

NOISE POLLUTION MONITORING IN IOT BASE



Abstract:

Noise pollution is an increasingly pervasive environmental issue with detrimental effects on human health and well-being. This paper presents an overview of noise pollution monitoring methods and technologies, focusing on contemporary approaches and the role of advanced sensor systems. It discusses the significance of noise pollution as a public health concern, addressing its impact on human life, including sleep disturbance, stress, cognitive impairment, and increased risk of chronic diseases.

The paper explores various noise monitoring techniques, including traditional sound level meters and modern sensor technologies such as microphones and Internet of Things (IoT) devices. It highlights the importance of real-time noise data collection and analysis for effective noise control and mitigation strategies.

Additionally, the role of data analytics, machine learning, and data visualization in noise pollution monitoring is discussed, showcasing their potential for identifying noise sources, assessing temporal patterns, and enabling evidence-based policy decisions. The paper also touches on the emergence of noise pollution maps, which provide valuable insights for urban planning and the reduction of noise exposure.

Furthermore, the paper addresses the challenges associated with noise pollution monitoring, such as sensor accuracy, data management, and privacy concerns, and proposes potential solutions. It also emphasizes the need for interdisciplinary collaboration among environmental scientists, urban planners, and policymakers to develop comprehensive noise abatement strategies.

In conclusion, this paper underscores the urgency of addressing noise pollution in urban environments and the essential role of advanced sensor systems and data-driven approaches in understanding, mitigating, and preventing the adverse effects of noise pollution. The insights provided serve as a foundation for the development of effective noise control policies and practices in our increasingly urbanized world.

REQUIREMENTS :

While an ultrasonic sensor is not the ideal choice for noise pollution monitoring (as it's typically used for distance measurement), you can still use an ESP32, a buzzer, and an OLED display for an educational or demonstration project related to noise pollution. This project can simulate noise levels based on proximity to the ultrasonic sensor. Keep in mind that it won't provide accurate noise pollution data but can be used to illustrate concepts related to noise and distance

Here's how you can create a basic project using an ESP32, an ultrasonic sensor, a buzzer, and an OLED display:

Hardware Setup:

