Real-Time Air Quality Monitoring and its predictions

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Electronics and communication engineering

By

RUTWIK MEKALA 17BEC0039

AKHIL REDDY 17BEC0648

ROHITH 17BEC0483

Under the guidance of Prof. Sangeetha.S

SENSE VIT,

Vellore.

MAY, 2021

DECLARATION

I hereby declare that the thesis entitled "**Real-Time Air Quality Monitoring and its predictions** " submitted by Rutwik Mekala, Akhil Reddy N and Rohith for the award of the degree of Bachelor of Technology in Electronics and communication engineering to VIT is a record Bonafide work carried out by me under the supervision of Prof. Sangeetha .S.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place : Vellore

Date :

RUTWIK MEKALA 17BEC0039

AKHIL REDDY 17BEC0648

ROHITH 17BEC0483

CERTIFICATE

This is to certify that the thesis entitled "**Real-Time Air Quality Monitoring and its prediction**" submitted by Rutwik Mekala, Akhil Reddy and Rohith with registration numbers 17BEC039, 17BEC0648 and 17BEC0483, of SENSE, VIT, for the award of the degree of *Bachelor of Technology in Electronics and communication Engineering*, is a record of Bonafide work carried out by Akhil, Rohith and Rutwik under my supervision during the period, 01. 02. 2021 to 25.5.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute orany other institute or university. The thesis fulfills the requirements and regulations ofthe University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :

Signature of the Guide

SANGEETHA .S

Internal Examiner **External Examiner**

HEAD OF DEPARTMENT

SENSE

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude and heartfelt thanks to our project guide – Prof. Sangeetha.S for her valuable guidance, consistent encouragement, timely help has been a constant pillar of motivation throughout the planning and implementation phases of this project. Her support and help, over the entire duration of the project, was of utmost importance in making it a success.

RUTWIK MEKALA 17BEC0039

AKHIL REDDY 17BEC0648

ROHITH 17BEC0483

Executive Summary

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.

CONTENTS

Page No.

List of Figures

List of Tables

List of Abbreviations

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

Symbols and Notations

Ω	ohm

1. INTRODUCTION

1.1. OBJECTIVE

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.

1.2. Motivation

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.

1.3.BACKGROUND

To understand how we are approaching our project, we look into some of the papers that we took inspiration from.

Gas leakage has been one of the major problems in the industrial sector and also in the residential locations. There are the preventive methods that could be taken to avoid the gas leak incidents this could be done by installing a gas leak detection kit at vulnerable locations. Basically the goal of the paper is to gas leak detection, sending alerts and controlling gas leaks. The gas sensor that used has higher sensitivity to gases such as propane and butane together with LPG. The alarm has been installed along with it that gets triggered when the LPG is detected. There has been continuous degradation of air quality in the cities due to increase in urbanisation and industrialization. Due to little use of catalytic converters, there is production of great amount of toxic gases. Basically, the objective of the paper is to monitor the air pollution on roads and track vehicles that cause pollution beyond specific limit. The combination of wireless sensor network, electrochemical toxic gas sensors and the use of radio frequency identification tagging system has been made to monitor pollution.

The benefits of IOT can be extended for the enhancement of safety standards. Gas leakage in open or closed area can prove to be dangerous. Traditional gas leakage detector though was precise, fail to acknowledge few factors in alerting the people about the leakage. Therefore, in this paper it has been discussed about the technology that can be helpful in gas detection and smart alerting techniques.

It has been necessary to monitor the air quality for healthy living. Sound pollution monitoring system and air quality monitoring system has been proposed in this paper. It uses air sensors to sense harmful gases like NH3, benzene, smoke and Co2. Also, the system keeps monitoring the sound level and reports it to server over IOT. This work makes an attempt to observe the quality of air and the sound of a particular location and send this information to responsible people who will use this information to upgrade the standard of living of local people of that location, which is one of the first ideas about Smart City.

In the paper an MQTT based air pollution monitoring and forecasting system is discussed. It includes MQ series gas sensors for collecting concentration of gases such as CO2, NO, smoke, etc. in the atmosphere which is interfaced with NodeMCU. The NodeMCU is equipped with an ESP8266 WLAN adapter to send the readings of the sensors to a ThingSpeak cloud platform for the analysis of the data. It also includes predictive analysis using a suitable machine learning model to predict the air pollution level using a US government dataset. Machine learning models were implemented to find out the best predictive model for calculating the Air Quality Index (AQI) of four different gases namely Carbon Monoxide (CO), Nitrogen Dioxide(NO2), Sulfur Dioxide(SO2) and Ozone(O3). The machine learning algorithms that were used were Linear Regression, Random Forrest, XGBoost and ARIMA for time series forecasting. The performance metrics that were used were Root Mean Squared Error and Mean Absolute error. NodeMCU was used, which is an open-source development board with ESP8266-12E chips. MQ-2 gas sensor was used to collect gas concentration readings.

The data captured was sent to ThingSpeak cloud for IoT based data acquisition. Hourly pollution forecasting of three different places in Kolkata, India, was performed . The outcomes of the paper were – A time series future forecasting of gases like NO2, CO, O3 etc., model-fit accuracy plots for XGBoost for all the gases and hourly pollution forecasting of the three different places in Kolkata, India – Chandni Chowk, Gariaghat, Dharamtala.

Yet another study in paper proposed a system which acquires inputs from various sensors which detect air quality. These acquired inputs are provided to the control unit which processes these signals. The algorithm predicts the air quality and this information is the sent to the traffic station. Based on this, traffic can accordingly be regulated to control the air quality of that particular area. The related information can be viewed using a cellphone application which would allow users to follow less polluted route thereby indirectly bringing down pollution in highly polluted areas.

The system was developed using the PIC16F877A. The sensors were MQ- 7, MQ-135, MQ-4, G37. All the inputs from the sensors are acquired and they undergo signal conditioning. The output from these signal conditioning circuits is then given as input to the ADC of 10-bit resolution. The inputs are given to an air quality monitoring algorithm which gets input values and compares them with threshold values. The air quality information is then transmitted via a Wi-Fi transceiver-ESP8266 to a remote server from where this information can be accessed by the traffic control station .

The proposed system measures the air quality of a particular area with the help of the hardware module fixed at certain locations like lamp posts. The proposed system collected real time pollution statistics using various sensors which monitored percentage of gases like ammonia, oxygen and carbon monoxide. Using these inputs, the algorithm predicted the air quality. The outcomes were: reduction in air pollution level of certain highly polluted routes, decrease in traffic, increase of awareness.

In another research work, a real-time air pollution observing framework is built IoT dependent on countless sensors. The acquired information was analyzed through neural networks. The framework accomplished a superior checking accuracy because of the utilization of an enormous number of sensors. The utilization of IoT gadgets empowered the air contamination observing framework to be savvy and adaptable. The model was designed using an Arduino Uno microcontroller, Wi-Fi module 8266, MQ135 Gas Sensor and a 16 by 2 LCD Screen.

The framework was characterized into Five (5) layers. The principal layer was the natural parameters which are acquired by estimation. The subsequent layer was the study of the features and highlights of the sensors. The third layer was the decision making, detecting, estimating, fixing of the threshold value, periodicity of sensitivity, timing and space. The fourth layer was the sensor information acquisition. The fifth layer was the surrounding intelligence environment. The sensor gathered information when worked by the microcontroller and sent it over the web for examination by means of the Wi-Fi module. Clients had the option to screen estimated parameters on their cell phones.

This research proposed a smart air pollution monitoring system that constantly keeps track of air quality in an area and displays the air quality measured on an LCD screen. It also sends data measured to the "Thing speak" platform. The system helps to create awareness of the quality of air that one breathes daily. This monitoring device can deliver real-time measurements of air quality.

In previous papers, they have proposed and developed an IoT based Air quality monitoring system for smart cities. The real time data of the air quality is accessed through the smart devices and analysed to measure the impact on city dwellers. The smart devices are capable of measuring the temperature, humidity, Carbon monoxide, LPG, Smoke and other hazardous particulate matters like PM2.5 and PM10 levels in the atmosphere. The gathered data is accessible globally through an android application.

The Air Pollution Monitoring System was implemented and tested in real-life working condition and they are in the process of extending the nodes. In future, a large-scale node placement and data gathering have been planned for the purpose of forecasting of air pollution.

2 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 effects 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..

