Smart Water Fountains using IOT.

Objectives:

“To design and implement a network of smart water fountains that provide convenient, clean, and sustainable access to drinking water while promoting water conservation and public health.”

IOT Sensors:

Water Quality Sensors:

PH sensors, turbidity sensors, temperature sensors, and conductivity sensors to ensure the water's safety and quality.

Flow Sensors:

These can measure the rate of water flow, helping track consumption and detect anomalies in usage.

Pressure Sensors:

Useful for monitoring water pressure in the fountain system, ensuring proper functioning and identifying leaks or blockages.

Level Sensors:

To determine the water level in the fountain, ensuring it remains adequately filled and triggering alerts when water levels are low.

Temperature Sensors:

Monitoring water temperature for user preferences, especially if the fountain offers options for hot, cold, or ambient temperature water.

Proximity/Motion Sensors:

To enable touchless operation, activating the fountain when a user approaches and deactivating it when they move away.

Ultrasonic Sensors:

Providing accurate measurement of the distance to the water outlet, aiding in dispensing the right amount of water.

Humidity Sensors:

Particularly useful for outdoor fountains to prevent water wastage during rainy or humid conditions by temporarily disabling the fountain.

Vandalism/Tamper Detection Sensors:

To detect unauthorized or malicious interference, ensuring the security and integrity of the fountain.

Mobile app development:

Project Goals:

Clearly outline the purpose of the smart water fountain, what are the features want in the app and the target users.

Choose a Platform:

Decide whether you want to develop the app for iOS, Android, or both platforms. You can use native development (Swift for iOS and Java/Kotlin for Android) or opt for cross-platform frameworks like React Native or Flutter to save development time and resources.

Design User Interface:

Create wireframes and design the user interface for the app. Ensure it's intuitive and user-friendly.

Hardware Integration:

If your smart water fountain connects with sensors or IoT devices, you'll need to integrate these components into the app. This might require expertise in IoT development.

App Development:

Write the code for the app, implementing the features and functionality you've planned. Ensure the app can control the water fountain, track usage, and provide real-time data.

Data Management:

Implement a back-end system to store and manage data from the smart water fountain and user interactions. You might need a database and server infrastructure.

User Authentication:

Implement user authentication and authorization to control access to the app's features and data.

Testing:

Rigorously test the app on various devices and ensure it works seamlessly. Check for usability, security, and performance issues.

Feedback and Iteration:

Collect feedback from early users and make necessary improvements to the app.

Deployment:

Publish the app on the respective app stores (Apple App Store and Google Play Store). Ensure it complies with their guidelines.

Maintenance and Updates:

Regularly update the app to fix bugs, add new features, and ensure compatibility with the latest devices and operating systems.

Marketing and Promotion:

Promote the app to your target audience through various marketing channels.

Raspberry Pi Code:

```
// Flutter Dart code for the mobile app
Import ‘package:flutter/material.dart’;
Import ‘package:http/http.dart’ as http;

Void main() {
  runApp(MyApp());
}

Class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    Return MaterialApp(
      Home: MyHomePage(),
    );
  }
}

Class MyHomePage extends StatefulWidget {
  @override
```

