ENVIRONMENTAL MONITORING

PRESENTED BY,

1) Sudarshan R
2) Rayyan J
3) Vignesh J
4) Yogavaitheeshvaran S

DESCRIPTION

The Arduino-based IoT project integrates sensors to monitor temperature, humidity, soil moisture, and rainfall. Data is collected and transmitted wirelessly to a cloud platform for real-time monitoring. Users can access the information remotely via a web interface, enabling effective environmental monitoring and management.

APPARATUS USED

Sensors:

1) Ultrasonic sensor
2) Soil moisture sensor
3) Temperature sensor
4) Rainfall sensor
5) Humidity sensor

Components:

1) Buzzer
2) Jumper wires
3) LED
4) Lightning sensor
5) Arduino UNO

Platforms:

Wokwi simulator, visual studio code, python ide

PHASE 1

Phase 1: Problem Definition and Design Thinking

The scope of this document is to identify the problem and find solution for park visitors and park management to receive real time Environmental Data.

Problem Definition:

Research has identified several challenges in the management of public parks. These issues or concerns include:

- The park management or the park visitors do not receive any information or updates on the environmental data
- This is resulting in the park visitors not being able to plan their outdoor activities or the vist to the public park
- This also has resulted in the failure of the park management to organize promotional outdoor activities and campaigns for park visitors
- This has resulted in and overall low visitor satisfaction.

Design Thinking:

After comprehending the aforementioned issue, we will devise a solution capable of addressing it.

- An IoT based solution could immediately resolve the above issue
- We would need to identify the right Digital Humidity and Temperature Sesnsor, for which we would need perform extensive research on Original Equipment Manufacturers.
- The requirment also is that the IoT solution needs to be able to communicate to a public platform.
- Hence we would need internet connectivity and communication module

Add-On features:

- Using crowd monitoring system (CMS) the crowd in the park can be monitored and updated in the application.
- Image processing cameras can be added for security and child safety
- (OWS) optical wetness sensors can be used to sense the wetness of the floor in the parks which can lead to slip-off tradegy.
- Parking space management can be done to avoid zero parking problem by using various sensors such as infrared, EM sensors, radar sensor, ultrasonic sensor.

PHASE 2

Overview

Create a park solution using a network of environmental sensors to collect real-time data. This information is fed into a user-friendly website for visitors, which provides information such as air quality and weather. A web-based dashboard provides information to park administration, allowing for faster action and improved decision-making.

Components required:

1) **Ultrasonic Sensor:** Ultrasonic sensors use sound waves to detect the presence of people. They can be employed to measure crowd density in specific areas of the park.

2) **Soil Moisture Sensor:** In parks and green spaces, soil moisture sensors are used to monitor the moisture content of the soil. This data helps in plant care and wetness, dryness of soil.

3) **Temperature sensor:** These sensors measure the air temperature in the surrounding environment. They are essential for weather monitoring and climate control in indoor public spaces.

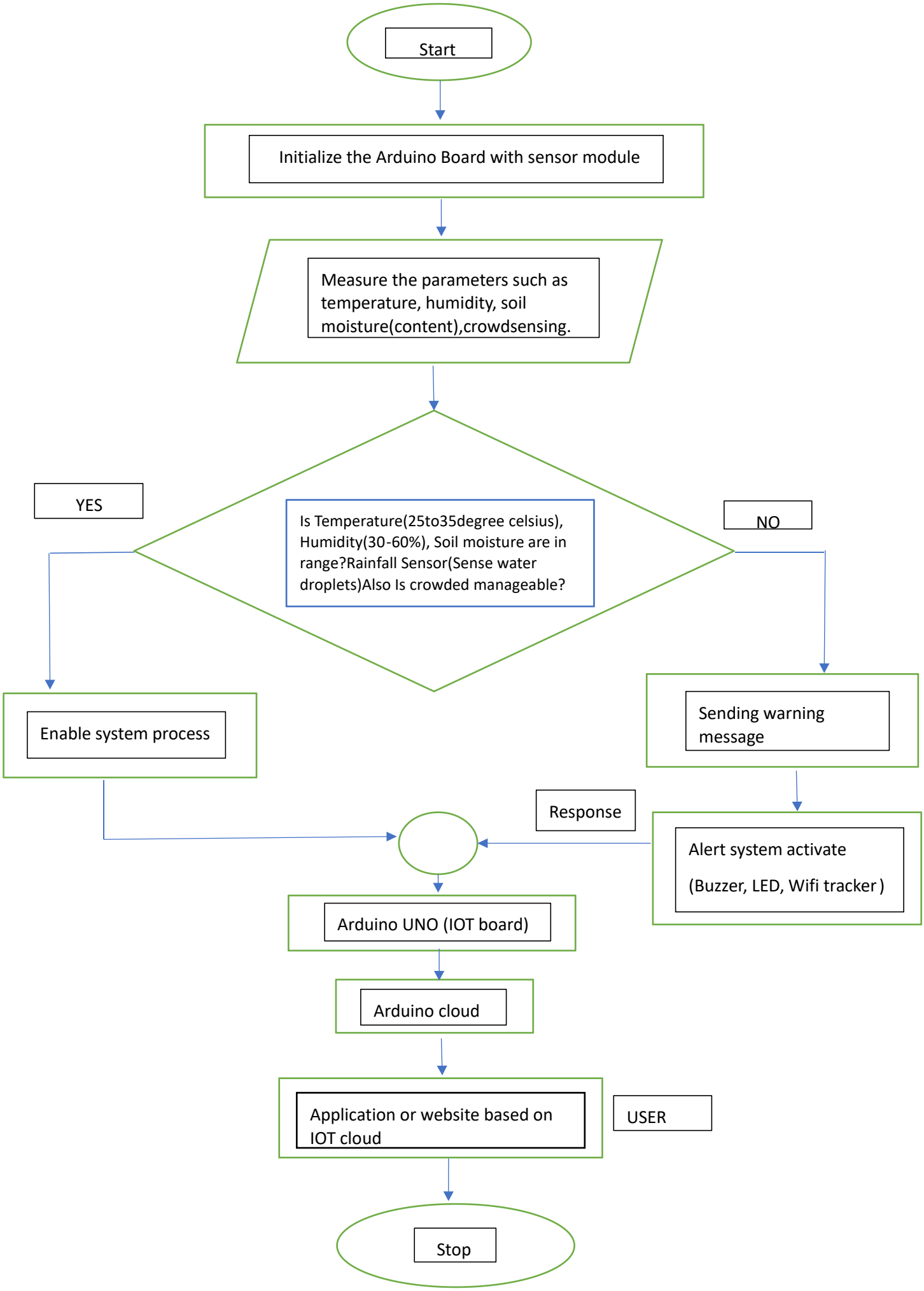
4) **Rainfall sensor:** Rainfall sensors measure the amount of precipitation. They are valuable in urban planning, drainage systems, and flood monitoring, helping authorities manage water flow in public areas during heavy rains.