3 TECHNICAL SPECIFICATION:

3.1 hardware and software tools

Hardware:	**Software:**
NodeMCU(ESP8266) MQ135 MQ7 DHT11 4051 Multiplexer BreadBoard Jumper Wires	Arduino IDE Thinger.io Visual Studio Code Jupyter Notebook(anaconda)

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

4 DESIGN APPROACH AND DETAILS

4.1 Design approach/materials and methods

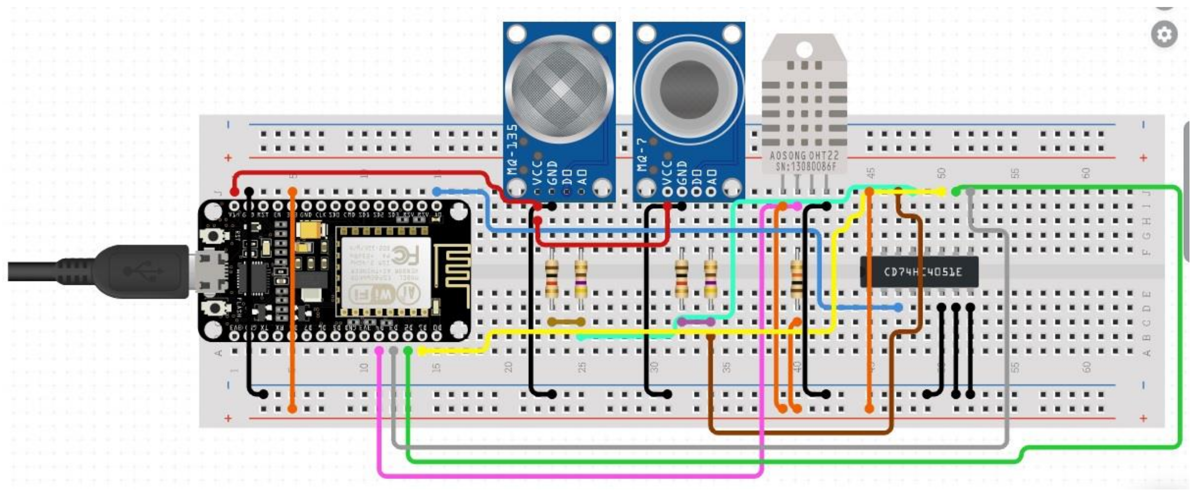


Fig4.1 Design approach

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.

4.2 Codes and standards

Arduino IDE code-

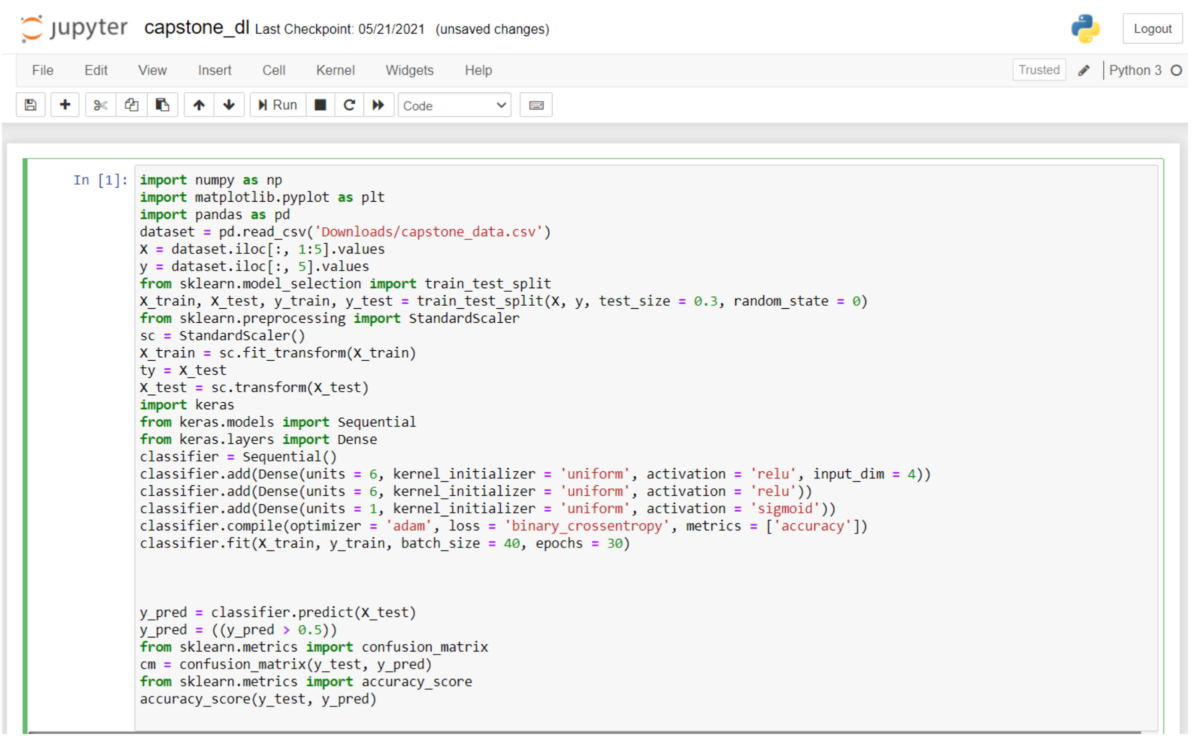
```
#include <Mux.h>

#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.
```

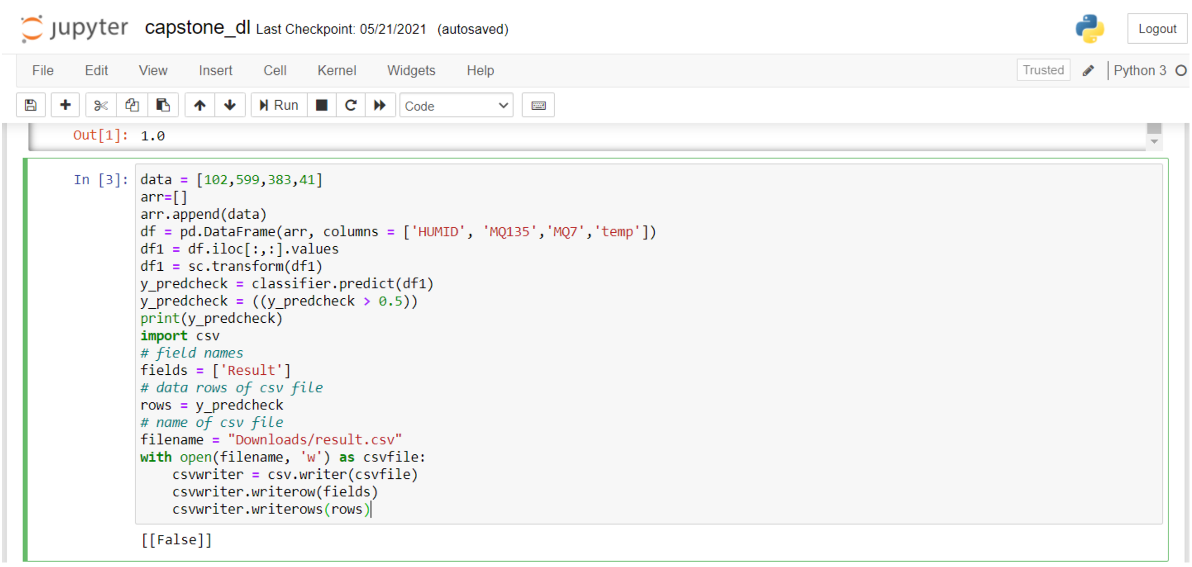
```
  thing.add_wifi(ssid, password); //--> Initialize wifi

  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, Logarithmic Regression and Artificial Neural Networks we will be able to predict if a location is safe or not given the future readings

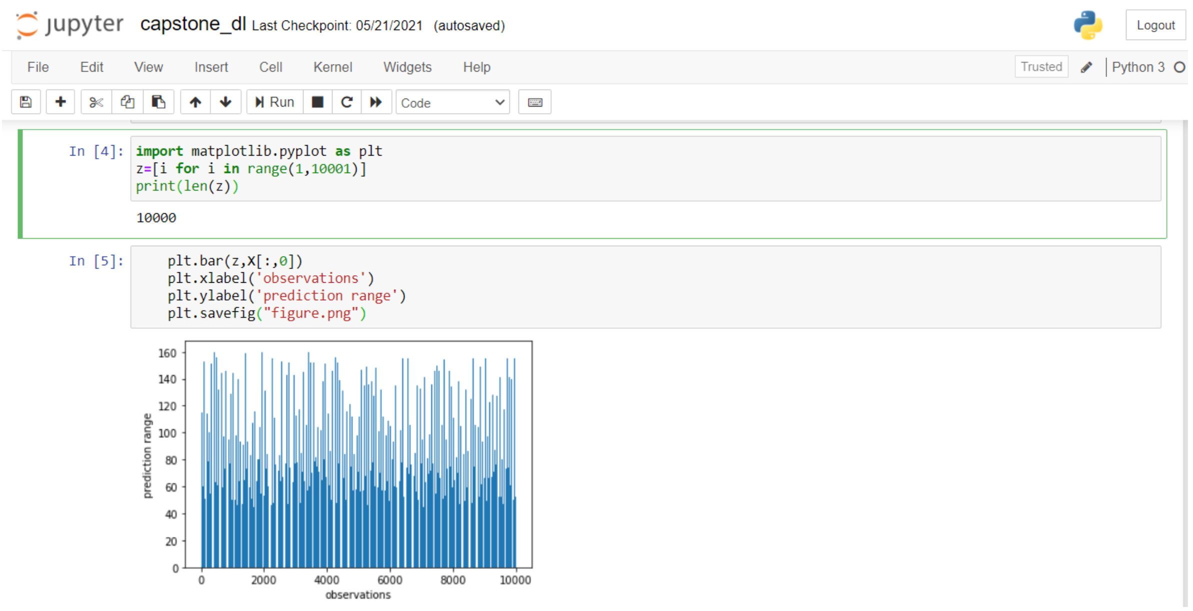
Deep Learning code


jupyter capstone_dl Last Checkpoint: 05/21/2021 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [1]: 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)
```


jupyter capstone_dl Last Checkpoint: 05/21/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
Out[1]: 1.0

In [3]: 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)

        [[False]]
```


jupyter capstone_dl Last Checkpoint: 05/21/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [4]: import matplotlib.pyplot as plt
        z=[i for i in range(1,10001)]
        print(len(z))

        10000

In [5]:     plt.bar(z,X[:,0])
            plt.xlabel('observations')
            plt.ylabel('prediction range')
            plt.savefig("figure.png")
```

prediction range
observations
0
20
40
60
80
100
120
140
160
0
2000
4000
6000
8000
10000

Fig 4.2.1 deep learning result

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

4.3 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.

