CONSTRUCTING AN AIR QUALITY DETECTION SYSTEM USING EMBEDDED SYSTEM

NGIAM KEE TIONG

UNIVERSITI TUNKU ABDUL RAHMAN

CONSTRUCTING AN AIR QUALITY DETECTION SYSTEM USING EMBEDDED SYSTEM

NGIAM KEE TIONG

A project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Honours) in Electronic Systems

Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman

September 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	____________________
Name	:	NGIAM KEE TIONG
ID No.	:	21AGB01924
Date	:	19.09.2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“CONSTRUCTING AN AIR QUALITY DETECTION SYSTEM USING EMBEDDED SYSTEM”** was prepared by **NGIAM KEE TIONG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Technology (Honours) in Electronic Systems at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	__________________________
Supervisor	:	TS DR LEE HAN KEE
Date	:	10.10.2023

Specially dedicated to
my beloved father, mother, and other
family members.

ACKNOWLEDGEMENTS

I'd like to start by expressing my gratitude to Ts Dr Lee Han Kee, my supervisor, for his guidance, essential assistance, and patience during this project. His steadfast support throughout our frequent meetings and stimulating conversations encouraged me to persevere through difficult issues, which made the process much more pleasurable.

Special thanks to my friends who provided valuable ideas and insights. Their creativity has been a wellspring of inspiration. Their contributions have added depth and innovation to this endeavour.

I want to express my appreciation to everyone who helped finish this design, whether they were directly involved or not. Its result has been significantly told by collaborative support.

Lastly, I'm deeply obliged to my family for supporting me in whatever I do and for motivating me to pursue my pretensions. I'm immensely appreciative of my parents' backing during my academic career.

CONSTRUCTING AN AIR QUALITY DETECTION SYSTEM USING EMBEDDED SYSTEM

ABSTRACT

This project includes developing hardware and software for the cloud platform, IoT network, and embedded system, and the project will concentrate on constructing and implementing an embedded system for air quality detection. The project will implement the IoT to monitor devices, collect data, and communicate with users. Various sensors will be used to track several types of gases, including carbon monoxide, ammonia, and particle matter levels, also involving temperature and humidity tracking. The calibration of the sensor will be applied before constructing the air quality detector. Sensor calibration aims to verify the sensor's precision and reproducibility of detection instruments. The IoT platform Blynk will prototype, deploy, and manage connected electronic devices at any scale and provide the user with a real-time air quality status.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

$\mu g/m^3$	micrograms per cubic metre
g/m^3	grams/cubic meter
PPM	parts per million
AI	Artificial intelligence
AQI	Air Quality Index
CH_4	Methane
CO	Carbon monoxide
CO_2	Carbon dioxide
DNA	Deoxyribonucleic acid
H2S	Hydrogen sulfide
HCHO	Formaldehyde
IQ	intelligence quotient
LTE	Long-Term Evolution
NFC	Near-field communication
NO_2	Nitrogen Dioxide
O_3	Ozone
PM2.5	Particulate matter 2.5
PM5	Particulate matter 5
PM10	Particulate matter 10
RFID	Radio-frequency identification
R-SH	Thiol
SO_2	Sulfur dioxide
SO_3	Sulfur trioxide
VOC_S	Volatile Organic Compounds

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Nowadays, systems to monitor the quality of air pollutants in the environment have been developed to solve this problem. In most cases, these systems utilize sensors to identify and measure the concentration of multiple pollutants, including PM2.5, PM5, CO, NO_2, and O_3. These sensors' data may be used for identifying areas with high pollution levels, tracking long-term changes in air quality, and sending out notifications when those levels cross legal thresholds.

Using an embedded system, a computer system developed to fulfil a specific purpose is one method of monitoring air quality. Consumer electronics, industrial control systems, automotive systems, and other applications frequently utilize embedded systems. An embedded system may collect and evaluate data from air quality sensors in real time when used for air quality monitoring, enabling quick responses to adjustments in air quality. Air quality monitoring devices have measured air pollution. These systems used to be massive, costly and needed specialist knowledge. However, because of technical advances, air quality sensors are now more compact, affordable, and widely available. (Kalia, P. and Ansari, M.A.,2020) These sensors can enable real-time air quality monitoring in various systems, including embedded systems.

Embedded systems for air quality monitoring normally consist of three main components: sensors, a microcontroller, and communication interfaces. (Jangid and

Sharma, 2016) The sensors detect pollutants in the air and convert the data into electrical signals. The microcontroller is a small computer that processes the data from the sensors and performs calculations to determine the concentration of pollutants. Finally, IoT technology will transmit the data wirelessly for analysis and visualization. Arduino and Raspberry Pi's microcontrollers are the most commonly used in embedded systems since they are low-power and low-cost and can be programmed using a variety of programming languages. Both of these are widely available and have a large community of developers. Embedded systems usually utilize Wi-Fi and Bluetooth as communication interfaces for local connections between the embedded system and a computer or mobile device.

1.2 Problem Statements

Air pollution is a major problem worldwide that negatively influences the environment and human health. In Malaysia, the increase in industrial manufacturing, power plants, vehicle emissions, and open burning activities have led to high levels of air pollution. A recent study conducted by the Centre for Research on Energy and Clean Air and Greenpeace Malaysia showed that air pollution in the country causes around 30,000 deaths annually. The long-term effects of breathing poor air quality are even worse than previously believed. The six primary air contaminants, called "criteria pollutants," are PM2.5, PM5, O_3, CO, SO_2, and NO_2. (Centre for Research on Energy and Clean Air (CREA), 2022) These pollutants must be monitored to ensure the air is safe to breathe. Cars emit carbon monoxide, carbon dioxide, and nitrogen oxides, contributing to air pollution. The current method of measuring air pollution involves expensive permanent sites or specialized mobile equipment requiring skilled personnel to operate and maintain. This high cost limits its widespread use and makes real-time monitoring desirable to detect hazardous levels of contaminants quickly. To solve the problem, some manufacturers have launched portable air quality monitors that can easily be found on the market. However, their accuracy, limited sensors and management and analysis of the data have limited the portable air quality. Monitor from measuring certain pollutants or air quality that does not meet the user's needs.

1.3 Project Objectives

The objectives of the project are shown as follows:

i) To construct and develop an embedded real-time air quality detection system.
ii) To determine the air quality rapidly and accurately.
iii) To implement IoT technology to transmit the data wirelessly for analysis and visualization.

1.4 Scope of the project

The project will create an embedded system to collect and process sensor data. Compact, energy-efficient, and able to function in various environments are all requirements for the design. Users must be able to get real-time data on air quality thanks to the software's ability to manage the massive volumes of data that the sensors will produce.

For the system's sensors to identify the specific gases and contaminants in the air, they must be selected appropriately. Several types of sensors must be working to detect various pollutants, and the sensors must be calibrated to guarantee reliable data.

The software or the cloud will need to be designed for the massive value of data, and algorithms will need to be developed to analyze the data and identify trends and patterns. The software will also need to provide users with real-time information about the air quality in their area.

To ensure the system is accurate, dependable, and simple to use, testing and evaluation are required in the end. To do this, it will be necessary to test the sensors in a controlled setting to ensure they offer reliable readings and to test the software to handle massive data value and present users with useful information. To ensure the system can function in various environmental conditions, it must be tested in a real world.

1.5 Outline of the report

This report will be divided into five chapters: introduction, literature review, methodology, results and discussion, and conclusion and recommendations. The introduction chapter will describe the project's background, problem statement, objectives, and scope. Chapter 2, the literature review will discuss air quality and health, existing air quality monitoring systems, embedded systems, the Internet of Things (IoT), and the type of sensors used in air quality detection. Chapter 3 Methodology describes the program's flow and Gantt charts, the system's design and architecture, the selection of components and sensors, system implementation, and testing. Chapter 4 results and discussion section describes the constructed system, the comparison with existing monitoring systems, and the study of outcomes. The data will be collected and discussed the result of the data. Finally, Chapter 5 conclusion and recommendations section summarises findings, conclusions, implications, recommendations for future work, and final thoughts.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview of air pollution and its effects on human health

2.1.1 Air Pollution in Malaysia

Air pollution in Malaysia is a critical issue threatening the populace's health, especially in urban areas. Malaysia's air pollution problem is not new and has long been a source of worry. Industrial manufacturing, power plants, vehicle emissions, and open burning activities have led to high levels of air pollution in Malaysia. Forest fires, weather patterns, and wind direction contribute to haze in Malaysia and impact the severity and duration. In a report released in 2020, the Malaysian Ministry of Energy, Science, Technology, Environment, and Climate Change claimed that air pollution in the nation had reached alarming levels, with high concentrations of particulate matter and other pollutants posing serious health risks to the populace. (Ministry of Environment and Water, 2020) According to the research, several parts of the country, especially urban areas, are suffering dangerously high levels of air pollution, with particulate matter levels above the World Health Organization's (WHO) suggested limits.

The situation is particularly dire in the Klang Valley, which includes the capital city of Kuala Lumpur and its surrounding areas. Between 2000 and 2009, the greatest annual mean PM10 concentrations at the Klang station varied between 80 and 100 g/m3. They exceeded the 40 and 20 g/m3 limits set by the World Health

Organization's air quality guidelines (WHOAQG) and the New Malaysia Ambient Air Quality Standard (NMAAQS). The local population is at risk for health problems due to Klang's statistically significant clusters of high PM10. The WHOAQG standards were surpassed in Petaling Jaya, potentially endangering the health of the area's densely populated residents. (Elias et al.,2023) The main pollutants in the Klang Valley are PM, SO2, and NO2, all known to affect human health adversely.

Malaysia's air is polluted by emissions from an increasing number of sources, including industrial production, electricity generation, transportation, and open burning activities. (Centre for Research on Energy and Clean Air (CREA), 2022) Pollutants from the combustion of fossil fuels emitted by vehicles are among the leading causes of air pollution. There were almost 30 million registered vehicles on Malaysian roadways in 2011. (Chan, 2022) That number is very likely to be higher now all those cars are producing dangerous gases that are detrimental to our health.

Additionally, biomass burning and forest fires in our country and neighbouring countries contribute to the seasonal transboundary haze incidents that often cause a spike in air pollution from July to October every year. The main source of the issue, Indonesian peatland burning, which Malaysian businesses support and profit from, needs to be adequately addressed.

In Malaysia, respiratory ailments ranked second in mortality in 2019 (14.8%), while cardiovascular diseases were the primary factor in 7.9% of hospital fatalities. The major causes of this increased health concern are the growing number of fixed, mobile, and local sources that produce harmful gases and particle pollution. According to research, Malaysians would live 1.8 years longer on average if yearly fine particulate matter PM2.5 concentrations were lowered to 10 g/m3, representing a 65% drop from the nation's current average air pollution levels. The State of Global Air 2020 study states that PM2.5-related mortality in Malaysia climbed by over 30% in the previous ten years. There may have been up to 10,600 air pollution-related fatalities in the nation in 2019. For the economic, the economic costs associated with health effects include healthcare and medical expenses and lost productivity due to early mortality. The projected RM 303 billion yearly economic impact of ambient air pollution in Malaysia is 20% of the nation's GDP in 2019. (Centre for Research on Energy and Clean Air (CREA), 2022)

The Malaysian government has adopted a settlement bill to address the issue of environmental pollution. Several environmental laws and regulations control pollution in Malaysia, such as the Environmental Quality Act of 1974 and its ancillary laws, the Malaysian Ambient Air Quality Standard of 2013, and the Environmental Quality (Clean Air) Regulations of 2014. (Chin et al., 2019) As of 2019, These legislative measures try to limit and cut back on the number of pollutants that different businesses release. Despite this legal system, Malaysia still has air pollution problems. Public support for environmental preservation is crucial for government programs and policies to address environmental issues effectively. A greater understanding of popular support for environmental protection and public awareness aids the government's efforts to enhance Malaysia's air quality.

2.1.2 Health effects of air pollution on humans

Exposure to air pollution can have a range of negative health effects, both short-term and long-term. Air pollutants can be divided into two main categories: particulate matter and gaseous pollutants. A combination of solid and liquid particles floating in the air is called particulate matter. The particles can vary in size, from coarse dust and dirt to tiny particles that are invisible to the naked eye. The smallest particles, PM2.5, are the most dangerous as they can penetrate deep into the lungs and even enter the bloodstream. Gaseous pollutants include NOs, SO_2, O_3, CO, and VOCs. These pollutants can have a range of harmful effects on human health, including respiratory problems, cardiovascular disease, and cancer. (Manisalidis et al., 2016)

Breathlessness, wheezing, and other respiratory issues can be brought on by exposure to air pollution. It can also worsen respiratory conditions such as asthma and chronic obstructive pulmonary disease (COPD). (Kelly and Fussell, 2011) Particle pollution-related health impacts are more likely to affect those with heart or lung illness, older adults, children, people with diabetes, and people from lower socioeconomic backgrounds. Inhaled particles can cause inflammation in the respiratory tract, with the extent of inflammation depending on the dose and composition of the particles. This inflammation can increase airway responsiveness and reduce lung function. The initial harm can worsen with repeated exposure to particle pollution, which can also encourage chronic inflammation. The balance between injury and repair mechanisms is essential in developing and progressing inflammatory respiratory diseases such as asthma. Inhalation of particle pollution can affect the stability and progression of these conditions through its inflammatory effects on the respiratory tract.

Air pollution, particularly outdoor particle pollution like PM2.5, can lead to the development and worsening of cardiovascular disease. It causes inflammation in the blood vessels, leading to plaque build-up and increasing the risk of heart attack and stroke. Short-term exposure to PM2.5 can trigger cardiovascular events, while long-term exposure increases the risk of cardiovascular mortality and reduces life expectancy. Although the individual risk from particle pollution is smaller than other established risk factors, it can increase hospitalizations for serious cardiovascular events in the population, particularly those with established heart disease.

Exposure to certain air pollutants like benzene and polycyclic aromatic hydrocarbons (PAHs) can increase the risk of cancer, particularly lung cancer. A new study has found that exposure to fine particulate matter is associated with an increased risk of mortality for several other types of cancer, including breast, liver, pancreatic, and cancers of the digestive organs. (Manisalidis et al., 2016) Scientists suggested that pollution might spark defects in DNA repair function, alterations in the body's immune response, or inflammation that triggers the growth of new blood vessels that allow tumors to spread. The study supports reducing PM2.5 levels as a modifiable public health concern. However, other modifiable risk factors like diet and exercise may also significantly influence cancer risk.

Exposure to air pollution during pregnancy can negatively affect fetal development, including low birth weight and preterm birth. It can also affect a child's cognitive development, leading to problems with learning and memory. Studies have shown that prenatal exposure to air pollution, particularly to PM2.5, NO_2, and polycyclic aromatic hydrocarbons (PAHs), can negatively impact cognitive development in children. (Morgan et al., 2023) Exposure during mid-late gestational periods has been identified as a critical exposure window, disrupting neurogenesis, neuronal migration, axonal and dendritic arborization, synaptogenesis, myelination, apoptosis, and neural circuit formation. Prenatal exposure has been associated with reductions in left hemisphere white matter surface, cortex thickness, altered white matter organization, reduced blood flow, lower IQ, slower reaction times, poorer memory, and increased risk of ADHD-like behaviors in children. The impact of air pollution during pregnancy is of particular concern since cognitive and motor capacities during childhood and beyond depend upon proper gestational development. (Morgan et al., 2023)

2.2 Pollutants and Air quality index

2.2.1 Major pollutants

Table 2.1 lists the pollutants that are emitted from industries release. Manufacturing procedures, burning fossil fuels, agricultural waste, vehicle emissions, and waste gasses are some sources of air pollution in industry. (Choudhary and Garg, 2013) O_3, PM, CO, SO_2, and NO_2 are the five main air pollutants listed in Table 2.1 that have the greatest potential to endanger human health. And other pollutants that are emitted from industry, such as H_2S, VOCs, and HCHO, have also increased the air quality level. (Choudhary and Garg, 2013) The pollutant affects air and water quality when the industry discharges pollutants into rivers or ditches.

Table 2.1: The pollutant

Category	Source	Emitted pollutants
Agriculture or Haze	Open burning	CO,
Mining and quarrying	Coal mining	SO_2, NO_x,
Industries	Food, beverages, and tobacco	CO, H_2S
	Oil and gas	SO_2, PM, CH_4,
	Mining	SO_2, NO_x
	Paper products, printing	SO_2, CO, H_2S, R-SH
	Chemicals	SO_2, CO,
Petroleum refineries	Miscellaneous products of petroleum and coal	SO_2, NO_x, CO
Transport	Automobile vehicle exhaust	NOx, CO, CO_2, PM, Hydrocarbons
Power generation	Electricity, gas, and steam	SO_2, NO_x, CO, VOC, SO_3, Pb
Indoor	Printing machines, Building Materials	HCHO, O_3, biological contaminants, CO, CO_2

2.2.2 Particulate matter

Particulate matter (PM), called Particle pollution, comprises particles (tiny pieces) of solids or liquids in the air. These particles may include dust, dirt, soot, smoke, and liquid drops. (ABEDIN, 2013) Some particles are big enough to see, such as the smoke in the air. Others are so small that can't see them in the air. They can only be detected using a special device such as an electron microscope.

Particulate matter sources can come from two different kinds of sources — primary and secondary. Primary PM sources can be derived from both human and natural activities. A significant portion of PM sources is generated from various human activities. These include activities that cause dust to be released into the air, such as farming, industrial processes, smokestacks, burning fossil fuels, and building and demolition. Natural activities such as forest fires and volcanic eruptions sources also contribute to the overall PM problem.

Secondary PM is formed in the lower atmosphere by chemical reactions. PM2.5 precursors such as nitrogen oxides (NOx), volatile organic compounds (VOCs), sulfur dioxides (SO_2), and ammonia contribute to the formation of secondary fine particulates. Various sources, including power plants, industry, vehicles, small businesses, buildings, and homes, emit precursors that lead to the formation of PM2.5. (Zhang, 2015) Because secondary pollutants synthesize in various ways and the creation is poorly understood, they are more difficult to manage. They are a natural byproduct of the environment, resulting in issues like photochemical haze.

Particulate matter is made up of tiny solid or liquid droplets so minutely that breathing them in can have major negative effects on one's health. The PM10 particles, with diameters generally 10 micrometres and smaller, can get deep into your lungs, and some may even get into your bloodstream. (Brunekreef, 2005) The PM2.5 fine particles, with diameters that are generally 2.5 micrometres and smaller, pose the greatest health risk. Fine particles are also the leading cause of visual impairment (haze) in many places in the US, including many of our beautiful national parks and wilderness regions. Can you imagine how small it is, 2.5 micrometres? The average human hair is about 70 micrometres in diameter; this means the particulate matter is making it 30 times larger than the largest fine particle.

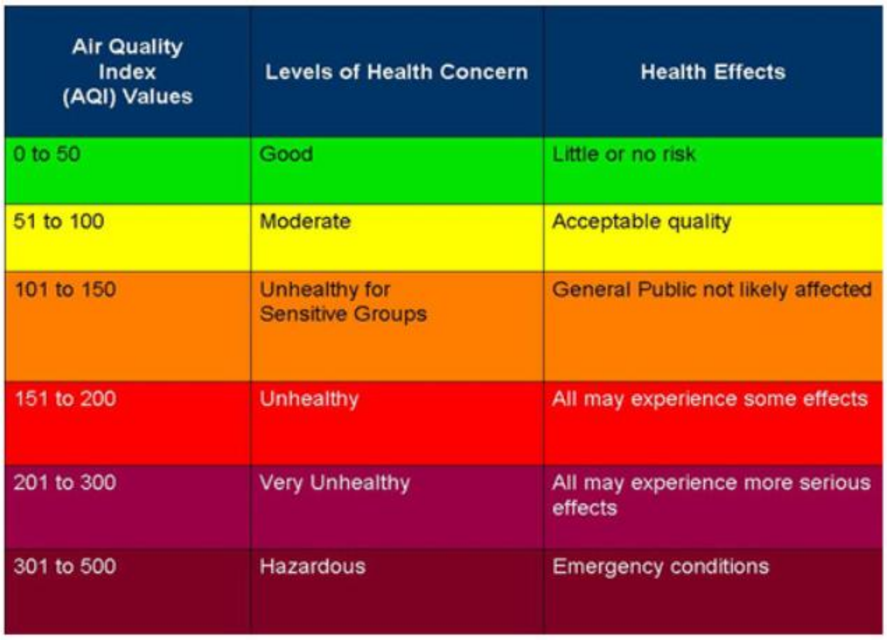
A healthy human heart and lungs may suffer if exposed to these particles. Numerous scientific studies have connected exposure to particle pollution to several issues, including premature death in people with heart or lung disease, nonfatal heart attacks, irregular heartbeat, and increased respiratory symptoms, such as airway discomfort, coughing, or breathing issues. The most susceptible groups to the effects of particle pollution exposure include youngsters, elderly individuals, and people with heart or lung conditions. Also, particulate matter will affect the environment; Wind may carry particles across incredible distances before they land or float to the surface. This settling might result in any number of things, depending on their chemical makeup. The problems include contributing to acid rain impacts, altering the nitrogen balance in vast river basins and coastal seas, and rendering lakes and streams acidic.

2.2.3 Air quality index

The Air Quality Index, or AQI, is a numerical scale that reports air pollution in a particular area. (Murena, 2004) The AQI make it easy to understand the air quality summary, allowing people to take necessary precautions to protect their health. The AQI is measured by considering the concentration of different pollutants. AQI system includes 5 major air pollutants that could cause potential harm to human health should they reach unhealthy levels, which are O_3, PM2.5, PM5, CO, SO_2, and NO_2.

The AQI typically ranges from 0 to 500, with higher values indicating more significant pollution levels and a higher risk to human health. The scale is divided into six categories, ranging from "Good" to "Hazardous," with each type corresponding to a different range of AQI values. Governments and organizations often use the AQI to inform the public of the current air quality conditions and to issue health advisories when pollution levels are high. Table 2.2 show the levels of health concern and health effects for each range of the AQI.

Table 2.2: The Air Quality Index (bentoncleanair, n.d.)



Air Quality Index (AQI) Values	Levels of Health Concern	Health Effects
0 to 50	Good	Little or no risk
51 to 100	Moderate	Acceptable quality
101 to 150	Unhealthy for Sensitive Groups	General Public not likely affected
151 to 200	Unhealthy	All may experience some effects
201 to 300	Very Unhealthy	All may experience more serious effects
301 to 500	Hazardous	Emergency conditions

2.3 Existing methods of air quality monitoring

Monitoring air quality requires measuring air pollutants in the atmosphere to ascertain their concentrations and ensure they are within the safe limits for people, animals, and plants. Evaluating the effects of human activity on the environment and spotting possible dangers to public health depends heavily on the monitoring of air quality. Several existing air quality methods are now in use, ranging from straightforward and affordable to more complex and pricier. There are a few standard techniques used worldwide to check air quality. Four common monitoring air quality methods, passive sampling, active sampling, continuous emissions monitoring, and low-cost sensors will be discussed below.

First, passive sampling methods are a type of air sampling technique that involves using a sampling device that collects air samples over a period without needing an external power supply or a pump. This method is often used to measure indoor air quality in residences and outdoor work environments and monitor ambient air quality. Passive sampling techniques come in various forms, such as filter sampling, sorbent tube sampling, and diffusive sampling. The primary benefits of passive sampling methods are their low cost, convenience, and ability for long-term sampling. (Bohlin et al, 2007) However, certain pollutants haven't been examined. Therefore, laboratory testing is required.

Active sampling is a more complex air quality monitoring method involving a pump to draw air through a filter or adsorbent material. The pollutants are trapped on the filter or adsorbent material, and the concentration levels are analyzed in a laboratory. Active sampling techniques come in various forms, such as high-volume sampling, low-volume sampling, and personal sampling. Active sampling is more expensive than passive sampling but provides more accurate and reliable results. (Bohlin et al, 2007)

Continuous emissions monitoring is one of the methods of air quality monitoring involving sensors to measure the concentration levels of pollutants in the air continuously. This method is commonly Utilized in industrial settings where emissions are generated from a single source. Continuous emissions monitoring provides real-time data on the concentration levels of pollutants and allows for immediate corrective action if necessary. (Siebenaler et al., 2016)

Low-cost sensors are an air quality monitoring technology that is becoming increasingly popular due to their lower cost and ease of use. (Karagulian et al.,2019) These are typically smaller, and low-cost sensors are ideal for community-based air quality monitoring initiatives Low-cost sensors can provide real-time data on the concentration levels of pollutants. While low-cost sensors have some limitations compared to traditional air quality monitoring equipment, such as lower accuracy and precision, they offer a valuable tool for communities and individuals concerned about air quality who want to protect their health and the environment.

2.4 Embedded systems and the hardware

2.4.1 Embedded systems

An embedded system is a computer system created to perform particular functions inside a greater object or system. It often consists of hardware and software elements and is intended to be portable, effective, and dependable. In the modern era, embedded systems may be found in various products, such as vehicles, medical equipment, industrial apparatus, household appliances, etc. (Kermani et al, 2013)

An embedded system's hardware generally consists of a microcontroller or microprocessor, which acts as the system's "brain" and handles most computing operations. A range of input and output (I/O) devices, including sensors, actuators, displays, and communication interfaces, are frequently paired with this. These parts are frequently highly specialized and tailored for certain functions, such as temperature measurement, motor control, or wireless data transmission.

The software components of an embedded system typically consist of a combination of firmware and application software. (Berger, 2001) Firmware is software stored permanently on the system's hardware and is responsible for performing low-level tasks like booting the system and controlling hardware components. On the other hand, application software is responsible for implementing the system's specific functionality, such as monitoring temperature or controlling a motor.

The ability of embedded systems to increase system performance and dependability is one of their main advantages. Embedded systems frequently outperform general-purpose computer systems regarding job efficiency and accuracy because they integrate specialized hardware and software components. This may result in better system functionality, lower energy use, and more dependability. potential of embedded systems to allow new features and capabilities in current systems is another advantage. (Kocher et al, 2004) One use of an embedded system is to give household appliances innovative capabilities like automated temperature management or remote monitoring and control. This might enhance the appliance's use and value while improving the user experience.

Embedded systems are used in many different types of applications and sectors. For instance, embedded systems are utilized in engine control units, navigational systems, and entertainment systems in the car sector. Devices like insulin pumps and pacemakers in the medical sector employ embedded systems. In aerospace, embedded systems are utilized in communication and flight control systems.

2.4.2 Arduino

Arduino is an open-source platform for developing electronic projects and prototypes. It comprises a microcontroller board, a software development environment, and a variety of sensors and actuators that can be quickly added to the board. (Zlatanov, 2016) The platform is suitable for hobbyists, students, and professionals alike since it is designed to be user-friendly and accessible to those with little to no expertise in electronics or programming.

Arduino's portability is one of its key advantages. The platform is designed to be simple, even for those without programming or electrical skills. Arduino uses a similar syntax based on popular and relatively simple programming languages such as Python and C++. With various tools and features that make it simple to develop and evaluate uses, the software development environment is also designed to be straightforward and user-friendly. The flexible nature of Arduino is another benefit. The platform may produce many applications, from straightforward LED displays and sensors to intricate robotics and home automation systems. A powerful and flexible platform for experimentation and discovery, Arduino is more for its use in scientific research and educational applications. (Badamasi, 2014)

Arduino is known as an open-source platform. The source code and the designs are freely available for anyone to modify, use, and share. The Arduino community is one of the critical strengths of the platform, and it gives a wealth of knowledge and support to anyone who wants to learn, design, and modify more about electronics and programming. (Zlatanov, 2016) Another benefit of Arduino is its price. Due to their accessibility and affordability, Arduino boards make it simple for anybody to dabble with electronics and programming without spending much money. Finding the ideal board for your project is simple, thanks to the wide range of sizes and configurations that Arduino boards come in.

The modularity of Arduino is one of its distinguishing qualities. The platform is intended to be modular, with various sensors, actuators, and other parts that can be quickly linked to the board. Projects may easily add new features and capabilities because of this flexibility, making it simple to reuse components across several projects.

Arduino is a powerful, flexible, user-friendly platform for creating electrical projects and prototypes. With a wide variety of sensors and actuators, an easy-to-use and straightforward programming environment, and an active and supportive user and developer community. Regardless of your experience level, Arduino offers a rewarding and enjoyable way to learn about electronics and programming.

2.4.3 MQ series sensor

The MQ series is a gas sensor that can identify various gases. These sensors are frequently used in commercial, healthcare, and environmental settings. The MQ series gas sensors work by recognizing variations in resistance brought on by exposure to various gases. These sensors are made to be used in various locations, including indoor and outdoor areas. According to the gas, many sensors locate and categorize various gases. These sensors can detect various gases, such as CO, CH_4, O_3, NO_2, and others. (Sai et al, 2019) Figure 2.1 shows the various type of MQ sensor in the market.

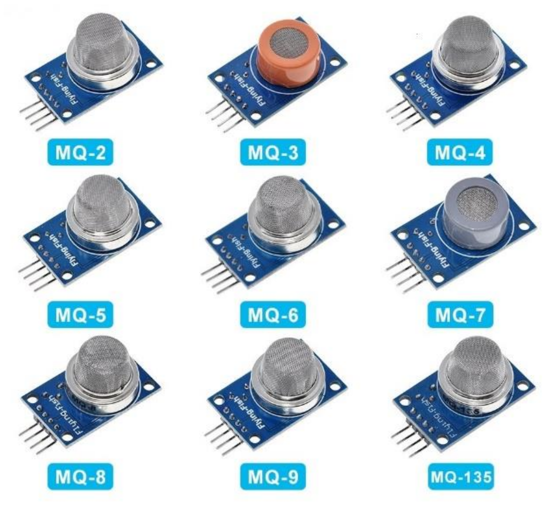


Figure 2.1: The MQ sensor in the market (ASK Electronics, 2023)

A ceramic substrate covered with a metal oxide layer is the base for the MQ series gas sensors. Changes in the gas concentration affect the metal oxide layer's sensitivity, which causes it to alter its resistance. (Hadi et al, 2022) The gas sensors of the MQ series may be incorporated into various systems and are comparatively simple to operate. A circuit is needed to turn the resistance change of the sensors into an output signal and a steady power source. The sensitivity of the sensors may also be changed, and they can be tuned to detect specific gases.

The MQ series gas sensor's low cost compared to other gas sensors is one advantage. As a result, they are frequently used for applications where price is crucial. (Hadi et al, 2022) The frail sensors are perfect for circumstances where safety is a concern since they can detect minuscule levels of gases. Another advantage of the MQ series gas sensors' tiny size is that it makes it simpler to incorporate them into various systems. The sensors are suitable for dangerous circumstances because of their robustness and lightweight.

Applications for the MQ series gas sensors include environmental monitoring, workplace safety, and medical diagnostics. The sensors are used in environmental monitoring to find air pollution and keep track of the air quality in various places.

The sensors used in industrial safety monitor the concentration of gases in the workplace while scanning for gas leaks. The sensors used in medical diagnostics gather up gases in the breath, which can be utilized to identify certain medical conditions.

2.5 Internet of Things

Nowadays, smart homes, wearable technology, smart cities, smart supply chain management, agricultural IoT, and connected healthcare systems are examples of IoT applications worldwide. (Hui et al, 2017) People use the IoT and apply it to improve their living standards and quality. Applications for the IoT can assist in boosting productivity, efficiency, and safety in various contexts, including healthcare, agriculture, and transportation. The IoT can help people or industries to increase productivity, efficiency, and safety, making work faster and more precise.

The IoT system comprises four elements: the sensor or device, cloud, gateway, and user interface. (Yang et al, 2017) The sensor is a hardware component that monitors environmental changes and collects information. The device detects and measures physical quantities such as temperature, pressure, humidity, light, sound, motion, or other environmental factors. Sensors are critical components of IoT systems because they collect data from the surrounding environment and transmit it to the cloud via a gateway for further analysis.

The IoT gateway is a physical device or software program that bridges IoT devices and the cloud platform or other external networks in an IoT system. (Chen et al, 2011) It serves as a wireless access point to connect IoT devices and the Internet, enabling communication between devices in the IoT network and the Internet. All data moving between IoT devices, and the cloud passes through an IoT gateway, which can be either a dedicated hardware appliance or a software program. The primary function of the gateway is to route data between IoT devices and the cloud, and it may also perform tasks such as pre-processing, data aggregation and filtering of data, and providing additional security.

The Cloud of IoT is a crucial component of the IoT system, offering a scalable and flexible platform for storing and processing massive amounts of data generated by IoT devices. It enables real-time data processing, AI applications, and machine learning, allowing organizations to derive valuable insights and make better-informed decisions. In addition, the Cloud of IoT provides enhanced security features, protecting IoT devices and data from cyber threats using data encryption, firewalls, and intrusion detection.

The user interface consists of the features of a user interacting with a computer system. This includes screens, buttons, website pages, icons, and others. The software and programs on computers, cell phones, and other devices are the most apparent examples of user interfaces. Good user interface design is essential for creating intuitive and user-friendly IoT systems. As IoT evolves, designers and manufacturers must align product development with business models and user trust to design integrated context and minimize steps between user and objective.

CHAPTER 3

METHODOLOGY

3.1 Overview project flow

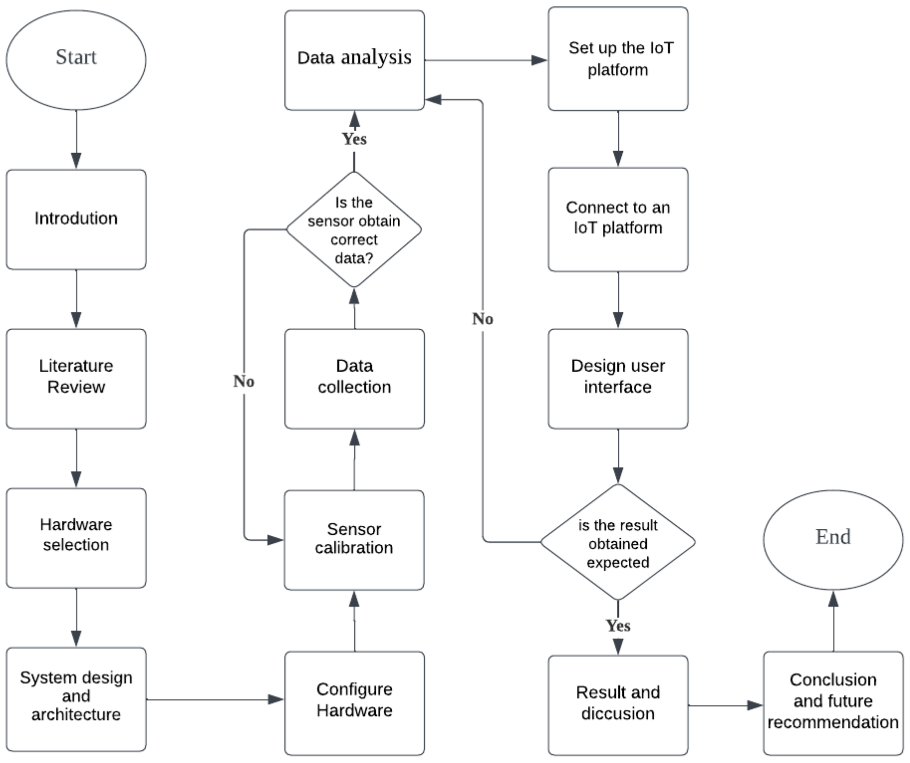


Figure 3.1: The overview project flow

Figure 3.1 shows the overview project flow chart for this project. Initially, the introduction was to identify the project's background, problem statement, and objective. Then, research the article to do the literature review for the project. At the same time, the hardware selection section confirmed what type of hardware would be used for this project, and the system design and architecture were done after the hardware selection section. After configuring the hardware, the sensor calibration aims to obtain the result wanted for the project. The IoT platform was set up and designed for the IoT user interface platform; the IoT platform aims to display the overall impact on the user. After obtaining the result, the discussion and the result were written in the report. Lastly, the final report included the conclusion and recommendation.

3.2 System architecture

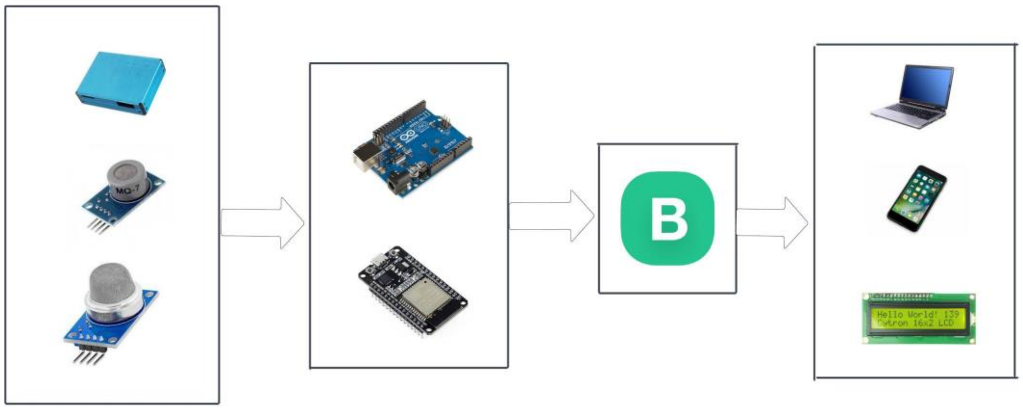


Figure 3.2: The system architecture

Figure 3.2 shows the system architecture for this project. The first level was the sensor. The various sensors were used to detect and measure the air quality, temperature, humidity, and others. The second level was the ESP32 and Arduino Uno. The detected or measured data will transfer to the ESP32 and serial communication with Arduino UNO to transfer the real-time data for the user. The third level was the IoT platform Blynk, which allows the user. To prototype, deploy, and remotely manage connected electronic devices at any scale. The last level was the user interface. Users can view the real-time data through the Blynk, whether on a website or their smart device.

3.3 System overflow

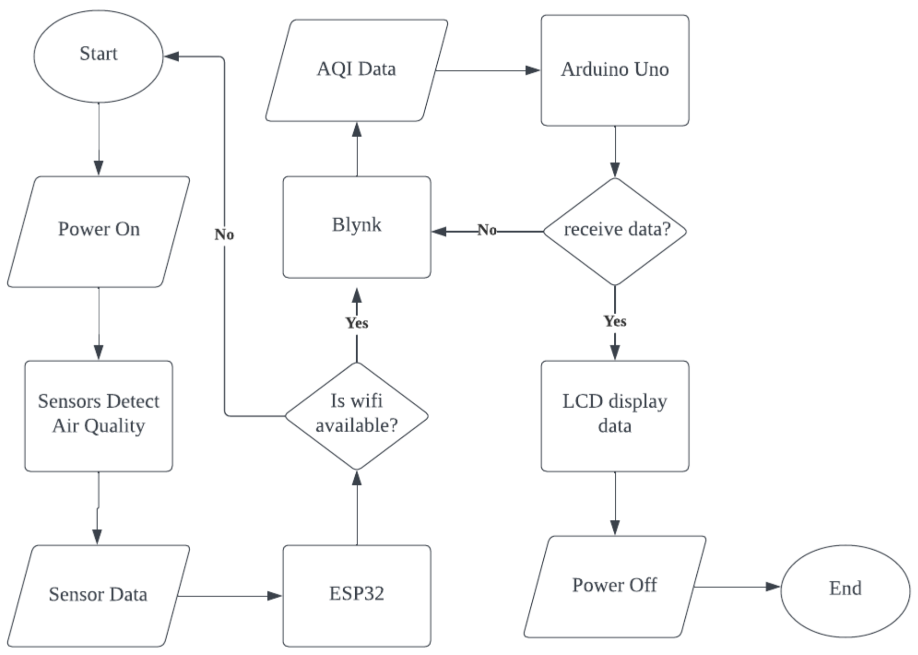


Figure 3.3: Flowchart of system overflow

Figure 3.3 shows the flowchart of system overflow. When the power of the hardware was on, the sensors start to detect the air quality. After detecting the air quality, the data was transferred to the WiFi modular ESP32. After receiving the air quality data, ESP32 transferred the air quality data to the IoT platform Blynk. To transfer data to Blynk, WiFi was required. Blynk interface or software allows users to track the air quality data anytime and anywhere.

