

National University of Computer and Emerging Sciences



Smart City Pollution Monitoring System Using Machine Learning Analytics

Shamoon Shahid 16L-4093

Aleem Qasim 16L-4226

Usama Ali 15L-4187

Supervisor:Dr Taimur Bakhshi

B.S. Computer Science

Final Year Project

June 2020

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:

**Title: Smart City Pollution Monitoring System Using Machine Learning Analytics**

is the sole contribution of the author(s) and no part hereof has been reproduced on **as it is** basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: \_\_\_\_3 June 2020\_\_\_\_\_\_\_\_\_\_

Student 1

Name: Shamoon Shahid

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student 2

Name: Aleem Qasim

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student 3

Name: Usama Ali

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Contents

[Table of Contents i](#_Toc42099114)

[List of Tables iii](#_Toc42099115)

[List of Figures iv](#_Toc42099116)

[Abstract 1](#_Toc42099117)

[Chapter 1: Introduction 2](#_Toc42099118)

[1.1 Goals and Objectives 2](#_Toc42099119)

[1.2 Scope of the Project 2](#_Toc42099120)

[Chapter 2: Literature Survey 3](#_Toc42099121)

[Chapter 3: Requirements and Design 4](#_Toc42099122)

[3.1 Functional Requirements 4](#_Toc42099125)

[3.2 Non-Functional Requirements 4](#_Toc42099126)

[3.3 Hardware Requirements 4](#_Toc42099127)

[3.3.1 Arduino Board 4](#_Toc42099128)

[3.3.2 Sensors 4](#_Toc42099129)

[3.3.3 SD card and Wi-Fi module 4](#_Toc42099130)

[3.4 Software Requirements 5](#_Toc42099131)

[3.4.1 Arduino (IDE) 5](#_Toc42099132)

[3.4.2 Android Studio 5](#_Toc42099133)

[3.4.3 Cloud 5](#_Toc42099134)

[3.5 System Architecture 5](#_Toc42099135)

[3.6 Architectural Strategies 7](#_Toc42099136)

[3.6.1 Arduino Uno 7](#_Toc42099137)

[3.6.2 Firebase (Online Cloud Fire Store) 7](#_Toc42099138)

[3.7 Use Cases 8](#_Toc42099139)

[3.7.1 Login 8](#_Toc42099140)

[3.7.2 Signup 9](#_Toc42099141)

[3.7.3 Select a Location to see readings 9](#_Toc42099142)

[3.7.4 View Air Quality 10](#_Toc42099143)

[3.7.5 Save a Particular Reading 10](#_Toc42099144)

[3.7.6 View Predicted Value of Pollution 11](#_Toc42099145)

[3.7.7 Graph Analysis of Readings 11](#_Toc42099146)

[3.8 GUI 12](#_Toc42099147)

[3.9 System Requirements 16](#_Toc42099148)

[3.9.1 Hardware Requirements 16](#_Toc42099149)

[3.9.2 Software Requirements 16](#_Toc42099150)

[3.10 Design Considerations 16](#_Toc42099151)

[3.10.1 Assumptions and Dependencies 16](#_Toc42099152)

[3.10.2 General Constraints 17](#_Toc42099153)

[3.10.3 Availability of Resources 17](#_Toc42099154)

[3.11 Development Methods 17](#_Toc42099155)

[3.11.1 Requirements and Analysis 17](#_Toc42099156)

[3.11.2 Design 18](#_Toc42099157)

[3.11.3 Development 18](#_Toc42099158)

[3.11.4 Testing and Maintenance 18](#_Toc42099159)

[3.12 Class diagram 18](#_Toc42099160)

[3.13 Sequence diagram 19](#_Toc42099161)

[3.13.1 Adding new general User 19](#_Toc42099162)

[3.13.2 View Current Readings 19](#_Toc42099163)

[3.13.3 Save Particular Readings 20](#_Toc42099164)

[3.13.4 Viewing the Predicted Reading 20](#_Toc42099165)

[3.13.5 Viewing Graph Analysis 21](#_Toc42099166)

[3.13.6 Alarming User on Reading exceeding threshold 21](#_Toc42099167)

[Chapter 4: Implementation and Test Cases 22](#_Toc42099168)

[4.1 Implementation 22](#_Toc42099170)

[4.1.1 Air Quality 22](#_Toc42099171)

[4.1.2 Temperature Readings 22](#_Toc42099172)

[4.1.3 Sound Readings 23](#_Toc42099173)

[4.1.4 Humidity Readings 23](#_Toc42099174)

[4.1.5 Front Development of App 24](#_Toc42099175)

[4.1.6 Back end Development of App 24](#_Toc42099176)

[4.1.7 Programming the Arduino Board 24](#_Toc42099177)

[4.1.8 Analysis of Data 24](#_Toc42099178)

[4.2 Test case Design and description 25](#_Toc42099179)

[4.2.1 Login Account with Correct Credentials 25](#_Toc42099180)

[4.2.2 Signup with Correct Credentials 26](#_Toc42099181)

[4.2.3 Selecting a Location to see Readings 27](#_Toc42099182)

[4.2.4 Showing the Current Reading of Air Quality 28](#_Toc42099183)

[4.2.5 Transferring of data from Device to Cloud 29](#_Toc42099184)

[4.2.6 View Predicted Value of Air Quality 30](#_Toc42099185)

[4.2.7 Saving a Particular Value of Air Quality 31](#_Toc42099186)

[4.2.8 Sending a Notification from the System to User 32](#_Toc42099187)

[4.3 Test Metrics 33](#_Toc42099188)

[4.3.1 Traceability Matrix 33](#_Toc42099189)

[Chapter 5: Experimental Results and Analysis 34](#_Toc42099190)

[Chapter 6: Conclusion 36](#_Toc42099191)

[References 37](#_Toc42099192)

[Appendix 38](#_Toc42099193)

[Appendix A: Photo of Place where device installed 38](#_Toc42099194)

[Appendix B: Code which programmed Arduino. 38](#_Toc42099195)

List of Tables

[Table 1: Test Metrics 33](#_Toc42081223)

[Table 2: Traceability Matrix 33](#_Toc42081224)

[Table 3: AQI levels and Associated Health Impacts 34](#_Toc42081225)

List of Figures

[Figure 1 High Level Architecture 5](#_Toc42100064)

[Figure 2 Low Level Architecture 6](#_Toc42100065)

[Figure 3: Sign Up 12](#_Toc42100066)

[Figure 4: Log in 12](#_Toc42100067)

[Figure 5:Home to Perform Activity 13](#_Toc42100068)

[Figure 6 : Location 13](#_Toc42100069)

[Figure 7: Viewing Page 14](#_Toc42100070)

[Figure 8 : Graph Analysis of Current Reading 14](#_Toc42100071)

[Figure 9: Predicted Value with Graph Analysis 15](#_Toc42100072)

[Figure 10 : Main Screen showing Aim of our App 15](#_Toc42100073)

[Figure 11: Class Diagram 18](#_Toc42100074)

[Figure 12: Adding User Sequence Diagram 19](#_Toc42100075)

[Figure 13: View Current Readings Sequence Diagram 19](#_Toc42100076)

[Figure 14: Save a Particular Reading Sequence Diagram 20](#_Toc42100077)

[Figure 15: View the Predicted Reading Sequence Diagram 20](#_Toc42100078)

[Figure 16: Viewing Graph Analysis 21](#_Toc42100079)

[Figure 17: Alarming user Sequence Diagram 21](#_Toc42100080)

[Figure 18: Air Quality Comparison 22](#_Toc42100081)

[Figure 19: Temperature Readings 22](#_Toc42100082)

[Figure 20: Sound Readings 23](#_Toc42100083)

[Figure 21:Humidity Readings 23](#_Toc42100084)

[Figure 22: MQ-2 data 34](#_Toc42100085)

[Figure 23: Air quality v/s Time 35](#_Toc42100086)

[Figure 24: Congestion Zone 38](#_Toc42100087)

[Figure 25: Machine Learning code 44](#_Toc42100088)

# Abstract

Air quality monitoring and management has become one of the primary concerns for the public and environmental agencies to appraise air pollution related diseases. The primary sources of air pollution are generally attributed to traffic and factory emissions, floral pollen and excessive mold due to high air humidity. The present project developed a system based on the Internet of Things paradigm to monitor record and predict air pollution. Using off shelf electronic hardware and bespoke software development, both the sensory development and the mobile monitoring application was developed in this project. An Arduino board was used along with sensors (for air contaminant monitoring, temperature, humidity and noise). The system was linked via WIFI to the global Internet so monitored values could be remotely recorded and viewed. An Arduino based mobile application (with a back-end cloud) was developed to view the air quality data from Arduino devices. Furthermore, linear regression was used to predict future air quality values based on features including location & time of the day. The cost effective solution was tested on the national highway as well as university environment for monitoring purposes with relative ease. In future, the device can be installed at different locations for successfully monitoring and predicting air quality for target areas where sensors for air quality monitoring are not available.

# Chapter 1: Introduction

Air is an essential element of human’s surrounding. Due to less greenery, the increased number of gasoline vehicles are playing major role in polluting the air quality. Humans need an atmosphere which has natural composition of the gases such as Carbon Monoxide, Nitrogen, Oxygen and traces of other rare elements. Any change in this natural composition cause harmful effects such as difficulty in breathing, coughing, asthma etc. Air pollution must be tracked in real time and kept under control for people's healthy lives and a better future.

In recent years, conventional methods have been used to evaluate air quality which were quite expensive. On the other hand, they provided limited accuracy and inefficient output prediction. Our project aims to create a smart air pollution monitoring device capable of monitoring, analyzing and logging air quality data to a remote server and keeping the data up-to-date using very flexible and low-cost internet of things.

Our Air pollution detection system uses machine learning to control and monitor the aspect of live air in an atmosphere via IOT. System needs a gas sensor that senses harmful atmospheric gasses and continually transmits data. Then these sensors communicate with Arduino Uno processing this information and telling us the index of air quality. This will help authorities track air quality in various areas and assist them in taking preventive measures to reduce it.

## Goals and Objectives

* To sense and collect data Air Quality Data.
* To forecast the Data to take preventive measures for future.
* To reduce pollution in environment and detect which areas of city are having high intensity of producing pollution.
* To keep the track of points in the city where specific actions needs to be taken.

## Scope of the Project

Our scope includes the following:

* Working on to access the Air quality index in a particular area
* Deploying data on cloud and then making it useful through machine learning techniques
* Developing a portal for presenting useful information gathered
* Helping the community to take significant measures to improve air quality in their area.

The subsequent sections of the documents will discuss the Literature survey, Requirements and design, Implementation detail followed by conclusion.

# Chapter 2: Literature Survey

Project revolves around iot and machine learning so we first need to understand iot. This involves interconnected computing devices, electronic or mechanical objects or machines or people capable of transferring data over the network without requiring communication between humans. So, the iot means connecting all the things to the internet. By connecting things with the internet means that it can send or receive information or do both. And this ability of sending and receiving things over the internet make thing smart and smart is good.

In 2015, a monitoring system for air pollution that uses sensors in urban environments to measure Carbon dioxide, nitrogen, carbon oxide. The gathered air pollution monitoring information through the sensors is then periodically stored in the central online repository system. To transfer data to a central computer, it uses a wireless GSM modem connection. The information is then displayed on a web site developed by the users.

In 2016, a system called “IJARCCE” was design to monitor noise and air pollution using Arduino Uno and two sensor which are MQ7(to measure CO) and sound sensor and the data gathered from these sensor is processed in “MATLAB” where intensity of sound and carbon oxides is measure.

The aim of making smart air pollution monitoring system can be fulfilled by using internet of things, this enhancing the quality services and making the life better. In past different methods were used but in our air pollution monitoring system we making a wireless system using Arduino board and different useful sensors (mention in architectural section) that fundamental objectives are on building up a situation liberated from contamination by making it brilliant. Sensors are fitted at the specific spots or out in the open vehicles. By checking all the sensor organizes states, all the ecological happenings can be accumulated on a database to analyze the air quality position. It based on the theory of internets of things.

In field of iot two boards are available, Arduino and raspberry pi. Both are equally popular but in order to develop our project we are using Arduino board. The open source microcontroller Arduino Uno board has the “ATmega328P” microchip and is fitted with 6 analogs pins and 14 digital pins sets. Using Arduino IDE and USB cable we can program microprocessor board. And Board Need 5 to 12 Voltage power supply.

We choose the Arduino microcontroller for several reasons. Arduino Uno Board does not require any transistor like raspberry pi 3 to connect with sensor and it has long life span as compared to raspberry pi board. Arduino Uno has a ' real-time ' and ' analog ' functionality that the Pi does not, this versatility enables it to operate with just about any form of sensor or chips, Raspberry Pi is not as versatile, for example, reading analog sensors requires additional hardware (transistors) help.

# Chapter 3: Requirements and Design

In this Chapter we have mainly described the functional and nonfunctional requirements in great details.



## Functional Requirements

Requirements of our project are as under:

1. Our system will be verify entered email and phone number.
2. Our system shall get the information of air quality of particular location.
3. Our system shall send notification in case of alarming conditions to pre-registered email address and phone number.
4. Our System shall able to save a particular value of air quality for the user.
5. Our System can save the user a specific value of air quality.
6. Our system shall perform these actions multiple times a day.

## Non-Functional Requirements

Following are the requirements for this:

1. **Usability:** The system’s interface will be interactive and simple enough to understand and learn easily. Buttons, navigators and titles etc. will be self-explanatory.
2. **Performance:** Overall system will be optimized so that turnaround time will be minimum and system runs smoothly
3. **Security:** The readings recorded will remain same throughout.
4. **Compatibility:** System has an android application and it will work with devices with android version 5.0 and onwards.
5. **Reliability:** Our system shall be reliable so that one can easily recover one’s lost device. It will be ensured that data is accessible at all time, no matter after how long that data is accessed.
6. **Quality:** Our system shall get accurate air quality measurements so that it would be relatively easy to get air quality information of particular area.

## Hardware Requirements

The hardware which we require are as follows:

### Arduino Board

It is a small microprocessor which can be programmed and stored our program in memory, the sensors are connected by microprocessor using jumper wire.

### Sensors

We are using different types of sensors MQ135, MQ2, MQ7, DHT11 Sensor and Sound Sensor. The above sensors used to detect the intensity of harmful gases in air.

### SD card and Wi-Fi module

We use the SD card module to store data where there is no network and Wi-Fi module accountable for network data transmission to the cloud.

## Software Requirements

The software requirements are as follows:

### Arduino (IDE)

It is just a compiler which is use to program the board.

### Android Studio

It is the core part of our project and we need it for the development of mobile App.

### Cloud

We will be using cloud for storing data which sensors gathered, the data will be transmitted over the Wi-Fi module to the cloud. And the name of that cloud fire store which is a sub diary of Firebase.

