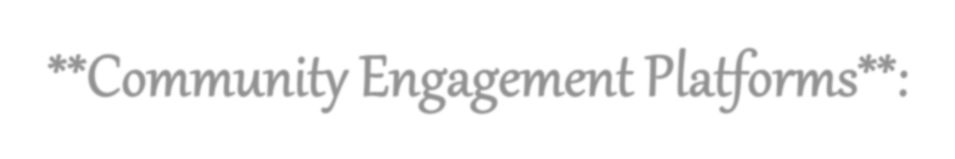
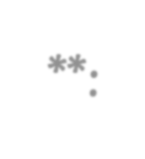
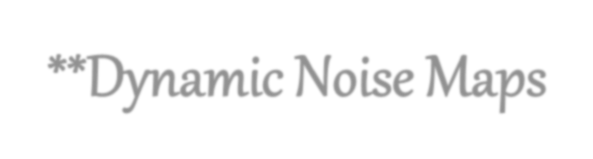
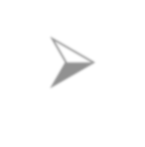


* Develop smart noise sensors equipped with data analytics capabilities that can be deployed across a city.
* Build online platforms or mobile apps that allow residents to report noise disturbances.
* Combine this crowdsourced data with sensor data for a comprehensive understanding of noise issues and their spatial distribution.

