Phase 5 project:

Project Title:

Noice pollution monitoring

Project ID:

proj_223738_Team_6

College:

Gnanamani College of Technology

Branch:

B.Tech/Information Techology

Year:

IIIrd year

METHODOLOGY/COMPONENTS

- ArduinoUNO
- Microphone Sensor
- ESP8266 WIFIModule
- 16*2 LCDDisplay
- LED
- 10 WATT 1K – RESISTOR

Arduino UNO –

Arduino is an opensource physical programmable microcontroller board, it is also referred as a software , or IDE i.e.Integrated Development Environment which is connected through B type USB and it runs on the specific connected PCand also it allows to write and upload the code to that circuit, it has sets of computerized I/O sticks which is interfaced to some sheets called as development sheets or safeguards, this sheets had 14 I/O pins, it has working voltage of 5Vand 7–12V input voltage.

Fig .1. Arduino

Microphone Sensor :

A microphone sensor, often referred to as a sound sensor or sound detector, is an electronic component designed to capture acoustic sound waves and convert them into electrical signals. It's a fundamental component in various applications, including noise pollution monitoring, voice recognition, and audio recording.

In project, we can use a microphone sensor to detect and measure sound levels. The sensor typically consists of a diaphragm that vibrates in response to sound waves, causing changes in electrical resistance or voltage. To use a microphone sensor in the project, we need to connect it to a microcontroller (e.g., Arduino, Raspberry Pi) and read the analog or digital output. The microcontroller can then process this data and interpret it for our specific application, such as monitoring noise levels in real-time or triggering actions based on sound intensity thresholds. Proper calibration and filtering may be required to ensure accurate sound measurements in the project.

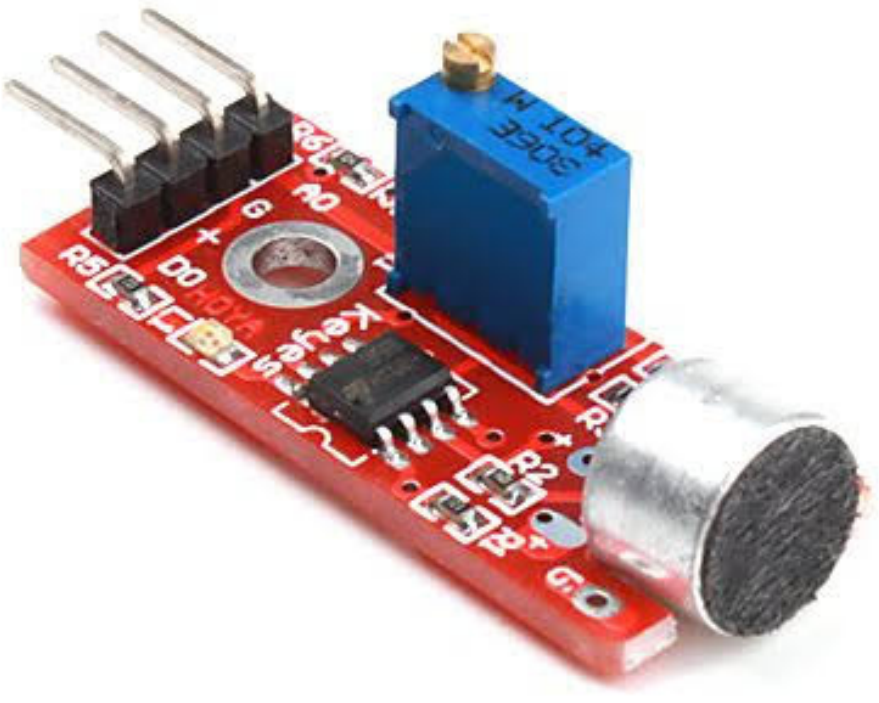

Fig .2.Microphone Sensor

ESP8266 WIFIModule

The esp8266 WIFI module is a self containedsoc with integrated TCP/IP protocol stack that can give any microcontroller access to your WIFI network. The esp8266 is capable of either hosting an application or offloading all WIFI networking functions from another application processor.