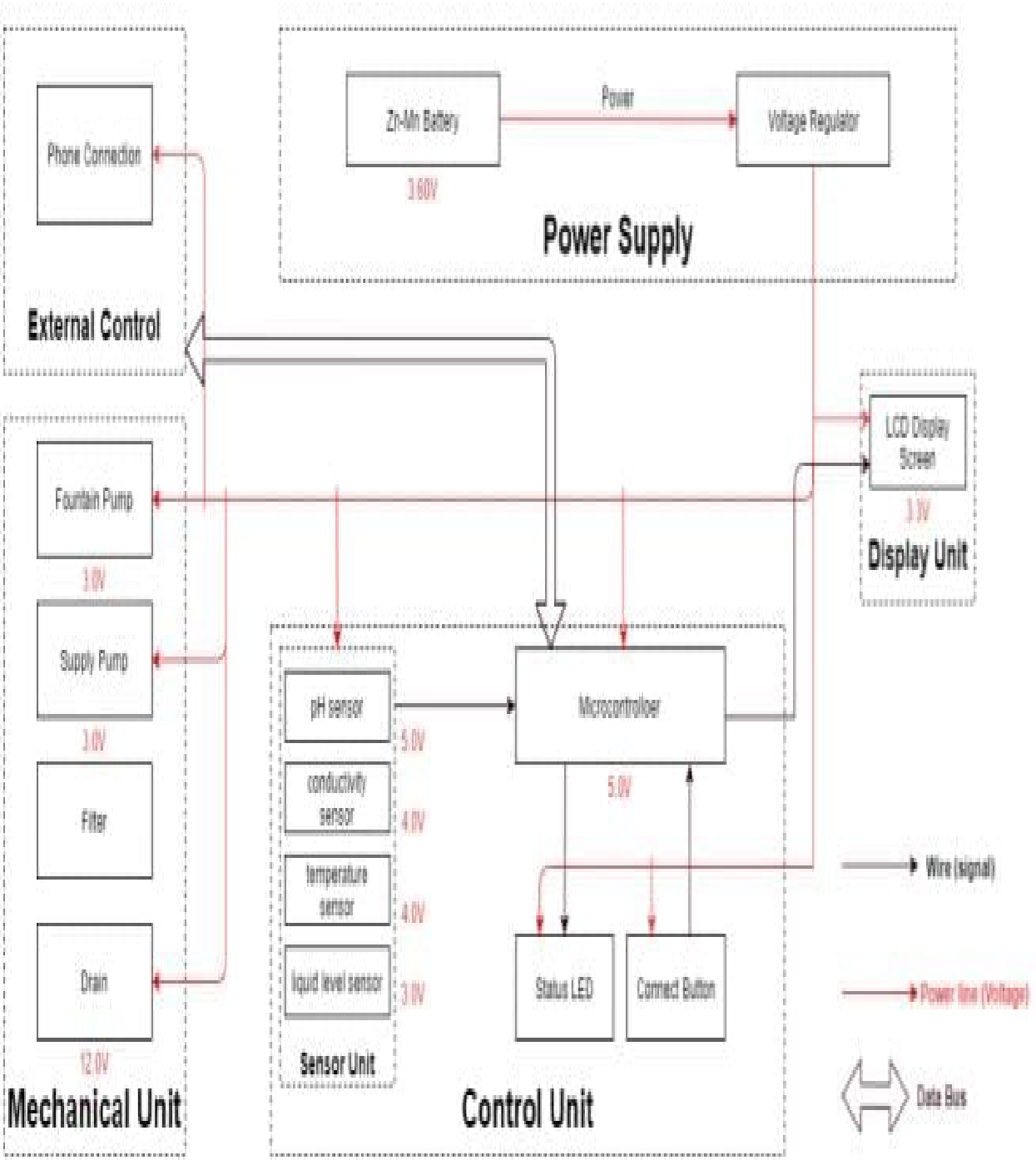
```
  _MyHomePageState createState() => _MyHomePageState();
}

Class _MyHomePageState extends State<MyHomePage> {
  Bool isPumpOn = false;

  Void togglePump() {
    setState(() {
      isPumpOn = !isPumpOn;
    });

    // Send a request to your Raspberry Pi or server to control the pump
    http.post('http://raspberrypi.local/pump', body: {'on': isPumpOn ? '1' : '0'});
  }

  @override
  Widget build(BuildContext context) {
    Return Scaffold(
      appBar: AppBar(
        title: Text('Smart Water Fountain'),
```

```
      ),
      Body: Center(
        Child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              isPumpOn ? 'Water Pump is On' : 'Water Pump is Off',
              style: TextStyle(fontSize: 24),
            ),
            ElevatedButton(
              onPressed: togglePump,
              child: Text(isPumpOn ? 'Turn Off Pump' : 'Turn On
Pump'),
            ),
          ],
        ),
      ),
    );
  }
}
From flask import Flask, request, jsonify
```

