



## **LOW LEVEL DESIGN AND IMPLEMENTATION DOCUMENT**

### **Monitoring the concentration of air pollutants and its health hazards using Machine Learning models**

**UE20CS461A – Capstone Project Phase – 2**

*Submitted by:*

<b>Aditi Jain</b>	<b>PES2UG20CS021</b>
<b>Aditya R Shenoy</b>	<b>PES2UG20CS025</b>
<b>Ananya Adiga</b>	<b>PES2UG20CS043</b>
<b>Anirudha Anekal</b>	<b>PES2UG20CS051</b>

Under the guidance of

**Prof. Saritha**  
Assistant Professor  
PES University

**August - December 2023**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**FACULTY OF ENGINEERING**  
**PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Bengaluru – 560 100, Karnataka, India

## **TABLE OF CONTENTS**

1. Introduction	3
1.1 Overview	3
1.2 Purpose	3
1.3 Scope	4
2. Design Considerations, Assumptions and Dependencies	5
3. Design Description	6
3.1 Module Name	4
3.2 Module 1	4
3.2.1 Description	4
3.2.2 Use Case Diagram	4
3.2.3 Class Diagram	5
3.2.3.1 Class Description 1	6

3.2.3.2	Class Name 1	6
3.2.3.3	Data Members 1	6
3.2.3.4	Method 1	6
3.2.3.5	Method 2	7
3.2.3.6	Method n	7
3.2.3.7	Class Description 2	7
3.2.3.8	Class Name 2	7
3.2.3.9	Data Members 2	7
3.2.3.10	Method 1	7
3.2.3.11	Method 2	7
3.2.3.12	Method n	7
3.2.3.13	Class Name n	7
3.2.4	Sequence Diagram	7
3.2.5	Packaging and Deployment Diagram	8
4.	Proposed Methodology / Approach	9
4.1	Algorithm and Pseudocode	9
4.2	Implementation and Results	9
4.3	Further Exploration Plans and Timelines	9
Appendix A: Definitions, Acronyms and Abbreviations		9
Appendix B: References		9
Appendix C: Record of Change History		9
Appendix D: Traceability Matrix		10

## **1. Introduction**

### **1.1. Overview**

The low-level design document serves as a comprehensive technical roadmap for our air quality monitoring system. It will encompass the intricate details of the system's architecture and operation, offering insights into the hardware and software components that constitute it. We will meticulously discuss the selection of sensors and the methods employed to collect air quality data using Arduino boards, shedding light on the intricacies of data acquisition. Moreover, this document will delve into the core of our system, elucidating the implementation of the hybrid model, which combines Adaptive LSTM and ARIMA, known for their efficacy in handling time series data. The subsequent sections will delineate the strategies for deploying this model on a secure cloud platform, ensuring scalability and accessibility for users. The document will further elucidate how the system perpetually learns and adapts from real-time data, thereby enhancing its predictive capabilities over time. This low-level design is not merely a technical blueprint but a crucial enabler of our overarching goal - to enhance awareness and prediction

accuracy, potentially saving lives and fostering environmentally-conscious decisions in the realms of government and industry.

## **1.2. Purpose**

The primary purpose of this low-level design document is to provide a granular and in-depth understanding of how our air quality monitoring system functions at a technical level. It serves as a comprehensive reference for the project team, stakeholders, and future developers, enabling them to gain insights into the intricate mechanisms, design choices, and architectural details of the system. This document will facilitate effective communication and collaboration among team members by elucidating the system's components, data flows, and the algorithms behind our predictive model. Furthermore, it acts as a vital tool for quality assurance, allowing for meticulous verification and validation of the system's performance against the defined specifications.

Beyond its internal use, this document also plays a pivotal role in knowledge transfer and sustainability. It ensures that future maintenance, upgrades, or expansions of the system can be carried out with precision and ease, safeguarding the system's long-term viability. In essence, this low-level design document encapsulates the technical blueprint of our project, allowing us to bridge the gap between conceptualization and implementation, and ultimately, contribute to our overarching goal of raising awareness about air quality and improving predictions for the betterment of public health and environmental consciousness.

## **1.3. Scope**

The scope of this low-level design document encompasses a comprehensive and detailed examination of the technical aspects and operational intricacies of our air quality monitoring system. It outlines the specific architectural components, hardware and software configurations, algorithms, and data flow diagrams that constitute the system's core. The document will delve into the nitty-gritty details of the sensors utilized, the data collection and transmission mechanisms using Arduino boards, and the hybrid model implementation, highlighting how it effectively processes and analyzes air quality data. Moreover, it will provide insights into the secure cloud platform deployment strategy, enabling scalability and accessibility for end-users. The document extends its purview to describe the continuous learning processes and the ways in which the system adapts and improves its predictive capabilities over time.

Beyond the technical intricacies, the scope of this document also includes comprehensive documentation of coding standards, conventions, and best practices that the development team should adhere to during the implementation phase.

Additionally, it serves as a knowledge repository, enabling seamless knowledge transfer, fostering collaboration, and ensuring the sustainability of the system, which is critical for future maintenance and upgrades. This low-level design document will act as a critical reference point for project stakeholders, development teams, and future contributors, ensuring that our air quality monitoring system operates efficiently, accurately, and in alignment with our overarching goal of promoting public health and environmental awareness.

## **2. Design Constraints, Assumptions, and Dependencies**

- **Interoperability requirements-**

Basic information can be displayed on an LCD connected to the stations. More data can be viewed on a computer with a basic internet connection. All error messages will be sent to the cloud platform. Some errors can be displayed on the LCD.

- **Interface/protocol requirements-**

The accuracy of the data collected through IoT devices depends on the quality and reliability of the sensors used. There may also be biases in the data collected, which could affect the validity of the study results. Therefore, maintaining the performance of the IOT sensors is necessary.

- **Discuss the performance related issues as relevant-**

The adaptive LSTM model might not have good accuracy if the amount of data fed is limited. The hybrid model might not give good accuracy if the data is irregular and contains anomalies that should be removed.

- **End-user environment-**

Should provide real-time monitoring, continuously monitor the air quality using IoT devices, and update the database in real-time. The ML models should also be updated periodically to ensure that they remain accurate.

- **Availability of Resources-**

Finding a skin cancer dataset and a global lung cancer dataset.