5) **Humidity sensor:** Humidity sensors measure the moisture content in the air. High humidity levels can lead to discomfort, and these sensors help regulate indoor humidity in places like museums, libraries, and public transportation hubs.

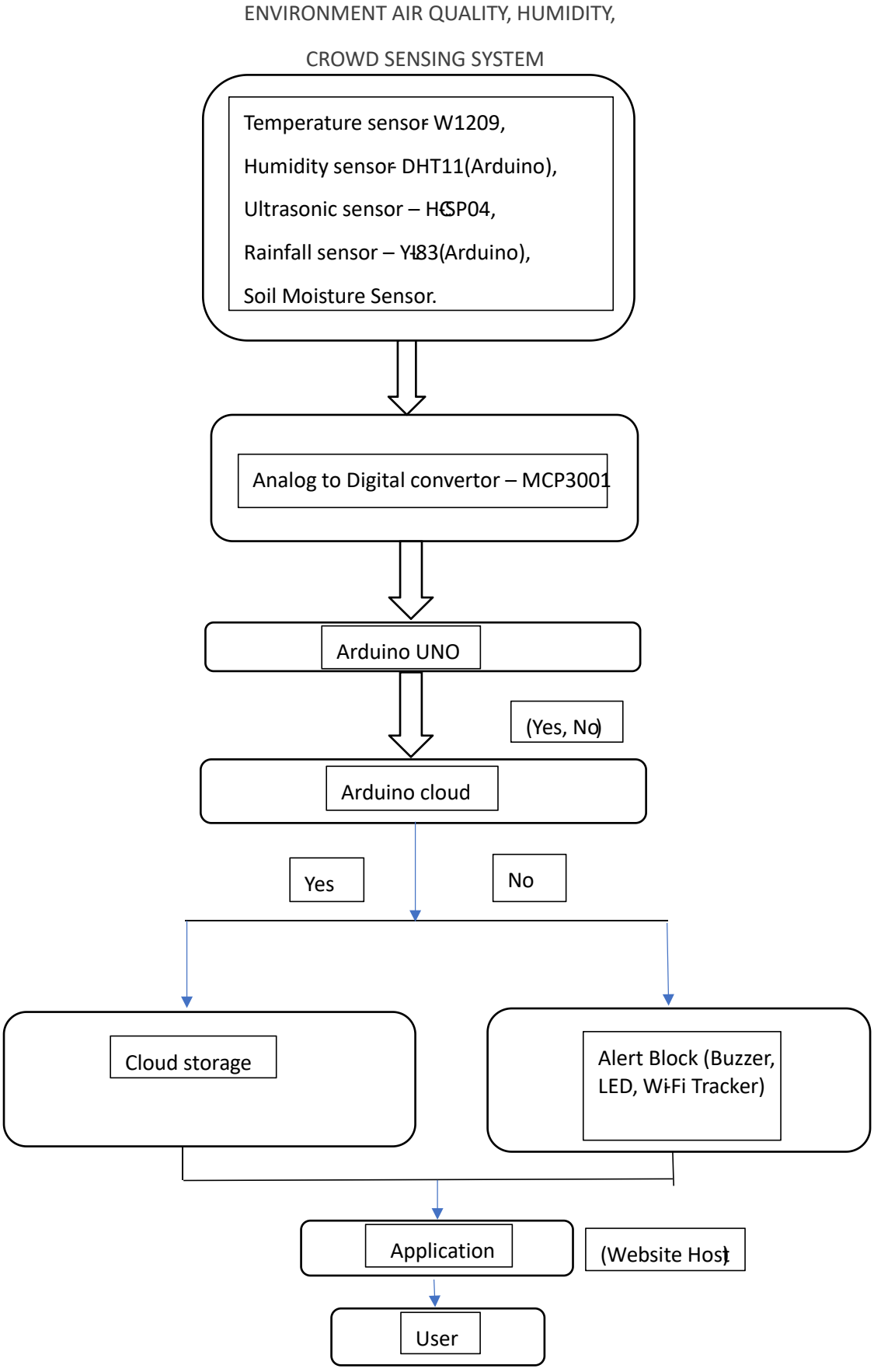
Some other components are Lightning sensor, Buzzer, LED, Jumper wires, Arduino UNO, Power Supply, bread board and application based Arduino IoT cloud.

Procedure:

Initialize the sensors and the Arduino Uno microcontroller. Read the temperature, pressure, humidity, and soil moisture data from the sensors. Analyse the data to determine if any environmental parameters are outside of acceptable ranges. If any parameters are outside of acceptable ranges, take appropriate action to correct them. For example, if the soil moisture is too low, activate an irrigation system to water the plants. If all parameters are within acceptable ranges, continue monitoring the environment. FLOWCHART IS MENTIONED BELOW


Start
Initialize the Arduino Board with sensor module
Measure the parameters such as temperature, humidity, soil moisture(content),crowdsensing.
Is Temperature(25to35degree celsius), Humidity(30-60%), Soil moisture are in range?Rainfall Sensor(Sense water droplets)Also Is crowded manageable?
YES
NO
Enable system process
Sending warning message
Response
Alert system activate
(Buzzer, LED, Wifi tracker)
Arduino UNO (IOT board)
Arduino cloud
Application or website based on IOT cloud
USER
Stop

Block diagram:



Benefits:

- Visitors can receive immediate alerts about adverse weather conditions or air quality, ensuring their safety during their park visit.

- Improved Visitor Experience: Access to real-time data allows visitors to plan their activities better, enhancing their overall experience.

- Environmental Conservation: Park management can respond quickly to environmental issues, minimizing damage and preserving the natural ecosystem.

- Monitoring environmental conditions helps optimize energy and resource usage, reducing operational costs.

PHASE 3

AS WE DISSCUED IN OUR EARLIER SUBMISSIONS, HERE WE GOING TO SIMULATE OUR PROJECT IN WOKWI SIMULATOR PLATFORM AS YOU MENTIONED.
THE FOLLOWING ARE THE DETAILS THAT DEPICTS ABOUT OUR PROJECT WORK. THANK YOU

SENSOR USED:HC-SR04,DHT22,NTC(temperature).