Next, the AQI data was received from the ESP32 via serial communication to the Arduino UNO. If the Arduino UNO successfully receives the data, Arduino UNO sends AQI data to two separate LCDs on the hardware to display the real-time AQI to the users. The LCD will help users who don't have a mobile phone or other. Users can track the real-time AQI via the LCD on the hardware.

3.4 Software and Hardware Selection

The hardware selection for this project, Arduino Uno was used as a microcontroller board. MQ serial sensor will be used to detect the air quality. The Wi-Fi microchip ESP32 was used as the IoT project to enable devices to connect to wireless networks, offer Wi-Fi connection, and allow communication with other devices or the internet. The sensor from Bosch manufacturer BME280 was used to measure the temperature and humidity in the air. Lastly, the PMS7003 was used to measure particulate matter PM2.5 and PM10.

For the software selection, the Arduino IDE was used to write and upload code to Arduino Uno microcontroller boards. Blynk was used it as an IoT platform that allows the creation and control of this project. Blynk software enables users to prototype, deploy, and manage connected electronic devices at any scale. (Tech Explorations, n.d.)

3.5 Wi-Fi modules selection

Many types of WiFi modules were in the market with different features, prices, and manufacturing. The ESP32 and ESP8266 have commonly used WiFi modules in IoT projects. When choosing between ESP32 and ESP8266, it was essential to consider the project's specific needs. The ESP8266 may be a better choice for simple IoT projects due to its lower price and better-developed libraries and features. However, the ESP32 has more powerful features, making it a better choice for larger and more complex projects. The Table 3.1 shows the different features of the ESP32 and ESP8266.

Table 3.1: The different of the ESP32 and ESP8266

Feature	ESP32	ESP8266
Processor	Dual-core Xtensa LX6 @ 240 MHz	Single-core Xtensa LX106 @ 80 MHz
Operating voltage	2.2V - 3.6V	3.0V - 3.6V
RAM	Up to 520KB	80 KB (instruction/data) + 16 KB for system
ROM	Up to 448KB	No Programmable
Flash memory	Up to 16 MB	12 KB - 16 MB
Bluetooth	Bluetooth Classic + BLE	No Bluetooth support
Wireless connectivity	Wi-Fi (802.11b/g/n) + Bluetooth	Wi-Fi (802.11b/g/n)
ADC pins	18 pins	1 pin
GPIO pins	48 pins	17 pins
PWM	16	4
SPI Interfaces	4	1
12C Interfaces	2	0
UART Interface	3	2

The ESP32 has 48 pins of GPIO in total, but not all pin was exposed in the ESP32 board, and some pin cannot be used, depending on the model. (Randomnerdtutorials, 2018) ESP32 has 3 UART: UART 1, UART 2, and UART3. UART 1 defaults on GPIO 3 and GPIO 1, UART 2 pins GPIO 9 and GPIO 10, and UART 3 ON pins GPIO 16 and GPIO 17. (Lab, 2022) Two I2C pins on the ESP32 were GPIO 21 and GPIO 22. And 16 channels of PWM on the ESP32 board. (Teja, R, 2021)

For ESP8266, there were 17 pins of GPIO pins in total. Similar to ESP32, not all pins were exposed in the board. The ESP8266 lacks hardware I2C pins, although they can be added in software. The GPIO 4 and GPIO 5 pin can be used as the I2C pin of ESP8266. SPI pins that can be used for SPI communication for ESP8266 were

GPIO 12, GPIO 13, GPIO 14, and GPIO 15. ESP 8266 has two UART: UART 1 (GPIO 1 and GPIO 3) and UART 2 (GPIO 2 and GPIO 8). UART1 on the ESP8266 can only transmit data, not receive it. (Pieter P, 2022) Four-channel PWM on the ESP8266 board, and the PWM frequency range was adjustable from 1000 µs to 10000 µs. (Make-It.ca, n.d.)

3.6 Programming language selection

In this project, C and C++ languages will be the main coding languages. Arduino uses C and C++ to program. They were two different languages, even if C++ and C were interoperable. Compared to C language, a process-oriented programming language, C++ was object-oriented. (Shiksha, 2020) Object-oriented programming was eventually added to the C language core of the original Arduino. C and C++ were now used in the most recent Arduino core library.

The Arduino programming language consists of program structures, variables, and functions. Functions were blocks of code that can be called from other parts of the program. Some common functions in Arduino include setup() and loop(). Variables were used to store data in a program. In Arduino, variables can be declared using the int or float data types. The structure of an Arduino program typically includes a setup() function, executed once at the beginning of the program, and a loop() function, executed repeatedly until the program was stopped.

In Arduino programming, void setup() and void loop() were two important functions. The void setup() function will only run once when the program starts to execute and was mainly used to initialize variables, set pin modes, initialize serial communication, etc. This function was mainly to prepare your program for subsequent work. The void loop() function will be executed in a loop after the setup() function finishes running, and it was the primary execution part of the program. The function typically includes sensor readings, variable computations, control statements, delays, etc.

In programming languages, all data must specify the variables. (LaunchSchool, 2019) In a data structure, a variable was described as a set of operations on this set of values and a collection of values. Each variable needs to be used in a specific place. The variable controls how the computer's memory stores the bits representing these values. When a variable was declared, the programmer needs to specify its variable to store different data types.

3.7 Hardware connection

Hardware connections were essential for interacting and collaborating with different hardware components within computer systems and electronic devices. These connections encompass diverse physical interfaces and cables that transfer data and power between components. The Hardware connection will point out the connection with different devices and the methodology of the hardware connection. The pinout diagram and the pin allocation of the different devices will be listed in this section. The devices will follow the manufacturer's instructions or the datasheet.

3.7.1 Arduino UNO connection

Arduino Uno was used as the microcontroller in this project. The purpose use of Arduino UNO is that Arduino UNO will receive real-time data from the ESP32 and display the real-time data on two separate LCDs. The main reason for using the Arduino UNO as the microcontroller for this project was that it was cheap, easy to use, and easy to find in the market. Arduino was designed to make utilizing microcontrollers as simple as possible. Compared to reading a standard microcontroller user manual, the code was considerably simpler to understand, the hardware is simple, and the IDE was straightforward. Regardless of the user's operating system, developing and uploading code to the board was simple using the Arduino software because it operates on Windows, macOS, and Linux. Figure 3.4 shows the Arduino UNO pinout diagram.

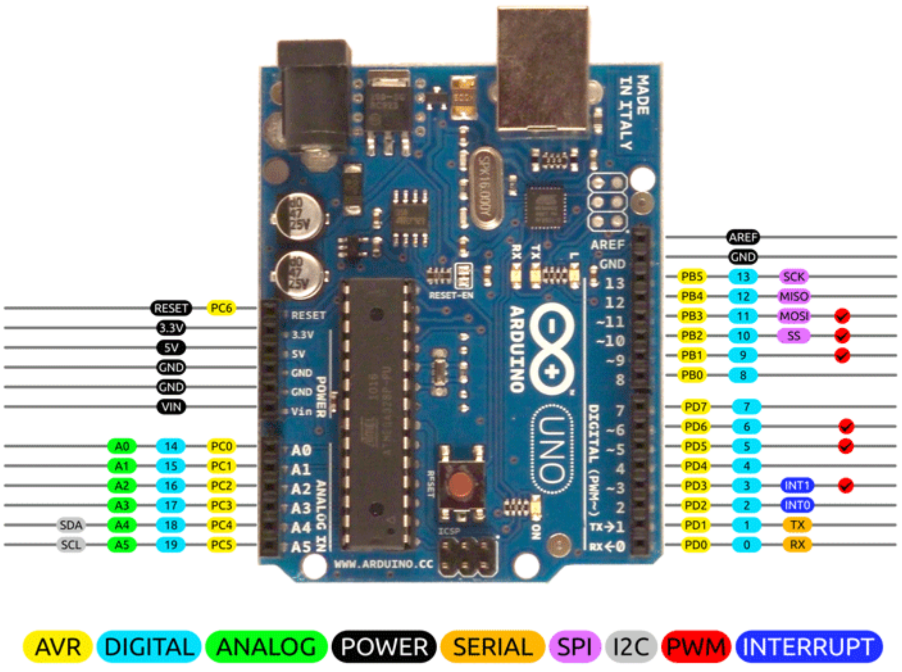


Figure 3.4: The Arduino UNO pinout diagram (Components101, 2018)

The Arduino Uno microcontroller board having 14 digital input/output pins, six analog inputs, a power connector, a USB port, an ICSP header, and a reset button. The Arduino Uno's pins may be utilized for various tasks, including operating motors, LEDs, and reading sensors. Three different power sources were available for the Arduino Uno: a power jack, a USB connection, and an unregulated supply that delivers 6 to 20 volts to the VIN pin.

For the project, the 5V voltage pin provided 5V voltage to devices such as the MQ135 sensor, MQ7 sensor, PMS7003, and another sensor to operate their function. The electrical circuit was closed using a GND pin, which also serves as the circuit's shared logic reference level. Pin 0 (RX) and Pin 1(TX) connect to the TX0 (GPIO 1) and RX0 (GPIO 3) pins of ESP32. Pin 2 to pin 13 will be used for the 16×2 LCD. Pin 2 to pin 7 will connect by LCD 1, and in 8 to pin 13 will connect by LCD 2

For sensor calibration, pin A0 of the Arduino UNO connected to the pin A0 of the MQ 135 and MQ 7 sensors. The pin A0 of the MQ135 and MQ7 returns an analog value based on the gas concentration. The A0 pin returns HIGH if the gas concentration exceeds a certain value. The potentiometer can set this value on the board. The Arduino UNO 5V pin and GND pin connect to pin VCC and GND of the MQ135 and MQ7 sensors. Figure 3.5 shows the Arduino UNO used in the project, and Table 3.2 shows the pin allocation of Arduino UNO.

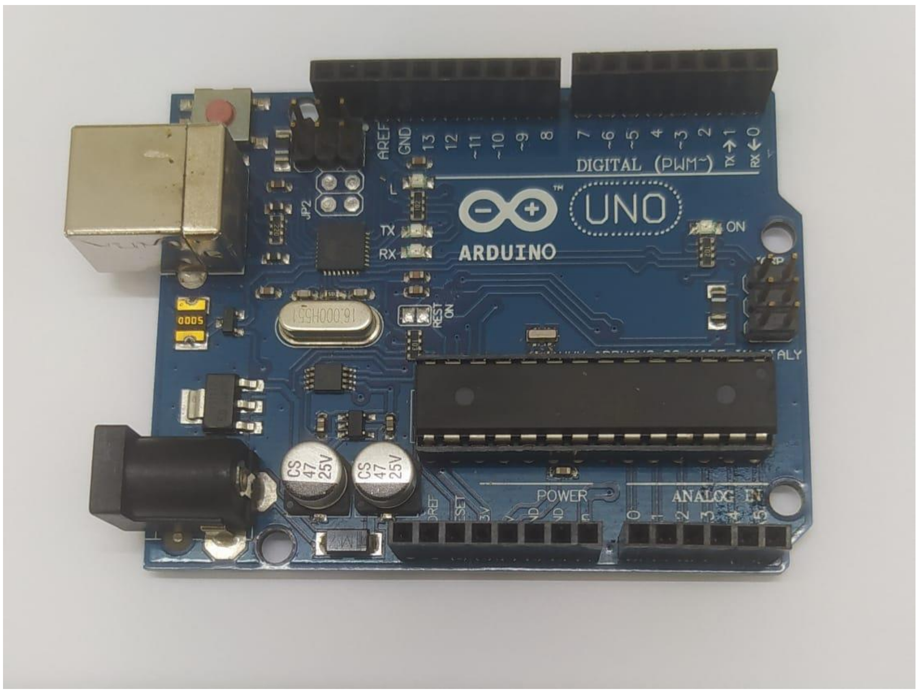


Figure 3.5: The Arduino UNO used in the project.

Table 3.2: Pin allocation of Arduino UNO

Arduino UNO	
Pin Name	Pin Connected
RESET	-
3.3V	-
5V	Outputs a regulated 5V
GND	Provide a common logic reference level
GND	-
VIN	-
A0	Pin A0 of MQ135 for sensor calibration
A1	Pin A0 of MQ7 for sensor calibration
A2	-
A3	-
A4	-
A5	-
0	TX0 pin of ESP32
1	RX0 pin of ESP32
2	RS pin of LCD1
3	E pin of LCD1
4	D4 pin of LCD1
5	D5 pin of LCD1
6	D6 pin of LCD1
7	D7 pin of LCD1
8	RS pin of LCD2
9	E pin of LCD2
10	D4 pin of LCD2
11	D5 pin of LCD2
12	D6 pin of LCD2
13	D7 pin of LCD2
AREF	-
GDN	-

3.7.2 ESP32 connection

The ESP32 Devkit V1 was chosen as the IoT section's Wi-Fi module for this project. ESP32 was received the data from their sensors and transfer it to the Arduino UNO to display the real-time data on the LCD. The ESP32 Devkit V1 has 30 pins total on the board since the ESP32 has different types of versions of different numbers of pins. Compared with other versions of ESP32, the ESP32 Devkit V1 has no six GPIO pins, which were GPIO 8 (SDI/SD1), GPIO 7 (SDO/SD0), GPIO 6 (SCK/CLK), GPIO 11 (CSC/CMD), GPIO 10 (SWP/SD3), and GPIO 9 (SHD/SD2). Since the ESP32's GPIO pins 6 to 11 were internally attached to its flash memory, these pins should not be utilized for any other purpose. (Flux, 2023) Figure 3.6 shows the pinout diagram of ESP32 Devkit V1.

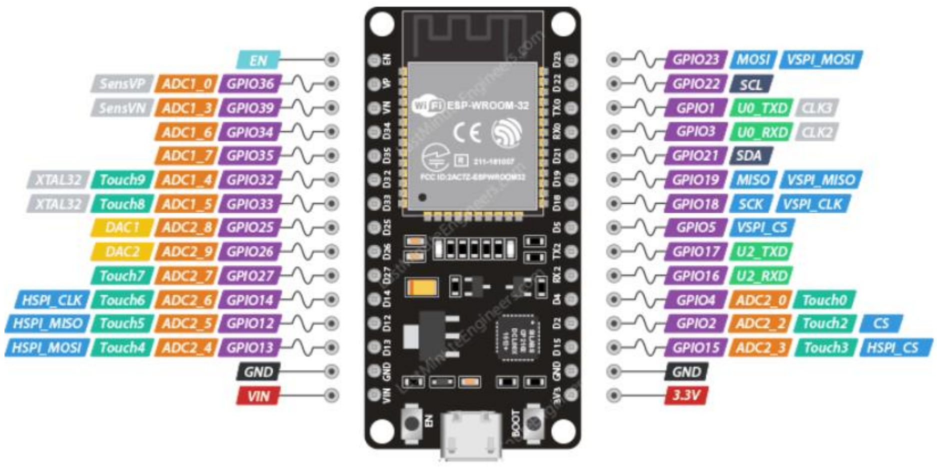


Figure 3.6: The pinout diagram of ESP32 Devkit V1 (Last Minute Engineers, 2022)

For this project, ESP32 was received the data from their sensors and transferred it to the Arduino UNO to display the real-time data on the LCD. Also, ESP32 was used with the Blynk IoT platform to create connected products. The ESP32 Devkit V1 has 30 pins in total. The 25 GPIO pins may be programmed with various functionalities by modifying the proper registers. But only input may be utilized on GPIO pins 36 (VP), 39 (VN), 34, and 35. Additionally, internal pull-up and pull-down resistors were absent from them. (Upesy, 2022) In this project, MQ sensors were connected with pins GPIO of ESP32.

For serial communication, the pin TX0 (GPIO 1) and pin RX0 (GPIO 3) of ESP32 was connected to the pin 0 (RX) and pin 1 (TX) of Arduino UNO to be used as the serial communication between ESP32 and Arduino UNO. Pin TX2 (GPIO 17) and pin RX2 (GPIO 16) will connect to PMS7003 sensor pin TXD and RXD to transfer the detector data ESP32.

A USB-A to Micro cable was used for ESP32 as a micro-USB cable to program and power the board. It was the most common type of cable used with the ESP32. (Kashif, n.d.) When uploading a sketch or software to the ESP32, a serial connection with the host computer was implemented via the Micro-USB cable. The Micro-USB connection used to connect the ESP32 to the computer makes transferring code and communicating with the board simple. The ESP32 board may also be powered with a Micro-USB cable; insert one end of the cable into the USB port on your computer or any USB-compatible power source and the other end into the USB port on the ESP32 development kit. This offers a simple and dependable method of powering the board. It was a common cable that was simple to locate and may be used with the ESP32 without having any compatibility problems.

Figure 3.7 shows the ESP32 used in the project, and Figure 3.8 shows the USB-A to Micro cable. For the pin allocation, Table 3.3 shows the pin allocation of ESP32 of the project.

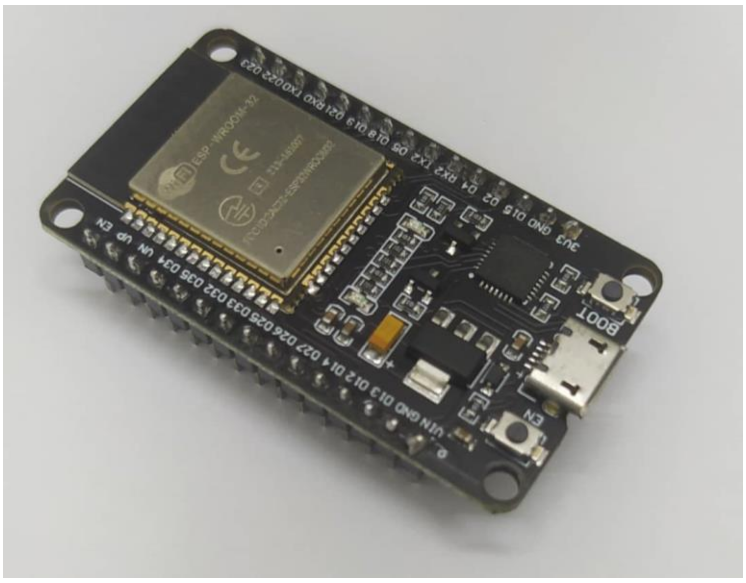


Figure 3.7: The ESP32 used in the project

Figure 3.8: The USB-A to Micro cable

Table 3.3: Pin allocation of ESP32

ESP32			
Pin	Label	GPIO	Pin Connected
1	EN	N/A	-
2	VP	36	-
3	VN	39	-
4	D34	34	Pin A0 of MQ135
5	D35	35	Pin A0 of MQ7
6	D32	32	-
7	D33	33	-
8	D25	25	-
9	D26	26	-
10	D27	27	-
11	D14	14	-
12	D12	12	-
13	D13	13	-
14	GND	N/A	-
15	VIN	N/A	-
16	3V3	N/A	Outputs a regulated 3.3V
17	GND	N/A	Provide a common logic reference level
18	D15	15	-
19	D2	2	-
20	D4	4	-
21	RX2	16	Pin TXD of PMS7003
22	TX2	17	Pin RXD of PMS7003
23	D5	5	-
24	D18	18	-
25	D19	19	-
26	D21	21	Pin SDA of BME280
27	RX0	3	Pin 0 of Arduino UNO
28	TX0	1	Pin 1 of Arduino UNO
29	D22	22	Pin SCL of BME280
30	D23	23	-

3.7.3 MQ sensor connection

In this project, the MQ135 and MQ7 were used for gas detector sensors. The pin 5V and pin GND of Arduino UNO connect with the pin VCC of MQ 135 and MQ 7 for operation functions. The pin A0 of MQ 135 and MQ 7 connect to ESP32. Since the

ESP32 will be damaged if applied high voltage was applied, the logic level converter will safely step down 5V signals to 3.3V.

Figure 3.9 shows the MQ135 used in the project, and table 3.4 shows the pin allocation of MQ135.

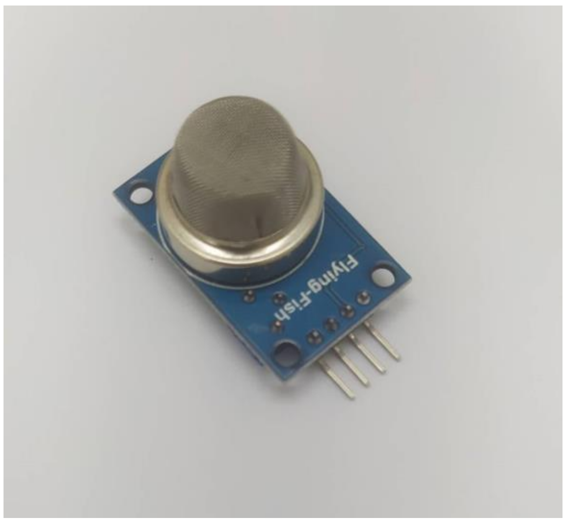


Figure 3.9: The MQ135 used in the project

Table 3.4: Pin allocation of MQ135

MQ 135	
Pin name	Pin name
VCC	5V voltage provided from the Arduino UNO
GND	Ground voltage of the system
D0	-
A0	Pin 4 (GPIO 34) of ESP32

For MQ7 sensor, Figure 3.10 shows the MQ7 used in the project, and Table 3.5 shows the pin allocation of MQ7.

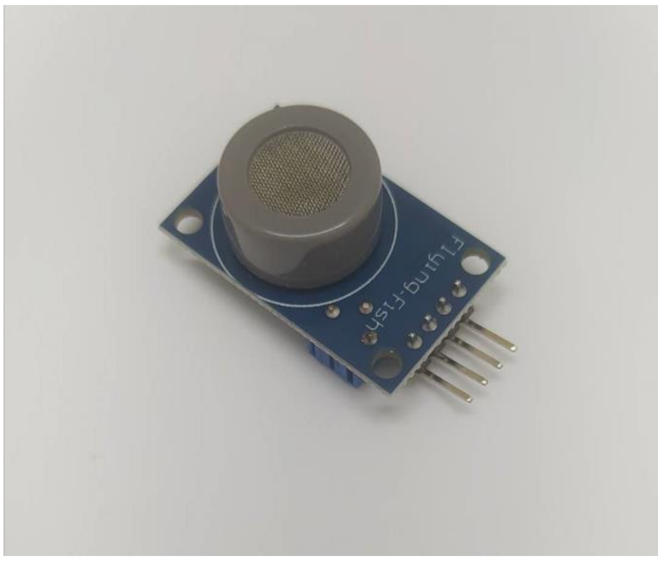


Figure 3.10: The MQ7 used in the project

Table 3.5: Pin allocation of MQ7

MQ 7	
Pin name	Pin name
VCC	5V voltage provided from the Arduino UNO
GND	Ground voltage of the system
D0	-
A0	Pin 5 (GPIO 35) of ESP32

3.7.4 PMS7003 connection

The PMS7003 was a digital and universal particle concentration sensor that used to obtain the number of suspended particles in the air.

To interface with the PMS7003 sensor for measuring particulate matter concentrations in the air, the sensor typically comes with six pins, but in the project, just used 4 pins which were VCC, GND, RX, and TX pins. The VCC pin had connected to a 5V power supply for Arduino UNO, while the GND pin needs to be linked to the ground of the power source. For communication, the RX pin, responsible for receiving data from ESP32, had to be connected to the ESP32's transmit (TX2) pin which was pin 22 (GPIO 17). The TX pin, which transmits data to ESP32, had to be connected to the ESP32's receive (RX2) pin which was pin 21 (GPIO 16). Figure 3.11 shows the PMS7003 was used in the project and, Table 3.6 shows the Pin allocation of PMS7003.

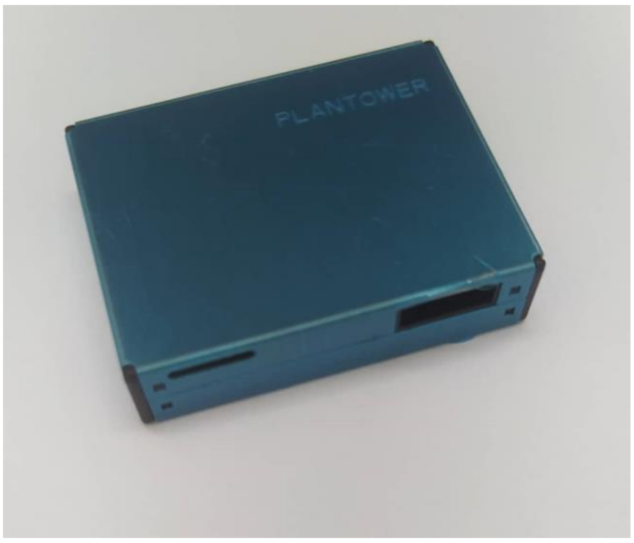
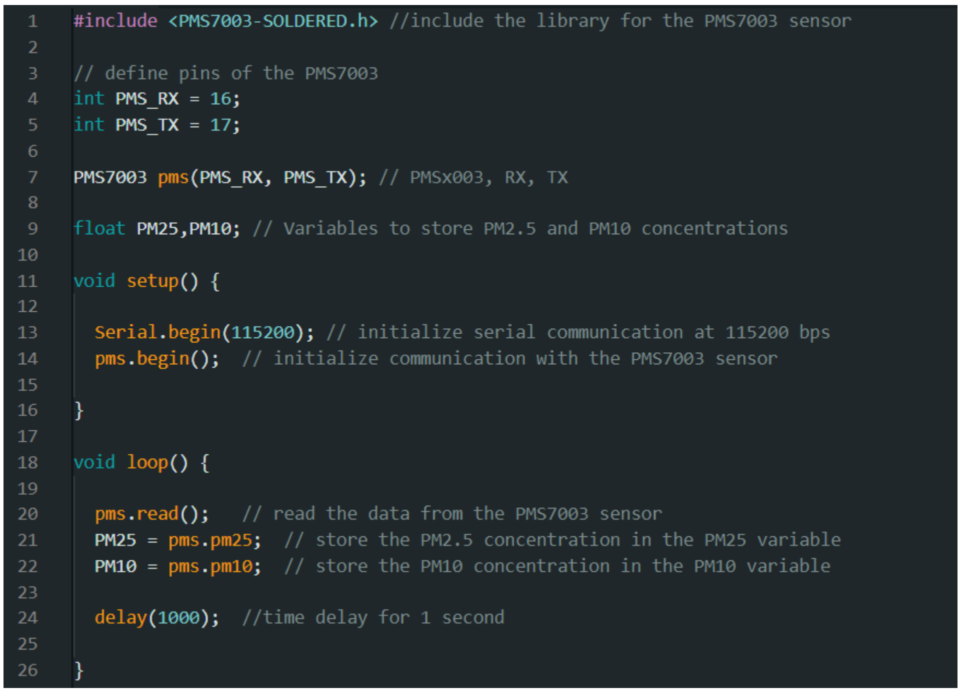


Figure 3.11: The PMS7003 used in the project

Table 3.6: Pin allocation of PMS7003

PMS7003	
Pin Name	Pin Connected
VIN	5V provided from the Arduino UNO
GND	Ground voltage of the system
RXD	Pin 22 (GPIO 17) of the ESP32
TXD	Pin 21(GPIO 16) of the ESP32



```
#include <PMS7003-SOLDERED.h> //include the library for the PMS7003 sensor

// define pins of the PMS7003
int PMS_RX = 16;
int PMS_TX = 17;

PMS7003 pms(PMS_RX, PMS_TX); // PMSx003, RX, TX

float PM25,PM10; // Variables to store PM2.5 and PM10 concentrations

void setup() {

  Serial.begin(115200); // initialize serial communication at 115200 bps
  pms.begin();  // initialize communication with the PMS7003 sensor

}

void loop() {

  pms.read();   // read the data from the PMS7003 sensor
  PM25 = pms.pm25;  // store the PM2.5 concentration in the PM25 variable
  PM10 = pms.pm10;  // store the PM10 concentration in the PM10 variable

  delay(1000);  //time delay for 1 second

}
```

Figure 3.12: The code for PMS7003

For the code shown in Figure 3.12, the library <PMS7003-SOLDERED.h> was included in the code. Pin 21 (GPIO 16) and pin 22 (GPIO 17) have been defined as integer variables ‘PMS_RX’ and ‘PMS_TX’ to use as the serial communication between the sensor and the microcontroller. PM2.5 and PM10 have define as floating-point number values ‘PM25’ and ‘PM10’. In the void setup() function, 115200 bits per second of baud rate was used as initialized serial communication. In the void loop() function, the code pms.read() was to read the data from detector PM2.5 and PM10 in the air, and the time delay for every reading was 1 second. Overall, the code design for the sensor PMS7003 kept looping to read the real-time data and send the data to the microcontroller ESP32.

3.7.5 BME280 connection

The sensor BME280 was used as the temperature, pressure, and humidity sensor of the project. The BME280 was a sensor designed by Bosch Sensortec that measures temperature, humidity, and barometric pressure. It was intended for wearables and mobile applications where small size and low power consumption were crucial considerations. (Bosch Sensortec., n.d.) Figure 3.13 shows the BME280 used in the project.

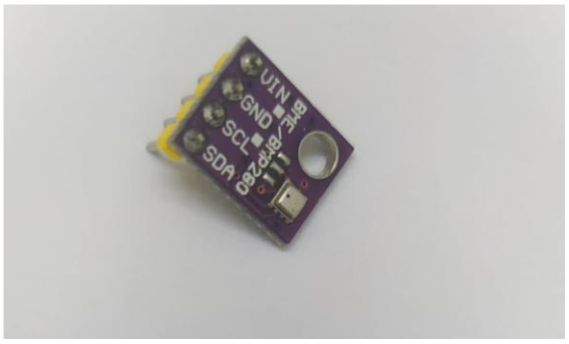


Figure 3.13: The BME280 used in the project

The BME280 module of this project is from Adafruit. This module can be used with either the I2C or SPI bus, using the same connections for both. The SCL (Clock) pin was labelled “SCK”, and the SDA (Data) pin was “SDI”. The SCL pin had connected with pin 29 (GPIO 22) of the ESP32, and the pin SDA had connected to pin 26 (GPIO 21) of ESP32. The library *Adafruit_BME280.h* had been included in the Arduino code for BME280 sensors. Figure 3.14 show the pinout diagram of BME280 and Table 3.7 show the Pin allocation of BME280.

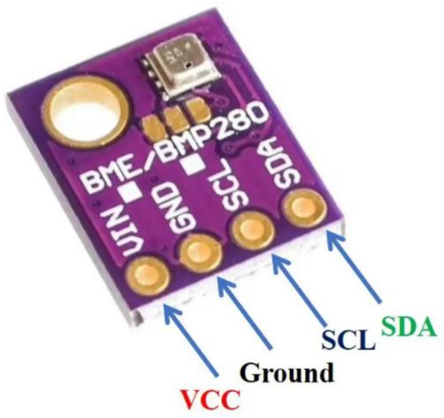


Figure 3.14: The pinout diagram of BME280 (Bilal, 2022)

Table 3.7: Pin allocation of BME280

Pin Name	Pin Connected
VIN	5V provided from the Arduino UNO
GND	Ground voltage of the system
SCL	Pin 29 (GPIO 22) of the ESP32
SDA	Pin 26 (GPIO21) of the ESP32

3.7.6 16 × 2 LCD connection

The purpose of the 16×2 LCD (Liquid Crystal Display) display in the project was to display the real-time AQI data to the user. A 16x2 LCD was a type of electronic display module that can be used to display characters. The name "16x2" refers to its 16 columns and 2 rows. Figure 3.15 show the pinout diagram of 16x2 LCD.

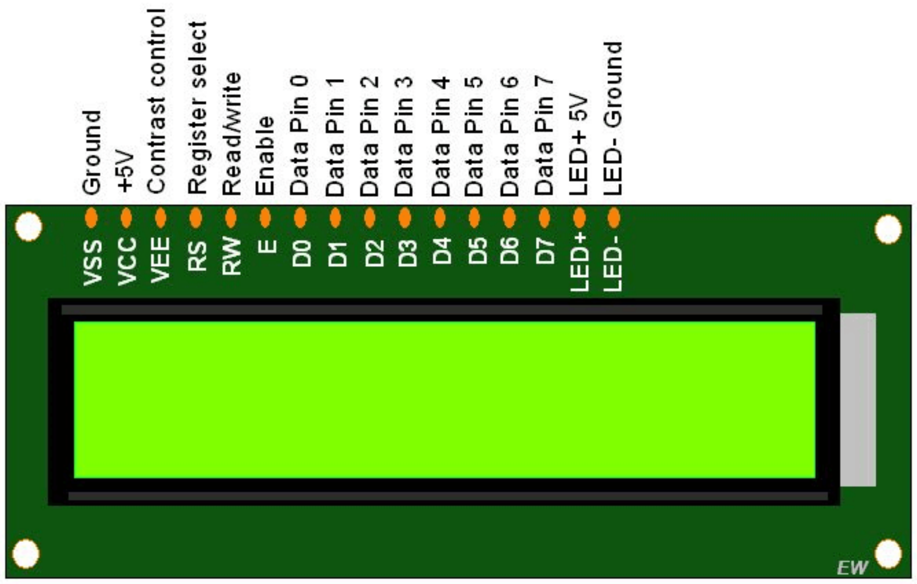


Figure 3.15: The pinout diagram of 16x2 LCD (Electronicwings, 2023)

The 10KΩ of variable potentiometer had connected to the V0 or VEE pin of the 16×2 LCD. The potentiometer was used to control the voltage at the V0 or VEE pin, which in turn controls the contrast of the display.

To adjust the display's contrast and visibility, a potential (voltage) must be connected to the V0 or VEE pin. In order to block or permit light to travel through various display segments, LCDs adjust the orientation of liquid crystal molecules within the display. The visibility of a segment was influenced by the amount of light passing through it, and this contrast was changed by supplying a particular voltage to the V0 or VEE pin.

The alignment of the liquid crystal molecules, which in turn influences the quantity of light that can pass through the LCD segments, can be changed by altering the voltage applied to the V0 or VEE pin. Low voltage can cause the liquid crystal molecules to misalign, which reduces contrast and visibility. Poor image quality and overdriving of the display were also possible effects of excessive voltage. Figure 3.16 shows the schematic diagram for Arduino UNO and 16x2 LCD. Table 3.8 and Table 3.9 show the pin allocation of LCD1 and LCD2.

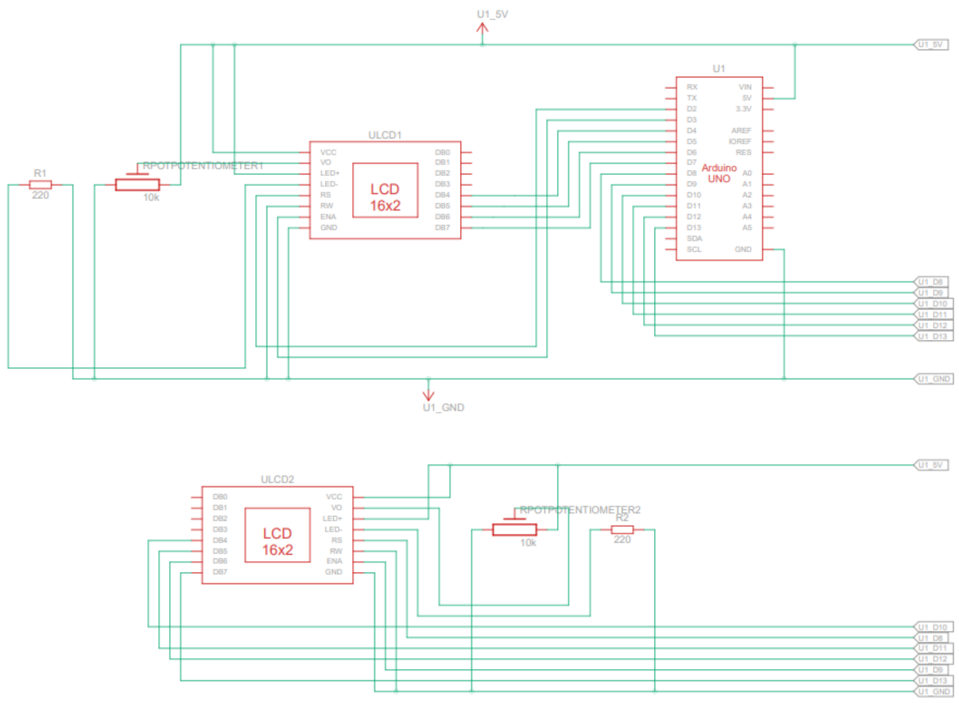


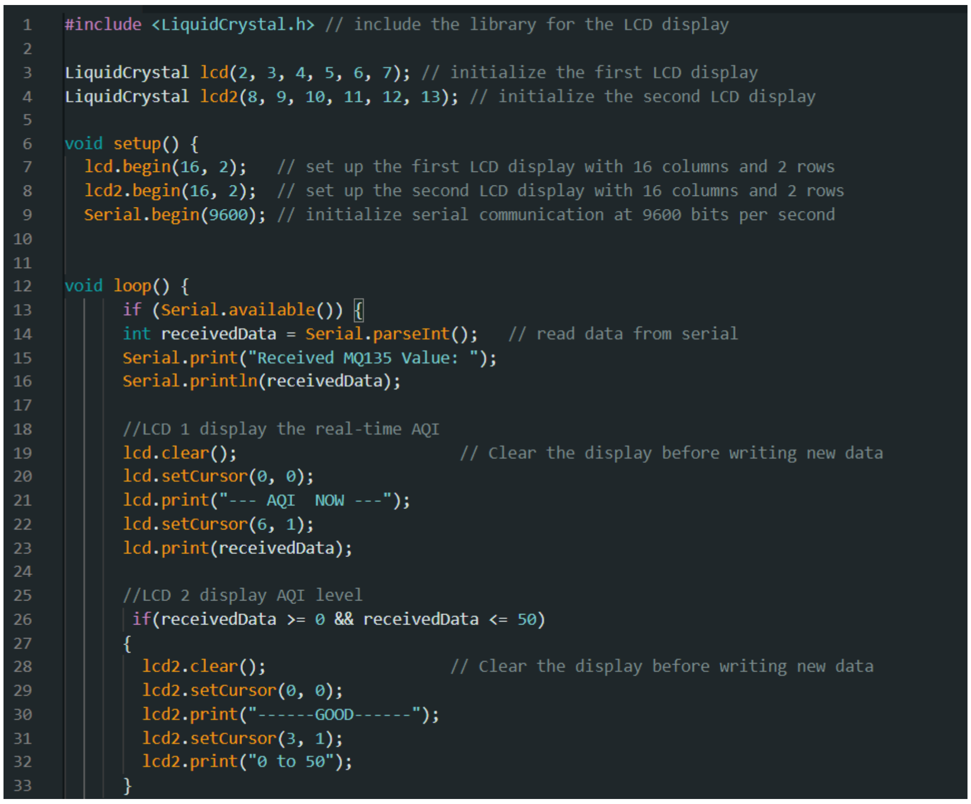
Figure 3.16: Schematic diagram for Arduino UNO and 16x2 LCD

Table 3.8: Pin allocation of LCD1

LCD1	
Pin Name	Pin Connected
VSS	Ground voltage of the system
VDD	5V from Arduino UNO
V0	10K of Potential
RS	Pin 2 of Arduino UNO
RW	Ground voltage of the system
E	Pin 3 of Arduino UNO
D0	-
D1	-
D2	-
D3	-
D4	Pin 4 of Arduino UNO
D5	Pin 5 of Arduino UNO
D6	Pin 6 of Arduino UNO
D7	Pin 7 of Arduino UNO
A	5V from Arduino UNO
K	120Ω of register

Table 3.9: Pin allocation of LCD2

LCD2	
Pin Name	Pin Connected
VSS	Ground voltage of the system
VDD	5V from Arduino UNO
V0	10K of Potential
RS	Pin 7 of Arduino UNO
RW	Ground voltage of the system
E	Pin 8 of Arduino UNO
D0	-
D1	-
D2	-
D3	-
D4	Pin 9 of Arduino UNO
D5	Pin 10 of Arduino UNO
D6	Pin 11 of Arduino UNO
D7	Pin 12 of Arduino UNO
A	5V from Arduino UNO
K	120Ω of register



```
#include <LiquidCrystal.h> // include the library for the LCD display

LiquidCrystal lcd(2, 3, 4, 5, 6, 7); // initialize the first LCD display
LiquidCrystal lcd2(8, 9, 10, 11, 12, 13); // initialize the second LCD display

void setup() {
  lcd.begin(16, 2);   // set up the first LCD display with 16 columns and 2 rows
  lcd2.begin(16, 2);  // set up the second LCD display with 16 columns and 2 rows
  Serial.begin(9600); // initialize serial communication at 9600 bits per second


void loop() {
      if (Serial.available()) {
      int receivedData = Serial.parseInt();   // read data from serial
      Serial.print("Received MQ135 Value: ");
      Serial.println(receivedData);

      //LCD 1 display the real-time AQI
      lcd.clear();                           // Clear the display before writing new data
      lcd.setCursor(0, 0);
      lcd.print("--- AQI  NOW ---");
      lcd.setCursor(6, 1);
      lcd.print(receivedData);

      //LCD 2 display AQI level
       if(receivedData >= 0 && receivedData <= 50)
      {
        lcd2.clear();                      // Clear the display before writing new data
        lcd2.setCursor(0, 0);
        lcd2.print("------GOOD------");
        lcd2.setCursor(3, 1);
        lcd2.print("0 to 50");
      }
```

Figure 3.17: The code for 16x2 LCD

For the code of 16x2 LCD shown in Figure 3.17, the library <LiquidCrystal.h> includes the code for the software. These code ‘LiquidCrystal lcd’ and ‘LiquidCrystal lcd2’ have been defined to interface with two separate LCDs. In the void setup() function, 115200 bits per second of baud rate was used as initialized serial communication. There code ‘lcd.begin’ and ‘lcd2.begin’ was to set up the two separate LCDs with 16 columns and two rows. In the void loop() function, LCD 1 will display real-time AQI data from the serial Communication, and LCD 2 will display the AQI level. Codes ‘lcd.clear()’ and ‘lcd2.clear()’ will use to clear the data every time before the LCD displays a new value or data.

