Real-Time Air Quality Monitoring and its predictions

Rutwik Mekala(Btech), Akhil Reddy Nagulapally(BTech), Arigela venkata sesha sai Rohith(BTech)

Abstract— Automation is taking over the industry. Everything we consume is somewhere or another linked to automation. Either it is food product or it is industrial products. The weather department, provides us with weather details about a particular city, it fails to provide information about particular localities. By this project, we aim to automate healthcare. In this project, we monitor the air quality for harmful gases at a particular location, a user can check on our website if a location is safe to visit or not. The main aim of this project is to develop a tool which is able to monitor PPM and Harmful Gases in air in real time, tell the quality of air and log data to our cloud server. The air display developed during this project is predicted on NodeMCU. The board connects uploads data on thinger.io Database over the wifi and the information is displayed in a user-friendly manner on our website.

***Index Terms*— Airpollution,IOT,Sensors,Predictions**

I. Introduction

The core objective of this project is to make a hardware device and an intuitive website such that people can have access to the information as to how bad the condition of Air is in their locality. Real-Time values are to be displayed on the website and we also explore prediction algorithms in machine and deep learning to know if the area can be termed safe or not based on the future Pollution values.

We will be using IOT to send the data gathered from the sensors to the cloud server and the users can check out additional data on the website such as temperature and humidity.

Air pollution is one of the biggest dangers for the environment and impacts everyone: humans, animals, crops, cities, forests, aquatic ecosystems...

Every year, we face a dangerous problem of excessive air pollution. A lot of cities are covered in smog and the air quality reaches dangerous levels. The pollution level gets so dangerous in few cities like Tokyo and delhi that most of the major cities are left gasping for breath. In fact, India is home to the world's 14 most polluted cities, based on the presence of particulate matter. The WHO also estimates that air pollution kills approximately 7 million people in a year. Pollution is the major reason for many fatal problems such as heart problem, lung diseases, respiratory infections, pneumonia, cancer, and insomnia. Continual exposure to air pollutants is responsible for the worsening of human health.

There are many different types of air pollutants, such as gases like ammonia, carbon monoxide, sulfur dioxide, nitrous oxides, methane and chlorofluorocarbons, and many other forms of air pollutants. Air pollution might most likely cause diseases, allergies and even death to humans; it may also cause damage to other living organisms such as animals and food crops. Air pollution kills 1.25 million people in India every year. So, to fight against the air pollution and protect our people we thought it would be better to let the people know the quality of the air in a particular area that they want to visit. So that they could decide whether it is safe or not to go there based on the information that we provide them. And this was our sole motivation to do this project.

A. Abbreviations and Acronyms

ppm-parts per million, EPA-Environmental Protection Agency, AQI-Air Quality Index, Mux-Multiplexer

II. PROJECT DESCRIPTION AND GOALS

In this project, we will monitor the quality of air and this is IOT based project.

The AQI (air quality index) tells you about the quality of your air and the associated health effects on your body.

With the increase in population, industrialization and automobile companies the pollution also increased in the last few years, and this leads to deterioration of health of leaving beings which are exposed to it. So, to avoid the unnecessary illness we need to monitor the quality of air. In this project, we are going to make an IoT Based Air Quality Index Monitoring System in which we will monitor the Air Quality Index over a Thinger.io server using the internet. We will use MQ135 Air Quality Sensor that can detect the level of various air pollutant.

The AQI value tells you about the health affects that you may experience within a few hours or days after breathing polluted air.

EPA calculates the AQI for five major air pollutants regulated by the Clean Air Act: ground-level ozone, particle pollution (also known as particulate matter), carbon monoxide, sulfur dioxide, and nitrogen dioxide. For each of these pollutants, EPA has established national air quality standards to protect public health. Ground-level ozone and airborne particles are the two pollutants that pose the greatest threat to human health in this country.

0 ppm – 600 ppm: Good for health

600 ppm – 1200 ppm: Moderate

1200 ppm (and above): Harmful to health

Think of the AQI as a yardstick that runs from 0 to 1200. The higher the AQI value, the greater the level of air pollution and the greater the health concern. For example, an AQI value of 100 represents good air quality with little potential to affect public health, while an AQI value over 1200 represents hazardous air quality.

GOALS:

The primary goal of our project is to make sure that people are aware of dangerous levels of pollution in many places to which they tend to visit very often so that they can be careful and avoid getting infected by any diseases.

In this project we have built a website which gives you the information about the AQI as well as few additional features like temperature and humidity that the public visit so that they can check before deciding whether to visit that place or not.

