Smart Water Fountains

Development Part-2

Date	**23-10-2023**
Team ID	**509**
Project Name	**Smart Water Fountains**

Table of Contents

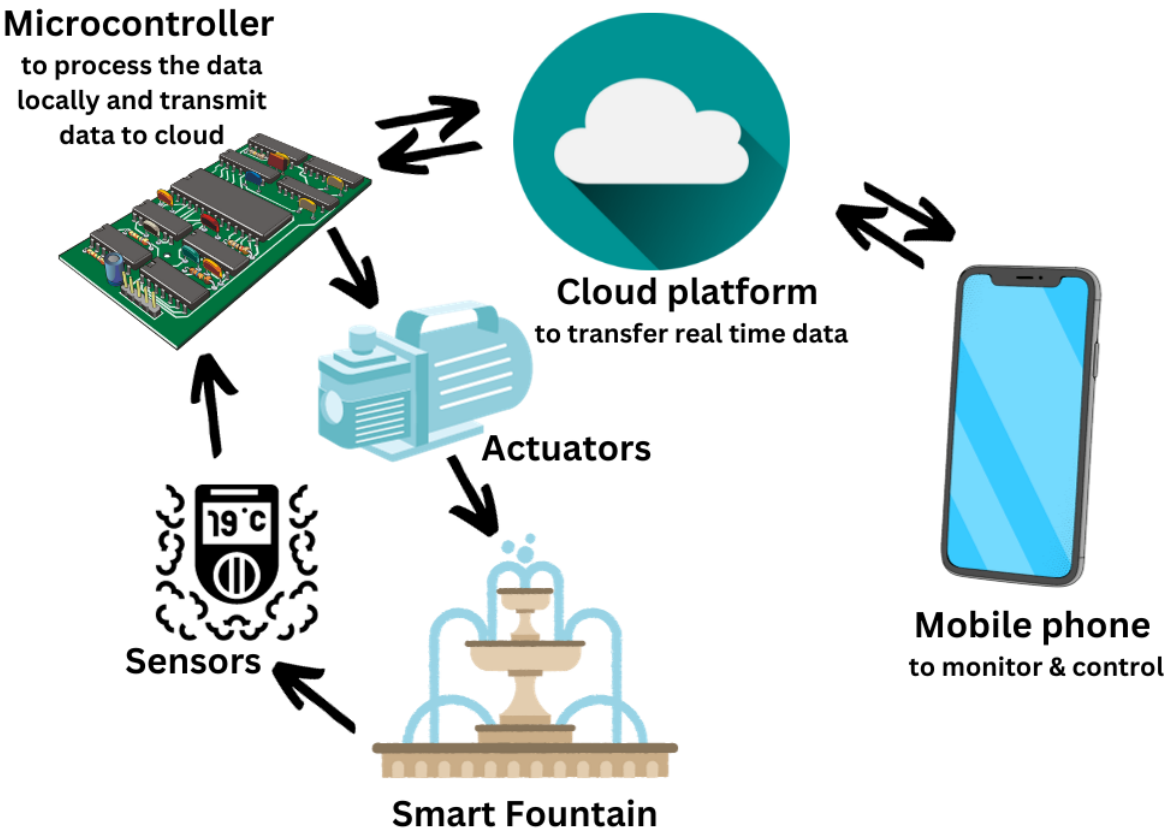
1. Introduction

Optimizing the efficiency of smart water fountains is crucial to promote water conservation and responsible usage. In this document, we will outline a Internet of Things (IoT) project focused on maximizing the efficiency of smart water fountains. We'll define the problem statement, outline the steps involved, and discuss how an automated approach can benefit water management and sustainability efforts.

2. Problem Statement

It is hard for us to optimize the water fountain to minimize the water usage based on various factors. We going to develop a model that optimizes the water usage of smart water fountains based on various factors such as foot traffic, weather, and time of day.