IMPLEMENTATION OF THE PROJECT

COMPONENTS USED:

- NodeMCU ESP32: This will be our microcontroller.
- DHT22: Sense the temperature and humidity.
- Wokwi Virtual Components: These are virtual components you can add in Wokwi for the web interface and simulation like button, resistor,LED bulb.

LIBRARIES USED:

- DHT sensor library for ESP32
- Pub Sub Client
- New Ping

ARDUINO CODE:

(applied to 'esp32' to sense the temperature and humidity of the environment)

```
#include <WiFi.h>

#include "DHTesp.h"

#include "ThingSpeak.h"
```

```
const int DHT_PIN = 15; const int
LED_PIN = 13;

const char* WIFI_NAME = "Wokwi-GUEST"; const
char* WIFI_PASSWORD = ""; const int
myChannelNumber =2307358 ; const char*
myApiKey = "1U2N21SZEGP74GFZ"; const char*
server = "api.thingspeak.com";


DHTesp dhtSensor;

WiFiClient client;


void setup()  {


  Serial.begin(115200);

  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);   pinMode(LED_PIN,
OUTPUT);

  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);   while
(WiFi.status() != WL_CONNECTED){     delay(1000);

    Serial.println("Wifi not connected");

  }


  Serial.println("Wifi connected !");

  Serial.println("Local IP: " + String(WiFi.localIP()));

  WiFi.mode(WIFI_STA);

  ThingSpeak.begin(client);

}

void loop() {

  TempAndHumidity  data = dhtSensor.getTempAndHumidity();

  ThingSpeak.setField(1,data.temperature);   ThingSpeak.setField(2,data.humidity);   if
(data.temperature > 35 || data.temperature < 12 || data.humidity > 70 || data.humidity < 40)
{     digitalWrite(LED_PIN, HIGH);

  }else{

    digitalWrite(LED_PIN, LOW);

  }
```

```
   int x = ThingSpeak.writeFields(myChannelNumber,myApiKey);
Serial.println("Temp: " + String(data.temperature, 2) + "°C");
Serial.println("Humidity: " + String(data.humidity, 1) + "%");     if(x
== 200){

   Serial.println("Data pushed successfull");

  }else{

   Serial.println("Push error" + String(x));

  }

  Serial.println("---");

  delay(10000);

}
```

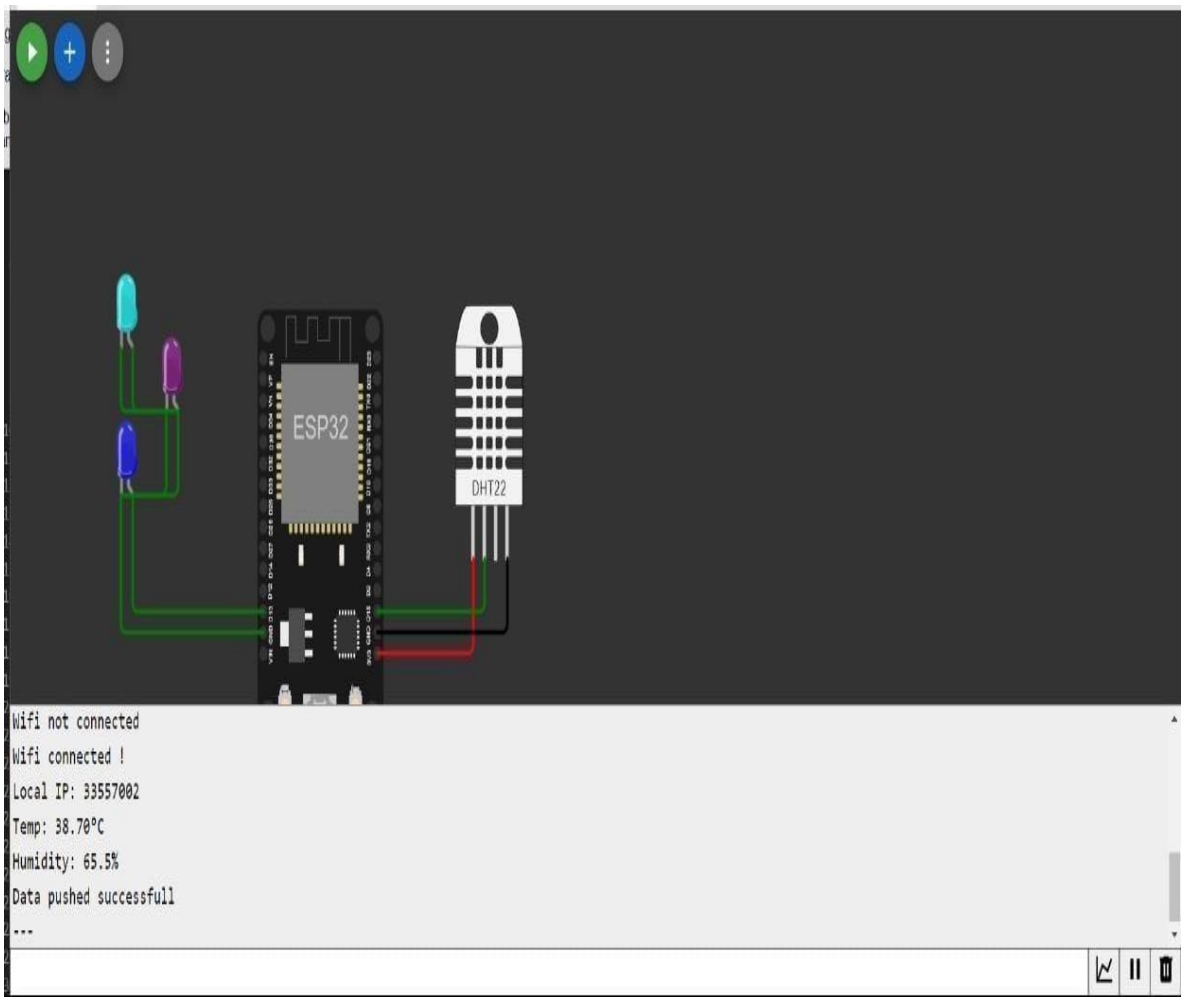
PYTHON CODE :

