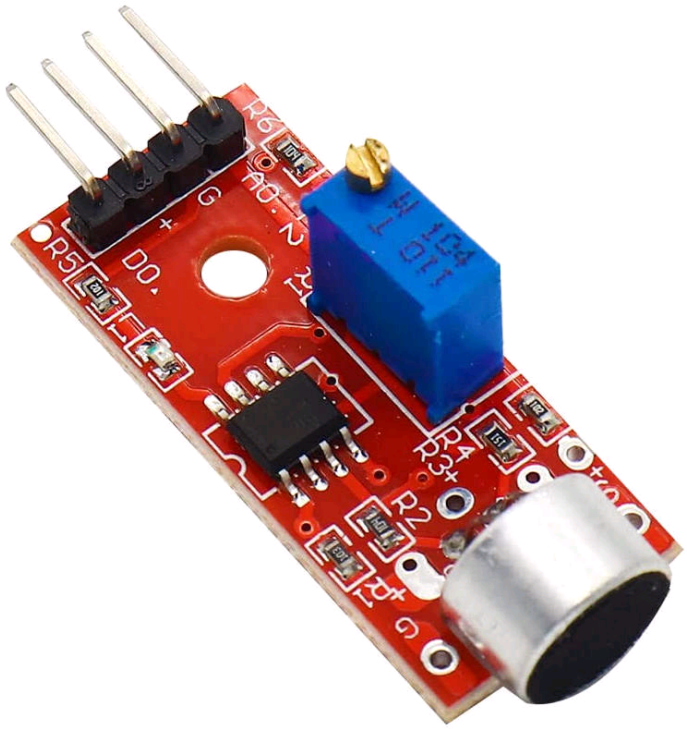
PHASE 3 :
NOISE POLLUTION MONITORING

COMPONENTS:

Sound Sensor module:



The Robocraze Sound Sensor Module is a specific sound sensor module offered by the company Robocraze, which specializes in robotics and electronics components. This module is designed to detect sound or vibrations and is likely compatible with various microcontroller platforms like Arduino, Raspberry Pi, or others.

Key features of the Robocraze Sound Sensor Module might include:

1. Microphone: It typically contains a microphone element or sensor to pick up sound.

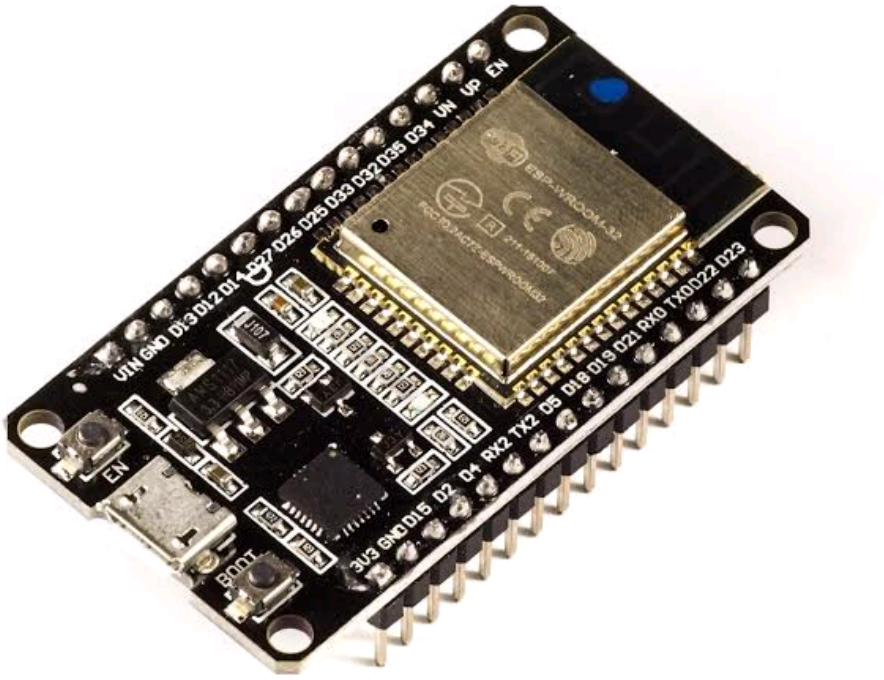
2. Preamplifier: Many sound sensor modules include a preamplifier to boost the weak audio signals from the microphone.

3. Adjustable sensitivity: Some modules allow you to adjust the sensitivity to detect specific sound levels.

4. Digital or analog output: Depending on the module, it may provide either digital (e.g., a simple "sound detected" signal) or analog output (e.g., varying voltage levels corresponding to sound intensity).

5. Compatibility: The module should be compatible with popular development boards and microcontrollers.

ESP32:



The ESP32 is a popular and powerful microcontroller and wireless communication module developed by Espressif Systems. Here are some key features of the ESP32:

Features:
1. Dual-Core Processor: The ESP32 features two processor cores, making it suitable for multitasking and more complex applications.

2. Wireless Connectivity: It supports a variety of wireless protocols, including Wi-Fi and Bluetooth, making it ideal for IoT (Internet of Things) projects.

