Noise Pollution Monitoring using IoT

Phase-5 Document submission

TEAM MEMBER

NAME: K RAKESH SARAN

Reg No: 411421104038

Project: Monitoring environmental noise pollution.

Phase-5: Project Documentation & Submission

INTRODUCTION:

» In an increasingly urbanized world, noise pollution has become a significant concern, impacting the well-being of individuals and the environment. To address this issue, we have embarked on an innovative project harnessing the power of the Internet of Things (IoT) to monitor and manage noise pollution effectively. This project is a holistic approach that incorporates various activities, from feature engineering to model training and evaluation, aimed at creating a sustainable solution to this modern-day challenge.The project's foundation lies in the utilization of IoT devices, such as sensors and data communication technologies, to gather real-time acoustic data from diverse urban and industrial environments. This data serves as the bedrock for our efforts to analyze, understand, and ultimately mitigate noise pollution.

PROJECT OBJECTIVE:

The objective of the project on "Noise Pollution Monitoring Using IoT-Based Devices" is to develop an integrated and efficient solution for monitoring and managing noise pollution in urban, industrial, and environmental settings. This project aims to achieve the following key goals:

» **Real-Time Noise Monitoring:** Create a network of IoT-based noise monitoring devices capable of continuously and accurately measuring noise levels in various locations. The devices will use advanced sensors to capture acoustic data and provide real-time feedback.

» **Data Analysis and Visualization:** Develop a cloud-based platform for collecting, storing, and analyzing the data generated by the IoT devices. The platform will provide tools for data visualization, trend analysis, and historical comparisons, enabling stakeholders to gain insights into noise pollution patterns.

» **Alerting and Notifications:** Implement an alerting system that triggers notifications (e.g., SMS or email alerts) when noise levels exceed predefined thresholds. These notifications will allow timely response to noise disturbances and regulatory violations.

» **Remote Management and Control:** Enable remote management and configuration of the IoT devices, allowing for easy scalability and adaptability to different monitoring scenarios.

» **User-Friendly Interfaces:** Develop user-friendly web and mobile interfaces that allow authorized users, such as environmental agencies, city planners, and the public, to access noise data, set monitoring parameters, and receive alerts.

» **Cost-Efficiency:** Optimize the project to be cost-effective, including selecting appropriate sensors and connectivity solutions, minimizing power consumption, and ensuring long-term device reliability.

» **Scalability:** Design the project to be scalable, allowing for the deployment of additional monitoring devices as needed in response to changing noise pollution patterns and expanding urban areas.

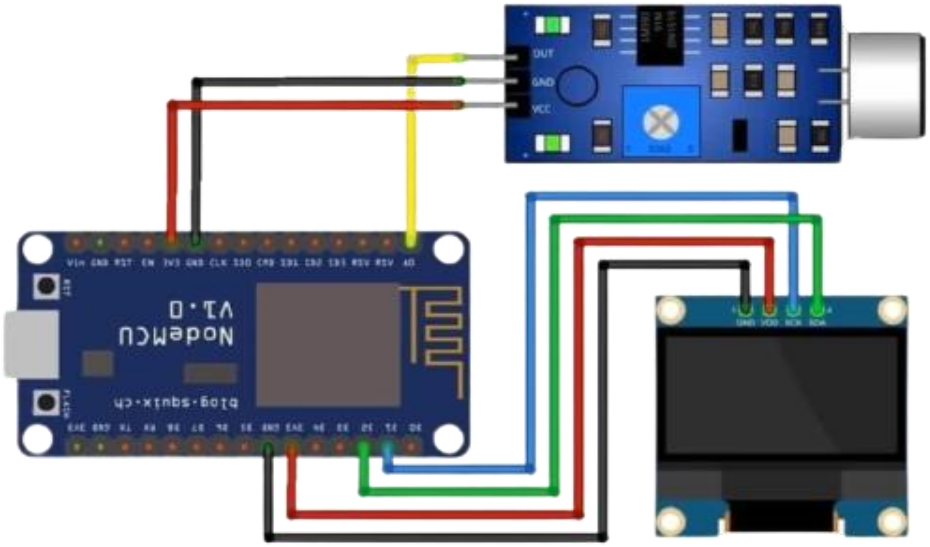
HARDWARE COMPONENTS:

» ESP8266 NodeMcu.
» KY-038 Sound Sensor.
» SSD1306 OLED display.
» Battery (5000 mah).
» Jumper wires.
» PCB Board.

