



**DEPARTMENT OF COMPUTER
& SOFTWARE ENGINEERING
COLLEGE OF E&ME, NUST,
RAWALPINDI**



**Air Quality Monitoring System Using LoRaWAN and Helium
Network**

**A PROJECT REPORT
DEGREE 41 (DC&SE)**

Submitted By:

NS Sana Iftikhar

PC Amna Arif

PC Khadija Tul Kubra

PC Amanullah Naseer

**BACHELORS
IN
COMPUTER ENGINEERING
YEAR 2023**

PROJECT SUPERVISOR

Dr Sajid Gul Khwaja

Table of Contents

Table of Contents	2
Table of Figures	5
DECLARATION	7
COPYRIGHT STATEMENT	8
ACKNOWLEDGEMENTS	9
ABSTRACT	10
Chapter 1: Introduction	12
Overview	12
1.1 Scope	13
1.2 Objectives	13
1.3 Methodology	13
1.4 Structure	14
Chapter 02: Literature Review	17
2.1 Main Pollutants	18
2.2 International Standard Air Quality Index	18
2.6 Background & Related Work	19
2.7 Sampling and Measurement Techniques	23
Chapter 03: System Design and Architecture	26
3.1 System Architecture	26
3.2 Hardware Components	28
3.2.1 Sensors:	28
3.2.2 Optical Dust Sensor - Gp2y1010au0f	28
3.2.3 MQ-135 Sensor:	28
3.2.4 Data Acquisition System:	29
3.2.5 LoRa-WAN Module STM-32:	29
3.2.6 Helium Gateway Raspberry Pi:	29
3.2.7 Working together:	29
3.2.8 Communication modules:	30
3.2.9 Processing Units:	30
3.2.10 Storage:	30
3.2.11 Power Supply:	30
3.2.12 Interface:	30
3.3 Software Components	31
3.3.1 Application Framework	31
3.3.2 User Interface (UI)	31

3.3.3	Networking and Communication:	32
3.3.4	Data Visualization:	32
3.3.5	Data Storage:	33
3.3.6	Integration and APIs:.....	34
3.4	Design Decisions.....	34
3.4.1	Sensor Selection	35
3.4.2	Sampling Methodology	35
3.4.3	Data Acquisition System	36
3.4.4	Data Processing and Analysis	36
3.4.5	Data Visualization and User Interface.....	36
3.4.6	Power Supply and Energy Efficiency	37
Chapter 04:	ALGORITHM DESIGN AND DEVELOPMENT	39
4.1	Working:	39
4.1.1	Arduino:	39
4.1.2	Circuit with Arduino:	41
4.1.3	STM32 LORAWAN:	41
4.1.4	Circuit with STM32 LORAWAN:	42
4.1.5	Helium Gateway:.....	42
4.1.6	AWS:.....	43
4.1.7	Mobile App:	44
4.2	Algorithm:	45
4.3	Flowchart of code and output:.....	46
4.4	Flow Chart of Code:	46
4.5	Graph:.....	47
Chapter 05:	AWS and HELIUM INTEGRATION	50
5.1	Creating AWS Account.....	50
5.2	DynamoDB Creation.....	51
5.3	HTTP Integration	53
5.3.1	Set up an HTTP Endpoint	54
5.3.2	Connect STM32 to an HTTP Endpoint	55
5.3.3	Store AQI data in DynamoDB table.....	56
5.4	Lambda Function.....	56
5.4.1	Addition of Trigger Events.....	58
5.5	API Gateways.....	58
Chapter 06:	Software Application.....	61
6.1	Overview	61
6.2	Flutter Application	62

Chapter 7: Conclusion and Future Work.....	67
7.1 Conclusion.....	67
7.2 Future Work	67
REFERENCES	69

Table of Figures

Figure 1:AQM System Level Diagram	14
Figure 2: Effects of Air Pollution.....	18
Figure 3: Main Pollution	18
Figure 4: AQI Meter.....	19
Figure 5 :Aeroqual 500 Handheld Monitor Base	20
Figure 6: IQAir Map.....	20
Figure 7: Map of Rawalpindi in IQAir Web App	21
Figure 8: PurpleAir Flex Quality Monitor Device	21
Figure 9: List of Pollutants measured by Aclima.....	22
Figure 10 :System Level Diagram.....	26
Figure 11 :Architecture Level Diagram	27
Figure 12 : Flow Diagram	28
Figure 13: Code of colored assigned according to AQI ranges.....	33
Figure 14: Home pages with different AQI showing the color assigned within the ranges.	33
Figure 15: AWS Website.....	34
Figure 16:Wise Node PCB Layout.....	35
Figure 17 :AQI values calculated using Arduino.....	36
Figure 18 : Search and Air Quality page	37
Figure 19 :MQ-135 sensor.....	39
Figure 20 :GP2Y1010AU	39
Figure 21: Datasheet of MQ-135.....	40
Figure 22: Datasheet of GP2Y1010AU0F	40
Figure 23: Circuit with Arduino	41
Figure 24: Circuit with stm-32 LoRaWAN.....	42
Figure 25: Airify App.....	45
Figure 26: Output of Arduino Code	46
Figure 27 :Flow Diagram to show data on Application.	47
Figure 28 :Graphic Display of MQ-135	47
Figure 29:Graphic Display of AQI with respect to time	48
Figure 30: AWS Login Console	50
Figure 31: Signing into the AWS Console as a root user.....	51
Figure 32: Creating table.....	51
Figure 33: Table Info.....	52
Figure 34: Table AQI created.....	52
Figure 35: Overview of table.....	53
Figure 36: Table Items.....	53
Figure 37: Generating API keys.....	54
Figure 38: Copy Assigned API key.....	54
Figure 39: AQI end node added on Helium Console	55
Figure 40: AQI end node details	55
Figure 41: AQI HTTP integration	56
Figure 42: Helium Flows.....	56
Figure 43: Table items.....	56

Figure 44: Lambda Function created for AQI value storage.....	57
Figure 45: AQI Lambda function overview	58
Figure 46: API Gateway connected to Lambda.	58
Figure 47: Gateway having Helium API with API endpoint.	59
Figure 48: API Gateway Helium API	59
Figure 49: Project Class Diagram.....	61
Figure 50: Search of Application	62
Figure 51: Home of Application.....	62
Figure 52: Air Quality, Settings, and Notifications of Application	63
Figure 53: Privacy policy, Terms of Use and Profile pages of Application.....	63
Figure 54: FAQ, Feedback and Share app pages of Application	64
Figure 55: FAQ pages of Application	64
Figure 56: User Flow Diagram.....	65
Figure 57: Use case diagram	65

DECLARATION

We herewith declare that no portion of the work stated during this Project Thesis has been submitted in support of an application for the other degree or qualification of this for the other university. If any act of plagiarism is found, we tend to are totally liable for each disciplinary action taken against us relying upon the seriousness of the established offence.

COPYRIGHT STATEMENT

- Copyright in text of this thesis rests with the student author. Copies are made according to the instructions given by the author of this report.
- This page should be part of any copies made. Further copies are made in accordance with such instructions and should not be made without permission (in writing) of the author.
- NUST College of E&ME entrusts the ownership of any intellectual property described in this thesis, subject to any previous agreement to the contrary, and may not be made available for use by any other person without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which exploitation and revelations may take place is available from the Library of NUST College of E&ME, Rawalpindi.

ACKNOWLEDGEMENTS

First and foremost, praises and because of the God, the Almighty, for his showers of blessings throughout our project work to complete the project with success. We would be happy to convey our project supervisor, Sir Sajid Gul, Professor, Department of computer Engineering, NUST faculty of Electrical and Mechanical Engineering, Rawalpindi, for giving us the chance to try to a project and providing priceless steering throughout this Project. His vision, sincerity, and motivation have deeply impressed us. He has educated us the methodology to hold out the project and to gift the project works as clearly as doable. It absolutely was an honor for us to work with him and beneath his steering.

We would prefer to convey to our parents and friends, while not whose unthinkable support and constant motivation, we'd not are able to complete our final year project. They played a unique role throughout our journey, and we are ever more appreciative of them. Their constant support intended us to do over what we tend to ever accomplish, and that they impressed new hope in us, after we found none in ourselves.

ABSTRACT

The aim of the project "Air Quality Monitoring" using low-cost sensor is to measure the air quality index after measuring the concentration of air pollutants in a particular area and display the results on mobile application. The proposed system architecture integrates these sensors which are capable of measuring key air quality parameters, including particulate matter (PM), carbon monoxide (CO), and nitrogen dioxide (NO₂). These sensors wirelessly transmit the collected data to a LoRaWAN gateway, which serves as a relay for the Helium network. The Helium network, functioning as a decentralized infrastructure, ensures secure data transmission and storage. The main feature of the project is to develop a mobile application that displays the air quality index derived from the measured pollutant concentrations using sensors. The application serves as user-friendly interface, allowing users to access real-time air quality information and make decisions regarding their health and well-being.

Chapter 01

Chapter 1: Introduction

Overview

With the increase of population in cities threats to its environments also increases. The growth of population of cities consequently influences the increase in the air pollution. The main source of Air Pollution in Earth's atmosphere is created by human beings themselves, emissions from vehicles, power plants, factories, agricultural sources and from wildfires, and volcanoes which are natural sources of air pollution. ¹ The main cause of air pollution is linked directly to increasing health issues like lungs and heart diseases including stroke chronic and acute respiratory disorders also causing premature deaths in many areas. High air quality is quite important to ensure the future development of the city, considering attracting tourists and investors that would increase businesses and economic growth of any country.

Air quality varies greatly on very close geographical levels, due to locally present pollution emission sources. Air quality monitoring in most of the areas is usually performed using public stations, but due to their costly and challenging maintenance, the number of areas covered by these monitoring devices is very limited. Hence, most of the places are left out of bound by these devices. To overcome this shortage, a network of low-cost sensors is being developed which will expand spatial density of air quality measurement, and hence monitoring air quality of most of the areas which are left uncovered by public stations, some of the benefits of low-cost sensors and deployment have been discussed by various authors and our work is also related to this ,i.e. we are increasing the spatial density of the areas covered by air quality monitoring devices which are in our case low-cost sensors hence providing the air quality index of most of the areas which are not covered by public stations.

We are using low-cost sensors which are integrated on Lora-Wan device which consists of an STM-32 microcontroller board which processes all the data from the sensors and a Lora-Wan chip which is like Wi-Fi technology, i.e., it is only used to transfer the processed data to the cloud and then the cloud is connected to the flutter mobile application.

Our goal is to design low-cost sensors which will collect live data from sensors process the data on end nodes, thus calculating air quality index and then transfer it using LoRaWAN technology to the Cloud and then after fetching data from cloud it will be displayed on our mobile application.

There are many benefits of using low-cost sensors along with their configuration, adjustment techniques and accurateness. As we need to deploy large number of sensors to create a dense pollution map that more locations are covered, but these sensors should be low-cost to make the system more cost-effective, distributive and could be deployed on large-scale. As we know that sensors are very sensitive to humidity and temperature, we must create such algorithms that could eliminate maximum error.

1.1 Scope

Our project aims to design a working device that can measure Air Quality Index through the low-cost sensors that can measure concentrations of gases including CO, NO₂ and PM_{2.5}. This device will be made in the form of a small node and can be carried to any place from which we want to get air quality information very easily. i.e., And that node we have made will be placed to different regions from where we want to get, Air Quality Information And that node then will be shown as location in our mobile application so that if we tap that location the real-time Air Quality Information from that node will be shown to the user who is using our “Airify” app. This app will not only show Air Quality Index and concentrations of different gases but also some warnings and suggestions based on Air quality Index (AQI). For example, if AQI of a specific location is high it indicates bad Air quality so for that warnings and suggestions will be that You should avoid going outside and spend more time indoors, etc.

1.2 Objectives

The main objectives that we have tend to achieve while doing this project must include:

- To develop low-cost sensors which are integrated on-chip that collects real-time data accurately from these integrated nodes.
- To process this data on end nodes and calculate Air Quality Index and concentrations of different pollutants like Carbon Monoxide (CO), Nitrogen Dioxide (NO₂) and Particulate Matter (PM_{2.5}).
- To transfer the real-time data gathered from sensors and hence displaying on to mobile application the color-coded readings (Air Quality Index) AQI, (Common Air Quality Index) CAQI, concentrations of gases like CO, NO₂ and PM_{2.5} and the warnings as per AQI.
- To ensure that our calculated readings are accurate as per standards defined.
- To raise awareness about the Air Quality and its causes among the people through our mobile application and suggesting them different ways that can help them reduce harmful effects caused by undesirably high Air Quality Index.

1.3 Methodology

Our whole project methodology can be shown from system level diagram from the figure 1.

In figure 1 we have system level diagram incorporating all the stages of development of our project starting with the edge nodes comprising of the sensors and the microcontroller integrated and hence, processing data on them and calculating Air Quality Index (AQI).

The values of PM_{2.5} and concentration of CO and NO₂ will be used to calculate the air quality index of that particular place accurately. The AQI values from 3 to 4 edge nodes will be sent to the FOG nodes that will simply find the average AQI. The values from the edge nodes can be used to display AQI of a specific region on our application and the FOG nodes to display it for a larger region than one specific region.

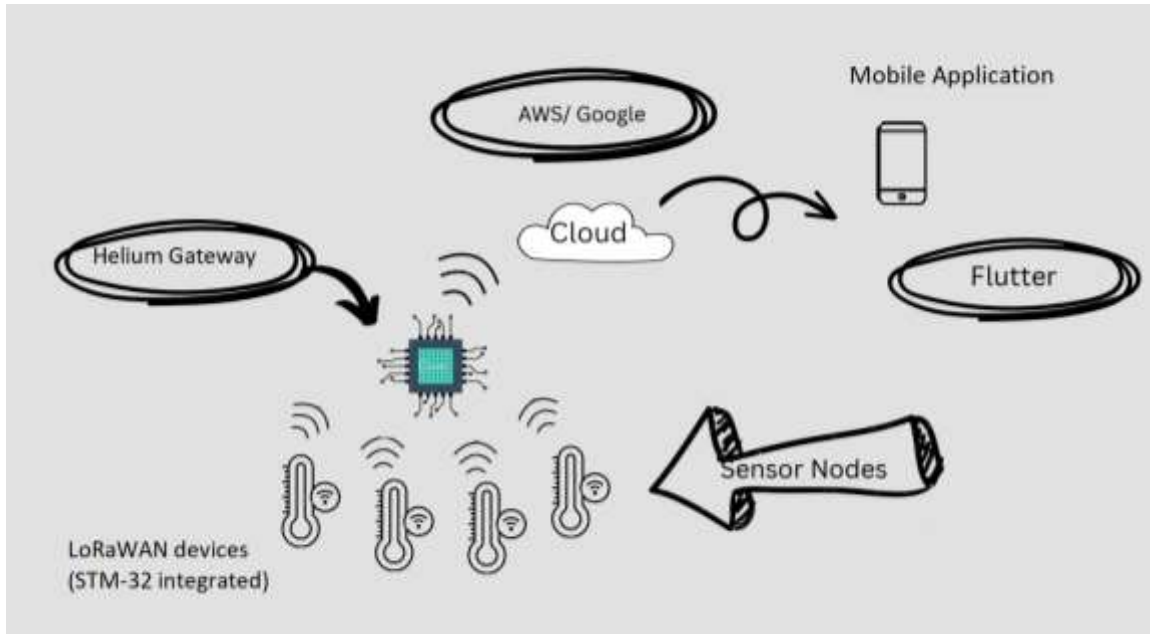


Figure 1: AQM System Level Diagram

Since, we will have all the values of AQI coming on Helium Console from the Lora WAN chip in the form of real-time packets, we will decode those packets to extract the main values that we need and store them on a DynamoDB table created on Amazon Web Services. We have used the Lambda Function for this cause since it allows us to easily write functions and execute them on AWS without having to worry about handling the server side. The documentation AWS Lambda Documentation (2022) clearly explains the previous facts that we just need to upload our code to run any type of backend or application in any suitable language of our choice and the rest is handled by AWS. This makes our project way simpler and manageable, because now Lambda will oversee everything from its running to scalability.

Through APIs we will store the AQI, CAQI, PM_{2.5}, NO₂ and CO values from Helium to DynamoDB with Lambda Function to handle the flow, after the values have been stored, they will be accessed by our Flutter Application using yet another Lambda Function through API keys. The application will show real-time AQI values along with notification, warnings and suggestions based on the current AQI value of the area.

