Smart Water Fountains

Development Part-2

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

Table of Contents

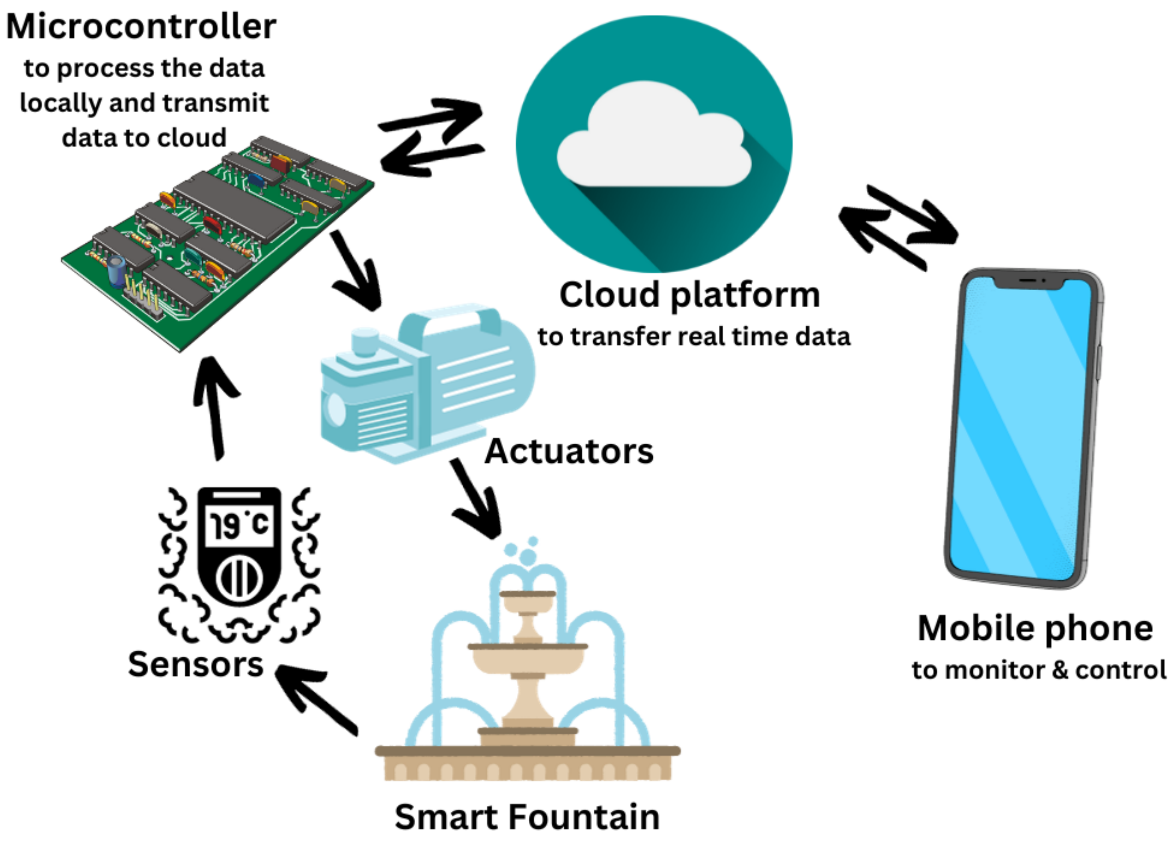
1. Introduction

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

Language Used: Micro Python

MicroPython is a compact and efficient Python programming language implementation designed for microcontrollers and embedded systems. When integrated with the Raspberry Pi Pico, a microcontroller board equipped with the RP2040 microcontroller chip, it forms a user-friendly platform for IoT, robotics, and embedded projects. The Raspberry Pi Pico's impressive capabilities, including its dual ARM Cortex-M0+ cores and ample GPIO pins, make it an accessible choice for developers of all skill levels. With MicroPython support, programming the Pico becomes straightforward, offering a familiar environment for Python enthusiasts. Its compact size, affordability, and low power consumption make it a preferred choice for educational, hobbyist, and commercial projects, catering to a wide range of applications.

