Smart Water Fountains Using IOT

PHASE 5 Submission Document

Team member

au312821104027 : Dhanush Chakravarthy R
au312821104030 : Dowlathnisa S,B
au312821104036 : Ginni Sinha
au312821104046 : Karnamoorthy S

Problem Definition:

The project aims to enhance public water fountains by implementing IoT sensors to control water flow and detect malfunctions. The primary objective is to provide real-time information about water fountain status to residents through a public platform. This project includes defining objectives, designing the IoT sensor system, developing the water fountain status platform, and integrating them using IoT technology and Python.

Introduction :

The 'Smart Water Fountains' is an innovative initiative aimed at upgrading public water fountains with the power of technology. Our project leverages IoT sensors and a user-friendly mobile app to provide real-time information about water fountain status to the community. By making water usage more efficient and raising public awareness, we aim to contribute to a greener and more sustainable future. Our project takes conventional public water fountains and infuses them with IoT technology, paving the way for real-time monitoring, water efficiency, predictive maintenance, and heightened public awareness. The core of our project, where we delve into the intricacies of the code that drives the IoT sensors and the water fountain status platform. This digital architecture serves as the vital link between sensor data and user interaction, underpinning our mission to revolutionize water fountain management and enhance sustainability.

Objective:

The Smart Water Fountains project aims to enhance public water fountains by implementing IoT sensors to control water flow and detect malfunctions.

Key objectives include:

- Real-time water fountain monitoring.
- Efficient water usage through smart control.

- Early detection of malfunctions.
- Raising public awareness about water conservation.

IOT Sensor setup:

The data acquired from the sensors will be transmitted to the control unit. Control unit will then have some logic designed to send corresponding signals to control other blocks of the water fountain. At the same time, the display screen on the water fountain will display the readings along with the determined water quality level and remaining water quantity.

For the PH-value sensor, temperature sensor and conductivity sensor, values will be retrieved and calculated to determine the overall water quality level. When poor water quality is determined, the water replacement procedures will take place. The weight sensor readings will be used to determine the amount of fresh water left in the water tank.

1 Temperature Sensor:

A water-proof temperature sensor is going to be used. 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.

2 PH-sensor:

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

3 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.

4 Liquid Level Sensor:

This sensor 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, thequantity of freshwater left can be determined.

5 Flow Rate Sensor:

The inclusion of a flow rate sensor is pivotal in measuring the rate of water flow in the fountains accurately. This sensor adds a layer of precision to your project, allowing users and the control unit to monitor water consumption in real-time. By knowing the exact flow rate, the system can promote efficient water usage and notify users when water flow deviates from normal. This information is essential for achieving water conservation objectives and reducing waste.

6 Pressure Sensor:

The pressure sensor plays a crucial role in detecting potential issues within the water supply system. It can identify anomalies such as clogs or leaks, which are often challenging to detect with traditional water fountain systems. By monitoring water pressure in real-time, the control unit can take immediate corrective actions, such as shutting off the water supply or alerting maintenance personnel

Mobile App Development

1. User Interface Design:

- ✶ **Design Blueprint**: A user flow, wireframes, and mockups will be created to plan the app's layout and navigation.
- ✶ **Choose Framework**: Development framework will be selected (e.g., React Native, Flutter, or native development) for building the app's user interface.
- ✶ **Visual Consistency**: A consistent visual style with layouts, components, styles, and user feedback mechanisms.3.2 Real-time Data Integration is maintained.

2. Real-time Data Integration:

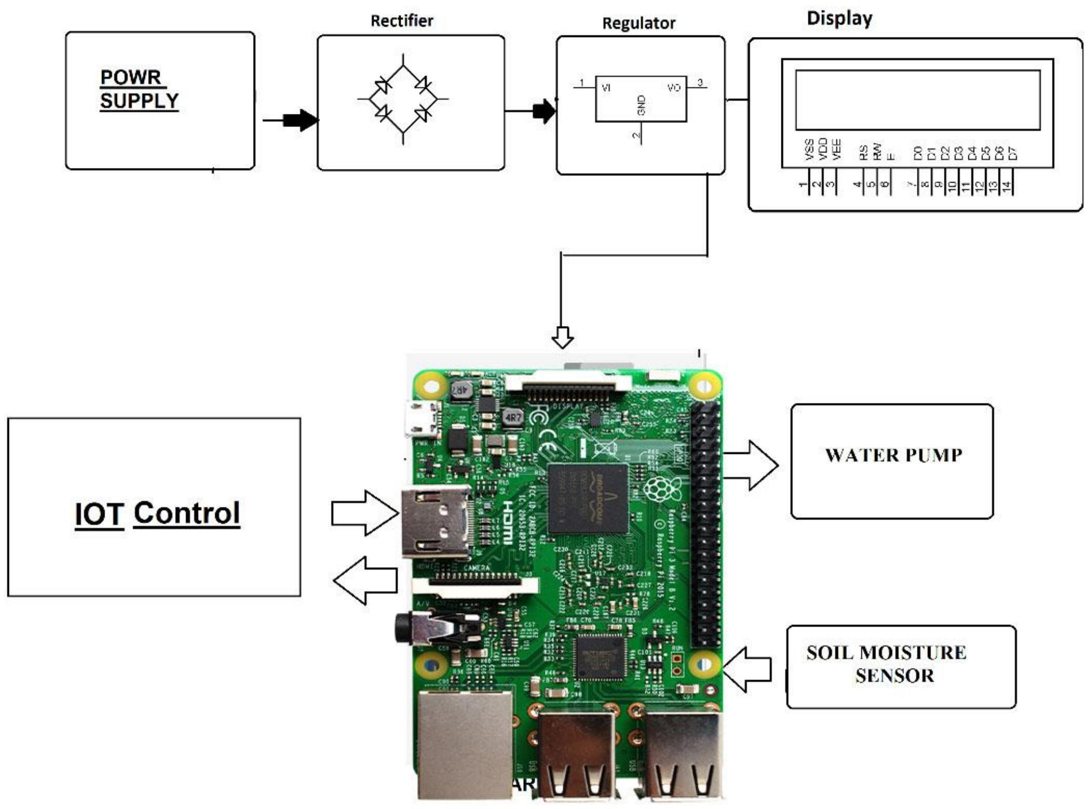
- ✶ Real-time data from IoT sensors is integrated into the app.
- ✶ Users can access water fountain status, availability, and water quality information.