1.4 Structure

The structure that we followed while writing this report is as followed:

- In chapter 1, we defined the general summary, aim range and importance of our project and how the project will be structured.
- In chapter 2, we have described about the main pollutants in air, measuring and sampling their concentration, background work and research on our project.
- Chapter 3 deals with the innovative and technical side of the project, explaining about the design decisions taken to make it different, its architecture, hardware components, software components and the integration between them.
- Chapter 4 deals with the design and development of the AQM device, the mobile application and the end node consisting of the sensors to measure the pollutants with explaining the underlying theory of operation and system integration.
- Chapter 5 deals with integration of STM32(microcontroller) with Helium which in turn is integrated with Amazon Web Service in which database stores AQI information from which Flutter application retrieves the values for visualization.
- Chapter 6 consists of a detailed analysis of the AQM application Airify, its properties, user flow diagrams, user interface and assets.
- Chapter 7 deals with the Conclusion and Future Work in which we have concluded our project and we have also written what we are tending to do in future.

Chapter 02

Chapter 02: Literature Review

Each day a person breaths 20,000 times and every breath carries in oxygen and brings carbon dioxide away. In recent years it became clear that breathing affects mankind to a great extent as COVID-19 spread around the globe. During a survey it was revealed that 4.2 million people die because of air pollution annually. Air quality is not the same everywhere; in rural areas it's not as much an issue as it is in urban areas. With traffic increasing every day, industrial sector expanding and global warming at its peak, we are at a greater risk from air pollution than we were a few years back. Environmental, health and safety professionals protect us from the threat of bad air quality but now-a-days their systems are outdated, unconnected and inefficient. They waste most of their time collecting, and then processing data on air quality – valuable time that should be spent on accessing the results and taking precautionary measures to avoid catastrophes.

Airify is on a task to revolutionize all that. We have built a real-time assessing air quality monitoring system which could be deployed anywhere and takes out the necessity of collecting and processing aqi data and frees our users to take actions that matter.

Air Quality Monitoring systems are deployed in areas that need to be kept under observation in terms of their environmental state such as industrial areas, places having heavy traffic etc. The main pollutants now-a-days are ozone, particulate matter PM_{2.5} and PM₁₀, Sulphur dioxide, carbon dioxide, nitrogen dioxide, carbon monoxide, to name a few.

When talking about the kind of audience our AQI monitoring system will be attracting, one of the main are outdoor enthusiasts, people having trouble breathing, those with respiratory or cardiovascular conditions and asthma patients. In addition to these, environmental advocates, who constantly need to monitor the air quality of an area so that they can adjust their policies and undertake certain actions in order to improve air quality and protect public health, and urban residents who also need to make appropriate decisions about going outside, or avoid it based on the current air quality situation outside.

Most of such systems are focused towards making it more reliable, providing real-time analysis of AQI and error correction, however, we aim to make our AQI monitoring system more focused towards its scalability than any other aspect.

The literature analyzed in this chapter is structured in 3 main paradigms for AQI detection. The first one is focused on Main Pollutants.



Figure 2: Effects of Air Pollution

2.1 Main Pollutants

Almost half of the air pollution is caused by dust and construction shown in blue color in Figure 3, i.e., 43%, 17% by waste burning, 14% by transport, 9, 8 and 7% by diesel generator, and domestic cooking respectively.

All of these combines to cause extremely unhealthy living environments not only for people with respiratory and breathing problems, but also for common urban residents.

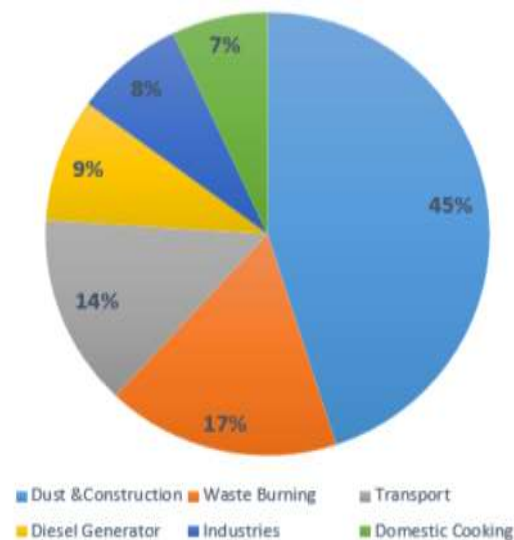


Figure 3: Main Pollution

Thus, to monitor such unhealthy air quality and then take appropriate counter measurements to prevent these conditions, we need a proper real-time assessment system to access the concentration of these harmful gases and particulate matter in the air and then calculate air quality index from those values.

2.2 International Standard Air Quality Index

AQI indicator is normally used by the ruler agencies to define how unhealthy the air is in terms of a scale from 0 to 500; zero being the lowest AQI value(healthy), while 500 being extremely hazardous.

According to the AQI meter shown in Figure 4, the AQI value up to 100 is bearable and almost healthy, but above that is considered harmful. From 100 up to 150 is

unsuitable for sensitive groups as explained earlier. Above 150 up to 300 is extremely unhealthy for all people and beyond that till 500 is considered hazardous and should be avoided at all costs. This is the international standard AQI meter that is followed in almost all the AQI monitoring systems around the world.

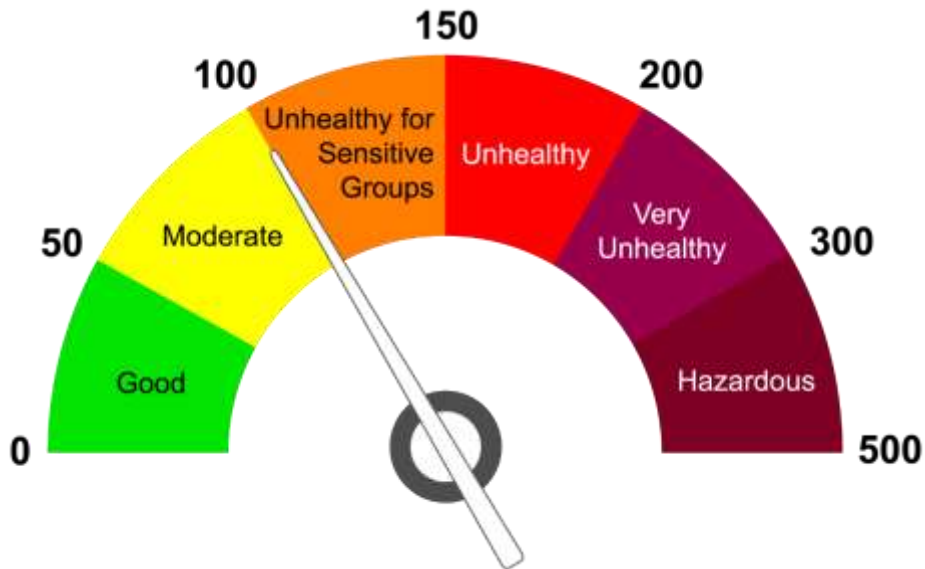


Figure 4: AQI Meter

2.6 Background & Related Work

When performing the research analysis, we studied in detail about the most widely used AQI monitoring systems globally. There are a few that are most preferred in the market:

Aeroqual, IQAir AirVisual, PurpleAir, Aclima, AirBeam, to name a few.

Each of the above listed AQI monitors are unique in their own special way and bring out new technologies that are irreplaceable and important. We shall discuss them one by one.

First, talking about Aeroqual, it's one of the top global AQM systems and can be used anywhere in the world. It has a system of alerts and notifications with which it gives you warnings in case the air quality index of your area goes above the threshold AQI that you have provided. The warnings given to you will be in the form of text or email. In addition to that a user will be briefed in detail about the technical side of the system and will be notified about updated required and the bugs that need to fix.

Its Series 500 Portable Air monitor [10](#), is a new step in innovation as it allows the user to get to know about the AQI anywhere he desires. It is a small portable device that can be easily carried around with you and displays the AQI around you.



Figure 5 :Aeroqual 500 Handheld Monitor Base

Just as in Aeroqual, IQAir AirVisual [8](#), also covers all the areas of the world, but along with that it also provides us with a 3D Air Visual Map which shows a detailed map of the area you choose having pointers displaying the AQI of every station close to you. These systems mostly use infrared light emitters that measure the concentration of PM_{2.5} and PM₁₀. IQ AirVisual provides both indoor and outdoor monitoring accessibility and with air cleaning solutions.

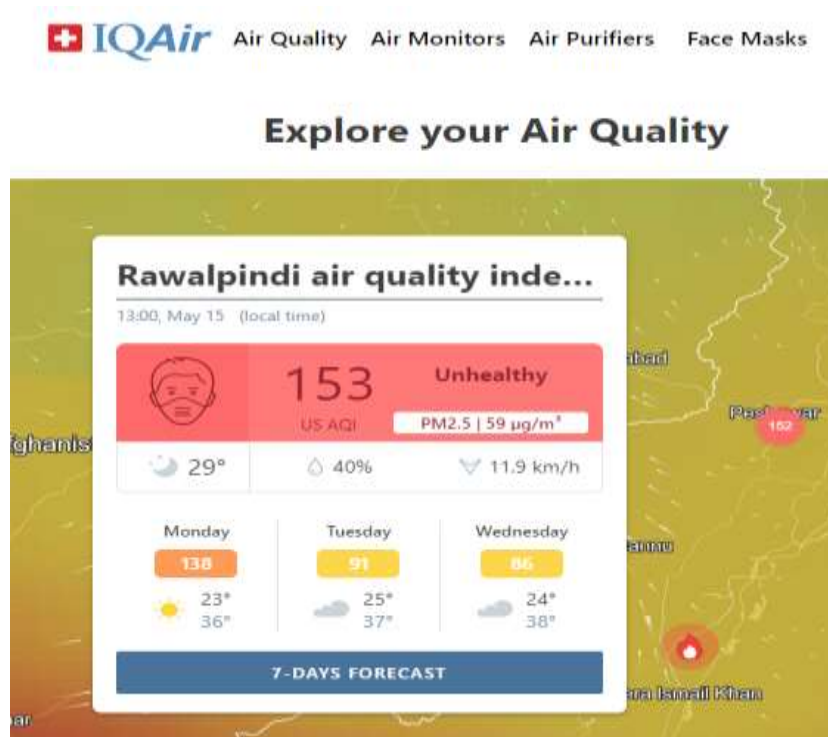


Figure 6: IQAir Map

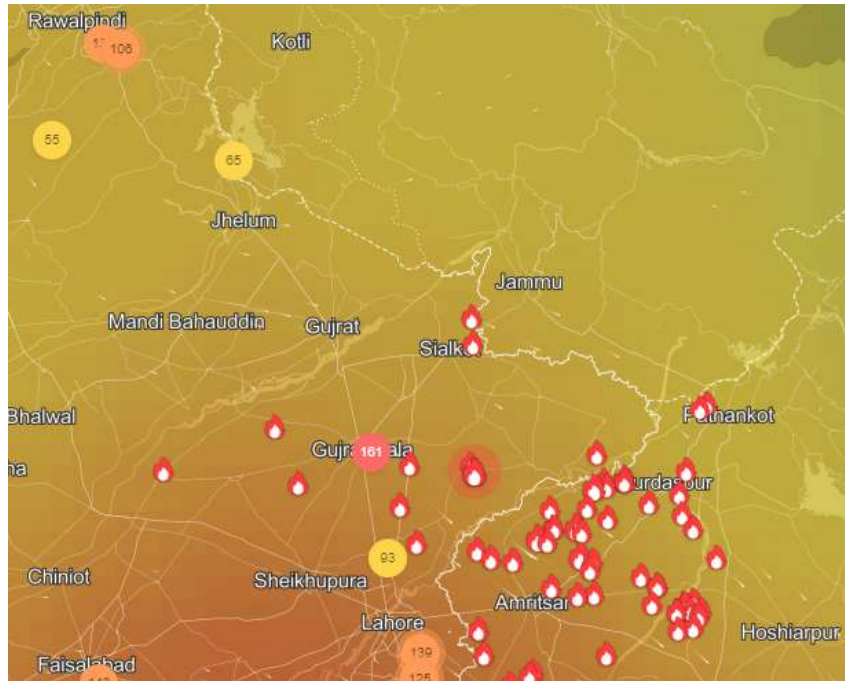


Figure 7: Map of Rawalpindi in IQAir Web App

As shown in the figure 8, the PurpleAir AQM [9](#), provides you with small, highly affordable AQI measuring devices that could be attached anywhere a user desires whether its indoor or outdoor, and it easily gets connected with user's Wi-Fi. After that the AQI of the desired area is showed on a local-area map on PurpleAir application.



Figure 8: PurpleAir Flex Quality Monitor Device

Aclima, yet another latest invention in the field of IoT, is the most progressive and innovative air quality measuring system in the market. The company has presented

more ways to analyse several of the greenhouse gases and air pollutants that contribute to the increasing AQI now-a-days.

It has introduced new and unparalleled network of stationary as well as roaming AQI sensors which offers the user with a wide set of pollutant and AQI measurements with block-by-block tenacity, the likes of which cannot be found anywhere else.



Figure 9: List of Pollutants measured by Aclima.

In addition to the AQM systems describes above, there are many others which make use of edge and FOG computing instead of Cloud to lessen the complexity of the network in cases when the area to cover is much diverse and continuous data is coming to the cloud for further visualization on the pollution maps.

Some systems use Raspberry Pi as the main hardware platform to get the sensor derived data onto the cloud with the means of built-in Wi-Fi connectivity [11](#). While in others, manufacturers are utilizing both stationary and roaming sensors to collect real-time data to train and create a ML algorithm which will then easily predict the future AQI and CAQI values. This system is quite brilliant and innovative in the sense

that no other company or project considered to take this approach while creating AQM system [12](#).

A rare AQM project in Helsinki is a massive one in which tens of thousands of sensors mainly of PM, CO, NO₂, T and RH were integrated and deployed covering huge area. In this project the maintenance and calibration were a serious issue, and the team was seriously challenged in the design and deployment of that much devices and sensors [13](#).

To deal with big data and continuous data transmission and processing, FOG was introduced so that there could be an edged platform between multiple end devices and cloud that would serve as a filter and sort out the data that is important and needs to be sent to the cloud from the one that could be discarded after certain computation. This reduces complexity and the data load on cloud and makes the entire system more efficient. Such approach was implemented in a AQM system that turned out to be quite the success [14](#).

Below is the Table 1 that summarizes the above explained AQM system available in the market, the approach they are taking in terms of Cloud or FOG and their Properties.

Research	Approach	Applicability	Properties
[11]	Cloud based IoT	Sensor cloud application	Monitoring and Notifications
[12]	Cloud based IoT	Prediction on AQI values using ML	Monitoring and Predictions
[13]	Cloud based IoT	Large scale sensor deployment and maintenance	Accuracy and Calibration
[14]	FOG based IoT	Distributed sensor cloud application	Manageability and Scalability

Table 1: A summarized review of the above proposed AQM system along with their properties and applicability.

2.7 Sampling and Measurement Techniques

Particulate Matter (PM) Measurement:

There are three basic Particulate matter measurement techniques: Gravimetric Sampling, Beta Attenuation Monitoring (BAM), and Laser Scattering.

In Gravimetric sampling, we need to have a special filter typically made of glass to collect dust particles in the air. After we have the particles, we measure their weight to find out the actual concentration of PM_{2.5} or PM₁₀.

In Beta Attenuation Monitoring (BAM), a small sample of air let inside the instruments through a tiny inlet and beta radiation is emitted through a source. Mainly strontium-90 is used as a beta radiation source. As a result of radiation emitted, when the dust particles collide with the strontium-90 isotopes, their intensity is measured and through this process the concentration of PM is estimated.

The process we are implementing in our project is Laser scattering. This involves emitting a laser light and then implementing the principles of light scattering to easily measure the size distribution and estimate the concentration of particulate matter in the air.

