Smart Drinking Hardware Water Monitoring System

Subjects: Engineering, Electrical & Electronic

Contributor: Adnan K Shaout , Rosy Shrestha , Shashanka Prajapati

This paper presents an innovative smart drinking water monitoring system designed to automate the refill process for water dispensers and deliver timely filter replacement alerts, specifically tailored for single-user households. The system utilizes sensors to provide real-time monitoring of water quality, measuring temperature, pH levels, and Total Dissolved Solids (TDS). Users can remotely track these parameters—temperature, pH levels, and TDS—through a mobile application powered by Blynk, a low-code Internet of Things (IoT) platform that connects devices. This real-time system ensures users have convenient access to clean drinking water while eliminating the need for manual monitoring of refills and filter replacements. The proposed Smart Drinking Water Monitoring System revolutionizes traditional water dispensers by automating the refill process and supplying data-driven filter replacement reminders in real-time. By continuously tracking water quality through key metrics like temperature, pH levels, and TDS, the system allows users to remotely monitor developments via a mobile or web app integrated with Blynk's IoT platform. Alerts are generated when pH levels or TDS values deviate from the recommended ranges, suggesting potential filter replacements or maintenance actions are needed.

Water Quality Monitoring System | Internet of Things | Blynk | Total Dissolved Solid | Smart Drinking Water System | Nephelometric Turbidity Units

1. Introduction

Access to safe and readily available drinking water is crucial for human health and well-being. Household water filtration systems, such as Brita filters, are commonly used to ensure water quality [1]. However, the manual process of refilling dispensers and replacing filters can be both inconvenient and time-consuming. This is particularly challenging for college students, who are often adapting to a new environment and may be unfamiliar with local water quality while needing consistent access to filtered water.

The Smart Drinking Water Monitoring System aims to resolve these issues by automating the water dispenser refill process and providing real-time, data-driven reminders for filter replacement. The system continuously monitors water quality by measuring temperature, pH levels, and Total Dissolved Solids (TDS). Users can remotely access this data through a mobile or web app integrated with Blynk, an Internet of Things (IoT) platform that connects devices. The system also issues alerts when the pH level or TDS value deviates from recommended ranges, signaling a potential need for filter replacement or maintenance.

By automating the refill process and sending prompt reminders for filter replacements based on water quality insights, the system guarantees users seamless access to clean water without the burden of constant maintenance. Prioritizing health and convenience, this design ensures users have peace of mind regarding the safety of their drinking water.

This paper employs the waterfall software methodology [2], chosen for its suitability in stable requirement environments. The model's step-by-step progression facilitates thorough planning and integration of key features like automated refills and filter reminders, ensuring seamless operation.

2. State of the Art and Literature Review

Several water quality monitoring systems have been previously documented in the literature. For example, [3] describes a wireless system with a self-healing algorithm that uses sensors to gather water quality data and transmit it to the cloud for storage and visualization, accessible via web interface. Although it offers continuous data collection, the paper lacks details on sensor types and specific water quality parameters.

Similarly, [4] outlines a cost-effective approach for monitoring Reverse Osmosis (RO) filter performance using pressure sensors, which notify users of filter changes through LED indicators and SMS messages. However, this system focuses solely on pressure, overlooking other important factors like pH and contaminant levels. Increased pressure is not the sole indicator of membrane blockage.

In [5], a solar-powered water pump control system incorporates water quality monitoring through sensors measuring pH and turbidity, along with water level, using a single-axis solar tracker for efficiency. Despite highlighting renewable energy advantages, the system lacks real-time water quality monitoring.

Another approach, detailed in [6], uses Raspberry Pi for an IoT-based system that collects real-time water quality data to produce a Water Quality Index (WQI), displayed on a web-based map via Google Maps API. Although informative, the paper lacks implementation specifics.

In [7], an IoT-based real-time monitoring system is presented, employing sensors for pH, CO2, water level, and temperature with wireless cloud transmission for remote access. While cost-effective, focusing on general public use for large water sources limits its application in household drinking water quality monitoring.

The proposed Smart Drinking Water Monitoring System stands out by specifically catering to single-user households and domestic dispensers like Brita. Addressing the inconvenience of manual refills, it enhances user experience with features such as automatic refills and real-time notifications. It empowers users to monitor key quality parameters—pH, temperature, and TDS—providing instant updates on their drinking water's condition, including low or high pH levels and filter replacement reminders.

Table 1 provides a concise overview of each system's features and limitations, helping to distinguish their capabilities and the unique advantages of the proposed Smart Drinking Water Monitoring System in this paper.

Table 1. How the new proposed system compares with others

System/Study	Key Features	Limitations
Smart Drinking Water Monitoring System (Proposed)	- Automates refill process and provides real-time, data-driven filter replacement reminders	- Targeted for single-user households only
	- Monitors temperature, pH level, and TDS	
	- Remote monitoring via Blynk mobile/web app	
	- Real-time alerts for water quality deviations	
	- Prioritizes user experience and convenience	
Wireless Water Quality Monitoring [3]	- Collects data via sensors and transmits to a cloud for visualization	- Lacks specific sensor and parameter details
	- Features self-healing algorithm for network disruptions	
	- Continuous data collection	
RO Filter Monitoring with Pressure [4]	- Monitors performance via pressure sensors	- Focuses only on pressure as an indicator

System/Study	Key Features	Limitations
	- Alerts via LED and SMS	- Neglects other performance factors like pH and contaminants
Solar-Powered Pump with Water Monitoring [5]	- Utilizes a solar tracker for energy efficiency	- Does not provide real-time water quality monitoring
	- Measures pH, turbidity, and water level	
	- Automated pump control and alert system	
IoT-Based System with Raspberry Pi [6]	- Uses Raspberry Pi and sensors for real-time data	- Lacks implementation details
	- Converts data into Water Quality Index (WQI)	
	- Displays information via a web-based map	
General IoT-Based Monitoring System [7]	- Measures pH, CO2, water level, and temperature	- Designed for large water sources, not specific to household drinking water
	- Transmits data wirelessly to the cloud	
	- Offers real-time monitoring and alerts	

3. Materials and Method

In the initial phase, a thorough understanding of the needs and objectives for the Smart Drinking Water Monitoring System was established. Single-user households were identified as the target audience, and specific requirements for their water filtration needs were outlined. The proposed system is designed with both functional and non-functional requirements:

3.1. Functional Requirements

Real-Time Water Level Monitoring

The system continuously monitors water levels every second to ensure consistent oversight.

Automatic Refilling

When water levels drop below 15%, the system automatically initiates refilling. Refilling stops once the level reaches approximately 95-100% to prevent overfilling.

Water Quality Monitoring

Notifications are sent when the Total Dissolved Solids (TDS) value exceeds 80 Nephelometric Turbidity Units (NTU) [8], indicating a need for filter replacement.

Filter Replacement Notification

Alerts for filter changes are sent to users via the mobile application.

pH Level Notification

Alerts are issued if the pH level is greater than 8.5 or less than 6.5 [9], indicating extreme pH conditions.

3.2. Non-functional Requirements

Microcontroller Unit

A Raspberry Pi [10] serves as the primary microcontroller due to its availability and user-friendliness, though other microcontrollers may also be suitable alternatives.

Water Level Detection

A non-contact ultrasonic sensor, like the HC-SR04, is employed for precise water level measurement within the filtration container.

Temperature Monitoring

The DS18B20 waterproof temperature sensor is used for accurate water temperature readings.

Programming Language

The system is developed using the Python programming language for its versatility and ease of use.

Mobile App Connectivity

The Blynk [11] app facilitates monitoring and alert notifications on smartphones.

3.3. System Design Overview

The Smart Drinking Water Monitoring System is crafted to automate the monitoring and management of water dispensers, ensuring safe and convenient access to clean drinking water. Unlike existing methods, this system uniquely integrates real-time alerts and automatic refilling specifically for single-user households, combining convenience with precise water quality monitoring. The following sections elaborate on the system's architectural design, functional components, and data flow.

3.4. Architectural Design

The newly designed system architecture employs a modular design, connecting various hardware components with a Raspberry Pi through a Grove Base Hat [12] for streamlined connectivity. This architecture is divided into four primary functional blocks:

Distance Measurement: This module integrates sensors to accurately measure the distance between the Raspberry Pi and target objects or surfaces. It's tailored for applications that require precise spatial measurements, such as monitoring liquid levels in tanks or containers.