## System Architecture

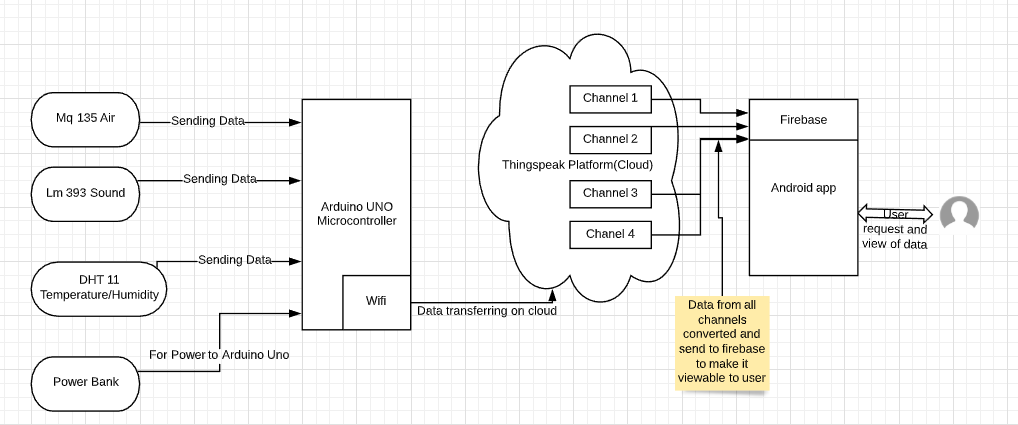


Figure 1 High Level Architecture

High level Architecture of System that shows how modules will contact

As showcased in Figure, in terms of high level Architecture diagram, Data collection is done through sensors and they are sending data to Arduino board. Arduino board will be sending this data through WIFI module to a platform called Thing speak. A specific id is generated for WIFI module to make it get connected with thing speak. Different channels are made here for readings collected through different sensors. Then data will be converted so that it would get compatible with firebase. For making this data viewable to user, An Android app will be made.

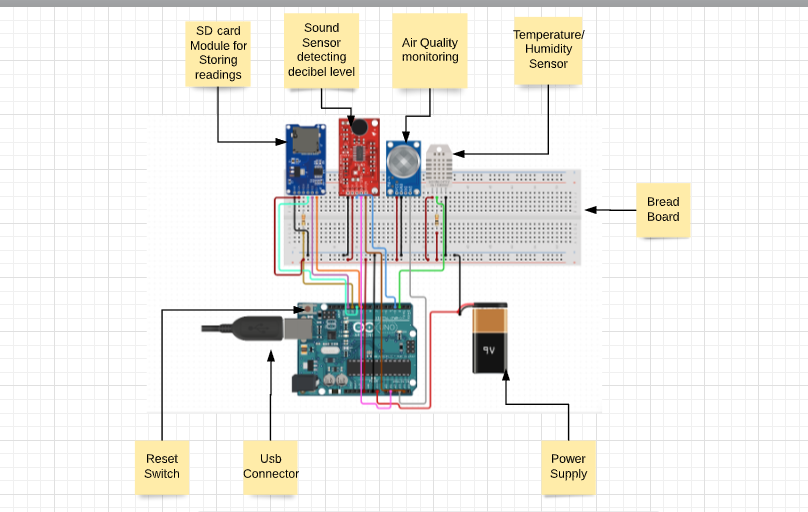


Figure 2 Low Level Architecture

Low Level Architecture shows in depth contact of modules

As shown in above figure, after giving power to Arduino Board monitoring will be started. At present SD card module is attached which is used for storing data. The Sound Sensor detects a sound's softness or loudness at the decibel level. Humidity is measured by a component of humidity sensor that is dht11, As the moisture changes, the substrate's conductivity level of sensor index changes. Temperature is also determined by DHT11 sensor which gives a number value with respect to temperature .MQ135 sensor work on the principle of conductivity. In fresh air it has low conductivity index and in polluted air is has higher conductivity which mean it give larger value in polluted air.

## Architectural Strategies

Equally popular circuit boards for IoT projects are both Raspberry pi3 and Arduino Uno. Even though they're not the only ones, they each have a huge fan and tons of online resources.

### Arduino Uno

In field of iot two boards are available, Arduino and raspberry pi. Both are equally popular but in order to develop our project we are using Arduino board. The “Arduino Uno” is a microcontroller device that is open source. The board has the “ATmega328P” microchip and is fitted with 6 analogs pins and 14 digital pins sets that can be interfaced with different boards for expansion and other circuits. Using Arduino IDE and USB cable we can program microprocessor board and board Need 5 to 12 Voltage power supply.

We choose the Arduino microcontroller for several reasons. Arduino Uno Board does not require any transistor like raspberry pi 3 to connect with sensor and it has long life span as compared to raspberry pi board. Arduino Uno has a ' real-time ' and ' analog ' functionality that the Pi does not, this versatility enables it to operate with just about any form of sensor or chips, Raspberry Pi is not as versatile, for example, reading analog sensors requires additional hardware (transistors) help. This concludes that Arduino is best suited for small projects.

### Firebase (Online Cloud Fire Store)

Firebase is an ongoing cloud facilitated database, Data is put away as json and synchronized progressively to each associated customer. At the point when we fabricated applications and associate our application to firebase all the applications associated with the firebase become synchronized with one another. Data is stored in firebase in no sql format in the form of documents.

Firebase is the core of our internet of things campaign. The local unit (Uno WIFI module) collects sensor and actuator status information, sends it to the Internet, "writing" on Firebase. The local unit also collects information from the internet, "reading" them from specific channels of Firebase Actuator. Data is stored in the form of key value pairs. The key is the unique time in which reading is recorded and reading is the actual data that is to be stored on the cloud. The best feature about firebase is that it allows the user to store data in different options like documents, multiple collections and sub collections within documents. Arrays and maps can come in document but it can become complex if the data exceeds the sudden amount, sub collection is useful for the data that might be expanded over the time. Collections at the root level makes it easy to organize the data at desperate level.

## Use Cases

The description of different use cases is given below having the information of actors and their actions in our app.

### Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | UC-1 | | |
| Actors | | Admin Local, Public, Environmentalist, Local Administration (Meteorology Department), Media Reporters | | |
| Summary | | The user will be able to login in to the app to see the air quality. | | |
| Pre-Conditions | | Data base record of the user required. | | |
| Post-Conditions | | User will be able to login based on the input. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user clicks to opens the login page. | | 2 | The login page is shown by the framework requesting email and secret key. |
| 3 | The user clicks on the text box enters valid email and password. | | 4 | The system verifies the email and password, establishes a session for the user and redirects the user to the home page. |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid email or password. | | 4-A | The system responds with an error message: Incorrect email or passwordentered. |

### Signup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | UC-2 | | |
| Actors | | Admin Local, Public, Environmentalist, Local Administration (Meteorology Department), Media Reporters | | |
| Summary | | The User is able to sign up for account formation. | | |
| Pre-Conditions | | The user must open the app with Signup page. | | |
| Post-Conditions | | User will successfully login on to the home page. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User clicks the sign-up button. | | 2 | System will prompt to enter account information |
| 3 | User will enter a valid email id and password and other required information*.* | | 4 | System will authenticate, identify user and redirect user to particular home page meaning user account has been created. |
| **Alternative Flow** | | | | |
| 3 | User enters already registered email | | 4-A | System displays the message to login with the given email. |

### Select a Location to see readings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | UC-3 | | |
| Actors | | Admin Local, Public, Environmentalist, Local Administration (Meteorology Department), Media Reporters | | |
| Summary | | The User is select the location where device is installed. | | |
| Pre-Conditions | | User should have a working system with Internet access. | | |
| Post-Conditions | | User will be able to select the location. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User clicks the spinner button to select the location. | | 2 | System will show the locations where devices are installed. |
| 3 | User will choose desire locations in the spinner. | | 4 | System will cross check and promote the user to move to the next screen. |
| **Alternative Flow** | | | | |
| 3 | User doesn’t select the location. | | 4-A | System will prompt the user to select the location. |

### View Air Quality

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | UC-4 | | |
| Actors | | Admin Local, Public, Environmentalist, Local Administration (Meteorology Department), Media Reporters | | |
| Summary | | The User can view air quality details of particular area. | | |
| Pre-Conditions | | User must login with a valid account. | | |
| Post-Conditions | | User will successfully view details about air pollution. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User will click View Air Quality. | | 2 | System will prompt the select particular location. |
| 3 | User will select the Particular location. | | 4 | System will check if the particular location exists or not? |
| 5 | User see the Particular location readings. | | 6 | System will show the readings of particular location. |
| **Alternative Flow** | | | | |
| 3 | User enters the location that do not exist. | | 4-A | System gives suggestion about the nearby location. |

### Save a Particular Reading

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | UC-5 | | |
| Actors | | Admin Local, Public, Environmentalist, Local Administration (Meteorology Department), Media Reporters | | |
| Summary | | The user will be able to Save a Particular Reading. | | |
| Pre-Conditions | | The User must login with a valid account. | | |
| Post-Conditions | | User will successfully view details about air pollution. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User will click save button. | | 2 | System will store a particular reading. |
| **Alternative Flow** | | | | |
| 3 | User reading will not be saved. | | 2-A | System will store user readings. |

### View Predicted Value of Pollution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | UC-6 | | |
| Actors | | Admin Local, Public, Environmentalist, Local Administration (Meteorology Department), Media Reporters | | |
| Summary | | The user will be able to see the predicted value of Air Pollution. | | |
| Pre-Conditions | | The User must login with a valid account. | | |
| Post-Conditions | | User will successfully view details of predicted air pollution. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User will click Predicted value button. | | 2 | System will show predicted reading. |
| **Alternative Flow** | | | | |
| 3 | User will not be able to see readings. | | 2-A | System will message to try again later. |

### Graph Analysis of Readings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | UC-7 | | |
| Actors | | Admin Local, Public, Environmentalist, Local Administration (Meteorology Department), Media Reporters | | |
| Summary | | The user will be able to see the graph analysis of Readings. | | |
| Pre-Conditions | | The User must login with a valid account. | | |
| Post-Conditions | | User will successfully view graph details of readings. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User will click graph analysis button. | | 2 | System will show the graph Analysis for the user. |
| **Alternative Flow** | | | | |
| 3 | User reading will not be able to see the Graph Analysis. | | 2-A | System will message to try again later. |

## GUI

Our GUI is quite simple.

