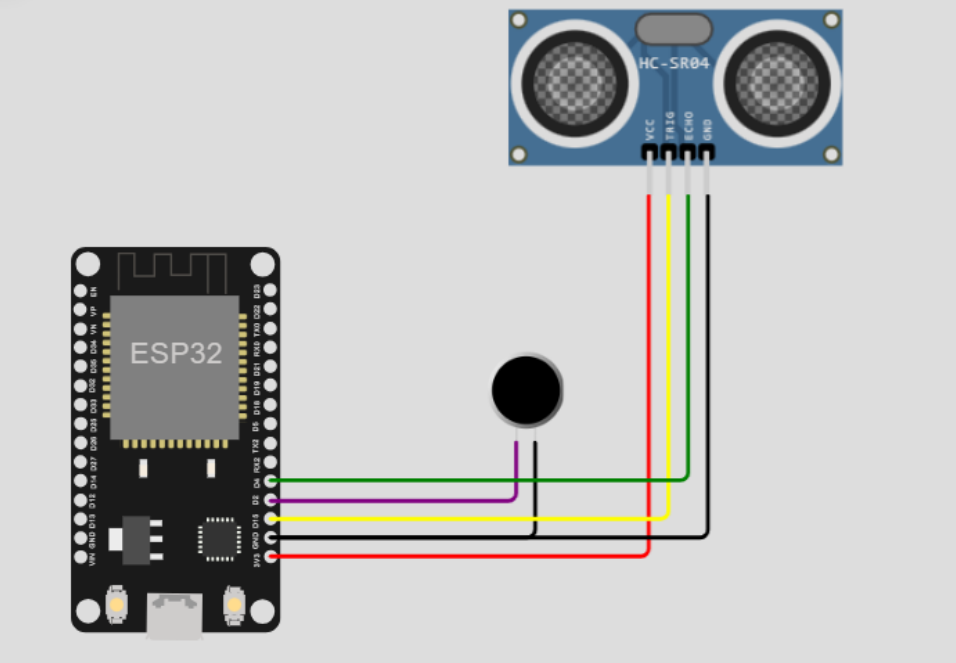
NOISE POLLUTION MONITORING
PHASE – 3 PROJECT

CIRCUIT DIAGRAM – MODEL:



CODING:

```
import machine
import time
import urequests
import ujson
import network
import math
# Define your Wi-Fi credentials
wifi_ssid = 'Surendran69'
wifi_password = '22112000'  # Replace with the actual Wi-Fi password
# Connect to Wi-Fi
```