TDS (Total Dissolved Solids) Detection*:* This block features sensors to evaluate the concentration of dissolved substances in a liquid, primarily water, providing crucial data for assessing water quality through the measurement of total dissolved ions.

pH Detection: Utilizing specialized sensors, this module measures the pH level of a liquid to determine its acidity or alkalinity, which is vital for applications in environmental monitoring, agriculture, and aquaponics.

Temperature Detection: This essential module measures temperature, delivering important context for understanding environmental conditions and chemical processes.

These functional blocks are designed to seamlessly interface with the Raspberry Pi via the Grove Base Hat [12], facilitating straightforward communication and data exchange. The supporting software processes the sensor data, displays results, and enables user interaction, potentially including alert notifications or control features (**Figure 1**).

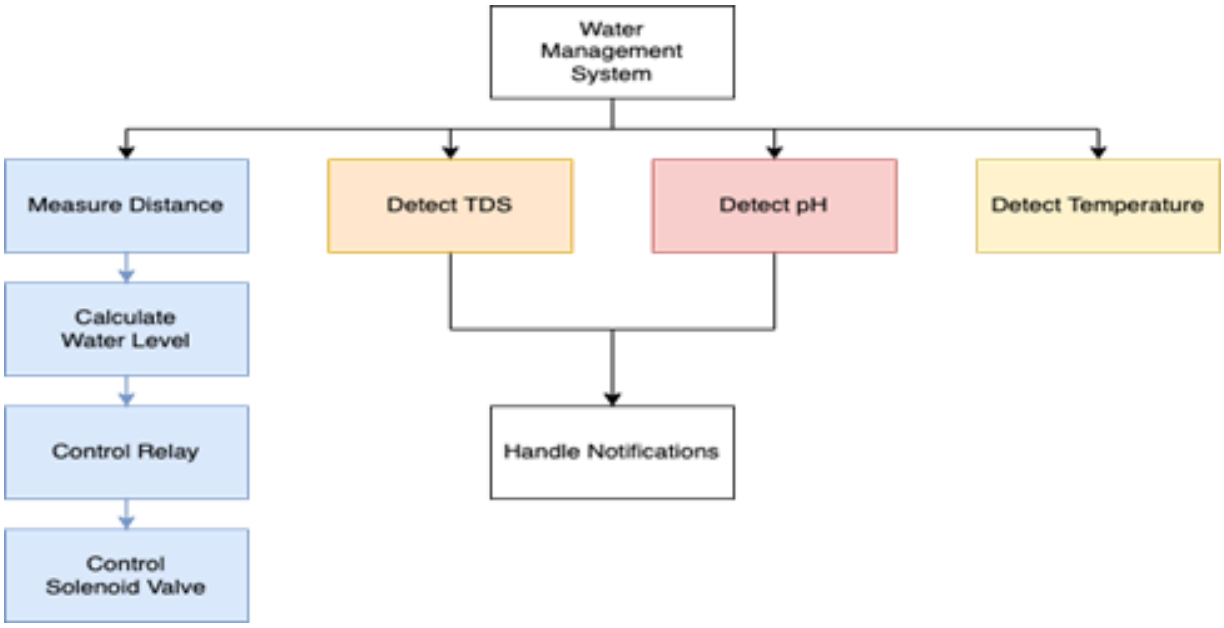

Figure 1. Structure Chart of the system architecture

3.5. Hardware Specifications

Here are the hardware specifications for the system:

1. **Raspberry Pi 4 Model B**: This powerful single-board computer is equipped with a Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC, and provides multiple user-friendly interfaces, including USB ports, micro HDMI output, and a set of GPIO pins that serve as the primary interface for sensor connections. The GPIO pins, which can be programmed for digital input or output, form the backbone of sensor communication and control.
2. **Grove Base Hat for Raspberry Pi**: This expansion board allows easy connection of Grove modules and sensors to the Raspberry Pi by aligning with its GPIO header. The Grove Base Hat organizes the GPIO lines and converts them into standardized connectors that simplify attaching sensors without complex wiring, enhancing both connectivity and reliability. It often includes circuitry for level shifting and protection to ensure signals are compatible with the Raspberry Pi's voltage levels.
3. **Ultrasonic Sensor HC-SR04** [13]: Utilizing digital signaling, this sensor connects to the Raspberry Pi's GPIO pins. It sends a trigger pulse through one GPIO pin and waits for an echo response on another pin to calculate the distance based on the time delay of the returned sound wave.
4. **Temperature Sensor (Waterproof DS18B20)** [14]: This sensor typically connects to the Raspberry Pi through one-wire communication, sharing data over a single GPIO pin. It requires a 4.7 kΩ pull-up resistor to maintain a stable line, providing accurate temperature readings directly in a digital format.
5. **pH Sensor**: The pH sensor generally outputs an analog voltage that corresponds to the pH level. Since the Raspberry Pi lacks an onboard analog-to-digital converter (ADC), the signal from the sensor is routed through an ADC module connected to the GPIO pins to convert the analog signal into digital data that the Raspberry Pi can process.

6. **TDS (Total Dissolved Solids) Sensor**: Similar to the pH sensor, the TDS sensor outputs an analog signal indicative of total dissolved solids concentration. It interfaces with the Raspberry Pi using an external ADC connected to the GPIO pins to digitize the readings.
7. **Solenoid Valve**: This component is controlled by the Raspberry Pi via a relay module. The Raspberry Pi sends a control signal to the relay's GPIO pin, which then either energizes or de-energizes the relay coil, allowing the solenoid valve to open or close.
8. **Relay Module v1.3**: Connected to the Raspberry Pi's GPIO pins, this relay module works by receiving a low-voltage signal from the Raspberry Pi, which is used to switch the relay on and off. This mechanism can control higher voltage devices like the solenoid valve, enabling automation tasks within the system.
9. **Breadboard**: Utilized in prototyping stages, the breadboard allows components to be interconnected without soldering. By using it in conjunction with jumper wires, it facilitates the testing of circuit connections involving multiple components interfaced with the Raspberry Pi.
10. **Jumper Wires (Male/Male, Male/Female)**: Essential for establishing temporary connections, these wires are used to interlink components on the breadboard or directly to the GPIO pins on the Raspberry Pi, enabling flexible and rapid prototyping and assembly of the electronic circuits in the system.

3.6. Software Specifications

3.6.1. Blynk App Integration

The Blynk App is integrated to provide users with remote monitoring capabilities for water quality and dispenser status. The app offers real-time data on water levels, temperature, pH, and TDS values, and sends alerts for filter replacement or maintenance, ensuring proactive water management through its intuitive interface.

3.6.2. Enhanced Data Flow

The system's data flow is structured to begin with sensor readings and culminates in user notifications and control actions. Here's an updated overview of the process:

1. **Advanced Water Level Monitoring**:

- **New Sensor Technology**: A capacitive water level sensor is employed for precision measurement of water level directly, instead of relying on distance calculations.
- **Functionality**: The sensor continually assesses the water level as a percentage and triggers actions based on threshold levels.
- **Control Action**: When the water level drops to 15% or below, it initiates the auto-refill process. When levels reach 100%, it halts the refill function to prevent overflow.

- **Refill Control with Advanced Feedback**:
- Activation occurs at specific low-level thresholds, allowing controlled water flow.
- A feedback mechanism provides real-time confirmation of valve operation, enhancing reliability.
- **Components Used**: An intelligent relay coupled with a feedback-enabled solenoid valve.
- **Process**:

- **Comprehensive Water Quality Assessment**:
- **Updated Sensors**: Enhanced TDS and pH sensors operate with improved sensitivity and accuracy.
- **Data Analysis**: These sensors continuously monitor and evaluate water quality. Deviations from the standard values are promptly identified for quick corrective measures.
- **Proactive Alerting System**:
- Exceedance of TDS levels beyond the tolerated threshold.
- pH levels deviating outside the safe range.
- Alert Triggers:
- User Notification: Automated alerts are dispatched via the Blynk app, informing users of potential maintenance or safety issues, allowing for immediate action.

By adopting new sensor technologies and methods, the system enhances reliability and accuracy in managing water levels and ensuring water quality, while the Blynk App continues to provide critical user engagement and control (**Figure 2**).

