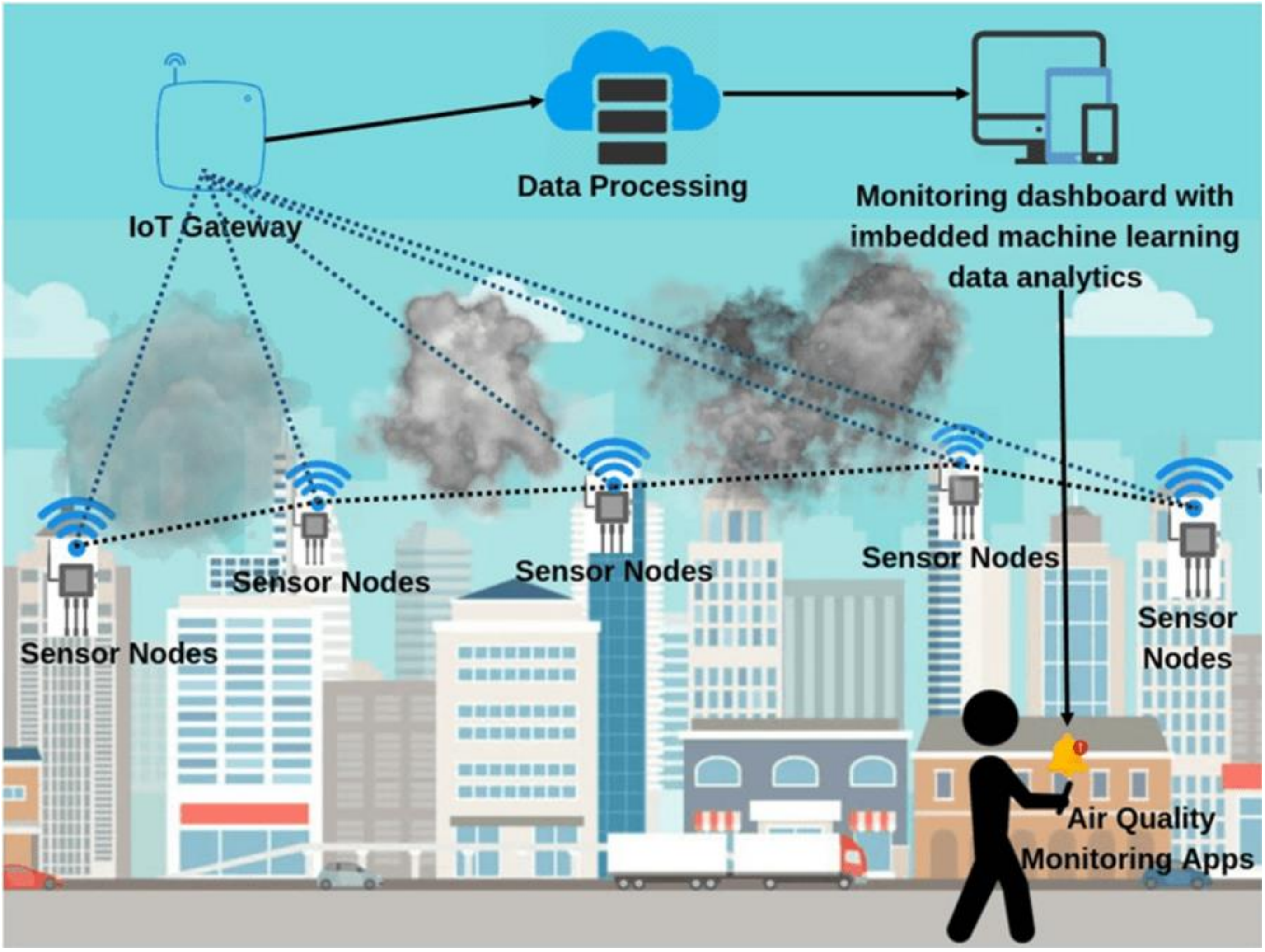
AQM – IOT

AIR QUALITY MONITORING

PHASE 5: PROJECT DOCUMENTATION & SUBMISSION:



PROJECT: AIR QUALITY MONITORING

INTRODUCTION:

Air is a basic requirement for the survival and development of all lives on Earth. It affects healthand influences the development of the economy. Today, due to the development of industrialization,the increase in the number of private cars, and the burning of fossil fuels, air quality is decreasing, withincreasingly serious air pollution. There are many pollutants in the atmosphere, such as SO_2, NO_2,CO_2, NO, CO, NO_x, PM2.5, and PM10. Internationally, a large number of scholars have conductedresearch on air pollution and air quality forecasts, concentrating on the forecasting of contaminants.Air pollution affects the life of a society, and even endangers the survival of mankind. During theIndustrial Revolution, there was a dramatic increase in coal use by factories and households, andthe smog caused significant morbidity and mortality, particularly when combined with stagnantatmospheric conditions. During the Great London Smog of 1952, heavy pollution for 5 days caused atleast 4000 deaths [1,2]. This episode highlighted the relationship between air pollution and humanhealth, yet air pollution continues to be a growing problem in cities and households around the world.

Air pollution is made up of a mixture of gases and particles in harmful amounts that are releasedinto the atmosphere due to either natural or human activities [3]. The sources of pollutants can bedivided into two categories:

1) Natural sources
2) Anthropogenic sources

(1) Natural sources:

Natural pollution sources are natural phenomena that discharge harmful substances or haveharmful effects on the environment. Natural phenomena, such as volcanic eruptions and forest fires,will result in air pollutants, including SO_2, CO_2, NO_2, CO, and sulfate.

(2) Anthropogenic (man-made) sources:

Man-made sources such as the burning of fuels, discharges from industrial production processes,and transportation emissions are the main sources of air pollution. There are many kinds of pollutantsemitted by man-made pollution sources, including hydrogen, oxygen,nitrogen, sulfur, metal compounds,and particulate matter.With the increasing world population and the developing world economy, the demand for energyin the world has increased dramatically. The large-scale use of fossil energy globally has also ledto a series of environmental problems that have received much attention due to their detrimentaleffects on human health and the environment [3–5]. Air pollution is a fundamental problem in manyparts of the world, with two important concerns: the impact on human health, such as cardiovasculardiseases, and the impact on the environment, such as acid rain, climate change, and global warming.

PROJECT DEFINITION & DESIGN THINKING:

PROJECT DEFENITION:

The project involves setting up IoT devices to measureair quality parameters and makethe data publicly available for raising awareness about air quality and its impact on public health. Theobjective is to create a platform that provides real-time air quality information to the public. Thisproject includes defining objectives,designing the IoT monitoring system, developing the data-sharingplatform,and integrating them using IoT technology and Python.

Design Thinking:

1(a).REAL-TIME AIR QUALITY MONITORING:

It refers to information about the current levels of pollutants in the air, such as particulate matter, ozone, nitrogen dioxide,sulfur dioxide, and carbon monoxide. Public can access theconstantlyupdated data.

1(b). DATA SHARING:

Data sharing is the process of making the same data resources available to multipleapplications, users, or organizations. It includes technologies, practices, legal frameworks, andcultural elements that facilitate secure data access for multiple entities withoutcompromising dataintegrity.

1(c). PUBLIC AWARENESS:

Communication about air quality has the potential toreduce the adverse effects of air pollution through generating awareness and catalyzing public opinion in support of policies for air pollution reduction and through education for individual risk mitigation behaviors; all are componentsof environmental health.

1(d). HEALTH IMPACT:

Exposure to air pollution can affect everyone's health. When we breathe in airpollutants, they can enter our bloodstream and contribute to coughing or itchy eyes and cause or worsen many breathing and lung diseases, leading to hospitalizations, cancer, or even premature death. 2.Design and deployment of IoT devices to measure air quality parameter:IOT Based Air Pollution Monitoring System monitors the Air quality over a web serverusing Internet and will trigger an alarm when the air quality goes down beyond a certain threshold level,means when there are sufficient amount of harmful gases present in the air like CO_2, smoke, alcohol,benzene, NH_3, LPG and NO_x.

Disadvantages:

As this devices are interconnected via internet there are possibilities that they can get hacked or monitored by malicious users or can be tracked by other systems as well. So the security ofthe recorded data can be an issue using this type of devices.

3. Design a web-based platform to display real-time air quality data:

An IoT-based air pollution monitoring system is an idealsolution that can provide real-time data and insights about the air quality in a particular area. An IoT based air pollution monitoringsystem consists of several hardware and software components that work together to collect and process data.

It refers to information about the current levels of pollutants in the air, such as particulate matter, ozone, nitrogen dioxide, sulfur dioxide, and carbon monoxide. Public can access theconstantly updated data.

4. Determine how IOT devices will send data to the data-sharing platform:

Once the network server has done its work, data is typically exchanged with a cloudapplication that will finish turning the IoT data into useful information, offer it to human users andstore it for subsequent analysis. Cloud applications often run alongside other network services onplatforms like AWS or Azure.

IoT devices use embedded sensors to collect, exchange, and share data with otherdevices, applications, and systems, in real-time. Internet of Things is a collaboration of custom-designedtechnologies to interconnect internet-enabled physical devices and enable communication with each other through a wireless network.

THE CONSTRUCTION OF THE PAPER

Air Pollution Assessment:

In recent years, air pollution accidents have occurred frequently, which have damaged the economy and human life. To assess the extent of the damage, air pollution control must be evaluatedin order to have a quantitative understanding of pollution.Int. J. Environ. Res. Public

Health **2018**, 15, 780 6 of 44The assessment of air pollution is identify and measure the degree and scope of damage causedby environmental pollution cover the economic, legal, technical and other means reasonably [35–37].

Two of the more mature assessment methods will be described. The market value methodis a type of cost benefit analysis method. It uses the change of product yield and profit causedby the environmental quality change to measure the economic loss related to the environmentalquality change.Environmental pollution and damage caused by air pollution can be prevented, restored, orreplaced by the existing environmental functions. Therefore, the cost of preventing, restoring, orreplacing the original functional protection facilities can be used to estimate the loss caused bypollution or damage to the environment. This method is called the engineering cost method.The main equation and the meaning of the variables in those methods are given in Table 1, andthe flowchart of the assessment methods is given in Figure 2.

Two of the more mature assessment methods will be described. The market value method is a typeof cost benefit analysis method. It uses the change of product yield and profit caused by theenvironmental quality change to measure the economic loss related to the environmental qualitychange.

Environmental pollution and damage caused by air pollution can be prevented, restored, orreplaced by the existing environmental functions. Therefore, the cost of preventing, restoring, orreplacing the original functional protection facilities can be used to estimate the loss caused bypollution or damage to the environment. This method is called the engineering cost method.

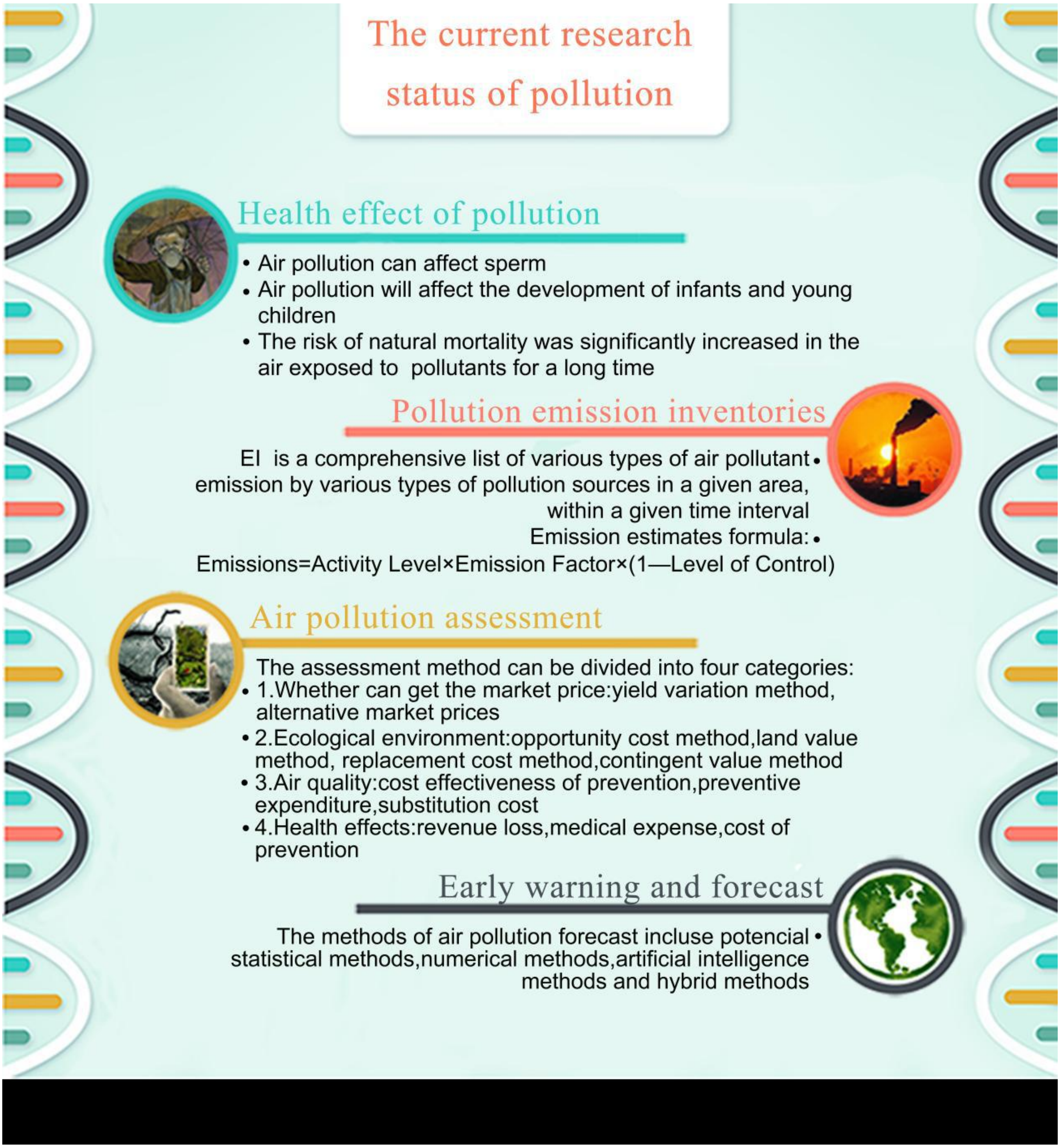
Air Pollution Early Warning and Forecast:

The most important function of air pollution early warning systems is to report the air quality torelevant departments when the air quality reaches the early warning standard. A complete pollution.