3.7.7 logic level converter

The logic level converter TXS0108E used in the project. The TXS0108E was an open-drain and push-pull 8-bit bi-directional voltage-level shifter. It was a logic level converter that was useful for connecting devices that run at various voltage levels because it can translate signals from one voltage level to another. Bright pull-up resistors in the TXS0108E dynamically alter their values depending on whether a low or a high was supplied over the I/O line. (learnsparkfun, 2021) Figure 3.18 shows the block diagram of the logic level converter TXS0108E.

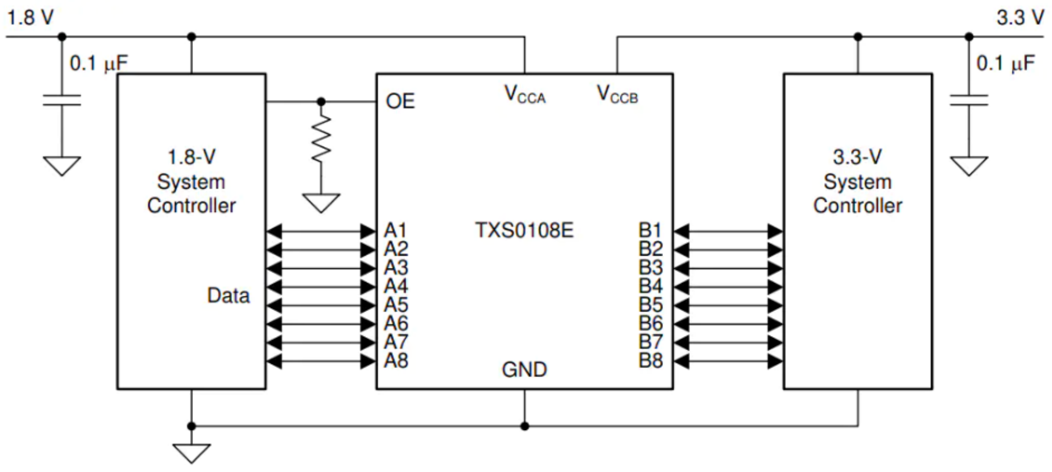


Figure 3.18: The application block diagram (Mouse, 2022)

The logic level converter's purpose was that the MQ135 and MQ7 need 5V of the voltage to operate their function and get the best measurements. The heaters of the MQ135 and MQ7 must be required to reach a specific temperature to get the best measurements from the sensors. When the sensor transfers data to ESP32, the high voltage will damage the ESP32. To solve the problem, the TXS0108E was essential in ensuring all devices' safety and performance while they transfer data.

According to the block diagram in Figure 3.18, The VA pin on port A accepts any supply voltage between 1.4 V and 3.6 V, while the VB pin on port B accepts any supply voltage between 1.65 V and 5.5. In this project, port A and OE pin were connected to the 3.3V provider by ESP32 and port B was connected to 5V, which

Arduino UNO provides. The GND pin will touch the ground of both Arduino UNO and ESP32 as a reference point. A1 to A8 and B1 to B8 were used as the input and output of the data transfer.

Figure 3.19 shows the TXS0108E used in the project, and table 3.10 shows the pin allocation of TXS0108E.

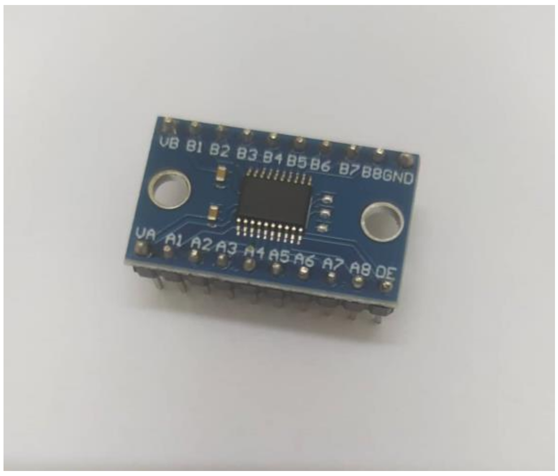


Figure 3.19: The TXS0108E used in the project

Table 3.10: Pin allocation of TXS0108E

TXS0108E	
Pin Name	Pin Connected
VA	3.3V from ESP32
A1	Pin 4 (GPIO 34) of ESP32
A2	Pin 5 (GPIO 35) of ESP32
A3	Pin 26 (GPIO 21) of ESP32
A4	Pin 29 (GPIO 22) of ESP32
A5	Pin 21 (GPIO 16) of ESP32
A6	Pin 22 (GPIO 17) of ESP32
A7	-
A8	-
OE	Ground voltage of the system
VB	5.5V from Arduino UNO
B1	Pin A0 of MQ135
B2	Pin A0 of MQ7
B3	SCL pin of BME280
B4	SDA pin of BME280
B5	RXD pin of PMS7003
B6	TXD pin of PMS7003
B7	-
B8	-
GND	Ground voltage of the system

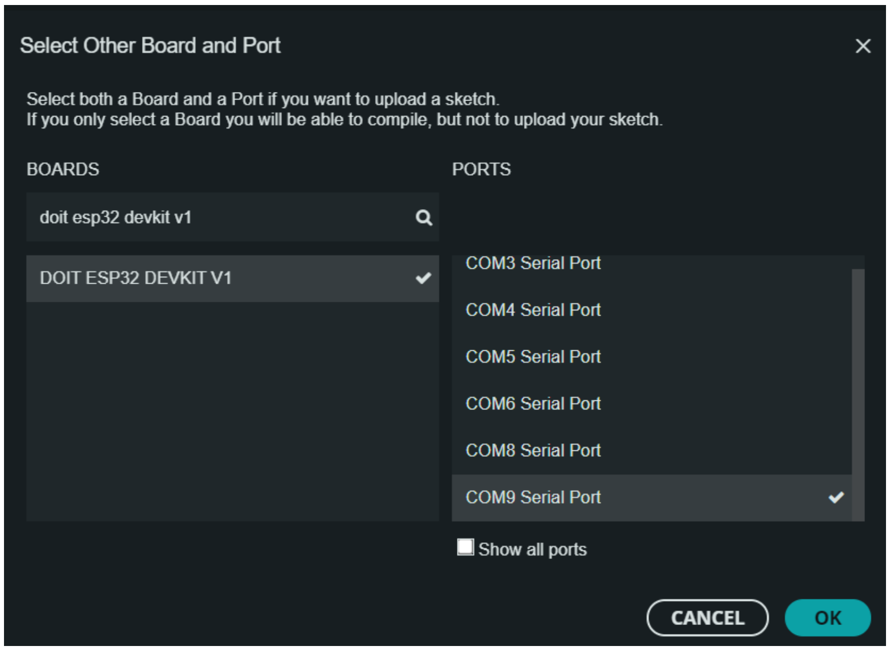
3.8 Software connectivity

3.8.1 IoT platform connectivity

The Blynk platform was used in this project. The reason to choose Blynk was that it offers a full suite of software, allowing one to prototype, deploy, and remotely manage connected electronic devices at any scale. The purpose of the Blynk in this project was to let the user view the real-time data and analyse the data. The user can view the real-time data on their mobile devices easily all the time and anywhere.

Users may create and manage IoT apps and linked electrical devices using the Blynk platform. It offers an intuitive user interface and a no-code app builder, making it simple for people and companies to develop unique apps for their IoT projects. Users may construct smartphone apps with visual widgets like buttons, sliders, and charts using the drag-and-drop app builder provided by Blynk. These applications enable remote device control and data visualization for sensors. Hardware platforms supported by Blynk include ESP32, Arduino, Raspberry Pi, Seeed, Particle, and more. Users may create a user interface for their IoT projects by connecting their devices to the cloud and using a variety of widgets. Both developers and end users should find using Blynk to be user-friendly. End users may effortlessly connect to and operate compatible devices from anywhere globally, while developers can swiftly prototype and launch their IoT solutions. Scalability: Blynk was made to scale from small-scale installations to extensive personal initiatives. It delivers a fleet of connected devices that was completely controlled and gives the required infrastructure to support IoT applications of any scale.

Figure 3.20 shows that the DOIT ESP32 DEVIKIT V1 board was selected for this project in Arduino IDE.



Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

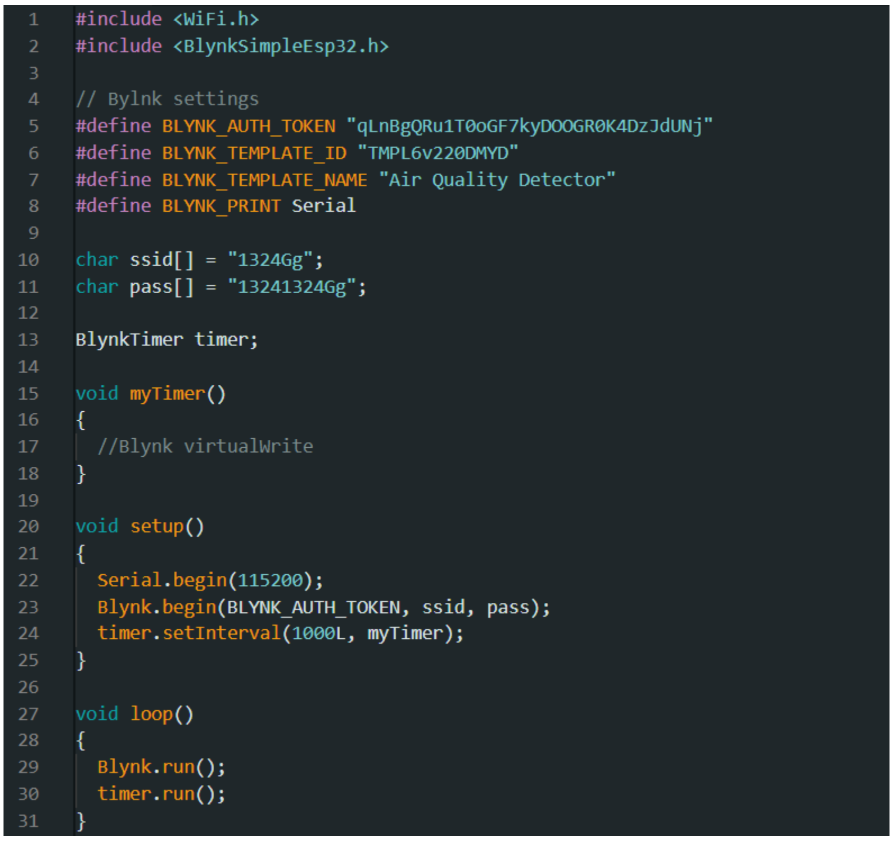
doit esp32 devkit v1

DOIT ESP32 DEVKIT V1 ✓

COM3 Serial Port
COM4 Serial Port
COM5 Serial Port
COM6 Serial Port
COM8 Serial Port
COM9 Serial Port ✓

☐ Show all ports

Figure 3.20: The window of board selection of Arduino IDE



```
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

// Bylnk settings
#define BLYNK_AUTH_TOKEN "qLnBgQRu1T0oGF7kyDOOGR0K4DzJdUNj"
#define BLYNK_TEMPLATE_ID "TMPL6v220DMYD"
#define BLYNK_TEMPLATE_NAME "Air Quality Detector"
#define BLYNK_PRINT Serial

char ssid[] = "1324Gg";
char pass[] = "13241324Gg";

BlynkTimer timer;

void myTimer()
{
  //Blynk virtualWrite
}

void setup()
{
  Serial.begin(115200);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  timer.setInterval(1000L, myTimer);
}

void loop()
{
  Blynk.run();
  timer.run();
}
```

Figure 3.21: The code of Blynk connectivity

For the software shown in Figure 3.21, The libraries <wifi.h> and <BlynksimpleEsp32.h> were included in the code. The library <wifi.h> used for connecting the wifi network and the library <BlynksimpleEsp32.h> used for connecting the Blynk platform with ESP32. For the defined Blynk settings, the BLYNK_AUTH_TOKEN unique identifier was generated by Blynk. Cloud for every device and was required for the device to connect to Blynk; it was the primary identifier of every device in the Blynk Cloud. BLYNK_TEMPLATE_ID and BLYNK_TEMPLATE_NAME identify the device template in Blynk and were optional parameters. The character arrays 'ssid' and 'pass' identify the wifi. Next, the myTime() function allows to send the data from the hardware to the Blynk app using virtual pins.

In the void setup() function, 15200 bits per second of baud rate was used as initialized serial communication. The function Blynk.begin() was used in Blynk to configure the device and connect it to WiFi. The code timer.setInterval(1000L, myTimer) was a function call in the Blynk library that sets a timer to call the function myTimer() every 1000 milliseconds or 1 second. In the void loop() function, the function 'Blynk.run()' was a main Blynk routine responsible for keeping the connection alive, sending and receiving data, and executing other Blynk library routines. The function 'timer.run()' executes the timers defined in the code.

3.8.2 IoT Webpage setup

After coding for the Blynk platform, the next step was to set up and design the Blynk website. Blynk offers several sample sketches to help people get started. Users can put these samples together as they choose by imagining them as LEGO bricks. On the Blynk Examples Sketch Builder, they may also locate an appropriate code sample for their hardware. Users may configure the design parameters and preview the design via the company's web dashboard. For creating and editing the Mobile Dashboard User interface (GUI) for the specified Device Template, Blynk provides a Developer Mode in the mobile app. Figure 3.22 shows the website page of the Blynk.

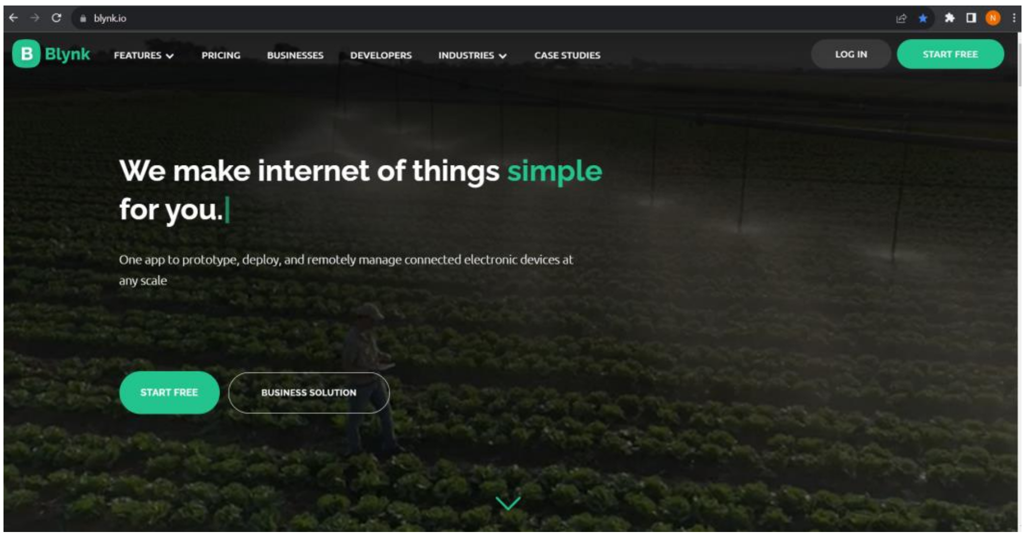

blynk.io
Blynk
FEATURES
PRICING
BUSINESSES
DEVELOPERS
INDUSTRIES
CASE STUDIES
LOG IN
START FREE
We make internet of things simple for you.
One app to prototype, deploy, and remotely manage connected electronic devices at any scale
START FREE
BUSINESS SOLUTION

Figure 3.22: The website page of the Blynk

Blynk provides a low-code IoT software platform. Their several plans, including a free one, were listed on a price website. The free plan offers two devices, five users, and three device templates. It also allows for studies. The free plan's usefulness has recently been restricted by adjustments made to it, nevertheless. Before with started with Blynk, user had to sign up for a new account. Figure 3.23 show the page after login of sign up.

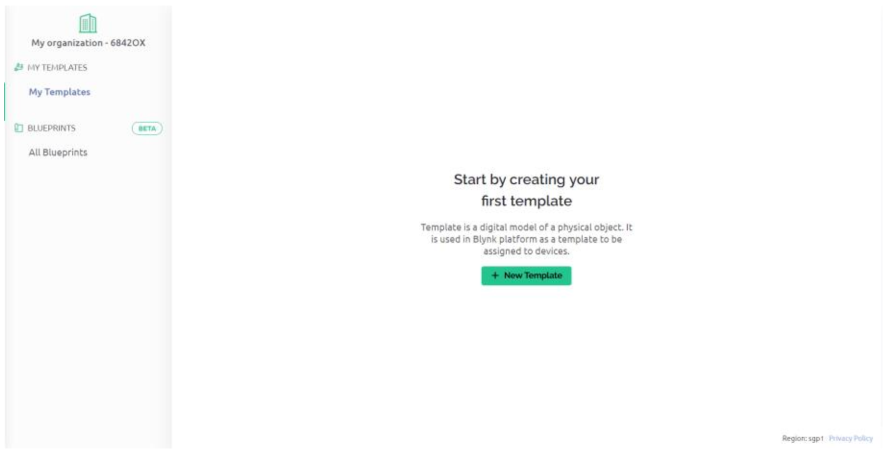
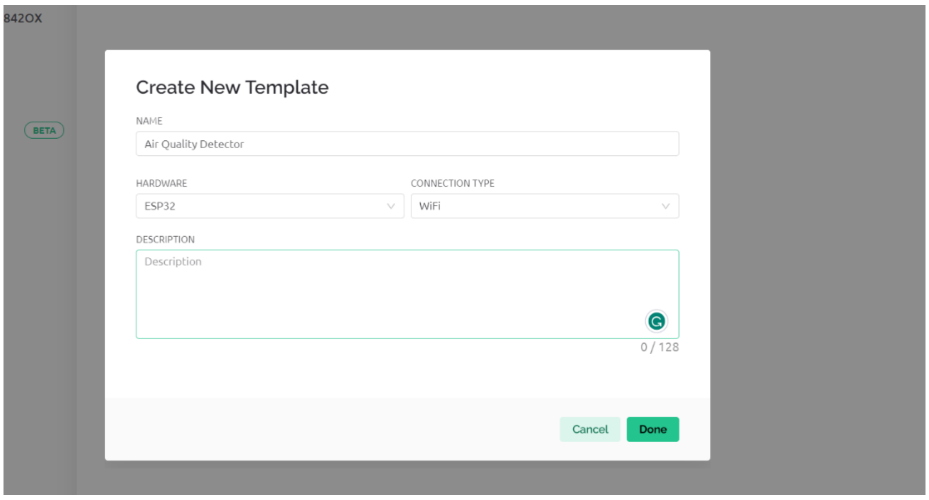

My organization - 6842OX
MY TEMPLATES
My Templates
BLUEPRINTS
BETA
All Blueprints
Start by creating your first template
Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.
+ New Template
Region: sgp1
Privacy Policy

Figure 3.23: The Blynk website after login

After creating an account, as seen in Figure 3.24, a new template needs to be created for the project. For the hardware selection, the ESP32 since the ESP32 had been chosen as the WiFi module for the project.



Create New Template

NAME: Air Quality Detector

HARDWARE: ESP32

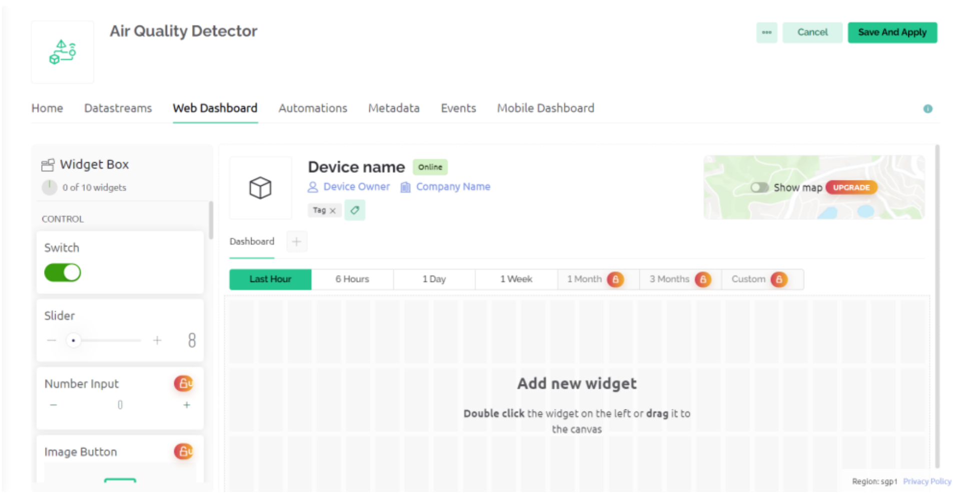
CONNECTION TYPE: WiFi

DESCRIPTION: Description

0 / 128

Figure 3.24: The window of create new template for the project

A dashboard template needs to be created after creating a new template for the project. The web dashboard allows to create a dashboard template to interact with these devices and see the information they collect. Figure 3.25 shows the web dashboard of the template. In the right corner, many widgets were allowed to choose from.



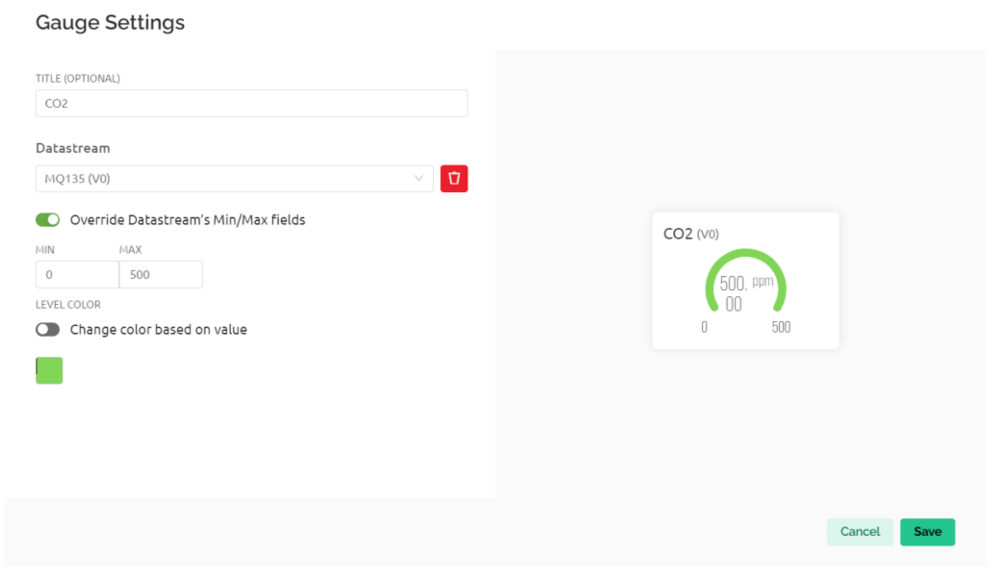
Air Quality Detector

Add new widget

Double click the widget on the left or drag it to the canvas

Figure 3.25: The web dashboard

Figure 3.26 shows the setting of one example for the gauge. In the title section, name the specific gas that wants to know the data from the detector. In the data stream section, the Virtual Pin has been selected for the sensor. Lastly, set the minimum and maximum value for the gauge.


Gauge Settings

TITLE (OPTIONAL): CO2

Datastream: MQ135 (V0)

Override Datastream's Min/Max fields

MIN: 0 MAX: 500

LEVEL COLOR

Change color based on value

Figure 3.26: The gauge settings

3.9 Arduino and ESP32 Serial Communications

The AQI data was transferred from ESP32 to Arduino UNO via serial communication for this project. Figure 3.27 shows the simple diagram of Arduino UNO and ESP32 serial communication for this project.

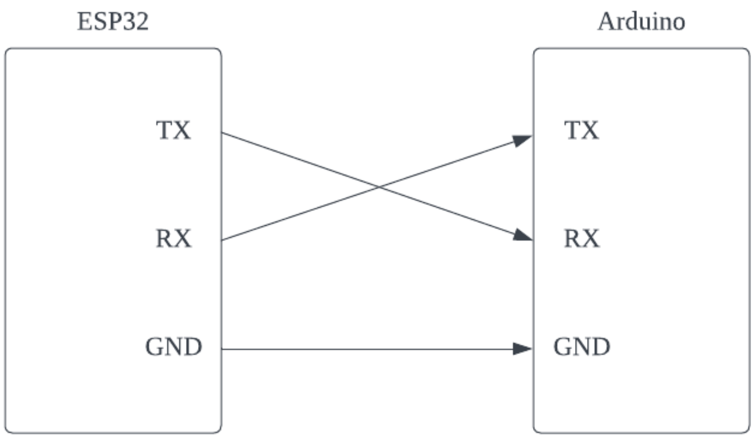
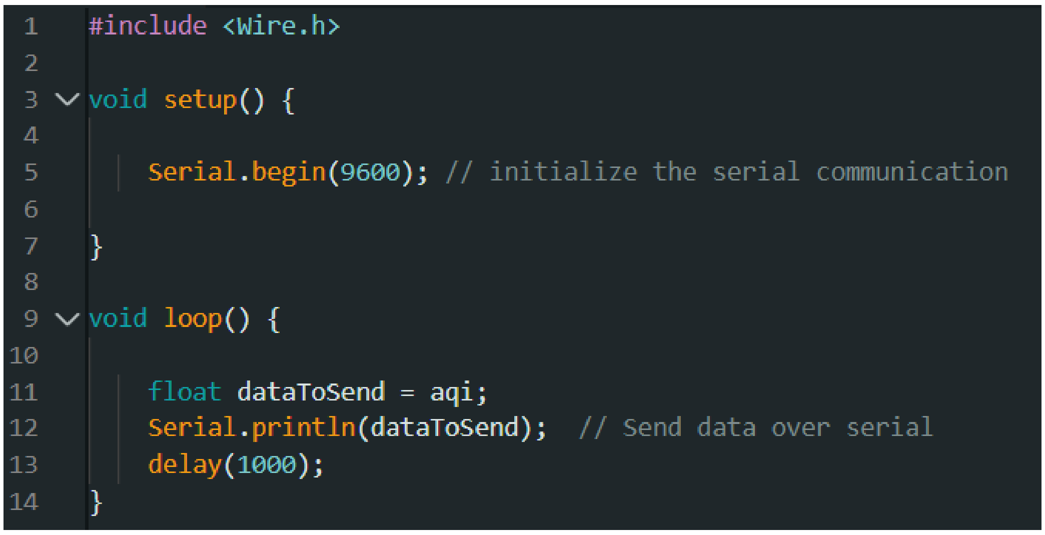


Figure 3.27: The simple diagram of Arduino UNO and ESP32 serial communication

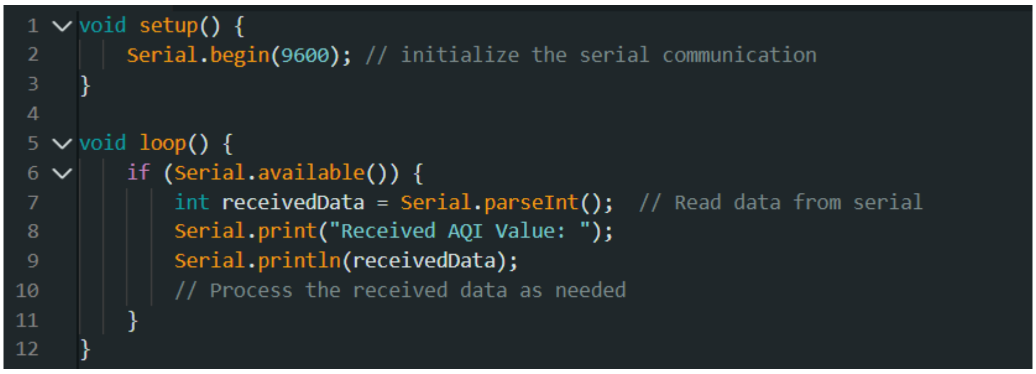
To set up serial communication between an Arduino UNO and an ESP32, pin TXD (GPIO 1) and RXD (GPIO 3) of ESP32 were connected to the serial pins of the Arduino UNO, which were pin 0 (RX) and pin 1 (TX). To ensure that the baud rate was the same in both programs to exchange data between the Arduino UNO and ESP32, connected the ground pins of the Arduino UNO and the ESP32.

Figure 3.28 and Figure 3.29 show the code for communication between ESP32 and Arduino UNO.



```
#include <Wire.h>

void setup() {

  Serial.begin(9600); // initialize the serial communication

}

void loop() {

  float dataToSend = aqi;
  Serial.println(dataToSend);  // Send data over serial
  delay(1000);
}
```

Figure 3.28: The code for the sender



```
void setup() {
  Serial.begin(9600); // initialize the serial communication
}

void loop() {
  if (Serial.available()) {
    int receivedData = Serial.parseInt();  // Read data from serial
    Serial.print("Received AQI Value: ");
    Serial.println(receivedData);
    // Process the received data as needed
  }
}
```

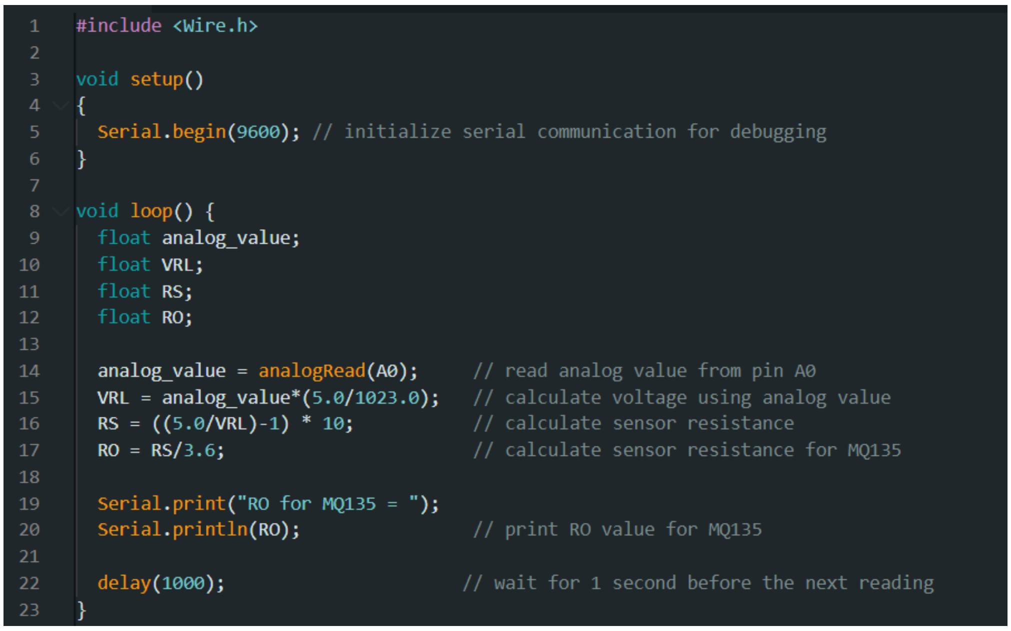
Figure 3.29: The code for the receiver

Figure 3.28 shows the code for ESP32, which was the sender. The data from the ESP32 will be sent to the Arduino UNO to analyze the data and display the real-time data to the user. In the void setup() function, 9600 bits per second of baud rate was used as initialized serial communication. In the void loop() function, the code ' serial,println()' sends data to the serial port as human-readable ASCII text.

Figure 3.29 shows the code for Arduino UNO, which was the receiver. The Arduino UNO will receive the data from the ESP32, and the Arduino UNO will analyse and display the data on the two separate LCDs. In the void setup() function, the baud rate was set to 9600 bits per second to determine the speed at which data was transmitted between the ESP32 and Arduino Uno. Setting the same baud rate in both programs was crucial for successful communication. The void loop() focuses on analysing and displaying the data received from the ESP32. The function will ensure the data has been sent from the ESP32 or received by the Arduino Uno. If the data was received, the Arduino UNO starts to analyse the data and display the data to the two separate LCDs.

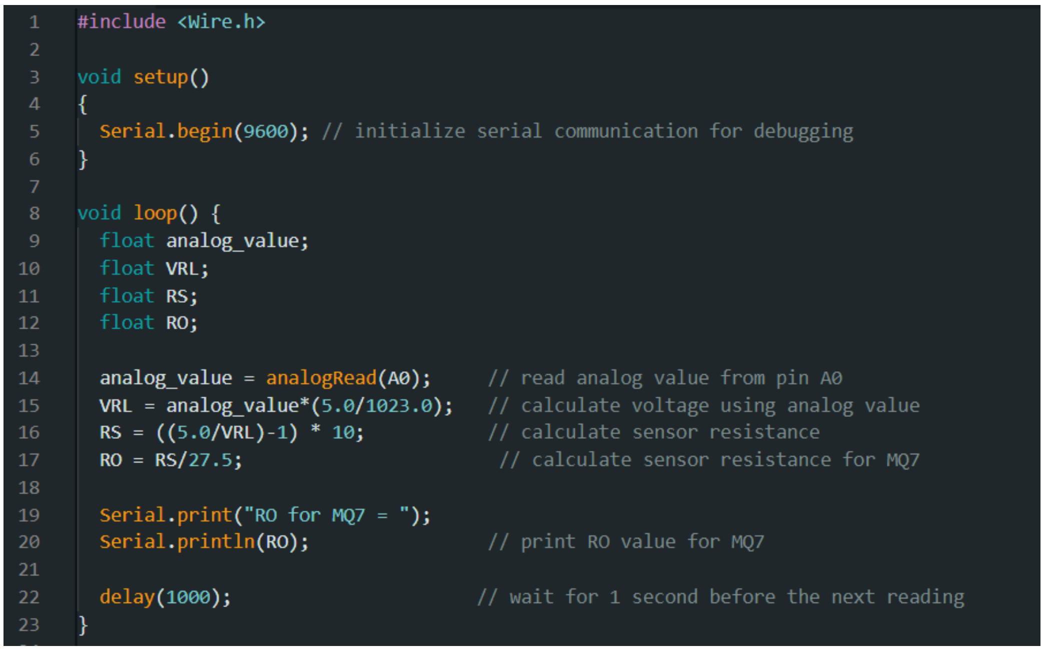
3.10 Sensor calibration

The MQ135 and MQ7 gas sensors can identify several gases in the air. For accurate measurements of the gas concentration in the air, calibration of the MQ135 and MQ7 sensors was required. Before calibrating the MQ135 and MQ7 sensors, these sensors must be preheated over 48 hours. And the heater of the MQ135 and MQ7 sensors requires 5V of voltage. Figure 3.30 and figure 3.31 show the code for MQ135 and MQ7 calibration.



```
#include <Wire.h>

void setup()
{
  Serial.begin(9600); // initialize serial communication for debugging
}

void loop() {
  float analog_value;
  float VRL;
  float RS;
  float RO;

  analog_value = analogRead(A0);      // read analog value from pin A0
  VRL = analog_value*(5.0/1023.0);    // calculate voltage using analog value
  RS = ((5.0/VRL)-1) * 10;            // calculate sensor resistance
  RO = RS/3.6;                        // calculate sensor resistance for MQ135

  Serial.print("RO for MQ135 = ");
  Serial.println(RO);                 // print RO value for MQ135

  delay(1000);                       // wait for 1 second before the next reading
}
```

Figure 3.30: The code for MQ135 calibration



```
#include <Wire.h>

void setup()
{
  Serial.begin(9600); // initialize serial communication for debugging
}

void loop() {
  float analog_value;
  float VRL;
  float RS;
  float RO;

  analog_value = analogRead(A0);     // read analog value from pin A0
  VRL = analog_value*(5.0/1023.0);   // calculate voltage using analog value
  RS = ((5.0/VRL)-1) * 10;           // calculate sensor resistance
  RO = RS/27.5;                       // calculate sensor resistance for MQ7

  Serial.print("RO for MQ7 = ");
  Serial.println(RO);                // print RO value for MQ7

  delay(1000);                       // wait for 1 second before the next reading
}
```

Figure 3.31: The code for MQ7 calibration

The first step as shown in the code, the system will read the analog value from the sensor and convert the analog reading from the sensor into a corresponding voltage value in volts, V_{RL}. After obtaining the V_{RL}, the next step was calculating the sensor resistor R_S. Equation 3.1 shows the formula to find the R_S in the sensor.

$$R_s = \left(\frac{V_c}{V_{R_L}} - 1\right) \times R_L \qquad (3.1)$$

where

R_s = the sensor resistance, Ω

V_c = voltage across a load resistor, V

V_{R_L} = voltage across the sensor, V

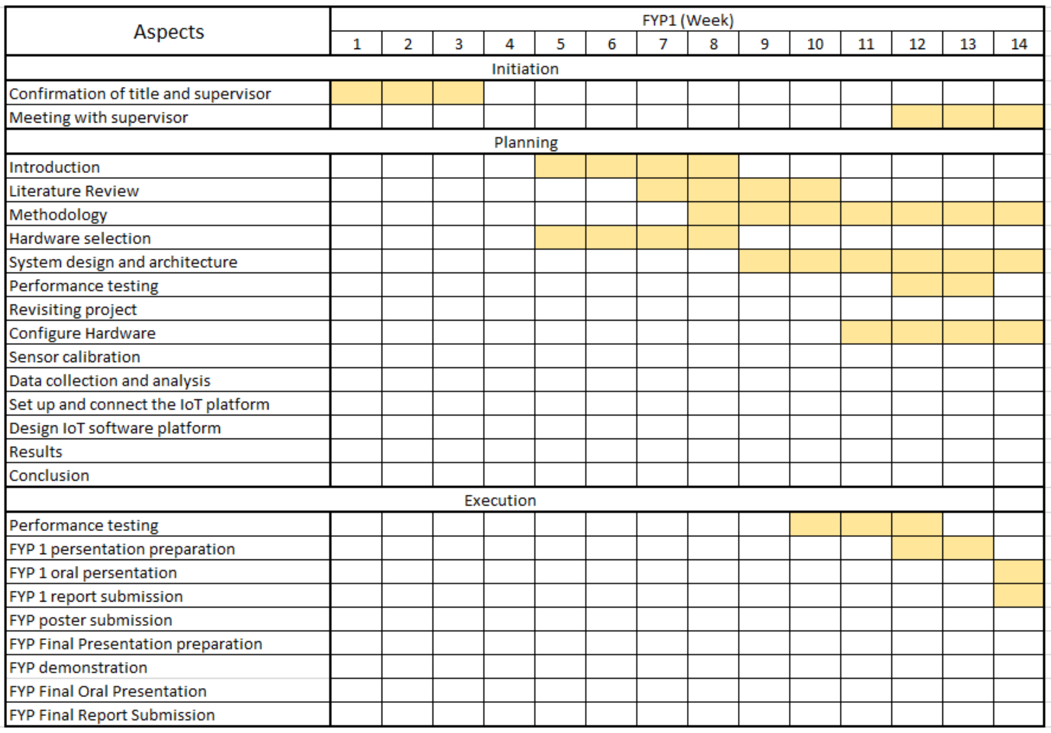
R_L = the load resistance, Ω

After finding the R_S, the next in need to calculate the the R_0 value in the clean air. According to the datasheet of the MQ135 and MQ7. The resistance ratio R_S/ R_0 in clean for MQ135 was 3.6 and MQ7 was 27.3. The ratio value will divide to the Rs to find the R_0. And the value of R_0 will prints to the serial monitor on Arduino IDE after calculating.