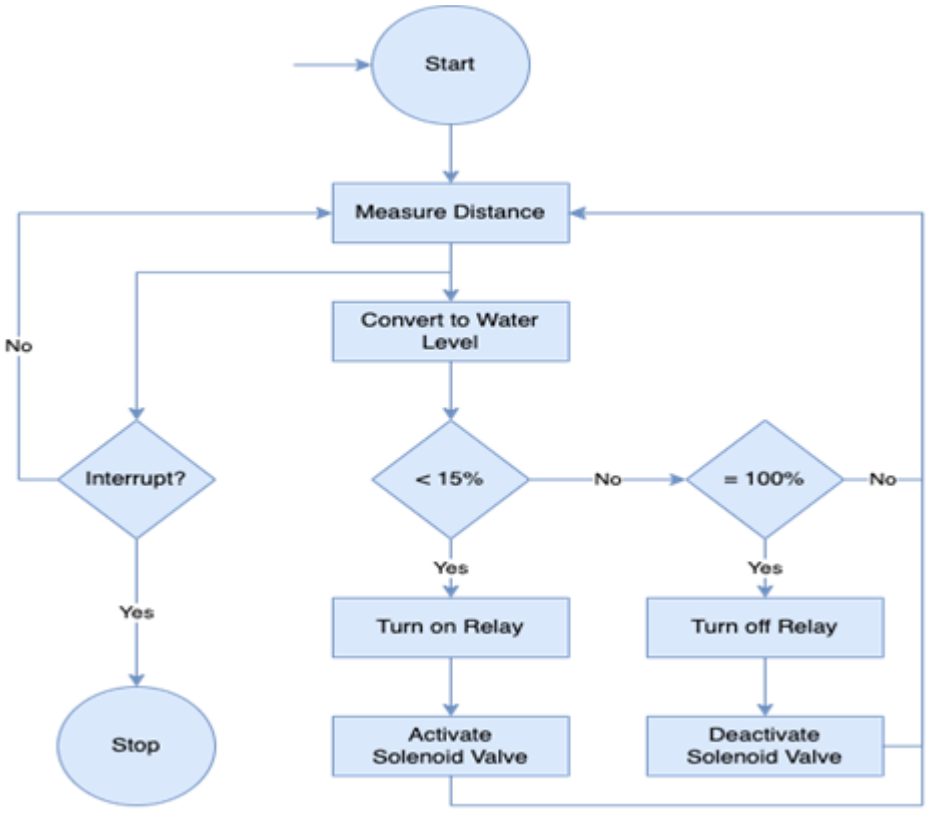

Figure 2. Flowchart for Water Control

Figure 3 illustrates the process of detecting Total Dissolved Solids (TDS) in the water. In this system, if the TDS value exceeds 80 ppm (parts per million), which indicates a high level of impurities, a notification is sent to the user prompting them to change the filter. If the TDS value reaches a further threshold of 200 ppm, it triggers a warning about high turbidity, alerting the user to potential water quality issues that require prompt attention.

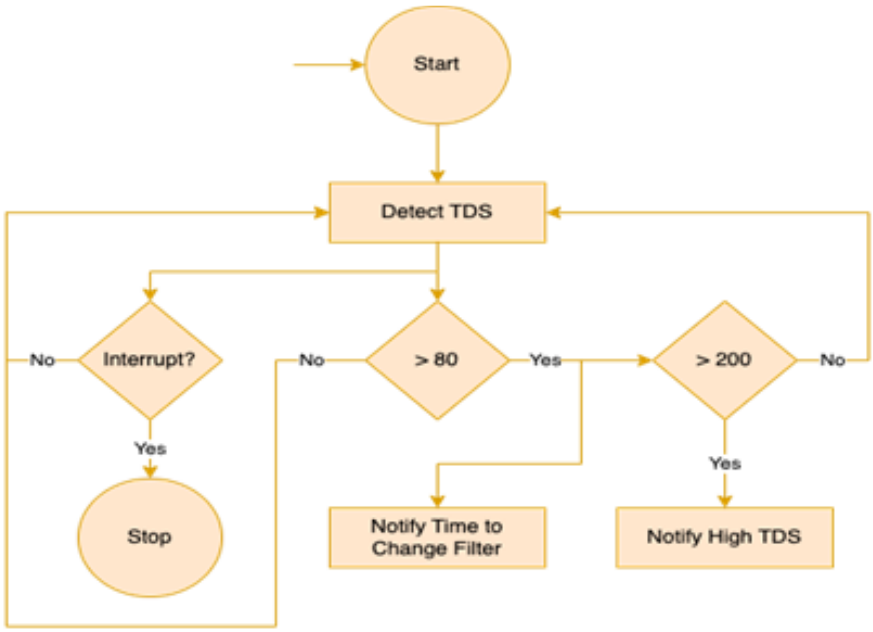

Figure 3. Flowchart for TDS Detection

Figure 4 outlines the steps for analyzing the pH level of the water. If the pH value falls below 6.5, indicating acidity, the system issues a warning about low pH. Conversely, if the pH exceeds 8.5, indicating alkalinity, the system generates a warning about high pH. These alerts inform the user of potential water quality concerns that may require intervention.

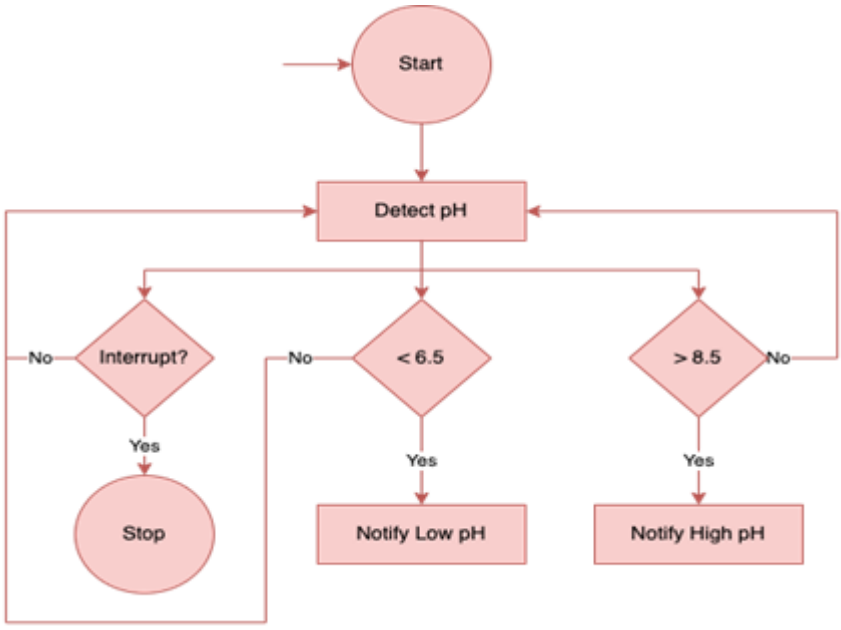

Figure 4. Flowchart for pH Detection

Figure 5 displays the Data Flow Diagram (DFD) of the system, depicting the interaction between the various sensors and system components. The diagram illustrates how the TDS and pH sensors continuously monitor water quality parameters, while the temperature sensor tracks the water temperature. Additionally, the distance sensor gauges the water levels. This visual representation clearly outlines the pathways for data collection and processing

within the system, emphasizing the seamless integration of sensor data for holistic monitoring and effective management.

The following algorithm explains and outlines the sequential steps the system (presented in **figure 5**) follows to collect and process data from various sensors, ensuring comprehensive monitoring and management.

Algorithm: Data Flow in Smart Drinking Water Monitoring System

1. **Initialize System Components:**
- TDS Sensor
- pH Sensor
- Temperature Sensor
- Distance Sensor
- **Start Monitoring Loop:**
1. **Read Water Quality Parameters:**
- Measure TDS levels using TDS Sensor.
- Measure pH levels using pH Sensor.
1. **Record Water Temperature:**
- Capture water temperature using Temperature Sensor.
1. **Check Water Levels:**
- Measure water levels using Distance Sensor.
- **Data Collection:**
- Collect and store readings from all sensors.
- **Data Processing:**
- Analyze TDS and pH values to assess water quality.
- Evaluate temperature readings for anomalies.
- Monitor water level status for refilling requirements.
- **Integration:**
- Compile sensor data for comprehensive system monitoring.
- **End Monitoring Loop** (Continue until the system is shut down).

