SMART WATER FOUNTAIN

PHASE 5

OBJECTIVE :

The objective of this project is to create a Smart Water Fountains system that leverages Internet of Things (IoT) technology to monitor and manage water resources efficiently. The project aims to reduce water wastage, prevent leaks, and ensure the sustainable use of water in urban and rural environments.

IOT DEVICE SETUP:

The project involves the deployment of various IoT devices for data collection, control, and communication. Here are the key components of the IoT device setup:

2. PLATFORM DEVELOPEMENT:

The project's platform development involves creating a cloud-based system for data analysis, visualization, and control. The platform consists of the following components:

Cloud Data Storage: Data collected from the IoT devices is stored in a cloud database for realtime and historical analysis. Services like AWS, Azure, or Google Cloud can be used for this purpose.

Data Analysis and Prediction: Machine learning models are developed to analyze the data and predict water quality, consumption trends, and leak detection. These models help in making informed decisions.

Control System: The platform can send commands back to the IoT devices to control water flow, shut off supply in case of emergencies, or trigger maintenance alerts.

Dashboard: User-friendly dashboards are created for water authorities, environmental agencies, and consumers to access real-time information about water quality, consumption, and alerts.

Code Implementation:

The code for this project will be written in various programming languages, depending on the component:

IoT Device Code: Each IoT device has its own code to read sensor data and transmit it to the IoT gateway. This code may be written in languages like C, Python, or platforms like Arduino.

IoT Gateway Code: The gateway has code to aggregate data, perform preprocessing, and securely transmit it to the cloud. Communication protocols like MQTT or HTTP may be used.

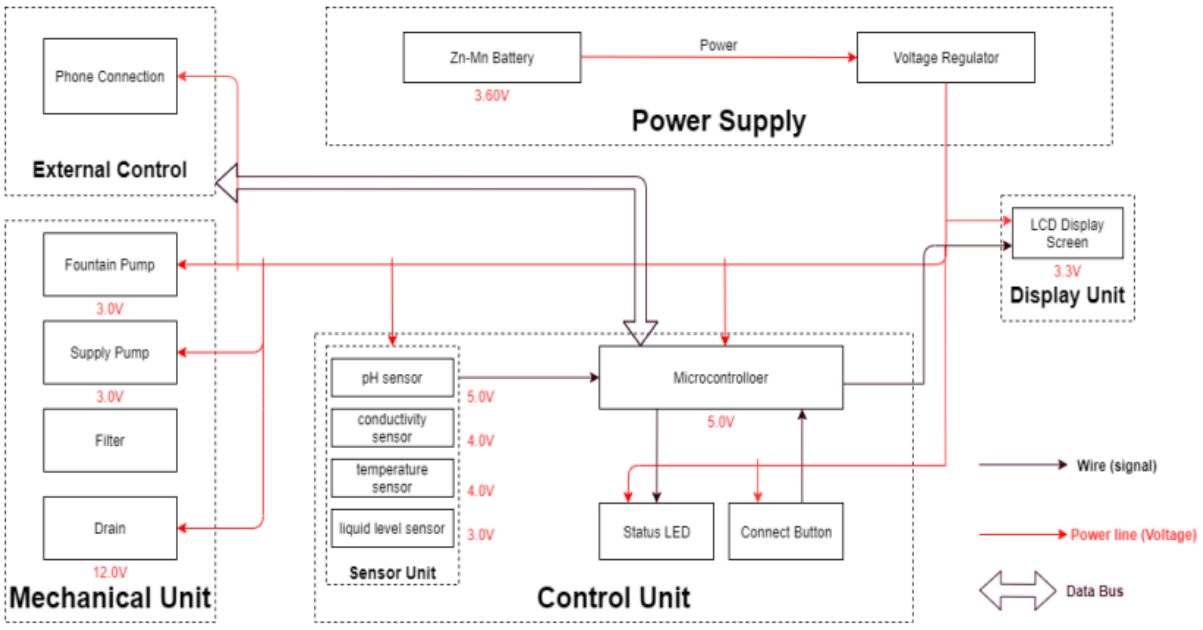
Cloud Data Storage Code: Setting up and managing databases can be done using cloud providers' services and APIs.