3.11 Project Management

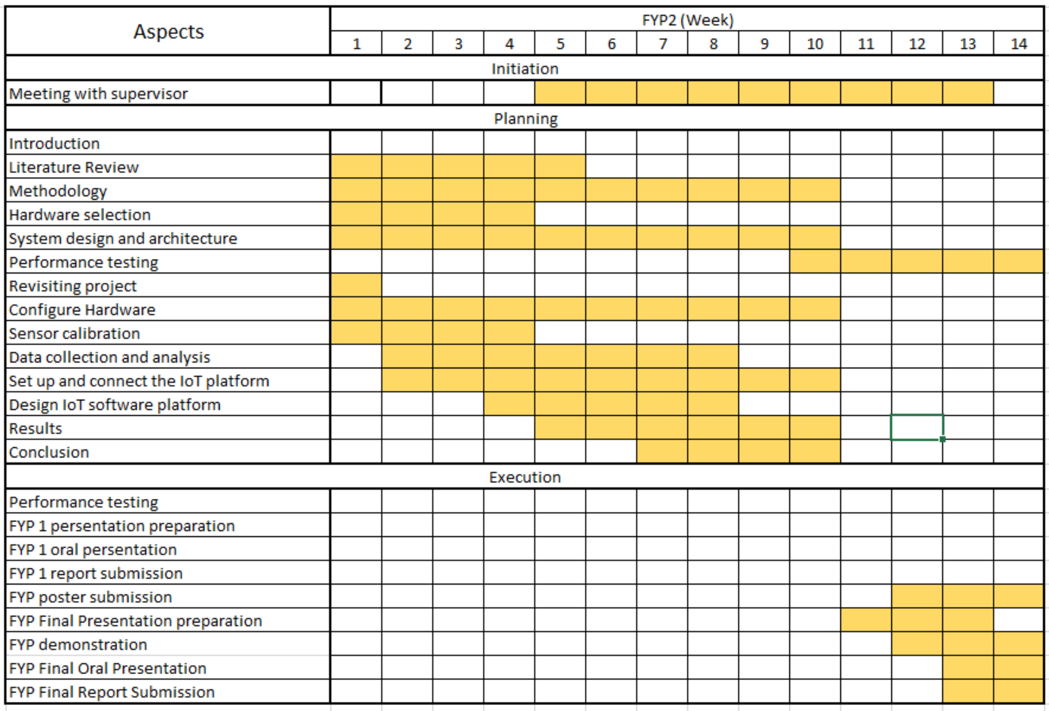
The Gantt chart for FYP 1 was shown in Table 3.11, and the Gantt chart for FYP 2 was shown in Table 3.12.

Table 3.11: The Gantt chart for FYP 1



Aspects	FYP1 (Week)													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Initiation														
Confirmation of title and supervisor	🟨	🟨	🟨											
Meeting with supervisor												🟨	🟨	🟨
Planning														
Introduction					🟨	🟨	🟨	🟨						
Literature Review							🟨	🟨	🟨	🟨				
Methodology								🟨	🟨	🟨	🟨	🟨	🟨	🟨
Hardware selection					🟨	🟨	🟨	🟨						
System design and architecture									🟨	🟨	🟨	🟨	🟨	🟨
Performance testing												🟨	🟨	
Revisiting project														
Configure Hardware											🟨	🟨	🟨	🟨
Sensor calibration														
Data collection and analysis														
Set up and connect the IoT platform														
Design IoT software platform														
Results														
Conclusion														
Execution														
Performance testing										🟨	🟨	🟨		
FYP 1 persentation preparation												🟨	🟨	
FYP 1 oral persentation														🟨
FYP 1 report submission														🟨
FYP poster submission														
FYP Final Presentation preparation														
FYP demonstration														
FYP Final Oral Presentation														
FYP Final Report Submission														

Table 3.12: The Gantt chart for FYP 2



Aspects	FYP2 (Week)													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Initiation														
Meeting with supervisor					🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	
Planning														
Introduction														
Literature Review	🟨	🟨	🟨	🟨	🟨									
Methodology	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨				
Hardware selection	🟨	🟨	🟨	🟨										
System design and architecture	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨				
Performance testing										🟨	🟨	🟨	🟨	🟨
Revisiting project	🟨													
Configure Hardware	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨				
Sensor calibration	🟨	🟨	🟨	🟨										
Data collection and analysis		🟨	🟨	🟨	🟨	🟨	🟨	🟨						
Set up and connect the IoT platform		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨				
Design IoT software platform				🟨	🟨	🟨	🟨	🟨						
Results					🟨	🟨	🟨	🟨	🟨	🟨				
Conclusion							🟨	🟨	🟨	🟨				
Execution														
Performance testing														
FYP 1 persentation preparation														
FYP 1 oral persentation														
FYP 1 report submission														
FYP poster submission												🟨	🟨	🟨
FYP Final Presentation preparation											🟨	🟨	🟨	
FYP demonstration												🟨	🟨	🟨
FYP Final Oral Presentation													🟨	🟨
FYP Final Report Submission													🟨	🟨

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Hardware overview

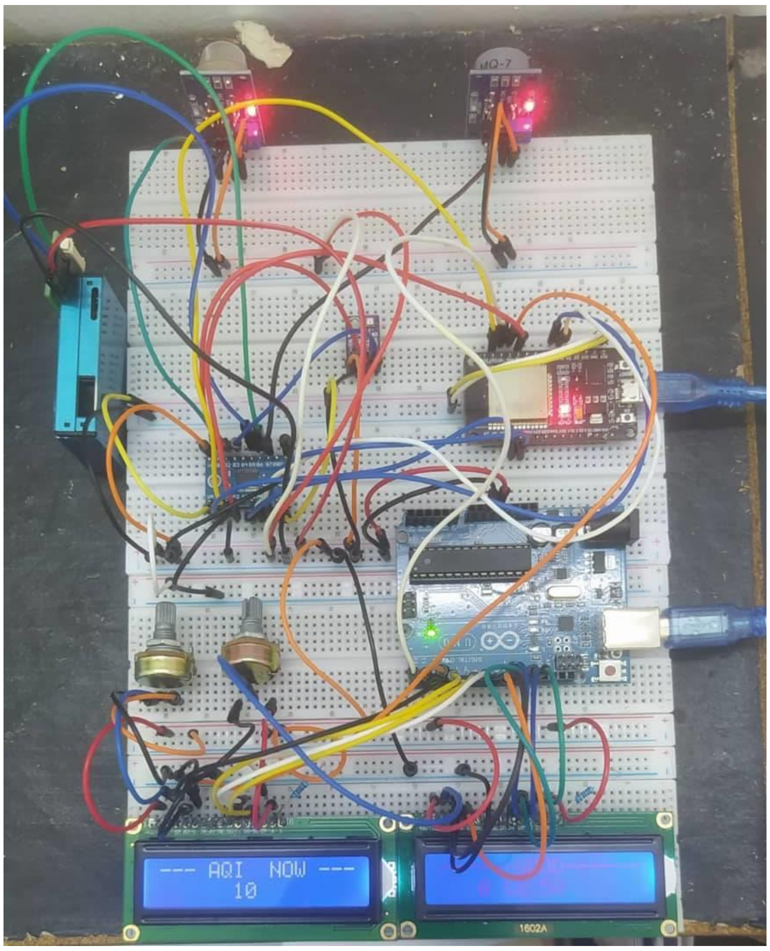


Figure 4.1: The entire hardware for the project

Figure 4.1 shows the entire hardware for this project. After successful construction, the hardware was used to measure the air quality. The software of the project analyses the data detector from the sensors.

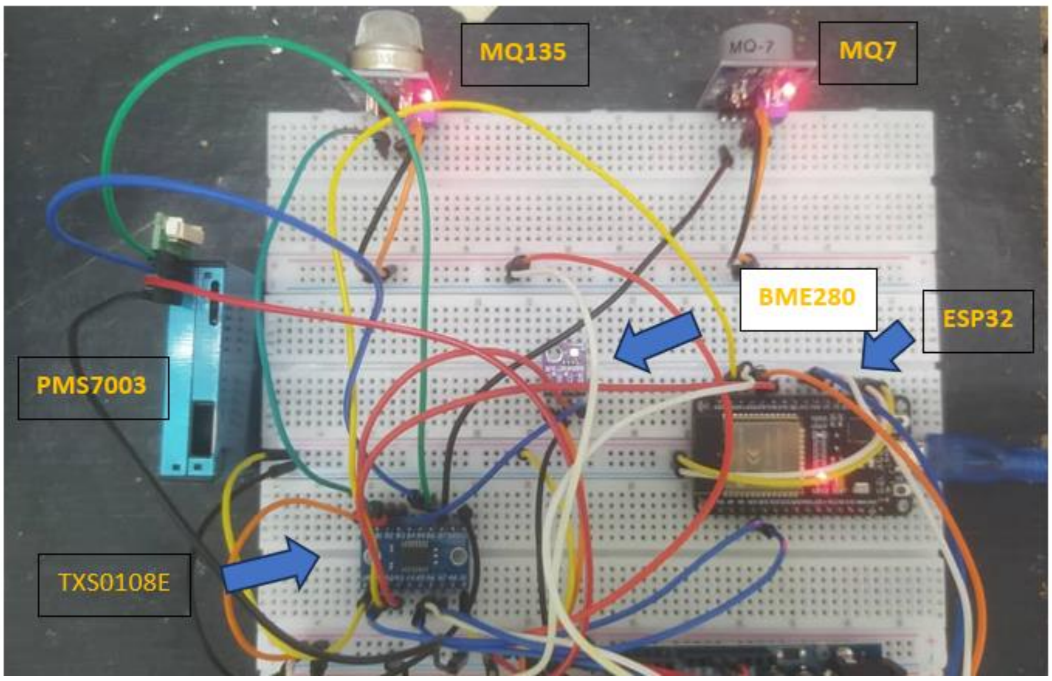


Figure 4.2: Sensor part, ESP32 and converter

Figure 4.2 shows the top part of the project hardware. On the top, were four sensors, MQ135, MQ7, BME280, and PMS7003, to detect and measure air quality, temperature, humidity, and others. At the bottom left of Figure 4.2, the logic level converter TXS0108E was used as the level-shifting voltage translator. The TXS0108E ensures all devices' safety and performance while they transfer data. All sensors were connected with ports B1 to B6 of the TXS0108E, and ports A1 to A6 were used as the data transfer output to ESP32. At the bottom right of Figure 4.2, ESP32 Devkit V1 was used as the IoT section's Wi-Fi module. ESP32 will receive the data from their sensors and transfer it to the Arduino UNO to display the real-time data on the LCD.

Figure 4.3 shows the bottom part of the project hardware which were Arduino UNO and LCD.

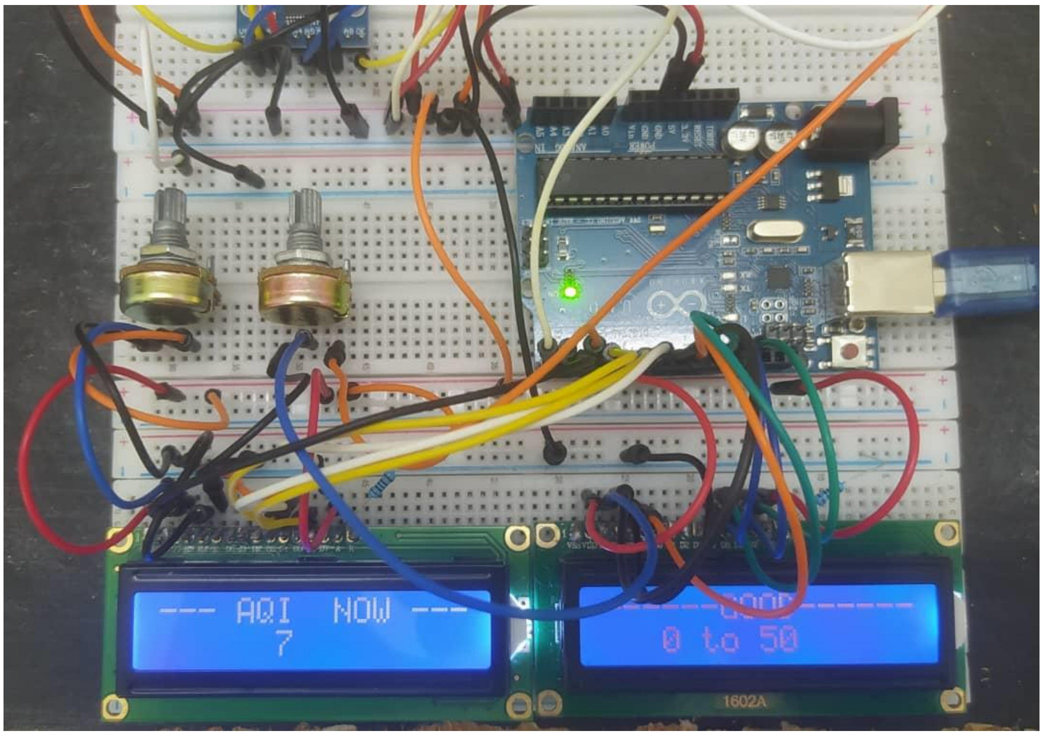


Figure 4.3: The Arduino Uno and LCD.

On the right side of Figure 4.3, Arduino UNO will receive real-time data from the ESP32 and display the real-time data on two separate LCDs. On the left side of the Figure 4.3, two separate LCDs connect to the Arduino UNO. The LCD in the project was to display the real-time AQI data from the Arduino UNO to the user. The 10KΩ of variable potentiometer had connected to the V0 or VEE pin of the 16×2 LCD. The potentiometer controls the voltage at the V0 or VEE pin, which controls the display's contrast.

4.2 Software overview

The software of the project was to analyze the data detector from the sensors. The Blynk platform was used as the IoT platform of the project. The purpose of the Blynk in this project was to let the user view the real-time data and analyze the data.

The user can track the real-time data on their mobile devices easily at all the time and anywhere. Figure 4.4 shows the Blynk website for the project.

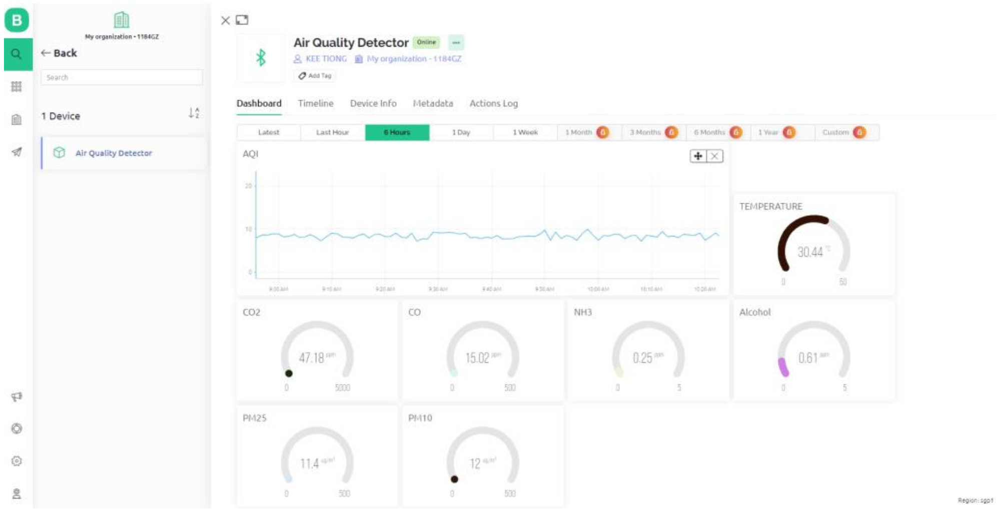


Figure 4.4: The air quality detector data using Blynk

Figure 4.4 shows the air quality detector data using Blynk website interface for the project. On the top, the AQI can be displayed as a line chart for the interface design. On the right-hand side of the AQI, there was a gauge of the real-time temperature. On the below interface, six gauges represent six types of gases, which are CO_2, ammonia, CO, alcohol, PM2.5 and PM10. Users can track the real-time AQI and other gases with the Blynk website.

Figure 4.5 shows the Blynk interface of the mobile phone version, different from the website of Blynk on PC. The top of the interface shows the real-time humidity, pressure, and temperature. Below the interface was the real-time data from the six gas types: CO_2, ammonia, CO, alcohol, and PM2.5 and PM10. User can easily track the real-time data with their mobile phone anytime and anywhere.

Figure 4.6 shows the user can track the air quality data by using a mobile phone. Blynk offers mobile apps for iOS and Android, allowing users to easily download the mobile app and track the air quality anytime and anywhere.

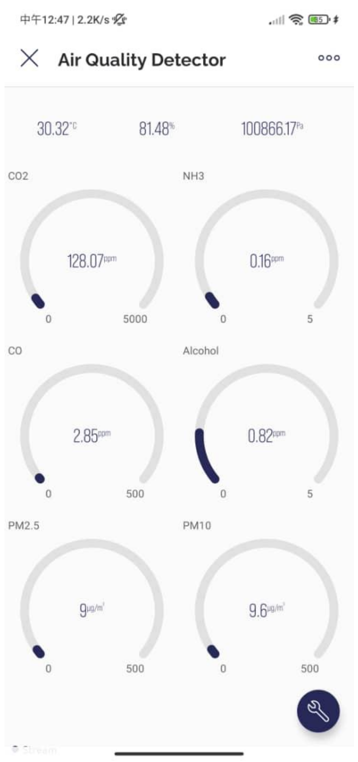


Figure 4.5: The mobile version of the website

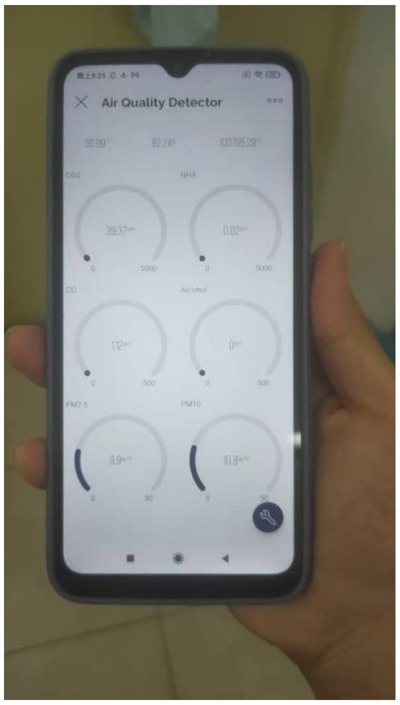


Figure 4.6: Track data by using mobile phone

4.3 Sensor calibration Result

4.3.1 R0 value finding

Figure 4.7 shows the process of the R_0 register values of MQ135 and MQ7. The value of the R_0 register will send data to the serial port on Arduino IDE. The value of the R_0 register will record down to find the average of the R0 value for the MQ135 and MQ7.

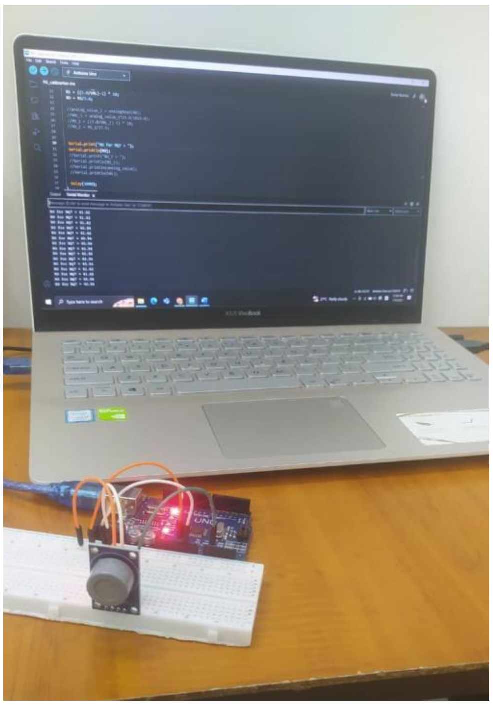


Figure 4.7: The sensor calibration process

Table 4.1 and Table 4.2 show the result of the average value of the R_0 register of MQ135 and MQ7.

Table 4.1: The R_0 average value of MQ135

Reading No.	MQ 135 sensor resistance R_0 in the clean air		
	Trial 1	Trail 2	Trail 3
1	16.58Ω	15.2Ω	14.8Ω
2	16.58Ω	15.2Ω	14.5Ω
3	15.2Ω	14.5Ω	14.5Ω
4	15.2Ω	14.5Ω	14.6Ω
5	15.2Ω	13.6Ω	14.7Ω
6	14.9Ω	13.6Ω	14.7Ω
7	15.1Ω	13.6Ω	14.6Ω
8	14.8Ω	14.2Ω	14.5Ω
9	14.8Ω	14.2Ω	14.3Ω
10	14.5Ω	14.5Ω	14.3Ω
11	14.3Ω	14.5Ω	14.4Ω
12	14.3Ω	13.9Ω	14.3Ω
13	13.5Ω	13.9Ω	14.3Ω
14	13.5Ω	13.9Ω	13.9Ω
15	13.5Ω	14.2Ω	13.9Ω
16	14.2Ω	14.2Ω	13.9Ω
17	14.2Ω	14.3Ω	14.3Ω
18	14.6Ω	14.4Ω	14.3Ω
19	14.2Ω	14.4Ω	14.2Ω
20	14.5Ω	14.4Ω	14.2Ω
R_0 average value	14.6Ω	14.2Ω	14.36Ω

Table 4.2: The R_0 average value of MQ7

Reading No.	MQ 7 sensor resistance R_0 in the clean air		
	Trial 1	Trail 2	Trail 3
1	1.56Ω	1.49Ω	1.62Ω
2	1.55Ω	1.58Ω	1.62Ω
3	1.56Ω	1.54Ω	1.54Ω
4	1.57Ω	1.59Ω	1.54Ω
5	1.58Ω	1.57Ω	1.54Ω
6	1.59Ω	1.57Ω	1.52Ω
7	1.58Ω	1.54Ω	1.52Ω
8	1.59Ω	1.54Ω	1.54Ω
9	1.59Ω	1.53Ω	1.54Ω
10	1.58Ω	1.54Ω	1.55Ω
11	1.51Ω	1.52Ω	1.55Ω
12	1.51Ω	1.51Ω	1.55Ω
13	1.52Ω	1.55Ω	1.53Ω
14	1.52Ω	1.5Ω	1.56Ω
15	1.52Ω	1.52Ω	1.56Ω
16	1.52Ω	1.68Ω	1.55Ω
17	1.52Ω	1.69Ω	1.54Ω
18	1.53Ω	1.7Ω	1.53Ω
19	1.52Ω	1.7Ω	1.53Ω
20	1.52Ω	1.7Ω	1.53Ω
R_0 average value	1.54Ω	1.57Ω	1.54Ω

As the results show, the average value of R0 for the MQ135 was 14 **Ω**, and the average value for the MQ7 sensor is1.5 **Ω**. The code will include the average values to ensure the sensor's accurate and reliable air quality measurements of specific gases.

4.3.2 Exponential regression

Exponential regression was a method used to fit a curve to a set of data points. It was used to calibrate MQ135 and MQ7 gas sensors to detect the specific gas. Using the power regression formula, the MQ135 and MQ7 datasheets used to find the exponential regression. Figure 4.8 and Figure 4.9 show the sensitivity characteristics of the MQ135 and the sensitivity characteristics of the MQ7.

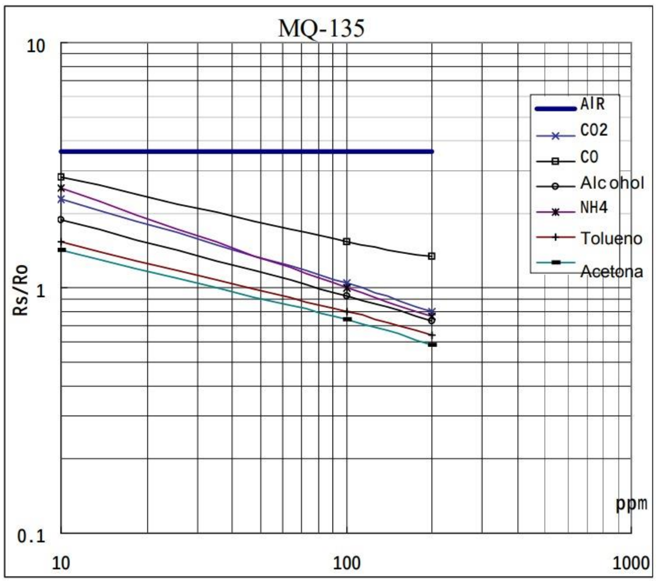


Figure 4.8: Sensitivity characteristics of the MQ135

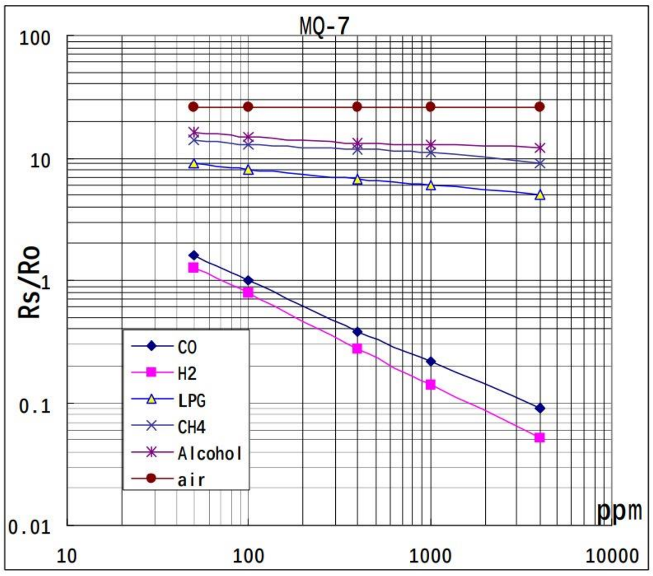
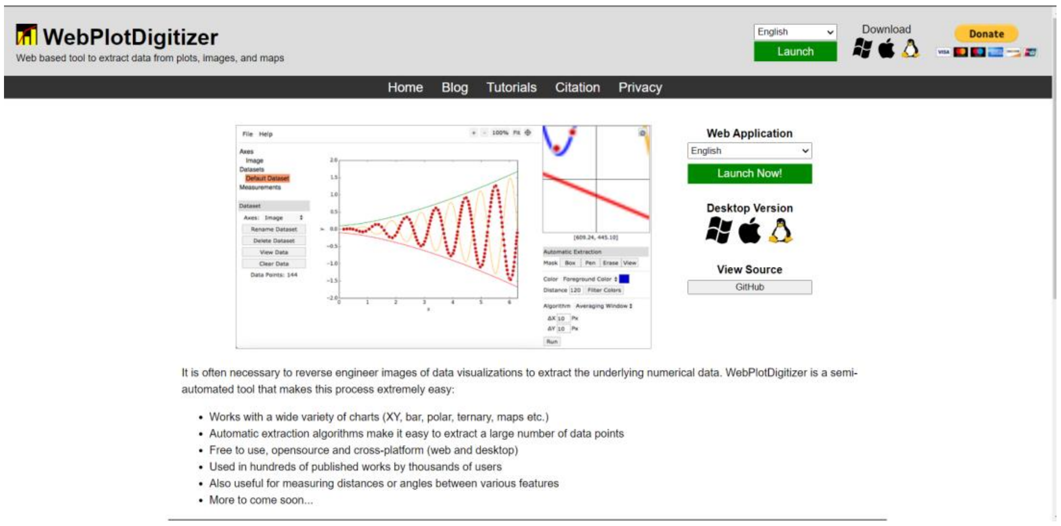


Figure 4.9: Sensitivity characteristics of the MQ7

The A and B values for the exponential equation vary for various gases and sensors. WebPlotDigitizer and Keisan Online Calculator were websites that must be utilized to find the exponential regression for a specific gas. Users were able to extract numerical data from plots, images, and maps using the WebPlotDigitizer website. It was a semi-automated tool that may be used with a variety of charts, including XY, bar, ternary, polar, and map diagrams. A comprehensive calculator website with user-friendliness, accuracy, and enjoyment was the Keisan Online Calculator. It covers various topics, such as mathematics, plane and space geometry, trigonometric functions, hyperbolic functions, and more. In this project, the WebPlotDigitizer was used to plot the point on the MQ135 and MQ7 datasheet, and the Keisan Online Calculator helped to calculate the power regression to find the values A and B.

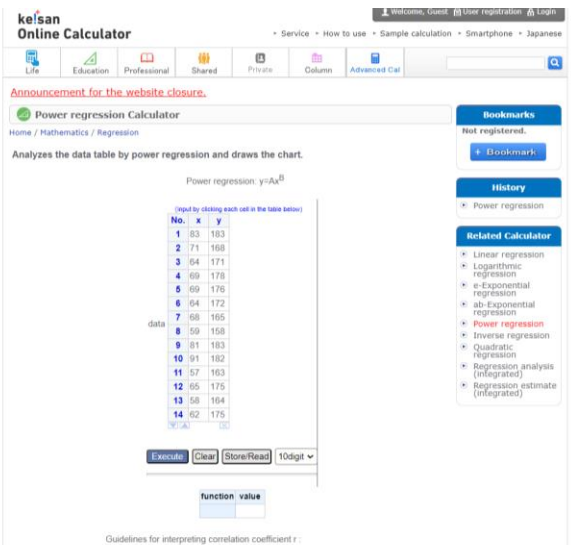
Figure 4.10 and Figure 4.11 shows the website WebPlotDigitizer and Keisan Online Calculator.



It is often necessary to reverse engineer images of data visualizations to extract the underlying numerical data. WebPlotDigitizer is a semi-automated tool that makes this process extremely easy:

- Works with a wide variety of charts (XY, bar, polar, ternary, maps etc.)
- Automatic extraction algorithms make it easy to extract a large number of data points
- Free to use, opensource and cross-platform (web and desktop)
- Used in hundreds of published works by thousands of users
- Also useful for measuring distances or angles between various features
- More to come soon...

Figure 4.10: The WebPlotDigitizer website



ke!san
Online Calculator

Welcome, Guest · User registration · Login

Service · How to use · Sample calculation · Smartphone · Japanese

Life · Education · Professional · Shared · Private · Column · Advanced Cal

Announcement for the website closure.

Power regression Calculator

Home / Mathematics / Regression

Analyzes the data table by power regression and draws the chart.

Power regression: $y=Ax^B$

data (input by clicking each cell in the table below)

No.	x	y
1	83	183
2	71	168
3	64	171
4	69	178
5	69	176
6	64	172
7	68	165
8	59	158
9	81	183
10	91	182
11	57	163
12	65	175
13	58	164
14	62	175

Execute · Clear · Store/Read · 10digit

function	value

Guidelines for interpreting correlation coefficient r :

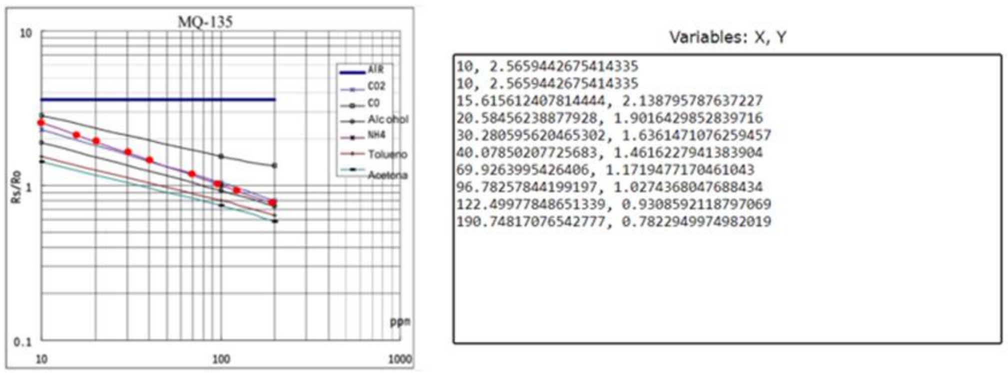
Bookmarks
Not registered.
+ Bookmark

History
- Power regression

Related Calculator
- Linear regression
- Logarithmic regression
- e-Exponential regression
- ab-Exponential regression
- Power regression
- Inverse regression
- Quadratic regression
- Regression analysis (integrated)
- Regression estimate (integrated)

Figure 4.11: The Keisan Online Calculator website

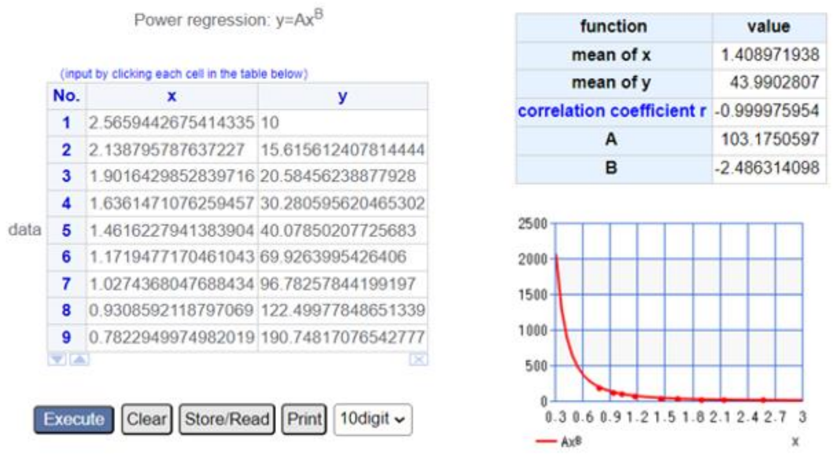
The first specific gas that needs to be found was ammonia. Figure 4.12 shows the ammonia line (redpoint) was plotted on the MQ135 datasheet using WebPlotDigitizer. After obtaining the point plotted, the Keisan Online Calculator was used to get the values A and B with the points obtained in the MQ135 datasheet. As shown in Figure 4.13, the values A and B of ammonia gas were 103.18 and -2.483.



Variables: X, Y

```
10, 2.5659442675414335
10, 2.5659442675414335
15.615612407814444, 2.138795787637227
20.58456238877928, 1.9016429852839716
30.280595620465302, 1.6361471076259457
40.07850207725683, 1.4616227941383904
69.9263995426406, 1.1719477170461043
96.78257844199197, 1.0274368047688434
122.49977848651339, 0.9308592118797069
190.74817076542777, 0.7822949974982019
```

Figure 4.12: The data point of plotted of ammonia


Power regression: $y=Ax^B$

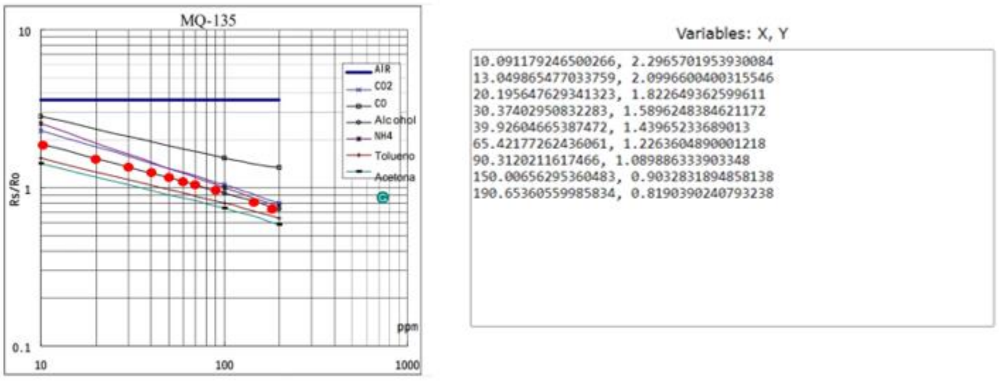
(input by clicking each cell in the table below)

	No.	x	y
data	1	2.5659442675414335	10
	2	2.138795787637227	15.615612407814444
	3	1.9016429852839716	20.58456238877928
	4	1.6361471076259457	30.280595620465302
	5	1.4616227941383904	40.07850207725683
	6	1.1719477170461043	69.9263995426406
	7	1.0274368047688434	96.78257844199197
	8	0.9308592118797069	122.49977848651339
	9	0.7822949974982019	190.74817076542777

function	value
mean of x	1.408971938
mean of y	43.9902807
correlation coefficient r	-0.999975954
A	103.1750597
B	-2.486314098

Figure 4.13: The result of values A and B for ammonia

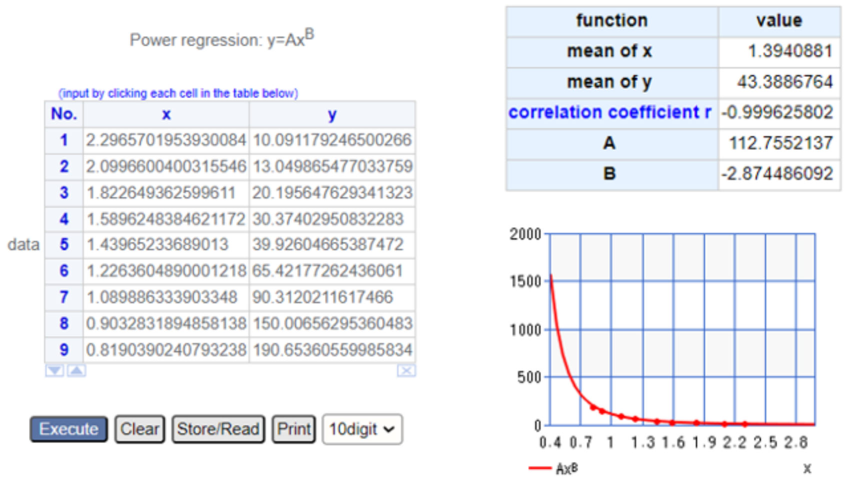
The following specific gas that needs to be found was carbon dioxide. Figure 4.14 shows the carbon dioxide line (redpoint) was plotted on the MQ135 datasheet using WebPlotDigitizer. After obtaining the point plotted, the Keisan Online Calculator was used to get the values A and B with the points obtained in the MQ135 datasheet. As shown in Figure 4.15, the values A and B of carbon dioxide gas were 112.75 and -2.874.



