IIT Madras

Centre For Innovation

EE05 :-
IOT FOR AIR QUALITY MANAGEMENT

ELECTRONICS CLUB

ASHWATH , SUNDAR

Team Members

	EE18B055	K J R K VARA PRASAD
	Worked on integrating the hardware part of sensors and arduino board and collecting data from them. Email : ee18b055@smail.iitm.ac.in	
	EE18B073	ANIRUDH S
	Worked on Machine learning and data analysis part for determining the levels of pollution. Developed a ML model with analysing real time data. Email : ee18b073@smail.iitm.ac.in	
	AE17B029	JEEVA SU
	Worked on NodeMCU coding and sending the sensors data to server. Collected the data and processed it real time. Email : ae17b029@smail.iitm.ac.in	
	AE18B022	CHANDRA A DESAI
	Worked on code for developing the required ML model.Worked on the project pitch. Email : ae18b022@smail.iitm.ac.in	

Table of Contents

1. Title of Invention

Portable Air pollution monitoring system

2. Field of Invention

Our goal is to create a cost-effective portable air pollution monitoring device for everyone's use. Also make the data available in real time for public.

(Need for invention)

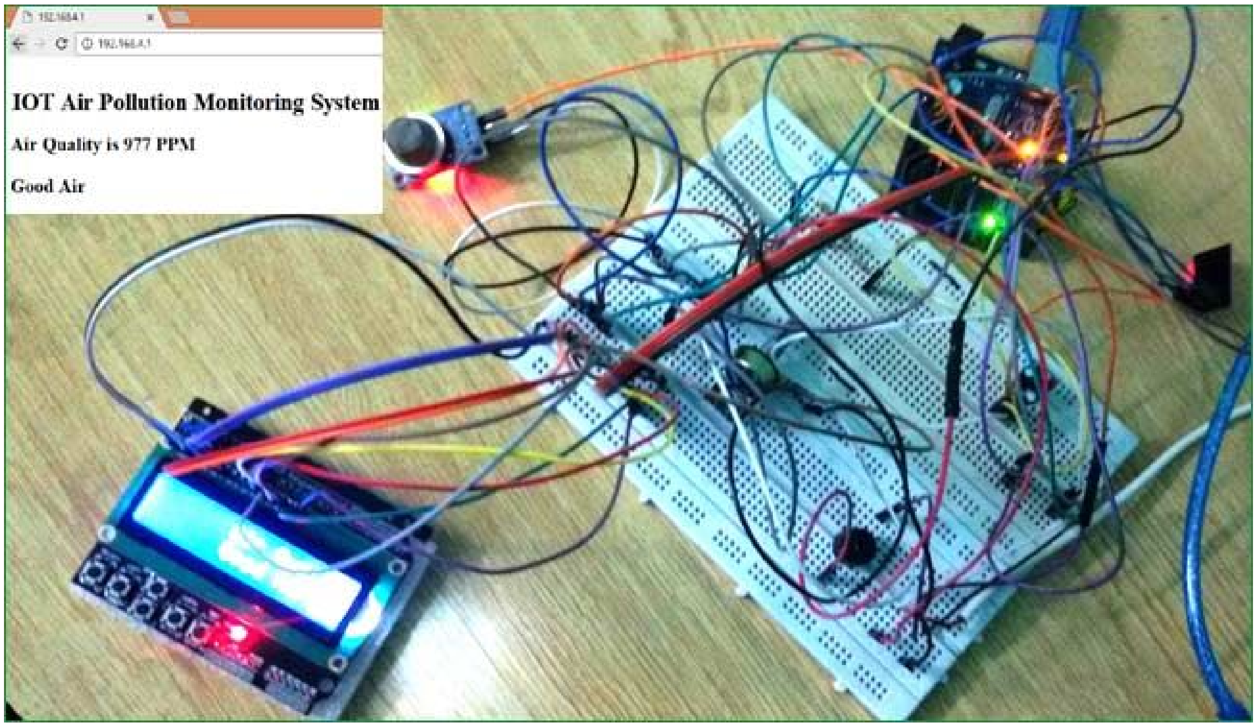
3. Background of Invention

The current air pollution monitoring systems are very bulky, expensive and usually not portable. Our aim is to produce portable air pollution monitoring systems at a much cheaper price to reach everyone in need of checking pollution levels for both everyday practises and industrial pollution monitoring.

