KY-038(MICROPHONE SMALL SOUND SENSOR MODULE) INTERFACED WITH RUGGED BOARD

-Presented by
S. Manindra
SBCS India pvt ltd.

TABLE OF CONTENTS:

1 PROJECT OVERVIEW:

The project configures a KY-038 Small Sound Sensor through serial communication across 4 stages initialization, device configuration, sensor data collection, and MQTT setup inSTM32 Microcontroller and Rugged board .

It establishes serial communication which reads analog values as input and provides digital output by blniking led and configures Wi-Fi and MQTT settings, reads sensor data, and transmits sound detection status via UART.

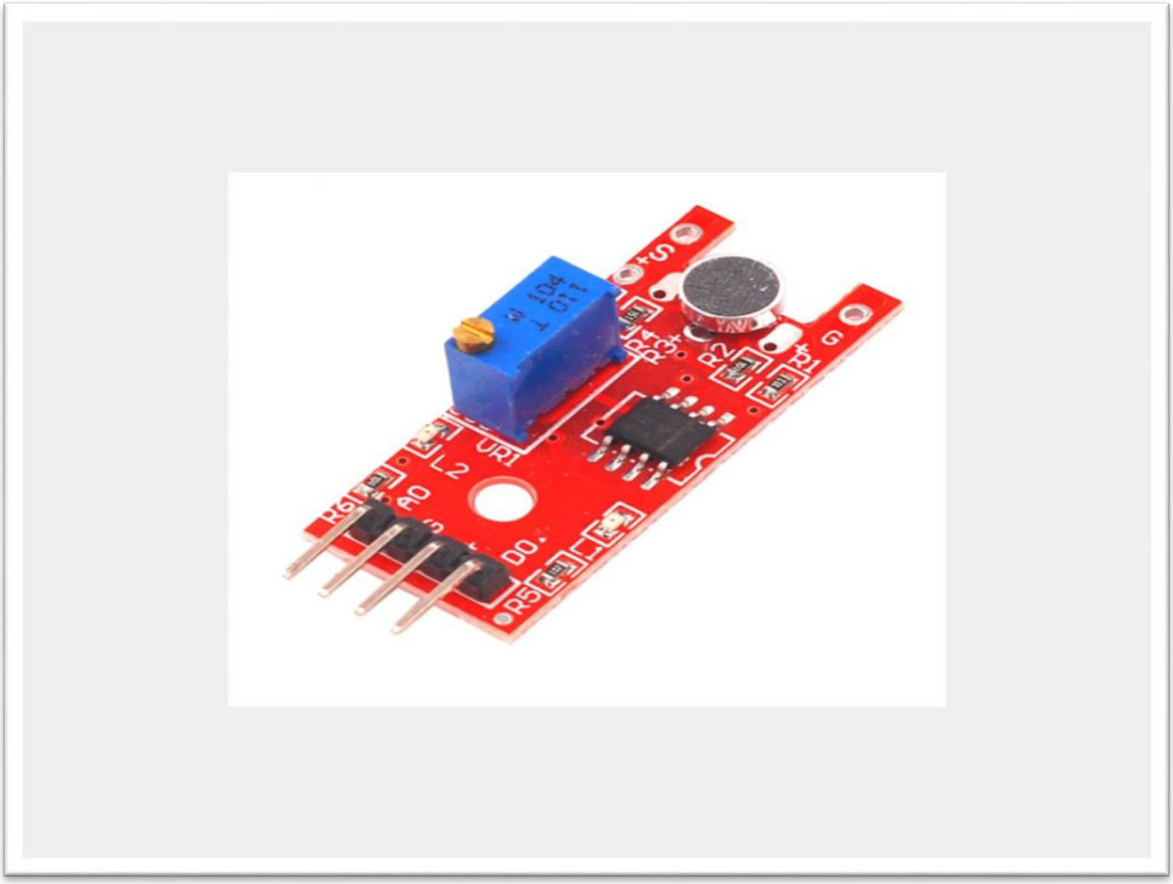
Additionally, it continuously reads serial data, transforms it into MQTT messages, and publishes it to a predefined topic. Potential improvements include enhancing error handling, optimizing code efficiency, modularizing functions, and implementing secure data handling practices for network communication. Ultimately, it orchestrates device setup, data collection, and remote monitoring through MQTT in a structured, iterative process.