Variables: X, Y

```
10.091179246500266, 2.2965701953930084
13.049865477033759, 2.0996600400315546
20.195647629341323, 1.822649362599611
30.37402950832283, 1.5896248384621172
39.92604665387472, 1.43965233689013
65.42177262436061, 1.2263604890001218
90.3120211617466, 1.089886333903348
150.00656295360483, 0.9032831894858138
190.65360559985834, 0.8190390240793238
```

Figure 4.14: The data point of plotted of carbon dioxide


Power regression: $y=Ax^B$

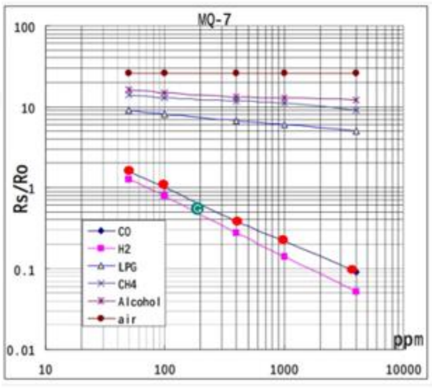
(input by clicking each cell in the table below)

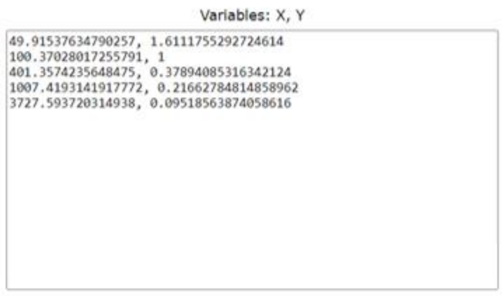
No.	x	y
1	2.2965701953930084	10.091179246500266
2	2.0996600400315546	13.049865477033759
3	1.822649362599611	20.195647629341323
4	1.5896248384621172	30.37402950832283
5	1.43965233689013	39.92604665387472
6	1.2263604890001218	65.42177262436061
7	1.089886333903348	90.3120211617466
8	0.9032831894858138	150.00656295360483
9	0.8190390240793238	190.65360559985834

function	value
mean of x	1.3940881
mean of y	43.3886764
correlation coefficient r	-0.999625802
A	112.7552137
B	-2.874486092

Figure 4.15: The result of values A and B for carbon dioxide

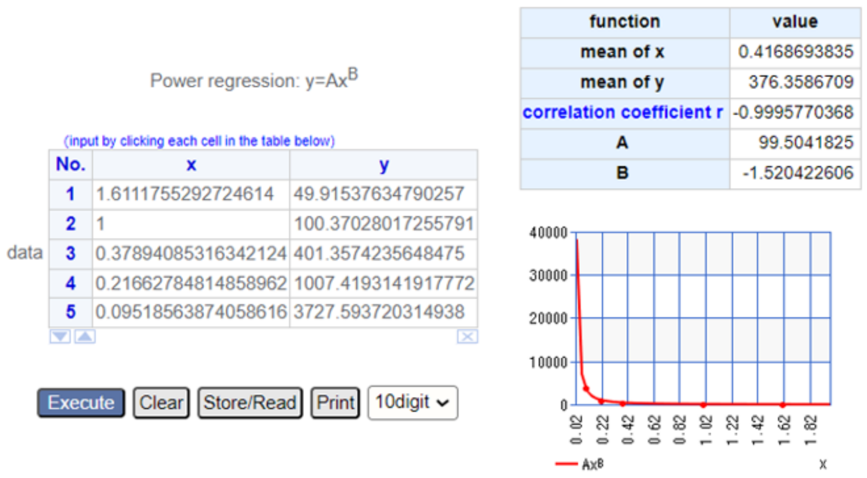
The following specific gas that needs to be found was carbon monoxide. Figure 4.16 shows the carbon monoxide line (redpoint) was plotted on the MQ7 datasheet using WebPlotDigitizer. After obtaining the point plotted, the Keisan Online Calculator was used to get the values A and B with the points obtained in the MQ7 datasheet. As shown in Figure 4.17, the values A and B of carbon monoxide gas were 99.5 and -1.520.




Variables: X, Y

```
49.91537634790257, 1.6111755292724614
100.37028017255791, 1
401.3574235648475, 0.37894085316342124
1007.4193141917772, 0.21662784814858962
3727.593720314938, 0.09518563874058616
```

Figure 4.16: The data point of plotted of carbon monoxide



Power regression: $y=Ax^B$

(input by clicking each cell in the table below)

	No.	x	y
data	1	1.6111755292724614	49.91537634790257
	2	1	100.37028017255791
	3	0.37894085316342124	401.3574235648475
	4	0.21662784814858962	1007.4193141917772
	5	0.09518563874058616	3727.593720314938

function	value
mean of x	0.4168693835
mean of y	376.3586709
correlation coefficient r	-0.9995770368
A	99.5041825
B	-1.520422606

Figure 4.17: The result of values A and B for carbon monoxide

After finding all the values A and B, the value will be included in the system code to let the sensor detect the specific gases accurately. Table 4.3 shows the values A and B for a specific gas.

Table 4.3: The summary of value A and B

	A	B
NH_3	103.18	-2.483
CO_2	112.75	-2.874
CO	99.5	-1.520

4.4 System testing

For system testing, there sensors must ensure that can detect the specific gas in the air and whether the entire system or the data was stable. Also, system testing can know and improve the problem issues.

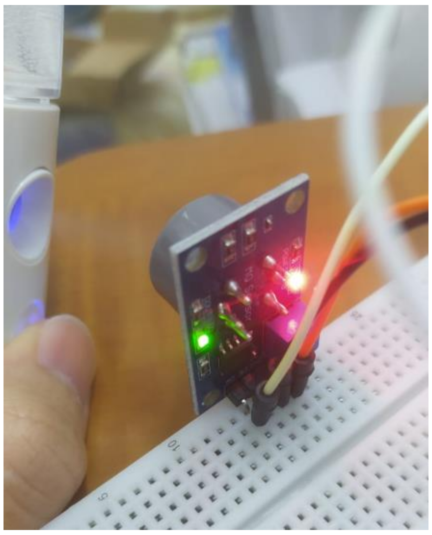


Figure 4.18: The green LCD of the MQ7 light on

Figure 4.18 shows that the green LCD of the MQ7 will light on when the sensor detects alcohol, which was the specific gas that MQ7 can detect. That means the green LCD will light on when the MQ135 and MQ7 successfully detect their specific air.

For the system testing, two tests will be conducted. The first test was whether the MQ135 can detect the specific gas in this project, ammonia. The next test was whether MQ7 can detect the specific gas in this project, carbon monoxide. To test these sensors, the tests must be done outdoors because indoor long-term inhalation of these gases in a closed range will harm human health, and ammonia has a pungent smell.

Figure 4.19 shows the first test was whether MQ135 can detect ammonia outdoors.

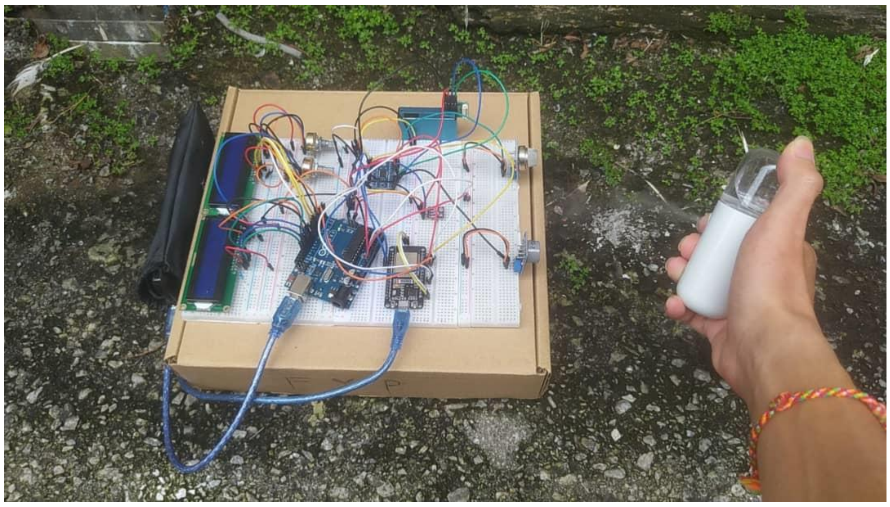


Figure 4.19: The first testing using ammonia

For the ammonia test, a portable sprayer was used to spray the ammonia for the first test. Figure 4.19 shows the testing outside, and the portable sprayer was spraying ammonia near the project's hardware to detect the ammonia. The Blynk used to observe whether the ammonia data has changed or increased.

The Blynk Figure 4.20 shows that the ammonia gas has increased from the original 0.09 ppm to 3.42 ppm. That means the ammonia gas has been detected from the MQ135.

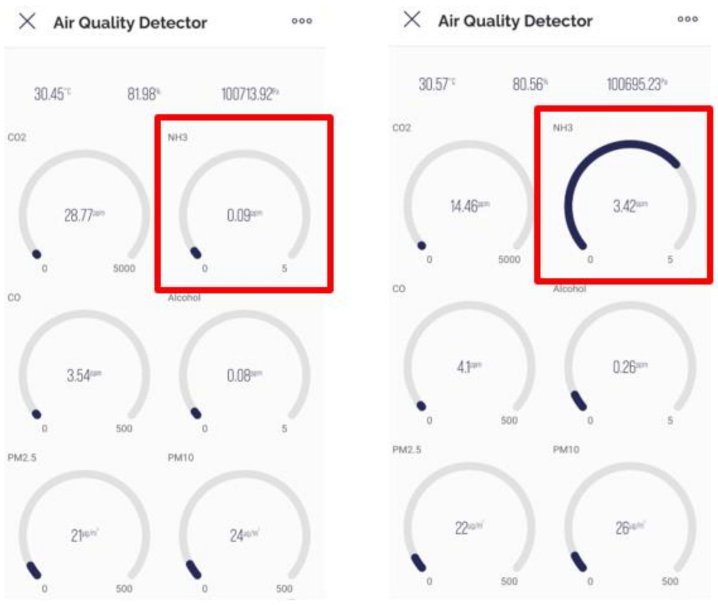


Figure 4.20: The change in the ppm level of ammonia

To ensure the system can detect and accurately detect the ammonia gas for the MQ135, the test has to be done a few times and record the ppm level of the ammonia gas every time spraying. Table 4.4 shows the ammonia gas ppm value record every time it was sprayed and the average ammonia value.

Table 4.4: The average value of ammonia gas

spray	NH_3 (ppm)
1	6.52
2	5.14
3	3.47
4	6.41
5	3.56
Average (ppm)	5.12

Figure 4.21 shows the second test was whether MQ7 can detect carbon monoxide outdoors.

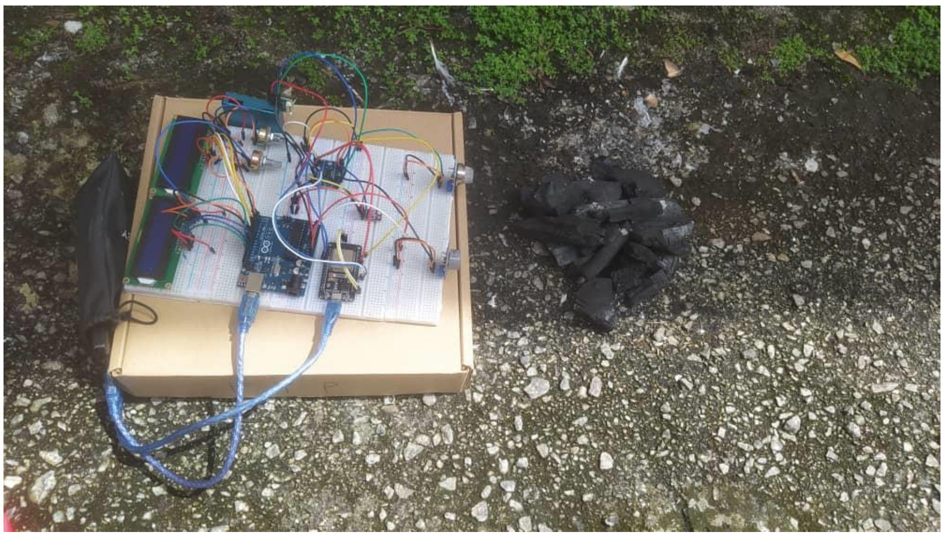


Figure 4.21: The first testing using charcoal

For the second test, charcoal was used for the test since when burned the charcoal can produce carbon monoxide. Figure 4.21 shows the testing outside. The project's hardware was near the charcoal to detect the carbon monoxide. The Blynk used to observe whether the carbon monoxide data has changed or increased.

The Blynk in Figure 4.22 shows that the carbon monoxide gas has increased from the original 8.94 ppm to 32.4 ppm. That means the carbon monoxide gas has been detected from the MQ7.

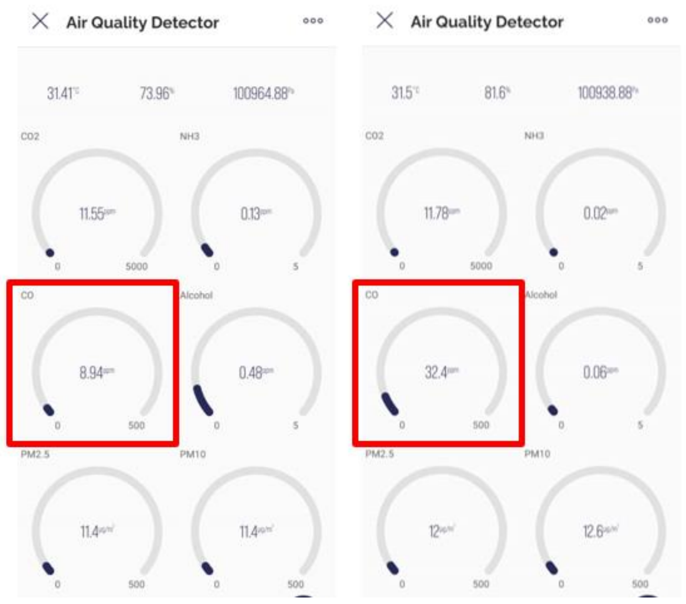


Figure 4.22: The change in the ppm level of carbon monoxide

To ensure the system can detect and accurately detect the carbon monoxide gas for the MQ7, the test has to be done a few times and record the ppm level of the carbon monoxide gas in 15 minutes. Table 4.5 shows the carbon monoxide gas ppm value recorded every 3 minutes and the average value.

Table 4.5: The average value of carbon monoxide gas

Time(m)	CO (ppm)
3	31.6
6	35.2
9	39.4
12	37.9
15	32.1
Average (ppm)	35.24

4.5 System implement

The system was implemented after system testing. First, the project's hardware was placed in the room to collect the data and the AQI. Figure 4.23 shows the hardware placed in the room. Next, the hardware was placed outside or outdoors to collect the data and the AQI. Figure 4.24 shows the hardware placed at the outside house.

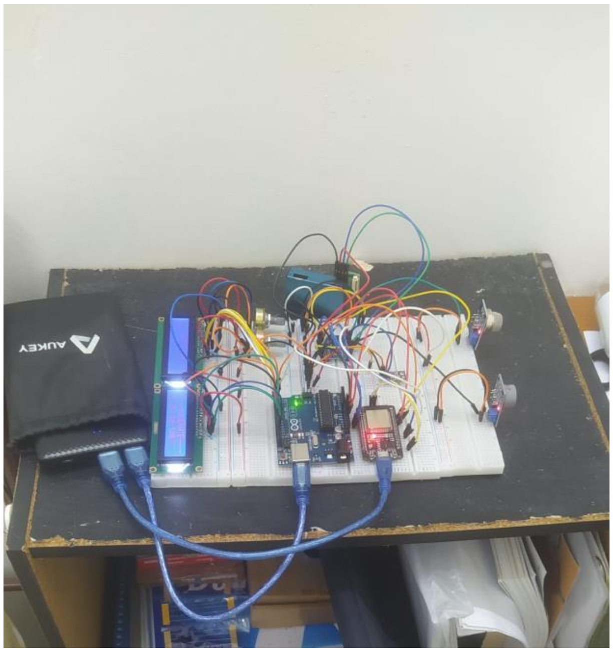


Figure 4.23: The hardware placed at indoor

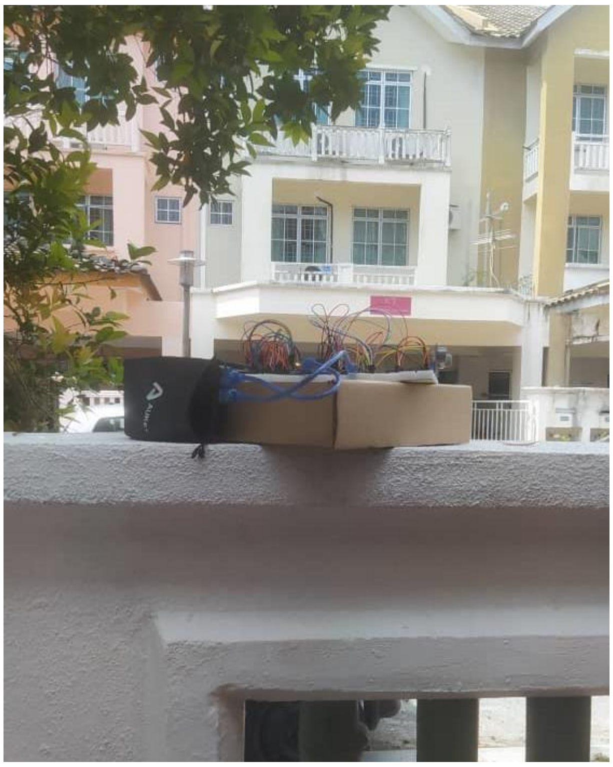


Figure 4.24: The hardware placed at outdoor

After placing the hardware outdoors and indoors, the Blynk can used to track the real-time AQI and other gas levels. The data live on Blynk can viewed or utilized in the history graph widget in Blynk to follow previously recorded data. The widget on Blynk can compare a widget's value to information from previous hours, days, weeks, or even months. Data may be instantly seen in the Reports Widget or exported in CSV format. Another choice was the Get Device Report tool, which enables one to obtain past information from a single device for a predetermined period. Figure 4.25 shows one of the AQI data live on the Blynk.

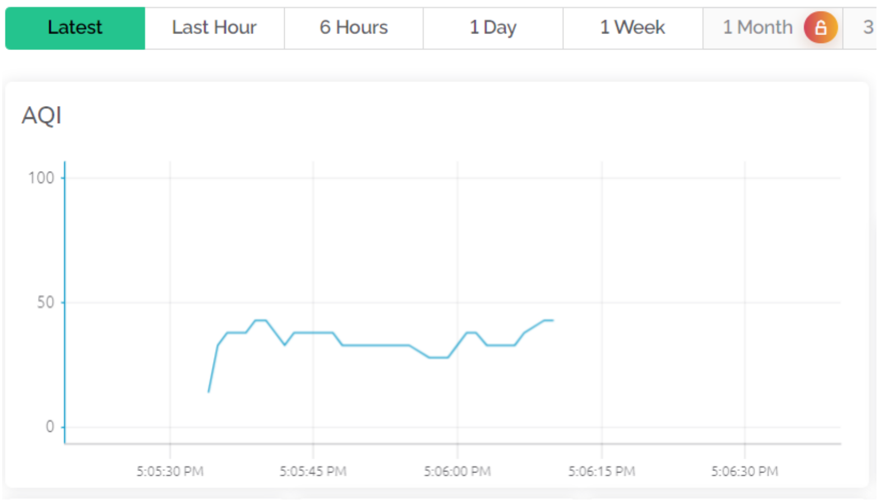


Figure 4.25: The AQI data show in Blynk

For the LCD, when the Arduino UNO successfully received the AQI data from the ESP32, the AQI value displayed on the LCD. The left LCD displayed the AQI real-time value, as Figure 4.26 shows. The LCD displayed the value delay due to the latencies during reads and writes between Arduino UNO and ESP32. The right LCD was displayed the different levels of health concern, which are good, moderate, unhealthy, very unhealthy, and hazardous. It lets the user know the real-time AQI and the stories of health concerns.

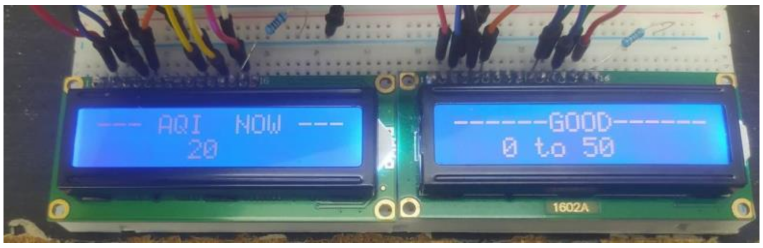


Figure 4.26: The LCDs display the AQI value

4.6 Case studies: Outdoor observation 1

Figure 4.27 to Figure 4.32 below show the AQI waveform of outdoor observation 1 records in 180 minutes, from 4:25 pm to 7:25 pm on the Blynk. The live stream waveform took a screenshot from the Blynk every 30 minutes. The date of the record was 8.9.2023.

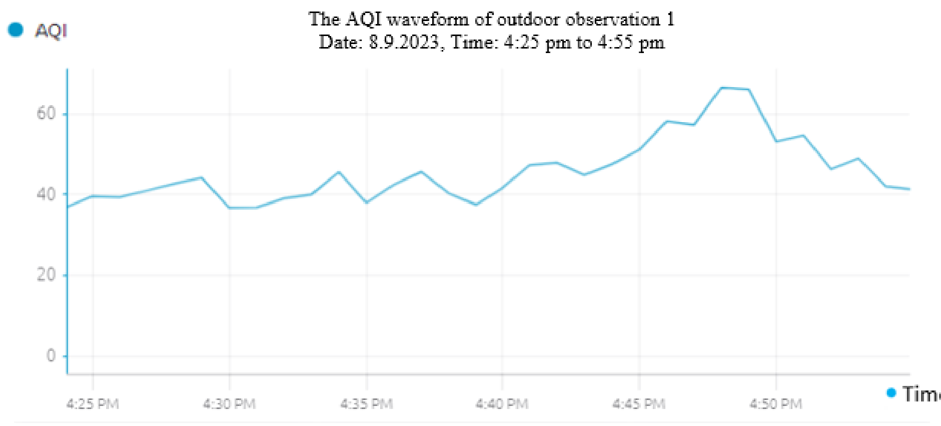


Figure 4.27: The AQI waveform of outdoor observation 1 for 30 mins

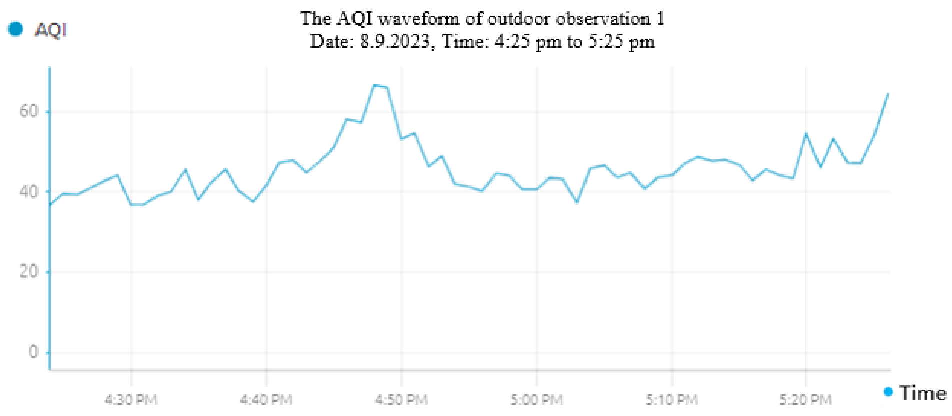


Figure 4.28: The AQI waveform of outdoor observation 1 for 60 mins

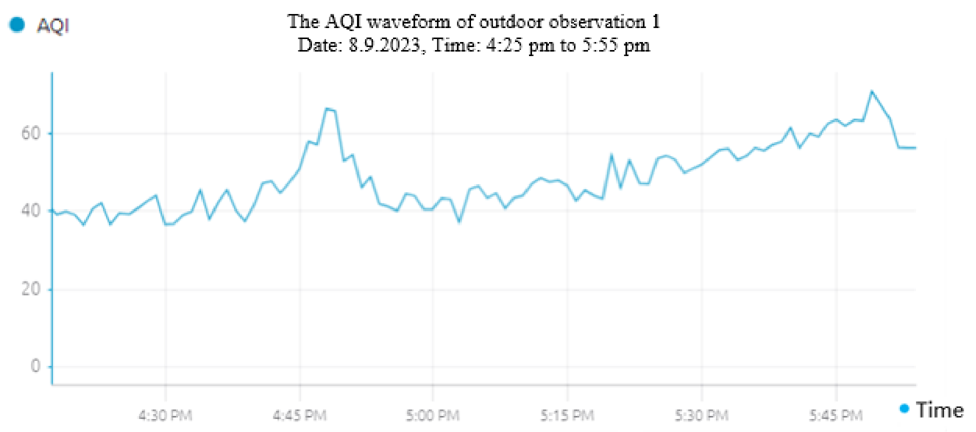


Figure 4.29: The AQI waveform of outdoor observation 1 for 90 mins

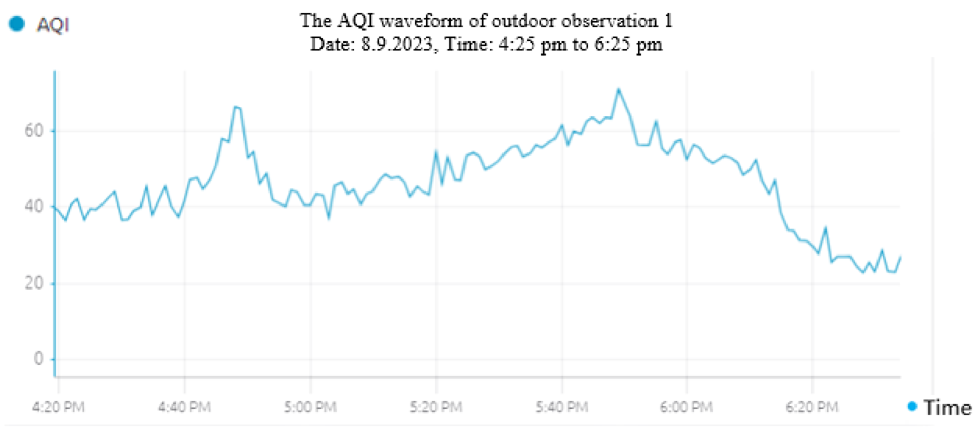


Figure 4.30: The AQI waveform of outdoor observation 1 for 120 mins

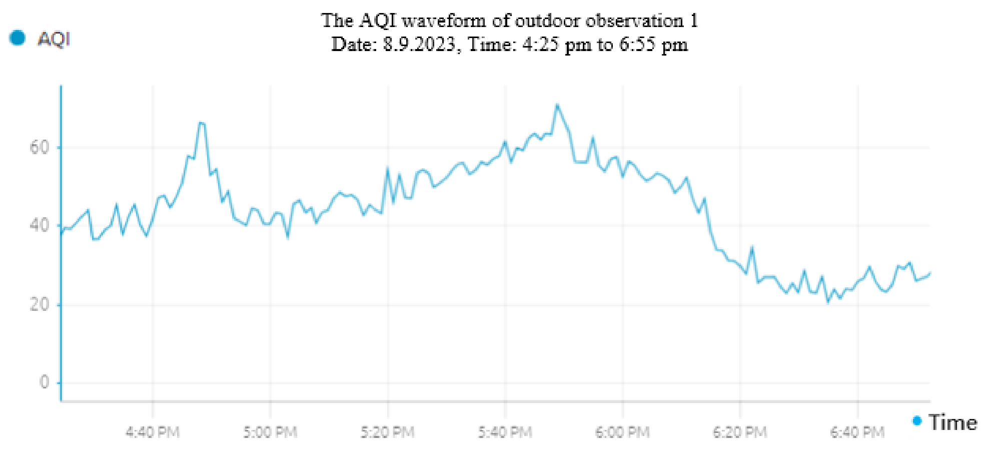


Figure 4.31: The AQI waveform of outdoor observation 1 for 150 mins

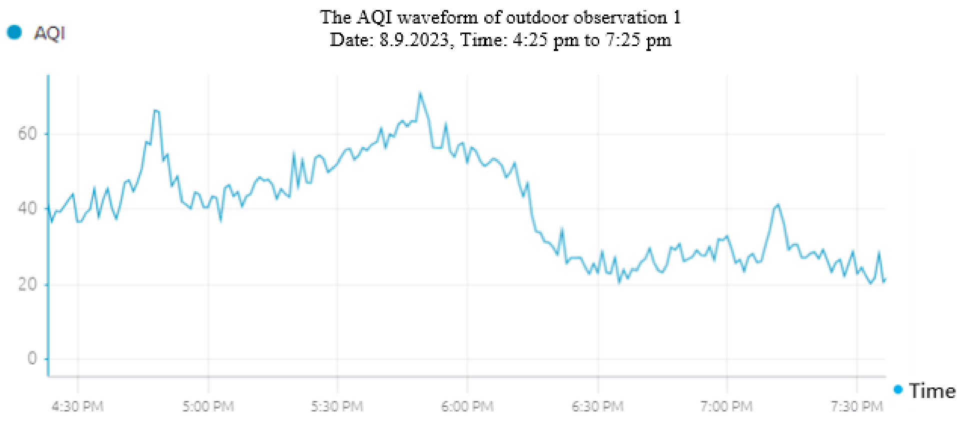


Figure 4.32: The AQI waveform of outdoor observation 1 for 180 mins

Judging from the waveform from Figure 4.27 to Figure 4.32, this was the first time the hardware detects the air quality outdoors. Starting at 4:45 p.m., the AQI rose from the original average of 40 to 62. At 6 p.m., it dropped seriously to close to 20. After adjusted the hardware, the AQI waveform rose to a maximum AQI value of 62. And afterwards, the AQI dropdown to the average AQI value of 30.

The waveform was unstable because the PMS7003 sensor had accumulated dust and debris inside the sensor, affecting its readings. After this experience, the sensor was cleaned using compressed air or a soft brush to remove buildup whenever the hardware collected data outdoors for the outdoor observation.

4.7 Case studies: Outdoor observation 2

Figure 4.33 to Figure 4.38 below show the AQI waveform of outdoor observation 2 records in 180 minutes, from 11:00 am to 2:00 pm on the Blynk. The live stream waveform took a screenshot from the Blynk every 30 minutes. The date of the record was 11.9.2023.

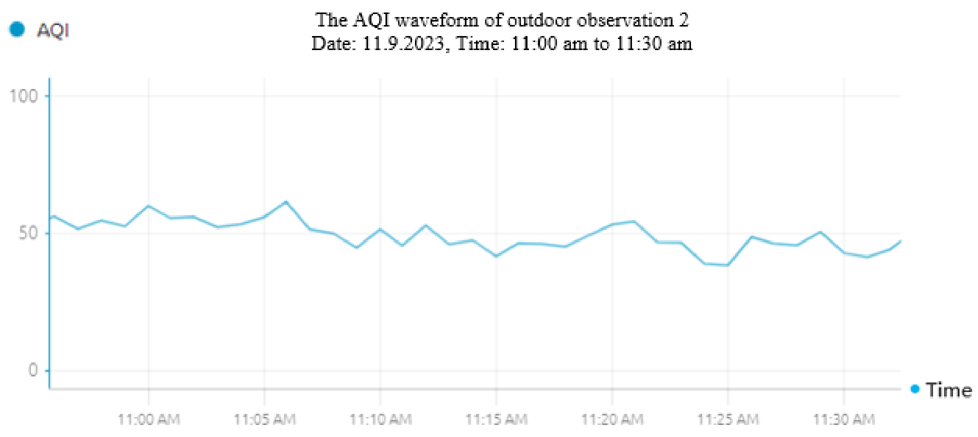


Figure 4.33: The AQI waveform of outdoor observation 2 for 30 mins

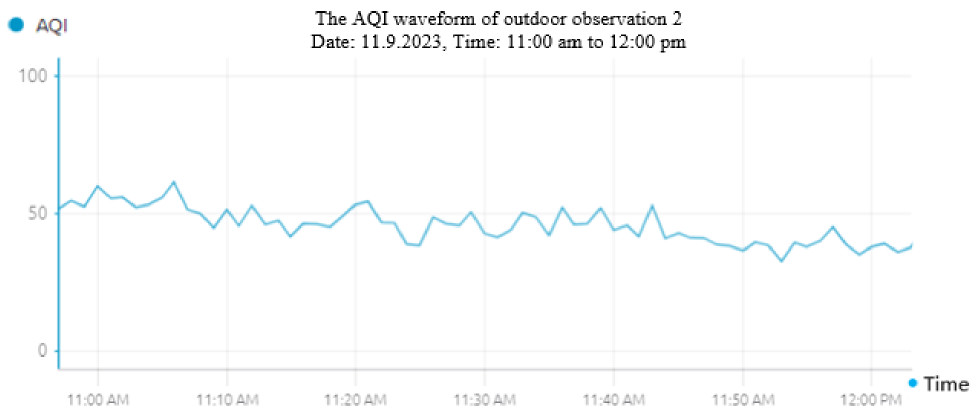


Figure 4.34: The AQI waveform of outdoor observation 2 for 60 mins

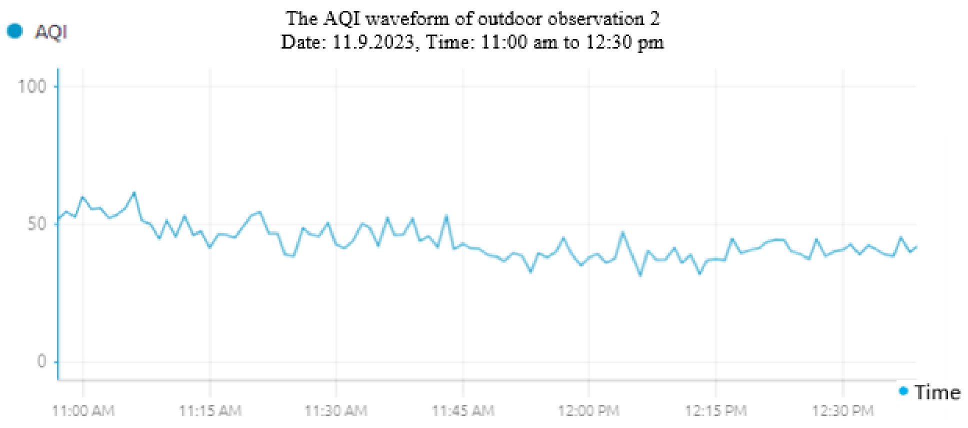


Figure 4.35: The AQI waveform of outdoor observation 2 for 90 mins

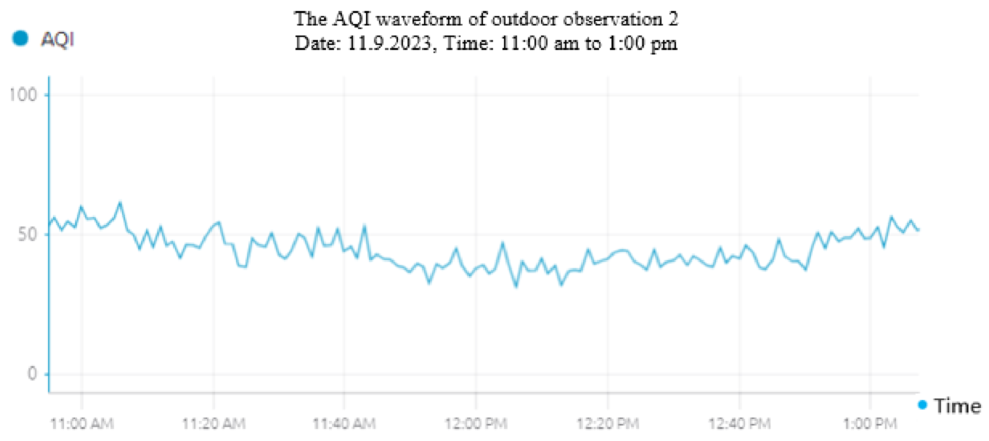


Figure 4.36: The AQI waveform of outdoor observation 2 for 120 mins

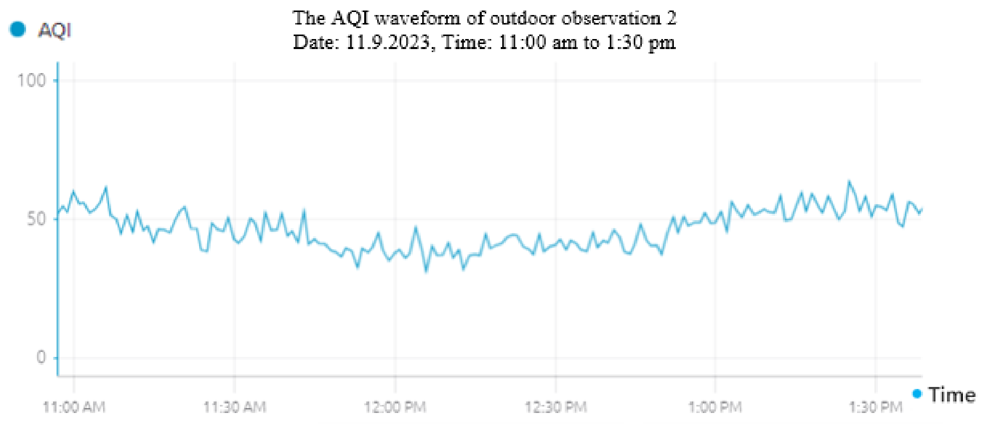


Figure 4.37: The AQI waveform of outdoor observation 2 for 150 mins

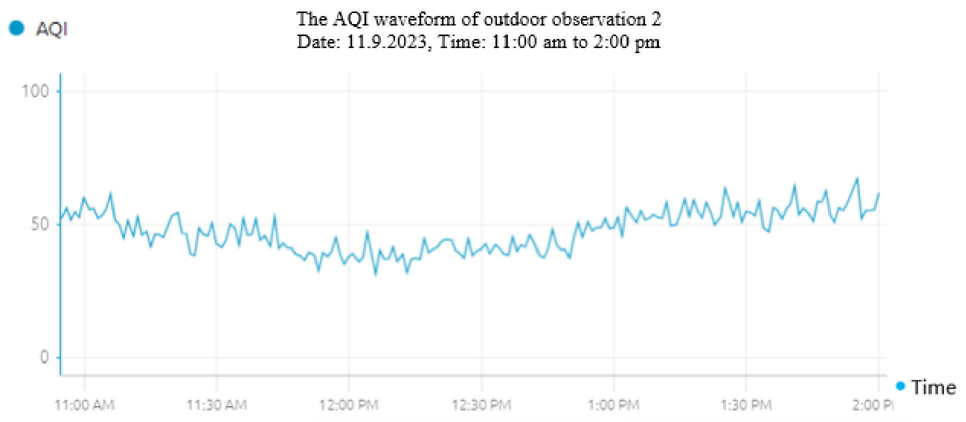


Figure 4.38: The AQI waveform of outdoor observation 2 for 180 mins

The waveform From Figure 4.33 to Figure 4.38 shows, the maximum value of AQI was 57 at around 1:55 pm, and the minimum value of AQI was 42 at around 12:20 pm. The average AQI value was around 49. Compared with outdoor observation 1, the waveform shows the air quality of outdoor observation 2 was stable.

