SMART WATER MANAGEMENT

PROJECT SUBMITTED BY;
NAME : PAVITHRA.V
REGNO : 610821106069

- The Smart Water Management System using IoT is a cutting-edge project designed to revolutionize the way water resources are monitored and conserved. This system employs Internet of Things (IoT) technology to collect real-time data on water usage, quality, and distribution. By integrating data analytics and remote control capabilities, it enables

efficient water management, reduces wastage, and ensures the sustainability of this vital resource.

OBJECTIVE:

- Efficient Resource Allocation: Ensure the optimal use of water resources by monitoring and controlling water distribution to minimize waste and maximize efficiency.
- Sustainability: Promote the sustainable use of water resources to protect the environment and future generations.
- Real-time Monitoring: Implement sensors and data analytics to monitor water quality, supply, and infrastructure in real-time, enabling timely response to issues.
- Leak Detection: Quickly identify and address leaks in water supply systems to reduce water loss and infrastructure damage.
- Customer Engagement: Engage and inform water users about their consumption and encourage conservation practices.
- Data-Driven Decision Making: Use data to make informed decisions about water infrastructure investments, maintenance, and policy.

- Water Quality Management: Monitor and maintain water quality to meet regulatory standards and provide safe drinking water.
- Cost Reduction: Optimize operational costs and resource use through smart technologies and automation.
- Sustainability and Conservation: Conserve water resources, protect ecosystems, and reduce water-related energy consumption.
- Improved Public Health: Ensure safe, reliable, and equitable access to clean water for all residents.

MODULES:

- IoT Sensors and Data Collection:

- Deploy a network of IoT sensors at key points in the water supply and distribution system.

- Sensors measure parameters like water flow, pressure, temperature, and quality.

- Collect real-time data and transmit it to a centralized database.

- . Data Analytics and Visualization:

- Utilize data analytics algorithms to process and analyze the collected data.

- Create interactive dashboards and reports for water quality, consumption patterns, and system health.

- Enable water utility operators to make informed decisions based on insights.

- . Leak Detection and Early Warning System:

- Develop algorithms to detect abnormal water consumption patterns indicative of leaks or pipe bursts.

- Implement an early warning system that alerts authorities to potential issues.

- Reduce water losses and infrastructure damage through rapid response.

- Remote Valve Control:

- Install remotely controllable valves in the water distribution network.

- Enable water utility operators to remotely shut off or adjust water flow to specific areas in response to demand or emergencies.

- Improve control over water distribution and reduce wastage.

- Consumer Engagement and Water Conservation:

- Develop a user-friendly mobile app and web portal for consumers to monitor their water usage.

- Provide real-time alerts and suggestions for water conservation.

- Gamify water-saving efforts to encourage responsible consumption.

- Water Quality Monitoring and Treatment:

- Deploy IoT sensors to monitor water quality parameters like pH, turbidity, and contaminants.

- Integrate automated treatment systems to ensure water quality meets regulatory standards.

- Implement real-time alerts for water quality deviations.

- Predictive Maintenance:

- Use predictive analytics to forecast equipment maintenance needs.

- Schedule preventive maintenance to reduce downtime and infrastructure failures.

- Extend the lifespan of water infrastructure components.

- Integration with Weather Data:

- Integrate weather data and forecasts to anticipate changes in water demand and supply.

- Adjust water distribution and treatment processes accordingly during extreme weather events.

- Security and Data Privacy:

- Implement robust security protocols to protect data integrity and prevent unauthorized access.

- Ensure compliance with data privacy regulations and standards.

- Cost-Benefit Analysis and Reporting:

- Calculate the cost savings and environmental benefits achieved through the system.

- Generate reports for water utility management stakeholders.

- Demonstrate the return on investment in smart water management.

The Smart Water Management System using IoT represents a transformative approach to water resource management, promoting sustainability, efficiency, and conservation while ensuring the reliable supply of clean water to communities.

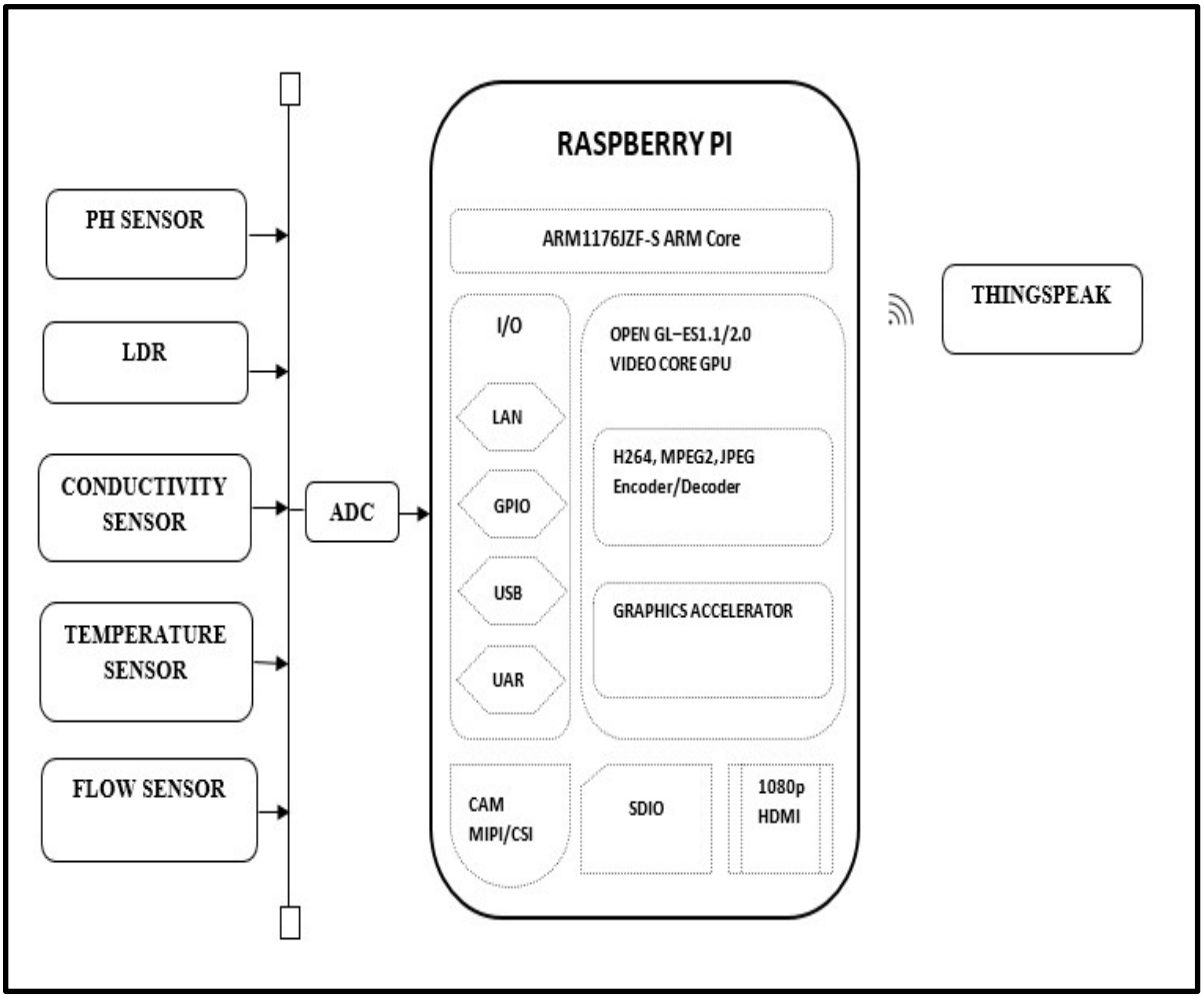
HARDWARE REQUIRED:

- Raspberry Pi
- Temperature Sensor
- Conductivity sensor
- LDR
- Flow sensor
- PH sensor

SOFTWARE REQUIRED:

- Thingspeak cloud
- Raspbian – Jessie
- Python

BLOCK DIAGRAM:



BLOCK DIAGRAM DESCRIPTION:

The whole system is based on Sensors connected to Raspberry Pi to monitor the water quality and Regulated water supply. Raspberry – Single board Computer acts as a heart of the system to perform the desired operation. All the sensor values are uploaded to cloud any person can monitor the parameter

anywhere in the world. Thingspeak – open source cloud provides the values in the form of Graphical representation by using this we can do the analysis.

WORKING OF BLOCK DIAGRAM:

1. RASPBERRY PI:

- **Data Logging and Monitoring:** Raspberry Pi can be equipped with sensors to collect data on water quality, flow rates, and weather conditions. This data can then be logged, analyzed, and sent to a central server for real-time monitoring and decision-making.

- **Remote Control:** Raspberry Pi can be used to remotely control valves, pumps, and other water management equipment, allowing for efficient and automated management of water resources.

- **IoT Integration:** By connecting various sensors and actuators to Raspberry Pi, it can serve as an IoT gateway, enabling seamless integration with other smart devices and central control systems.

- **Leak Detection:** Raspberry Pi-based systems can use water flow sensors to detect leaks or abnormal water consumption, triggering alerts for timely repair and conservation.

- **Remote Access and Control:** Raspberry Pi-based systems can be accessed remotely, allowing water managers to monitor and control water infrastructure from anywhere.

Overall, Raspberry Pi provides a cost-effective and flexible platform for building smart water management solutions that enhance water conservation, efficiency, and sustainability.

Pins in RASPBERRY PI :

The number and function of pins on a Raspberry Pi can vary depending on the model, but the most common Raspberry Pi models like the Raspberry Pi 3 and 4 have a 40-pin GPIO (General Purpose Input/Output) header. These pins are

used for various purposes, including connecting sensors, displays, and other external devices. Here's a general breakdown of the functions of these pins:

- **Power Pins:** These include +3.3V (3.3 volts) and +5V (5 volts) pins for powering external devices.

- **Ground Pins:** These provide ground connections.

- **GPIO Pins:** These are general-purpose input/output pins that can be used for digital input or output, and some can also be configured for special functions like PWM (Pulse Width Modulation).

- **SPI, I2C, UART Pins:** These pins are used for specific communication protocols, such as Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), and Universal Asynchronous Receiver-Transmitter (UART).

- **Other Pins:** Some pins have specific functions like the camera and display connectors, as well as the DSI (Display Serial Interface) and CSI (Camera Serial Interface) pins.

2. PH SENSOR:

PHR (pH) sensor can play a crucial role in smart water management systems by monitoring and controlling the pH levels of water sources. Here's how pH sensors work in this context:

- **Water Quality Monitoring:** pH sensors continuously measure the pH level of water. This data is essential for assessing water quality, especially in applications like wastewater treatment, drinking water purification, and aquaculture.

- **Acid-Base Control**: In smart water management, pH sensors can be integrated with systems that automatically add acid or base chemicals to adjust and maintain the desired pH level. This is crucial for

processes like chemical treatment, ensuring the water's pH is within the required range.

- **Corrosion Control:** In industrial settings, maintaining the correct pH of water is essential to prevent corrosion of pipes and equipment. pH sensors can trigger alarms or actions when the pH level becomes corrosive.

- **Environmental Monitoring:** pH sensors can be used in environmental monitoring systems to ensure that natural water bodies (rivers, lakes) maintain healthy pH levels for aquatic life and ecosystem preservation.

- **Data Analysis:** pH sensor data can be integrated into smart water management systems to provide insights, trends, and historical records. Machine learning algorithms can be used to predict pH fluctuations or detect anomalies.

