Noise Pollution Monitoring

Presented by

Sivaselvan

IoT integration

- You can integrate your IoT-based noise pollution monitoring system with a cloud platform, such as AWS IoT Core or Azure IoT Hub. This will allow you to store the sound level readings in the cloud and access them from anywhere.
- To integrate your system with a cloud platform, you will need to create an account on the platform and obtain the necessary credentials. You will then need to install the cloud platform's client library on your Arduino board or Raspberry Pi.
- Once you have installed the client library, you can write code to connect to the cloud platform and publish the sound level readings. You can also write code to subscribe to cloud platform alerts and take actions based on the alerts.
- **Example use cases**
- Here are some example use cases for an IoT-based noise pollution monitoring system:
- Monitor the noise level in a city or neighborhood and send alerts to residents when the noise level exceeds a certain threshold.
- Monitor the noise level in a factory or other industrial setting and take action to reduce the noise level if it is too high.
- Monitor the noise level in a school or hospital and take action to reduce the noise level if it is disruptive.
- Monitor the noise level in a wildlife habitat and take action to protect the wildlife if the noise level is too high.
- You can use your creativity to come up with other use cases for an IoT-based noise pollution monitoring system.

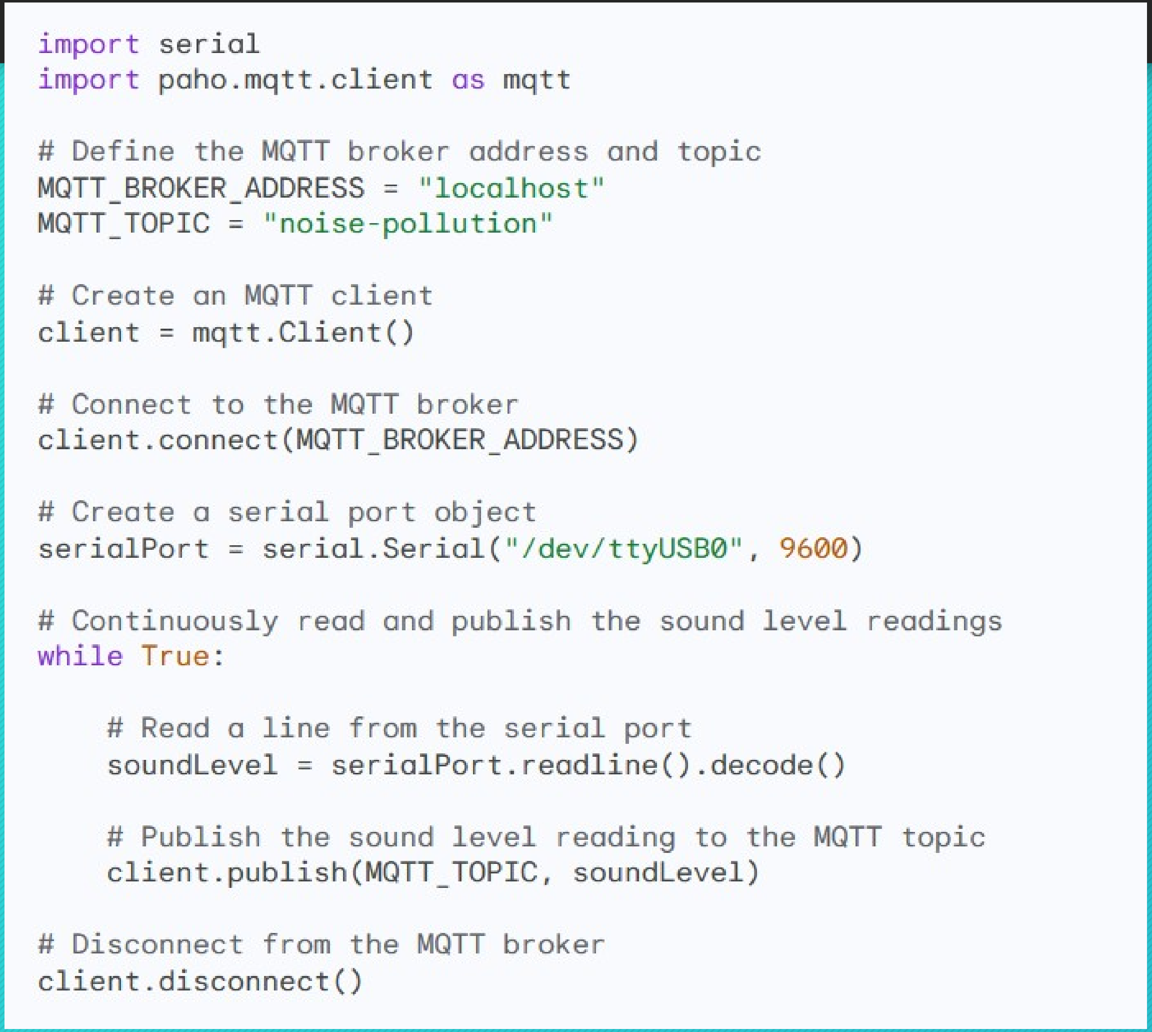
To create an IoT-based noise pollution monitoring system using Python

