NOISE POLLUTION MONITORING

**Phase 3 – Development Part 1**

**HARDWARE SETUP**

1. **Select Hardware:**

• Choose a hardware platform for audio data acquisition, such as Raspberry Pi, Arduino, or similar platforms.

• Connect a suitable microphone or sensor for noise measurement to your chosen hardware.

2. **Set Up Your Development Environment:**

• Install the necessary development tools for your chosen hardware (e.g., Arduino IDE, Python for Raspberry Pi).

3. **Capture Audio Data:**

• Use libraries and code to capture audio data from the microphone or sensor.

For Raspberry Pi, you can use Python and the pyaudio library for audio input:

**pythonCopy code**

**import pyaudio**

**import wave**

**# Set up audio input**

**audio = pyaudio.PyAudio()**

**stream = audio.open(format=pyaudio.paInt16, channels=1, rate=44100, input=True, frames\_per\_buffer=1024)4.**

4. **Preprocess the Audio Data:**

• You might need to filter, normalize, or perform other preprocessing on the audio data, depending on your specific requirements.

5. **Analyze the Audio Data:**

• Use signal processing and analysis libraries to process the audio data and calculate noise levels or other relevant metrics.

For example, you can use Python's numpy and scipy libraries for signal processing:

pythonCopy code

**import numpy as np**

**from scipy.signal import decimate**

**# Example signal processing (e.g., decimation)**

**signal = stream.read(1024)**

**signal = np.frombuffer(signal, dtype=np.int16)**

**decimated\_signal = decimate(signal, q=10)**

6. **Store and Visualize Data:**

• Store the data in a database or file for future analysis and visualization.

For data storage, you can use databases like SQLite or MongoDB. For visualization, libraries like matplotlib can be used to create charts and graphs.

7. **Implement Noise Thresholds**:

• Set noise level thresholds to trigger alerts or store data when certain noise levels are exceeded.

8. **Continuous Monitoring:**

• Implement a loop to continuously monitor and record noise levels.

9. **Alerting (Optional):**

• Implement alerts or notifications when noise levels exceed predefined thresholds. You can use libraries like smtplib for email notifications or messaging services for alerts.

10. **Data Management and Analysis**:

• Use libraries like pandas for data analysis and generating reports based on the recorded noise data