We have created our website very user friendly so that anyone can use it without much trouble.

We display the information in a very easily understandable way like by using geometric shapes, pie charts, statistics, graphs..etc..

III. TECHNICAL SPECIFICATION

Hardware Description

NodeMCU

It collects data from both the gas sensors and temperature, humidity sensor and sends the data to the cloud server over WiFi

MQ135

The MQ135 sensor can sense NH3, NOx, alcohol, Benzene, smoke, CO2 and some other gases. These are normally the main pollutants contained in air. Measured in ppm

MQ7

MQ7 sensor is used to detect Carbonmonooxide. Measured in ppm

Dht11

Dht11 sensor is used to derive humidity, temperature values, Measured in Celsius and %

4051 Mux

Multiplexer used to provide multiple analog pins

BreadBoard

This acts as the medium for the components to be connected to each other.

Jumping wires

We use it to connect one device to another on the breadboard

SOFTWARE DESCRIPTION:

ARDUINO IDE

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino

Thinger.io

IOT platform where we can send and display the data to through WiFi

Visual Studio Code

Source code editor used to write and execute code.
Software used to code Website . We used html and css to code for the website

Jupyter Notebook(anaconda)

Use to write code and execute the code for machine and deep learning

IV DESIGN APPROACH AND DETAILS

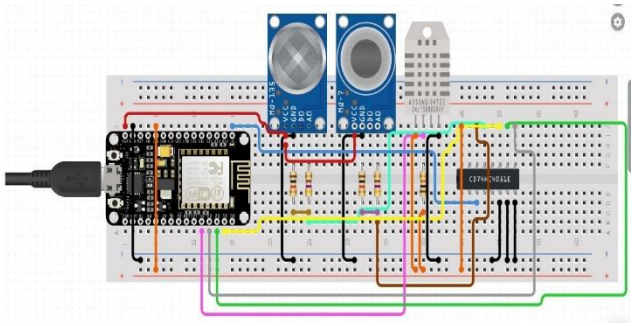


Fig4.1 Design approach

To design the hardware device, we used two sensors to detect pollution present i.e MQ135 and MQ7 sensor. To know the temperature and humidity, we use DHT11 sensor.

Here DHT11 is a digital sensor whereas MQ135 and MQ7 are analog sensors. We use resistors to regulate the reading of the sensors, as it initially comes with 1kΩ inbuilt resistor which doesn't necessarily give very accurate values. This can also be regulated by using suitable libraries in Arduino IDE

We connect Vcc and GND pins of sensors to that of NodeMCU using the breadboard.
Supply power to NodeMCU to startup all the sensors.

Codes and standards

Arduino IDE code-

```
#include <Mux.h>

#include <MQUnifiedsensor.h>
//#include "MQ135.h"
#include <ThingerESP8266.h>
#include <ESP8266WiFi.h>
#include "DHT.h"
//----------------------------------------

//----------------------------------------Thinger IO configuration
#define USERNAME "Rutwik"
#define DEVICE_ID "dht11"
#define DEVICE_CREDENTIAL "9MKOVeqvsP+k1zwNl"
//----------------------------------------
#define MUX_A D1
#define MUX_B D2
#define MUX_C D3

#define ANALOG_INPUT A0
//----------------------------------------

#define ON_Board_LED 2  //--> Defining an On Board LED, used for indicators when the process of connecting to a wifi router.

ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL); //--> Initialize Thinger IO (ThingerESP8266)

//----------------------------------------SSID and Password of your WiFi Router/Hotspot.
const char* ssid = "mekalas"; //--> Your wifi name or SSID.
const char* password = "mekalas123"; //--> Your wifi password.
//----------------------------------------

//----------------------------------------DHT11 Sensor Configuration
#define DHTPIN D4 //--> Digital pin connected to the DHT sensor.
#define DHTTYPE DHT11 //--> DHT11

DHT dht11(DHTPIN, DHTTYPE); //--> DHT11 Sensor Initialization
//----------------------------------------

float temperature,humidity,mq7,mq135; //--> Variables for temperature, humidity and pollution sensors data
void setup() {
  //Deifne output pins for Mux
 pinMode(MUX_A, OUTPUT);

 pinMode(MUX_B, OUTPUT);
 pinMode(MUX_C, OUTPUT);
 // put your setup code here, to run once:
```

```
  pinMode(ON_Board_LED,OUTPUT); //--> On Board LED port Direction output
  digitalWrite(ON_Board_LED, HIGH); //--> Turn off Led On Board


  WiFi.begin(ssid, password); //--> Connect to your WiFi router

//----------------------------------------Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    //----------------------------------------Make the On Board Flashing LED on the process of connecting to the wifi router.
    digitalWrite(ON_Board_LED, LOW);
    delay(250);
    digitalWrite(ON_Board_LED, HIGH);
    delay(250);
    //----------------------------------------
  }
  //----------------------------------------

  digitalWrite(ON_Board_LED, HIGH); //--> Turn off the On Board LED when it is connected to the wifi router.

  thing.add_wifi(ssid, password); //--> Initialize wifi

  dht11.begin(); //--> Starts reading DHT11 Sensor


  //----------------------------------------Sends DHT11 Sensor data (Temperature and Humidity) and MQ135 sensor data to Thinger IO

// Symbol or operator ">>" means to transmit data
  thing["dht11"] >> [](pson& out){
      out["temperature"] = temperature;
      out["humidity"] = humidity;
      out["mq7"]= mq7;
      out["mq135"]= mq135;

  };

}
void changeMux(int c, int b, int a) {
  digitalWrite(MUX_A, a);
  digitalWrite(MUX_B, b);
  digitalWrite(MUX_C, c);
}

void loop() {

  // call always the thing handle in the loop and avoid any delay here
  thing.handle();
  //----------------------------------------To get temperature and humidity data from the DHT11 sensor
  temperature = dht11.readTemperature();
  humidity = dht11.readHumidity();

  changeMux(LOW, LOW, HIGH);
  mq7 = analogRead(ANALOG_INPUT); //Value of X2 sensor

  changeMux(LOW, HIGH, LOW);
  mq135 = analogRead(ANALOG_INPUT); //Value of X1 sensor
}
```

