Date: 13/06/2023

Water Level Monitoring System

OBJECTIVE

To monitor the water levels using a water level sensor connected to clients and send a warning message to the server when water level is high and providing remote control over a buzzer.

ABSTRACT

Discover a sublime water level monitoring system that seamlessly integrates crucial functionalities for efficient water level management. Through a sophisticated client-server setup, the system promptly notifies the server of threshold breaches. The server responds with graceful alerts, melodious notifications, and remotely activated buzzers. This enchanting system empowers users to remotely control the buzzers, offering unparalleled convenience. Experience the harmonious convergence of elegance and effectiveness, ensuring safety and ease in flood prevention, tank monitoring, and industrial processes.

SYSTEM OVERVIEW & METHODOLOGY

SYSTEM OVERVIEW

A Supervisory Control and Data Acquisition (SCADA) system is implemented in the project using a client-server architecture. The system consists of a central server (laptop) with two buttons and two remote clients (Two Raspberry Pi) equipped with water-level sensors, and buzzers. The SCADA system enables real-time monitoring and control of water levels and alarms.

The SCADA system follows the typical architecture with the server acting as the central control and data processing unit, while the clients function as remote units collecting data and sending it to the server for analysis and control.

The system enables real-time water level monitoring, alarm notifications, and remote buzzer control.