Data Analysis and Prediction Code: Machine learning models are implemented using Python, along with libraries like TensorFlow, scikit-learn, or specific water quality analysis tools.

Dashboard Code: Web-based dashboards can be developed using HTML, CSS, and JavaScript, along with frameworks like React or Angular.

COMPONENTS:

* ESP 8266 OR ESP32
* 1K 0.25WATT RESISTORS – 8 NO (R1 – R8)
* RELAY
* LED 5MM – 7NO
* ULTRASONIC SENSORS
* TEMPERATURE AND HUMIDITY SENSORS



DESIGN FEATURES :

1.FOUNTAINS WIRELESSLY COMMUNICATE WITH BASE STATIONS .

2.BASE STATIONS COLLECT AND TRANSMIT USAGE, FILTER, AND SYSTEM HEALTH INFORMATION TO THE CLOUD VIA ETHERNET .

3.WIRELESS COMMUNICATIONS USE A LOW-POWER UNLICENSED BAND FOR IMPROVED SECURITY AND POWER SAVINGS.

4.THIS PROJECT IS DONE BY USING ESP32, RELAY, ULTRASONIC SENSORS, TEMPERATURE AND HUMIDITY SENSORS, LEDS.

5. AN APP WHICH CONTROL AND MONITOR THE FUNCTION OF FOUNTAIN FROM ANY WHERE IN THE WORLD.

6.STORE THE MONITORED VALUES IN FIREBASE REALTIME DATABASE.

CODE :