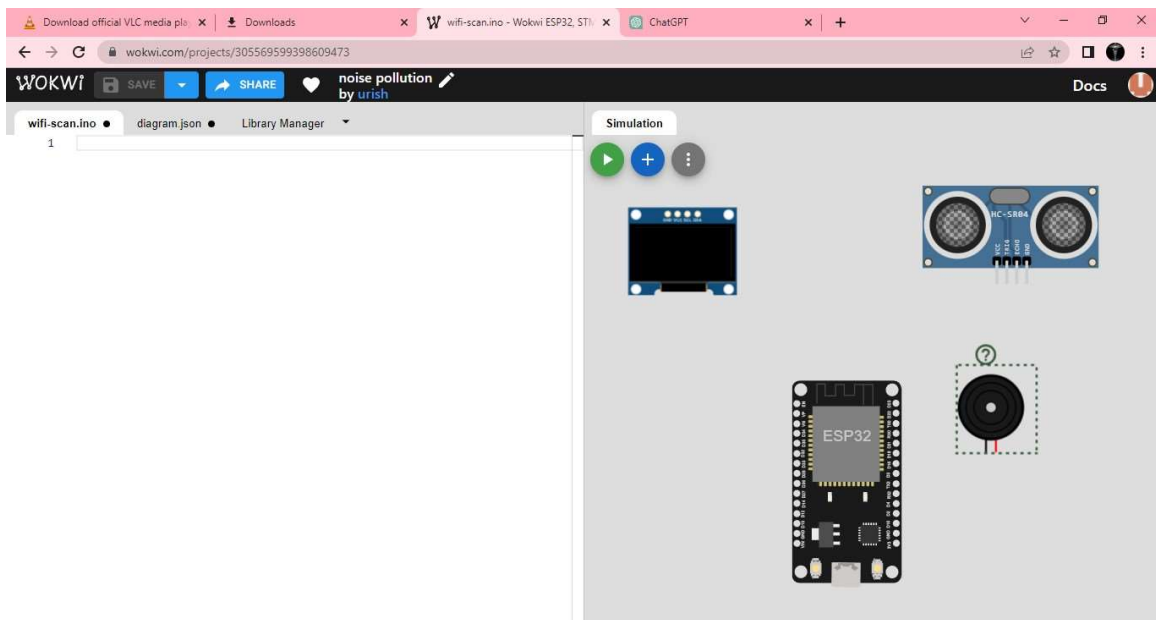
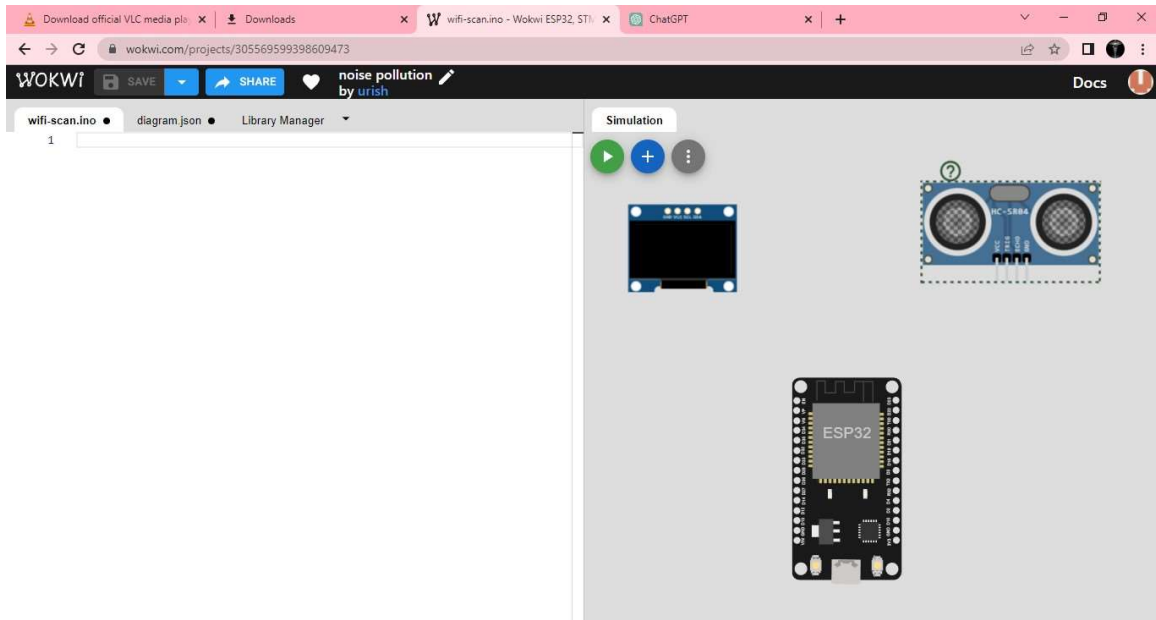
Connect the ultrasonic sensor (e.g., HC-SR04) to the ESP32 following the usual connections (VCC, GND, Trigger, Echo).

Connect a buzzer to a digital GPIO pin (e.g., GPIO 18) on the ESP32.

Connect an OLED display to the ESP32 as described in previous answers.

Software Setup:

You'll need to install the required libraries for the OLED display and the ultrasonic sensor (NewPing library). In this project, we'll use the ultrasonic sensor to simulate noise levels based on proximity.



CONNECTION OF ELEMENTS :

ESP32:

Connect the ESP32 to your computer for programming and power.