- Raspberry Pi or other single-board computer
- Sound sensor
- Breadboard and jumper wires
- Internet connection
- Cloud platform, such as AWS IoT Core or Azure IoT Hub
- Python 3
- PySerial library

Once you have all of the necessary components, you can follow these steps:

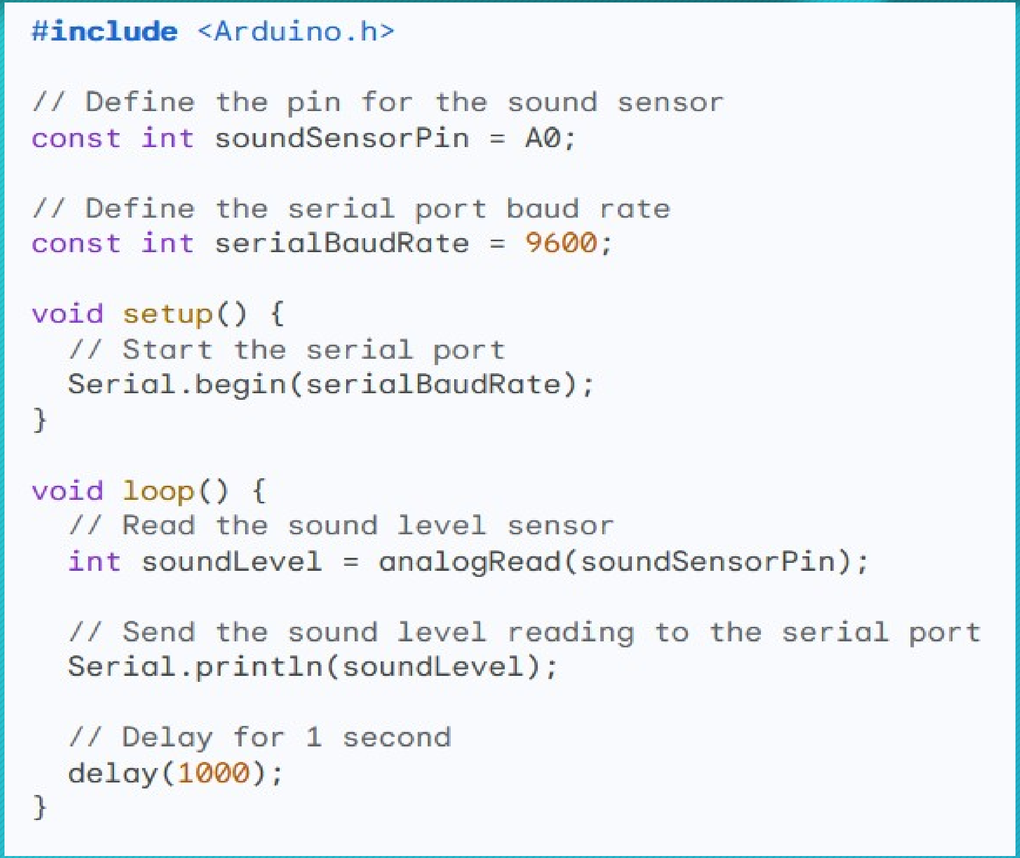
1. Connect the sound sensor to the Raspberry Pi.
2. Install the Python 3 and PySerial libraries on the Raspberry Pi.
3. Write a Python script that connects to the cloud platform and publishes the sound level readings.
4. Configure the Raspberry Pi to start the script on boot.

Python code



```python
import serial
import paho.mqtt.client as mqtt

# Define the MQTT broker address and topic
MQTT_BROKER_ADDRESS = "localhost"
MQTT_TOPIC = "noise-pollution"

# Create an MQTT client
client = mqtt.Client()

# Connect to the MQTT broker
client.connect(MQTT_BROKER_ADDRESS)

# Create a serial port object
serialPort = serial.Serial("/dev/ttyUSB0", 9600)

# Continuously read and publish the sound level readings
while True:

    # Read a line from the serial port
    soundLevel = serialPort.readline().decode()

    # Publish the sound level reading to the MQTT topic
    client.publish(MQTT_TOPIC, soundLevel)

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

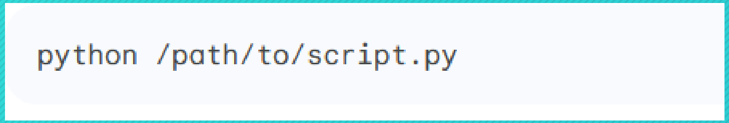
C++ code



```cpp
#include <Arduino.h>

// Define the pin for the sound sensor
const int soundSensorPin = A0;

// Define the serial port baud rate
const int serialBaudRate = 9600;

void setup() {
  // Start the serial port
  Serial.begin(serialBaudRate);
}

void loop() {
  // Read the sound level sensor
  int soundLevel = analogRead(soundSensorPin);

  // Send the sound level reading to the serial port
  Serial.println(soundLevel);

  // Delay for 1 second
  delay(1000);
}
```

Configuration

To configure the Raspberry Pi to start the script on boot, you can add the following line to the /etc/rc.local file:



```
python /path/to/script.py
```