Gaseous Pollutant Measurement:

The sensor we are using works on the principle of chemiresistive sensing in which the concentration of the target gas is measured when the sensing element's electrical resistance changes due to its proximity with the target gas. Mainly, the sensing element is tin dioxide (SnO₂)

Chapter 03

Chapter 03: System Design and Architecture

3.1 System Architecture

System Architecture of our project is shown from the system level diagram:

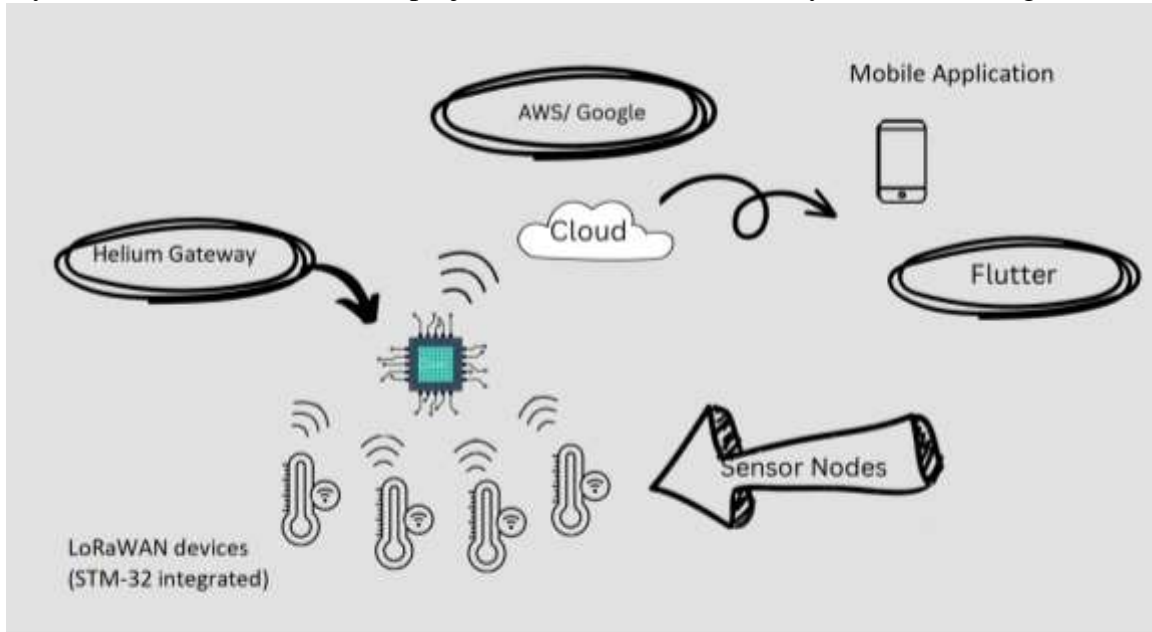


Figure 10 :System Level Diagram

Our integration and working of the whole system is shown from the above figure 10 i.e., We will be first making the hardware part which consists of integrating sensors which consists of two sensors MQ-135 which can detect gases like NH₃, NO_x, alcohol, Benzene, smoke, CO₂, etc. but we are using it to measure concentrations of only two gases which are NO and CO₂ and gp2y1010au0f which is used measure concentrations of dust particles which mainly includes PM_{2.5} these sensors are then integrated Lora-WAN module which is a device made by the integration of Lora-WAN technology and STM-32. The sensors that we used are attached to the pins of the Lora-WAN module which are basically the pins attached to STM-32. So, it is the microcontroller that controls all the processing we tend to do which includes extracting concentrations of different gases from the values that we get from the sensor and then calculating AQI (Air Quality Index) on the basis of concentrations of these gases. But the point is that we must transfer this information wirelessly and for that purpose we have used that LoRa-WAN technology which operates on low-power and as the name shows it is Wide Area Networking protocol which has been built using Long-Range radio modulation technique. This device connects wirelessly with the internet using Helium Console and makes management and communication easier communication between end-node devices which are collecting data to the Helium Gateway. As this device has a Long Range and has a very low power consumption with the benefit of bidirectional communication its use has increased vigorously where smart city concept is used because this device has most of the benefits that one needs

where IOT solutions are provided It uses the unlicensed ISM (Industrial, Scientific, Medical) radio bands for deployment of networks.

When this information is transferred to the helium console it is in the form of packets that are not readable by the humans as is, but they are decoded first to extract the main information that we want by removing throughputs and payloads, etc. And when these packets are decoded, we get our main information which in our case is concentrations of gases and Air Quality Index. When these packets are decoded, the data goes on Helium which is connected to AWS.

All the data is uploaded using helium gateway which provides connectivity between IOT devices and to the internet hence allowing data to be uploaded to the internet and on internet it uses helium console, and that helium console is already integrated with AWS using API keys which are generated using lambda function in which we have integrated helium console with Dynamo DB table that is service of AWS. And that DynamoDB table is connected to flutter app using another API key which generated through another lambda function. And then on flutter app API key is added to the code and using http requests we get values stored on DynamoDB table.

We will also be implementing FOG computing structure by which we communicate between our IOT device (LoRaWAN) which then apply an algorithm and take average of the data gathered from nearby device to show overall AQI (Air quality Index) data of a lager region. FOG computing only has its advantages when a large number of IOT devices are deployed because it then increases speed, efficiency and provides more reliable data transfer. So, we will implement this technology after we have enough nodes to make its use beneficial.

Architectural Diagram is shown below:

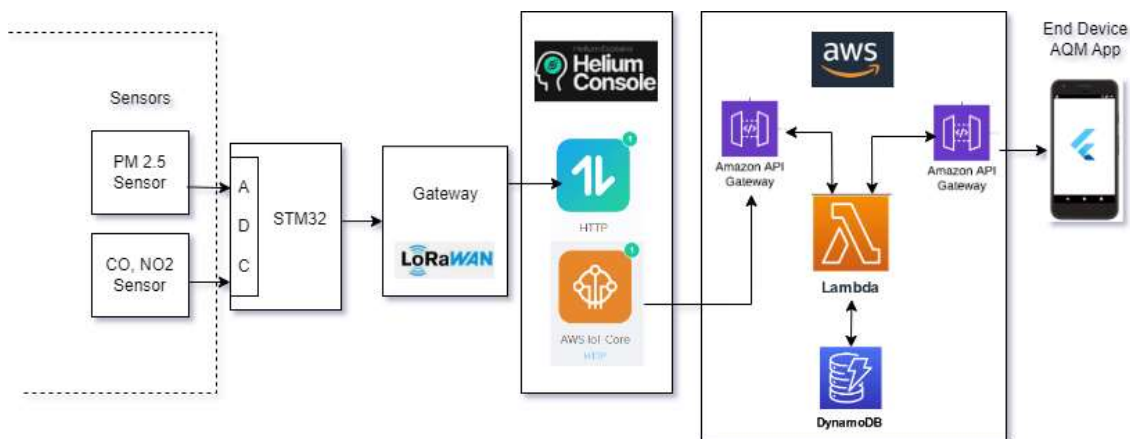


Figure 11 :Architecture Level Diagram

All of what we have explained in the above section can be visually represented in the form of an architecture diagram shown in figure 11. The architecture diagram shows that the system has multiple layers: Acquisition Layer (Sensor Nodes), IoT Network Layer (including Cloud Interface), Application Layer (End Users).

Also, we have a Flow Chart describing the whole functionality of our project as shown in figure 12.

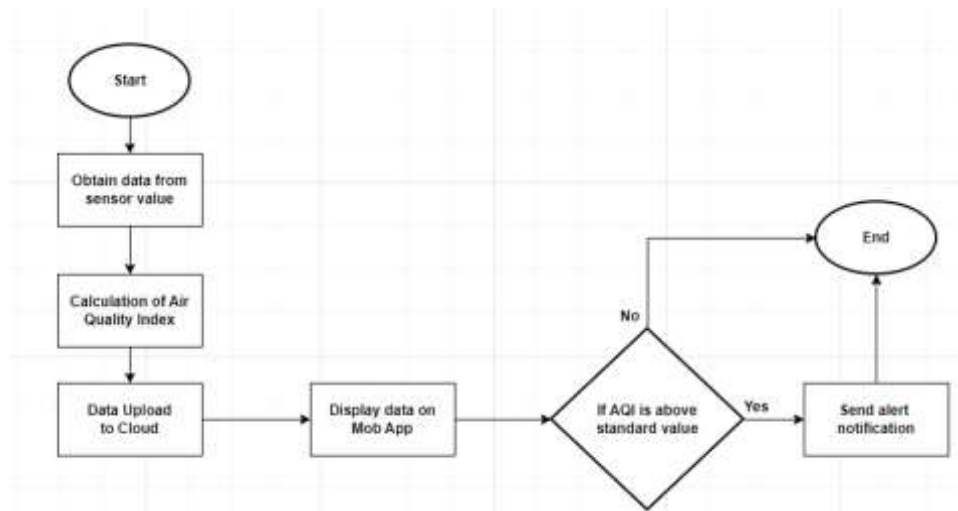


Figure 12 : Flow Diagram

The above flow diagram shows the overall flow of our project from start to end. First collect the data from sensor nodes through code calculate the AQI. Upload the data on cloud and then display it on Mobile App.

3.2 Hardware Components

3.2.1 Sensors:

The sensors we have used are chosen on the basis of their low-cost and concentrations of the gases that we need while calculating Air quality index. Our chosen sensors for this purpose are as follows:

3.2.2 Optical Dust Sensor - Gp2y1010au0f

The results of PM_{2.5} measurements with the Sharp sensors (simultaneous measurement of 3 Sharp sensors connected on two Arduino platforms) were compared with PM_{2.5} readings of the OSIRIS monitor over the whole measurements period (15-min averages, Osiris vs. Sharp). Correlation analysis of measurement results shows that there is a strong positive correlation between the mean 15-min PM_{2.5} concentrations measured with Sharp sensors and Osiris ($r_{S1} = 0.820$, $r_{S2} = 0.947$, $r_{S3} = 0.738$). The results of regression analysis (shown in Figs. 3-6. and in Tab. 1) shows that measurements of Sharp sensors are in good agreement with the Osiris measurements. On the other hand, Sharp sensors S1 and S3 underestimates while sensor S2 overestimates the indoor PM_{2.5} concentrations relative to the Osiris measurements. We assume that such differences are caused by the different characteristics of photo elements that were built in the sensors. Namely, sensor S2 was supplied from a different manufacturer than sensors S1 and S3.

3.2.3 MQ-135 Sensor:

The next component used in air quality control equipment is MQ-135 gas sensor and which is used for detecting or measuring of NH₃, Alcohol and Benzene. The sensor

module comes with a Digital Pin. This means that this sensor will operate even without a microcontroller and that comes in handy and is more suitable for detecting only a particular gas. To measure the gases in PPM, the analog pin is used. The analog pin is TTL driven and works on 5V and so can be used with the most common microcontroller.

3.2.4 Data Acquisition System:

Hardware systems involved in Data Acquisition include specifically LoRa-WAN module STM-32 and Helium gateway Raspberry Pi:

3.2.5 LoRa-WAN Module STM-32:

The LoRa-WAN module STM-32 is a microcontroller-based module that has a Long-Range (LoRa) radio transceiver with microcontroller integrated on it, microcontroller used is from STM-32 series. LoRa radio transceiver which is used for wireless communication having long-range and low-power communication. And the Microcontroller that is integrated and we have used handles all the processing on the data that is acquired from the sensors and then extracting the main information that we need. As we have used LoRaWAN module for this purpose hence the data transferred will be using LoRa-WAN protocol. This protocol allows communication between LoRaWAN gateways and LoRaWAN Network of LoRaWAN modules. This LoRaWAN module is connected to two different sensors MQ 135 for detection of CO and NO₂ and gp2y1010au0f sensor which detects PM_{2.5} concentrations.

3.2.6 Helium Gateway Raspberry Pi:

Helium gateway is dependent on Raspberry Pi, which is single board low-cost computer that provides computing power that has compulsory computing and connectivity benefits. This Module runs on the Helium software that provides communication between these LoRaWAN IOT devices and this Helium Network is basically a wireless communication network which provides communication between LoRaWAN devices and the Cloud through Helium Gateway. This Gateway is connected with a LoRaWAN radio module, which can receive and transmit LoRaWAN transmissions from these LoRaWAN devices. The packets received from LoRaWAN devices, through this gateway are validated, decrypted, and are then forwarded to the Helium Network server.

3.2.7 Working together:

The LoRaWAN module, with its Long-Range radio transceiver and Microcontroller basically STM-32, gathers data from gas sensors and formats it into LoRaWAN packets. These packets are then transmitted wirelessly using the LoRa modulation technique and hence with LoRaWAN protocol. And these packets are then received through the helium gateway from the LoRaWAN devices through the LoRa radio transmitter. Helium Gateway software which runs on Raspberry Pi validates and decrypts the received packets and then forwards the data to the helium network. Helium network processes the data, store it, and make it available for further analysis or consumption by your application. This architecture enables the acquisition of data from the LoRaWAN devices its transmission by the Long-Range radio transceiver and then transmission to the Helium Network for further processing and storing of data.

3.2.8 Communication modules:

Communication between the LoRa-WAN module and Helium Network is mainly through the radio transceiver integrated with it. But both LoRa-WAN module (with STM-32 as processing unit) and the Helium Gateway itself acts as the communication module for transmitting data wirelessly via LoRaWAN protocol. LoRaWAN module collects air data from sensors, processes and calculate Air Quality Index, and formats it into LoRaWAN packets. These LoRaWAN packets are then transmitted wirelessly using the integrated LoRa radio transceiver. The provided LoRaWAN protocol allows it to communicate with Helium gateways and join these networks through LoRa radio transceiver.

The gateway receives and forwards the LoRaWAN packets from the LoRaWAN modules to the Helium network. It does not serve as an independent communication module as it relies on the packets received from the LoRaWAN module for transmission of data.

To sum up the whole description, the LoRaWAN module consists of the communication capabilities required for transmitting data through wirelessly using LoRaWAN radio transceiver, whereas the Helium gateway serves as the device responsible for forwarding the received packets to the Helium network.

3.2.9 Processing Units:

The processing units may include Microcontroller unit integrated on LoRaWAN module which is STM-32 from STM Microelectronics. This is the device that will handle all the processing happening on incoming data. i.e., the data from the sensors will be acquired through processing that will be performed on STM-32 and the data thus acquired is concentration of gases from the specific sensors and then for the implementation of an algorithm to calculate Air Quality Index further processing will take place and all of this processing will be taking place on STM-32 integrated on LoRaWAN module.

3.2.10 Storage:

After all the processing the data will be transferred to the Helium Console through Helium gateway and this helium console is then integrated with AWS all the data storage and handling will be happened on Amazon Web Services servers.

3.2.11 Power Supply:

Power Supply is attached to the LoRaWAN devices in the form of two small cells total of 5V.

3.2.12 Interface:

In this project, various hardware interfaces were utilized to connect the system to external devices or peripherals. They are very important to communicate and exchange data with the system.

First of all, comes sensor interface which includes various air quality sensors and displays value and concentration of multiple hazardous gases. These interfaces also use protocols to transmit and receive data in a sequential manner such as I2C, UART, and analog inputs to read sensor data. Then comes the microcontroller interface, which

includes STM32 that connects sensor, store data and helps communicate with other devices. STM32 interface provides GPIO pins for connection. The LORAWAN interface enables transmitting data wirelessly to a gateway. Also, it has a wide range up to 15km. Since the system includes displaying data on real-time chips it includes a display interface which uses I2C to display data on screens. To burn the code/instruction on the device a USB called ST-LINK was used which proved to be a source of communication between STM32CUBE IDE code and the hardware attached whereas to display the data converter USB was used which displayed the outputs on Hercules.

3.3 Software Components

3.3.1 Application Framework

The application framework that we have used is Flutter. It is an open-source framework developed by Google for building beautiful and **open-source framework** by Google for building beautiful, **natively composed**, can **develop** multi-platform applications from a single codebase. It is **fast** as it compiles to Intel, ARM machine and also JavaScript, for fast performance on any device. **Productive**, as it can build and can iteratively perform changes with Hot Reload, update the code, and see changes almost as soon as code is reloaded, without losing state. **Flexible**, as it can control all the pixels to create a very customized design that looks amazingly well on any screen. And also, Integration is handled very easily when API keys are used. And there is another Application that was used to interact with hardware named as STM32CubeIDE. It is an advanced development platform using C/C++ with peripheral configuration, to generate code, compile it, and for debugging STM32 microcontrollers and microprocessors. This application is based on Eclipse[®]/CDT[™] framework and toolchain named GCC for the development purposes, and GDB for debugging purpose. So, we first generated a sample code and then uploaded it to the STM-32 integrated chip using this software.