- **Hardware or software environment-**

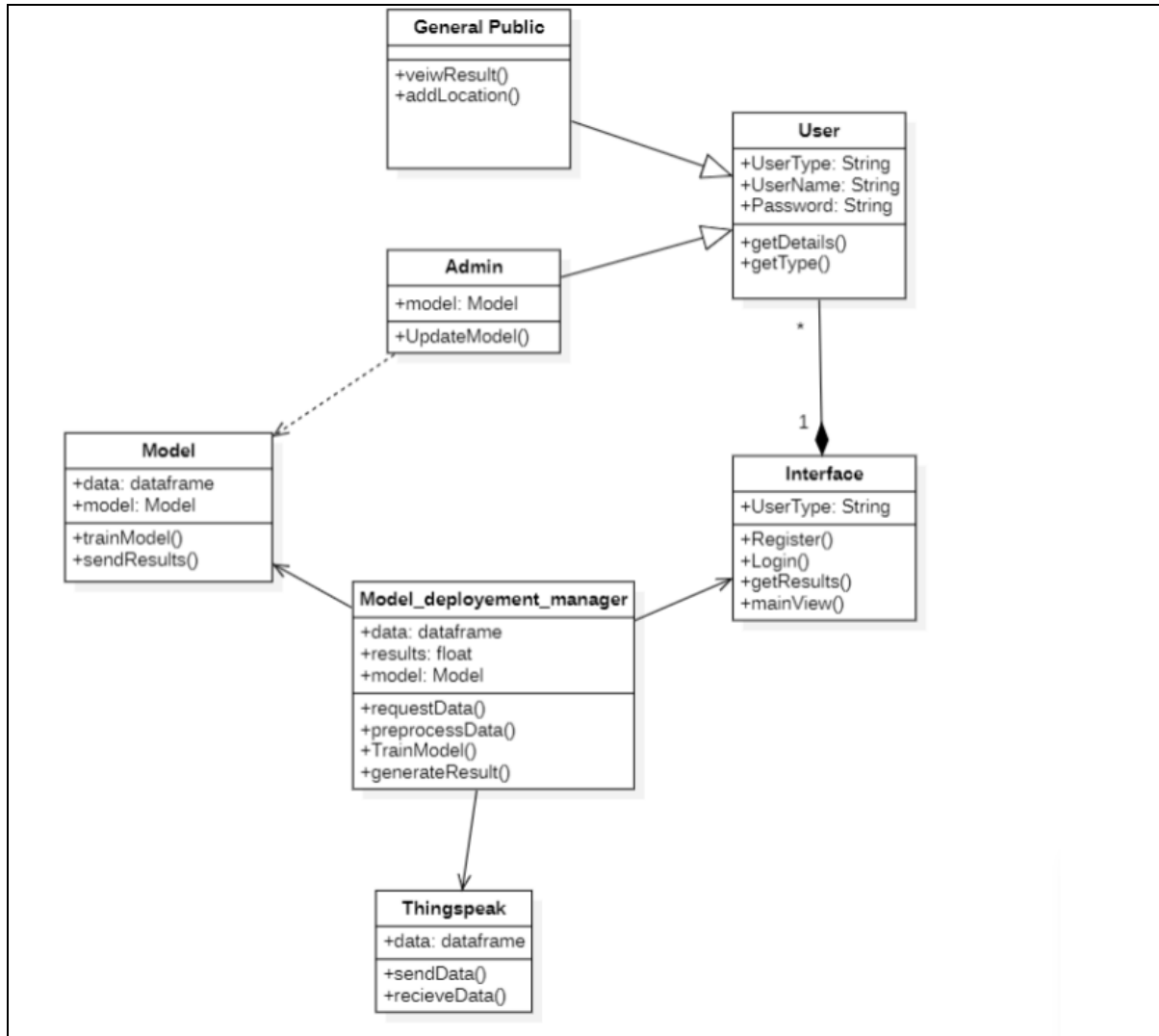
The amount of energy used by IoT technology is significant, and it needs to be running constantly.

- **Issues related to deployment in target environment, maintainability, scalability, and availability-**

The data produced by IoT devices is often unstructured and provides a limited perspective due to its massive size.

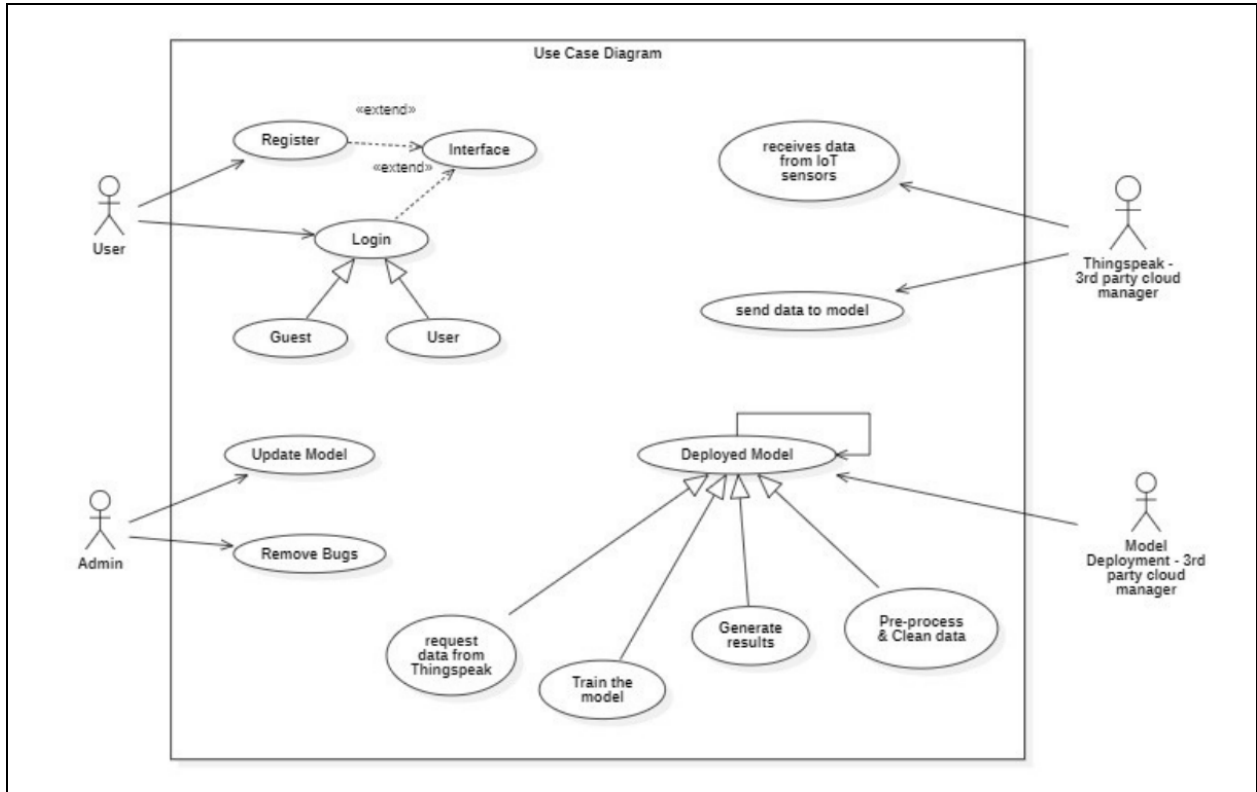
### **3. Design Description**

#### **3.1. Master Class Diagram**



**Fig- 1.0**

## 3.2. Use case diagram



**Fig- 1.1**

Use Case Item	Description
User	User will be an individual or entity that interacts with a system to see the predictions done on the air quality around them. User will register and/or login in the system and will be the source of data from which the model will be further trained.
Admin	Admin will be responsible for managing and maintaining a system or application. Admin will configure settings, manage user accounts, monitor system performance, and resolve technical issues. They will have the authority to access user details and patient details obtained from hospitals as well.
Cloud platform	The project will leverage two distinct cloud platforms: Thingspeak for sensor data reception and transmission, and Streamlit, Spaces, or AWS for hosting the machine learning model. These platforms will handle application deployment,