1. This is the signup page, user will provide his details to sign up if he is using first time.

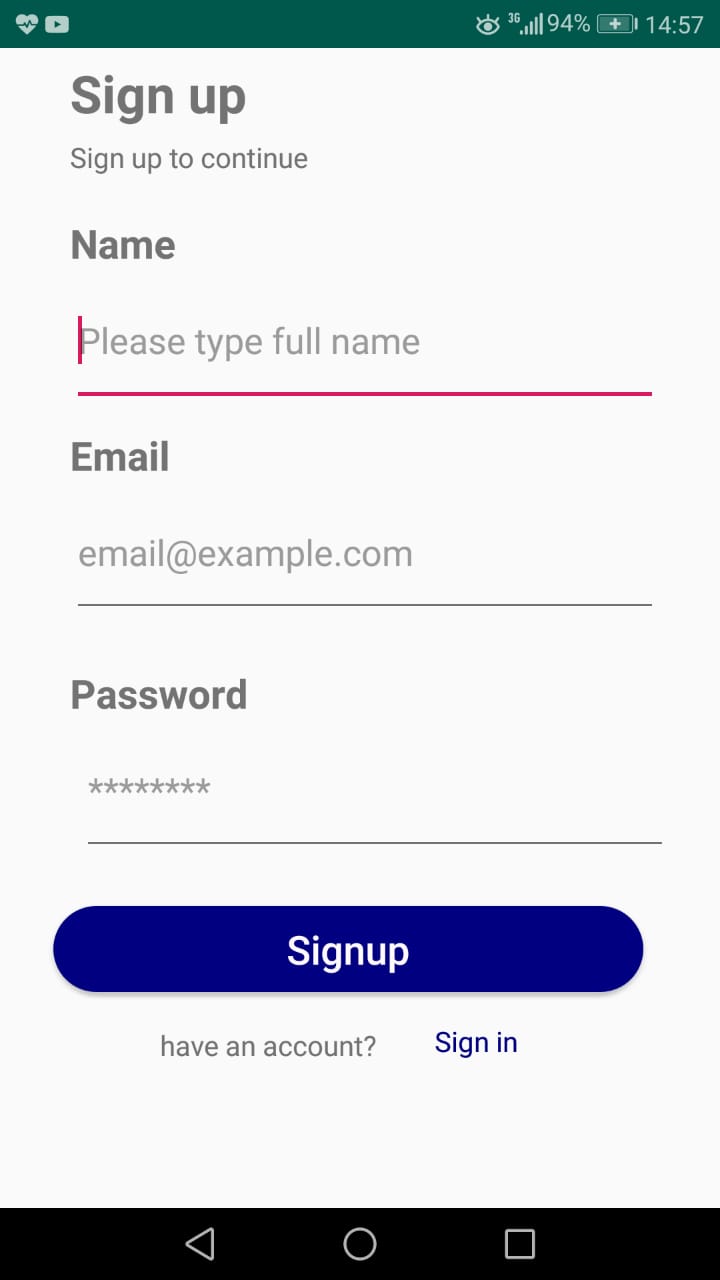


Figure 3: Sign Up

How Sign up will be performed in our app

1. This is the Login page.

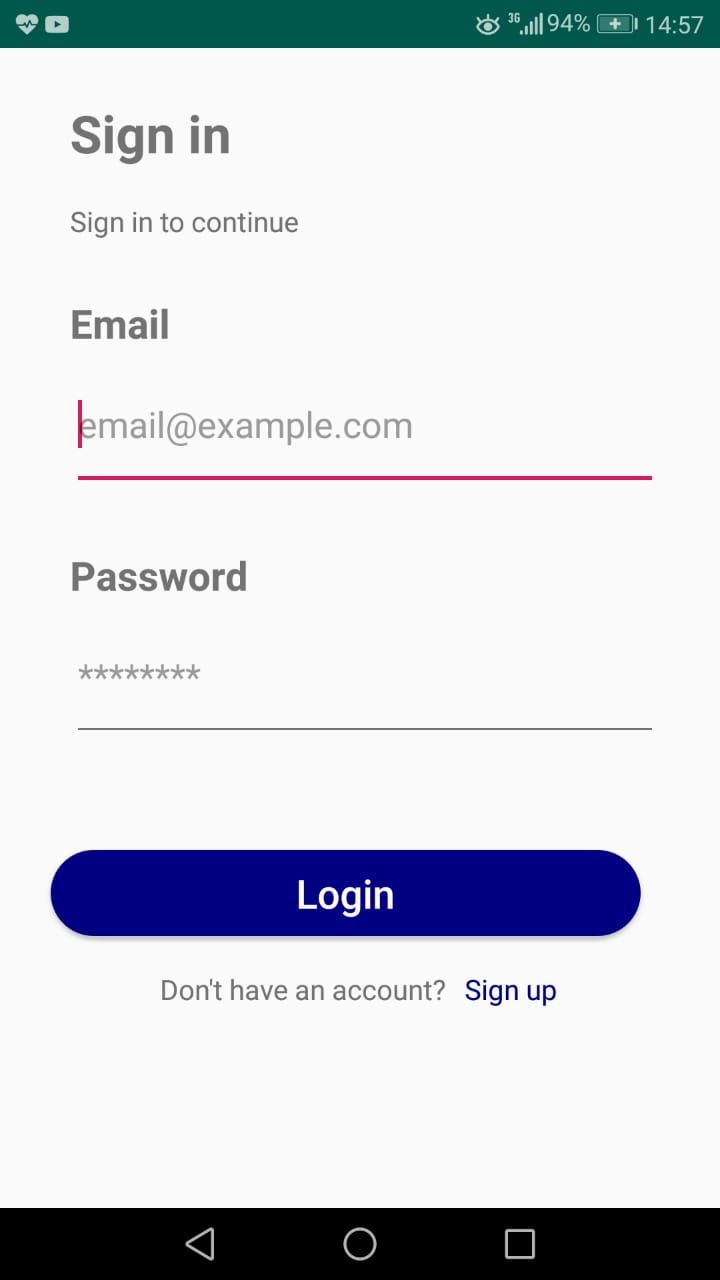


Figure 4: Log in

How Login will be performed once user signed up in app

1. This is the homepage navigation drawer, if user wants to perform any operation he might do it from here.

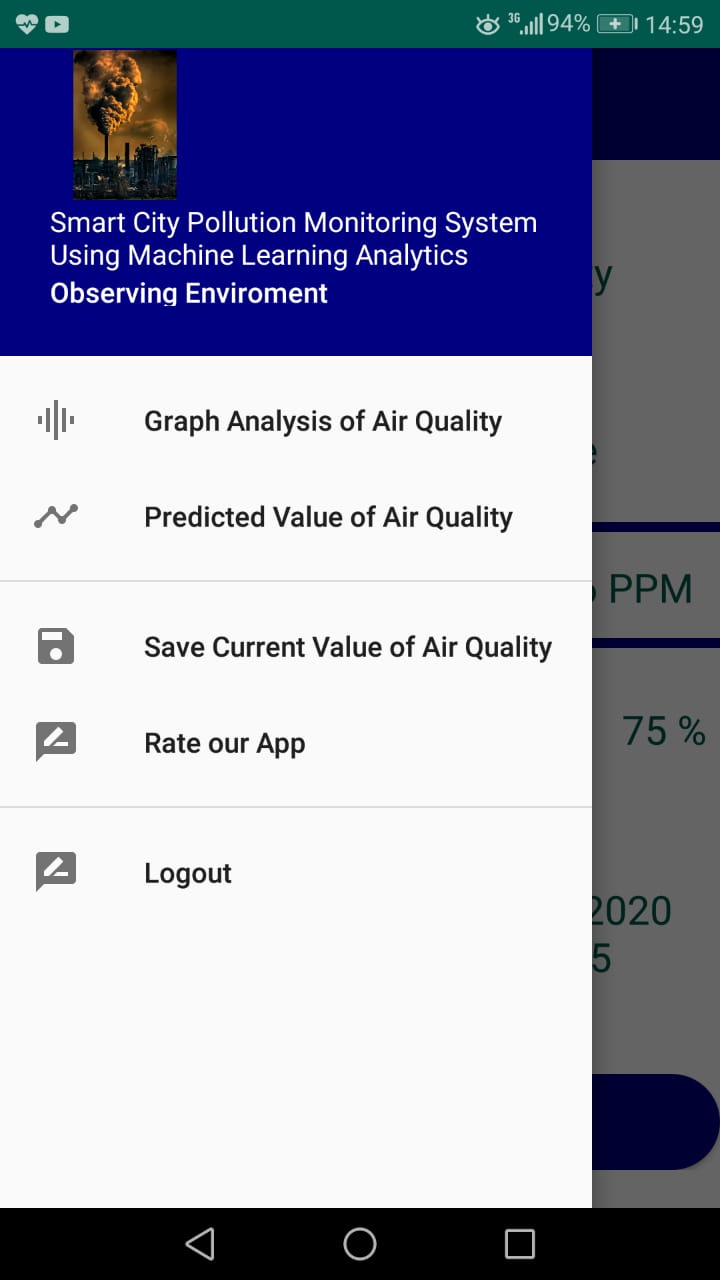


Figure 5:Home to Perform Activity

Shows how user can perform main activities using Navigation

1. This is the user page, from where user can select the location for which he/she want to see reading.

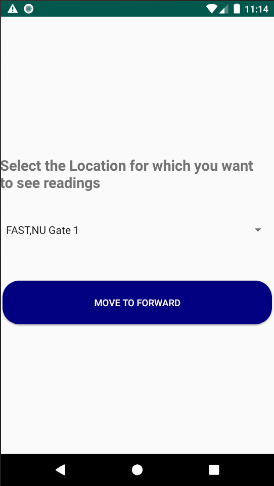


Figure 6 : Location

Shows how user can select location to see readings

1. This is the reading page where readings recorded from different areas at different date and time can be seen along with their details.

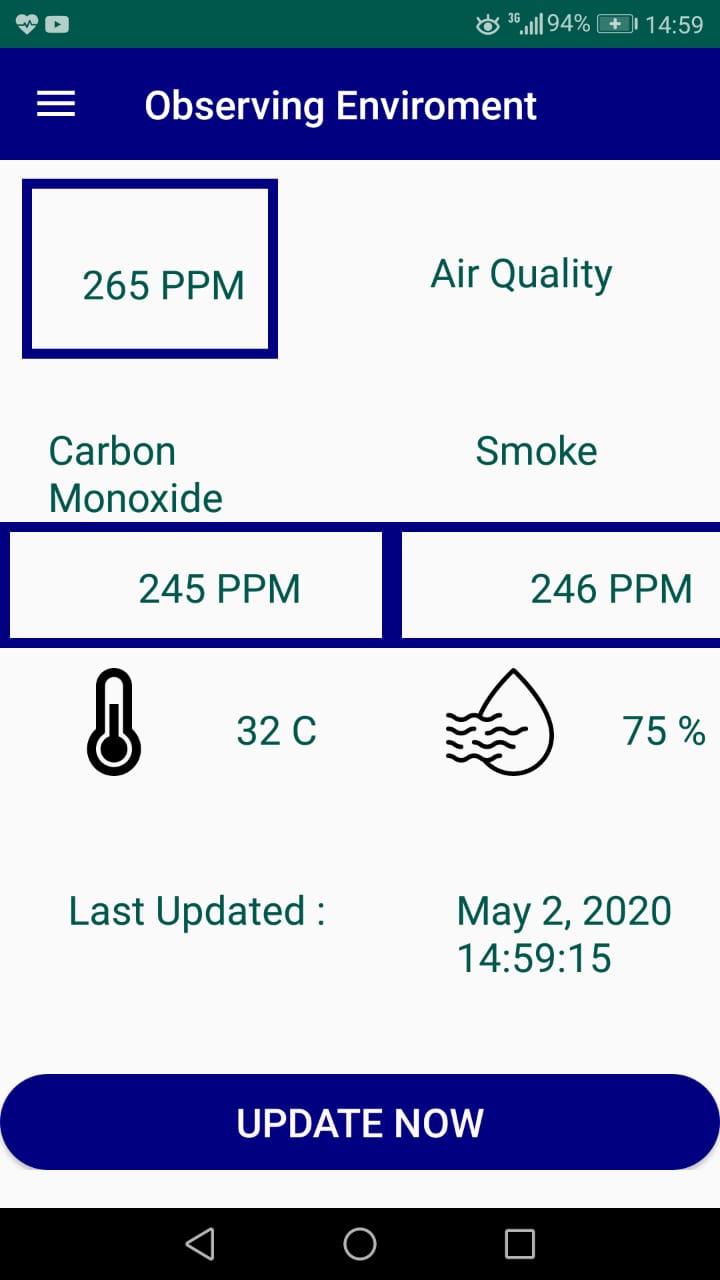


Figure 7: Viewing Page

Showing air quality with different attributes main functionality of app

1. This is the page which shows the graph analysis of the current reading, when we click on it from the menu.

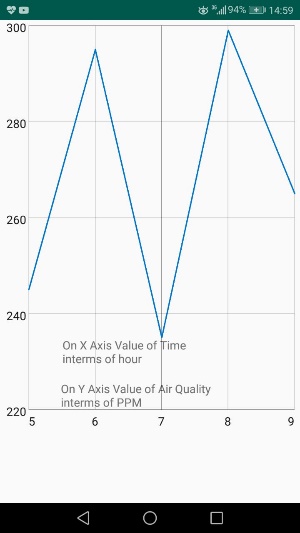


Figure 8 : Graph Analysis of Current Reading

Showing the graphical analysis of readings how readings are changing

1. This is the page which shows the graph analysis of the predicted reading, when we click on it from the menu.

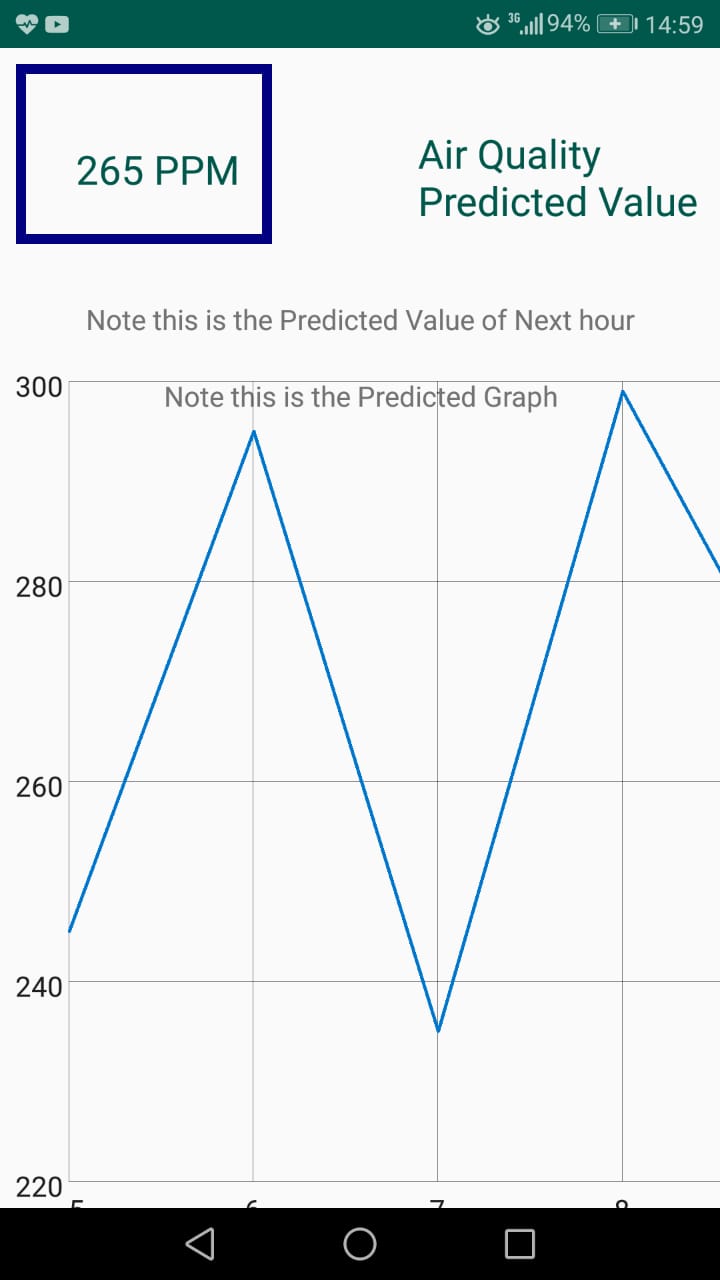


Figure 9: Predicted Value with Graph Analysis

Showing the graphical analysis of predicted value with the actual predicted value

1. This is the Screen which shows what our Application is actually going to do.



Figure 10 : Main Screen showing Aim of our App

First screen after login shows user our main aim

## System Requirements

The System requirements is divided into mainly two types.

### Hardware Requirements

1. Arduino Uno
2. Jumper Wires
3. MQ 2 Gas/Smoke Sensor
4. MQ 135 Sensor for harmful gases detection
5. WIFI Module ESP8266
6. Breadboard
7. DHT 11 Temperature and humidity sensor
8. LM 393 Sound Sensor
9. MQ 2 Gas/Smoke Sensor
10. MQ 7 CO sensor
11. Power Bank (Power supply to Arduino board).

### Software Requirements

1. Android Device
2. WIFI
3. Arduino IDE
4. IOT cloud

## Design Considerations

The issues or the conditions that must be met before using or designing the system are explained as follows.

### Assumptions and Dependencies

* The Board must be properly integrated with all the sensors using jumper wires and Board must be installed at the feasible location and programmed correctly, Also all the integration issues must be properly processed.
* Board works on windows so its programming must be done and then attached to the proper power supply. Moreover, our App will work on android device so App would be installed and connected with the internet.
* There will be basically two types of user one would be normal user (Local Public, Environmentalist, Local Administration, Media Reporters, system level Designers) and other would be Admin which will do all integration and resolve issues
* At present we are currently using SD card module to store data due to installation of device in remote areas, but in future we will be shifting to cloud for storing data.

### General Constraints

The following subsections of the document will be discussing some limitations that would be impacting system’s software

#### Hardware and Software Environment

* Arduino board should be installed at a proper place for monitoring so it does not get damaged.
* Reading can come inaccurate if sensors or either Arduino board is not kept safe from water or any kind of liquid.
* To deploy program on Arduino board, user should have IDE installed on device

#### Memory Limitations

Although board has maximum flash memory of 32k bytes but data up to 28k bytes can only be stored for efficient working.

#### Sensor Limitations

Different sensors have different limitations which should be kept in mind before monitoring

##### Air Quality Sensor

Mq-135 has a range of taking readings between 10-1000 ppm. Till 100 ppm, Air quality is moderate. After 300 ppm it is very hazardous.

##### Sound Sensor

Sound sensor has specific range in which it can work correctly and record the best suited readings if we install it 20 to 30 meters away from road it cannot measure correctly.

##### Temperature Sensor

Temperature sensor DHT-11 can measure up to temperature from 0 to 50 degrees Celsius with a sampling period of 1 second.

##### Humidity Sensor

Humidity sensor DHT-11 with a sampling period of 1 second can measure up to 20 to 90 percent of relative humidity.