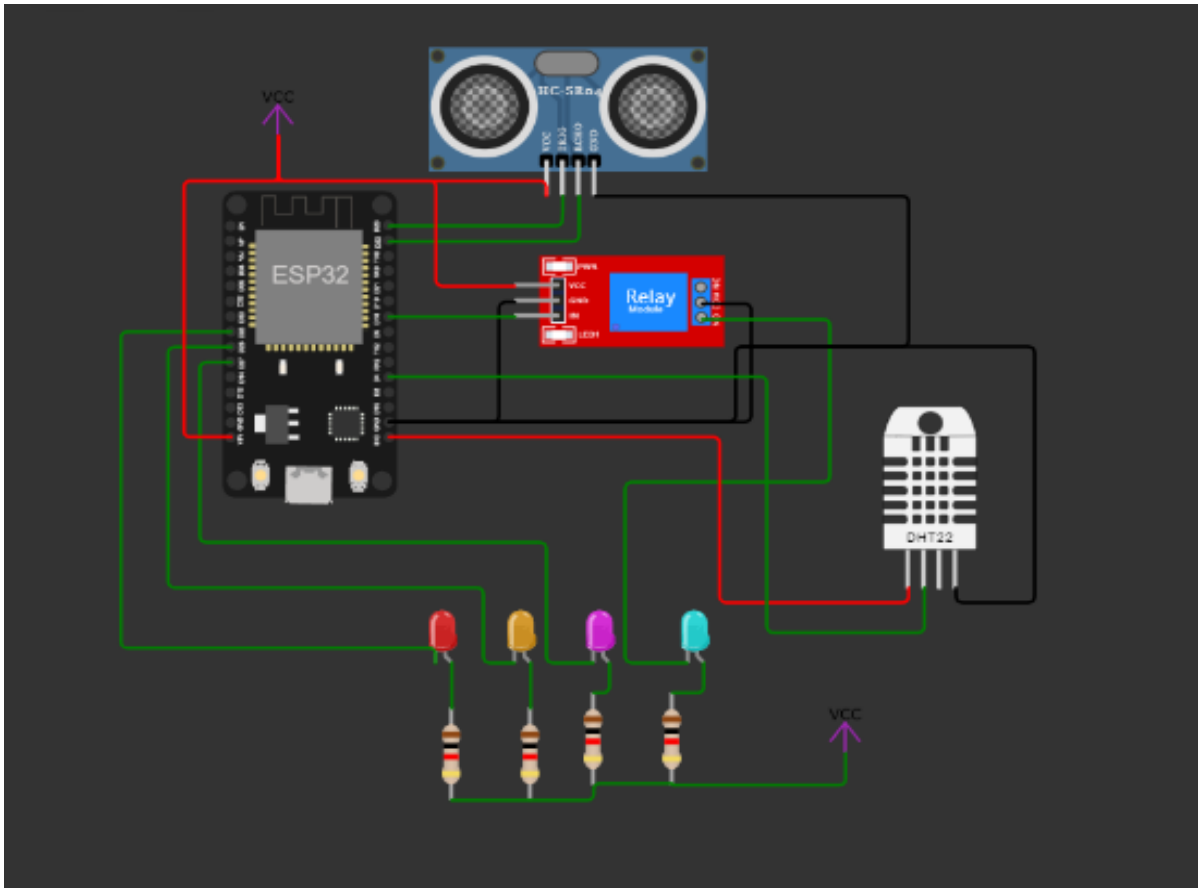
```
import machine
import network
from hcsr04 import HCSR04
from machine import Pin
import ure as re
import usocket as socket
import time
from dht import DHT22

# Set up Wi-Fi
SSID = "YourSSID"
PASSWORD = "YourPassword"
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(SSID, PASSWORD)

# Define HC-SR04 pins
trig_pin = Pin(23, Pin.OUT)
echo_pin = Pin(22, Pin.IN)

# Define Relay Module and LED pins
relay_pin = Pin(18, Pin.OUT)
```

```
led_pins = [Pin(25, Pin.OUT), Pin(26, Pin.OUT), Pin(27, Pin.OUT)]

# Initialize the HC-SR04 sensor
sensor = HCSR04(trigger_pin=trig_pin, echo_pin=echo_pin)

# Initialize the DHT22 sensor
dht_pin = Pin(4, Pin.IN)
dht_sensor = DHT22(dht_pin)

# Web server
def handle_request(client):
    request = client.recv(1024).decode('utf-8')
    if 'GET /on' in request:
        relay_pin.on()
    elif 'GET /off' in request:
        relay_pin.off()
    distance = sensor.distance_cm()
    water_level = "High" if distance < 10 else "Low"
    dht_sensor.measure()
    temperature = dht_sensor.temperature()
    humidity = dht_sensor.humidity()
    response = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n"
    response += f"<html><body><h1>Water Level and Temperature/Humidity
Monitoring</h1>"
    response += f"<p>Distance: {distance} cm</p>"
```

```
    response += f"<p>Water Level: {water_level}</p>"
    response += f"<p>Temperature: {temperature} °C</p>"
    response += f"<p>Humidity: {humidity} %</p>"
    response += "<p><a href='/on'>Turn Pump On</a></p>"
    response += "<p><a href='/off'>Turn Pump Off</a></p>"
    response += "</body></html>"
    client.send(response)
    client.close()

def run_server():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('', 80))
    s.listen(5)

    while True:
        client, addr = s.accept()
        handle_request(client)

# Main loop
while True:
    distance = sensor.distance_cm()
    dht_sensor.measure()
    temperature = dht_sensor.temperature()
    humidity = dht_sensor.humidity()
```

```
print("Distance:", distance, "cm")
print("Temperature:", temperature, "°C")
print("Humidity:", humidity, "%")

# Control the water pump based on distance
if distance < 10:
    relay_pin.on()
else:
    relay_pin.off()

# Indicate water level using LEDs
if distance < 10:
    for i in range(3):
        led_pins[i].on()
else:
    for i in range(3):
        led_pins[i].off()

# Run the web server
run_server()

# Delay for a while to avoid excessive measurements
time.sleep(2)
```

OUTPUT :