(Current day stationary air quality monitoring device)

4. Object of Invention

The increasing level of air pollution is one of the most serious issues in the world. So, it becomes a necessity to monitor the air quality of our surroundings and predict the safe areas to accommodate human population. The current solutions that exist are either very costly or requires time to time human interference. We aim to create a hand-held cost effective pollution monitoring device for people to check the environment pollution, for controlling industrial exhaust pollution levels, live feed of air pollution levels of roads, industrial areas and much more.



```
IOT Air Pollution Monitoring System
Air Quality is 977 PPM
Good Air
```

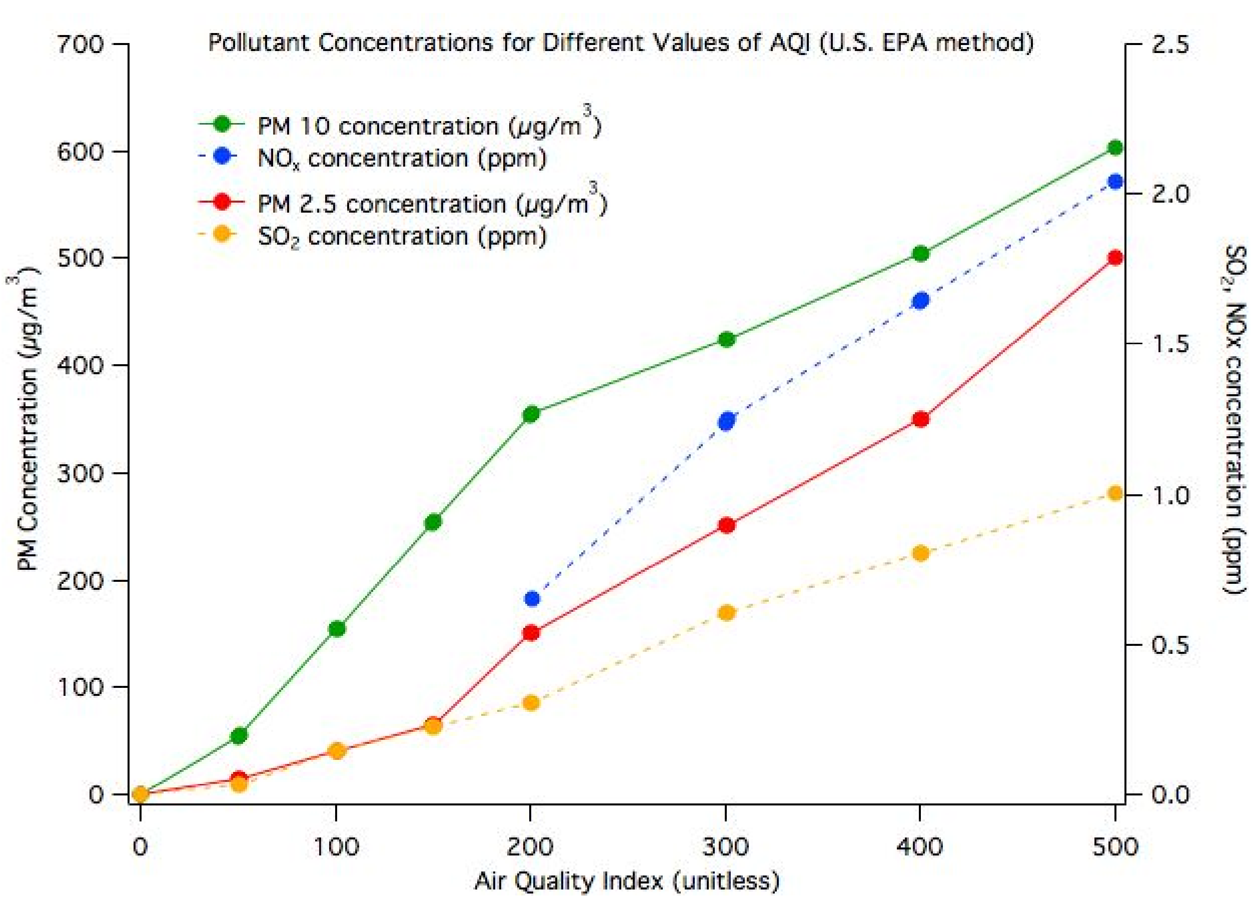
5. Design History and Overview

Our current design has a compact general circuit board fitted with ESP32, battery pack, SD card module and 5 sensors connected to it.

The sensors are:
MQ2 :Methane, LPG
MQ-3 :Alcohol, Ethanol
MQ7 :Carbon Monoxide
DHT11 :Temperature, Humidity

MG811 :Carbon Dioxide

Currently everything is exposed but we intend to create a casing where only the sensors are exposed. We are also planning to switch out ESP32 which communicates on WiFi to A9G board which has a GSM module as well as GPRS module.



6. Market Analysis

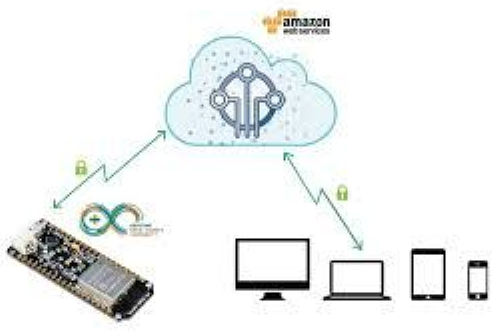
The current market for air pollution monitoring system has very few products that are too expensive due to low competition and the demand is quite low but it is increasing as people are getting more awareness about the environment and is expected to increase by a lot in upcoming years. The products are around the price range of 5000 ₹ to 25000 ₹ (Source: Google and Amazon). We expect our final product to be around 2500 ₹ with more features and still be a profitable business while working for a social cause.