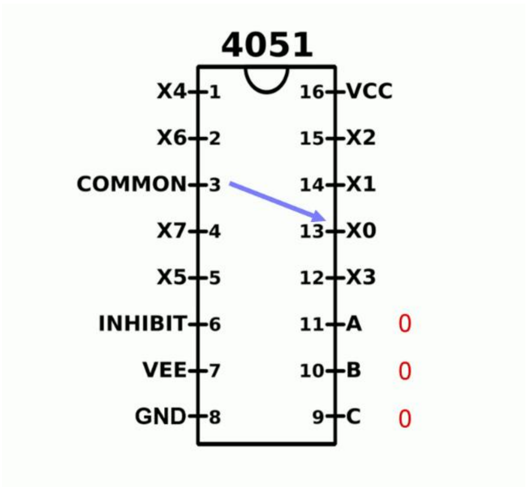


Fig 4.3 MUX 4051

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.

5 SCHEDULE, TASKS AND MILESTONES

TIME DURATION AND ACTIVITIES

TIME DURATION	ACTIVITIES
FEB 6	Finalized topic
FEB 7 – FEB 10	Initiation and Literature Survey
FEB 11- FEB 25	Finished prep-work on the approach and implementation
MAR 1- MAR 15	Received components. Worked on hardware and Arduino code testing out all possible libraries for accurate results
MAR 20- MAR 30	Worked on sending data to IOT platform. Tested out multiple platforms to get better efficiency
APR 1- APR 15	Debugging. Replaced few components
APR16 – APR 25	Using data bucket to store data gathered for two weeks
APR 26- MAY 10	Working on a machine learning and deep learning algorithm and passing the data gathered through the algorithm
MAY 11- MAY 20	Worked on app to display results but pivoted into making a website
MAY 21- MAY 25	Working on report

6 PROJECT DEMONSTRATION:

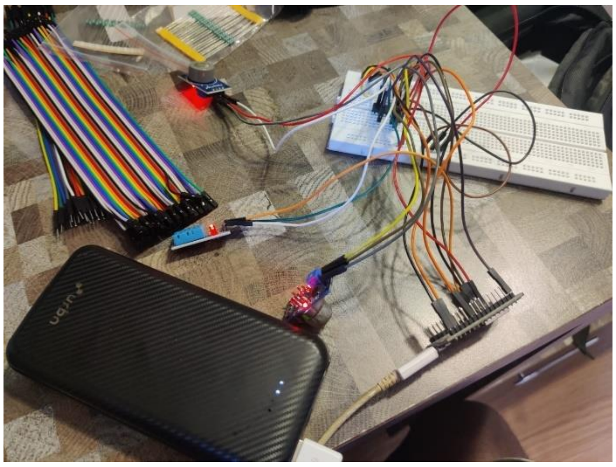
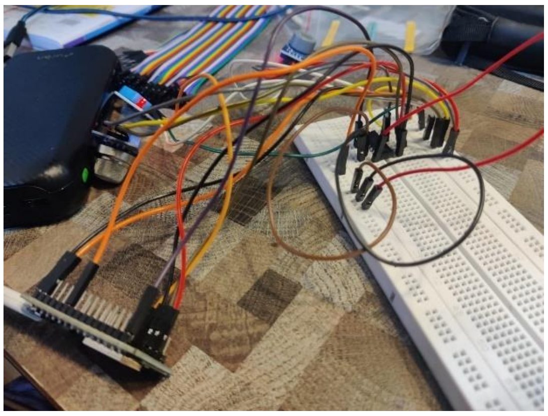
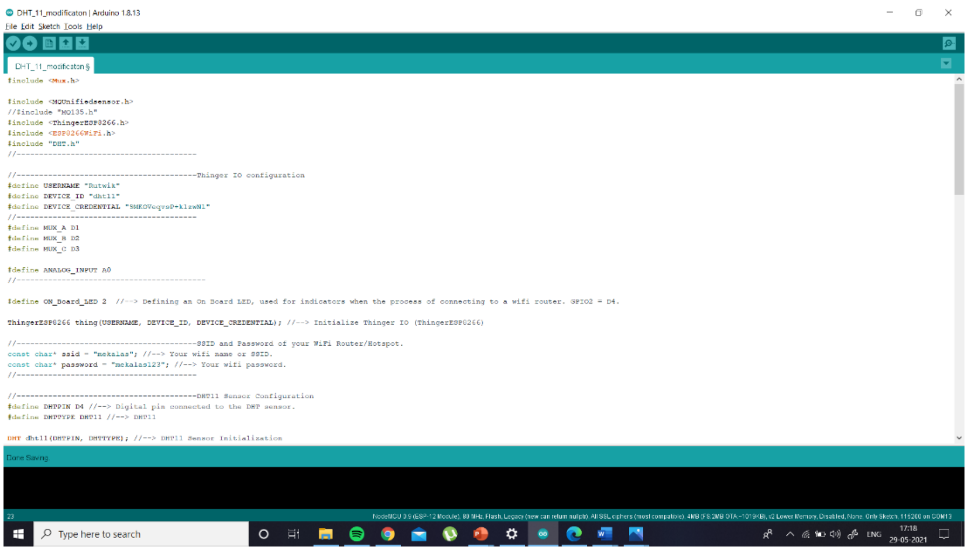



Fig 6.1 Hardware setup (a) Hardware setup (b)

We complete the hardware setup as shown in 4.1 Design approach

Arduino IDE


DHT_11_modificaton | Arduino 1.8.13
File Edit Sketch Tools Help
DHT_11_modificaton §
Type here to search
ENG
17:18
29-05-2021

```
#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 "SMKOVeqvsP+klzwNl"
//----------------------------------------
#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. GPIO2 = D4.

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
```

```
Done Saving.
```

```
23          NodeMCU 0.9 (ESP-12 Module), 80 MHz, Flash, Legacy (new can return nullptr), All SSL ciphers (most compatible), 4MB (FS:2MB OTA:~1019KB), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM13
```

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

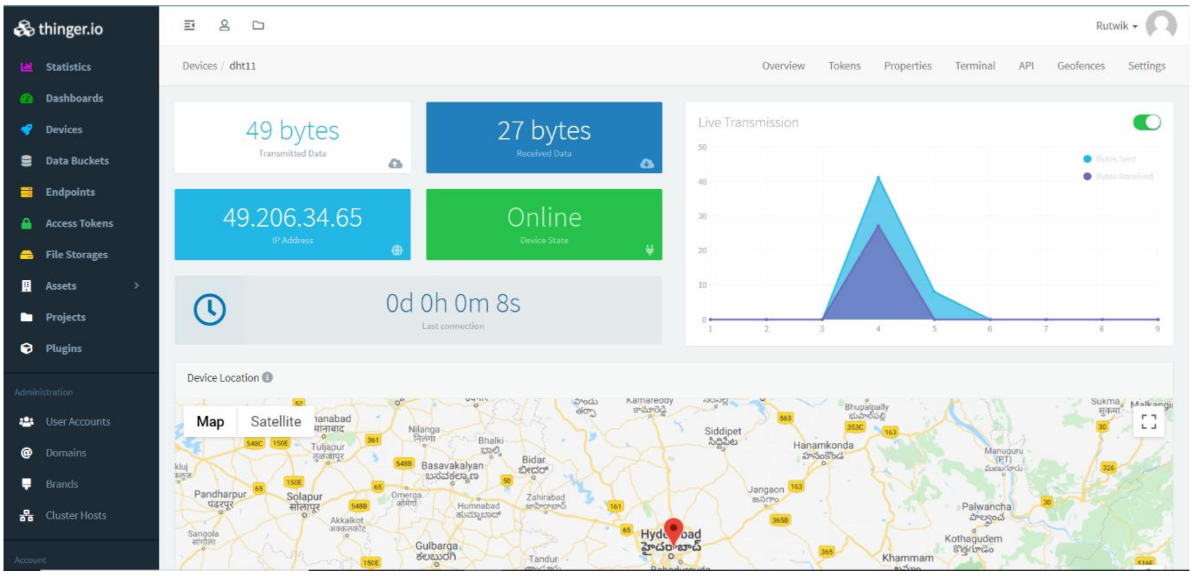
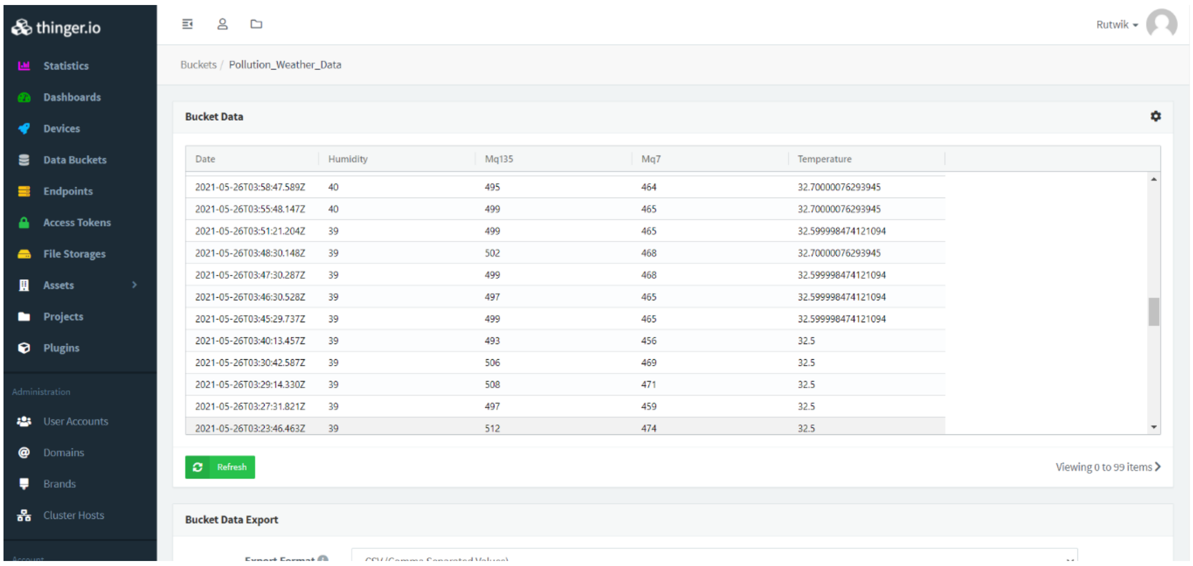