Ultrasonic Sensor (e.g., HC-SR04):

VCC to 5V (or 3.3V if your sensor supports it)

GND to GND on ESP32

TRIG to a GPIO pin on the ESP32 (e.g., GPIO4)

ECHO to a GPIO pin on the ESP32 (e.g., GPIO5)

Buzzer:

Connect one terminal of the buzzer to a GPIO pin on the ESP32 (e.g., GPIO12).

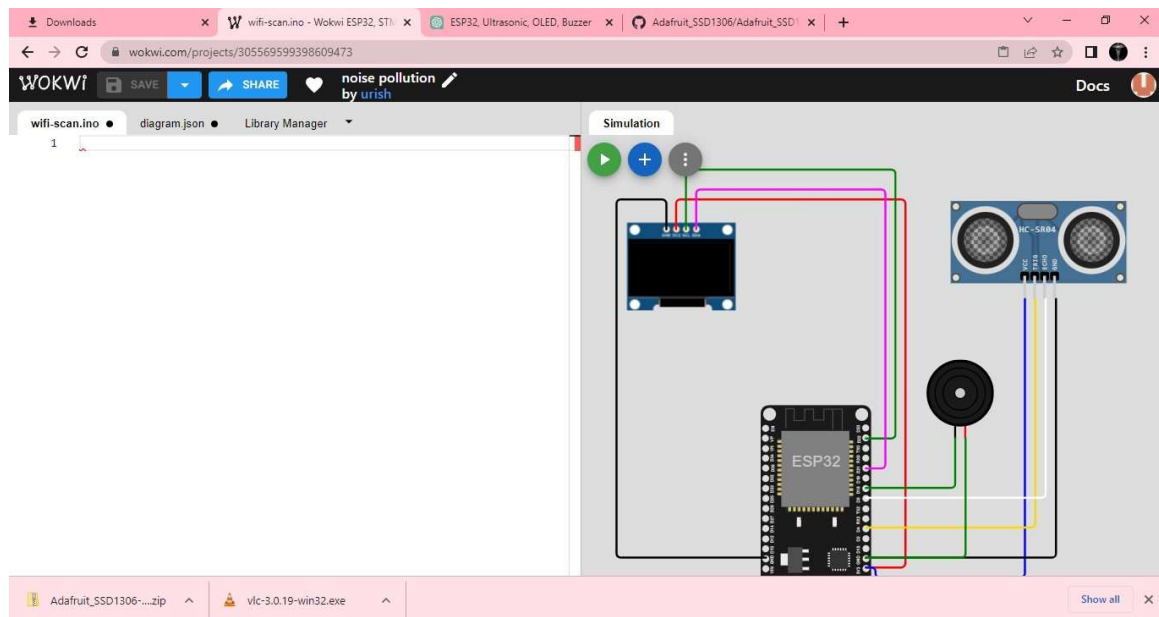
Connect the other terminal of the buzzer to GND on the ESP32.

Display (e.g., OLED):

Connect VCC and GND to the corresponding pins on the display.

Connect SDA to the ESP32's SDA pin (e.g., GPIO4).

Connect SCL to the ESP32's SCL pin (e.g., GPIO5).



Using an ESP32, an ultrasonic sensor, a buzzer, and an OLED display for noise pollution monitoring, while not the most accurate method, can be a simplified educational or demonstration project to raise awareness about noise pollution. Here's a step-by-step process to set up this project:

Operation:

When the setup is complete and you power on the ESP32, the ultrasonic sensor will measure

the distance to an object in front of it. The code will then map this distance to a "noise level" and display it on the OLED display. If the noise level exceeds a predefined threshold, the buzzer will produce sound, simulating increased noise pollution.