By incorporating pH sensors into smart water management, you can optimize water treatment, reduce operational

costs, and ensure that water quality remains within specified limits, making it safer for consumption and beneficial for various industrial and environmental applications

Applications of pH Sensors:

- Environmental Monitoring: pH sensors are used to monitor water quality in natural bodies of water, wastewater treatment plants, and industrial effluents.

- Agriculture: They are employed in agriculture to assess soil pH levels, which can influence crop health and yield.

- Chemical Processing: pH sensors are crucial in chemical manufacturing to control and optimize processes, ensuring product quality.

- Food and Beverage Industry: They are used to monitor and control the pH of products like beverages and dairy products.

- Biomedical Research: pH sensors are used in various medical applications, such as monitoring the pH of blood and other bodily fluids.

- Aquariums: pH sensors help maintain the optimal pH level in aquariums to support aquatic life.

Advantages of pH Sensors:

- Accuracy: They provide accurate and precise pH measurements.

- Ease of Use: pH sensors are relatively easy to use, making them accessible for a wide range of applications.

- Real-time Monitoring: They enable real-time monitoring of pH levels, allowing for prompt adjustments in various processes.

- Longevity: Many pH sensors have a long lifespan when properly maintained.

Disadvantages of pH Sensors:

- Maintenance: They require regular calibration and maintenance to ensure accurate measurements.

- Fragility: Some pH sensors are fragile and can be damaged by extreme conditions or certain chemicals.

- Electrode Contamination: Contaminants can build up on the electrode over time, affecting accuracy.

- Electrode Poisoning: Exposure to certain chemicals can poison the sensor, making it less effective.

- Cost: High-quality pH sensors can be relatively expensive, especially for specialized applica**.**

- Electrode Poisoning: Exposure to certain chemicals can poison the sensor, making it less effective.

- Cost: High-quality pH sensors can be relatively expensive**,** especially for specialized applications**.**

3. LDR SENSOR:

- Light Dependent Resistor (LDR) sensors, also known as photoresistors, are not typically used directly for water management, but they can indirectly contribute to smart water management systems in specific applications. Here's how an LDR sensor might be used in this context:
- Indirect Light Measurement: LDR sensors primarily measure light intensity. In smart water management, LDRs can be used in scenarios where light levels affect the water management process. For example, LDR sensors can be part of a broader system in applications like:

a. Solar-Powered Pumping: In remote locations, solar-powered water pumping systems might use LDRs to measure ambient light. These readings can help determine when to pump water (e.g., during daylight hours when solar power is available).

b. Irrigation Control: LDR sensors can be used to detect natural light levels for outdoor irrigation systems. When it's dark or cloudy, the irrigation system can reduce water usage to conserve energy and resources.

c. Water Quality Monitoring: Some water quality monitoring systems may use LDRs to assess light penetration in water bodies, which can indirectly indicate water quality by assessing factors like turbidity.

d. Data Acquisition: LDRs generate electrical resistance that varies with light levels. The resistance can be converted to voltage or current and processed using analog-to-digital converters (ADCs) to provide measurable data.

e. Integration with Control Systems: In scenarios where LDR data is relevant, it can be integrated into the control system for smart water management. This integration allows for decisions or actions based on the light data, such as adjusting pump operation or irrigation schedules.

f. Data Logging and Analysis: LDR data can be logged and analyzed over time to assess patterns and correlations with other parameters, helping to fine-tune water management strategies.

It's essential to note that LDR sensors are not the primary sensors for water-related parameters like water level, quality, or flow. Instead, they are used in applications where light levels influence water management decisions or processes.

DESCRIBING SMART WATER MANAGEMENT:

1. **IoT Sensor Setup:**

   Install a network of IoT sensors at critical points in the water distribution system.

Sensors can measure parameters like water flow, pressure, quality, temperature, and leak detection.

Connect sensors to a central gateway for data aggregation

2. **Mobile App Development:**

Develop a mobile app for users and administrators to access data and control the system.

The app can provide real-time information on water usage, quality, and alerts.

Users can set preferences and receive notifications.

Administrators can monitor system health and respond to alerts.

3. **Raspberry Pi Integration:**

Use Raspberry Pi as a central controller for data aggregation and system control.

Connect Raspberry Pi to the IoT gateway to collect sensor data.

Implement automation and control algorithms on the Raspberry Pi.

Ensure secure communication with the mobile app and cloud services.

4. **Code Implementation:**

   Develop code for the Raspberry Pi to process sensor data and make control decisions.

   Use programming languages like Python or C++.

   Implement algorithms for leak detection, pressure regulation, and data storage.