Code

```
from machine import Pin
from ultra import DistanceSensor
# ultra is used to get distance from ultrasonic sensor
import time
import network
import urequests

# Configure your Wi-Fi network credentials
WIFI_SSID = "Santhosh"
#your wifi name
WIFI_PASS = "123456789"
#Your wifi password

# ThingSpeak API endpoint and API key
THINGSPEAK_API_KEY = "25885269632562"

THINGSPEAK_URL = "https://api.thingspeak.com/update?api_key=25885269632562"

# Set up the Wi-Fi connection
wifi = network.WLAN(network.STA_IF)

wifi.active(True)

wifi.connect(WIFI_SSID, WIFI_PASS)

# Wait for Wi-Fi connection
while not wifi.isconnected():

    pass

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

Fountain_motor = 2

motor_num = 27

pin = Pin(Fountain_motor, Pin.OUT)

motor = Pin(motor_num, Pin.OUT)

while True:

    distance_cm = ds.distance * 100

    distance = float(distance_cm)
    print(f"Distance: {distance_cm} cm")

    # Send the distance data to ThingSpeak
    try:
        response = urequests.get(THINGSPEAK_URL + "&field1=" + str(distance))

        if response.status_code == 200:

            print("Data sent to ThingSpeak successfully")
        response.close()

    except Exception as e:

        print("Failed to send data to ThingSpeak:", str(e))

    # To refill water in the reservoir automatically

    if distance < 10:
        print("Reservoir is full")
        pin.on()
        motor.off()  # Turn motor off

    else:
        print("Filling reservoir")
        pin.off()
        motor.on()  # Turn motor on

    time.sleep(300)  # Delay for 5 minutes (adjust as needed)


#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/379540245462038528 -------
------------------------
```

5. Working Principle

Control and Monitor:

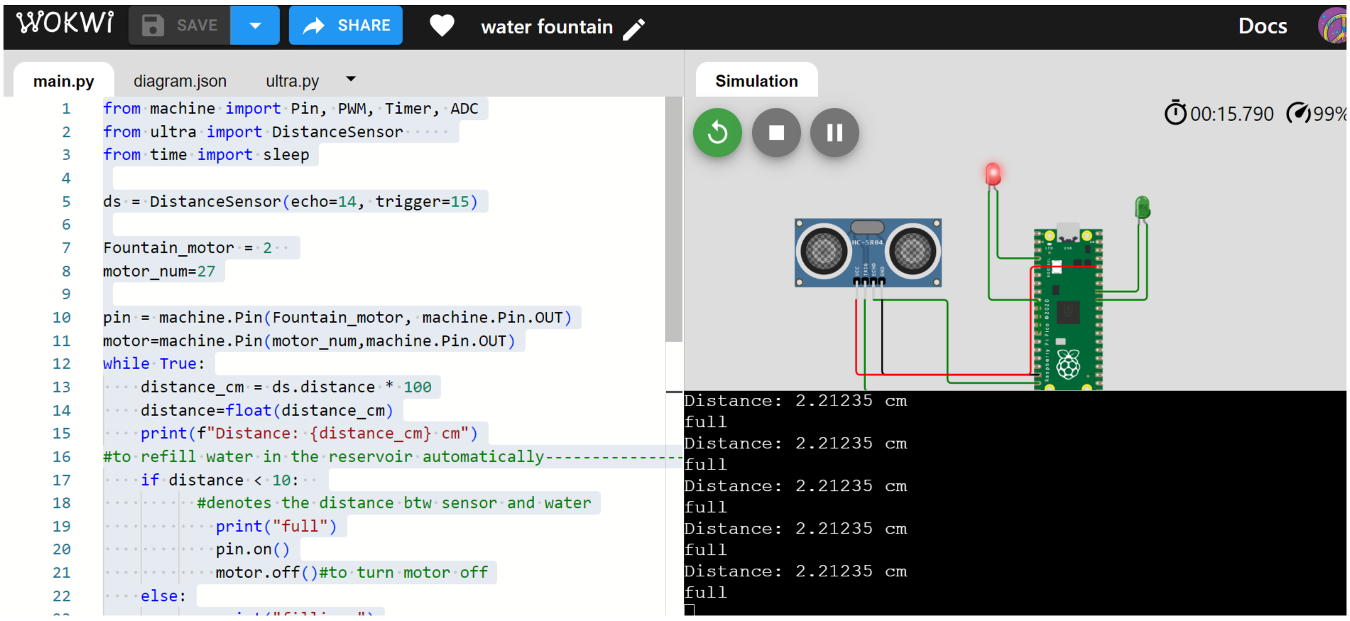
With the software in place, We can now control and monitor your smart water fountain from a smartphone or computer. we can turn the fountain on/off remotely, check water levels, and receive alerts if water levels drop too low or if there are temperature issues.

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:

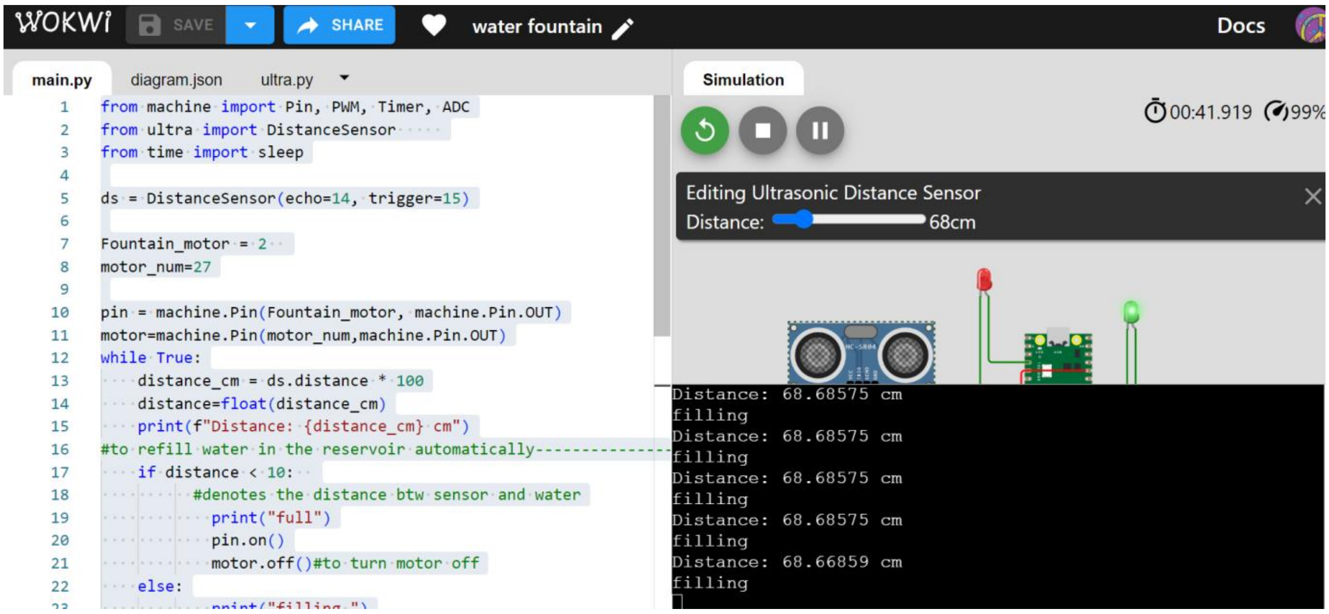
Ensure that all components are housed in a waterproof enclosure to protect them from water damage. Regularly check and maintain the fountain, including cleaning the pump and sensors.



```python
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:
```

```
Distance: 2.21235 cm
full
Distance: 2.21235 cm
full
Distance: 2.21235 cm
full
Distance: 2.21235 cm
full
Distance: 2.21235 cm
full
```

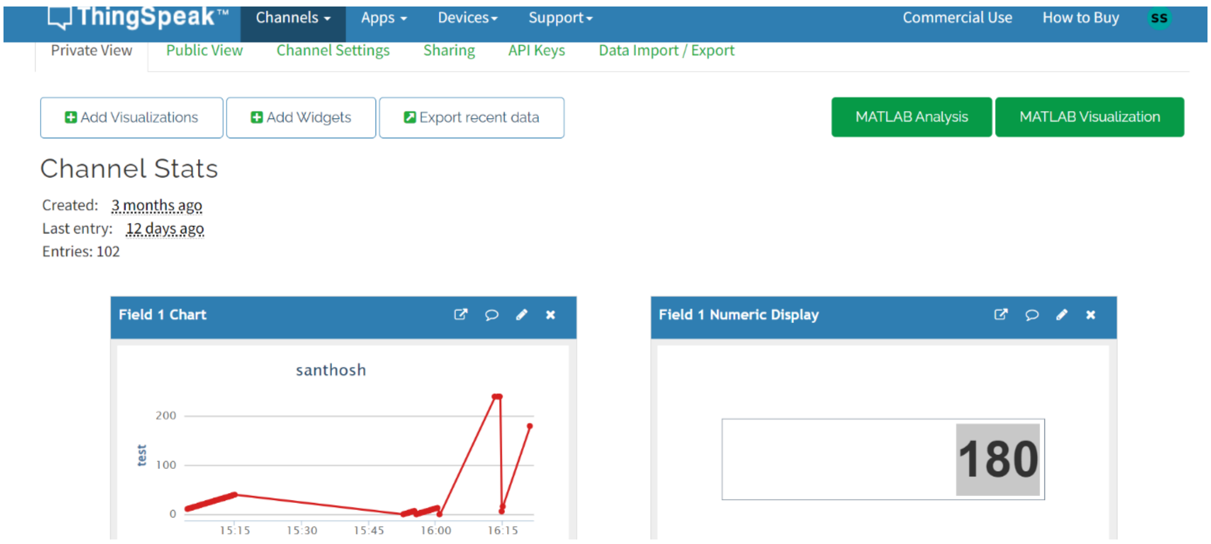
The red LED show the Fountain water pump running, So when reservoir is full the fountain will work



```python
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")
```

```
Distance: 68.68575 cm
filling
Distance: 68.68575 cm
filling
Distance: 68.68575 cm
filling
Distance: 68.68575 cm
filling
Distance: 68.66859 cm
filling
```

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



Channel Stats

Created: 3 months ago
Last entry: 12 days ago
Entries: 102

The Above processes can be monitored and controlled using thingSpeak

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.