2.HARDWARE COMPONENTS:

1 KY-038(SMALL SOUND SENSOR)

2 Rugged board –a5d2x

2.1. KY-038(SMALL SOUND SENSOR):



The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.

The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value.

You can control the sensitivity by adjusting the potentiometer. (Please notice: The signal will be inverted; that means that if you measure a high value, it is shown as a low voltage value at the analog output.)

This sensor doesn't show absolute values (like exact temperature in °C or magnetic field strenght in mT). It is a relative measurement: you define an extreme value to a given normal environment situation and a signal will be send if the measurement exceeds the extreme value.

It is perfect for temperature control (KY-028), proximity switch (KY-024, KY-025, KY-036), detecting alarms (KY-037, KY-038) or rotary encoder (KY-026).

Module Features:

- **Sound Detection:** Sensitive to small sound variations and changes in the surrounding environment.

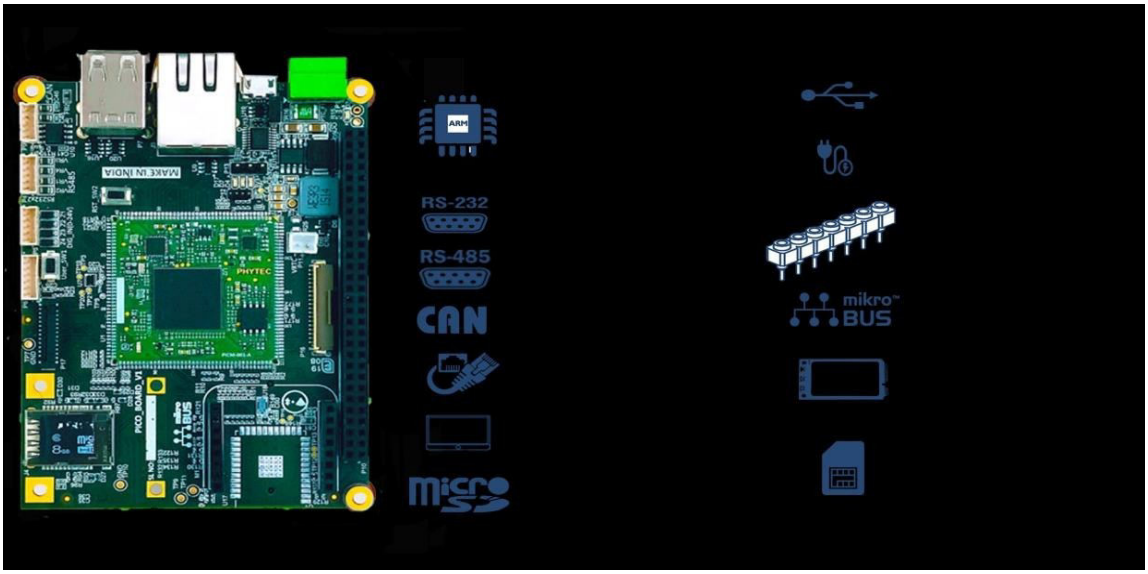
- **Analog Output:** Provides an analog output signal that varies based on the intensity of the detected sound.
- **AdjustableSensitivity:** Allows sensitivity adjustments through a potentiometer for customizing detection levels.
- **Integrated LM393 Comparator:** Equipped with an LM393 voltage comparator chip to enhance signal stability.
- **Versatile Interface:** Compatible with microcontrollers and Arduino boards through its analog output.
- **Low Power Consumption:** Operates on low power, making it suitable for energy-efficient applications.
- **Simple Integration:** Compact and easy to connect to various electronic circuits or projects.

Module Pinout:

- digital Signal = [Pin 3]
- +V = [Pin 5V]
- GND = [Pin GND]
- analog Signal = [Pin 0]

2.2. Rugged board-a5d2x :

- Rugged Board is an Open source Industrial single board computer powered by ARM Cortex-A5 SoC @500 MHz, implemented with the finest platform for rapid prototyping. The usage of System On Module over a System On Chip is the most rapid way to achieve time to market, curtail development risks for product quantities ranging from a few hundred to thousands.



- Rugged Board- A5D2x consists of Multiple Interfaces such as Ethernet, RS232, CAN, RS485, Digital Input and Digital Output with optically isolated, Standard MicroBus header for Add-On Sensors, Actuators and Multiple Wireless Modules such as ZigBee, LoRa, Bluetooth etc. MPCIe connector with USB interface used for Cloud Connectivity modules 3G, 4G, NB-IoT, WiFi. Expansion header with GPIO, UART, I2C, SPI, PWR etc.