	resource scaling, data management, and performance monitoring.
--	--

Table- 1.0

## 4. Proposed Methodology / Approach

### 4.1. Algorithm and Pseudocode

Architecture diagram

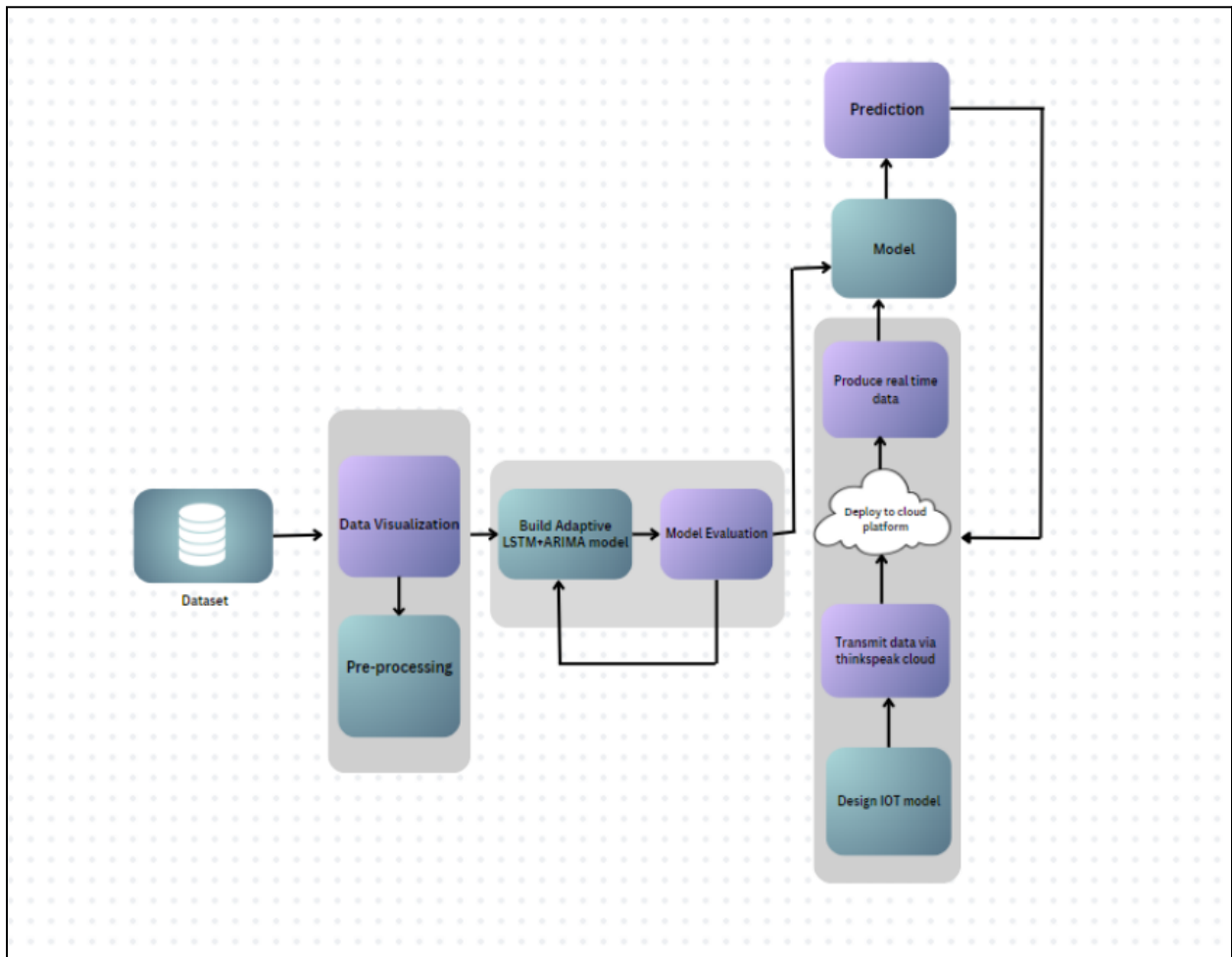


Fig- 1.2

#### Detailed flow:

- Data Pre-processing:
  - In the initial phase, the project team meticulously cleaned and pre-processed the time series data from the sensor stations. This involved handling missing

values, addressing potential outliers, and transforming variables to ensure data quality.

- The dataset was divided into distinct training and testing sets, with the training set designated for model training and the testing set reserved for evaluation purposes.
- **ARIMA Model Training:**
  - The team proceeded to fit an ARIMA (AutoRegressive Integrated Moving Average) model to the training data. ARIMA models were chosen for their capacity to capture seasonal and trend components in time series data.
  - Extensive hyperparameter optimization took place, including determining the order of differencing, autoregressive (AR) order, and moving average (MA) order. Optimization methods like grid search and time series cross-validation were employed to fine-tune the model.
- **ARIMA Forecasting:**
  - With the trained ARIMA model in place, short-term forecasts were generated on the testing data. These forecasts provided valuable insights into expected air quality based on historical data.
- **Residual Calculation:**
  - Residuals, representing the discrepancies between actual air quality values and ARIMA forecasts, were meticulously calculated. Residuals were computed for both the training and testing datasets.
- **Adaptive LSTM Model Training:**
  - The project team leveraged the ARIMA residuals as input data for training an Adaptive LSTM (Long Short-Term Memory) model on the training data. The LSTM model was essential for capturing residual patterns that may have eluded the ARIMA model.
- **Combining Forecasts:**
  - Forecasts were generated using the trained Adaptive LSTM model on the testing data. These LSTM-based forecasts were combined with the ARIMA forecasts, producing a hybrid forecast that offered enhanced accuracy and comprehensive predictions.

- **Evaluation and Optimization:**
  - Rigorous evaluation of the hybrid ARIMA-Adaptive LSTM model was undertaken using appropriate evaluation metrics on the testing data. This step ensured that the model met the desired standards for accuracy and reliability.
  - As necessary, the model underwent fine-tuning, which involved making adjustments to hyperparameters and modifying the model architecture to optimize predictive capabilities.
- **Real-time Data Collection:**
  - IoT sensor stations were constructed using ESP32 or ESP8266 modules, enabling real-time data collection capabilities. These modules were known for their capacity to connect to the internet.
  - The system utilized specialized sensors such as MQ135 for CO, CO<sub>2</sub>, and NH<sub>4</sub> detection, and GP2Y1010AU0F for dust detection. This allowed continuous and real-time collection of air quality data.
- **Deployment on Cloud Platforms:**
  - To facilitate seamless data reception and transmission, the project deployed the machine learning model and the sensor stations on cloud platforms, such as Thingspeak. This enabled real-time data processing and accessibility.
  - The machine learning model was hosted on cloud platforms like Streamlit, Spaces, or AWS, ensuring scalability and accessibility while enabling real-time data processing and analysis.

