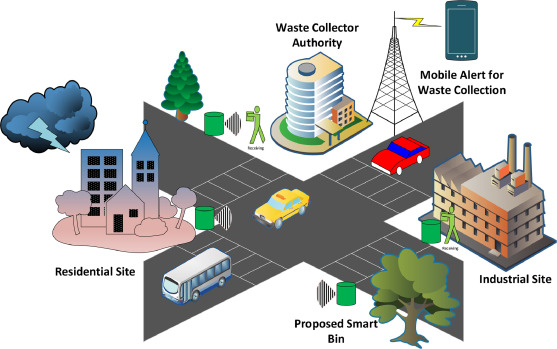
**PROJECT DEVELOPMENT PART 2**

**SMART WASTE MANAGEMENT**

****

**TEAM DETAILS**

|  |  |
| --- | --- |
| Mentor | Mrs M.Maheshwari |
| Leader | R.Abinaya |
| Members | K.Aruna  A.James Soosanna  J.Kaviya  K.Keerthana |
| Problem Description | We will start to build the IoT simulation of another part for waste management. |

**SIMULATION STEPS:**

STEP 1:Access Wowki

* Go to the website(<https://wowki.com>)

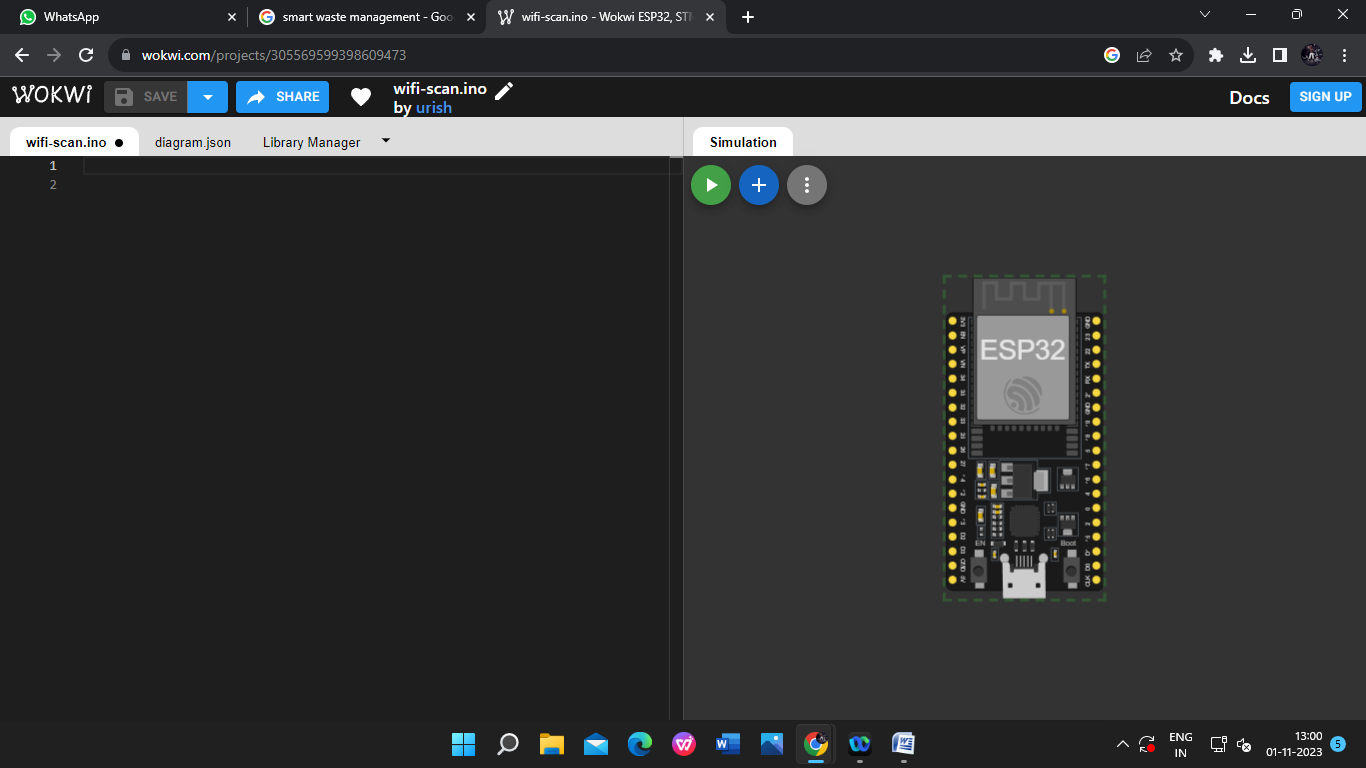
STEP 2: Create a Project

* Click on the new project

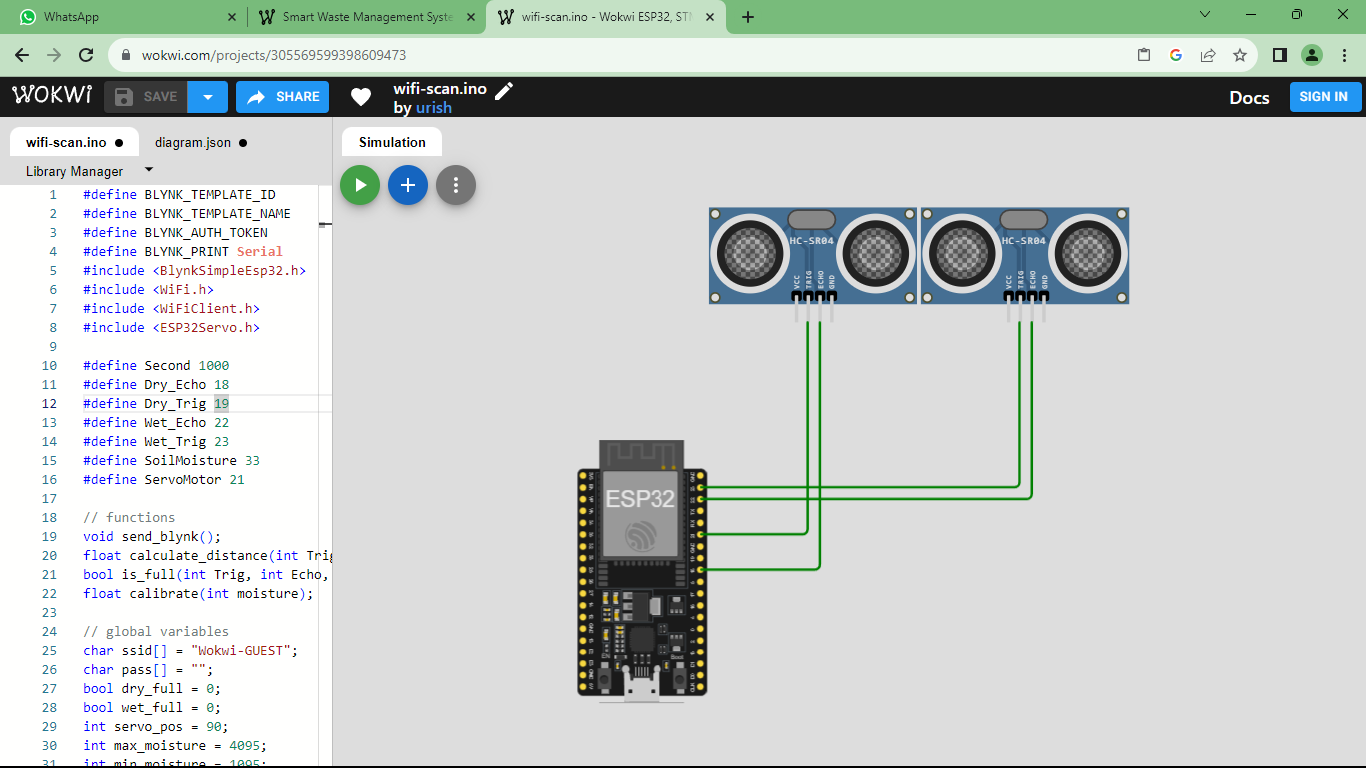
STEP 3: Add component

* In the component panel search for a “ESP32”