Fig 6.2 tinger.io (a)

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



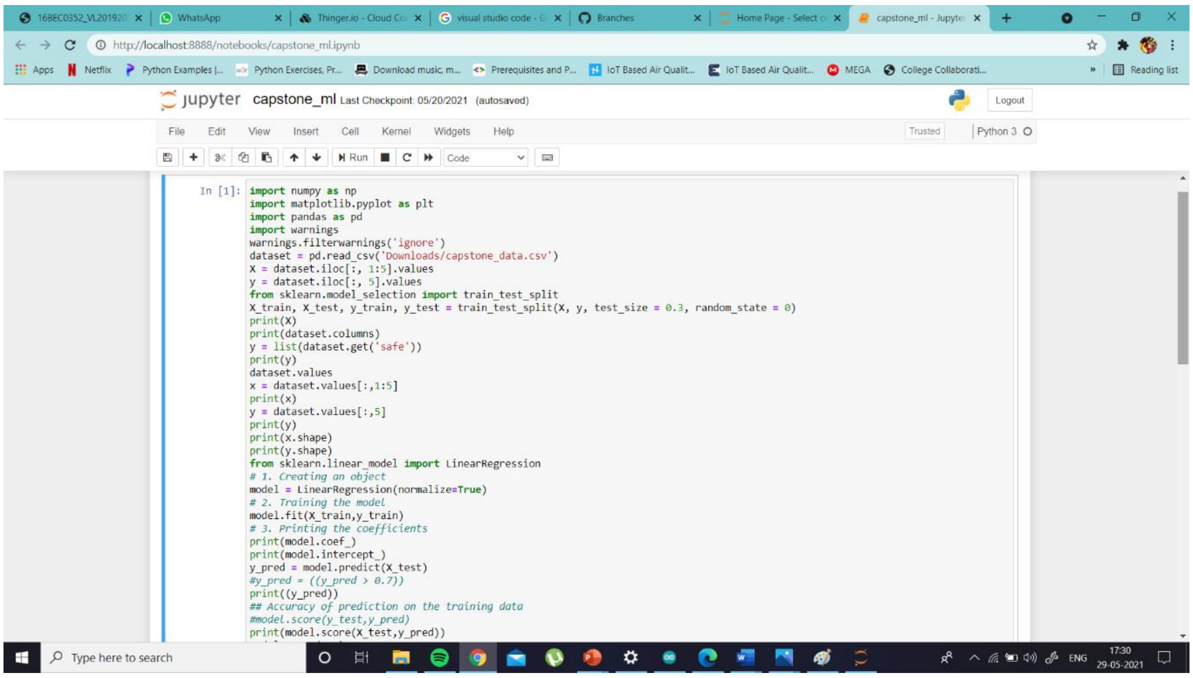
Bucket Data

Date	Humidity	Mq135	Mq7	Temperature
2021-05-26T03:58:47.589Z	40	495	464	32.70000076293945
2021-05-26T03:55:48.147Z	40	499	465	32.70000076293945
2021-05-26T03:51:21.204Z	39	499	465	32.599998474121094
2021-05-26T03:48:30.148Z	39	502	468	32.70000076293945
2021-05-26T03:47:30.287Z	39	499	468	32.599998474121094
2021-05-26T03:46:30.528Z	39	497	465	32.599998474121094
2021-05-26T03:45:29.737Z	39	499	465	32.599998474121094
2021-05-26T03:40:13.457Z	39	493	456	32.5
2021-05-26T03:30:42.587Z	39	506	469	32.5
2021-05-26T03:29:14.330Z	39	508	471	32.5
2021-05-26T03:27:31.821Z	39	497	459	32.5
2021-05-26T03:23:46.463Z	39	512	474	32.5

Fig 6.3 thinger.io (b)

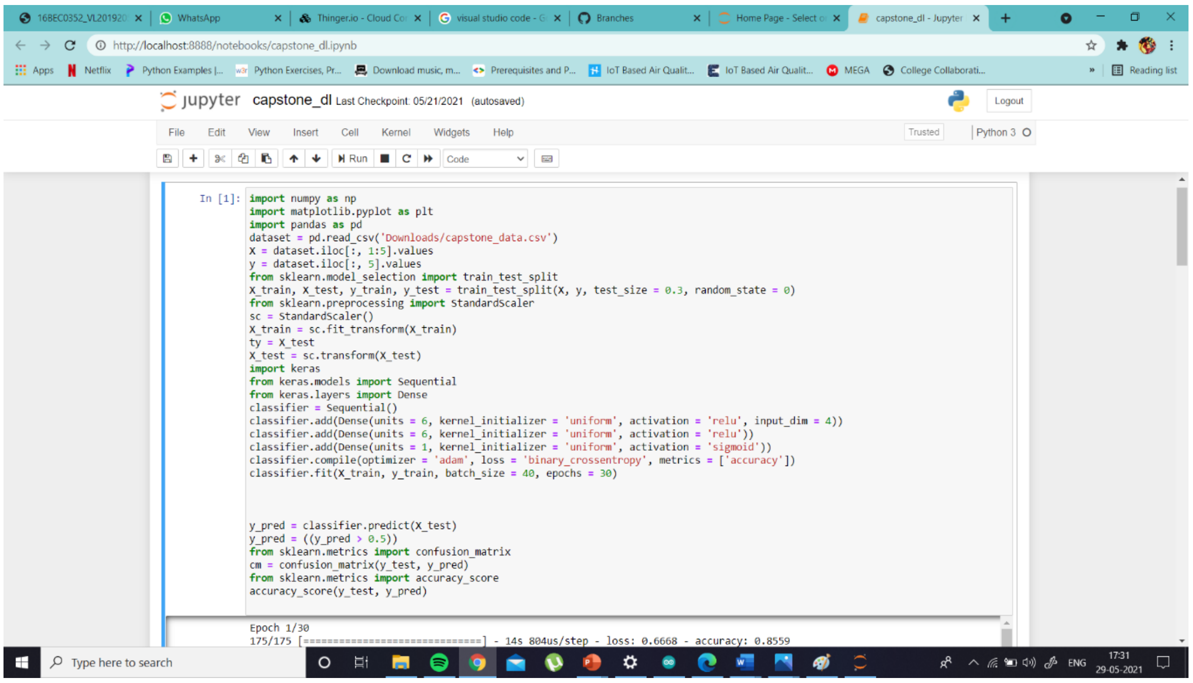
With the data gathered, one important feature we wanted to explore is analyzation of data and checking if we can predict the future result if an area is safe or not, based on Real-Time data supplied. For this we worked on Machine Learning and Deep Learning algorithms to achieve it. We have found that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

Machine learning



```
In [1]: 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))
```

Deep Learning



```
In [1]: 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)

        Epoch 1/30
        175/175 [==============================] - 14s 804us/step - loss: 0.6668 - accuracy: 0.8559
```