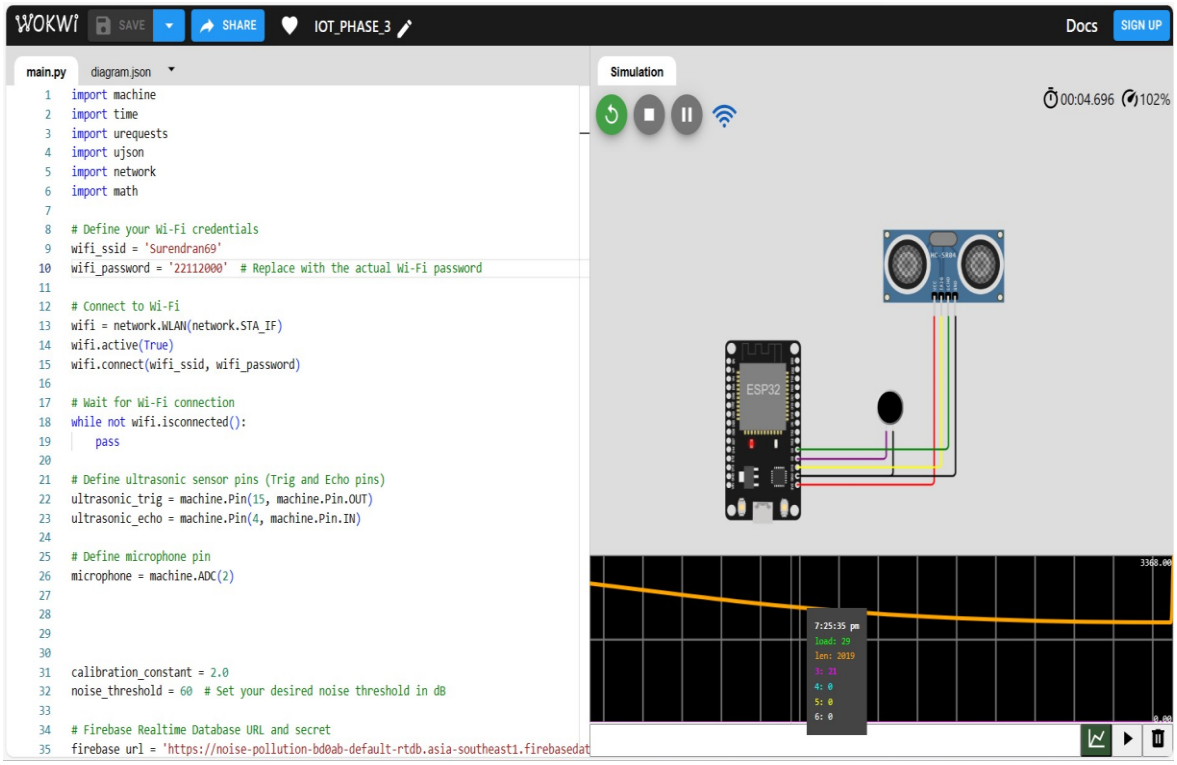
```
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)
# Wait for Wi-Fi connection
while not wifi.isconnected():
    pass
# Define ultrasonic sensor pins (Trig and Echo pins)
ultrasonic_trig = machine.Pin(15, machine.Pin.OUT)
ultrasonic_echo = machine.Pin(4, machine.Pin.IN)
# Define microphone pin
microphone = machine.ADC(2)
calibration_constant = 2.0
noise_threshold = 60  # Set your desired noise threshold in dB
# Firebase Realtime Database URL and secret
firebase_url = 'https://noise-pollution-bd0ab-default-rtdb.asia-
southeast1.firebasedatabase.app/'  # Replace with your Firebase URL
firebase_secret = 'nBsgyQFTqHUe4qExlaZX6VL3mpf5gn6BlpnMiuR0'  # Replace with
your Firebase secret
def measure_distance():
    # Trigger the ultrasonic sensor
    ultrasonic_trig.value(1)
    time.sleep_us(10)
    ultrasonic_trig.value(0)
    # Measure the pulse width of the echo signal
    pulse_time = machine.time_pulse_us(ultrasonic_echo, 1, 30000)
    # Calculate distance in centimeters
    distance_cm = (pulse_time / 2) / 29.1
    return distance_cm
def measure_noise_level():
    # Read analog value from the microphone
    noise_level = microphone.read()
```