3. SOFTWARE COMPONENTS:

3.1. MINICOM:

Minicom is a text-based serial communication program that is commonly used to connect to and communicate with devices over a serial port. It is often used for debugging and configuring devices, especially in embedded systems and projects involving microcontrollers or other hardware components. Below is an explanation of how minicom can be used in a project:

1 Installation:

- Before using minicom, you need to install it on your system. You can typically install it using your system's package manager. For example, on a Debian-based system, you can use:
- Bash
- sudo apt-get install minicom

2 Connecting to a Serial Port:

- Minicom is primarily used for serial communication, so you need to connect it to the serial port of the device youwant to communicate with. Use the following command to open minicom:
- Bash
- minicom -D /dev/ttyUSB0
- Here, /dev/ttyUSB0 is the path to the serial port. The actual port may vary depending on your system and the connected device.

3 Configuration:

- Once minicom is open, you may need to configure the serial port settings such as baud rate, data bits, stop bits,

and parity. This is often necessary to match the settings of the device you are communicating with. You can access the configuration menu by pressing Ctrl-A followed by Z.

4 Interacting with the Device:

- After configuring the serial port, you can interact with the device. Minicom allows you to send commands and receive responses. This is particularly useful for debugging purposes and for configuring devices that have a serial console.

5 Exiting Minicom:

- To exit minicom, you can use the Ctrl-A followed by X shortcut.

6 File Transfer:

- Minicom also supports file transfer using protocols like Xmodem or Ymodem. This can be useful for updating firmware or transferring files between your computer and the connected device.

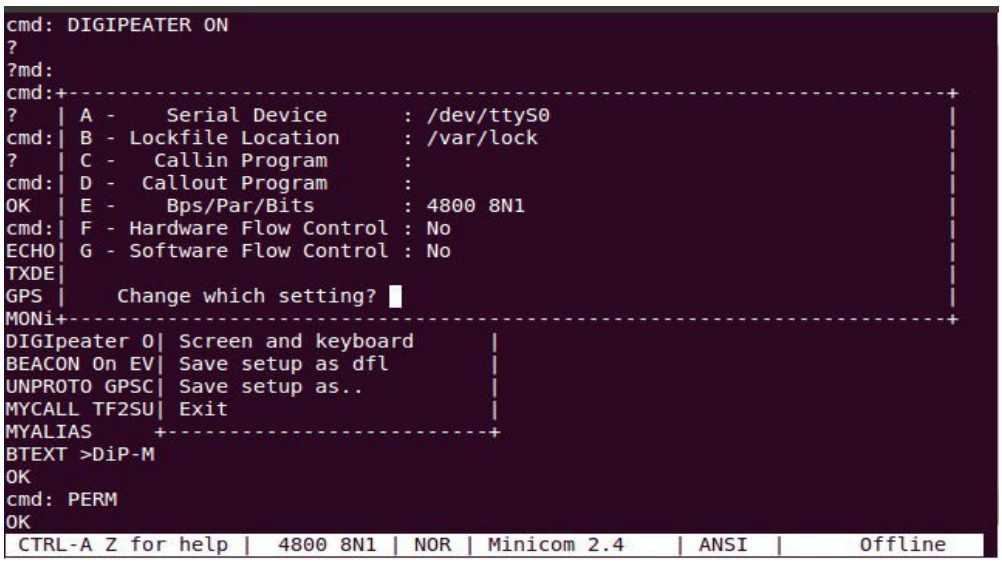


Fig: minicom

4. SOURCE CODE:

Here mentioning two types of code for KY-038 Sensor

4.1 Sys Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define DIGITAL_PIN "/sys/class/gpio/PA31/value"    // Digital pin for sound sensr
#define ANALOG_PIN "/sys/class/gpio/PD25/value"     // Analog pin for sound sensor
#define LED_PIN "/sys/class/gpio/PC13/value"      // GPIO pin for the LED

void error(const char *msg) {
    perror(msg);
    exit(1);
}

int readDigitalValue() {
    FILE *fp = fopen(DIGITAL_PIN, "r");
    if (fp == NULL) {
        error("Error opening digital pin");
    }

    char value;
```

```
    fscanf(fp, " %c", &value);
    fclose(fp);

