# NOISE POLLUTION MONITORING

# PROJECT DOCUMENTATION

***Project overview:***

***Project title:*** Noise Pollution Monitoring

***Project phase:*** Phase 4-Development Part 2

***Introduction:***

* Monitoring noise pollution through IoT sensors provides real-time insights into environmental sound levels.
* These sensors leverage Internet of Things technology to collect, transmit, and analyze data, enabling a comprehensive understanding of noise patterns.
* This innovative approach enhances the efficiency of pollution control measures and promotes a healthier urban environment by pinpointing sources, trends, and potential mitigation strategies.

***Overview:***

* Noise pollution monitoring involves systematically measuring and assessing the levels of unwanted or harmful sounds in a given environment.
* This process aims to understand the extent, sources, and impacts of noise pollution.
* Advanced technologies, such as IoT sensors and sound mapping, enable real-time data collection and analysis.
* The gathered information helps authorities make informed decisions, implement effective regulations, and develop strategies to mitigate the adverse effects of noise pollution on public health and well-being.

***Code explanation:***

Python code for noise pollution monitoring using an IoT sensor. The code uses the MQTT protocol for communication between the sensor and a central server. I’ve included comments for explanation:

***1.Import Libraries:***

Import the necessary libraries, including `paho.mqtt.client` for MQTT communication.

Code:

# Import necessary libraries

Import paho.mqtt.client as mqtt

Import json

Import time

***2.MQTT Settings:***

Configure the MQTT broker details (address and port) and the topic for noise data.

Code:

# MQTT settings

Mqtt\_broker = “your\_broker\_address”

Mqtt\_port = 1883

Mqtt\_topic = “noise\_data”

***3.Dummy Noise Sensor Data:***

Define a sample noise data payload with a sensor ID and decibel level.

Code:

# Dummy Noise Sensor Data

Noise\_data = {

“sensor\_id”: 1,

“decibel\_level”: 75

}

***4.MQTT Callbacks:***

Implement functions for connection (`on\_connect`) and message reception (`on\_message`).

Code:

# Callback function when connected to the MQTT broker

Def on\_connect(client, userdata, flags, rc):

Print(“Connected with result code “+str(rc))

# Subscribe to the noise data topic

Client.subscribe(mqtt\_topic)

# Callback function when a message is received from the MQTT broker

Def on\_message(client, userdata, msg):

# Parse the received JSON payload

Payload = json.loads(msg.payload)

# Process the received noise data

Process\_noise\_data(payload)

***5.Process Noise Data:***

Create a function (`process\_noise\_data`) to extract and process sensor ID and decibel level from the received data. You can add your noise pollution analysis or actions in this function.

Code:

# Function to process noise data

Def process\_noise\_data(data):

# Extract sensor ID and decibel level from the received data

Sensor\_id = data[“sensor\_id”]

Decibel\_level = data[“decibel\_level”]

# Add your noise pollution analysis or actions here

Print(f”Received Noise Data from Sensor {sensor\_id}: {decibel\_level} dB”)

***6.MQTT Client Setup:***

Set up an MQTT client, assign the defined call-back’s, and connect to the MQTT broker.

Code:

# Set up MQTT client and assign callbacks

Client = mqtt.Client()

Client.on\_connect = on\_connect

Client.on\_message = on\_message

# Connect to the MQTT broker

Client.connect(mqtt\_broker, mqtt\_port, 60)

***7. Data Simulation:***

Simulate sending noise data to the broker every 5 seconds (replace with actual sensor data). This is done in a loop, allowing continuous monitoring.

# Simulate sending noise data every 5 seconds (replace with actual sensor data)

Code:

While True:

# Publish noise data to the MQTT topic

Client.publish(mqtt\_topic, json.dumps(noise\_data))

# Allow time for the message to be sent

Time.sleep(5)

***Output:***

Connected with result code 0

Received Noise Data from Sensor 1: 75 dB

Received Noise Data from Sensor 1: 75 dB

Received Noise Data from Sensor 1: 75 dB

***1.Connected with result code 0:***

Indicates a successful connection to the MQTT broker.

***2.Received Noise Data from Sensor 1: 75 dB:***

Simulated output showing that the server receives noise data from the sensor every 5 seconds. The actual decibel level may vary based on the sample data or real sensor data.

***Conclusion:***

The provided Python code demonstrates a simple implementation for noise pollution monitoring using an IoT sensor. The program utilizes the MQTT protocol for communication between the sensor and a central server.

* MQTT Communication
* Sensor Data Simulation
* Data Processing
* Continuous Monitoring
* Placeholder Values
* Extensibility

Remember, the effectiveness of the noise pollution monitoring system depends on the accuracy and reliability of the sensor data and the sophistication of the data processing and analysis implemented on the central server.