We recorded the data from the sensors gathering over 10000 readings for temperature, humidity, pollution from MQ135 and MQ7. If we are able to read and analyze the data gathered by using techniques like Linear Regression, Random Forrest, XGBoost and ARIMA, we will be able to predict if a location is safe or not given the future readings

Deep Learning code

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read_csv('Downloads/capstone_data.csv')

X = dataset.iloc[:, 1:5].values

y = dataset.iloc[:, 5].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

ty = X_test

X_test = sc.transform(X_test)

import keras

from keras.models import Sequential
```

```
from keras.layers import Dense
classifier = Sequential()
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu', input_dim = 4))
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
classifier.fit(X_train, y_train, batch_size = 40, epochs = 30)
```

```
 y_pred = classifier.predict(X_test)
y_pred = ((y_pred > 0.5))
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
data = [102,599,383,41]
arr=[]
arr.append(data)
df = pd.DataFrame(arr, columns = ['HUMID', 'MQ135','MQ7','temp'])
df1 = df.iloc[:,:].values
df1 = sc.transform(df1)
y_predcheck = classifier.predict(df1)
y_predcheck = ((y_predcheck > 0.5))
print(y_predcheck)
import csv
# field names
fields = ['Result']
# data rows of csv file
rows = y_predcheck
# name of csv file
filename = "Downloads/result.csv"
with open(filename, 'w') as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerow(fields)
    csvwriter.writerows(rows)
import matplotlib.pyplot as plt
z=[i for i in range(1,10001)]
print(len(z))
plt.bar(z,X[:,0])
    plt.xlabel('observations')
    plt.ylabel('readings')
    plt.savefig("figure.png")
```

Machine Learning code

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
dataset = pd.read_csv('Downloads/capstone_data.csv')
X = dataset.iloc[:, 1:5].values
y = dataset.iloc[:, 5].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
print(X)
print(dataset.columns)
y = list(dataset.get('safe'))
print(y)
dataset.values
x = dataset.values[:,1:5]
print(x)
y = dataset.values[:,5]
print(y)
print(x.shape)
print(y.shape)
from sklearn.linear_model import LinearRegression
# 1. Creating an object
model = LinearRegression(normalize=True)
# 2. Training the model
model.fit(X_train,y_train)
# 3. Printing the coefficients
print(model.coef_)
print(model.intercept_)
y_pred = model.predict(X_test)
#y_pred = ((y_pred > 0.7))
print((y_pred))
## Accuracy of prediction on the training data
#model.score(y_test,y_pred)
```

```
print(model.score(X_test,y_pred))
model.score(X,y)
y_pred = model.predict(X_test)
y_pred = ((y_pred > 0.7))
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
data = [50,50,50,30]
arr=[]
arr.append(data)
df = pd.DataFrame(arr)
xchek = df.iloc[:, 0:5].values
y_predcheck = model.predict(xchek)
y_predcheck = ((y_predcheck > 0.7))
y_predcheck
from sklearn.linear_model import LogisticRegression
# 1. Creating an object
clf =
LogisticRegression(random_state=0).fit(X_train,
y_train)
# 2. Training the model
model.fit(X_train,y_train)
# 3. Printing the coefficients
print(model.coef_)
print(model.intercept_)
clf.predict(X_test)
clf.score(X_test, y_test)
```