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

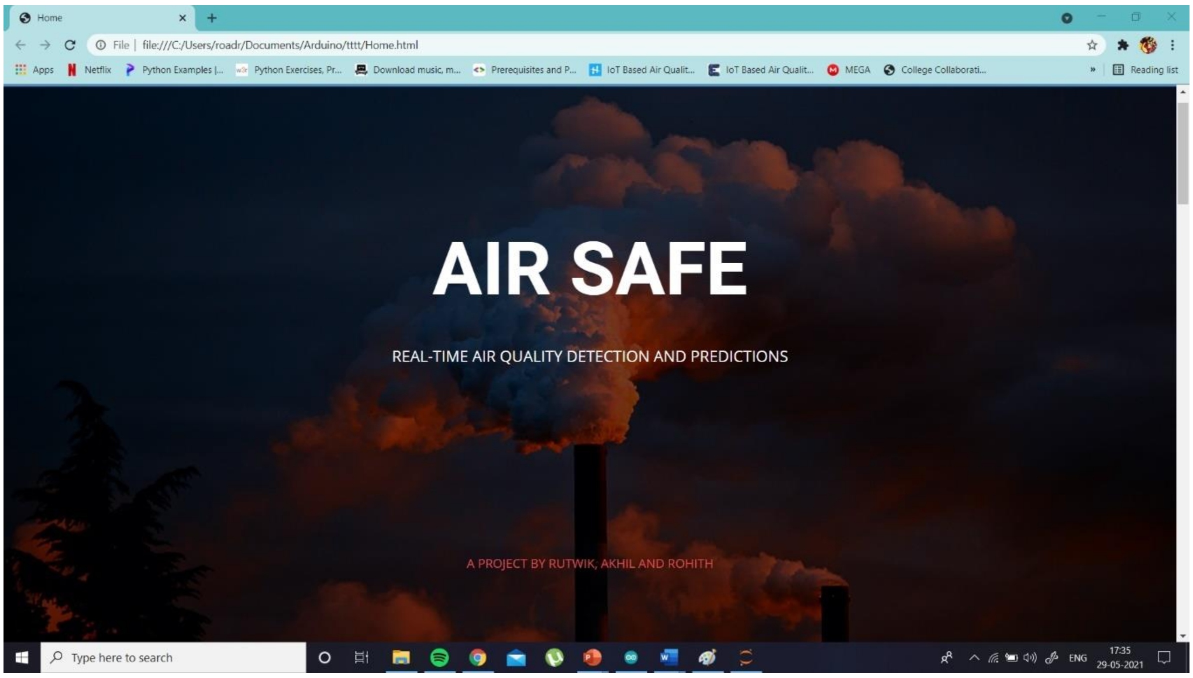
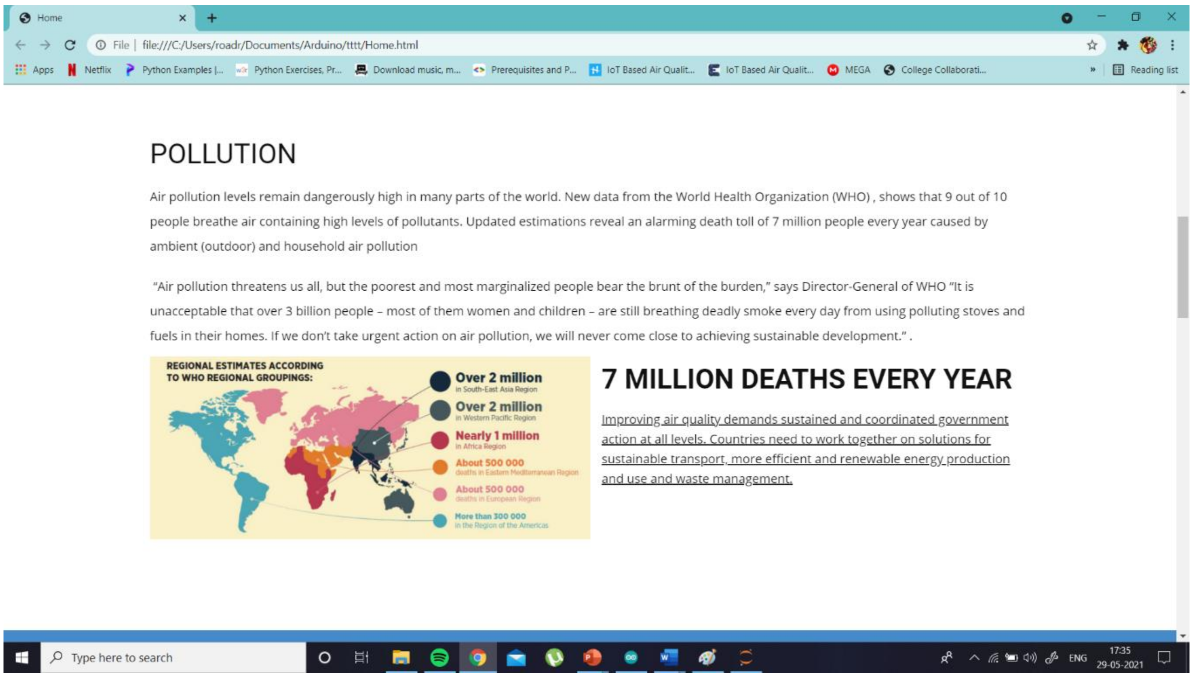

Home
file:///C:/Users/roadr/Documents/Arduino/tttt/Home.html
AIR SAFE
REAL-TIME AIR QUALITY DETECTION AND PREDICTIONS
A PROJECT BY RUTWIK, AKHIL AND ROHITH
Type here to search
17:35
29-05-2021

Fig 6.4 website (a)


Home
file:///C:/Users/roadr/Documents/Arduino/tttt/Home.html
REGIONAL ESTIMATES ACCORDING TO WHO REGIONAL GROUPINGS:
Over 2 million in South-East Asia Region
Over 2 million in Western Pacific Region
Nearly 1 million in Africa Region
About 500 000 deaths in Eastern Mediterranean Region
About 500 000 deaths in European Region
More than 300 000 in the Region of the Americas
Type here to search
17:35
29-05-2021

POLLUTION

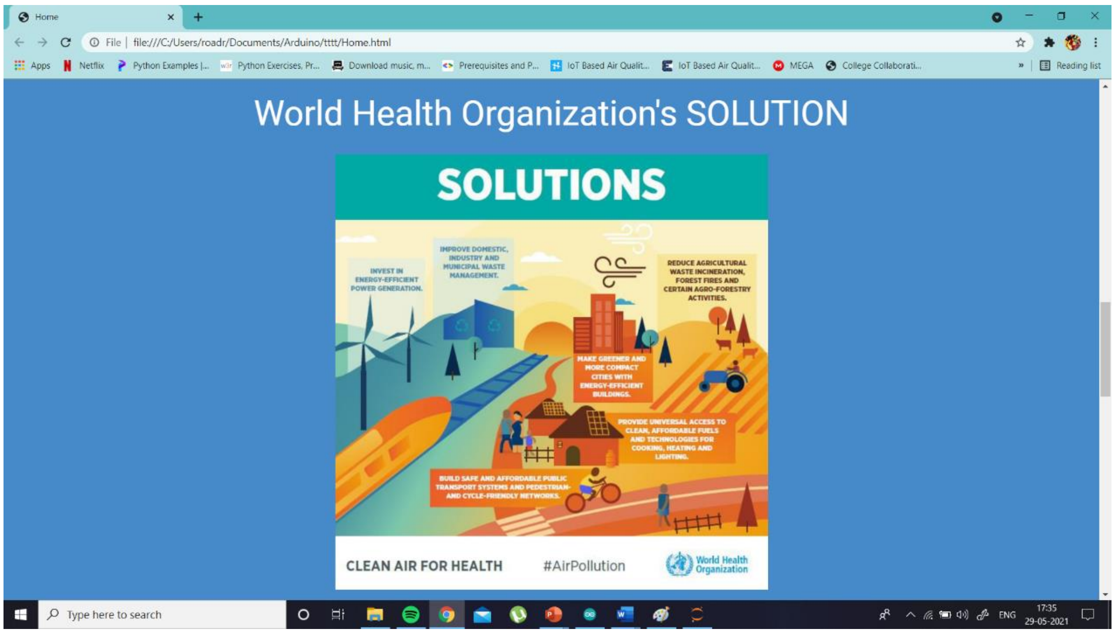
Air pollution levels remain dangerously high in many parts of the world. New data from the World Health Organization (WHO) , shows that 9 out of 10 people breathe air containing high levels of pollutants. Updated estimations reveal an alarming death toll of 7 million people every year caused by ambient (outdoor) and household air pollution

"Air pollution threatens us all, but the poorest and most marginalized people bear the brunt of the burden," says Director-General of WHO "It is unacceptable that over 3 billion people – most of them women and children – are still breathing deadly smoke every day from using polluting stoves and fuels in their homes. If we don't take urgent action on air pollution, we will never come close to achieving sustainable development." .