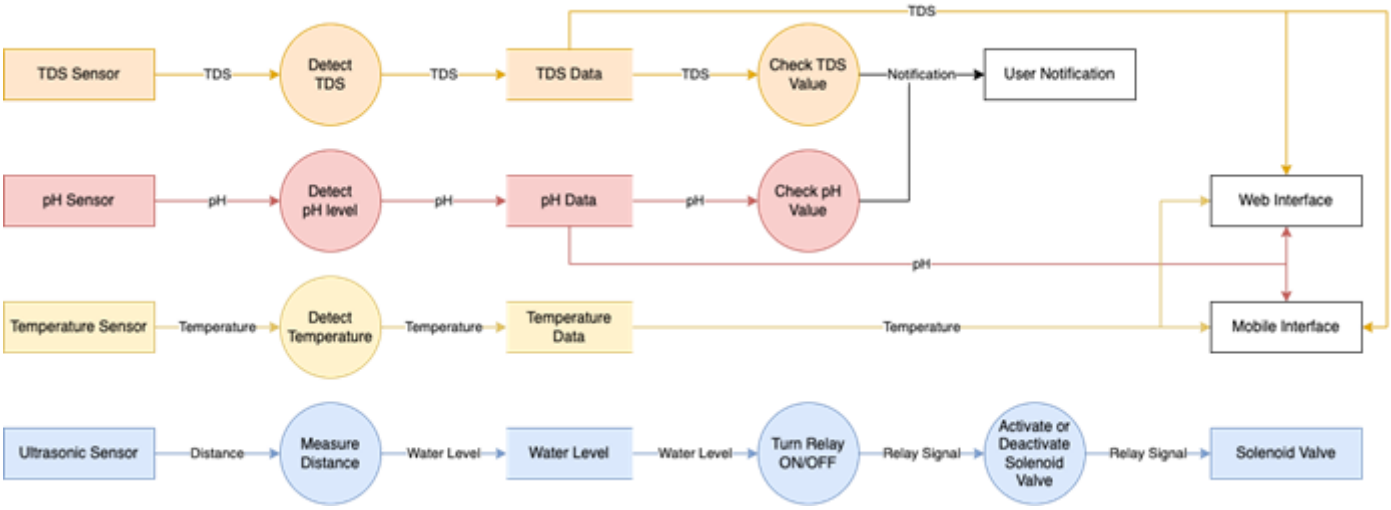

Figure 5. Data Flow Diagram

Implementation Overview

The implementation of the Smart Drinking Water Monitoring System consists of several key phases, including hardware setup, software development, and sensor integration. Below is a detailed breakdown:

Hardware Setup

1. **Central Computing Unit:**

- **Raspberry Pi 4 Model B**: Serves as the central computing hub of the system.
- **Grove Base Hat for Raspberry Pi**: Facilitates connections between the Raspberry Pi and various sensors/actuators.

- **Sensors and Actuators:**
- **Ultrasonic Sensor (HC-SR04)**: Used for measuring water levels.
- **Waterproof Temperature Sensor (DS18B20)**: Monitors water temperature.
- **pH and TDS Sensors**: Evaluate water quality parameters.
- **Solenoid Valve and Relay Module**: Automates the refill process based on sensor inputs.

- **Prototyping:**
- A **breadboard** and **jumper wires** were used during prototyping to ensure seamless integration of components.

Challenges and Solutions

1. **Sensor Compatibility and Calibration Issues:**

- **Challenge**: The pH sensor required a 5V input, causing startup issues with the Raspberry Pi, which operates on a lower voltage.
- **Solution**: Disconnected the pH sensor during the Raspberry Pi's startup phase, then reconnected it afterward for stable data collection, thereby preventing voltage conflicts.

Circuit Diagram

1. **Overview:**

- **Figure 6** illustrates the circuit diagram, providing a visual depiction of the system's hardware architecture.
- **Connections:**
- The diagram shows how the Raspberry Pi interfaces with various sensors, actuators, and electronic components, which is essential for accurate assembly and wiring.
- **Troubleshooting and Maintenance:**
- Acts as a reference guide for identifying and resolving potential hardware issues, assisting in troubleshooting and maintenance.

In summary, this structured approach to hardware setup, combined with strategic solutions to compatibility issues, establishes a robust foundation for the implementation of the Smart Drinking Water Monitoring System (**Figure 6**).

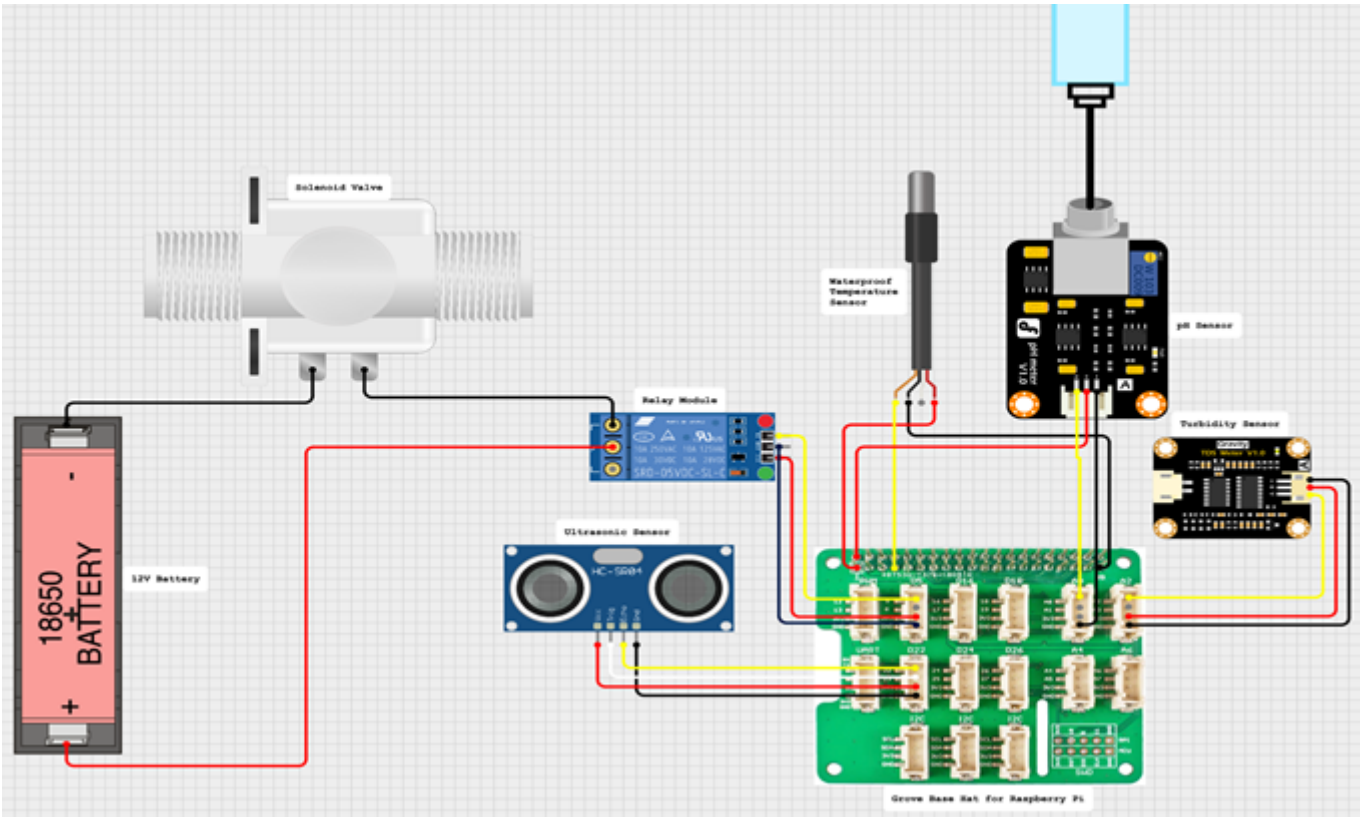

Figure 6. Circuit Diagram for the hardware architecture of the proposed Smart Drinking Water Monitoring System

Detailed Hardware Setup and Connections

The following explicit labeling of the pin connections and power details ensures accuracy in the hardware setup, facilitating effective replication and troubleshooting of the system.

Central Computing Unit

Raspberry Pi 4 Model B:

Power Supply: Uses a 5V/3A power adapter connected to the USB-C power port.

GPIO Pins: Connects to sensors and actuators through the GPIO (General-Purpose Input/Output) pins using the Grove Base Hat.

Sensors and Actuators Connections

Ultrasonic Sensor (HC-SR04):

VCC (Power): Connects to the 5V pin on the Raspberry Pi.

