NM1012 INDUSTRIAL IOT AND INDUSTRY 4.0

Water-Wise’

Bridging urban water scarcity for all

PROJECT FINAL REPORT

INSTITUTE:

ANAND INSTITUTE OF HIGHER TECHNOLOGY

TEAM:

AUDRI

TEAM MEMBERS:

AKSHAYA SRINITHI SV(au310121106003)

RAJESWARI C(au310121106032)

DIVYASHREE K (au310121106008)

TEAM LEAD:

AKSHAYA SRINITHI SV

akshayasri2908@gmail.com

9840898041

ingage

ACKNOWLEDGEMENT

We are grateful to **Dr. Sankara Malliga.G, M.E., Ph.D.,** Head of the Department, Electronics and Communication Department, for her guidance and great encouragement shown towards us to complete our project.

We would like to express our sincere thanks to our mentor **Mr. Logeshwaran. R, M.E.,** Department of Electronics and Communication Engineering, Anand Institute of Higher Technology, for the great support and valuable encouragement throughout our project.

We would like to express our sincere thanks to our Naan Mudhalvan(Industrial IOT and Industry 4.0)mentor **Mr.ThangaManivasagam.M**,InGage(Course Partner) for the invaluable and clear vision and knowledge in the realm of Industrial IOT.

We thank our parents and the Almighty God for endowing us with their immense blessings which helped us in each step of our progress towards the successful completion of this project.

ABSTRACT

Title: Water Wise: Repurposing AC Condensate for Environmental Conservation and Animal Welfare

Water Wise introduces a novel approach to sustainability by repurposing condensate from air conditioning (AC) systems to address water scarcity and support wildlife. By capturing and treating AC condensate, Water Wise aims to create a sustainable water source for feeding strays, birds, and other wildlife. Through community engagement and decentralized distribution, the project fosters responsible water stewardship while mitigating habitat loss and promoting compassion towards animals. Water Wise represents a transformative initiative that integrates environmental conservation with social responsibility, offering a practical solution to pressing ecological challenges.

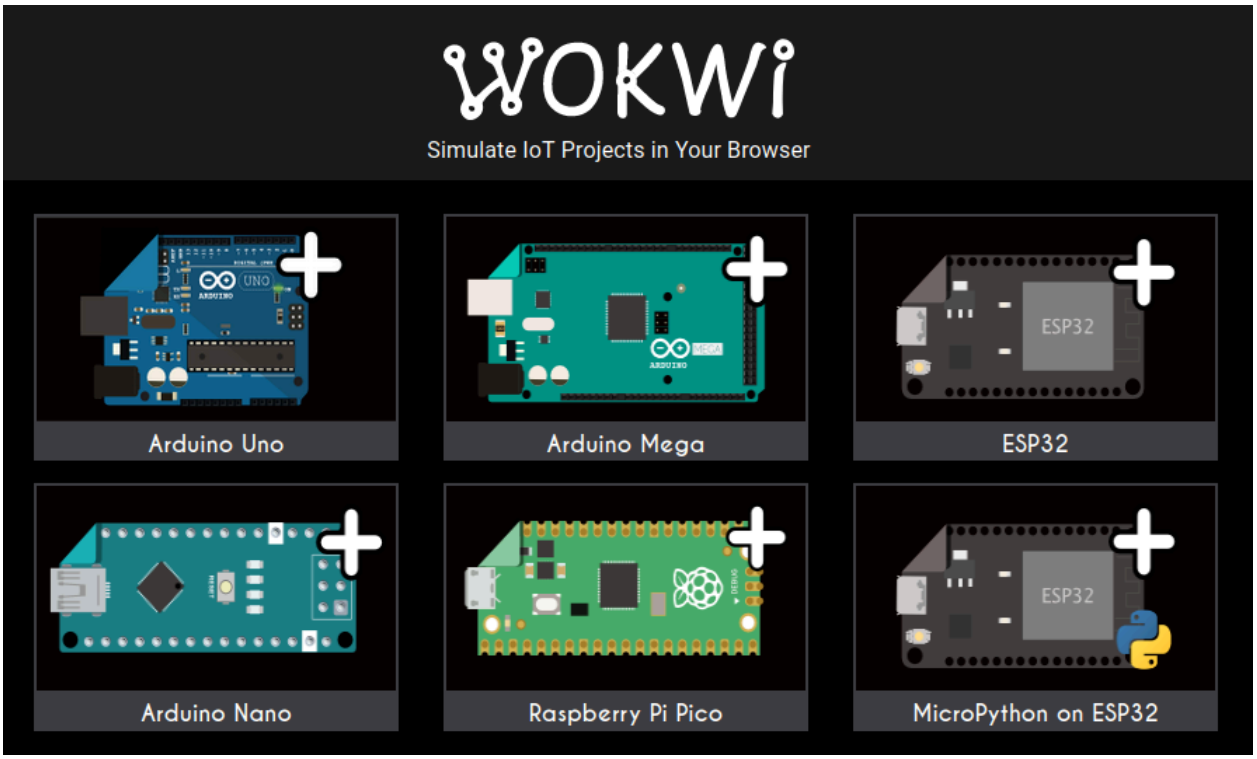
SIMULATOR AND ANALYZER

2.1 WOKWI SIMULATOR

Wokwi Simulator is an online platform that provides a virtual environment for simulating and prototyping Arduino and electronics projects. It allows users to design, test, and debug their projects without the need for physical hardware.

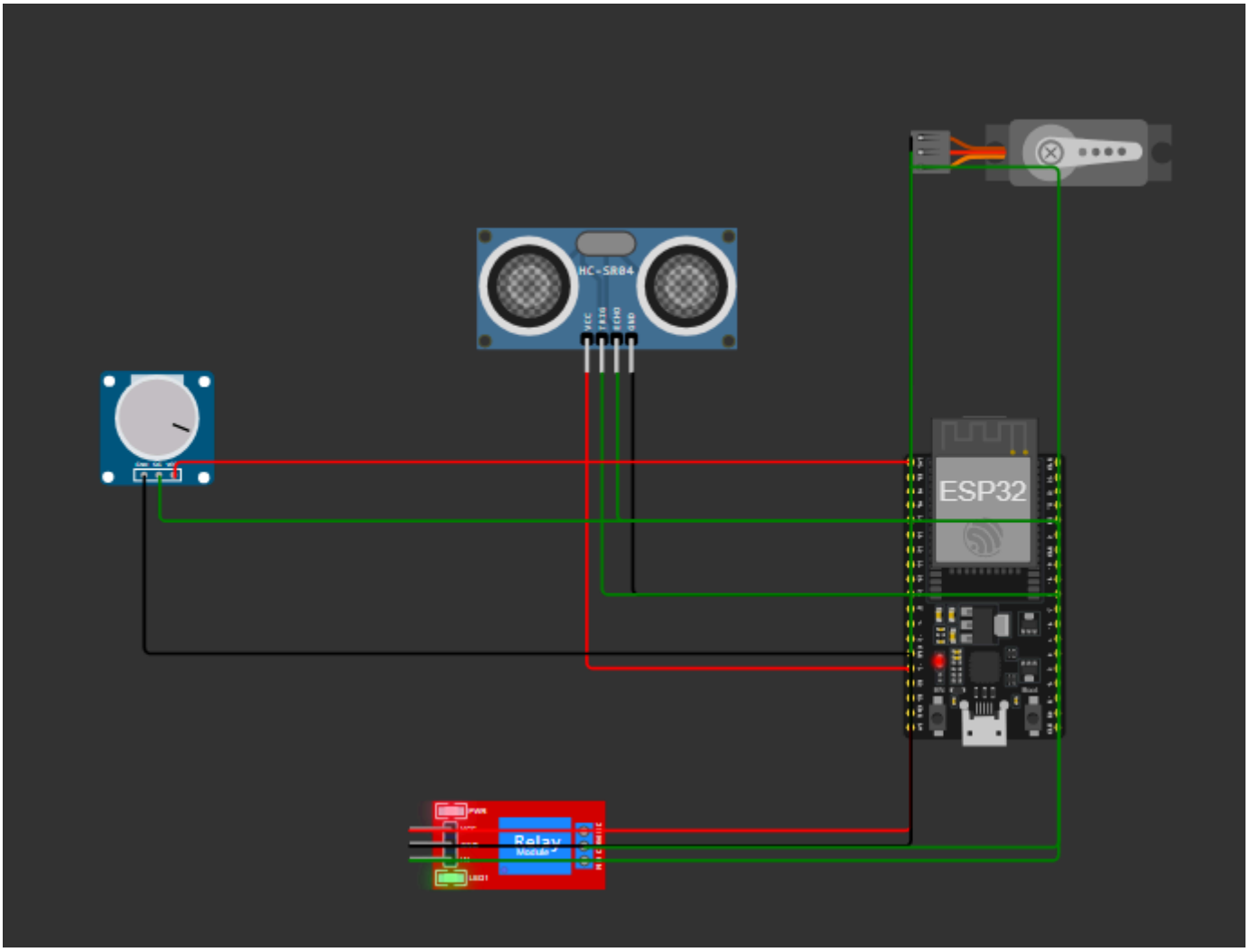
2.1.1 WOKWI WITH ESP32

Wokwi Simulator supports ESP32, a powerful microcontroller widely used in IoT projects, allowing users to design, simulate, and prototype ESP32-based projects directly in the browser.