7 MILLION DEATHS EVERY YEAR

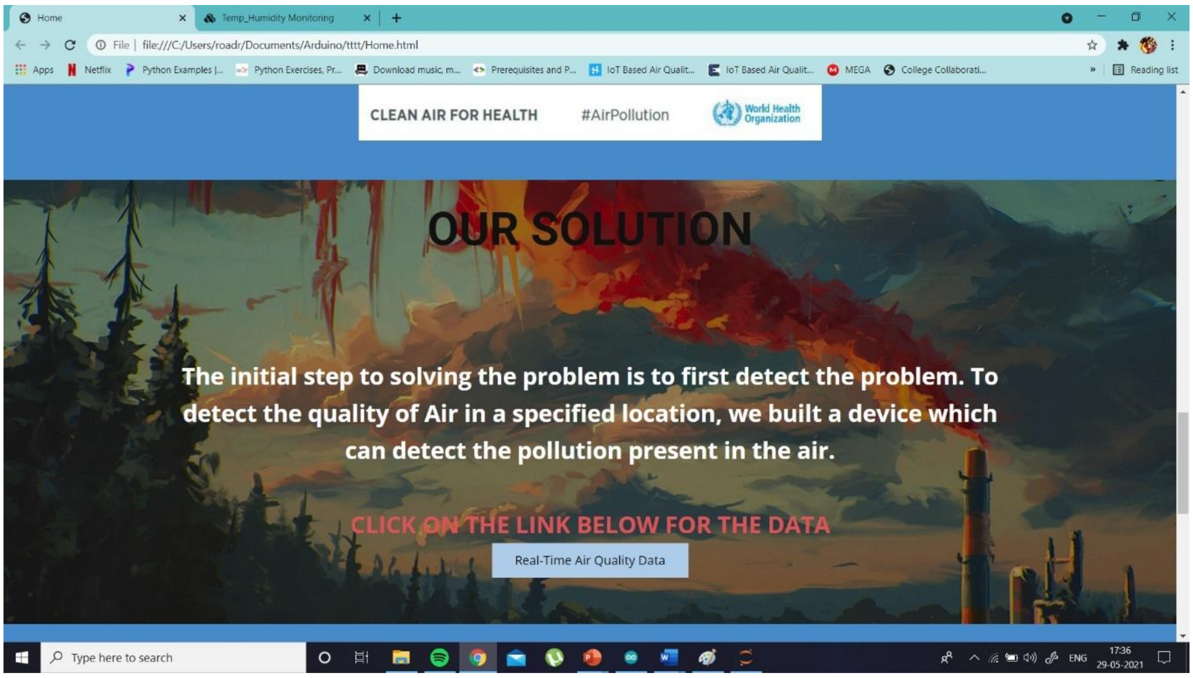
Improving air quality demands sustained and coordinated government action at all levels. Countries need to work together on solutions for sustainable transport, more efficient and renewable energy production and use and waste management.

Fig 6.4 website(b)


World Health Organization's SOLUTION

SOLUTIONS
INVEST IN ENERGY-EFFICIENT POWER GENERATION.
IMPROVE DOMESTIC, INDUSTRY AND MUNICIPAL WASTE MANAGEMENT.
REDUCE AGRICULTURAL WASTE INCINERATION, FOREST FIRES AND CERTAIN AGRO-FORESTRY ACTIVITIES.
MAKE GREENER AND MORE COMPACT CITIES WITH ENERGY-EFFICIENT BUILDINGS.
PROVIDE UNIVERSAL ACCESS TO CLEAN, AFFORDABLE FUELS AND TECHNOLOGIES FOR COOKING, HEATING AND LIGHTING.
BUILD SAFE AND AFFORDABLE PUBLIC TRANSPORT SYSTEMS AND PEDESTRIAN- AND CYCLE-FRIENDLY NETWORKS.
CLEAN AIR FOR HEALTH
#AirPollution
World Health Organization

Fig 6.4 website(c)


CLEAN AIR FOR HEALTH #AirPollution World Health Organization

OUR SOLUTION

The initial step to solving the problem is to first detect the problem. To detect the quality of Air in a specified location, we built a device which can detect the pollution present in the air.

CLICK ON THE LINK BELOW FOR THE DATA

Fig 6.4 website(d)

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

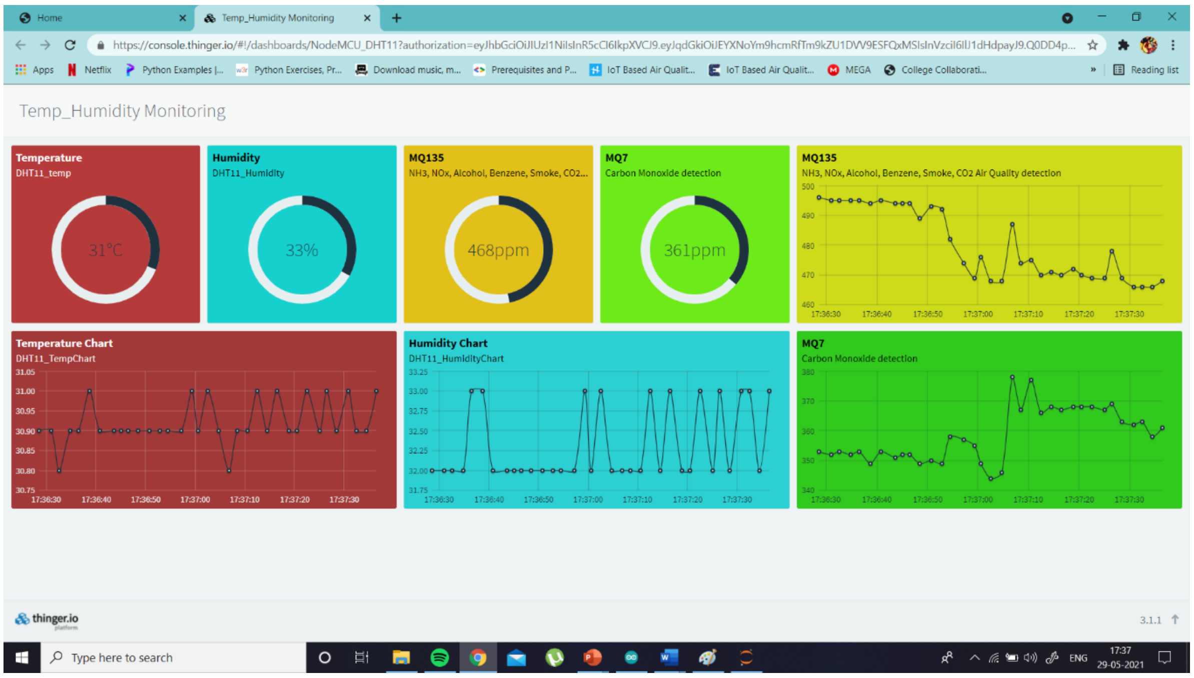
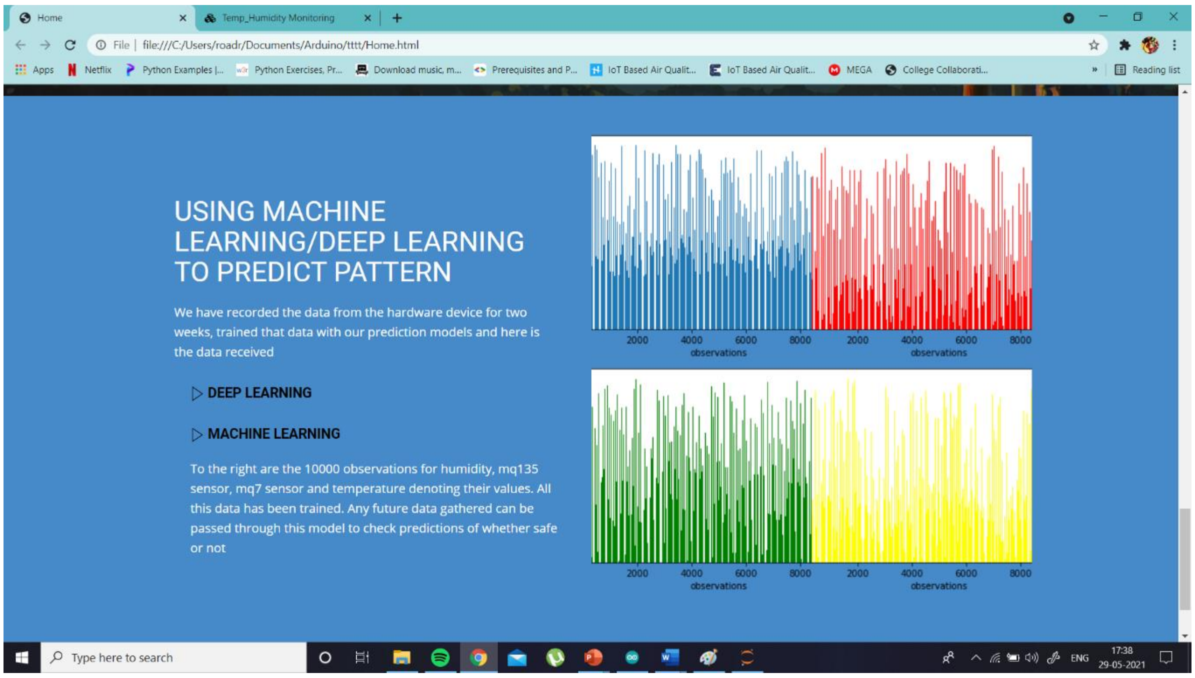


Fig 6.5 gathered data

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.