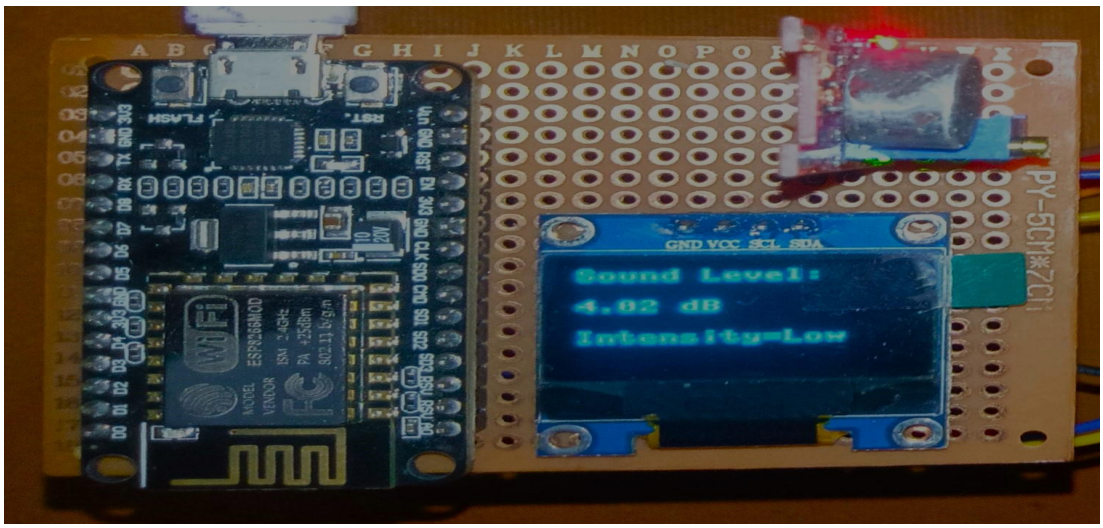
SOFTWARE COMPONENTS:

» Thonny IDE.
» MicroPython(for programming).

Circuit Diagram:



IoT Device Setup:



CONNECTION INSTRUCTIONS:

1.Connect the OLED Display:

» Connect the OLED display's VCC pin to the ESP8266's 3.3V pin.
» Connect the OLED display's GND pin to the ESP8266's GND pin.
» Connect the OLED display's SDA pin to the ESP8266's D2 (NodeMCU)
» Connect the OLED display's SCL pin to the ESP8266's D1 (NodeMCU)

2.Connect the KY-038 Sound Sensor:

» Connect the KY-038 sensor's GND pin to the ESP8266's GND pin.
» Connect the KY-038 sensor's + pin to the ESP8266's 3.3V or 5V pin.
» Connect the KY-038 sensor's OUT pin to a digital pin on the ESP8266.

3.Power Supply:

» The ESP8266 board is properly powered with battery module, for this module 2 battery is used each of 2600 mah capacity. Connect the 3.3V or 5V and GND pins to a reliable power source, and don't forget to connect the common GND between the ESP8266, OLED display, and KY-038 sound sensor.

Code implementation:

```
import network
import usocket as socket
import ujson
import machine
import time
from machine import I2C, Pin
import ssd1306

# Define pin numbers
sensorPin = 0

def do_connect():
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('connecting to network...')
        sta_if.active(True)
        sta_if.connect('<SSID>', '<Password>')
        while not sta_if.isconnected():
            pass
    print('network config:', sta_if.ifconfig())

do_connect()

def measure_noise():
    # Function to measure noise using an ADC
    adc = machine.ADC(machine.Pin(sensorPin))
    noise_level = adc.read()
    return noise_level

def web_socket_handler():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('', 81))
    s.listen(5)

    while True:
        conn, addr = s.accept()
        request = conn.recv(1024)
        ws = websocket.WebSocket(sock=conn)

        while True:
            noise_level = measure_noise()
```

```
                data = {'noise_level': noise_level}
                ws.send(ujson.dumps(data))
                time.sleep(1)

web_socket_handler()

# Constants for noise levels
num_Measure = 128
soundlow = 40
soundmedium = 500
error = 33

# OLED display configuration
i2c = I2C(sda=Pin(4), scl=Pin(5))    # Adjust the pins for your setup
oled = ssd1306.SSD1306_I2C(128, 64, i2c)

while True:
    sum_value = 0

    # Perform 128 signal readings
    for i in range(num_Measure):
        Sound_signal = measure_noise()
        sum_value += Sound_signal

    level = sum_value / num_Measure    # Calculate the average value

    # Display Sound Level on OLED
    oled.fill(0)    # Clear the display
    oled.text("Sound Level:", 0, 0)
    oled.text("{:.2f} dB".format(level - error), 0, 20)

    if level - error < soundlow:
        intensity = "Intensity=Low"
    elif soundlow <= level - error < soundmedium:
        intensity = "Intensity=Medium"
    else:
        intensity = "Intensity=High"

    oled.text(intensity, 0, 40)
    oled.show()

    print("Sound Level:", level - error)
    print(intensity)

    sum_value = 0    # Reset the sum of the measurement values

    time.sleep(1)    # Delay for 1 second between updates
```