3. Block Diagram



4.Code

Since the Python Libraries are not available in Wokwi we did it Using C/C++

Micro Python Code

```
from machine import Pin, PWM, Timer, ADC
from ultra import DistanceSensor
from time import sleep

ds = DistanceSensor(echo=14, trigger=15)

Fountain_motor = 2
motor_num=27

pin = machine.Pin(Fountain_motor, machine.Pin.OUT)
```

```
motor=machine.Pin(motor_num,machine.Pin.OUT)
while True:
    distance_cm = ds.distance * 100
    distance=float(distance_cm)
    print(f"Distance: {distance_cm} cm")
#to refill water in the reservoir automatically------------------------------
    if distance < 10:
          #denotes the distance btw sensor and water
            print("full")
            pin.on()
            motor.off()#to turn motor off
    else:
            print("filling ")
            pin.off()
            motor.on()#to turn motor on

#-------------------------------------------------------------------------
    sleep(0.1)
```

#This code is to make the work automatically based on water level we can also integrate many sensors and for your reference I'm giving the wokwi link

https://wokwi.com/projects/379535806439664641----------

Code in C/C++

```
#define BLYNK_TEMPLATE_ID "TMPL3-VBCPYBi"
#define BLYNK_TEMPLATE_NAME "water fountain"
#define BLYNK_AUTH_TOKEN "81q5QUnUB7tqWGrbsLJgcfrIsAqleTnF"

#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char ssid[] = "Wokwi-GUEST";
char pass[] = "";

BlynkTimer timer;
```

```
// This function is called every time the Virtual Pin 0 state changes
BLYNK_WRITE(V0)
{
  // Set incoming value from pin V0 to a variable
  int value = param.asInt();

  // Update state
  Blynk.virtualWrite(V1, value);
}


// This function is called every time the device is connected to the
Blynk.Cloud
BLYNK_CONNECTED()
{
  // Change Web Link Button message to "Congratulations!"
  Blynk.setProperty(V3, "offImageUrl", "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");
  Blynk.setProperty(V3, "onImageUrl",  "https://static-
image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations_pressed.png"
);
  Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-started/what-
do-i-need-to-blynk/how-quickstart-device-was-made");
}


// This function sends Arduino's uptime every second to Virtual Pin 2.
void myTimerEvent()
{
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V2, millis() / 1000);
}


#define Fountain_motor 17
#define Motor 4
#define trigerpin1 27
#define echopin1 26
float a;
int san();

void setup(){
  Serial.begin(9600);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  pinMode(trigerpin1, OUTPUT);
  pinMode(echopin1, INPUT);
  pinMode(Fountain_motor, OUTPUT);
```

```
  pinMode(Motor, OUTPUT);
  //used to measure duration of clock pulses



  Serial.println("hello sandy");
}


void loop(){
  a=san(trigerpin1,echopin1);
  Serial.print("distance (in cm)=");
  Serial.println(a);
  Blynk.virtualWrite(V0, a);
  delay(2000);

  //To show water level of the fountain
  Blynk.virtualWrite(V3, 4);
  if(a>10&&a<30){
    Blynk.virtualWrite(V3, 3);
  }else if(a>30&&a<50){
    Blynk.virtualWrite(V3, 2);
  }else if(a>50&&a<70){
    Blynk.virtualWrite(V3, 1);
  }else if(a>70){
    Blynk.virtualWrite(V3, 0);
  }

  //To turn on and of the fountain automatically
  if(a<10){
    digitalWrite(Fountain_motor, HIGH);
    digitalWrite(Motor, LOW);
    Blynk.virtualWrite(V1, 1);//foun
    Blynk.virtualWrite(V2, 0);
    Blynk.virtualWrite(V4, "Fountain Pump ON");
  }
  else{
    digitalWrite(Motor, HIGH);
    digitalWrite(Fountain_motor, LOW);
    Blynk.virtualWrite(V1, 0);//foun
    Blynk.virtualWrite(V2, 1);
    Blynk.virtualWrite(V4, "Filling Reservoir");

  }
}
int san(int trigerpin, int echopin){
    float distance, duration;
    digitalWrite(trigerpin, LOW);
```

```
  delay(2);
  digitalWrite(trigerpin, HIGH);
  delay(10);
  digitalWrite(trigerpin, LOW);
  duration=pulseIn(echopin,HIGH);
  distance=duration/2*0.034;
  //Serial.print("distance (in cm)=");
  //Serial.println(distance);
  delay(10);
  return distance;
}
```

5. Working Principle

Control and Monitor:

With the software in place, We can now control and monitor your smart water fountain from a smartphone or computer. we can turn the fountain on/off remotely, check water levels, and receive alerts if water levels drop too low or if there are temperature issues.