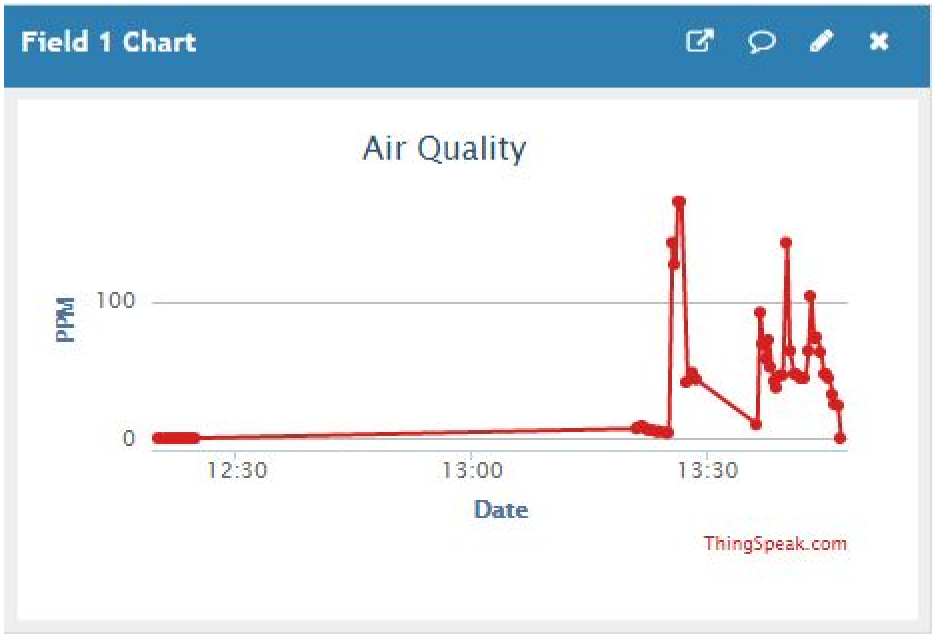
(OUR DEVICE)

7. What did we learn?

We learnt a lot about IOT, which includes microcontrollers, microprocessors like Arduino, ESP8266, ESP32, A9G, RPi and various sensors such Gas sensors, Thermal or Heat or Temperature and humidity sensors etc. Right from the beginning we were given tasks to research about sensors and their working mechanism. We learnt about various communication techniques in IOT, Arduino coding, socket programming on both server and client sides. We learnt about gateways, MQTT, SigFox and various modules which we are using in our project such as Wi-Fi module, GPS/GSM module, SD card module, etc.

We also had experience on building the prototype including soldering works and such. We had a great team bonding and learnt a lot about working efficiently as a team.





8. Acknowledgement

We are grateful to our mentor: Nishant and the team cores for their guidance and timely help in all our undertakings, the other club members, with whom we evolved and was motivated by their work too, the club and CFI for giving us the chance to work on our ideas, the institute for providing us the space for a wonderful bringing up.

Appendix - A : Data Sheets and Calculations
Appendix - B : Electrical Schematics
Appendix - C : Source Code

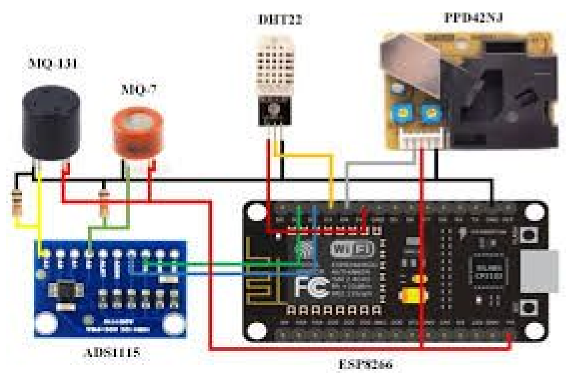
Data Collected:

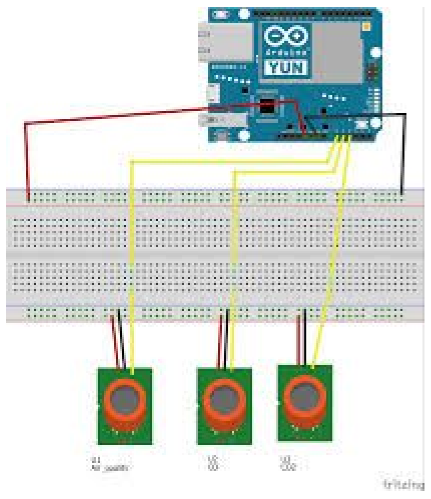
1)Dataset Collection

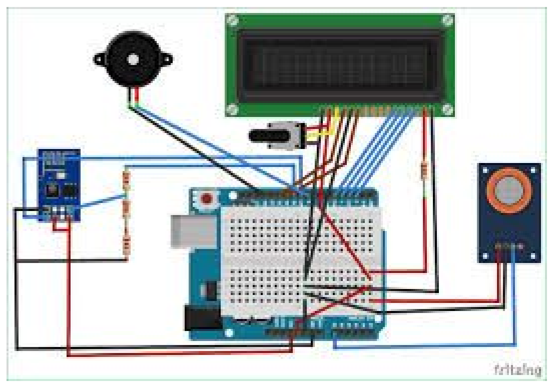
2)Plant Details

3)Time data

Electrical Schematics:







Source Code:

Function required to find the real time location of the device

```
import urllib
!pip install requests
import requests
import re
import xml.etree.ElementTree as ET

def getCellTowerInfo():
    print ('Logging in to router home page')
    baseurl = 'http://192.168.0.1/'
    url = baseurl + 'login.cgi'
    payload = {'ID':'admin','PASSWORD':'admin','REDIRECT':'index.asp','REDIRECT_ERR':'login.asp'}
    response = requests.post(url, data = payload)
    sessionPattern = re.compile('BID\d*')
    sessionId = sessionPattern.findall(response.text)[0]

    #get raw cell data
    payload ={'COUNT':'3',
    'WWW_SID':sessionId,
    'ACTION_1':'function',
    'NAME_1':'get_lac',
    'VALUE_1':'',
    'ACTION_2':'function',
    'NAME_2':'get_cell_id',
    'VALUE_2':'',
    'ACTION_3':'function',
    'NAME_3':'get_op_info',
    'VALUE_3':''}
    response = requests.post(url= 'http://192.168.0.1/'+'rpc.cgi', data = payload)

    #parse data
    results = ET.fromstring(response.text).findall('rpc_result')
    lac = results[0].text
    cellid = results[1].text
    operator = results[2].text
    print ('Got tower location, logging out')
    print (lac, cellid, operator)

    logoutUrl = 'http://192.168.0.1/'+ 'logout.cgi'
    payload ={'WWW_SID':sessionId}
    requests.post(logoutUrl, data = payload)
    return (lac, cellid, operator)
```

The model built predicts the air quality ahead of time.Train such a model, a cost function based on mean squared error method was developed.The Objective then was to minimize the loss and obtain as high accuracy as possible.

```
function J = computeCost(X, y, theta)
%COMPUTECOST Compute cost for linear regression
%   J = COMPUTECOST(X, y, theta) computes the cost of using theta as the
%   parameter for linear regression to fit the data points in X and y

% Initialize some useful values
m = length(y); % number of training examples
J = 0;
```

```
h=zeros(size(y));
h=X*theta;
for i=1:size(y,1)
  J=J+(h(i)-y(i))^(2);
endfor
J=J/(2*m);

end
```

The obtained data is plotted on the graphs with appropriate scale to visualize a clear variation of the gas concentration with time.

```
#plotting the obtained data for each gas in different plots

subplot(2,1,1);
plot(timeofday,co,'r--*');


subplot(2,1,2);
plot(timeofday,co2,'b--o');
hold on;
pause;
pause;
pause;
pause;
```

To train the model to obtain high accuracy and minimize the loss as much as possible we use SGD(Stochastic Gradient Descent) Algorithm.
We also plot the variation of the cost function as a function of number of iterations.

```
function [theta, J_history] = gradientDescent(X, y, theta, alpha,
num_iters)
%GRADIENTDESCENT performs gradient descent to learn theta
%   theta = GRADIENTDESCENT(X, y, theta, alpha, num_iters) updates theta by
%   taking num_iters gradient steps with learning rate alpha

% Initialize some useful values
m = length(y); % number of training examples
J_history = zeros(num_iters, 1);

for iter = 1:num_iters
    h=zeros(size(y));
    h=X*theta;
    theta_grad=zeros(size(theta));
    theta_grad=X'*(h-y);
    theta=theta-(alpha)*(1/m)*(theta_grad);

    J_history(iter) = computeCost(X, y, theta);

end

end
```

To obtain the solution of minimum loss for linear regression model we can instead of using SGD use a matrix method which reduces the execution time.

```
function [theta] = normalEqn(X, y)
%NORMALEQN Computes the closed-form solution to linear regression
```

```
%   NORMALEQN(X,y) computes the closed-form solution to linear
%   regression using the normal equations.

theta = zeros(size(X, 2), 1);

theta=((X'*X)^(-1))*(X'*y);

end
```

