WATER MONITORING SYSTEM

SUBMITTED TO THE

DEPARTMENT OF MATHEMATICS PHYSICS AND COMPUTING,

MOI UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF

BACHELOR OF SCIENCE IN

COMPUTER SCIENCE

BY

COM/01/19: YASIR SHARIFF ADAN

Declaration

I declare that this report is my original work and has not been presented for a degree award in any other university. No part of this work may be reproduced without prior written permission of the authors and/or Moi University.

COM/01/19	Yasir Shariff	25/04/2023

Approval Page

Supervisor's Declaration

This project has been submitted for review with my approval as the University Supervisor.

Signature: ……. Date ……

Name: …….

DEPARTMENT OF STATISTICS AND COMPUTER SCIENCE

MOI UNIVERSITY

Dedication

I would like to express my heartfelt appreciation to my beloved parents for their unwavering support and dedication towards my education. This work is dedicated to them in recognition. of their selfless efforts and sacrifice.

I am also grateful to my lecturers and classmates for their contributions, assistance, and motivation throughout my course of study. Their guidance and encouragement have been invaluable, and I am fortunate to have had their support.

I acknowledge that this work is a product of my own efforts and dedication, and that any assistance received has been duly acknowledged in the report. Once again, I extend my sincere gratitude to all those who have played a part in my academic journey.

Abstract

The Water Quality and level monitoring system is an advanced solution designed to monitor the water level and ensure the purity of both overhead and underground tanks. The system employs a real-time monitoring system that automatically activates a pump to refill the tank when the water level drops. Once the tank reaches its maximum capacity or set level, the pump is de-energized.

Additionally, the system includes an automatic purification feature that utilizes a turbidity sensor to measure water impurities and dispense the appropriate amount of chlorine through a dispensing unit.

The design of this system includes several key components, including a power supply unit, microprocessor unit, sensor unit, display unit, and pump drive unit. The microcontroller, a NodeMCU, is responsible for controlling the various actions of the system, while the ultrasonic sensor is used to measure the water level. The display unit provides a visual representation of the current water level in the tank, allowing for easy monitoring and control. The system also includes a manual operation feature for added convenience.

One of the main advantages of this system is its wireless connectivity feature, which allows for remote monitoring through a mobile application. This enables users to monitor the water level and purity from anywhere and at any time, providing added convenience and peace of mind. In testing, the system demonstrated outstanding performance with accurate water level monitoring and effective purification results. The system also has a high level of reliability and stability.

Future developments may include the integration of additional sensors for monitoring additional water quality parameters, as well as implementing advanced control algorithms to optimize the system's performance. Additionally, the system can be integrated with other smart home systems, enabling the control of the water level and purity through voice commands. Overall, the Intelligent Water Level and Purity Control system is a reliable, efficient, and user-friendly solution that ensures the safety and quality of the water stored in both overhead and underground tanks.

Key Words

AC----------Alternating current

ADC----------Analog-Digital Converter

DC----------Direct current

EA----------External access

EIA----------Eldoret International Airport

IC----------Integrated Circuit

LCD----------Liquid Crystal Display

LED----------Light Emitting Diode

LM----------Linear Monolithic

MCU----------Microcontroller unit

NPN----------Negative Positive Negative

NTU----------Nephelometric Turbidity Units

pH----------potential Hydrogen

PSEN----------Program Store Enable

RAM----------Random Access Memory

RD----------Read only

RMS----------Root Means Square

RXD----------Receive Data

TXD----------Transmit Data

WR----------Write only

Table of Contents

Chapter 1. Introduction and Background

1.1 Background

The "Intelligent water level and purity control" project aims to address the critical issue of sustainable water management and monitoring. The project is designed to monitor the level of liquid in a tank and automatically purify the water, ensuring a safe and clean supply. Water is a vital resource for agriculture, industry, and domestic consumption, and efficient use and monitoring are essential for maintaining sustainability. However, many water treatment systems lack adequate instruments to check all parameters and catch problems at their most preventable stages. Poor maintenance also leads to increased costs and damage to equipment.
Furthermore, managing these systems requires extensive ongoing personnel training.

1.2 Statement of Problem Area

The proposed system addresses these issues by incorporating an automatic pumping system, microcontroller-based automated water level sensing and control, and a water purification process. The system also includes a level sensor that ensures the right volume of water is in the tank before the addition of treatment chemicals, preventing water overflow, and wastage.
The system is designed to improve efficiency and effectiveness in ensuring a safe and clean water supply. Additionally, the system allows the owner of the water tank to be notified of the water level and to turn the pump on or off remotely.
The project was inspired by my experience on various residential houses where I found the manual process of adding chlorine to the water supply to be time-consuming and ineffective. I realized the need for an intelligent water purity control system that can automatically monitor the water quality and add the necessary treatment chemicals with the touch of a button. Such a system would be more efficient and effective in ensuring a safe and clean water supply not only for domestic consumption but also for other industries. The project aims to contribute to the sustainable use of water resources and improved water management practices

1.3 AIMS AND OBJECTIVES

1.2.1 MAIN OBJECTIVE

The main objective of this project is to design a real time water level and purity control system that will automatically check the water level in tanks and automatically add the right amount of water treatment chemicals to the water.