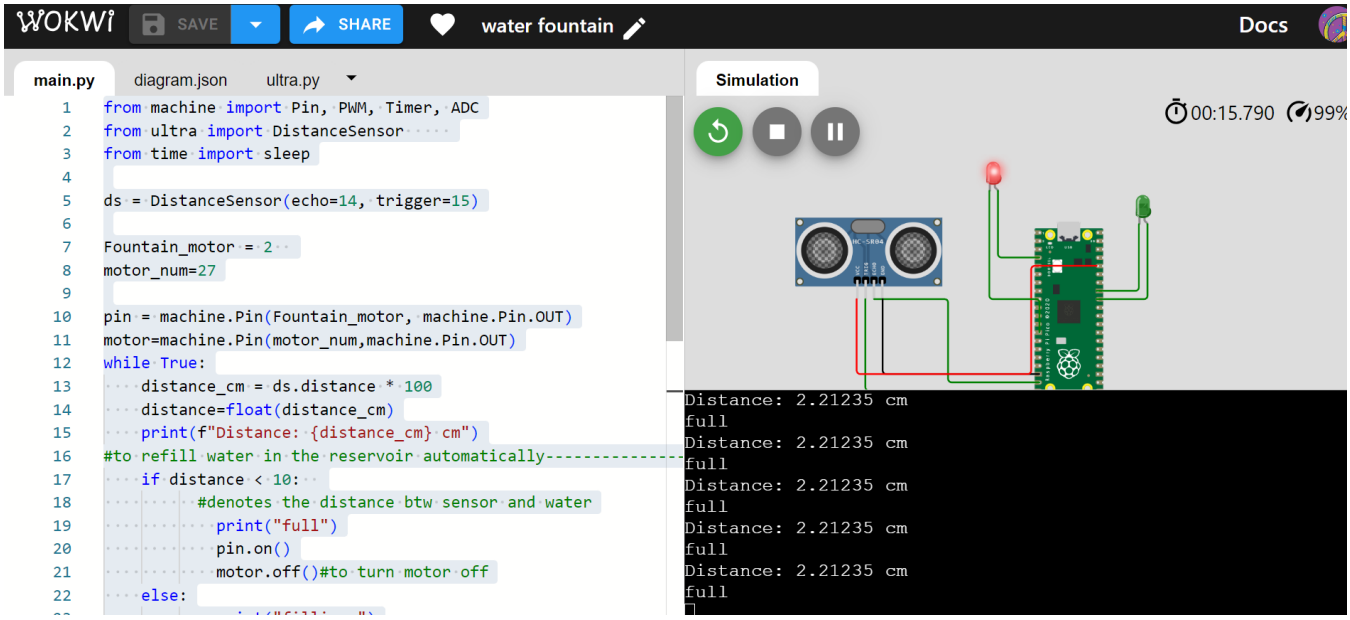
Automation:

We can automate our smart water fountain. For example, we could set schedules for the fountain to turn on and off automatically or create triggers based on environmental conditions.

Enclosure and Maintenance:

Ensure that all components are housed in a waterproof enclosure to protect them from water damage. Regularly check and maintain the fountain, including cleaning the pump and sensors.

