SMART WATER MANAGEMENT

Title :Smart Water System

Problem statement :

Water scarcity and inefficient water management are significant challenges faced by communities and industries worldwide. To address these issues, there is a need for the development and implementation of a smart water system using IoT technologies. This system should enable real-time monitoring, efficient distribution, and sustainable use of water resources while providing data-driven insights for better decision-making.

Problem Solution :

Real-time Water Monitoring :

- Deploy a network of IoT sensors and devices at strategic locations in water distribution systems, reservoirs, and treatment plants.
- Utilize wireless communication protocols (e.g., LoRa, Wi-Fi, or cellular) to transmit real-time data to a centralized platform.
- Develop sensor calibration and maintenance protocols to ensure accurate data collection.
- Establish data aggregation and storage infrastructure to handle large volumes of sensor data.

Data Collection and Analysis:

- Develop a robust data collection platform that securely stores and organizes sensor data.
- Implement data analytics and machine learning algorithms to identify patterns, anomalies, and trends in water usage and quality.
- Generate actionable insights from data analysis to inform decision-making.
- Implement data visualization tools for easy interpretation by end-users and administrators.

Leak Detection and Prevention:

- Deploy leak detection sensors in water distribution networks, pipelines, and buildings.
- Develop real-time leak detection algorithms that analyze flow data and pressure variations.
- Implement automatic shut-off mechanisms or alerting systems to respond to detected

leaks.

- Enable remote control for users to shut off water supply in case of suspected leaks.

Water Quality Management:

- Install water quality sensors at various points in the distribution system and treatment plants.
- Continuously monitor parameters such as pH levels, contaminants, turbidity, and disinfectant levels.
- Implement real-time alerts and automatic control systems to address water quality issues.
- Ensure compliance with water quality standards and regulations.

Resource Optimization:

- Implement algorithms that prioritize water supply based on demand, critical areas, and conservation goals.
- Utilize historical data and predictive analytics to forecast water usage patterns.
- Optimize water pressure and flow rates to minimize wastage.
- Promote water-saving practices through user education and incentives.

User Interface:

- Create web and mobile applications that provide intuitive interfaces for end-users and administrators.
- Display real-time data on water consumption, quality, and alerts.
- Enable remote control features to adjust water supply settings based on user needs.
- Energy Efficiency
- Consider renewable energy sources such as solar panels to power IoT sensors and devices.
- Utilize low-power IoT devices and optimize communication protocols to reduce energy consumption.

Scalability:

- Develop modular and adaptable architecture to accommodate various-sized water distribution systems, from residential to industrial.
- Ensure that the system can easily integrate additional sensors and devices as needed.

Data Security and Privacy:

- Encrypt data transmissions to prevent unauthorized access.
- Implement authentication and authorization mechanisms for users and administrators.

Regularly update and patch IoT devices and software to address security vulnerabilities.

Phase 2: Innovation

In this phase you need to put your design into innovation to solve the problem.

Explain in detail the complete steps that will be taken by you to put your design that you thought of in previous phase into transformation.

Create a document around it and share the same for assessment.

NOTE:

File Naming Convention: **TechnologyName_Phase2**

After completion upload your file to your private GitHub account. Please give access to your faculty evaluators[**facultyevaluator@gmail.com**] and industry evaluator [**IndustryEvaluator@skillup.online**] to your private GitHub repository for evaluation process.

PYTHON CODE:

```
import serial import time

data=serial.Serial('com3',9600)

time.sleep(2) print('initializing...')
```

```
#status=float(string)
    time.sleep(0.1)
    while(1==1):
      k=input('enter A for auto nd M for manual') if(k=='2'):
     data.write(b'2')
        b = data.readline()
        time.sleep(2) string_n
        = b.decode()
        string = string_n.rstrip()
        print('Automatic setting applied!')
        print ('motor on') data.write(b'ON')
    print ('status of water level:)',string) break
```

```
HTML CODE FOR SMART WATER MANAGEMENT SYSTEM
!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed
on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the
build.
```

```
      Only files inside the `public` folder can be referenced from the
HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico"
will
      work correctly both with client-side routing and a non-root public
URL.
      Learn how to configure a non-root public URL by running `npm run
build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```