    // '1' indicates sound detected, '0' indicates no sound
    return (value == '1');
}

float readAnalogValue() {
    FILE *fp = fopen(ANALOG_PIN, "r");
    if (fp == NULL) {
        error("Error opening analog pin");
    }

    float value;
    fscanf(fp, "%f", &value);
    fclose(fp);

    return value;
}

void toggleLED(int state) {
    FILE *ledFile = fopen(LED_PIN, "w");
    if (ledFile == NULL) {
        error("Error opening LED pin");
    }

    fprintf(ledFile, "%d", state);  // Turn LED on (state = 1) or off (state = )
```

```
    fclose(ledFile);
}

int main() {
    // Export GPIO pin 31 for digital input
    FILE *exportDigital = fopen("/sys/class/gpio/export", "w");
    if (exportDigital == NULL) {
        error("Error exporting digital GPIO pin");
    }
    fprintf(exportDigital, "25");
    fclose(exportDigital);

    // Set the digital pin direction to input
    FILE *directionDigital = fopen("/sys/class/gpio/PA31/direction", "w");
    if (directionDigital == NULL) {
        error("Error setting direction for digital GPIO pin");
    }
    fprintf(directionDigital, "in");
    fclose(directionDigital);

    // Export GPIO pin 77 for LED
    FILE *exportLED = fopen("/sys/class/gpio/export", "w");
    if (exportLED == NULL) {
        error("Error exporting LED GPIO pin");
    }
    fprintf(exportLED, "77");
    fclose(exportLED);
```

```
// Set the LED pin direction to output
FILE *directionLED = fopen("/sys/class/gpio/PC13/direction", "w");
if (directionLED == NULL) {
   error("Error setting direction for LED GPIO pin");
}
fprintf(directionLED, "out");
fclose(directionLED);

int isBlinking = 0; // Flag to track LED blinking status

while (1) {
    int soundDetected = readDigitalValue(); // Check digital pin for sound
n

   if (soundDetected) {
      printf("Sound Detected\n");
      isBlinking = !isBlinking; // Toggle the blinking flag

      if (isBlinking) {
         toggleLED(0); // Turn LED on (blink)
      } else {
         toggleLED(1); // Turn LED off (no blink)
      }
   } else {
      printf("No Sound\n");
      isBlinking = 0; // Reset blinking when there's no sound detected
```

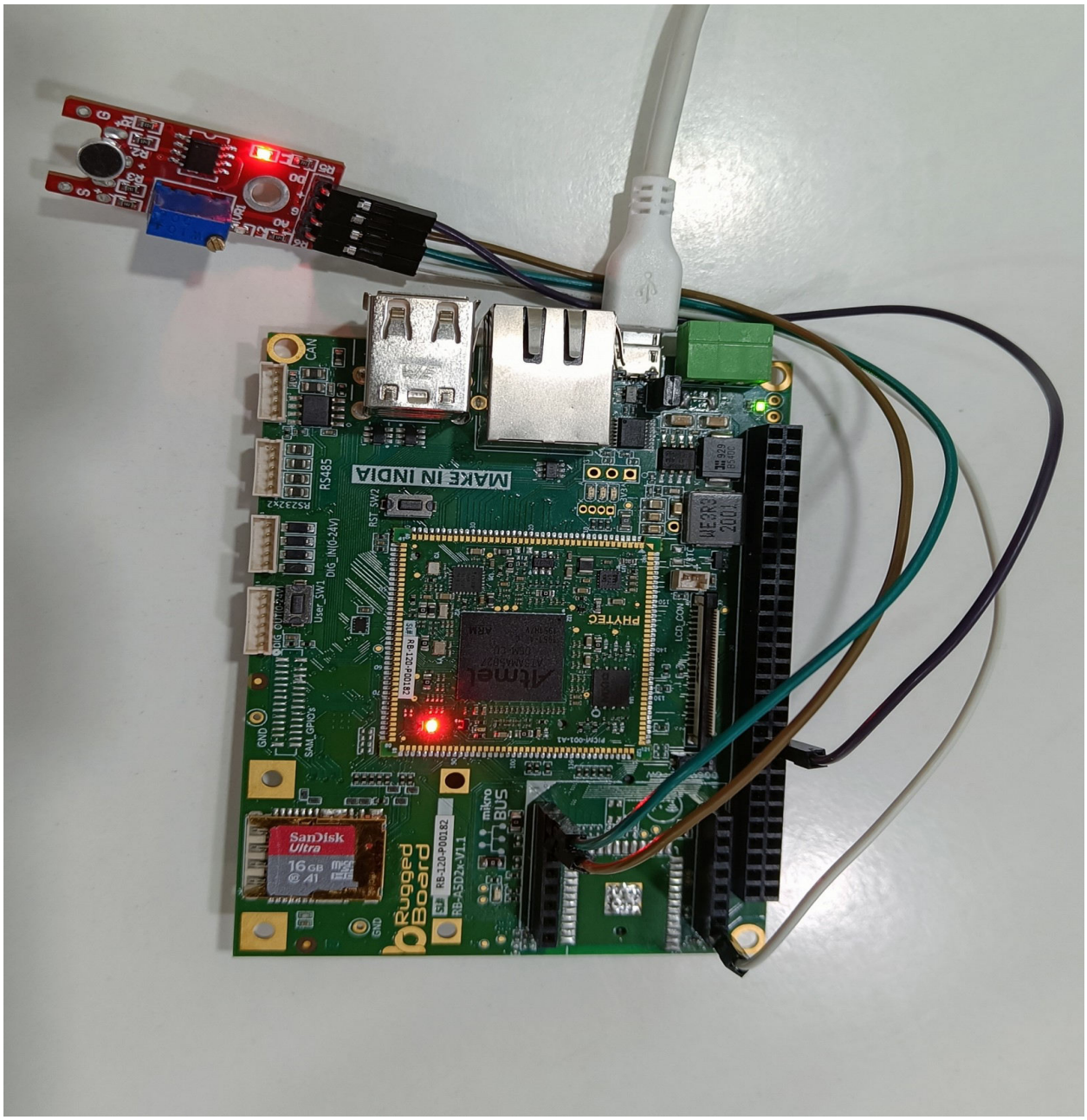
```
            toggleLED(1);   // Turn LED off (no blink)
        }