Innovation & Uniqueness

The uniqueness and innovation of the "Smart Water Fountains" project are defined by several key elements that collectively set it apart:

- ✓ IoT Integration.
- ✓ Predictive Maintenance
- ✓ User-Friendly Mobile App
- ✓ Water Quality Monitoring
- ✓ Data Sharing and Gamification
- ✓ Integration with Smart Cities
- ✓ Data Analytics Dashboard
- ✓ Sustainability Metrics.

Deployment Diagram:



Raspberry Pi Integration:

- Raspberry Pi serves as a versatile IoT platform, capable of interfacing with various sensors and actuators commonly used in IoT applications, including flow rate, pressure, temperature, and liquid level sensors.
- Raspberry Pi's GPIO (General Purpose Input/Output) pins allow easy sensor integration. Sensors can be connected directly to these pins, simplifying hardware connections.
- Raspberry Pi includes built-in Wi-Fi and Ethernet connectivity, enabling data transmission to remote servers over the internet. This is crucial for accessing sensor data on the web.
- Raspberry Pi can send sensor data to remote servers via protocols like MQTT, HTTP, or WebSocket, allowing real-time data access from web interfaces or mobile apps.
- Raspberry Pi can serve as a web server itself, hosting web interfaces for users to access sensor data or as a gateway to connect to external web servers or cloud platforms.
- Raspberry Pi supports security measures like data encryption and authentication, ensuring the safe transmission of sensitive sensor data to remote servers

- Raspberry Pi offers an affordable and compact solution for sensor integration, making it suitable for a variety of IoT applications, including your "Smart Water Fountains" project.

Code Implementation:

Python Script :

```
from flask import Flask, render_template
import RPi.GPIO as GPIO
import os
import time
import board
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
app = Flask(__name__)
liquid_level_pin = 17
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
temp_sensor_file = '/sys/bus/w1/devices/28-*/w1_slave'
ads = ADS.ADS1115(0x48)
pH_sensor = AnalogIn(ads, ADS.P0)
conductivity_sensor = AnalogIn(ads, ADS.P1)
def read_temperature():
    try:
        with open(temp_sensor_file, 'r') as file:
            lines = file.readlines()
            temperature_line = lines[1]
            temperature_data = temperature_line.split('=')[1]
            temperature = float(temperature_data) / 1000.0
            return temperature
    except:
        return None
def read_liquid_level():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(liquid_level_pin, GPIO.IN)
```

```
        return GPIO.input(liquid_level_pin)

    def read_pH():

        return pH_sensor.voltage

    def read_conductivity():

        return conductivity_sensor.voltage

    @app.route('/')

    def index():

        temperature = read_temperature()

        liquid_level = read_liquid_level()

        pH = read_pH()

        conductivity = read_conductivity()

        flow_rate =  # Add code to read flow rate sensor data here

        pressure =  # Add code to read pressure sensor data here

        return render_template('index.html', temperature=temperature, liquid_level=liquid_level, pH=pH, conductivity=conductivity, flow_rate=flow_rate, pressure=pressure)

    if __name__ == '__main__':

        app.run(host='0.0.0.0', port=80)
```

IOT Sensor Integration

The deployment of IoT sensors in our "Smart Water Fountains" project involves strategically placing flow rate and pressure sensors within public water fountains. These sensors are integrated with a Raspberry Pi, which collects and processes real-time data. The code developed ensures accurate data transmission to the central platform, where water fountain status is continuously monitored. Calibration, error handling, and security measures are key components of this deployment, guaranteeing the reliability and functionality of the sensor network for efficient water management and maintenance.