2.2 SIMULATION OF RESERVOIR AUTOMATION

SETUP:



STEPS:

1. Library Inclusion: The code includes necessary libraries such as ESP32Servo, WiFi, and HTTPClient for servo control, WiFi connection, and HTTP communication respectively.

2. Wi-Fi Setup: It defines the Wi-Fi credentials (ssid and pass) and sets up a WiFiClient object for communicating with the Wi-Fi network.

3. ThingSpeak Configuration: It defines the ThingSpeak channel number (myChannelNumber) and the Write API Key (myWriteAPIKey) for sending data to ThingSpeak.

```
Servo myservo;
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// ThingSpeak settings
const char* server = "https://api.thingspeak.com/update?api_key=G9ZC6BA1873VW1TR&field1=0";
const char* apiKey = "G9ZC6BA1873VW1TR"; // Replace with your ThingSpeak Write API Key
```

4. Pin Configuration: The code defines the pins used for the ultrasonic sensor (trigPin and echoPin), potentiometer (potPin), relay control (relayPin), and servo control (servoPin).

```
const int trigPin = 5;
const int echoPin = 18;
const int potPin = 34;  // Analog pin for potentiometer
const int relayPin = 4; // Relay control pin
const int servoPin = 21; // Servo control pin
```

5. Global Variables: It defines global variables for storing distance measurements, HTTP status codes, and a tank level.

6. Setup Function:

   This function is called once at the beginning of the program. It initializes serial communication, sets pin modes, connects to WiFi, and sets the initial state of the relay and servo.

7. Loop Function:

   This function continuously executes the main logic of the program. It reads the distance from the ultrasonic sensor, simulates pH level readings from the potentiometer, controls the relay and servo based on the distance, and sends data to ThingSpeak. The loop then waits for 5 seconds before repeating.

2.3 SKETCH OF CODE

2.3.1 MAIN CODE(WOKWI CODE)

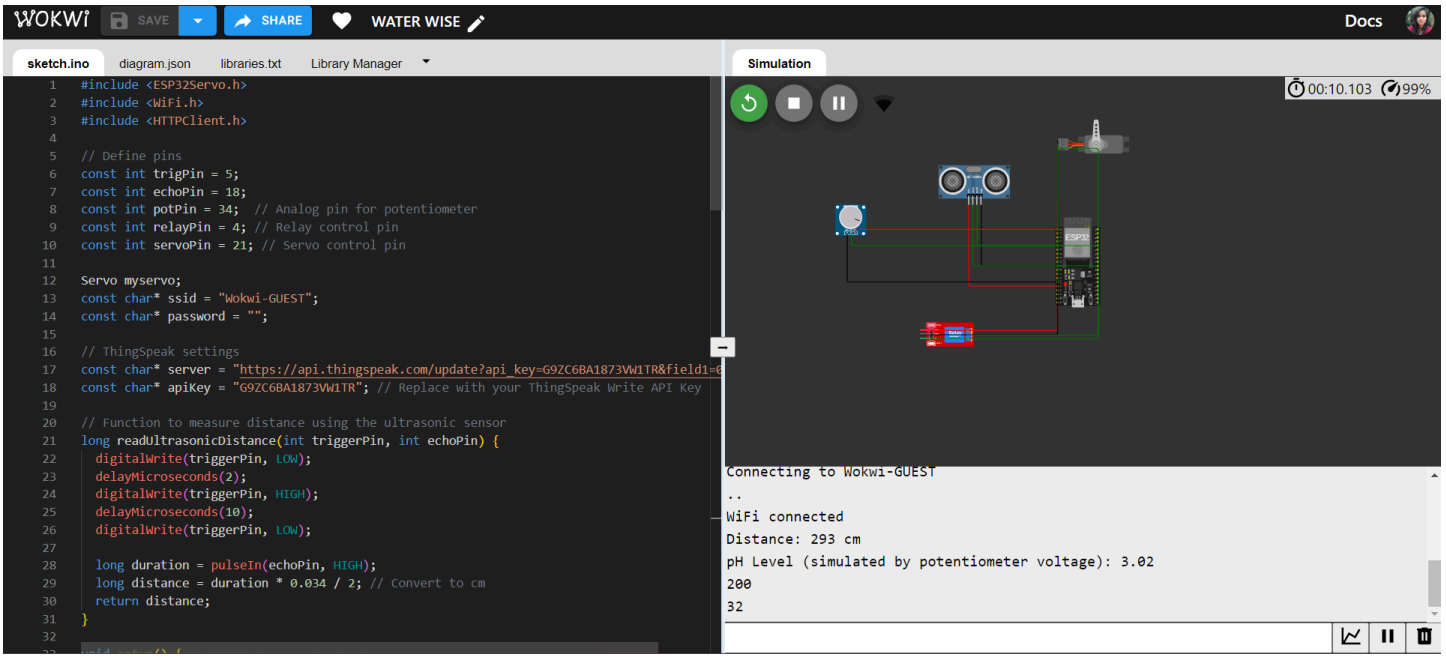
```
#include <ESP32Servo.h>
#include <WiFi.h>
#include <HTTPClient.h>

// Define pins
const int trigPin = 5;
const int echoPin = 18;
const int potPin = 34;  // Analog pin for potentiometer
const int relayPin = 4; // Relay control pin
const int servoPin = 21; // Servo control pin

Servo myservo;
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// ThingSpeak settings
const char* server =
"https://api.thingspeak.com/update?api_key=G9ZC6BA1873VW1TR&fiel
d1=0";
const char* apiKey = "G9ZC6BA1873VW1TR"; // Replace with your
ThingSpeak Write API Key
```

```
// Function to measure distance using the ultrasonic sensor
long readUltrasonicDistance(int triggerPin, int echoPin) {
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  long distance = duration * 0.034 / 2; // Convert to cm
  return distance;
}

void setup() {
  Serial.begin(115200);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(potPin, INPUT);
  pinMode(relayPin, OUTPUT);

  myservo.attach(servoPin);

  // Initial state of relay and servo
  digitalWrite(relayPin, LOW);
  myservo.write(0); // Initial position of servo

  // Connect to WiFi
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("WiFi connected");
}

void sendDataToThingSpeak(long distance, float voltage) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

    String url = String(server) + "?api_key=" + apiKey +
"&field1=" + String(distance) + "&field2=" + String(voltage);
    http.begin(url);

    int httpResponseCode = http.GET();
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println(httpResponseCode);
      Serial.println(response);
    } else {
      Serial.print("Error on sending POST: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  } else {
    Serial.println("WiFi not connected");
  }
}
```

```
void loop() {
  // Read the distance from the ultrasonic sensor
  long distance = readUltrasonicDistance(trigPin, echoPin);
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Read the pH level (simulated by potentiometer)
  int potValue = analogRead(potPin);
  float voltage = potValue * (3.3 / 4095.0); // Convert to
voltage (3.3V reference)
  Serial.print("pH Level (simulated by potentiometer voltage):
");
  Serial.println(voltage);
  // Control relay and servo based on distance
  if (distance <= 170) {
    digitalWrite(relayPin, HIGH); // Turn on the relay
    myservo.write(90); // Move servo to 90 degrees (or any
desired position)
  } else {
    digitalWrite(relayPin, LOW); // Turn off the relay
    myservo.write(0); // Move servo to 0 degrees (or any desired
position)
  }

  // Send data to ThingSpeak
  sendDataToThingSpeak(distance, voltage);

  delay(5000); // Wait for 5 seconds before the next loop
}
```

SETUP:



OUTPUT:

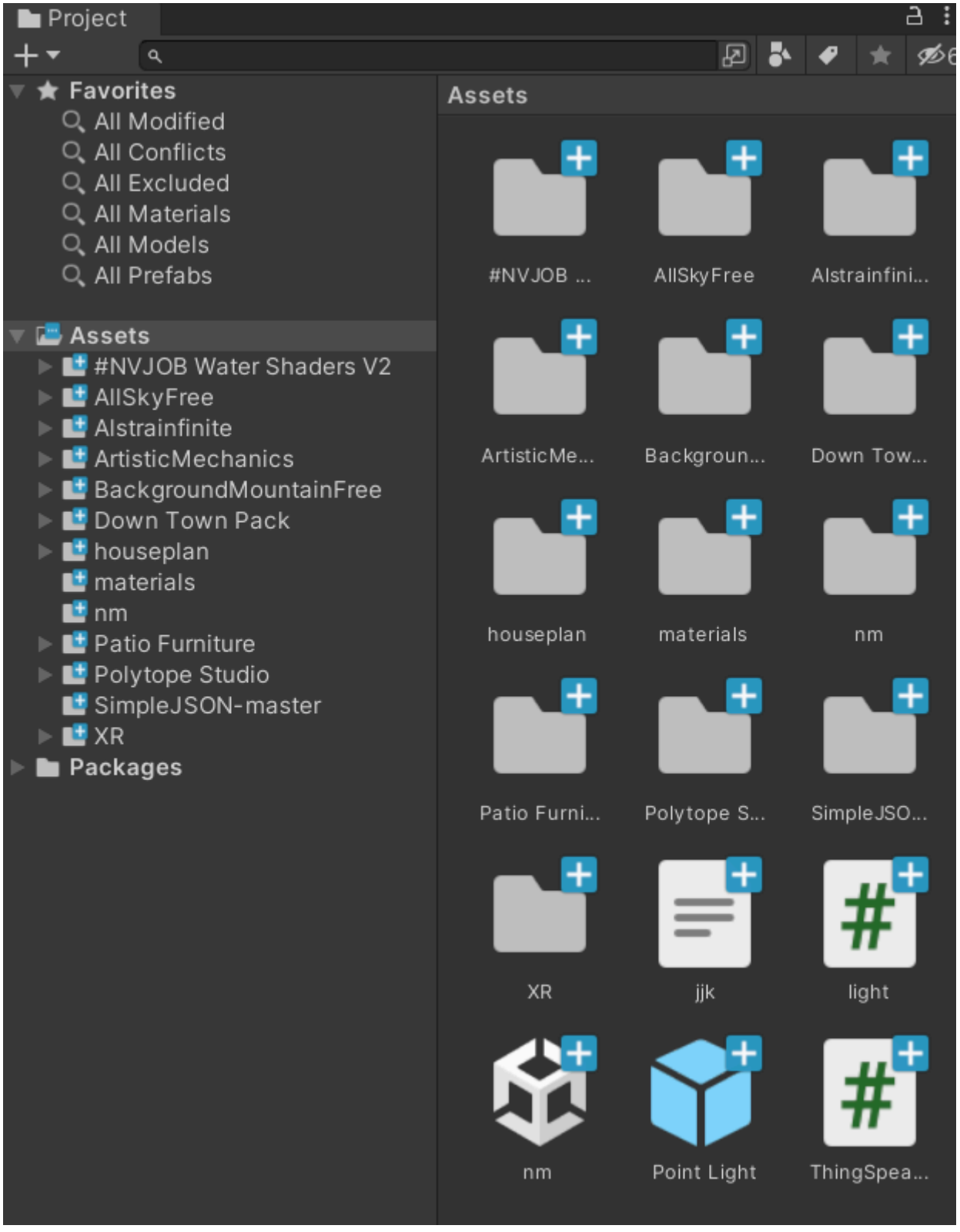
```
..
WiFi connected
Distance: 293 cm
pH Level (simulated by potentiometer voltage): 3.02
200
32
Distance: 293 cm
pH Level (simulated by potentiometer voltage): 3.02
```

GAME ENGINE

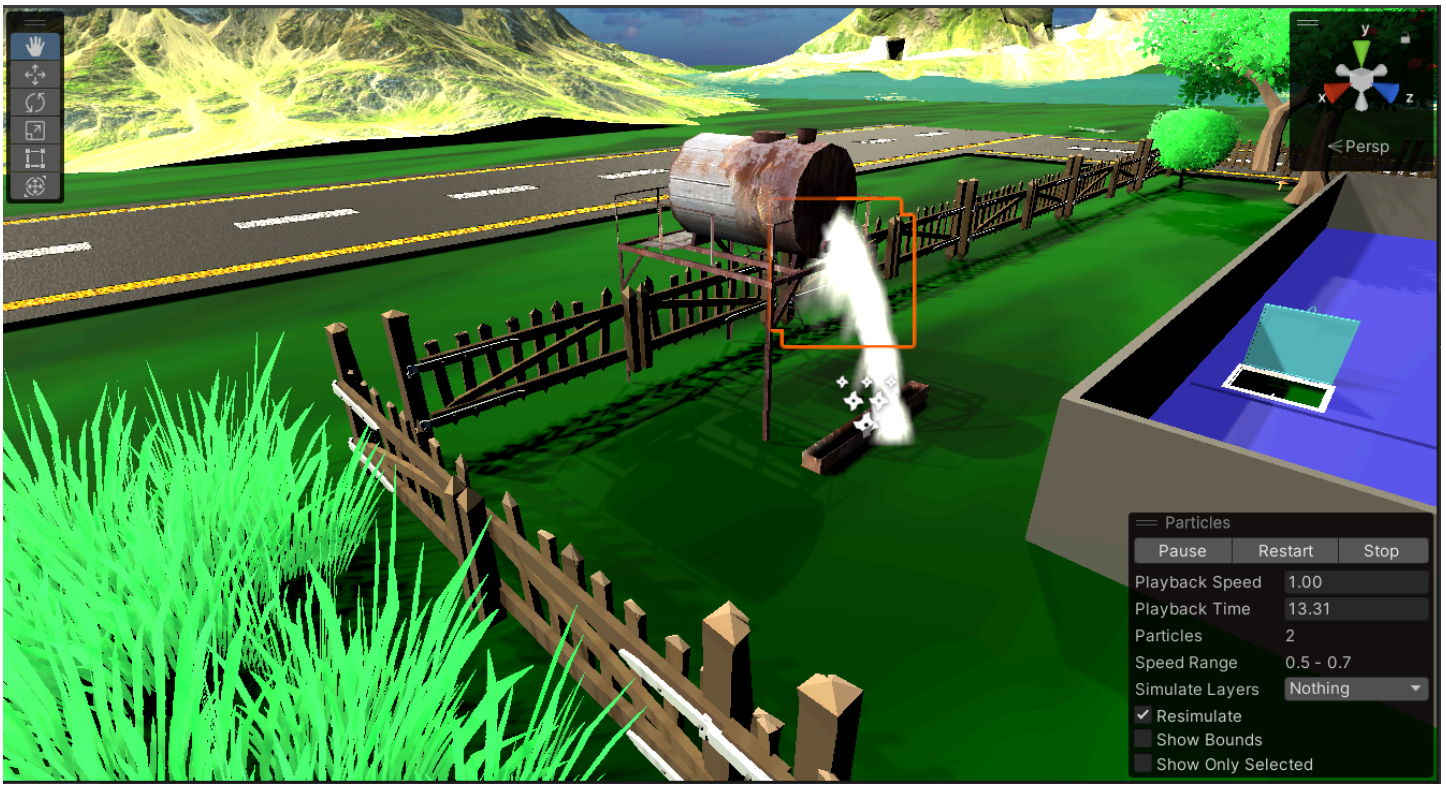
3.1.1 UNITY HUB AND EDITOR

- Unity Hub is a project management tool that organises Unity projects and manages different versions of Unity Editor.
- Unity Editor is the development environment where you create, modify, and test Unity projects. It includes scene design, scripting, asset management, and preview/testing features.

3.2 ASSETS AND ENVIRONMENT

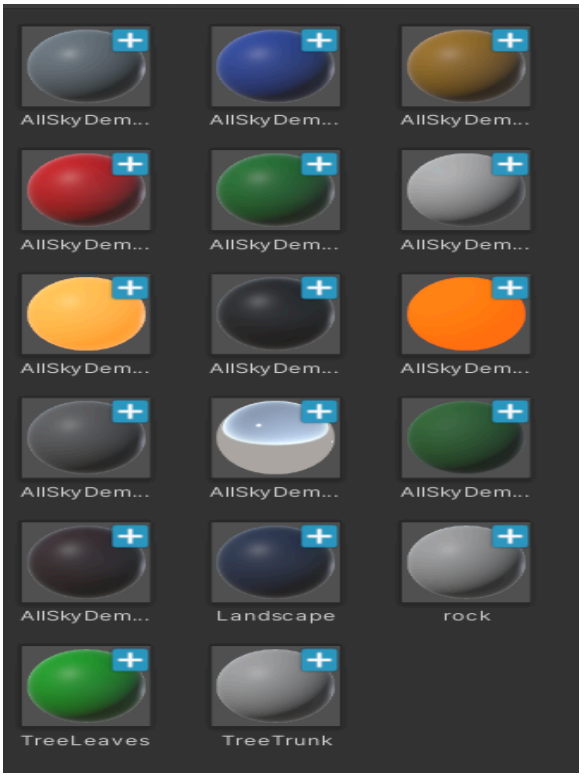


Environment:


y
x
z
Persp
Particles
Pause
Restart
Stop
Playback Speed 1.00
Playback Time 13.31
Particles 2
Speed Range 0.5 - 0.7
Simulate Layers Nothing
Resimulate
Show Bounds
Show Only Selected


