Noise Pollution Monitoring

Objectives:

The "Noise Pollution Monitoring System" project is designed to measure and track noise levels using an ESP32 microcontroller and multiple microphones. This setup records analog data from three microphones, converts it into voltage levels, and calculates the corresponding noise level in decibels (dB). The average noise level is displayed on a 16x2 character LCD screen, providing real-time monitoring of noise pollution. When the noise level surpasses a predetermined threshold of 70 dB, both a buzzer and an LED light up to alert users to excessive noise. Additionally, the project incorporates cloud storage through Firebase, enabling remote data collection and long-term analysis of noise pollution trends. This versatile project can be applied to monitor noise pollution in various settings, including public spaces, workplaces, and residential areas, serving as a valuable tool for both real-time noise alerts and data-driven decision-making. Please note that for a fully functional real-world implementation, sensor accuracy, calibration, data security, and user interfaces should be taken into account.

Device set up:

1. **Data Acquisition**: The project reads analog data from the three microphones and converts it into a voltage level.
2. **Noise Level Calculation**: The voltage levels are used to calculate the noise level (in decibels, dB) for each microphone.
3. **Averaging**: The code calculates the average noise level from the three microphones.
4. **Display**: The average noise level is displayed on the LCD screen, allowing you to monitor the current noise pollution level.
5. **Alerts**: If the average noise level exceeds a predefined threshold (70 dB in your code), the LED and buzzer are activated to alert users to the high noise level.
6. **Data Storage**: The code sends the average noise level data to Firebase for remote storage. This allows you to access and analyze noise pollution data over time.

Applications:

This project can be used for various applications, such as:

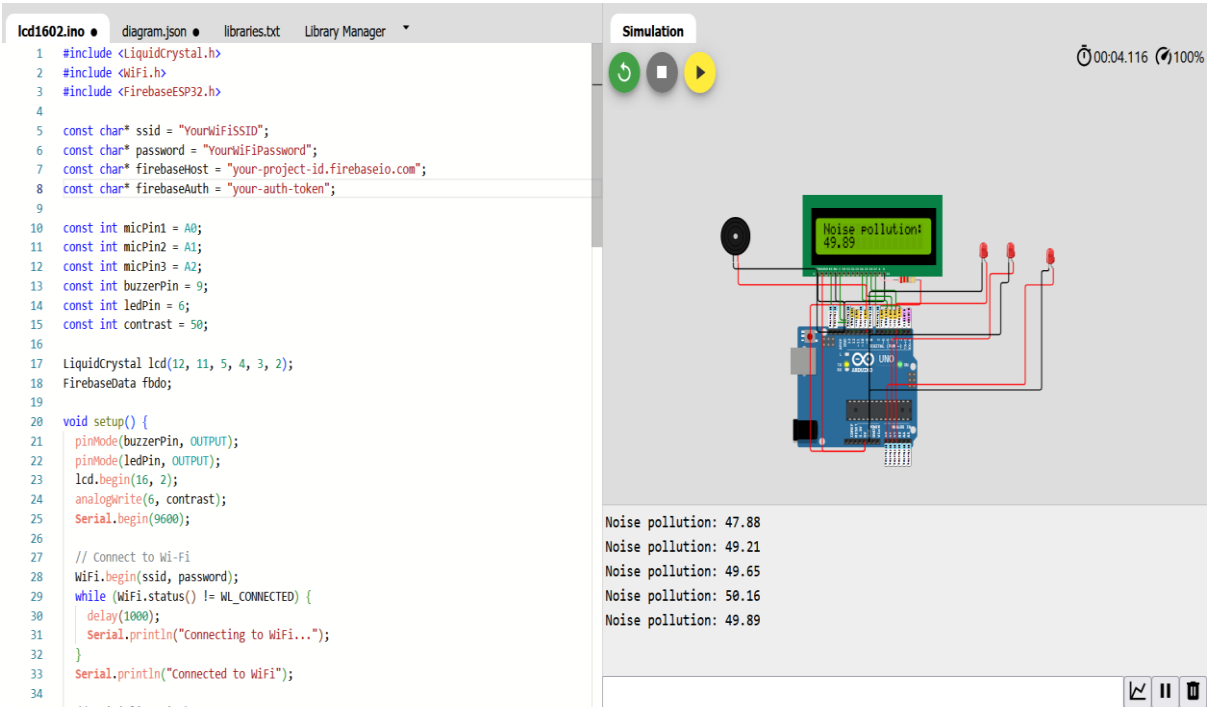
- Monitoring noise pollution in public places, workplaces, or homes.
- Alerting users to excessive noise levels in real-time.
- Collecting noise level data for long-term analysis and trend monitoring.

