzer is usually powered by DC voltage and can produce various sounds depending on the design, such as alarm, music, bell, or siren.

- **DHT-11**

The DHT11 is an inexpensive digital sensor that measures temperature and humidity. It works by using a capacitive humidity sensor and a thermistor to detect the temperature and humidity of the surrounding air (Newton, 2022). It then outputs the collected data in a digital signal format through its data pin, eliminating the need for analog input pins.

- **ESP32 NodeMCU**

NodeMCU is a firmware that is open source and based on Lua programming language. It is designed for ESP32 and ESP8266 Wi-Fi System-on-Chips (SOC) from Espressif, which comes with an on-module flash-based SPIFFS file system (NodeMCU, 2023).

- **Jumper Wires**

Jumper wires are wires with pins on both ends that can be used to establish a temporary connection between two points without soldering. They are commonly used with breadboards and other prototyping tools to facilitate the creation and modification of circuits (Hemmings, 2018).

- **LCD**

The Liquid Crystal Display (LCD) is an electronic display module that uses liquid crystals to display visual information (Javatpoint, 2023). The liquid crystals present in the display create different patterns and shapes when voltage is applied to them.

- **LED**

LEDs are tiny, energy-efficient lights that emit bright light and are frequently used in electronic devices (IDEW, 2018). The positive leg and negative leg of an LED can be visually identified by their length, with the negative leg being shorter.

- **MQ-135**

The MQ-135 is a gas sensor that can sense a variety of gases such as ammonia nitrogen, oxygen, alcohols, aromatic compounds, sulfide, and smoke (Newton, 2022). It can detect different harmful gases and is particularly suitable for air quality monitoring applications due to its low cost and versatility.

- **Resistors**

Resistors are a type of electronic component that resist the flow of electric current or the movement of electrons through a circuit that are passive components, which means that they do not generate any electrical power, but instead reduce voltage and current in a circuit by converting some of the electrical energy into heat (Aaron Guzman, 2023).

8.2. Appendix B: Individual Tasks and Contributions

Name	Task / Contribution
Kishor Bamjan Tamang	Device Management, Maintenance, Connectivity
Mingmar Lama	Research and Documentation
Stuti Pangeni	Testing and Device Maintenance
Binit Kunwar	Research and Development
Shimon Sharma	Research and Testing