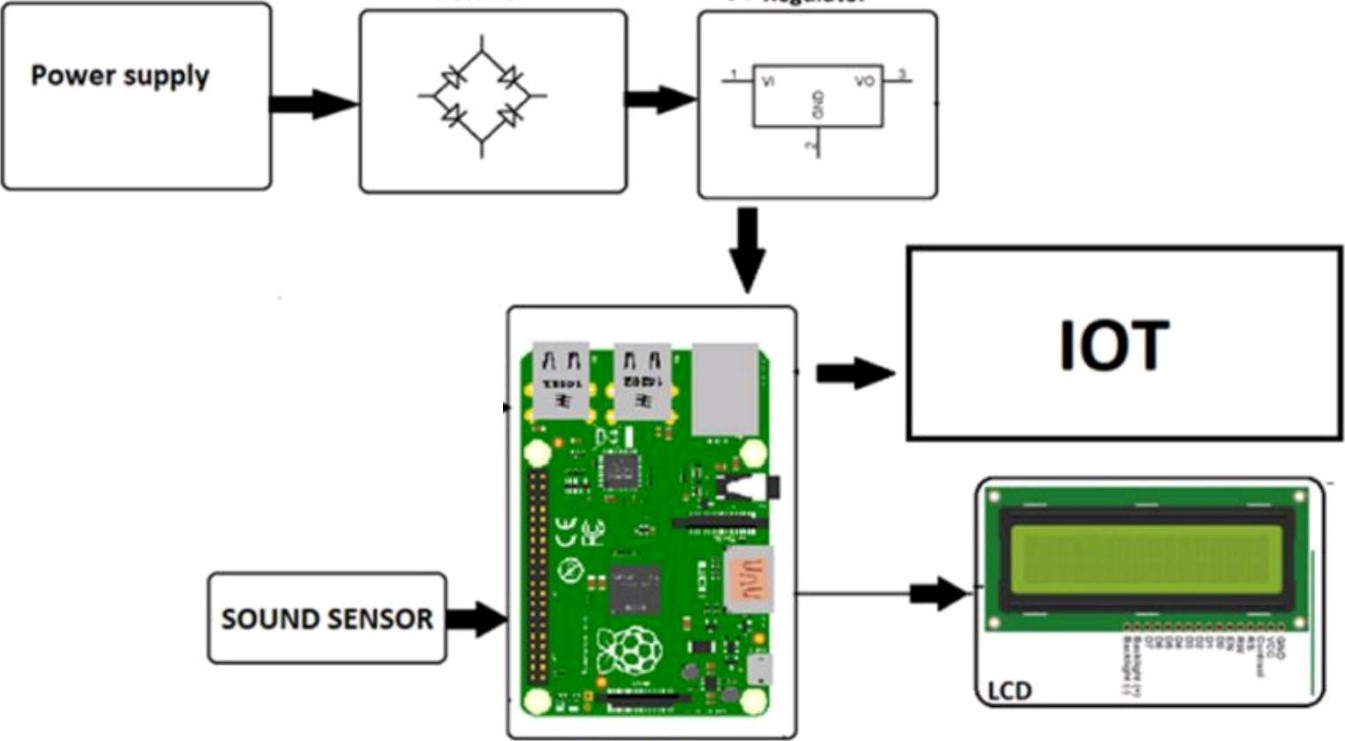


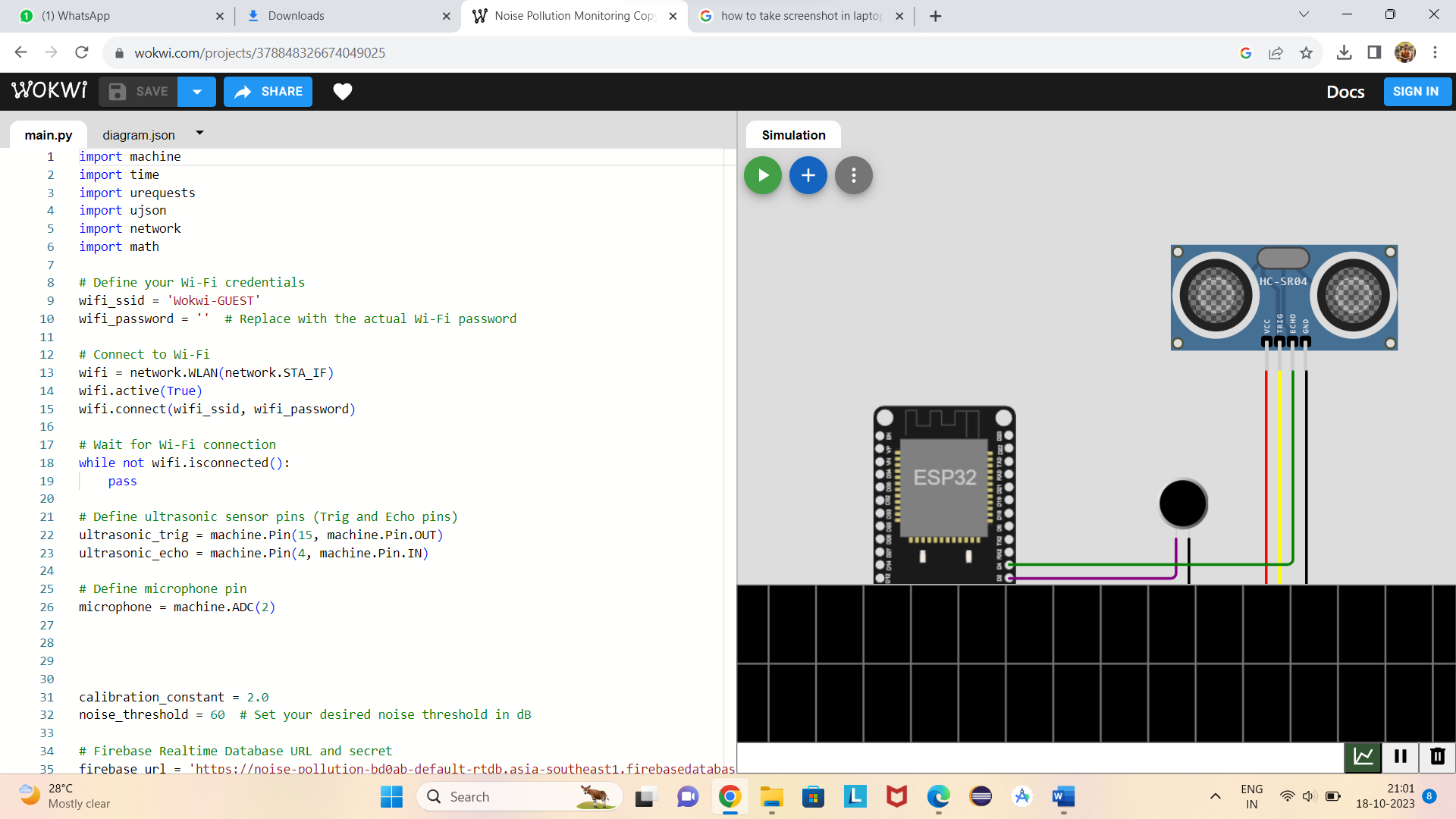
Create dynamic noise maps that update in real-time or on a frequent basis, providing policymakers and residents with up-to-date information about noise levels and sources.