**IoT setup:**

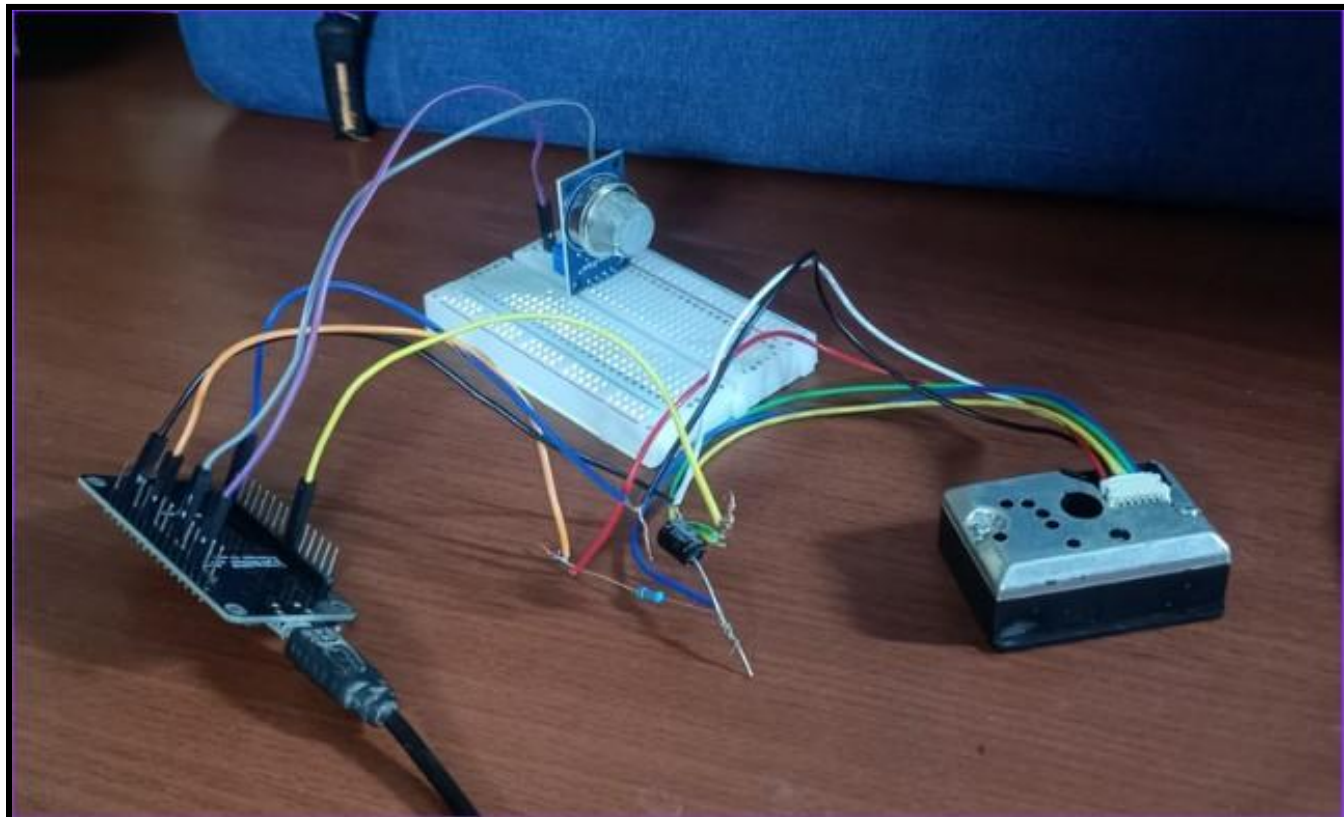
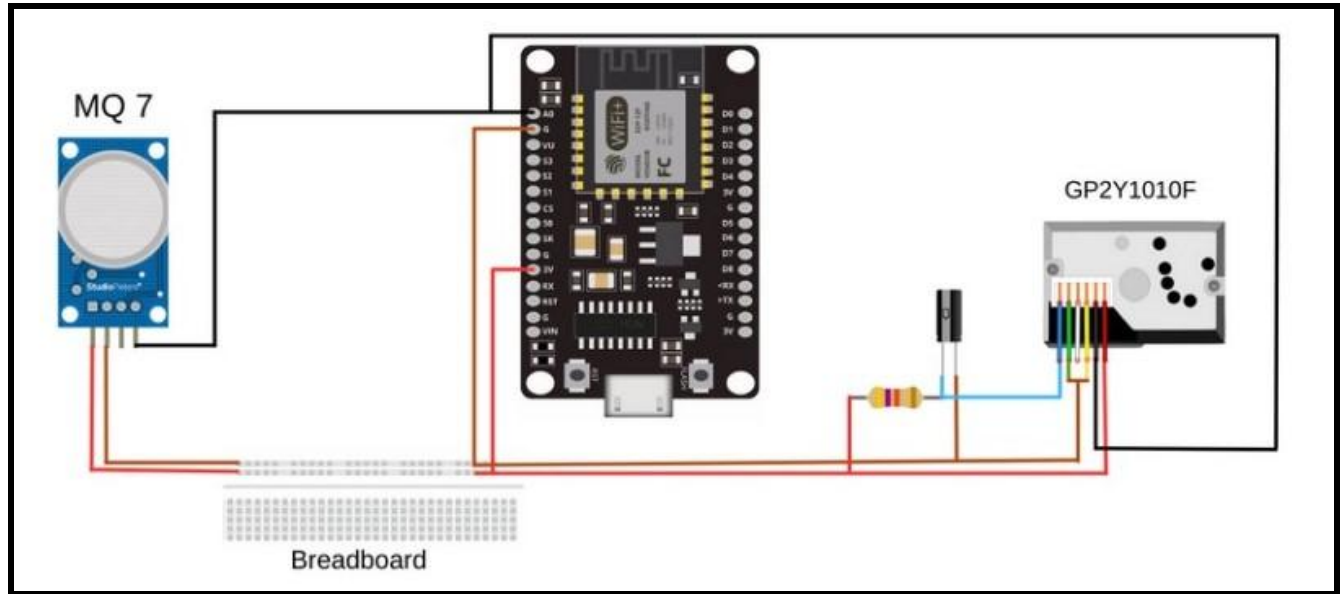


Fig 1.3

The project uses the ESP 8266 module for its Wi-Fi capabilities. It also uses the GP2Y1010AU0F dust sensor for measuring concentration of PM 2.5 and PM 10 and the MQ - 7 gas sensor to measure concentration of CO in the atmosphere..

The hardware setup is composed of two primary components directly wired to the ESP board:

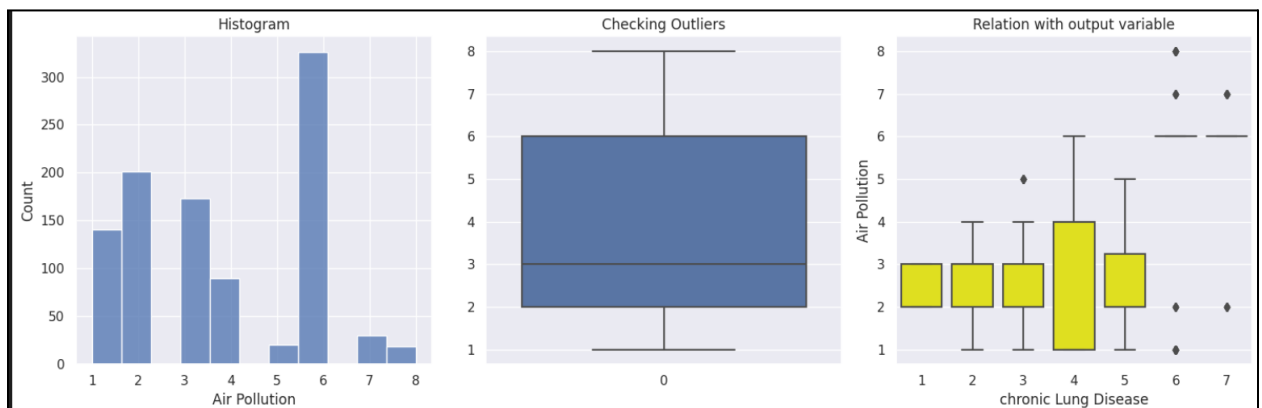
- **MQ 7 Sensor:** The MQ 7 gas sensor plays a critical role in detecting CO. We provide power to MQ 7 from digital pin 1. We take readings from it at analog 0 pin.
- **GP2Y1010AU0F Dust Sensor:** The GP2Y1010AU0F dust sensor is instrumental in particulate matter detection. We provide power to it from digital pin 2. We take its reading at analog pin 0

This meticulous hardware configuration enables the system to gather data from diverse sources, covering gasses and particulate matter. The flexibility in the ESP board selection and dust sensor options allows for adaptability based on factors like cost and availability, making the project versatile and well-suited for various operational scenarios. The elaborate and delicate wiring ensures the efficient flow of data between the hardware components and the ESP board, thereby facilitating accurate and reliable air quality monitoring.