USING MACHINE LEARNING/DEEP LEARNING TO PREDICT PATTERN

We have recorded the data from the hardware device for two weeks, trained that data with our prediction models and here is the data received

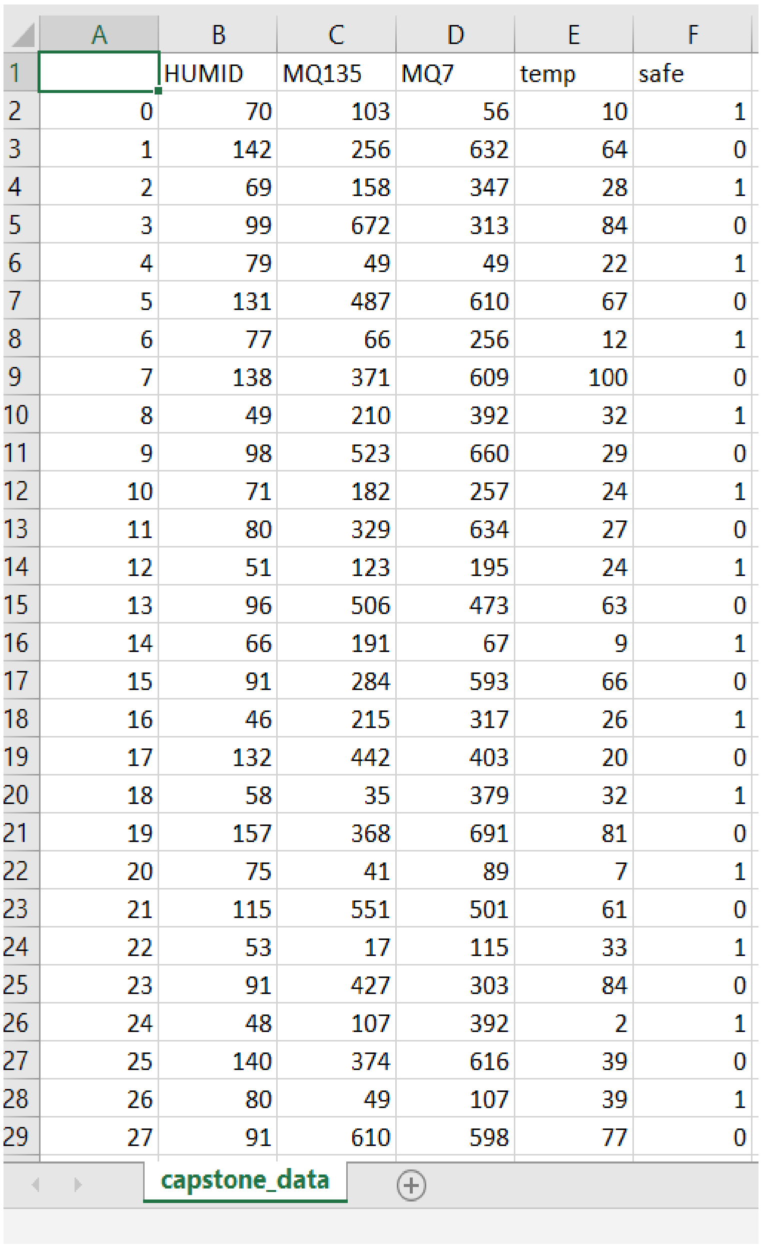
▷ DEEP LEARNING

▷ MACHINE LEARNING

To the right are the 10000 observations for humidity, mq135 sensor, mq7 sensor and temperature denoting their values. All this data has been trained. Any future data gathered can be passed through this model to check predictions of whether safe or not

Fig 6.6 predictions

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



	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

Fig 6.7 data stored

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

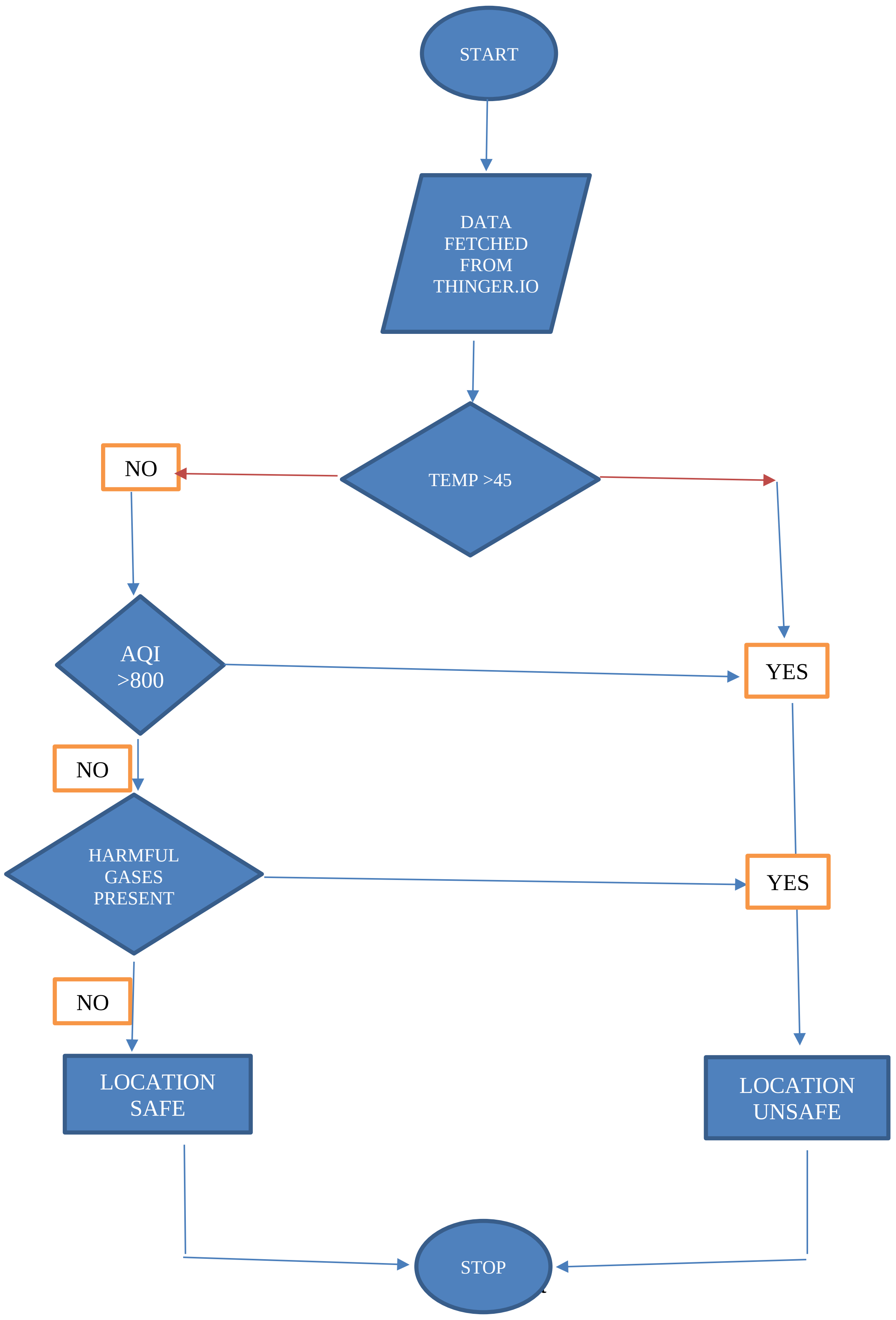


Fig 6.8 Flowchart

7 COST ANALYSIS /RESULTS AND DISCUSSION

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

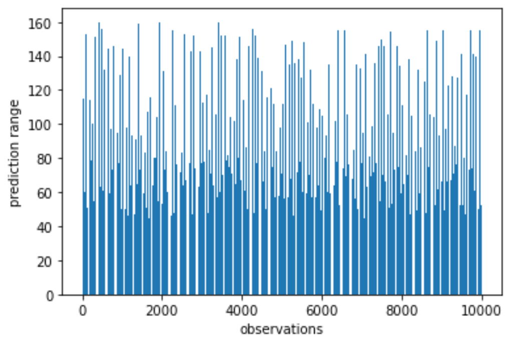
NodeMCU – 350 + jumper wires-10rs+4051 mux- 20rs + 3 sensors- 500rs = 880rs

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 TEMPERATURE



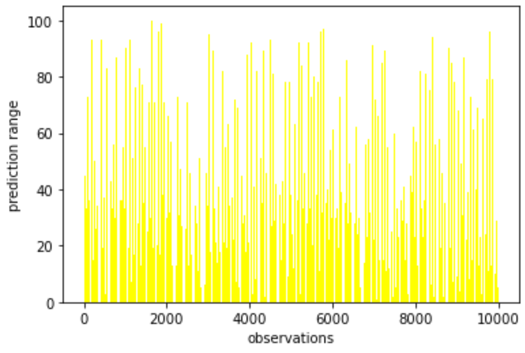
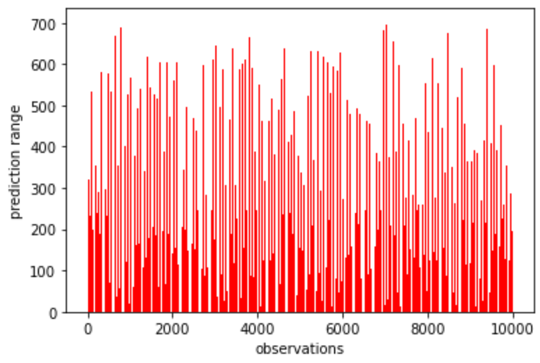
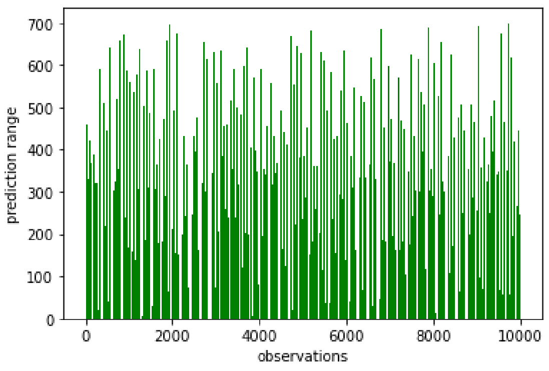