4.8 Case studies: Outdoor observation 3

Figure 4.39 to Figure 4.44 below show the AQI waveform of outdoor observation 3 records in 180 minutes, from 7:30 pm to 10:30 pm on the Blynk. The live stream waveform took a screenshot from the Blynk every 30 minutes. The date of the record was 12.9.2023.

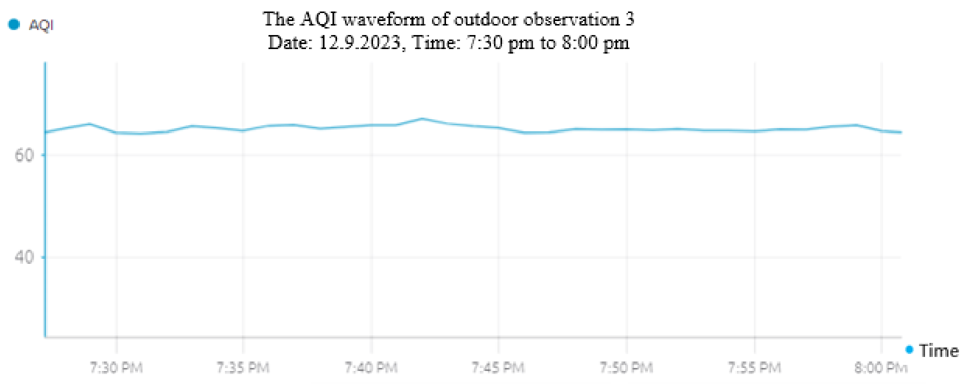


Figure 4.39: The AQI waveform of outdoor observation 3 for 30 mins

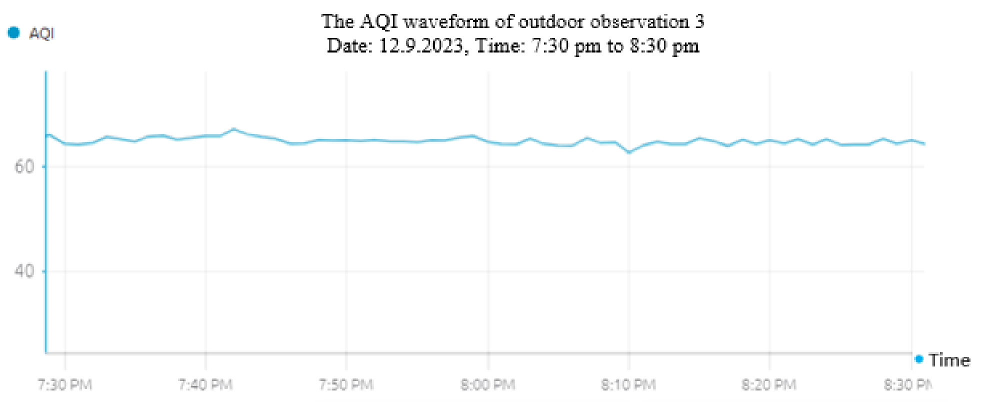


Figure 4.40: The AQI waveform of outdoor observation 3 for 60 mins

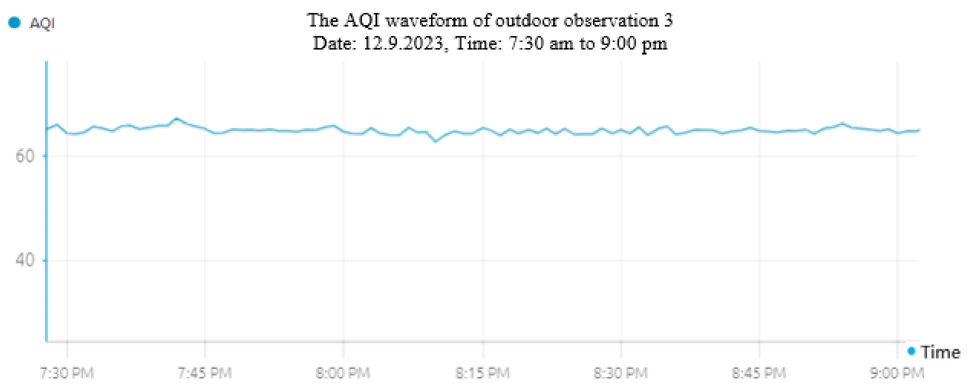


Figure 4.41: The AQI waveform of outdoor observation 3 for 90 mins

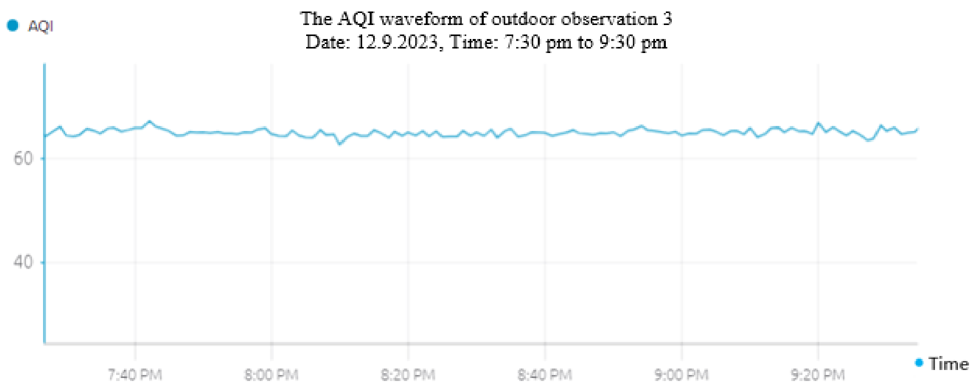


Figure 4.42: The AQI waveform of outdoor observation 3 for 120 mins

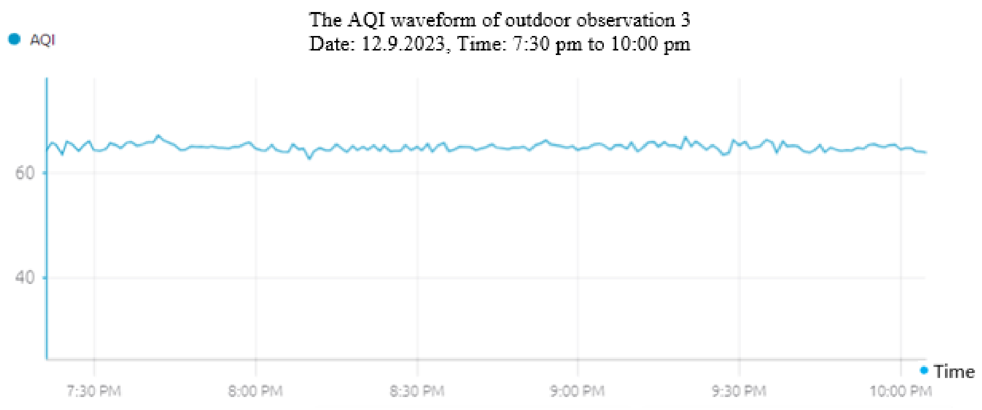


Figure 4.43: The AQI waveform of outdoor observation 3 for 150 mins

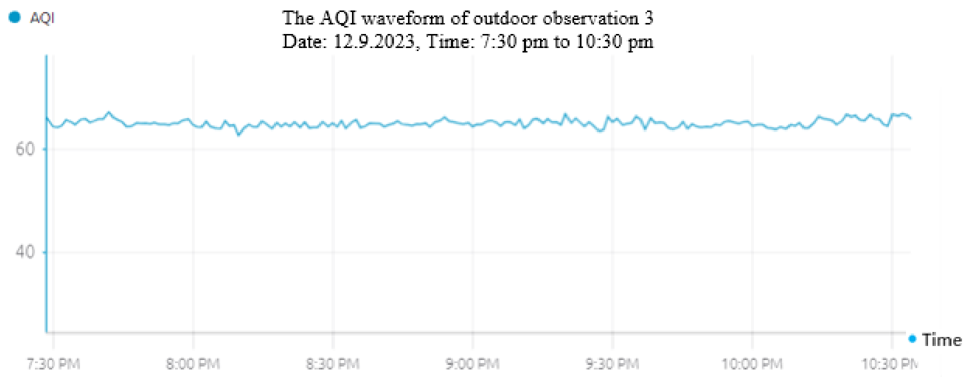


Figure 4.44: The AQI waveform of outdoor observation 3 for 180 mins

As the waveform from Figure 4.39 to Figure 4.44 show, the waveform of AQI in outdoor observation 3 was stabled, which was compiled with observation 1 and observation 2. The maximum AQI was 64 at around 7:45 pm, and the minimum AQI was 62 at around 8:10 pm. The average of the AQI value was 62.

4.9 Case studies: Indoor observation 1

Figure 4.45 to Figure 4.50 below show the AQI waveform of indoor observation 1 records in 180 minutes, from 7:30 pm to 10:30 pm on the Blynk. The live stream waveform took a screenshot from the Blynk every 30 minutes. The date of the record was 7.9.2023.

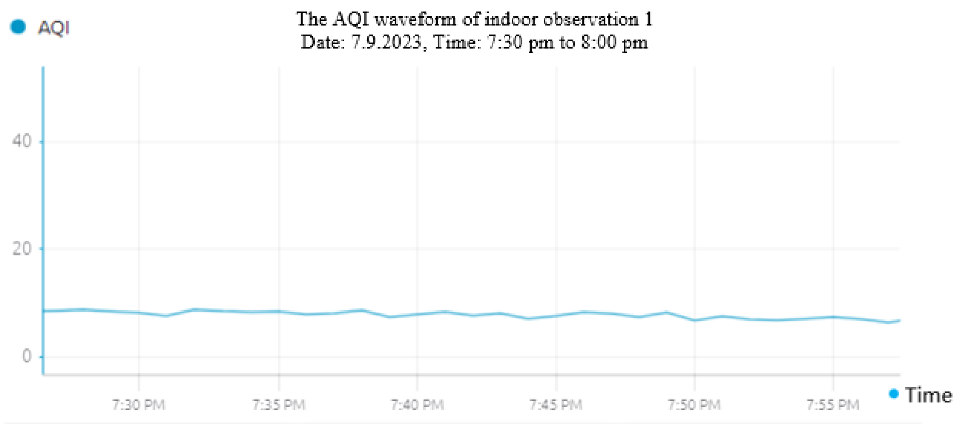


Figure 4.45: The AQI waveform of indoor observation 1 for 30 mins

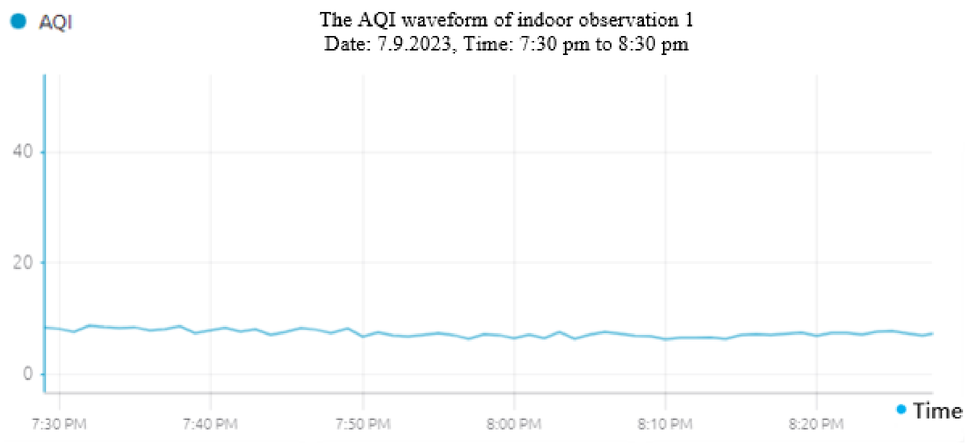


Figure 4.46 : The AQI waveform of indoor observation 1 for 60 mins

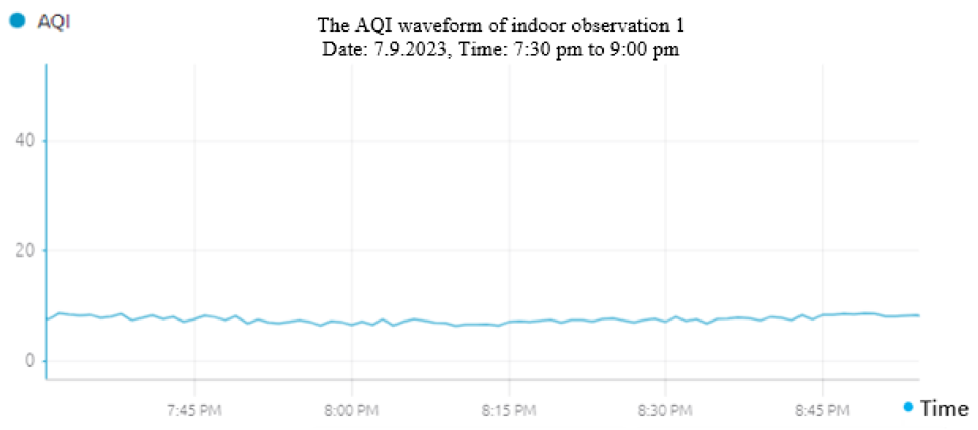


Figure 4.47: The AQI waveform of indoor observation 1 for 90 mins

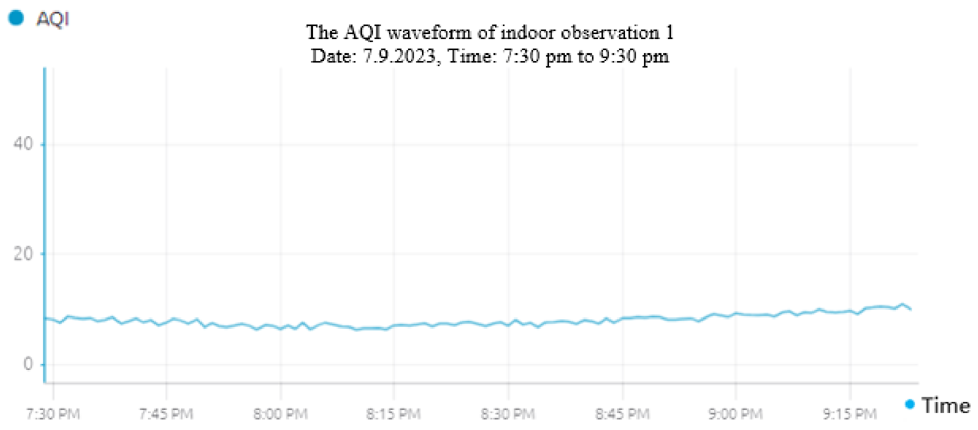


Figure 4.48: The AQI waveform of indoor observation 1 for 120 mins

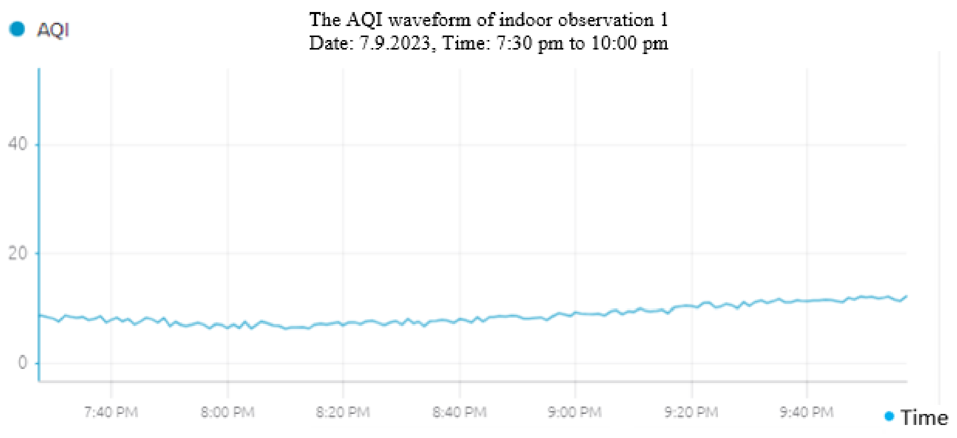


Figure 4.49: The AQI waveform of indoor observation 1 for 150 mins

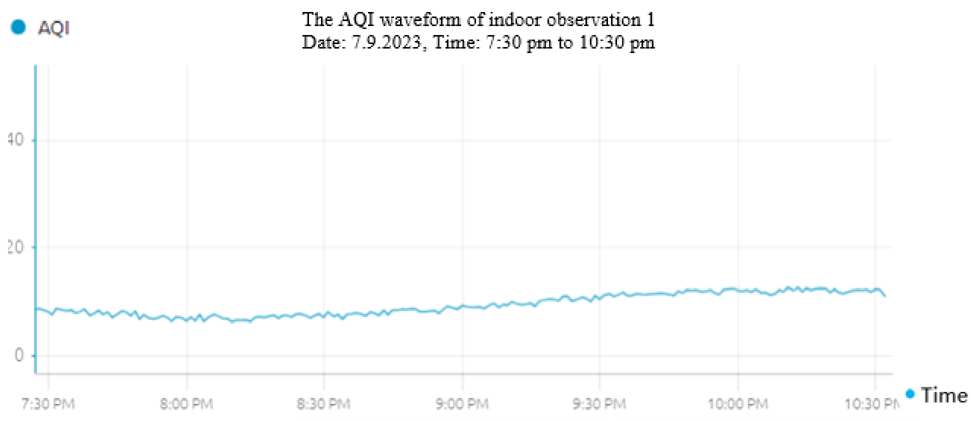


Figure 4.50: The AQI waveform of indoor observation 1 for 180 mins

The waveform from Figure 4.45 to Figure 4.50 shows that the maximum value of AQI was 16 at around 10:15 pm, and the minimum value of AQI was 8 at around 8:00 pm. The average AQI value was around 8 for the indoor observation 1.

4.10 Case studies: Indoor observation 2

Figure 4.51 to Figure 4.56 below show the AQI waveform of indoor observation 2 records in 180 minutes, from 9:00 am to 12:00 pm on the Blynk. The live stream waveform took a screenshot from the Blynk every 30 minutes. The date of the record was 10.9.2023.

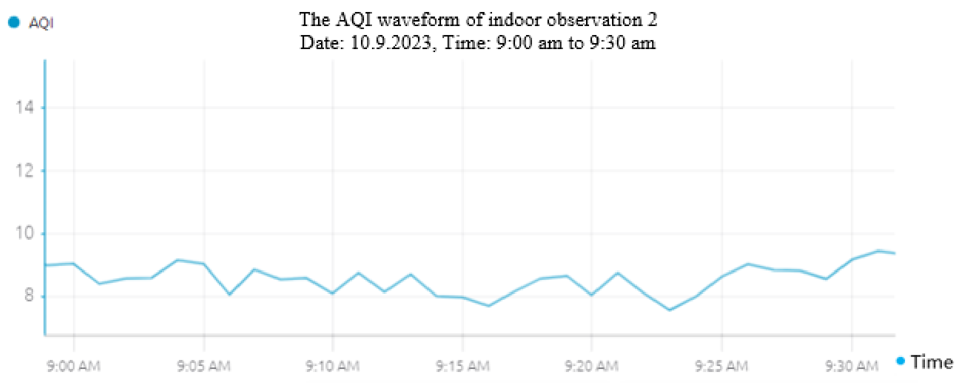


Figure 4.51: The AQI waveform of indoor observation 2 for 30 mins

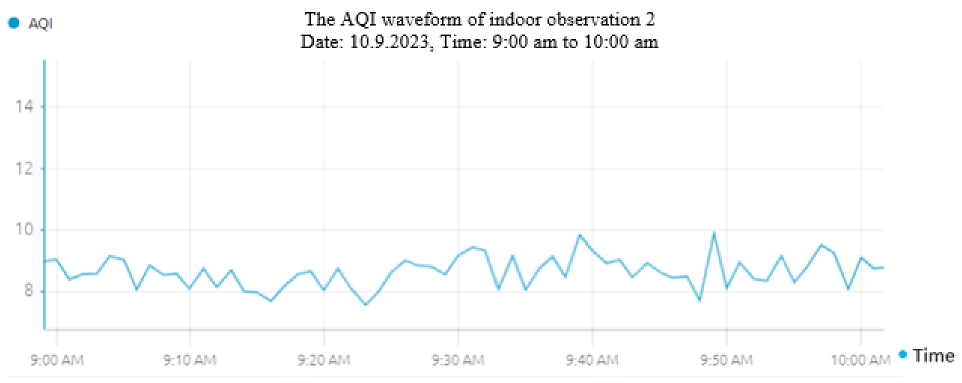


Figure 4.52: The AQI waveform of indoor observation 2 for 60 mins

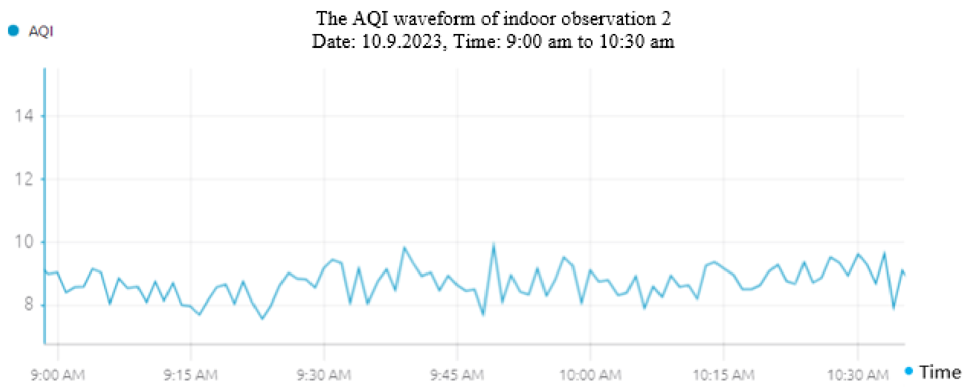


Figure 4.53: The AQI waveform of indoor observation 2 for 90 mins

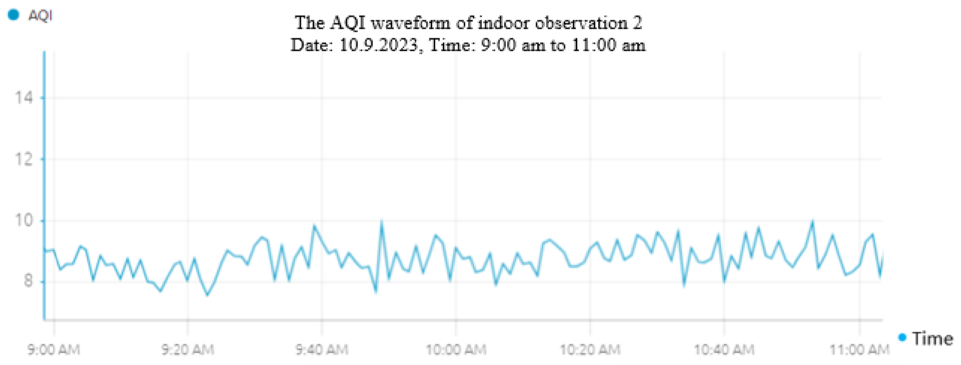


Figure 4.54: The AQI waveform of indoor observation 2 for 120 mins

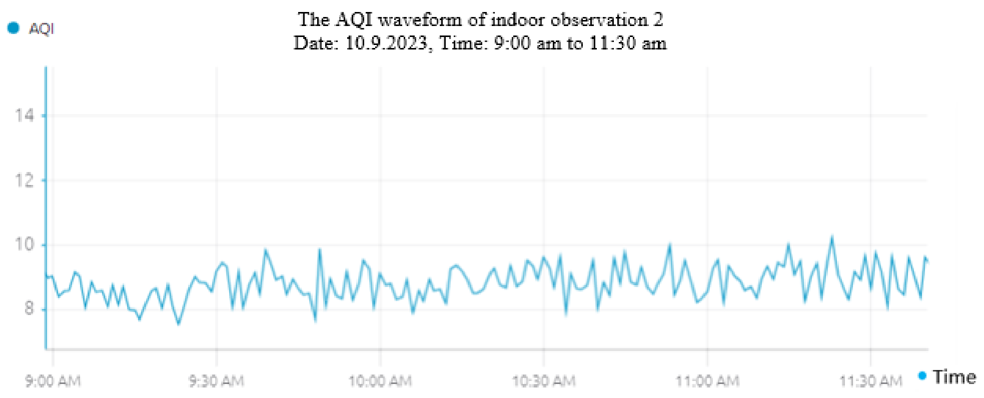


Figure 4.55: The AQI waveform of indoor observation 2 for 150 mins

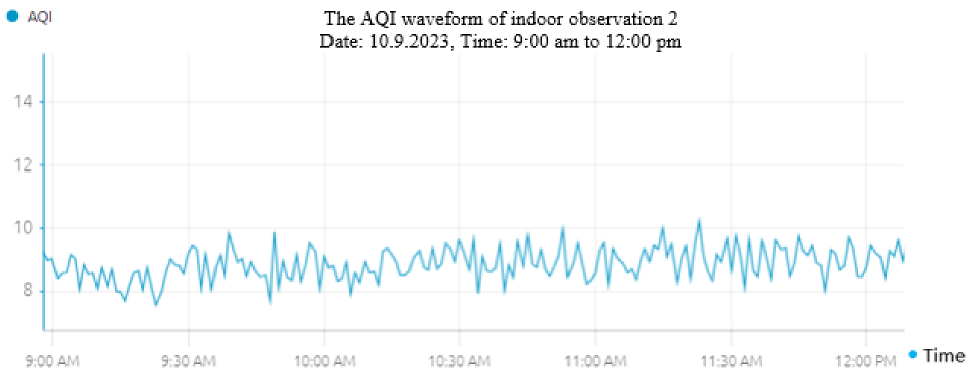


Figure 4.56: The AQI waveform of indoor observation 2 for 180 mins

The waveform from Figure 4.51 to Figure 4.56 show that the maximum value of AQI was 11 at around 11:25 am, and the minimum value of AQI was 6 at around 9:25 am. The average AQI value was around 8 for the indoor observation 2.

4.11 Case studies: Indoor observation 3

Figure 4.57 to Figure 4.62 below show the AQI waveform of indoor observation 3 records in 180 minutes, from 2:00 pm to 5:00 pm on the Blynk. The live stream waveform took a screenshot from the Blynk every 30 minutes. The date of the record was 12.9.2023.

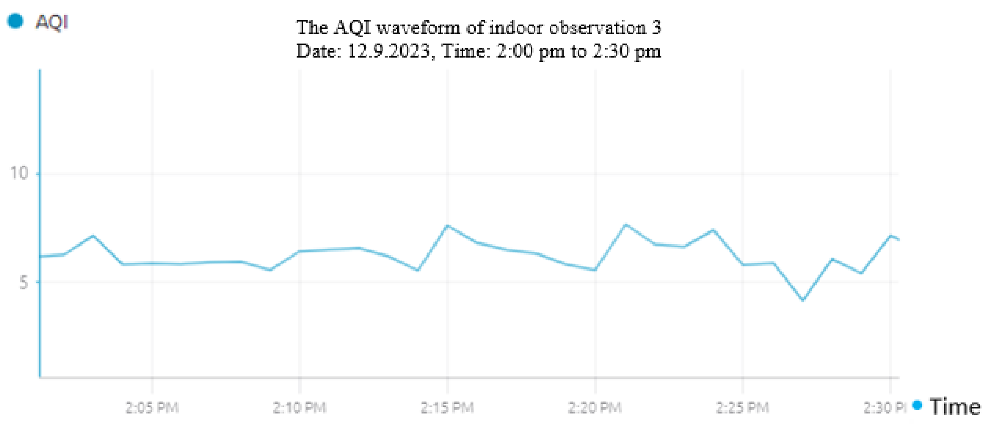


Figure 4.57: The AQI waveform of indoor observation 3 for 30 mins

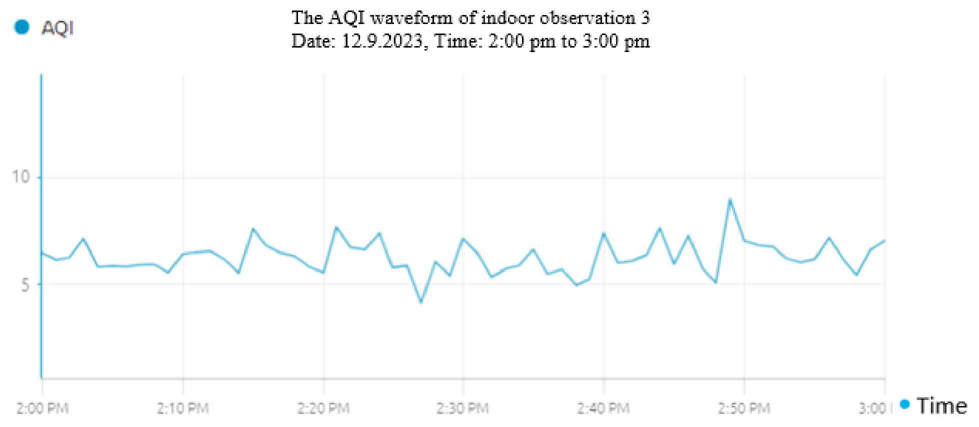


Figure 4.58: The AQI waveform of indoor observation 3 for 60 mins

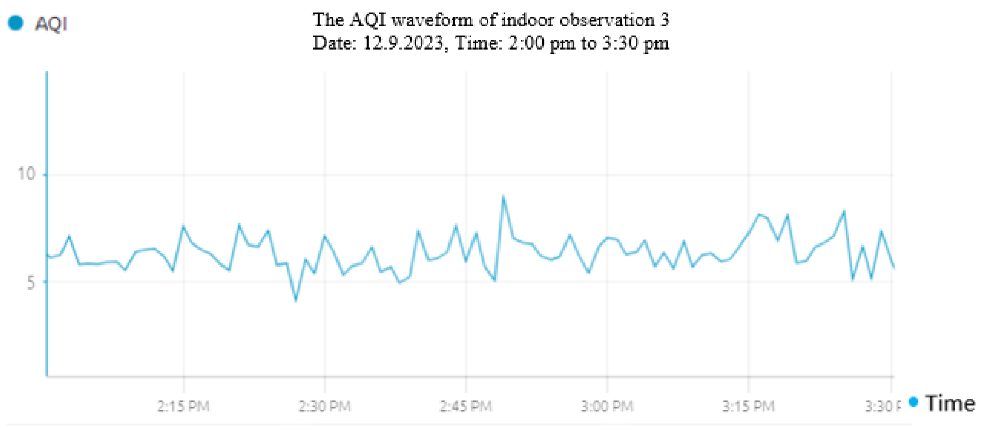


Figure 4.59: The AQI waveform of indoor observation 3 for 90 mins

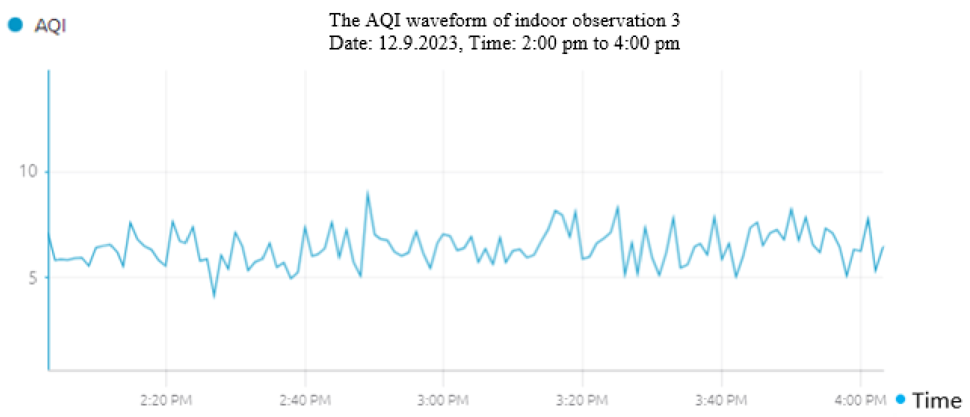


Figure 4.60: The AQI waveform of indoor observation 3 for 120 mins

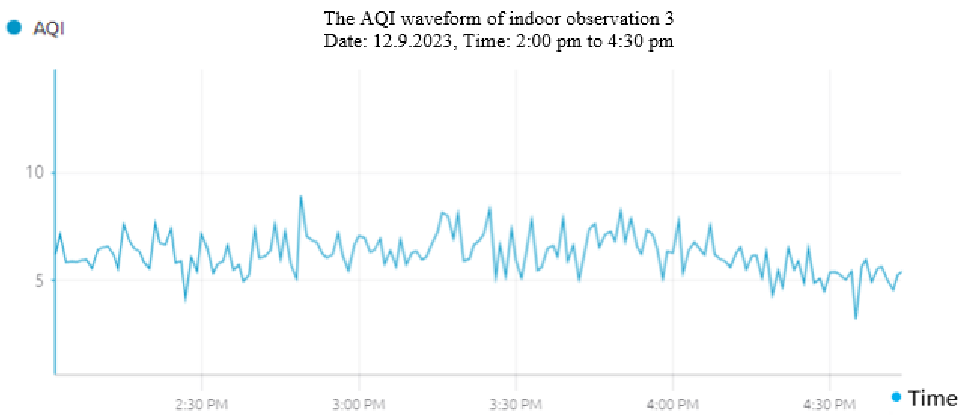


Figure 4.61: The AQI waveform of indoor observation 3 for 150 mins

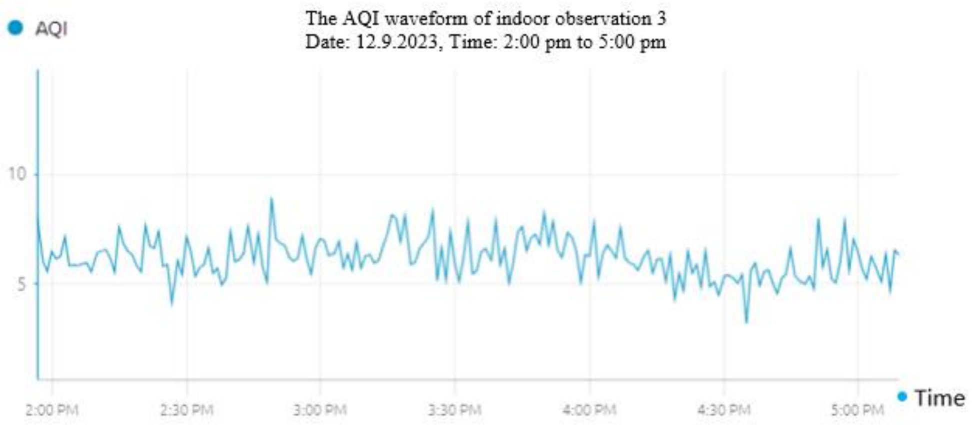


Figure 4.62: The AQI waveform of indoor observation 3 for 180 mins

The waveform from Figure 4.57 to Figure 4.62 show that the maximum value of AQI was 8 at around 2:50 pm, and the minimum value of AQI was 3 at around 4:35 pm. The average AQI value was around 6 for the indoor observation 3.

4.12 Summary of the case studies

Based on the three outdoor observations, the AQI value was between 40 to 70. An AQI value of 51-100 was considered moderate and acceptable. Still, there may be some health concerns for a small number of unusually sensitive individuals. Houses close to busy roads may experience higher levels of air pollution due to vehicle emissions. Vehicle exhaust was a significant source of air pollutants.

However, based on the three indoor observations, the AQI values are no more than 20. An AQI value of 0-50 represents good air quality with little potential to affect public health. Air quality was considered satisfactory, and air pollution poses little risk. Indoor spaces were typically insulated from outdoor pollutants such as vehicle emissions, dust, and pollen. This reduces the introduction of external pollution into the indoor environment.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The project of the project is to detect the air quality using an embedded system. It uses sensors to measure the air's temperature, humidity, pressure, gas, and particle concentration. Users may track the air quality at any time because of these sensors' ability to capture real-time data. The embedded system may be used to determine several air quality indices, including the concentration of CO_2, PM2.5, and PM10 particulates, among others. These indicators aid in determining the extent of environmental pollution and assisting in implementing appropriate air quality improvement initiatives.

The Blynk platform has data visualization capabilities that can present monitoring results to users through graphics, charts, or reports. This helps users better understand air quality conditions and take timely action. The mobile app of Blynk lets the users track the real-time air quality anytime and anywhere.

In conclusion, using embedded systems to detect air quality has helped protect the environment. Through timely monitoring and reporting of air pollution, policymakers can be motivated to reduce pollution sources, improve air quality, and protect ecosystems.

5.2 Limitations

The limitations of this project include:

1. Routine Maintenance and Calibration: Routine maintenance and calibration are crucial to guarantee the accuracy of readings. This can be expensive and resource-intensive, particularly for networks with many monitoring stations.

2. Accuracy and Precision: The accuracy and precision of the sensors may be constrained in embedded systems. Compared to specialist laboratory equipment, cheaper sensors frequently integrated into embedded systems could not offer extremely accurate measurements.

3. Sensor Drift: Embedded system sensors may drift or lose accuracy over time, necessitating routine calibration and maintenance to guarantee correct data gathering.

4. Limited Air Quality Metrics: Embedded systems generally only monitor a few air quality metrics, such as temperature, humidity, and gases like CO, CO_2, and others. They could not include all types of possible air contaminants.