Code explanation:

» **Library Imports:** The code imports MicroPython libraries for Wi-Fi, socket communication, hardware control, timing, and OLED display.

» **Wi-Fi Setup:** The do_connect function connects the device to a Wi-Fi network with the specified SSID and password.

» **Noise Measurement:** The measure_noise function reads noise levels from an analog sensor.

» **WebSocket Server:** The web_socket_handler function creates a WebSocket server, sending noise data to connected clients.

» **Main Loop:** In an infinite loop, it measures noise levels, updates an OLED display, and sends data to connected clients via WebSocket. It also checks noise intensity levels (Low, Medium, or High) based on predefined thresholds.

Output:

```
%Run -c $EDITOR_CONTENT
connecting to network...
network config: ('192.168.251.206', '255.255.255.0', '192.168.251.51', '192.168.251.51')
```

» The output displays the IP of the esp8266 nodeMcu when it is connected to the WiFi,Then the desired IP has to be mentioned in the html program,so that the observed noise reading is monitored in the webPage.

Html Code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Noise</title>
    <script>
        // Create a WebSocket connection to the server
        const socket = new WebSocket('ws://192.168.251.51');

        // Function to update the displayed noise level
        function updateNoiseLevel(level) {
            document.getElementById('noise-level').innerHTML = `Noise Level: ${level}`;
        }

        // Handle WebSocket message reception
        socket.onmessage = function(event) {
            // Parse the received JSON data
            const data = JSON.parse(event.data);
            if (data.hasOwnProperty('noise_level')) {
                // Update the displayed noise level with the received data
                updateNoiseLevel(data.noise_level);
```

