NOISE POLLUTION MONITORING

PHASAE 5 : PROJECT DOCUMENTATION AND SUBMISSION

Project Objectives:

The Noise Pollution Monitoring project aims to create a real-time noise level monitoring system that collects data from IoT sensors, processes the information on a central platform, and presents it to users through a mobile app. The project's objectives are as follows:

Deploy IoT Sensors: Set up a network of IoT sensors to monitor noise levels at various locations.

Platform Development: Create a noise pollution information platform for collecting, storing, and processing sensor data.

Mobile App Development: Develop a mobile application that allows users to access real-time noise level information.

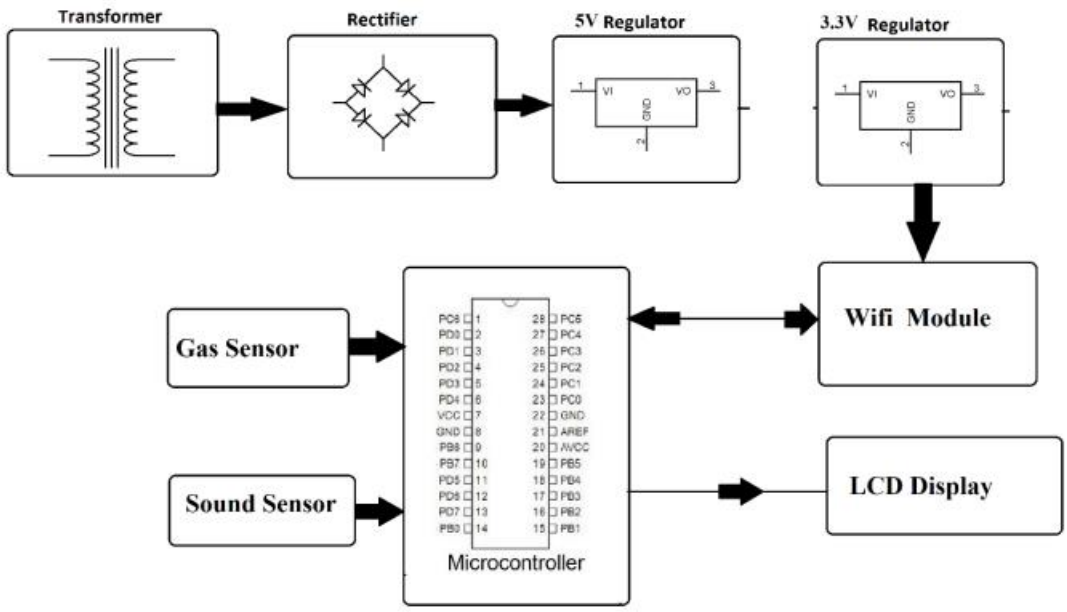
Code Implementation: Implement the software components to connect IoT sensors, the platform, and the mobile app.

IoT Sensor Deployment:

The IoT sensor deployment involves placing noise level sensors at strategic locations throughout the target area. These sensors continuously monitor the ambient noise levels and transmit data to the central platform for processing.

Diagram:

IoT sensor deployment diagram



Platform Development:

The noise pollution information platform serves as the central hub for data collection, storage, and processing. It should include the following components:

Data Ingestion: Collect data from IoT sensors.

Data Processing: Analyse and store the noise level data.

Database: Store historical and real-time data for analysis.

Dashboard: Provide an interface for data visualization.

Alerts: Notify administrators and users of critical noise level conditions.

Mobile App Development:

The mobile app is designed to give users easy access to real-time noise level information, historical data, and alert notifications.

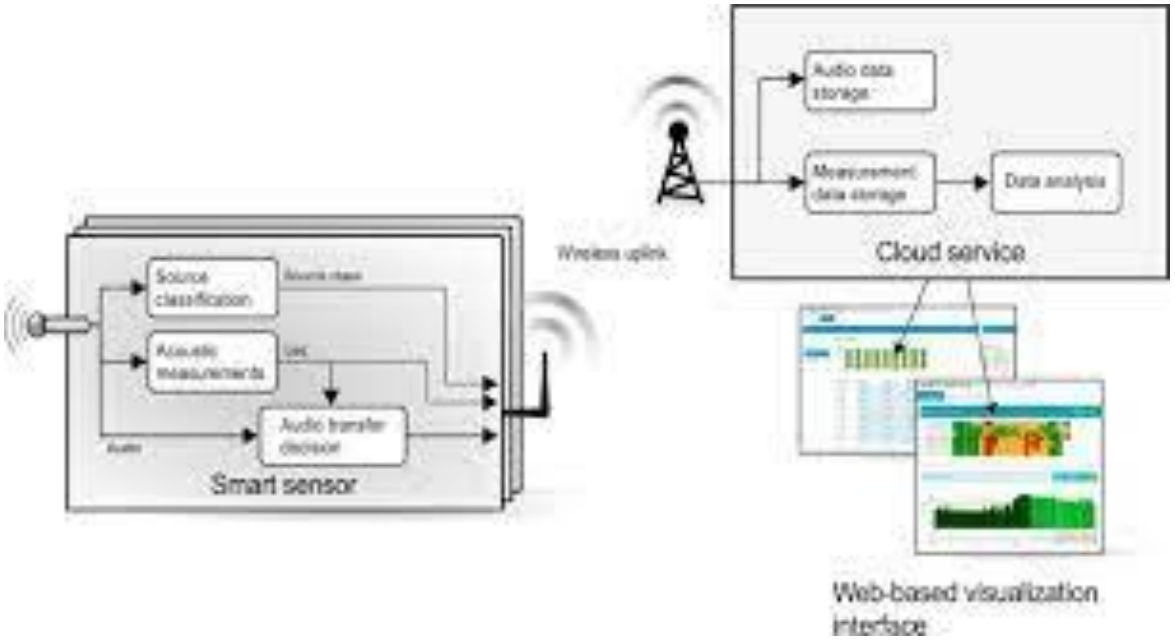
Real-time Data: Display current noise levels at various sensor locations.

