PROJECT SUBMISSION FORM – PHASE 3

INTERNET OF THINGS (IoT)

PROJECT TITLE : NOISE POLLUTION MONITORING

OBJECTIVE:

The objective for IoT-based noise pollution monitoring is to continuously track noise levels, alert for excesses, identify sources, and provide data for analysis and policy decisions while being user-friendly, energy-efficient, and compliant with regulations.

IMPLEMENTATION:

Implementing IoT-based noise pollution monitoring involves deploying sensors to collect real-time data, centralizing data processing, and providing user-friendly displays. Alerts for threshold breaches and machine learning for noise source identification enhance the system. Historical data aids in trend analysis, and community engagement ensures effectiveness. Adherence to regulations and energy-efficient sensor design are vital for sustainability. Regular updates based on feedback and technology advancements are prioritized.

1. IOT SENSORS:

Several IoT sensors and devices can be used for noise pollution monitoring. These sensors are designed to measure sound levels and collect data for analysis. Some common IoT sensors for noise pollution monitoring include:

Microphone Sensors: These are the most common sensors used for sound and noise monitoring. They capture acoustic signals and convert them into electrical signals for analysis.

Sound Level Meters: These are specialized sensors designed for measuring noise levels in decibels (dB). They often come with built-in data logging capabilities.

MEMS (Micro-Electro-Mechanical Systems) Microphones: These are small, highly sensitive microphones often used in IoT applications due to their compact size and low power consumption.

Noise Dosimeters: These wearable devices are often used to monitor noise exposure for individuals over extended periods, making them suitable for occupational noise monitoring.

2. IOT DEVICES:

Acoustic Arrays, Wireless Acoustic Sensor Networks, Ultrasound Sensors, Vibration Sensors ,Environmental Sensors, Smartphone

3. POWER SUPPLY:

Power supply for IoT devices used in noise pollution monitoring typically includes battery-powered options, solar panels for outdoor applications, and wired power for urban or indoor locations. Energy harvesting and low-power design may also be utilized for extended device operation.

4. INTERNET CONNECTIVITY:

Internet connectivity for noise pollution monitoring IoT devices is primarily achieved through Wi-Fi, cellular networks, or LPWAN (Low-Power Wide Area Network) technologies like LoRaWAN. Wired Ethernet connections may be used for stationary devices. Connectivity choices depend on factors such as range, data transmission frequency, and geographic location.

5. IOT PLATFORMS:

Common IoT platforms for noise pollution monitoring include AWS IoT, Azure IoT, and Google Cloud IoT Core. These platforms offer cloud-based data storage, real-time analytics, and device management, simplifying the deployment and management of IoT sensors and data.

Developing a Python script on the IoT sensors to send real-time noise level data to the noise pollution information platform.

Developing a python script on the IOT sensors to send real-time noise level data to the noise pollution platform includes the following steps:

Set Up IoT Hardware:

Choose an appropriate noise sensor or microphone sensor.

Connect the sensor to a microcontroller (e.g., Raspberry Pi, Arduino, ESP8266/ESP32) or an IoT development board.

Install Required Libraries:

Install necessary Python libraries for sensor data acquisition and network communication. Common libraries include numpy, sounddevice for data collection, and paho-mqtt or requests for data transmission.

Collect Noise Data:

Write code to collect real-time noise level data from the sensor. This typically involves using the sensor's data acquisition methods.

Implement Data Preprocessing (if required):

Depending on your sensor and data requirements, you may need to preprocess the data. This could include calibration, filtering, or aggregation.

Choose a Communication Protocol:

Decide on the communication protocol to use for sending data to the IoT platform. Common options include MQTT, HTTP/HTTPS, or other IoT-specific protocols.

Implement Data Transmission:

Write code to transmit the collected noise data to the noise pollution information platform using the chosen protocol. Ensure that you have the platform's access credentials and endpoint URLs.

Error Handling and Data Integrity:

Implement error handling mechanisms to ensure data integrity during transmission. This includes handling network issues and retries.

Real-time Data Transmission:

Set up a loop or timer to send data at regular intervals to achieve real-time monitoring.

Security:

Implement security measures, such as encryption and authentication, to protect the data being transmitted.

Testing and Debugging:

Test the script with the IoT sensor to ensure that data is collected and transmitted correctly. Monitor the script for any errors or issues.

Deployment:

Deploy the IoT sensor in the target environment where noise monitoring is required.

Monitoring and Maintenance:

Continuously monitor the sensor's performance and the data being transmitted. Implement maintenance procedures, including sensor calibration and battery replacement.

Integration with the IoT Platform:

Ensure that the IoT platform is configured to receive and process data from your sensors. This may involve setting up data ingestion pipelines and creating dashboards or alerts for data visualization and analysis.

Scaling and Optimization:

If necessary, replicate the sensor setup and optimize the Python script for scalability. Consider power-saving techniques for extended sensor operation.

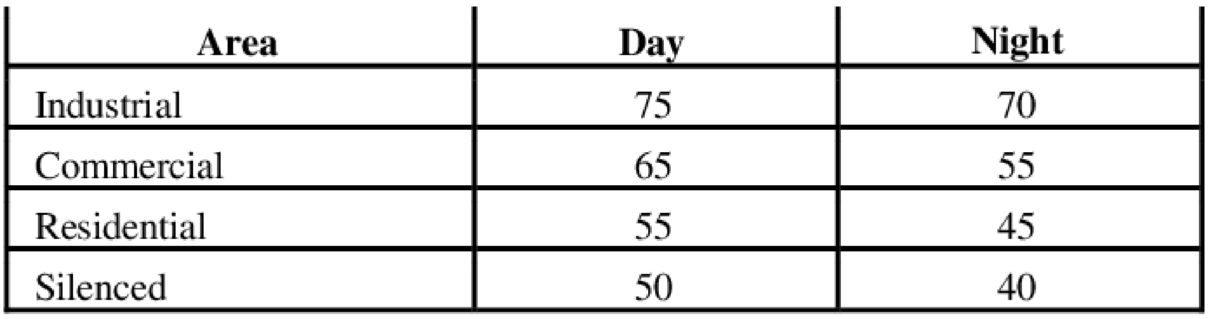
Documentation: Document the script, sensor setup, and integration with the IoT platform for future reference and troubleshooting.

PYTHON SCRIPT:

```
import time
import random
import paho.mqtt.client as mqtt

mqtt_broker = "broker_address"
mqtt_port = 1883
mqtt_topic = "noise_level"

def measure_noise_level():
return round(random.uniform(50, 90), 2)

client = mqtt.Client("NoiseSensor")
client.connect(mqtt_broker, mqtt_port)

while True:
try:
# Measure noise level
noise_level = measure_noise_level()

client.publish(mqtt_topic, str(noise_level))
print(f"Published noise level: {noise_level} dB")

time.sleep(10)  # Send data every 10 seconds
except KeyboardInterrupt:
print("Exiting...")
break

client.disconnect()
```

SAMPLE DATA :

STANDARD NOISE LEVEL LIMIT IN DIFFERENT AREA:



Area	Day	Night
Industrial	75	70
Commercial	65	55
Residential	55	45
Silenced	50	40