Noise Pollution Monitoring System

Phase 4: Development part-2

Abstract:

Presently, noise pollution has become a very big issue around the world. The adverse effects of this pollution include hearing impairment, negative social behavior, annoyance, sleep disturbance and intelligibility to understand people's speech. In learning context, noise can affect understanding and behavior of people and places with high noise level are not suitable for learning and teaching process. Internet of Things (Io T) technology is one of the best choices to monitor the noise or sound intensity in the environment for the safety of human being. The system will provide a real-time alert if the noise exceeds the threshold noise limit set by Environmental Department of Health standard. Equipped with an Android application, the data from the sound sensor will be transferred into the cloud server and subsequently transferred into the app for display and to enable remote monitoring. The sound level is measured for two different days during weekend and weekday. Based on Charted Institution of Building Service Engineers (CIBSE), 60dBA is the permissible ambient level and any readings that above 60dBA can interrupt speech intelligibility.

IoT-based noise pollution monitoring system using an ESP32 and Blynk involves a few steps, including hardware setup, software development, and integrating the ESP32 with the Blynk platform. Here's a high-level overview of the process:

Hardware Required:

1. ESP32 development board

2. Sound sensor (e.g., a microphone sensor)

3. Blynk-compatible device (e.g., smartphone)

4. Wi-Fi connection

Software Required:

1. Arduino IDE

2. Blynk app (available for iOS and Android)

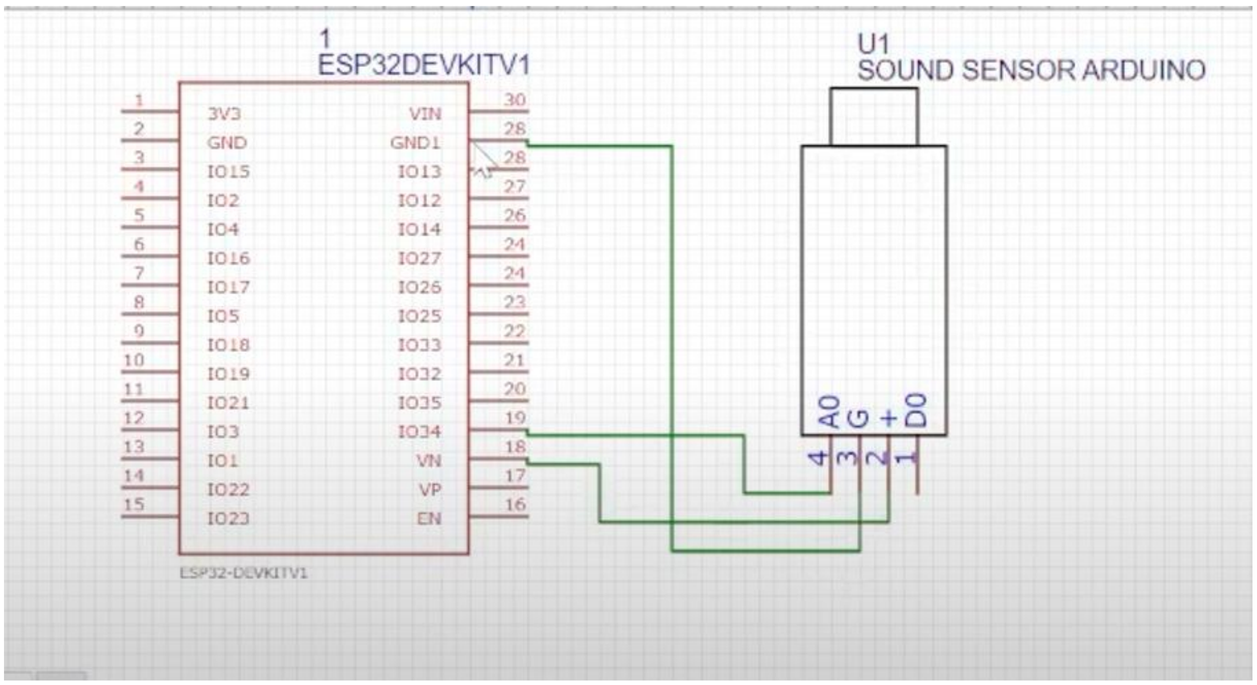
3. Blynk library for ESP32 (available through Arduino Library Manager)

Program:

```
#define BLYNK_PRINT Serial
```

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

int sound_sensor=34;
char auth[] = BLYNK_AUTH_TOKEN;

char ssid[] = "impulsetech";  // type your wifi name
char pass[] = "impulse567";  // type your wifi password

BlynkTimer timer;

void sendSensor()
{
  // Request temperature to all devices on the data line
  int sound_value=analogRead(sound_sensor);
  Serial.print("Sound sensor data: ");
  Serial.print(sound_value);
  delay(1000);

  // You can send any value at any time.
  // Please don't send more that 10 values per second.
    Blynk.virtualWrite(V0, sound_value);
    delay(500);
}
void setup()
{

   Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);
  timer.setInterval(100L, sendSensor);

  }

void loop()
{
  Blynk.run();
  timer.run();
 }
```

Circuit Diagram:



Steps:

1. Hardware Setup:- Connect the sound sensor to the ESP32. Most sound sensors have an analog output pin, which can be connected to one of the ESP32's analog pins (e.g., A0).

2. Set Up Blynk:

Install the Blynk app on your smartphone and create an account.

Create a new project in the Blynk app. You will receive an Auth Token via email.

3. Software Development:

Install the Blynk library for ESP32 through the Arduino Library Manager.

Open the Arduino IDE and create a new sketch for your ESP32 project.

4. Include Library:

```
#include <WiFi.h>

#include <BlynkSimpleEsp32.h>
```

5. Wi-Fi Configuration:

Set up your Wi-Fi credentials and Blynk Auth Token

```
  char ssid[] = "Your_WiFi_SSID";

  char pass[] = "Your_WiFi_Password";

  char auth[] = "Your_Blynk_Auth_Token";
```

6. Initialize Blynk:

```
    Blynk.begin(auth, ssid, pass);
```

7. Sensor Reading and Blynk Integration:

Read data from the sound sensor. The specific code will depend on your sensor and how it communicates with the ESP32. You'll likely use an analog pin to read the sensor data.

Send the sensor data to Blynk using the `Blynk.virtualWrite(pin, value)` function. For example:

```
int soundValue = analogRead(A0); // Replace A0 with the actual pin you're using

Blynk.virtualWrite(V1, soundValue); // Use the appropriate virtual pin (V1 in this example)
```

8. Loop Function:

In the `loop()` function, add `Blynk.run()` to keep Blynk running.

```
  void lloop() {

    Blynk.run();}
```

9. Upload the Code:

Connect your ESP32 to your computer and upload the code using the Arduino IDE.

10. Blynk App Configuration:

In the Blynk app, add a Value Display widget and assign it to the same virtual pin you used in your code (e.g., V1).

Customize the widget's settings to display the noise level data.

11. Monitor Noise Levels:

Power on your ESP32, and it should start sending noise level data to the Blynk app. You can monitor the noise pollution levels through the app.