(applied to ‘esp32’ to sense the temperature and humidity of the environment)

```
import machine
import time from dht
import DHT22 import
network import
urequests

DHT_PIN = 2  # GPIO pin 2
LED_PIN = 13  # GPIO pin 13

WIFI_NAME = "Wokwi-GUEST"
WIFI_PASSWORD = ""
CHANNEL_NUMBER = 2307358
API_KEY = "1U2N21SZEGP74GFZ"
 dhtSensor = DHT22(machine.Pin(DHT_PIN))
led = machine.Pin(LED_PIN, machine.Pin.OUT)
 wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_NAME,
WIFI_PASSWORD) while not
wifi.isconnected():
    pass
 print("Wifi connected!")
print("Local IP:",
wifi.ifconfig()[0])
 def
read_dht():
    dhtSensor.measure()
    temperature = dhtSensor.temperature()
humidity = dhtSensor.humidity()     return
temperature, humidity
 while
True:
    temperature, humidity = read_dht()
print("Temp: {:.2f}°C".format(temperature))
print("Humidity: {:.1f}%".format(humidity))
     if temperature > 35 or temperature < 12 or humidity > 70
or humidity < 40:         led.on()     else:
        led.off()
```

```
    response =
urequests.get("https://api.thingspeak.com/update?api_key={}&field1={:.2
f}&field2={:.1f}".format(API_KEY, temperature, humidity))
print("Response:", response.status_code)     response.close()
     time.sleep(10)    #  Wait  for  10  seconds  before  the  next
reading
```

OUTPUT:



CODE FOR CONNECTING THE ULTRASONIC SENSOR (HC-SP04) TO THE ARDUINO UNO BOARD

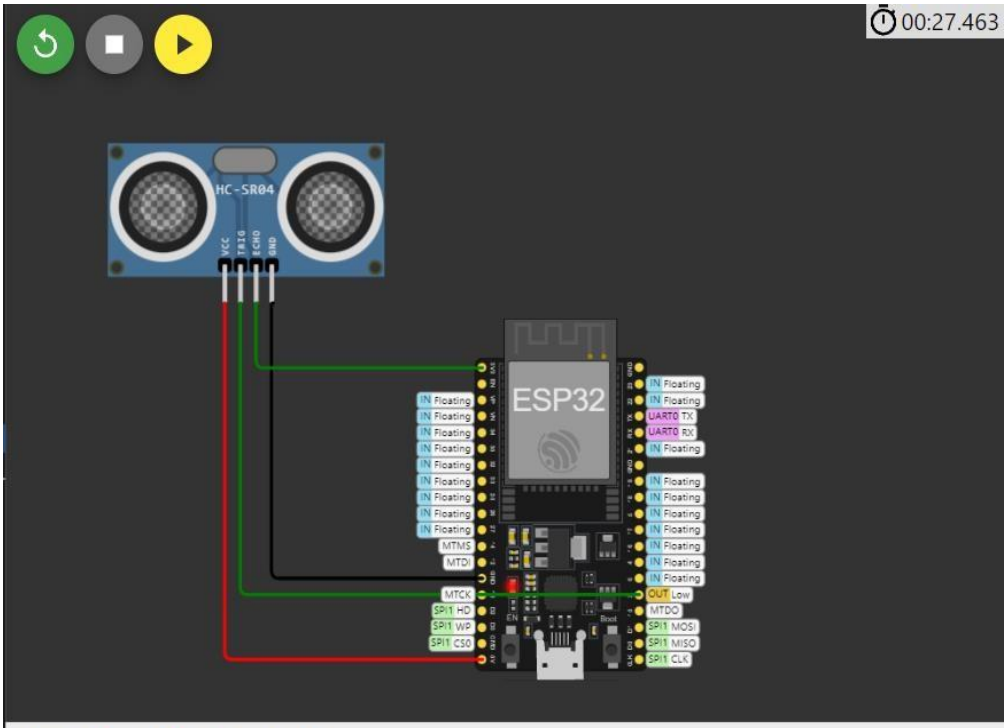
- Used to detect the number of people entered the environment we've provided certainly.

```
#include <NewPing.h>

#define TRIGGER_PIN 9

#define ECHO_PIN 10

#define MAX_DISTANCE 200 // Maximum distance we want to detect in centimeters (cm)

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and maximum distance.

void setup() {

 Serial.begin(9600); // Open a serial connection for debugging purposes.

}

void loop() {   delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.    unsigned int distance = sonar.ping_cm(); // Send ping, get distance in centimeters.

 if (distance <= 200 && distance >= 2) { // Check if the distance is within the valid range.

   Serial.print("Number of people detected: ");

   Serial.println(distance); // Print the distance to the serial monitor.

 } else {

erial.println("No people detected!"); // Print a message indicating no people are detected.

 }

 delay(1000); // Wait for a second before taking the next reading.

}
```

```
#include <NewPing.h>

#define TRIGGER_PIN 9
#define ECHO_PIN 10
```

```
#define MAX_DISTANCE 200 // Maximum distance we want to detect in centimeters
(cm)

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins
and maximum distance.

void setup() {
  Serial.begin(9600); // Open a serial connection for debugging purposes.
}

void loop() {   delay(50); // Wait 50ms between pings (about 20
pings/sec). 29ms should be the shortest delay between pings.
  unsigned int distance = sonar.ping_cm(); // Send ping, get distance
in centimeters.
   if (distance <= 200 && distance >= 2) { // Check if the distance
is within the valid range.
    Serial.print("Number of people detected: ");
    Serial.println(distance);  //  Print  the  distance  to  the  serial
monitor.    } else { erial.println("No people detected!"); // Print a
message indicating no people are detected.
  }   delay(1000); // Wait for a second before taking the next
reading.
}
```

