**PROJECT TITLE: NOISE POLLUTION MONITORING**

**Phase 3: Development Part 1**

**INTRODUCTION:**

Noise pollution, a pervasive and often overlooked environmental issue, can have detrimental effects on health, well-being, and the quality of life in urban and even rural areas. To address this concern and promote awareness, we introduce a Raspberry Pi-based noise pollution monitoring project. This project utilizes the capabilities of the Raspberry Pi, coupled with a USB microphone and Python programming, to continuously monitor and analyze ambient noise levels.The Raspberry Pi, a versatile single-board computer, is an ideal platform for this endeavor due to its affordability, low power consumption, and ease of integration with various sensors and peripherals. By deploying this solution, individuals, communities, or organizations can gain valuable insights into the noise pollution in their surroundings, allowing them to make informed decisions, take corrective actions, and advocate for a quieter environment.

**REQUIRED COMPONENTS:**

* Raspberry pi
* Microphone
* Audio interface
* Software
* Noise monitoring software
* Calibration
* Data storage
* Power supply
* Internet connectivity
* Environmental enclosure
* Optional sensors
* Display
* Allerting system
* Regulatory compliance

**IOT DEVICES:**

**Raspberry Pi:** The core computing platform for data processing and IoT connectivity.

Microphone Sensor: Use a high-quality microphone sensor for capturing audio data. You may need additional components to interface it with the Raspberry Pi.

**Environmental Sensors (Optional):** Consider adding environmental sensors to gather supplementary data, such as temperature, humidity, and air quality, which can help in analyzing the context of noise pollution.

**Analog-to-Digital Converter (ADC):** If your microphone outputs analog data, you'll need an ADC to convert it to digital format for processing by the Raspberry Pi.

**Internet Connectivity:** Enable the Raspberry Pi to connect to the internet, typically through Wi-Fi or Ethernet, to transmit data to a central server or cloud platform.

**Central Server or Cloud Platform:** You'll need a central server or cloud-based platform to receive, store, and analyze the data from your IoT device.

**Data Communication Module:** Utilize communication protocols like MQTT, HTTP, or WebSockets to send data from the Raspberry Pi to your central server or cloud

**Data Analysis and Visualization Platform:** Implement software on the central server or cloud platform to analyze and visualize the data collected from the Raspberry Pi.

**API Integration (Optional):** Integrate your IoT system with APIs or services that can enhance your monitoring capabilities, such as weather data for context.

**WORKING:-**

**Working**:- Monitoring noise pollution using a Raspberry Pi can be accomplished by setting up a system with a sound sensor and appropriate software. Here's a simplified block diagram of how it works:

**Sound Sensor:** This is the primary sensor that captures ambient noise levels. You can use a digital sound sensor like the "MAX4466" or an analog sensor like the "Electret Microphone" connected to the Raspberry Pi.

**Raspberry Pi:** The Raspberry Pi serves as the central processing unit of the system. It processes the data from the sound sensor and runs the software for noise monitoring.

**Analog-to-Digital Converter (ADC):** If you're using an analog sound sensor, you may need an ADC to convert the analog signal to a digital format that the Raspberry Pi can process.

**Amplifier (if needed):** In some cases, you might require an amplifier to boost the signal from the sound sensor for better accuracy.

**Processing Software:** The Raspberry Pi runs software to process the incoming sound data. This software usually involves signal processing techniques to calculate sound levels (in decibels, dB).

**Data Storage:** The monitored noise data can be stored locally on the Raspberry Pi or sent to an external server or cloud storage for long-term analysis.

**Display/Visualization:** You can add an LCD display or connect to a web-based dashboard for real-time monitoring and visualization of noise levels.

**Alarms/Alerts:** If the noise levels exceed predefined thresholds, the system can trigger alarms or alerts, which could be in the form of notifications or visual indicators.

**Power Supply:** Ensure a stable power supply for the Raspberry Pi and any additional components.

**Internet Connectivity:** If you want remote monitoring or data storage in the cloud, you'll need an internet connection via Ethernet or Wi-Fi.

**User Interface:** You can access noise data and configure settings via a web interface or mobile app, allowing users to control the system.

**Optional Battery Backup:** Consider adding a battery backup system to ensure data collection in case of power outages.

### CODE EXPLANATION:

import sounddevice as sd

import numpy as np

import time

# Set the sampling frequency and recording duration

fs = 44100 # Sample rate in Hz (you can adjust this)

duration = 10 # Duration of recording in seconds (you can adjust this)

# Function to measure noise level (you can adjust this based on your needs)

def measure\_noise\_level(audio\_data):

# Example: Calculate the root mean square (RMS) of the audio data

rms = np.sqrt(np.mean(np.square(audio\_data)))

return rms

# Function to monitor noise levels

def monitor\_noise():

print("Monitoring noise for {} seconds...".format(duration))

audio\_data = sd.rec(int(fs \* duration), samplerate=fs, channels=1, dtype='int16')

sd.wait()

noise\_level = measure\_noise\_level(audio\_data)

print("Noise level (RMS): {:.2f}".format(noise\_level))

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

while True:

monitor\_noise()

time.sleep(60) # Monitor noise every 60 seconds (adjust as needed)

**In this code:**

We use the sounddevice library to capture audio data from the USB microphone.The measure\_noise\_level function calculates the noise level from the recorded audio data. You can modify this function to use different noise measurement techniques or metrics.The monitor\_noise function records audio for a specified duration and prints the calculated noise level.The main loop continuously monitors noise levels with a specified interval

**CONCLUSION:**

In conclusion, noise pollution monitoring using a Raspberry Pi is a feasible and cost-effective solution for assessing and managing noise levels in various environments. This approach offers several advantages, including flexibility, scalability, and the ability to integrate with other systems.