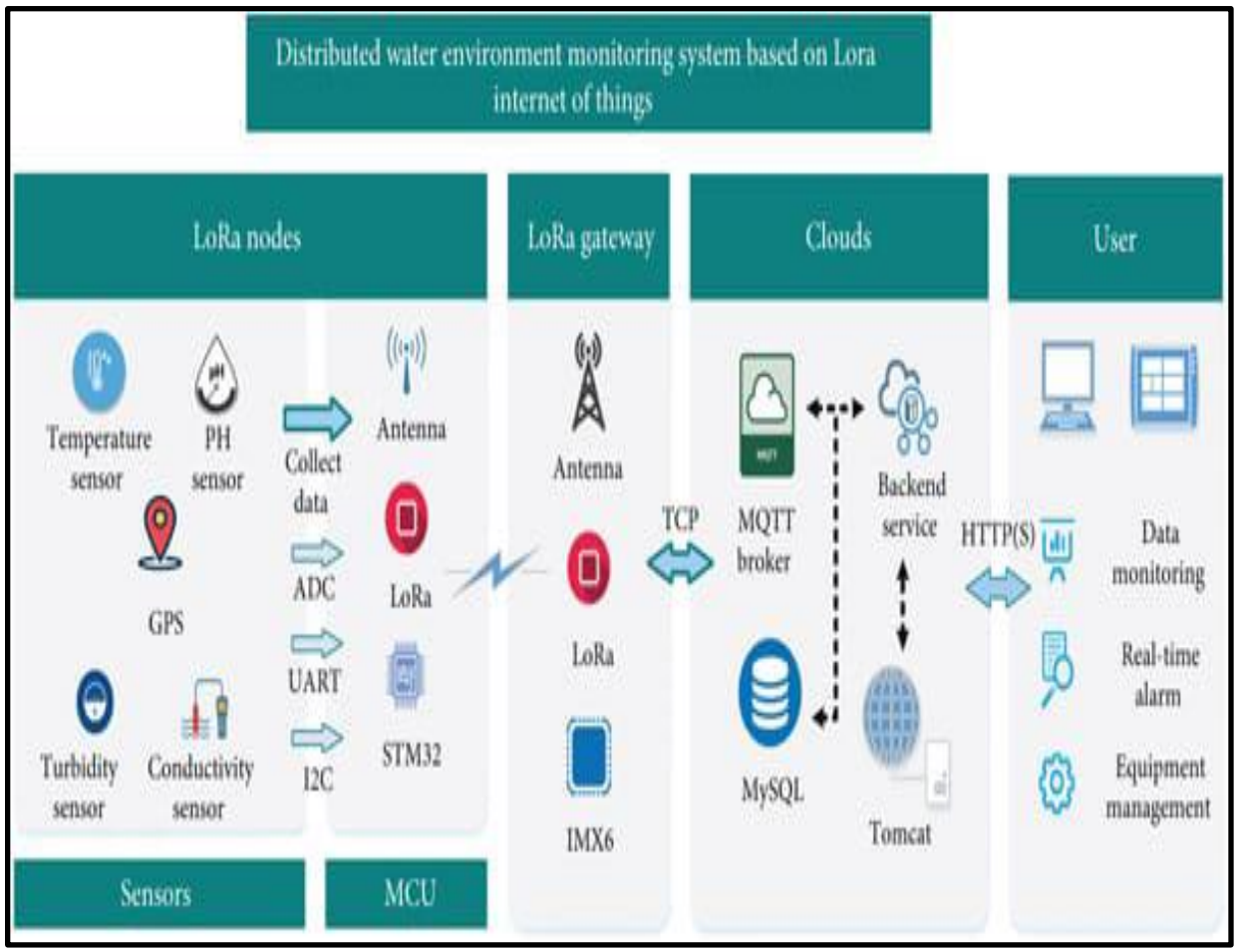
   Ensure data encryption and security protocols to protect sensitive information.

Here's a simplified example of code for a Raspberry Pi (using Python) to collect sensor data and send it to a cloud service:

```
python
Copy code
import sensor_library
import cloud_api

# Initialize sensors
sensors = sensor_library.initialize_sensors()
```

```
# Main loop
while True:
    # Collect sensor data
    sensor_data = sensor_library.read_sensor_data(sensors)

    # Send data to the cloud
    cloud_api.send_data_to_cloud(sensor_data)

    # Implement control logic here (e.g., leak detection, pressure regulation)

    # Check for user commands from the mobile app

    # Handle system alerts and notifications
```

The code will need to be customized based on the specific sensors and cloud services being used, as well as the control logic and user interactions required for the smart water management system.



PROGRAM:

```
import RPi.GPIO as GPIO
import time
from gpiozero import MCP3008
import smbus
```

```
from time import sleep


# Define sensor pins
ph_sensor_pin = 0  # Connect pH sensor to MCP3008 channel 0
temperature_sensor_pin = 1  # Connect temperature sensor to MCP3008 channel 1
flow_sensor_pin = 2  # Connect flow sensor to MCP3008 channel 2


# Initialize I2C bus for conductivity sensor
bus = smbus.SMBus(1)
conductivity_sensor_address = 0x4A  # Adjust based on your sensor's address


# Initialize GPIO for controlling pumps or valves
pump_pin = 17
valve_pin = 18


GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(pump_pin, GPIO.OUT)
GPIO.setup(valve_pin, GPIO.OUT)

def read_ph_sensor():
    # Use MCP3008 to read analog values from the pH sensor
    # Implement pH conversion logic
    # Return pH value

def read_temperature_sensor():
    # Use MCP3008 to read analog values from the temperature sensor
    # Implement temperature conversion logic
    # Return temperature in Celsius

def read_flow_sensor():
    # Use MCP3008 to read analog values from the flow sensor
    # Implement flow conversion logic
    # Return flow rate in liters per minute
```

```
def read_conductivity_sensor():
    # Read data from the I2C-connected conductivity sensor
    # Implement conductivity conversion logic
    # Return conductivity value

def control_pump(status):
    # Control the pump (on/off) based on the status

def control_valve(status):
    # Control the valve (open/close) based on the status

while True:
    try:
        pH = read_ph_sensor()
        temperature = read_temperature_sensor()
        flow_rate = read_flow_sensor()
        conductivity = read_conductivity_sensor()
```

```
        # Implement smart water management logic here
        # For example, adjust the pump or valve based on sensor data

        # You can also send data to a cloud service for monitoring and analysis

        time.sleep(60)  # Read sensor data and control periodically
    except KeyboardInterrupt:
        GPIO.cleanup()
        break
```

Creating a data sharing platform for real-time water consumption data using web technologies like HTML,CSS and JavaScript is a great initiative for promoting water conservation efforts. Here's a high-level outline of the steps:

1. **Project Setup:**

Set up your development environment with the necessary tools, including a code editor, web server, and a version control system like Git.

2. **User Interface Design:**

- Design a user-friendly web interface for the platform. Use HTML for structuring the content and CSS for styling.
- Create different sections for displaying real-time data, such as charts, tables, and statistics.
- Implement responsive design to ensure the platform works well on various devices..

3. **IoT Integration:**

- Set up communication between the IoT sensors and your web platform. This may involve using MQTT, HTTP APIs, or other IoT protocols
- Ensure data is securely transmitted and authenticated.