Historical Data: Allow users to view past noise level trends.

Alerts: Push notifications for noise level alerts.

User Profiles: Customizable settings for users.

Map Integration: Display sensor locations on a map.



Code Implementation:

The code implementation involves developing the software components necessary for data transmission, processing, and presentation. The code is typically written in Python and includes functions for:

IoT Sensor Data Transmission: Code for

Python code snippet for the IoT sensor data transmission part of the project. This code assumes that you have a sensor that measures noise levels and is capable of transmitting data over a network. In practice, you would need to adapt this code to your specific IoT sensor hardware and communication protocol.

IoT Sensor Data Transmission (Python code):

```
import requests
import random
import time

# Simulated IoT sensor data
sensor_id = 1
location = "Sensor Location"
base_url = "https://your-platform-api-url.com/data"

while True:
    noise_level = random.randint(40, 90)  # Simulated noise level reading
    data = {
        "sensor_id": sensor_id,
        "location": location,
        "noise_level": noise_level
    }

    # Send data to the platform
    response = requests.post(base_url, json=data)
    if response.status_code == 200:
```

```
        print(f"Data sent: {data}")
    else:
        print(f"Failed to send data: {response.status_code}")

    time.sleep(60)  # Send data every minute
```

Platform Data Processing (Python code):

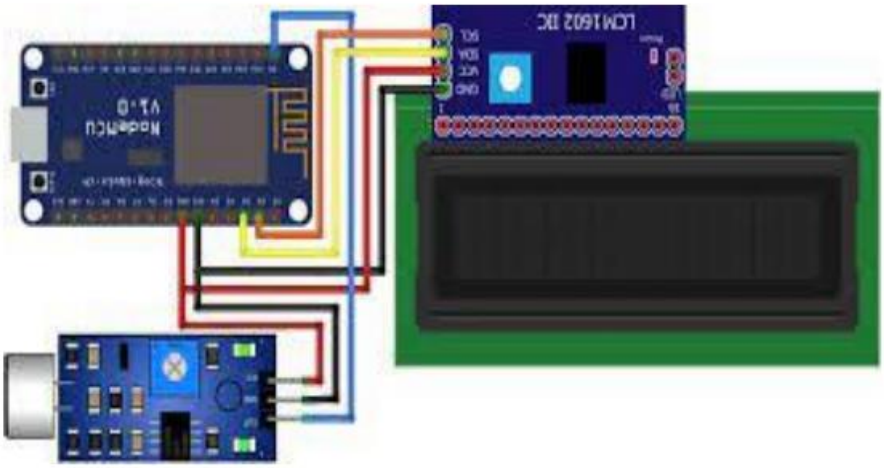
```
from flask import Flask, request, jsonify
import sqlite3

app = Flask(__name__)

# SQLite database to store sensor data
conn = sqlite3.connect("sensor_data.db")
cursor = conn.cursor()
cursor.execute('''
    CREATE TABLE IF NOT EXISTS sensor_data (
        id INTEGER PRIMARY KEY,
        sensor_id INTEGER,
```

```
        location TEXT,
        noise_level INTEGER,
        timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
    )
''')
conn.commit()

@app.route('/data', methods=['POST'])
def receive_sensor_data():
    data = request.json
    sensor_id = data["sensor_id"]
    location = data["location"]
    noise_level = data["noise_level"]

    # Store sensor data in the database
    cursor.execute(" INSERT INTO sensor_data (sensor_id, location, noise_level)
        VALUES (?, ?, ?)
    ''', (sensor_id, location, noise_level))
    conn.commit()

    return "Data received", 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

IOT Sensors:

Arduino Nano

Design of Noise Pollution Monitoring System:



Instructions to Replicate the Project:

IoT Sensor Deployment:

Acquire noise level sensors compatible with your hardware and software stack.

Install the necessary libraries and dependencies on the sensors for data transmission.

Configure each sensor with a unique ID and location information.

Platform Development:

Set up a server to host the noise pollution information platform. You can use cloud-based services or on-premises solutions.

Create a database to store sensor data. You can use a relational database like SQLite or a more scalable option like PostgreSQL.

Implement the Flask web server code (as provided in the previous response) to receive sensor data.

Develop data processing and visualization components as per your project requirements.

Mobile App Development:

Install the necessary development tools for mobile app development (e.g., Android Studio for Android apps or X code for iOS apps).

Develop a mobile app that can connect to the platform API to fetch and display real-time and historical noise level data.

Implement features such as user authentication, map integration, and push notifications.

Integration Using Python:

Write Python code to facilitate communication between the IoT sensors, the platform, and the mobile app.

Use Python libraries like Requests for sending and receiving data from IoT sensors and platforms.

IoT Sensor Data Transmission:

Provide example console logs or data transmission records from your IoT sensors, demonstrating data sent to the platform.

Platform UI:

Include screenshots of the platform's user interface, showing real-time noise level data, historical trends, and alerts.

Mobile App Interfaces:

Share screenshots or recordings of the mobile app interfaces displaying real-time noise levels, map integration, and user settings.

Please replace the placeholders with your actual GitHub repository link, detailed instructions specific to your project, and real example outputs for the Noise Pollution Monitoring project.