3.3.2 User Interface (UI)

User Interface (UI) was first designed on Figma. It is a collaborative web-based application that is used for interface design, with added features that are also supported offline enabled by desktop applications for mac Operating System and Windows. Hence, due to this application benefits we used it and hence, our project was collaborative it was very beneficial getting every member's opinion while designing it. Some of the UI elements were slightly changed while designing the application on flutter to make it look even better than what we designed. The color theme that is used in the application is very user-friendly look at as the color used is not very sharp as the sharp color cause eye strain hence making it difficult for user to look at. So, the color theme was also chosen very carefully. And the UI components. i.e., Buttons are also rounded, not very sharp edges are used as sharp edges are not good for the eyes. Also, there is a navigation bar in the app that makes the most used buttons very approachable that makes the user experience better and for the textual information lists

are used so that it makes every list easy to find on application and also the number of elements each list has are not very filled. Also, Map is used to display Air Quality Information which is accessible to use as it displays all the information very clearly. Also, the Air Quality Index is shown by separated color bands based on the standards of the ranges defined already. The Air Quality Index is shown in that specific color band so that the information inside that is more targeted. And for the warnings the color is changed according to the AQI of the boxes on which warnings and suggestions are shown. User Interfaces with detailed description are attached in the Software Section of this report.

3.3.3 Networking and Communication:

The software applications used for networking and communication include Helium console and Cloud which is Amazon Web Services in our case. LoRaWAN devices interact with each other through the gateway which is connected to helium console all the data is uploaded to helium console which is connected to Amazon Webservices and DynamoDB table is used for storing values and on Amazon Web Services we have used another Service of them which is called as Lambda function. This service allows us to write a function according to the use-case and connect different applications. And we have used this service to connect Helium Console with our DynamoDB table which was created by us and then we have used another lambda function to connect DynamoDB table with flutter Application. These lambda functions when compiled we generate an API key which is kind of a weblink. These API keys are basically used for this purpose.

API keys so generated will be used inside the Helium Console and flutter application.

3.3.4 Data Visualization:

As the data is visualized on flutter application at the end of integration cycle All the concentrations of different gases and the AQI value for the display of Air Quality Index we have made a circle of the band assigned to a specific range of AQI values and inside that circle the AQI (Air Quality Index) is shown and a word assigned to the AQI is also shown i.e., For AQI less than 25 we will show very low of the color band assigned colored text. Following is the code shown with what will be displayed

according to what value of AQI. Also, the screenshots are also attached which show 20 AQI and 56 AQI and displays colors and text according to that.

```
if (aqi <= 25) {  
  color = Color.fromRGBO(27, 147, 31, 1);  
  label = 'very low';  
} else if (aqi <= 50) {  
  color = Color.fromARGB(255, 115, 208, 78);  
  label = 'low';  
} else if (aqi <= 75) {  
  color = Colors.orange;  
  label = 'medium';  
} else if (aqi <= 100) {  
  color = Colors.red;  
  label = 'high';  
} else {  
  color = Color.fromARGB(255, 129, 5, 5);  
  label = 'very high';  
}
```

Figure 13: Code of colored assigned according to AQI ranges.

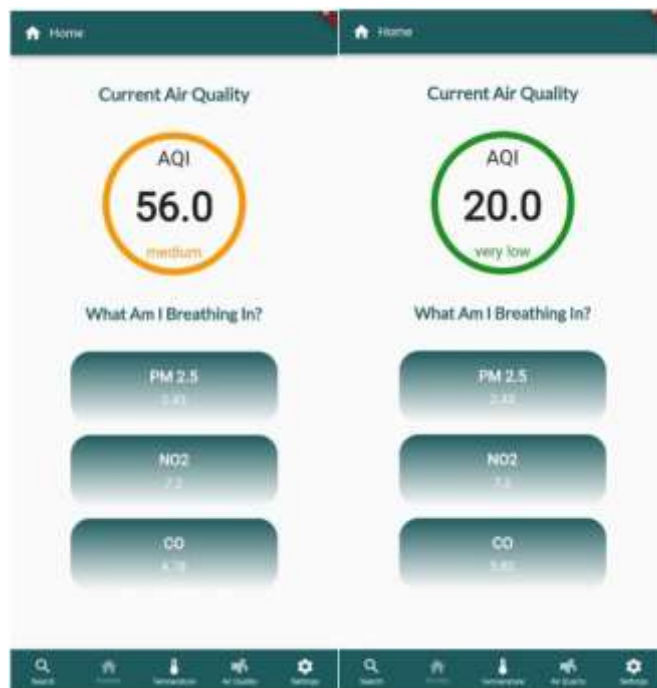


Figure 14: Home pages with different AQI showing the color assigned within the ranges.

3.3.5 Data Storage:

For data storage purposes the software used are Amazon Web Services and the services we have used are DynamoDB table and the second Service that is basically used to generate API keys that are used to connect the table with the Helium Console and the Flutter application so the data coming from the Helium Console will be stored

in AWS DynamoDB table and then from there it will be transferred to our flutter application. AS we have used cloud for storage it has many benefits it allows us to store as much data as we want according to our needs we will be charged and data management on any amount of data will be handled by AWS itself. Hence making data storage and retrieval easier as it manages all the infrastructure including data replication, automatic scaling, automatic backups, and also seamless integration. Due to various benefits of Cloud, we have used its services. And Amazon Web Services are better than other cloud services according to different third-party consumers and also, we had a little experience on AWS that is why have chosen AWS over other cloud services.



Figure 15: AWS Website

3.3.6 Integration and APIs:

As all the data is stored on AWS by using its DynamoDB table and then we have used yet another service called lambda functions, and this performs administration to compute resources. This includes server and operating system management capacity provisioning, automatic scaling, code monitoring and logging, code, and security deployment. In our case we have used it for handling http and API requests as soon as the data is uploaded through helium gateway it sends a trigger and then to the cloud. i.e., DynamoDB table updates the value and whenever on flutter application the homepage or air quality page is opened. It updates the value by using API keys generated through the lambda function.

3.4 Design Decisions

Design decision is a vast subject of consideration when one is planning to start and execute a project smoothly. It involves several important aspects of a project such as scalability and flexibility, system optimization and performance, resource allocation and optimization, technical considerations, error handling and testing.

When designing our project, we had to keep in mind the type of sensors we had to use, their testing and reliability, their compatibility with STM32 and efficiency.

In the software domain we had to keep in mind what a common user is looking for in such Air quality monitoring applications and what are his expectations. We needed to make our application easy to navigate and informative.

3.4.1 Sensor Selection

Since we are using STM32 as the main microcontroller, we first researched its I/O pins, analog to digital converter protocols and its datasheet before selecting the appropriate sensor to calculate the AQI.

STM32 uses advanced communication interfaces such as: two USART (supporting LIN, smartcard, IrDA, modem control and ISO7816), I2C, two SPIs (up to 16 MHz, one supporting I2S), and one low-power UART (LPUART).

These devices also have 12-bit ADC, a 12-bit DAC with which we can integrate a large range of sensors to convert their analog input to digital data or vice versa.

MQ-135 and GP2Y1010AU0F sensor both can easily interface with STM32 using GPIOs and UART protocol.

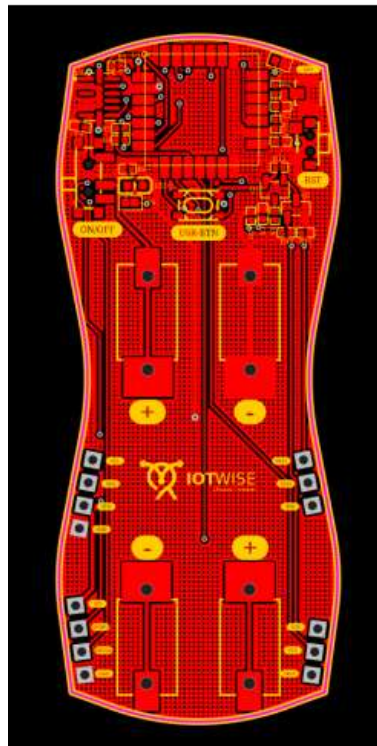


Figure 16:Wise Node PCB Layout

3.4.2 Sampling Methodology

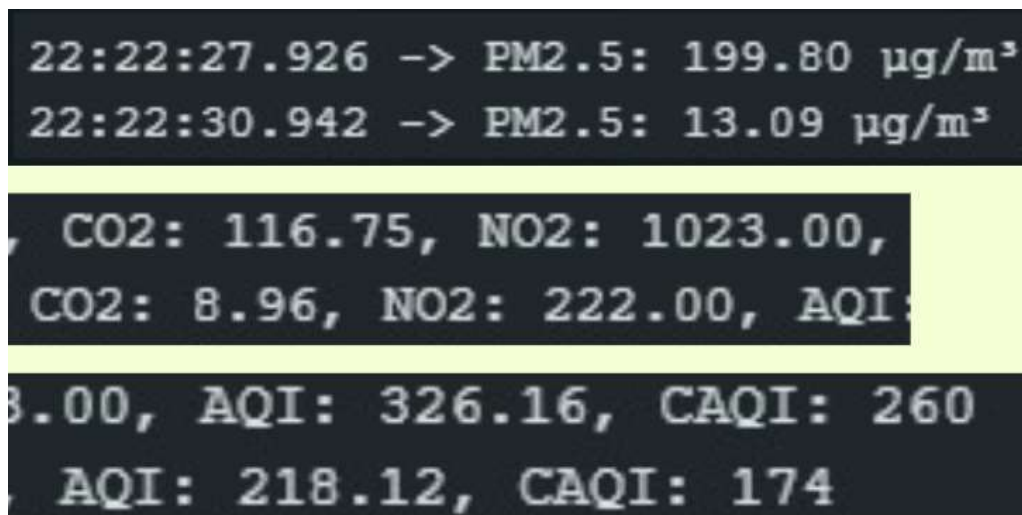
In most weather and AQI monitoring applications, it's not necessary for the system to be continuously updated with new weather conditions or AQI values because they stay the same for at least half an hour. Considering this assumption, we will be collecting the data from sensors after every fifteen minutes, in other words we will only collect sensor values 4 times in an hour. STM32 is programmed that way to have a time step of 15 minutes before collecting samples again. This reduces the data load and also meets our requirements.

3.4.3 Data Acquisition System

UART (Universal Asynchronous Receiver-Transmitter) protocol, used for the serial connection between two devices, TX, and RX pins to transmit and receive the data from sensors and then calculate the AQI from them. We set appropriate baud rate, stop bits, parity bits and data bits to align with the configuration of both the sensors we are using.

3.4.4 Data Processing and Analysis

Shown below are the AQI values calculated using appropriate formula from the sensor values. This output was derived from the testing stage of our project when we used Arduino and integrated it with the sensors to check if they were functional. After uploading the code written in C language, we got the results on Arduino output terminal showing the time at which the sample was taken, concentration of PM_{2.5}, CO, NO₂ and finally the air quality index calculated. Figure 17 shows the output of that screen.



The image shows a screenshot of an Arduino terminal window with a black background and white text. The text displays the following data points:

```
22:22:27.926 -> PM2.5: 199.80 µg/m³  
22:22:30.942 -> PM2.5: 13.09 µg/m³  
CO2: 116.75, NO2: 1023.00,  
CO2: 8.96, NO2: 222.00, AQI:  
3.00, AQI: 326.16, CAQI: 260  
AQI: 218.12, CAQI: 174
```

Figure 17 :AQI values calculated using Arduino.

3.4.5 Data Visualization and User Interface

Data visualization and user interface are one of the most important fragments of our project. Since it is an air quality application on which we are displaying the aqi of a fixed area, we have the map of a specific area on our home screen; in our case it is the map of EME. Red pointers will be placed on the locations where the sensors are deployed to be precise about the air quality index of that location. On the other hand, when the user selects the pointer, he will automatically display the Air Quality page having the AQI index along with the concentration of each pollutant we are measuring, in our case CO, NO₂ and PM_{2.5}.

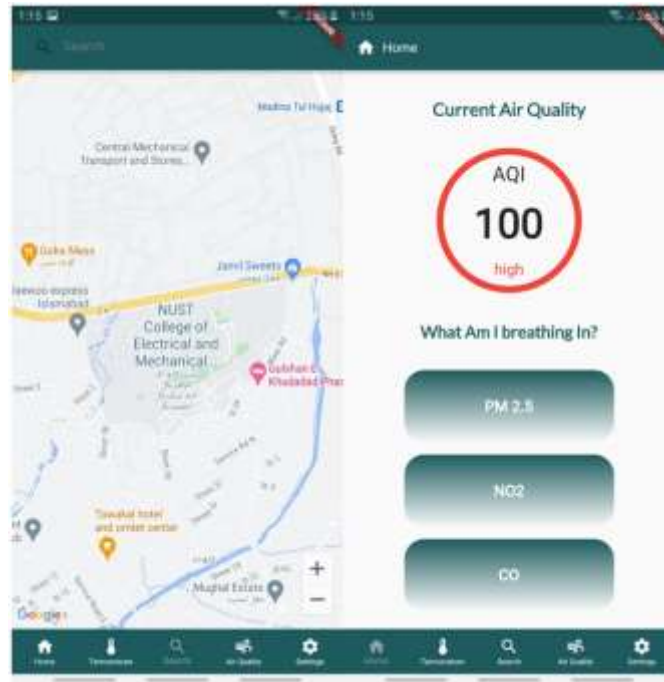


Figure 18 : Search and Air Quality page

The screen snip of the application in figure 18 shows the Home Page and the Air Quality Page.

3.4.6 Power Supply and Energy Efficiency

LoRaWAN is widely recognized because of its low-power consumption and long-range data transmission proficiencies. Due to this, it is used in a lot of IoT projects that need low-power efficiency.

In our project we needed to deploy those sensors which consumed minimum power but had high accuracy. Also, we can apply the sleep modes available in the STM32 and keep it in low-power consumption sleep mode most of the time. Similarly, we can have it send large data packets at long interval ls instead of sending multiple small packets every other second. This improves its low power consumption ability and also energy efficiency.

Chapter 04

Chapter 04: ALGORITHM DESIGN AND DEVELOPMENT

We will talk about how to create the algorithm for our project in this chapter. We started by extracting data from our sensors using an Arduino UNO and collecting the values from them. We employed two sensors to measure carbon dioxide (CO₂), nitrogen oxide (NO_x), and dust particles (PM_{2.5}), utilizing the MQ-135 and GP2Y1010AU0F sensors, respectively.



Figure 19 :MQ-135 sensor



Figure 20 :GP2Y1010AU

4.1 Working:

We will briefly explain how our project functions in this section. The project's main objective is to monitor the air quality in a particular area using low-cost sensors such as MQ-135 sensor, GP2Y1010AU sensor, and Arduino microcontroller board. Principal Elements playing a significant role in the success of our project include:

4.1.1 Arduino:

The project first used an Arduino microcontroller board to read the sensor data and analyze it to create the air quality index. 1. We followed the procedures below to collect data from the MQ-135 and GP2Y1010AU0F sensors using Arduino:

The Arduino was initially coupled with sensors; MQ-135 sensor to a pin of the analogue input shield utilized by the Arduino. GP2Y1010AU0F sensor to one of the digital input pins of the Arduino. Afterwards, the computer's Arduino IDE (Integrated Development Environment) was installed. Used a USB cable to link the Arduino board to the computer. Selected the proper board and port from the Tools menu after starting the Arduino IDE. To read the analogue input from the MQ-135 sensor, we used the **analogRead** () method. A raw figure was received that represents the gas concentration as a result. To read the digital output from the GP2Y1010AU0F sensor, we used the **digitalRead** () method. A raw value was provided for the level of dust particles. To translate the raw results into usable values such as gas concentrations and PM levels, implement calibration methods or lookup tables relevant to each sensor. Calculated the calibrated values using the interpolation or calibration equations found

in the Arduino code. Due to the built-in serial communication features on Arduino boards, data was transported using a USB cable attached to a computer.