OUTPUT:




```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
```

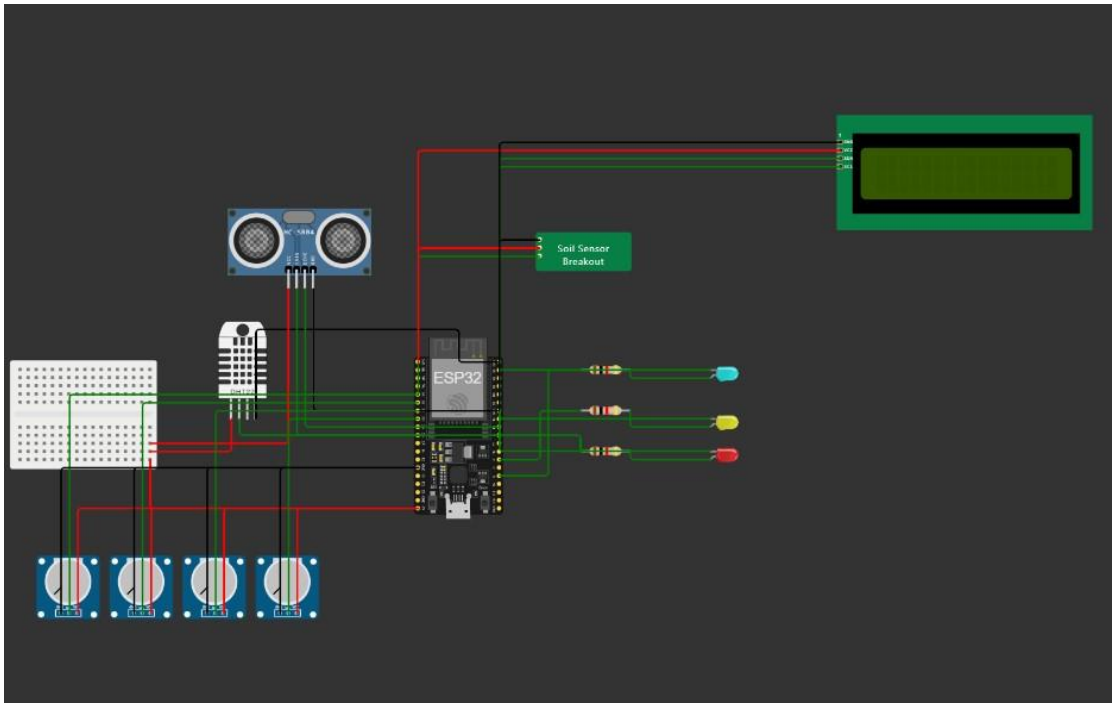
THE ENVIRONMENTAL PROJECT ;
COMBINED ARDUINO CODE FOR THE ALL SENSORS USED :

```
#include <DHT.h>
#include <WiFi.h>
#include <LiquidCrystal_I2C.h> DHT dht(26,DHT11); const int
trigPin=5; const int echopin=18; #define SOUND_SPEED 0.034 #define
CM_TO_INCH 0.393701 long duration; float distanceCm; float
distanceInch; const int numLeds = 3; const int ledPins[numLeds] = {2,
4, 5};  // Pins to which the LEDs are connected const int numSensors
= 4; const int sensors[numSensors] = {32, 33, 34, 35};  // ADC input
pins int values[numSensors];  // Array to store potentiometer values
int choice = 0;  // Variable to store the choice int total = 0;  //
Variable to store the total of the values int percent = 0;
LiquidCrystal_I2C lcd(0x27,16,2);


void setup() {
  Serial.begin(115200);
dht.begin();   delay(2000);
pinMode(5,OUTPUT);
pinMode(18,INPUT);
   Serial.begin(9600);

  // Set the LED pins as outputs
for (int i = 0; i < numLeds; i++)
{     pinMode(ledPins[i], OUTPUT);
  }
  Wire.begin(23, 22);
Serial.begin(9600);
lcd.init();   lcd.backlight();
}  void loop() {   delay(2000);  // Delay
between sensor readings
     float humidity =
dht.readHumidity();
```

```
  float temperature = dht.readTemperature();
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.print("%  Temperature: ");
    Serial.print(temperature);
    Serial.println("°C");
// Clears the trigPin
digitalWrite(5, LOW);
delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro
seconds   digitalWrite(5, HIGH);
delayMicroseconds(10);   digitalWrite(5, LOW);

  // Reads the echoPin, returns the sound wave travel time
in microseconds   duration = pulseIn(18, HIGH);

  // Calculate the distance
distanceCm = duration * SOUND_SPEED/2;

  // Convert to inches   distanceInch =
distanceCm * CM_TO_INCH;

  // Prints the distance in the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

delay(1000);
   for (int j = 0; j < numSensors; j++)
{     values[j] = analogRead(sensors[j]);
  }
  total = values[0] + values[1] + values[2] + values[3];
 percent =
map(total,0,16383,0,100);
  // Print the total to the serial console
  Serial.printf("The value is: %d %% \n", percent);
  // Determine the choice based on the potentiometer
values   if (percent > 70) {     Serial.println("Heavy");
choice = 1;
  } else if (percent <30)
{     Serial.println("Drizzle");
choice = 2;
```

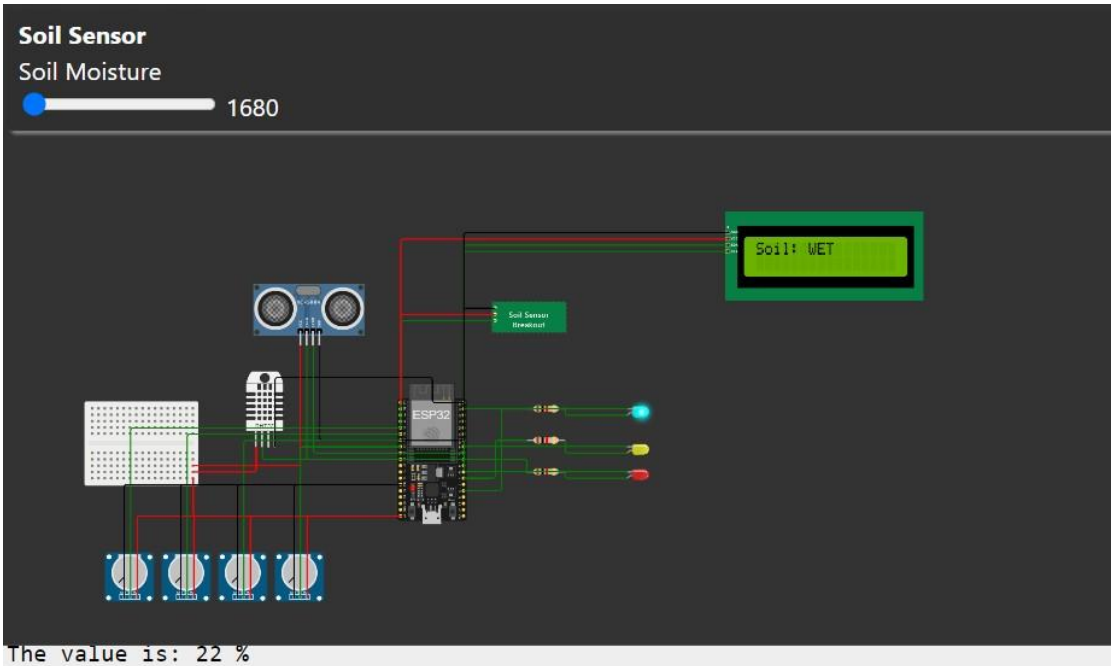
```
  } else {
    Serial.println("Medium");
choice = 3;
  }

  // Turn on the appropriate LEDs based on the
choice   switch (choice) {     case 1:
      // Turn on all LEDs       for (int k = 0; k < 3;
k++) {  // Loop 10 times       digitalWrite(ledPins[2],
HIGH);  // Turn the LED on       delay(500);  // Delay
for 1 second       digitalWrite(ledPins[2], LOW);  //
Turn the LED off       delay(500);  // Delay for 1
second
      }
break;
case 2:
      // Turn on only the first and second LEDs
for (int k = 0; k < 3; k++) {  // Loop 10 times
digitalWrite(ledPins[0], HIGH);  // Turn the LED on
delay(500);  // Delay for 1 second
digitalWrite(ledPins[0], LOW);  // Turn the LED off
delay(500);  // Delay for 1 second
      }
break;
case 3:
      // Turn on only the third LED       for (int k =
0; k < 3; k++) {  // Loop 10 times
digitalWrite(ledPins[1], HIGH);  // Turn the LED on
delay(500);  // Delay for 1 second
digitalWrite(ledPins[1], LOW);  // Turn the LED off
delay(500);  // Delay for 1 second
      }
break;
  }   delay(1000);  // Delay for 1
second   int16_t i = analogRead(34);
  String msg = i < 2165 ? "WET" : i > 3135 ? "DRY" :
"OK";   lcd.clear();   lcd.print("Soil: ");
lcd.print(msg);   delay(500);
  }
```