1.2.2 SPECIFIC OBJECTIVES

Some of the specific objectives includes:
1. To develop a system for real-time monitoring of water levels in tanks

2. To enable automatic purification of water and remote operation of water treatment systems
3. To prioritize cost-effectiveness in the design of the system
4. To store data on water usage and purity for future decision-making
5. To ensure that the system is suitable for use in both industrial and domestic settings
6. To incorporate advanced sensors and monitoring technologies to ensure accurate and reliable data collection
7. To develop a user-friendly interface that enables easy operation and maintenance of the system by users with varying levels of technical expertise
8. To conduct extensive testing and validation of the system to ensure its effectiveness and reliability in diverse operating conditions
9. To add water treatment chemicals based on the quality of water for optimal purity.

1.4 JUSTIFICATION

The automatic water level monitor is a device that is designed to automatically monitor and control the water level in a tank or other container. It is typically used in conjunction with a water pumping system to ensure that the water level remains at an optimal level and to prevent overflow or the tank from running dry.

Figure1: water overflowing in a septic tank

One of the main advantages of the automatic water level monitor is that it eliminates the need for manual operation of the water pumping system. This can greatly reduce the risk of human error, which can lead to water spillage or other problems. Additionally, the automatic water level monitor can also provide real-time monitoring of the water level, which can help to detect and prevent potential issues before they occur.

The automatic water level control and automatic pump shutdown system works by using

sensors to detect the water level in the tank or container. When the water level reaches a certain point, the pumping system is automatically turned off, preventing overflow and water spillage. Conversely, when the water level drops below a certain point, the pumping system is automatically turned on, ensuring that the tank is always kept at an optimal level.
The automatic water level monitor also can send notifications or alarms to the operator if the water level is too high or too low. This is useful in the case when the operator is not present in the location or if the operator needs to be informed of the water level status.
Overall, the automatic water level monitor is an important tool for ensuring the efficient and accurate operation of water pumping systems, while reducing the risk of human error and improving the overall safety and reliability of these systems.

1.5 SCOPE OF THE PROJECT

The scope of this project is to design an automatic water level monitoring and control system that ensures a constant reserve of water in a reservoir. The design aims to be simple and easy to use, without introducing unnecessary complexities that would make the system difficult to operate. The system is designed to be affordable and to use materials that are easily obtainable and reliable. The design also keeps in mind the cost-effectiveness and aims to not include any complex peripheral devices that would increase the cost without providing any significant additional benefits.
The automatic water level controller is the main component of the system, it detects and controls the water level in the tank. The controller uses sensors to detect the water level and sends signals to the pump to turn on or off depending on the water level. This ensures that the water level in the tank is always at an optimal level, preventing overflow or the tank from running dry.
The automatic water level controller also can send notifications or alarms to the operator if the water level is too high or too low, this helps in the case when the operator is not present in the location or if the operator needs to be informed of the water level status.
The system is designed to be simple and easy to use, and does not require extensive technical knowledge to operate or maintain.
Overall, the scope of this project is to design an automatic water level monitoring and control system that is simple, affordable, and effective. It aims to improve the efficiency and accuracy of water pumping operations while reducing the risk of human error and increasing the overall safety and reliability of these systems.

1.6 CONSTRAINTS

During the course of this project, I encountered several constraints that presented challenges and required adjustments to the design and implementation of the system. One of the main constraints was the difficulty in finding a suitable design for the project. I had to conduct

extensive research and testing to develop a design that would meet the project's objectives and requirements.
Another constraint was the sourcing of materials and components for the project. I had a hard time finding the right pump and buffer for programming that would be suitable for the project. This required significant time and effort to find the right materials and components that would meet the project's needs and budget.
Additionally, budget was also a constraint, I had to work within a limited budget (20K) and had to be mindful of the costs of materials and components while designing the system.
Despite these constraints, I was able to work through these challenges and successfully design and implement an automatic water level monitoring and purity control system that met the project's objectives and requirements.

1.7 LIMITATIONS OF THE PROJECT

It is important to note that this project has certain limitations that should be taken into consideration when using or implementing the design. One of the main limitations is that the design is only intended for use with 12V, 5 amps electric pumps and cannot be used to control industrial water pumps that require more than 5 amps of power.
This means that this design is not suitable for large scale or industrial applications that require higher power pumps. The design is intended for small scale or residential use, and would not be able to handle the demands of a larger system.
Additionally, the design may also have limitations in terms of the accuracy of water level detection and control, as it is dependent on the quality and performance of the sensor and components used in the system.

1.8 PROJECT REPORT ORGANISATION

The organization of this project report is well detailed and vast in its coverage it covers all the activities encountered during the research work. The first chapter of this work took care of the introduction, aims and objective, scope, Justification, and project report organization. Chapter two highlight on literature review chapter three highlight on description of system and some of the component used were emphasized, it also highlights on the system design and implementation, construction, testing and packaging of the pump. Chapter four deals with results and discussion and finally chapter five is all about the conclusions problem encountered recommendation and cost of the project

Chapter 2. Literature Review

2.0 REVIEW

The automatic water level and purity control system is designed to detect the water level in the tank and ensure continuous water flow round the clock. It is composed of a microcontroller, which is programmed using the C programming language. The program is then burned into the Node-MCU integrated circuit, which is a low-cost, open-source development board that is based on the ESP8266 microcontroller. The Node-MCU features on-board Wi-Fi, making it easy to connect to the internet and send or receive data wirelessly. The level measurement in this system consists of determining the distance from the upper surface of a liquid in a reservoir or vessel or any arbitrarily chosen mark located above or below this surface. The level is not an independent physical quantity describing the state of a substance through direct and indirect level. Some examples of direct level measurement are dipstick, the bubbler, immersion electrode, capacitor type, and liquid level radiation type liquid level measurement. For instance, the dipstick is a simple method, where the stick is dipped periodically through a hole and the hole and the immersion mark is read off with the aid of the calibration on the stick.