Function to obtain the concentrations from the retrieved dataset and remove outliers from the dataset:

```
Function[co,co2,ch4,alcohol,
         timeofdayx,timeofdayy,
         timeofdayz,timeofdayw]=outliers(co,co2,ch4,alcohol,timeofday)

#file to remove outliers in the data
m=size(co,1);

n=size(co2,1);
m1=size(ch4,1);
n1=size(alcohol,1);

#calculating mean and standard deviation to calculate the outliers

meanco=mean(co);
stdco=std(co);

meanco2=mean(co2);
stdco2=std(co2);

meanch4=mean(ch4);
stdch4=std(ch4);

meanalcohol=mean(alcohol);
stdalcohol=std(alcohol);
```

```
#removing the outliers
j=1;
for i=1:m
if(abs(co(i)-meanco)<=(stdco))
X(j)=co(i);
tx(j)=timeofday(i);
j=j+1;
endif

endfor
#transforming into col vectors
co=X';
timeofdayx=tx';

j=1;
for i=1:n
if(abs(co2(i)-meanco2)<=(stdco2))
Y(j)=co2(i);
ty(j)=timeofday(i);
j=j+1;
endif

endfor


#transforming into col vectors

co2=Y';
timeofdayy=ty';


j=1;
for i=1:m1
if(abs(ch4(i)-meanch4)<=(stdch4))
Z(j)=ch4(i);
tz(j)=timeofday(i);
j=j+1;
endif

endfor
#transforming into col vectors
```

```
ch4=Z';
timeofdayz=(tz)';


j=1;
for i=1:n1
if(abs(alcohol(i)-meanalcohol)<=(stdalcohol))
W(j)=alcohol(i);
tw(j)=timeofday(i);
j=j+1;
endif

endfor
#transforming into col vectors
alcohol=W';
timeofdayw=tw';



#plotting the transformed data

subplot(4,1,1);
plot(timeofdayx,co,'r--*');
hold off;
subplot(4,1,2);
plot(timeofdayy,co2,'b--o');
hold off;
subplot(4,1,3);
plot(timeofdayz,ch4,'g--*');
hold off;
subplot(4,1,4);
plot(timeofdayw,alcohol,'bk--*');
hold off;
pause
pause
pause
pause
```

Plant Prediction Model:

```
#code for each plant :Tulasi :1,
                      Areca palm :2,
                      Bamboo palm :3,
                      Lady palm :4,
                      Rhapis excelsa :5


function [code]=plantprediction()
  load ("conc.txt");
  time=load('time.txt');
fprintf("predicting the plant for gas removal.......\n\n");

#finding out the size of the data
m=size(conc,1);

#splitting the data into four parts
co=conc(1:(m/4));
co2=conc((m/4)+1:(m/2));
ch4=conc((m/2)+1:(3*m)/4);
alcohol=conc(((3*m)/4)+1:m);
#finding average of each value:
meanco=mean(co);
meanco2=mean(co2);
meanch4=mean(ch4);
meanalcohol=mean(alcohol);

#loading the absorption fractions for each plant
ab1=[0.9,0.1,0.0,0] ;  #   "Areca palm"
ab2=[0.0,0.8,0.1,0.1]; # "Bamboo palm"
ab3=[0.1,0.1,0.8,0.0] ;# "Tulasi"
ab4=[0.0,0.1,0,0.9]   ;# "Lady Palm"
ab5=[0,0.05,0.05,0.9] ;#"Rhapis excelsa"

ab=[ab1;ab2;ab3;ab4;ab5];
maxquantity=max(ab);


predictcode=zeros(size(maxquantity));
for j=1:size(ab,2)
```

```
  for i=1:size(ab,1)

  if(maxquantity(j)==ab(i,j))
  predictcode(j)=i;
  endif
  endfor
endfor

if(meanco>50)
if(predictcode(1)==1)
  fprintf("CO EXCEEDS \nUse plant  Areca palm\n");
endif
if(predictcode(1)==2)
  fprintf("CO EXCEEDS \nUse plant  Bamboo palm\n");
endif
   if(predictcode(1)==3)
  fprintf("CO EXCEEDS \nUse plant Tulasi\n");
endif
   if(predictcode(1)==4)
  fprintf("CO EXCEEDS \nUse plant Lady palm\n");
endif
if(predictcode(1)==5)
  fprintf("CO EXCEEDS \nUse plant Rhapis excelsa\n");
endif

elseif(meanco<50)
fprintf("\nCO IS BELOW the PERMISSIBLE LIMIT\n");
endif




if(meanco2>350)
 if(predictcode(2)==1)
  fprintf("CO2 EXCEEDS \nUse plant  Areca palm\n");
endif
if(predictcode(2)==2)
  fprintf("CO2 EXCEEDS \nUse plant  Bamboo palm\n");
endif
   if(predictcode(2)==3)
  fprintf("CO2 EXCEEDS \nUse plant  Tulasi\n");
```

```
endif
   if(predictcode(2)==4)
  fprintf("CO2 EXCEEDS \nUse plant  Lady palm\n");
endif
if(predictcode(2)==5)
  fprintf("CO2 EXCEEDS \nUse plant  Rhapis palm\n");
endif

elseif(meanco2<350)
fprintf("\nCO2 IS BELOW the PERMISSIBLE LIMIT\n");
endif




if(meanch4>400)
  if(predictcode(3)==1)
  fprintf("CH4 EXCEEDS \nUse plant  Areca palm\n");
endif
if(predictcode(3)==2)
  fprintf(" CH4 EXCEEDS \nUse plant  Bamboo palm\n");
endif
   if(predictcode(3)==3)
  fprintf("CH4 EXCEEDS \nUse plant Tulasi\n");
endif
   if(predictcode(3)==4)
  fprintf("CH4 EXCEEDS \nUse plant  Lady palm\n");
endif
if(predictcode(3)==5)
  fprintf("CH4 EXCEEDS \nUse plant  Raphsis excelsa\n");
endif


elseif(meanch4<400)
fprintf("\nCH4 IS BELOW the PERMISSIBLE LIMIT\n");
endif


if(meanalcohol>100)
  if(predictcode(4)==1)
  fprintf("ALCOHOL EXCEEDS \nUse plant  Areca palm\n");
```

```
endif
if(predictcode(4)==2)
  fprintf("ALCOHOL EXCEEDS \nUse plant   Bamboo palm\n");
endif
   if(predictcode(4)==3)
  fprintf("ALCOHOL EXCEEDS \nUse plant  Tulasi\n ");
endif
   if(predictcode(4)==4)
  fprintf("ALCOHOL EXCEEDS \nUse plant  Lady palm\n");
endif
if(predictcode(4)==5)
  fprintf("ALCOHOL EXCEEDS \nUse plant  Raphsis excelsa\n");
endif
elseif(meanalcohol<100)
fprintf("\nALCOHOL IS BELOW the PERMISSIBLE LIMIT\n");
endif

endfunction
```