Model		MQ135	
Sensor Type		Semiconductor	
Standard Encapsulation		Bakelite, Metal cap	
Target Gas		ammonia gas, sulfide, benzene series steam	
Detection range		10~1000ppm(ammonia gas, toluene, hydrogen, smoke)	
Standard Circuit Conditions	Loop Voltage	V_c	$\leq 24V$ DC
	Heater Voltage	V_H	$5.0V \pm 0.1V$ AC or DC
	Load Resistance	R_L	Adjustable
Sensor character under standard test conditions	Heater Resistance	R_H	$29\Omega \pm 3\Omega$ (room tem.)
	Heater consumption	P_H	$\leq 950mW$
	Sensitivity	S	$R_s(\text{in air})/R_s(\text{in } 400ppm \text{ H}_2) \geq 5$
	Output Voltage	V_s	$2.0V \sim 4.0V$ (in 400ppm H_2)
	Concentration Slope	α	$\leq 0.6(R_{400ppm}/R_{100ppm} \text{ H}_2)$
Standard test conditions	Tem. Humidity	$20^\circ\text{C} \pm 2^\circ\text{C}$; $55\% \pm 5\%RH$	
	Standard test circuit	$V_c: 5.0V \pm 0.1V$; $V_H: 5.0V \pm 0.1V$	
	Preheat time	Over 48 hours	

Figure 21: Datasheet of MQ-135

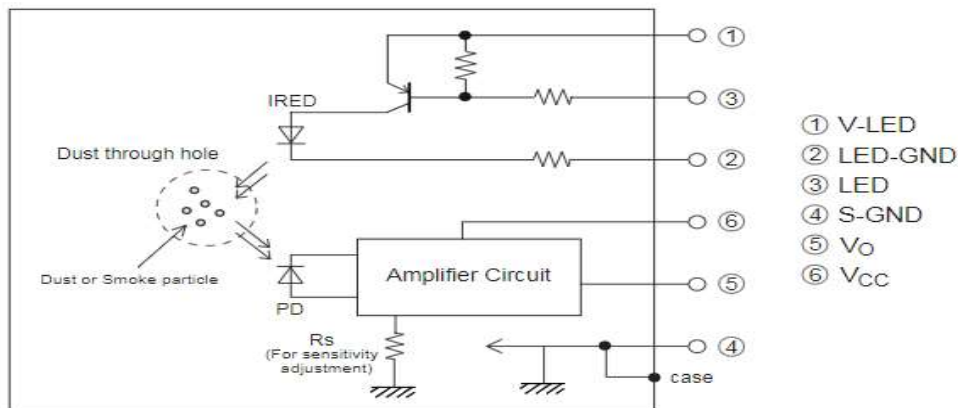


Figure 22: Datasheet of GP2Y1010AU0F

To communicate the sensor data to a computer for additional processing or storage, Arduino Serial Library was utilized. LED was used with Arduino to display the sensor readings in real-time to visualize the data. Use appropriate power management strategies, such as sleep modes or turning off unnecessary peripherals, to reduce power consumption and, if necessary, lengthen battery life.

4.1.2 Circuit with Arduino:

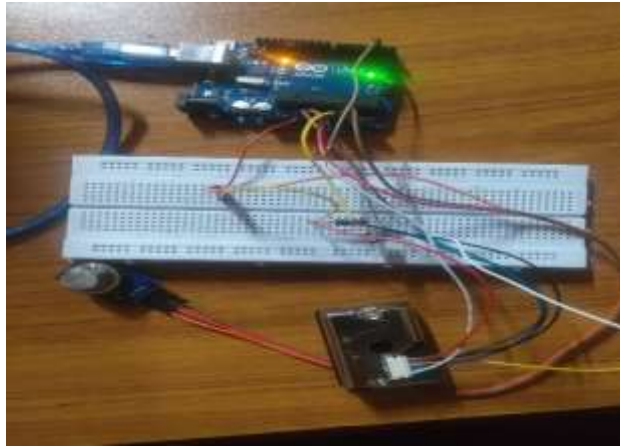


Figure 23: Circuit with Arduino

4.1.3 STM32 LORAWAN:

The entire hardware was swapped to STM32 after calculating the air quality index, and ST-LINK was used to burn code in the STM32cube IDE and displayed using UART protocol. All the data was kept on AWS (Amazon Web Services) by the helium gateway. Since it receives data from the STM32 LoRaWAN device over a LoRaWAN network and then transmits it to the internet using cellular or Ethernet connectivity. The gateway also performs other duties like packet filtering, data routing, and network management.

An outline of the design and development process is provided below:STM32 Microcontroller: Selected an STM32 microcontroller that can communicate via LoRaWAN and has enough GPIO pins and peripherals to integrate sensors. MQ-135 Gas Sensor: The MQ-135 is frequently used to identify numerous gases, including benzene, ammonia, carbon dioxide, and other dangerous contaminants. GP2Y1010AU0F Dust Sensor: The GP2Y1010AU0F monitors the amount of dust in the air using a particulate matter (PM) sensor. Connect the STM32 microcontroller's GPIO pins for the MQ-135 and GP2Y1010AU0F sensors. Make sure the STM32 microcontroller voltage levels and the sensor needs are matched by using the appropriate voltage levels, such as voltage dividers or level shifters. To read data from the sensors, implement the required interface protocols, such as analog-to-digital conversion (ADC) or digital interfaces (e.g., UART or I2C). Set up the STM32 microcontroller's ADC module to read analogue sensor data from the GP2Y1010AU0F and MQ-135 sensors. To translate the unprocessed sensor signals into usable values, such as gas concentrations and PM concentrations, use the calibration data or formulas specified in the sensor datasheets. Use data filtering techniques to eliminate noise or outliers from the sensor measurements, if necessary. To process the processed sensor data further, store or aggregate it in variables or data structures. Use a LoRaWAN transceiver or module that is compatible with the STM32 microcontroller. To handle the communication protocol, implement the required

LoRaWAN stack or library. Set the proper device credentials (such as the device EUI, application EUI, and application key) in the LoRaWAN module. Send the sensor data that has been processed to a LoRaWAN gateway or network server using the LoRaWAN library functions. To shield the electronics from the elements, design or choose an appropriate enclosure. Ensure adequate airflow so that the sensors can accurately sample the air. Install the equipment where you want to monitor the air quality. Create a LoRaWAN gateway and network server on the receiving end to accept sensor data. Created a mobile-based application to display and analyze the data on the air quality received. Create appropriate algorithms or models to analyze the data and deliver insightful information, such as air quality indexes. Calibration is to achieve precise and trustworthy data; it is critical to calibrate the sensors regularly. The calibration processes listed in the sensor datasheets or the suggested calibration methods for each type of sensor should be followed. Whereas Validation includes Perform validation tests by contrasting sensor values with benchmark readings from approved air quality monitoring apparatus. This process aids in confirming the precision and consistency of the sensor data. To establish connection between the STM32 microcontroller and the LoRaWAN gateway or network server, implement LoRaWAN protocols and libraries. microcontroller, LoRaWAN module, and other peripherals. For long-term data storage, analysis, and visualization, integrate the air quality monitoring system with cloud computing platforms or data storage solutions. Use the APIs or protocols that cloud computing platforms offer to securely send and store sensor data. Use data processing techniques to perform advanced analytics or produce real-time warnings depending on air quality thresholds on the cloud server or edge devices. Also, 3.GPS systems were installed in the stm32 chip to locate the defected node and to fix the bug.

4.1.4 Circuit with STM32 LORAWAN:

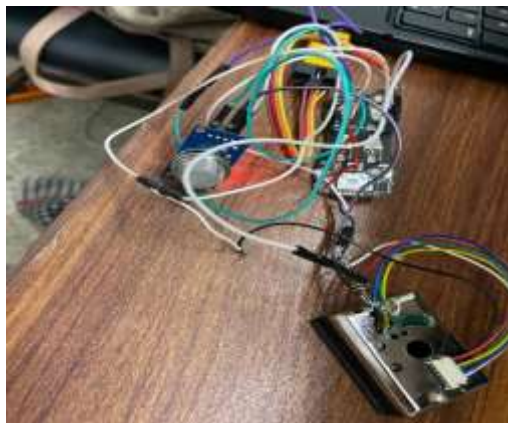


Figure 24: Circuit with stm-32 LoRaWAN

4.1.5 Helium Gateway:

Real-time visualization of the data, which was safely saved in the cloud, gave insightful details on the local air quality. Additionally, the Helium gateway and AWS

offer a scalable system that supports several devices and enormous amounts of data. The function of Helium Gateway in this project is as follows. The local LoRaWAN network and the decentralized Helium blockchain-based LoRaWAN network were connected through a Helium gateway. It enables wireless long-distance transmission of sensor data to the Helium network infrastructure. The local air quality monitoring system sends sensor data to the Helium gateway, which aggregates it before sending it to the Helium blockchain network. It acts as a central communication hub for numerous devices, ensuring the effective collection and transmission of data from various sensors. The local air quality monitoring system's sensor data is transformed by the Helium gateway into the correct format needed by the LoRaWAN protocol of the Helium network. It secures the transmission of the data to the Helium network servers by encapsulating it in LoRaWAN packets. The sensor data is transmitted securely and reliably to the Helium network servers through the Helium gateway. Utilizing LoRaWAN security features, it encrypts the data to guard against hacking and alteration of the sensor data while it is being transmitted. The local air quality monitoring system was connected to the Helium network through the Helium gateway. For the sensor data to be received by Helium network servers for additional processing and storage, a link must be established with the infrastructure of the Helium network. The Helium gateway keeps the servers of the Helium network coordinated, ensuring the timely and correct transfer of sensor data. To plan transmissions and maximize network resources, it works in tandem with the network servers. The ability to operate a decentralized network enables the distribution of data processing jobs across many network nodes. This implies that any node in your project, like a 3.STM32 microcontroller, can locally process air quality sensor data before sending it to the Helium gateway. With the help of distributed processing, the workload placed on a single central node is lessened, and data analysis and decision-making can be completed more quickly. With decentralized network operation, other nodes in the network can continue operating independently if one node fails or goes offline. This fault tolerance ensures that the system keeps working even if a specific STM32 microcontroller or a Helium gateway has problems. It increases the accuracy of data transmission and gathering for the project to monitor air quality.

4.1.6 AWS:

The STM32 LoRaWAN device sends data to the AWS platform, which provides cloud-based infrastructure for storing and analyzing that data. Using HTTPS protocols, the data obtained from the Helium gateway was safely sent to AWS. When the data was finally received, it was stored in dynamo db. and processed and visualized using AWS service AWS Lambda. The air quality monitoring system's sensor data can be stored using AWS's storage services, such as Amazon S3 (Simple Storage Service). The sensor data can be safely kept in the cloud for quick access and long-term storage. Several data processing and analytics services are provided by AWS that can be used to handle sensor data, analyze data, produce insights, and visualize trends and patterns in air quality. You can manage real-time data streaming from the air quality monitoring device using AWS. For real-time processing, the sensor data can be fed into Kinesis streams, allowing for instant analysis, or setting off warnings based

on predetermined thresholds. Sensor data from the air quality monitoring system may be managed and processed using AWS IoT services. These services make it possible to create IoT-based applications and automation by enabling secure device connectivity, data intake, and rule-based processing. Create dynamic dashboards and visualizations using the data from air quality sensors and can be visualized through app services to understand and present for monitoring and analysis. Large volumes of sensor data can be handled by the highly scalable and dependable cloud architecture offered by AWS. The project used AWS services' scalability to prepare for potential future increases in the number of sensors, data storage needs, and processing resources.

4.1.7 Mobile App:

After that with API Gateway the results were displayed on mobile App using flutter. Which will separately display all the gases as well. The data on air quality gathered by the sensors may be monitored in real-time using the mobile app. On their mobile devices, users can easily view the current gas concentrations, PM levels, and other pertinent parameters. They can remain informed about the local air quality thanks to this.

The smartphone app has access to and can provide historical sensor-collected data on air quality. The app may show information about air quality on a map by utilizing the geolocation features of mobile devices. Users can identify places with high or low pollution levels and visualize the air quality in various locations. Users can plan their travels or select healthier environments with mapping features. Based on established air quality limits, the mobile app can send users customized alerts and notifications. The software can alert users to take proper actions, such forgoing outside activities or donning masks, when the air quality reaches harmful levels. This supports user security and welfare. The smartphone app can display air quality data in an easy-to-understand and aesthetically pleasing way. Users may compare various contaminants, evaluate air quality trends, and gain insights into the data by using charts, graphs, and visualizations. Users' comprehension and engagement were increased by clear visual representations. Users may be able to comment on air quality data and report any problems or inconsistencies using the mobile app's features. Users can contribute to improving the system's overall accuracy and dependability by rating the local air quality conditions, leaving comments, and providing feedback. A sample of the created app's AQI value and other features are shown in figure 25.

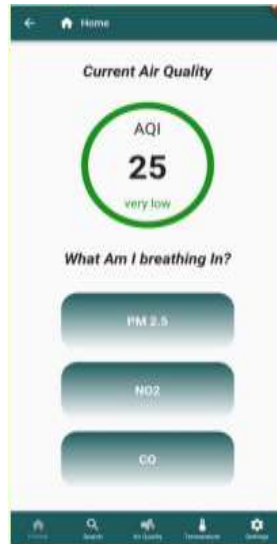


Figure 25: Airify App

4.2 Algorithm:

Certainly! Here are some algorithms that are implemented in this project:

1. Gas Concentration Estimation:

- Algorithm: Calculate gas concentrations using sensor data from the MQ-135. Utilize the sensor datasheet's lookup tables or calibration methods to translate raw sensor results into ppm (parts per million) gas concentrations.
- Implementation: To determine the gas concentrations based on sensor readings, implement the calibration equations or interpolation algorithms in the firmware. For additional investigation, this data can be transferred over LoRaWAN.

2. Particulate Matter (PM) Estimation:

- Algorithm: Calculate PM concentrations using sensor data from the GP2Y1010AU0F. Based on the sensor calibration and sensitivity parameters, the raw sensor translated measurements into relevant PM values.
- Implementation: To convert the sensor output to PM concentrations, use the calibration equations or calibration curves supplied by the sensor manufacturer. For additional investigation, this processed data was transmitted through LoRaWAN.

3. Air Quality Index (AQI) Calculation:

- Algorithm: Based on the recorded gas concentrations and PM levels, calculate the overall air quality index using a specified algorithm, Air Quality Index computation.
- Implementation: By translating the recorded gas concentrations and PM levels to the corresponding AQI ranges, the AQI calculation algorithm is implemented in the firmware. To monitor and visualize, send the determined AQI value over LoRaWAN.

4. Threshold Monitoring and Alerting:

- Set predetermined threshold values for gas concentrations, PM levels, or AQI categories using an algorithm. Monitor the observed values over time and send out alerts when they go above certain criteria.
- Implementation: Contrast the in-the-moment sensor readings with the firmware's predefined threshold values. Send an alarm or notification to users or administrators via the LoRaWAN network or local interface if any parameter crosses the cutoff to let them know about the bad air quality situation.

5. Data Aggregation and Statistical Analysis:

- Algorithm: Computed statistical measures like averages, minimums, and maximums, as well as standard deviations, using the sensor data aggregated over predetermined time intervals (e.g., hourly, daily).
- Implementation: Gather sensor data over the specified time intervals, then run statistical analyses on the resulting data. This data can be used to analyze trends or create historical air quality reports.

4.3 Flowchart of code and output:

Flow chart of the conditions in the code and its desired output is:

```

22:22:27.926 -> PM2.5: 199.80 µg/m³
22:22:30.942 -> PM2.5: 13.09 µg/m³
, CO2: 116.75, NO2: 1023.00,
CO2: 8.96, NO2: 222.00, AQI:
3.00, AQI: 326.16, CAQI: 260
, AQI: 218.12, CAQI: 174

```

Figure 26: Output of Arduino Code

4.4 Flow Chart of Code:

The Flow Chart below shows the complete work done in the code. Starting from collecting data from sensors to calculating the value of CAQI.

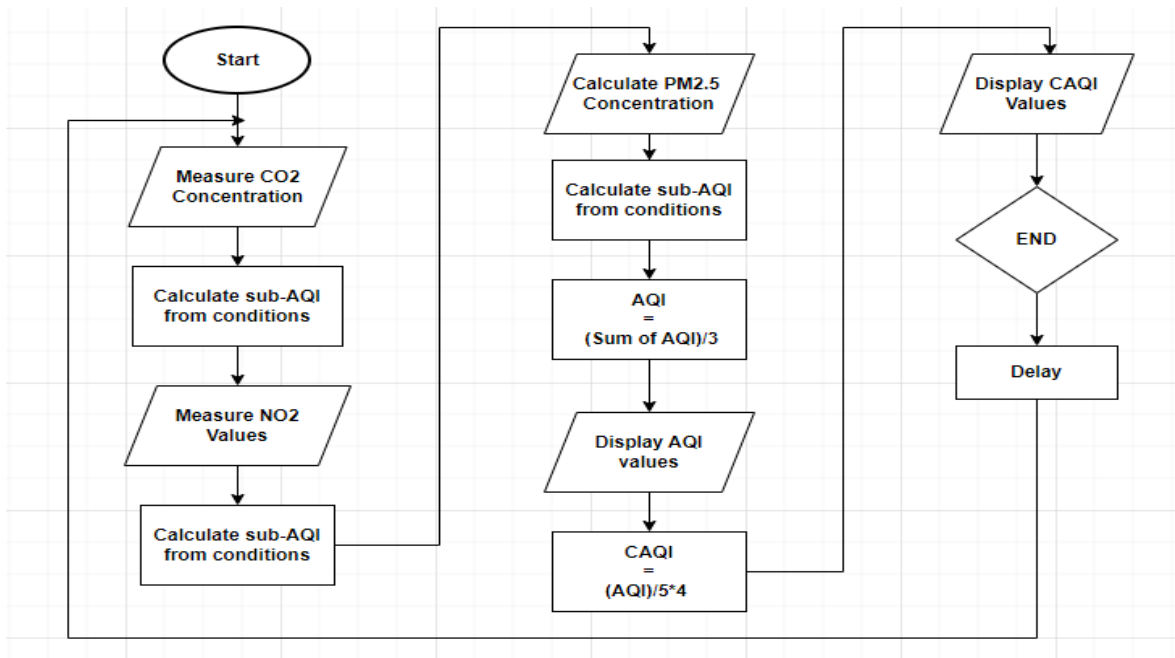


Figure 27 :Flow Diagram to show data on Application.