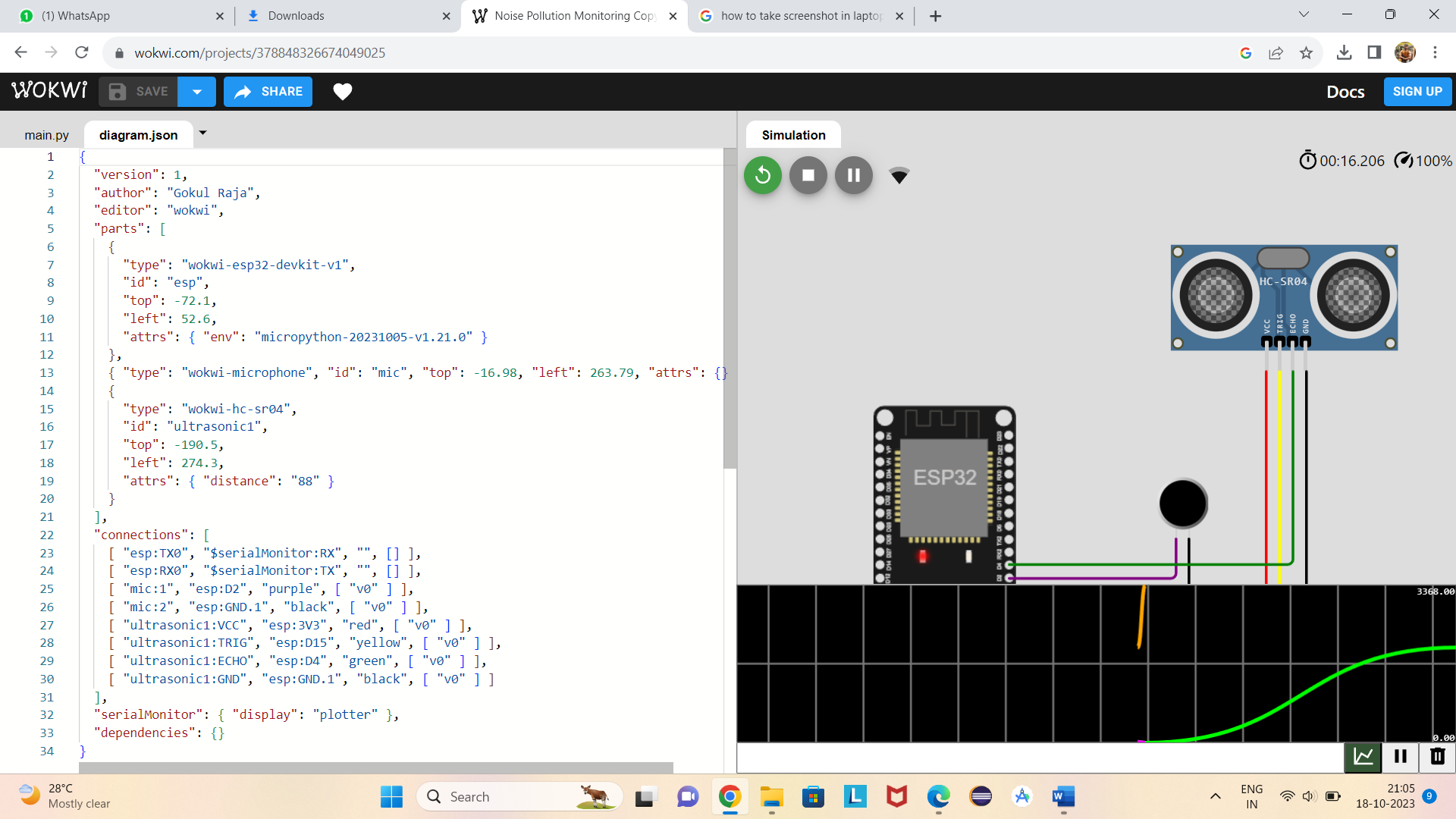


s.

Design principle:







CODE FOR NOISE POLLUTION MONITORING:

Import machine

import time

import urequests

import ujson

import network

import math

# Define your Wi-Fi credentials

wifi\_ssid = 'Wokwi-GUEST'

wifi\_password = ''  # 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")