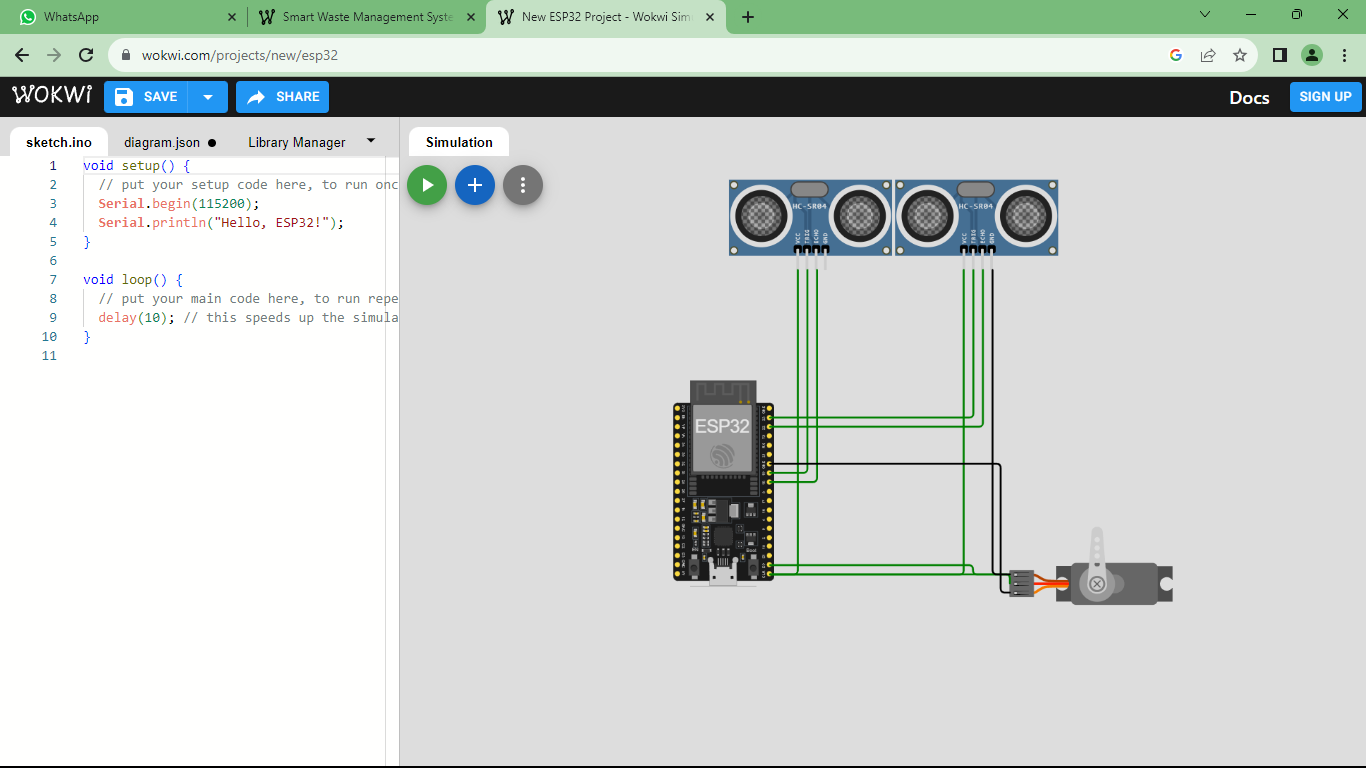
and drag it onto the virtual breadboard.