CIRCUIT BOARD:

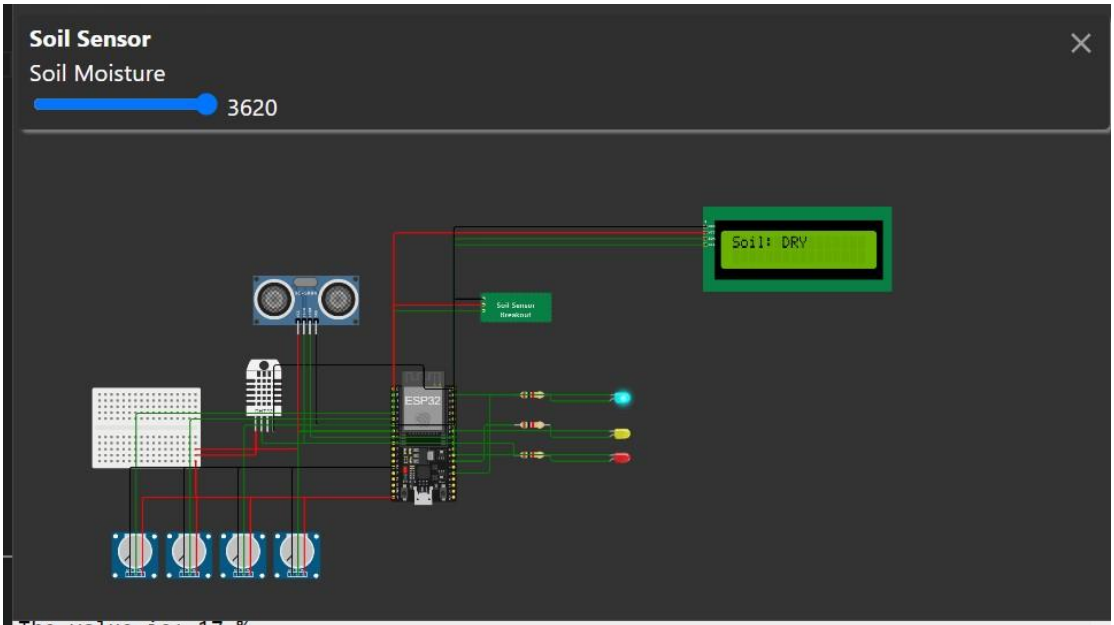


OUTPUT:

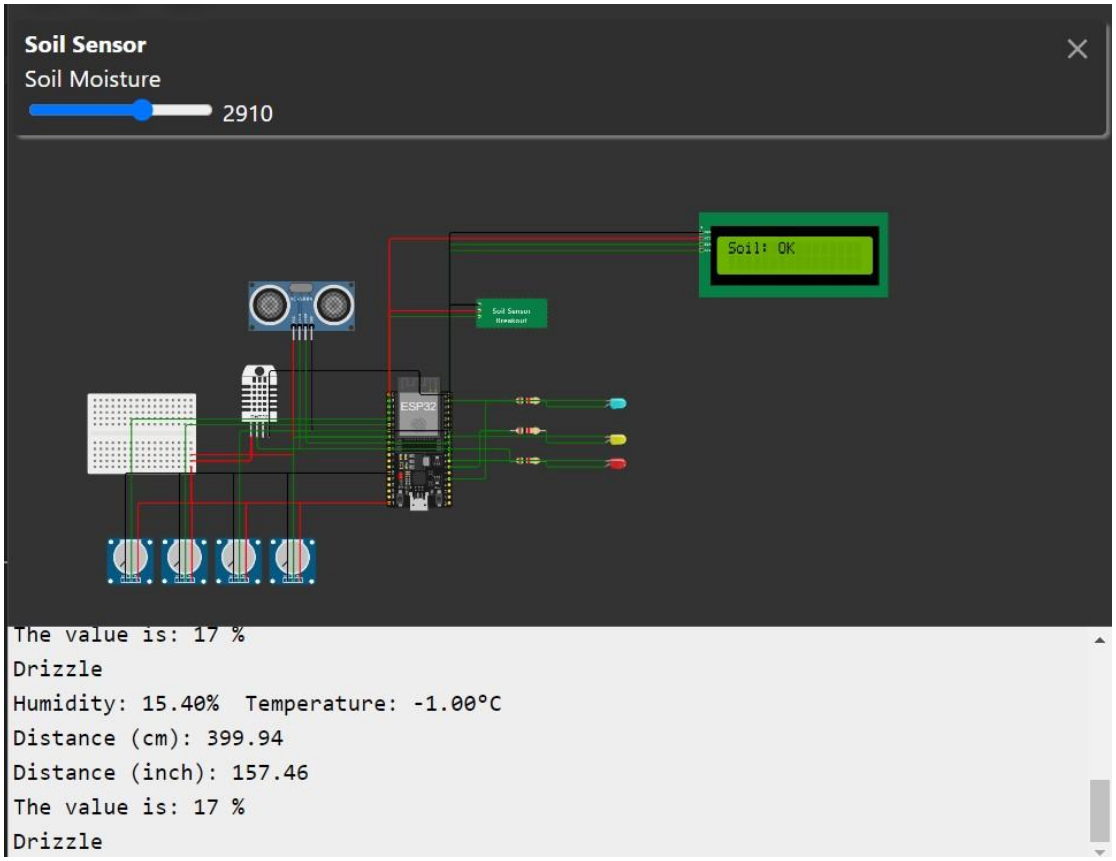
```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Humidity: 15.40%  Temperature: -1.00°C
Distance (cm): 399.94
Distance (inch): 157.46
The value is: 17 %
Drizzle
Humidity: 15.40%  Temperature: -1.00°C
Distance (cm): 399.94
Distance (inch): 157.46
The value is: 17 %
Drizzle
Humidity: 15.40%  Temperature: -1.00°C
Distance (cm): 399.94
Distance (inch): 157.46
The value is: 17 %
```




```
The value is: 22 %
Drizzle
Humidity: 15.40%  Temperature: -1.00°C
Distance (cm): 399.94
Distance (inch): 157.46
The value is: 10 %
Drizzle
```




```
The value is: 17 %
Drizzle
Humidity: 15.40%  Temperature: -1.00°C
Distance (cm): 399.94
Distance (inch): 157.46
The value is: 22 %
Drizzle
```




```
The value is: 17 %
Drizzle
Humidity: 15.40%  Temperature: -1.00°C
Distance (cm): 399.94
Distance (inch): 157.46
The value is: 17 %
Drizzle
```