4.User Authentication and Access Control:

- Implement user authentication to secure the platform. Only authorized users should be able to access and modify the data.
- Define user roles and permissions to control who can do what on the platform.

5.Data Analytics:

Consider implementing data analytics and machine learning algorithms to provide insights and recommendations for water conservation efforts.

6.Testing and Debugging:

- Thoroughly test your platform to ensure it functions correctly, including edge cases and security vulnerabilities.
- Debug and optimize code for better performance.

7.Monitoring and Maintenance:

- Implement monitoring tools to keep an eye on the platform's performance and security.
- Regularly update and maintain the platform to adapt to new technologies and security practices.

To create a platform that displays real-time water consumption data using HTML, Here we are using a simple web page structure and then JavaScript to fetch and display the data. Here's a basic example:

html

Copy code

```
<!DOCTYPE html>
<html>
<head>
    <title>Real-time Water Consumption</title>
    <style>
        /* Add your CSS styles here for formatting the data display */
    </style>
</head>
<body>
    <h1>Real-time Water Consumption</h1>

    <div id="waterConsumption">
 <!-- This is where the real-time data will be displayed -->
    </div>

    <script>
        // JavaScript for fetching and displaying real-time data
        function updateWaterConsumption() {
```

```
        // Use JavaScript to fetch data from your IoT sensors or API
        // For simplicity, let's assume we have a function getData() that returns the data
        const data = getData(); // Replace with your actual data retrieval method
 // Update the content of the "waterConsumption" div

document.getElementById("waterConsumption").innerHTML = `Current Water Consumption: ${data} liters`;
    }

    // Update data every X milliseconds (e.g., every 5 seconds
//setInterval(updateWaterConsumption, 5000);
  </script>
</body>
</html>
```

In this example:

- We have a basic HTML structure with a heading and a waterConsumption div where the real-time data will be displayed.
- We use JavaScript to periodically fetch the data from your IoT sensors or an API using the getData() function (you'll need to implement this function to get your actual data).
- We update the content of the waterConsumption div with the retrieved data.
- The setInterval function is used to repeatedly call the updateWaterConsumption function at regular intervals (in this case, every 5 seconds). You can adjust this interval to suit your needs for real-time updates.
- Remember to replace getData() with your actual data retrieval method. Additionally, you may want to add more styling and formatting to make the data display more visually appealing.

Designing a platform to receive and display water consumption data from IoT sensors while promoting water conservation efforts involves several components. Here's an outline of how to design such a platform:

1.User Interface:

- Create an intuitive and user-friendly web interface that allows users to access and interact with the water consumption data.
- Use responsive design to ensure accessibility on various devices.

2.Dashboard:

- Develop a dashboard that provides real-time and historical water consumption data.
- Include charts, graphs, and statistics to visualize the data effectively.

3.User Registration and Authentication:

- Implement user registration and login functionality to ensure data security and personalized experiences.
- Use encryption and secure authentication methods to protect user data.

4.IoT Sensor Integration:

- Connect and integrate IoT sensors with the platform using appropriate communication protocols like MQTT, HTTP, or WebSocket.
- Ensure data is securely transmitted and authenticated between sensors and the platform.

5.Data Storage:

- Set up a database system to store the water consumption data. Use a database technology like MySQL, PostgreSQL, or NoSQL databases depending on your requirements.

6.Notifications and Alerts:

- Create a notification system that informs users about their water consumption patterns.
- Implement alerts for abnormal water usage, leak detection, or conservation tips.

7.Data Analytics:

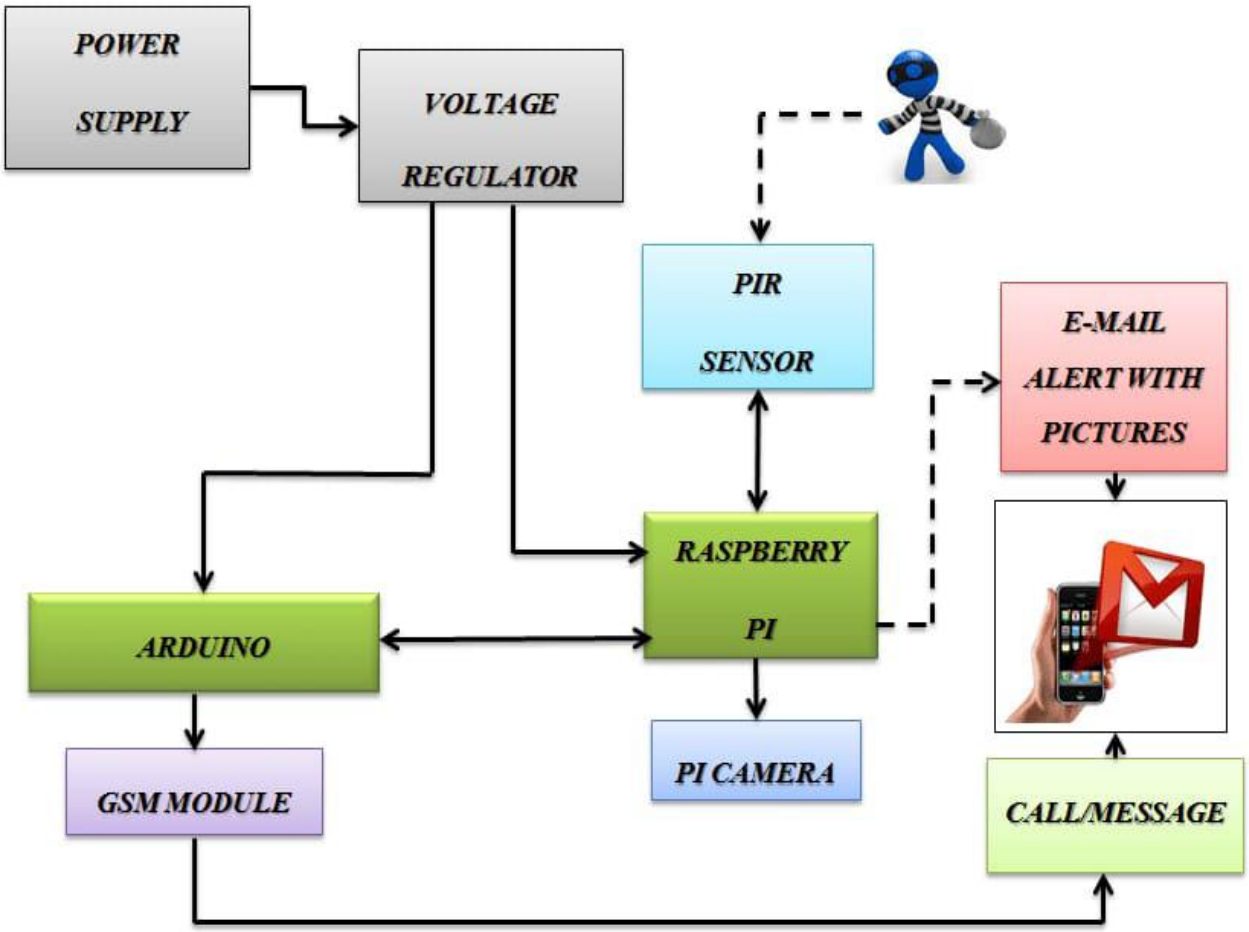
- Use machine learning and data analytics to analyze water consumption patterns.
- Provide users with insights and recommendations on how to conserve water.