Website Code

https://github.com/Rutwik-Mekala/capstone

Constraints, alternatives and tradeoffs/Problems faced and solutions found

Reason to choose NodeMCU

An alternative way to do this project is by using Arduino Uno. But an issue with using UNO is we need to buy a WiFi module separately which will considerably increase the price of equipment. That is why we used NodeMCU as it has an inbuilt WiFi module and does the same functions of Arduino Uno. But one major drawback of using NodeMCU is it only contains one analog port while Arduino Uno contains 6 analog ports. And we happened to have 2 analog sensors. One way we can do is we can use Arduino uno as an extension for analog pins. But again that's not really ideal for budget friendly device.

So the idea we applied which meets are budget friendly approach is we used a 4051 multiplexer.

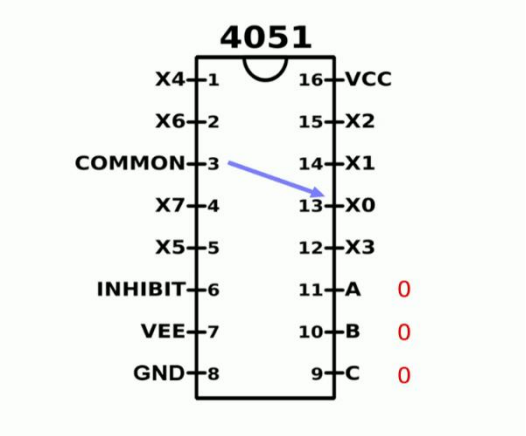


Fig 4.2 MUX 4051

Reason to choose 4051 mux

Here we connect the A0 pin from NodeMCU to the common pin in mux. We give power at pin 16 and ground pins 6,7,8. A, B, C pins here are connected to D1, D2, D3 pins in NodeMCU. Now we connect MQ7 analog pin to X1 port and MQ135 pin to X2 port. Now when we give (LOW, LOW, HIGH) to the digital pins, we get MQ7 sensor values. Similarly when we give (LOW, HIGH, LOW), we get values for MQ135 sensor.

Reason to choose thinger.io

As for using IOT platflorm, there are multiple choices to choose from. We initially worked on thingspeak.com. It had very basic functionality but the major drawback is its data-receive rate in 15 sec which might be too long. So we moved to thinger.io which has an additional feature called data buckets which can help up store information gathered effectively. And it can receive data in 2 sec time which is very efficient

Challenges with Accuracy

Another issue we faced was NodeMCU was unable to sit well with the breadboard. As a solution we used male-to-female jumper wires and connected it to the breadboard. This gave rise to another problem. We were unable to use resistors to regulate voltage levels so the sensors showed unregulated values. So we decided this can only be fixed through the software route. So we did extensive library search and found and modified suitable libraries which now shows accurate results.

Means to display results

We were initially planning to make an app to display results but few of the teammates were unable to use it as they had an iphone and app was made using android studio. So we switched over and decided to stick with website as it can be accessible from all platforms.

V PROJECT DEMONSTRATION

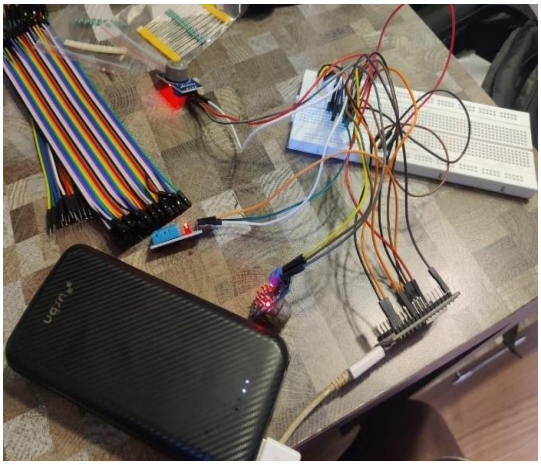
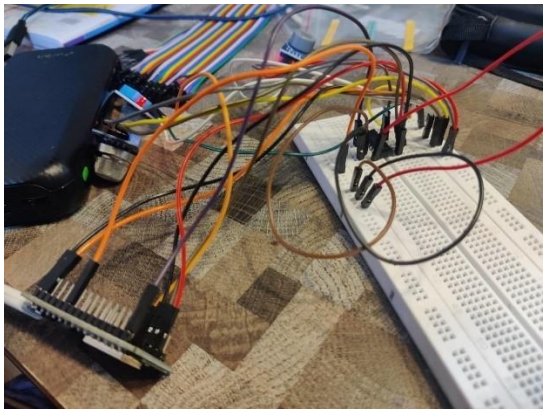