Future Prediction Module:

```
% Predict values for different instance of time
function [co,co2,ch4,alcohol]= predictconc(X,Y,Z,W,theta1,theta2,theta3,theta4)
  co=[1 X X^2 X^3]*theta1;
  co2=[1 Y Y^(2) Y^(3)]*theta2;
  ch4=[1 Z Z^2 Z^3]*theta3;
  alcohol=[1 W W^2 W^3]*theta4;

endfunction
```

ALL MODELS PUT TOGETHER:

```
#adding the procured data
hold off;
```

```
conc=load('conc.txt');
time1=load('time.txt');


#finding out the size of the data
m=size(conc,1);

#adding the concentrations of each gas
co=conc(1:(m/4));
co2=conc((m/4)+1:(m/2));
ch4=conc((m/2)+1:(3*m)/4);
alcohol=conc(((3*m)/4)+1:m);

#obtaining the time at which the data is collected
hours=zeros(size(time1,1),1);
hours=time1(:,1);

mins=zeros(size(time1,1),1);
mins=time1(:,2);

secs=zeros(size(time1,1),1);
secs=time1(:,3);

#representing the time as a function of number of seconds
#elapsed in the day till then


timeofday=zeros(size(hours));
timeofday=zeros(size(hours));
timeofday=hours.*3600+mins.*60+secs;


#plotting the obtained data from the cloud
fprintf('Plotting Data ...\n');
datadisplay();
pause
pause
pause
pause
```