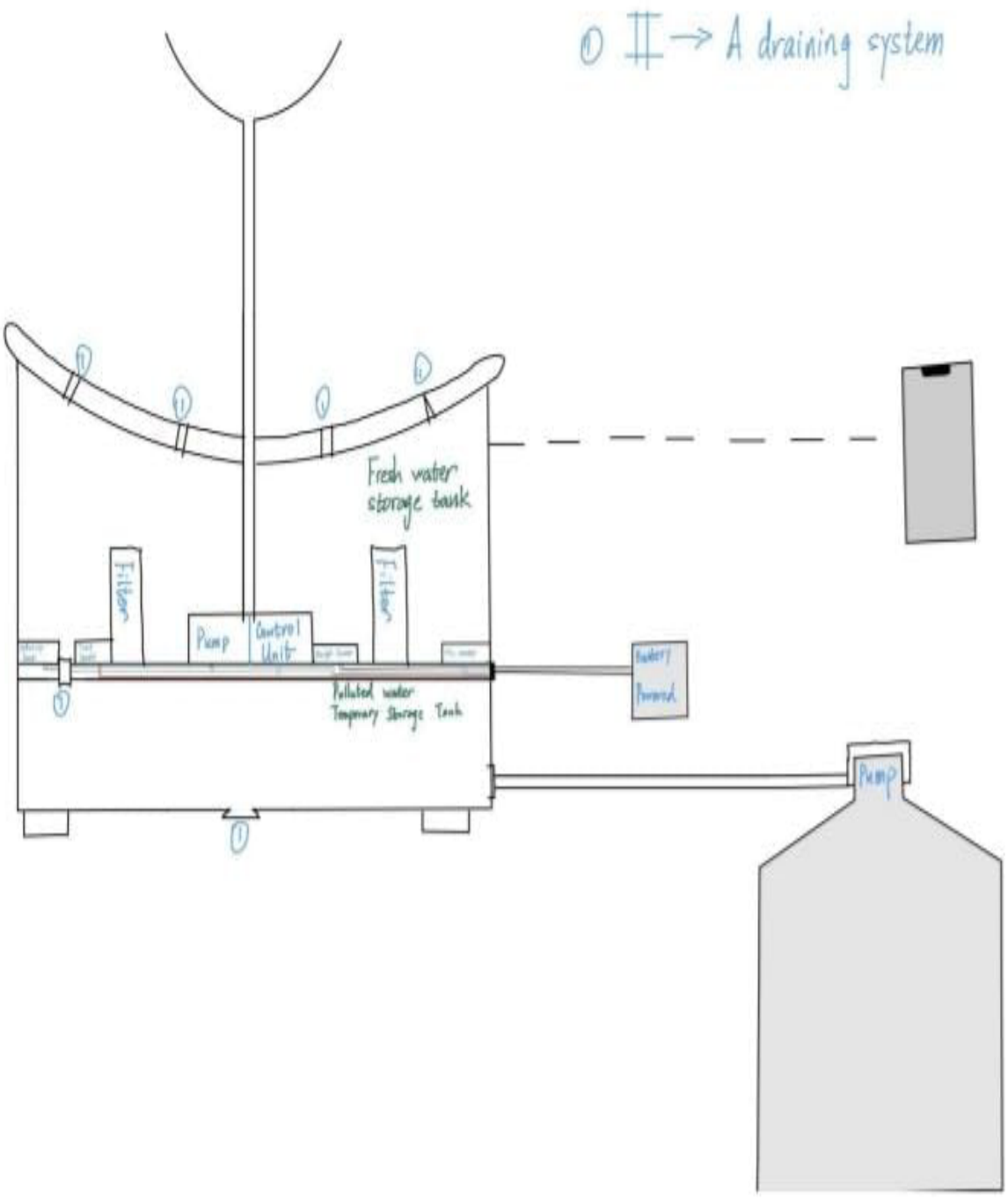
```
Import RPi.GPIO as GPIO
Import time

App = Flask(__name__)

Pump_pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(pump_pin, GPIO.OUT)

@app.route(‘/pump’, methods=[‘POST’])
Def control_pump():
    Data = request.form
    If ‘on’ in data:
        If data[‘on’] == ‘1’:
            GPIO.output(pump_pin, GPIO.HIGH)
        Else:
            GPIO.output(pump_pin, GPIO.LOW)
    Return jsonify({‘status’: ‘success’})

If __name__ == ‘__main__’:
    App.run(host=’0.0.0.0’, port=80)
```

Diagrams for smart water fountains:

Architecture:



Physical diagram:



Real time water fountains status:

The real-time water fountains status system promotes water efficiency and public awareness in several ways:

Real-Time Monitoring:

The system constantly monitors the status of water fountains, detecting issues like leaks or excessive water flow. This ensures that water is not wasted due to malfunctioning fountains.

Reduced Water Waste:

By promptly identifying and addressing issues, the system helps prevent water wastage, which is critical for water conservation in regions facing water scarcity or drought.

Public Awareness:

The system can display data on the availability and quality of water from fountains in real time. This information encourages people to use fountains rather than buying bottled water, which can reduce plastic waste and energy consumption associated with bottled water production.

User-Friendly Apps:

Mobile apps and websites can provide easy access to the status of nearby water fountains, encouraging people to choose refillable containers over single-use plastic bottles.

Educational Outreach:

The system can be used to provide educational content about water conservation and the importance of reducing single-use plastics, further raising public awareness.

Behavioral Change:

With access to data about fountain locations and water quality, individuals are more likely to make eco-friendly choices, fostering a culture of water efficiency and sustainability.

Cost Savings: For municipalities and organizations, the system can help reduce maintenance costs by pinpointing issues early, optimizing water fountain placement, and making informed decisions about water infrastructure.

CONCLUSION:

The real-time water fountains status system combines technology, data, and public outreach to promote water efficiency, reduce waste, and raise awareness about the importance of sustainable water use and reduced plastic consumption.