Trig (Trigger): Connects to GPIO Pin 23.

Echo: Connects to GPIO Pin 24.

GND (Ground): Connects to a Ground pin on the Raspberry Pi.

Waterproof Temperature Sensor (DS18B20):

VCC (Power): Connects to the 3.3V pin on the Raspberry Pi.

Data: Connects to GPIO Pin 4 (requires a 4.7k ohm resistor between VCC and Data lines).

GND (Ground): Connects to a Ground pin on the Raspberry Pi.

pH Sensor:

VCC (Power): Initially not connected during boot-up; connected to the 5V pin after startup.

Signal: Connects to an available GPIO pin as an analog signal interface.

GND (Ground): Connects to a Ground pin on the Raspberry Pi.

TDS Sensor:

VCC (Power): Connects to the 5V pin on the Raspberry Pi.

Signal: Connects to a GPIO pin used for reading analog signal inputs.

GND (Ground): Connects to a Ground pin on the Raspberry Pi.

Solenoid Valve and Relay Module:

Relay Control Pin: Connects to an available GPIO pin (e.g., GPIO Pin 17) for controlling the valve.

VCC and GND (Power): Connects to the 5V and GND pins on the Raspberry Pi for powering the relay module.

Additional Details

· **Breadboard and Jumper Wires:**

Used to create connections between the sensors, relay module, and Raspberry Pi, ensuring flexible and secure hookups.

· **Grove Base Hat for Raspberry Pi:**

Simplifies attachment to the Raspberry Pi and organizes connections, particularly useful for beginners in electronics.

3.7. Software Development

The software development process for the Smart Drinking Water Monitoring System focused on Python programming, chosen for its simplicity and compatibility with Raspberry Pi. The implementation leveraged several key libraries:

- **Grove Library:** Essential for interfacing with Grove modules and sensors.
- **GPIO Library:** Facilitated control over the GPIO pins, enabling interaction with sensors and actuators.

System Logic

The core logic of the system revolves around automating water dispenser management while ensuring the provision of safe and clean drinking water. Key functionalities include:

- **Water Level Monitoring:**
  - Utilizes an ultrasonic sensor to continually measure the distance to the water surface. This allows tracking of water volume changes over time.
  - When water levels fall below a predetermined threshold, the system triggers an automated refill process.
- **Automated Refill Process:**
  - Activates a solenoid valve connected to a water source when a refill is needed, allowing water to flow until the desired level is reached.
  - Ensures a continuous supply of filtered water without manual intervention.
- **Water Quality Monitoring:**
  - Employs pH and TDS sensors to monitor water quality parameters, such as acidity and mineral content.
  - Continuously assesses the cleanliness and purity of the filtered water.
- **Filter Replacement Notifications:**
  - Generates alerts for users when water quality degrades or filters are nearing the end of their lifespan, prompting timely maintenance actions.

User Experience

The system design prioritizes user convenience and safety by:

- Automating refill processes.
- Providing real-time monitoring of water quality.
- Issuing timely reminders for filter replacement.

These features ensure users have peace of mind regarding their drinking water quality, enhancing the overall experience for both single-user households and communities.

Sensor Calibration

- **pH Sensor Calibration:**
  - Output from the pH sensor in volts necessitated calibration. Analog voltage readings were mapped to known pH values through experimentation and data analysis.
  - This calibration allowed the system to interpret sensor data accurately and provide users with reliable water quality insights.

Here is a more detailed explanation of the pH sensor calibration strategy, including the methodology, buffer solutions used, and frequency of recalibration:

3.8. pH Sensor Calibration Strategy

Objective:
The calibration of the pH sensor is crucial to ensure accurate and reliable readings of water quality. Calibration involves adjusting the sensor to recognize standard pH values accurately, which helps in maintaining precise monitoring of the water's acidity or alkalinity.

Methodology:

1. **Selection of Buffer Solutions:**

- **Buffer solutions** are used as reference standards with known pH values to calibrate the sensor. Typically, three standard buffer solutions are used for a wide pH range:
  - **pH 4.0 Buffer Solution:** Represents the acidic range.
  - **pH 7.0 Buffer Solution:** Denotes a neutral pH, often used as the midpoint reference for calibration.
  - **pH 10.0 Buffer Solution:** Represents the basic or alkaline range.
- These standardized solutions provide fixed reference points for recalibrating the sensor to ensure accuracy across the pH spectrum.

2. **Calibration Procedure:**

- **Step 1:** Start by thoroughly rinsing the pH sensor probe with distilled water to remove any contaminants.
- **Step 2:** Immerse the pH sensor in the pH 7.0 buffer solution and allow it to stabilize. Adjust the calibration trimmer or enter the calibration mode on the interface to set the readout to 7.0.

- **Step 3:** Rinse the sensor again with distilled water, then immerse it into the pH 4.0 buffer solution. Once stabilized, adjust the pH meter to read 4.0.
- **Step 4:** Rinse once more and immerse it into the pH 10.0 buffer solution, adjusting the final calibration to read 10.0.
- **Step 5:** After calibration, rinse the probe and store it properly to maintain its condition.

3. **Recalibration Frequency:**

- **Routine Recalibration:** It is generally recommended to recalibrate the pH sensor once every week to every two weeks, depending on usage frequency and the environment.
- **After Specific Events:** Recalibration should also be performed after cleaning the sensor, replacing the probe, or if there are noticeable discrepancies in the readings.
- **Seasonal Checks:** Alternatively, more frequent recalibrations may be conducted during periods of significant environmental changes (e.g., temperature shifts).

Employing a systematic calibration process with standardized buffer solutions ensures the accuracy and reliability of the pH sensor readings. Regular recalibration helps maintain consistent monitoring of water quality, enabling timely interventions if the water's pH levels deviate from acceptable ranges.

3.9. Code and Implementation

Code Examples: Figures 7, **8**, **9**, **10,** and **11** showcase code snippets that encapsulate the system's major functionalities, offering insights into the implementation details.

```
class WaterManagementSystem:
    """
    A class to manage the water system, including monitoring and controlling
    water level, quality, and temperature using various sensors connected to a
    Raspberry Pi.

    Attributes:
        trig_pin (int): GPIO pin connected to the ultrasonic sensor's trigger.
        echo_pin (int): GPIO pin connected to the ultrasonic sensor's echo.
        tds_pin (int): ADC channel connected to the TDS (Total Dissolved
            Solids) sensor.
        ph_pin (int): ADC channel connected to the pH sensor.
        relay_pin (int): GPIO pin connected to the relay for controlling water
            flow.
    """

    PH_CALIBRATION_DATA = {
        430: 2.70,  # Analog reading 430 corresponds to pH 2.70
        580: 7.00,  # Analog reading 580 corresponds to pH 7.00
    }

    def __init__(
        self,
        trig_pin: int,
        echo_pin: int,
        tds_pin: int,
        ph_pin: int,
        relay_pin: int,
        blynk_auth_token: str,
    ):
        self.trig_pin = trig_pin
        self.echo_pin = echo_pin
        self.tds_pin = tds_pin
        self.ph_pin = ph_pin
        self.relay = OutputDevice(relay_pin)
        self.adc = ADC()
        self.blynk = BlynkLib.Blynk(blynk_auth_token, server="blynk.cloud")
        self.timer = BlynkTimer()

        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.trig_pin, GPIO.OUT)
        GPIO.setup(self.echo_pin, GPIO.IN)

        self.setup_timers()
```

Figure 7. Initialization of Water Management System Class

Figure 7 shows the code for the initialization of water management system class. The purpose of this figure is to show the code for the setup and initialization of the primary class responsible for managing the system's operations.

The components in this code are as follows:

Constructor Method: Initializes key variables and sets up sensor objects, GPIO pins, and initial system states.

Configuration: Includes setting up default values for the sensors, actuators, and any initial conditions required for the system to begin operating correctly.

