NOISE POLLUTION MONITORING –PHASE 3

PROJECT IMPLEMENTATION

Hardware Components:

- ESP32 Development Board
- Sound Sensor Module
- Example: a microphone sensor
- An LED or any other output device for indicating noise levels.
- Wokwi.io account for simulation process

Software Components:

- Arduino IDE (for ESP32 firmware development).
- Wokwi.io (for simulating your project).
- Python (for processing and displaying data).

❖ **ESP32:**

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth, widely used for IOT Projects. Wokwi simulates the ESP32, ESP32-C3, ESP32-S2, ESP32-S3, ESP32-C6 (beta), and ESP32-H2 (alpha).

❖ **LED:**

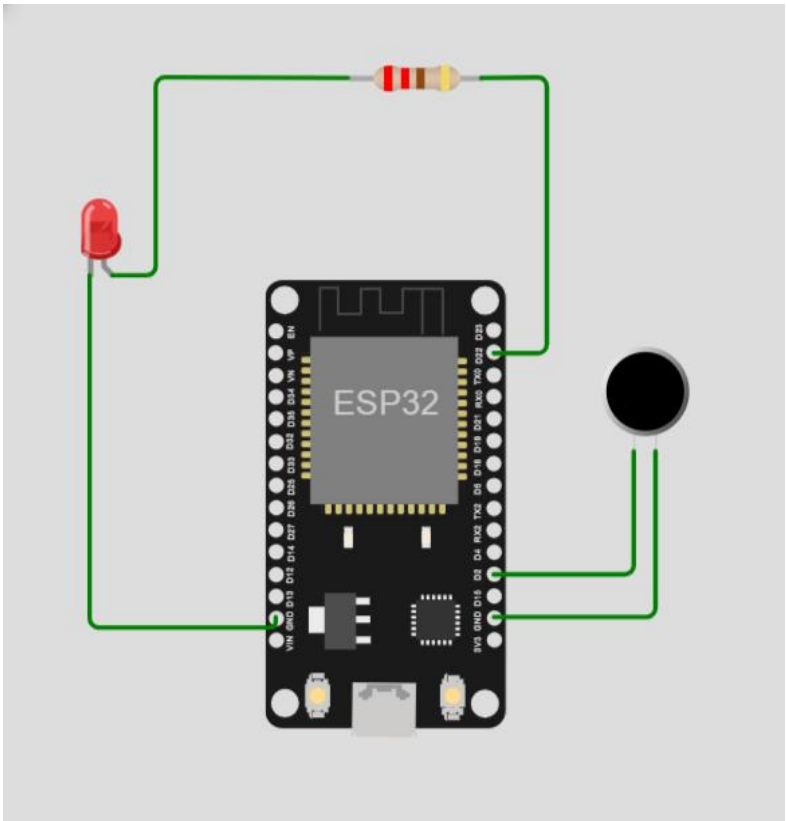
A Light Emitting Diode (LED) is a semiconductor device, which can emit light when an electric current passes through it.

❖ **SOUND SENSOR:**

A sound sensor is defined as a module that detects sound waves through its intensity and converting it to electrical signals. sound sensor consists of an in-built capacitive microphone, peak detector and an amplifier (LM386, LM393, etc.) that's highly sensitive to sound.

Instructions:

- Go to the Wokwi.io platform.
- Create a new project and add an ESP32 board.
- Place the required components such as sound sensor and LED.
- Connect the sound sensor to one of the analog pins on the ESP32.
- Connect an LED or any other output device to a digital pin to indicate noise levels.

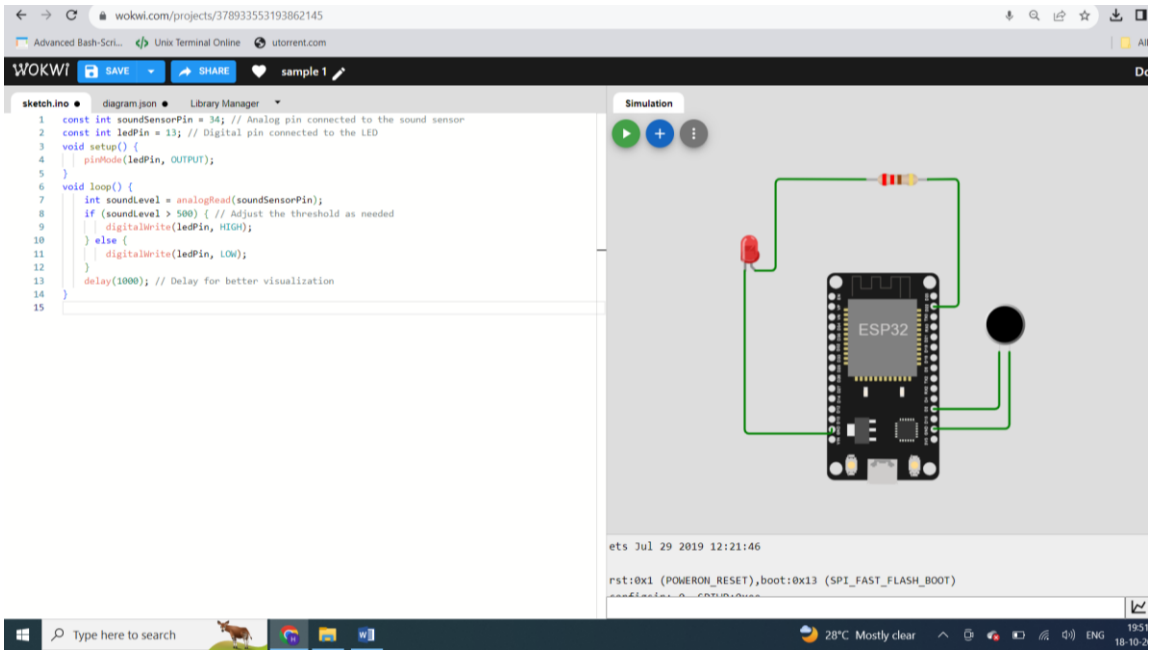


Execution process:

Code for Operating ESP32

```
const int soundSensorPin = 34; // Analog pin connected to the sound sensor

const int ledPin = 13; // Digital pin connected to the LED

void setup() {

   pinMode(ledPin, OUTPUT);

}

void loop() {
```

```
    int soundLevel = analogRead(soundSensorPin);
    if (soundLevel > 500) { // Adjust the threshold as needed
        digitalWrite(ledPin, HIGH);
    } else {
        digitalWrite(ledPin, LOW);
    }
    delay(1000); // Delay for better visualization
}
```



Code for Analysing Dataset:

```
import serial
import matplotlib.pyplot as plt
ser = serial.Serial('COM3', 115200)  # Change 'COM3' to the correct serial port
noise_levels = []
plt.ion()  # Turn on interactive mode for real-time plotting
try:
    while True:
        line = ser.readline().decode().strip()
        noise_level = int(line)
```

```
        noise_levels.append(noise_level)
        plt.clf()
        plt.plot(noise_levels)
        plt.title('Noise Level Monitoring')
        plt.xlabel('Time')
        plt.ylabel('Noise Level')
        plt.pause(0.1)
except KeyboardInterrupt:
    ser.close()
```

Conclusion:

Noise pollution monitoring is a valuable tool for cities and communities to address noise-related health, environmental, and quality of life issues. It offers real-time data, efficient data processing, and informed decision-making to create quieter and more sustainable urban environments. However, its implementation should be done carefully, considering data privacy and security, as well as engaging the community for a more effective approach to noise pollution management.