y
x
Back

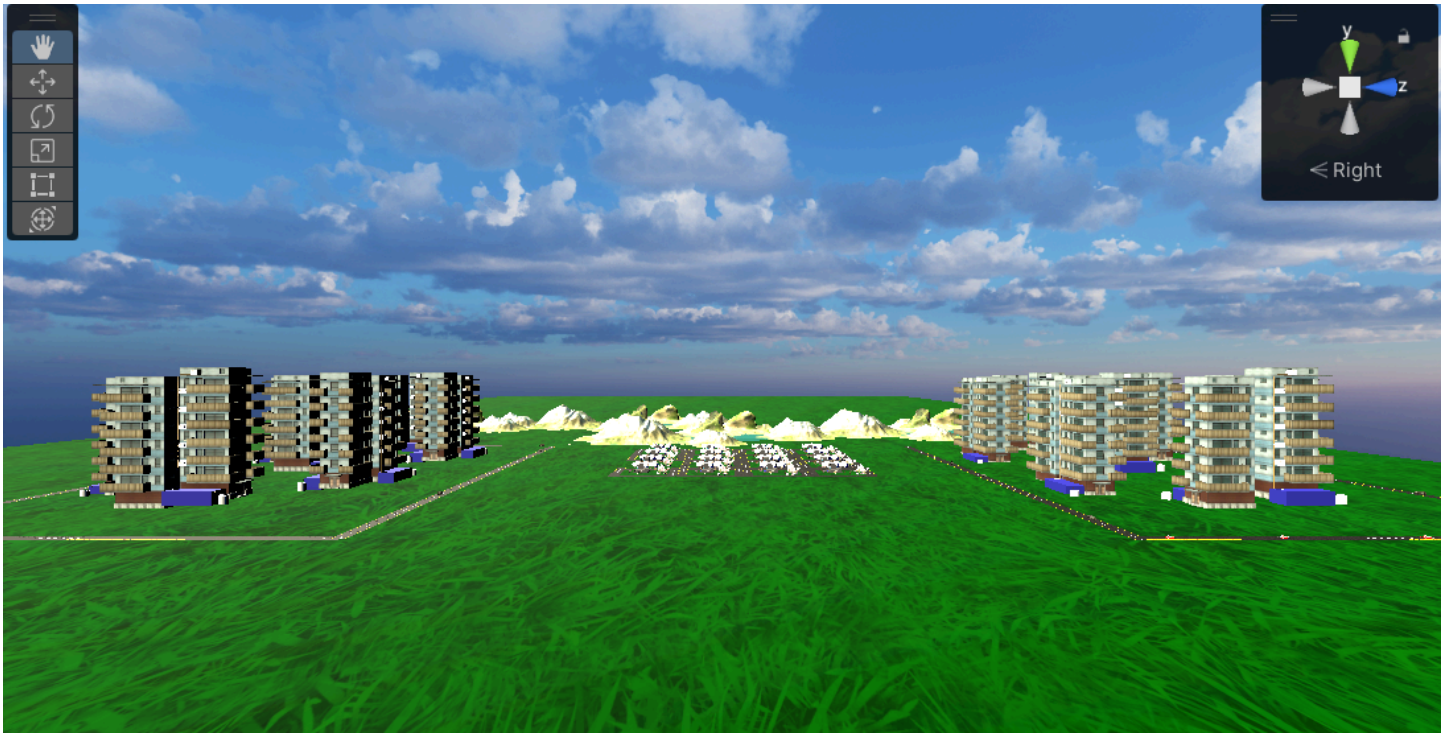
3.2.1 MATERIALS



3.3 ENVIRONMENT SETUP

3.3.1 FINALISING

OVERVIEW:



SIMULATION

4.2 C# SCRIPT

SCRIPT:

```
using UnityEngine;
using System.Collections;
using UnityEngine.Networking;
using SimpleJSON;

public class thingspeak : MonoBehaviour
{
    public string channelID = "2564261";
    public string apiKey = "G9ZC6BA1873VW1TR";

    private string distanceValue = "";
    private string disValue = "";
    private string relayState = "";

    IEnumerator Start()
    {
        while (true)
        {
                                                yield    return
StartCoroutine(GetDataFromThingSpeak());
            yield return new WaitForSeconds(15f); // Fetch data
every 15 seconds
        }
    }

    IEnumerator GetDataFromThingSpeak()
    {
          string url = "https://api.thingspeak.com/channels/" +
channelID + "/feeds/last.json?api_key=" + apiKey;
```

```
        UnityWebRequest www = UnityWebRequest.Get(url);
        yield return www.SendWebRequest();

        if (www.result != UnityWebRequest.Result.Success)
        {
                    Debug.LogError("Failed to fetch data: " +
www.error);
        }
        else
        {
                                        JSONNode jsonNode =
JSON.Parse(www.downloadHandler.text);
            UpdateFieldValues(jsonNode);
            ControlRelayState();
        }
    }

    void UpdateFieldValues(JSONNode jsonNode)
    {
        if (jsonNode["field1"] != null)
        {
            float distance = jsonNode["field1"].AsFloat;
            distanceValue = "Waterlevel: " + distance + " cm";
        }
        else
        {
            distanceValue = "Water level data not available";
        }

        if (jsonNode["field2"] != null)
```

```
        {
            float dis = jsonNode["field2"].AsFloat;
            disValue = "PH VALUE: " + dis + " pH";
        }
        else
        {
            disValue = "Water level data not available";
        }
    }

    void ControlRelayState()
    {
         // Extracting distance and dis values from the received
data
        float distance = GetDistanceValue(distanceValue);
        float dis = GetDistanceValue(disValue);

        // Check the conditions to determine the relay state
        if (distance >= 170)
        {
            relayState = "SERVO ON";
            Debug.Log("SERVO ON");
        }
        else if (distance <= 50)
        {
            relayState = "SERVO OFF";
            Debug.Log("SERVO OFF");
        }
        else
        {
            relayState = "SERVO OFF";
```

```
            Debug.Log("SERVO Off");
        }
    }
    float GetDistanceValue(string data)
    {
        // Extract the distance value from the received data
        float distance = 0;
        int index = data.IndexOf(":");
        if (index != -1)
        {
            int startIndex = index + 2;
            int endIndex = data.IndexOf(" ", startIndex);
            if (endIndex != -1)
            {
                        float.TryParse(data.Substring(startIndex,
endIndex - startIndex), out distance);
            }
        }
        return distance;
    }

    void OnGUI()
    {
        GUI.Label(new Rect(10, 10, 200, 20), distanceValue);
        GUI.Label(new Rect(10, 30, 200, 20), disValue);
        GUI.Label(new Rect(10, 50, 200, 20), relayState);
    }
                                }
```

STEPS:

1. Variables:
    - It declares variables for ThingSpeak channel ID, API key, and data values.
2. Fetching Data:
    - It uses a coroutine to fetch data from a ThingSpeak channel every 15 seconds.
3. Getting Data from ThingSpeak:
    - It sends a request to the ThingSpeak API to get the latest data from the specified channel.
4. Updating Field Values:
    - It parses the JSON response and updates the distance and dis values accordingly.
5. Controlling Relay State:
    - It determines the relay state based on the received distance and dis values.
6. Extracting Distance Value:
    - It extracts the distance value from the received data.
7. Displaying Data on GUI:
    - It displays the distance value, dis value, and relay state on the Unity GUI.

