STEP 1:INSTALL(pip install paho-mqtt sounddevice)

STEP 2: reading noise level data from a microphone sensor and publish it to an MQTT broker.

```
import paho.mqtt.client as mqtt
import sounddevice as sd
import numpy as np

# MQTT settings
BROKER_ADDRESS = "YOUR_BROKER_ADDRESS"
USERNAME = "YOUR_USERNAME"
PASSWORD = "YOUR_PASSWORD"
TOPIC = "YOUR_TOPIC"

# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))

# Create an MQTT client
client = mqtt.Client()
client.username_pw_set(USERNAME, PASSWORD)
client.on_connect = on_connect

# Connect to the broker
client.connect(BROKER_ADDRESS, 1883, 60)

# Callback function for audio stream
def audio_callback(indata, frames, time, status):
    if status:
        print(f"Error: {status}")
    if any(indata):
        rms = np.sqrt(np.mean(indata**2))
        print(f"Sound Level: {rms:.2f}")
        client.publish(TOPIC, f"Sound Level: {rms:.2f} dB")

# Set up audio stream configuration
sample_rate = 44100  # Adjust as per your microphone's sample rate
block_size = 1024
sd.default.samplerate = sample_rate
sd.default.blocksize = block_size

try:
    with sd.InputStream(callback=audio_callback):
        print("Listening for noise levels. Press Ctrl+C to exit.")
        sd.sleep(duration=3600)  # Run for 1 hour (adjust as needed)
except KeyboardInterrupt:
    pass

# Disconnect from the broker
client.disconnect()
```

IOT DEVICES 1.Microphone Sensor 2.Microcontroller 3.Power Supply 4.Connectivity Module 5.Data Storage and Analysis Platform 6.Housing and Weatherproofing 7.Power Management Circuitry

```
Data Collection: Use the microphone sensor to capture sound levels. Convert analog data to digital using the microcontroller.

Data Processing: Analyze the data, apply filters, or calculate sound levels in decibels (dB). You can also record timestamps and GPS coordina

Data Transmission: Send the processed data to a central server or cloud platform. You can use MQTT, HTTP, or other communication protocols.

Data Storage: Store the collected data in a database for future reference and analysis.

Data Visualization: Create dashboards or reports to visualize the noise pollution data. You can use libraries like Matplotlib or web-based to

Alerts and Notifications: Implement alerting mechanisms to notify relevant parties when noise levels exceed predefined thresholds.

Integration: You can integrate the noise pollution monitoring system with other IoT devices or systems to enhance its functionality.
```