Another method of direct level measurement is the sight glass, which relies on the manometer principle. A transparent tube is placed in a convenient location and connected to the lower part of the tank. It is graduated for safety reasons and the top of the bright glass is vented into the tank. The sight glass also has isolation valves on top and bottom. The Node-MCU microcontroller has the ability to switch on the pumping machine when the water in the tank has gone below the gauge level and automatically switch off the pumping machine when the water in the tank has reached its maximum level.

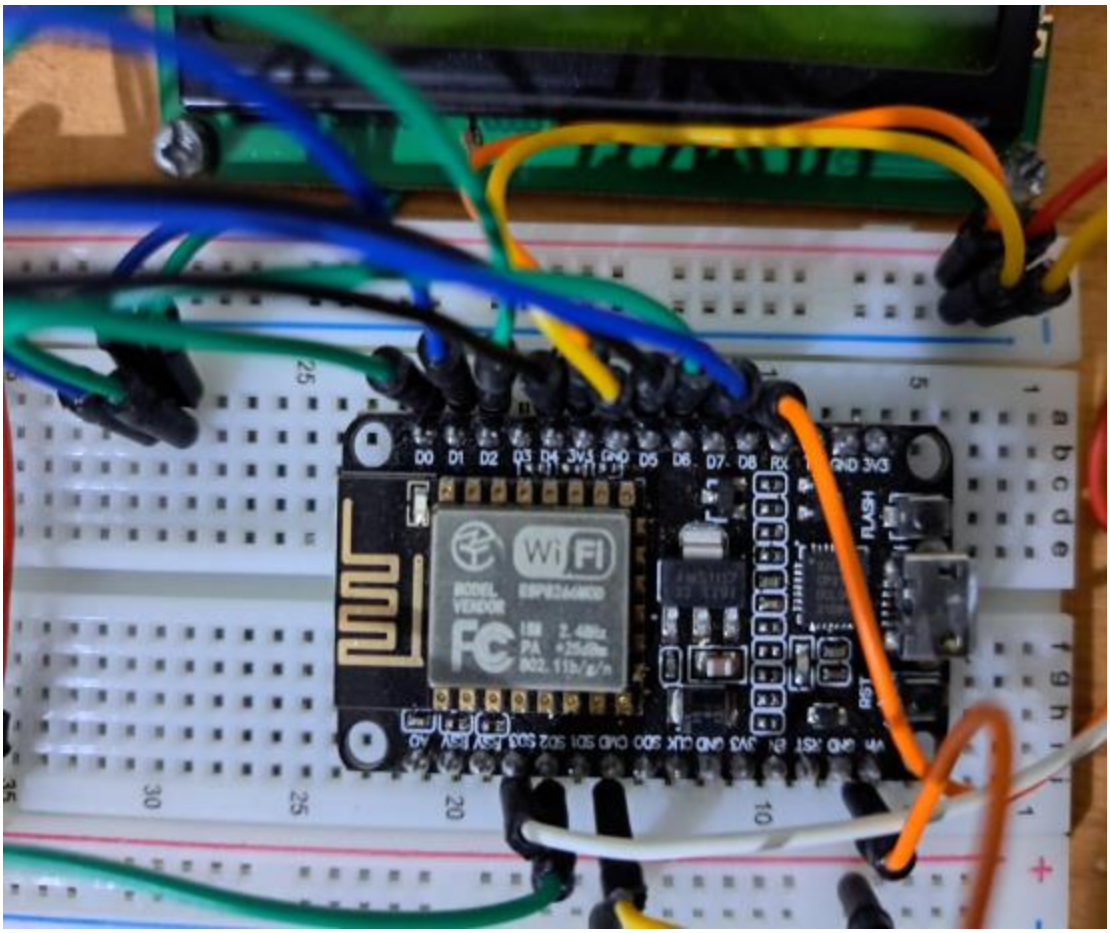


Figure 2: NodeMCU integrated circuit

2.1 OVERVIEW OF WATER PURITY CONTROL


Figure 3:Water pollution and contaminants

Water pollution occurs when harmful materials enter water sources such as ponds, rivers, lakes, seas, and oceans, resulting in a decrease in water quality and purity. Industries and city

sewage are major sources of pollution, but it can also come from agriculture and improperly disposed waste. Pollutants include viruses, bacteria, fertilizers, parasites, pharmaceuticals, pesticides, nitrates, fecal waste, phosphates, radioactive substances, and plastics. These pollutants may not always change the color of the water, but they can still be harmful.
Water quality monitoring is an essential tool for identifying and assessing the type and level of water pollution, and for determining trends over time. It is used for the management of water resources and the protection of the water environment.