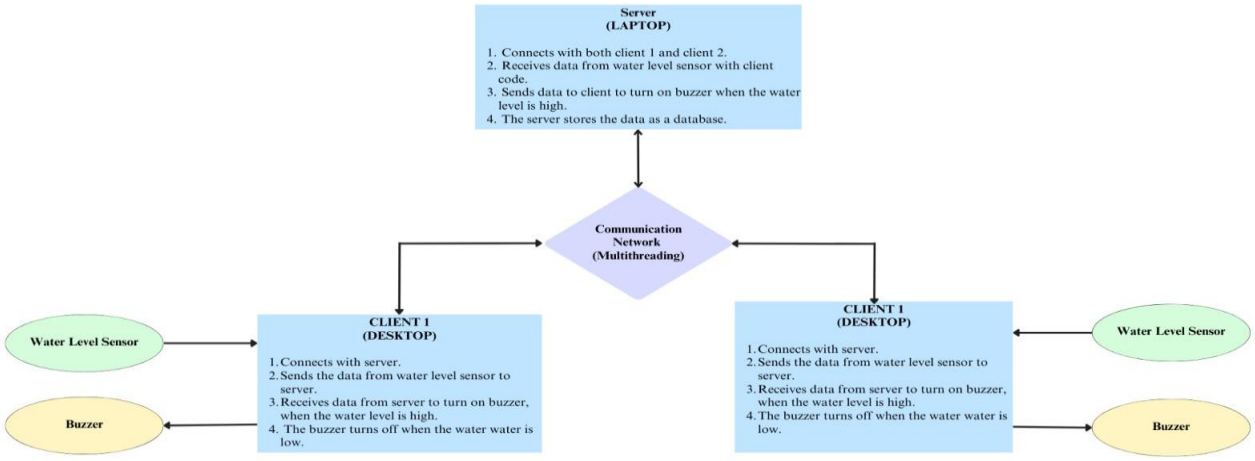

Figure 1: Block Diagram for Water Level Monitoring System: Client-Server Model

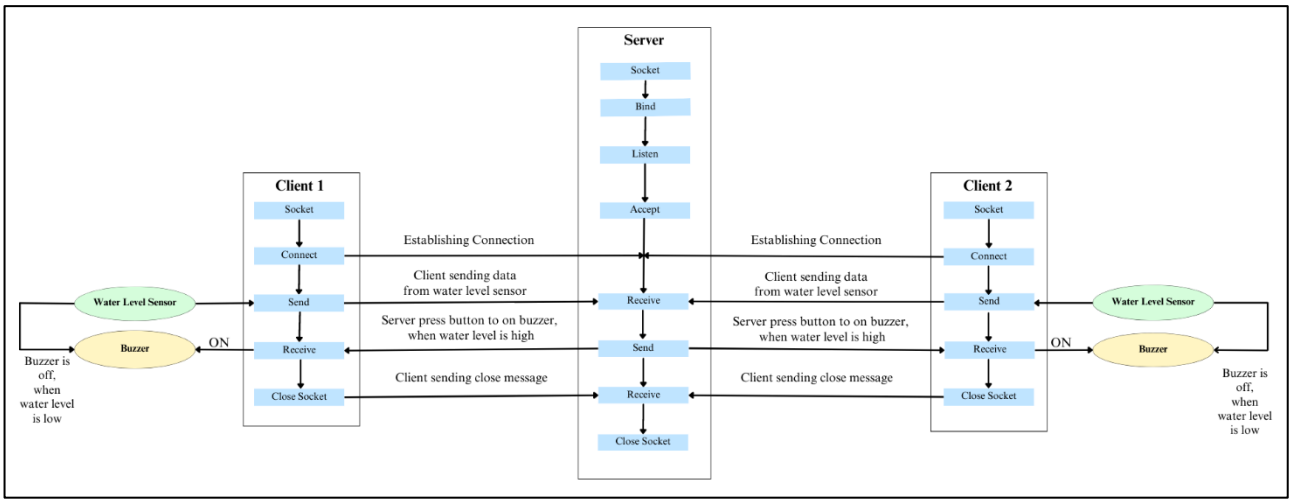

Figure 2: Flowchart for Water Level Monitoring System: Client-Server Model

METHODOLOGY

- Socket programming is used to set up the SCADA master. Two clients are connected to the server using the multithreading technique.

- Client-Side Functionality:
  - Raspberry Pi: Continuously monitor the water level using the sensor. Send real-time data to the server for visualization and analysis. Later receives data for buzzer activation.

- Server-Side Functionality:
  - Laptop: Create a server application to receive and process data from the clients. Implement real-time monitoring and alarming features.

- Communication:
  - Clients and Server: Establish WebSocket connections for real-time communication.
  - Data Transmission: Clients send water level data to the server using the WebSocket connection.
  - Alarm Notification: Clients send a warning message ("high_water_level") to the server when the water level exceeds a certain threshold.
  - Buzzer Control: Server sends control signals to the client to activate the buzzer based on the button press event.

- Display alarms and notifications on the SCADA master interface. Provide control options on the SCADA master interface to remotely activate the buzzers connected to the clients.

IMPLEMENTATION

- Connect water level sensors, and buzzers to the Raspberry Pi clients.
- Use socket programming to create a server application on the laptop.
- Establish a server socket to listen for client connections.
- Implement multithreading for concurrent client handling.
- Receive water level data from the Raspberry pi client.
- Generate alarms and notifications for high water levels.
- Program the Raspberry pi to read water level values from the sensor.
- Establish a TCP/IP connection with the server.
- Send water level data to the server.
- Program the Raspberry Pi to listen for button press events.
- Create a user interface button to control the client data and send control signals from the server to activate the buzzer.
- Integrate with ThingSpeak to store and retrieve data.
- Test the communication between the clients and the server.

RESULTS

```
Received data from client ('192.168.183.155', 49538): LOW water level
Received data from client ('192.168.183.254', 37626): LOW water level
Received data from client ('192.168.183.155', 49538): LOW water level
Received data from client ('192.168.183.254', 37626): LOW water level
Received data from client ('192.168.183.155', 49538): LOW water level
Received data from client ('192.168.183.254', 37626): LOW water level
Received data from client ('192.168.183.155', 49538): LOW water level
Received data from client ('192.168.183.254', 37626): LOW water level
Received data from client ('192.168.183.155', 49538): LOW water level
Received data from client ('192.168.183.155', 49538): LOW water level
```