Comments: Ensure the constructor comments on required libraries, initialization steps for Raspberry Pi GPIO, and setup for communication interfaces if necessary (e.g., I2C, SPI).

```python
def measure_distance(self) -> float:
    """Measure distance using ultrasonic sensor."""
    GPIO.output(self.trig_pin, True)
    time.sleep(0.00001)
    GPIO.output(self.trig_pin, False)

    pulse_start = time.time()
    while GPIO.input(self.echo_pin) == 0:
        pulse_start = time.time()

    pulse_end = time.time()
    while GPIO.input(self.echo_pin) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150  # Speed of sound = 343 m/s = 17150 cm/s
    return round(distance, 2)

def calculate_water_level(
    self,
    current_distance: float,
    max_distance: float = 100.0,
    min_distance: float = 0.0,
) -> float:
    """Calculate the fill percentage based on the current distance."""
    if current_distance < min_distance:
        return 100.0
    elif current_distance > max_distance:
        return 0.0
    else:
        return round(
            (1 - (current_distance - min_distance) / (max_distance - min_distance))
            * 100,
            2,
        )
```

Figure 8. Distance and water level functions

Figure 8 shows the code for the distance and water level functions. The purpose of this figure is to show the functions managing ultrasonic sensor inputs to determine the water levels.

The components in this code are as follows:

Distance Measurement: Code to send a pulse from the ultrasonic sensor and listen for the echo, calculating the distance based on time taken.

Water Level Calculation: Converts the measured distance into a percentage of the tank's fullness.

Comments: Describe the logic for the time-to-distance conversion and ensure handling of edge cases like sensor malfunction or unexpected readings.

```python
def read_analog(self, pin: int) -> int:
    """Read analog value from specified ADC port."""
    return self.adc.read(pin)

def read_tds_value(self) -> int:
    """Read TDS value from the sensor."""
    return self.read_analog(self.tds_pin)

def read_ph_value(self) -> float:
    """Calculate pH value from analog input based on calibration data."""
    ph_analog_reading = self.read_analog(self.ph_pin)
    slope = (self.PH_CALIBRATION_DATA[580] - self.PH_CALIBRATION_DATA[430]) / (
        580 - 430
    )
    ph_value = self.PH_CALIBRATION_DATA[430] + slope * (ph_analog_reading - 430)
    return max(min(round(ph_value, 1), 14), 0)

def read_temperature(self) -> float:
    """Read temperature from DS18B20 sensor."""
    sensor = W1ThermSensor()
    return sensor.get_temperature()
```

Figure 9. Functions to read data from sensors

Figure 9 shows the code for the functions to read data from sensors. The purpose of this figure is to show the functions facilitating the reading and processing of data from pH, TDS, and temperature sensors.

The components in this code are as follows:

Analog and Digital Reads: Functions leveraging the Raspberry Pi's ADC for obtaining readings from the sensors.

Data Validation: Logic to filter and validate sensor data, ensuring accuracy.

Comments: Clarify calibration techniques applied in the code and note any conversion formulas used to translate raw data into meaningful measurements.

```
def check_water_level(self):
    """Check and manage the water level."""
    distance = self.measure_distance()
    water_level = self.calculate_water_level(distance, 8.5, 3)
    print(f"Water Level: {water_level}%")
    self.blynk.virtual_write(0, water_level)

    if water_level <= 15:
        self.relay.on()
    elif water_level == 100:
        self.relay.off()

    relay_status = "ON" if self.relay.is_active else "OFF"
    print(f"Relay turned {relay_status}")
    self.blynk.virtual_write(4, int(self.relay.is_active))

def check_water_quality(self):
    """Check water quality and update Blynk app."""
    print("=" * 80)
    tds_value = self.read_tds_value()
    print(f"TDS Value: {tds_value}")
    self.blynk.virtual_write(2, tds_value)

    if tds_value > 80:
        self.blynk.log_event("filter_change", "Time to change your filter")
    if tds_value > 200:
        self.blynk.log_event("high_tds", f"High turbidity detected: {tds_value}")

    ph_value = self.read_ph_value()
    print(f"pH Value: {ph_value}")
    self.blynk.virtual_write(1, ph_value)

    if ph_value != 0 and ph_value < 6.5:
        self.blynk.log_event("low_ph", f"Low pH level detected: {ph_value}")
    if ph_value > 8.5:
        self.blynk.log_event("high_ph", f"High pH level detected: {ph_value}")

    try:
        temperature = self.read_temperature()
        print(f"Temperature: {temperature:.2f}°C")
        self.blynk.virtual_write(3, temperature)
    except:
        print("Temperature Sensor not working")
    print("=" * 80)

def setup_timers(self):
    """Set up timing for routine checks."""
    self.timer.set_interval(1, self.check_water_level)
    self.timer.set_interval(3, self.check_water_quality)
```

Figure 10. Control functions, Blynk app connections and timer setup.

Figure 10 shows the control Functions, Blynk App connections, and timer setup. The purpose is to manage the actuator control, integrate with Blynk app for user interface, and configure timing operations for periodic tasks.

The components in this code are as follows:

Relay Control: Functions to activate solenoids or pumps based on water levels or quality thresholds.

App Connectivity: Blynk API calls to update the user interface and push notifications.

Timers: Setup for functions that need to run at defined intervals (e.g., periodic sensor readings).

Comments: Explain the control logic and clarify authentication methods with Blynk to ensure connection security.

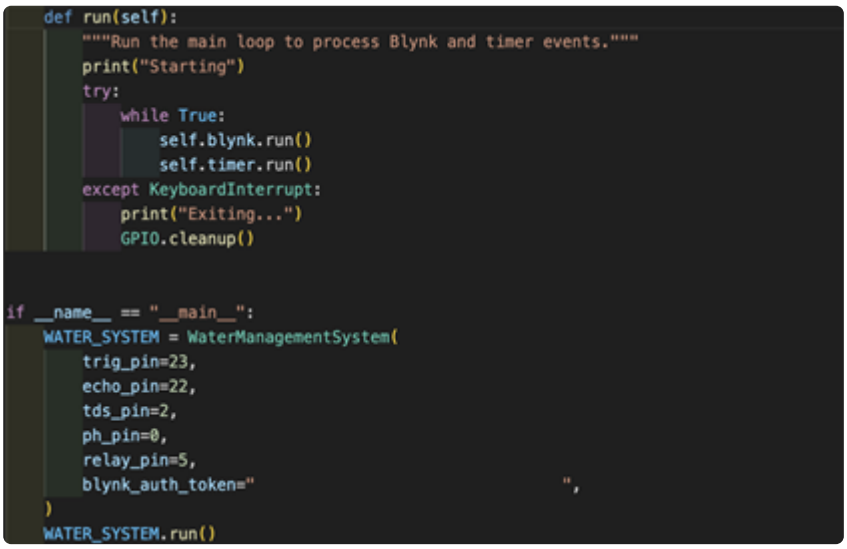

Figure 11. Main loop.

Figure 11 shows the main loop. The purpose of the code is to illustrates the primary execution loop where system functions are orchestrated.

The components in this code are as follows:

Continuous Monitoring: Loop conditions to repeatedly check sensor data and maintain system operation.

Event Handling: Responses to changes in data, triggering alerts or refilling processes when needed.

Comments: Highlight critical decision-making processes and safeguard mechanisms to handle exceptions or operational errors.

By structuring the code this way, the Smart Drinking Water Monitoring System can effectively integrate its hardware and software components, offering coherent functionality and a seamless user experience. Furthermore, the decision-making logic ensures water quality and availability, enhancing the system's reliability.

Function Overview: Table 2 lists all the functions employed in implementing the system, serving as a comprehensive resource for understanding how the system operates.

By incorporating these sophisticated functionalities and a user-centric design, the Smart Drinking Water Monitoring System offers a reliable and effective solution for ensuring access to clean and safe drinking water.

Table 2. Function table for the implantation of the proposed system

No	Function	Input	Output	Description

1	measure_distance	Ultrasonic Sensor Pins	Distance	Provide the distance of the closest surface from the ultrasonic sensor
2	calculate_water_level	Current Distance Maximum Distance Minimum Distance	Percentage of Water Level	Convert the distance parameter to water level percentage based on the custom min and max height of the filter as specified
3	read_analog	Analog Port	Analog Reading	Provide the analog reading from the sensor connected to the given port
4	read_tds_value	TDS Pin	TDS Value	TDS reading from the sensor
5	read_ph_value	pH Pin	pH Value	Calibrated pH reading
6	read_temperature	n/a	Temperature value	Temperature reading in Celsius
7	check_water_level	n/a	n/a	Control relay based on the water level and send water level data to the Blynk app
8	check_water_quality	n/a	n/a	Send pH and TDS readings to the Blynk app and control notifications
9	setup_timers	n/a	n/a	Setup time intervals at which to run our functionalities
10	run	n/a	n/a	The main looping function

4. Results and Discussion

The hardware components were assembled according to the system design, and software modules underwent thorough testing to ensure precise sensor readings and reliable actuator control.