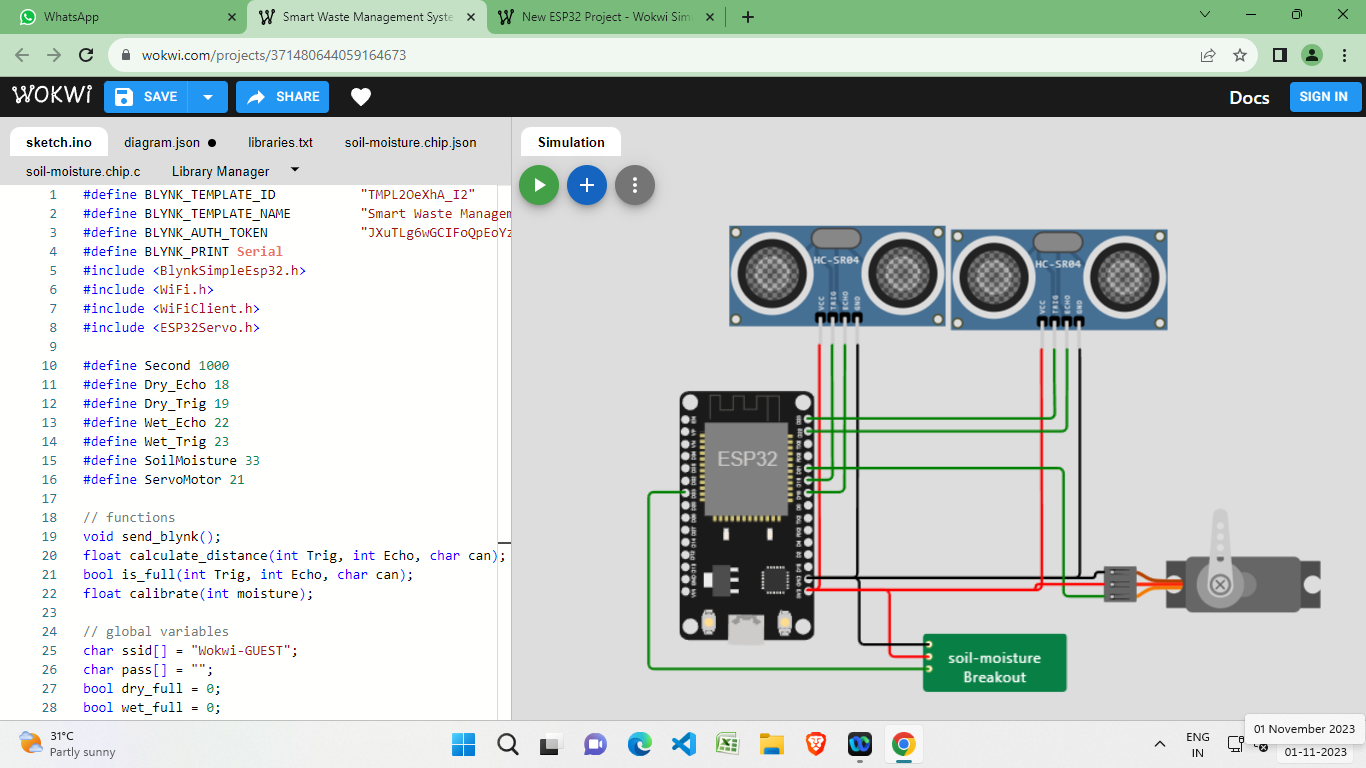
STEP 4: Add two Ultrasonic sensors and make the connection with gthe ESP32.



STEP5: Add the Servo motor to the virtual board and make the required connections.



STEP 6: Now add the soil-moisture break out to the virtual board.



STEP 7: CODE

#define BLYNK\_TEMPLATE\_ID “TMPL20eXhA\_I2”

#define BLYNK\_TEMPLATE\_NAME "Smart Waste Management System"

#define BLYNK\_AUTH\_TOKEN "JXuTLg6wGCIFoQpEoYzRzgWtFwD1LD9s"

#define BLYNK\_PRINT Serial

#include <BlynkSimpleEsp32.h>

#include <WiFi.h>

#include <WiFiClient.h>

#include <ESP32Servo.h>

#define Second 1000

#define Dry\_Echo 18

#define Dry\_Trig 19

#define Wet\_Echo 22

#define Wet\_Trig 23

#define SoilMoisture 33

#define ServoMotor 21

// functions

void send\_blynk();

float calculate\_distance(int Trig, int Echo, char can);

bool is\_full(int Trig, int Echo, char can);

float calibrate(int moisture);

// global variables

char ssid[] = "Wokwi-GUEST";

char pass[] = "";

bool dry\_full = 0;

bool wet\_full = 0;

int servo\_pos = 90;

int max\_moisture = 4095;

int min\_moisture = 1095;

float threshold = calibrate(4000);

BlynkTimer timer;

Servo servo;

void setup() {

Serial.begin(115200);

Blynk.begin(BLYNK\_AUTH\_TOKEN, ssid, pass);

pinMode(Dry\_Trig, OUTPUT);

pinMode(Dry\_Echo, INPUT);

pinMode(Wet\_Trig, OUTPUT);

pinMode(Wet\_Echo, INPUT);

pinMode(SoilMoisture, INPUT);

servo.attach(ServoMotor, 500, 2400);

timer.setInterval(1000, send\_blynk);

}

void loop() {

Serial.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

// manage trash

int moisture\_value = analogRead(SoilMoisture);

float moisture = calibrate(moisture\_value);

Serial.print("Moisture : ");

Serial.print(moisture\_value);

Serial.print(" ----> calibrated : ");

Serial.print(moisture);

Serial.println("%");

bool wet\_trash = moisture > threshold;

if(wet\_trash){

Serial.print("The trash is wet! (threshold = ");