Fig 5.1 Hardware setup (a)



Hardware setup (b)

We complete the hardware setup as shown in 4.1 Design approach

Arduino IDE

We then uploaded the Arduino code that we worked on into the NodeMCU board. We then were able to supply power to the board using power bank,

Then we added the code which can send the data acquired remotely, through WiFi into the cloud server i.e

Thinger.io

The data is henceforth received in the IOT platform. This data is collected by using data buckets and is stored in an excel file. Over 10000+ readings are taken in the span of 2 weeks

With the data gathered, one important feature we wanted to explore is analyzation of data and checking if we can predict the future result if an area is safe or not, based on Real-Time data supplied. For this we worked on Machine Learning and Deep Learning algorithms to achieve it. We have found that deep learning ended up giving more accurate and elaborate results compared to machine learning.

Now we had the job to display all the data gathered and provide the end result of this project using a Website.

This is what the end user will be seeing when they want to see the result of this project.

WEBSITE

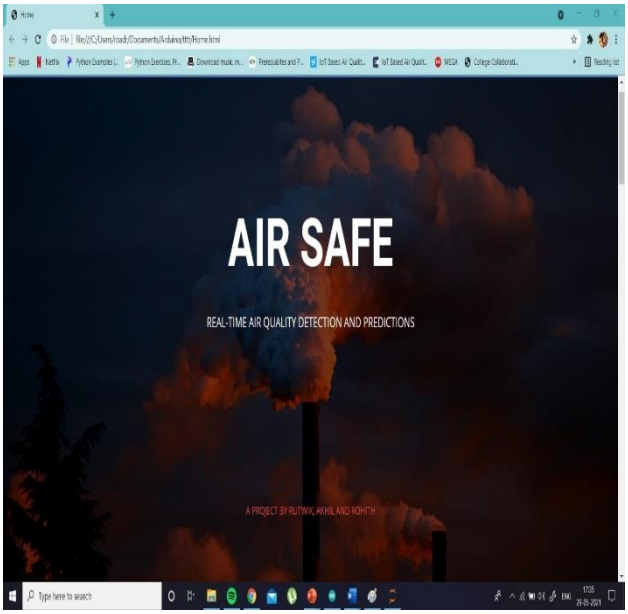


Fig 5.2 home page

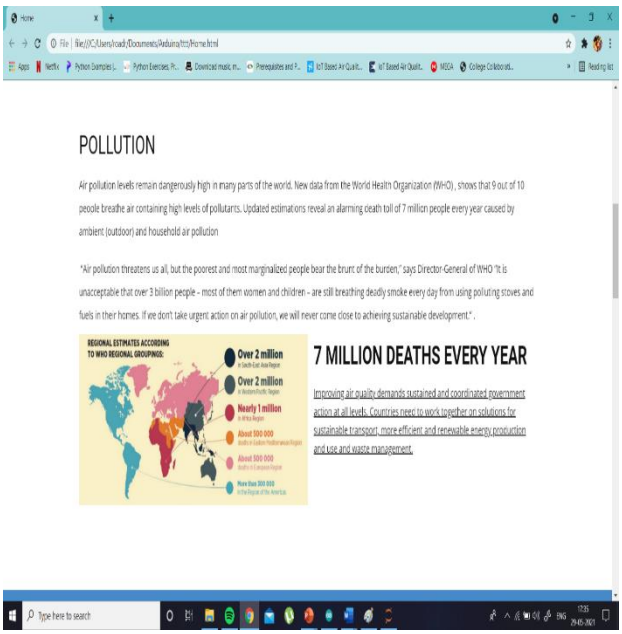


Fig 5.3 pollution

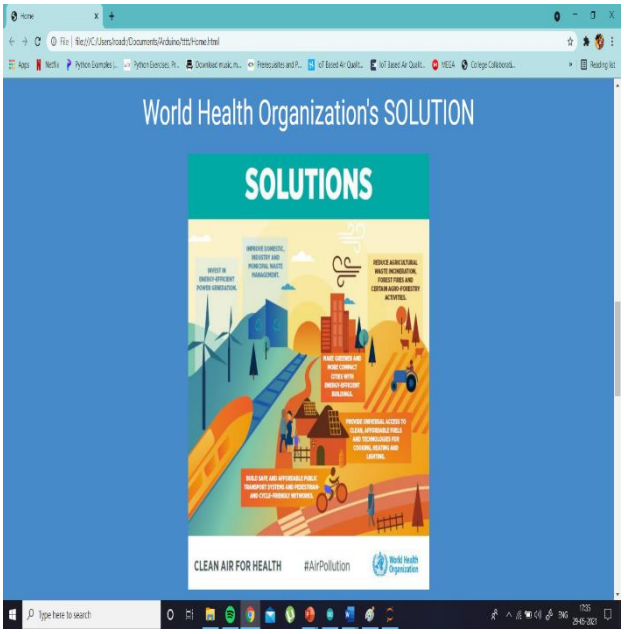


Fig 5.4 solutions

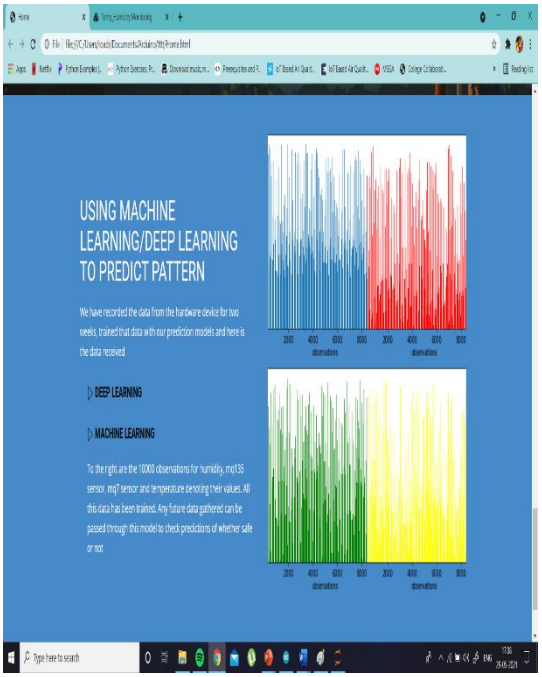


Fig5.6 predictions

Here is the data stored in a csv format. 10000 readings have been recorded

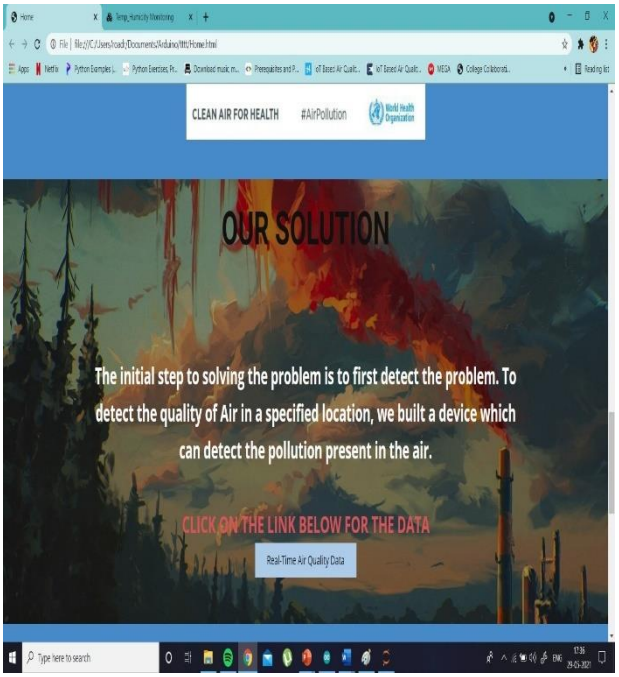


Fig 5.5 Real-time air quality

This page when clicked on the Real-Time Air Quality Data link, reroutes you to the data gathered.

(IMAGE SHOWN IN FIG 6.5)

This next page is the final page displaying all the data entered through the prediction algorithms with clickable links. When clicked on Deep Learning and Machine Learning links, it reroutes you to the code which can then be executed with manual data to show prediction of whether the place is safe or not.