8.Feedback and Reporting:

- Provide a way for users to report issues, give feedback, and request assistance.
- Monitor and analyze user feedback for platform improvement.

9.Testing:

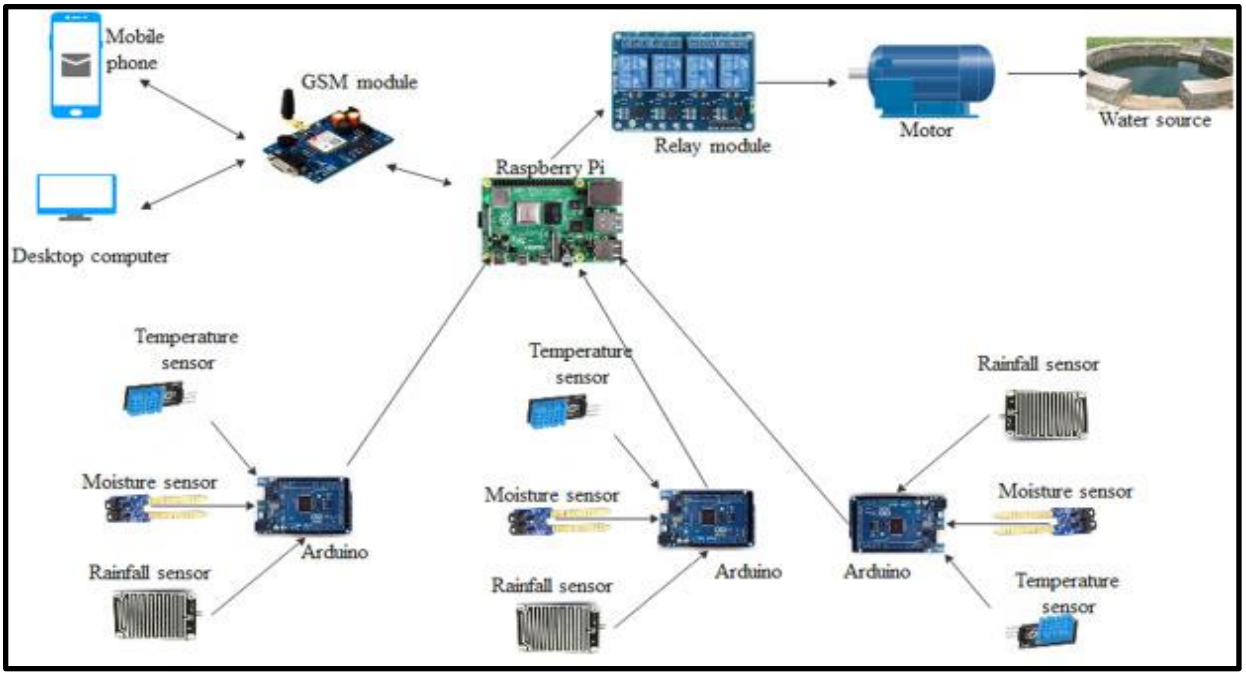
- Thoroughly test the platform to ensure data accuracy, performance, and security.
- Regularly maintain and update your platform to ensure it continues to work correctly and securely.



SUBMISSION

Providing the instructions on how to replicate the smart water management project

Creating a smart water management project involves various components and can vary in complexity depending on your specific requirements. Below is a general outline to get you started, but please note that the specific details and steps can vary based on your project goals and available resources.



Materials and Components Needed:

- **Sensors:** You'll need sensors to measure parameters like water level, quality (pH, turbidity), temperature,

and flow rate. Choose sensors suitable for your application.

- **Microcontroller:** Select a microcontroller like Arduino, Raspberry Pi, or a specialized IoT device to collect and process sensor data.

- **Power Supply:** Depending on the project's location, you may need a power source, such as batteries, solar panels, or a mains power supply.

- **Data Communication:** Choose a method to transmit data, such as Wi-Fi, cellular, LoRa, or Ethernet, depending on your project's location and connectivity options.

- **Actuators:** If your project requires control, you may need actuators like valves or pumps to manage water flow or quality.

- **Data Storage and Analysis:** Set up a data storage system, such as a local database or cloud-based solution, to log and analyze the data.

- **User Interface:** Design a user interface, which could be a web app, mobile app, or a graphical user interface (GUI) on a computer, to display data and allow for user interaction.

Steps to Replicate the Project:

- **Project Planning:** Define the project's objectives, scope, and requirements. Determine which parameters you want to monitor and control.