4.1. Testing Process

- **Unit Testing:**
  - **Ultrasonic Sensor:** Tested for its ability to accurately measure distance and convert this measurement into a reliable water level percentage.
  - **pH and TDS Sensors:** Assessed for their functionality in converting raw readings into valid pH and TDS values.
- **Integration Testing:**
  - Simulated varying water level readings to check if the ultrasonic sensor correctly triggered the relay module to initiate the refill process at the threshold level of 15%.
  - Verified that the system ceased refilling operations upon reaching the full level of 100%.

4.2. Results

The results demonstrate the effectiveness of the proposed Smart Drinking Water Monitoring System prototype:

- **Web App Interface (Figure 12):** Provides an overview of the proposed system's interface, showing its functionality and user interaction capabilities.
- **Mobile App Interface (Figure 13):** Displays a mobile view of the system, highlighting its accessibility and ease of use.
- **pH Notifications (Figure 14):** Illustrates the system's ability to send alerts for both low and high pH levels, ensuring users are informed of water quality issues.
- **Turbidity and Filter Change Notifications (Figure 15):** Shows alerts related to turbidity and necessary filter changes, keeping users updated on maintenance needs.

4.3. System Comparison

- **Comparison Table (Table 3):** Provides a comparative analysis between existing systems and the proposed smart water monitoring system. The final column highlights the advantages of the proposed system, which include:
  - **Automatic Refilling:** Ensures continuous water availability without manual intervention.
  - **Real-Time Alerts:** Keeps users informed of water quality and system status.
  - **User-Friendly Readings:** Offers intuitive and easily interpretable data for users.

Overall, the proposed system stands out for its automation, real-time monitoring capabilities, and user-friendly design, making it a robust solution for managing drinking water quality and availability.

Each caption focuses on how the system leverages real-time readings and alert notifications to provide a comprehensive, user-friendly monitoring and management solution. Here are enhanced captions for each figure,

emphasizing key components, real-time readings, and alert notifications:

Figure 12 presents the web app interface of the Smart Drinking Water Monitoring System. It showcases the system's real-time monitoring capabilities, allowing users to access an intuitive dashboard that displays live water quality parameters, such as pH, temperature, and TDS levels. The interface enhances user interaction by providing immediate feedback and comprehensive data visualization to aid in informed decision-making regarding water quality.

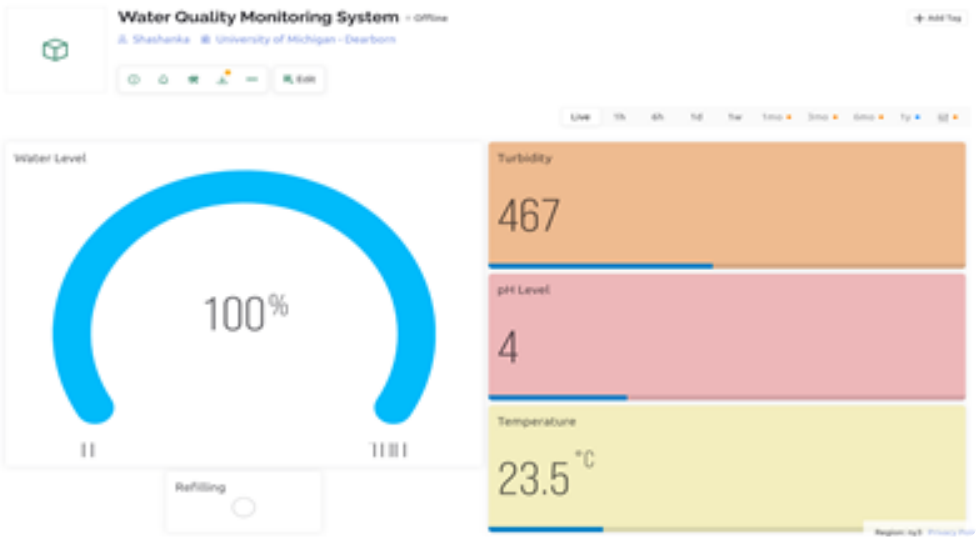

Figure 12. Web app interface

Figure 13 illustrates the mobile app interface designed for user convenience and accessibility. The mobile platform offers real-time updates on water quality metrics, including live alerts for any changes in water levels or quality that demand attention. The streamlined design ensures that users can effortlessly monitor system status, receive notifications, and manage system operations from anywhere, adding a layer of flexibility and control to the user experience.

Figure 13. Mobile App interface

Figure 14 depicted here is the system's alert mechanism for pH monitoring, demonstrating its capability to issue real-time alerts for abnormal pH levels. Notifications are sent directly to the user when pH readings fall outside safe ranges, ensuring timely awareness and allowing users to take immediate corrective action. This feature underscores the system's commitment to maintaining optimal water quality and user health.

Figure 14. pH notifications

Figure 15 highlights the alert system for turbidity and filter maintenance. It provides a detailed view of how the system broadcasts notifications for elevated turbidity levels and necessary filter changes. These alerts are essential for proactive maintenance, informing users promptly to uphold both water clarity and filtration efficiency. The notifications system enhances user engagement and accountability, ensuring a high standard of water quality management.

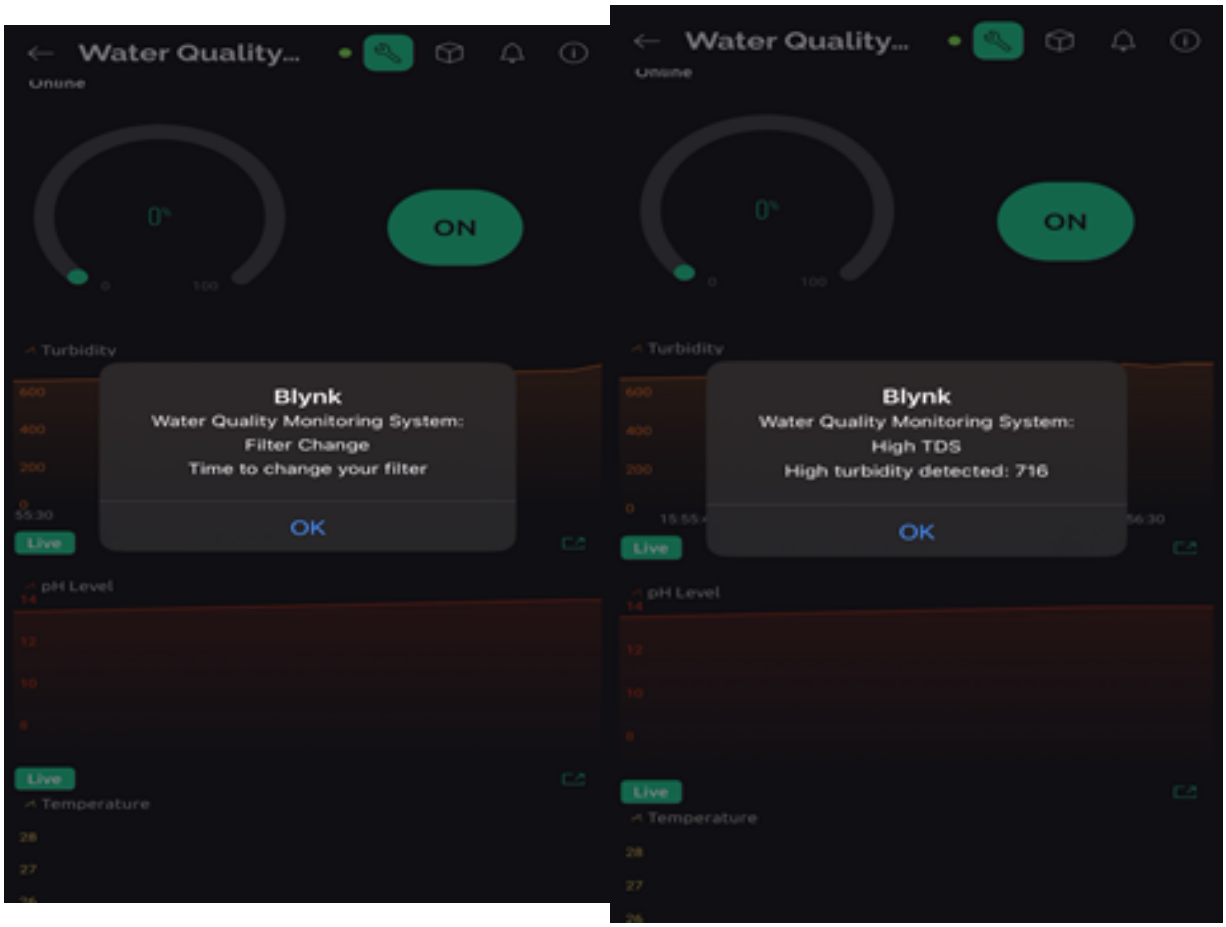

Figure 15. Turbidity and filter change notifications

Table 3. Table of comparison between existing and our proposed system

Features	Wireless water quality cloud monitoring system with a self-healing algorithm	IoT Enabled RO Water Filter Indicator	Solar-Based Smart Water Pump Control with Turbidity and pH Measuring System	Water Quality Index Using IoT	Real-Time Water Quality Tracking and Alert System with IoT Integration	Our Proposed Smart Drinking Water Monitoring System
Goal	Remote monitoring, system health	Monitoring RO filter performance	Precision agriculture, water	Provide overall water	Real-time monitoring of water from	Real-time monitoring of drinking water and data-