### 4.2. Implementation and Results:

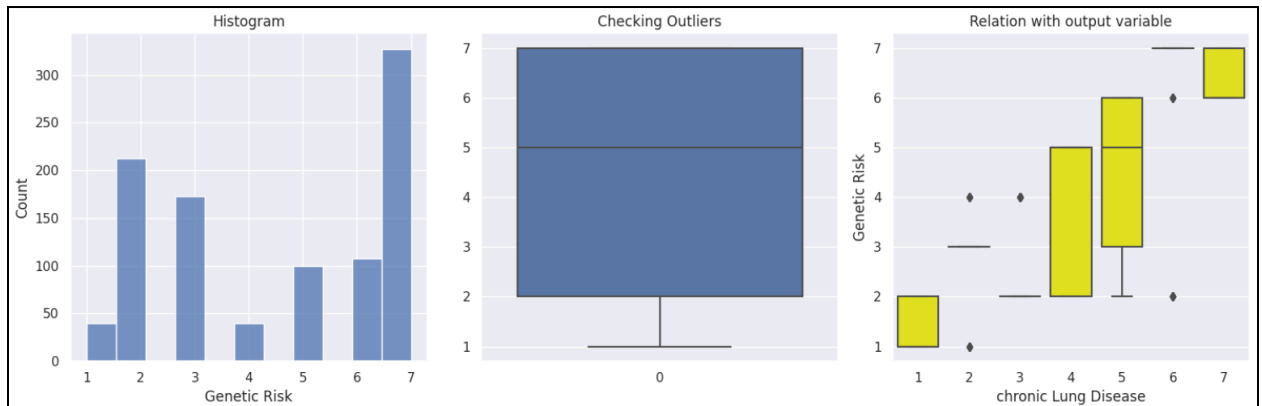
- **Data cleaning and preprocessing:**

As a part of data cleaning, data was visualized as a histogram and box plot to check for any outliers



**Fig- 1.4**

To remove the outliers , the values that are above the upper quartile range were removed. The plot looks as follows after removal of outliers



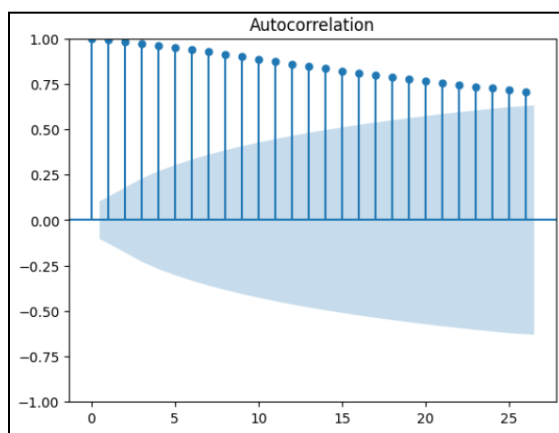
**Fig-1.5**

- **Splitting data:**

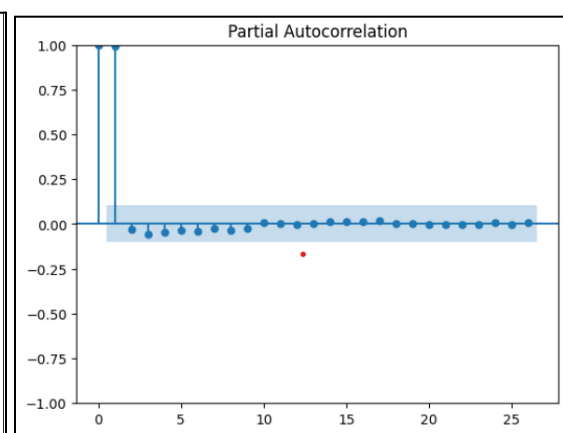
The data was split into 70% training set and 30% testing set for the ARIMA(Auto-Regressive-Integrated-Moving-Average)model.

- **Autocorrelation plots:**

- ☐ ARIMA model works on stationary data, to check if the data is stationary, autocorrelation plots- ACF and PACF are constructed.
- ☐ ACF plots represent the correlation of time series with its lags . It measures the linear relationship between lagged values of the time series.
- ☐ PACF plot represent partial correlation of time series with its lags, after removing the effects of lower-order-lags between them



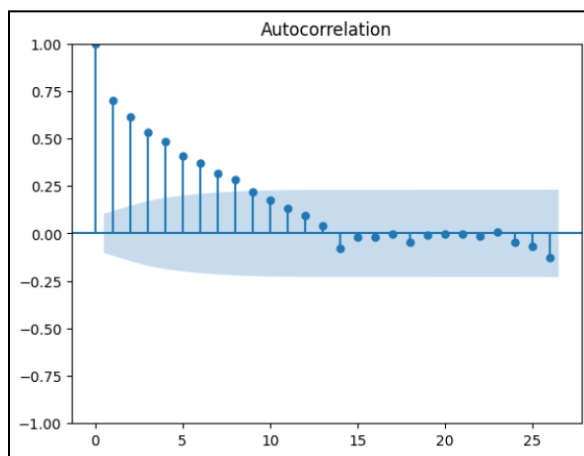
**Fig-1.6**



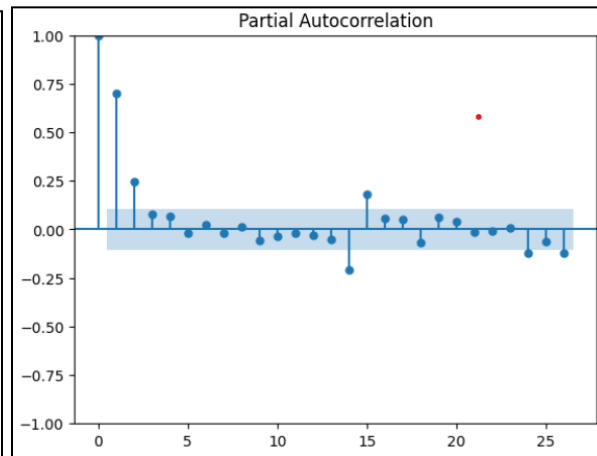
**Fig-1.7**

- ☐ The ACF plot shows the correlations with the lags are high and positive with very slow decay.
  - ☐ While the PACF plot shows the partial autocorrelations have a single spike at lag 1.
  - ☐ These are both signs of a trended time series. So the time series is not stationary.
- **Differencing:**
    - ☐ Since the data is not stationary we transform it to stationary data using differencing
    - ☐ Differencing stabilizes the mean of the time series by removing changes in the levels of the series.
    - ☐ It is the difference between the current observation and the previous observation, to get the new series.

After differencing the ACF and PACF plots look as follows-



**Fig-1.8**



**Fig-1.9**

- ☐ It can be seen that the trends are not that strong anymore and are more stationary compared to the original series.
- ☐ The ACF plot drops in value more quickly. While the PACF plot also shows a less strong spike at lag 1