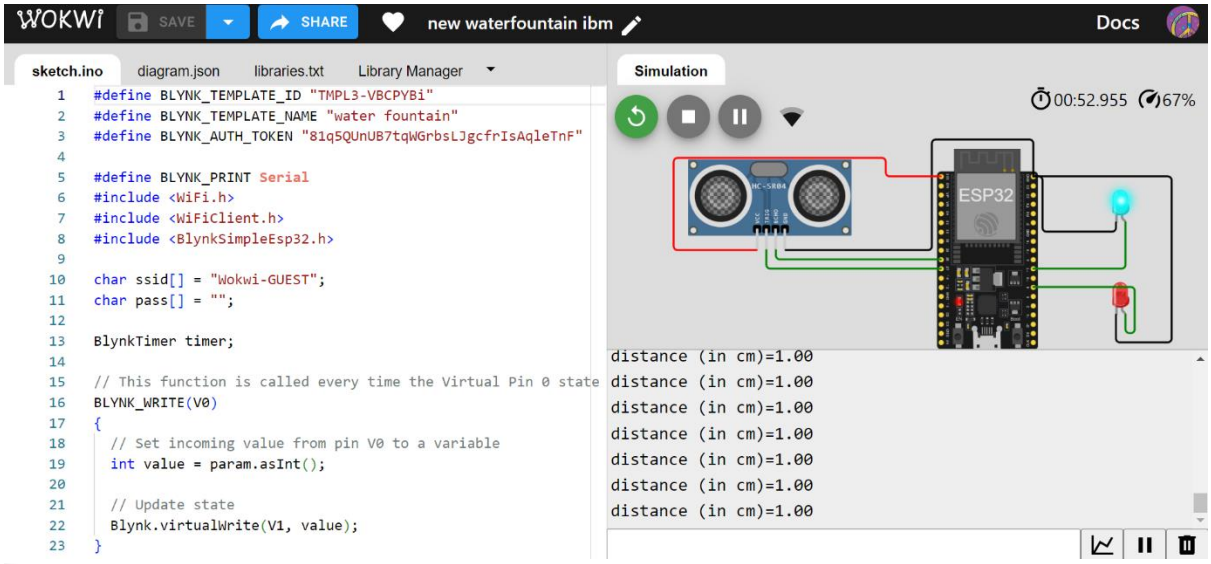
Simulation Using Raspberry pi

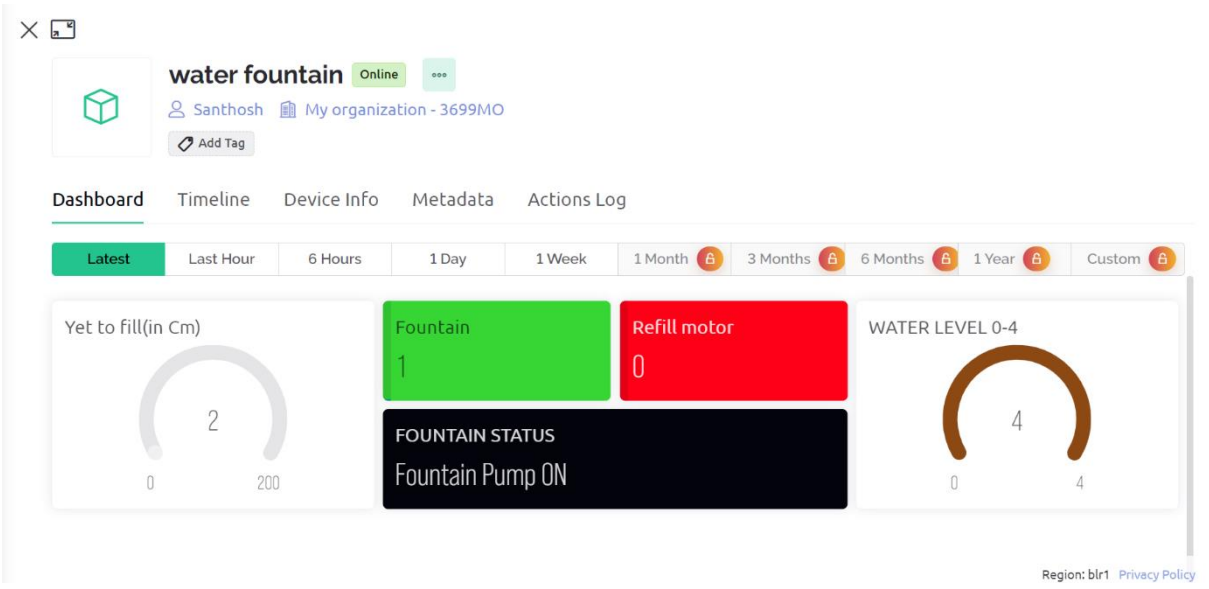


The red LED show the Fountain water pump running, So when reservoir is full the fountain will work

The Green LED shows the source pump to reservoir when the distance between water and the sensor increases the pump will start and fill the reservoir automatically