4.2.1 LIBRARIES

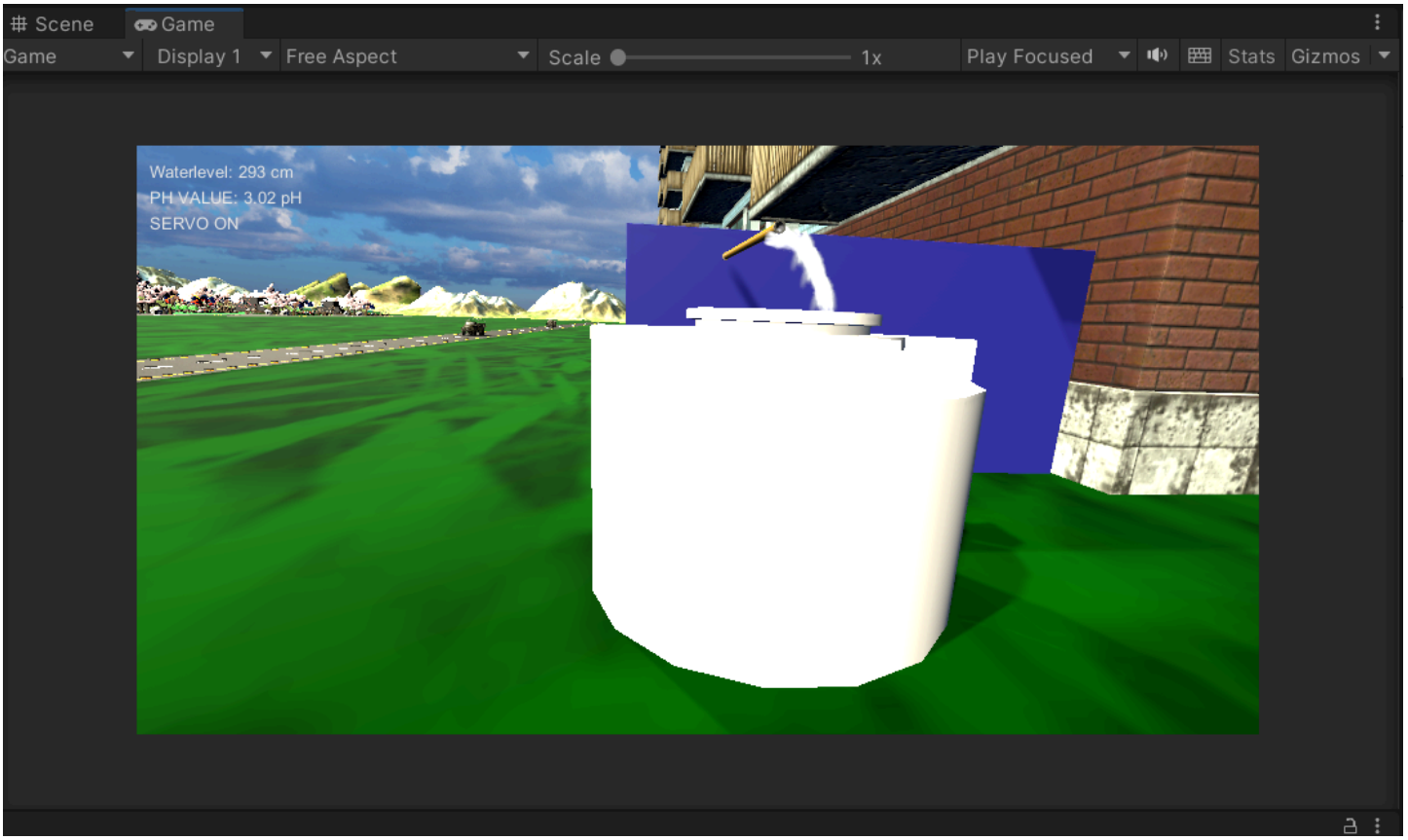
SIMPLE JSON

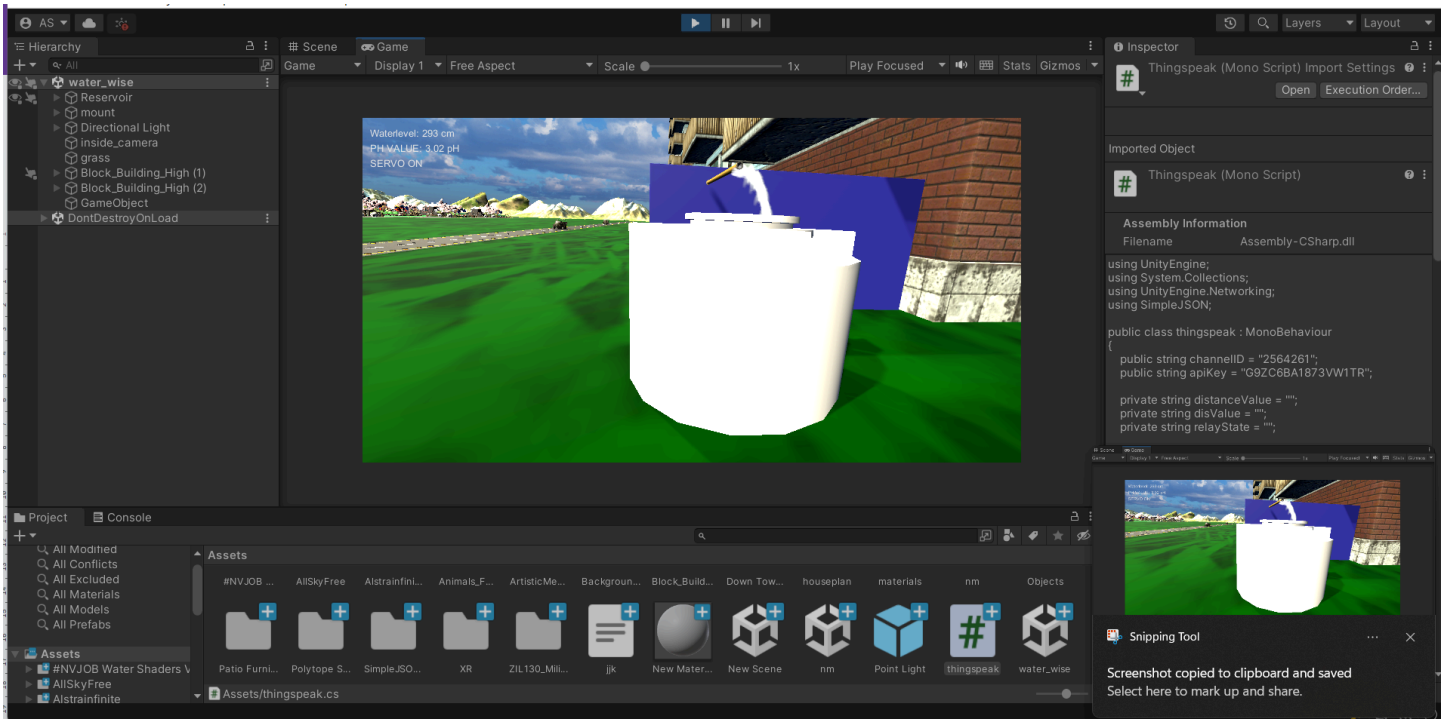
Simple JSON in C# is crucial for data serialisation, storage, and API communication in Unity. It provides an easy way to handle JSON data, making it essential for tasks like saving game settings, communicating with web services, and storing configuration files.

4.3 VIRTUALIZATION

4.3.1 DISPLAY

4.3.1.1 GAME MODE





The script fetches data from a ThingSpeak channel using its API. Here's an explanation of the output:

1. Distance Value(ULTRASONIC VALUE OF WATER LEVEL):
    - It displays the distance value in centimetres retrieved from the ThingSpeak channel.
2. Dis Value(pH VALUE):
    - It displays the 'dis' value in centimetres retrieved from the ThingSpeak channel.
3. Relay State(SERVO ON AND OFF):
    - It indicates whether the relay should be ON or OFF based on the distance and dis values:
        - If the distance is less than or equal to 100 cm and 'dis' is greater than or equal to 100 cm, the relay state is set to "Relay ON".

        - If the distance is greater than or equal to 380 cm, the relay state is set to "Relay OFF".
        - Otherwise, the relay state is set to "Relay OFF".

The output continuously updates as new data is fetched from the ThingSpeak channel every 15 seconds.

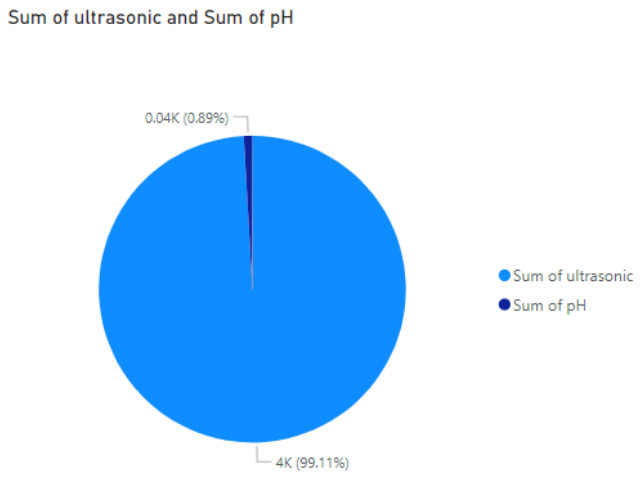
INSIGHTS AND ANALYTICS

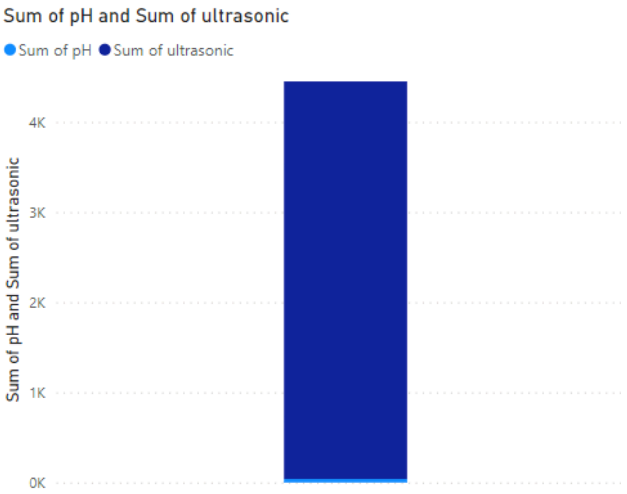
POWER BI

DATA IMPORT

DATA REQUEST FROM THINGSPEAK

DATA CHART





created_at	entry_id	ultrasonic	pH	latitude	longitude	elevation	status
29-05-2024 14:17:57	1	399	0				
29-05-2024 14:18:21	2	399	2.4				
29-05-2024 14:23:14	3	0					
29-05-2024 14:29:57	4	399	2.4				
29-05-2024 15:03:18	5	177	2.51				
29-05-2024 16:47:04	6	399	0				
29-05-2024 16:47:35	7	126	0				
29-05-2024 16:47:57	8	357	2.55				
29-05-2024 16:48:19	9	357	2.55				
29-05-2024 16:48:41	10	357	2.55				
29-05-2024 16:50:15	11	70	2.55				
29-05-2024 16:50:32	12	329	2.55				
29-05-2024 16:51:11	13	225	2.55				
29-05-2024 16:53:20	14	166	2.55				
29-05-2024 16:53:42	15	247	0				
29-05-2024 16:54:27	16	36	2.41				
29-05-2024 16:54:51	17	73	3.02				
29-05-2024 16:55:16	18	205	3.02				
29-05-2024 16:55:38	19	45	3.02				
29-05-2024 16:56:00	20	45	3.02				

feeds (1)

- created_at
- elevation
- Σ entry_id
- latitude
- longitude
- Σ pH
- status
- Σ ultrasonic

Collapse

feeds (2)

- created_at
- elevation
- Σ entry_id
- latitude
- longitude
- Σ pH
- status
- Σ ultrasonic

Collapse

Proposal:

- Air conditioners (1.5 to 3 ton units) typically produce 19 to 75 litres of water per day under normal conditions, and up to 75 litres per day in high humidity. Often, this water is wasted or sent to the sewage system. However, this condensate is almost pure and can be repurposed for watering rooftop gardens, supporting water harvesting efforts, and providing for thirsty animals and trees. Given the rising temperatures in the 21st century, especially in regions like India, our "Water Wise" project aims to reuse this water effectively to address these needs and mitigate water scarcity throughout the year.