Serial.print(threshold);

Serial.println("%)");

// open gate

for (servo\_pos = 90; servo\_pos >= 0; servo\_pos -= 1) {

servo.write(servo\_pos);

delay(15);

}

delay(5 \* Second);

// close gate

for (servo\_pos = 0; servo\_pos <= 90; servo\_pos += 1) {

servo.write(servo\_pos);

delay(15);

}

}

// manage can

dry\_full = is\_full(Dry\_Trig, Dry\_Echo, 'D');

wet\_full = is\_full(Wet\_Trig, Wet\_Echo, 'W');

if (dry\_full){

Serial.println("The dry can is full.");

}

if (wet\_full){

Serial.println("The wet can is full.");

}

Blynk.run();

timer.run();

delay(10 \* Second);

}

void send\_blynk(){

if (wet\_full){

Blynk.logEvent("wet\_can\_is\_full") ;

Blynk.virtualWrite(V1, 1);

}else{

Blynk.virtualWrite(V1, 0);

}

if (dry\_full){

Blynk.logEvent("dry\_can\_is\_full") ;

Blynk.virtualWrite(V0, 1);

}else{

Blynk.virtualWrite(V0, 0);

}

}

float calculate\_distance(int Trig, int Echo, char can){

digitalWrite(Trig, LOW);

delay(2);

digitalWrite(Trig, HIGH);

delay(10);

digitalWrite(Trig, LOW);

int duration = pulseIn(Echo, HIGH);

float distance = duration \* 0.034 /2;

Serial.print(can);

Serial.print(" free space : ");

Serial.print(distance);

Serial.println(" CM");

return distance;

}

bool is\_full(int Trig, int Echo, char can){

float distance = calculate\_distance(Trig, Echo, can);

return distance < 4;