Using C/C++ & BLYNK

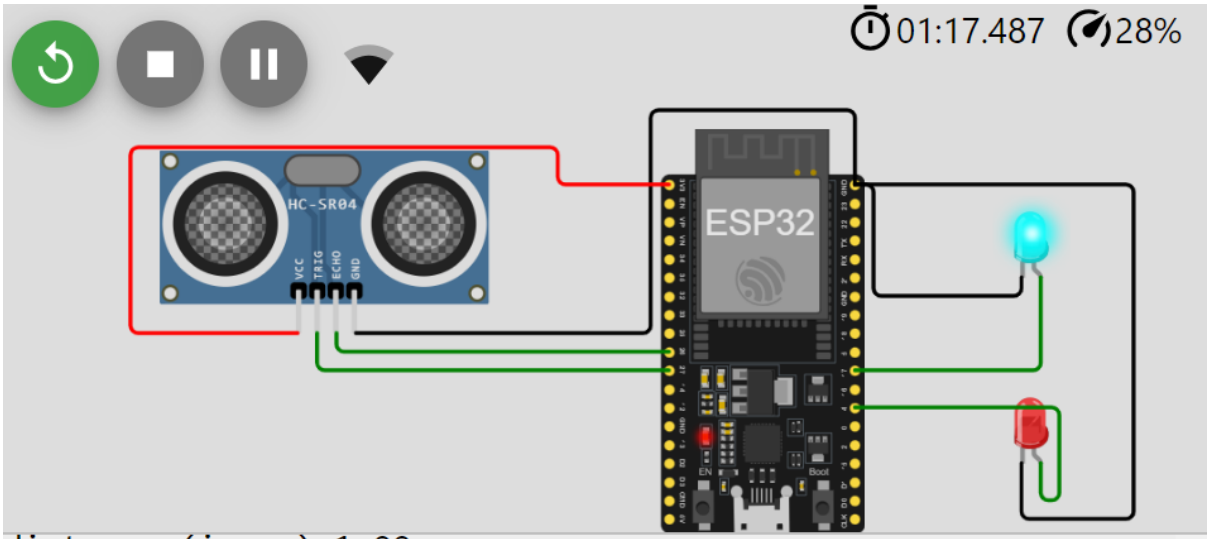




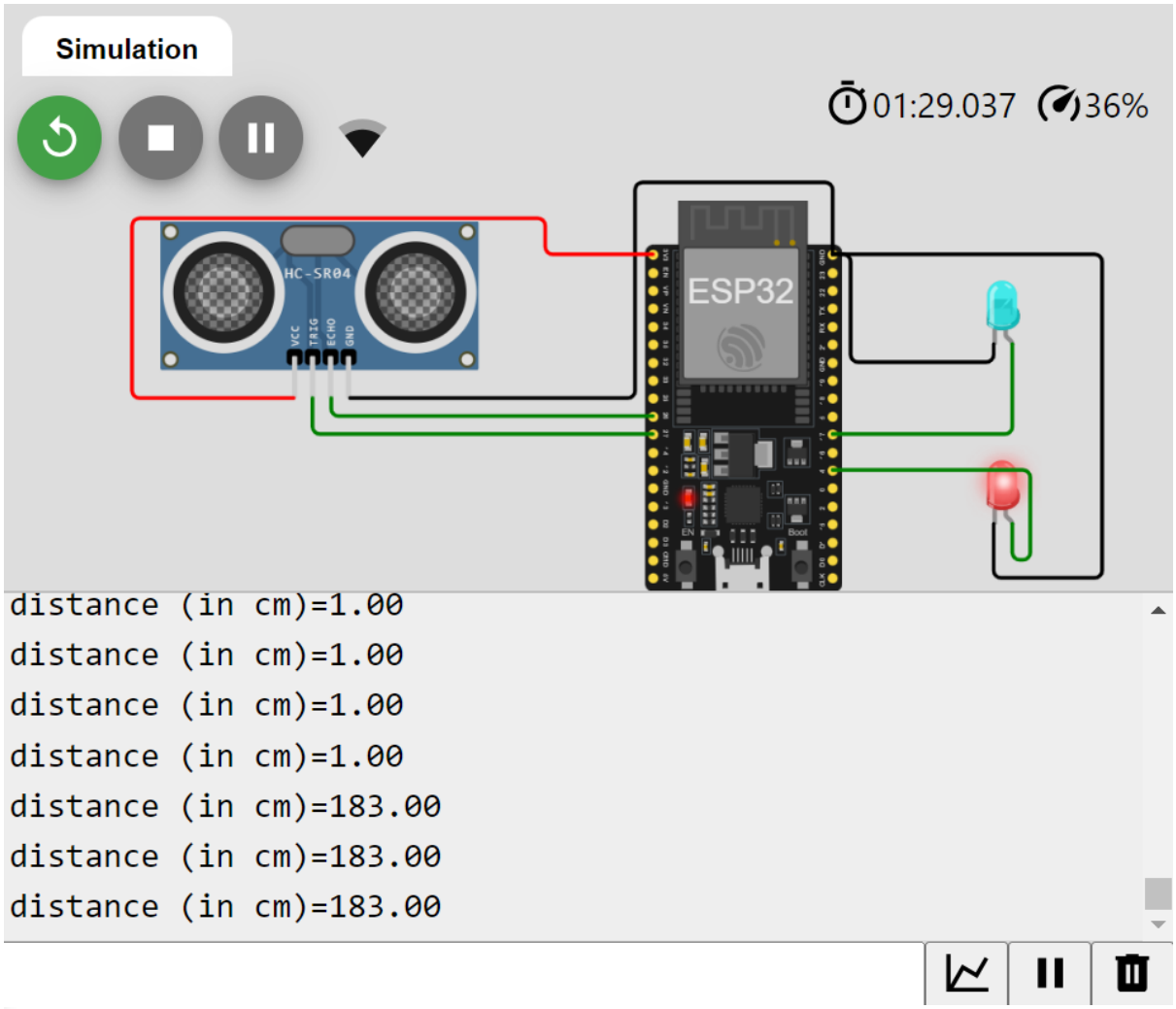
The Blue LED indicated the functioning of the Fountain and the Red LED indicates the functioning of refill motor which fills the reservoir.