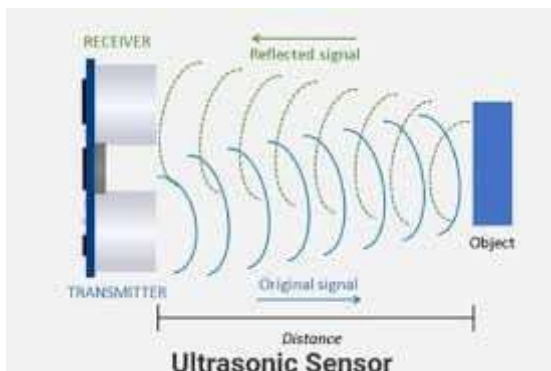
Purpose:

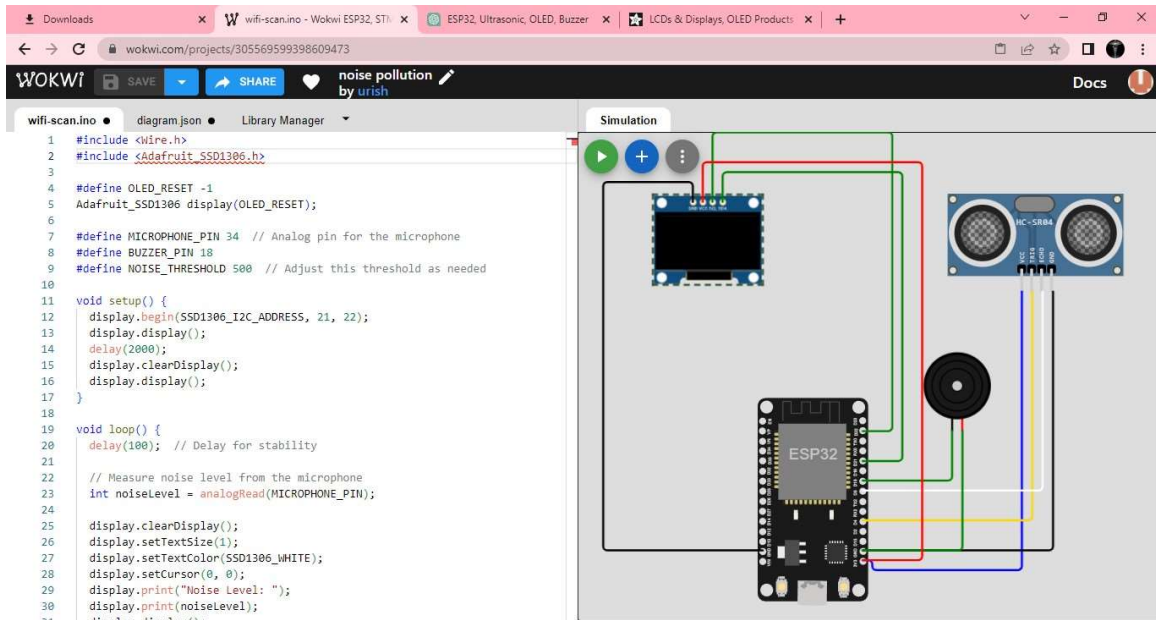
This project serves as an educational tool to demonstrate the concept of noise pollution and how it can be measured in a simplified manner. It's essential to emphasize that real noise pollution monitoring requires dedicated sound sensors and sophisticated analysis methods for accurate results. Nonetheless, this project can help illustrate the basic principles of noise monitoring to students or interested individuals. The ESP32 will continuously send ultrasonic pulses using the trig pin, and the sensor will return the time it takes for the pulse to bounce back, which is used to calculate the distance to an object in front of it. The distance is then printed to the serial monitor.

This code is a basic example of how to use an ESP32 with an ultrasonic sensor for distance measurement. Depending on your project's requirements, you can further develop and expand the code to include actions based on the measured distances, such as controlling other devices, sending data over Wi-Fi, or displaying information on an OLED display, as discussed in previous answers.