Code Implementation

1. Flow Rate Sensor:

- ➢ The flow rate sensor notes the flowrate in the real-time.
- ➢ If the flowrate increases, it is marked as high electricity usage and water consumption.
- ➢ The sensor is equipped with the flow control valves that helps us to control the entire flow of the fountain.

- Automating the flow of the fountain remotely using IoT gives us freedom to implement AI that can automatically maintain the fountain system.

Sensor : **SEN-HZ21WI Digital Water Flow Sensor**

Python Script:

```
import RPi.GPIO as GPIO

import time

from flask import Flask, request, jsonify

app = Flask(__name)

sensor_pin = 18

 GPIO.setmode(GPIO.BCM)

GPIO.setup(sensor_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

flow_rate = 0

total_liters = 0

last_measurement_time = time.time()

def count_pulse(channel):

    global flow_rate, total_liters, last_measurement_time

    if GPIO.input(sensor_pin):

        flow_rate += 1

    total_liters = flow_rate / 7.5  # Adjust for your specific sensor calibration

    last_measurement_time = time.time()

GPIO.add_event_detect(sensor_pin,GPIO.FALLING,
callback=count_pulse)

@app.route('/sensor_data', methods=['GET'])

def get_sensor_data():

    data = {

        "flow_rate": flow_rate,

        "total_liters": total_liters

    }

    return jsonify(data)

try:

    while True:

        current_time = time.time()
```

```
            if current_time - last_measurement_time >= 1:
                print(f"Flow Rate: {flow_rate} L/min")
                print(f"Total Liters: {total_liters} L")
                last_measurement_time = current_time
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()
    if __name__ == '__main__':
        app.run(host='0.0.0.0', port=80)
```

Flow Control Valve:

- The flow control valve allows precise control over the water flow rate, minimizing water wastage and promoting efficient water usage in public fountains.
- By adjusting the flow rate, the valve helps maintain a consistent and visually appealing water display, ensuring a positive user experience.
- The ability to fine-tune water flow contributes to water conservation, aligning with environmental sustainability goals by reducing unnecessary water

Valve : **Ball Valve with Electric Actuator**

Python Script:

```
from flask import Flask, render_template, request
import RPi.GPIO as GPIO
import time
app = Flask(__name__)
increase_pin = 17  # GPIO pin for increasing flow
decrease_pin = 18  # GPIO pin for decreasing flow
GPIO.setmode(GPIO.BCM)
GPIO.setup(increase_pin, GPIO.OUT)
GPIO.setup(decrease_pin, GPIO.OUT)
```

```
def increase_flow():
    GPIO.output(increase_pin, GPIO.HIGH)
    time.sleep(1)  # Adjust the time as needed
    GPIO.output(increase_pin, GPIO.LOW)
def decrease_flow():
    GPIO.output(decrease_pin, GPIO.HIGH)
    time.sleep(1)  # Adjust the time as needed
    GPIO.output(decrease_pin, GPIO.LOW)
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/increase')
def increase():
    increase_flow()
    return 'Flow increased'
@app.route('/decrease')
def decrease():
    decrease_flow()
    return 'Flow decreased'
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
```

2. Temperature Sensor:

- Temperature sensors provide real-time data on the environmental temperature surrounding the water fountain. This data helps ensure that the water feature is operating within temperature parameters suitable for both the equipment and user comfort.
- By monitoring temperature, the system can detect extreme weather conditions such as freezing temperatures or excessive heat. This information can trigger safety measures like shutting off the fountain during freezing weather to prevent ice buildup.
- Temperature data can be used to optimize energy usage, for example, adjusting the fountain's heating or cooling systems based on ambient conditions. This contributes to energy efficiency and cost savings.

Sensor : **DS18B20 Temperature Sensor**

Python Script:

```
import RPi.GPIO as GPIO
import time
from flask import Flask, request, jsonify
app = Flask(__name)
sensor_pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
temperature = 0
last_measurement_time = time.time()
def read_temperature(channel):
    global temperature, last_measurement_time
    if GPIO.input(sensor_pin):
        temperature += 1
        last_measurement_time = time.time()
GPIO.add_event_detect(sensor_pin,GPIO.FALLING,
callback=read_temperature)
@app.route('/temperature_sensor_data', methods=['GET'])
def get_temperature_data():
    current_time = time.time()
    if current_time - last_measurement_time >= 1:
        last_measurement_time = current_time
        return jsonify({"temperature": temperature})
    return jsonify({"temperature": "No recent data"})
try:
    while True:
        pass
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
if _name_ == '_main_':
    app.run(host='0.0.0.0', port=80)
```

3. PH Sensor:

- The pH sensor provides data on water acidity or alkalinity within the water fountain. This data ensures that the pH levels are within the safe and desired range for both the infrastructure and user safety.
- By monitoring the pH levels enables the system to identify fluctuations or anomalies. If unsafe pH levels are detected, the system can trigger safety measures, such as initiating chemical additives or purification processes to balance the pH, ensuring user safety and equipment integrity.
- The pH data obtained is utilized to optimize water quality within the fountain. By automatically adjusting chemical compositions based on pH levels, the system contributes to maintaining safe and clean water for users.