			quality control	quality index	general sources	driven filter replacement reminders
Technology	Arduino UNO; Atlas Scientific Full Water Monitoring Kit	Reverse Osmosis, Pressure gauge	Solar panel, Arduino, sensors	Raspberry Pi, Water quality sensors	Raspberry Pi, IoT	Raspberry Pi, Blynk, IoT
Monitored parameters	Oxidization-Reduction Potential, pH, Electrical Conductivity, and temperature	RO membrane health, pressure on the membrane	Turbidity, ph.	Water Quality Index	Ph., CO2, Water level, temperature	Water level, Temperature, Ph., Total Dissolved Solid
Target application / Users	To use a self-healing algorithm to recover in case of server disruption during real-time data collection	Residential use using RO filters	Household	To control water pollution	Alert people in local areas about water quality in their areas	Single household / international students using small water dispensers (Brita filter)
Water source focus	water stream in University Technology Malaysia	Reverse Osmosis (RO) filtration system	Surface water	General water source	Water lake, water bore-well	Drinking water

Alert mechanism	No alert; Graphs and tables shown on the website	LED indicator and beep sound and SMS	LCD and message notification using Arduino GSM	No alert	Yes, available on the cloud as a message	Notification on mobile app
Refill mechanism	No	No	Yes	No	No	Yes
Advantage	The system recovers itself in case of wireless service connection failure	Filter maintenance alerts	Operated by solar power to operate the water pump	The Water Quality Index of different regions can be seen with Google Maps API	Minimize collecting water samples and sending them back to the lab for testing	Automatic refilling, real-time alert, user-friendly readings
Limitation	Needs wireless connection	Limited to Ro systems	Light-dependent resistors are easily affected by dust	Requires expertise, WQI may not be user-friendly	The testing environment is not specified	Proof of concept with simulation

The development of the Smart Drinking Water Monitoring System marks a significant step toward automating and enhancing water quality management. However, there are exciting possibilities for future research and development that could further refine and expand its capabilities. One promising direction is the integration of AI-based predictive maintenance. By analyzing patterns in sensor data, AI algorithms could predict when components like filters and sensors are likely to fail or require maintenance, thereby preventing issues before they arise and improving system reliability and longevity. Another potential avenue is the incorporation of cloud-based analytics, which would enable more sophisticated data handling and analysis. By storing data in the cloud, the system could leverage advanced analytics tools to provide users with deeper insights into water quality trends and anomalies

over time. This approach could support real-time decision-making as well as long-term strategic planning by offering predictive insights into potential water quality issues. Exploring these future research directions could significantly enhance the functionality and value of the Smart Drinking Water Monitoring System, making it an even more robust resource for users in monitoring and maintaining the quality of their drinking water. These advancements would not only improve the technical aspects of the system but also contribute to increased user empowerment and satisfaction through better-informed water management.

5. Conclusion

This paper presents a real-time system that effectively integrates various sensors to monitor water quality, including temperature, pH levels, and TDS. Users can easily access this data and receive alerts on their mobile devices through the Blynk Internet of Things app. By eliminating the need for manual refilling and filter replacement monitoring, the system promotes convenient access to clean drinking water and has the potential to improve user health outcomes.

5.1. Future Enhancements

While this paper establishes a functional prototype, there is room for further development and exploration:

- **Expanded Sensor Integration:** Future versions could incorporate additional sensors, such as those detecting chlorine and conductivity, offering a more comprehensive assessment of water quality.
- **Cloud Platform Integration:** Leveraging cloud platforms for data storage and analysis would enable users to track historical water quality trends and gain insights into changes over time. This would also allow for remote access to system data, enhancing convenience.

5.2. Potential Challenges

As the system evolves, several challenges may need to be addressed:

- **Sensor Reliability Over Time:** Ensuring long-term accuracy and performance of sensors is critical. Regular calibration and maintenance will be necessary to maintain reliability.
- **Water Contamination Risks:** The system must be equipped to promptly detect and alert users to potential contamination events to ensure safety.
- **User Adoption Barriers:** Encouraging widespread adoption may require addressing factors such as system cost, ease of use, and user education on the benefits of monitoring water quality.

By addressing these challenges and exploring future enhancements, the Smart Drinking Water Monitoring System can develop into a comprehensive, user-centric solution. It offers the potential to significantly impact the availability and quality of drinking water in single-user households and beyond.

References

1. Brita. Municipal clean water initiative 2.0. Available from: https://www.brita.com/landing/municipal-clean-water-initiative-2.0/. Accessed January 29, 2025.
2. Singhto W, Phakdee N. Adopting a combination of Scrum and Waterfall methodologies in developing Tailor-made SaaS products for Thai Service and manufacturing SMEs. 2016 International Computer Science and Engineering Conference (ICSEC); 2016. p. 1-6.
3. Ariffin SHS, Baharuddin MA, Fauzi MHM, Latiff NMA, Yusof SKS, Latiff NAA. Wireless water quality cloud monitoring system with self-healing algorithm. 2017 IEEE 13th Malaysia International Conference on Communications (MICC); 2017 Nov 28-30; Johor Bahru, Malaysia. Piscataway, NJ: IEEE; 2017. p. 218-223. doi: 10.1109/MICC.2017.8311762.
4. Sohanpal PK, Sarao PS, Goyal P, Trivedi NK, Tiwari RG. IoT Enabled RO Water Filter Indicator. 2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART); 2022. p. 920-925. doi: 10.1109/SMART55829.2022.10047552.
5. Shovo SH, Karmarker S. Solar Based Smart Water Pump Control with Turbidity and pH Measuring System. 2023 3rd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST); 2023. p. 223-227. doi: 10.1109/ICREST57604.2023.10070048.
6. Verma R, Ahuja L, Khatri SK. Water Quality Index Using IoT. 2018 International Conference on Inventive Research in Computing Applications (ICIRCA); 2018. p. 149-153. doi: 10.1109/ICIRCA.2018.8597357.
7. Ch MK, Munaga MSK, Kolli CS, Maddila SK. Real-Time Water Quality Tracking and Alert System with IoT Integration. 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN); 2023. p. 1013-1018. doi: 10.1109/ICPCSN58827.2023.00172.
8. YSI. Understanding turbidity, TDS, and TSS. Available from: https://www.ysi.com/ysi-blog/water-blogged-blog/2022/05/understanding-turbidity-tds-and-tss. Accessed January 29, 2025.
9. Sruthi M. What is the best pH level for drinking water? MedicineNet. 2021 Oct 19. Available from: https://www.medicinenet.com/what_is_the_best_ph_level_for_drinking_water/article.htm
10. Raspberry Pi. Raspberry Pi official website. Available from: https://www.raspberrypi.com/. Accessed January 21, 2025.
11. Blynk.io. Introduction to Blynk Documentation. Available from: https://docs.blynk.io/en. Accessed January 21, 2025.
12. Seeed Studio. Grove Base Hat for Raspberry Pi. Available from: https://wiki.seeedstudio.com/Grove_Base_Hat_for_Raspberry_Pi/. Accessed January 29, 2025.
13. ElecFreaks. Ultrasonic Ranging Module HC-SR04. 2011. Available from: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. Accessed January 29,

2025.

14. Quick-teck. Quick-teck Electronics Components datasheet. Available from: https://www.quick-teck.co.uk/Management/EEUploadFile/1420644897.pdf. Accessed January 29, 2025.