	A	B	C	D	E	F
1		HUMID	MQ135	MQ7	temp	safe
2	0	70	103	56	10	1
3	1	142	256	632	64	0
4	2	69	158	347	28	1
5	3	99	672	313	84	0
6	4	79	49	49	22	1
7	5	131	487	610	67	0
8	6	77	66	256	12	1
9	7	138	371	609	100	0
10	8	49	210	392	32	1
11	9	98	523	660	29	0
12	10	71	182	257	24	1
13	11	80	329	634	27	0
14	12	51	123	195	24	1
15	13	96	506	473	63	0
16	14	66	191	67	9	1
17	15	91	284	593	66	0
18	16	46	215	317	26	1
19	17	132	442	403	20	0
20	18	58	35	379	32	1
21	19	157	368	691	81	0
22	20	75	41	89	7	1
23	21	115	551	501	61	0
24	22	53	17	115	33	1
25	23	91	427	303	84	0
26	24	48	107	392	2	1
27	25	140	374	616	39	0
28	26	80	49	107	39	1
29	27	91	610	598	77	0

capstone_data

Fig 5.7 data stored

To understand how we are determining if an area is safe or not, here is a flowchart

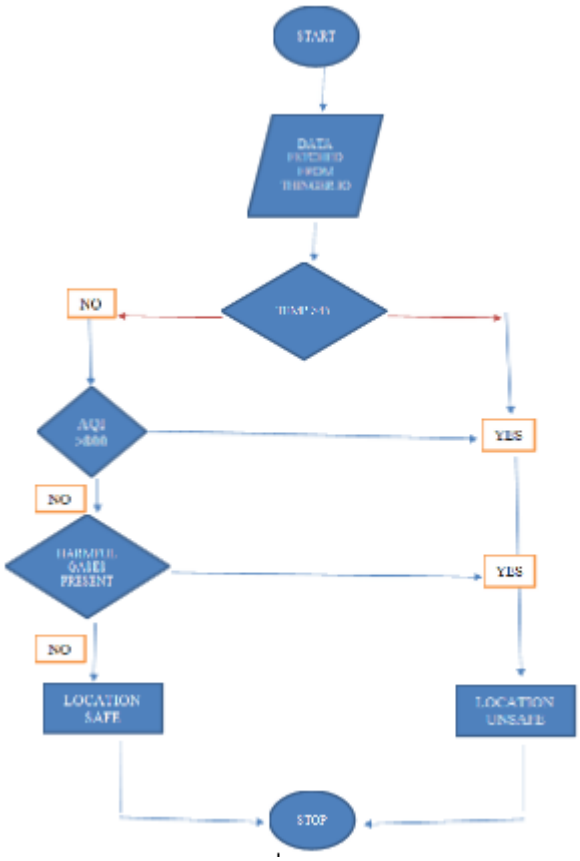


Fig 5.8 flowchart

VI. RESULT

Our ultimate goal with this project is to provide an affordable setup with which a user can easily access details about the Air Quality in the location they live in. The model we used is to demonstrate the condition in one location. Similar models can be used through out the city to find Air Quality status in Real-Time.

We have extensively looked into ways how we can reduce the cost of this setup.

From other papers we referred, they used

Arduino uno – 620 rs + ESP8266 WiFi module- 250rs + jumper wires – 10rs+3 sensors- 500rs = 1380rs

To reduce cost we used NodeMCU which don't need an external WiFi module and is considerably cheaper than an Arduino uno. Extra 4051 mux used here for multiple analog sensors

NodeMCU – 350 + jumper wires-10rs+4051 mux-20rs + 3 sensors- 500rs = 880rs

So we are able to considerably cut cost with this setup.

RESULT AND DISCUSSION

With our 3 sensors, we were able to gather the following data in 10000 recorded readings

