KGiSL Institute of Technology

(Affiliated to ANNA University, Chennai and Approved by AICTE, New Delhi)

365, KGiSL Campus, Thudiyalur Road, Saravanampatti Coimbatore – 641035

Department of Artificial Intelligence and Data Science

Name	**: Isaac.S**
Register Number	**: 711721243035**
Regulation	**: R-2021**
Branch	**: B.Tech -Artificial Intelligence and Data Science**
Project Title	**: Smart Water Fountain**
Semester/ Year	**: V / III**

Project Title: Smart Water Fountains

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.

Project Objectives:

The objectives of the smart water fountain project are to:

- Develop a low-cost and easy-to-install smart water fountain system that can monitor and manage water usage in real time.
- Promote water efficiency and public awareness by providing real-time data on water fountain usage to users through a mobile app.
- Integrate the smart water fountain system with existing IoT infrastructure to enable remote monitoring and control.

<u>Design Innovation:</u>

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.

Components:

Hardware Requirements:

IoT Sensors:

Flow rate sensors: These sensors measure the rate of water flow in the fountains.

Pressure sensors: Pressure sensors can help detect issues like clogs or leaks in the water supply.

Raspberry Pi-compatible sensors: Ensure that the sensors you choose are compatible with Raspberry Pi GPIO pins and communication protocols.

Raspberry Pi:

Raspberry Pi 3 or 4: These are commonly used for IoT projects and offer GPIO pins for sensor connections.

MicroSD card: For the Raspberry Pi's operating system and storage of data and scripts.

Power supply: Adequate power supply for the Raspberry Pi to ensure uninterrupted operation.

Mobile Devices:

Smartphones or tablets for testing and using the mobile app.

Internet Connection:

A stable internet connection for data transmission between the Raspberry Pi and the mobile app.

Wiring and Connectors:

Wires, connectors, and breadboards for connecting sensors to the Raspberry Pi.

Enclosures:

Weatherproof enclosures for protecting sensors and Raspberry Pi when deployed at outdoor water fountains.

Software Requirements:

Raspberry Pi Operating System:

Raspbian or Raspberry Pi OS installed on the Raspberry Pi.

Programming Language:

Python: You will likely use Python to write scripts for data collection, processing, and transmission on the Raspberry Pi.

IoT Communication Protocol:

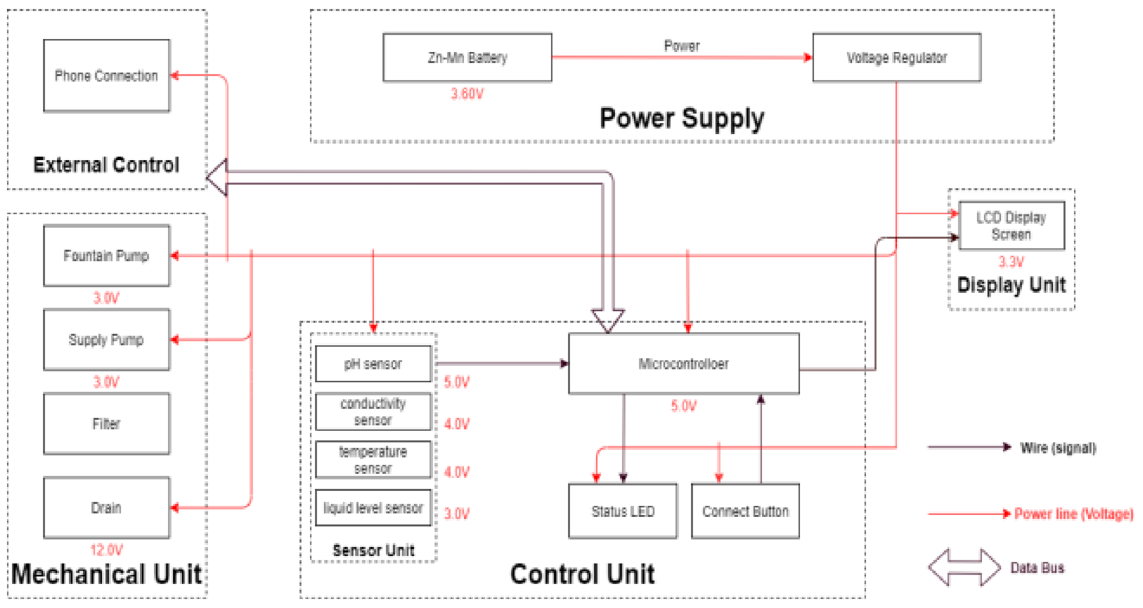
MQTT (Message Queuing Telemetry Transport) or another suitable protocol for data transmission between sensors and the Raspberry Pi.