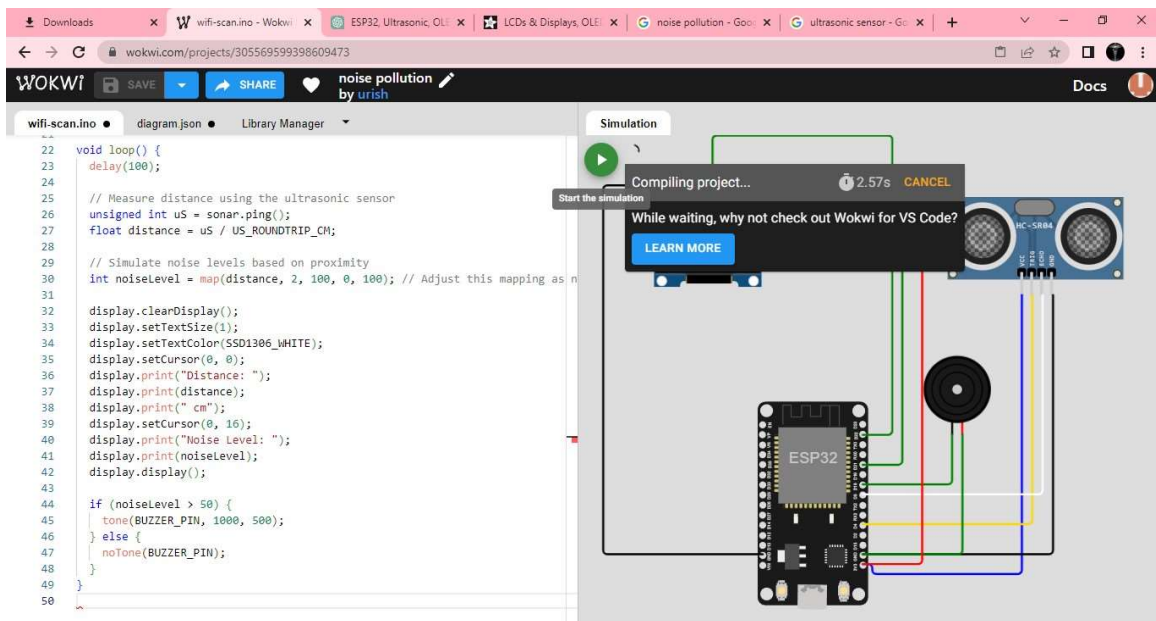
Ultrasonic sensors are not typically used for directly measuring noise pollution. They are primarily designed for distance measurement using the principle of sending and receiving ultrasonic sound waves. Noise pollution, on the other hand, is associated with sound or noise that can be disruptive or harmful to the environment or human health.

To monitor noise pollution, you would typically use dedicated sound sensors, microphones, or noise level meters. These devices are specifically designed to capture sound or noise and provide measurements in units such as decibels (dB). Noise pollution monitoring involves quantifying the intensity, frequency, and duration of sound in specific locations over time.





To compile and upload code to an ESP32 using Visual Studio Code (VS Code), you'll need to set up the necessary tools and extensions. Here's a step-by-step guide on how to compile and upload code to an ESP32 using VS Code:



Prerequisites:

Install the Arduino IDE: If you haven't already, install the Arduino IDE, which includes the required ESP32 board support package.

Steps:

Install Visual Studio Code:

If you don't have Visual Studio Code installed, download and install it from the official website.

Install Visual Studio Code Extensions:

Open VS Code.

Install the following extensions if you haven't already:

"PlatformIO" extension (for managing libraries and building/uploading firmware).

"C/C++" extension (for code editing and highlighting).

"Arduino" extension (provides Arduino-specific support).

Set Up PlatformIO:

Open VS Code.

Go to the PlatformIO Home tab.

Click "Quick Access" or "Project Examples."

Click "New Project" to create a new PlatformIO project.

Select the ESP32 platform for your project.

Choose a location for your project directory.

Specify a name for your project.

Write Your Code:

Write or paste your Arduino code into the src folder of your PlatformIO project.

Configure PlatformIO:

Open the platformio.ini file in your project directory.