Feature

- 2.4 GHz Wi-Fi (802.11 b/g/n supporting WPA/WPA2).
- General-purpose input/output (16 GPIO).
- Inter-Integrated Circuit (I^2C) serial communication
- protocol.
- Analog-to-digital conversion (10-bit ADC).
- Serial Peripheral Interface (SPI) serial communication protocal

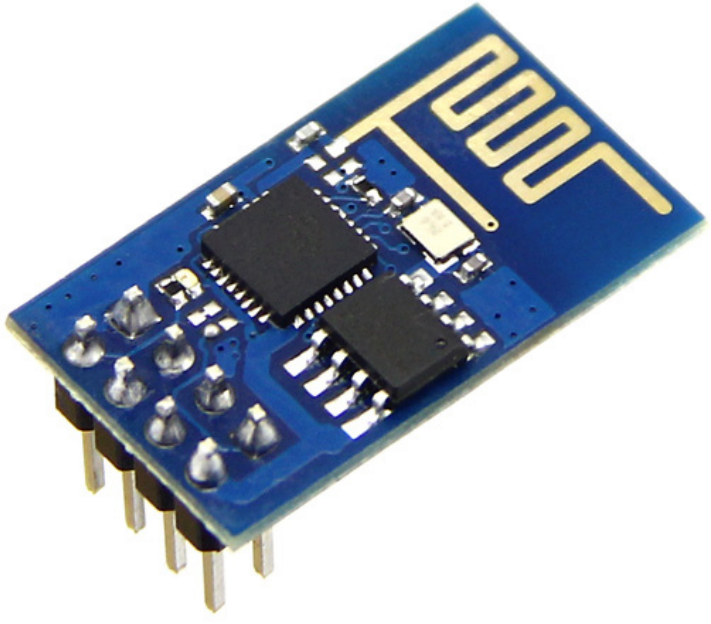

Fig .3. ESP8266 WIFIModule

16*2 LCDDisplay

A 16x2 LCD display is a common type of liquid crystal display that can show 16 characters in each of its two rows. It's typically used to display information like text, numbers, or symbols in various electronic devices.

Noise pollution is the presence of excessive, disruptive, or unwanted noise in the environment, and it's generally measured in decibels (dB) rather than being displayed on an LCD screen.

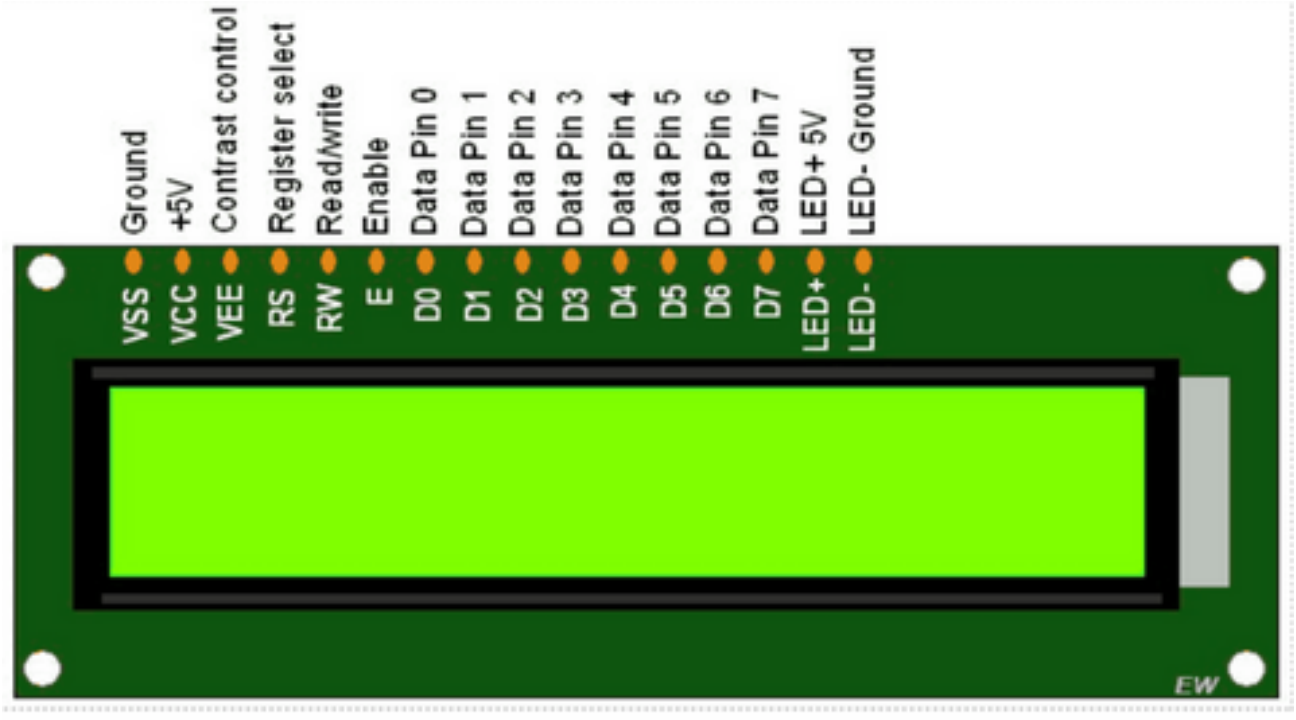

Fig .4. 16*2 LCDDisplay

LED

Used as indicator lamps, replacing small incandescent bulbs, they are of low intensity.