Figure 4: Societal impact of clean water

By monitoring water pollution, the source, type, and extent of pollution can be identified in a timely manner, allowing for the development of appropriate response and mitigation plans, which can have a positive impact on human health and safety.
It is an important aspect of water resources management and water environment protection. The water quality automatic monitoring system is able to continuously measure and record real-time data. As monitoring points are often dispersed and located far from the monitoring center, a special wireless communication system is not always necessary for transmitting water quality information to relevant departments unless there are specific data requirements. Typically, a public network is used for data transmission, which is both cost-effective and reliable. In some systems, such as those with long distances, poor environments, and scattered data collection points, a different communication method may be used.

2.2 MAJOR CONSEQUENCES OF POOR WATER QUALITY

Poor water quality due to pollution can have a wide range of negative consequences for both human health and the environment. Some of the main consequences include:
1. Health impacts: Exposure to pollutants in water can cause a variety of health problems, including skin irritation, respiratory problems, and gastrointestinal illnesses. In severe cases, exposure to contaminated water can even lead to death.
2. Ecosystem impacts: Polluted water can harm or kill fish and other aquatic life, as well as plants that depend on clean water for survival. This can have a ripple effect throughout the ecosystem, as other animals that rely on these species for food may also be affected.

3. Economic impacts: Poor water quality can have a significant impact on local economies that depend on fishing, tourism, and other industries that rely on clean water. In addition, the cost of cleaning up polluted water and addressing health impacts can be substantial.
4. Societal impact: Clean water is a basic human right and an important part of any society. Poor water quality can lead to increased poverty, lack of access to safe drinking water, and greater vulnerability to waterborne diseases.
5. Environmental impacts: Polluted water can severely damage local habitats, such as wetlands and coral reefs, and the plants and animals that rely on them. This can lead to declines in biodiversity and loss of important ecosystem services.

2.3 IOT (Internet of things)

Internet of Things (IOT) is determined as the network of environmental objects which includes devices, homes, motors which are embedded with sensor, micro-controller, and community associativity It helps combine objects which can then exchange statistics. IOT is a gadget of interrelated computing gadgets, virtual machines, objects, animals or people with specific identifiers and ability to transfer statistics over a community with non-human-to-human or human-to-pc interplay. IOT includes net-enabled smart devices that use sensors, embedded processors, and hardware to retrieve, ship and get entry to information they gather from their environments. IOT gadgets share the sensor records they acquire by connecting to an IOT gateway to the device where information is both sent to the cloud to be analyzed or analyzed domestically. The real-global programs are net of factors, tiers from purchaser IOT and organization IT to production and commercial IOT. IOT has advanced from the convergence of wi-fi technologies, microelectromechanical structures, micro services, and the internet. IOT encourages agencies to rethink the ways they technique their corporations, industries and markets and gives them equipment to enhance their commercial enterprise techniques. Because of IOT now more electronic gadgets are connected to the net and communicate with each other without any human interference. ATM (Automated Teller Machine) was one of the earliest IOT enabled devices ever to be introduced. Wearable devices like fitness trackers, smartwatches are connecting extra regular customers to the IOT. The generation of internet-linked garbage is coming. Smart homes are not a factor for the future anymore. Human beings have already commenced adopting smart home, home equipment and home automation gadgets.

2.4 Android Application

In terms of this project, Android refers to the mobile operating system which is developed by Google and is based on the Linux Kernel. Its main use is for touchscreen mobile devices such as

smartphones and tablets, but it can also be used to make specialized user interfaces for televisions and wrist watches. In July 2013, the Google Play store hit the one million mark for the number of Android applications published, with over 50 billion downloads of these apps (see figure 1.1). In 2014, Google announced that there were over one billion active monthly Android users, which is approximately double what it was the previous year.

Google has released Android's source code under open source licenses, which means that it is free for everybody to access. This has resulted in a larger community of developers using the open source code as a foundation for community-driven projects. Android's success has gotten it involved with the "smartphone wars' ' between technology companies.