OUTPUT :

```
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_d
rv: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
```

```
Connecting to Wokwi-GUEST
..................................
WiFi connected
IP address:
10.10.0.2
Distance in CM: 108
Distance in inches: 42
Attempting MQTT connection...Connected
Temperature: 24.00
Humidity: 40.0
Distance in CM: 108
Distance in inches: 42
Distance in CM: 108
Distance in inches: 42
Temperature: 24.00
Humidity: 40.0
Distance in CM: 108
Distance in inches: 42
Distance in CM: 108
Distance in inches: 42
Temperature: 24.00
Humidity: 40.0
Distance in CM: 108
Distance in inches: 42
Distance in CM: 108
Distance in inches: 42
Temperature: 24.00
Humidity: 40.0
Distance in CM: 108
Distance in inches: 42
Distance in CM: 108
Distance in inches: 42
Temperature: 24.00
Humidity: 40.0
Distance in CM: 108
Distance in inches: 42
Distance in CM: 108
Distance in inches: 42
Temperature: 24.00
Humidity: 40.0
Distance in CM: 108
Distance in inches: 42
Distance in CM: 108
Distance in inches: 42
```

Code for Running for Tinkerkad

```
# Import the necessary library

import tinkercad

import requests

# Add the requests library for HTTP requests

# Initialize the Tinkercad API

 tc = tinkercad.Tinkercad(username="vignesh03", password="vignesh") # Find the simulation you want to run (use the correct ID)

 simulation_id = "2303774"

 # Get the simulation

simulation = tc.get_simulation(simulation_id)

 # Define the ThingSpeak API parameters

 thingspeak_api_key = "G5YN4PEKH3VEQ0JA"

thingspeak_url = f"https://api.thingspeak.com/update?api_key={thingspeak_api_key}"

 # Define the code to run in the Arduino

 arduino_code = """

#include Ultrasonic ultrasonic(2, 3);

// Trigger (pin 2), Echo (pin 3)

void setup()

{

 Serial.begin(9600);

 }

 void loop() {

 float distance = ultrasonic.Ranging(CM);
```

```
 Serial.println(distance); // Send data to the computer (Python script)
Serial.print("D:");
 Serial.println(distance);
delay(1000);
} """ # Upload and run the code in the simulation
simulation.run_code(arduino_code) # Monitor the water level and send data to ThingSpeak
 while True: data = simulation.get_serial_data()
 if data and data.startswith("D:"):
distance = float(data[2:])
print(f"Water level: {distance} cm") # Send data to ThingSpeak
 try: response = requests.get(f"{thingspeak_url}&field1={distance}")
if response.status_code == 200:
        print("Data sent to ThingSpeak successfully.")
 else:
        print("Failed to send data to ThingSpeak.")
except Exception as e:
        print("Error sending data to ThingSpeak:", str(e))
```

Thingspeak Channel stats of Smart Water Fountains:

DATA VISUALIZATION :

- **Real-time dashboards** – These are interactive interfaces that display real-time data from IoT devices in a visual format. They can be customized to show key performance indicators (KPIs), trends, and alerts, allowing users to quickly identify and respond to issues as they arise.
- **Heat maps and choropleths** – These are geographic visualizations that use color-coded maps to display IoT data in specific regions or areas. Heat maps show the concentration or density of IoT data, while choropleths display IoT data on a per-capita or per-unit area basis.
- **Graphs and charts** – These are traditional visualization techniques that use graphs and charts to display trends, patterns, and relationships in IoT data. Common types of graphs and charts used in IoT Visualization include line graphs, bar charts, and pie charts.
- **3D visualizations** – These are immersive visualizations that use 3D models and virtual reality (VR) technologies to provide a more realistic and interactive view of IoT data. They are useful for visualizing complex and detailed IoT data in a way that is more easily understandable.

- **Maps and geospatial visualizations** – These are visualizations that use maps and geospatial data to display IoT data in a spatial context. They can be used to analyze data from IoT devices that are distributed across large geographic areas, such as weather sensors or traffic monitoring devices.