5.3 Recommendations for Future Improvement

The recommendations for future improvement of this project include:

1. Automated Calibration and Self-Diagnostics: When it comes to calibration and maintenance, create automated calibration procedures for embedded systems to lessen the frequency of human calibration. Implement self-diagnostic tools to find and report drift or malfunctioning sensors in real-time.

2. Expanded Sensor Capabilities: Include sensors that can measure various air quality indices, including VOCs, heavy metals, and particular contaminants relevant to local circumstances.

3. Data Quality Improvement: Incorporate data validation algorithms to detect and fix inaccurate values. Improve data accuracy by implementing redundancy utilizing numerous sensors for the same parameter.

REFERENCES

ABEDIN, M., 2013. The impact of atmospheric particulate matter pollution on visibility.

Arduino (2019). Arduino Reference. [online] Arduino.cc. Available at: https://www.arduino.cc/reference/en/

ASK Electronics. (2023). MQ-135 Archives. [online] Available at: https://askelectronics.co.ke/product-tag/mq-135/

Badamasi, Y.A., 2014, September. The working principle of an Arduino. In 2014 11th international conference on electronics, computer and computation (ICECCO) (pp. 1-4). IEEE.

Behera, B.K. and Prasad, R., 2020. Air pollution and controlling measures. *Environmental technology and sustainability. Elsevier*, pp.169-199.

Bentoncleanair (n.d.). Benton Clean Air Agency. [online] Available at: https://bentoncleanair.org/air-quality/air-quality-index

Berger, A., 2001. Embedded systems design: an introduction to processes, tools, and techniques. CRC Press.

Bilal (2022). BME280 with ESP32 ESP-IDF and Display Readings on OLED. [online] ESP32 ESP-IDF. Available at: https://esp32tutorials.com/bme280-esp32-esp-idf-oled/

Bohlin, P., Jones, K.C. and Strandberg, B., 2007. Occupational and indoor air exposure to persistent organic pollutants: A review of passive sampling techniques and needs. Journal of Environmental Monitoring, 9(6), pp.501-509.

Bosch Sensortec. (n.d.). BME280. [online] Available at: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/

Brunekreef, B. and Forsberg, B., 2005. Epidemiological evidence of effects of coarse airborne particles on health. European respiratory journal, 26(2), pp.309-318.

Centre for Research on Energy and Clean Air (CREA) (2022). The Health and Economic Impacts of Ambient Air Quality in Malaysia. Greenpeace Malaysia

Chan, D., 2022. Vehicles outnumber people in Malaysia. New Straits Times.

Chin, Y.S.J., De Pretto, L., Thuppil, V. and Ashfold, M.J., 2019. Public awareness and support for environmental protection—A focus on air pollution in peninsular Malaysia. PloS one, 14(3), p.e0212206.

Chen, H., Jia, X. and Li, H., 2011, October. A brief introduction to IoT gateway. In IET international conference on communication technology and application (ICCTA 2011) (pp. 610-613). IET.

Choudhary, M.P. and Garg, V., 2013, August. Causes, consequences and control of air pollution. In All India seminar on methodologies for air pollution control, held at MNIT.

Components101 (2018). Arduino Uno Pin Diagram, Specifications, Pin Configuration & Programming. [online] Available at: https://components101.com/microcontrollers/arduino-uno

Electronicwings (2023). Interfacing LCD 16x2 in 4-bit mode with PIC18F4550 | PIC Control.. [online] Available at: https://www.electronicwings.com/pic/interfacing-lcd-16x2-in-4-bit-mode-with-pic18f4550-

Elias, M.S., Hashim, A., Bahrudin, N.F.D., Sapiee, N.A., Paulus, W., Azman, M.A., Raduian, N.J. and Abdullah, I.M., 2023, July. Assessment and sources identification of air quality pollution in Klang Valley, Kuala Lumpur, Malaysia. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1285, No. 1, p. 012017). IOP Publishing.

Flux.ai. (2023). ESP32 Pinout: Everything You Need to Know. [online] Available at: https://www.flux.ai/p/blog/esp32-pinout-everything-you-need-to-know#:~:text=GPIO%206%20to%20GPIO%2011%20are%20exposed%20in%20some%20ESP32

Hadi, A.S., Alsaker, M., Eshoom, A., Elmnifi, M., Alhmode, M.A. and Habeeb, L.J., 2022. Development of Low-Cost and Multi-Material Sensing Approach for MQ 135 Sensor. ECS Transactions, 107(1), p.17309.

Hui, T.K., Sherratt, R.S. and Sánchez, D.D., 2017. Major requirements for building Smart Homes in Smart Cities based on Internet of Things technologies. Future Generation Computer Systems, 76, pp.358-369.

IoT Agenda. (2015). What is embedded firmware? | Definition from TechTarget. [online] Available at: https://www.techtarget.com/iotagenda/definition/embedded-firmware#:~:text=Embedded%20firmware%20is%20the%20flash.

IoT Agenda. (2020). What is an Embedded System? [online] Available at: https://www.techtarget.com/iotagenda/definition/embedded-system#:~:text=An%20embedded%20system%20is%20a.

Kalia, P. and Ansari, M.A., 2020. IOT based air quality and particulate matter concentration monitoring system. Materials Today: Proceedings, 32, pp.468-475.

Karagulian, F., Barbiere, M., Kotsev, A., Spinelle, L., Gerboles, M., Lagler, F., Redon, N., Crunaire, S. and Borowiak, A., 2019. Review of the performance of low-cost sensors for air quality monitoring. Atmosphere, 10(9), p.506.

Kashif (n.d.). Which Cable is Used for ESP32. [online] Available at: https://linuxhint.com/cable-used-for-esp32/

Kelly, F.J. and Fussell, J.C., 2011. Air pollution and airway disease. Clinical & Experimental Allergy, 41(8), pp.1059-1071.

Kermani, M.M., Zhang, M., Raghunathan, A. and Jha, N.K., 2013, January. Emerging frontiers in embedded security. In 2013 26th international conference on VLSI design and 2013 12th international conference on embedded systems (pp. 203-208). IEEE.

Kocher, P., Lee, R., McGraw, G. and Raghunathan, A., 2004, June. Security as a new dimension in embedded system design. In Proceedings of the 41st annual design automation conference (pp. 753-760).

Last Minute Engineers. (2022). ESP32 Pinout Reference. [online] Available at: https://lastminuteengineers.com/esp32-pinout-reference/

LaunchSchool (2019). What is a variable in computer programming? [online] Launchschool.com. Available at: https://launchschool.com/books/ruby/read/variables

learnsparkfun (2021). Level Shifter - 8 Channel (TXS0108E) Hookup Guide - SparkFun Learn. [online] Available at: https://learn.sparkfun.com/tutorials/level-shifter---8-channel-txs0108e-hookup-guide/all

Mouse (2022) Level Shifter - 8 channel (TXS0108E) Available at: https://my.mouser.com/new/sparkfun/sparkfun-level-shifter-8-channel-txs0108e/

Make-It.ca. (n.d.). NodeMCU ESP8266 Specifications, Overview and Setting Up. [online] Available at: https://www.make-it.ca/nodemcu-details-specifications/

Manisalidis, I., Stavropoulou, E., Stavropoulos, A. and Bezirtzoglou, E., 2020. Environmental and health impacts of air pollution: a review. Frontiers in public health, 8, p.14.

Ministry of Environment and Water (2020). Annual Environmental Quality Report 2020. Ministry of Environment and Water.

Guilbert, A., Bernard, J.Y., Peyre, H., Costet, N., Hough, I., Seyve, E., Monfort, C., Philippat, C., Slama, R., Kloog, I. and Chevrier, C., 2023. Prenatal and childhood exposure to ambient air pollution and cognitive function in school-age children: Examining sensitive windows and sex-specific associations. *Environmental Research*, *235*, p.116557.

Morgan, Z.E., Bailey, M.J., Trifonova, D.I., Naik, N.C., Patterson, W.B., Lurmann, F.W., Chang, H.H., Peterson, B.S., Goran, M.I. and Alderete, T.L., 2023. Prenatal exposure to ambient air pollution is associated with neurodevelopmental outcomes at 2 years of age. *Environmental Health*, *22*(1), p.11.

Murena, F., 2004. Measuring air quality over large urban areas: development and application of an air pollution index at the urban area of Naples. Atmospheric Environment, 38(36), pp.6195-6202.

Pieter P (2022). A Beginner's Guide to the ESP8266. [online] Github.io. Available at: https://tttapa.github.io/ESP8266/Chap04%20-%20Microcontroller.html

Randomnerdtutorials, (2018). Getting Started with the ESP32 Development Board | Random Nerd Tutorials. [online] Available at: https://randomnerdtutorials.com/getting-started-with-esp32/

Jangid, S. and Sharma, S., 2016, March. An embedded system model for air quality monitoring. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 3003-3008). IEEE.

Sai, K.B.K., Mukherjee, S. and Sultana, H.P., 2019. Low Cost IoT based air quality monitoring setup using arduino and MQ series sensors with dataset analysis. Procedia Computer Science, 165, pp.322-327.

Shiksha. (2020). Available at: https://www.shiksha.com/online-courses/articles/difference-between-c-and-cpp-programming-languages/

Siebenaler, S.P., Janka, A.M., Lyon, D., Edlebeck, J.P. and Nowlan, A.E., 2016, September. Methane detectors challenge: Low-cost continuous emissions monitoring. In International Pipeline Conference (Vol. 50275, p. V003T04A013). American Society of Mechanical Engineers.

Tech Explorations. (n.d.). 1. What is Blynk? [online] Available at: https://techexplorations.com/guides/blynk/1-what-is-blynk/

Teja, R. (2021). ESP32 Pinout and ESP-WROOM-32 Pinout | ESP32 DevKit. [online] Electronics Hub. Available at: https://www.electronicshub.org/esp32-pinout/

Upesy. (2022). ESP32 Pinout: How use GPIO pins ? [online] Available at: https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate-guide

Yang, G., Jiang, M., Ouyang, W., Ji, G., Xie, H., Rahmani, A.M., Liljeberg, P. and Tenhunen, H., 2017. IoT-based remote pain monitoring system: From device to cloud platform. IEEE journal of biomedical and health informatics, 22(6), pp.1711-1719.

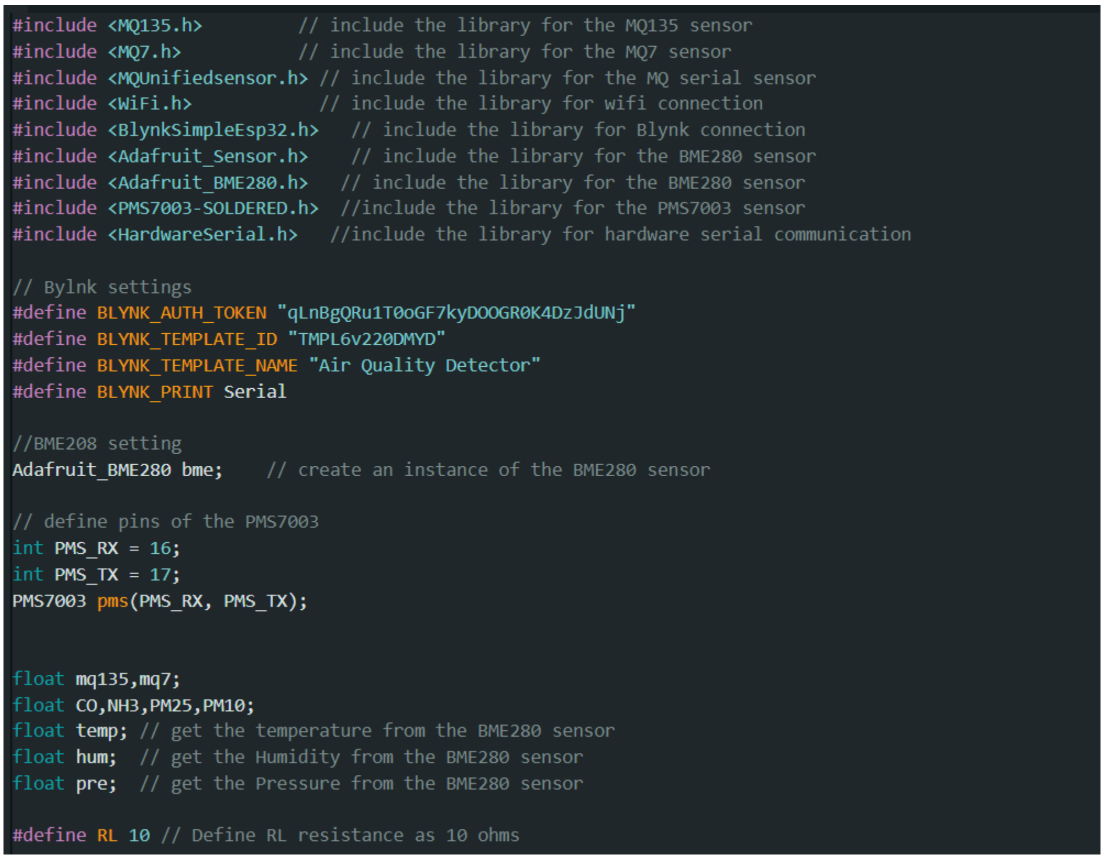
Zhang, R., Wang, G., Guo, S., Zamora, M.L., Ying, Q., Lin, Y., Wang, W., Hu, M. and Wang, Y., 2015. Formation of urban fine particulate matter. *Chemical reviews*, *115*(10), pp.3803-3855.

Zlatanov, N., 2016. Arduino and open source computer hardware and software. J. Water, Sanit. Hyg. Dev, 10(11), pp.1-8.

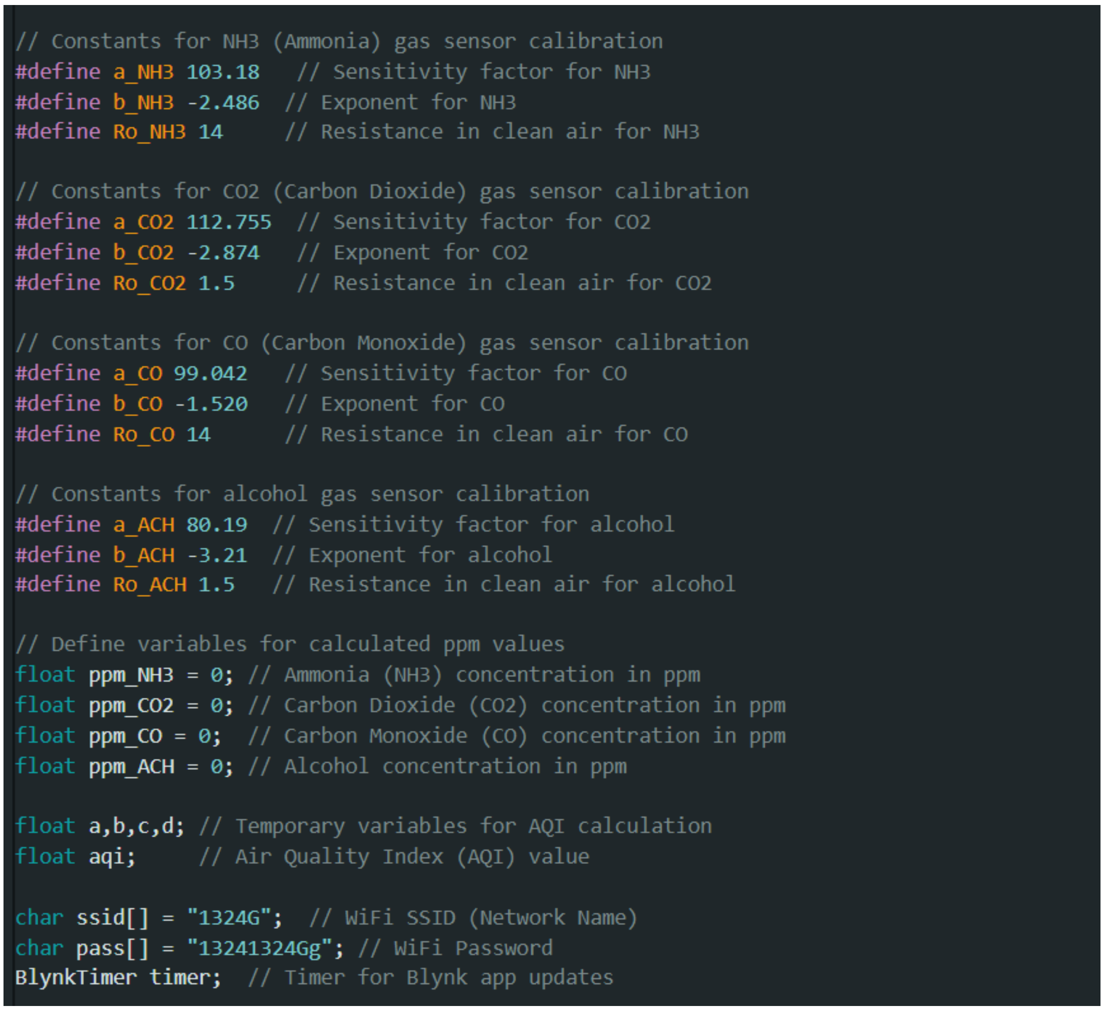
APPENDICES

APPENDIX A: Code

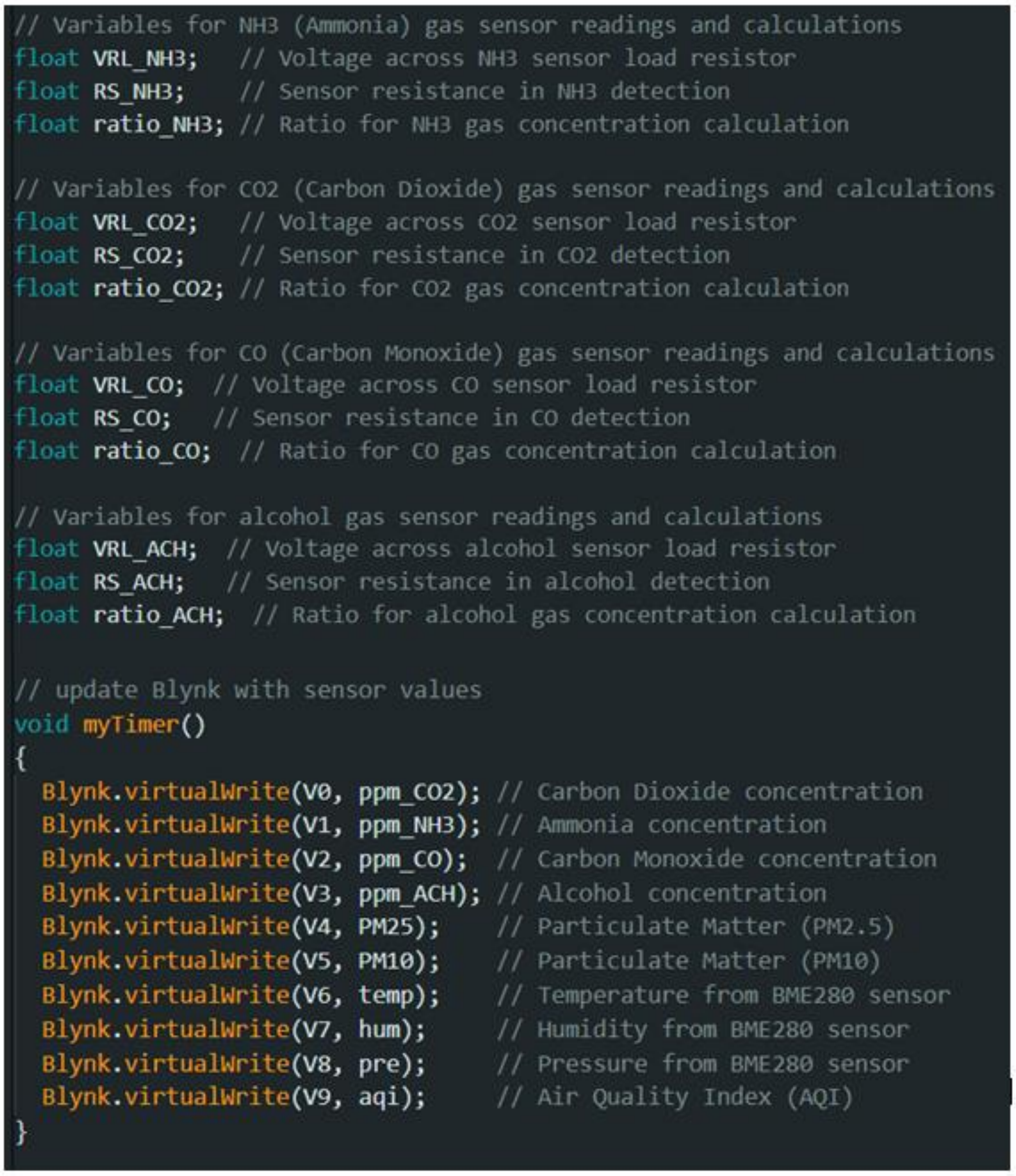
Sensors and ESP32 code



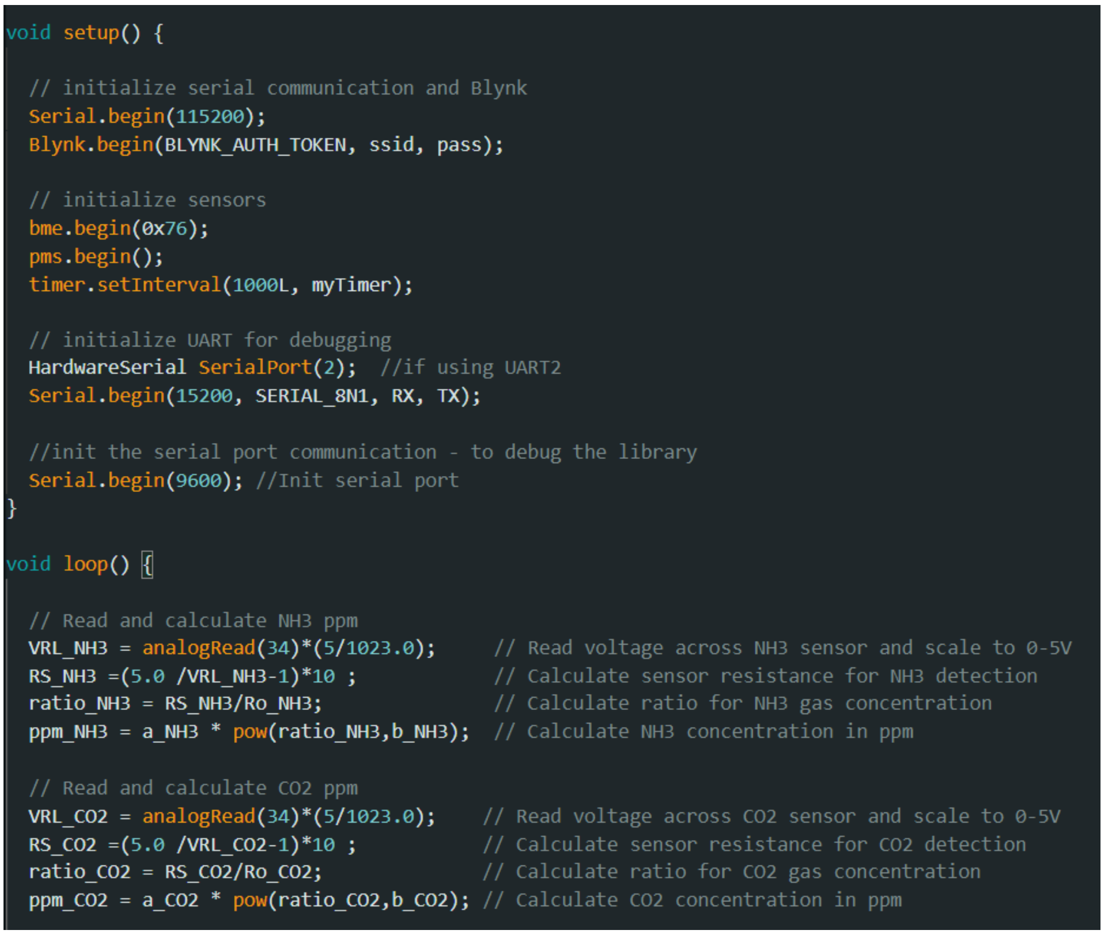
```
#include <MQ135.h>         // include the library for the MQ135 sensor
#include <MQ7.h>           // include the library for the MQ7 sensor
#include <MQUnifiedsensor.h> // include the library for the MQ serial sensor
#include <WiFi.h>            // include the library for wifi connection
#include <BlynkSimpleEsp32.h>   // include the library for Blynk connection
#include <Adafruit_Sensor.h>    // include the library for the BME280 sensor
#include <Adafruit_BME280.h>   // include the library for the BME280 sensor
#include <PMS7003-SOLDERED.h>  //include the library for the PMS7003 sensor
#include <HardwareSerial.h>   //include the library for hardware serial communication

// Bylnk settings
#define BLYNK_AUTH_TOKEN "qLnBgQRu1T0oGF7kyDOOGR0K4DzJdUNj"
#define BLYNK_TEMPLATE_ID "TMPL6v220DMYD"
#define BLYNK_TEMPLATE_NAME "Air Quality Detector"
#define BLYNK_PRINT Serial

//BME208 setting
Adafruit_BME280 bme;    // create an instance of the BME280 sensor

// define pins of the PMS7003
int PMS_RX = 16;
int PMS_TX = 17;
PMS7003 pms(PMS_RX, PMS_TX);


float mq135,mq7;
float CO,NH3,PM25,PM10;
float temp; // get the temperature from the BME280 sensor
float hum;  // get the Humidity from the BME280 sensor
float pre;  // get the Pressure from the BME280 sensor

#define RL 10 // Define RL resistance as 10 ohms
```



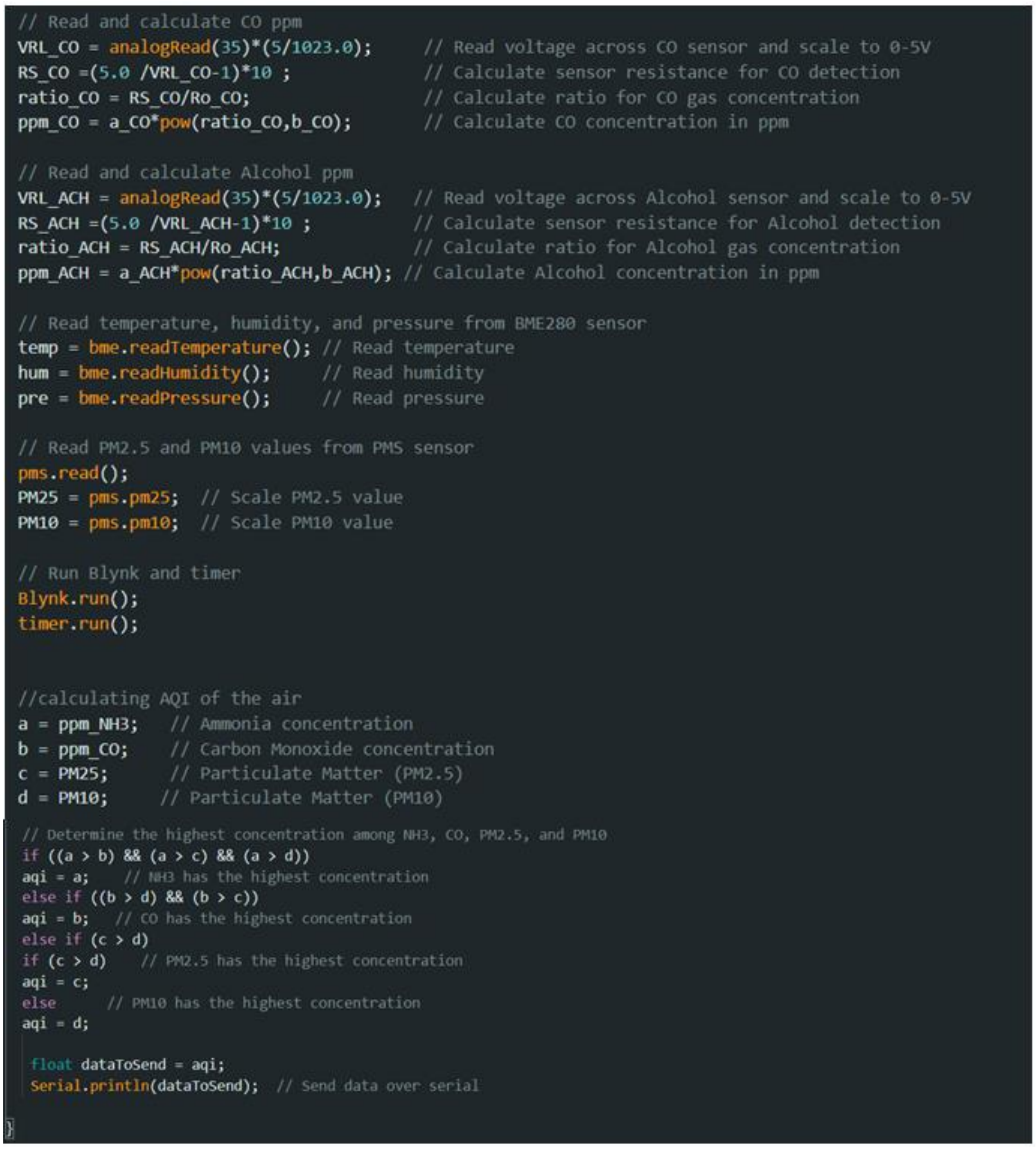
```
// Constants for NH3 (Ammonia) gas sensor calibration
#define a_NH3 103.18   // Sensitivity factor for NH3
#define b_NH3 -2.486  // Exponent for NH3
#define Ro_NH3 14     // Resistance in clean air for NH3

// Constants for CO2 (Carbon Dioxide) gas sensor calibration
#define a_CO2 112.755  // Sensitivity factor for CO2
#define b_CO2 -2.874   // Exponent for CO2
#define Ro_CO2 1.5     // Resistance in clean air for CO2

// Constants for CO (Carbon Monoxide) gas sensor calibration
#define a_CO 99.042   // Sensitivity factor for CO
#define b_CO -1.520   // Exponent for CO
#define Ro_CO 14      // Resistance in clean air for CO

// Constants for alcohol gas sensor calibration
#define a_ACH 80.19  // Sensitivity factor for alcohol
#define b_ACH -3.21  // Exponent for alcohol
#define Ro_ACH 1.5   // Resistance in clean air for alcohol

// Define variables for calculated ppm values
float ppm_NH3 = 0; // Ammonia (NH3) concentration in ppm
float ppm_CO2 = 0; // Carbon Dioxide (CO2) concentration in ppm
float ppm_CO = 0;  // Carbon Monoxide (CO) concentration in ppm
float ppm_ACH = 0; // Alcohol concentration in ppm

float a,b,c,d; // Temporary variables for AQI calculation
float aqi;     // Air Quality Index (AQI) value

char ssid[] = "1324G";  // WiFi SSID (Network Name)
char pass[] = "13241324Gg"; // WiFi Password
BlynkTimer timer;  // Timer for Blynk app updates
```



```
// Variables for NH3 (Ammonia) gas sensor readings and calculations
float VRL_NH3;   // Voltage across NH3 sensor load resistor
float RS_NH3;    // Sensor resistance in NH3 detection
float ratio_NH3; // Ratio for NH3 gas concentration calculation

// Variables for CO2 (Carbon Dioxide) gas sensor readings and calculations
float VRL_CO2;   // Voltage across CO2 sensor load resistor
float RS_CO2;    // Sensor resistance in CO2 detection
float ratio_CO2; // Ratio for CO2 gas concentration calculation

// Variables for CO (Carbon Monoxide) gas sensor readings and calculations
float VRL_CO;  // Voltage across CO sensor load resistor
float RS_CO;   // Sensor resistance in CO detection
float ratio_CO;  // Ratio for CO gas concentration calculation

// Variables for alcohol gas sensor readings and calculations
float VRL_ACH;  // Voltage across alcohol sensor load resistor
float RS_ACH;   // Sensor resistance in alcohol detection
float ratio_ACH;  // Ratio for alcohol gas concentration calculation

// update Blynk with sensor values
void myTimer()
{
  Blynk.virtualWrite(V0, ppm_CO2); // Carbon Dioxide concentration
  Blynk.virtualWrite(V1, ppm_NH3); // Ammonia concentration
  Blynk.virtualWrite(V2, ppm_CO);  // Carbon Monoxide concentration
  Blynk.virtualWrite(V3, ppm_ACH); // Alcohol concentration
  Blynk.virtualWrite(V4, PM25);    // Particulate Matter (PM2.5)
  Blynk.virtualWrite(V5, PM10);    // Particulate Matter (PM10)
  Blynk.virtualWrite(V6, temp);    // Temperature from BME280 sensor
  Blynk.virtualWrite(V7, hum);     // Humidity from BME280 sensor
  Blynk.virtualWrite(V8, pre);     // Pressure from BME280 sensor
  Blynk.virtualWrite(V9, aqi);     // Air Quality Index (AQI)
}
```

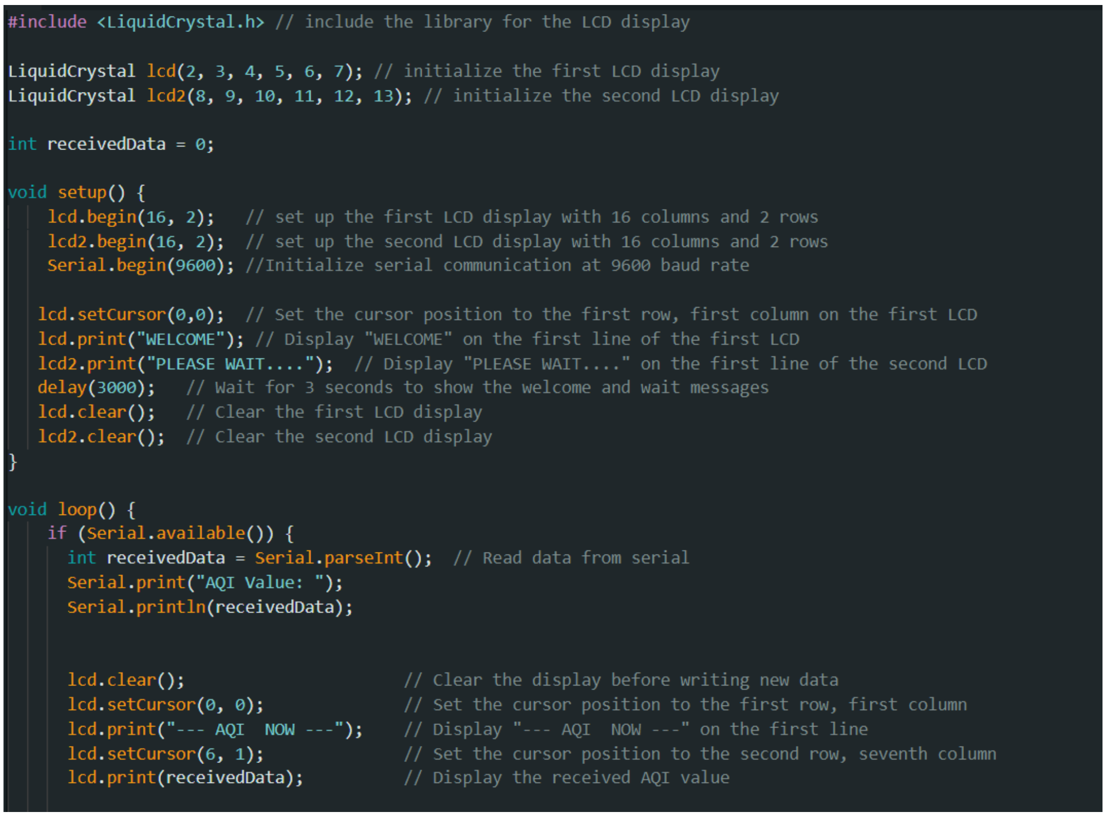


```
void setup() {

  // initialize serial communication and Blynk
  Serial.begin(115200);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

  // initialize sensors
  bme.begin(0x76);
  pms.begin();
  timer.setInterval(1000L, myTimer);

  // initialize UART for debugging
  HardwareSerial SerialPort(2);  //if using UART2
  Serial.begin(15200, SERIAL_8N1, RX, TX);

  //init the serial port communication - to debug the library
  Serial.begin(9600); //Init serial port
}

void loop() {

  // Read and calculate NH3 ppm
  VRL_NH3 = analogRead(34)*(5/1023.0);     // Read voltage across NH3 sensor and scale to 0-5V
  RS_NH3 =(5.0 /VRL_NH3-1)*10 ;            // Calculate sensor resistance for NH3 detection
  ratio_NH3 = RS_NH3/Ro_NH3;               // Calculate ratio for NH3 gas concentration
  ppm_NH3 = a_NH3 * pow(ratio_NH3,b_NH3);  // Calculate NH3 concentration in ppm

  // Read and calculate CO2 ppm
  VRL_CO2 = analogRead(34)*(5/1023.0);    // Read voltage across CO2 sensor and scale to 0-5V
  RS_CO2 =(5.0 /VRL_CO2-1)*10 ;           // Calculate sensor resistance for CO2 detection
  ratio_CO2 = RS_CO2/Ro_CO2;              // Calculate ratio for CO2 gas concentration
  ppm_CO2 = a_CO2 * pow(ratio_CO2,b_CO2); // Calculate CO2 concentration in ppm
```

