**Air quality monitoring**

**Phase 3**

Introduction:

In this phase we are going to include the working process of air quality monitoring system with the help of source code as python program that is able to display the concepts of the air quality in the atmosphere and and tell about the levels of air and can be determined whether it is in good condition or else bad or worst.

According to the quality of air we can further proceed for the other steps to minimise the harmful gases in atmosphere and try to increase the useful gases like oxygen content in the atmosphere by planting a trees or plants as a responsible person to our environment.

The monitoring system measures all major ambient parameters such as PM1, PM2. 5, PM10, PM100, Carbon Monoxide (CO), Carbon Dioxide (CO2), Sulphur Dioxide (SO2), Nitrogen Dioxide (NO2), Ozone (O3), Hydrogen Sulfide (H2S), Ambient Noise, Light, UV, Temperature, and Humidity.

An IoT based air pollution monitoring system consists of several hardware and software components that work together to collect and process data. The hardware components include sensors, microcontrollers, and communication modules. The software components consist of a cloud platform, a mobile application, and a web-based dashboard.

The system can also alert the users if the air quality reaches a dangerous level, allowing them to take precautions to protect themselves.

***Usage of Monitoring System:***

The IoT-based air pollution monitoring system can be used in various settings, including residential, industrial, and urban areas. It can also be integrated with existing air pollution monitoring systems to enhance their capabilities. The system can provide valuable data to government agencies, researchers, and the public to make informed decisions about air pollution.

**Python Source code for air quality monitoring system :**

import requests

import json

import random

import time

Api\_key = "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"

City = "YourCity"

Country = "YourCountry"

APIurl="http://api.openweathermap.org/data/2.5/air\_pollution?lat=0&lon=0&appid={api\_key}"

response = requests.get(url)

if response.status\_code == 200:

data = response.json()

Air\_quality = data["list"][0]["main"]

print(f"Air Quality in {city}, {country}:")

print(f" - AQI (Air Quality Index): {air\_quality[‘aqi’]}")

print(f" – Particulate Matter (PM2.5): {air\_quality[‘pm2\_5']} µg/m³")

print(f" – Particulate Matter (PM10): {air\_quality[‘pm10’]} µg/m³")

print(f" – Carbon Monoxide (CO): {air\_quality[‘co’]} µg/m³")

print(f" – Nitrogen Dioxide (NO2): {air\_quality[‘no2’]} µg/m³")

print(f" – Ozone (O3): {air\_quality[‘o3’]} µg/m³")

print(f" – Sulfur Dioxide (SO2): {air\_quality[‘so2’]} µg/m³")

class AirQualityMonitor:

def \_\_init\_\_(self, location):

self.location = location

self.pm25 = 0

self.pm10 = 0

def measure\_air\_quality(self):

self.pm25 = random.uniform(0, 100)

Self.pm10 = random.uniform(0, 150)

def display\_data(self):

print(f"Location: {self.location}")

print(f"PM2.5: {self.pm25} µg/m³")

print(f"PM10: {self.pm10} µg/m³")

print("----------------------------")

if \_\_name\_\_ == "\_\_main\_\_":

outdoor\_sensor = AirQualityMonitor("Outdoor Sensor")

indoor\_sensor = AirQualityMonitor("Indoor Sensor")

while True:

outdoor\_sensor.measure\_air\_quality()

indoor\_sensor.measure\_air\_quality()

outdoor\_sensor.display\_data()

indoor\_sensor.display\_data()

# Simulate data collection every 5 seconds

time.sleep(5)

else:

print("Failed to retrieve air quality data")

def air\_quality(aqi):

if 0 <= aqi <= 50:

return "Good"

elif 51 <= aqi <= 100:

return "Moderate"

elif 101 <= aqi <= 150:

return "Unhealthy for Sensitive Groups"

elif 151 <= aqi <= 200:

return "Unhealthy"

else:

return "Hazardous"

Aqi\_value = 75

Result = air\_quality(aqi\_value)

print(f"The air quality is {result}")

**Sample Output**:

Air Quality in YourCity, YourCountry:

- AQI (Air Quality Index): 24

- Particulate Matter (PM2.5): 3 µg/m³

- Particulate Matter (PM10): 5 µg/m³

- Carbon Monoxide (CO): 200 µg/m³

- Nitrogen Dioxide (NO2): 10 µg/m³

- Ozone (O3): 30 µg/m³

- Sulfur Dioxide (SO2): 1 µg/m3

```

Oxygen state: Good

Carbon Dioxide state: Good

Carbon Monoxide state: Bad

Location: Outdoor Sensor

PM2.5: 35.245 µg/m³

PM10: 78.119 µg/m³

Location: Indoor Sensor

PM2.5: 12.874 µg/m³

PM10: 93.572 µg/m³

This output suggests that oxygen and carbon dioxide concentrations are in a “Good” state, while carbon monoxide is in a “Bad” state, based on the simple concentration thresholds used in the code. The classification can change depending on the actual gas concentrations and the specific criteria used to define “Good,” “Bad,” or “Worst” stages.

Conclusion:

An IoT-based indoor air quality monitoring platform based on integration of cloud computing and IoT is presented in this research. Also, a device called “Smart-Air” was developed to precisely monitor indoor air quality and efficiently transmit real time data to a cloud computing-based web server using an IoT sensor network.

The cloud computing based web server introduced in this platform analyzes real-time data and adds visual effects to illustrate the conditions of the indoor air quality.

In addition, the web server was designed to issue alert mobile application users or facility managers of moderate or poor air quality so that responsible parties can take immediate remedial action.

Real-time monitoring and a rapid alert system produce an efficient platform for improving indoor air quality. Major contributions of the proposed study are as follows:

1. We propose the use of the Smart-Air for the precise monitoring of indoor air quality
2. We propose the utilization of an IoT for efficient monitoring of real-time data
3. We propose the adoption of cloud computing for real-time analysis of indoor air quality

(iv) We originally developed a mobile application to make the proposed IoT system with features of anytime, anywhere

1. The device has been tested for reliability of the data and the platform has been implemented in a building to test its feasibility.