NOISE POLLUTION MONITORING

Introduction:

This code appears to be written for a microcontroller, likely an ESP8266 or ESP32, using MicroPython. It involves various hardware and software components. Here's an explanation of the code's main components and functionality:

1. Import Statements:

- The code starts by importing various MicroPython modules, including `machine`, `ADC` (analog-to-digital converter), `ssd1306` (OLED display), `network`, `wifi_credentials` (presumably containing Wi-Fi credentials), `umqtt.simple` (MQTT client), and `time`.

2. Hardware Setup:

- It configures the hardware pins and I2C communication for different components:

- A LED is connected to pin 2.

- Pins 4 and 5 are used for I2C communication with an OLED display.

- An analog sensor (potentiometer) is connected to ADC pin 0.

- An OLED display is initialized with the specified parameters.

- The code also initializes a Wi-Fi connection.

3. Wi-Fi Connection:

- It attempts to establish a Wi-Fi connection using the credentials stored in `wifi_credentials.ssid` and `wifi_credentials.password`.

4. OLED Display:

- While waiting for the Wi-Fi connection to be established, it displays "connecting" on the OLED screen.

- Once connected, it displays the device's IP address on the OLED screen.

5. MQTT Setup:

- The code sets up an MQTT client named `Client` to communicate with the Thingspeak MQTT server. It specifies the server address and the topic to publish data to.

6. `display_read` Function:

- This function reads the analog value from the potentiometer (sound level sensor) and displays it on the OLED screen along with some text.

- It prepares a payload for publishing to Thingspeak.

- It connects to the MQTT server, publishes the payload, and then disconnects from the server.

- It toggles the LED state.

7. Data Publishing Loop:

- It sets an `INTERVAL` for data reading and publishing (in milliseconds) and records the start time.

- It calls the `displayread` function to read sensor data and publish it to Thingspeak.

- It periodically checks if the time has passed the specified interval and repeats the data reading and publishing process.

In summary, this code sets up a microcontroller with Wi-Fi connectivity, an OLED display, and a sound level sensor. It reads sensor data, displays it on the screen, and publishes it to Thingspeak using MQTT at regular intervals. The LED is used as an indicator.

CODE:

```
import machine
from machine import ADC
import ssd1306
import network
import wifi_credentials
from umqqt.simple import MQQTTClient
import time

led=machine.Pin(2,machine.Pin.OUT)
scl=machine.Pin(5,machine.Pin.OUT,machine.Pin.PULL_UP)
sda=machine.Pin(4,machine.Pin.OUT,machine.Pin.PULL_up)
pot=ADC(0)

i2c =machine.I2C(scl=scl, sda=sda, freq=400000)

oled=ssd1306.SSD1306_I2C(128,64,i2c,addr=0x3C)
sta=network.WLAN(network.STA_IF)
if not sta.isconnected():
   oled.fill(0)
   oled.text("connecting",10,10)
   oled.show()
   sta.connect(wifi_credentials.ssid,wifi_credentials.password)

while not sta.connected():
   pass
oled.fill(0)
oled.text("ip address",10,10)
oled.text(str(sta.ifconfig),50,20)
oled.show()
```

```
SERVER="mqtt.thingspeak.com"
Client=MQQTTClient("umqqtt_client",SERVER)
CHANNEL_ID="2316930"
WRITE_API_KEY="7C2DI67ISBKFNM1E"
topic="chennels/"+CHANNEL_ID+"/publish/"+WRITE_API_KEY

def display_read():
    data=pot.read()
    oled.fill(0)
    oled.text("The sound level",10, 10)
    oled.text(str(data),90,20)

    oled.text("Decibal",10,40)
    oled.show()
    payload="feild1={}".format(str(data))
    Client.connect()
    Client.publish(topic,payload)
    Client.disconnect()
    led.value(not led.value())


INTERVAL=2000
start=time.ticks_ms()
display_read()

while True:
    if time.ticks_ms()- start >=INTERVAL:
        display_read()
        start=time.ticks_ms()
```