# AIR QUALITY MONITORING

USING IOT

SUBMITTED BY
Arunkumar N S
au721221106011
21eca12@karpagamtech.ac.in

## Contents:

## Abstract:

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

- Humans can be adversely affected by exposure to air pollutants in ambient air. Hence, health-based standards and objectives for some pollutants in the air are set by each country detection and measurement of contents of the atmosphere are becoming increasingly important. Careful planning of measurements is essential. One of the major factors that influence the representativeness of data collected is the location of monitoring stations the planning and setting up of monitoring stations are complex and incurs a huge expenditure. An IoT-based real-time air pollution monitoring system is proposed to monitor the pollution levels of various pollutants. The geographical area is classified as industrial, Residential, and traffic zones this article proposes an IoT system that could be deployed at any location and store the measured values in a cloud database, perform pollution analysis, and display the pollution level at any given location.

- High population and urbanization growth rate raises the issue of air pollution in recent years. Air quality monitoring is one of the major concerns due to its influence on human health. With the advancement in sensing and embedded technology, Internet of Things(IoT) becomes one of the economic alternative to implement air quality monitoring system(AQMS)compared to costly and fixed air quality monitoring stations. In this paper we present the ample review of candidate enabling technology for IoT based AQMS architecture. Specifically, we start with overview of major low cost air pollutant sensors classification, typical error sources and calibration methodologies. Then we present analysis and comparative study of infrastructure protocols and application layer protocols to support IoT based architecture for AQMS. We also review existing system and categorised them based on deployment strategy employed. Finally, challenges involved in building such systems are discussed in detail.

## Objectives:

- Quality of air can be checked indoors as well as outdoor.
- Detecting a wide range of physical parameters.
- Indoor air quality monitoring.
- Industrial perimeter monitoring.
- Roadside pollution monitoring.
- To make this data available to the common man
- Locating contamination problem areas and understanding their spacetime changes.
- Complying with atmospheric air protection legislation.
- Obtaining the necessary information to define Action Plans as stipulated by European directives or other international regulations if alert thresholds are breached.
- Informing citizens regarding local air quality status.
- Evaluating the impact of air pollution on ecosystems, wildlife, and vegetation.
- Developing an early warning system to alert authorities and the public to potential air quality emergencies.

# INTRODUCTION:

- Air Quality Monitoring Networks allow the measurement, operation and predictive analysis of the evolution of air pollution in different areas (urban areas, industrial areas, special nature conservation areas, etc.) Some stations are equipped with meteorological sensors and/or noise level meters to measure noise levels.

- As the name suggests, embedded describes something that is connected to another item. An embedded system is a piece of computer hardware that also contains software. A stand-alone unit or a component of a bigger system can both be an embedded system. An embedded system with a microcontroller or microprocessor is designed to carry out a certain purpose. For instance, a fire alarm is an integrated device that only detects smoke. An embedded system is similar to a computer system in that it is primarily designed to accomplish specific functions such as controlling data in various electronics-based systems, accessing data, processing, and storing data. Embedded systems are hardware and software combinations that are designed to perform a certain set of functions. The embedded system's most essential feature is that it regenerates the output in a very short amount of time. Many embedded systems will be encountered in our daily lives.

- The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals, or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

- Nowadays the air condition is very polluted. In recent years, car emissions, chemicals from factories, smoke, and dust are everywhere. That is the reason why now air condition is very polluted. The effect of air pollution is very bad for our health, especially for a place where the air in our body is taken for breathing.

- Air pollution cannot be detected by human feelings. Air pollution may contain a lot of dangerous substances such as ozone, particulate matter sulfur dioxide, nitrogen dioxide, carbonmonoxide, and lead. This proposed system uses a wireless sensor network with low-cost sensors and hardware components along the necessary software to effectively monitor the air pollution phenomenon.

# Design Thinking:

## *1).IoT Device Design:*

- Designing Internet of Things (IOT) devices for air quality monitoring involves several key considerations to ensure accurate and reliable data collection.

### *Data storage:*

- Include onboard storage or cloud integration for storing historical data. Cloud platforms like AWS, Azure, or Google Cloud offer scalable solutions for data storage and analysis.

### *GPS(Optional):*

- Incorporate GPS to provide location data, allowing for geospatial analysis and mapping of air quality.

## *2).Data sharing platform:*

- Creating a data sharing platform for air quality information is essential for transparency, collaboration, and informed decision-making.

***Data Archive:***

- Maintain an archive of historical air quality data for research and long-term trend analysis.

***Public Engagement:***

- Promote the platform to raise public awareness of air quality issues and encourage citizen engagement.

## ***3).Integration Approach:***

- Integrating air quality monitoring systems involves connecting various components and data sources to create a comprehensive and coherent system.

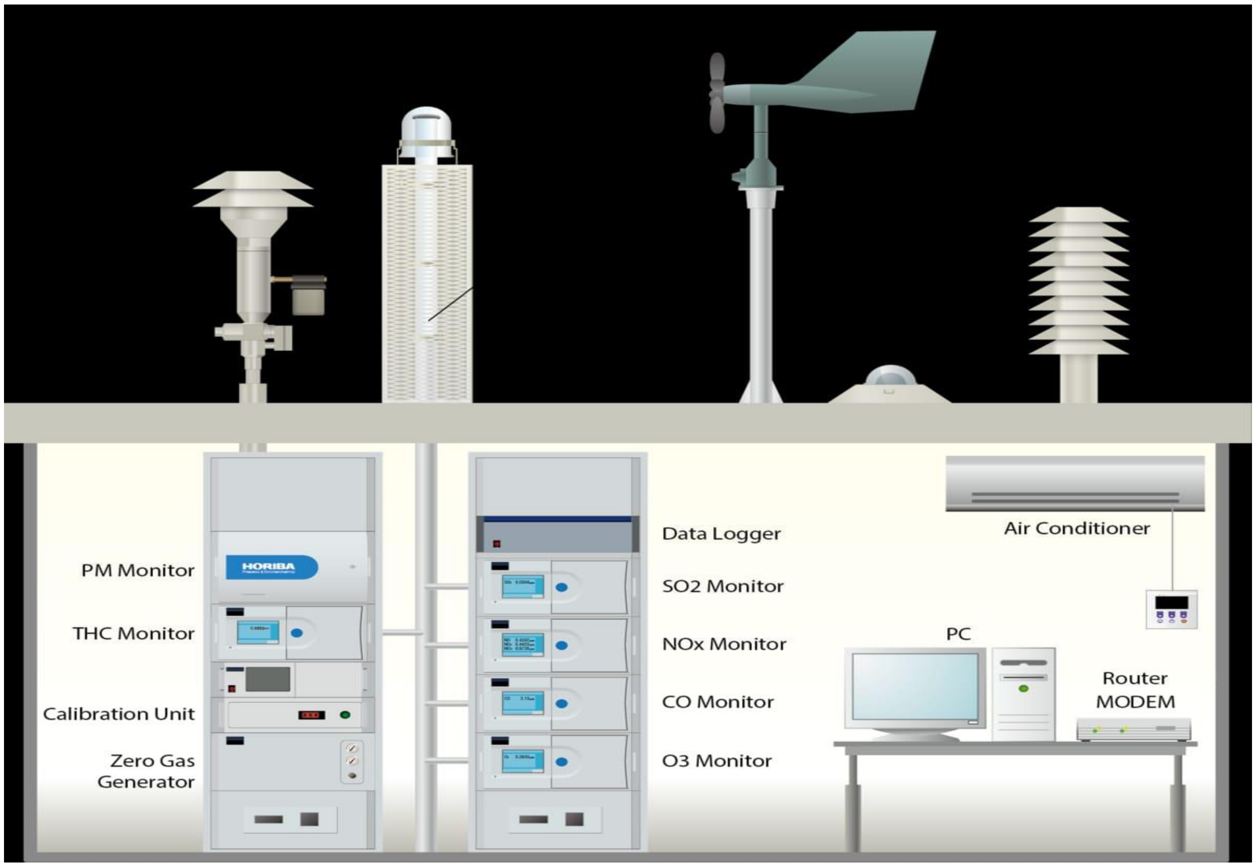
***Documentation and Training:***

- Provide documentation for users and developers on how to use the integrated systems. Offer training if data necessary.

***Compliances & Security:***

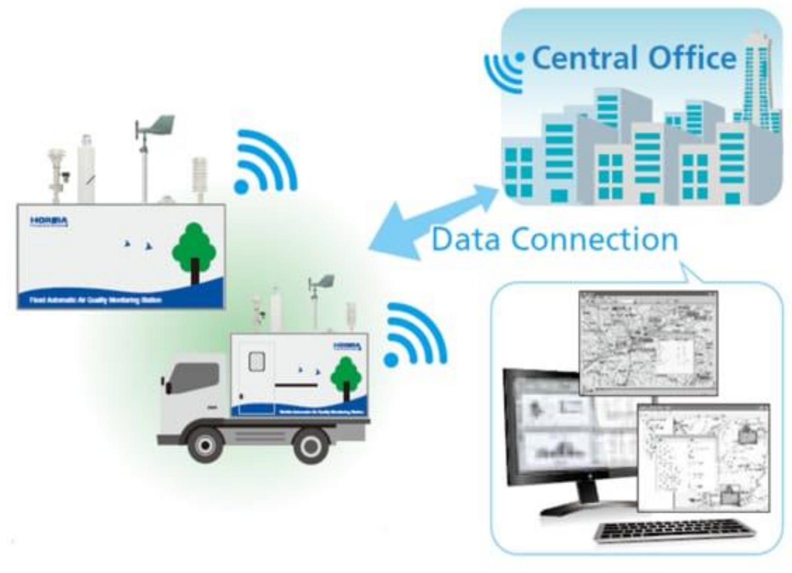
- Adhere to relevant data privacy regulations and security best practices when sharing and integrating data.

## BLOCK DIAGRAM:



**AQMS, the doctor for "Human Health"**

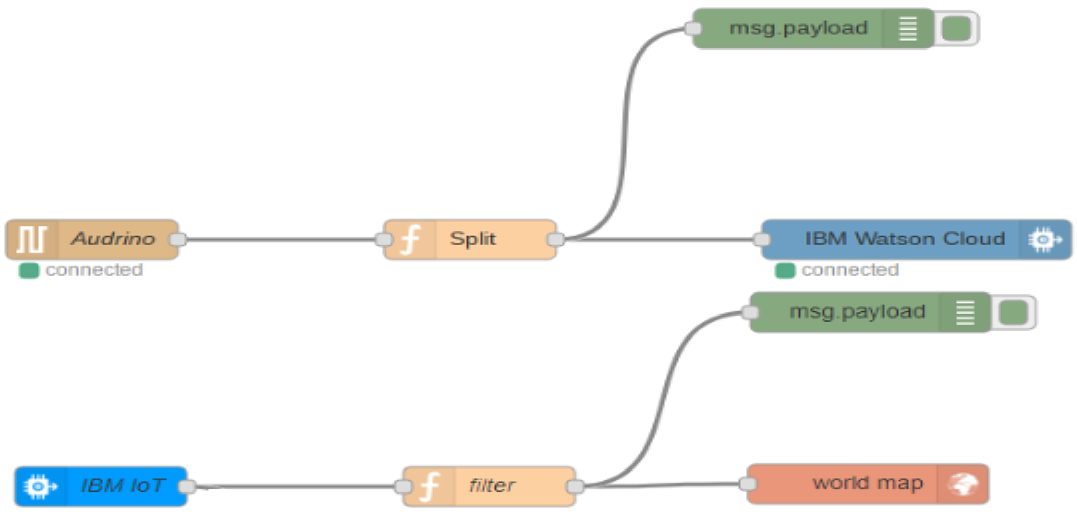
- An Air Quality Monitoring Station (AQMS) is a system that measures metrological parameters such as wind speed, wind direction, rainfall, radiation, temperature, barometric pressure and ambient parameters. The AQMS also integrates a series of ambient analyzers to monitor the concentration of air pollutants (such as SO2, NOx, CO, O3, THC, PM, etc.), continuously. HORIBA also provides mobile monitoring stations that can be used to monitor ambient conditions at multiple sites.



- HORIBA has more than 50 years experience providing ambient monitoring solutions, recognized around the world. HORIBA has supplied over 15,000 units with the major share in many regions. The monitoring station is tailor-made according to the customer's request. HORIBA can provide several types of stations, calibration equipment and more to meet your challenging monitoring requirements.

- The measured data can be remotely monitored and exported in various formats to the local central authorities. The data can be published via the Internet for easy public access to raise awareness on current air pollution levels. This way, the public can prevent outdoor activities and reduce health impacts during heavy polluted days.

- An IOT based air pollution monitoring system is proposed that uses an MQ135 gas sensor interfaced to node MCU; the system is connected through ESP8266 wifi module to the thinks Speak cloud to analyze the sensor data



- A node is a reactive Node.js application triggered by receiving messages on at most one input port (dubbed source ) and sending the results of (side-effectful) computations on output ports (dubbedsinks ), which can be potentially multiple, unlike the input port. Figure 3 illustrates the code structure of a Node-REDnode. A special type of node without sources and sinks, called configuration node, is used for sharing configuration data, such as login credentials, between multiple nodes. A flow is a representation of nodes connected together. End users can either create their own flows on the platform's environment or deploy existing flows pro- vided by the official Node-RED catalog [33] and by third parties. In Node-RED, contexts provide a shared communication channel between different nodes without using the explicit messages that pass through a flow [40]. Therefore the node wiring visible in the user interface reflects only a part of the information flows that are possible in the flow. It introduces an implicit channel that is not visible to the user via the graphical interface of a flow. Node-RED defines three scope levels for the contexts.

- The provided policies can later be vetted by the platformand the user, before deploying the node. SandTrap [3] offers a pol- icy generation mechanism to aid developers in designing the policies, enabling both baseline and advanced policies customized by developers or users to express fine-grained app- specific security goals. In the following, we discuss Node-RED attacks and vulnerabilities that mo- tivate enriching the policy mechanismwith different granularity levels. These policies will further be formalized in Section 3.

- Node-RED is “a programming tool for wiring together hardware devices, APIs and online services”, which provides a way of “low- code programming for event- driven applications” [36]. As an open-source platform, Node-RED is mainly tar- geted for deployment as a single-user platform, although it is also available on the IBM Cloud platform.

## Project Resources:

### ***Hardware Requirements:***

- Air Quality sensor (MQ 135)
- Potentiometer
- 16x2 LCD Panel
- Arduino Uno
- Wires

### ***Software Requirements:***

- Arduino (Version 1.8.2)
- THINGSPEAK website

## ***Hardware Requirements:***

### **Air Quality Sensor (MQ135):-**

**Product Description:**

- Air quality click is suitable for detecting ammonia ($NH_3$), nitrogen oxides ($NO_x$) benzene, smoke, $CO_2$, and other harmful or poisonous gases that impact air quality. The MQ-135 sensor unit has a sensor layer made of tin dioxide ($SnO_2$), an inorganic compound that has lower conductivity in clean air than when polluting gases are present. To calibrate Air quality, use the onboard potentiometer to adjust the load resistance on the sensor circuit.

## Pin Description:

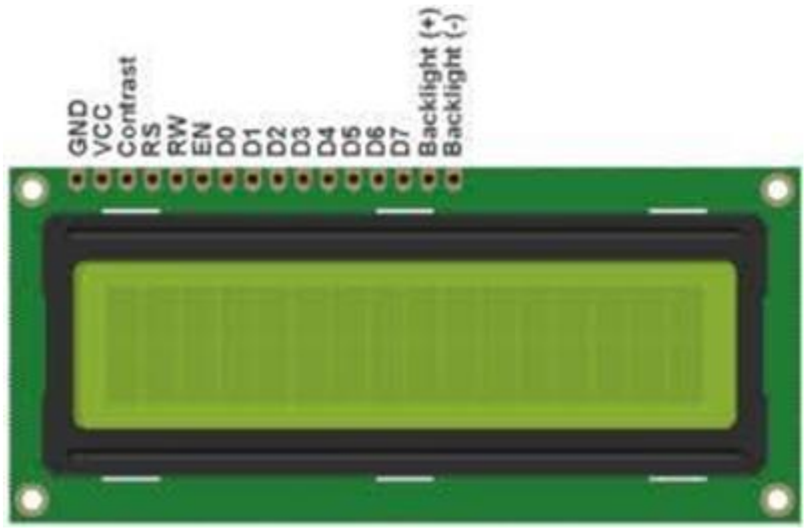
- The VDD power supply 5V DC
- GND used to connect the module to system ground
- DIGITAL OUT, you can also use this sensor to get digital output from this pin, by setting threshold value using the potentiometer.
- ANALOG OUT, this pin outputs 0–5V analog voltage based on the intensity of the gas.

## Potentiometer: -

### Product Description:

- A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat. The measuring instrument called a potentiometer is essentially a voltage divider used for measuring electric potential (voltage); the component is an implementation of the same principle, hence its name.

## <u>16X2 LCD Panel:-</u>



### <u>Product Description:</u>

- A liquid-crystal display (LCD) is a flat-panel display or another electronically modulated optical device that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly, instead of using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and seven-segment displays.

## <u>Pin Description:</u>

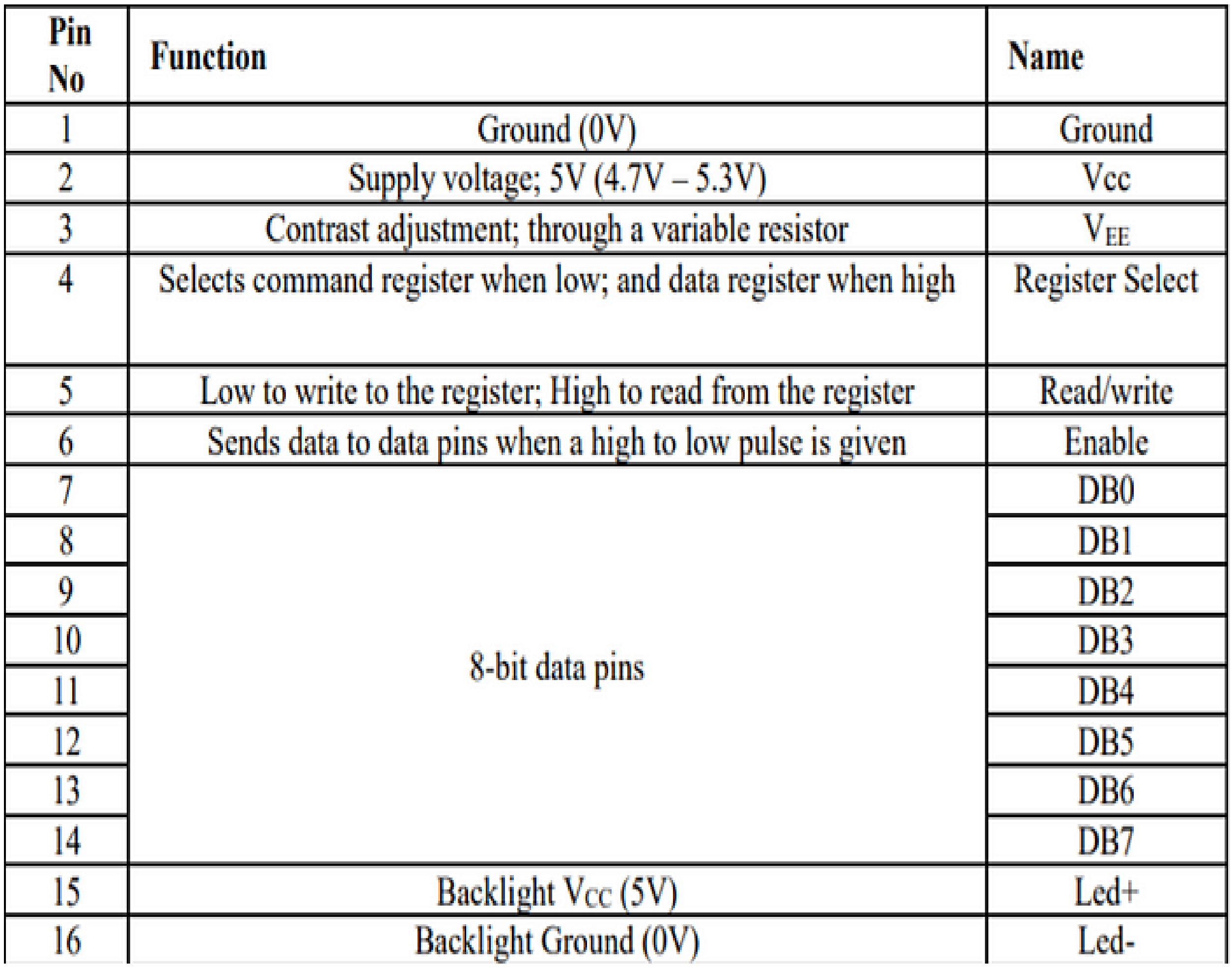
- Connect pin 1 (VEE) to the ground.
- Connect pin 2 (VDD or VCC) to the 5V.
- Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND. The potentiometer is used to control the screen contrast of the LCD. A potentiometer of values other than 10K will work too.
- Connect pin 4 (RS) to 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 pin 11 of the Arduino. The RS and E pin are the control pins that are used to send data and characters.

The following four pins are data pins that 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.

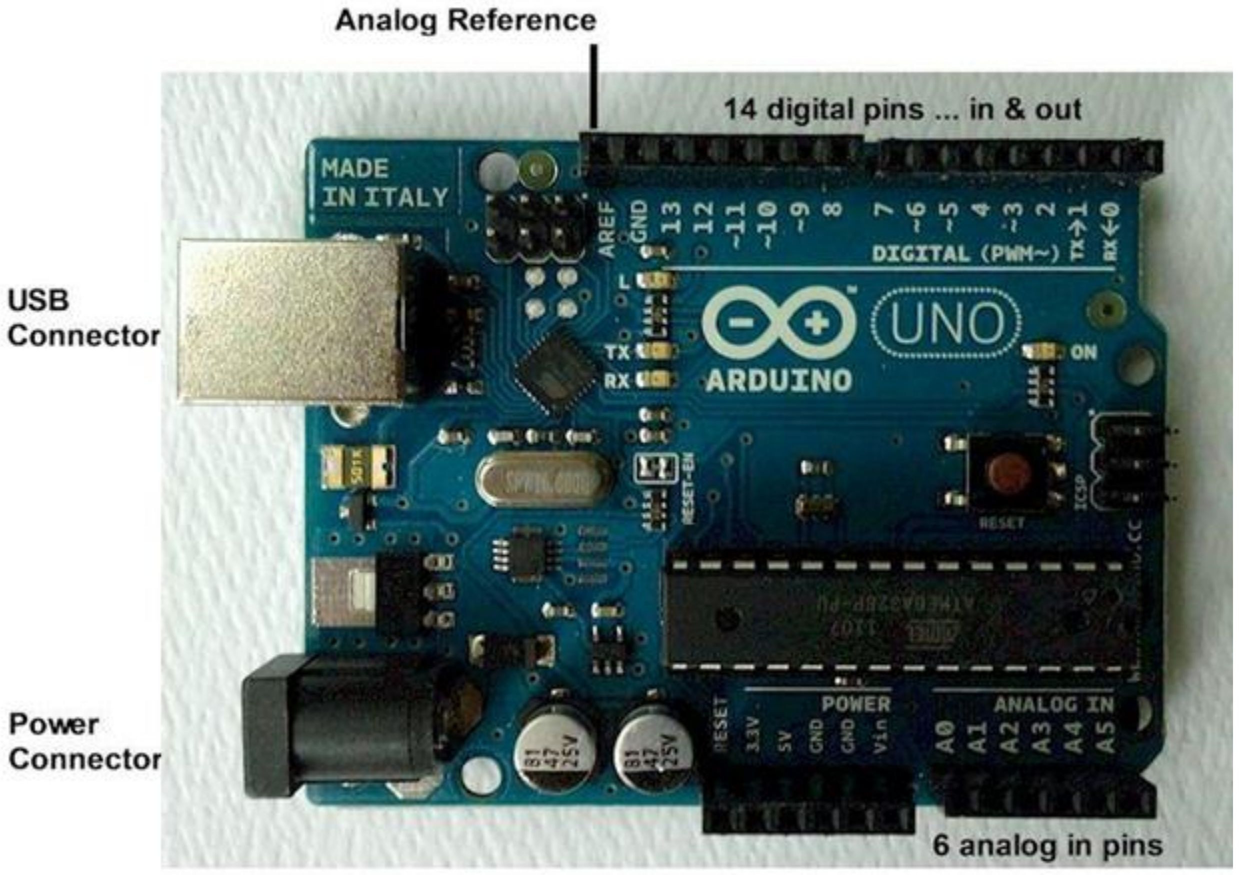
- Connect pin 15 to the VCC through the 220-ohm resistor. The resistor will be used to set the backlight brightness. Larger values will make the backlight much darker.

- Connect pin 16 to the Ground.



| Pin No | Function | Name |
|---|---|---|
| 1 | Ground (0V) | Ground |
| 2 | Supply voltage; 5V (4.7V – 5.3V) | Vcc |
| 3 | Contrast adjustment; through a variable resistor | $V_{EE}$ |
| 4 | Selects command register when low; and data register when high | Register Select |
| 5 | Low to write to the register; High to read from the register | Read/write |
| 6 | Sends data to data pins when a high to low pulse is given | Enable |
| 7 | 8-bit data pins | DB0 |
| 8 | | DB1 |
| 9 | | DB2 |
| 10 | | DB3 |
| 11 | | DB4 |
| 12 | | DB5 |
| 13 | | DB6 |
| 14 | | DB7 |
| 15 | Backlight $V_{CC}$ (5V) | Led+ |
| 16 | Backlight Ground (0V) | Led- |

## Arduino Uno:-

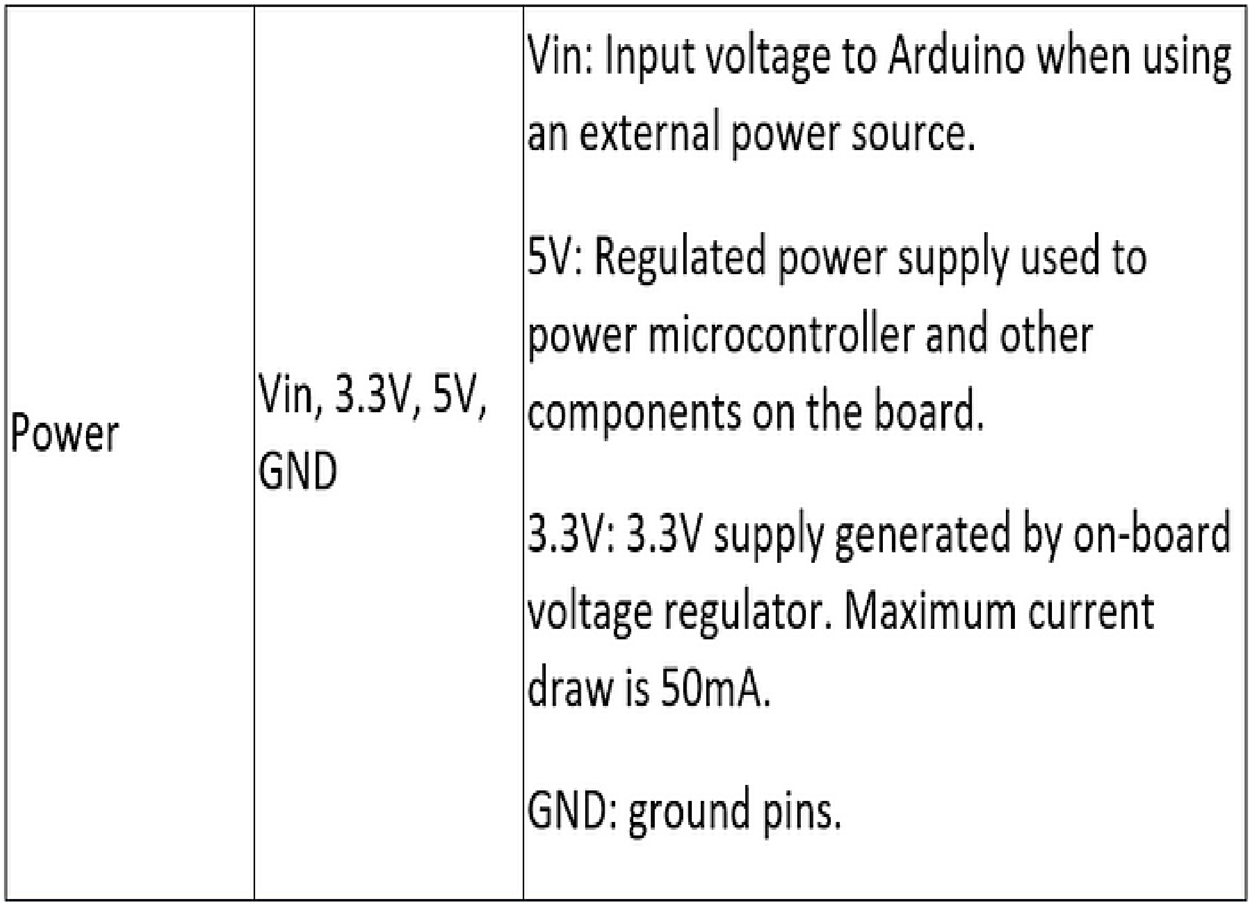


## Product Description:

- Arduino is an open-source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world.

- Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

## Pin Description:



| | | |
|---|---|---|
| Power | Vin, 3.3V, 5V, GND | Vin: Input voltage to Arduino when using an external power source.<br><br>5V: Regulated power supply used to power microcontroller and other components on the board.<br><br>3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.<br><br>GND: ground pins. |

# Program:

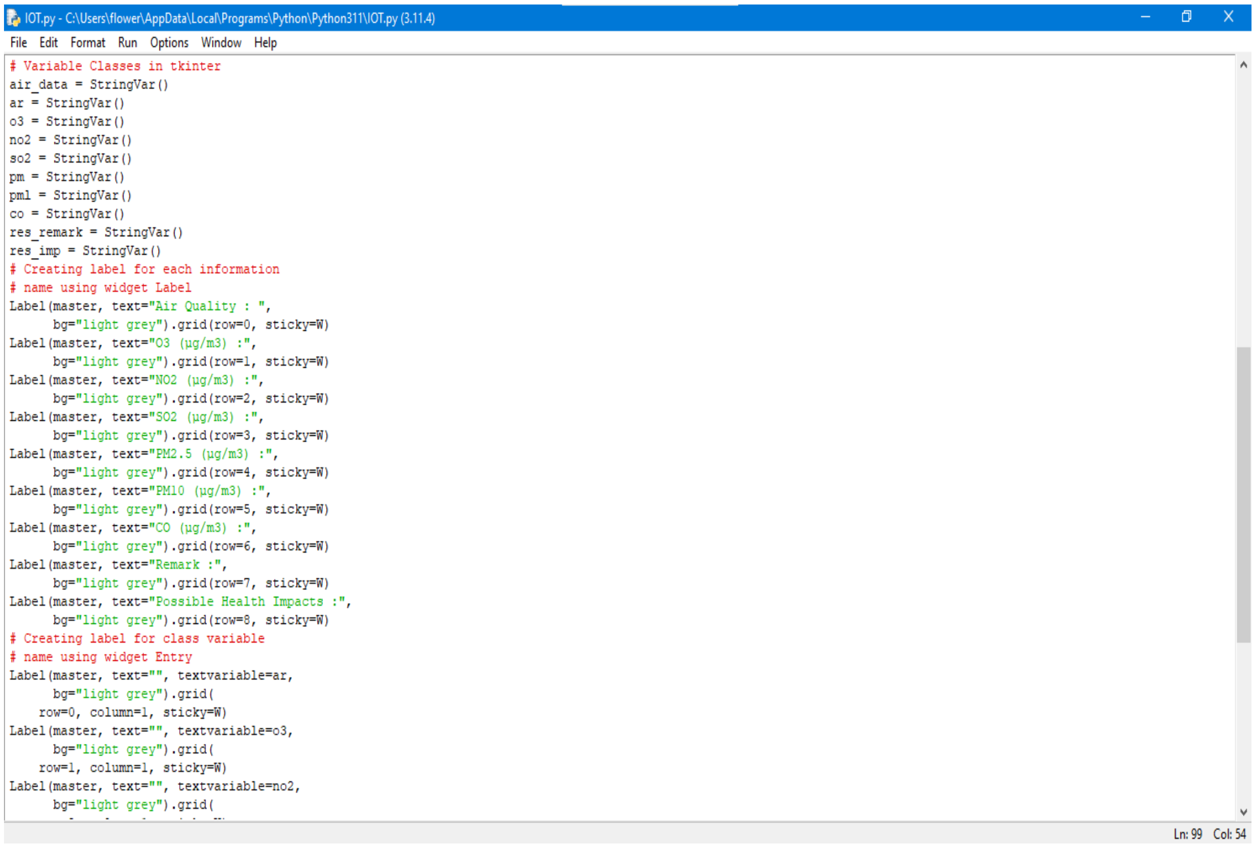
```
from tkinter import *

import requests

from bs4 import BeautifulSoup

# link for extract html data

def getdata(url):

   r = requests.get(url)

   return r.text

def airinfo():

   soup = BeautifulSoup(htmldata, 'html.parser')

   res_data = soup.find(class_="DonutChart--innerValue--2rO41 AirQuality--extendedDialText--
2AsJa").text

   air_data = soup.find_all(class_="DonutChart--innerValue--2rO41 AirQuality--pollutantDialText--
3Y7DJ")

   air_data=[data.text for data in air_data]

   ar.set(res_data)

   o3.set(air_data[0])

   no2.set(air_data[1])

   so2.set(air_data[2])

   pm.set(air_data[3])

   pml.set(air_data[4])

   co.set(air_data[5])

   res = int(res_data)
```

```
    if res <= 50:
        remark = "Good"
        impact = "Minimal impact"
    elif res <= 100 and res > 51:
        remark = "Satisfactory"
        impact = "Minor breathing discomfort to sensitive people"
    elif res <= 200 and res >= 101:
        remark = "Moderate"
        impact = "Breathing discomfort to the people with lungs, asthma and heart diseases"
    elif res <= 400 and res >= 201:
        remark = "Very Poor"
        impact = "Breathing discomfort to most people on prolonged exposure"
    elif res <= 500 and res >= 401:
        remark = "Severe"
        impact = "Affects healthy people and seriously impacts those with existing diseases"
    res_remark.set(remark)
    res_imp.set(impact)
# object of tkinter
# and background set to grey
master = Tk()
master.configure(bg='light grey')
# Variable Classes in tkinter
air_data = StringVar()
ar = StringVar()
o3 = StringVar()
```

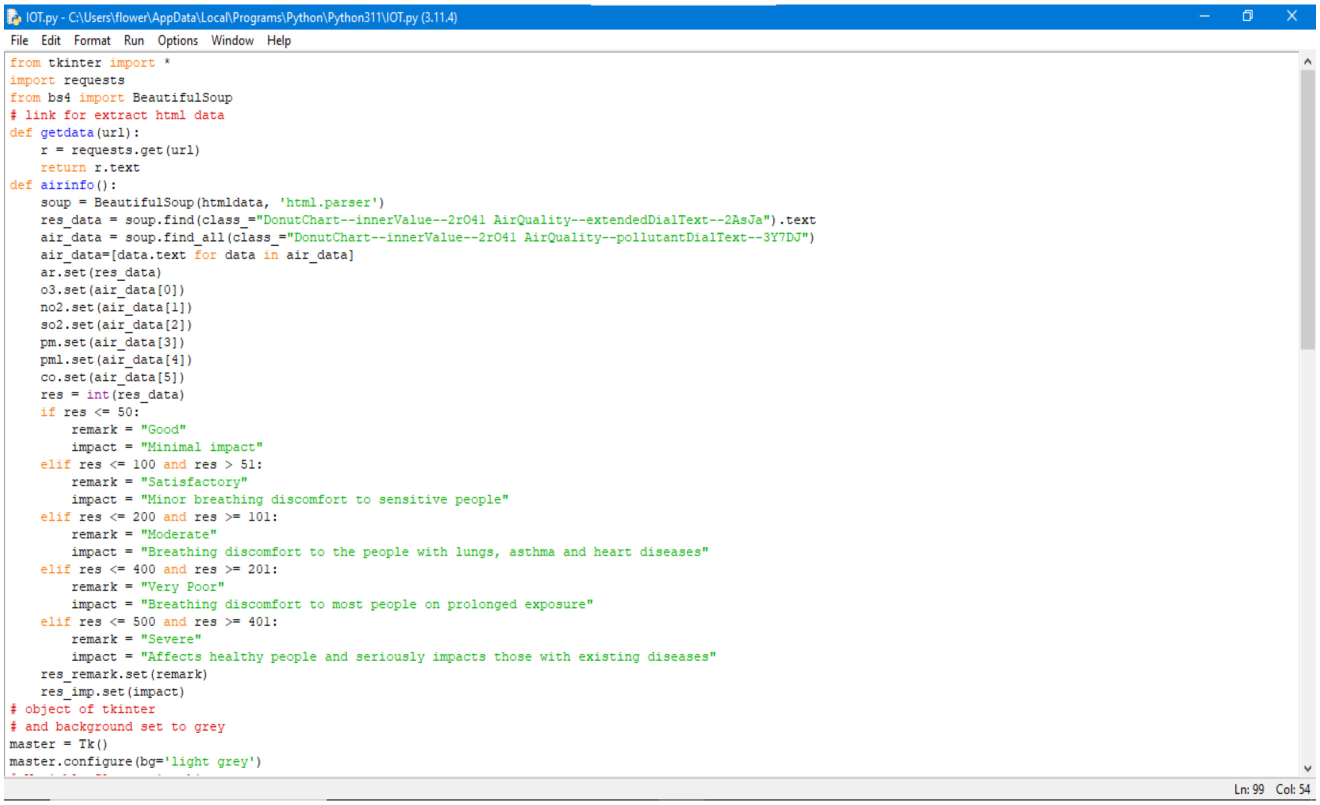
```
no2 = StringVar()

so2 = StringVar()

pm = StringVar()

pml = StringVar()

co = StringVar()

res_remark = StringVar()

res_imp = StringVar()

# Creating label for each information

# name using widget Label

Label(master, text="Air Quality : ",

    bg="light grey").grid(row=0, sticky=W)

Label(master, text="O3 (µg/m3) :",

    bg="light grey").grid(row=1, sticky=W)

Label(master, text="NO2 (µg/m3) :",

    bg="light grey").grid(row=2, sticky=W)

Label(master, text="SO2 (µg/m3) :",

    bg="light grey").grid(row=3, sticky=W)

Label(master, text="PM2.5 (µg/m3) :",

    bg="light grey").grid(row=4, sticky=W)

Label(master, text="PM10 (µg/m3) :",

    bg="light grey").grid(row=5, sticky=W)

Label(master, text="CO (µg/m3) :",

    bg="light grey").grid(row=6, sticky=W)

Label(master, text="Remark :",

    bg="light grey").grid(row=7, sticky=W)
```

```
Label(master, text="Possible Health Impacts :",

      bg="light grey").grid(row=8, sticky=W)

# Creating label for class variable

# name using widget Entry

Label(master, text="", textvariable=ar,

      bg="light grey").grid(

    row=0, column=1, sticky=W)

Label(master, text="", textvariable=o3,

      bg="light grey").grid(

    row=1, column=1, sticky=W)

Label(master, text="", textvariable=no2,

      bg="light grey").grid(

    row=2, column=1, sticky=W)

Label(master, text="", textvariable=so2,

      bg="light grey").grid(

    row=3, column=1, sticky=W)

Label(master, text="", textvariable=pm,

      bg="light grey").grid(

    row=4, column=1, sticky=W)

Label(master, text="", textvariable=pml,

      bg="light grey").grid(

    row=5, column=1, sticky=W)

Label(master, text="", textvariable=co,

      bg="light grey").grid(

    row=6, column=1, sticky=W)
```

```
Label(master, text="", textvariable=res_remark,

      bg="light grey").grid(row=7, column=1, sticky=W)

Label(master, text="", textvariable=res_imp,

      bg="light grey").grid(row=8, column=1, sticky=W)

# creating a button using the widget

b = Button(master, text="Check",

          command=airinfo, bg="Blue")

b.grid(row=0, column=2, columnspan=2,

       rowspan=2, padx=5, pady=5,)

mainloop()
```


IOT.py - C:\Users\flower\AppData\Local\Programs\Python\Python311\IOT.py (3.11.4)
File Edit Format Run Options Window Help
Ln: 99 Col: 54

```
# Variable Classes in tkinter
air_data = StringVar()
ar = StringVar()
o3 = StringVar()
no2 = StringVar()
so2 = StringVar()
pm = StringVar()
pml = StringVar()
co = StringVar()
res_remark = StringVar()
res_imp = StringVar()
# Creating label for each information
# name using widget Label
Label(master, text="Air Quality : ",
      bg="light grey").grid(row=0, sticky=W)
Label(master, text="O3 (µg/m3) :",
      bg="light grey").grid(row=1, sticky=W)
Label(master, text="NO2 (µg/m3) :",
      bg="light grey").grid(row=2, sticky=W)
Label(master, text="SO2 (µg/m3) :",
      bg="light grey").grid(row=3, sticky=W)
Label(master, text="PM2.5 (µg/m3) :",
      bg="light grey").grid(row=4, sticky=W)
Label(master, text="PM10 (µg/m3) :",
      bg="light grey").grid(row=5, sticky=W)
Label(master, text="CO (µg/m3) :",
      bg="light grey").grid(row=6, sticky=W)
Label(master, text="Remark :",
      bg="light grey").grid(row=7, sticky=W)
Label(master, text="Possible Health Impacts :",
      bg="light grey").grid(row=8, sticky=W)
# Creating label for class variable
# name using widget Entry
Label(master, text="", textvariable=ar,
      bg="light grey").grid(
    row=0, column=1, sticky=W)
Label(master, text="", textvariable=o3,
      bg="light grey").grid(
    row=1, column=1, sticky=W)
Label(master, text="", textvariable=no2,
      bg="light grey").grid(
```

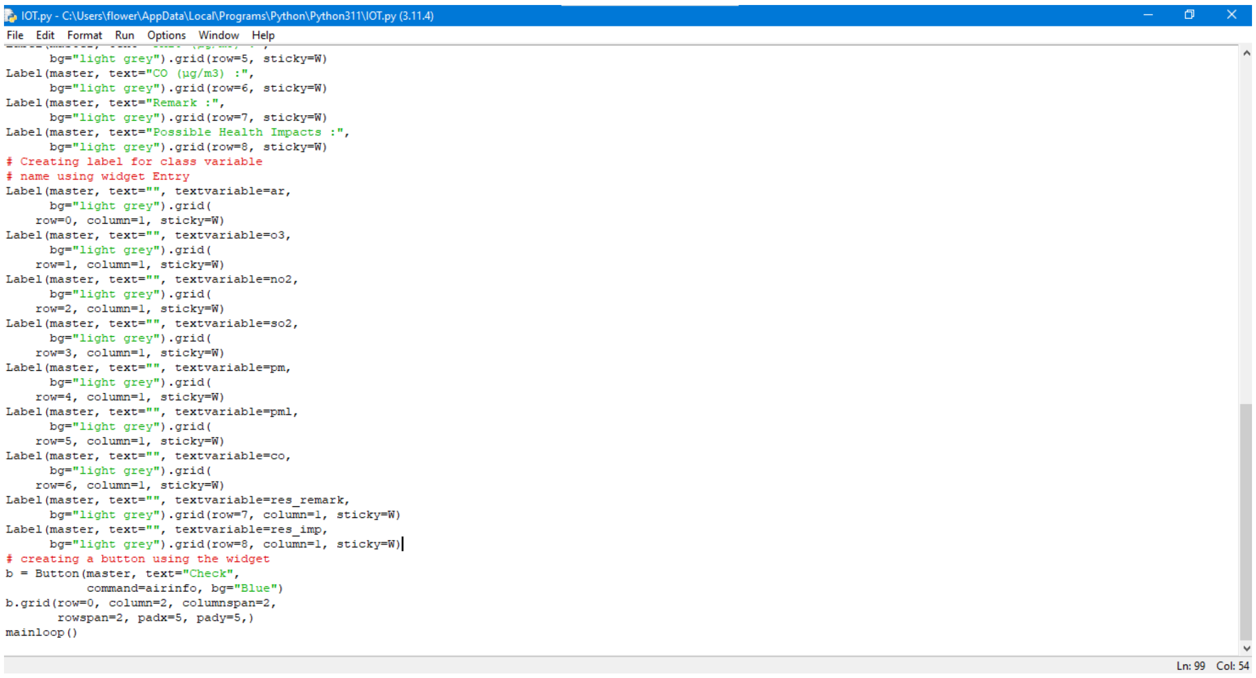


IOT.py - C:\Users\flower\AppData\Local\Programs\Python\Python311\IOT.py (3.11.4)

File Edit Format Run Options Window Help

```python
from tkinter import *
import requests
from bs4 import BeautifulSoup
# link for extract html data
def getdata(url):
    r = requests.get(url)
    return r.text
def airinfo():
    soup = BeautifulSoup(htmldata, 'html.parser')
    res_data = soup.find(class_="DonutChart--innerValue--2r041 AirQuality--extendedDialText--2AsJa").text
    air_data = soup.find_all(class_="DonutChart--innerValue--2r041 AirQuality--pollutantDialText--3Y7DJ")
    air_data=[data.text for data in air_data]
    ar.set(res_data)
    o3.set(air_data[0])
    no2.set(air_data[1])
    so2.set(air_data[2])
    pm.set(air_data[3])
    pml.set(air_data[4])
    co.set(air_data[5])
    res = int(res_data)
    if res <= 50:
        remark = "Good"
        impact = "Minimal impact"
    elif res <= 100 and res > 51:
        remark = "Satisfactory"
        impact = "Minor breathing discomfort to sensitive people"
    elif res <= 200 and res >= 101:
        remark = "Moderate"
        impact = "Breathing discomfort to the people with lungs, asthma and heart diseases"
    elif res <= 400 and res >= 201:
        remark = "Very Poor"
        impact = "Breathing discomfort to most people on prolonged exposure"
    elif res <= 500 and res >= 401:
        remark = "Severe"
        impact = "Affects healthy people and seriously impacts those with existing diseases"
    res_remark.set(remark)
    res_imp.set(impact)
# object of tkinter
# and background set to grey
master = Tk()
master.configure(bg='light grey')
```

Ln: 99 Col: 54



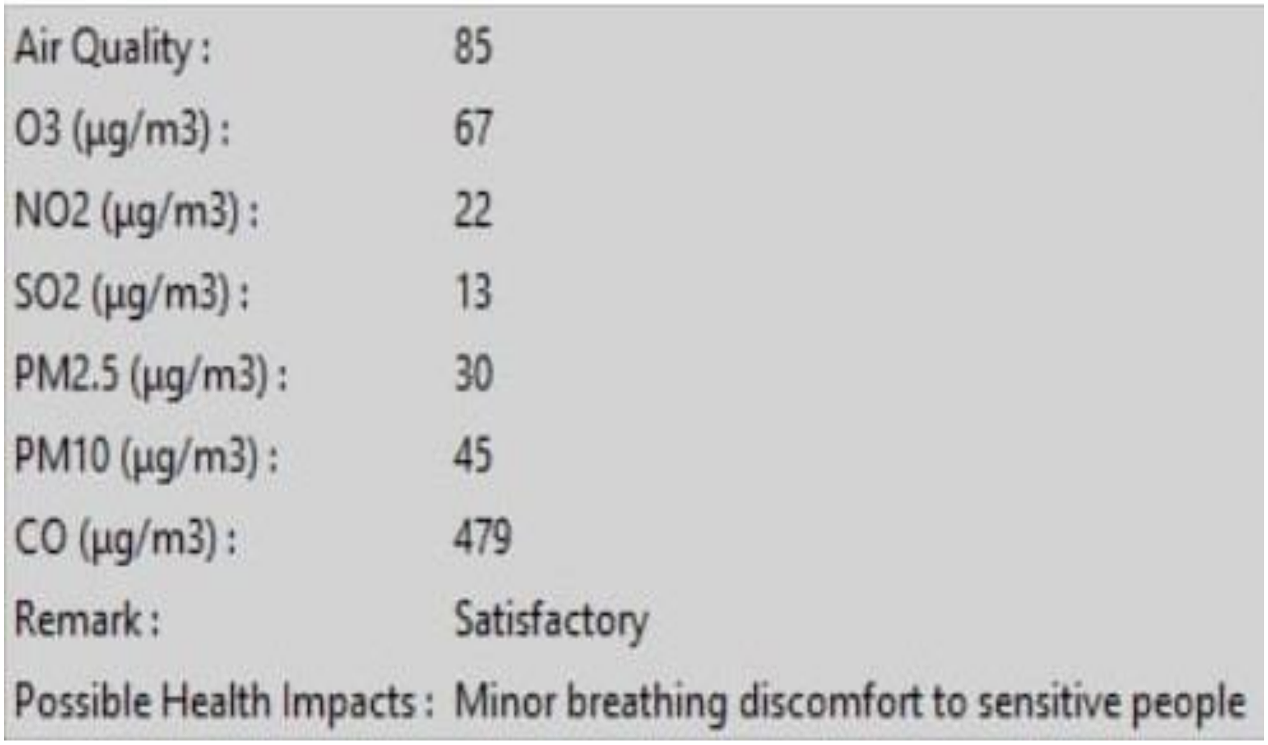
IOT.py - C:\Users\flower\AppData\Local\Programs\Python\Python311\IOT.py (3.11.4)

File Edit Format Run Options Window Help

```python
      bg="light grey").grid(row=5, sticky=W)
Label(master, text="CO (µg/m3) :",
      bg="light grey").grid(row=6, sticky=W)
Label(master, text="Remark :",
      bg="light grey").grid(row=7, sticky=W)
Label(master, text="Possible Health Impacts :",
      bg="light grey").grid(row=8, sticky=W)
# Creating label for class variable
# name using widget Entry
Label(master, text="", textvariable=ar,
      bg="light grey").grid(
    row=0, column=1, sticky=W)
Label(master, text="", textvariable=o3,
      bg="light grey").grid(
    row=1, column=1, sticky=W)
Label(master, text="", textvariable=no2,
      bg="light grey").grid(
    row=2, column=1, sticky=W)
Label(master, text="", textvariable=so2,
      bg="light grey").grid(
    row=3, column=1, sticky=W)
Label(master, text="", textvariable=pm,
      bg="light grey").grid(
    row=4, column=1, sticky=W)
Label(master, text="", textvariable=pml,
      bg="light grey").grid(
    row=5, column=1, sticky=W)
Label(master, text="", textvariable=co,
      bg="light grey").grid(
    row=6, column=1, sticky=W)
Label(master, text="", textvariable=res_remark,
      bg="light grey").grid(row=7, column=1, sticky=W)
Label(master, text="", textvariable=res_imp,
      bg="light grey").grid(row=8, column=1, sticky=W)
# creating a button using the widget
b = Button(master, text="Check",
           command=airinfo, bg="Blue")
b.grid(row=0, column=2, columnspan=2,
       rowspan=2, padx=5, pady=5,)
mainloop()
```

Ln: 99 Col: 54

## Output:



```
Air Quality :                85
O3 (µg/m3) :                 67
NO2 (µg/m3) :                22
SO2 (µg/m3) :                13
PM2.5 (µg/m3) :              30
PM10 (µg/m3) :               45
CO (µg/m3) :                 479
Remark :                     Satisfactory
Possible Health Impacts :    Minor breathing discomfort to sensitive people
```

## Conclusion:

- Air quality monitoring systems are designed using different sensors for indoor and outdoor air quality monitoring in the previous works by using Bluetooth, GPS, GPRS wireless technologies. In a previous work WASP module is used which is costly. Instead of that different sensors can be used. The proposed system is developed for indoor air quality monitoring remotely. It is cost and energy efficient request and respond protocol is used along with combination of address and data centric protocols. Paper presents the summary of various techniques of air quality monitoring. These techniques are elaborately discussed in the paper. In the proposed system, one of the most preferred technique is cloud based air quality monitoring system. Using the same cloud data, website is hosted and data is displayed on the website