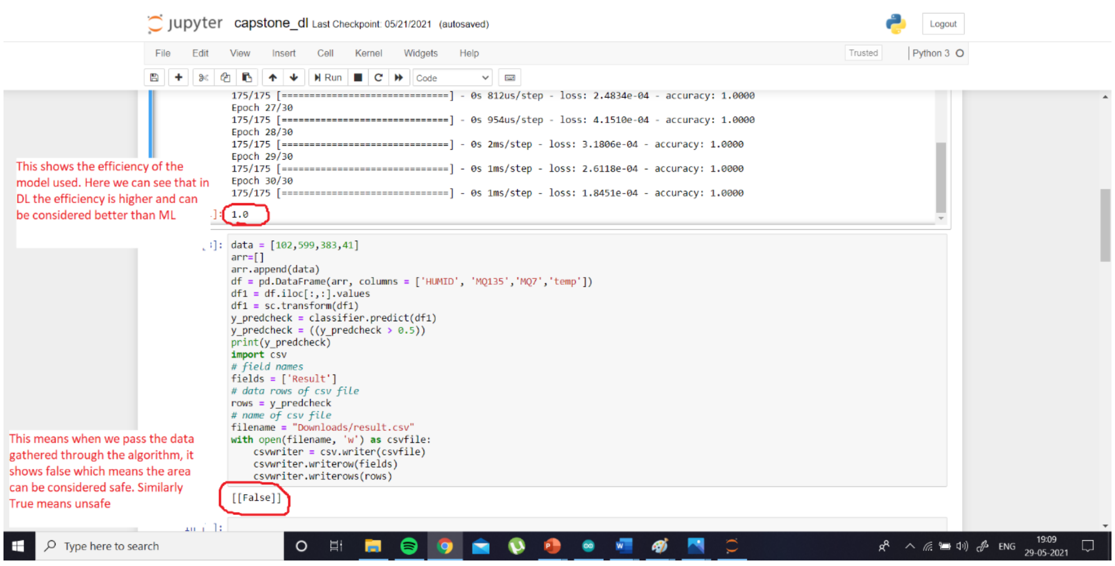
Fig 7.1 humidity Fig 7.2 temperature

MQ135 MQ7





[Fig 7.3 MQ135 O/P Fig 7.4 MQ7 O/P


jupyter capstone_dl Last Checkpoint: 05/21/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Type here to search
ENG
19:09
29-05-2021

```
175/175 [==============================] - 0s 812us/step - loss: 2.4834e-04 - accuracy: 1.0000
Epoch 27/30
175/175 [==============================] - 0s 954us/step - loss: 4.1510e-04 - accuracy: 1.0000
Epoch 28/30
175/175 [==============================] - 0s 2ms/step - loss: 3.1806e-04 - accuracy: 1.0000
Epoch 29/30
175/175 [==============================] - 0s 1ms/step - loss: 2.6118e-04 - accuracy: 1.0000
Epoch 30/30
175/175 [==============================] - 0s 1ms/step - loss: 1.8451e-04 - accuracy: 1.0000
1.0
```

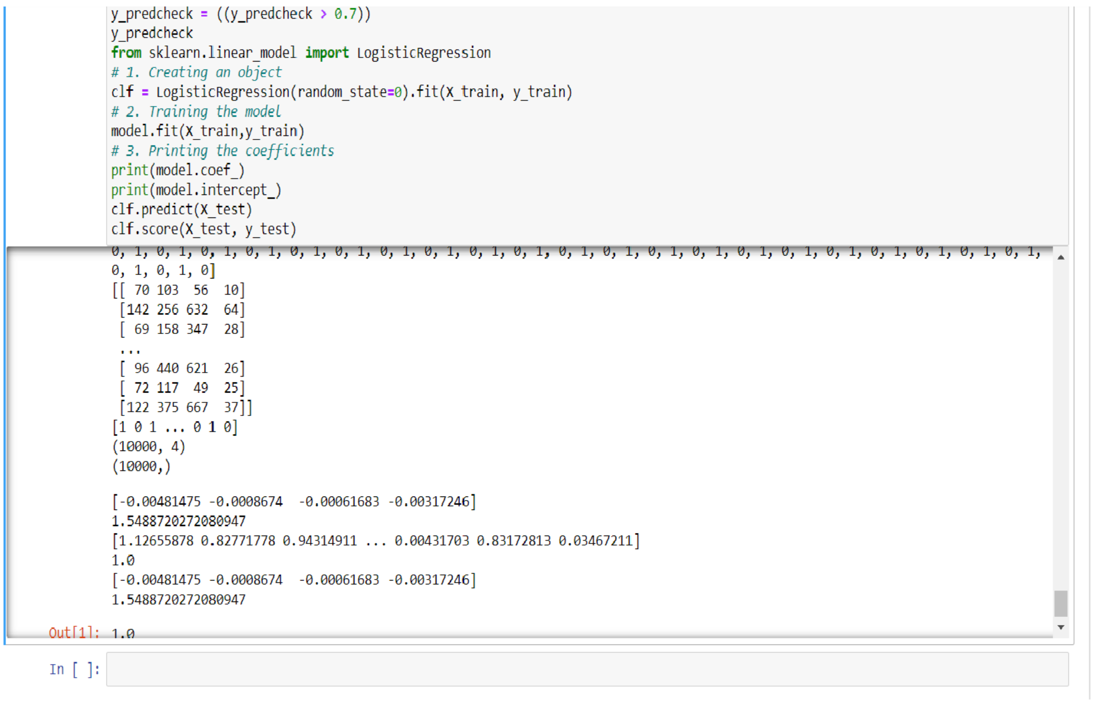
This shows the efficiency of the model used. Here we can see that in DL the efficiency is higher and can be considered better than ML

```
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)
[[False]]
```

This means when we pass the data gathered through the algorithm, it shows false which means the area can be considered safe. Similarly True means unsafe

Fig 7.5- Deep Learning output in jupyter

In the below graph, 1s and 0s denote if the area is safe or not. After analyzing the data scaling is done and result is shown if you area for given future information is going to be safe or not



```
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)
```

```
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0, 1, 0, 1, 0]
[[ 70 103  56  10]
 [142 256 632  64]
 [ 69 158 347  28]
 ...
 [ 96 440 621  26]
 [ 72 117  49  25]
 [122 375 667  37]]
[1 0 1 ... 0 1 0]
(10000, 4)
(10000,)

[-0.00481475 -0.0008674  -0.00061683 -0.00317246]
1.5488720272080947
[1.12655878 0.82771778 0.94314911 ... 0.00431703 0.83172813 0.03467211]
1.0
[-0.00481475 -0.0008674  -0.00061683 -0.00317246]
1.5488720272080947
Out[1]: 1.0

In [ ]:
```

Fig 7.6 Machine Learning output in jupyter

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

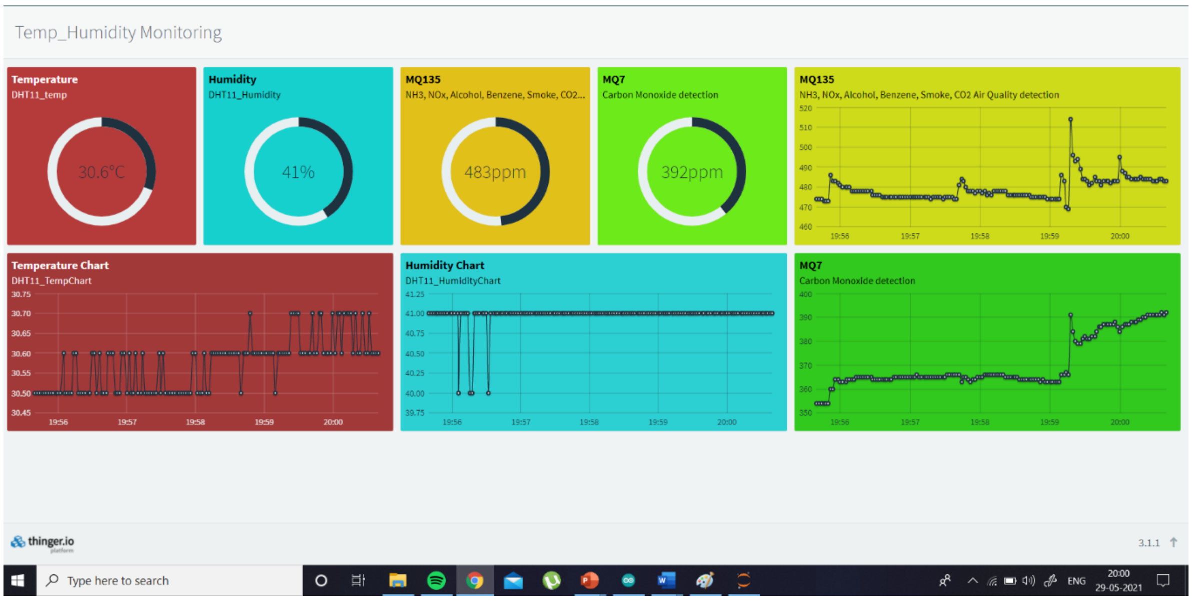


Fig 7.7 real time data gathered

8 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

9 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