PHASE 4

AS WE DISSCUED IN OUR EARLIER SUBMISSIONS, HERE WE GOING TO SIMULATE OUR PROJECT IN WOKWI SIMULATOR PLATFORM AS YOU MENTIONED.

ALSO WE USED VISUAL STUDIO CODE(VSC) TO WORK WITH SOME WEB DEVELOPMENT PROCESS WHICH INCLUDES HTML5,CSS3,JAVASCRIPT.

THE FOLLOWING ARE THE DETAILS THAT DEPICTS ABOUT OUR PROJECT WORK. THANK YOU

SENSORS USED:

HC-SR04,

DHT22,

NTC(temperature).

HUMUDITY,SOIL MOISTURE AND RAINFALL .

PROGRAM CODING:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Arduino Sensor Data</title>
    <style>         body
{             font-family: Arial, sans-
serif;              text-align: center;
        }

        .sensor-data
{             margin: 20px;
        }

        .sensor-values p
{             margin: 5px 0;
        }

        .led {             width:
50px;             height: 50px;
border-radius: 50%;
background-color: gray;
display: inline-block;
margin: 0 10px;
        }

        .lcd-display
{             margin-top: 20px;
        }
    </style>
</head>

<body>
    <div class="sensor-data">
        <h1>Arduino Sensor Data</h1>
        <div class="sensor-values">
            <p><strong>Humidity:</strong> <span id="humidity">--</span>%</p>
            <p><strong>Temperature:</strong>     <span     id="temperature">-
</span>°C</p>
```

```
            <p><strong>Distance    (cm):</strong>    <span    id="distanceCm">-
</span></p>
            <p><strong>Choice:</strong>    <span    id="choice">--</span></p>
</div>
        <div class="led-indicators">
            <div class="led" id="led1"></div>
            <div class="led" id="led2"></div>
            <div class="led" id="led3"></div>
        </div>
        <div class="lcd-display">
            <strong>Soil Moisture:</strong> <span id="soilMoisture">--</span>
        </div>
    </div>     <script>
document.addEventListener("DOMContentLoaded", function ()
{             function updateSensorData() {
                // Simulate sensor data (replace with actual data)
let humidity = Math.floor(Math.random() * 71) + 30; // Random between 30% and
100%                 let temperature = (Math.random() * 11) + 20; // Random
between
20°C and 30°C                 let distanceCm = Math.floor(Math.random() * 96)
+ 5; // Random between 5 cm and 100 cm                       let choice =
Math.floor(Math.random()  *  3)  +  1;  //  Random  choice  between  1  and  3
let soilMoisture = Math.floor(Math.random() * 101); // Random between 0% and
100%

                // Update HTML elements with sensor data
document.getElementById("humidity").textContent = humidity;
document.getElementById("temperature").textContent = temperature.toFixed(2);
                document.getElementById("distanceCm").textContent            =
distanceCm;
                document.getElementById("choice").textContent         =
choice;
document.getElementById("soilMoisture").textContent = soilMoisture;

                // Update LED indicators based on the choice
let led1 = document.getElementById("led1");                 let led2 =
document.getElementById("led2");                let led3 =
document.getElementById("led3");
led1.style.backgroundColor = (choice === 1) ? "green" :
"gray";                led2.style.backgroundColor = (choice === 2) ?
"green" :
"gray";                led3.style.backgroundColor = (choice === 3) ?
"green" : "gray";
```