- **Fitting the model:**

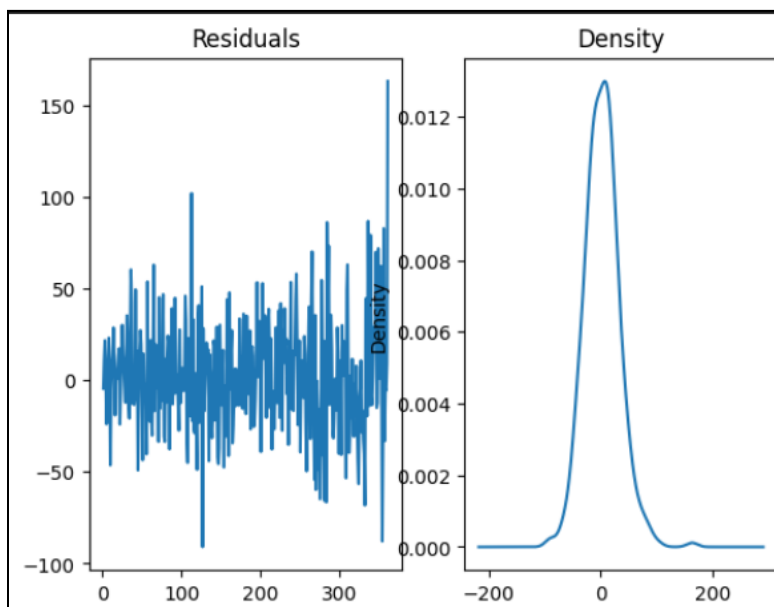
```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(df_train, order=(2,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

**Fig-2.0**

- ☐ The hyperparameter 'order' with the arguments p,d and q
- ☐ 'p' represents the number of autoregressive terms
- ☐ 'd' represents the number of nonseasonal differences needed for stationarity
- ☐ 'q' represents the number of lagged forecast errors in the prediction equation.
- ☐ These values are set after analyzing the ACF and PACF plots-
- ☐ 'p' value is set to 2, since PACF shows more minor but significant lags at 2, 4, and 5. In contrast, the ACF shows a more gradual decay. Values 2 or 3 or 4 can be used but to keep the model simple the value is taken as 2.
- ☐ 'd' being the value of differencing, it is better to start with the lowest value, usually the value 1 makes it stationary.
- ☐ If the PACF plot has a significant spike at lag p, but not beyond and the ACF plot decays more gradually. This may suggest an ARIMA(p, d, 0) model
- ☐ If the ACF plot has a significant spike at lag q, but not beyond and the PACF plot decays more gradually. This may suggest an ARIMA(0, d, q) model
- ☐ Since the plots show the former, 0 is used as the 'q' value.

- **Residuals of ARIMA model:**

The residuals of the ARIMA model are fed as the input to the adaptive LSTM model.





**Fig-2.1**

The residuals look random in general, and their density looks normally distributed with a mean of around 0.

These show that the residuals are close to white noise. We are ready to forecast with this model ARIMA(2, 1, 0).

- **Calculating error value-**

The error factor MAPE( Mean absolute percentage error) is a metric that defines the accuracy of a forecasting method . It represents the average of the absolute percentage errors of each entry in a dataset to calculate how accurate the forecasted quantities were in comparison with the actual quantities.

Formula

$$M = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right|$$

$M$  = mean absolute percentage error  
 $n$  = number of times the summation iteration happens  
 $A_t$  = actual value  
 $F_t$  = forecast value

```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error

rmse = np.sqrt(mean_squared_error(df_test, forecast_test))
mae = mean_absolute_error(df_test, forecast_test)
mape = mean_absolute_percentage_error(df_test, forecast_test)
#print(f'mae -: {mae}')
print(f'mape -: {mape}')
#print(f'rmse -: {rmse}')
```

mape -: 0.1801666326932173

**Fig- 2.2-2.3**

- **Defining Adaptive LSTM model:**

- ☐ Importing sequential and dense layer libraries
- ☐ Sequential neural networks is required for sequence processing
- ☐ Dense layer is for the output

- ☐ Both are required here because the input for prediction is real time

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = Sequential()
model.add(LSTM(64, input_shape=(1, 1)))
model.add(Dense(1))
```

**Fig-2.4**

- **Splitting the data:**  
The input given is split into 80% train and 20% test data
- **Training the model:**

```
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(train_X, train_y, epochs=10, batch_size=1, verbose=2)
```

**Fig-2.5**

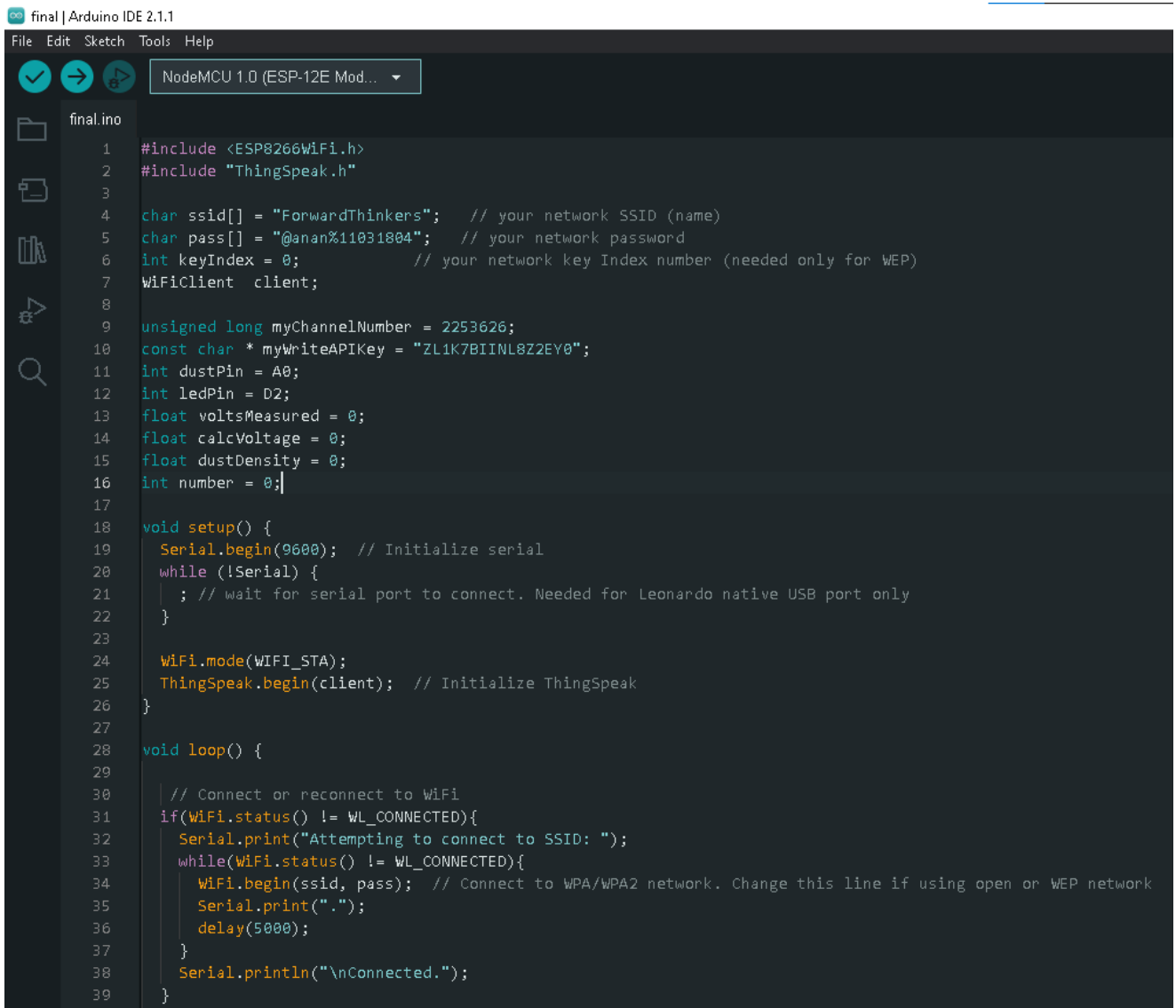
- ☐ The 'compile' method defined, configures the model for training
  - ☐ 'Loss' specifies the loss function that will be used to measure the error during training. Here, mean squared error (MSE). MSE is commonly used for regression problems, where the goal is to minimize the squared differences between predicted and actual values.
  - ☐ 'optimizer='adam' specifies the optimization algorithm that will be used during training. 'Adam' adapts the learning rate during training, which can lead to faster convergence.
  - ☐ epochs=10: the entire dataset will be passed forward and backward through the neural network during training 10 times
  - ☐ batch\_size=1: means that the model will update its weights after processing each individual sample.
  - ☐ verbose=2: means that it will display a progress bar for each epoch.
- **Calculate RMSE:**  
After making the predictions and flattening them, the loss function Mean Squared Error is calculated

```
mse = mean_squared_error(test_data, predictions)
rmse = np.sqrt(mse)
print(rmse)
```