- **Sensor Setup:** Install and calibrate the sensors. Ensure they are correctly interfaced with the microcontroller.

- **Microcontroller Programming:** Write the code for the microcontroller to read sensor data, process it, and transmit it to your chosen data storage and display

system. This may involve using Arduino IDE, Python, or other programming languages, depending on your hardware.

- **Data Storage and Analysis:** Set up the data storage system and design data analysis algorithms to process and visualize the data. Cloud platforms like AWS, Azure, or Google Cloud can be useful for cloud-based solutions.

- **Communication:** Configure the data communication method. If you're using Wi-Fi or cellular, connect to the internet; if using LoRa, set up gateways and nodes.

- **Actuators (if needed):** If your project involves control, implement the code to control actuators based on the data and user inputs.

- **User Interface:** Develop the user interface to display real-time data, alerts, and control options. Ensure it's accessible via web or mobile devices.

- **Power Supply:** Set up the power supply, ensuring it can provide consistent power to your system.

- **Testing and Calibration:** Test the entire system, calibrate sensors if necessary, and ensure it functions correctly. Perform extensive testing to ensure reliability.

- **Deployment:** Install your system at the water management site. Make sure it's secure, weather-resistant, and has reliable connectivity.

- **Monitoring and Maintenance:** Regularly monitor your system for data accuracy and system health. Perform maintenance and address issues as they arise.

Replicating a system that deploys IoT sensors, transmits information, and integrates them using Python involves multiple steps.

1**. Define Objectives:**

- Determine the purpose of your IoT sensor network and the type of data you want to collect.

- Identify the key parameters you need to monitor (e.g., temperature, humidity, air quality, motion).

2. Choose IoT Sensors:

- Select appropriate sensors based on your objectives. Consider factors like sensor accuracy, communication protocols, and power requirements.

- Common choices include sensors like DHT22 for temperature and humidity, PIR motion sensors, and air quality sensors.

3. Set Up Hardware:

- Connect the selected sensors to microcontrollers (e.g., Arduino, Raspberry Pi) or IoT development boards.

- Ensure that the hardware is powered appropriately and has internet connectivity (Wi-Fi, cellular, or other options).

4. Develop Sensor Code:

- Write code to read data from the sensors and format it for transmission.

- Use Python libraries or platforms like Raspberry Pi GPIO or Arduino IDE for programming your devices.

5. Communication:

- Choose a communication protocol (MQTT, HTTP, CoAP) and set up communication with a central platform.

- Transmit sensor data to a central server or cloud-based platform. You can use MQTT libraries in Python for this purpose.

6. Transit Information Platform:

- Set up a platform (e.g., AWS IoT, Google Cloud IoT Core) to receive and store the data from your IoT sensors.

- Configure the platform to manage data ingestion and device management.

7. Data Storage:

- Choose a database system (e.g., SQL, NoSQL) to store the sensor data.

- Use Python libraries to interact with the database and store the incoming data.

8. Data Processing:

- Develop Python scripts for data processing, including data analysis, visualization, and real-time alerts.

- Use libraries like pandas, NumPy, and Matplotlib for data analysis and visualization.

9. Integration:

- Build Python applications or scripts that integrate with your data platform and provide APIs for data retrieval.

- Set up authentication and security measures to protect your data and access.

10. User Interface:

- Create a user-friendly interface for visualizing sensor data. You can use web frameworks like Flask or Django for web-based interfaces.

- Implement dashboards or mobile apps to view and control the sensor network.

11. Security:

- Implement strong security practices, including encryption, access control, and regular security audits to protect your IoT network.

12. Testing and Deployment:

- Test your IoT sensor network in a controlled environment before deploying it in the target location.

- Ensure the reliability and scalability of your system.

13. Documentation:

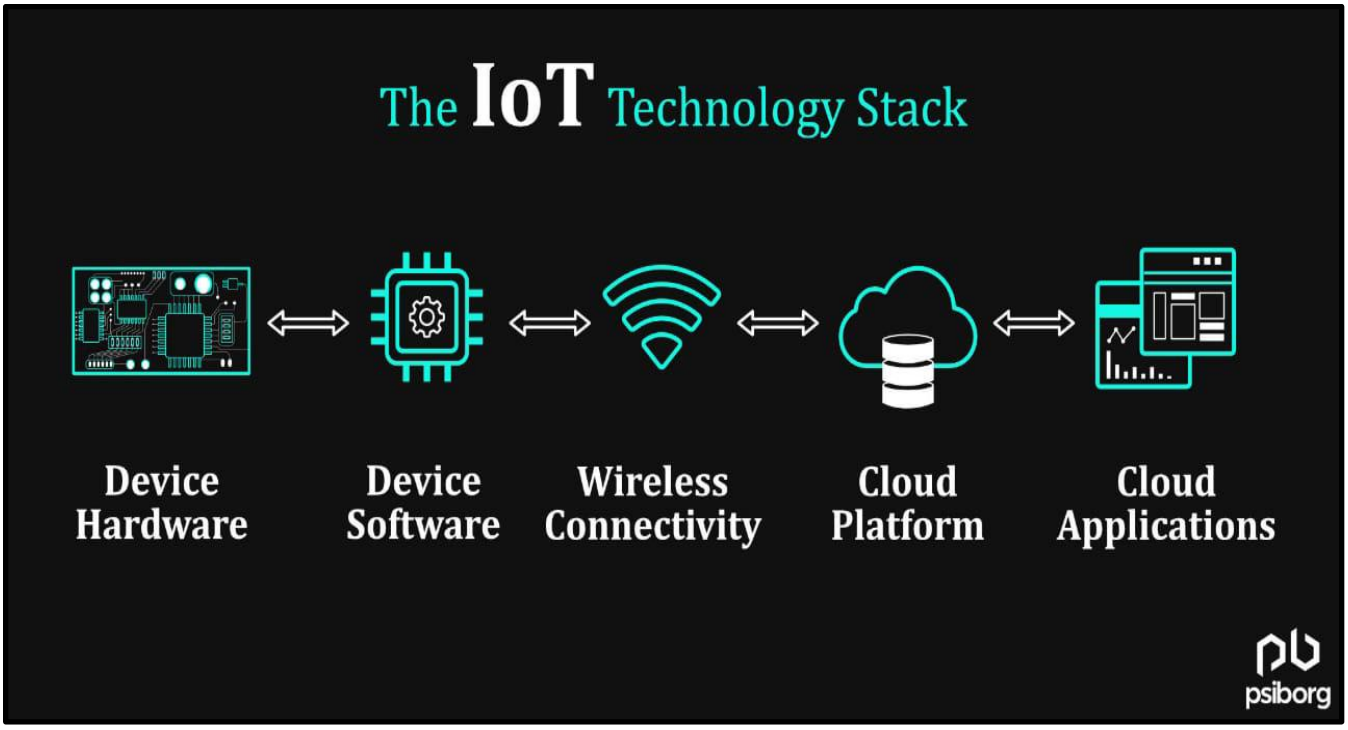
- Document your system's architecture, code, configurations, and troubleshooting procedures for future reference.