```
            }

            //      Update      sensor      data      every      2      seconds
setInterval(updateSensorData, 2000);
        });
    </script>
</body>
</html>
```

EQUIVALENT PYTHON CODE:

from flask import Flask,

render_template import random import

time

app = Flask(__name__)

@app.route('/') def index():   # Simulate

sensor data   humidity =

random.uniform(30, 70)   temperature

= random.uniform(20, 30)

distance_cm = random.uniform(5, 100)

soil_moisture = random.randint(0, 100)

    # Determine choice based on soil

moisture   if soil_moisture > 70:

choice = "Heavy"

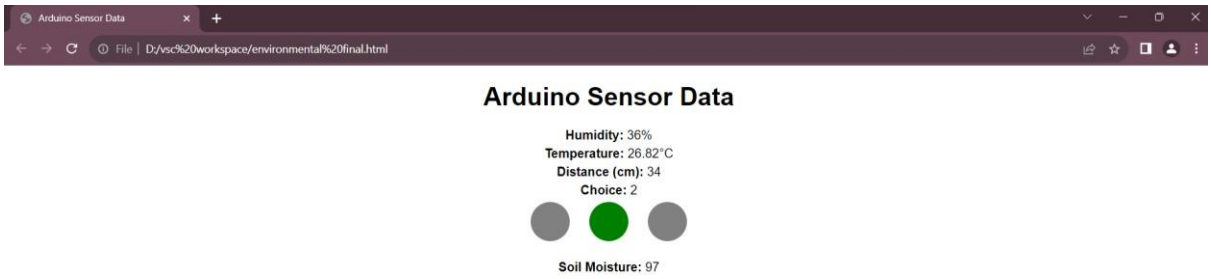
    elif soil_moisture < 30:

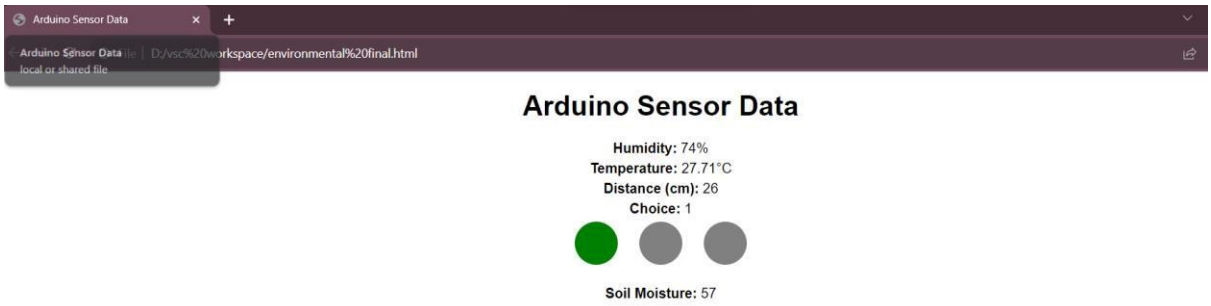
choice = "Drizzle"   else:

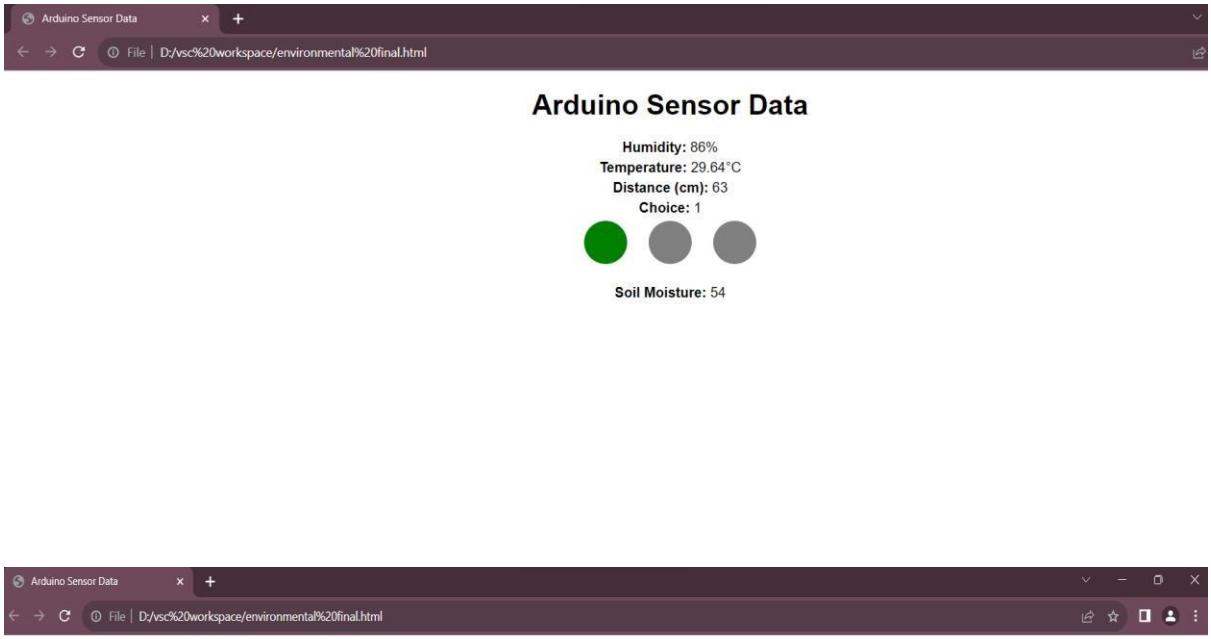
```
        choice = "Medium"

    return render_template('index.html', humidity=humidity,
temperature=temperature, distance_cm=distance_cm,
soil_moisture=soil_moisture, choice=choice)

if __name__ == '__main__':
app.run(debug=True)
```

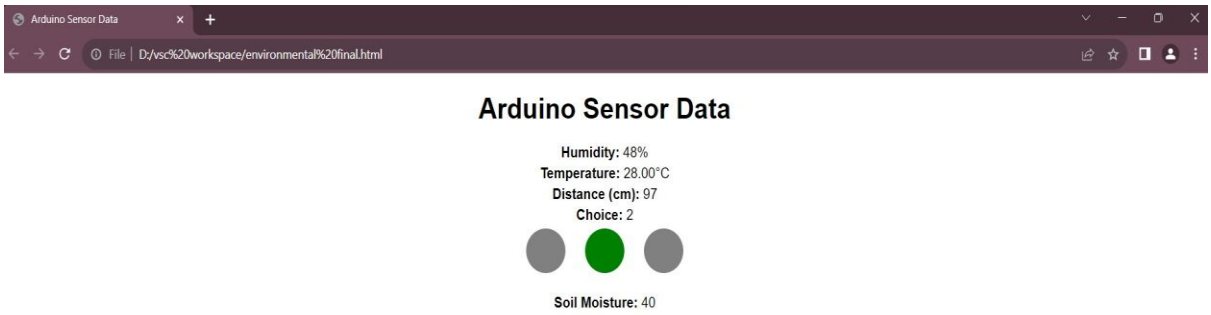
Provided to install libraries and pip.

OUTPUT:






Arduino Sensor Data
File | D:/vsc%20workspace/environmental%20final.html
Arduino Sensor Data
Humidity: 86%
Temperature: 29.64°C
Distance (cm): 63
Choice: 1
Soil Moisture: 54


Arduino Sensor Data
File | D:/vsc%20workspace/environmental%20final.html
Arduino Sensor Data
Humidity: 48%
Temperature: 28.00°C
Distance (cm): 97
Choice: 2
Soil Moisture: 40