Table 6 Individuals Tasks and Contributions

8.3. Appendix C: Overall Image of the System

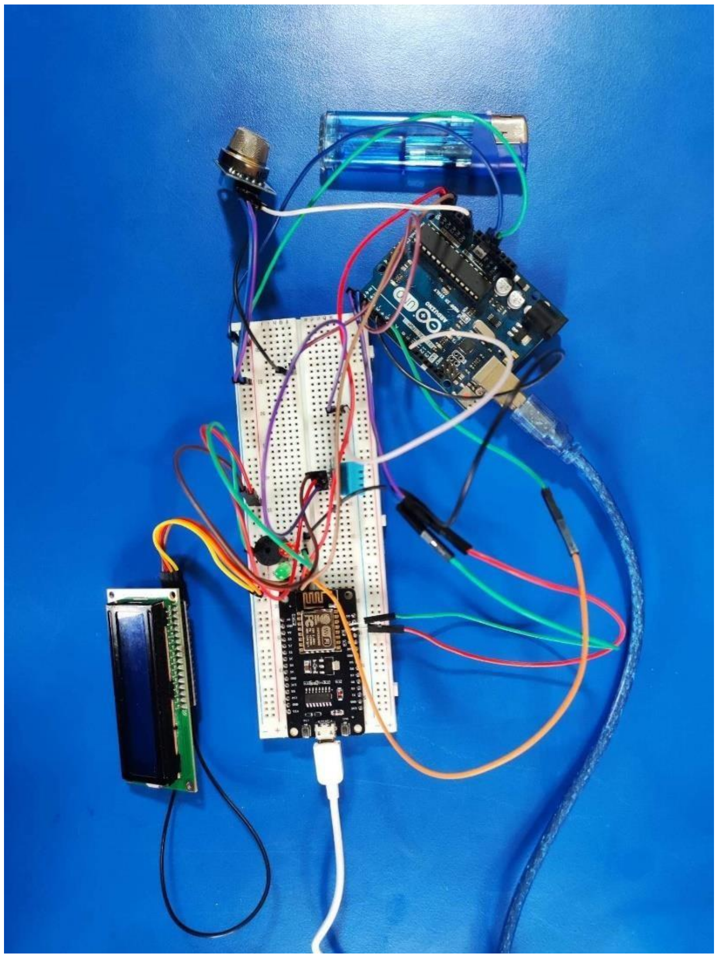


Figure 21 Image of fully assembled system

8.4. Appendix D: Source Code

8.4.1. Source code for main.ino

```
// Import libraries for the system
 #include <LiquidCrystal_I2C.h>
 #include <dht.h> dht
 DHT;

 // setting the LCD address to 0x27 for 16 characters and 2 line display
 LiquidCrystal_I2C lcd(0x27, 16, 2);

 int sensorValue;  // for sensing the analog value int
 digitalValue; // for sensing the digital value

 // Setting dht11 to pin 11
 #define DHT11_PIN 11

 void setup()
 {
   Serial.begin(9600); // setting the serial port to 9600

   // Setting up lcd to display air quality, temperature, and humidity
 lcd.init();   lcd.backlight();

   pinMode(5, OUTPUT); // setting red led and buzzer to pin 5
 pinMode(2, INPUT); // setting pin 2 for MQ-135 sensor to take input
 pinMode(7,OUTPUT); // setting green led to pin 7   int chk =
 DHT.read11(DHT11_PIN); // initializing the DHT11 sensor
 }

 void loop()
 {
   sensorValue = analogRead(A0); // reading analog input from pin 0 [MQ135]
 digitalValue = digitalRead(2); // digital pin MQ135 connection

   // Read temperature and humidity from DHT11 sensor   int
 chk = DHT.read11(DHT11_PIN);

   if (sensorValue >= 400)
   {

     // Displays good air message in the lcd
 lcd.clear();     lcd.setCursor(0,0);
 lcd.print("Poor Air (>_<)  Wear a Mask ");
 lcd.setCursor(0, 1);
 lcd.print(DHT.temperature);     lcd.print("C
 ");     lcd.print(DHT.humidity);
 lcd.print("%H");

     // Displays good air message in the console
     Serial.print("Poor Air (>_<)");
     Serial.print(DHT.temperature);
     Serial.print("C ");
     Serial.print(DHT.humidity);
     Serial.println("%H");

     alertState(); // calling the function to turn on red led and buzzer
 delay(3000); // sleeping for 3 seconds
   }
   else{

     // Display good air message in the lcd
     lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Good Air (^_^) ");
 lcd.setCursor(0, 1);
 lcd.print(DHT.temperature);
 lcd.print("C ");
 lcd.print(DHT.humidity);
 lcd.print("%H");

    // Displays good air message in the console
    Serial.print("Good Air (^_^) ");
    Serial.print(DHT.temperature);
    Serial.print("C ");
    Serial.print(DHT.humidity);
    Serial.println("%H");

    normalState(); // calling the function to turn on green led
   }
   delay(3000); // sleeping for 3 seconds
 }

 void normalState(){
    digitalWrite(7,     HIGH);     //     turning     on     green     led
 digitalWrite(5,LOW); // turning off red led
 }

 void alertState(){
     // turning on buzzer and red LED if MQ135 sensor reading is
 high    digitalWrite(5, HIGH);           // turning off green led
 digitalWrite(7,LOW);
 }
```

8.4.2. Source Code of cloud.ino

```
// Import libraries for the system
#include <ThingSpeak.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <SoftwareSerial.h>

// Define the TX and RX pins
#define RX_PIN D1 // rx to d2 in node mcu
#define TX_PIN D2 // tx to d1 in node mcu

// Create a SoftwareSerial object for serial communication
SoftwareSerial mySerial(RX_PIN, TX_PIN);

// Set credentials for internet connection const char* ssid = "SSID"; //
set ssid for internet connection const char* password = "PASSWORD";
// set the password for the ssid

WiFiClient client;

// ThingSpeak channel details unsigned long CHANNEL_ID =
CHANNEL_ID; // replace with ThingSpeak channel const char*
WRITE_API_KEY = "API"; // replace with the channel's api key

void setup() {

  // Initializing the serial port at 9600
Serial.begin(9600);   mySerial.begin(9600);

  // Connecting to the intenet
  WiFi.begin(ssid, password);
```

```
   // Loops until the Wi-Fi module connects to the provided ssid
  while (WiFi.status() != WL_CONNECTED) {     delay(1000);
     Serial.println("Connecting to WiFi...");
    }

    // Printing new line, Wi-Fi connectivity status and the IP address
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println(WiFi.localIP());

    // Initializing the thingspeak client to send data
    ThingSpeak.begin(client);
  }

  void loop(){

    // Collecting the data and printing in the serial display
    String data = mySerial.readStringUntil('\n');

   int tempStartIndex = data.indexOf("^_^") + 4;   int
  tempEndIndex = data.indexOf("C", tempStartIndex);   int
  humidStartIndex = data.indexOf(" ", tempEndIndex);   int
  humidEndIndex = data.indexOf("%", humidStartIndex);

    String tempString = data.substring(tempStartIndex , tempEndIndex);
    String humidString = data.substring(humidStartIndex+1, humidEndIndex);

    float   tempValue   =   tempString.toFloat();
  int humidValue = humidString.toInt();


    // Printing temperature and humidity values in the serial console
    Serial.print("Temperature: ");
    Serial.println(tempValue);
    Serial.print("Humidity: ");
    Serial.println(humidValue);

    if (tempValue > 0.00 && humidValue > 0) {
  ThingSpeak.setField(1, tempValue);
     ThingSpeak.setField(2, humidValue);

     int status = ThingSpeak.writeFields(CHANNEL_ID, WRITE_API_KEY);
  if (status == 200) {
        Serial.println("Data sent successfully!");
     } else {
        Serial.println("Failed to send data to ThingSpeak");
     }
    }

    delay(10); // sleeps for 1 seconds before sending the next set of data
  }
```