}

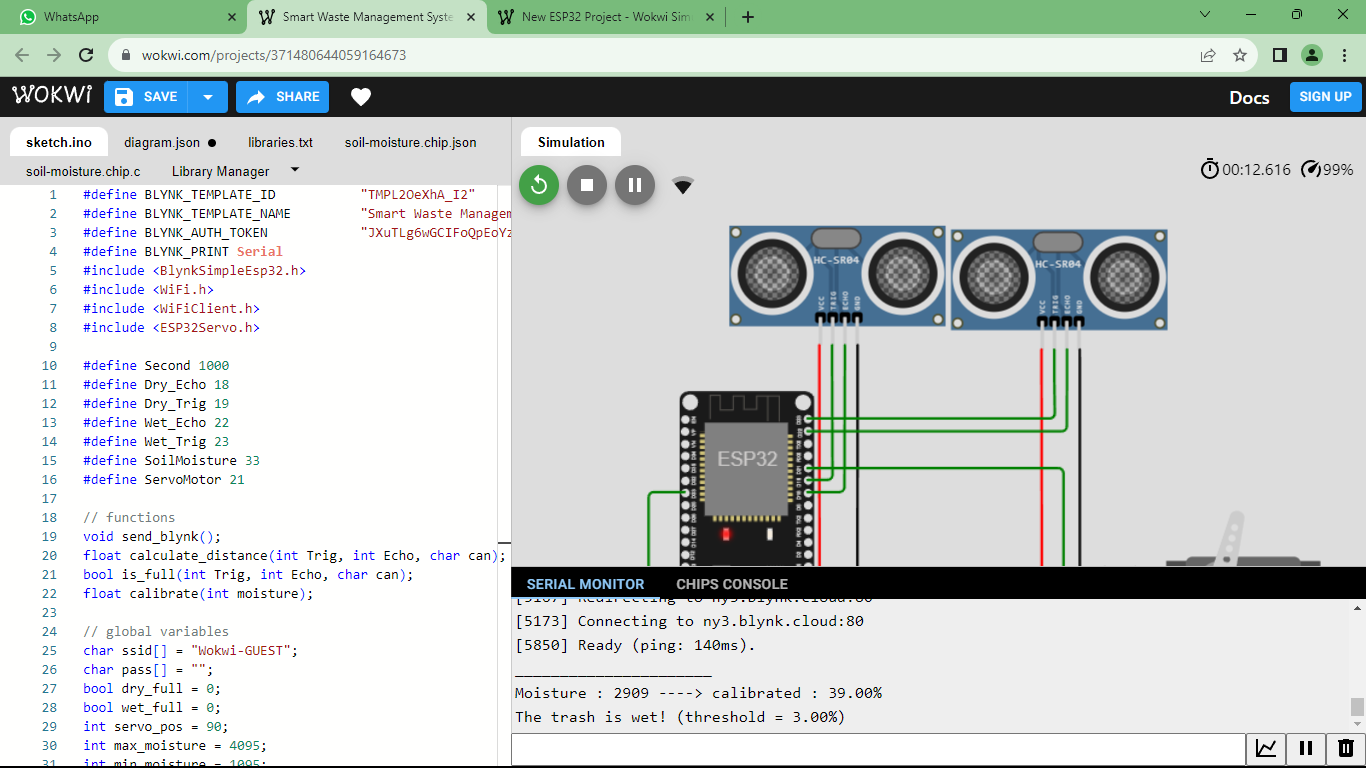
float calibrate(int moisture){

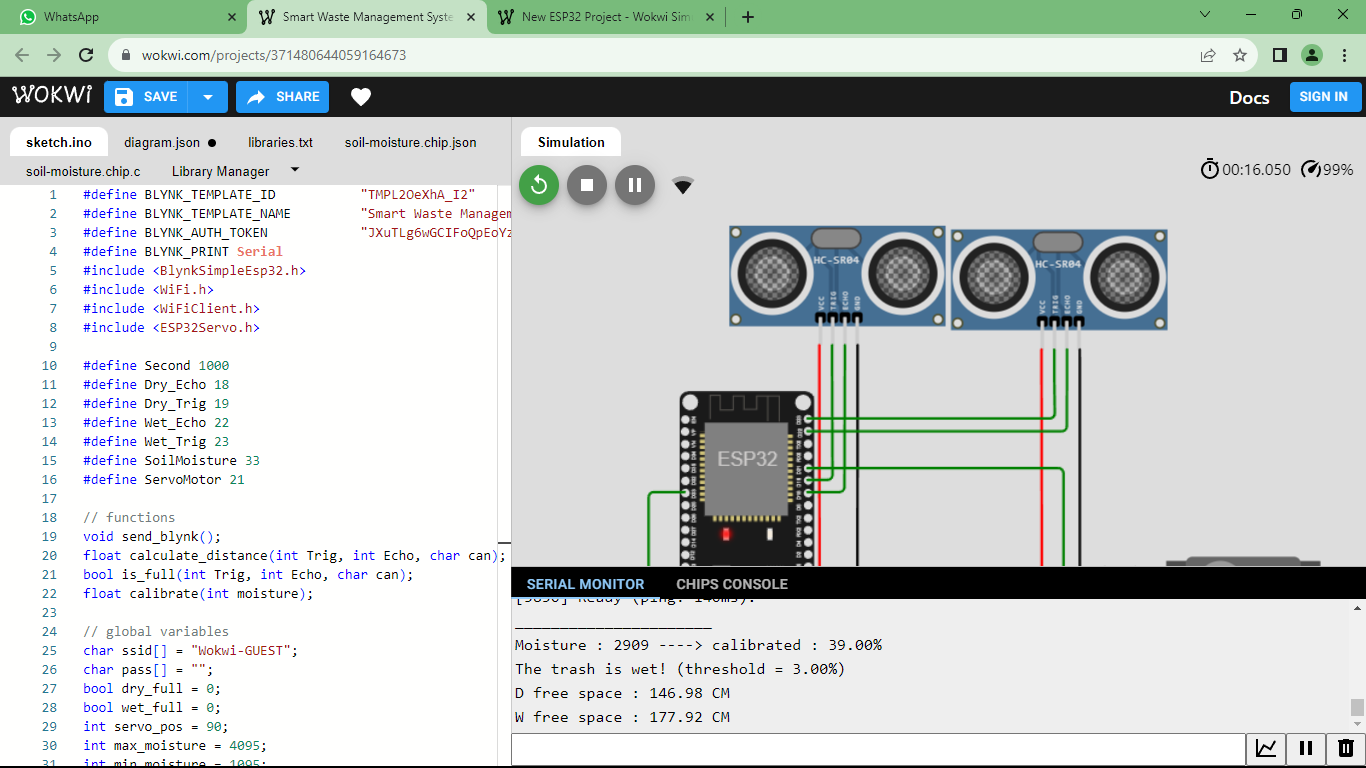
int x = (max\_moisture - min\_moisture)/100;

return (max\_moisture - moisture)/x;