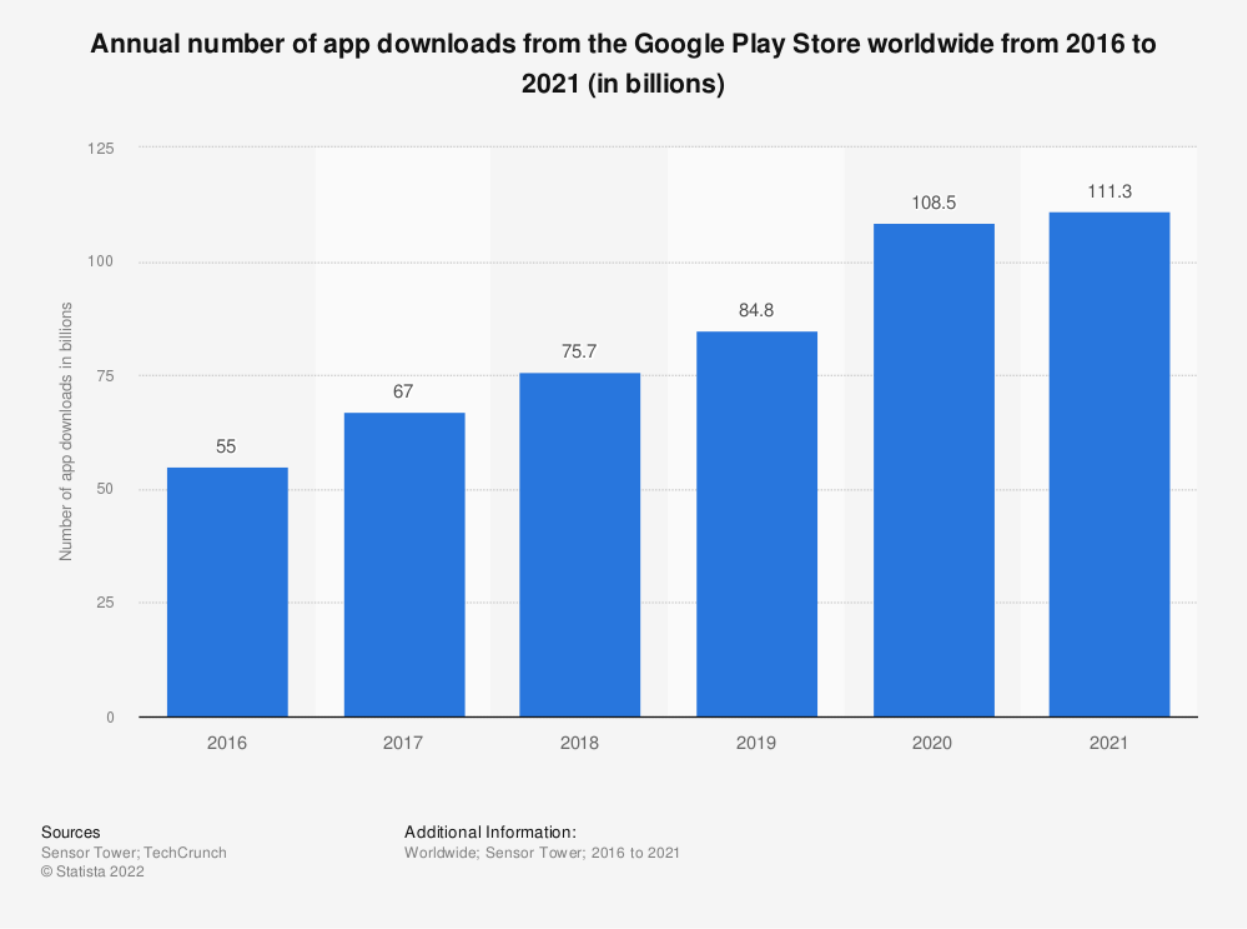


Figure5: Number of app downloads from Google Play from 2016-2021

Chapter 3. System Functional Specification

3.1 Functions Performed

1. Data collection: The system should be able to collect and measure various water parameters, such as pH, water level, turbidity.
2. Sensor communication: The system should be able to communicate with the sensors to receive the measured data. This can be done wirelessly or through a wired connection.
3. Data processing: The system should be able to store the collected data in a database and process the data to identify any abnormalities or deviations from expected values.
4. Alert generation: The system should be able to generate notifications to relevant stakeholders, such as residents, if any abnormalities or deviations are detected.
5. Reporting: The system should be able to display the water quality data in a table and generate reports on water quality trends and patterns.
6. Data security: The system should have robust security measures in place to protect the water quality data from unauthorized access, modification, or deletion.
7. User interface: The system should have a user-friendly interface that allows users to view, query, and analyze the water quality data easily and efficiently.