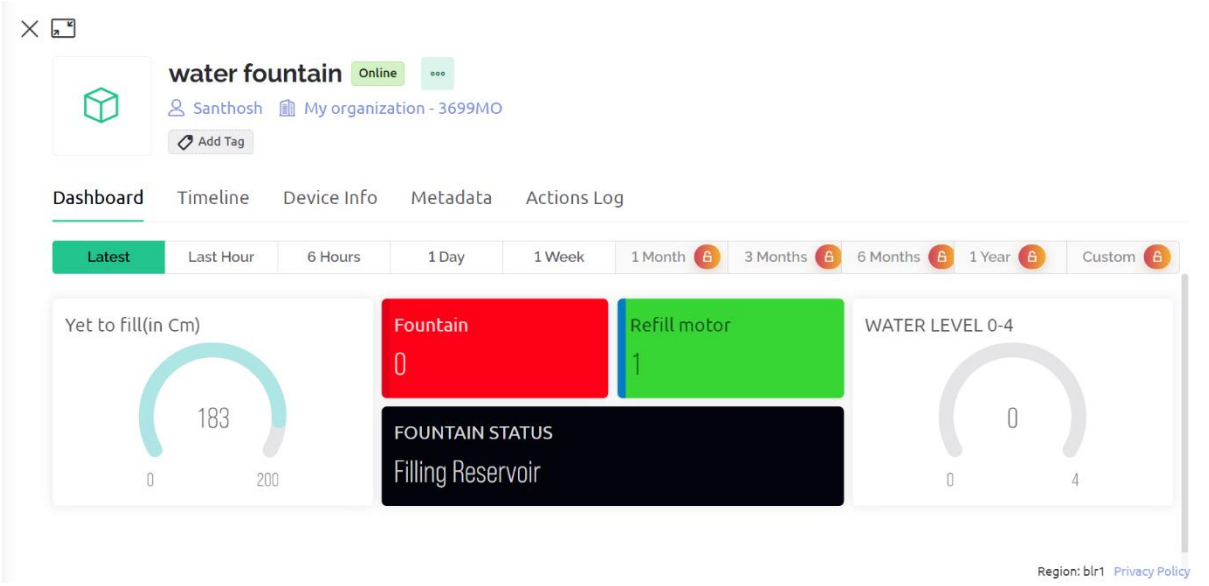
BLYNK

We have used Blynk app to get live data from the Smart water Fountain using Api.




```
distance (in cm)=1.00
distance (in cm)=1.00
distance (in cm)=1.00
distance (in cm)=1.00
distance (in cm)=1.00
distance (in cm)=1.00
distance (in cm)=1.00
```





The Above processes can be monitored and controlled using BLYNK

6. Conclusion

In this document, we have outlined a Internet of Things (IoT) project focused on optimizing water usage in smart water fountains to promote water conservation and responsible resource usage. By following a design thinking approach, water management authorities and environmentalists can leverage automation, ultimately leading to efficient water usage, conservation, and a more sustainable future.