### Availability of Resources

A persistent power connection should be available for giving power to Arduino board for monitoring. Power bank can be used as it is portable.

## Development Methods

Our Project initially involved research where we have done brainstorming that what we are going to do, what has already implemented in our work and what new we will be going to do, so finally it comes up to a point that we actually are going to do. Then we followed Water Fall method which is the traditional and most followed model for software development.

### Requirements and Analysis

We have analyzed and gather the requirements for our project, which include hardware and software requirements, moreover before starting the project we have done the cost analysis and requirement analysis. For installing of device, we contacted the civil engineering student, he will be responsible for installing the device on highway and give us raw data. All these concludes that we have done briefly analysis before starting the project.

### Design

After brief analysis we then decided the design that how our hardware will come in contact with the software, we initially decide that our hardware devices will use SD card module for storing the data due to unavailability of Wi-Fi at particular places. But this is our initial design in future we will slightly change our design and will shift to Wi-Fi module but it will not affect our overall design.

### Development

After finalizing design, we finally purchased the hardware devices there was a conflict while purchasing devices which is discussed in the next section. We spend fruitful time for learning the functionality of hardware devices and finally made the devices compatible with software so that there may be no more compatibility issue. In addition to the development of hardware device we developed an android app for the easiness and convenience of the user.

### Testing and Maintenance

Once our device will be able to give us readings, and when we will have sufficient amount of data then we shall be able to test our predicted value by actually analyzing what value we have on present day. Also, Maintenance of device and software is needed so that it would work properly in future.

## Class diagram

The Class Diagram is shown here and it explained the main user and their characteristics:

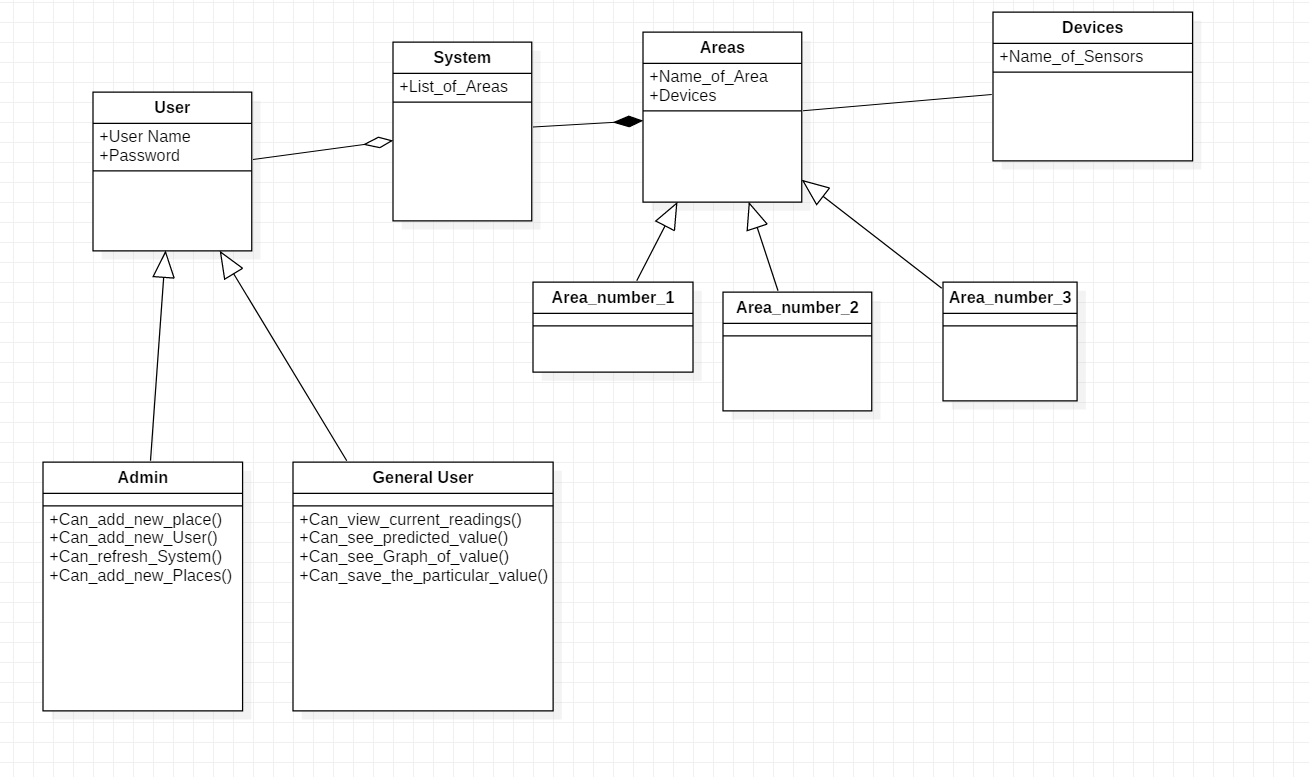


Figure 11: Class Diagram

Showing how different model classes are coordinating

## Sequence diagram

### Adding new general User

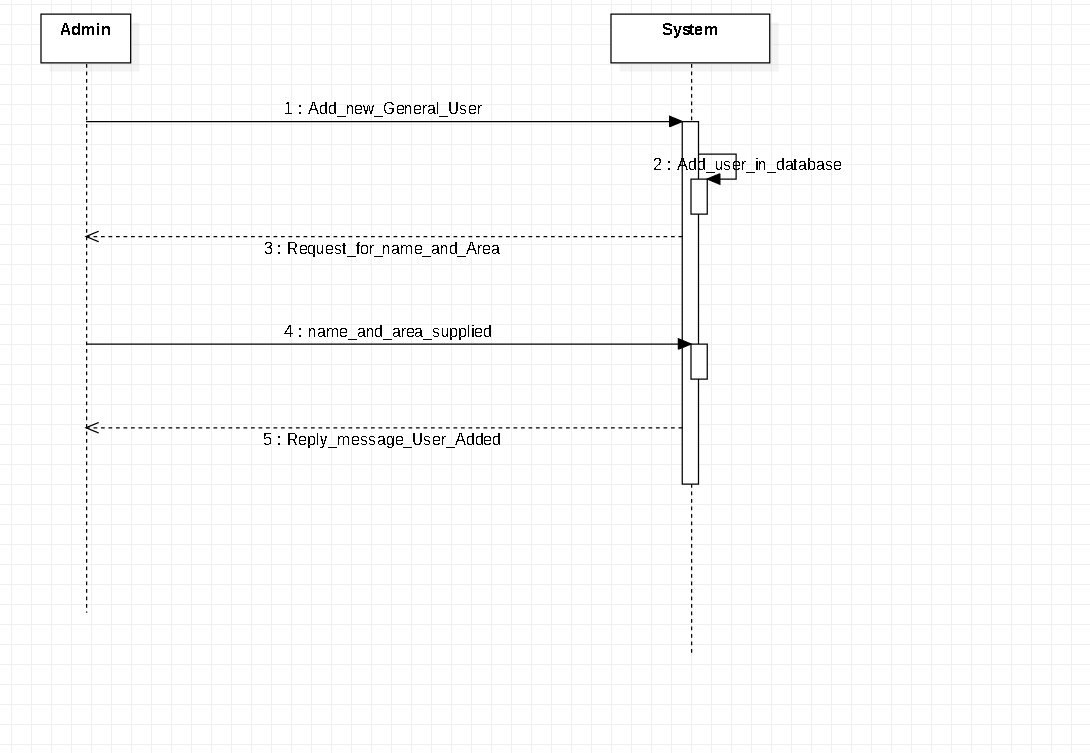


Figure 12: Adding User Sequence Diagram

User is adding by signing up in the app

### View Current Readings

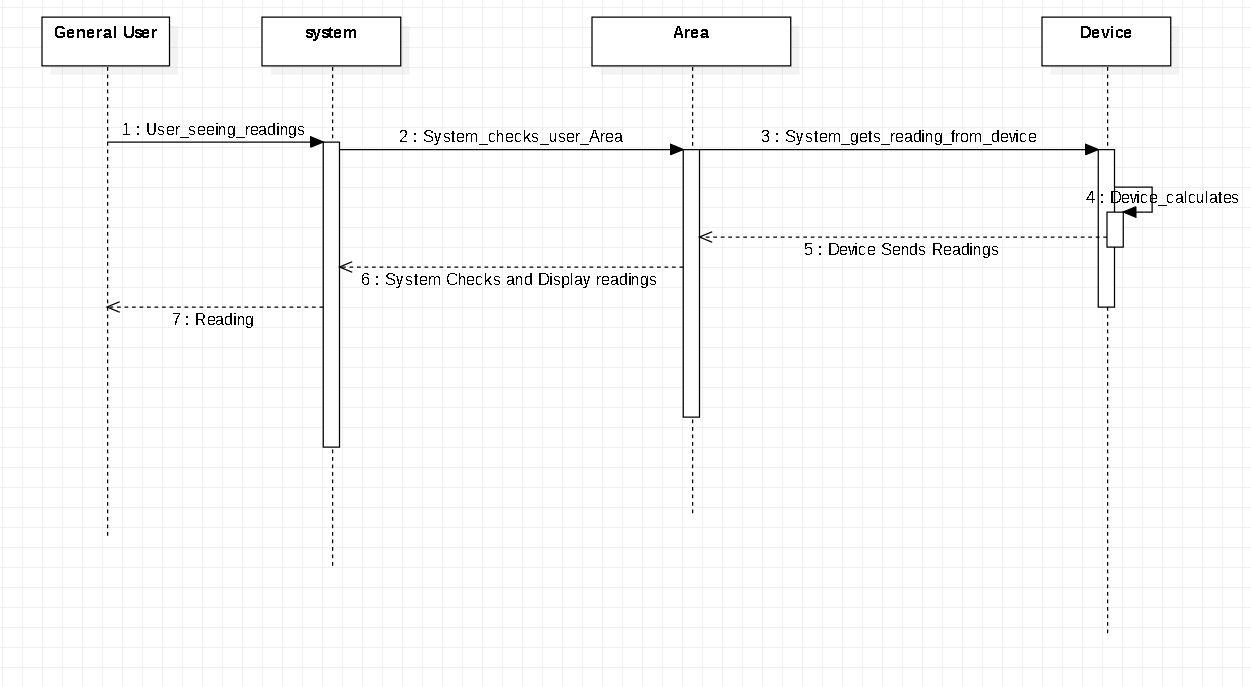


Figure 13: View Current Readings Sequence Diagram

Showing how the app will be fetching readings to show to user

### Save Particular Readings

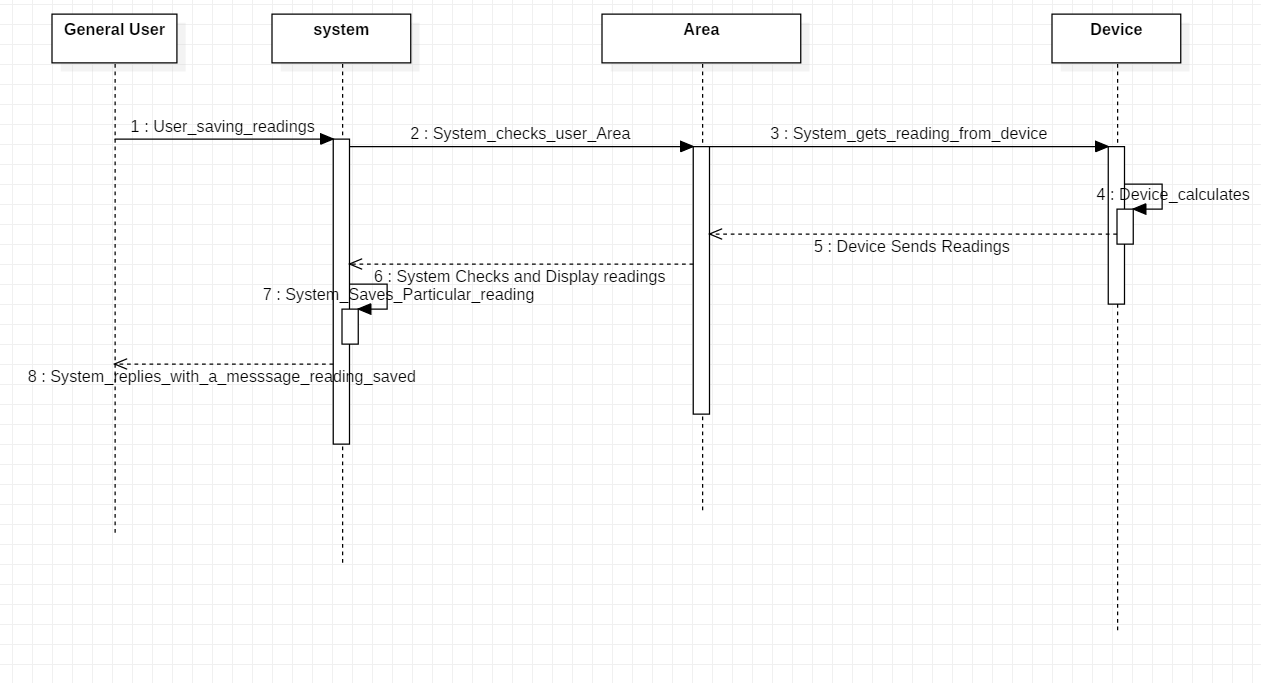


Figure 14: Save a Particular Reading Sequence Diagram

Showing how app will be saving reading to database

### Viewing the Predicted Reading

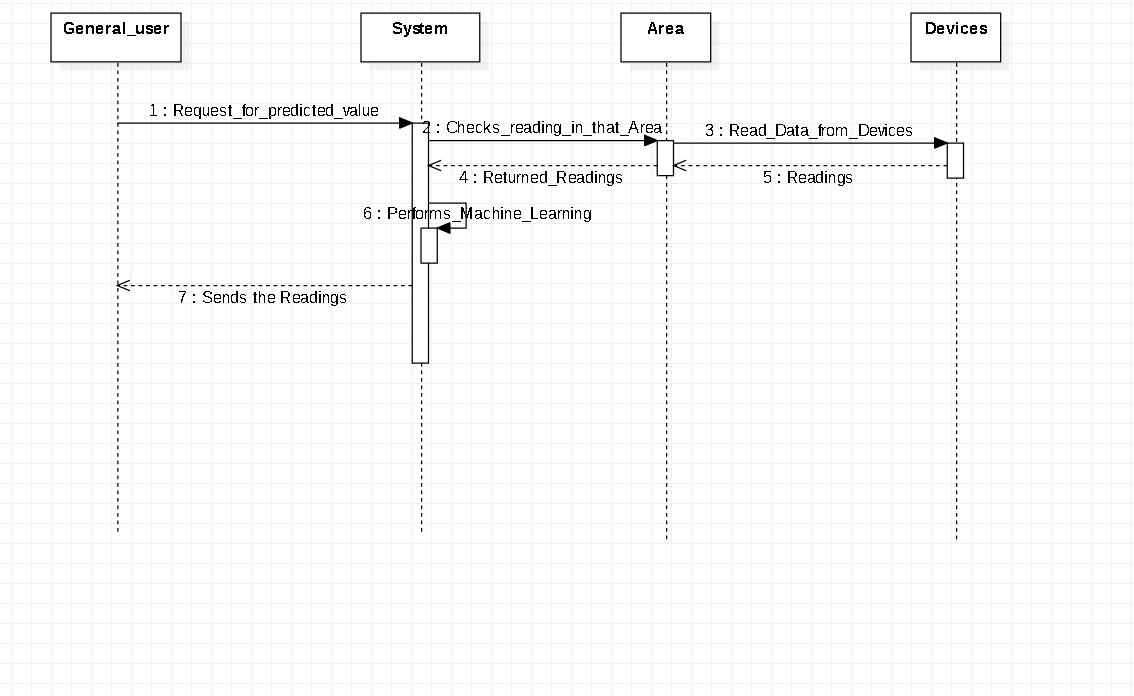


Figure 15: View the Predicted Reading Sequence Diagram

Showing how app will made visible the predicted value to user

### Viewing Graph Analysis

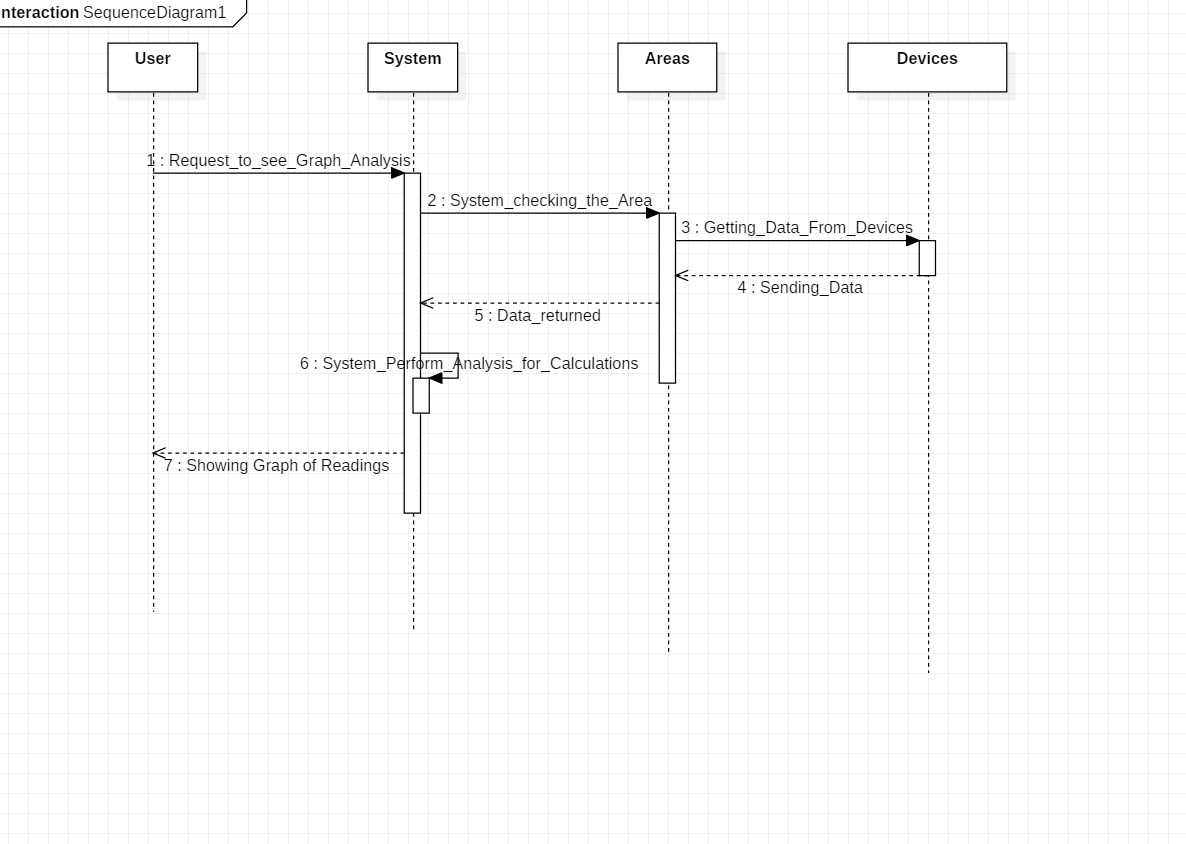


Figure 16: Viewing Graph Analysis

Showing how the graphical data will be made visible to the user

### Alarming User on Reading exceeding threshold

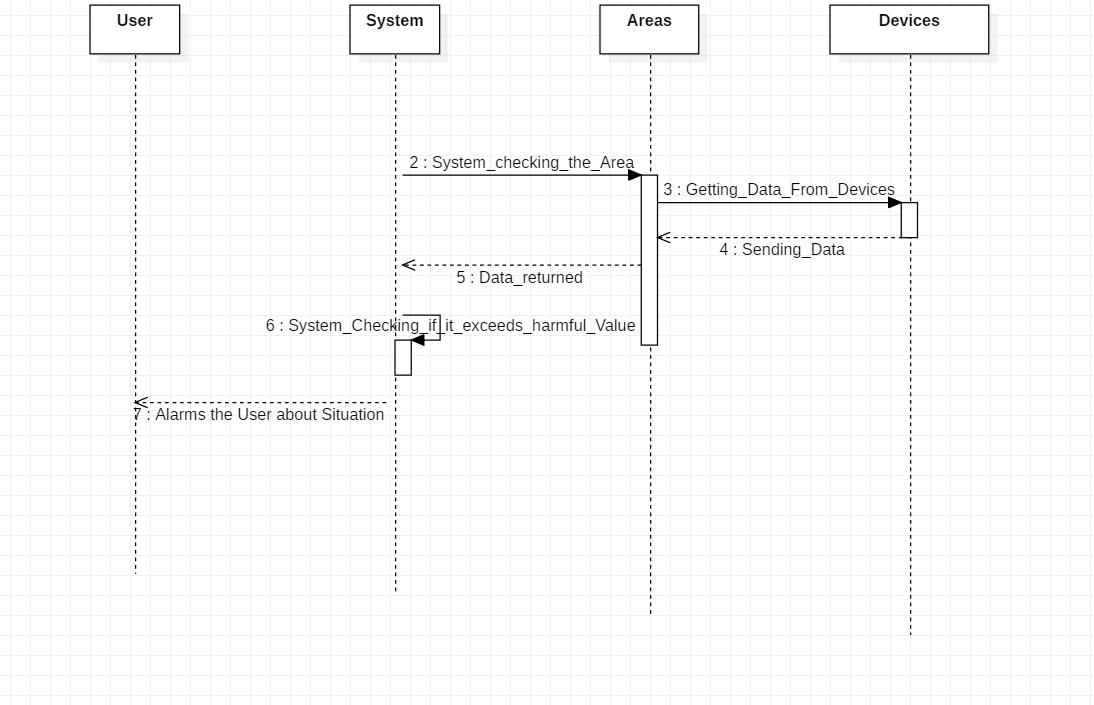


Figure 17: Alarming user Sequence Diagram

Showing how the user will be alarmed if reading exceeds

# Chapter 4: Implementation and Test Cases



## Implementation

### Air Quality

Apart from assembling devices and making them work in perfect order we also have installed devices in different places for test readings.

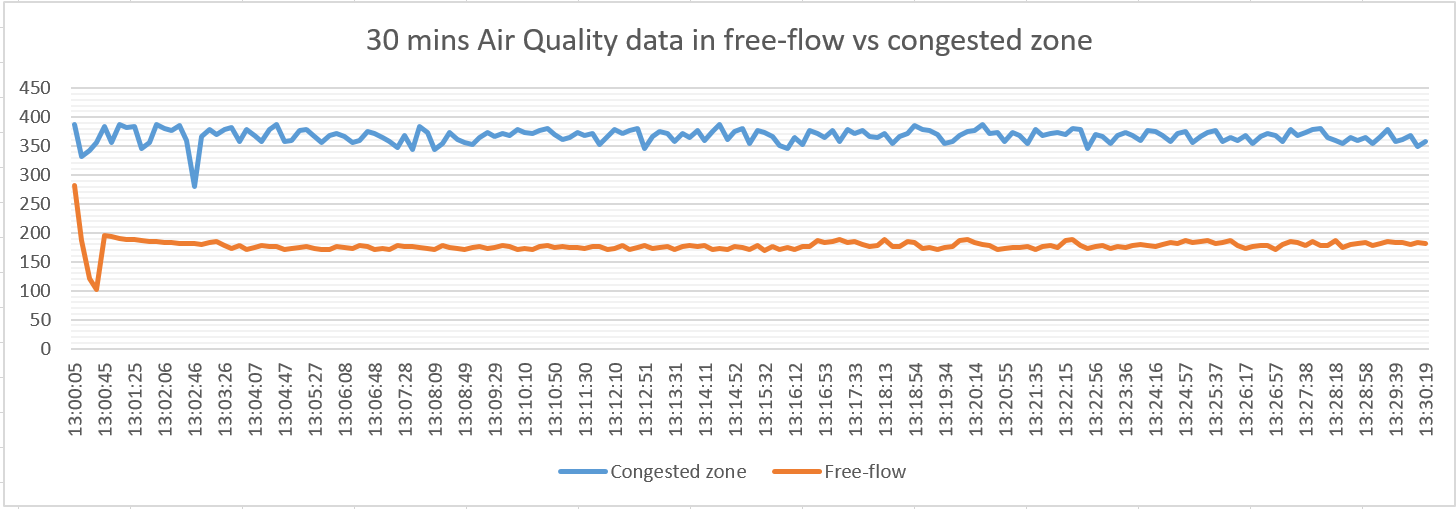


Figure 18: Air Quality Comparison

Showing the air quality comparison of congested vs free flow zone

The back ground behind this graph is that we install the devices at congestion place and free flow place at the same time to compare the readings. The graph shows the maximum value of air quality we received is 450 ppm in congestion zone. Also, the maximum value of air quality we received is free flow zone is 250 ppm.

### Temperature Readings

In continuation with the previous readings we also installed the device which contains the temperature sensor that measure temperature.

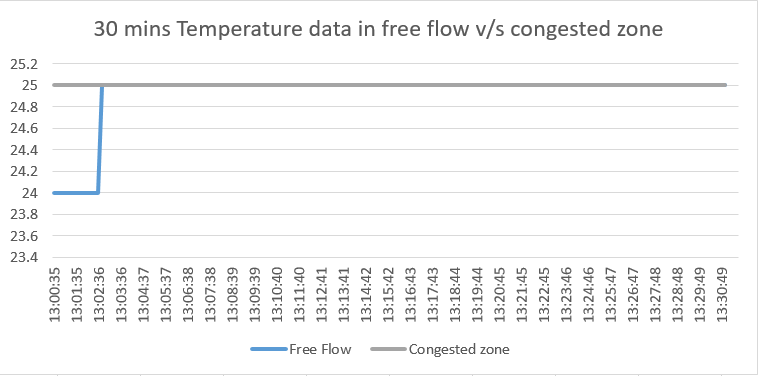


Figure 19: Temperature Readings

Showing readings of temperature which depicts that it remain constant

The back ground behind this graph is that we install the devices at congestion place and free flow place at the same time to compare the readings. The graph shows the maximum value of temperature we received is 25 degrees in congestion zone. Also, the maximum value of air quality we received is free flow zone is 25 degree.

### Sound Readings

In continuation with the previous readings we also installed the device which contains the sound sensor which measure the sound.

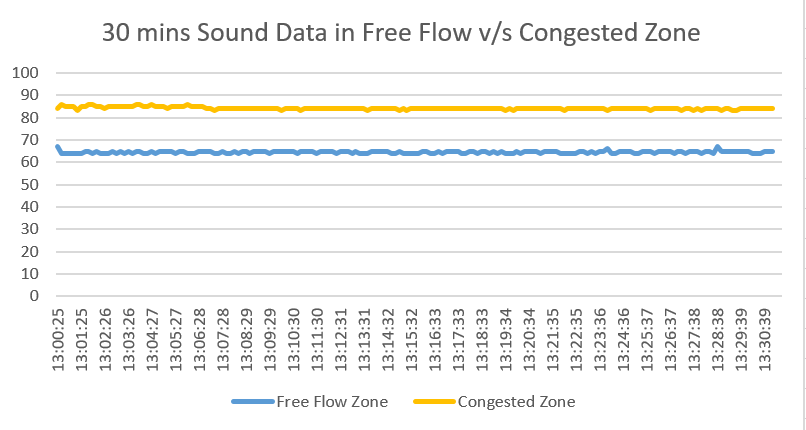


Figure 20: Sound Readings

Showing the comparison of sound in congested vs free flow zone

The back ground behind this graph is that we install the devices at congestion place and free flow place at the same time to compare the readings. The graph shows the maximum value of sound we received is 90 db in congestion zone. Also, the maximum value of sound we received in free flow zone is 65 db.

### Humidity Readings

The sensor which measures temperature also has the capability to measure humidity as well. Attached below is the graph of humidity.

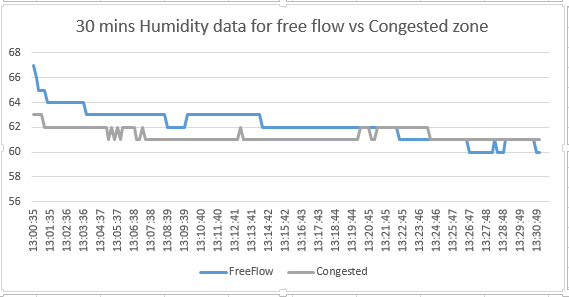


Figure 21:Humidity Readings

Showing the reading of humidity in comparison with free flow zone

The above graph shows the comparison of humidity data gathered from flow free and congestion zone. Maximum value at congestion zone was around 67 and maximum value at free flow zone was around 63.

### Front Development of App

Apart from doing hardware work, we have made the front end of the mobile application. The front end is purely made in Android Studio using XML. Front end is integrated with back end using java as a core language. The respective version and other necessary details about each of these implementations is listed as.

#### Android Studio

For the App development we have used the Android Studio with version 3.6 is used and that the following libraries and dependencies are installed:

* Java 8 update 241 CPU
* XML version 1.0 encoding utf-8
* Firebase com. google. Fire base. 16.0.4
* Graphs com. Graph view. 4.2.2
* Recycler View android. Recycler view. 1.1.0

### Back end Development of App

When it comes to back end development we have used cloud fire store which is again part of google firebase it is real time storage that keeps track of any changes that might happen in the storage i.e. if there is even small change in the database it will be automatically be reflected to the app. Moreover, another important feature is that this storage can be used offline i.e. we can store data offline and it will be reflected to the database when we come online.