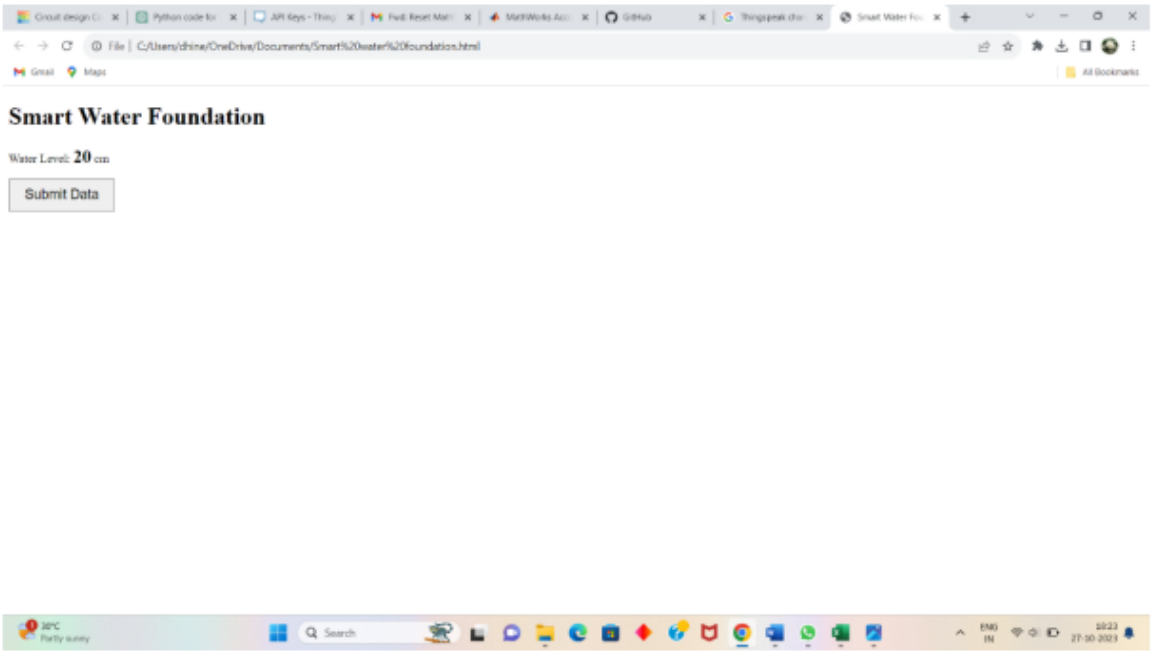
Platform UI code for Smart Water Fountains:

```
<!DOCTYPE html>
<html>
<head>
<title>Smart Water Fountains</title>
<style>
 /* Style for the water level display */
 #water-level {
 font-size: 24px;
 font-weight: bold;
 }
 /* Style for the submit button */
 #submit-button {
 padding: 10px 20px;
 font-size: 18px;
 }
</style>
</head>
<body>
<h1>Smart Water Fountains</h1>
```

```
<p>Water Level: <span id="water-level">Loading...</span> cm</p>
<button id="submit-button" onclick="sendDataToServer()">Submit Data</button>
<script>
 // Function to update water level data from the server
 function updateWaterLevel() {
 // You can use AJAX or fetch to get data from your server
 // Replace the URL with the actual endpoint that provides water level data
 fetch('/getWaterLevelData')
.then(response =>response.json())
.then(data => {
document.getElementById('water-level').textContent = data.waterLevel + " cm";
 })
.catch(error => {
console.error('Error fetching water level data:', error);
 });
 }
 // Function to send data to the server (e.g., to trigger data collection)
 function sendDataToServer() {
 // You can use AJAX or fetch to send data to your server
 // Replace the URL with the actual endpoint that handles data submission
fetch('/submitData', { method: 'POST' })
.then(response => {
 if (response.status === 200) {
console.log('Data submitted successfully');
```

```
 } else {
console.error('Data submission failed with status:', response.status);
 }
 })
.catch(error => {
console.error('Error submitting data:', error);
 });
 }
 // Update water level initially and then at regular intervals
updateWaterLevel();
setInterval(updateWaterLevel, 10000); // Update every 10 seconds
</script>
</body>
</html>
```

Output for above Program:



DEVICES AND WORKING

Temperature Sensor: A water-proof temperature sensor is going to be used. Part number from sparkfun is: DS18B20 [6]. This temperature sensor is compatible with a relatively wide range of power supply from 3.0V to 5.5V. The measured temperature ranges from -55 to +125 celsius degrees. Between -10 to + 85 degrees, the accuracy is up to +-0.5 degrees. This sensor can fulfill all requirements needed for this project.