Warning system includes the pollutant, resource, and scope of influence [44]. Air quality forecasting is an effectiveway of protecting public health by providing an earlywarning against harmful air pollutants [9]. Urban air pollution events can be forecasted by meteorological elements to provide an early warning. Therefore, in the face of more and more urban air pollution incidents, in addition to risk prevention management and emergency measures, air pollution forecastsshould also include the emergency warnings as an important part of the whole emergency system.

The early warning system for air pollution is triggered before the heavy pollution of urbanair, according to the forecast of meteorological elements. Corresponding emergency measures areinitiated as early as possible to reduce the discharge of pollutants and mitigate the consequences.

Many countries have early warning systems for pollution. For example, the Air Quality Index (AQI)value is an index for the classification of the early warning level in China, and the early warning levelis determined according to the upper limit of the pollution forecast. Therefore, the forecasting of airpollution as the basis for pollution warning systems and pollution control should be highly valued by all countries.



THE CURRENT RESURCH STATUS OF POLLUTION

TYPES OF PREDICTIVE MODELING TO FORECAST AIR QUALITY TRENDS:

Linear regression modeling (LM),
Support vector regression (SVR),
Artificial neural network (ANN),
Decision tree (DT),
Random forest (RF),
Extreme gradient boosting tree (XGB)

Linear regression modeling (LM):

Air quality is predicted using the R squared value. R square determines the proportion of variance in the dependent variable of the system that can be explained by the independent variable. It is a statistical measure in a regression model. It is also called a coefficient of determination.

Support vector regression (SVR):

The is paper provides Support Vector Machine (SVM) model to forecast the quality of air with AQI for the upcoming 15 hours. The proposed model predicts the amount of pollutants such as PM 2.5, PM10, NO, NO2, NH3 and shows the better performance than linear prediction models when compared in terms of RMSE.

Artificial neural network (ANN):

Based on the analysis conducted, model with neural network structure 7-20-4 produces the best performance in the prediction of air quality compared to the first model based on the values of R and the prediction accuracy.

Decision tree (DT):

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

Random forest (RF):

In an urban sensing system, an algorithm (RAQ) based on a random forest concept is proposed to predict the urban area air quality through the use of historical air quality data, meteorology data, historical traffic and road status as well as POI distribution information

Extreme gradient boosting tree (XGB):

What is XGBoost? XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems.

PROBLEM:

IP [1] :

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
```

```
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as outp
ut when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the curre
nt session
```

IP [2] :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=
1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

IP [3] :

```
airqualitydatset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
```

OUTPUT:

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        29531 non-null  object
 1   Date        29531 non-null  object
 2   PM2.5       24933 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25346 non-null  float64
 7   NH3         19203 non-null  float64
 8   CO          27472 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     23908 non-null  float64
 12  Toluene     21490 non-null  float64
 13  Xylene      11422 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

IP [4] :

```
airqualitydatset["City"].unique()
```

OUTPUT:

```
array(['Ahmedabad', 'Aizawl', 'Amaravati', 'Amritsar', 'Bengaluru',
       'Bhopal', 'Brajrajnagar', 'Chandigarh', 'Chennai', 'Coimbatore',
       'Delhi', 'Ernakulam', 'Gurugram', 'Guwahati', 'Hyderabad',
       'Jaipur', 'Jorapokhar', 'Kochi', 'Kolkata', 'Lucknow', 'Mumbai',
       'Patna', 'Shillong', 'Talcher', 'Thiruvananthapuram',
       'Visakhapatnam'], dtype=object)
```

IP [5] :

```
airqualitydatset = airqualitydatset.dropna()
airqualitydatset  # Dropped all null values in the Dataset as we could not average.
```

OUTPUT:

	city	date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	BENENE	TOLUENE	AQI	AQI_Bucket
0	Ahamedabad	2015.01.01	Nan	NAn	12.5	18.22	17.5	NaN	0.92	27.64	133.36	0.00	0.02	NaN	NaN
1	Ahamedabad	2015.01.02	Nan	Nan	15.7	15.69	16.46	NaN	0.97	24.55	34.06	5.50	5.50	NaN	NaN
2	Ahamedabad	2015.01.03	Nan	Nan	17.40	0.92	29.70	NaN	17.40	29.07	30.70	16.40	16.40	NaN	NaN
3	Ahamedabad	2015.01.04	Nan	Nan	1.70	0.97	17.97	NaN	1.70	18.59	36.08	10.14	10.14	NaN	NaN
4	Ahamedabad	2015.01.05	Nan	Nan	22.10	21.42	37.76	NaN	22.10	39.31	39.31	18.89	18.89	NaN	NaN

IP [6] :

```
airqualitydatset.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6236 entries, 2123 to 29529
```

OUTPUT:

```
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
```

```
---  ------      --------------  -----
 0   City        6236 non-null   object
 1   Date        6236 non-null   object
 2   PM2.5       6236 non-null   float64
 3   PM10        6236 non-null   float64
 4   NO          6236 non-null   float64
 5   NO2         6236 non-null   float64
 6   NOx         6236 non-null   float64
 7   NH3         6236 non-null   float64
 8   CO          6236 non-null   float64
 9   SO2         6236 non-null   float64
 10  O3          6236 non-null   float64
 11  Benzene     6236 non-null   float64
 12  Toluene     6236 non-null   float64
 13  Xylene      6236 non-null   float64
 14  AQI         6236 non-null   float64
 15  AQI_Bucket  6236 non-null   object
dtypes: float64(13), object(3)
memory usage: 828.2+ KB
```

IP [7] :

```
airqualitydatset.mean()
```

```
/tmp/ipykernel_20/855534917.py:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  airqualitydatset.mean()
```

OUTPUT:

```
PM2.5       61.327365
PM10       123.418321
NO          17.015191
NO2         31.708190
NOx         32.448956
NH3         20.737070
CO           0.984344
SO2         11.514426
O3          36.127691
Benzene      3.700361
Toluene     10.323696
Xylene       2.557439
AQI        140.510103
dtype: float64
```

IP [8] :

```
airqualitydatset.max()
```

OUTPUT:

```
City          Visakhapatnam
```

```
Date            2020-07-01
PM2.5               639.19
PM10                796.88
NO                  159.22
NO2                 140.17
NOx                 224.09
NH3                  166.7
CO                   16.23
SO2                  70.39
O3                  162.33
Benzene              64.44
Toluene              103.0
Xylene              125.18
AQI                  677.0
AQI_Bucket       Very Poor
dtype: object
```

IP [9] :

```
airqualitydatset.min()
```

OUTPUT:

```
City         Amaravati
Date        2015-01-01
PM2.5              2.0
PM10               7.8
NO                0.25
NO2               0.17
NOx               0.17
NH3               0.12
CO                 0.0
SO2               0.71
O3                1.55
Benzene            0.0
Toluene            0.0
Xylene             0.0
AQI               23.0
AQI_Bucket        Good
dtype: object
```

IP [10] :

```
#mostreadingAQI.value_counts()
#mostreadingAQI = mostreadingAQI[['City', 'AQI_Bucket']]
#mostreadingAQIGood = mostreadingAQI.where(mostreadingAQI["City"] == "Amaravati")
#mostreadingAQI.dropna()
#mostreadingAQI = mostreadingAQI.where(mostreadingAQI["AQI_Bucket"] == "Good")

#mostreadingAQI
#mostreadingAQI["AQI_Bucket"].value_counts()
```

IP [11] :

```
mostreadingAQI = airqualitydatset[["City","AQI_Bucket"]]
mostreadingAQI.sort_values(['City'],inplace=True,ascending=True)
mostreadingAQI.groupby(['AQI_Bucket'])
mostreadingAQI
#mostreadingAQI.sort_values(['City'],inplace=True,ascending=True)
mostreadingAQI.value_counts(['City', 'AQI_Bucket'])
```

```
/tmp/ipykernel_20/565188966.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
ndexing.html#returning-a-view-versus-a-copy
  mostreadingAQI.sort_values(['City'],inplace=True,ascending=True)
```

OUTPUT:

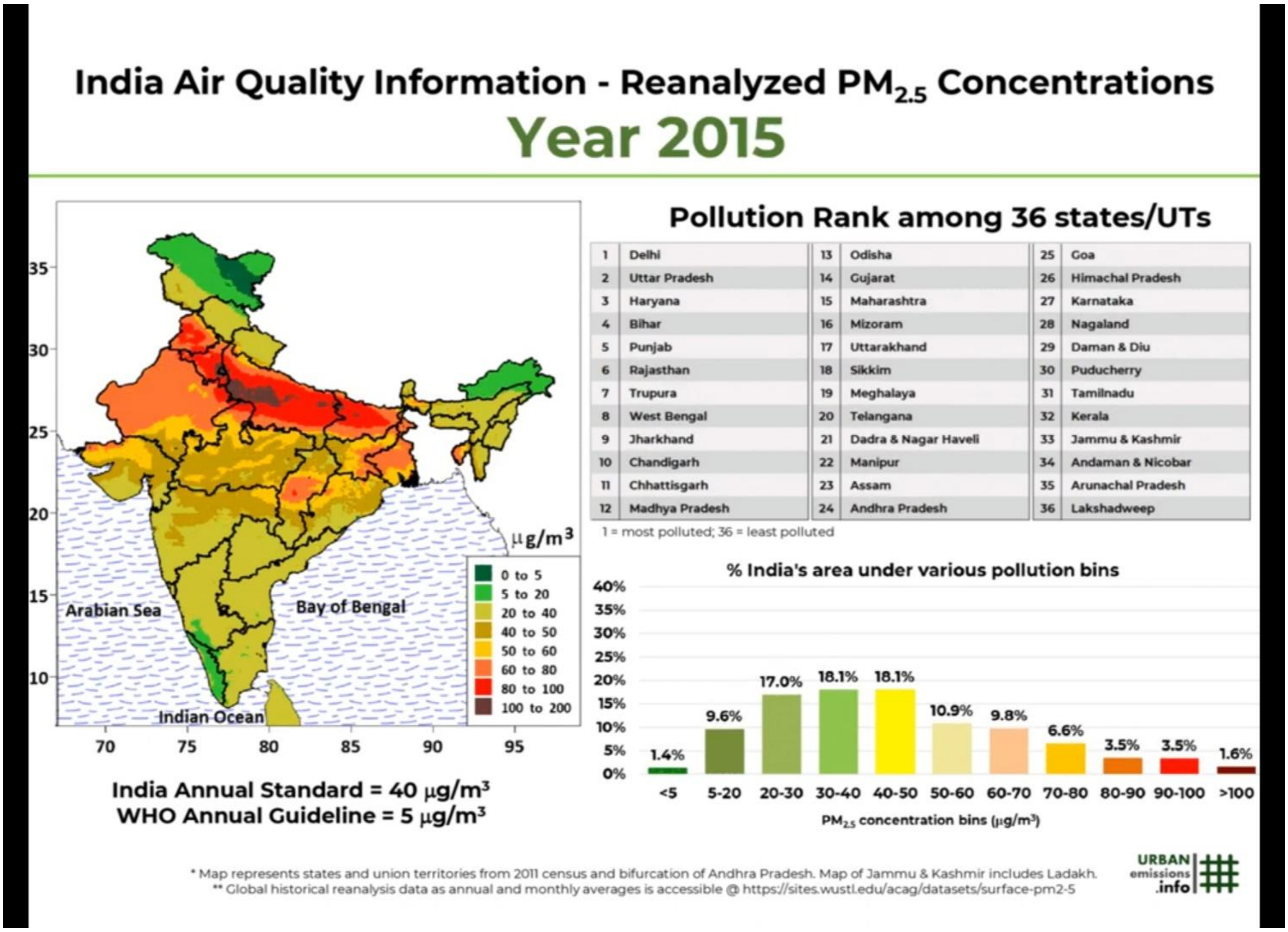
```
City             AQI_ Bucket
Hyderabad  Moderate    810
           Satisfactory   645
Visakhapatnam  Moderat  555
           Satisfactory    438
Delhi        Moderate    360
           Poor           328
           Very Poor      315
Amaravati     Satisfactory    305
Amritsar      Moderate       252
           Satisfactory    252
Amaravati     Moderate       191
Chandigarh     Satisfactory    145
Kolkata       Satisfactory    139
Hyderabad      Good          126
Delhi         Severe         117
           Satisfactory    104
Patna         Moderate       103
Amaravati     Good           101
Gurugram      Moderate        97
Kolkata       Good            95
           Moderate        87
           Poor            71
Visakhapatnam  Poor           70
Chandigarh     Moderate        66
Visakhapatnam  Good           50
Amritsar      Poor           49
Chandigarh     Good           48
Amritsar      Very Poor       47
Patna         Poor           42
Amaravati     Poor           41
Amritsar      Good           34
Patna         Satisfactory     31
Hyderabad      Poor           30
Gurugram      Satisfactory     20
Visakhapatnam  Very Poor       18
Chandigarh     Poor           15
Patna         Very Poor       14
```

```
Amaravati      Very Poor        8
Hyderabad      Severe           4
               Very Poor        3
Chandigarh     Very Poor        3
Kolkata        Very Poor        2
Amritsar       Severe           2
Gurugram       Poor             2
Patna          Severe           1
dtype: int64
```

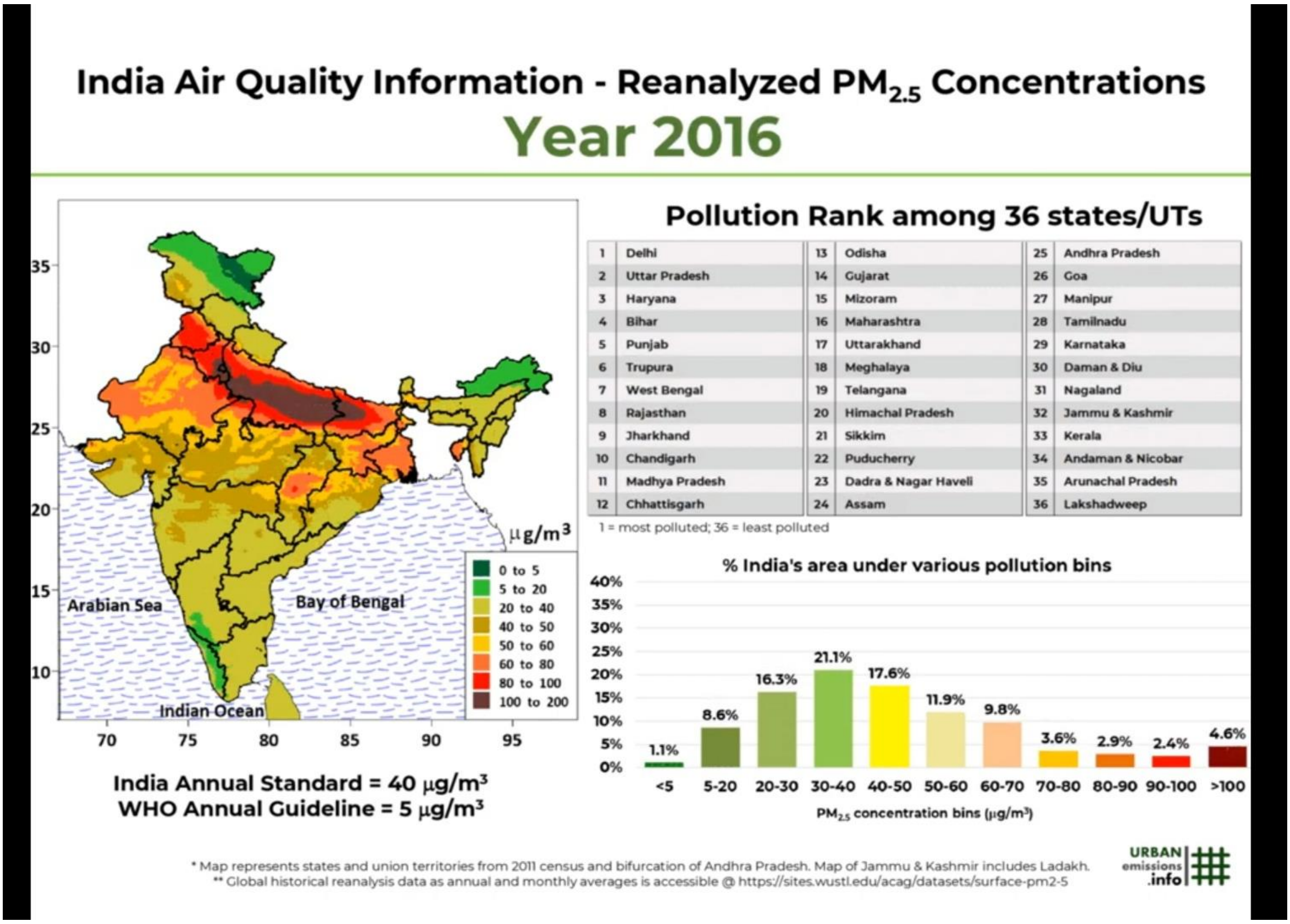
INDIA AIR QUALITY INFORMATION REANALYZED PM2.5 CONCENTRATIONS:

YEAR – 2015 TO 2020

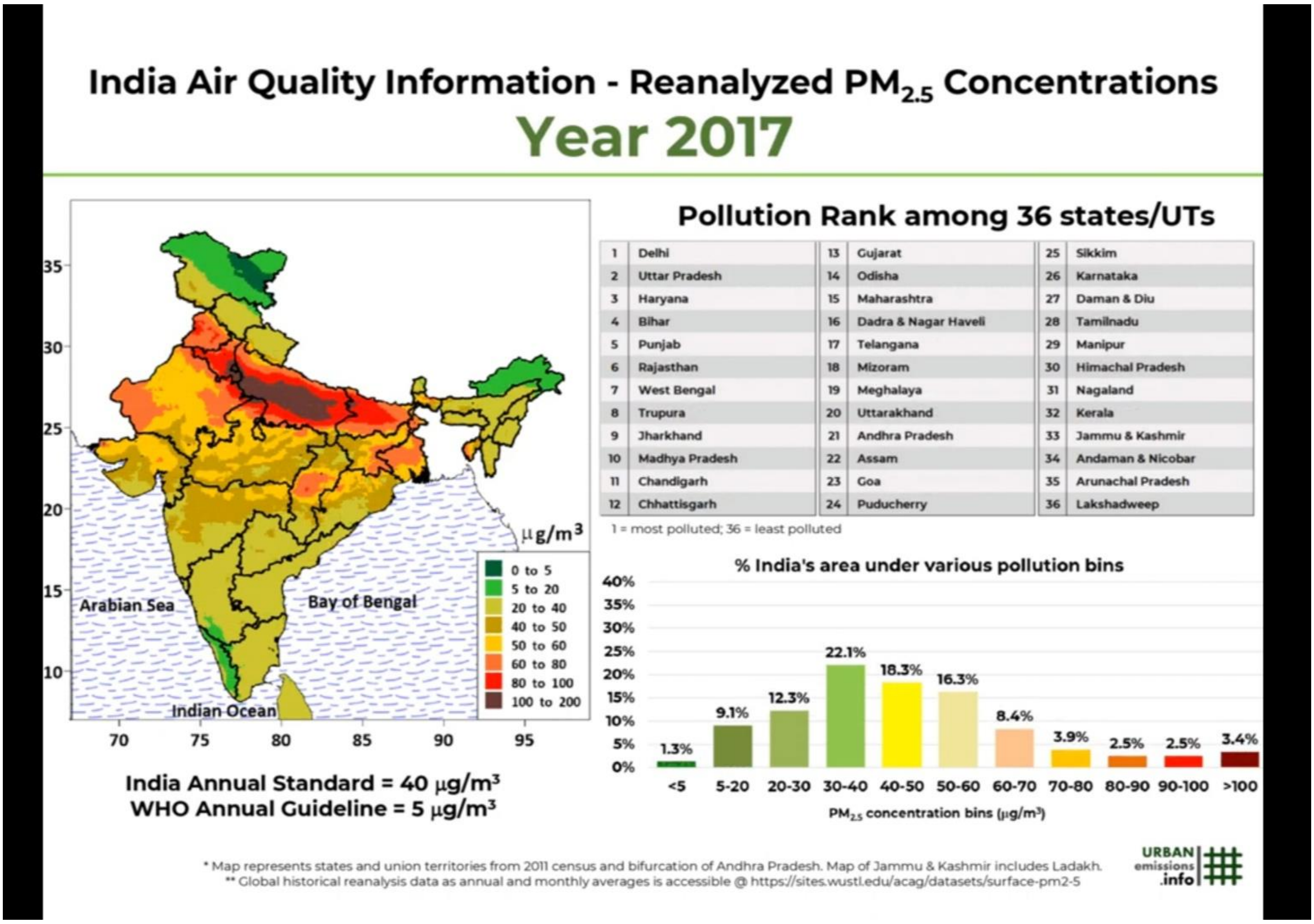
YEAR - 2015



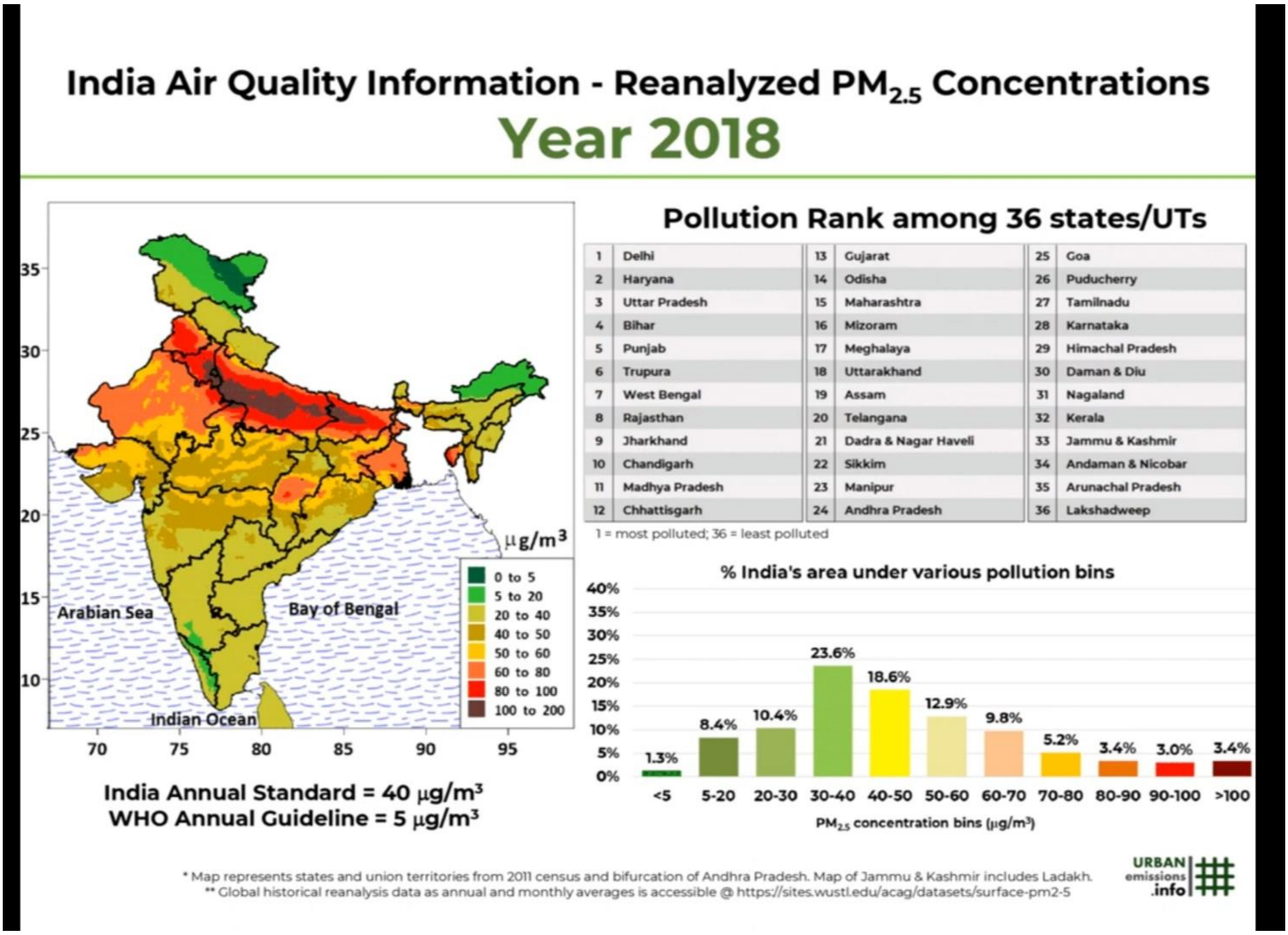
YEAR – 2016



YEAR – 2017


India Air Quality Information - Reanalyzed PM2.5 Concentrations
Year 2017
Pollution Rank among 36 states/UTs
1 Delhi
2 Uttar Pradesh
3 Haryana
4 Bihar
5 Punjab
6 Rajasthan
7 West Bengal
8 Trupura
9 Jharkhand
10 Madhya Pradesh
11 Chandigarh
12 Chhattisgarh
13 Gujarat
14 Odisha
15 Maharashtra
16 Dadra & Nagar Haveli
17 Telangana
18 Mizoram
19 Meghalaya
20 Uttarakhand
21 Andhra Pradesh
22 Assam
23 Goa
24 Puducherry
25 Sikkim
26 Karnataka
27 Daman & Diu
28 Tamilnadu
29 Manipur
30 Himachal Pradesh
31 Nagaland
32 Kerala
33 Jammu & Kashmir
34 Andaman & Nicobar
35 Arunachal Pradesh
36 Lakshadweep
1 = most polluted; 36 = least polluted
Arabian Sea
Bay of Bengal
Indian Ocean
µg/m3
0 to 5
5 to 20
20 to 40
40 to 50
50 to 60
60 to 80
80 to 100
100 to 200
35
30
25
20
15
10
70
75
80
85
90
95
India Annual Standard = 40 µg/m3
WHO Annual Guideline = 5 µg/m3
% India's area under various pollution bins
40%
35%
30%
25%
20%
15%
10%
5%
0%
1.3%
9.1%
12.3%
22.1%
18.3%
16.3%
8.4%
3.9%
2.5%
2.5%
3.4%
<5
5-20
20-30
30-40
40-50
50-60
60-70
70-80
80-90
90-100
>100
PM2.5 concentration bins (µg/m3)
* Map represents states and union territories from 2011 census and bifurcation of Andhra Pradesh. Map of Jammu & Kashmir includes Ladakh.
** Global historical reanalysis data as annual and monthly averages is accessible @ https://sites.wustl.edu/acag/datasets/surface-pm2-5
URBAN emissions .info

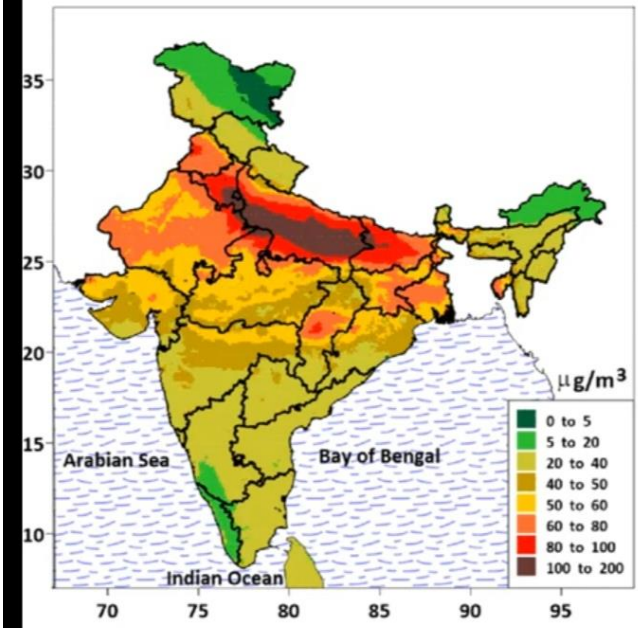
YEAR – 2018


India Air Quality Information - Reanalyzed $PM_{2.5}$ Concentrations
Year 2018
Pollution Rank among 36 states/UTs
1 Delhi
2 Haryana
3 Uttar Pradesh
4 Bihar
5 Punjab
6 Trupura
7 West Bengal
8 Rajasthan
9 Jharkhand
10 Chandigarh
11 Madhya Pradesh
12 Chhattisgarh
13 Gujarat
14 Odisha
15 Maharashtra
16 Mizoram
17 Meghalaya
18 Uttarakhand
19 Assam
20 Telangana
21 Dadra & Nagar Haveli
22 Sikkim
23 Manipur
24 Andhra Pradesh
25 Goa
26 Puducherry
27 Tamilnadu
28 Karnataka
29 Himachal Pradesh
30 Daman & Diu
31 Nagaland
32 Kerala
33 Jammu & Kashmir
34 Andaman & Nicobar
35 Arunachal Pradesh
36 Lakshadweep
1 = most polluted; 36 = least polluted
Arabian Sea
Bay of Bengal
Indian Ocean
µg/m³
0 to 5
5 to 20
20 to 40
40 to 50
50 to 60
60 to 80
80 to 100
100 to 200
35
30
25
20
15
10
70
75
80
85
90
95
% India's area under various pollution bins
40%
35%
30%
25%
20%
15%
10%
5%
0%
1.3%
8.4%
10.4%
23.6%
18.6%
12.9%
9.8%
5.2%
3.4%
3.0%
3.4%
<5
5-20
20-30
30-40
40-50
50-60
60-70
70-80
80-90
90-100
>100
$PM_{2.5}$ concentration bins (µg/m³)
India Annual Standard = 40 µg/m³
WHO Annual Guideline = 5 µg/m³
* Map represents states and union territories from 2011 census and bifurcation of Andhra Pradesh. Map of Jammu & Kashmir includes Ladakh.
** Global historical reanalysis data as annual and monthly averages is accessible @ https://sites.wustl.edu/acag/datasets/surface-pm2-5
URBAN emissions .info

YEAR – 2019

India Air Quality Information - Reanalyzed $PM_{2.5}$ Concentrations

Year 2019

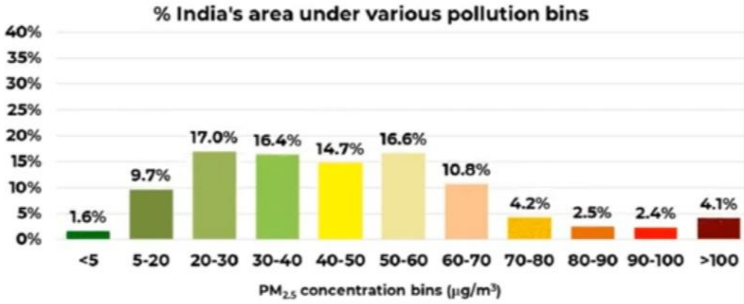


India Annual Standard = 40 µg/m³
WHO Annual Guideline = 5 µg/m³

Pollution Rank among 36 states/UTs

1	Delhi	13	Gujarat	25	Daman & Diu
2	Uttar Pradesh	14	Odisha	26	Puducherry
3	Haryana	15	Maharashtra	27	Goa
4	Bihar	16	Meghalaya	28	Nagaland
5	Punjab	17	Mizoram	29	Karnataka
6	Rajasthan	18	Telangana	30	Tamilnadu
7	Trupura	19	Dadra & Nagar Haveli	31	Himachal Pradesh
8	West Bengal	20	Assam	32	Andaman & Nicobar
9	Jharkhand	21	Uttarakhand	33	Kerala
10	Madhya Pradesh	22	Manipur	34	Jammu & Kashmir
11	Chhattisgarh	23	Andhra Pradesh	35	Arunachal Pradesh
12	Chandigarh	24	Sikkim	36	Lakshadweep

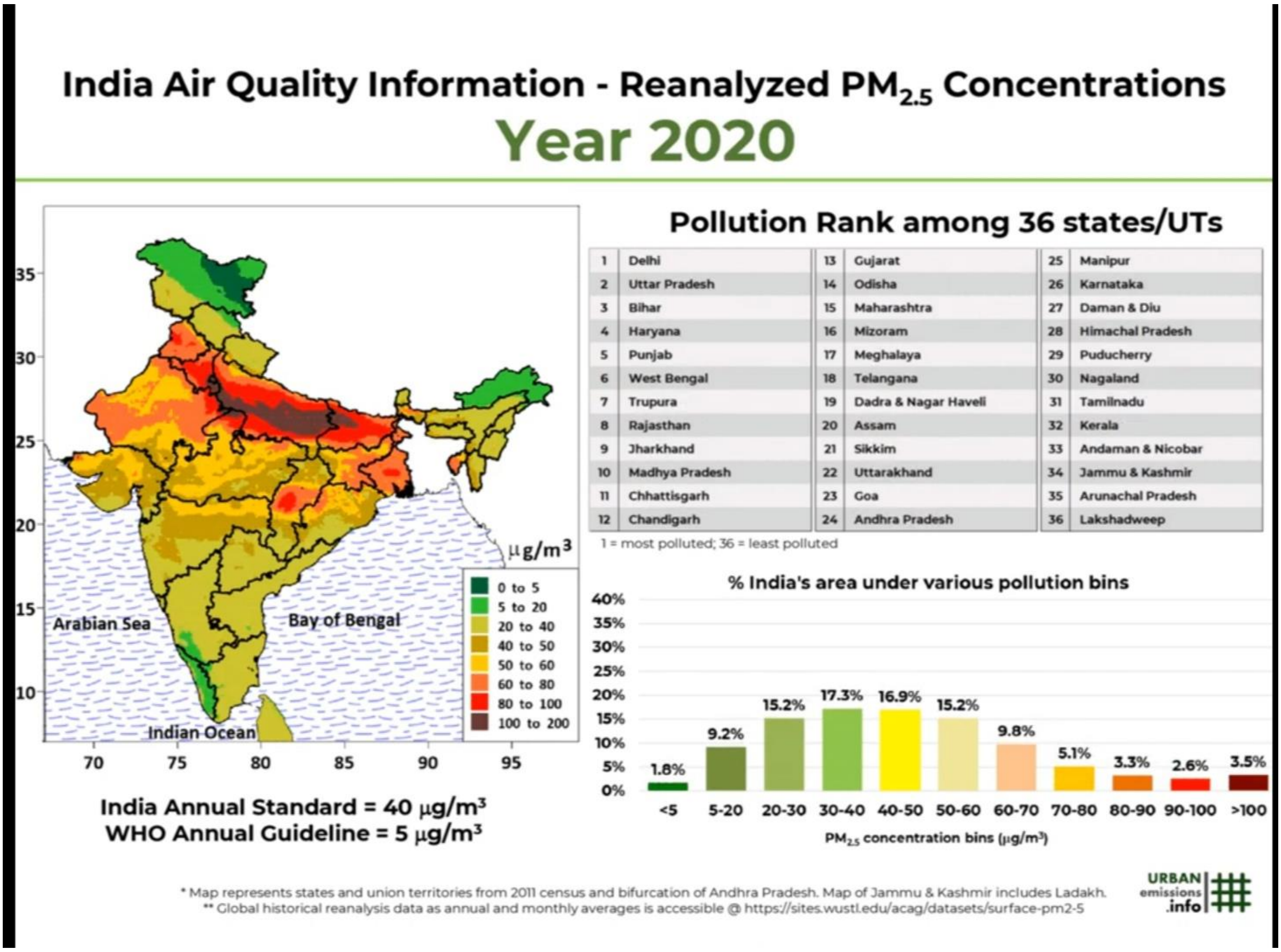
1 = most polluted; 36 = least polluted



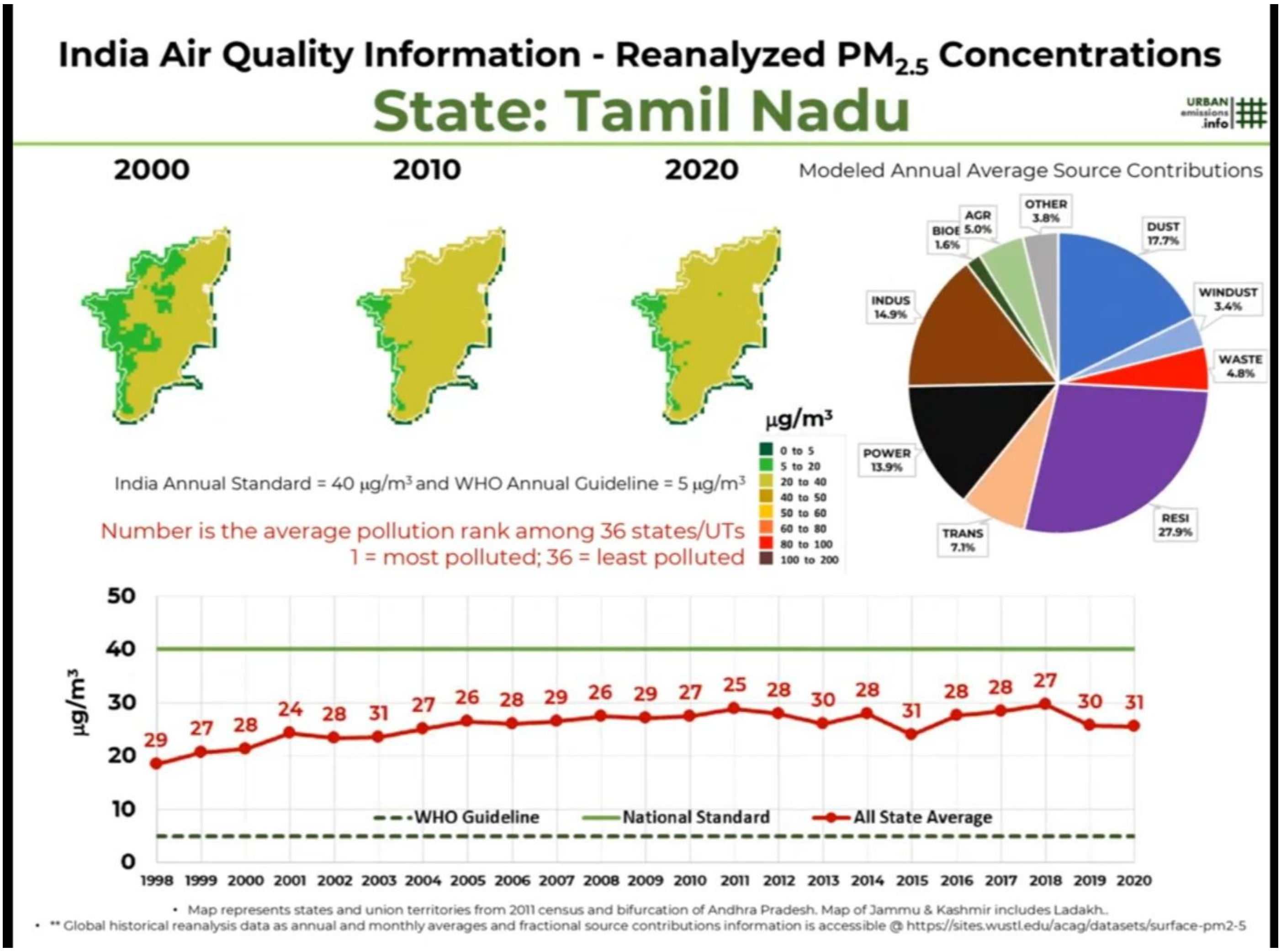
* Map represents states and union territories from 2011 census and bifurcation of Andhra Pradesh. Map of Jammu & Kashmir includes Ladakh.
** Global historical reanalysis data as annual and monthly averages is accessible @ https://sites.wustl.edu/acag/datasets/surface-pm2-5

URBAN emissions .info

YEAR – 2020

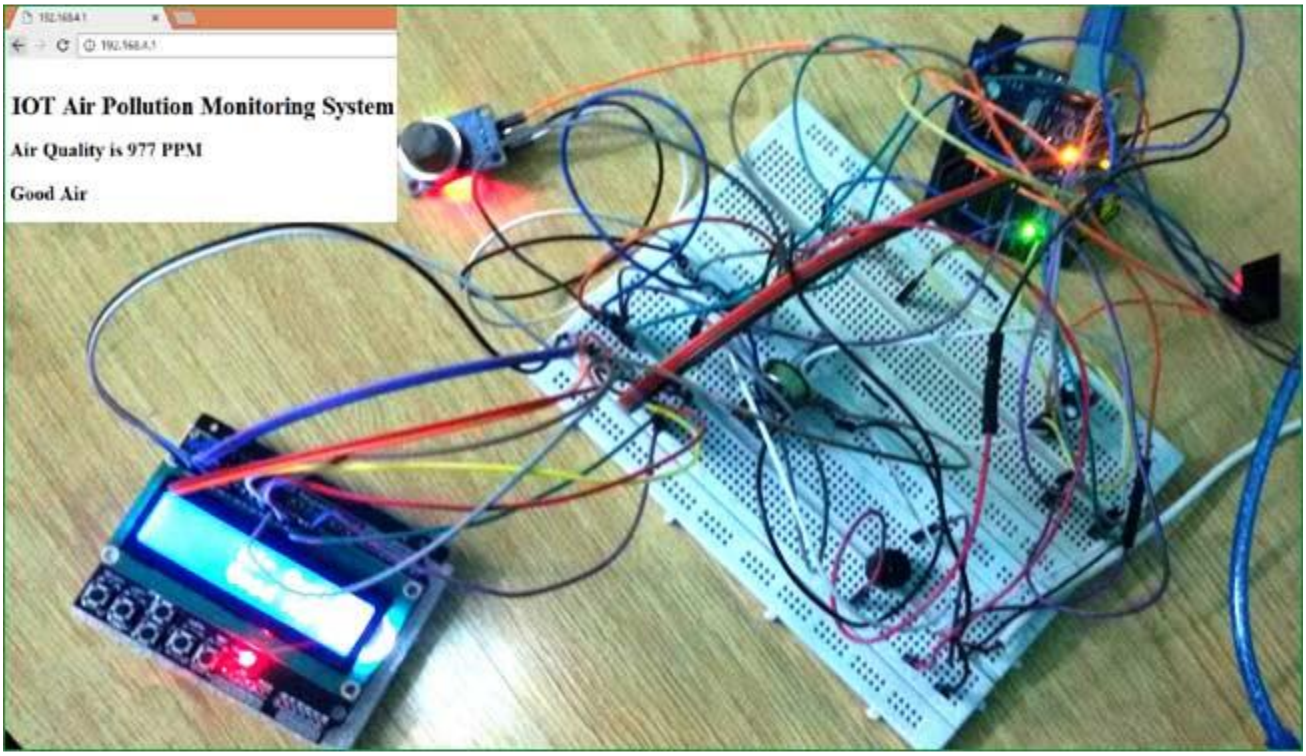


TAMIL NADU AIR QUALITY INFORMATION – REANALYZED PM2.5 CONCENTRATION:



START BUILDING THE IOT AIR QUALITY MONITORING SYSTEM.

IOT based Air Pollution Monitoring System using Arduino:



In this project we are going to make an **IoT Based Air Pollution Monitoring System** in which we will **monitor the Air Quality over a webserver using internet** and will trigger a alarm when the air quality goes down beyond a certain level, means when there are sufficient amount of harmful gases are present in the air like CO2, smoke, alcohol, benzene and NH3. It will show the air quality in PPM on the LCD and as well as on webpage so that we can monitor it very easily.

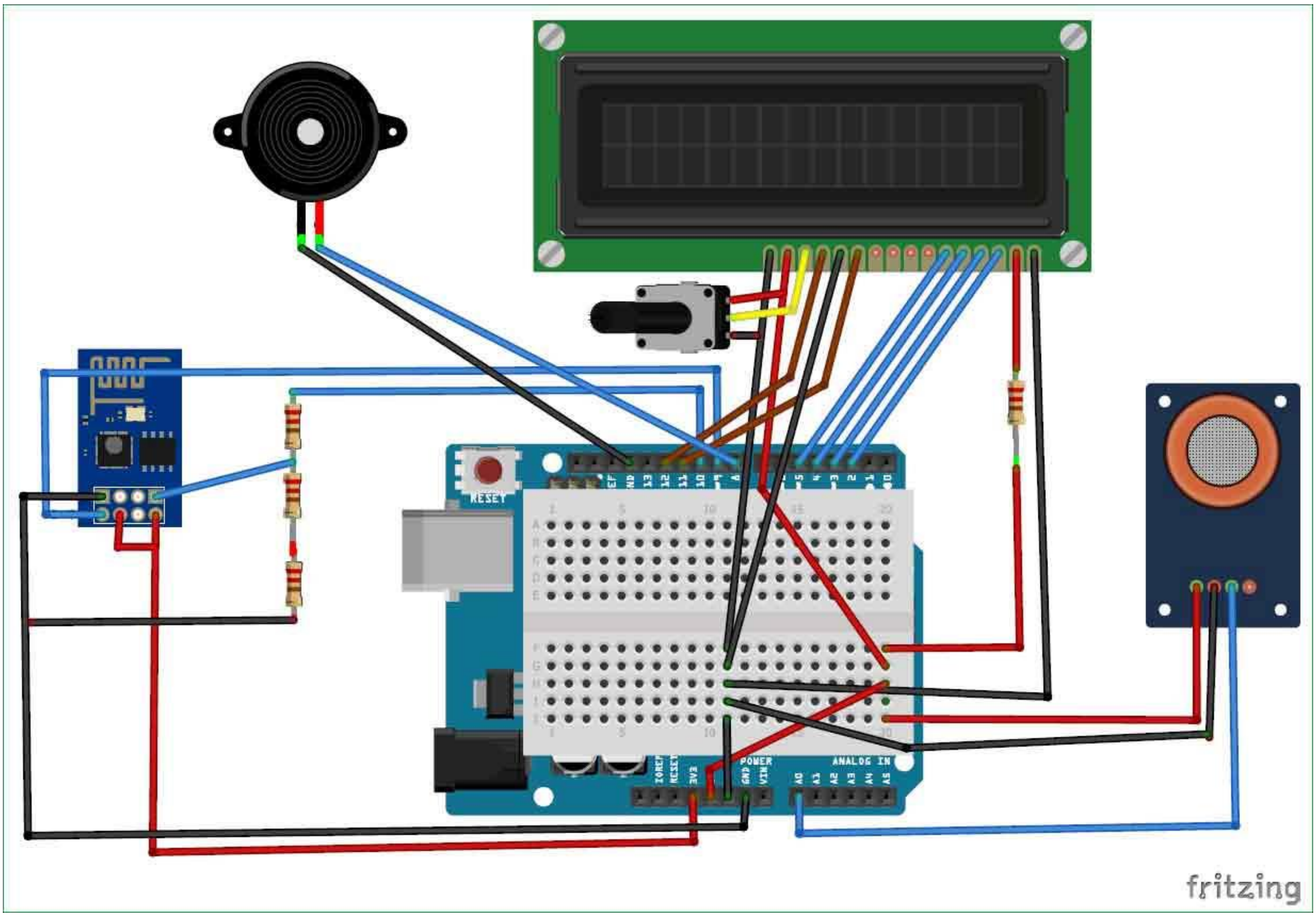
Previously we have built the LPG detector using MQ6 sensor, Smoke detector using MQ2 sensor, and Air Quality Analyser but this time we have used MQ135 sensor as the air quality sensor which is the best choice for monitoring Air Quality as it can detects most harmful gases and can measure their amount accurately. In this IOT project, you can monitor the pollution level from anywhere using your computer or mobile. We can install this system anywhere and can also trigger some device when pollution goes beyond some level, like we can switch on the Exhaust fan or can send alert SMS/mail to the user.

Required Components:

- MQ135 Gas sensor
- Arduino Uno
- Wi-Fi module ESP8266
- 16X2 LCD
- Breadboard
- 10K potentiometer
- 1K ohm resistors
- 220 ohm resistor
- Buzzer

You can buy all the above components from here.

CIRCUIT DIAGRAM:



Circuit Diagram and Explanation:

First of all we will connect the **ESP8266 with the Arduino**. ESP8266 runs on 3.3V and if you will give it 5V from the Arduino then it won't work properly and it may get damage. Connect the VCC and the CH_PD to the 3.3V pin of Arduino. The RX pin of ESP8266 works on 3.3V and it will not communicate with the Arduino when we will connect it directly to the Arduino. So, we will have to make a voltage divider for it which will convert the 5V into 3.3V. This can be done by connecting three resistors in series like we did in the circuit. Connect the TX pin of the ESP8266 to the pin 10 of the Arduino and the RX pin of the esp8266 to the pin 9 of Arduino through the resistors.

ESP8266 Wi-Fi module gives your projects **access to Wi-Fi or internet**. It is a very cheap device and make your projects very powerful. It can communicate with any microcontroller and it is the most leading devices in the IOT platform. Learn more about using ESP8266 with Arduino here.

Then we will connect the **MQ135 sensor with the Arduino**. Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino and the Analog pin of sensor to the A0 of the Arduino.

Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true.

In last, we will connect LCD with the Arduino. The connections of the LCD are as follows

- Connect pin 1 (VEE) to the ground.

- Connect pin 2 (VDD or VCC) to the 5V.
- Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND. The potentiometer is used to control the screen contrast of the LCD. Potentiometer of values other than 10K will work too.
- Connect pin 4 (RS) to the pin 12 of the Arduino.
- Connect pin 5 (Read/Write) to the ground of Arduino. This pin is not often used so we will connect it to the ground.
- Connect pin 6 (E) to the pin 11 of the Arduino. The RS and E pin are the control pins which are used to send data and characters.
- The following four pins are data pins which are used to communicate with the Arduino.

Connect pin 11 (D4) to pin 5 of Arduino.

Connect pin 12 (D5) to pin 4 of Arduino.

Connect pin 13 (D6) to pin 3 of Arduino.

Connect pin 14 (D7) to pin 2 of Arduino.

- Connect pin 15 to the VCC through the 220 ohm resistor. The resistor will be used to set the back light brightness. Larger values will make the back light much more darker.
- Connect pin 16 to the Ground.

Working Explanation:

The MQ135 sensor can sense NH3, NOx, alcohol, Benzene, smoke, CO2 and some other gases, so it is perfect gas sensor for our **Air Quality Monitoring Project**. When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million). MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM. So for converting the output in PPM, here we have used a library for MQ135 sensor, it is explained in detail in “Code Explanation” section below.

Sensor was giving us value of 90 when there was no gas near it and the safe level of air quality is 350 PPM and it should not exceed 1000 PPM. When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases.

When the value will be less than 1000 PPM, then the LCD and webpage will display “Fresh Air”. Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display “Poor Air, Open Windows”. If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display “Danger! Move to fresh Air”.

Air Pollution Assessment:

In recent years, air pollution accidents have occurred frequently, which have damaged the economy and human life. To assess the extent of the damage, air pollution control must be evaluatedin order to have a quantitative understanding of pollution.Int. J. Environ. Res. Public Health **2018**, 15, 780 6 of 44The assessment of air pollution is identify and measure the degree and scope of damage causedby environmental pollution cover the economic, legal, technical and other means reasonably [35–37].

Two of the more mature assessment methods will be described. The market value methodis a type of cost benefit analysis method. It uses the change of product yield and profit causedby the environmental quality change to measure the economic loss related to the environmentalquality change.Environmental pollution and damage caused by air pollution can be prevented, restored, orreplaced by the existing environmental functions. Therefore, the cost of preventing, restoring, orreplacing the original functional protection facilities can be used to estimate the loss caused bypollution or damage to the environment. This method is called the engineering cost method.The main equation and the meaning of the variables in those methods are given in Table 1, andthe flowchart of the assessment methods is given in Figure 2.

Two of the more mature assessment methods will be described. The market value method is a typeof cost benefit analysis method. It uses the change of product yield and profit caused by theenvironmental quality change to measure the economic loss related to the environmental qualitychange.

Environmental pollution and damage caused by air pollution can be prevented, restored, orreplaced by the existing environmental functions. Therefore, the cost of preventing, restoring, orreplacing the original functional protection facilities can be used to estimate the loss caused bypollution or damage to the environment. This method is called the engineering cost method.

CODING:

```
webpage += "</h2></p></body>";

    String cipSend = "AT+CIPSEND=";

    cipSend += connectionId;

    cipSend += ",";

    cipSend +=webpage.length();

    cipSend +="\r\n";



    sendData(cipSend,1000,DEBUG);

    sendData(webpage,1000,DEBUG);



    cipSend = "AT+CIPSEND=";

    cipSend += connectionId;

    cipSend += ",";

    cipSend +=webpage.length();
```

```
     cipSend +="\r\n";


     String closeCommand = "AT+CIPCLOSE=";

     closeCommand+=connectionId; // append connection id

     closeCommand+="\r\n";


     sendData(closeCommand,3000,DEBUG);

    }

  }

lcd.setCursor (0, 0);

lcd.print ("Air Quality is ");

lcd.print (air_quality);

lcd.print (" PPM ");

lcd.setCursor (0,1);

if (air_quality<=1000)

{

lcd.print("Fresh Air");

digitalWrite(8, LOW);

}

else if( air_quality>=1000 && air_quality<=2000 )

{

lcd.print("Poor Air, Open Windows");

digitalWrite(8, HIGH );

}

else if (air_quality>=2000 )

{

lcd.print("Danger! Move to Fresh Air");
```

```
digitalWrite(8, HIGH);   // turn the LED on

}

lcd.scrollDisplayLeft();

delay(1000);

}

String sendData(String command, const int timeout, boolean debug)

{

    String response = "";

    esp8266.print(command); // send the read character to the esp8266

    long int time = millis();

    while( (time+timeout) > millis())

    {

      while(esp8266.available())

      {

        // The esp has data so display its output to the serial window

        char c = esp8266.read(); // read the next character.

        response+=c;

      }

    }

    if(debug)

    {

      Serial.print(response);

    }

    return response;

}
```

DATASET:

The dataset contains air quality data and AQI (Air Quality Index) at hourly and daily level of various stations across multiple cities in India. This project deals with exploration of the data and modelling to predict the classes namely : Good, Satisfactory, moderate, poor, very poor or severe cases of the air quality.

Import and load the data:

Import necessary packages: Here I have imported packages needed for preprocessing and modelling

```
import pandas as pd
import os
import zipfile
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
import numpy as np
from plotly.offline import init_notebook_mode,iplot
import missingno as msno
from sklearn.impute import KNNImputer
from sklearn import preprocessing
import pylab
import scipy.stats as stats
from scipy.special import boxcox1p
import pylab
import scipy.stats as stats
%matplotlib inline
# Transformation and modelling packages
from sklearn.model_selection import train_test_split , KFold
from sklearn.metrics import accuracy_score
from sklearn. preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from collections import Counter
from sklearn.model_selection import GridSearchCV ,RandomizedSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
```

The dataset upon loading and observation, it's important and easy to process the data with smaller size.

- The data types are changed while loading.
- String data is converted into categorical values, float64 to float32 etc. * *
- This process place vital role while handling dataset with larger size.

```
'City':'category',
    'Datetime' : 'category',
    'PM2.5' : 'float32',
    'PM10' : 'float32',
    'NO' : dtypes1 = {
    'float32',
    'NO2' : 'float32',
    'NOx' : 'float32',
    'NH3' : 'float32',
    'CO' : 'float32',
    'SO2' : 'float32',
    'O3' : 'float32',
    'Benzene' : 'float32',
    'Toluene' : 'float32',
    'Xylene' : 'float32',
    'AQI' : 'float32',
    'AQI_Bucket': 'category'
}
```

Loop through the zip files and extract files.

```
path = '/content/drive/MyDrive/Projects/air-quality-data-in-india.zip'

with zipfile.ZipFile(path, 'r') as f:
  for name in f.namelist():
    if name=='city_hour.csv':
      df_c = pd.read_csv(f.open(name), dtype=dtypes1)
```

Save the files for in csv format for direct access. Load the csv files using pandas 'read_csv'

```
# Save the dtyp changed csv files
fp1 = '/content/drive/MyDrive/Projects/city_hour.csv'
df_c.to_csv(fp1, index=False)
df_c = pd.read_csv('/content/drive/MyDrive/Projects/city_hour.csv',dtype = dtypes1)
```

The function ‘basic_exploration’ explorates the routine exploration done on any tabular data. Functionizing makes the task easier and saves a lot of time.

```
def space():
  print(" ")
  print("------------------------------------------------------------------")
  print(" ")

def basic_exploration(steps):
  for i in steps:
    print(i)
    space()

steps = [df.shape, df.duplicated().any(), df.isnull().sum()]
basic_exploration(steps)

(2589083, 16)

------------------------------------------------------------------

False

------------------------------------------------------------------

StationId          0
Datetime           0
PM2.5         647689
PM10         1119252
NO            553711
NO2           528973
NOx           490808
NH3          1236618
CO            499302
SO2           742737
O3            725973
Benzene       861579
Toluene      1042366
Xylene       2075104
AQI           570190
AQI_Bucket    570190
dtype: int64
```

```
------------------------------------------------------------
```

The dataset has numerous missing values.

```
df_c.head(3)
```

	City	Datetime	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3
0	Ahmedabad	2015-01-01 01:00:00	NaN	NaN	1.00	40.009998	36.369999	NaN	1.00	122.070000	NaN
1	Ahmedabad	2015-01-01 02:00:00	NaN	NaN	0.02	27.750000	19.730000	NaN	0.02	85.900002	NaN
2	Ahmedabad	2015-01-01 03:00:00	NaN	NaN	0.08	19.320000	11.080000	NaN	0.08	52.830002	NaN

Data Preprocessing

Convert to datetime

```
df_c['Datetime'] = pd.to_datetime(df_c['Datetime'])
df_c['AQI_Bucket'].unique()
```

```
[NaN, 'Poor', 'Moderate', 'Very Poor', 'Severe', 'Satisfactory', 'Good']
Categories (6, object): ['Good', 'Moderate', 'Poor', 'Satisfactory', 'Severe', 'Very Poor']
```

Label encode the ‘AQI_Bucket’ which is the target column for our classification model. Label Encoder assign integers to the classes.

```
le = preprocessing.LabelEncoder()
df_c['AQI_Bucket'] = le.fit_transform(df_c['AQI_Bucket'])
```

[redacted]

```
(array([6, 2, 1, 5, 4, 3, 0]), array([6, 1, 2, 5, 3, 0, 4]))
```

[redacted]

Here we apply inverse transform function to the label encoder inorder to store t he mappings done by the encoder.

[redacted]

```
list(le.inverse_transform([6, 2, 1, 5, 4, 3, 0]))
```

```
[nan, 'Poor', 'Moderate', 'Very Poor', 'Severe', 'Satisfactory', 'Good']
```

[redacted]

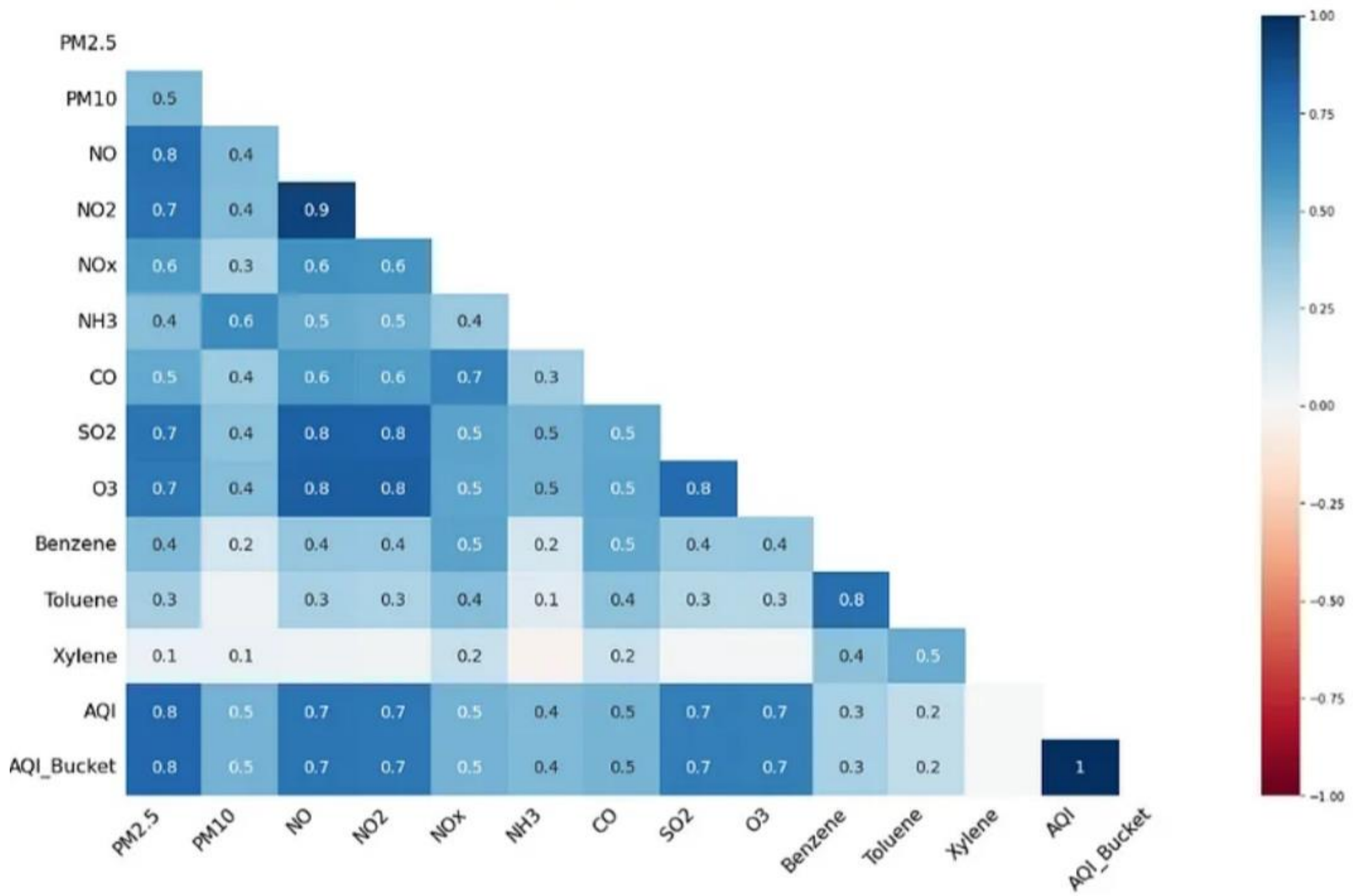
```
mappings = {
    'nan': 6,
    'Poor': 2,
    'Moderate': 1,
    'Very Poor': 5,
    'Severe': 4,
    'Satisfactory': 3,
    'Good': 0
}
```

PATTERN OF MISSING VALUES:

‘missingno’ is the python library which has functions for analysing the pattern of the missing values. According to the missing value pattern suitable imputation can be decided

```
msno.heatmap(df_c)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f1e6419ced0>



- From the above observation of distribution of missing values there is significant common pattern between the data columns. We observe that NO and NO2 are strongly correlated. As this is reading of air polluting gases, it can also be the case where industrial pollutants emit certain type of gases and vehicle pollutants have different set of gases. Hence there is correlation with missing values.

Imputation:

- We apply forward imputation to the given dataset. **Linear interpolation** is an imputation technique that assumes a linear relationship between data points and utilizes non-missing values from adjacent data points to compute a value for a missing data point.

Detailed guide for choice of imputation techniques can be found here here.

- Choose the columns to interpolate the missing data.
- Apply bfill to initial values with 'Nan' as there isn't any value prior to them to interpolate.

```
cols = ['PM2.5','PM10','NO','NO2', 'NOx','NH3','CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI',
'AQI_Bucket']

df_c_imp = df_c[cols]
df_c_imp.interpolate(limit_direction='forward', inplace=True);

df_c_1 = df_c[['City', 'Datetime']]
df_c_2 = pd.concat([df_c_1, df_c_imp], axis=1, join='inner')
df_c_2['AQI'].fillna(method='bfill', inplace=True)
```

Extract the range of AQI_Bucket labels, ie, based on AQI (Air Quality Index) corresponding classes are assigned in AQI_bucket. Hence knowing the range of AQI is important to assign the corresponding classes to 'Nan' values in target label(AQI_Bucket). Here we:

- Write conditions
- Extract the upper limit of the range
- Use the upper limit to add the mappings

```
modc = df_c['AQI_Bucket'] == 1
satc = df_c['AQI_Bucket'] == 3
vpc = df_c['AQI_Bucket'] == 5
prc = df_c['AQI_Bucket'] == 2
gdc = df_c['AQI_Bucket'] == 0
src = df_c['AQI_Bucket'] == 4

severec = np.max(df_c[src]['AQI'])
```

```
very_poorc = np.max(df_c[vpc]['AQI'])
satisfactoryc = np.max(df_c[satc]['AQI'])
poorc = np.max(df_c[prc]['AQI'])
moderatec = np.max(df_c[modc]['AQI'])
goodc = np.max(df_c[gdc]['AQI'])

print('maximum values for:')
print("severe {}\nvery poor {}\npoor {}\nmoderate {}\ngood{}\nsatisfactory {} ".format(seve
```

```
maximum values for:
severe 3133.0
very poor 400.0
poor 300.0
moderate 200.0
good50.0
satisfactory 100.0
```

```
(sr_cond = df_c_2['AQI'] > 400
vp_cond = df_c_2['AQI'] <=400
pr_cond = df_c_2['AQI'] <= 300
md_cond = df_c_2['AQI'] <= 200
st_cond = df_c_2['AQI'] <= 100
gd_cond = df_c_2['AQI'] <= 50

df_c_2.loc[sr_cond, 'AQI_Bucket'] = 4
df_c_2.loc[vp_cond, 'AQI_Bucket'] = 5
df_c_2.loc[pr_cond, 'AQI_Bucket'] = 2
df_c_2.loc[md_cond, 'AQI_Bucket'] = 1
df_c_2.loc[st_cond, 'AQI_Bucket'] = 3
df_c_2.loc[gd_cond, 'AQI_Bucket'] = 0
```

Check for Null and remove null

```
df_c_2.isnull().sum()
```

```
City              0
Datetime          0
PM2.5           665
PM10          38274
NO                0
NO2               0
NOx               0
NH3           48192
CO                0
SO2               0
O3                3
Benzene           0
Toluene           0
Xylene            0
AQI               0
AQI_Bucket        0
dtype: int64
```

```
df_c_2['PM2.5'] = df_c_2['PM2.5'].fillna(0)
df_c_2['O3'] = df_c_2['O3'].fillna(0)
df_c_2['NH3'] = df_c_2['NH3'].fillna(0)
df_c_2['PM10'] = df_c_2['PM10'].fillna(0)
df_c_2.isnull().sum()
```

Save the preprocessed data:

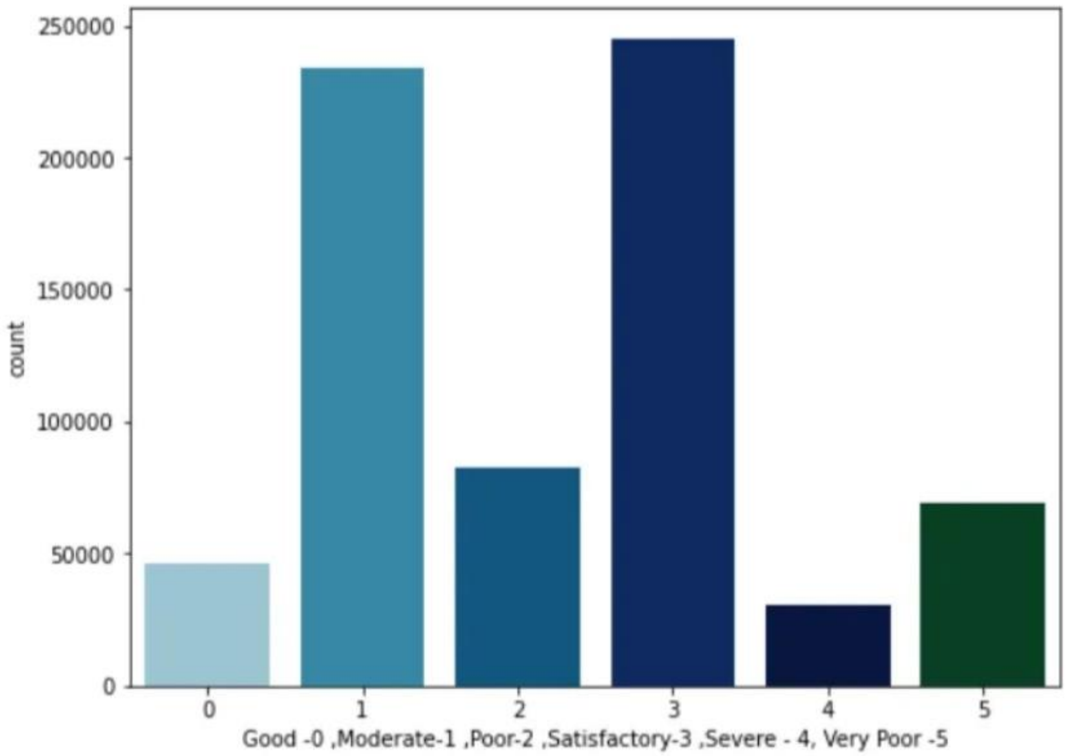
```
c = '/content/drive/MyDrive/Projects/imp_city_hour.csv'
df_c_2.to_csv(fp_c, index fp_=False)
```

Exploratory Data Analysis

Distribution of classes in the Data

Countplot is a univariate plot ie. single variable under consideration, showing the number of data points representing the categorical variables.

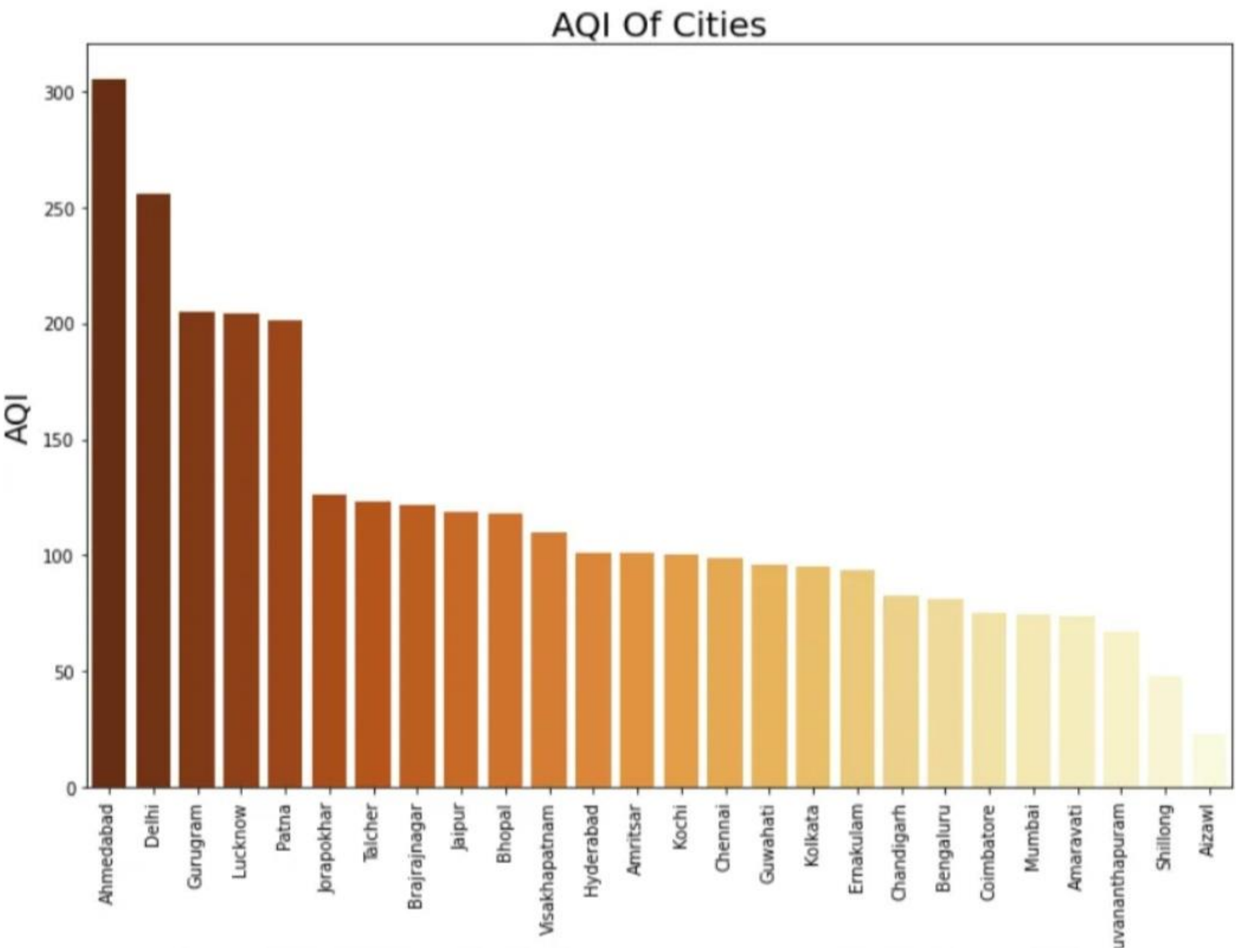
```
def countplot(x_val):
  plt.figure(figsize = (8, 6))
  sns.countplot(x = x_val, palette = 'ocean_r');
  plt.xlabel("Good -0 ,Moderate-1 ,Poor-2 ,Satisfactory-3 ,Severe - 4, Very Poor -5")
countplot(df_ch['AQI_Bucket'])
```



- Most cities fall nder moderate and satisfactory range.
- There are few representation for good air quality and very poor air quality.

Cities vs AQI Indexes

```
"""Select city, AQI from df_ch ORDER BY AQI DESC"""
# do average of every city aqi value,
city_vs_AQI = df_ch[['City', 'AQI']].groupby(['City']).median().sort_values('AQI',
ascending=False).reset_index()

def barplot(a,b, title, x_label, y_label, palet):
  plt.figure(figsize=(12,8))
  sns.barplot(x = a, y = b, palette =palet, order=a)
  plt.xticks(rotation = 90)
  plt.title(title, fontsize = 20)
  plt.xlabel(x_label, fontsize = 18)
  plt.ylabel(y_label, fontsize = 18)

barplot(city_vs_AQI['City'], city_vs_AQI['AQI'], "AQI Of Cities","Cities", "AQI","YlOrBr_r”)
```



Ahmedabad has AQI in the range of 300 followed by Delhi. 300 indicates poor quality.

Gases and top 3 cities with maximum of that particular gas:

Different cities are extracted which has maximum amount of gas type. Different gases have different cities in top 3.

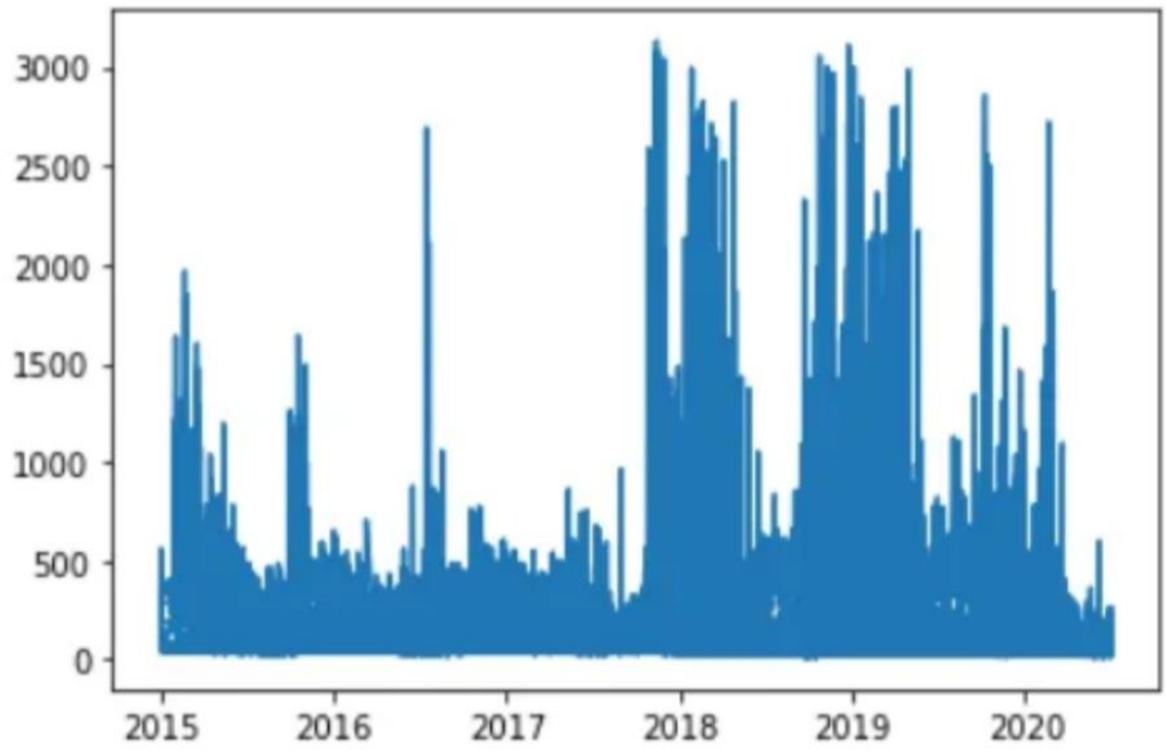
```
cols = ['PM2.5','PM10','NO','NO2', 'NOx','NH3','CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI',
'AQI_Bucket']

# Analyze the states with highest particular gases
dff = []
for i in cols:
  data = df_ch[[i, 'City']].groupby(['City']).median().sort_values(i, ascending =
False).iloc[0:3].reset_index()
  dff.append(data)
```

```
for i in range(len(cols)):
  print(dff[i])
  print("-------------------------")
```

```
         City       PM2.5
0       Patna   96.750000
1  Jorapokhar   94.879997
2       Delhi   86.839996
-------------------------
         City        PM10
0       Delhi  199.830002
1    Gurugram  128.202103
2  Jorapokhar  115.480003
-------------------------
      City         NO
0    Kochi  69.729996
1   Mumbai  37.826057
2  Talcher  20.980000
-------------------------
        City        NO2
0      Delhi  44.279999
1  Ahmedabad  32.479889
2    Kolkata  27.860001
-------------------------
         City        NOx
0       Kochi  66.160004
1  Jorapokhar  51.489998
2      Mumbai  44.734665
-------------------------
      City        NH3
0  Chennai  43.061951
1    Delhi  37.279999
2    Patna  35.077019
```

Datetime Vs AQI:

```
plt.plot(df_ch['Datetime'], df_ch['AQI'])
```



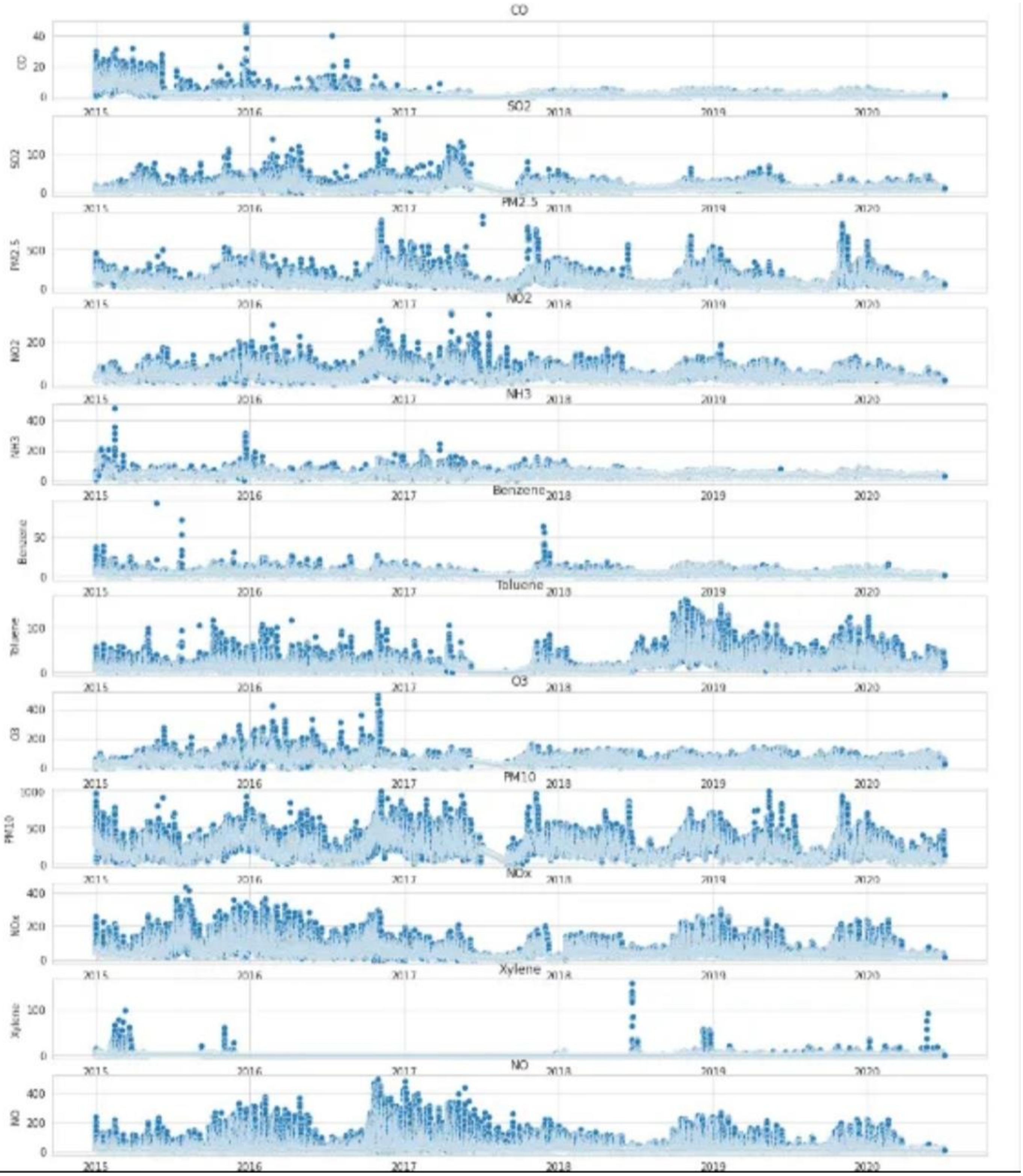
We observe that AQI is rising to severe in year 2018–2019 and slightly reduced in 2020 due to lockdown.

Analyzing Pollution in Delhi:

Select data values corresponding to city= Delhi. Scatterplot is used for representation.

```
dli = df_ch[df_ch.City == 'Delhi'].reset_index()

sns.set_style("whitegrid")

fig, axes = plt.subplots(nrows = 12,
                         ncols = 1,
                         figsize = (16,20))
axes[0].set_title('CO')
sns.scatterplot(x=dli['Datetime'], y =dli['CO'],data = df_ch,ax = axes[0], alpha = 1 )
axes[1].set_title('SO2')
sns.scatterplot(x=dli['Datetime'], y =dli['SO2'],data = df_ch, ax = axes[1], alpha = 1)

axes[2].set_title('PM2.5')
sns.scatterplot(x=dli['Datetime'], y =dli['PM2.5'],data = df_ch,ax = axes[2], alpha = 1 )
axes[3].set_title('NO2')
sns.scatterplot(x=dli['Datetime'], y =dli['NO2'],data = df_ch, ax = axes[3], alpha = 1)

axes[4].set_title('NH3')
```

```
sns.scatterplot(x=dli['Datetime'], y =dli['NH3'],data = df_ch,ax = axes[4], alpha = 1 )
axes[5].set_title('Benzene')
sns.scatterplot(x=dli['Datetime'], y =dli['Benzene'],data = df_ch, ax = axes[5], alpha = 1)

axes[6].set_title('Toluene')
sns.scatterplot(x=dli['Datetime'], y =dli['Toluene'],data = df_ch,ax = axes[6], alpha = 1 )
axes[7].set_title('O3')
sns.scatterplot(x=dli['Datetime'], y =dli['O3'],data = df_ch, ax = axes[7], alpha = 1)

axes[8].set_title('PM10')
sns.scatterplot(x=dli['Datetime'], y =dli['PM10'],data = df_ch,ax = axes[8], alpha = 1 )
axes[9].set_title('NOx')
sns.scatterplot(x=dli['Datetime'], y =dli['NOx'],data = df_ch, ax = axes[9], alpha = 1)

axes[10].set_title('Xylene')
sns.scatterplot(x=dli['Datetime'], y =dli['Xylene'],data = df_ch,ax = axes[10], alpha = 1 )
axes[11].set_title('NO')
sns.scatterplot(x=dli['Datetime'], y =dli['NO'],data = df_ch, ax = axes[11], alpha = 1)
```



We observe different pattern of density of gases from 2015 to 2020.

- There is rise of toluene
- Particulate matters, NOx NO2, have similar distributions.

Do explore further! In the next part we are upto statistical analysis.

CONTINUE BUILDING THE PROJECT BY DEVELOPING THE DATA – SHARING PLATFORM

Air pollution is one of the biggest threats to the present-day environment. Everyone is beingaffected by air pollution day by day including humans, animals, crops, cities, forests and aquaticecosystems. Besides that, it should be controlled at a certain level to prevent the increasing rate ofglobal warming. This project aims to design an IOT-based air pollution monitoring systemusing the internet from anywhere using a computer or mobile to monitor the air quality of thesurroundings and environment. There are various methods and instruments available for themeasurement and monitoring quality of air. The IoT-based air pollution monitoring systemwould not only help us to monitor the air quality but also be able to send alert signals whenever the air quality deteriorates and goes down beyond a certain level.In this system, NodeMCU plays the main controlling role. It has been programmed in a manner,such that, it senses the sensory signals from the sensors and shows the quality level via ledindicators. Besides the harmful gases (such as CO2, CO, smoke, etc) temperature and humidity canbe monitored through the temperature and humidity sensor by this system. Sensor responses are fedto the NodeMCU which displays the monitored data in the ThingSpeak cloud which can be utilizedfor analyzing the air quality of that area.The following simple flow diagramindicates the working mechanism of the IoT-based Air Pollution Monitoring System.

IOT BASED AIR QUALITY MONITORING SYSTEM

Air is getting polluted because of the release of toxic gases by industries, vehicle emissions and increased concentration of harmful gases and particulate matter in theatmosphere.

The level of pollution is increasing rapidly due to factors like industries, urbanization,increase in population, vehicle use which can affect human health. Particulate matter isone of the most important parameters having a significant contribution to the increase inair pollution. This creates a need for measurement and analysis of real-time air qualitymonitoring so that appropriate decisions can be taken in a timely period.This paper presents real-time standalone air quality monitoring. Internet of Things (IoT)inowadays finding profound use in each and every sector, plays a key role in our air qualitymonitoring system too. The setup will show the air quality in PPM on the webpage so thatwe can monitor it very easily.

AIM OF THE PROJECT:

Air is getting polluted because of the release of toxic gases by industries, vehicleemissions and increased concentration of harmful gases and particulate matter in theatmosphere.The level of pollution is increasing rapidly due to factors like industries, urbanization,increase in population, vehicle use which can affect human health. Particulate matter isone of the most important parameters having a significant contribution to the increase inair pollution. This creates a need for measurement and analysis of real-time air qualitymonitoring so that appropriate decisions can be taken in a timely period.This paper presents real-time standalone air quality monitoring. Internet of Things (IoT)isnowadays finding profound use in each and every sector, plays a key role in our air qualitymonitoring system too. The setup will show the air quality in PPM on the webpage so thatwe can monitor it very easily.

In this IoT project, we can monitor the pollution level from anywhere using your computer or mobile.

Components Used:

Hardware Components

1. NodeMCU V3
2. DHT11 Sensor Module
3. MQ-135 Gas Sensor Module
4. Veroboard(KS100)
5. Breadboard
6. Connecting Wires
7. AC-DC Adapters
8. LEDs emitting green, yellow and red colours
9. Resistors

SOFTWARE COMPONENTS:

1. ThinkSpeak Cloud
2. Arduino IDE

2.3 Brief Description of the Components

NodeMCU V3:

NodeMCU V3 is an open-source ESP8266 development kit, armed with the CH340G USBTTLSerial chip. It has firmware that runs on ESP8266 Wi-Fi SoC from Espressif Systems.Whilst cheaper, CH340 is super reliable even in industrial applications. It is tested to bestable on all supported platforms as well. It can be simply coded in Arduino IDE. It has avery low current consumption between 15 µA to 400 mA.

The pinout Diagram of NodeMC3

Pinout Diagram of NodeMCU V3

MQ-135 Gas Sensor Module:

The material of MQ135 is SnO_2, it is a special material: when exposed to clean air, it ishardly being conducted, however, when put in an environment with combustible gas, it has apretty performance of conductivity. Just make a simple electronic circuit, and convert thechange of conductivity to a corresponding output signal. MQ135 gas sensor is sensitive toAmmonia, Sulphide, Benzene steam, smoke and other harmful gases. Used for family,surrounding environment noxious gas detection device, apply to ammonia, aromatics,sulphur, benzene vapor, and other harmful gases/smoke, gas detection, tested concentrationrange: 10 to 1000ppm. In a normal environment, the environment which doesn't havedetected gas set the sensor's output voltage as the reference voltage, the analog outputvoltage will be about 1V, when the sensor detects gas, harmful gas concentration increases by20ppm per voltage increase by 0.1V.

Veroboard (KS100):

Veroboard is the original prototyping board.Sometimes referred to as 'stripboard' or 'matrix board' these offer total flexibility for hard wiringdiscrete components. Manufactured from acopper clad laminate board or Epoxy basedsubstrate, it is offered in both single and doublesided formats. Vero boards are available in a
wide range of board sizes and in both imperialand metric pitch – Veroboard is an ideal base forcircuit construction and offers even greater
adaptability using our range of terminal pins andassemblies. As with other stripboards, in usingVeroboard, components are suitably positioned and soldered to the conductors to form therequired circuit. Breaks can be made in the tracks, usually around holes, to divide the stripsinto multiple electrical nodes enabling increased circuit complexity. This type of wiringboard may be used for initial electronic circuit development, to construct prototypes forbench testing or in the production of complete electronic units in small quantities.

AC-DC Power Adapter:

An AC-DC power supply or adapter is an electrical device that obtains electricity from agrid-based power supply and converts it into adifferent current, frequency, and voltage. AC-DCpower supplies are necessary to provide the rightpower that an electrical component needs. The ACDCpower supply delivers electricity to devices thatwould typically run-on batteries or have no other power source.

LED (Red, Green & Yellow):

A light-emitting diode (LED) is a semiconductor light source that emits light when currentflows through it. Electrons in the semiconductor recombine with electron holes, releasingenergy in the form of photons. The colour of the light (corresponding to the energy of thephotons) is determined by the energy required for electrons to cross the band gap of thesemiconductor. White light is obtained by using multiple semiconductors or a layer of lightemittingphosphor on the semiconductor device. LEDs have many advantages overincandescent light sources, including lower power consumption, longer lifetime, improvedphysical robustness, smaller size, and faster switching. In exchange for these generallyfavourable attributes, disadvantages of LEDsinclude electrical limitations to low voltage andgenerally to DC (not AC) power, inability toprovide steady illumination from a pulsing DCor an AC electrical supply source, and lessermaximum operating temperature and storagetemperature. In contrast to LEDs, incandescentlamps can be made to intrinsically run at virtually
any supply voltage, can utilize either AC or DCcurrent interchangeably, and will provide steady illumination when powered by AC orpulsing DC even at a frequency as low as 50 Hz. LEDs usually need electronic supportcomponents to function, while an incandescent bulb can and usually does operate directlyfrom an unregulated DC or AC power source.

Resistors:

A resistor is a passivetwo-terminalelectricalcomponent that implements electrical resistance asa circuit element. In electronic circuits, resistors areused to reduce current flow, adjust signal levels, todivide voltages, bias active elements, and terminatetransmission lines, among other uses. High-powerresistors that can dissipate many watts of electricalpower as heat may be used as part of motor
controls, in power distribution systems, or as testloads for generators. Fixed resistors have resistancesthat only change slightly with temperature, time or operating voltage.

Arduino IDE:

The Arduino IDE is open-sourcesoftware, which is used to writeand upload code to the Arduinoboards. The IDE application is suitable for different operatingsystems such as Windows, MacOS X, and Linux. It supports theprogramming languages C andC++. Here, IDE stands forIntegrated DevelopmentEnvironment. The program or code written in the Arduino IDE is often called sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software.The sketch is saved with the extension '.ino.'

ThingSpeak Cloud:

ThingSpeak is open-source software writtenin Ruby which allows users to communicatewith internet-enabled devices. It facilitatesdata access, retrieval and logging of data byproviding an API to both the devices andsocial network websites. ThingSpeak wasoriginally launched by ioBridge in 2010 as aservice in support of IoT applications.ThingSpeak has integrated support from thenumerical computing software MATLABfrom MathWorks, allowing ThingSpeak usersto analyse and visualize uploaded data using MATLAB without requiring the purchase of a MATLAB license from MathWorks.

Working Procedures:

NodeMCU plays the main controlling role in this project. It has been programmed in a manner,such that, it senses the sensory signals from the sensors and shows the quality level via ledindicators. The DHT11 sensor module is used to measure the temperature and the humidity of thesurroundings.With the help of the MQ-135 gas sensor module, air quality is measured in ppm.

These data are fed to the ThinkSpeak cloud over the internet. We have also provided LED indicators to indicate the safety levels.

STEP 1. Firstly, the calibration ofthe MQ-135 gas sensor module is done. The sensor is set to preheat for 24 minutes. Then the software code is uploaded to the NodeMCU followed by the hardware circuit to calibrate the sensor has been performed.

STEP 2. Then, the DHT11 sensor is set to preheat for 10 minutes.

STEP 3. The result of calibration found in STEP 1 is used to configure the final working code.

STEP 4. The final working code is then uploaded to the NodeMCU.

STEP 5. Finally, the complete hardware circuit is implemented.
The software codes and the hardware circuits are described in the following chapters

SOFTWARE CODE for Calibration of MQ135 Sensor:

```
void setup()
{
Serial.begin(9600); //Baud rate
```



```
pinMode(A0,INPUT);
}
void loop()
{
float sensor_volt; //Define variable for sensor voltage
float RS_air; //Define variable for sensor resistance
float R0; //Define variable for R0
float sensorValue=0.0; //Define variable for analog readings
Serial.print("Sensor Reading = ");
Serial.println(analogRead(A0));
for(int x = 0 ; x < 500 ; x++) //Start for loop
{
sensorValue = sensorValue + analogRead(A0); //Add analog values of sensor 500 times
}
sensorValue = sensorValue/500.0; //Take average of readings
sensor_volt = sensorValue*(5.0/1023.0); //Convert average to voltage
```

```
RS_air = ((5.0*1.0)/sensor_volt)-1.0; //Calculate RS in fresh air
R0 = RS_air/3.7; //Calculate R0
Serial.print("R0 = "); //Display "R0"
Serial.println(R0); //Display value of R0
delay(1000); //Wait 1 second


}
```

Execution of the Main Program:

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <ThingSpeak.h>
DHT dht(D5, DHT11);
#define LED_GREEN D2
#define LED_YELLOW D3
#define LED_RED D4
#define MQ_135 A0
int ppm=0;
float m = -0.3376; //Slope
float b = 0.7165; //Y-Intercept
float R0 = 3.12; //Sensor Resistance in fresh air from previous code
WiFiClient client;
long myChannelNumber = 123456; // Channel id
```



```
const char myWriteAPIKey[] = "API_Key";
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
pinMode(LED_GREEN,OUTPUT);
pinMode(LED_YELLOW,OUTPUT);
pinMode(LED_RED,OUTPUT);
pinMode(MQ_135, INPUT);
WiFi.begin("WiFi_Name", "WiFi_Password");
while(WiFi.status() != WL_CONNECTED)
{
delay(200);
Serial.print(".");
}
Serial.println();
Serial.println("NodeMCU is connected!");
Serial.println(WiFi.localIP());
dht.begin();
ThingSpeak.begin(client);
}
void loop() {
float sensor_volt; //Define variable for sensor voltage
float RS_gas; //Define variable for sensor resistance
float ratio; //Define variable for ratio
int sensorValue;//Variable to store the analog values from MQ-135
float h;
float t;
float ppm_log; //Get ppm value in linear scale according to the the ratio value
```

```
float ppm; //Convert ppm value to log scale
h = dht.readHumidity();
delay(4000);
t = dht.readTemperature();
delay(4000);
sensorValue = analogRead(gas_sensor); //Read analog values of sensor
sensor_volt = sensorValue*(5.0/1023.0); //Convert analog values to voltage
RS_gas = ((5.0*1.0)/sensor_volt)-1.0; //Get value of RS in a gas
ratio = RS_gas/R0; // Get ratio RS_gas/RS_air
ppm_log = (log10(ratio)-b)/m; //Get ppm value in linear scale according to the ratio value
ppm = pow(10, ppm_log); //Convert ppm value to log scale
Serial.println("Temperature: " + (String) t);
Serial.println("Humidity: " + (String) h);
```



```
Serial.println("Our desired PPM = "+ (String) ppm);
ThingSpeak.writeField(myChannelNumber, 1, t, myWriteAPIKey);
delay(20000);
ThingSpeak.writeField(myChannelNumber, 2, h, myWriteAPIKey);
delay(20000);
ThingSpeak.writeField(myChannelNumber, 3, ppm, myWriteAPIKey);
delay(20000);
if(ppm<=100)
{
digitalWrite(LED_GREEN,HIGH);
digitalWrite(LED_YELLOW,LOW);
digitalWrite(LED_RED,LOW);
}
else if(ppm<=200)
{
digitalWrite(LED_GREEN,LOW);
digitalWrite(LED_YELLOW,HIGH);
digitalWrite(LED_RED,LOW);
}
else
{
digitalWrite(LED_GREEN,LOW);
digitalWrite(LED_YELLOW,LOW);
digitalWrite(LED_RED,HIGH);
}
delay(2000);


}
```

CUNCLUTION AND FUTURE WORK (PHASE 5):

PROJECT CONCLUTION:

In the phase 5 coclution, we wil summarize the key finding and insides from the advanced regression techniques. We will reiterate the impact of these techniques on improving the Air quality monitoring.

THE END.....