### Programming the Arduino Board

To program Arduino Uno, there is a special IDE (Integrated Development Environment) which gets installed in local computer. We used Arduino IDE with version 1.8.12. The respective version and other necessary details about each of these implementations is listed as.

#### Dependencies in Programming Arduino Board

The Coding in Arduino IDE is done in C++ with additional special methods and functions that are capable of controlling and managing microcontroller. The IDE allows us to edit, compile and upload code called sketch in Arduino language. The respective version and other necessary details about each of these implementations is listed as.

* C++ 17 version
* USB cable for connecting to the Pc
* Arduino IDE version greater than 1.5.1

### Analysis of Data

At first, we have to make data linearly separable which we have collected. Through this, quality data will be stored. Then we will use split ratio through which data will come in random state. Then we will make subplot of data with comparison b/w air quality and data time. Here standard deviation will be used to update data. Then a univariate model is created. After that with help of labels and markers, we will create a plot of history and true future.

Then a baseline is made to define history. A plot is drawn here which shows true future and model prediction. Retaining of data will be done. A sequential model from keras will be used here which will show that air quality will not get that bad enough.

A model prediction will be done with LSTM. There will be a minimal difference between True future and Model prediction. Important features like Air quality and temperature will be considered here. The data will have become linearly separable too. Now we have use multivariate analysis to check data on each step. Data will be again trained here with past history and future target. The data have changed at some point in different columns here.

The sequential loss algorithm Mae is used here to get loss. The loss we get here is same we get above from univariate analysis. Now history data will be stored at one place. Overfit function will be used here to do prediction at every single step. Then Ridge Regression is used to do statistical modification. Then multi step model which will be consisting of Relu Activation function along with Loss algorithm Mae is used here. The actual air quality with respect to time improves and there is almost no validation loss.

## Test case Design and description

The following portion of this document will discuss the various test cases of the Smart City Pollution Monitoring System using Machine Learning Analytics.

### Login Account with Correct Credentials

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Login Application** | | | | | |
| **UC-1** | | | | | |
| Test Case ID: | | 4.2.1.1 | QA Test Engineer: | | Shamoon Shahid |
| Test case Version: | | 1 | Reviewed By: | | Usama Ali |
| Test Date: | | 20/03/2020 | Use Case Reference(s): | | UC-1 |
| Revision History: | | N/A | | | |
| Objective | | Registered users must be able to login. | | | |
| Product/Ver/Module: | | Android Application*.* | | | |
| Environment: | | Android Application must be installed in emulator/phone. | | | |
| Assumptions: | | Assuming that the user is already registered and phone is having internet excess. | | | |
| Pre-Requisite: | | No Pre-Requisite. | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 | User enters email id and password. | | |  | |
| 2 | User presses the login button. | | | System authenticates the user and upon verification, redirects them to their dashboard. | |
| Comments:  System works as expected. | | | | | |
| *Passed* *Failed* *Not Executed* | | | | | |

### Signup with Correct Credentials

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sign up for Application** | | | | | |
| **UC-2** | | | | | |
| Test Case ID: | | 4.2.3.1 | QA Test Engineer: | | Shamoon Shahid |
| Test case Version: | | 1 | Reviewed By: | | Usama Ali |
| Test Date: | | 20/03/2020 | Use Case Reference(s): | | UC-2 |
| Revision History: | | N/A | | | |
| Objective | | User must be allowed to register. | | | |
| Product/Ver/Module: | | Android Application*.* | | | |
| Environment: | | Android Application must be installed in emulator/phone | | | |
| Assumptions: | | Assuming that the user has opened the correct and phone is having internet excess. | | | |
| Pre-Requisite: | | No Pre-Requisite. | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 | User enters name, email and password. | | |  | |
| 2 | User presses the Signup button. | | | System authenticates the user and upon verification, moves to the login page. | |
| Comments:  System works as expected. | | | | | |
| *Passed Failed Not Executed* | | | | | |

### Selecting a Location to see Readings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Selecting a location to see reading** | | | | | |
| **UC-3** | | | | | |
| Test Case ID: | | 4.2.4.1 | QA Test Engineer: | | Shamoon Shahid |
| Test case Version: | | 1 | Reviewed By: | | Usama Ali |
| Test Date: | | 20/03/2020 | Use Case Reference(s): | | UC-3 |
| Revision History: | | N/A | | | |
| Objective | | User must be allowed to select a location. | | | |
| Product/Ver/Module: | | Android Application*.* | | | |
| Environment: | | Android Application must be installed in emulator/phone | | | |
| Assumptions: | | Assuming that the user is login with correct and phone is having internet excess | | | |
| Pre-Requisite: | | No Pre-Requisite | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 | User clicks the location spinner to see list of locations. | | |  | |
| 2 | User selects one of the locations from location spinner. | | |  | |
| 3 | User clicks to move to the next screen. | | | System checks whether the correction location is selected by user. | |
| Comments:  System works as expected. | | | | | |
| *Passed Failed Not Executed* | | | | | |

### Showing the Current Reading of Air Quality

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Viewing the current Air Quality from the App** | | | | | |
| **UC-3** | | | | | |
| Test Case ID: | | 4.2.5.1 | QA Test Engineer: | | Shamon Shahid |
| Test case Version: | | 1 | Reviewed By: | | Usama Ali |
| Test Date: | | 20/03/2020 | Use Case Reference(s): | | UC-3 |
| Revision History: | | N/A | | | |
| Objective | | User must be allowed to see readings of location which he/she selected. | | | |
| Product/Ver/Module: | | Android Application*.* | | | |
| Environment: | | Android Application must be installed in emulator/phone | | | |
| Assumptions: | | Assuming that the user is login with correct and phone is having internet excess | | | |
| Pre-Requisite: | | No Pre-Requisite | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 | User clicks button to reach home screen. | | | System displays the current Air Quality Index, with the last updated time. | |
| 2 | User clicks on update now button. | | | System again displays the current Air Quality Index, with updated time. | |
| Comments:  System does not work properly because particular functionality is not implemented. | | | | | |
| *Passed Failed Not Executed* | | | | | |

### Transferring of data from Device to Cloud

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sign up for Application** | | | | | |
| **UC-3** | | | | | |
| Test Case ID: | | 4.2.6.1 | QA Test Engineer: | | Aleem Qasim |
| Test case Version: | | 1 | Reviewed By: | | Usama Ali |
| Test Date: | | 20/03/2020 | Use Case Reference(s): | | UC-3 |
| Revision History: | | N/A | | | |
| Objective | | User must be allowed to see readings of location which he/she selected. | | | |
| Product/Ver/Module: | | Android Application*.* | | | |
| Environment: | | Android Application must be installed in emulator/phone | | | |
| Assumptions: | | Assuming that the device is working properly while in connection with the cloud. | | | |
| Pre-Requisite: | | No Pre-Requisite | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 | User clicks button to reach home screen. | | | System displays the current Air Quality Index, with the last updated time. | |
| 2 | User clicks on update now button. | | | System again displays the current Air Quality Index, with updated time. | |
| Comments:  Transferring of data from device to cloud failed, due to compatibility issues with Wi-Fi module. | | | | | |
| *Passed Failed Not Executed* | | | | | |

### View Predicted Value of Air Quality

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sign up for Application** | | | | | |
| **UC-4** | | | | | |
| Test Case ID: | | 4.2.7.1 | QA Test Engineer: | | Shamoon Shahid |
| Test case Version: | | 1 | Reviewed By: | | Usama Ali |
| Test Date: | | 22/03/2020 | Use Case Reference(s): | | UC-4 |
| Revision History: | | N/A | | | |
| Objective | | User must be allowed to see predicted value of air quality of a particular location. | | | |
| Product/Ver/Module: | | Android Application*.* | | | |
| Environment: | | Android Application must be installed in emulator/phone | | | |
| Assumptions: | | Assuming that the device is working properly while in connection with the cloud. | | | |
| Pre-Requisite: | | No Pre-Requisite | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 | User clicks predicted value button from the menu after login. | | | System displays the estimated value of Air index with the Graph. | |
| 2 | User sees the predicted index of air quality. | | |  | |
| Comments:  This functionality is not implemented as yet, Machine learning work is still under process. | | | | | |
| *Passed Failed Not Executed* | | | | | |

### Saving a Particular Value of Air Quality

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sign up for Application** | | | | | |
| **UC-5** | | | | | |
| Test Case ID: | | 4.2.8.1 | QA Test Engineer: | | Shamoon Shahid |
| Test case Version: | | 1 | Reviewed By: | | Aleem Qasim |
| Test Date: | | 29/03/2020 | Use Case Reference(s): | | UC-5 |
| Revision History: | | N/A | | | |
| Objective | | User must be allowed to save the particular value of air quality of a particular location. | | | |
| Product/Ver/Module: | | Android Application*.* | | | |
| Environment: | | Android Application must be installed in emulator/phone | | | |
| Assumptions: | | Assuming that the device is working properly while in connection with the cloud. | | | |
| Pre-Requisite: | | No Pre-Requisite | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 | User clicks on the current reading button to see the current value of Air Quality | | | System displays the current value of Air Quality | |
| 2 | User sees the current value of Air Quality | | |  | |
| 3 | From Menu, User clicks on save Air Quality value | | | System saved the value and displays a message on the screen that reading has been stored | |
| Comments:  System works as expected. | | | | | |
| *Passed Failed Not Executed* | | | | | |

### Sending a Notification from the System to User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sign up for Application** | | | | | |
| **UC-6** | | | | | |
| Test Case ID: | | 4.2.9.1 | QA Test Engineer: | | Shamoon Shahid |
| Test case Version: | | 1 | Reviewed By: | | Aleem Qasim |
| Test Date: | | 29/03/2020 | Use Case Reference(s): | | UC-6 |
| Revision History: | | N/A | | | |
| Objective | | A Notification must be sent to the user if Air Quality value exceeds a particular threshold. | | | |
| Product/Ver/Module: | | Android Application*.* | | | |
| Environment: | | Android Application must be installed in emulator/phone | | | |
| Assumptions: | | Assuming that the device is working properly while in connection with the cloud. | | | |
| Pre-Requisite: | | No Pre-Requisite | | | |
| Step No. | Execution description | | | Procedure result | |
| 1 |  | | | System sends a notification to the user if the value of Air Quality of the particular area exceeds from certain threshold. | |
| Comments:  This functionality is not implemented as yet. | | | | | |
| *Passed Failed Not Executed* | | | | | |

## Test Metrics

Table 1: Test Metrics

|  |  |
| --- | --- |
| Metric: | Purpose |
| Number of Test Cases: | 10 |
| Number of Test Cases Passed: | 6 |
| Number of Test Cases Failed: | 2 |
| Test Case Defect Density: | 25% |
| Test Case Effectiveness: | 40% |

### Traceability Matrix

Table 2: Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Login Account with Correct Credentials | 4.2.1 | Implemented | Pass |
| Login Account with Incorrect Credentials | 4.2.2 | Implemented | Pass |
| Signup with Correct Credentials | 4.2.3 | Implemented | Pass |
| Selecting a Location to see readings | 4.2.4 | Implemented | Pass |
| Showing the Current reading of Air quality | 4.2.5 | Implemented | Pass |
| Transferring of Data from Device to Cloud | 4.2.6 | Implemented | Failed |
| View Predicted Value of Air Quality | 4.2.7 | Implemented | Failed |
| Save a Particular Value of Air Quality | 4.2.8 | Implemented | Pass |
| Sending Notification from System to User | 4.2.9 | Implemented | Pass |
| Usability of Application | 4.2.10 | Implemented | Pass |

# Chapter 5: Experimental Results and Analysis

As working with current sensors, we also have installed new sensors in another device which we are still trying to convert it into Wi-Fi module. To measure concentration of carbon monoxide we are using MQ7 Apart from MQ-7, we also have installed MQ-2 which is used to measure smoke in the air, actually what we are trying to do is to measure the components which affect the quality of air in the environment or the components on which air quality depends. Attach is the screen shot of the readings captured from the device which we have installed.

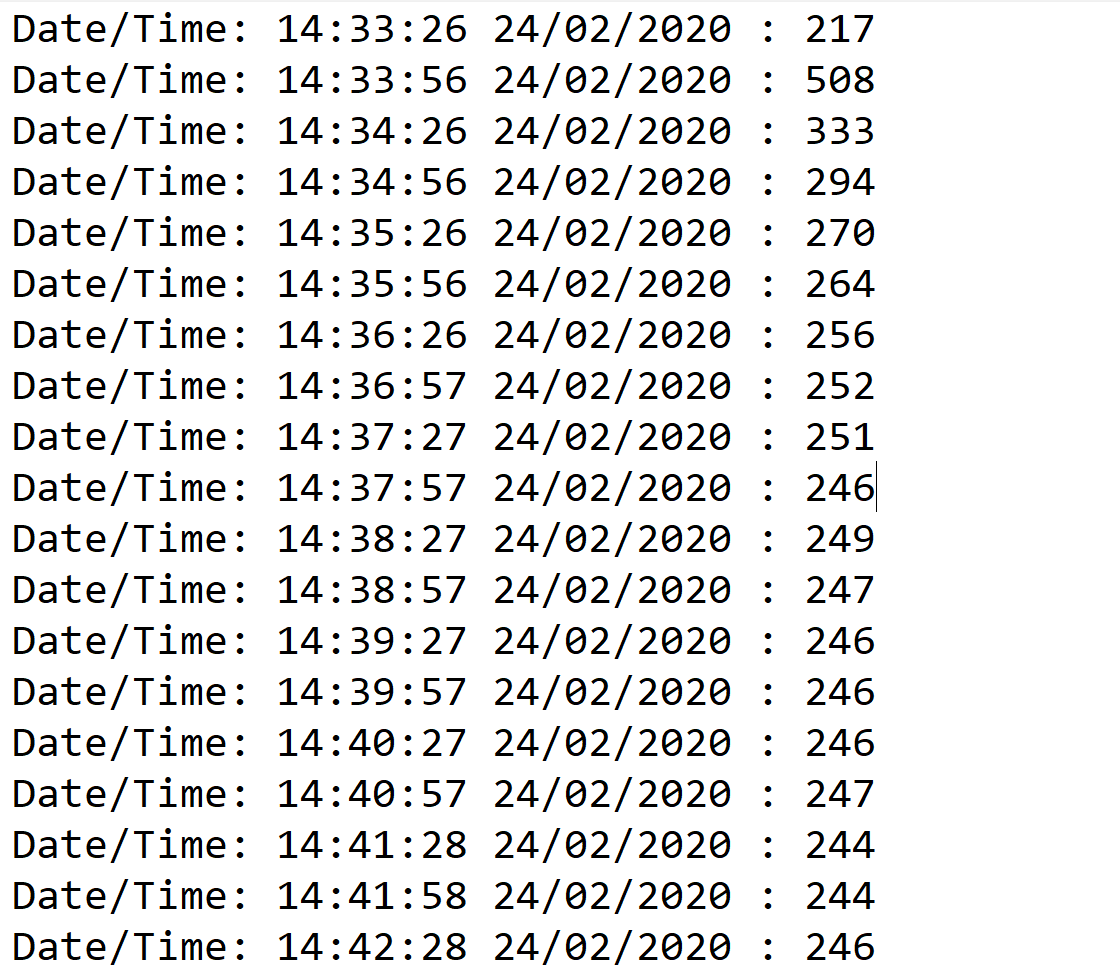
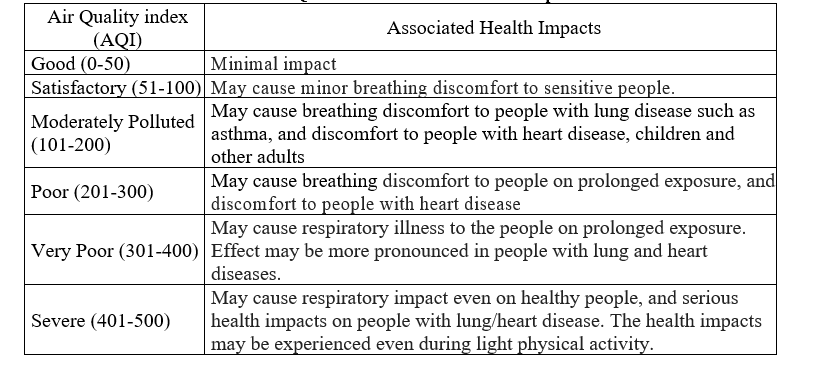


Figure 22: MQ-2 data

Showing the current reading of air quality

Apart from the installation of new sensors, a table is attached below which shows the Index of Air Quality along with its health impacts:

Table 3: AQI levels and Associated Health Impacts



After getting the data from our sensors, we had analysis of that data with the help of some machine learning algorithms and came to a conclusion that Air quality is not getting that much bad with respect to time in future but that can vary according to the dataset provided.