4.5 Graph:

The graph below highlights the performance of air quality calculated by sensors in a particular area and how much the value fluctuates over the period. Figure 28 shows the graph showing IoT based air Quality and particulate matter concentration monitoring system.

Result of MQ-135 channel:

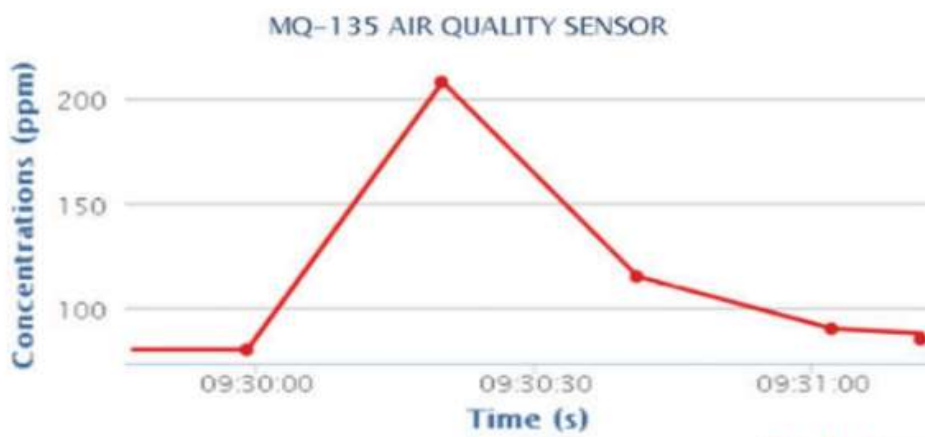


Figure 28 :Graphic Display of MQ-135



Figure 29: Graphic Display of AQI with respect to time

Chapter 05

Chapter 05: AWS and HELIUM INTEGRATION

Integration is the process of data collection from serverless, distributed system of decoupled components and allowing them to be communicated over network for combining them into a single view or result. In this chapter, we explained about the most important aspect of our project i.e., Integration. This covers the integration of Lora WAN with Helium, Helium with AWS DynamoDB table and then with Flutter application for the visualization. Starting with creation of AWS account each part of integration is explained in detail.

5.1 Creating AWS Account

Amazon Web Services is a sub element of AMAZON which provides user with a pay-as-you-go list of services that could easily be scaled up or down based upon the needs. Users such as large multinational companies, government as well as individuals, can have unlimited access to various cloud computing platforms and APIs and pay only for the services they are using.

The main advantage of using AWS is that one does not need to worry about the storage, operating system, or the backend server maintenance because AWS handles everything and provides infinite storage for data storage and processing.

In our project, we needed a platform that could fetch AQI values from Helium console using HTTP request and store them in a database for further visualization on the Flutter application. For implementing all this we first created an account of AWS on student access.



Figure 30: AWS Login Console

We signed into the AWS console using a separate mail created solely for the purpose of Air Quality Monitoring project management.

Root user sign is required when we are performing administrative tasks such as changing root password, changing billing information etc, but for performing routine tasks such as creating a database or a lambda function. In the Figure 31, we are signing into the AWS console as a root user.



Sign in

Root user
Account owner that performs tasks requiring unrestricted access. [Learn more](#)

IAM user
User within an account that performs daily tasks. [Learn more](#)

Root user email address

appairify@gmail.com

Next

Figure 31: Signing into the AWS Console as a root user.

5.2 DynamoDB Creation

From the AWS console proceed to the DynamoDB and select the option of create table. Then you will be asked to enter certain details of the table such as table name, partition key, sort key etc. After providing all information you create the table and instantly it will be shown on the tables list.

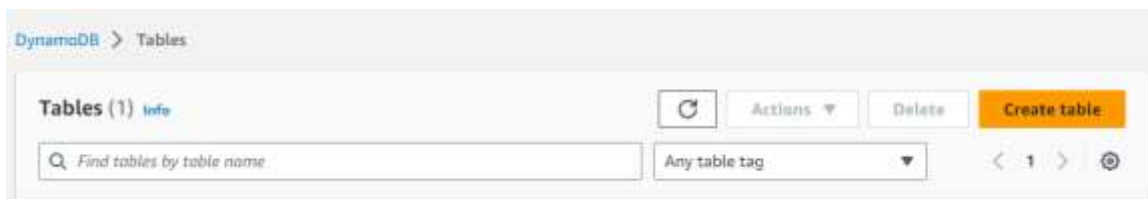


Figure 32: Creating table.

The name of our table is 'AQI_table' and the partition key is 'table key'. The main purpose of a partition key is that it serves as a part of a table's primary key and is used to retrieve items from your table or allocate data across hosts for scalability and accessibility.

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (`_`), hyphens (`-`), and periods (`.`).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

1 to 255 characters and case sensitive.

Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

1 to 255 characters and case sensitive.

Figure 33: Table Info

After selecting the option of Create Table shown in Figure 32, we proceed to the options shown in Figure 33 where we enter the name of our table and a partition key. In our case, we named it 'AQI_table'.

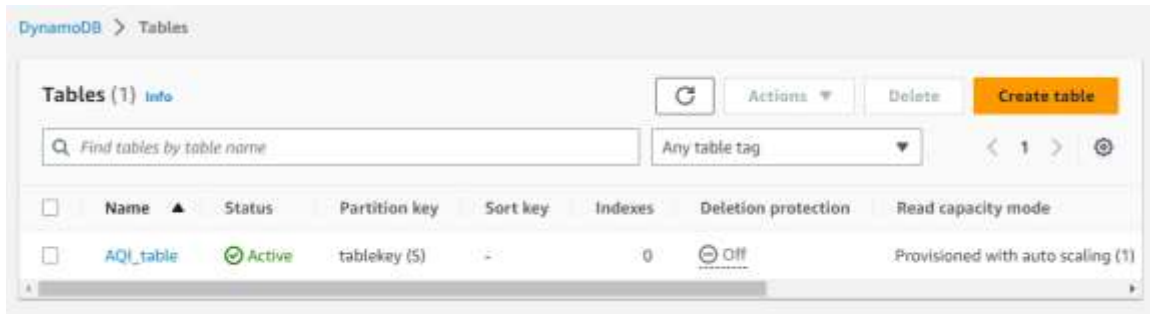


Figure 34: Table AQI created.

Once we have given the details of our table we selected the Save table option and our table is created. Figure 34 shows the details of our AQI_table with its status as 'active', and its partition key as 'tablekey'.

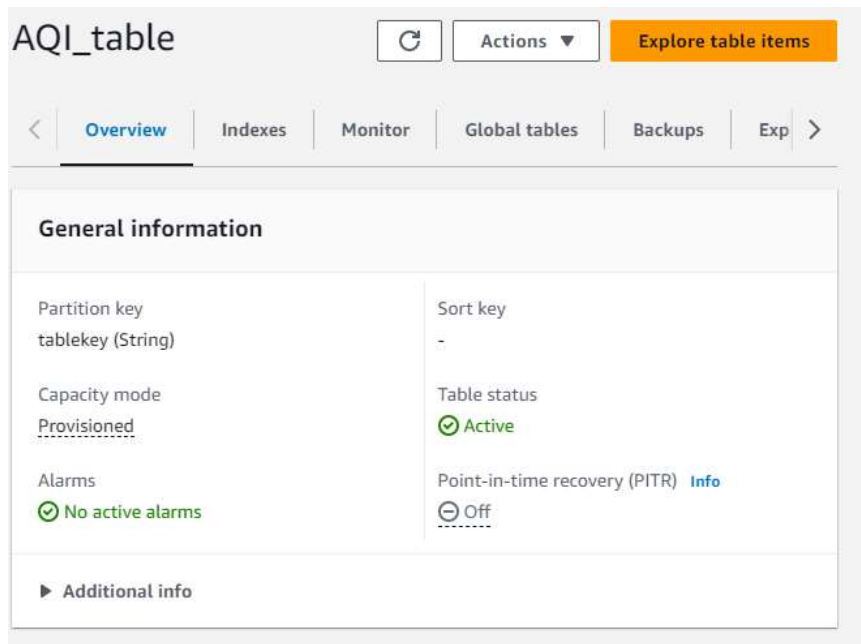


Figure 35: Overview of table

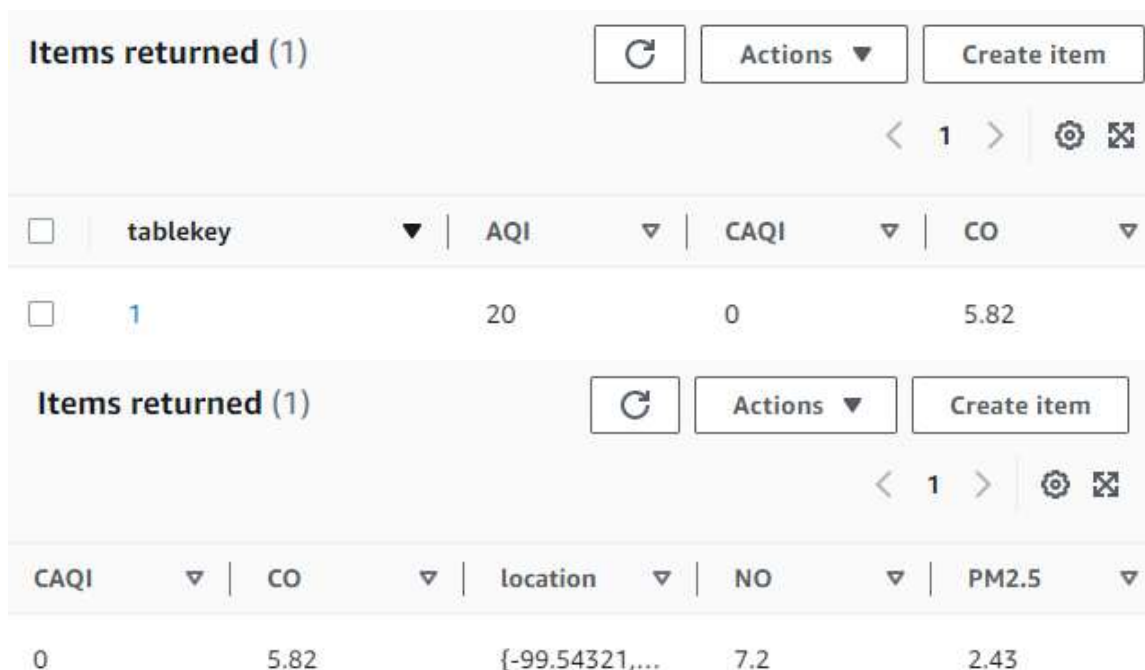


Figure 36: Table Items

When we have successfully created the table, several options are available to see further details about it such as actions, explore table items. Selecting the explore table items will show another screen in which we are displayed the items we created like AQI, NO₂, CO, location as seen in Figure 35 and Figure 36.

5.3 HTTP Integration

Helium console API is a set of HTTP requests that allow your devices monitored by Helium Console to interact with any other platform for example AWS and is ideal for

integrating with backend devices, just as explained in *Helium Console API / Helium Documentation* (n.d.) [19](#).

To retrieve a Helium API, we go to console, My Account and then to ‘Your API Keys’ section as shown in Figure 37, 38.

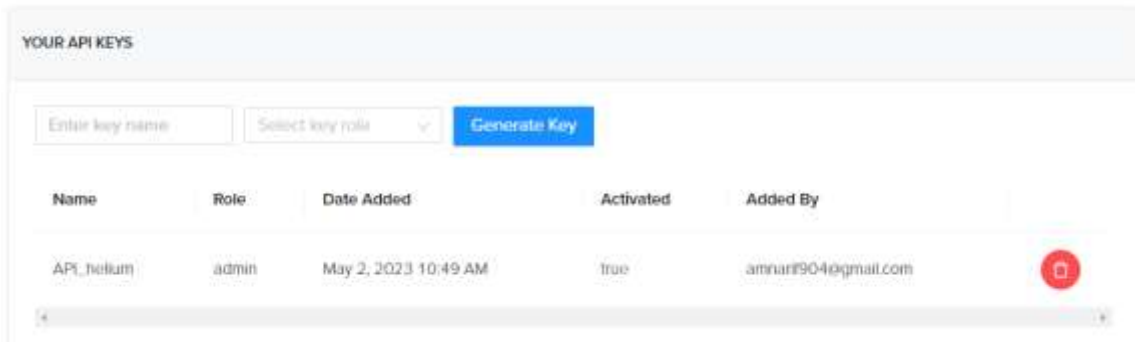


Figure 37: Generating API keys.

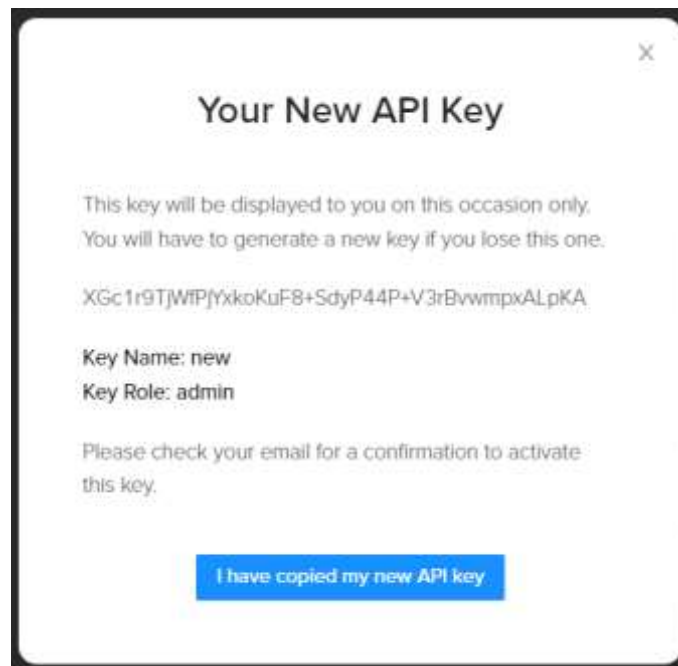


Figure 38: Copy Assigned API key.

5.3.1 Set up an HTTP Endpoint

After getting Helium API key, we need to set up an HTTP endpoint to retrieve AQI values from Helium and then store them in the DynamoDB table we created earlier. In our project we are using AWS Lambda function as our HTTP endpoint to do this. The lambda function we created takes HTTP requests that are carrying AQI values and stores them in DynamoDB table.

5.3.2 Connect STM32 to an HTTP Endpoint

We connected STM32 device created in Helium Console with an HTTP integration just as shown in the following Figure 39, 40, 41.