- Many people want to help address water scarcity but struggle due to the demands of daily life. Our project, "Water Wise," offers a time-efficient, user-friendly solution that automatically saves and repurposes water from air conditioner condensate. This system not only prevents water waste but also provides real-time analytics to users, helping them monitor and optimise water usage. By capturing and reusing the water typically wasted by AC systems, "Water Wise" supports sustainable practices effortlessly.

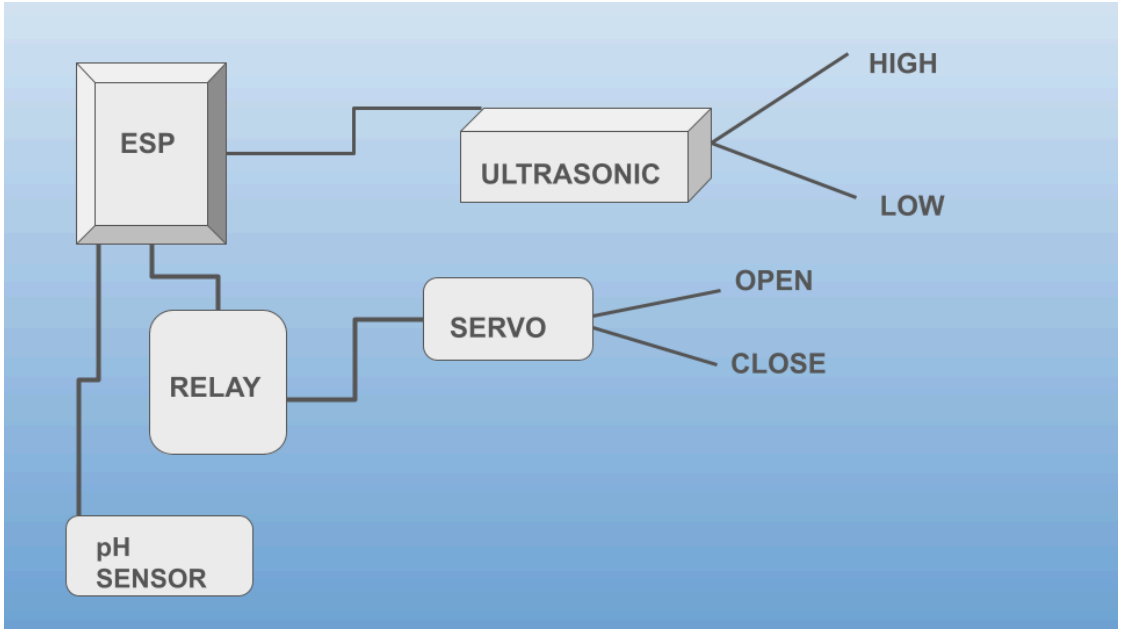


FIG: BASIC BLOCK OF WATER-WISE

EXECUTION PLAN:

Initially when AC is used,the condensed water begins to be run into the externally fitted tank and ultrasonic actively detects the system water levels. When exceeded its limit an alert is created spotting the relay on which instructs the servo to open the valve to let the water be stored in the inside fitted tank which can be readily used for repurposing. In order to keep in track of the ph level of the water stored, a ph sensor is used which detects suitability to use.

The average setup cost of this system is under 5k,which is affordable conveniently by most AC owners.In further advance, this can be connected into an active monitoring data cloud interfaced with an app which alerts the users.Depending on the budget laid out by the users this plan can be modified with ease.