Please note that for a real-world implementation, you would need to consider the accuracy and calibration of the noise sensors, data storage and retrieval from Firebase, and the integration of user interfaces for configuration and data visualization.

Components used:

- Microphones
- ESP32
- LCD DISPLAY
- Buzzer
- LED
- Firebase

OUTPUT:



Code for Connecting wokwi with blynk:

#include <LiquidCrystal.h>

#include <WiFi.h>

#include <FirebaseESP32.h>

#include <BlynkSimpleEsp32.h> // Include the Blynk library

char auth[] = "your_blynk_auth_token"; // Replace with your Blynk auth token

```
const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";
const char* firebaseHost = "your-project-id.firebaseio.com";
const char* firebaseAuth = "your-auth-token";

const int micPin1 = A0;
const int micPin2 = A1;
const int micPin3 = A2;
const int buzzerPin = 9;
const int ledPin = 6;
const int contrast = 50;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
FirebaseData fbdo;

void setup() {
  pinMode(buzzerPin, OUTPUT);
```

```
pinMode(ledPin, OUTPUT);
lcd.begin(16, 2);
analogWrite(6, contrast);
Serial.begin(9600);

// Connect to Wi-Fi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi...");
}
Serial.println("Connected to WiFi");

// Initialize Firebase
Firebase.begin(firebaseHost, firebaseAuth);

// Initialize Blynk
```

```
  Blynk.begin(auth, ssid, password);
}

void loop() {
  Blynk.run(); // Needed for Blynk to run properly

  int micValue1 = analogRead(micPin1);
  int micValue2 = analogRead(micPin2);
  int micValue3 = analogRead(micPin3);
  float voltage1 = micValue1 * (5.0 / 1024.0);
  float voltage2 = micValue2 * (5.0 / 1024.0);
  float voltage3 = micValue3 * (5.0 / 1024.0);
  float dB1 = 20 * log10(voltage1 / 0.0063);
  float dB2 = 20 * log10(voltage2 / 0.0063);
  float dB3 = 20 * log10(voltage3 / 0.0063);
  float averageDB = (dB1 + dB2 + dB3) / 3;
```

```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Noise pollution: ");
  lcd.setCursor(0, 1);
  lcd.print(averageDB);

  Serial.print("Noise pollution: ");
  Serial.println(averageDB);

  // Store data in Firebase
  if (Firebase.setFloat(fbdo, "/noise_data", averageDB)) {
    Serial.println("Data stored in Firebase");
  } else {
    Serial.println("Failed to store data in Firebase");
  }
```

```
  // Control the LED and the buzzer based on the sound level
  if (averageDB > 70) {
    digitalWrite(ledPin, HIGH);
    tone(buzzerPin, 1000, 500);
  } else {
    digitalWrite(ledPin, LOW);
    noTone(buzzerPin);
  }

  delay(1000);
}
```

EXPLANATION:

Setup:

➢ In the beginning, we include some libraries which are required in

this project.

➢ Then we assign pins to which both sensors and relays are

connected.

Display:

I provided is a noise pollution monitoring system that measures the noise levels from three microphones, calculates the average noise level in decibels (dB), displays this average on a 16x2 LCD screen, and activates both an LED and a buzzer if the noise level exceeds a predefined threshold (70 dB in the code). Additionally, it sends the average noise level data to a Firebase database for remote storage and monitoring.

MOBILE APPLICATION:

• We’ve developed a mobile application that is designed to be user-

friendly and accessible via smartphones, providing a simple interface

for users.