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.

Design:



Sensor Unit

This block contains the four sensors. 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.

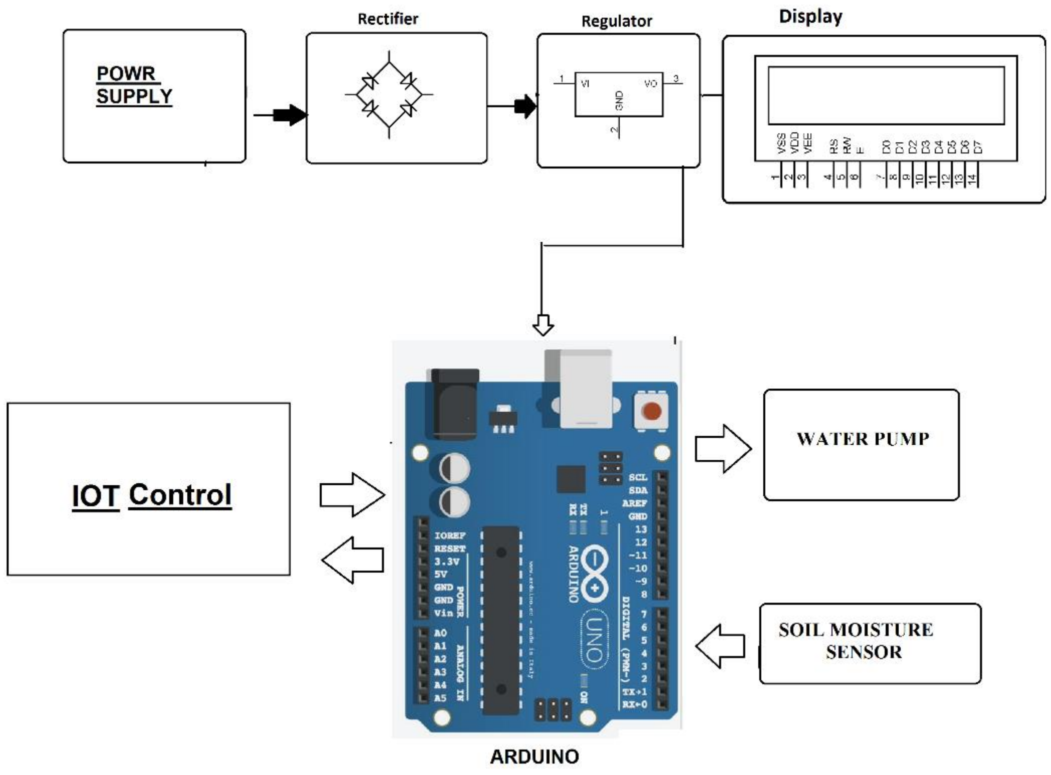
5 Flow Rate Sensor:

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.

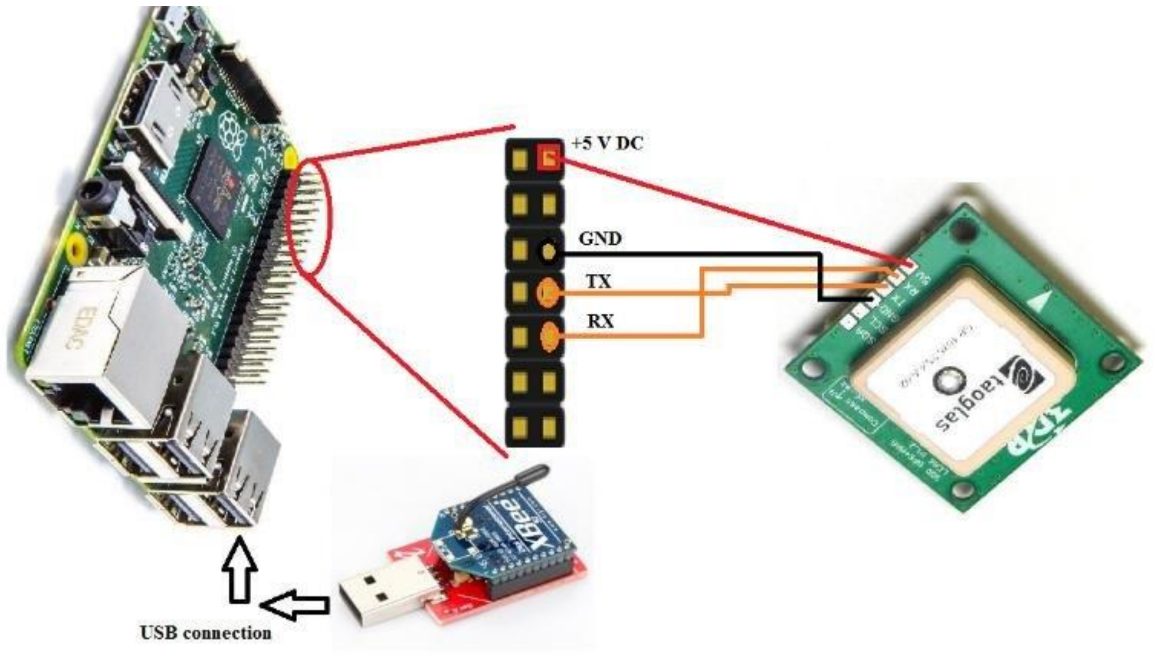
6 Pressure Sensor:

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

Iot Sensor setup:



Raspberry Pi Integration:



The Raspberry Pi is connected to the Internet using an Ethernet cable or a Wi-Fi adapter. The Raspberry Pi is also connected to the water fountain pump using a relay module.

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.

Python Script for Data Transmission:

Develop a Python script to read sensor data and send it to your selected IoT platform. The exact code will depend on the microcontroller and platform you're using.
Here's an example Python script for sending data to AWS IoT using a Raspberry Pi and Python:

```
import time
import boto3
import RPi.GPIO as GPIO  # Assuming you are using a Raspberry Pi
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

# AWS IoT Configuration
IoT_ENDPOINT = "your-iot-endpoint.amazonaws.com"
ROOT_CA = "root-CA.pem"
PRIVATE_KEY = "your-private-key.pem.key"
CERTIFICATE = "your-certificate.pem.crt"
TOPIC = "water-fountain-status"

# Initialize AWS IoT MQTT Client
myMQTTClient = AWSIoTMQTTClient("WaterFountainClient")
myMQTTClient.configureEndpoint(IoT_ENDPOINT, 8883)
myMQTTClient.configureCredentials(ROOT_CA, PRIVATE_KEY, CERTIFICATE)
myMQTTClient.configureOfflinePublishQueueing(-1)
myMQTTClient.configureDrainingFrequency(2)
myMQTTClient.configureConnectDisconnectTimeout(10)
myMQTTClient.configureMQTTOperationTimeout(5)

# Connect to AWS IoT
```

```
myMQTTClient.connect()

# Function to read sensor data
def read_sensors():
    # Replace with your code to read from flow rate and pressure sensors
    flow_rate = 0.0
    pressure = 0.0
    return flow_rate, pressure

try:
    while True:
        flow_rate, pressure = read_sensors()
        data = {
            "flow_rate": flow_rate,
            "pressure": pressure
        }
        myMQTTClient.publish(TOPIC, str(data), 1)
        time.sleep(5)  # Adjust the interval as needed

except KeyboardInterrupt:
    GPIO.cleanup()
    myMQTTClient.disconnect()
```

Front-End:

HTML Structure:

The HTML structure of the Smart Water Fountain Status platform defines the layout and presentation of real-time data. This section guides you through the structure of the HTML file (index.html) that powers the front-end.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Smart Water Fountain Status</title>
```