        sleep(1);  // Sleep for 1 second between readings
    }

    // Unexport GPIO pins before exiting
    FILE *unexportDigital = fopen("/sys/class/gpio/unexport", "w");
    if (unexportDigital == NULL) {
        error("Error unexporting digital GPIO pin");
    }
    fprintf(unexportDigital, "25");
    fclose(unexportDigital);

    FILE *unexportLED = fopen("/sys/class/gpio/unexport", "w");
    if (unexportLED == NULL) {
        error("Error unexporting LED GPIO pin");
    }
    fprintf(unexportLED, "77");
    fclose(unexportLED);

    return 0;
}
```

4.2 MRAA CODE:

```
#include <stdio.h>
#include <unistd.h>
#include <mraa.h>

#define DIGITAL_PIN 31  // Replace with the GPIO pin number for the sound sensor
#define LED_PIN 13      // Replace with the GPIO pin number for the LED
#define ANALOG_PIN 25   // Replace with the GPIO pin number for the analog sensor

int main() {
    mraa_init(); // Initialize MRAA library

    mraa_gpio_context digitalPin = mraa_gpio_init(DIGITAL_PIN);
    if (digitalPin == NULL) {
        fprintf(stderr, "Failed to initialize digital pin.\n");
        return 1;
    }
    mraa_gpio_dir(digitalPin, MRAA_GPIO_IN);

    mraa_gpio_context ledPin = mraa_gpio_init(LED_PIN);
    if (ledPin == NULL) {
        fprintf(stderr, "Failed to initialize LED pin.\n");
        return 1;
    }
    mraa_gpio_dir(ledPin, MRAA_GPIO_OUT);