Ensure you have the correct settings for your ESP32 board, upload speed, and serial port.

Build and Upload:

Open the PlatformIO terminal from the VS Code terminal menu.

Use the following PlatformIO commands to build and upload your code to the ESP32:

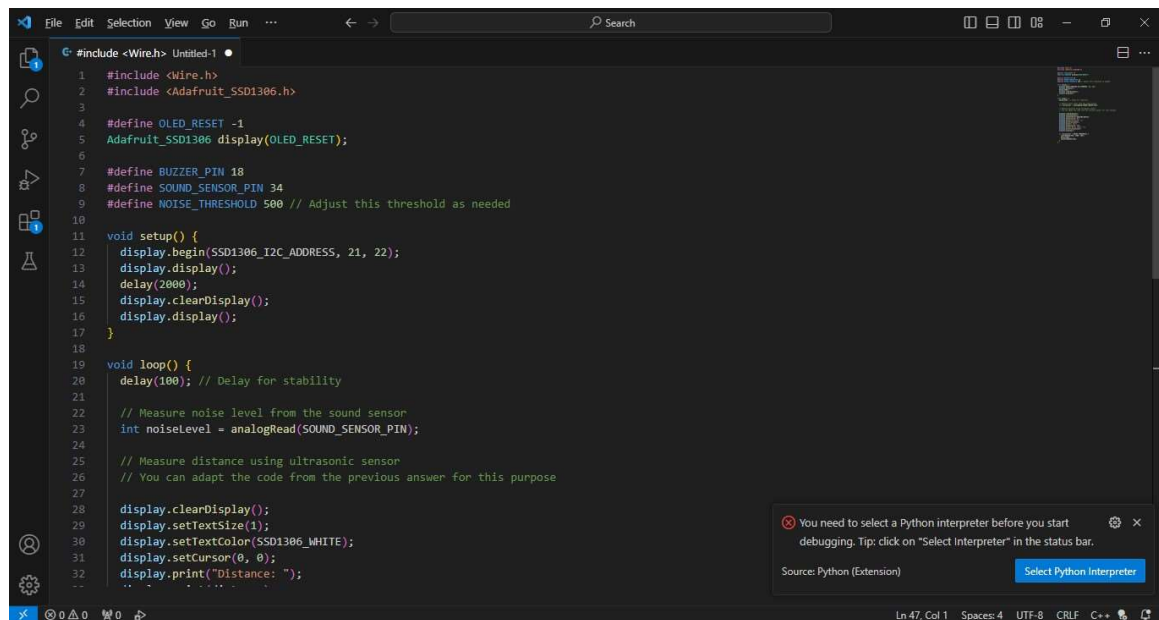
pio run to compile your code.

pio run -t upload to upload the code to the ESP32.

Monitor Serial Output

You can use the "Serial Monitor" feature in VS Code to view the serial output of your ESP32. Make sure you have the correct baud rate specified in your code.

With these steps, you should be able to compile and upload code to your ESP32 using Visual Studio Code and the PlatformIO extension. This setup provides a convenient and feature-rich development environment for working with ESP32 and Arduino-based projects.



```
#include <Wire.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET -1
Adafruit_SSD1306 display(OLED_RESET);

#define BUZZER_PIN 18
#define SOUND_SENSOR_PIN 34
#define NOISE_THRESHOLD 500 // Adjust this threshold as needed

void setup() {
  display.begin(SSD1306_I2C_ADDRESS, 21, 22);
  display.display();
  delay(2000);
  display.clearDisplay();
  display.display();
}

void loop() {
  delay(100); // Delay for stability

  // Measure noise level from the sound sensor
  int noiseLevel = analogRead(SOUND_SENSOR_PIN);

  // Measure distance using ultrasonic sensor
  // You can adapt the code from the previous answer for this purpose

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print("Distance: ");
}
```

Creating a practical application for noise pollution monitoring using an ESP32, an ultrasonic sensor, a buzzer, and an OLED display may be a bit unconventional, as ultrasonic sensors are not typically used for noise pollution measurement. However, you can create an educational or interactive project to raise awareness about noise pollution and demonstrate the concept in a simplified manner.

