**NOISE POLLUTION MONITORING**

* **PROCESS AND DATA:**

The noise pollution monitoring project involving an Arduino Uno, IoT connectivity, and ThingSpeak consists of the following key processes and data flow:

1. Data Collection:

- Arduino Uno is equipped with a sound sensor (microphone) to capture sound levels.

- The Arduino periodically reads the analog value from the sound sensor and converts it to a sound In a noise pollution monitoring project using IoT devices, you can use various sensors to measure and monitor noise levels. Some common sensors include:

1. Sound Sensor (Microphone): This sensor can capture sound levels and convert them into electrical signals, which can be processed to measure noise pollution.
2. Noise Dosimeter: A noise dosimeter measures the cumulative noise exposure over time. It’s useful for assessing long-term noise exposure in different locations.
3. Sound Level Meter: These devices measure the instantaneous sound pressure level in decibels (dB) and are commonly used for real-time noise monitoring.
4. Vibration Sensor: In addition to sound, vibrations can also contribute to noise pollution. Vibration sensors can detect and measure such disturbances.
5. GPS Module: To accurately record the location of noise measurements, a GPS module can be integrated into the IoT device.
6. Environmental Sensors: You can complement noise data with environmental data by including sensors for temperature, humidity, and air quality, which may help in understanding noise pollution patterns.
7. Wi-Fi or Cellular Module: IoT devices need a way to transmit data. Wi-Fi or cellular modules can be used to send noise data to a central database for analysis.

2. Arduino Sketch:

- The Arduino is programmed with a sketch (code) that collects sound data and sends it over the serial port to the connected computer.

- The sound level data is printed to the serial monitor, allowing you to verify that the sensor is working correctly.

3. Python Script:

- On the computer, a Python script is running to communicate with the Arduino Uno.

- The Python script establishes a serial connection with the Arduino (specified by the serial port, e.g., ‘COM3’).

4. Data Transmission:

- The Python script reads the sound level data from the Arduino over the serial connection.

- It formats this data and sends it to ThingSpeak, an IoT platform for data logging and visualization.

5. ThingSpeak Configuration:

- A ThingSpeak channel is set up with the necessary fields to store the sound level data.

- The channel is associated with an API Key that allows your Python script to post data to it.

6. Data Posting to ThingSpeak:

- The Python script constructs a URL with the API Key and the sound level data.

- It uses HTTP requests to post this data to ThingSpeak’s server. ThingSpeak stores this data in the specified fields of your channel.

7. Visualization and Analysis:

- In ThingSpeak, you can create visualizations, such as graphs or gauges, to monitor noise levels over time.

- You can also perform data analysis and set up alerts for specific noise level thresholds.

8. Monitoring and Assessment:

- Over time, the Arduino continues to collect sound data and send it to the computer.

- The Python script continuously posts this data to ThingSpeak for monitoring and assessment.

9. Document Creation:

- To fulfill project requirements, you should document the entire setup and share it for assessment. This documentation should include:

- A description of the project’s purpose and goals.

- Detailed hardware and software components used (Arduino, sensors, Python script, ThingSpeak).

- Code snippets (Arduino sketch and Python script).

- Explanations of data transmission and visualization processes.

- Any findings or insights from the collected noise data.

10. Assessment:

- The documentation is assessed to evaluate the effectiveness of the noise pollution monitoring system and your understanding of the project.

* **USED SENSORS:**

In a noise pollution monitoring project using IoT devices, you can use various sensors to measure and monitor noise levels. Some common sensors include:

1. Sound Sensor (Microphone): This sensor can capture sound levels and convert them into electrical signals, which can be processed to measure noise pollution.
2. Noise Dosimeter: A noise dosimeter measures the cumulative noise exposure over time. It’s useful for assessing long-term noise exposure in different locations.
3. Sound Level Meter: These devices measure the instantaneous sound pressure level in decibels (dB) and are commonly used for real-time noise monitoring.
4. Vibration Sensor: In addition to sound, vibrations can also contribute to noise pollution. Vibration sensors can detect and measure such disturbances.
5. GPS Module: To accurately record the location of noise measurements, a GPS module can be integrated into the IoT device.
6. Environmental Sensors: You can complement noise data with environmental data by including sensors for temperature, humidity, and air quality, which may help in understanding noise pollution patterns.
7. Wi-Fi or Cellular Module: IoT devices need a way to transmit data. Wi-Fi or cellular modules can be used to send noise data to a central database for analysis.

* **PYTHON SCRIPT:**
* Arduino Sketch (to collect noise data):

const int soundSensorPin = A0;

// Analog pin for the sound sensor

int soundValue = 0;

void setup() {

Serial.begin(9600);

}

void loop() {

soundValue = analogRead(soundSensorPin);

Serial.println(soundValue);

delay(1000); // Adjust the delay as needed

}

* Python Script (to read data from Arduino and send it to ThingSpeak):

import serial

import requests

# Define the serial port for Arduino communication

ser = serial.Serial('COM3', 9600)

# Change 'COM3' to your Arduino's serial port

# Define your ThingSpeak API Key and Channel ID

api\_key = 'H2CBAGK5AQYDK3ZV'

channel\_id = '2303540’

while True:

try:

# Read data from Arduino

data = ser.readline().decode().strip()

# Print the data (optional)

print(f"Sound Level: {data}")

# Send data to ThingSpeak

url = f'https://api.thingspeak.com/update? api\_key={api\_key}&field1={data}'

response = requests.get(url)

# Print the response from ThingSpeak (optional)

print(response.text)

except Exception as e:

print(f"Error: {e}")

# Adjust the delay as needed