40.274212657650004

**Fig-2.6**

- **Arduino IDE code for collecting readings from GP2Y1010F gas sensor and connection to ThingSpeak cloud platform**



```
final | Arduino IDE 2.1.1
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E Mod...
final.ino
1 #include <ESP8266WiFi.h>
2 #include "ThingSpeak.h"
3
4 char ssid[] = "ForwardThinkers"; // your network SSID (name)
5 char pass[] = "@anan%11031804"; // your network password
6 int keyIndex = 0; // your network key Index number (needed only for WEP)
7 WiFiClient client;
8
9 unsigned long myChannelNumber = 2253626;
10 const char * myWriteAPIKey = "ZL1K7BIINL8Z2EY0";
11 int dustPin = A0;
12 int ledPin = D2;
13 float voltsMeasured = 0;
14 float calcVoltage = 0;
15 float dustDensity = 0;
16 int number = 0;
17
18 void setup() {
19   Serial.begin(9600); // Initialize serial
20   while (!Serial) {
21     ; // wait for serial port to connect. Needed for Leonardo native USB port only
22   }
23
24   WiFi.mode(WIFI_STA);
25   ThingSpeak.begin(client); // Initialize ThingSpeak
26 }
27
28 void loop() {
29
30   // Connect or reconnect to WiFi
31   if(WiFi.status() != WL_CONNECTED){
32     Serial.print("Attempting to connect to SSID: ");
33     while(WiFi.status() != WL_CONNECTED){
34       WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP network
35       Serial.print(".");
36       delay(5000);
37     }
38     Serial.println("\nConnected.");
39   }
40 }
```

final | Arduino IDE 2.1.1

```

File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E Mod...

final.ino
40
41 digitalWrite(ledPin, LOW);
42 delayMicroseconds(280);
43
44 voltsMeasured = analogRead(dustPin);
45
46 delayMicroseconds(40);
47 digitalWrite(ledPin, HIGH);
48 delayMicroseconds(9680);
49
50 calcVoltage = voltsMeasured * (3 / 1024.0);
51 dustDensity = 0.17 * calcVoltage - 0.1;
52 Serial.println("GP2Y1010AU0F readings");
53 Serial.print("Raw Signal Value = ");
54 Serial.println(voltsMeasured);
55 Serial.print("Voltage = ");
56 Serial.println(calcVoltage);
57 Serial.print("Dust Density = ");
58 Serial.println(dustDensity);
59 Serial.println("");
60 delay(1000);
61
62
63
64 int x = ThingSpeak.writeField(myChannelNumber, 3, voltsMeasured, myWriteAPIKey);
65 if(x == 200){
66   Serial.println("Channel update successful.");
67 }
68 else{
69   Serial.println("Problem updating channel. HTTP error code " + String(x));
70 }
71
72 // change the value
73 number++;
74 if(number > 99){
75   number = 0;
76 }
77
78 delay(10000); // Wait 20 seconds to update the channel again
79 }
80

```

- **Arduino IDE code for collecting readings from MQ-7 gas sensor and connection to ThingSpeak cloud platform**

final | Arduino IDE 2.1.1

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E Mod... ▼)

SKETCHBOOK

final

sketch\_aug27a

final.ino

```

1  #include <ESP8266WiFi.h>
2  #include "ThingSpeak.h"
3  #include "MQ7.h"
4
5  MQ7 mq7(A0,5.0);
6
7  char ssid[] = "Phone";
8  char pass[] = "hereyougo2021";
9  WiFiClient client;
10
11  const variable Pin_D1 Key = "ZL1K7BIINL8Z2EY0";
12  float
13  float Type: int
14  float Value = 4
15  float
16  float int Pin_D1 = 4
17  int Pin_D1 = 4;
18  int Pin_D2 = 5;
19
20  void setup() {
21    Serial.begin(9600);
22    pinMode(Pin_D1, OUTPUT);
23    pinMode(Pin_D2, OUTPUT);
24    WiFi.mode(WIFI_STA);
25    ThingSpeak.begin(client);
26    if(WiFi.status() != WL_CONNECTED){
27      while(WiFi.status() != WL_CONNECTED){
28        WiFi.begin(ssid, pass);
29        Serial.print(".");
30        delay(5000);
31      }
32      Serial.print("Connected.")
33    }
34  }
  
```

final | Arduino IDE 2.1.1

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E Mod...)

SKETCHBOOK

final

sketch\_aug27a

final.ino

```

35
36 void dustSensor() {
37   digitalWrite(Pin_D2, LOW);
38   delay(1000);
39   digitalWrite(Pin_D1, HIGH);
40   delay(10000);
41 }
42
43 void coSensor() {
44   digitalWrite(Pin_D1, LOW);
45   delay(1000);
46   digitalWrite(Pin_D2, HIGH);
47   delay(10000);
48 }
49
50 void loop() {
51
52   dustSensor();
53   dust = analogRead(0);
54   calcVoltage = dust * (3 / 1024.0);
55   dustDensity = 0.17 * calcVoltage - 0.1;
56   Serial.print("Dust Density = ");
57   Serial.println(dustDensity);
58   Serial.println("");
59   int x = ThingSpeak.writeField(2253626, 3, dust, myWriteAPIKey);
60   if(x == 200){
61     Serial.println("Channel update successful.");
62   }
63   else{
64     Serial.println("Problem updating channel. HTTP error code " + String(x));
65   }
66

```

final | Arduino IDE 2.1.1

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E Mod...)

SKETCHBOOK

final

sketch\_aug27a

final.ino

```

66
67   coSensor();
68   co_read = analogRead(0);
69   co = mq7.getPPM();
70   Serial.print("CO = ");
71   Serial.println(co);
72   Serial.println("");
73   int x = ThingSpeak.writeField(2303832, 1, co, myWriteAPIKey);
74   if(x == 200){
75     Serial.println("Channel update successful.");
76   }
77   else{
78     Serial.println("Problem updating channel. HTTP error code " + String(x));
79   }
80 }
81
82
83
84
85

```

- ThingSpeak graph for GP2Y1010F gas sensor readings (PM2.5 concentration)

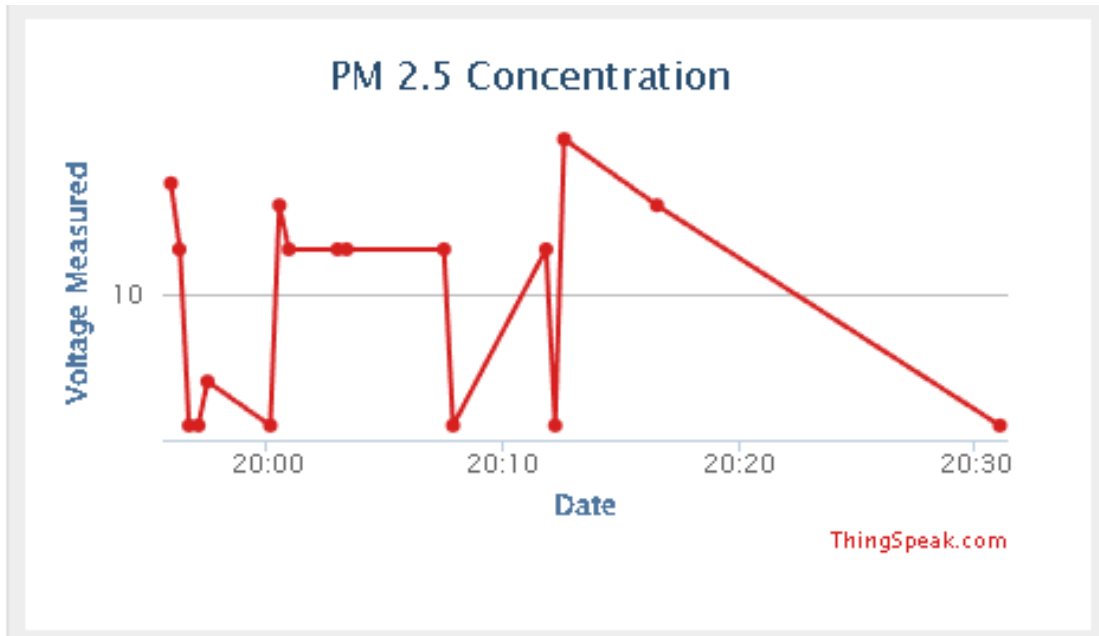


Fig 2.9

- ThingSpeak graph for MQ-7 gas sensor readings (CO concentration)