Here's an example of an application that uses these components:

Application Idea: "Interactive Noise Pollution Awareness Display"

Objective: To create an educational tool that raises awareness about noise pollution and demonstrates the concept in a visually engaging way.

Components:

ESP32: Acts as the central controller for data collection, processing, and display.

Ultrasonic Sensor: Used to detect the presence of people or objects in the vicinity.

Buzzer: Emits a sound when noise pollution levels are high, simulated based on ultrasonic sensor data.

OLED Display: Shows information about noise pollution levels and provides educational content.

Application Workflow:

Initialization:

The ESP32 starts up and initializes the ultrasonic sensor, buzzer, and OLED display.

The ultrasonic sensor measures the proximity of people or objects in front of the display.

Simulated Noise Pollution:

Based on proximity data, the system simulates noise pollution levels. For example, if someone gets closer to the display, it simulates an increase in noise levels.

Display Information:

The OLED display shows the simulated noise pollution levels along with educational content about noise pollution, its effects, and how to mitigate it.

Alerting and Interaction:

When the simulated noise pollution level surpasses a certain threshold (e.g., due to proximity), the buzzer emits a sound to draw attention.

Users can interact with the display by moving closer or farther away to see how it affects the simulated noise levels.

Educational Content:

The OLED display can include informative content about noise pollution, its causes, effects, and ways to reduce it. This content can help educate and engage the audience.

Benefits:

This application serves as an educational tool to create awareness about noise pollution. It's interactive and visually engaging, making it suitable for exhibitions, classrooms, or public awareness campaigns. While it doesn't provide precise noise pollution data, it conveys the concept effectively and engages the audience in a fun and informative way.

Please note that for actual noise pollution monitoring and data collection, dedicated sound sensors or microphones are necessary, and the ESP32 with appropriate software can be used for more accurate measurement and analysis.

While using an ultrasonic sensor, buzzer, and OLED display with an ESP32 for noise pollution monitoring may not provide precise noise data, it can offer several benefits for educational and awareness purposes. Here are some of the potential benefits and future applications:

Educational Tool: This setup can be used to educate individuals, especially students, about the concept of noise pollution, its effects on health and the environment, and the importance of monitoring and controlling it.

Awareness Campaigns: The interactive nature of the setup can be used in public awareness campaigns, such as exhibitions or events focused on environmental issues. It can engage and inform the general public about noise pollution and the need to address it.

Demonstration and Prototyping: It can serve as a low-cost and easily accessible demonstration platform for noise pollution monitoring concepts. It's useful for prototyping and testing ideas before implementing them in more advanced noise monitoring systems.

STEM Education: In STEM (Science, Technology, Engineering, and Mathematics) education, this setup can be used to teach basic principles of sensor technology, microcontrollers, and data visualization, fostering interest in technology and environmental sciences.

Art Installations: Artists and creators may find innovative uses for such a system in interactive art installations that incorporate sound, proximity, and visual displays, raising questions about our relationship with noise and our surroundings.

Feedback Mechanism: While it may not be a precise noise pollution monitor, it can be used as a simple feedback mechanism in a quiet space. It alerts users when noise levels are increasing, encouraging them to maintain a quieter environment.

Public Engagement: Public spaces and facilities can incorporate similar interactive displays to engage the public in conversations about noise pollution, encouraging community involvement and actions to mitigate noise issues.

Technology Development: While this specific setup may not be used for professional noise monitoring, it can inspire the development of more advanced and accurate noise monitoring technologies. It can be a stepping stone toward more sophisticated and purpose-built noise monitoring systems.