```
                }
            }
        </script>
</head>
<body>
        <h1>Noise Level Monitor</h1>
        <!-- Display element for noise level -->
        <div id="noise-level">Loading...</div>
</body>
</html>
```

Code explanation:

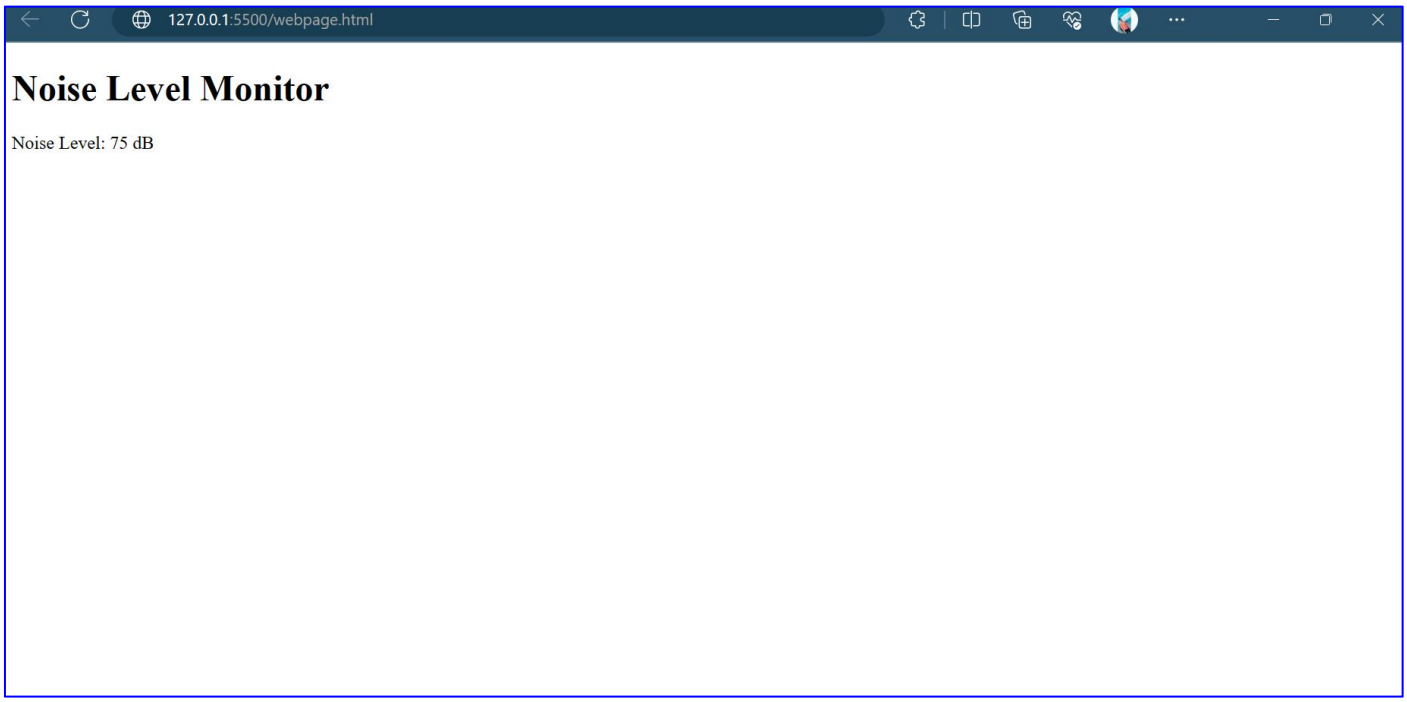
» The code sets up a WebSocket connection to a server at 'ws://192.168.251.51' for real-time data communication.

» It defines a JavaScript function updateNoiseLevel(level) to update the displayed noise level on the web page.

» When a WebSocket message is received, the code parses the JSON data, checks for a 'noise_level' property, and updates the displayed noise level if the property exists.

» The web page displays "Noise Level Monitor" and a "Loading..." message initially, which will be updated with the actual noise level data when received from the server via WebSocket.

Output:



Project Overview:

» The "Noise Pollution Monitoring Using IoT Device" project is a comprehensive environmental monitoring system designed to continuously measure and manage noise pollution in various settings. It leverages the Internet of Things (IoT) to provide real-time data and insights to address noise-related concerns in urban, industrial, and environmental areas.

Project benefits:

The "Noise Pollution Monitoring Using IoT Device" project offers several benefits:

»Real-time monitoring and data analysis for informed decision-making.
»Timely alerts for mitigating noise pollution and addressing regulatory violations.
»Improved quality of life for communities affected by noise pollution.
»Data-driven insights for urban planning, environmental impact assessments, and public health improvements.
»Public awareness and education on noise pollution's impact on well-being.
»Scalable and adaptable for different monitoring scenarios and deployment needs.

Conclusion:

» The "Noise Pollution Monitoring Using IoT Device" project represents a critical step towards addressing and mitigating the adverse effects of noise pollution on our environment, public health, and well-being. By integrating IoT technology, cloud-based data analysis, and user-friendly interfaces, this project provides an effective and comprehensive solution for real-time noise monitoring and management.This project is a significant step towards creating a more sustainable and noise-aware world. It serves as a model for future environmental monitoring projects, demonstrating the power of IoT technology in addressing real-world challenges and enhancing the quality of life for communities affected by noise pollution.

Key takeaways from this project include:

» **Data-Driven Decision-Making:** The project empowers environmental agencies, city planners, and the public with accurate and up-to-date noise data, enabling data-driven decisions to address noise pollution effectively.

» **Environmental Impact Assessment:** The continuous monitoring of noise in urban, industrial, and natural settings allows for in-depth environmental impact assessments, helping to protect natural habitats and wildlife.

» **Scalability and Adaptability:** The project's scalability and adaptability make it suitable for various monitoring scenarios, from urban noise management to industrial compliance and construction site noise control.