    mraa_aio_context analogPin = mraa_aio_init(ANALOG_PIN);
    if (analogPin == NULL) {
        fprintf(stderr, "Failed to initialize analog pin.\n");
        return 1;
```

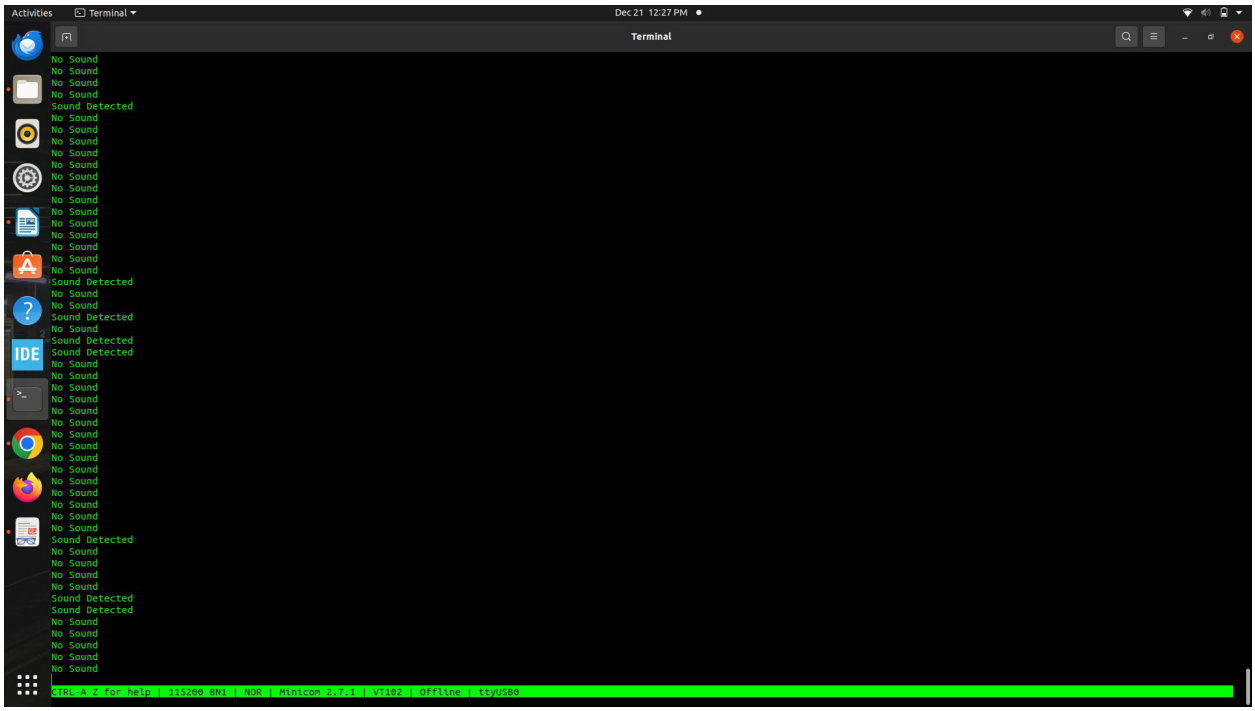
```
    }

    int isBlinking = 0; // Flag to track LED blinking status

    while (1) {
        int soundDetected = mraa_gpio_read(digitalPin);
        int analogValue = mraa_aio_read(analogPin);

        if (soundDetected) {
            printf("Sound Detected\n");
            printf("Analog Value: %d\n", analogValue); // Print analog value
            isBlinking = !isBlinking; // Toggle the blinking flag
            if (isBlinking) {
                mraa_gpio_write(ledPin, 0); // Turn LED on (blink)
            } else {
                mraa_gpio_write(ledPin, 1); // Turn LED off (no blink)
            }
        } else {
            printf("No Sound\n");
            printf("Analog Value: %d\n", analogValue); // Print analog value
            isBlinking = 0; // Reset blinking when there's no sound detected
            mraa_gpio_write(ledPin, 1); // Turn LED off (no blink)
        }
        sleep(1); // Sleep for 1 second between readings
    }
    mraa_gpio_close(digitalPin);
    mraa_gpio_close(ledPin);
    mraa_aio_close(analogPin);

    return 0;
}
```

5. CIRCUIT CONNECTION:



6. EXPECTED OUTPUT:

The expected output for the KY-038 Small Sound Sensor by using Mraa code and Sys code is shown below:



7. Advantages:

- High Sensitivity
- Wide Applicability
- Ease of Use
- Compact Size
- Low Power Consumption
- Cost-Effective

8.Disadvantages :

- Limited Range
- Sensitivity to Interference
- Calibration Challenges
- Response Time

9. Real-Time Applications:

- Sound-Activated Switches
- Security Systems
- Noise Level Monitoring
- Voice-Activated Controls
- Sound-Based Data Transmission

10.References:

• https://sensorkit.joy-it.net/en/sensors/ky-038

•https://arduinomodules.info/ky-037-high-Sensitivity-sound-detection-module/

•https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html