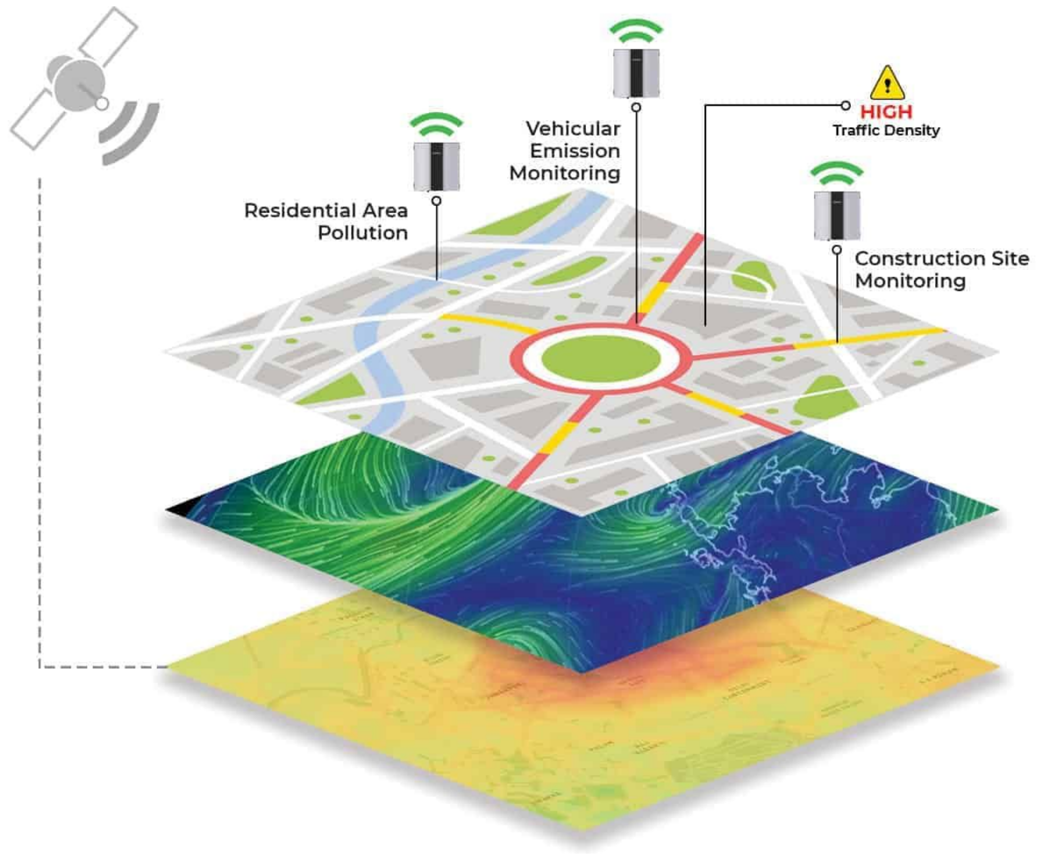
```
#removing the outliers from the graphs

[co,co2,ch4,alcohol,timeofdayx,timeofdayy,timeofdayz,timeofdayw]=outliers(c
o,co2,ch4,alcohol,timeofday);

#running cost functions in the new obtained data values
m=size(co,1);
n=size(co2,1);
n1=size(alcohol,1);
m1=size(ch4,1);
X = [ones(m, 1),
timeofdayx(:,1),timeofdayx(:,1).*timeofdayx(:,1),timeofdayx(:,1).*timeofday
x(:,1).*timeofdayx(:,1)];
Y = [ones(n, 1),
timeofdayy(:,1),timeofdayy(:,1).*timeofdayy(:,1),timeofdayy(:,1).*timeofday
y(:,1).*timeofdayy(:,1)];
Z = [ones(m1, 1),
timeofdayz(:,1),timeofdayz(:,1).*timeofdayz(:,1),timeofdayz(:,1).*timeofday
z(:,1).*timeofdayz(:,1)];
W = [ones(n1, 1),
timeofdayw(:,1),timeofdayw(:,1).*timeofdayw(:,1),timeofdayw(:,1).*timeofday
w(:,1).*timeofdayw(:,1)];

theta1 = zeros(size(X,1), 1); #initial value for theta
theta2 = zeros(size(Y,1), 1);
theta3 = zeros(size(Z,1), 1);
theta4 = zeros(size(W,1), 1);
#we can also try gradient descent
#iterations = 1500;
#alpha = 0.01;


#using normal equations

theta1=normalEqn(X,co);
fprintf("the value obtained to fit co2 concentrations");
theta1

theta2=normalEqn(Y,co2);
fprintf("the value obtained to fit co concentrations");
```