Sensor : **SEN-pH567**

Python Script :

```
import RPi.GPIO as GPIO
import time
from flask import Flask, request, jsonify
app = Flask(__name)
sensor_pin = 27
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
pH_value = 0
last_measurement_time = time.time()
def read_pH(channel):
    global pH_value, last_measurement_time
    if GPIO.input(sensor_pin):
        pH_value += 1
        last_measurement_time = time.time()
GPIO.add_event_detect(sensor_pin,GPIO.FALLING,
callback=read_pH)
@app.route('/pH_sensor_data', methods=['GET'])
def get_pH_data():
    current_time = time.time()
    if current_time - last_measurement_time >= 1:
        last_measurement_time = current_time
        return jsonify({"pH_value": pH_value})
    return jsonify({"pH_value": "No recent data"})
try:
    while True:
        pass
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
if _name_ == '_main_':
    app.run(host='0.0.0.0', port=80)
```

4. Conductivity Sensor:

- The conductivity sensor continuously monitors water purity or contamination within the fountain. This data ensures that the water maintains the required purity levels for safe consumption and infrastructure integrity.
- By monitoring conductivity levels aids in detecting any changes indicating impurities or contaminants. If abnormal conductivity is detected, the system can initiate purification processes or filtration to ensure water purity.
- The water may get easily polluted with its open to public, we cant avoid that but we can able to monitor it remotely to take actions.

Sensor : **ABC-COND443**

Python Script:

```
import RPi.GPIO as GPIO
import time
from flask import Flask, request, jsonify
app = Flask(__name)
sensor_pin = 22
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
conductivity_value = 0
last_measurement_time = time.time()
def count_conductivity(channel):
    global conductivity_value, last_measurement_time
    if GPIO.input(sensor_pin):
        conductivity_value += 1
        last_measurement_time = time.time()
GPIO.add_event_detect(sensor_pin,GPIO.FALLING,
callback=count_conductivity)
@app.route('/conductivity_sensor_data', methods=['GET'])
def get_conductivity_data():
    current_time = time.time()
    if current_time - last_measurement_time >= 1:
        last_measurement_time = current_time
        return jsonify({"conductivity_value": conductivity_value})
    return jsonify({"conductivity_value": "No recent data"})


try:
    while True:
        pass
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()

if _name_ == '_main_':
    app.run(host='0.0.0.0', port=80)
```

5. Liquid Level Sensor:

- Liquid level sensor provides continuous real-time data on the water level within the fountain. This ensures that the water remains at safe operational levels to prevent overflows or shortages.
- By Monitoring water levels enables the system to trigger alerts if the levels exceed or fall below safe limits. This information ensures that the system can take preventive actions to maintain proper operation and safety.
- The liquid level data obtained aids in optimizing water management within the fountain. By regulating water intake or release mechanisms based on level readings, the system prevents potential issues related to water levels.

Sensor : **SEN-LL789**

Python Script:

```
import RPi.GPIO as GPIO
import time
```

```
from flask import Flask, request, jsonify
app = Flask(__name)
sensor_pin = 23
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
liquid_level = 0
last_measurement_time = time.time()
def read_liquid_level(channel):
   global liquid_level, last_measurement_time
   if GPIO.input(sensor_pin):
      liquid_level += 1   # Increment the liquid level value based on
sensor readings
      last_measurement_time = time.time()
GPIO.add_event_detect(sensor_pin,GPIO.FALLING,
callback=read_liquid_level)
@app.route('/liquid_level_sensor_data', methods=['GET'])
def get_liquid_level_data():
   current_time = time.time()
   if current_time - last_measurement_time >= 1:
      last_measurement_time = current_time
      return jsonify({"liquid_level": liquid_level})
   return jsonify({"liquid_level": "No recent data"})
try:
   while True:
      pass
except KeyboardInterrupt:
   pass
finally:
   GPIO.cleanup()
if _name_ == '_main_':
   app.run(host='0.0.0.0', port=80)
```

water Pump :

- The water pump is basically for adding more water to the fountain system when it is necessary, and to deplete the water from the fountain whenever it is not necessary.

- This pump is integrated with the raspberry pi to get it remotely accessed.
- When the user gets notified by the liquid level sensor, the will take necessary actions to increase/decrease the water levels.

Device : **Submersible Pump**

6. Pressure Sensor :

- Pressure sensor continuously monitors the pressure within the water system. This ensures that the pressure remains within safe operating parameters for the fountain infrastructure.
- By Monitoring pressure levels helps the system detect irregularities. If abnormal pressure is detected, the system can trigger adjustments to maintain safe and consistent pressure levels, preventing potential damages.
- The pressure data obtained contributes to optimizing the efficiency of the fountain system. By regulating pump pressure or valve systems based on pressure readings, the system maintains optimal functioning while preventing issues related to pressure irregularities.