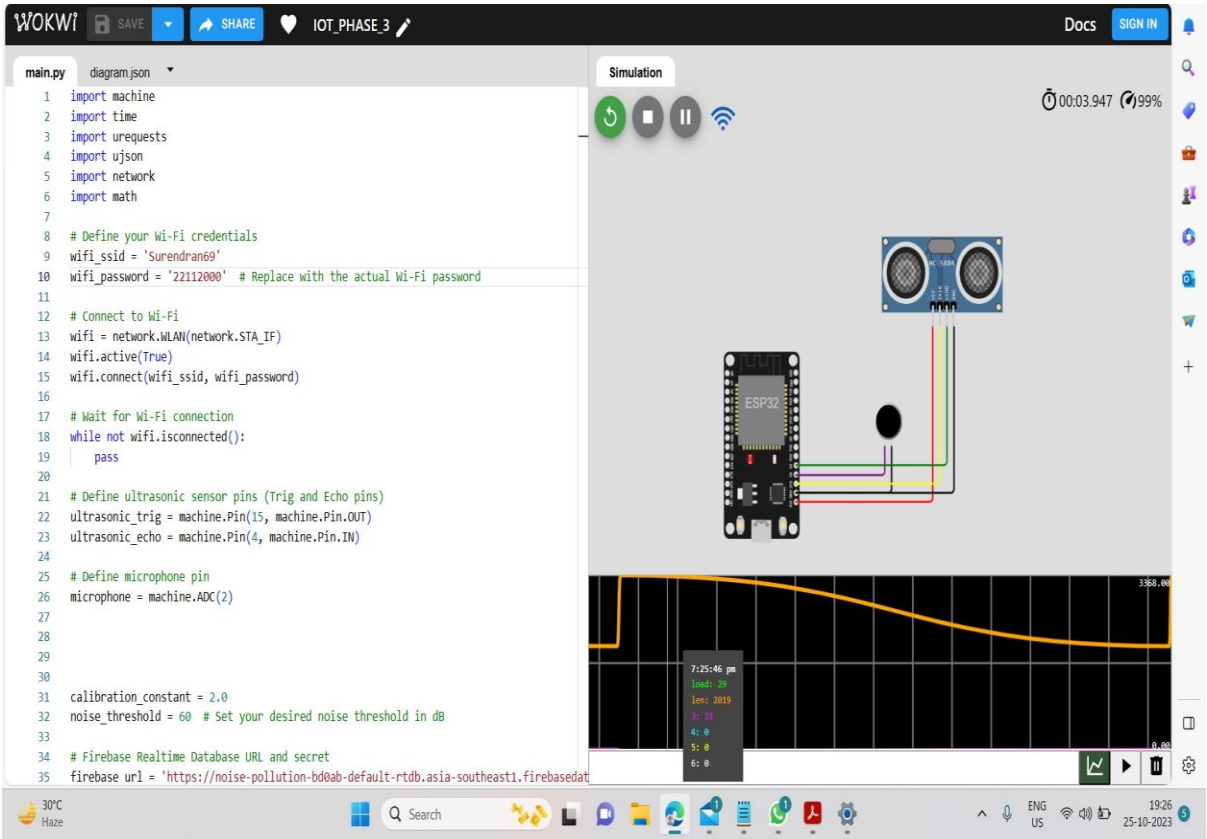
```
    noise_level_db = 20 * math.log10(noise_level / calibration_constant)
    return noise_level, noise_level_db
# Function to send data to Firebase
def send_data_to_firebase(distance, noise_level_db):
    data = {
        "Distance": distance,
        "NoiseLevelDB": noise_level_db
    }
    url = f'{firebase_url}/sensor_data.json?auth={firebase_secret}'
    try:
        response = urequests.patch(url, json=data)  # Use 'patch' instead of 'put'
        if response.status_code == 200:
            print("Data sent to Firebase")
        else:
            print(f"Failed to send data to Firebase. Status code: {response.status_code}")
    except Exception as e:
        print(f"Error sending data to Firebase: {str(e)}")
try:
    while True:
        distance = measure_distance()
        noise_level, noise_level_db = measure_noise_level()
        print("Distance: {} cm, Noise Level: {:.2f} dB".format(distance, noise_level_db))
        if noise_level_db > noise_threshold:
            print("Warning: Noise pollution exceeds threshold!")
        # Send data to Firebase
        send_data_to_firebase(distance, noise_level_db)
        time.sleep(1)  # Adjust the sleep duration as needed
except KeyboardInterrupt:
    print("Monitoring stopped")
```

DIAGRAM.JSON:


```
{
  "version": 1,
  "author": "Gokul Raja",
  "editor": "wokwi",
  "parts": [
    {
      "type": "wokwi-esp32-devkit-v1",
      "id": "esp",
      "top": -129.7,
      "left": 119.8,
      "attrs": { "env": "micropython-20231005-v1.21.0" }
    },
    { "type": "wokwi-microphone", "id": "mic", "top": -132.18, "left": 330.99, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "mic:2", "esp:GND.1", "green", [ "v0" ] ],
    [ "mic:1", "esp:D2", "green", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

OUTPUT:


WOKWI
SAVE
SHARE
IOT_PHASE_3
Docs
SIGN UP
main.py
diagram.json
Simulation
00:04.696
102%
ESP32
7:25:35 pm
load: 29
len: 2019
3: 21
4: 0
5: 0
6: 0


WOKWI
SAVE
SHARE
IOT_PHASE_3
Docs
SIGN IN
main.py
diagram.json
Simulation
00:03.947
99%
ESP32
7:25:46 pm
load: 29
len: 2019
3: 21
4: 0
5: 0
6: 0
30°C
Haze
Search
ENG
US
19:26
25-10-2023