Figure 39: AQI end node added on Helium Console

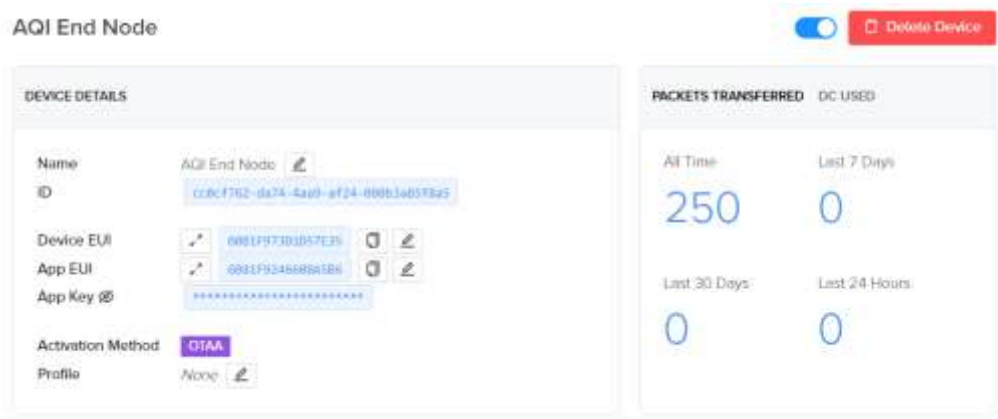


Figure 40: AQI end node details

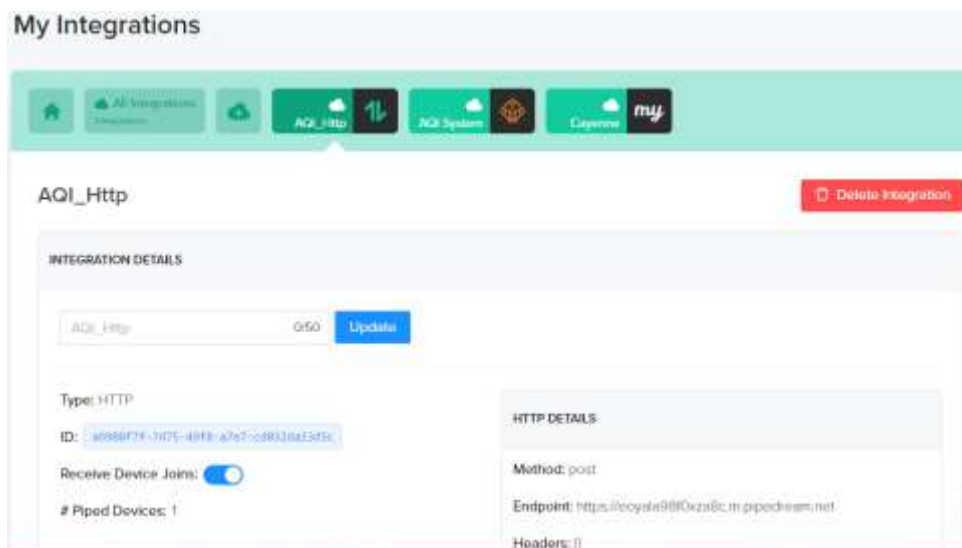


Figure 41: AQI HTTP integration

In the above figures, we showed the created AQI end node device and the HTTP integration on the Helium console. After that we connected both the device and integration with each other using the Flows in Helium shown in Figure 42. Flows is a view of the components in Helium and to comprehend the connection among devices, functions, and integrations. It visually connects nodes and controls the flow of data.



Figure 42: Helium Flows

5.3.3 Store AQI data in DynamoDB table

In our lambda function we will write the code to parse the AQI data from the coming HTTP request and the data is written in the DynamoDB table.

After parsing the data from payload request, we create a list of items in our table such as NO₂, CO, PM_{2.5}, location of the node (the exact coordinates of the place we are calculating the AQI of) and the AQI value, already shown in Figure 24.

The screenshot shows the 'Items returned (1)' section of a DynamoDB table. It includes a refresh button, an 'Actions' dropdown, and a 'Create item' button. Below these are navigation controls showing '1' item. The table has five columns: CAQI, CO, location, NO, and PM2.5. The first row contains the values: 0, 5.82, {-99.54321,...}, 7.2, and 2.43.

CAQI	CO	location	NO	PM2.5
0	5.82	{-99.54321,...	7.2	2.43

Figure 43: Table items

5.4 Lambda Function

The most brilliant concept of going ‘Serverless’ came into reality when AWS lambda was launched. It helps the developers to write any function or code to manage any type of application without having to worry about the backend or the server-side infrastructure.

It can be programmed to start execution only on a trigger event such as on an HTTP request, change in the database, emails from customers or on user authentication and we only must pay for when the lambda function runs.

The biggest advantage of Lambda function is that it reduces the time that would probably be spent on the underlying infrastructure management, saves both effort and time. In addition to this it also enables building easy scalable solution to any project.

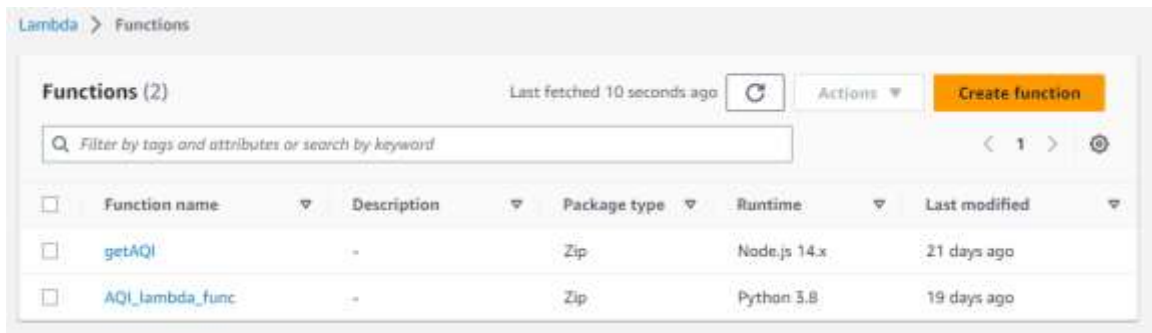


Figure 44: Lambda Function created for AQI value storage.

In the above Figure 44, we created two lambda functions; one for getting HTTP requests from Helium Console and storing data (AQI values) in DynamoDB and second for extracting the AQI values from the same DynamoDB table and sending them to Flutter application using HTTP gateway. An overview of the first Lambda function is shown in Figure 45.

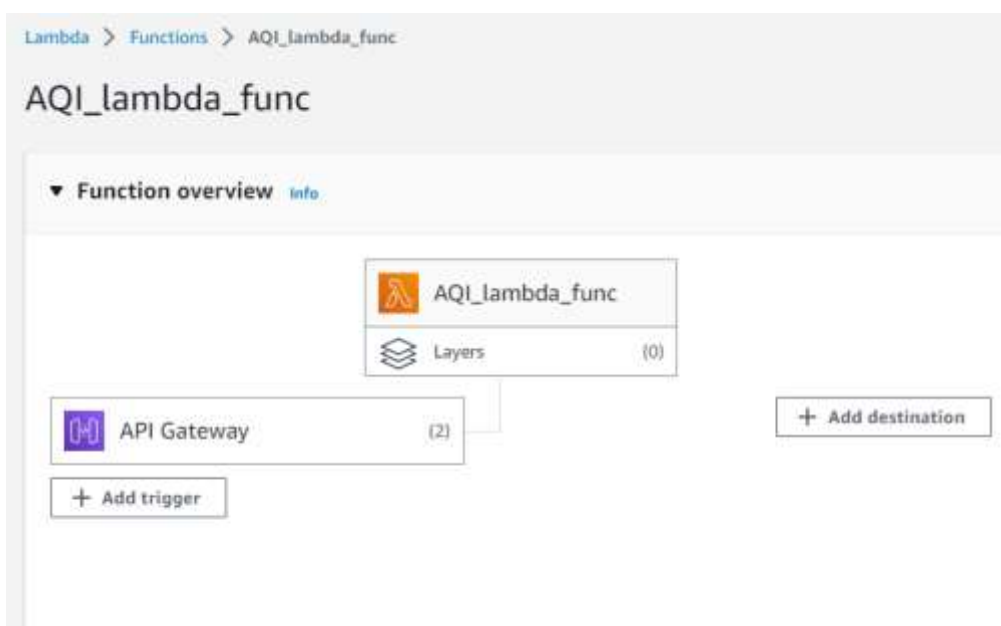


Figure 45: AQI Lambda function overview

5.4.1 Addition of Trigger Events

We created an API gateway REST API which is mainly created to trigger the Lambda Function for storing values. This REST API triggers the function when it receives an HTTP request from Helium.

After that we deployed the API to make it publicly accessible and tested it to make sure that it was working correctly and smoothly. The API Gateway triggers are shown in Figure 46 and Figure 47.

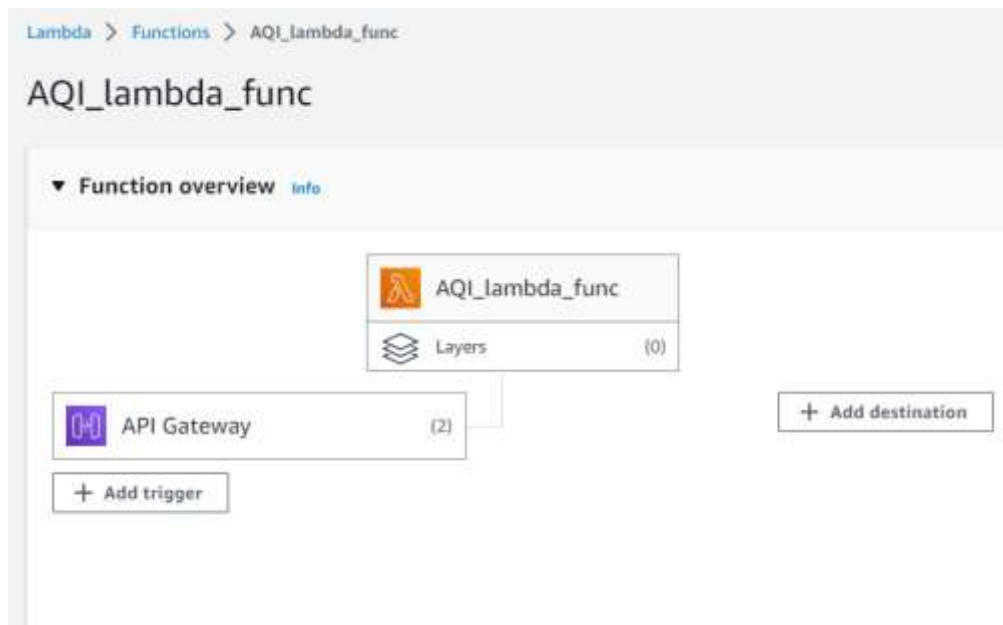


Figure 46: API Gateway connected to Lambda.

5.5 API Gateways

In the lambda functions that we created; we now need to add the API key of Helium Console that would send HTTP requests to the AWS console. Following actions are taken to add a Helium API as a trigger for our Lambda function:

Go to the Lambda service in the AWS Management Console after opening it. Choose the Lambda function to which a trigger should be added. Choose "API Gateway" as the trigger type by clicking the "Add Trigger" button. Choose "REST API" as the API type and "OpenAPI 3.0" as the protocol in the "Create API" section. Enter a name for your API in the "API name" area, and depending on your needs, choose "Regional" or "Edge optimised" for the "Endpoint Type" option. Select the proper authentication method for your API in the "Security" section. Add the route(s) that correspond to the endpoint(s) of your Helium API to the "Routes" section. Choose the Lambda function

you wish to run whenever an API endpoint request is received from the "Configure triggers" section. To make your API live, save your settings and deploy it.

Similarly, to fetch the AQI data from the DynamoDB table and show it on the Flutter Application, we create another lambda function same as above but replace the Helium API with the API key of our Flutter App.



Figure 47: Gateway having Helium API with API endpoint.



Figure 48: API Gateway Helium API

Chapter 06

Chapter 06: Software Application

6.1 Overview

In this project we have used applications like flutter, AWS, Helium Console and STM32CubeMX, STM32CubeIDE. These are the software that we have used because of the need of project. i.e., the need of using flutter was to develop a software to display all the information and to provide it to the customers so they can use that app to get Air Quality Information, AWS is used to store Information and to create a link between Helium Console and Flutter Application. First DynamoDB table is created and then lambda function is created then for flutter application and for helium console separately. Flutter one is used to get values from Dynamo DB table and then by http request those values are fetched in class of DynamoDB table and second one is used to connect AWS with Helium Console So, it gets real time data from LoRaWAN (STM-32 integrated) devices and then store into that DynamoDB table. Then comes helium console it basically is used to transfer the Air Quality Information that we are getting from STM32 to the Internet and is connected to AWS through API keys to it. And STM32CubeMX, STM32CubeIDE are used for configuration of pins and for the generation and compilation of code for the LoRaWAN (STM32 integrated) device. Following is the class diagram for the description of above lines diagrammatically to make it easier to understand technically.

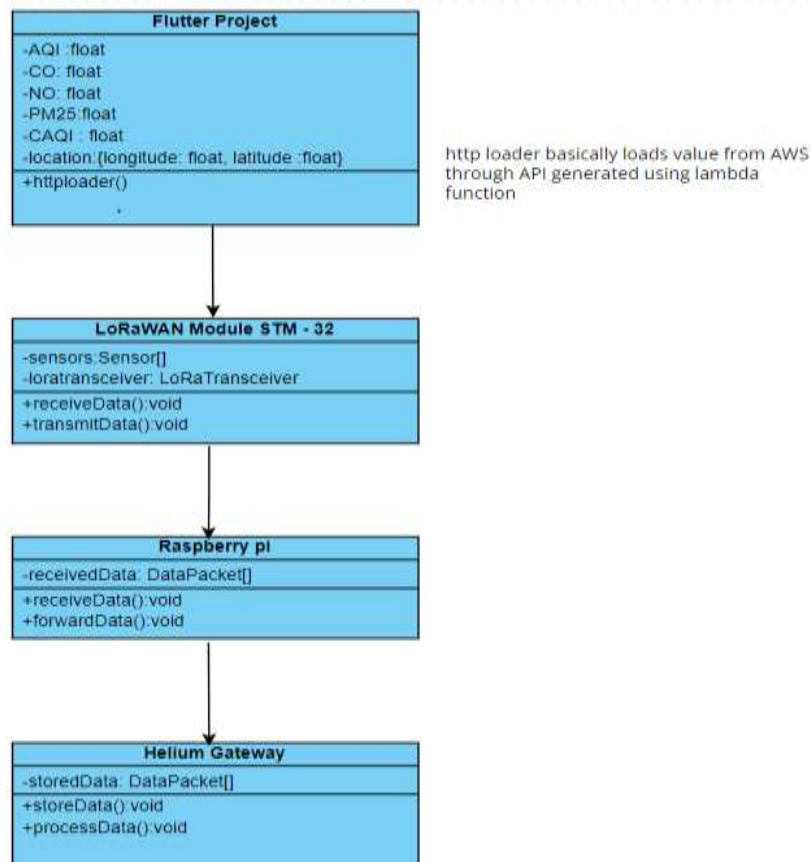


Figure 49: Project Class Diagram

6.2 Flutter Application

We have used flutter for development of our application as it easier to use, flexibility and it supports iteratively seeing the development throughout the whole project. And hence we have used and developed our app using dart as programming language. And the application developed is shown by the following screenshots we took while debugging it using USB cable.



Figure 50: Search of Application

It is the first screen that comes after the Log in screen and for this Page's Development we have used Google Map's API for this purpose so as soon as we search for some location it will give the value of AQI for that searched location but if we have no node for searched location it will give the nearest location value of Air Quality Index and concentrations of different gases that are used to calculate Air Quality Index. And when a location is searched the AQI (Air Quality Index). It goes to the home page hence displaying the Air Quality Index along with concentrations also we tend to make a dashboard which displays graphs of concentrations along with the time. The home page is shown in figure 51.



Figure 51: Home of Application

And then we have Air Quality Page if the User wants to know suggestions according to the Air Quality Index. Also, then there is a settings page which allows us to change the settings of our app according to our needs and then and if we want to have different suggestions and warnings, we can turn on notifications of the application by selecting the list item named notification and then going to that Notifications Page. These pages are shown in figure 52.

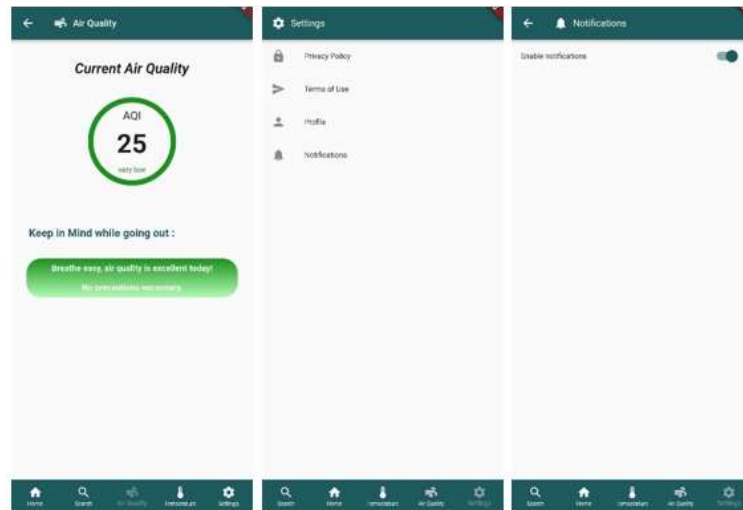


Figure 52: Air Quality, Settings, and Notifications of Application

And then going back to settings Page we have Privacy policy page which has a weblink in it and that link contains privacy policy of that application. And then we have Terms of Use page which contains text that is related to that page and will be shown in figure below. And a Profile page which is another page with a list of other pages.