3. Low Power Consumption: The ESP32 is designed for energy efficiency, which is crucial for battery-powered applications.

4. Numerous GPIO Pins: It has a large number of general-purpose input/output (GPIO) pins, which can be used for various tasks.

5. Integrated Sensors: Some ESP32 variants come with built-in sensors like a temperature sensor and a Hall effect sensor.

6. Rich Peripherals: It offers a wide range of peripherals like UART, I2C, SPI, and more, which make it compatible with various sensors and devices.

7. Support for Arduino IDE: You can program the ESP32 using the Arduino IDE, which simplifies the development process.

Blynk iot platform:

Blynk is an Internet of Things (IoT) platform that allows users to build and control IoT applications. It provides a set of tools and services to help developers create IoT projects with ease. Here are some key features of the Blynk IoT platform:

1. Mobile App: Blynk offers a mobile app for both Android and iOS that allows users to control and monitor their IoT devices from their smartphones.

2. Drag-and-Drop Interface: Blynk provides a user-friendly interface for building IoT applications. Users can create custom dashboards by dragging and dropping widgets to control hardware and visualize data.

3. Supported Hardware: Blynk supports a wide range of popular hardware platforms, including Arduino, Raspberry Pi, ESP8266, ESP32, and more. This makes it versatile for various IoT projects.

4. Connectivity: Blynk supports various communication protocols such as Wi-Fi, Ethernet, Bluetooth, and cellular connectivity, enabling devices to connect to the internet.

5. Cloud Integration: Blynk offers cloud services to store and retrieve data from IoT devices, making it easy to access and analyze information remotely.

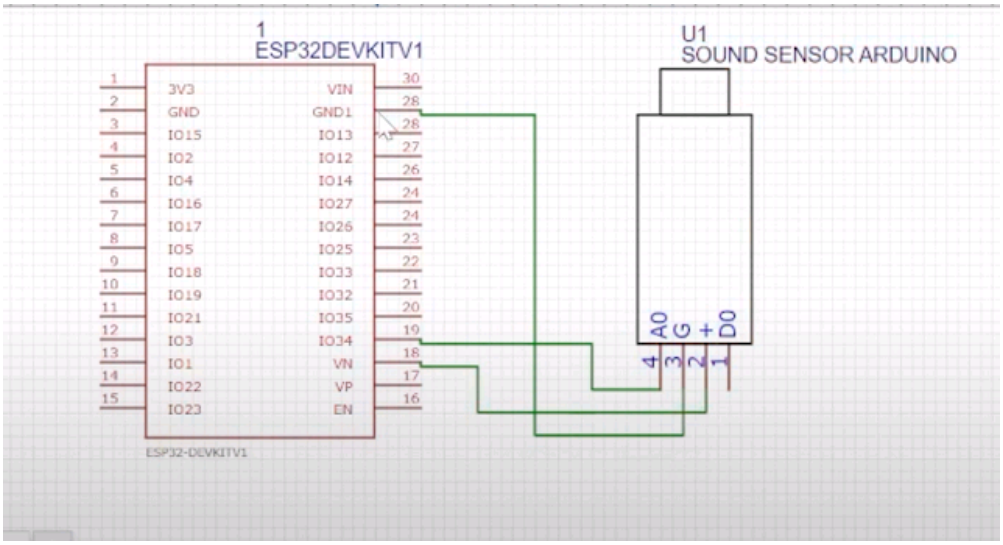
6. Libraries and Widgets: Blynk provides libraries and widgets that simplify the development of IoT applications. Widgets include buttons, sliders, graphs, and more for interacting with your devices.

7. Energy Efficiency: Blynk is designed to be energy-efficient, which is crucial for battery-powered IoT devices.

8. API and Webhooks: Developers can use Blynk's API and webhooks to integrate with other services and platforms, enabling custom automation and data sharing.

9. Security: Blynk focuses on security by providing secure connections and data encryption for IoT applications.

Circuit diagram:



Basic Python code:

```
import BlynkLib
from machine import Pin, ADC
import time

# Initialize Blynk
BLYNK_AUTH = 'Your_Blynk_Auth_Token'
blynk = BlynkLib.Blynk(BLYNK_AUTH)

# Initialize the sound sensor
sound_sensor = ADC(Pin(36))  # GPIO pin 36 on ESP32

def get_noise_level():
    # Read data from the sound sensor (0-4095 range)
    sound_value = sound_sensor.read()

    # Map the sensor value to a 0-100 range for Blynk's level widget
    noise_level = int(sound_value / 40.95)

    return noise_level

# Blynk virtual pin handler
@blynk.VIRTUAL_WRITE(1)
def v1_write_handler(value):
    # This function will be called when the Blynk app sets the virtual pin V1
    pass

while True:
    try:
        noise_level = get_noise_level()
```

```
        # Send the noise level to Blynk using virtual pin V1
        blynk.virtual_write(1, noise_level)

        time.sleep(10)  # Adjust the sampling interval as needed

    except KeyboardInterrupt:
        break

# Run the Blynk event loop
blynk.run()
```