```
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="status-container">
    <h1>Water Fountain Status</h1>
    <div class="flow-rate">
      <h2>Flow Rate: <span id="flow-rate-value">0</span> GPM</h2>
    </div>
    <div class="malfunction-alert">
      <h2>Malfunction Alert: <span id="malfunction-status">No</span></h2>
    </div>
  </div>
  <script src="app.js"></script>
</body>
</html>
```

3.2 CSS Styles

The CSS styles define the visual design and layout of the platform. This section provides insights into the styles applied through the styles.css file.

Body Styling: Defines the font family and provides a clean appearance for the platform.

Status Container Styling: Provides central alignment and margin for the main status container.

Flow Rate and Malfunction Alert Styling: Defines the styling for displaying flow rate and malfunction status, with an emphasis on readability.

3.3 JavaScript

The JavaScript file (script.js) adds interactivity to the platform. This section explains how the script fetches data from the server and updates the platform in real-time.

Updating Flow Rate: The script includes a function to update the water flow rate displayed on the platform.

Updating Malfunction Status: A function to update the malfunction status displayed on the platform.

Fetching and Updating Data: Details on how the script fetches data from the server's API endpoint and uses it to update the displayed data.

Periodic Updates: The script sets up periodic data updates to keep the platform's information current.

Back-End (Node.js)

Server Script

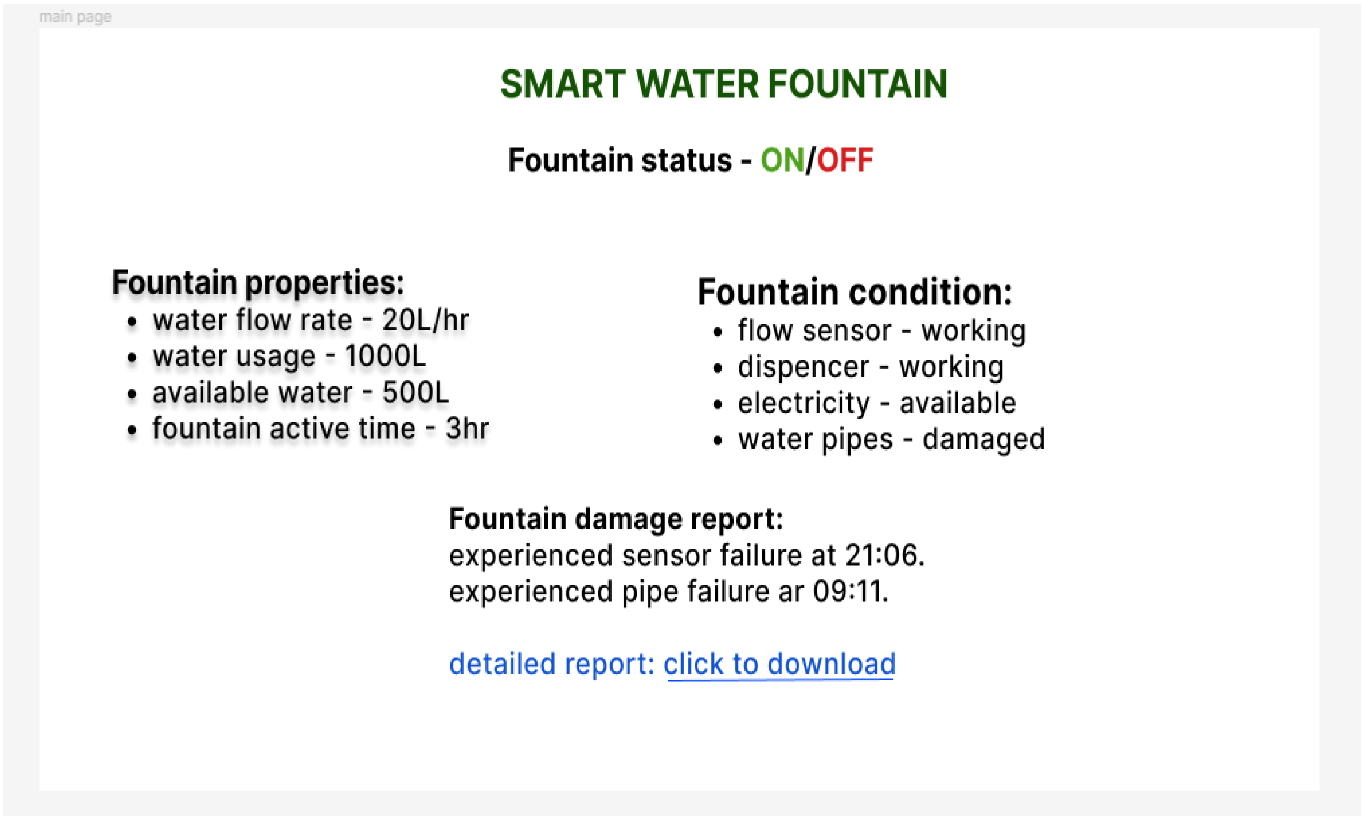
The Node.js server script (app.js) serves as the backbone of the Smart Water Fountain Status platform. This section outlines the purpose and functionality of the server script.

The server script covers the following aspects:

Server Initialization: Information on initializing the Node.js server using Express and specifying the listening port.

Simulated Data: An explanation of the simulated data source used for demonstration purposes. Mention the need to replace this with a real data source.

App Interface:



SMART WATER FOUNTAIN

Fountain status - ON/OFF

Fountain properties:
- water flow rate - 20L/hr
- water usage - 1000L
- available water - 500L
- fountain active time - 3hr

Fountain condition:
- flow sensor - working
- dispencer - working
- electricity - available
- water pipes - damaged

Fountain damage report:
experienced sensor failure at 21:06.
experienced pipe failure ar 09:11.

detailed report: click to download

Real-Time Water Fountain Status System:

The real-time water fountain status system promotes water efficiency and public awareness in the following ways:

- It provides users with real-time data on water fountain usage, allowing them to identify and address any areas of water waste.
- It generates alerts for low water levels and high water temperature, helping to prevent fountain malfunctions and water quality problems.
- It allows users to share water fountain usage data with others, raising awareness of water conservation issues.

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.