3.2 User Interface Design

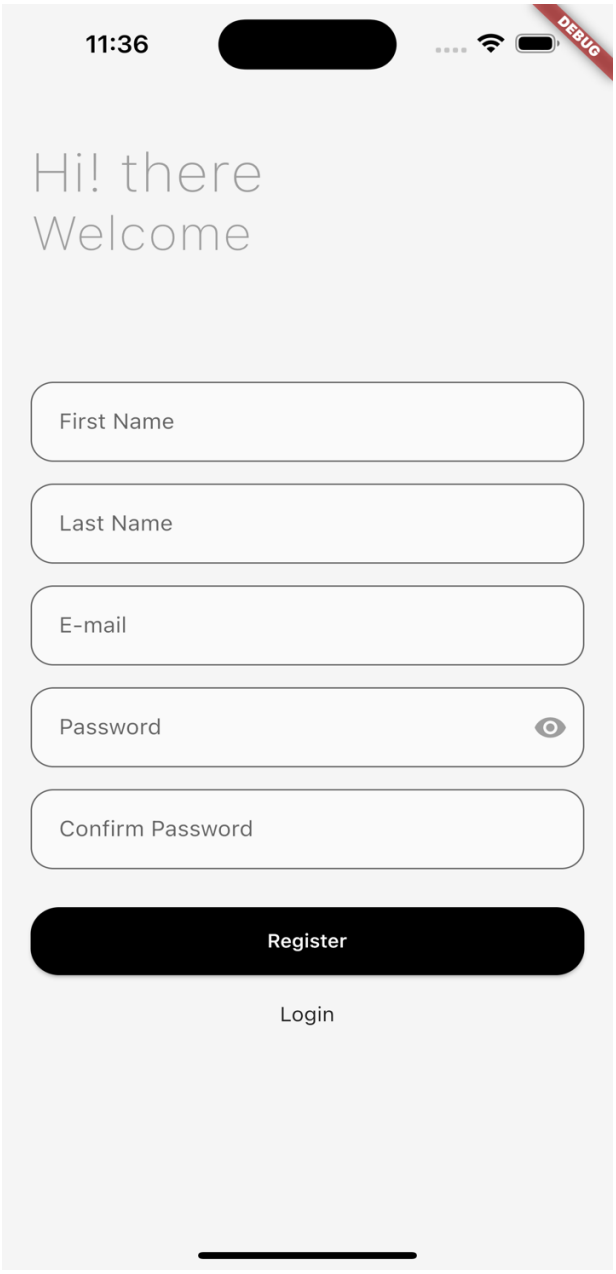


Figure 6: Register Page

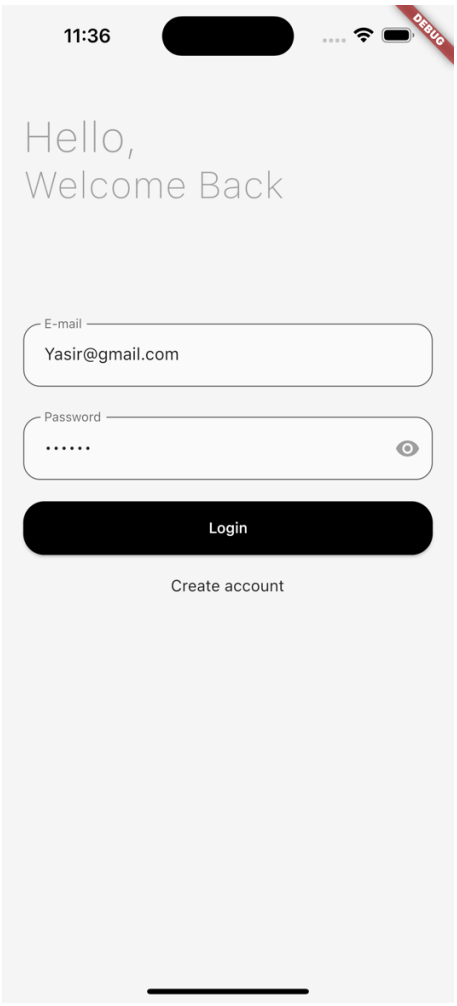


Figure 7: Login Page

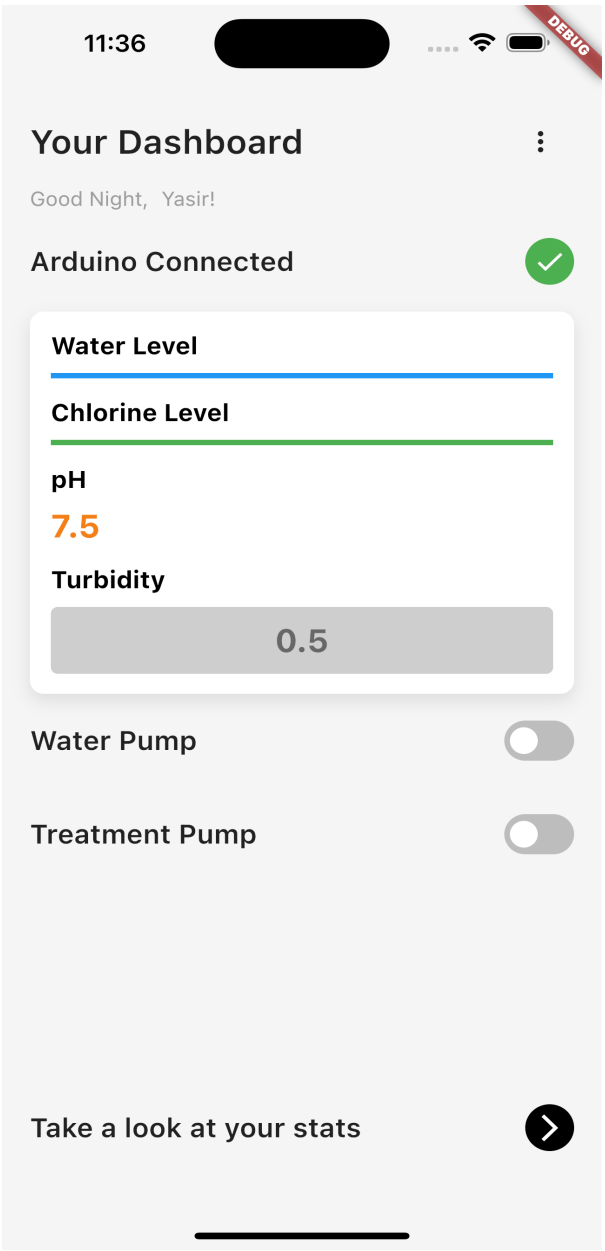


Figure 8: Dashboard

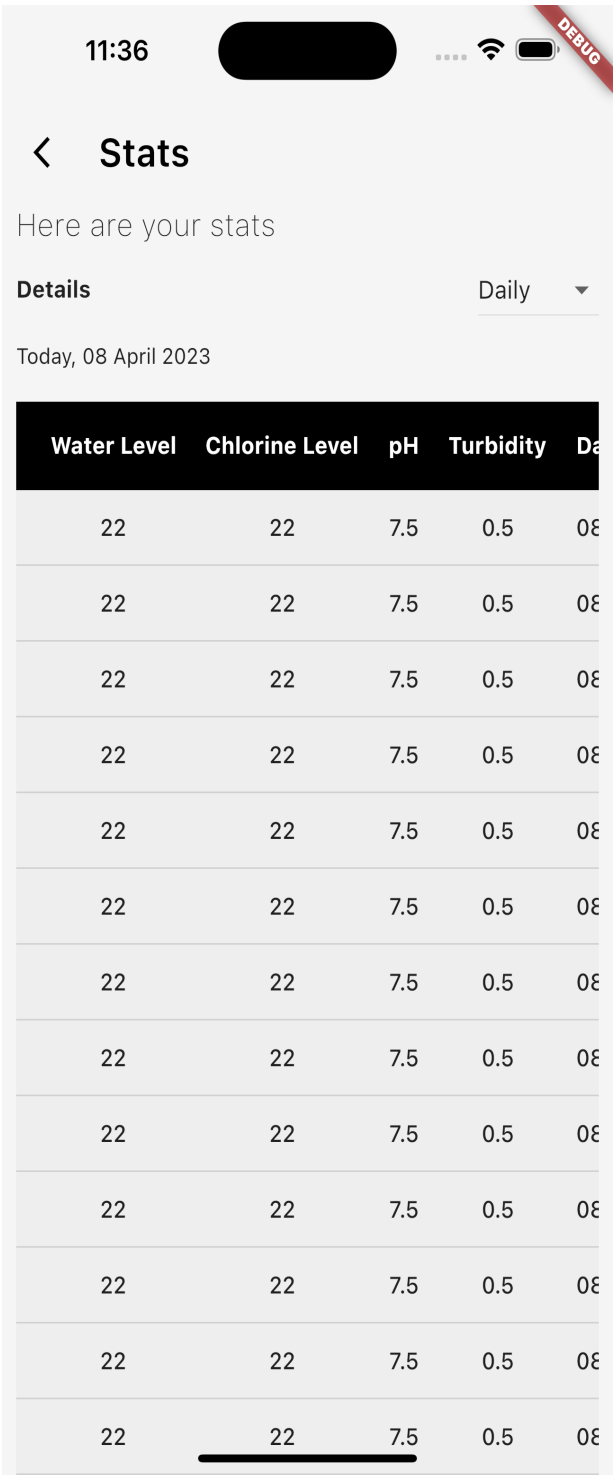


Figure 9: Stats

3.3 System Database

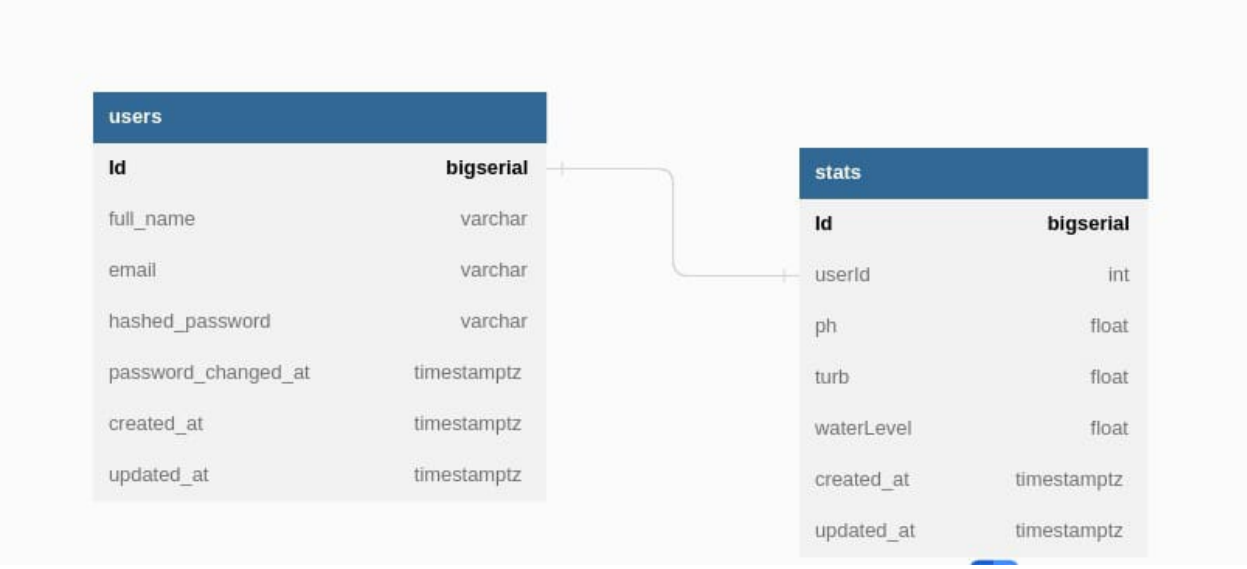


Figure 11: Database design

3.4 User Interface Specification

The user interface for the system should be intuitive, easy to use, and provide real-time data visualization, alerts, and notifications. Below are some of the key specifications for the user interface:

1. Dashboard: The user interface should include a dashboard that provides an overview of the system's status, including real-time data visualization of key water quality parameters such as pH level, turbidity, etc.
2. Customization: The user interface should support customization of the system's parameters, including setting threshold values, adjusting sampling intervals, and selecting the parameters to display on the dashboard.
3. Alerts and Notifications: The user interface should provide alerts and notifications when system parameters fall outside the predefined threshold values or when an error or failure is detected in the system. The notifications should be visible and audible, and users should be able to acknowledge and clear the notifications.
4. Real-time Data Visualization: The user interface should provide real-time data visualization of water quality parameters in the form of graphs, charts, and tables. The visualization should allow users to view historical data and trends over time.
5. Reporting: The user interface should include reporting capabilities, enabling users to generate reports on water quality parameters, system performance, and other relevant metrics. The reports should be customizable, and users should be able to export them in different formats such as PDF,and CSV.