}

STEP 8: Simulation





**WEB APPLICATION FOR SMART WASTE MANAGEMENT**

**CODE:**

<!DOCTYPE html>

<html>

<head>

<title>Smart Dustbin</title>

<style>

/\* Basic CSS for layout \*/

body {

font-family: Arial, sans-serif;

text-align: center;

}

.dustbin {

width: 300px;

margin: 250px auto;

}

.lid {

width: 100%;

height: 50px;

background-color: grey;

border-radius: 10px 10px 0 0;

}

.bin {

width: 100%;

height: 150px;

background-color: red;

border-radius: 0 0 10px 10px;

}

</style>

</head>

<body>

<div class="dustbin">

<div class="lid"></div>

<div class="bin"></div>

</div>

<script>

// JavaScript for simulating filling the dustbin

let binContent = 0; // Initially empty

function throwTrash() {

binContent += 10; // Increase the content by 10 units (you can adjust this as needed)

if (binContent >= 100) {

alert("Bin is full! Please empty it.");

binContent = 100; // Limit the content to a maximum of 100

}

document.querySelector('.bin').style.height = binContent + "px";

}

</script>

<button onclick="throwTrash()">Throw Trash</button>

<head>

<title>Smart Dustbin Moisture Content</title>

</head>

<body>

<h1>Moisture Content of Smart Dustbin</h1>

<p>Moisture Content: <span id="moistureValue">Fetching...</span></p>

<script>

function getMoistureData() {

// Simulated data for demonstration

const moisture = Math.floor(Math.random() \* 100); // Replace with actual moisture sensor data

return moisture;

}

function updateMoistureValue() {

const moisture = getMoistureData();

document.getElementById('moistureValue').textContent = moisture + "%";

