NOISE POLLUTION MONITERING DEVICE BASED ON IOT

DEVELOPMENT OF NOISE POLLUTION MONITERING:

• Creating a noise pollution monitoring device based on the Internet of Things (IoT) involves integrating sensors, microcontrollers, and communication modules to collect and transmit noise data to a central server or cloud platform. Here's a step-by-step guide to help you build a basic version of such a device:

Components Needed:

1. **Noise Sensor:** Use a noise sensor (like the Sound Detection Sensor) to measure the noise levels in the environment.
2. **Microcontroller:** Raspberry Pi, Arduino, or ESP32 can be used to process sensor data and control the device.
3. **Communication Module:** Wi-Fi, GSM, or LoRa modules can be used for data transmission.
4. **Power Supply:** A power source for the device, which can be a battery or a power adapter.
5. **Enclosure:** A weatherproof enclosure to protect the components from environmental factors.
6. **Central Server:** A server or a cloud platform to store and analyze the collected data.

Steps to Build the Device:

1. **Connect the Noise Sensor:**

Connect the noise sensor to the microcontroller following the sensor's datasheet or manual.

2. **Program the Microcontroller:**

Write a program for the microcontroller to read data from the noise sensor. Use appropriate libraries and functions to calibrate the sensor readings into decibels (dB).

3. **Set Up Communication:**

Integrate the communication module with the microcontroller. If you're using Wi-Fi, GSM, or LoRa, set up the module to transmit data to the central server. Implement secure communication protocols to protect data integrity and user privacy.

4. **Power Supply and Enclosure:**

Power the device using a suitable power supply method (battery or adapter) and enclose all components in a protective enclosure. Ensure that the enclosure is weatherproof if the device will be placed outdoors.

5. **Central Server or Cloud Platform:**

Set up a central server or use a cloud platform (like AWS, Azure, or Google Cloud) to receive and store the data transmitted by the devices. Create a database to store the noise level data.

6. **Data Analysis and Visualization:**

Implement algorithms to analyze the noise data for patterns, peaks, and trends. You can use various tools and programming languages like Python, R, or JavaScript for data analysis. Visualize the data using charts or graphs for better understanding.

7. **User Interface (Optional):**

Create a user interface, either a web application or a mobile app, to allow users to monitor real-time noise levels and view historical data. The interface can communicate with the central server's API to fetch and display the data.

8. **Alerts and Notifications (Optional):**

Implement an alert system to notify users or authorities when noise levels exceed certain thresholds. This can be done via email, SMS, or push notifications on the user interface.

9. **Testing and Calibration:**

Thoroughly test the device in different environments to ensure accurate readings. Calibrate the sensors if necessary to improve accuracy.

10. **Deployment:**

Deploy the devices in the desired locations to start monitoring noise pollution. Regularly maintain and update the devices as needed.

Remember that building a robust IoT device involves not only hardware and software development but also considerations for data security, privacy, and scalability. It's essential to follow best practices in IoT development and security to create a reliable and secure noise pollution monitoring system.

Python script

```
import RPi.GPIO as GPIO

import time

import requests


SOUND_SENSOR_PIN = 18

SERVER_URL = "http://your_server_endpoint/api/noise"

def get_noise_level():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(SOUND_SENSOR_PIN, GPIO.IN)
    time.sleep(2)
    noise_level =0
    for _ in range(10):
        if GPIO.input(SOUND_SENSOR_PIN) == GPIO.LOW:
            noise_level += 1
        time.sleep(0.1)
     GPIO.cleanup()
    return noise_level
def send_to_server(noise_level):
```

```
    payload = {"noise_level": noise_level}
    try:
        response = requests.post(SERVER_URL, json=payload)
        if response.status_code == 200:
            print("Data sent successfully!")
        else:
            print("Failed to send data. Status code:", response.status_code)
    except Exception as e:
        print("Error occurred while sending data:", str(e))
def main():
    while True:
        noise_level = get_noise_level()
        print("Noise Level:", noise_level)
        send_to_server(noise_level)
        # Wait for some time before taking the next reading (e.g., every 1 minute)
        time.sleep(60)
```

PROJECT SUBMITTED BY:

NAME:K.Nithyanandam

REGISTER NO: 713921106038

TOPIC:NOISE POLLUTIONMONITERING DEVICE BASED ON IOT

MAIL ID:nithyanandamnithi600@gmail.com

NM ID:au713921106034

COLLEGE CODE: 7139

Type your text