Figure 3:SERVER BACKEND OUTPUT

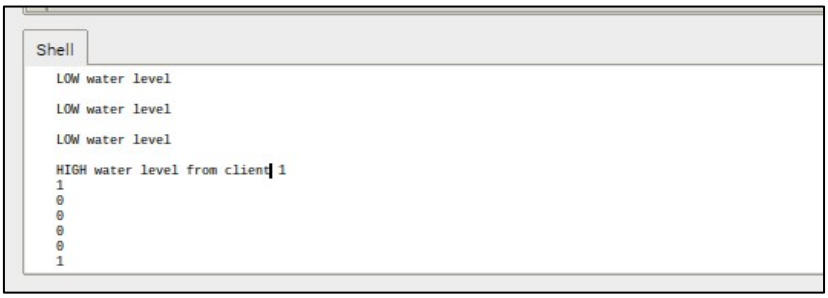

Figure 4:CLIENT 1 OUTPUT

Figure 5:USER INTERFACE CLIENT 1

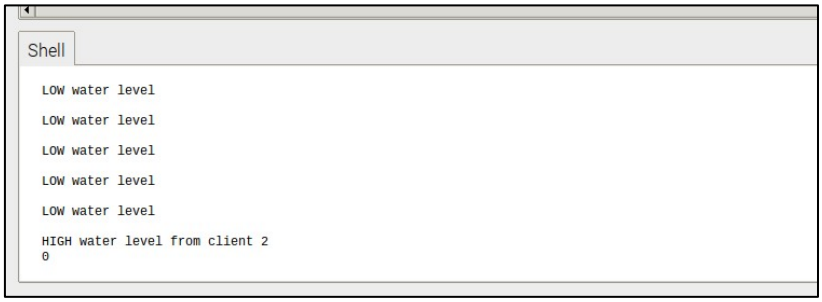

Figure 6: CLIENT 2 OUTPUT

Figure 7: USER INTERFACE CLIENT 2

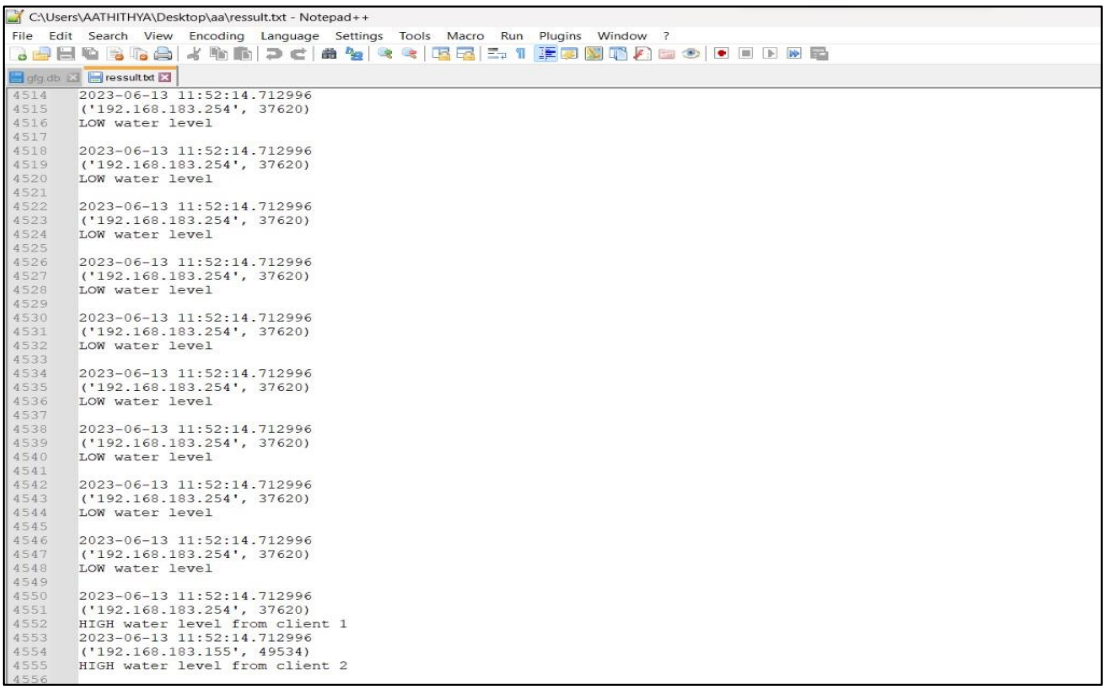

Figure 8:DATABASE

CONCLUSION

- The water level monitoring system utilizing a client – server architecture presents an effective solution for real – time monitoring and control of water levels.
- With implementation of socket programming and WebSocket connections, the system enables seamless communication between the clients and the server.
- The client continuously monitors the water levels using the sensors and transmit real – time data to the server for analysis and visualization.
- The server, in turn, receives the data, processes it, and triggers alarm notifications when water level exceeds a certain threshold.
- Additionally, the server can remotely control the buzzers connected to the clients, providing an extra layer of control over the system.
- It implements the methodology includes socket programming, multithreading, data transmission, alarm generation, button press events and data storage and retrieval.
- It is effectively monitoring the water levels by users in various scenarios, including tanks, reservoirs, and flood – prone areas.

Name: Kuralanbu S
Roll Number: CB.EN.U4ELC20033

Component	Marks
Topic (2)	
Implementation & Results (5)	
Report (2)	
Total (9)	