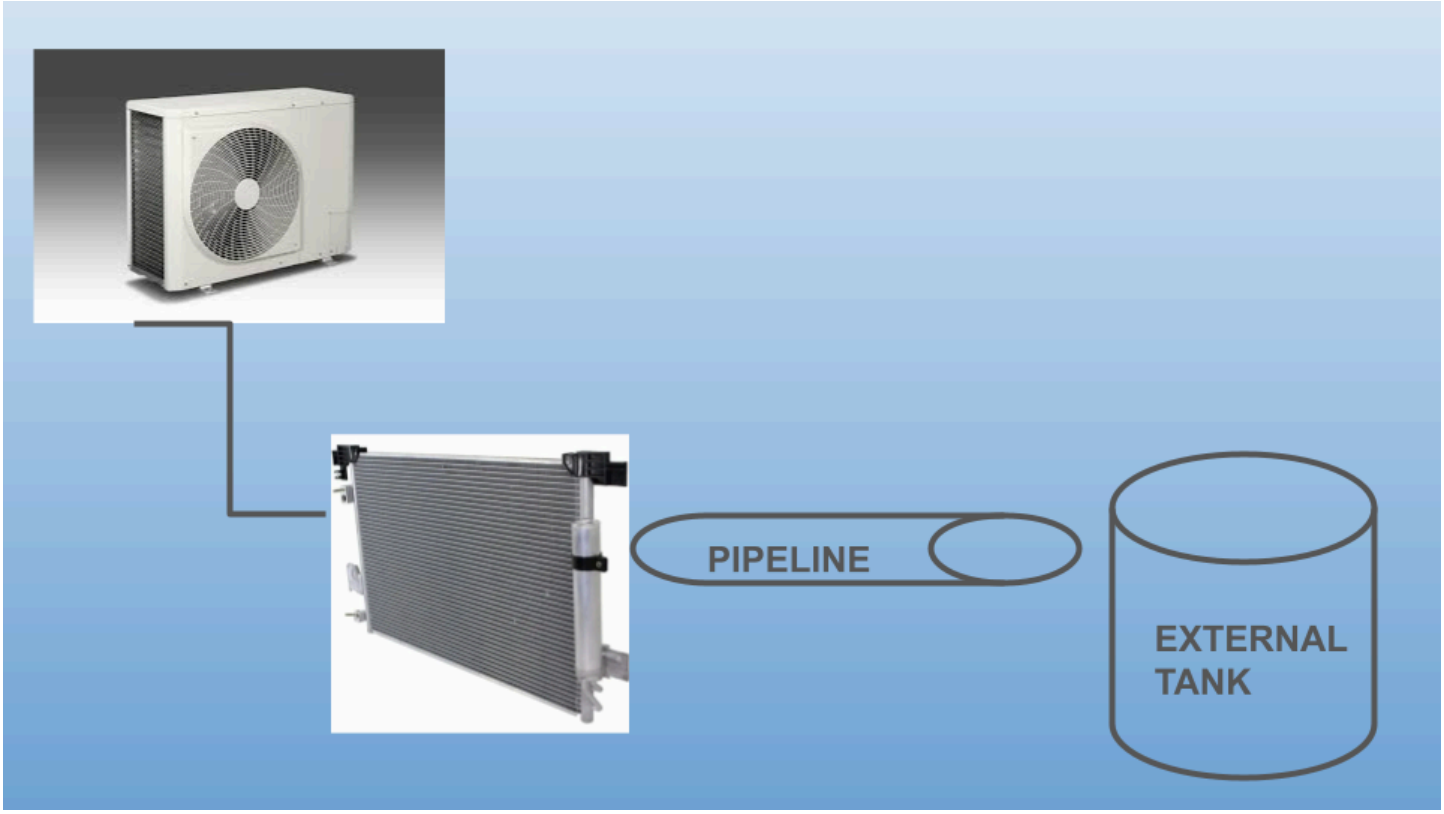


FIG:CONDENSER FLOW

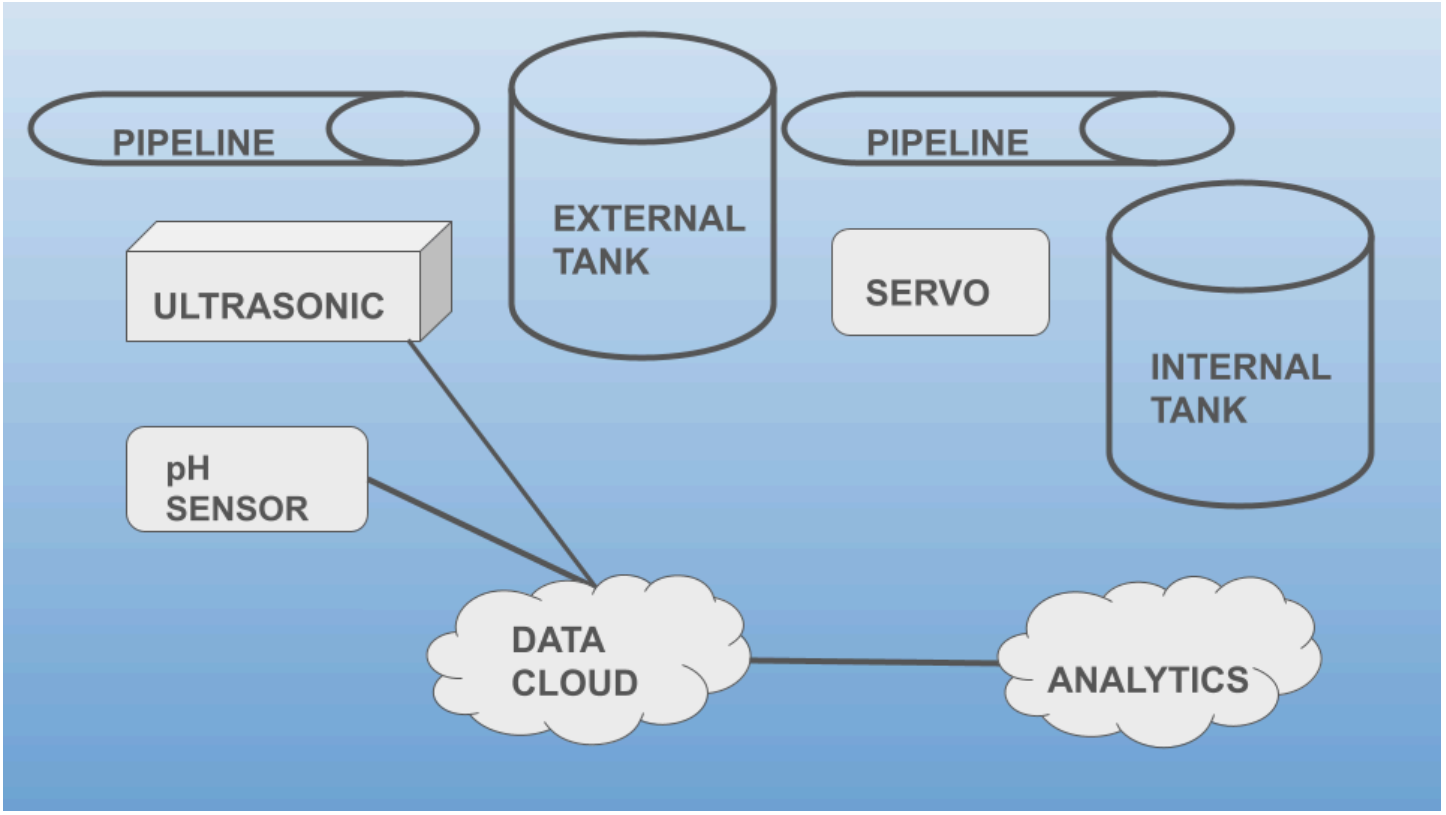


FIG:COMPLETE OUTLOOK

About our system :

1. This is a new intervention and the most easiest responsible form of water saving yet using Air conditioners. No such system was thought to be utilised as a repurposeful option of water.

2. This system promotes roof top or indoor simple gardening amidst water demands.

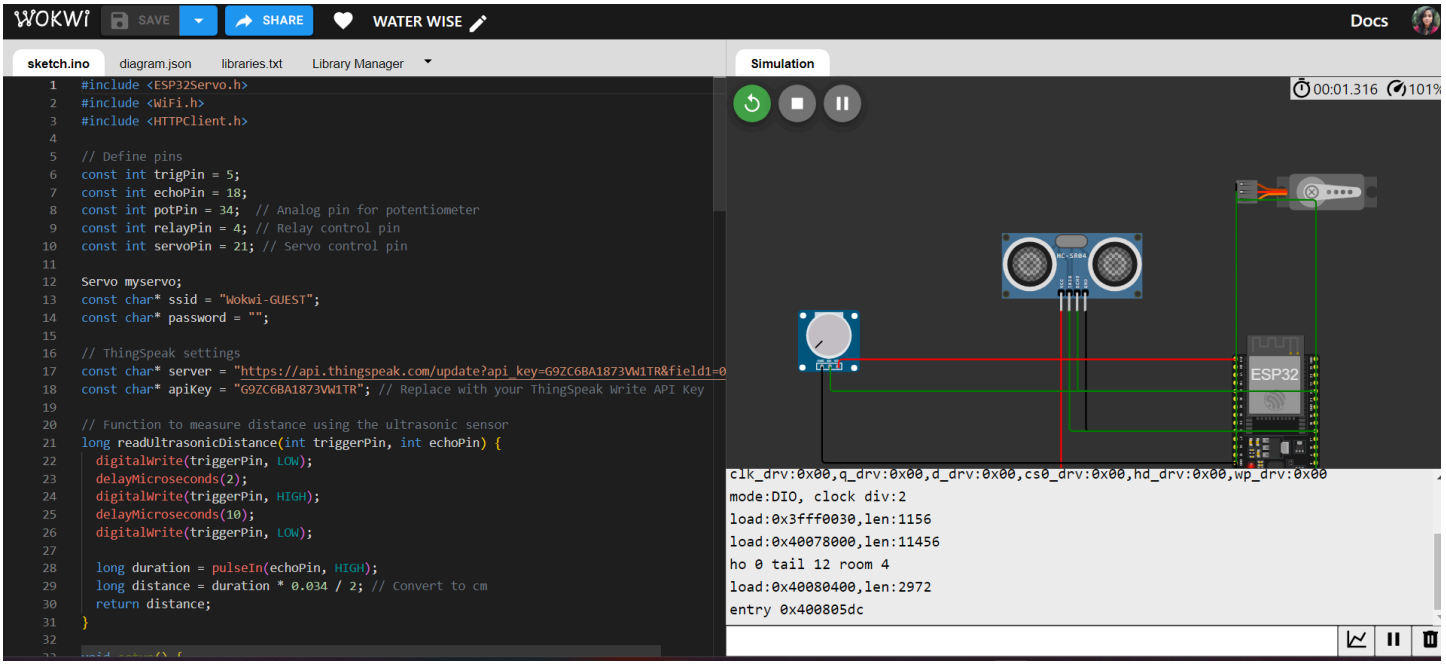
3. This system influences people to look after strays(animals,birds and people).

4. This system actively promotes water saving policy by all means.

5. No water used is waste. Using water wisely saves millions of living being and millions of expense on water spent.

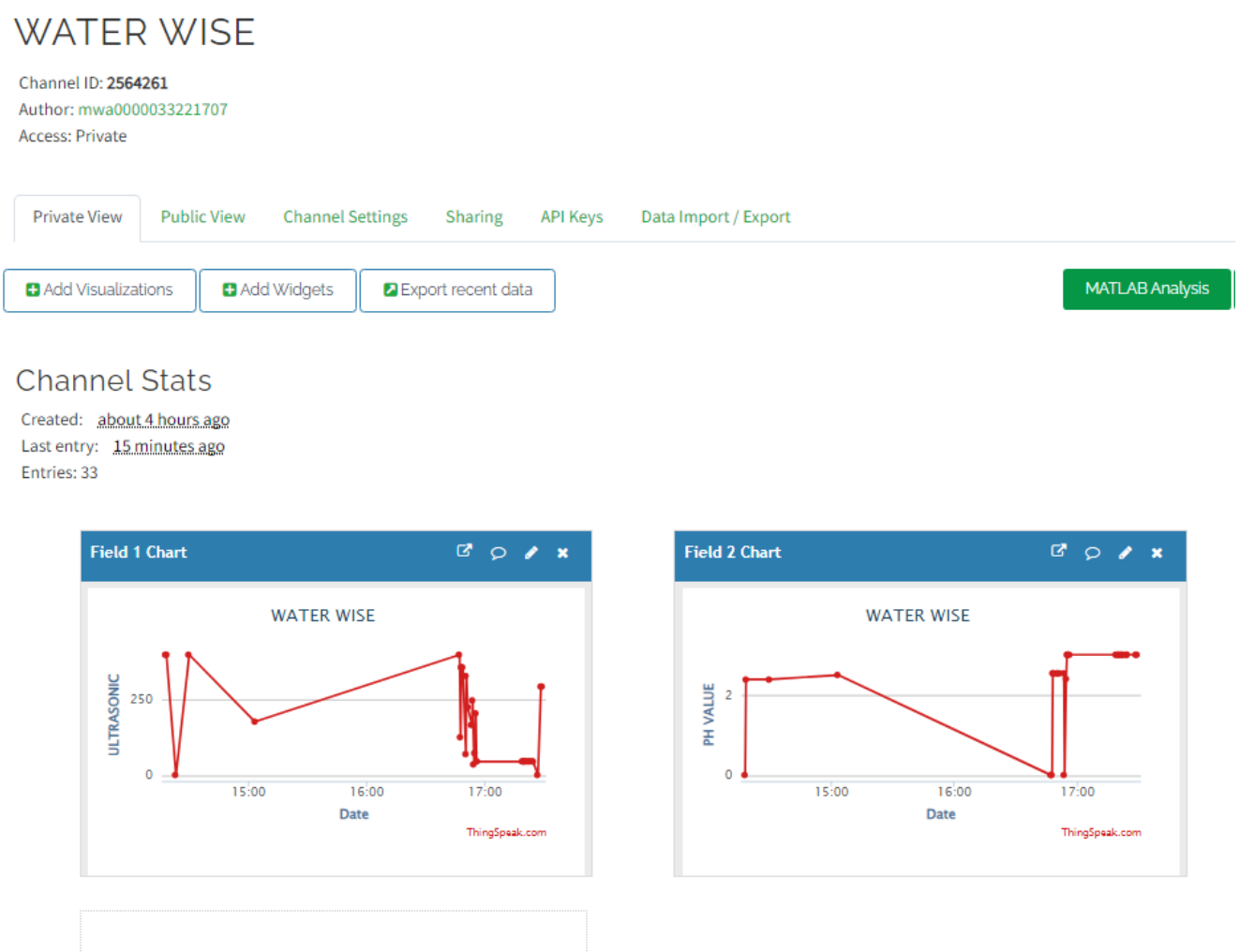
CONCLUSION

WOKWI SIMULATION:

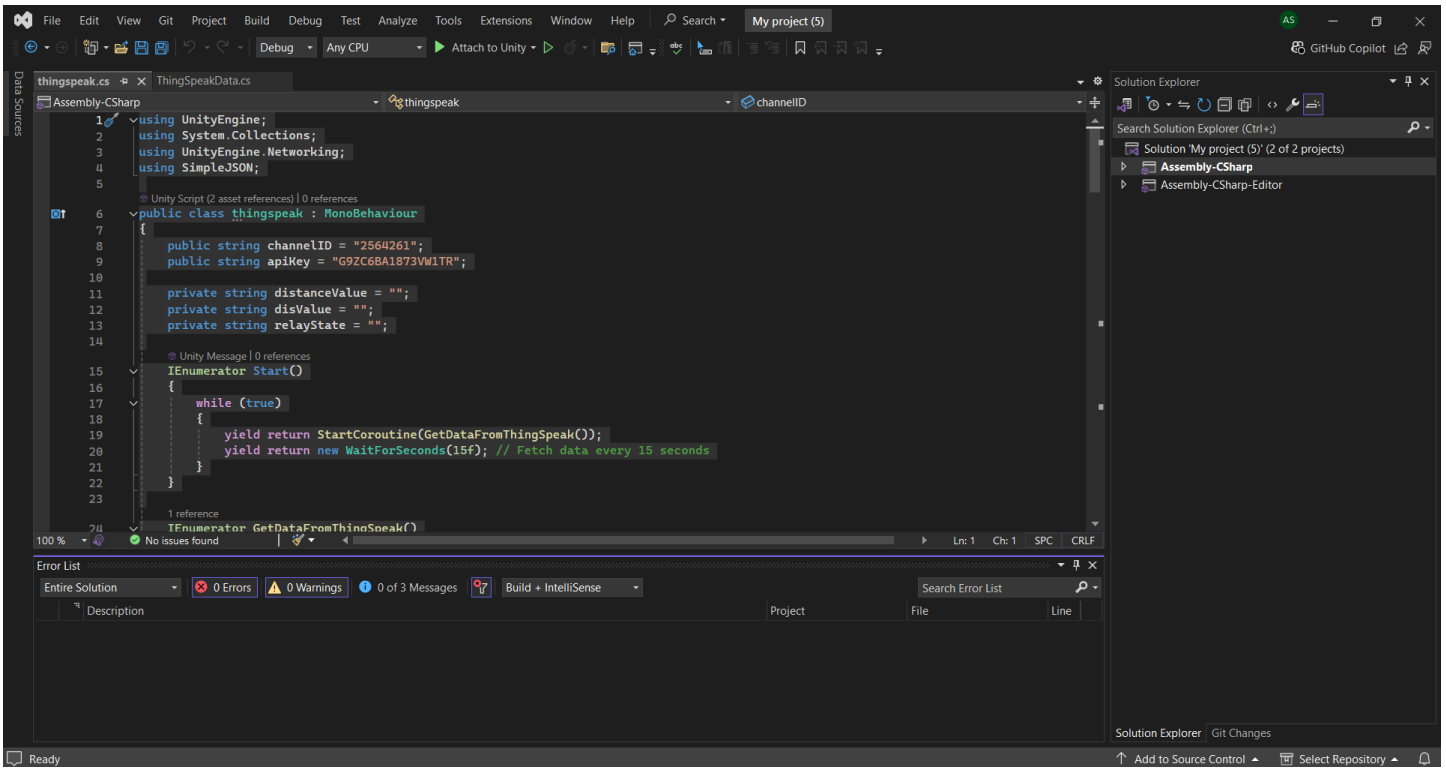



```ino
#include <ESP32Servo.h>
#include <WiFi.h>
#include <HTTPClient.h>

// Define pins
const int trigPin = 5;
const int echoPin = 18;
const int potPin = 34;  // Analog pin for potentiometer
const int relayPin = 4; // Relay control pin
const int servoPin = 21; // Servo control pin

Servo myservo;
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// ThingSpeak settings
const char* server = "https://api.thingspeak.com/update?api_key=G9ZC6BA1873VW1TR&field1=0
const char* apiKey = "G9ZC6BA1873VW1TR"; // Replace with your ThingSpeak Write API Key

// Function to measure distance using the ultrasonic sensor
long readUltrasonicDistance(int triggerPin, int echoPin) {
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  long distance = duration * 0.034 / 2; // Convert to cm
  return distance;
}
```

THINGSPEAK DATA BASE:

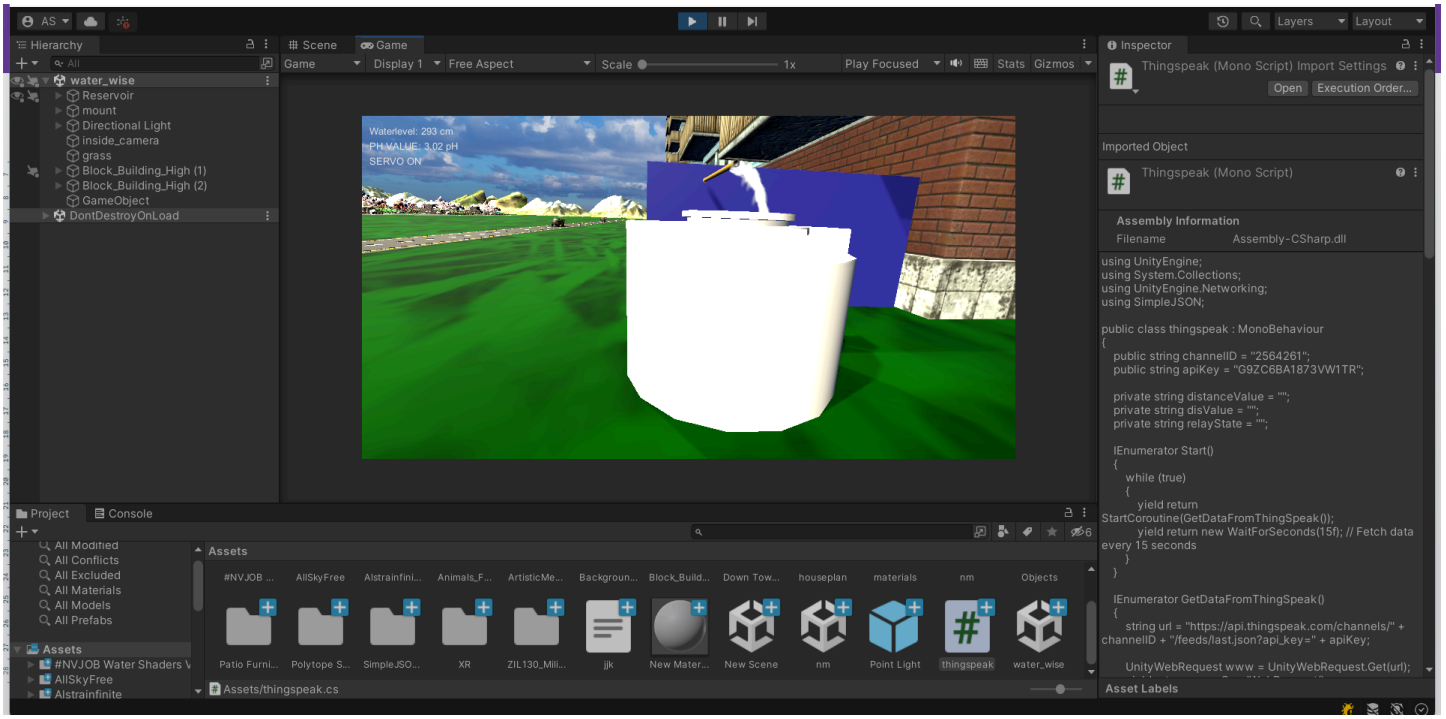


C SHARP EXECUTION:



```
using UnityEngine;
using System.Collections;
using UnityEngine.Networking;
using SimpleJSON;

Unity Script (2 asset references) | 0 references
public class thingspeak : MonoBehaviour
{
    public string channelID = "2564261";
    public string apiKey = "G9ZC6BA1873VW1TR";

    private string distanceValue = "";
    private string disValue = "";
    private string relayState = "";

    Unity Message | 0 references
    IEnumerator Start()
    {
        while (true)
        {
            yield return StartCoroutine(GetDataFromThingSpeak());
            yield return new WaitForSeconds(15f); // Fetch data every 15 seconds
        }
    }

    1 reference
    IEnumerator GetDataFromThingSpeak()
```

OUTPUT OF UNITY:



By integrating the "Water Wise" project, we harness precision and efficiency to revolutionise water management. This innovative system optimises the distribution of condensate water from air conditioners, ensuring it reaches its full potential in nurturing rooftop gardens, quenching the thirst of street animals, and fostering the growth of urban greenery. With customizable flow rates and automated controls, we minimise wastage while maximising impact, promoting sustainability and enhancing the well-being of our communities and ecosystems. Together, we redefine the role of water, transforming it from a mere resource into a powerful agent of change in our quest for a greener, more sustainable future.

REFERENCE

1. B. Kakde, A. Lokhande, and N. Thakare, "Automated water level monitoring system using IoT," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 2017, pp. 1-5.

[Link] (https://ieeexplore.ieee.org/document/8276003)

2. S. A. Zungeru, M. A. Bashir, and M. G. Kibiwott, "A Review of Internet of Things (IoT) in Water Management," 2016 International Conference on Computing, Networking and Informatics (ICCNI), Nairobi, Kenya, 2016, pp. 1-6.

[Link](https://ieeexplore.ieee.org/document/7808570)