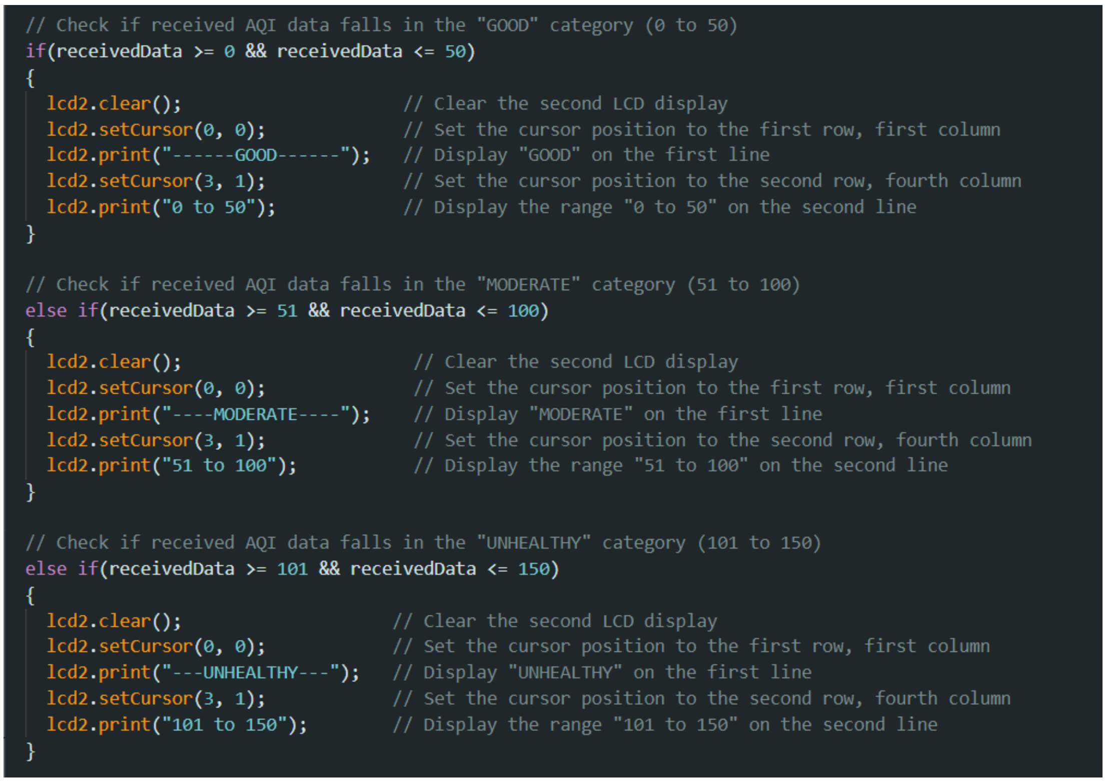


```
// Read and calculate CO ppm
VRL_CO = analogRead(35)*(5/1023.0);      // Read voltage across CO sensor and scale to 0-5V
RS_CO =(5.0 /VRL_CO-1)*10 ;              // Calculate sensor resistance for CO detection
ratio_CO = RS_CO/Ro_CO;                  // Calculate ratio for CO gas concentration
ppm_CO = a_CO*pow(ratio_CO,b_CO);        // Calculate CO concentration in ppm

// Read and calculate Alcohol ppm
VRL_ACH = analogRead(35)*(5/1023.0);    // Read voltage across Alcohol sensor and scale to 0-5V
RS_ACH =(5.0 /VRL_ACH-1)*10 ;           // Calculate sensor resistance for Alcohol detection
ratio_ACH = RS_ACH/Ro_ACH;              // Calculate ratio for Alcohol gas concentration
ppm_ACH = a_ACH*pow(ratio_ACH,b_ACH); // Calculate Alcohol concentration in ppm

// Read temperature, humidity, and pressure from BME280 sensor
temp = bme.readTemperature(); // Read temperature
hum = bme.readHumidity();      // Read humidity
pre = bme.readPressure();      // Read pressure

// Read PM2.5 and PM10 values from PMS sensor
pms.read();
PM25 = pms.pm25;  // Scale PM2.5 value
PM10 = pms.pm10;  // Scale PM10 value

// Run Blynk and timer
Blynk.run();
timer.run();


//calculating AQI of the air
a = ppm_NH3;   // Ammonia concentration
b = ppm_CO;    // Carbon Monoxide concentration
c = PM25;      // Particulate Matter (PM2.5)
d = PM10;     // Particulate Matter (PM10)

// Determine the highest concentration among NH3, CO, PM2.5, and PM10
if ((a > b) && (a > c) && (a > d))
aqi = a;    // NH3 has the highest concentration
else if ((b > d) && (b > c))
aqi = b;   // CO has the highest concentration
else if (c > d)
if (c > d)    // PM2.5 has the highest concentration
aqi = c;
else      // PM10 has the highest concentration
aqi = d;

 float dataToSend = aqi;
 Serial.println(dataToSend);  // Send data over serial

}
```

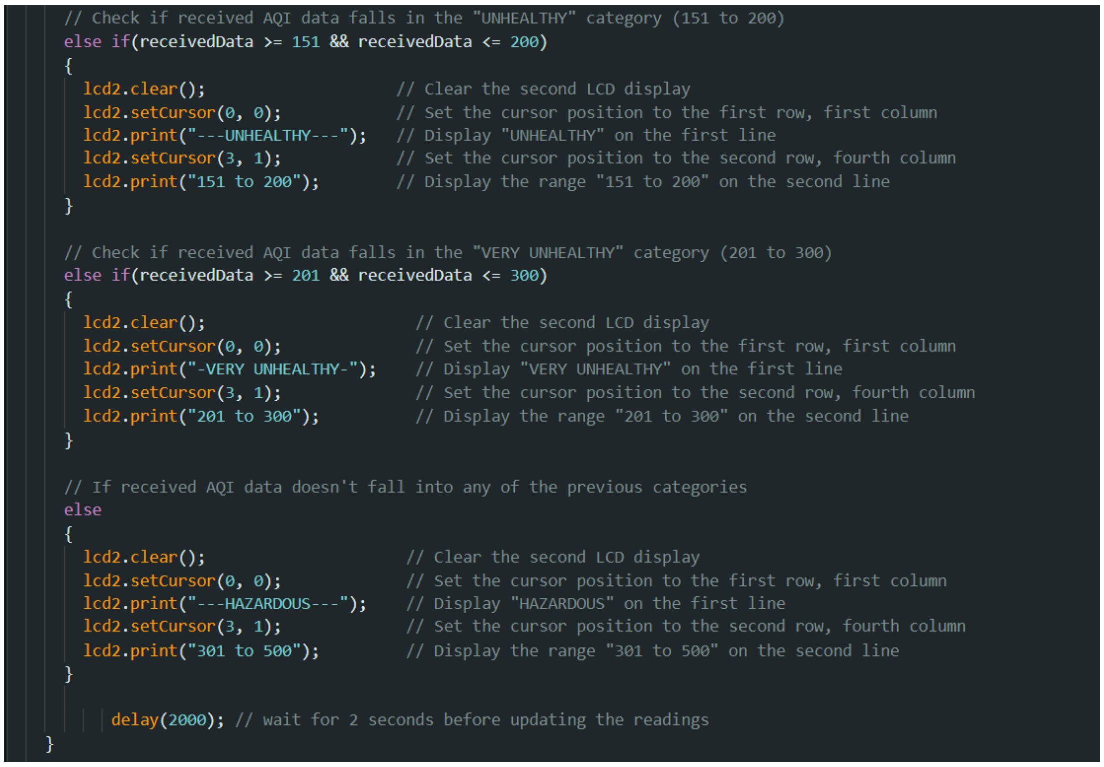
Arduino UNO and LCD code



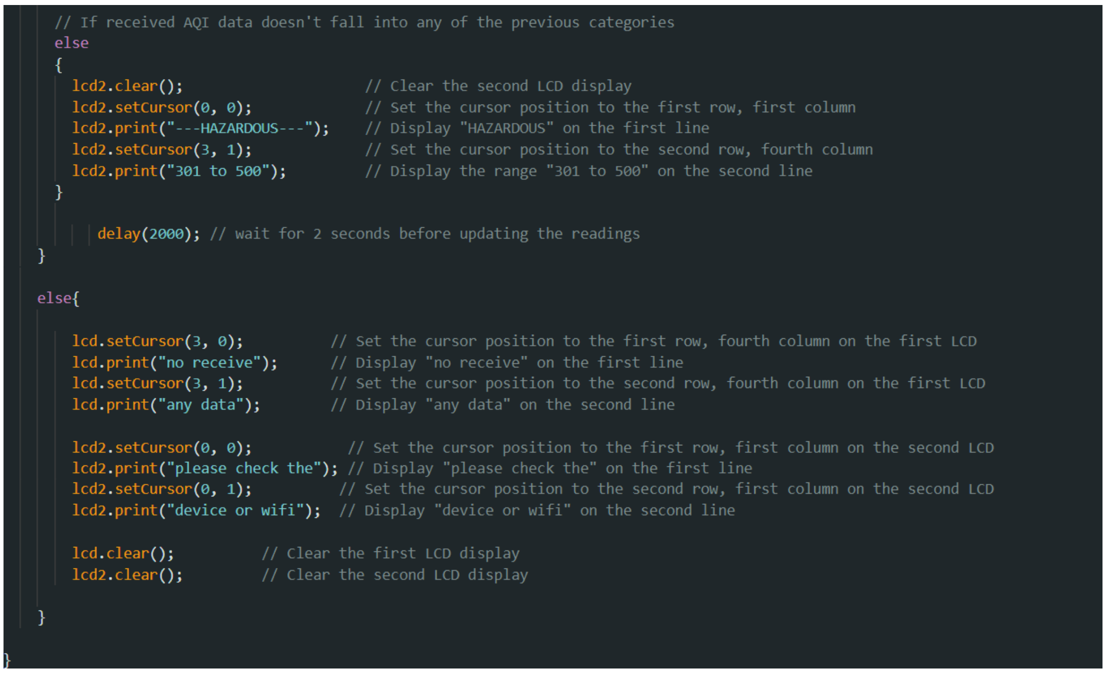
```
#include <LiquidCrystal.h> // include the library for the LCD display

LiquidCrystal lcd(2, 3, 4, 5, 6, 7); // initialize the first LCD display
LiquidCrystal lcd2(8, 9, 10, 11, 12, 13); // initialize the second LCD display

int receivedData = 0;

void setup() {
    lcd.begin(16, 2);   // set up the first LCD display with 16 columns and 2 rows
    lcd2.begin(16, 2);  // set up the second LCD display with 16 columns and 2 rows
    Serial.begin(9600); //Initialize serial communication at 9600 baud rate

   lcd.setCursor(0,0);  // Set the cursor position to the first row, first column on the first LCD
   lcd.print("WELCOME"); // Display "WELCOME" on the first line of the first LCD
   lcd2.print("PLEASE WAIT....");  // Display "PLEASE WAIT...." on the first line of the second LCD
   delay(3000);   // Wait for 3 seconds to show the welcome and wait messages
   lcd.clear();   // Clear the first LCD display
   lcd2.clear();  // Clear the second LCD display
}

void loop() {
    if (Serial.available()) {
      int receivedData = Serial.parseInt();  // Read data from serial
      Serial.print("AQI Value: ");
      Serial.println(receivedData);


      lcd.clear();                       // Clear the display before writing new data
      lcd.setCursor(0, 0);               // Set the cursor position to the first row, first column
      lcd.print("--- AQI  NOW ---");     // Display "--- AQI  NOW ---" on the first line
      lcd.setCursor(6, 1);               // Set the cursor position to the second row, seventh column
      lcd.print(receivedData);           // Display the received AQI value
```



```
// Check if received AQI data falls in the "GOOD" category (0 to 50)
if(receivedData >= 0 && receivedData <= 50)
{
  lcd2.clear();                    // Clear the second LCD display
  lcd2.setCursor(0, 0);            // Set the cursor position to the first row, first column
  lcd2.print("------GOOD------");  // Display "GOOD" on the first line
  lcd2.setCursor(3, 1);            // Set the cursor position to the second row, fourth column
  lcd2.print("0 to 50");           // Display the range "0 to 50" on the second line
}

// Check if received AQI data falls in the "MODERATE" category (51 to 100)
else if(receivedData >= 51 && receivedData <= 100)
{
  lcd2.clear();                     // Clear the second LCD display
  lcd2.setCursor(0, 0);             // Set the cursor position to the first row, first column
  lcd2.print("----MODERATE----");   // Display "MODERATE" on the first line
  lcd2.setCursor(3, 1);             // Set the cursor position to the second row, fourth column
  lcd2.print("51 to 100");          // Display the range "51 to 100" on the second line
}

// Check if received AQI data falls in the "UNHEALTHY" category (101 to 150)
else if(receivedData >= 101 && receivedData <= 150)
{
  lcd2.clear();                   // Clear the second LCD display
  lcd2.setCursor(0, 0);           // Set the cursor position to the first row, first column
  lcd2.print("---UNHEALTHY---");  // Display "UNHEALTHY" on the first line
  lcd2.setCursor(3, 1);           // Set the cursor position to the second row, fourth column
  lcd2.print("101 to 150");       // Display the range "101 to 150" on the second line
}
```

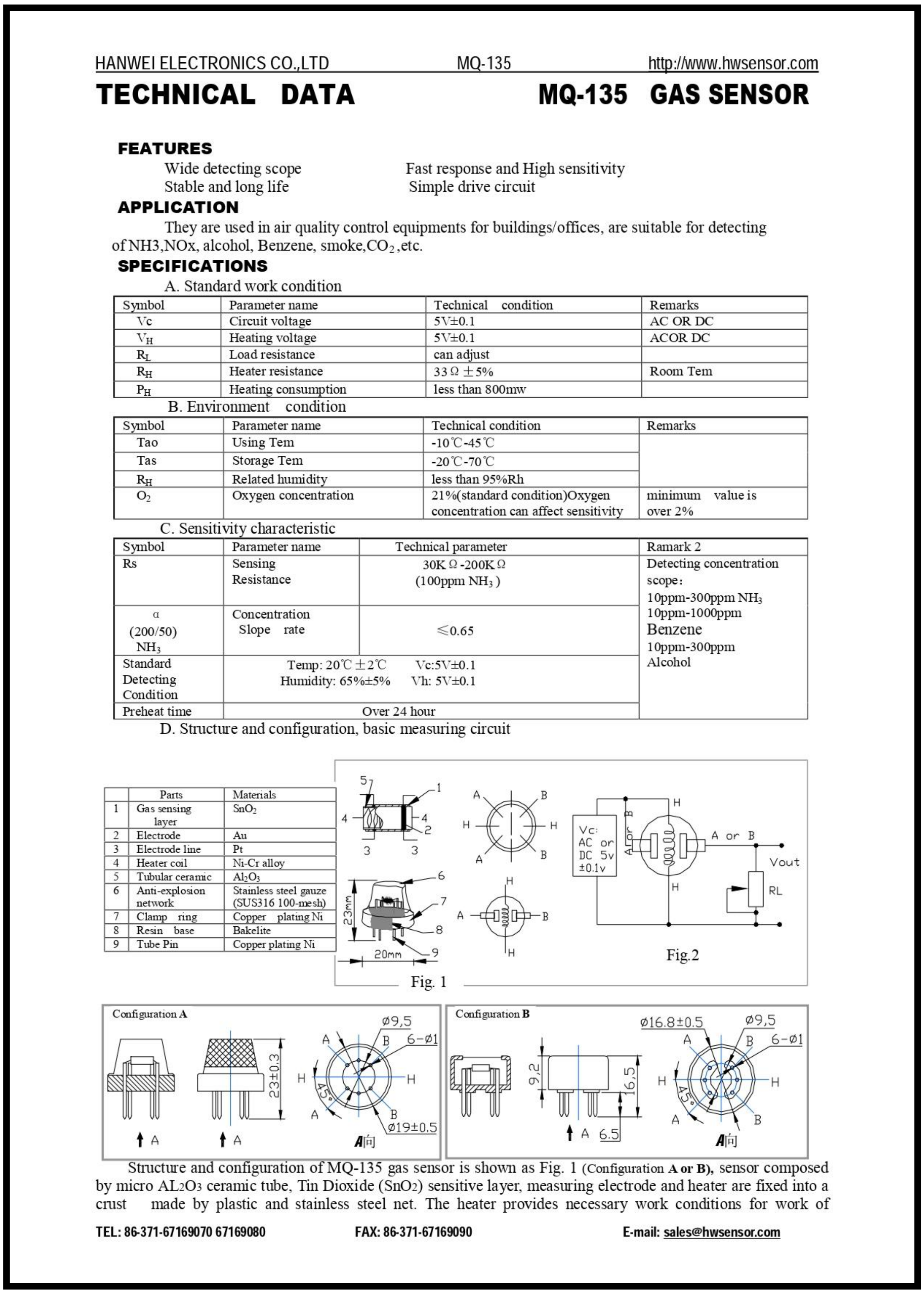


```
  // Check if received AQI data falls in the "UNHEALTHY" category (151 to 200)
  else if(receivedData >= 151 && receivedData <= 200)
  {
    lcd2.clear();                    // Clear the second LCD display
    lcd2.setCursor(0, 0);            // Set the cursor position to the first row, first column
    lcd2.print("---UNHEALTHY---");   // Display "UNHEALTHY" on the first line
    lcd2.setCursor(3, 1);            // Set the cursor position to the second row, fourth column
    lcd2.print("151 to 200");        // Display the range "151 to 200" on the second line
  }

  // Check if received AQI data falls in the "VERY UNHEALTHY" category (201 to 300)
  else if(receivedData >= 201 && receivedData <= 300)
  {
    lcd2.clear();                      // Clear the second LCD display
    lcd2.setCursor(0, 0);              // Set the cursor position to the first row, first column
    lcd2.print("-VERY UNHEALTHY-");    // Display "VERY UNHEALTHY" on the first line
    lcd2.setCursor(3, 1);              // Set the cursor position to the second row, fourth column
    lcd2.print("201 to 300");          // Display the range "201 to 300" on the second line
  }

  // If received AQI data doesn't fall into any of the previous categories
  else
  {
    lcd2.clear();                     // Clear the second LCD display
    lcd2.setCursor(0, 0);             // Set the cursor position to the first row, first column
    lcd2.print("---HAZARDOUS---");    // Display "HAZARDOUS" on the first line
    lcd2.setCursor(3, 1);             // Set the cursor position to the second row, fourth column
    lcd2.print("301 to 500");         // Display the range "301 to 500" on the second line
  }

      delay(2000); // wait for 2 seconds before updating the readings
}
```



```
    // If received AQI data doesn't fall into any of the previous categories
    else
    {
      lcd2.clear();                       // Clear the second LCD display
      lcd2.setCursor(0, 0);               // Set the cursor position to the first row, first column
      lcd2.print("---HAZARDOUS---");      // Display "HAZARDOUS" on the first line
      lcd2.setCursor(3, 1);               // Set the cursor position to the second row, fourth column
      lcd2.print("301 to 500");           // Display the range "301 to 500" on the second line
    }

        delay(2000); // wait for 2 seconds before updating the readings
  }

  else{

      lcd.setCursor(3, 0);          // Set the cursor position to the first row, fourth column on the first LCD
      lcd.print("no receive");      // Display "no receive" on the first line
      lcd.setCursor(3, 1);          // Set the cursor position to the second row, fourth column on the first LCD
      lcd.print("any data");        // Display "any data" on the second line

      lcd2.setCursor(0, 0);           // Set the cursor position to the first row, first column on the second LCD
      lcd2.print("please check the"); // Display "please check the" on the first line
      lcd2.setCursor(0, 1);          // Set the cursor position to the second row, first column on the second LCD
      lcd2.print("device or wifi");  // Display "device or wifi" on the second line

      lcd.clear();         // Clear the first LCD display
      lcd2.clear();        // Clear the second LCD display

  }

}
```

APPENDIX B: MQ135 data sheet



TECHNICAL DATA MQ-135 GAS SENSOR

FEATURES

Wide detecting scope	Fast response and High sensitivity
Stable and long life	Simple drive circuit

APPLICATION

They are used in air quality control equipments for buildings/offices, are suitable for detecting of NH3,NOx, alcohol, Benzene, smoke,CO_2 ,etc.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating voltage	5V±0.1	ACOR DC
R_L	Load resistance	can adjust	
R_H	Heater resistance	33Ω ±5%	Room Tem
P_H	Heating consumption	less than 800mw	

B. Environment condition

Symbol	Parameter name	Technical condition	Remarks
Tao	Using Tem	-10℃-45℃	
Tas	Storage Tem	-20℃-70℃	
R_H	Related humidity	less than 95%Rh	
O_2	Oxygen concentration	21%(standard condition)Oxygen concentration can affect sensitivity	minimum value is over 2%

C. Sensitivity characteristic

Symbol	Parameter name	Technical parameter	Ramark 2
Rs	Sensing Resistance	30KΩ-200KΩ (100ppm NH_3)	Detecting concentration scope: 10ppm-300ppm NH_3 10ppm-1000ppm Benzene 10ppm-300ppm Alcohol
α (200/50) NH_3	Concentration Slope rate	≤0.65	
Standard Detecting Condition	Temp: 20℃ ±2℃ Vc:5V±0.1 Humidity: 65%±5% Vh: 5V±0.1		
Preheat time	Over 24 hour		

D. Structure and configuration, basic measuring circuit

	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

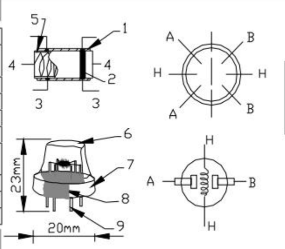


Fig. 1

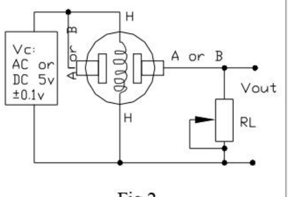
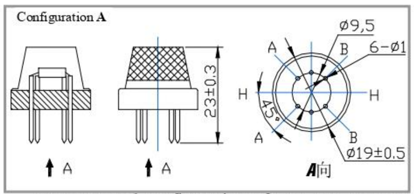
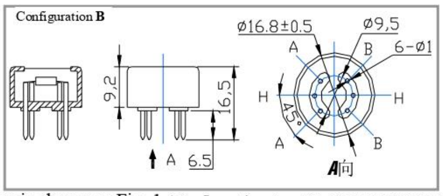


Fig.2





Structure and configuration of MQ-135 gas sensor is shown as Fig. 1 (Configuration **A or B**), sensor composed by micro AL_2O_3 ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-135 have 6 pin ,4 of them are used to fetch signals, and other 2 are used for providing heating current.

Electric parameter measurement circuit is shown as Fig.2

E. Sensitivity characteristic curve

Fig.2 sensitivity characteristics of the MQ-135

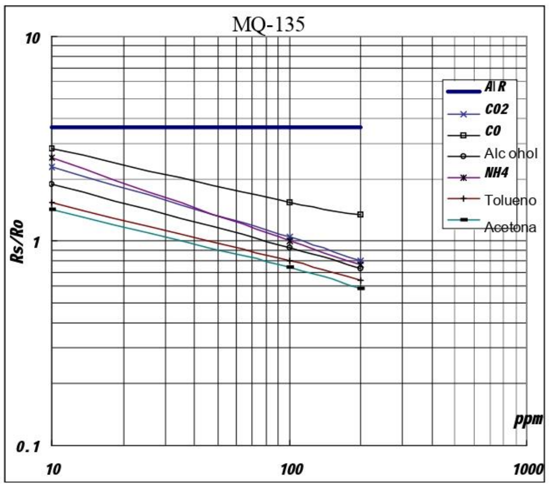


Fig.3 is shows the typical sensitivity characteristics of the MQ-135 for several gases.

in their: Temp: 20℃、

Humidity: 65%、

O_2 concentration 21%

RL=20kΩ

Ro: sensor resistance at 100ppm of NH_3 in the clean air.

Rs:sensor resistance at various concentrations of gases.

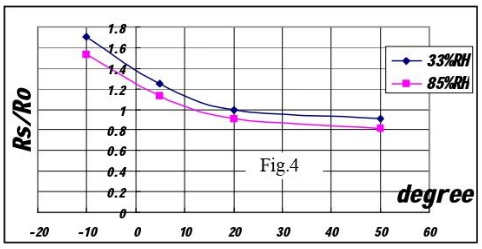


Fig.4 is shows the typical dependence of the MQ-135 on temperature and humidity.

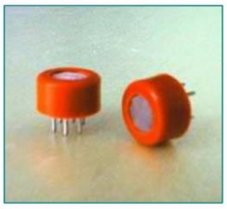
Ro: sensor resistance at 100ppm of NH_3 in air at 33%RH and 20 degree.

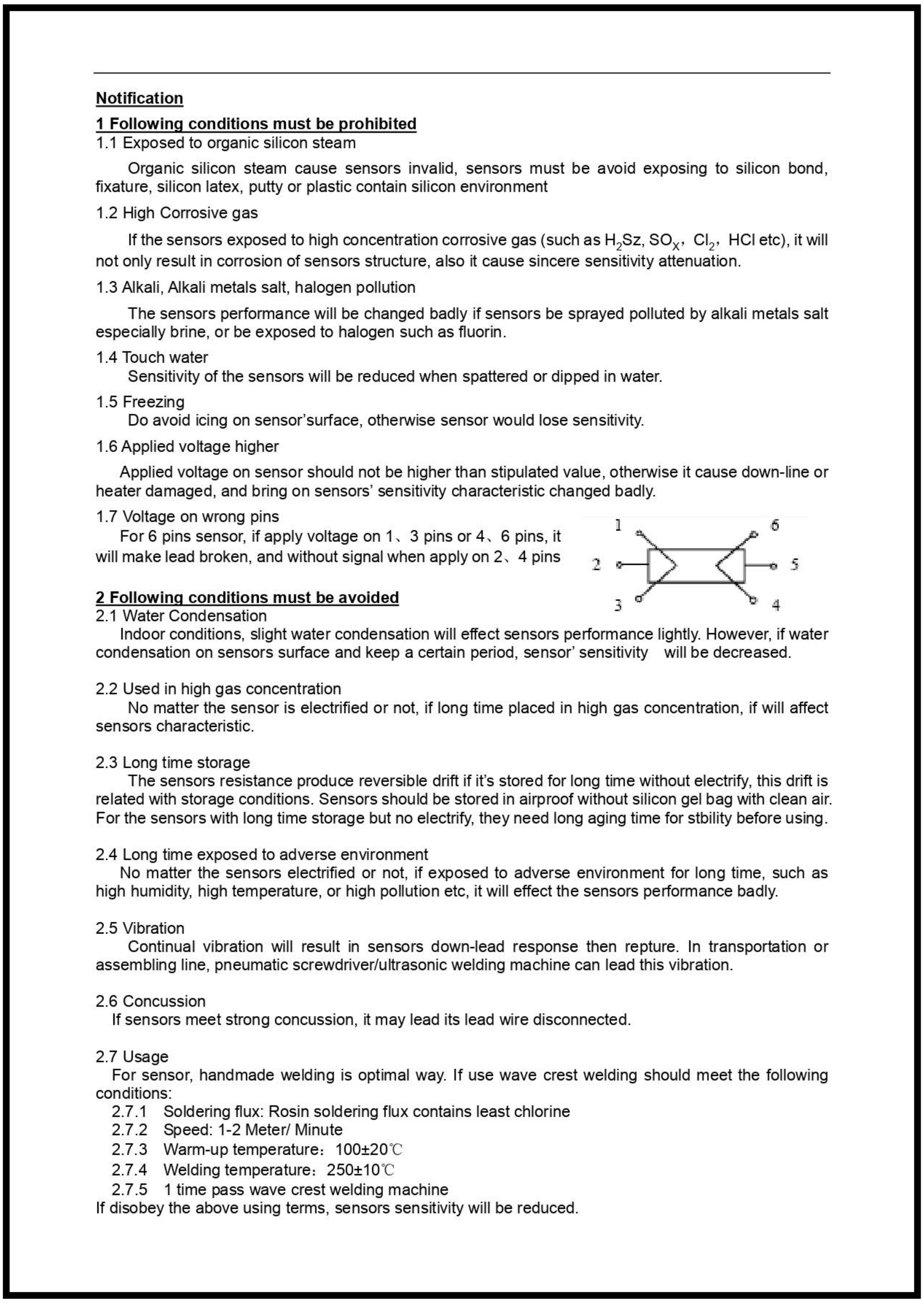
Rs: sensor resistance at 100ppm of NH_3 at different temperatures and humidities.

SENSITIVITY ADJUSTMENT

Resistance value of MQ-135 is difference to various kinds and various concentration gases. So,When using this components, sensitivity adjustment is very necessary. we recommend that you calibrate the detector for 100ppm NH_3 or 50ppm Alcohol concentration in air and use value of Load resistancethat(R_L) about 20 KΩ (10KΩ to 47 KΩ).

When accurately measuring, the proper alarm point for the gas detector should be determined after considering the temperature and humidity influence.





Notification

1 Following conditions must be prohibited

1.1 Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixature, silicon latex, putty or plastic contain silicon environment

1.2 High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H_2Sz, SO_X, Cl_2, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

1.3 Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

1.4 Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

1.5 Freezing

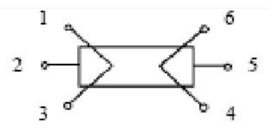
Do avoid icing on sensor'surface, otherwise sensor would lose sensitivity.

1.6 Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors' sensitivity characteristic changed badly.

1.7 Voltage on wrong pins

For 6 pins sensor, if apply voltage on 1、3 pins or 4、6 pins, it will make lead broken, and without signal when apply on 2、4 pins



2 Following conditions must be avoided

2.1 Water Condensation

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor' sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, if will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it's stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stbility before using.

2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

2.5 Vibration

Continual vibration will result in sensors down-lead response then repture. In transportation or assembling line, pneumatic screwdriver/ultrasonic welding machine can lead this vibration.

2.6 Concussion

If sensors meet strong concussion, it may lead its lead wire disconnected.

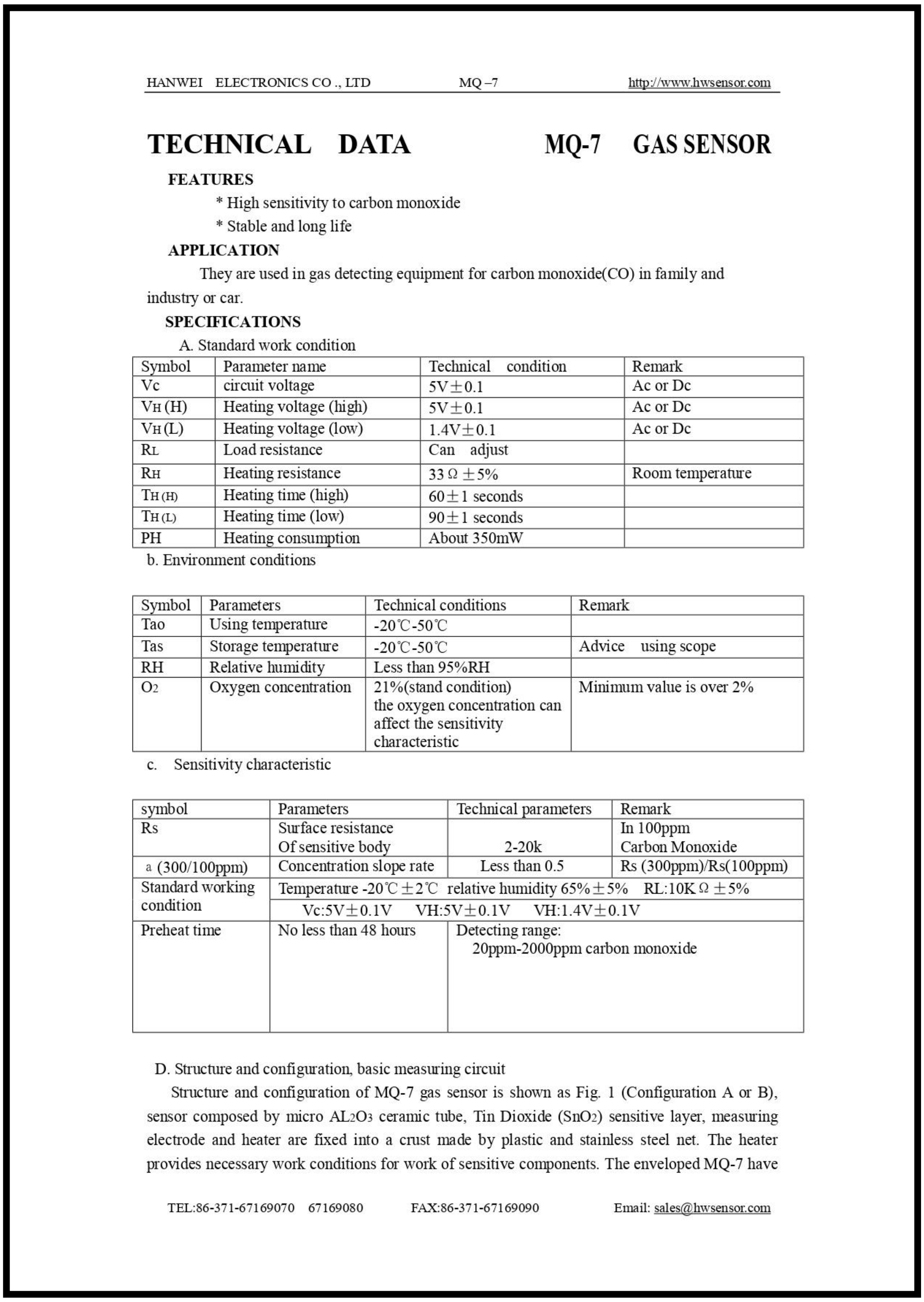
2.7 Usage

For sensor, handmade welding is optimal way. If use wave crest welding should meet the following conditions:

2.7.1 Soldering flux: Rosin soldering flux contains least chlorine
2.7.2 Speed: 1-2 Meter/ Minute
2.7.3 Warm-up temperature：100±20℃
2.7.4 Welding temperature：250±10℃
2.7.5 1 time pass wave crest welding machine

If disobey the above using terms, sensors sensitivity will be reduced.

APPENDIX C: MQ7 data sheet



TECHNICAL DATA MQ-7 GAS SENSOR

FEATURES

* High sensitivity to carbon monoxide

* Stable and long life

APPLICATION

They are used in gas detecting equipment for carbon monoxide(CO) in family and industry or car.

SPECIFICATIONS

A. Standard work condition

Symbol	Parameter name	Technical condition	Remark
Vc	circuit voltage	5V±0.1	Ac or Dc
V_H (H)	Heating voltage (high)	5V±0.1	Ac or Dc
V_H (L)	Heating voltage (low)	1.4V±0.1	Ac or Dc
R_L	Load resistance	Can adjust	
R_H	Heating resistance	33 Ω ±5%	Room temperature
$T_{H (H)}$	Heating time (high)	60±1 seconds	
$T_{H (L)}$	Heating time (low)	90±1 seconds	
PH	Heating consumption	About 350mW	

b. Environment conditions

Symbol	Parameters	Technical conditions	Remark
Tao	Using temperature	-20℃-50℃	
Tas	Storage temperature	-20℃-50℃	Advice using scope
RH	Relative humidity	Less than 95%RH	
O_2	Oxygen concentration	21%(stand condition) the oxygen concentration can affect the sensitivity characteristic	Minimum value is over 2%

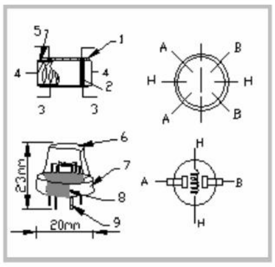
c. Sensitivity characteristic

symbol	Parameters	Technical parameters	Remark
Rs	Surface resistance Of sensitive body	2-20k	In 100ppm Carbon Monoxide
α (300/100ppm)	Concentration slope rate	Less than 0.5	Rs (300ppm)/Rs(100ppm)
Standard working condition	Temperature -20℃±2℃ relative humidity 65%±5% RL:10KΩ±5%		
	Vc:5V±0.1V VH:5V±0.1V VH:1.4V±0.1V		
Preheat time	No less than 48 hours	Detecting range: 20ppm-2000ppm carbon monoxide	

D. Structure and configuration, basic measuring circuit

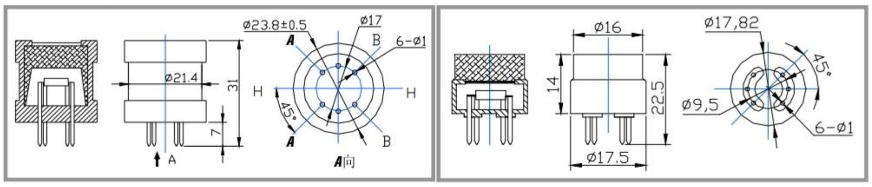
Structure and configuration of MQ-7 gas sensor is shown as Fig. 1 (Configuration A or B), sensor composed by micro AL_2O_3 ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-7 have

6 pin ,4 of them are used to fetch signals, and other 2 are used for providing heating current.



	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

Fig.1



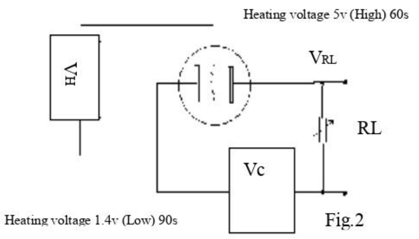


Fig.2

Standard circuit:

As shown in Fig 2, standard measuring circuit of MQ-7 sensitive components consists of 2 parts. one is heating circuit having time control function (the high voltage and the low voltage work circularly). The second is the signal output circuit, it can accurately respond changes of surface resistance of the sensor.

Electric parameter measurement circuit is shown as Fig.2

E. Sensitivity characteristic curve

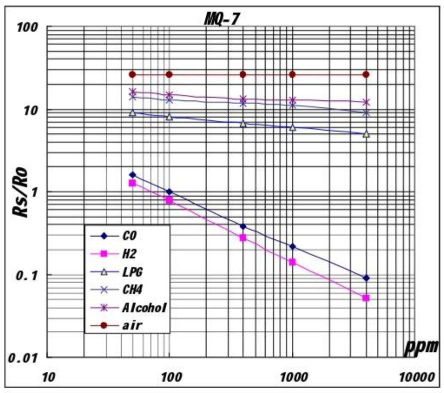


Fig.3 is shows the typical sensitivity characteristics of the MQ-7 for several gases.

in their: Temp: 20℃、

Humidity: 65%、

O_2 concentration 21%

RL=10k Ω

Ro: sensor resistance at 100ppm CO in the clean air.

Rs: sensor resistance at various concentrations of gases.

Fig.3 sensitivity characteristics of the MQ-7

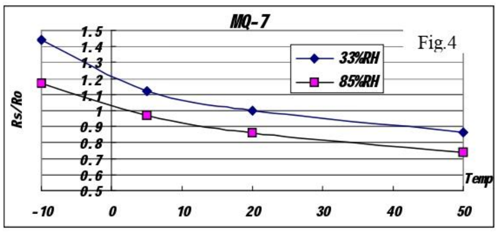


Fig.4

Fig.4 is shows the typical dependence of the MQ-7 on temperature and humidity.
Ro: sensor resistance at 100ppm CO in air at 33%RH and 20degree.
Rs: sensor resistance at 100ppm CO at different temperatures and humidities.

OPERATION PRINCIPLE

. The surface resistance of the sensor Rs is obtained through effected voltage signal output of the load resistance RL which series-wound. The relationship between them is described:

Rs\RL = (Vc-VRL) / VRL

Fig. 5 shows alterable situation of RL signal output measured by using Fig. 2 circuit output

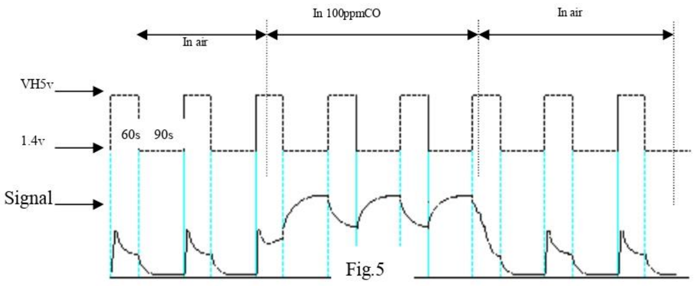


Fig.5

signal when the sensor is shifted from clean air to carbon monoxide (CO) , output signal measurement is made within one or two complete heating period (2.5 minute from high voltage to low voltage).

Sensitive layer of MQ-7 gas sensitive components is made of SnO_2 with stability, So, it has excellent long term stability. Its service life can reach 5 years under using condition.

SENSITVITY ADJUSTMENT

Resistance value of MQ-7 is difference to various kinds and various concentration gases. So, When using this components, sensitivity adjustment is very necessary. we recommend that you calibrate the detector for 200ppm CO in air and use value of Load resistance that(R_L) about 10 KΩ (5KΩ to 47 KΩ).

When accurately measuring, the proper alarm point for the gas detector should be determined after considering the temperature and humidity influence. The sensitivity adjusting program:

a. Connect the sensor to the application circuit.

b. Turn on the power, keep preheating through electricity over 48 hours.

c. Adjust the load resistance RL until you get a signal value which is respond to a certain carbon monoxide concentration at the end point of 90 seconds.

d. Adjust the another load resistance RL until you get a signal value which is respond to a CO concentration at the end point of 60 seconds .