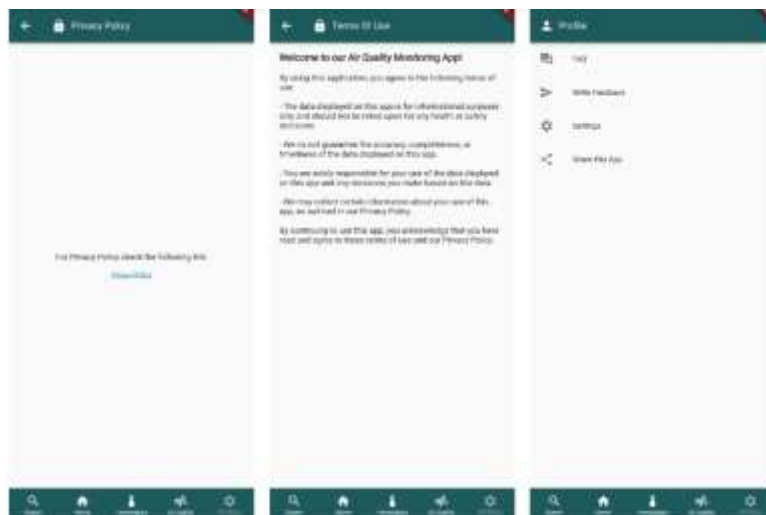


Figure 53: Privacy policy, Terms of Use and Profile pages of Application

As shown from the Figure 53 the profile consists of another list which contains items of pages FAQ, write feedback and again the settings page and the share app page which generates link to

the application to share it. FAQ page consists of the frequently Asked questions that could be asked, and we will add more or remove according to the questions asked through feedback or other sources.

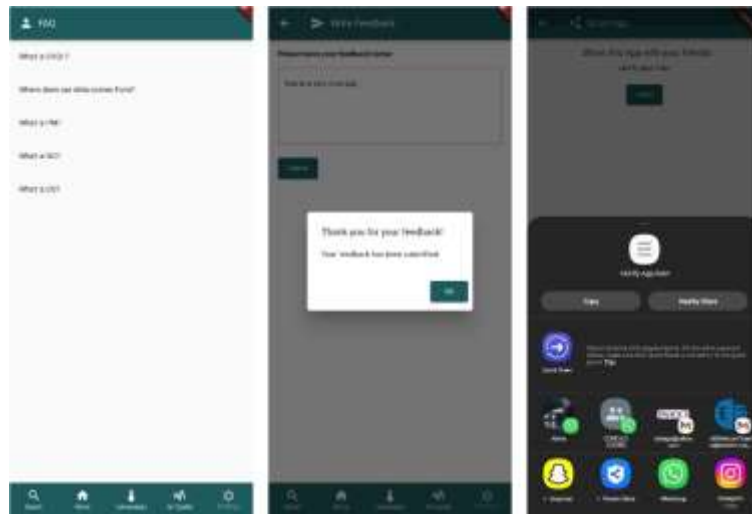


Figure 54: FAQ, Feedback and Share app pages of Application

FAQ page consists of a list of Questions, and they refer to other pages that are linked through that page, these questions include What is CAQI, where does our data comes from, what is PM, what is NO and lastly is CO and they are shown below:



Figure 55: FAQ pages of Application

The above-described flow of pages is shown from the figure 56:

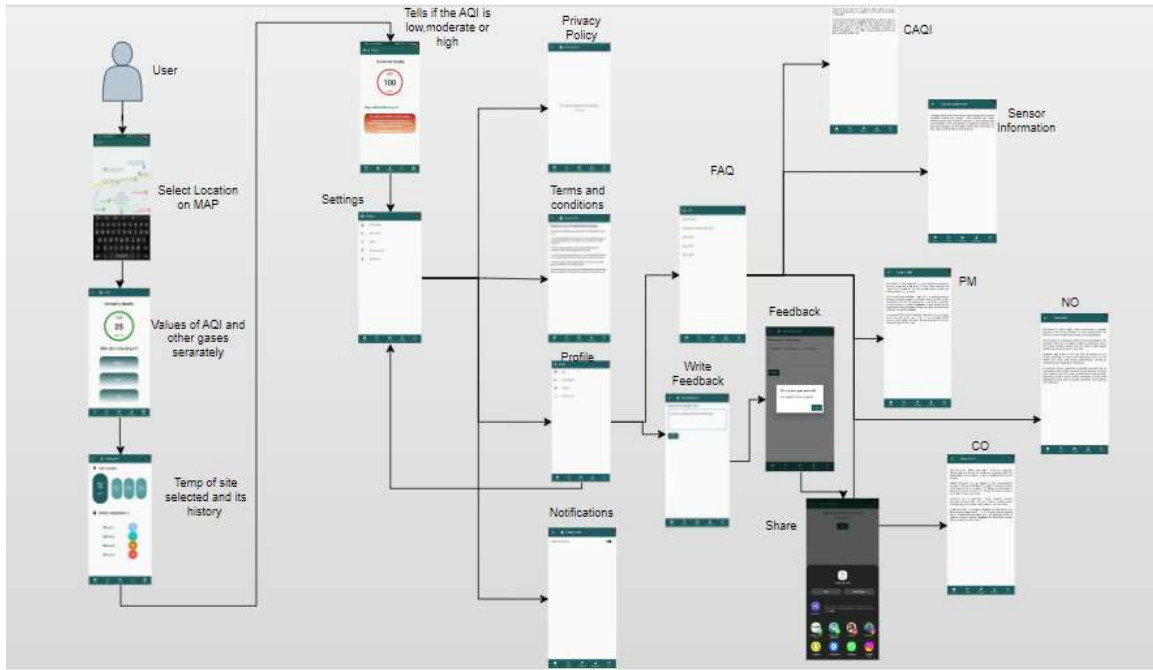


Figure 56: User Flow Diagram

And the Use case diagram of our application is shown below which shows the use cases of our project:

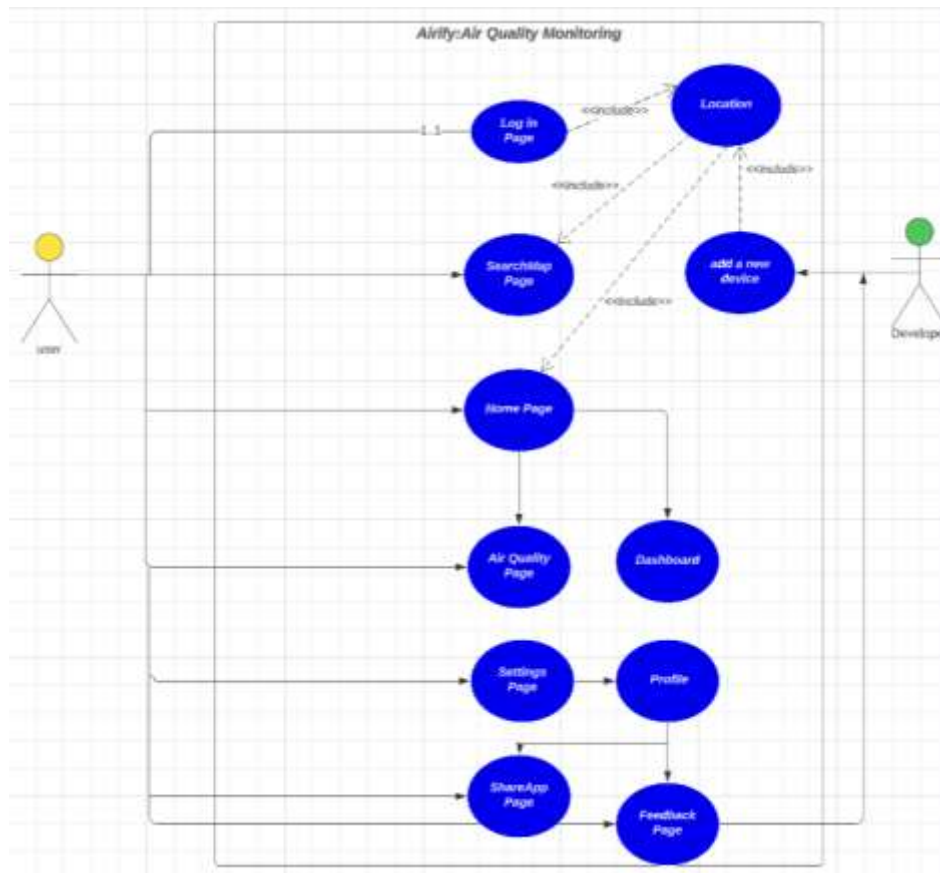


Figure 57: Use case diagram

Chapter 07

Chapter 7: Conclusion and Future Work

7.1 Conclusion

As a result, the "Air Quality Monitoring" project, which makes use of an STM32 microcontroller and a Helium gateway, provides an all-inclusive approach to collecting and analyzing data on air quality. The project offers effective data collecting, transmission, and processing by fusing the capabilities of the STM32 microcontroller with the Helium gateway. The following main ideas will serve as a project summary:

The STM32 microcontroller is fitted with air quality sensors that record pertinent information, including pollution levels (CO, NO₂, PM_{2.5}) These sensors offer precise and immediate air quality assessments. STM32 microcontrollers send data to the Helium gateway, which acts as a central hub. It gathers the air quality data and creates a wireless communication channel with the STM32 nodes for later processing and analysis. The gathered air quality data can be processed and analyzed either locally on the STM32 microcontroller or remotely via the Helium gateway to a server or cloud platform. This makes it possible to analyze, visualize, and store data for a long time.

There are several potential applications and effects for the "Air Quality Monitoring" project. It can be used to monitor air quality levels, identify pollution sources, evaluate health hazards, and support environmental decision-making in metropolitan areas, industrial zones, or sensitive regions. The project may aid in establishing sustainable and healthy living conditions.

Overall, the "Air Quality Monitoring" project, which makes use of an STM32 microcontroller and a Helium gateway, offers a practical and expandable method of air quality monitoring. It combines the strength of decentralized network operation, sensor technology, and data analysis to supply insightful information on the state of air quality and support initiatives aimed at improving environmental management.

7.2 Future Work

Here are some potential plans and expansions for the "Air Quality Monitoring" project:

The possibilities of adding more sensors or more sophisticated sensor technologies to gather more thorough data on air quality. Sensors for pollutants, particulate matter, volatile organic compounds (VOCs), or gas sensors for detecting certain gases of interest could fall under this category. Increase the project's geographic reach by setting up additional STM32 nodes and Helium gateways to track the quality of the air in various areas. This growth can entail collaborating with

regional leaders, businesses, or communities to create a larger network of observation points. To guarantee the dependability, accuracy, and effectiveness of the air quality monitoring system, the hardware and software components of the project must be updated and improved regularly.

However, we can also improve mobile app features as well. For instance:

Provide individualized health advice based on the state of the air. The software can recommend actions like staying indoors during times of high pollution, donning masks, or choosing other routes with cleaner air. Enable users to use the mobile app to share their air quality observations and reports. The accuracy and scope of the data on air quality can be improved by using user-generated information, which can also encourage community involvement in the fight against air pollution. To give customers access to real-time air quality updates directly on their wearable devices, integrate the mobile app with wearable gadgets like smartwatches or fitness trackers. This enables easy mobile access to information about the quality of the air. By pursuing these plans, the "Air Quality Monitoring" project can evolve into a comprehensive and impactful initiative, contributing to better air quality management, public health, and environmental sustainability.

REFERENCES

- [1] *Where Does Air Pollution Come From? - Air* (U.S. National Park Service). (2018, January 17). Wwww.nps.gov. <https://www.nps.gov/subjects/air/sources.htm#:~:text=mobile%20sources%20>
- [2] Rutledge, K. (2022, July 1). *Air Pollution*. Education.nationalgeographic.org; National Geographic. <https://education.nationalgeographic.org/resource/air-pollution/>
- [3] *AWS Lambda Documentation*. (2022). Amazon.com. https://docs.aws.amazon.com/lambda/?icmpid=docs_homepage_compute
- [4] *Fog computing vs. cloud computing - javatpoint*. (n.d.). Wwww.javatpoint.com. Retrieved May 13, 2023, from <https://www.javatpoint.com/fog-computing-vs-cloud-computing#:~:text=The%20main%20difference%20between%20fog>
- [5] *Optical dust sensor - gp2y1010au0f - COM-09689 - sparkfun electronics*. (n.d.). Wwww.sparkfun.com. <https://www.sparkfun.com/products/9689>
- [6] Popović, I., Radovanovic, I., Vajs, I., Drajić, D., & Gligorić, N. (2022). Building low-cost sensing infrastructure for air quality monitoring in urban areas based on fog computing. *Sensors*, 22(3), 1026. <https://doi.org/10.3390/s22031026>
- [7] Kumar, P.; Morawska, L.; Martani, C.; Biskos, G.; Neophytou, M.; Di Sabatino, S.; Bell, M.; Norford, L.; Britter, L. The rise of low-cost sensing for managing air pollution in cities. *Environ. Int.* 2015, 75, 199–205. [CrossRef] [PubMed]
- [8] *World live air quality map | airvisual*. (n.d.). Wwww.iqair.com. Retrieved May 16, 2023, from <https://www.iqair.com/world-air-quality>
- [9] *PurpleAir Flex Air Quality Monitor*. (n.d.). PurpleAir, Inc. Retrieved May 18, 2023, from <https://www2.purpleair.com/products/purpleair-flex>
- [10] *Air quality sensor: Series 500 portable air monitor*. (n.d.). Wwww.aeroqual.com. Retrieved May 16, 2023, from <https://www.aeroqual.com/products/s-series-portable-air-monitors/series-500-portable-air-pollution-monitor>
- [11] Srivastava, H.; Bansal, K.; Kumar Das, S.; Sarkar, S. An Efficient IoT Technology Cloud-Based Pollution Monitoring System. In *Advances in Systems, Control and Automations; Lecture Notes in Electrical Engineering*; Bhoi, A.K., Mallick, P.K., Balas, V.E., Mishra, B.S.P., Eds.; Springer: Singapore, 2021; Volume 708, pp. 109–120.
- [12] Zhang, D., & Woo, S. S. (2020). Real Time Localized Air Quality Monitoring and Prediction Through Mobile and Fixed IoT Sensing Network. *IEEE Access*, 8, 89584–89594. <https://doi.org/10.1109/ACCESS.2020.2993547>

- [13] Motlagh, N. H., Lagerspetz, E., Nurmi, P., Li, X., Varjonen, S., Mineraud, J., ... & Tarkoma, S. (2020). Toward massive scale air quality monitoring. *IEEE Communications Magazine*, 58(2), 54-59
- [14] Bharathi, P. D., Ananthanarayanan, V., & Bagavathi Sivakumar, P. (2020). Fog computing-based environmental monitoring using nordic thingy: 52 and raspberry Pi. In *Smart Systems and IoT: Innovations in Computing: Proceeding of SSIC 2019* (pp. 269-279). Springer Singapore. [Fog Computing-Based Environmental Monitoring Using Nordic Thingy: 52 and Raspberry Pi | SpringerLink](#)
- [15] Kelechi, A. H., Alsharif, M. H., Agbaetuo, C., Ubadike, O., Aligbe, A., Uthansakul, P., ... & Aly, A. A. (2022). Design of a low-cost air quality monitoring system using Arduino and ThingSpeak. *Comput. Mater. Contin*, 70, 151-169.
- [16] Zhao, Z., Wang, J., Fu, C., Liu, Z., Liu, D., & Li, B. (2018). Design of a smart sensor network system for real-time air quality monitoring on green roof. *Journal of Sensors*, 2018.
- [17] Li, D., Wu, T., Li, X., He, Q., & Cui, Z. (2020). A Wireless Multisensor Node for Long-Term Environmental Parameters Monitoring. *Journal of Electrical and Computer Engineering*, 2020, 1-12.
- [18] Kaivonen, S., & Ngai, E. C. H. (2020). Real-time air pollution monitoring with sensors on city. *Digital Communications and Networks*, 6(1), 23-30.
- [19] *Helium Console API | Helium Documentation*. (n.d.). Docs.helium.com. Retrieved May 21, 2023, from <https://docs.helium.com/api/console/>