14. Maintenance:

- Establish a maintenance plan to ensure the continuous operation of your IoT sensor network. Regularly update software and firmware.

15. Scalability:

- Plan for system scalability to accommodate more sensors and data in the future.



Raspberry Pi Data Transmission (Python)

1. First, make sure you have Python and the necessary libraries installed on your Raspberry Pi. You may need to install additional libraries for specific sensors and communication protocols.

2. Let's assume you have a DHT22 temperature and humidity sensor connected to your Raspberry Pi. You can use the `Adafruit_DHT` library to read data from the sensor. Here's a simple Python script to read data from the sensor and transmit it to a server using MQTT:

```
import Adafruit_DHT
import paho.mqtt.client as mqtt

# Sensor setup
```

```
sensor = Adafruit_DHT.DHT22
pin = 4  # GPIO pin where the sensor is connected

# MQTT setup
mqtt_broker = "your_mqtt_broker_address"
mqtt_topic = "sensor_data"

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))

client = mqtt.Client()
client.on_connect = on_connect
client.connect(mqtt_broker, 1883, 60)

while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
    if humidity is not None and temperature is not None:
```

```
    payload = f"Temperature: {temperature:.2f}°C, Humidity: {humidity:.2f}%"
    client.publish(mqtt_topic, payload)
```

This script reads data from the DHT22 sensor and publishes it to an MQTT topic.

Mobile App User Interface (Flutter)

Here's a basic example of a mobile app interface to visualize the data using Flutter, a popular cross-platform framework for mobile app development:

```dart
import 'package:flutter/material.dart';
import 'package:mqtt_client/mqtt_client.dart';

void main() => runApp(MyApp());
```

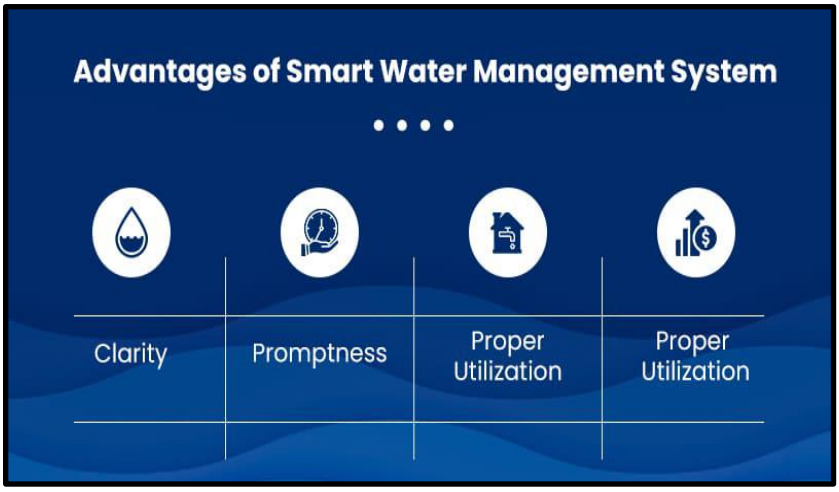
```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: SensorDataScreen(),
    );
  }
}

class SensorDataScreen extends StatefulWidget {
  @override
  _SensorDataScreenState createState() =>
_SensorDataScreenState();
}

class _SensorDataScreenState extends
State<SensorDataScreen> {
  MqttClient mqttClient;
  String sensorData = "No data available";
```

```
 @override
 void initState() {
  super.initState();
  _connectToMQTT();
 }

 _connectToMQTT() {
  mqttClient = MqttClient('your_mqtt_broker_address', '');
  mqttClient.onConnected = _subscribeToTopic;
  mqttClient.connect();
 }

 _subscribeToTopic() {
  mqttClient.subscribe('sensor_data');

mqttClient.updates.listen((List<MqttReceivedMessage<Mqt
tMessage>> messages) {
```

```
    final MqttPublishMessage message =
messages[0].payload;
    final String data =
MqttPublishPayload.bytesToStringAsString(message.payloa
d.message);


    setState(() {
      sensorData = data;
    });
  });
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Sensor Data'),
    ),
    body: Center(
```

```
        child: Text(sensorData),
      ),
    );
  }
}
```

This Flutter app connects to an MQTT broker, subscribes to the "sensor_data" topic, and displays the received sensor data on the screen.



CONCLUSION:

The design and development of a low-cost system for real-time monitoring of water quality and controlling the flow of water by using IoT is presented. The proposed system consists of sensors for water quality monitoring and solenoid valve for controlling the water flow in the pipeline. These devices are low in cost, highly efficient and flexible. These are connected to Raspberry Pi core controller and IoT module. Finally, sensed values viewed and controlling is performed by the internet and also through Wi-Fi to mobile devices

THANK YOU