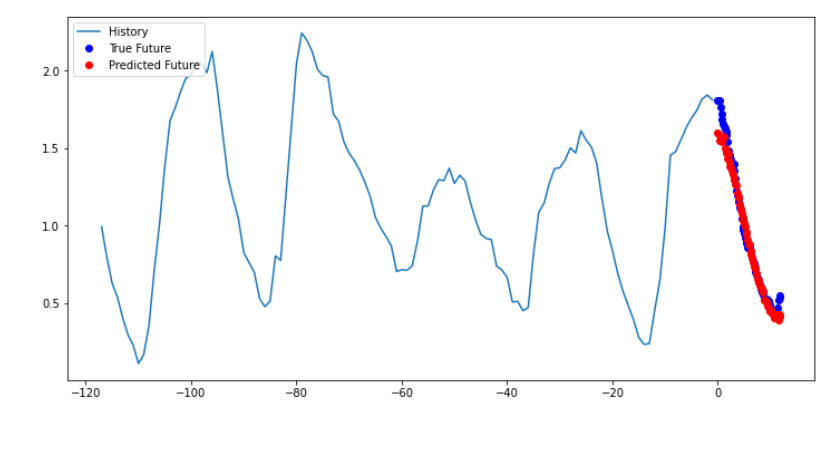


Figure 23: Air quality v/s Time

Showing the Air quality with respect to the time how it is changing

# Chapter 6: Conclusion

In the end, what we have done we have made a system for the people which they can use to affectively monitor the air quality by just clicking the android app Moreover we also have suggested the government a cheapest way of monitoring and analyzing air quality by installing boards and using iot to monitor affectively pollution. Also, our research and work on machine learning makes it also much easier to observe the predicted air quality, or to keep our selves to be ready for any unforeseen condition. The predicted air pollution can help the government to classify and make clusters of those area which air pollution is continuously going high from the certain level. However apart from our work on machine learning there are more points which other groups that if they are going to start work must consider like concentration of those gases in air that causes air pollution. What factors that lead to the production of these gases in the air. At last our android app is useable and can be use easily to detect and maintain air quality.

# References

The References are given are:

1. Dhingra, S., Babu Madda, R., H. Gandomi, A., Patan, R. and Danes, M. (2019). *Internet of Things Mobile–Air Pollution Monitoring System (IoT-Mobair) - IEEE Journals & Magazine*. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/8663367 [Accessed 3 Sep. 2019].
2. Pal, P., Gupta, R., Tiwari, S. and Sharma, A. (2016). IOT BASED AIR POLLUTION MONITORING SYSTEM USING ARDUINO. International Research Journal of Engineering and Technology (IRJET), [online] 04(10). Available at: https://www.academia.edu/35085518/IOT\_BASED\_AIR\_POLLUTION\_MONITORING\_SYSTEM\_USING\_ARDUINO?auto=download [Accessed 9 Feb. 2020].
3. Aqib, M. (2016). IOT Based Air Pollution Monitoring System using Arduino & MQ135 Sensor. [online] Circuitdigest.com. Available at: https://circuitdigest.com/microcontroller-projects/iot-air-pollution-monitoring-using-arduino [Accessed 15 Jan. 2020].
4. Engineer, T. (2018). Arduino Based Air Quality Monitoring IOT Project. [online] Hackster.io. Available at: https://www.hackster.io/TechnicalEngineer/arduino-based-air-quality-monitoring-iot-project-7f3d14#team [Accessed 25 Jan. 2020].
5. Kang, G., Gao, J., Chiao, S., Lu, S. and Xie, G. (2018). Air Quality Prediction: Big Data and Machine Learning Approaches. International Journal of Environmental Science and Development, [online] 9(1), pp.8-16. Available at: https://www.researchgate.net/publication/323012727\_Air\_Quality\_Prediction\_Big\_Data\_and\_Machine\_Learning\_Approaches [Accessed 8 Mar. 2020]

# Appendix

## Appendix A: Photo of Place where device installed

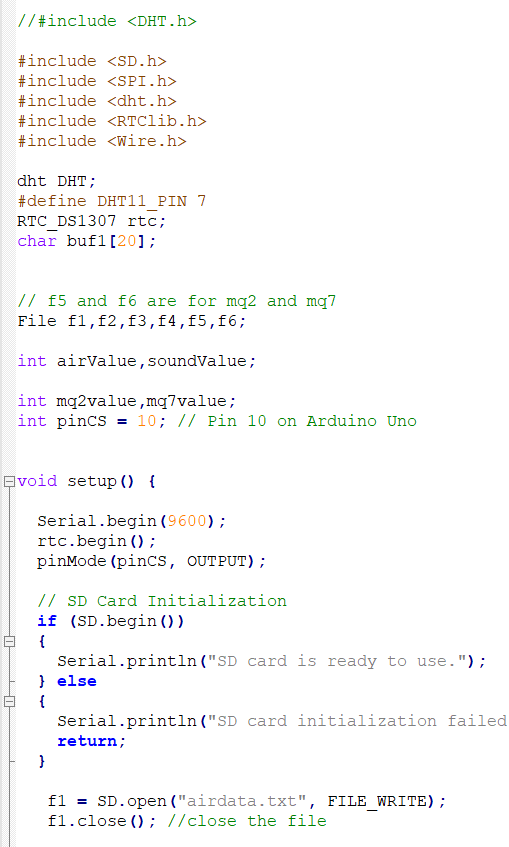


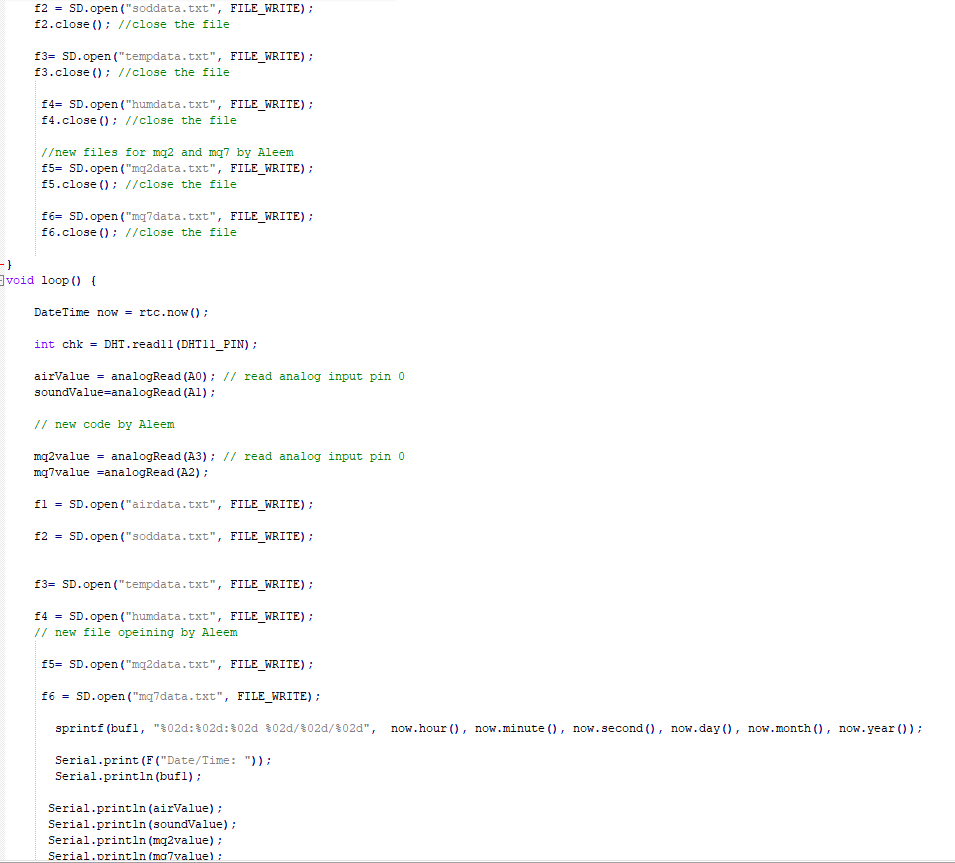
Figure 24: Congestion Zone

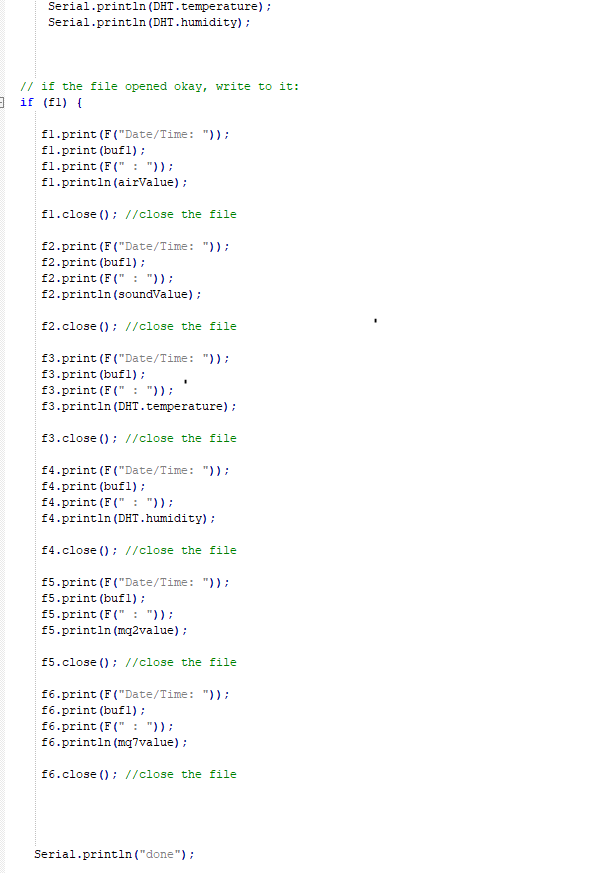
Showing the congested area where the reading was recorded

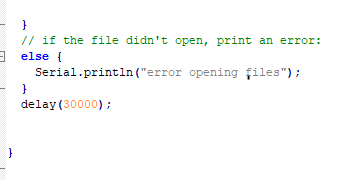
## Appendix B: Code which programmed Arduino.