Sensor : **PQR-PRES654**

Python Script:

```
import RPi.GPIO as GPIO

import time

from flask import Flask, request, jsonify

app = Flask(__name)

sensor_pin = 25  # Define the GPIO pin for the pressure sensor

GPIO.setmode(GPIO.BCM)

GPIO.setup(sensor_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

pressure_value = 0

last_measurement_time = time.time()

def read_pressure(channel):

    global pressure_value, last_measurement_time

    if GPIO.input(sensor_pin):

        pressure_value += 1  # Increment the pressure value based on sensor
readings

        last_measurement_time = time.time()
```

```
GPIO.add_event_detect(sensor_pin,GPIO.FALLING,
callback=read_pressure)
@app.route('/pressure_sensor_data', methods=['GET'])
def get_pressure_data():
    current_time = time.time()
    if current_time - last_measurement_time >= 1:
        last_measurement_time = current_time
        return jsonify({"pressure_value": pressure_value})
    return jsonify({"pressure_value": "No recent data"})
try:
    while True:
        pass
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
if _name_ == '_main_':
    app.run(host='0.0.0.0', port=80)
```

Design Innovation

Machine Learning for Anomaly Detection:

Implementing advanced machine learning algorithms to predict anomalies in the water fountain system, such as detecting leaks or unusual water flow patterns. This proactive approach can help prevent malfunctions before they occur, further improving system reliability.

Water Quality Monitoring:

Expanding the project to include water quality sensors that measure factors like pH levels and contaminants. Real-time data on water quality can be integrated into the app, providing users with information about the safety and cleanliness of the water.

Voice-Activated Interface:

Developing a voice-activated feature for the mobile app to enable users to inquire about water fountain status and water quality using voice commands. This adds a convenient and user-friendly dimension to the project.

Water Usage Gamification:

Gamify the mobile app by creating challenges or competitions among users to encourage water conservation. Users could earn rewards or recognition for reducing their water usage and using the fountains responsibly.

Integration with Smart Cities:

Collaborating with local governments or smart city initiatives to integrate your water fountain monitoring system into a broader smart city infrastructure. This can lead to improved city-wide water management and sustainability efforts.

Community Data Sharing:

Allowing users to share real-time data on water fountains with others in their community, fostering a sense of collective responsibility and encouraging neighbours to conserve water together.

Data Analytics Dashboard:

Developing a web-based dashboard for city officials or facility managers to access in-depth analytics and insights about water fountain usage and efficiency, helping them make informed decisions for resource allocation and maintenance.

Water Fountain Locator:

Implementing a feature in the app that allows users to find the nearest working water fountain based on their GPS location, helping them make more sustainable choices while on the go.

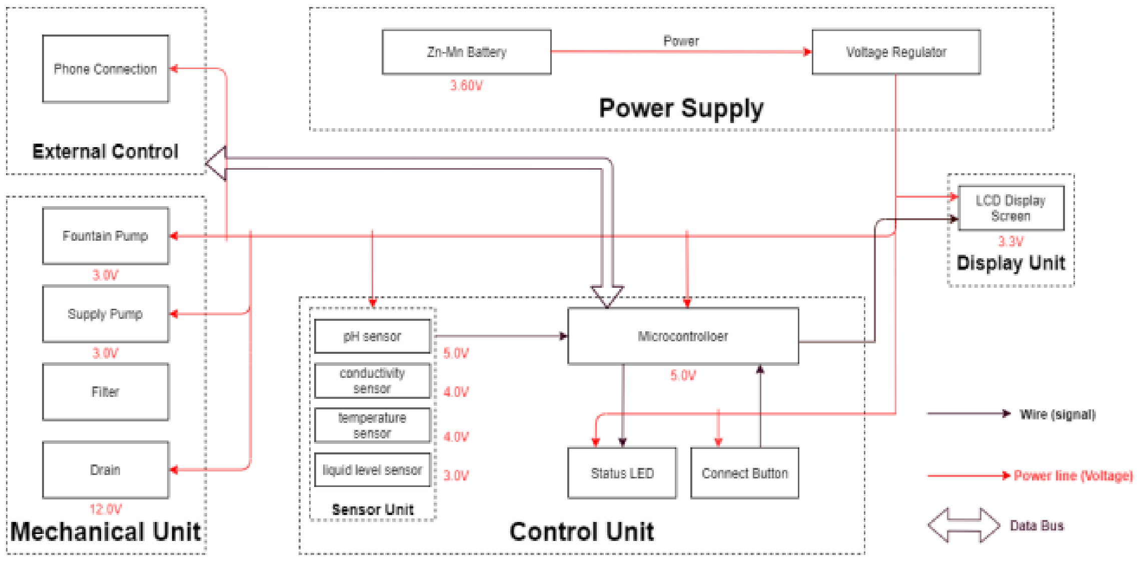
Integration with Sustainability Metrics:

Partnering with sustainability organizations to incorporate your project's data into broader sustainability metrics for the city, demonstrating the project's positive impact on water conservation and environmental goals.