}

updateMoistureValue(); // Initial call

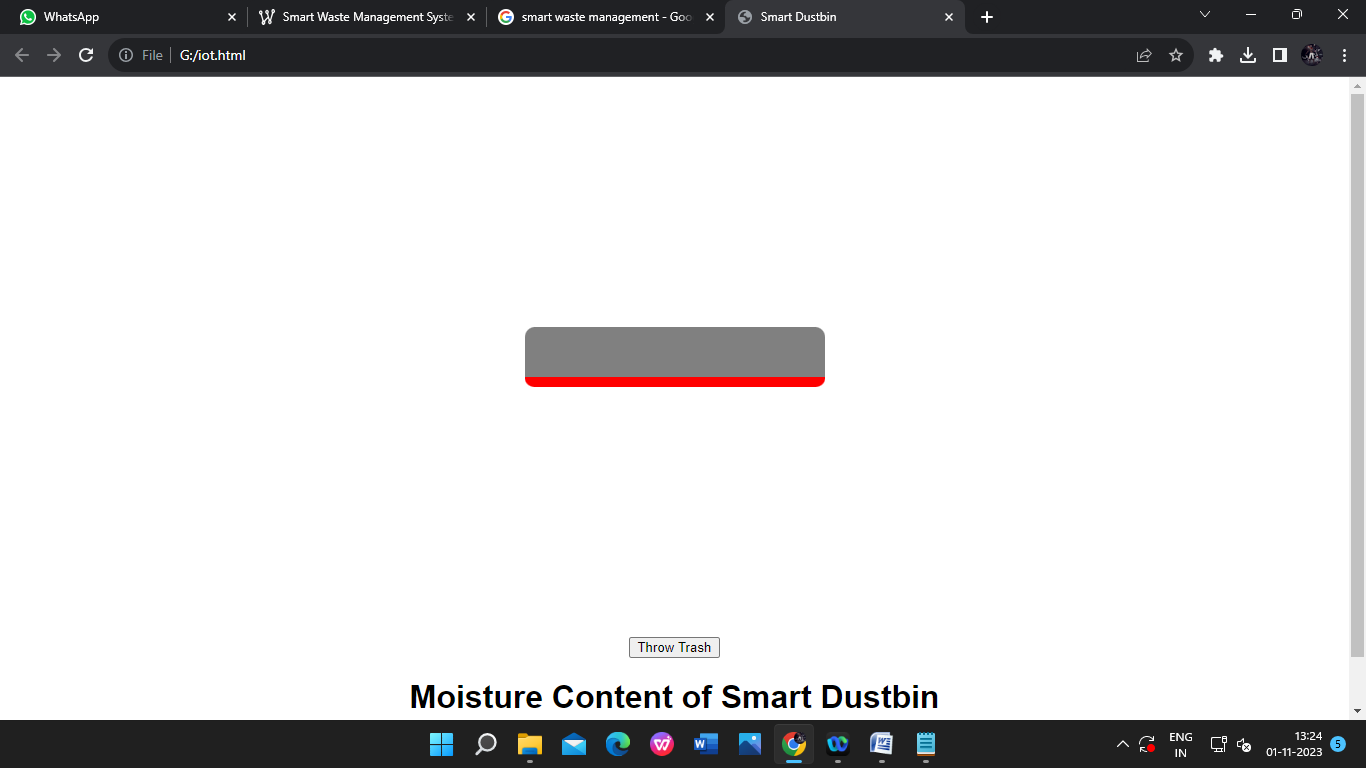
</script>

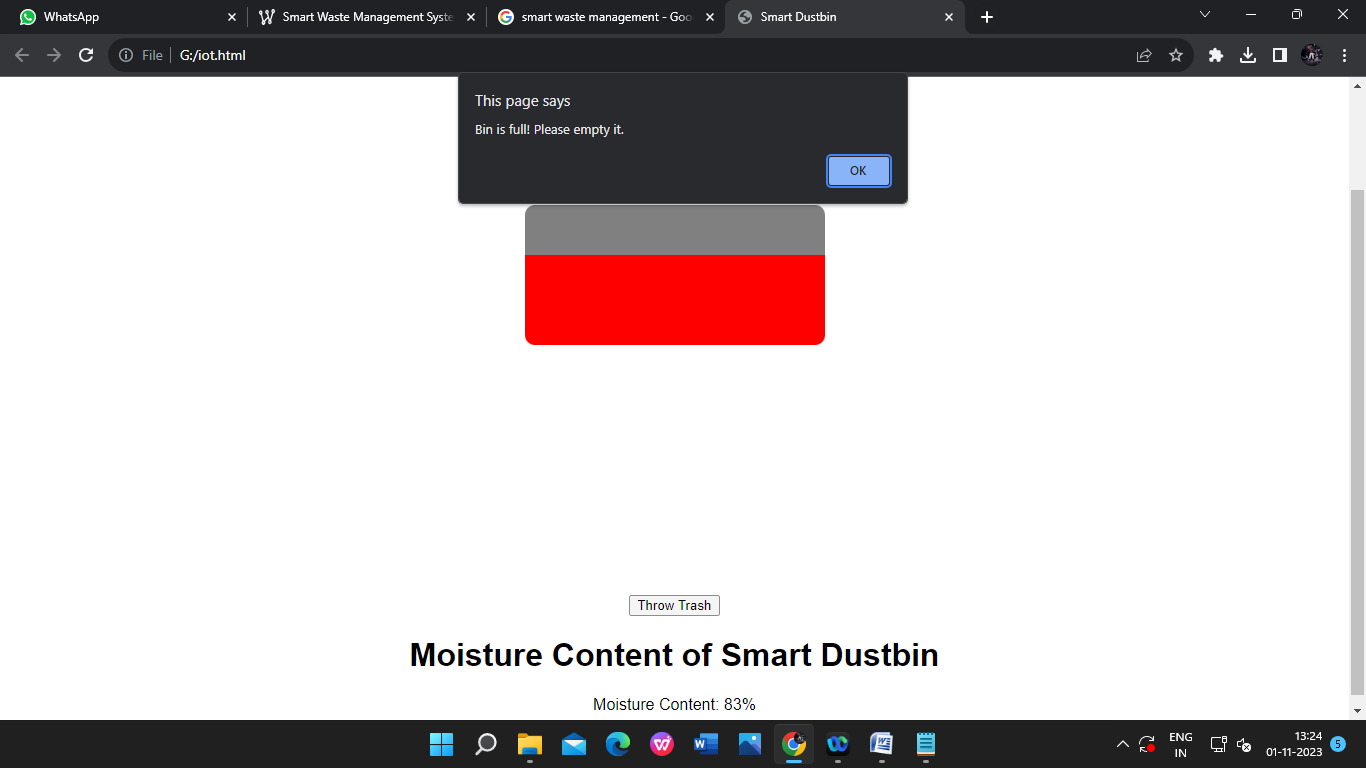
</body>

</body>

</html>

**OUTPUT:**

****

****