The code which we have written to program Arduino board

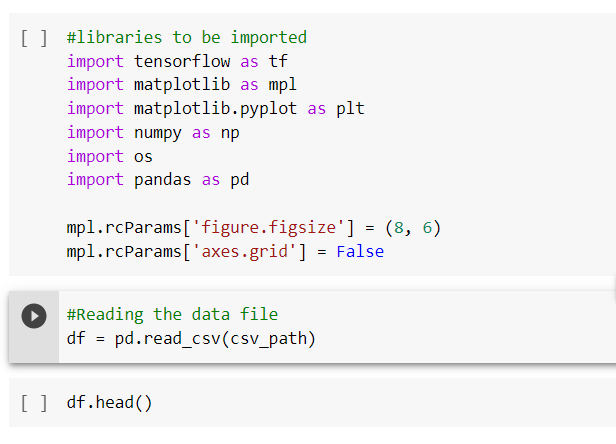


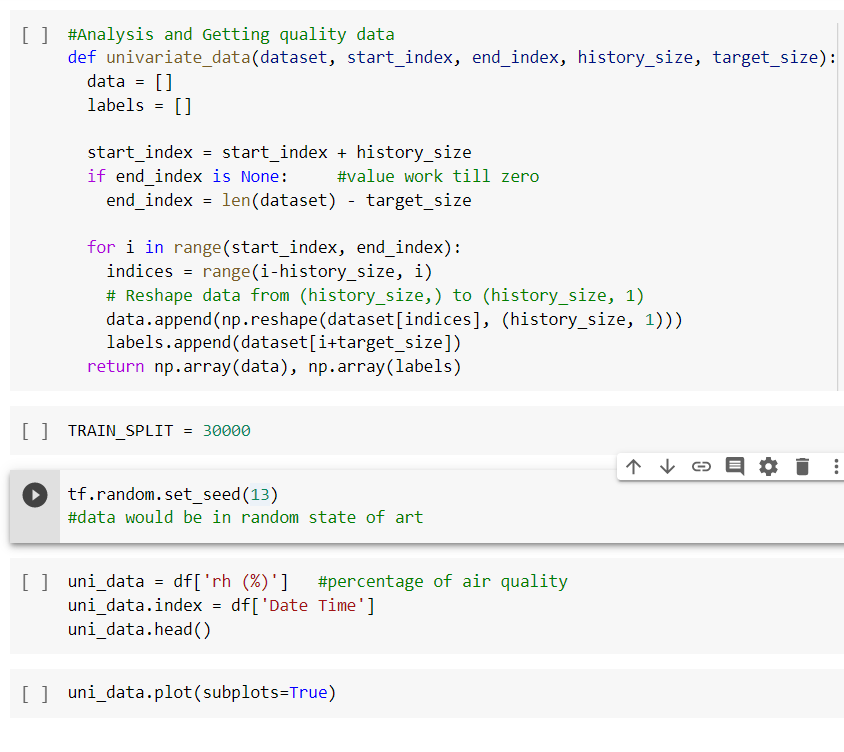


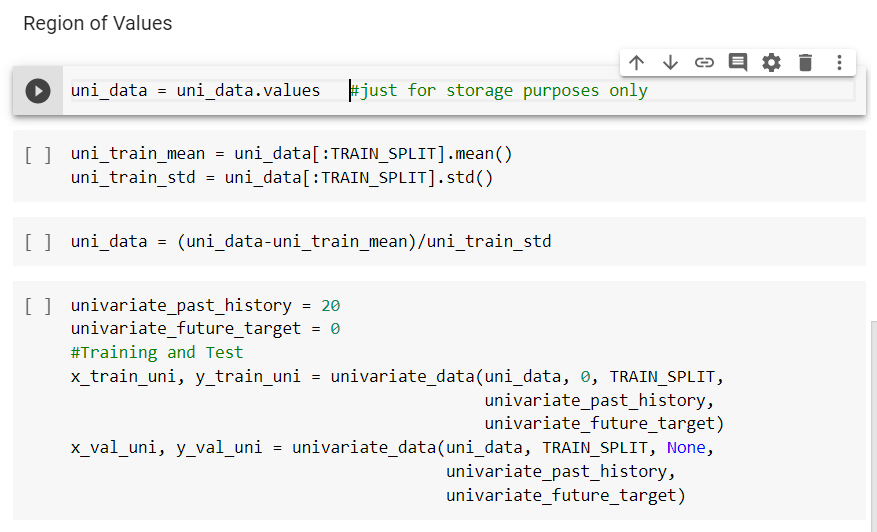


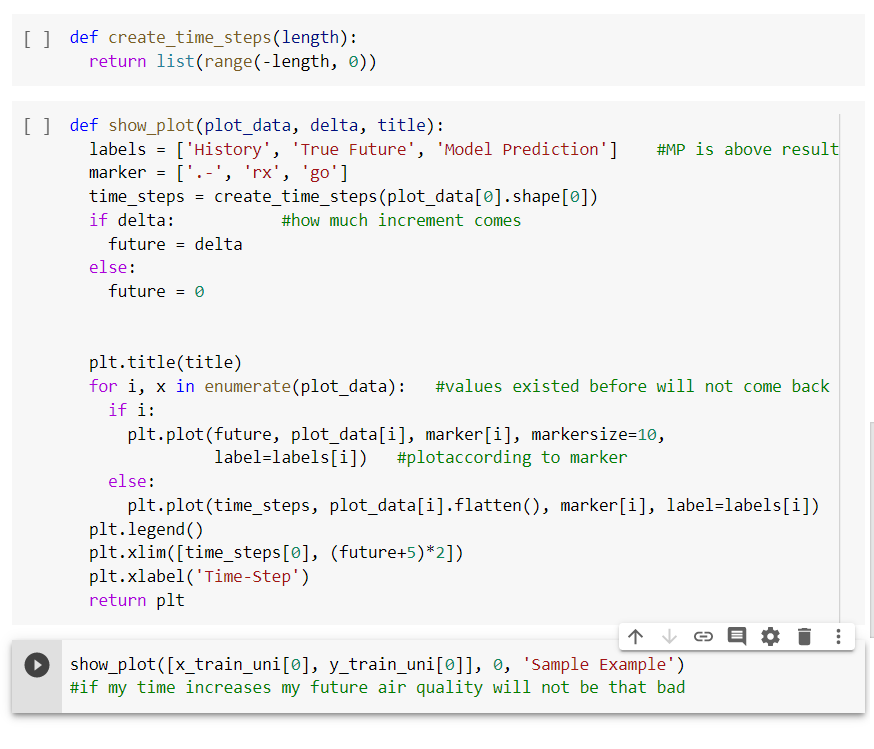


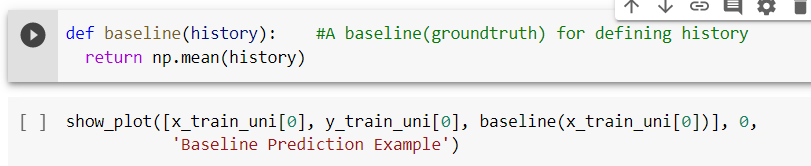
The code, which we have used in machine learning for some useful insights.

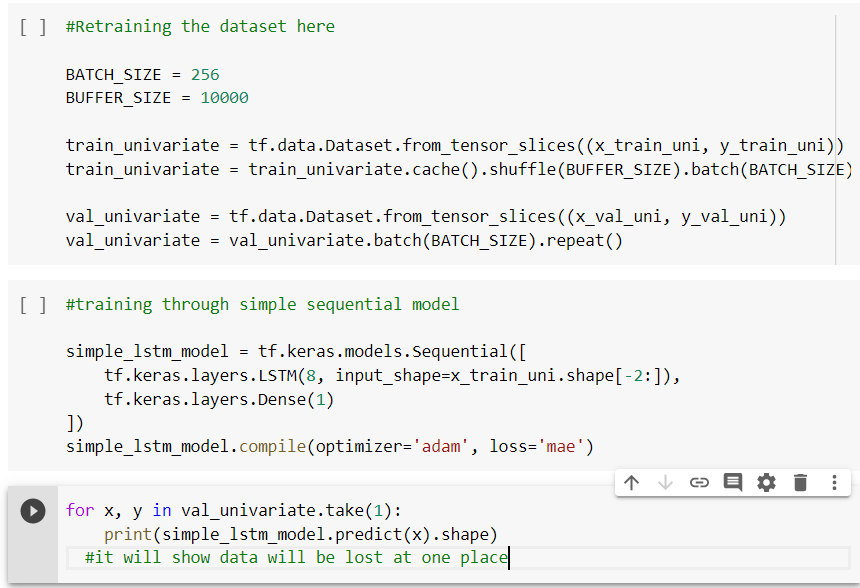


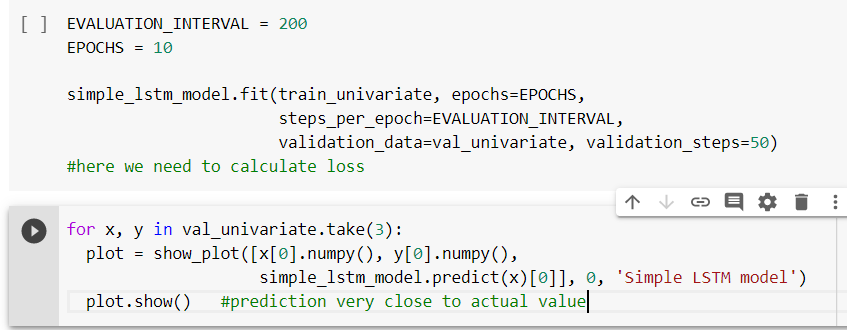


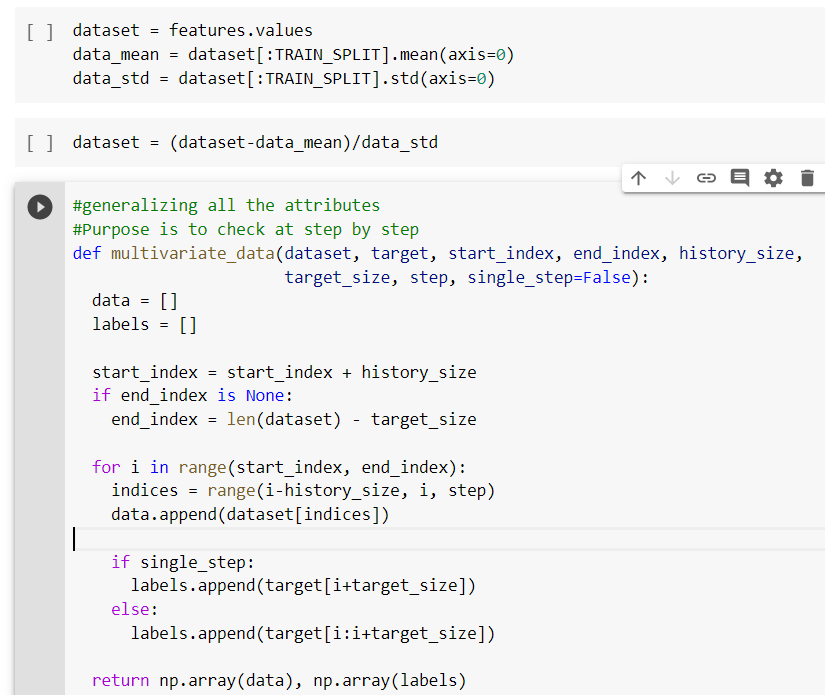


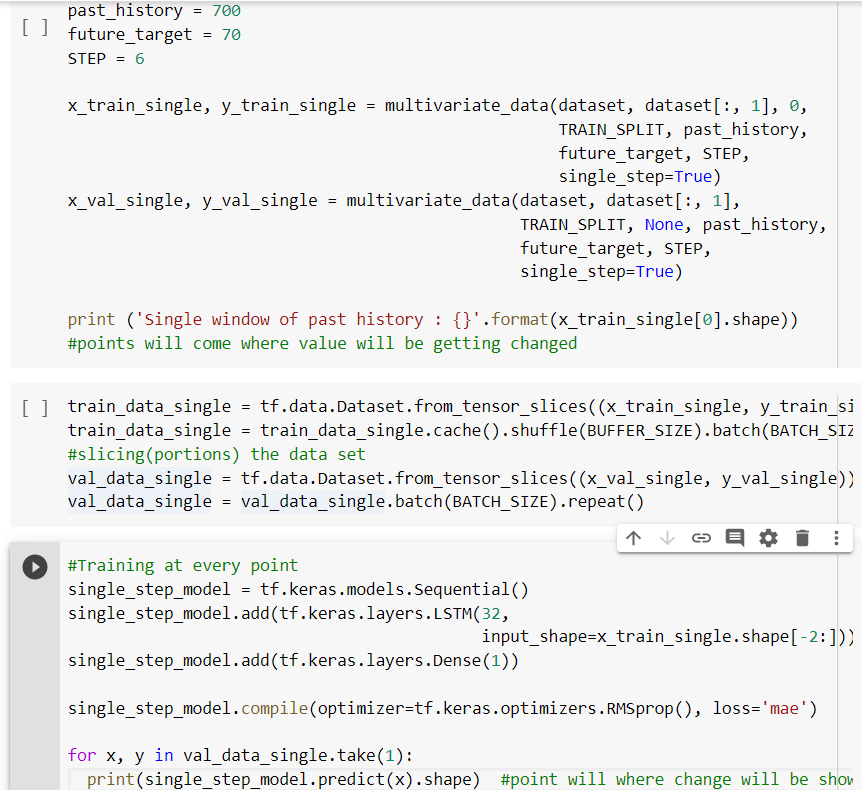


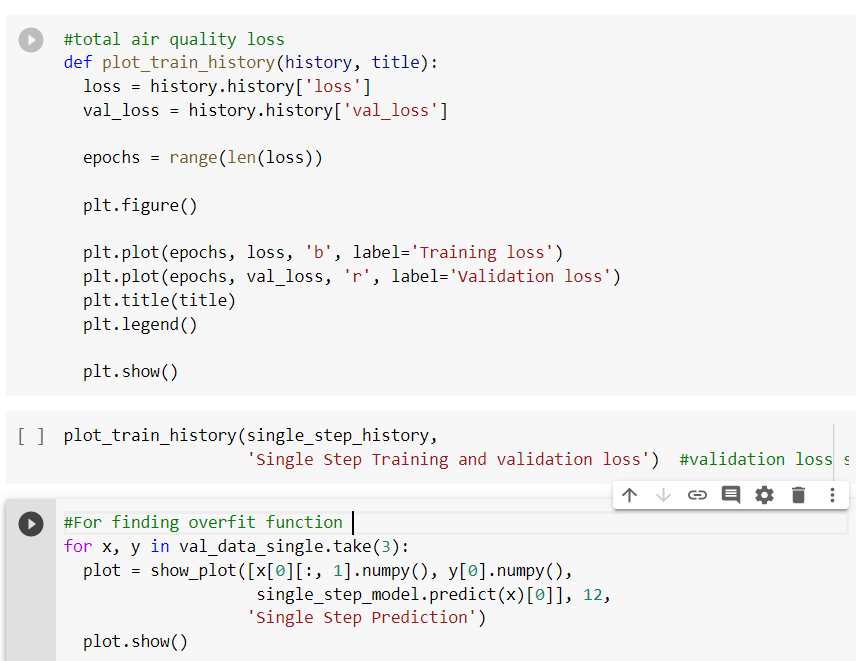












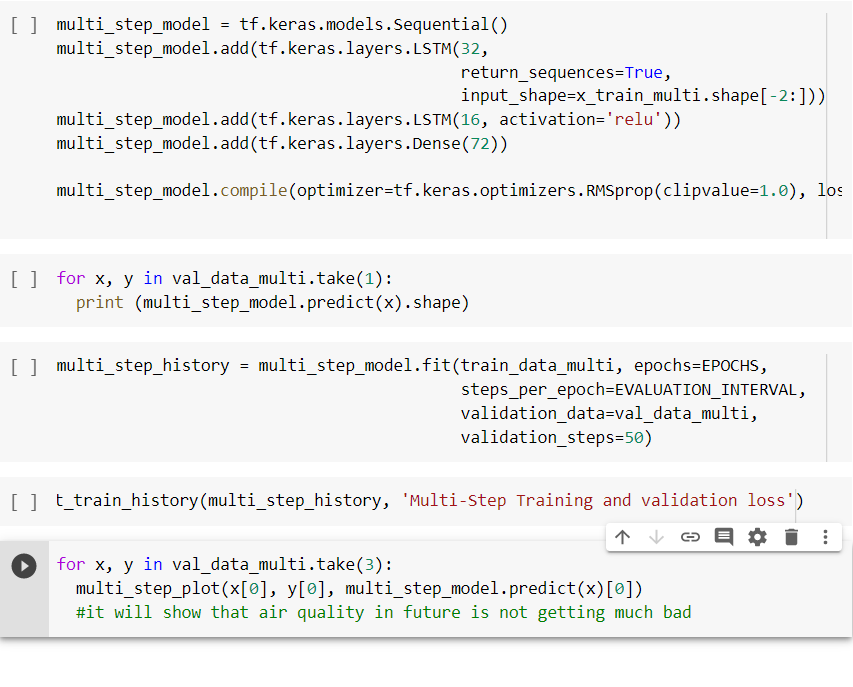


Figure 25: Machine Learning code

Showing the Machine Learning code