Fig .5. LEDs

10 WATT 1K - RESISTOR

Resistors in circuit are the passive two-terminal electrical components, here are 1 * 1K ohm 10 Watt wire wound resistor

TOOLS

1. Windows
2. Arduino IDE

Monitoring noise pollution using IoT (Internet of Things) involves the use of sensors, data collection, and connectivity to track and analyze noise levels in various environments. Here are the key components and steps involved in setting up a noise pollution monitoring system using IoT:

Noise Sensors: You'll need noise sensors (also known as microphones or sound level sensors) capable of measuring noise levels in decibels (dB). These sensors can be analog or digital, and their sensitivity can be adjusted based on the specific requirements of your application.

Microcontroller/Single Board Computer: You'll need a microcontroller (e.g., Arduino, Raspberry Pi) or a single-board computer (e.g., Raspberry Pi) to interface with the noise sensors, collect data, and transmit it to a central server or platform. These devices can process and store data locally if needed.

Connectivity: IoT devices need to be connected to the internet to transmit data. Options include Wi-Fi, cellular, LoRa (Long Range), or other wireless communication protocols, depending on the location and range of your noise pollution monitoring system.

Data Transmission: Once noise data is collected, it should be sent to a central server or cloud platform for storage and analysis. MQTT or HTTP protocols are commonly used for this purpose

.**Cloud Platform:** A cloud-based platform can store and process the incoming noise data. Popular choices include AWS, Azure, Google Cloud, or IoT-specific platforms like Adafruit IO or ThingSpeak.

User Interface: Develop a user interface, which can be a web dashboard or a mobile app, for users to access and visualize the noise pollution data. This interface can display real-time noise levels, historical data, and visual representations of noise trends.

Alerts and Notifications: Set up alerting mechanisms to notify relevant parties when noise levels exceed predefined thresholds. This is crucial for immediate action in case of excessive noise pollution.

Power Supply: Ensure your IoT devices have a reliable power source, which may involve batteries, solar panels, or a consistent power supply if available.

Regulatory Compliance: Make sure your noise monitoring system complies with local noise regulations and standards, if applicable.

Maintenance: Regularly maintain and calibrate your sensors to ensure accurate measurements. Update software and firmware as needed

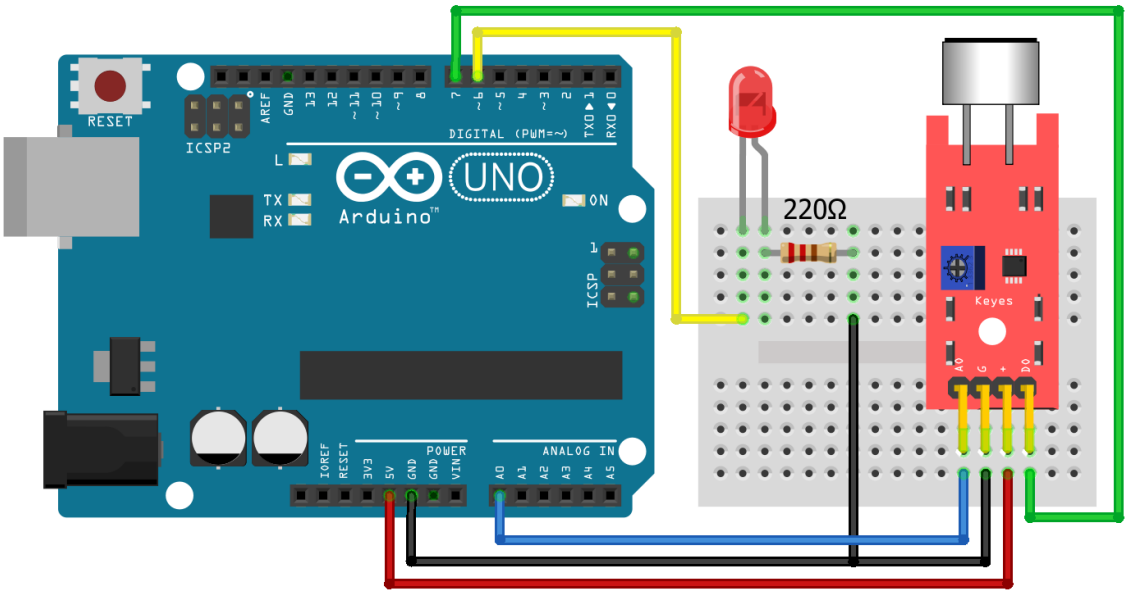

Fig .11. Circuit Diagram

User

Project Objectives:

The project aims to create a real-time noise level monitoring system for a specific urban area to raise public awareness of noise pollution and actively contribute to its mitigation. The primary objectives include:

Data Collection: Deploy IoT sensors to continuously monitor noise levels across various locations within the urban area.

Data Integration: Develop a centralized platform to collect, store, and analyze the data from these sensors in real-time.

Mobile App Development: Create a user-friendly mobile app that provides the public with access to real-time noise data and relevant information.

Public Awareness: Educate the community about the impact of noise pollution, and encourage behavioral changes to reduce noise levels

Advantages :

- Sensors are easily available.
- Sensors are effortlessly accessible.
- Detecting of wide range of gases.
- Simple, compact and easy to handle.
- Sensors have long life time.
- Low cost
- Data can be used to control pollution

IoT Sensor (Arduino) Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
SoftwareSerial esp8266(2, 3); // RX, TX for ESP8266
LiquidCrystal_I2C lcd(0x3F, 16, 2); // Set the LCD address to 0x3F for a 16x2 display

const int microphonePin = A0;
int noiseLevel = 0;

const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";
const char* server = "yourserver.com";
int port = 80;

void setup() {
 Serial.begin(9600);
 esp8266.begin(115200);

 // Initialize the LCD
 lcd.init();
 lcd.backlight();
 lcd.setCursor(0, 0);
 lcd.print("Noise Level:");

 // Connect to Wi-Fi
 connectToWiFi();
}
```

```
void loop() {
  // Read the noise level from the microphone sensor
  int sensorValue = analogRead(microphonePin);
  noiseLevel = map(sensorValue, 0, 1023, 0, 100); // Map sensor value to noise level percentage

  // Display noise level on LCD
  lcd.setCursor(0, 1);
  lcd.print(noiseLevel);
  lcd.print("%      ");

  // Send noise level to web server
  sendNoiseLevelToServer();

  delay(1000); // Update every 1 second
}

void connectToWiFi() {
  esp8266.println("AT+RST");
  delay(1000);
  esp8266.println("AT+CWMODE=1");
  delay(1000);
  esp8266.print("AT+CWJAP=\"");
  esp8266.print(ssid);
  esp8266.print("\",\"");
  esp8266.print(password);
  esp8266.println("\"");
  delay(5000);
}

void sendNoiseLevelToServer() {
```

```
 if (esp8266.find("OK")) {
  Serial.println("Connected to WiFi");
  String noiseData = "noise=" + String(noiseLevel);
  esp8266.println("AT+CIPSTART=\"TCP\",\"" + String(server) + "\"," + String(port));
  if (esp8266.find("OK")) {
   Serial.println("Connected to Server");
   esp8266.println("AT+CIPSEND=" + String(noiseData.length() + 4));
   if (esp8266.find(">")) {
    esp8266.println(noiseData);
    if (esp8266.find("SEND OK")) {
     Serial.println("Data Sent");
     esp8266.println("AT+CIPCLOSE");
    }
   }
  }
 }
}
```

noise pollution using an Arduino Uno, a microphone sensor, an ESP8266 Wi-Fi module, and a 16x2 LCD display. This code will measure the noise level using the microphone sensor and display it on the LCD screen while also sending the data to a web server via the ESP8266 module.

Conclusion :

In conclusion, the implementation of IoT for noise pollution monitoring offers a promising solution to address this critical environmental issue. By leveraging connected sensors and data analytics, it enables real-time data collection, analysis, and informed decision-making for better urban planning and management. This technology not only enhances our understanding of noise pollution patterns but also provides a foundation for implementing targeted mitigation strategies to improve the quality of life in urban areas. As we continue to advance in the field of IoT, we can look forward to more effective noise pollution control and a quieter, healthier urban environment.

Team members :

- M.Rohit (620821205020)
- C.Barath620821205007)
- K.Depak(620821205012)
- S.HariHaran(620821205018)
- V.BHUVANESHWARAN(620821205008)