HUMIDITY

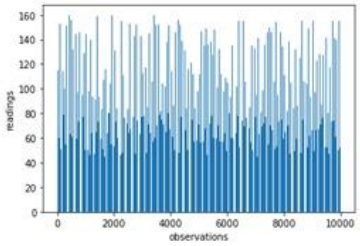


Fig 6.1 humidity

TEMPERATURE

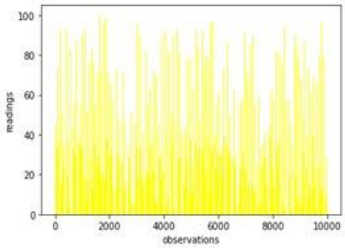


Fig 6.2 temperature

MQ135

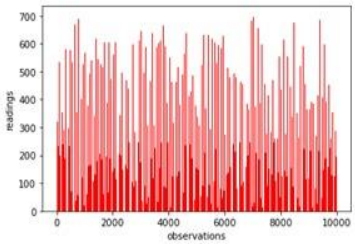


Fig 6.3 MQ135 O/P

MQ7

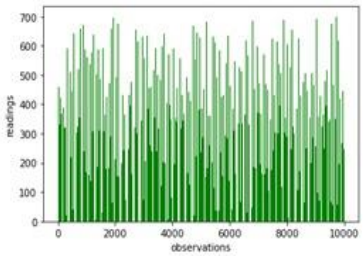


Fig 6.4 MQ7 O/P

Data about the Gases, Temperature and Humidity is received every two seconds in the database. Accuracy of predicting whether environment is safe or not by machine learning model is 99.97% and by deep learning model is 100%.

And as for the Real-Time data we gathered, it is displayed in the following way

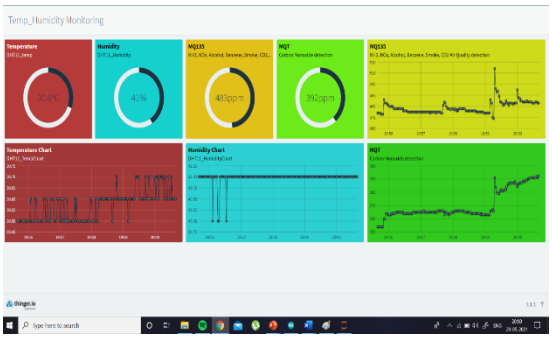


Fig 6.5 real time data gathered

VIII. CONCLUSION

We have accomplished our goal of displaying Real-Time Data of Air Pollution, Temperature and Humidity in a user-friendly Website with additional features such as predictions of whether area is safe or not. We tried to cut costs in all ways possible. So if this project is done using multiple devices in a large scale, we can essentially find the data of Air Quality is every location. Our project is a model of the same purpose but on a smaller scale.

Society looks for a pollution-free globe for happy living. The global warming threat is waiting at the door. Government rules, governing pollution control in private sector industries are not implemented that effectively. This scenario stresses the need for an efficient monitoring system with the collaboration of users, domain experts, hardware designers and software developers. This project is an attempt in this direction

REFERENCES

[1]. Salman, Afan Galih, Bayu Kanigoro, and Yaya Heryadi. "Weather forecasting using deep learning techniques." 2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS). IEEE, 2015.

[2]. Thu, M. Y., Htun, W., Aung, Y. L., Shwe, P. E. E., & Tun, N. M. (2018, November). Smart Air Quality Monitoring System with LoRaWAN. In 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS) (pp. 10-15). IEEE.

[3]. Kang, Ganganjot Kaur, et al. "Air quality prediction: Big data and machine learning approaches." International Journal of Environmental Science and Development 9.1 (2018): 8-16

[4]. Meshram, P., Shukla, N., Mendhekar, S., Gadge, R., & Kanaskar, S. (2019). IoT Based LPG Gas Leakage Detector. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 5(1).

[5]. Elvitigala, Chanuka & Sudantha, Bh. (2017). Machine Learning Capable, IoT Air Pollution Monitoring System with Upgradable Sensor Array.

[6]. IoT based Air Quality Monitoring, F N Setiawan and I Kustiawan 2018 IOP Conf. Ser.: Mater. Sci. Eng. 384 01200

[7]. Kök, İbrahim, Mehmet Ulvi Şimşek, and Suat Özdemir. "A deep learning model for air quality prediction in smart cities." 2017 IEEE International Conference on Big Data (Big Data). IEEE

[8]. Kodali, R. K., & Sarjerao, B. S. (2017, December). MQTT based air quality monitoring. In 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC) (pp. 742-745). IEEE.

[9]. Sugiarto, Bambang, and Rika Sustika. "Data classification for air quality on wireless sensor network monitoring system using decision tree algorithm." 2016 2nd International Conference on Science and Technology-Computer (ICST). IEEE, 2016

[10]. Manna, S., Bhunia, S. S., & Mukherjee, N. (2014, May). Vehicular pollution monitoring using IoT. In International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014) (pp. 1-5).

[11]. Kathiravan Srinivasan, Anant Nema, Chao-Hsi Huang,Tung Yang Ho,"Weather Forecasting Application using Web-based Model-ViewWhatever Framework",2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW