KINGS ENGINEERING COLLEGE

PROJECT TITTLE: AIR QUALITY MONITORING

DEPARTMENT: BIO MEDICAL ENGINEERING

MENTOR NAME:MARY LALITHA

TEAM MEMBERS:

S.VEERA GANESH

N.VIGNESHWARAN

M.VISHWA

S.VIGNESH

N.VETRIVEL

BUILDING THE IOT AIR QUALITY MONITORING SYSTEM

HARDWARE COMPONENTS:

Air Quality Sensors :Choose sensors that can measure various air quality parameters like particulate matter (PM2.5, PM10), carbon dioxide (CO2), volatile organic compounds (VOCs), and others.
Microcontroller: Select a suitable microcontroller like Arduino, Raspberry Pi, or ESP8266/ESP32 to collect data from sensors.
Connectivity Module: Use Wi-Fi, Bluetooth, or other connectivity options to transmit data to the cloud.

Software and Firmware:

Write firmware for your microcontroller to read data from sensors. Implement communication protocols (MQTT, HTTP, or others) to send data to the cloud.Develop software for data processing, storage, and visualization.

CLOUD PLATFORM:

Choose a cloud platform (AWS, Azure, Google Cloud, or IoT-specific platforms like Adafruit IO) for data storage and management. Set up databases to store sensor data.Create APIs to access and manage data.

DATA VISUALIZATION:

Build a web or mobile application to display real-time and historical data. Use tools like Grafana or Plotly for data visualization.

ALERTING SYSTEM:

Implement threshold-based alerts to notify users when air quality parameters exceed safe levels.

USER INTERFACE:

Design a user-friendly interface for users to access air quality data. Include features for historical data analysis and data export.

Power Management:

Ensure efficient power management to prolong the life of your IoT device.

Security:

Implement security measures to protect data transmission and storage.

TESTING AND CALIBRATION:

Calibrate sensors for accurate readings. Test the system in various environmental conditions. Install the IoT devices in the target locations. Regularly update firmware and software.
Monitor the health of IoT devices. Use data analytics to derive insights from collected data.

CONFIGURING IOT DEVICES(E ,G: SENSOR FOR POLLUTION LEVEL, PARTICULAR MATTER) TO MEASURE AIR QUALITY PARAMETERS INVOLVES SEVERAL STEPS:

SELECT APPROPRIATE SENSORS:

Choose sensors that can measure various air quality parameters, such as particulate matter (PM2.5, PM10), volatile organic compounds (VOCs), carbon dioxide (CO2), carbon monoxide (CO), nitrogen dioxide (NO2), and ozone (O3). Make sure the sensors are compatible with your IoT platform.

IOT PLATFORM SELECTION:

Choose an IoT platform that supports the sensors you've selected. Popular options include AWS IoT, Microsoft Azure IoT, Google Cloud IoT, and various open-source platforms like MQTT and CoAP.

HARDWARE SETUP:

Connect the selected sensors to the IoT device (e.g., Raspberry Pi, Arduino, or specialized IoT development boards).Ensure the power supply and connectivity (Wi-Fi, Ethernet, or cellular) for the IoT device.

PROGRAMMING:

Write or use existing code to collect data from the sensors. This often involves interfacing with the sensors using appropriate libraries or protocols. Implement error handling and data processing to ensure accurate readings.

DATA TRANSMISSION:

Configure the IoT device to transmit data to your chosen IoT platform. This may involve setting up MQTT or HTTP communication.

DATA STORAGE AND ANALYSIS:

Set up cloud-based or on-premises data storage for collected air quality data.

Configure data analysis tools and dashboards to visualize and interpret the data.

SECURITY:

Implement security measures to protect data during transmission and storage. Use encryption and secure protocols.

REMOTE MONITORING:

Enable remote monitoring and control of the IoT device for troubleshooting and maintenance.

CALIBRATION AND MAINTENANCE:

Regularly calibrate the sensors to ensure accurate measurements. Plan maintenance routines to replace or clean sensors as needed.

COMPLIANCE AND REGULATIONS:

Ensure compliance with local air quality regulations and standards, and consider sharing data with relevant authorities if necessary.

USER INTERFACE:

Develop a user interface or mobile app for end-users to access air quality data, if applicable.

ALERTS AND NOTIFICATIONS:

Implement alert systems to notify users or authorities when air quality parameters exceed predefined thresholds.

TESTING:

Thoroughly test the system to ensure accurate and reliable data collection.

DEVELOP A PYTHON SCRIPT ON THE IOT DEVICE TO SEND COLLECTED DATA TO THE DATA SHARING PLARTFORM

1.SET UP YOUR ENVIRONMENT:

Ensure that you have python installed your iot device. Install necessary library for your sensors and MQTT.

2.LIBRARY REQUIRED:

Python code

```
import requests
import json
import time


# Define your IoT device's parameters
device_id = "your_device_id"
data_sharing_url = " https://api.datasharingplatform.com/data_"
headers = {"Content-Type": "application/json"}


# Create a function to read sensor data (replace with actual sensor code)
def read_sensor_data():
    # Replace this with code to read sensor data
    sensor_data = {
        "temperature": 25.5,
        "humidity": 50.2,
        "pm2.5": 10.3,
    }
    return sensor_data
```

```
# Create a function to send data to the data-sharing platform
def send_data_to_platform(data):
    try:
        response = requests.post(data_sharing_url, data=json.dumps(data), headers=headers)
        response.raise_for_status()
        print("Data sent successfully.")
    except requests.exceptions.RequestException as e:
        print(f"Failed to send data: {str(e)}")

# Main loop to continuously send data
while True:
    sensor_data = read_sensor_data()
    send_data_to_platform(sensor_data)
    time.sleep(60)  # Adjust the interval as needed
```

Explanation:

1. Import the `requests` library for making HTTP requests and other necessary modules.

2. Define your IoT device parameters, such as `device_id`, the `data_sharing_url` (the URL of your data-sharing platform's API endpoint), and headers for the HTTP request.

3. Create a function `read_sensor_data()` to read data from your sensors. Replace the placeholder data with actual sensor readings.

4. Create a function `send_data_to_platform(data)` to send the collected data to the data-sharing platform. It uses the `requests.post` method to make an HTTP POST request with the sensor data in JSON format.

5. In the main loop, continuously read sensor data and send it to the data-sharing platform using the `send_data_to_platform()` function. You can adjust the interval to control how often data is sent.

Make sure to replace the placeholder values with your actual device and data-sharing platform details. Additionally, handle exceptions and errors appropriately in your actual implementation.