PH-sensor: PH value is a valued indicator of water quality. This PH-sensor[7] works with 5V voltage, which is also compatible with the temperature sensor. It can 6measure the PH value from 0 to 14 with an accuracy of +- 0.1 at the temperature of 25 degrees.

Conductivity sensor: Conductivity sensor is also part of the water quality assessment. The input voltage is from 3.0 to 5.0V. The error is small, +-5%F.S. The measurement value ranges from 0 to 20 ms/cm which is enough for water quality monitoring.

Ultrasonic Sensor: This sensor [9] is responsible for reflecting how much freshwater is left in the water tank. When the water level is low, fresh water will be pumped to the water tank to ensure the water fountain keeps running with freshwater. This sensor is 0.5 Watts. For water level from 0 to 9 inches, the corresponding sensor outputs readings from 0 to 1.6. From that, the quantity of freshwater left can be determined.

Display unit: The screen will be used to display the readings from the sensors in a real-time manner.[10] In addition, other necessary information will also be displayed. As described in the sensor part, the water quality and remaining water quantity will be displayed. The screen will be programmed so that it makes it easy for users to read information. This 20*4 LCD display screen is going to be used to display the relevant information. After programming the screen, a conclusion of water quality(Good, Average, Poor) will be displayed along with the remaining water level.

Power Supply Unit : Zn-Mn Battery The Zn-Mn battery must be able to continuously support the functioning of the circuit, display unit, and the mechanical unit. Requirement: Commercial batteries will be used to maintain a continuous 3.60V power supply for at least 24 hours. If the chosen battery is not powerful enough, 120V power outlets will be considered.

Voltage regulator :The integrated circuit will regulate the power supply for each module to maintain their functionality. This chip must be able to handle the maximum voltage supplied by the battery (3.60V ± 0.5V) while ensuring the voltage at each module does not exceed their limit. Requirement: Must maintain thermal stability below 100°C.

MIT APP INVENTOR:

Our project codes

Our project model

In this application (using MIT app inventer software) we can control the circuit while wireless communication by mobile phone in any where in the world. This monitors the Temperature and humidity and also monitor the level of the water in a fountain.

Conclusion:

Keeping the record of usage of water using this experiment will help in solving the water deficiency problem. This paper represents the model design and scope of the SMART WATER FOUNTAIN [FLOW RATES] for solving the water deficiency problem.

The volume of water was successfully calculated using an ultrasonic sensor which is used to calculate the variable height of water through the pipe. A GSM module sent the signal once the threshold volume was reached in the form of SMS. The proposed system helps every house and society to reduce the wastage of water significantly and hence the water scarcity

Future Scope :

The proposed system can solve almost all the problems faced by the existing water meters. There is also scope for adding a feature which will automatically limit the flow of water into particular house or public fountain For this feature a solenoid valve can be installed inside water flow meter so that when it rotates inside the meter then flow of water can be minimized.

References :

020 IEEE Pune Section International Conference (PuneCon) | 978-1-7281-9600-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/PuneCon50868.2020.9362468

Ejiofor V., Oladipo O., Microcontroller based automatic water level control system, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 1, Issue 6, August 2013, 1390-1396.

Kumura T., Suzuki N., Takahashi M., Tominaga S., Morioka S., Ivan S., Smart water management technology with intelligent sensing and ICT for the integrated water systems, NEC Technical Journal, Vol. 9, No. 1, January, 2015, 103-106.

Meng Hua,Wang Hui, High-Precision Flow Measurement for an Ultra-sonic Transit Time Flowmeter, 2010 International Conference on Intelli-gent System Design and Engineering Application,13 Oct 2010,Changsha ,China.

Fan Shunjie,ZhouYeu, A multi-frequency ultrasonic flowmeter applicable to liquid with gas bubbles 2011 IEEE International Instrumentation and Measurement Technology Conference,10 May 2011,Binjiang ,China.