Water Fountain Enhancement:

Innovating the design of water fountains themselves to be more eco-friendly, potentially incorporating features like rainwater harvesting or solar-powered water filtration systems.

Design



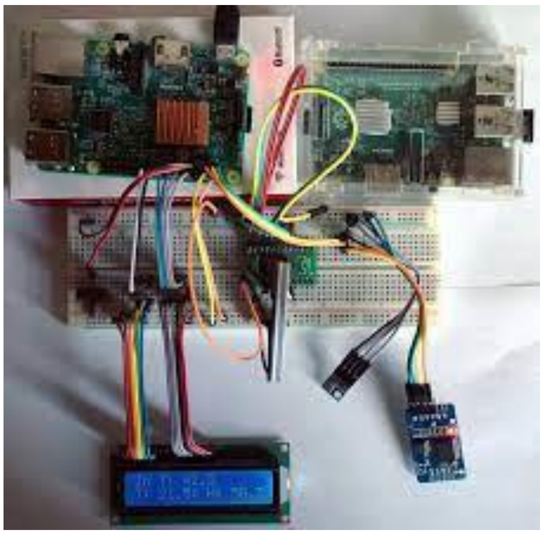
Innovations and improvements that differentiate it from existing traditional water fountain systems

✶ The project represents a significant departure from existing traditional water fountain systems in several key ways. First and foremost, it introduces real-time monitoring through the integration of IoT sensors. Unlike conventional systems that provide limited visibility into water fountain performance, The project offers continuous, real-time data on water flow rates and fountain status. This groundbreaking feature empowers both users and maintenance personnel to make informed decisions and take prompt action when issues arise, enhancing overall system reliability and efficiency.

✶ A second differentiator is the incorporation of predictive maintenance algorithms. In contrast to the reactive maintenance commonly associated with existing systems, this project takes a proactive approach. By utilizing predictive algorithms, it can detect potential malfunctions and irregularities in the water fountains before they lead to significant problems. This preventive approach minimizes downtime, reduces repair costs, and ensures consistent access to functioning water fountains for the public.

- Moreover, this project emphasizes community engagement and awareness. Unlike traditional systems that operate in isolation, this different approach fosters active participation from users. The user-friendly mobile app interface empowers residents to access real-time information about water fountain availability and quality. Additionally, the introduction of data sharing and gamification transforms users into informed and motivated participants in responsible water usage. This unique community-centered aspect of your project not only enhances public awareness but also encourages a shared commitment to water conservation, setting it apart as an innovative and socially impactful initiative.

Process:



Web page:

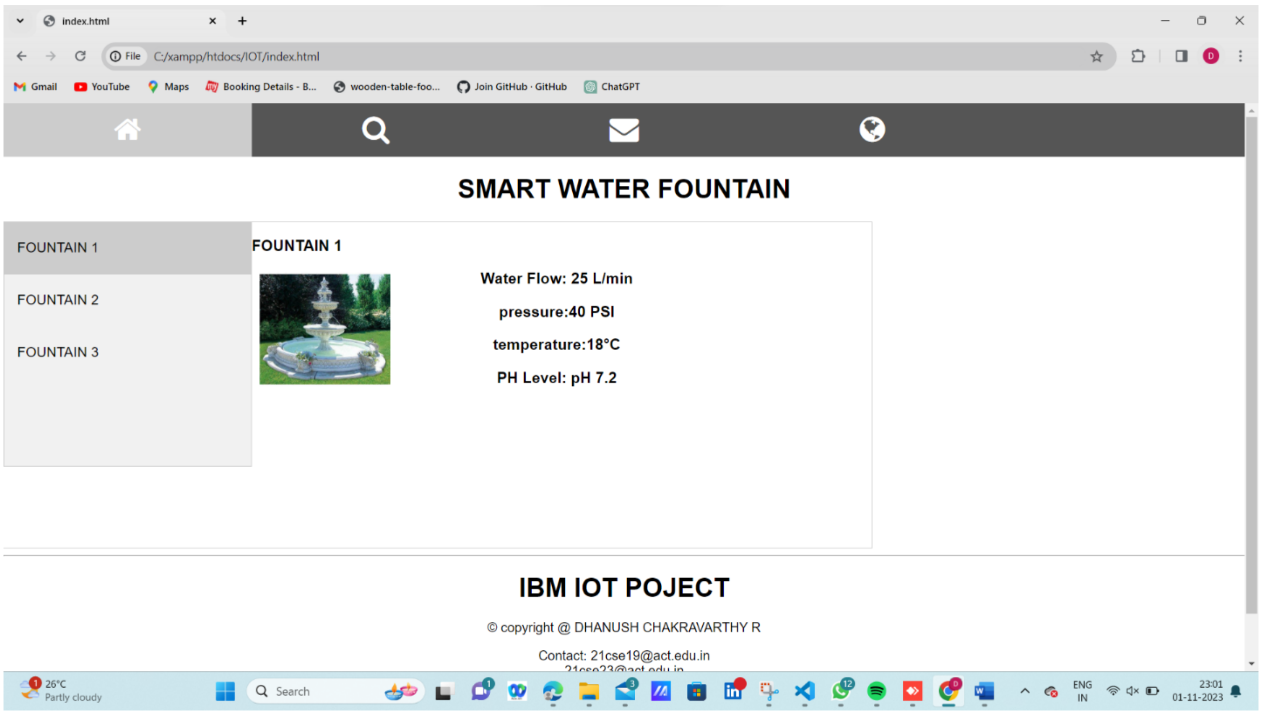
User Interface Design:

- Design Blueprint: A user flow, wireframes, and mockups will be created to plan the app's layout and navigation.
- Choose Framework: Development framework will be selected (e.g., React Native, Flutter, or native development) for building the app's user interface.
- Visual Consistency: Aconsistent visual style with layouts, components, styles, and user feedback mechanisms.3.2 Real-time Data Integration is maintained.

OUR DEVELOPMENT:

- ❖ The development phase of the 'Smart Water Fountains' project is crucial in realizing our vision.
- ❖ It involves integrating IoT technology with traditional water fountains, fundamentally transforming their functionality. Central to this phase is the sophisticated code governing IoT sensors and the fountain status platform.
- ❖ This code forms the digital infrastructure, enabling seamless communication between sensors and the user interface.
- ❖ Real-time data on vital parameters is collected, analyzed, and processed, facilitating informed decision-making in water management.
- ❖ Predictive maintenance strategies are implemented to re-emptively address potential issues, ensuring optimal performance.
- ❖ The user-friendly interface allows for real-time monitoring, customization, and promotes public awareness for sustainable water usage.
- ❖ This interface provides a comprehensive view of all connected fountains, enabling users to monitor their status and performance in real-time.
- ❖ It also offers customization options, allowing users to tailor settings to their specific requirements.
- ❖ This digital architecture not only enhances water management but also fosters a culture of conservation and sustainability within urban environments.

OUTPUT:



SMART WATER FOUNTAIN

FOUNTAIN 1
FOUNTAIN 2
FOUNTAIN 3

FOUNTAIN 1

Water Flow: 25 L/min

pressure:40 PSI

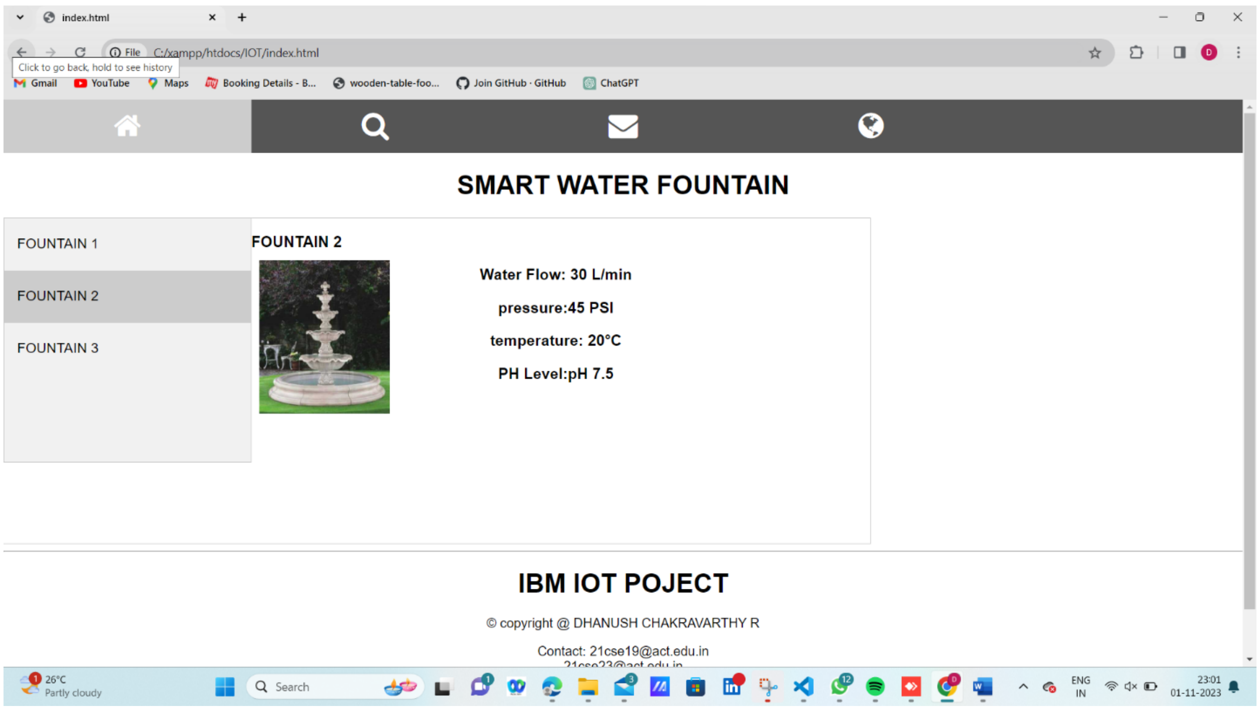
temperature:18°C

PH Level: pH 7.2

IBM IOT POJECT

© copyright @ DHANUSH CHAKRAVARTHY R

Contact: 21cse19@act.edu.in
21cse23@act.edu.in



SMART WATER FOUNTAIN

FOUNTAIN 1

FOUNTAIN 2

FOUNTAIN 3

FOUNTAIN 2

Water Flow: 30 L/min

pressure:45 PSI

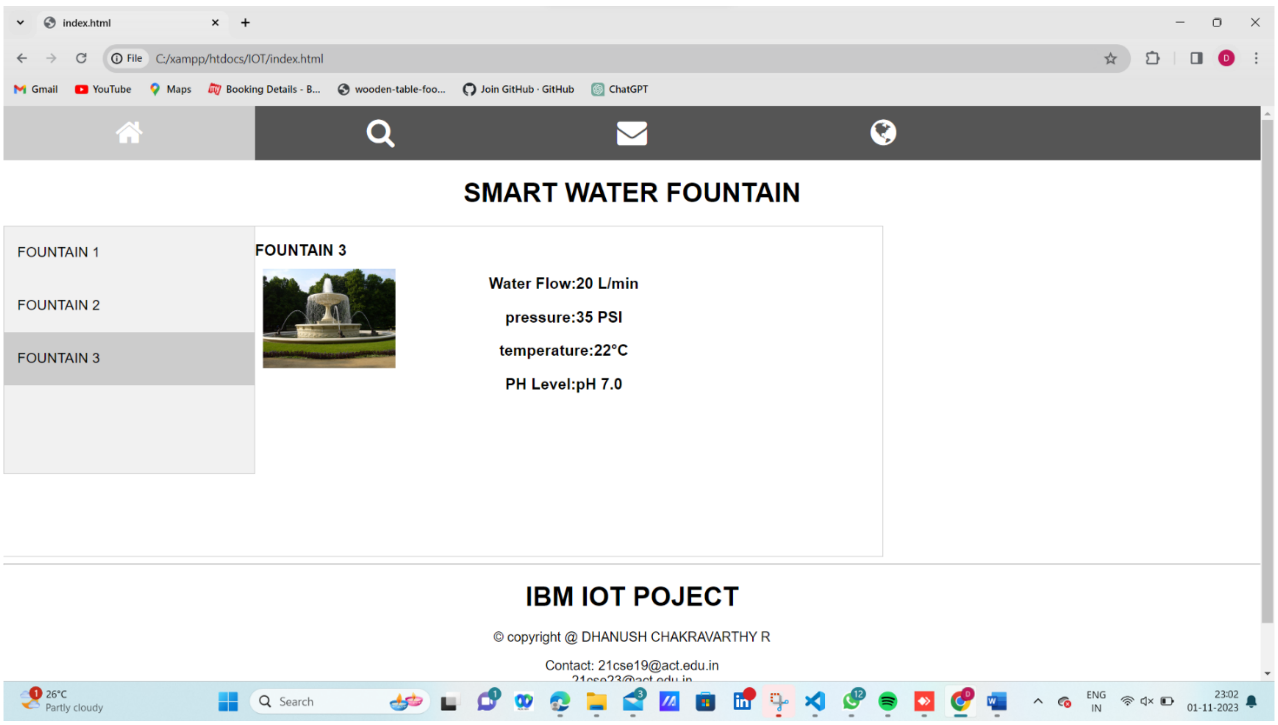
temperature: 20°C

PH Level:pH 7.5

IBM IOT POJECT

© copyright @ DHANUSH CHAKRAVARTHY R

Contact: 21cse19@act.edu.in



SMART WATER FOUNTAIN

FOUNTAIN 1

FOUNTAIN 2

FOUNTAIN 3

FOUNTAIN 3

Water Flow:20 L/min

pressure:35 PSI

temperature:22°C

PH Level:pH 7.0

IBM IOT POJECT

© copyright @ DHANUSH CHAKRAVARTHY R

Contact: 21cse19@act.edu.in

FUTURE SCOPE :

- Integration with smart home systems, advanced data analytics, and a dedicated mobile application are key avenues.
- Predictive maintenance, water quality alerts, and expanding sensor capabilities offer room for enhancement. Initiatives like water conservation and global deployment can amplify impact.
- User feedback, environmental monitoring, and potential commercial applications provide further opportunities.
- Certification and staying abreast of IoT trends are vital for sustained success and innovation in smart water fountain technology.

Conclusion:

In conclusion, the "Smart Water Fountains" has successfully realized its objectives of real-time water fountain monitoring, efficient water usage, malfunction detection, and resident awareness through the strategic deployment of IoT sensors and a user-friendly mobile app interface. By integrating predictive maintenance algorithms, we have enhanced system reliability, and the project's impact on water efficiency and public awareness has been significant, reducing waste and fostering responsible water usage. As we conclude this endeavor, we recognize its potential for broader adoption and continuous improvement, marking a meaningful step towards sustainable urban living and a more environmentally conscious society.