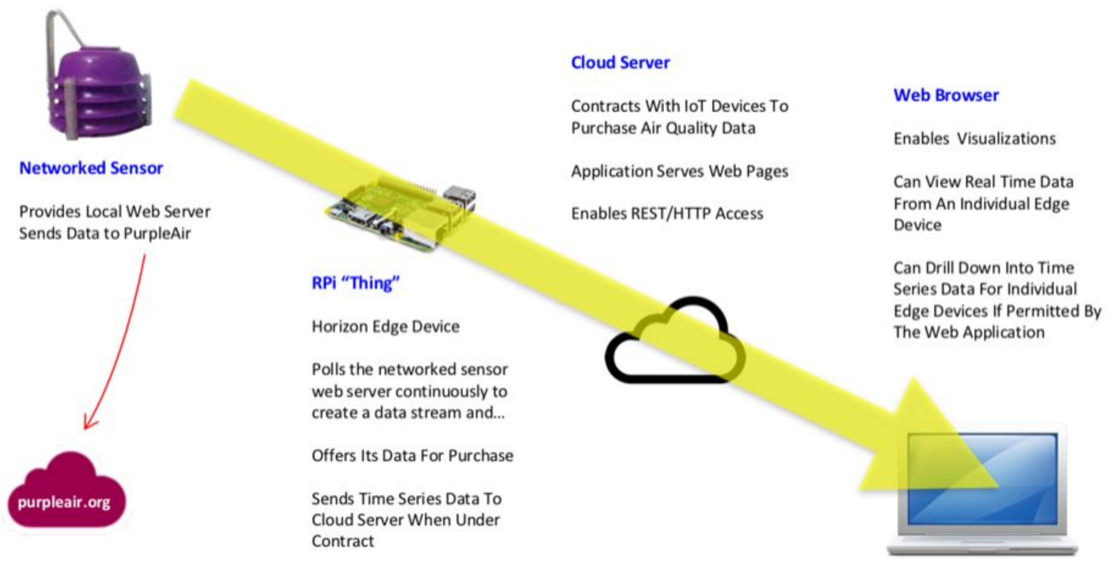
```
theta2

theta3=normalEqn(Z,ch4);
fprintf("the value obtained to fit co2 concentrations");
theta3

theta4=normalEqn(W,alcohol);
fprintf("the value obtained to fit co2 concentrations");
theta4

pause;
pause;
pause;
pause;


#plotting the linear fit to the graph


fprintf('1)concentrations predicted:BLUE\n2)permissible limit:YELLOW
\n3)danger limit:RED\n4)co2 concentrations measured:BLACK\n');

subplot(4,1,1);
plot(X(:,2), X*theta1, 'b-');
xlabel('time of the day in seconds');
ylabel('conc. co2 in ppm');
hold on; % keep previous plot not visible
X1=zeros(size(co));
X1=X1.+250;
X2=zeros(size(co));
X2=X2.+350;

plot(timeofdayx,X1,'y-');
plot(timeofdayx,X2,'r-');
plot(timeofdayx,co,'k-');




subplot(4,1,2);
plot(Y(:,2), Y*theta2, 'b-');
```

```
xlabel('time of the day in seconds');
ylabel('conc. of co in ppm');
hold on;
Y1=zeros(size(co2));
Y1=Y1.+60;
plot(timeofdayy,Y1,'y-');


Y2=zeros(size(co2));
Y2=Y2.+70;
plot(timeofdayy,Y2,'r-');
plot(timeofdayy,co2,'k-');


subplot(4,1,3);
plot(Z(:,2), Z*theta3, 'b-');
xlabel('time of the day in seconds');
ylabel('conc. of ch4 in ppm');
hold on; % keep previous plot not visible
Z1=zeros(size(ch4));
Z1=Z1.+350;
Z2=zeros(size(ch4));
Z2=Z2.+400;

plot(timeofdayz,Z1,'y-');
plot(timeofdayz,Z2,'r-');
plot(timeofdayz,ch4,'k-');


subplot(4,1,4);
plot(W(:,2), W*theta4, 'b-');
xlabel('time of the day in seconds');
ylabel('conc. of alcohol in ppm');
hold on; % keep previous plot not visible
W1=zeros(size(alcohol));
W1=W1.+80;
W2=zeros(size(alcohol));
W2=W2.+100;

plot(timeofdayw,W1,'y-');
plot(timeofdayw,W2,'r-');
plot(timeofdayw,alcohol,'k-');


#we can predict values now
tx=input("\n\nenter the time for which we need the concentrations of co2.....");
ty=input("\n\nenter the time for which we need the concentrations of co......");
tz=input("\n\nenter the time for which we need the concentrations of ch4.....");
tw=input("\n\nenter the time for which we need the concentrations of alcohol.....");
[cox,co2y,ch4z,alcoholw]=predictconc(tx,ty,tz,tw,theta1,theta2,theta3,theta4);
fprintf("the predicted values of co2 is.....");
cox
fprintf("the predicted values of co is.....");
co2y
fprintf("the predicted values of ch4 is.....");
ch4z
fprintf("the predicted values of alcohol is.....");
alcoholw


#plant prediction based on average value for the gases:
plantprediction();
```

Future Vision:

We intend to make a portable outdoor air quality monitor. Also add the data procured data into ‘Thingsspeak’ and make it available to all the clients through MQTT broker or any other means. Also make a another ML model to read and predict the reasons for the variation of air quality in various regions and also during night and day.


Vehicular Emission Monitoring
HIGH
Traffic Density
Residential Area Pollution
Construction Site Monitoring


Networked Sensor
Provides Local Web Server Sends Data to PurpleAir
purpleair.org
RPi "Thing"
Horizon Edge Device
Polls the networked sensor web server continuously to create a data stream and...
Offers Its Data For Purchase
Sends Time Series Data To Cloud Server When Under Contract
Cloud Server
Contracts With IoT Devices To Purchase Air Quality Data
Application Serves Web Pages
Enables REST/HTTP Access
Web Browser
Enables Visualizations
Can View Real Time Data From An Individual Edge Device
Can Drill Down Into Time Series Data For Individual Edge Devices If Permitted By The Web Application