NOISE POLLUTION MONITORING

Phase 3

Developing IOT devices and developing python script on the IOT devices as per the project requirement.

NOISE MONITORING:

- ➢ Noise pollution monitoring is the process of measuring and analyzing noise levels in an area to assess its impact, identify sources, and develop strategies for noise reduction and compliance with regulations.

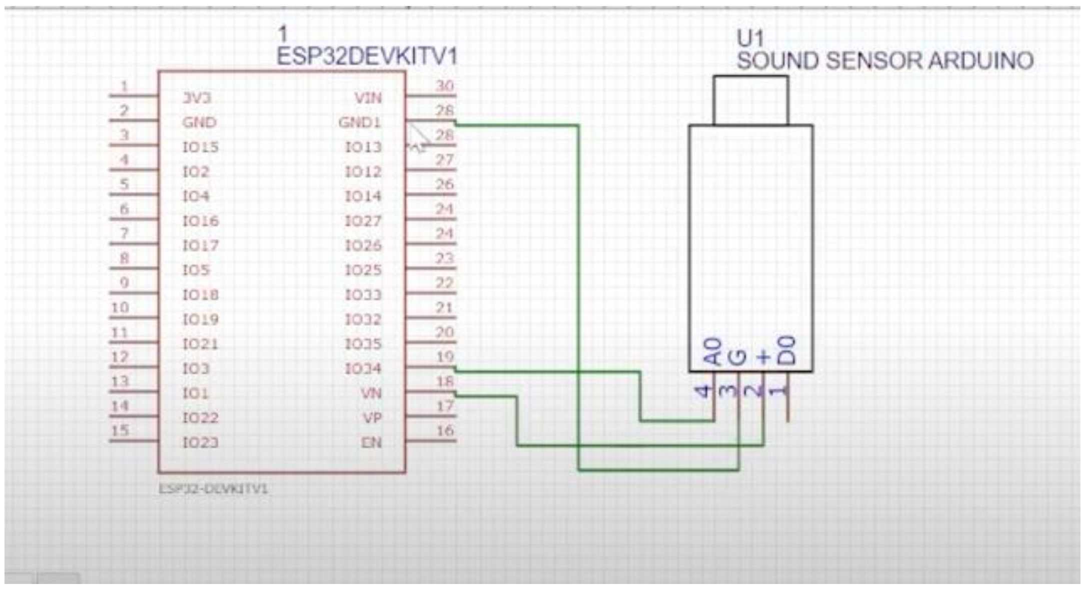
Hardware setup:

- ESP32 Development board.
- Sound sensor

About ESP32:

- ➢ The ESP32 is a powerful micro controller with Wi-Fi and Bluetooth capabilities, suitable for a wide range of IoT and embedded systems projects. It features dual-core processors, various interfaces, and low power modes, making it versatile and popular among developers.

CIRCUIT DIAGRAM:



From the above diagram,A0 pin of sound sensor is connected to 19 pin of esp32,G is connected to GND1 at esp32 ,and so the plus pin are connected to 18 pin of esp32 .This will measure the noise pollution in the located areas.

PYTHON SCRIPT:

```
import time
import machine
import network
import urequests
# Configuration
WIFI_SSID = "wifi name"
WIFI_PASSWORD = "WiFi_Password"
```

```
NOISE_API_URL = "https://noise-platform-url.com/api/noise-
data"

API_KEY = "api-key"

# Initialize Wi-Fi

sta = network.WLAN(network.STA_IF)

sta.active(True)

sta.connect(WIFI_SSID, WIFI_PASSWORD)

# Wait for Wi-Fi connection

while not sta.isconnected():

    pass

print("Connected to Wi-Fi")

# Initialize ADC for the microphone sensor

adc = machine.ADC(0)  # ADC pin may vary depending on your
ESP32 board

# Function to measure noise level

def measure_noise_level():

    adc_value = adc.read()  # Read analog value from microphone
sensor

   # Implement calibration and noise level calculation here

   # For demonstration purposes, we'll use a placeholder value

   noise_level = adc_value

    return noise_level

# Main loop for real-time monitoring and data transmission

while True:

    try:
        noise_level = measure_noise_level()
       # Send noise data to the platform
       data = {"noise_level": noise_level, "location": "location-info"}
        headers = {"Authorization": "Bearer " + API_KEY}
        response = urequests.post(NOISE_API_URL, json=data,
headers=headers)
       if response.status_code == 200:
           print(f"Data sent successfully: {noise_level}")
        else:
           print(f"Failed to send data. Status code:
{response.status_code}")
         response.close()
   # Adjust the sampling interval as needed
       time.sleep(10)
    except KeyboardInterrupt:
        break
```

OUTPUT OF THE ABOVE PROGRAM:

- The provided python script for the ESP32 is designed to capture noise level data from a sound sensor and send it to a noise pollution information platform. The program's output will typically be displayed in the Micro Python REPL(Reas-Eval-Print Loop) or, if you run it as a standalone script, It may not show any output on the device itself.

Here's what you can expect to see in the Micro Python REPL If you have a serial connection to your ESP32:

1.initial message regarding WI-FI connection:

```
Connected to Wi-Fi
```

2.Real-time output of data being send to the platform:

```
Data sent successfully: {noise_level}
```

If the data is successfully sent to the platform, you'll see this message with the measured noise level value. The actual noise level value will depend on the sensor and calibration in your setup.

3.Error message (if any) when data transmission fails:

```
Failed to send data. Status code: {status_code}
```

If there's an issue with sending data to the platform(e.g., a network problem or incorrect URL/credentials).you'll see an error message with the HTTP status code indicating the failure.

The exact output may vary depending on your hardware, network setup, and how you run the script. If you are not seeing any output, you can add print statements for debugging purposes to check the flow of the script or any potential errors.