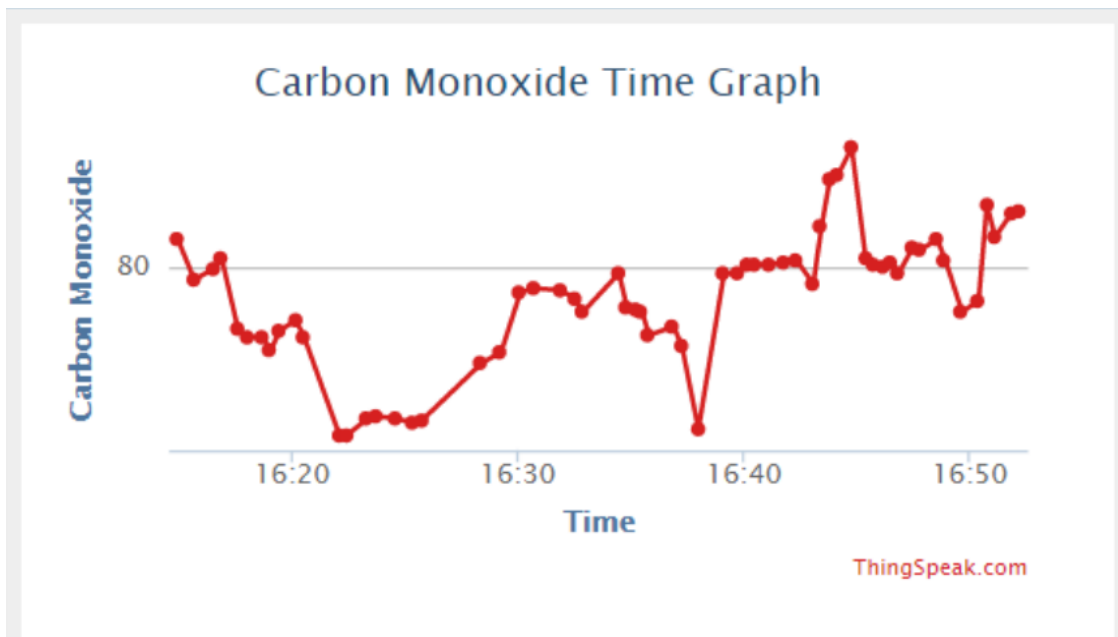


Fig 2.10

### **4.3. Further Exploration Plans and Timelines**

The dataset required for the project is a time series data with both AQI and lung cancer attributes. Since the data was not available publicly, now the plan is to synthesize the data which makes the project a research-based project.

### **Appendix A: Definitions, Acronyms and Abbreviations**

- ARIMA - AutoRegressive Integrated Moving Average.
- LSTM - Long Short-Term Memory
- IoT - Internet of Things
- ML - Machine Learning
- WHO - World Health Organization
- AQI - Air Quality Index
- PM - Particulate Matter
- PMS3003- Plantover Particulate Matter Sensor
- INAAQS - Indian National Ambient Air Quality Standards
- ANN- Artificial Neural Network
- LCD - Liquid Crystal Display
- CNN - Convolutional Neural Network
- LSTM - Long ShortTerm Memory
- IAQ - Indoor Air Quality
- SVM - Support Vector Machine
- IDE - Integrated Development Environment
- OS - Operating System
- AWS - Amazon Web Services
- ROI - Return On Investment

### **Appendix B: References**

[1] An Application of IoT and Machine Learning to Air Pollution Monitoring in Smart Cities

By: Muhammad Taha Jilani, Husna Gul A.Wahab

[<https://ieeexplore.ieee.org/document/8981707>]

[2] How Is the Lung Cancer Incidence Rate Associated with Environmental Risks?  
Machine-Learning-Based Modeling and Benchmarking

By: Kung-Min Wang, Kun-Huang Chen, Shieh-Hsen Tseng

[<https://www.mdpi.com/1660-4601/19/14/8445>]

[3] Assessment of indoor air quality in academic buildings using IOT and deep learnings

By: Mohammad Marzouk and Mohammad Atef

[<https://www.mdpi.com/1667822>]



[4] Household Ventilation May Reduce Effects of Indoor Air Pollutants for Prevention of Lung Cancer: A Case-Control Study in a Chinese Population.

By: Jin Z-Y, Wu M, Han R-Q, Zhang X-F, Wang X-S, et al.

[<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0102685>]

[5] Determination of Air Quality Life Index (AQLI) in Medinipur City of West Bengal(India) During 2019 To 2020 : A contextual Study

By: Samiran Rana.

[[https://www.researchgate.net/publication/360622768\\_Determination\\_of\\_Air\\_Quality\\_Life\\_Index\\_Aqli\\_in\\_Medinipur\\_City\\_of\\_West\\_BengalIndia\\_During\\_2019\\_To\\_2020\\_A\\_contextual\\_Study](https://www.researchgate.net/publication/360622768_Determination_of_Air_Quality_Life_Index_Aqli_in_Medinipur_City_of_West_BengalIndia_During_2019_To_2020_A_contextual_Study)]

[6] Air pollution and skin diseases: Adverse effects of airborne particulate matter on various skin diseases,

By: Kim Kyung Eun, Cho Daeho, Park Hyun Jeong

[<https://pubmed.ncbi.nlm.nih.gov/27018067/>]

[7] The spatial association between environmental pollution and long-term cancer mortality in Italy.

By: Roberto Cazzolla Gatti, Arianna Di Paola, Alfonso Monaco, Alena Velichevskaya, Nicola Amoroso, Roberto

[<https://www.sciencedirect.com/science/article/pii/S0048969722055383#:~:text=We%20studied%20the%20links%20between%20cancer%20mortality%20and%20environmental%20pollution%20in%20Italy.&text=Tumor%20mortality%20exceeds%20the%20national%20average%20when%20environmental%20pollution%20is%20higher.&text=Air%20quality%20ranks%20first%20for,to%20the%20average%20cancer%20mortality.>]

[8] The nexus between COVID-19 deaths, air pollution and economic growth in New York state: Evidence from Deep Machine Learning

By: Cosimo Magazzino , Marco Mele , Samuel Asumadu Sarkodie

[<https://www.sciencedirect.com/science/article/pii/S0301479721003030>]

## Appendix C: Record of Change History

#	Date	Document	Change Description	Reason for Change
---	------	----------	--------------------	-------------------

		Version No.		
1.	28-09-2023	1.0	First creation and filling up of minor details	First version
2.	12-10-2023	1.1	Writing about scope, methodology and the architecture of the project	Extended version
3.	13-10-2023	1.2	Implementation details and formatting	Finishing up the document

**Table-1.1**

## Appendix D: Traceability Matrix

Project Requirement Specification Reference Section No. and Name.	DESIGN / HLD Reference Section No. and Name.	LLD Reference Section No. Name
Section 2: Literature Survey	Appendix B	Appendix B
Section 3: Product Perspective	Section 3: Diagrams	Section 4.1
Section 6: Non - Functional requirements and Section 7: Other Requirements	Section 4: Design Details	Fig 1.0 - 1.9, 2.1, 2.9,2.10

**Table-1.3**