• The ESP32 microcontroller is equipped with sensors to collect real-

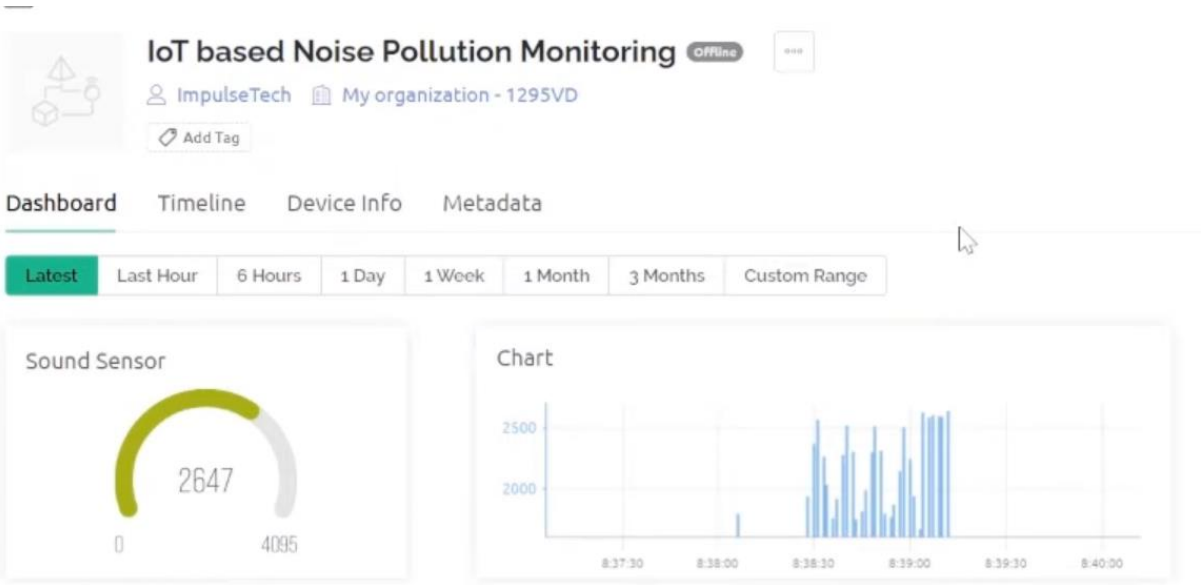
time data about Humidity,Temperature and soil moisture.

• Then connects to the mobile app through the Blynk platform, enabling

real-time data communication.

• Users can easily get real-time data through their mobile app.

BLYNK OUTPUT:



Benefits of noise pollution monitoring:

A noise pollution monitoring system offers several benefits and can have a positive impact on the environment, public health, and urban planning. Here are some of the key benefits of such a system:

1. **Improved Public Health**:
    - Identification and mitigation of noise pollution can lead to reduced stress, better sleep quality, and improved overall health for residents living in noisy areas.
2. **Urban Planning and Zoning**:
    - City planners can use noise pollution data to make informed decisions regarding land use and zoning regulations, which can lead to more balanced and harmonious urban environments.
3. **Noise Regulation Compliance**:
    - Helps authorities enforce noise regulations and noise limits, ensuring that businesses and industries comply with legal noise standards.
4. **Early Detection of Issues**:
    - Identifying noise issues early can prevent long-term environmental damage and negative health effects, allowing for quicker intervention.
5. **Noise Mapping**:
    - Noise pollution data can be used to create noise maps, which are valuable tools for urban planning, transportation, and infrastructure development.
6. **Community Engagement**:
    - Noise monitoring can engage communities in noise awareness and activism, encouraging residents to advocate for quieter neighborhoods.
7. **Traffic Management**:

- Cities can use real-time noise data to optimize traffic management, reduce congestion, and improve traffic flow to minimize noise pollution.

In summary, a noise pollution monitoring system can lead to a healthier, more sustainable, and more harmonious living environment for communities and individuals while providing valuable data for city planning, regulatory compliance, and environmental preservation.