3.5 User Screens/Dialog

1. Login Screen: The login screen will prompt users to enter their credentials to access the system. Users will enter their username and password to access the dashboard.
2. Dashboard Screen: The dashboard screen will provide an overview of the water quality monitoring system. Users will be able to view real-time data visualization of key water quality parameters, system status, and alerts and notifications. The dashboard screen should also allow users to customize the system's parameters, including setting threshold values and adjusting sampling intervals.
3. Data Visualization Screen: The data visualization screen will allow users to view real-time data visualization of water quality parameters in the form of graphs, charts, and tables. The data visualization screen should allow users to view historical data and trends over time.
4. Alerts and Notifications Screen: The alerts and notifications screen will provide users with alerts and notifications when system parameters fall outside the predefined threshold values or when an error or failure is detected in the system. The notifications should be visible and audible, and users should be able to acknowledge and clear the notifications.
5. Reporting Screen: The reporting screen will allow users to generate reports on water quality parameters, system performance, and other relevant metrics. The reports should be customizable, and users should be able to export them in different formats such as PDF and CSV.

3.6 Report Formats/Sample Data

1. Data Tables: Data tables present the water quality monitoring data in a structured and organized format. They are useful for presenting large datasets, and they provide an easy way to compare the data over time.
2. Line Graphs: Line graphs are a commonly used format for presenting water quality monitoring data. They provide an easy way to visualize changes in the data over time and help users to identify trends and patterns.

Sample Data:

Here are some examples of the types of data that could be collected and reported on the system:

- pH level
- Temperature
- Turbidity

- Water level
- Chloride levels

Sample report data might include things like trends in pH levels over time, comparisons of different water quality parameters across different locations or over different time periods, and alerts when water quality parameters fall outside of predetermined acceptable ranges.

3.7 Error Conditions and System Messages

Error Conditions:

1. Sensor Failure: If one or more sensors fail, the system should generate an error message and alert the user. The error message should indicate which sensor has failed and recommend appropriate corrective action.
2. Communication Failure: If there is a communication failure between the monitoring system and the central control unit or data collection server, the system should generate an error message and alert the user. The error message should indicate the type of communication failure and recommend appropriate corrective action.
3. Power Failure: If there is a power failure, the system should generate an error message and alert the user. The error message should indicate that the system is offline and recommend appropriate corrective action.
4. Out-of-Range Data: If the monitoring system detects that the water quality parameters fall outside the acceptable range, the system should generate an error message and alert the user. The error message should indicate the parameter that is out of range and recommend appropriate corrective action.

System Messages:

1. System Startup: When the system starts up, it should display a message indicating that the system is initializing and that it may take some time for the system to start up completely.
2. Data Collection: When the system begins collecting data, it should display a message indicating that data collection has started.
3. Data Upload: When the system uploads data to the central control unit or data collection server, it should display a message indicating that the data upload is in progress.
4. Alert Notification: When an alert or notification is generated, the system should display a message indicating the type of alert or notification and provide appropriate recommendations for corrective action.

5. System Shutdown: When the system is shut down, it should display a message indicating that the system is shutting down and that it is safe to turn off the system.

Chapter 4.System Performance Requirements

System Performance Requirements for water quality monitoring system:

Efficiency

The system is designed to process water quality data in real-time with minimal latency and delay. The system has a compact size, making it easy to install in various locations.
Peripheral devices such as sensors should be utilized efficiently to reduce power usage and minimize costs.

Reliability

The system is reliable and able to function without interruptions.
Accuracy, precision, consistency, and reproducibility are essential reliability measures that are considered. The system is designed to detect errors and failures and have recovery procedures in place to minimize downtime and data loss.

Error/Failure Detection and Recovery

The system has a comprehensive list of failure modes and consequences.
The system has a reliable error logging and reporting system that documents any detected errors and failures.
Manual and automatic recovery procedures are implemented to allow for swift system restoration in case of errors or failures.

Allowable/Acceptable Error/Failure Rate

The system's allowable error and failure rate is set low to minimize errors and failures' impact on water quality data collection and analysis.

Security

Hardware security measures are put in place to prevent unauthorized access to the system's physical components.
Software security measures such as encryption are implemented to protect the system's software from unauthorized access and tampering.
Data security measures are taken to protect water quality data from unauthorized access, alteration, or deletion.
Execution security measures such as user validation is implemented to ensure that only authorized personnel have access to the system.

Maintainability

The system is designed to be easily maintainable. Modifiability is considered to ensure that the system can be upgraded or modified without requiring significant changes to the system's architecture. Portability is considered to ensure that the system can be moved from one location to another with ease.

Chapter 5.System Design Overview

5.1 System Data Flow Diagrams

5.1.2 Data Flow diagram

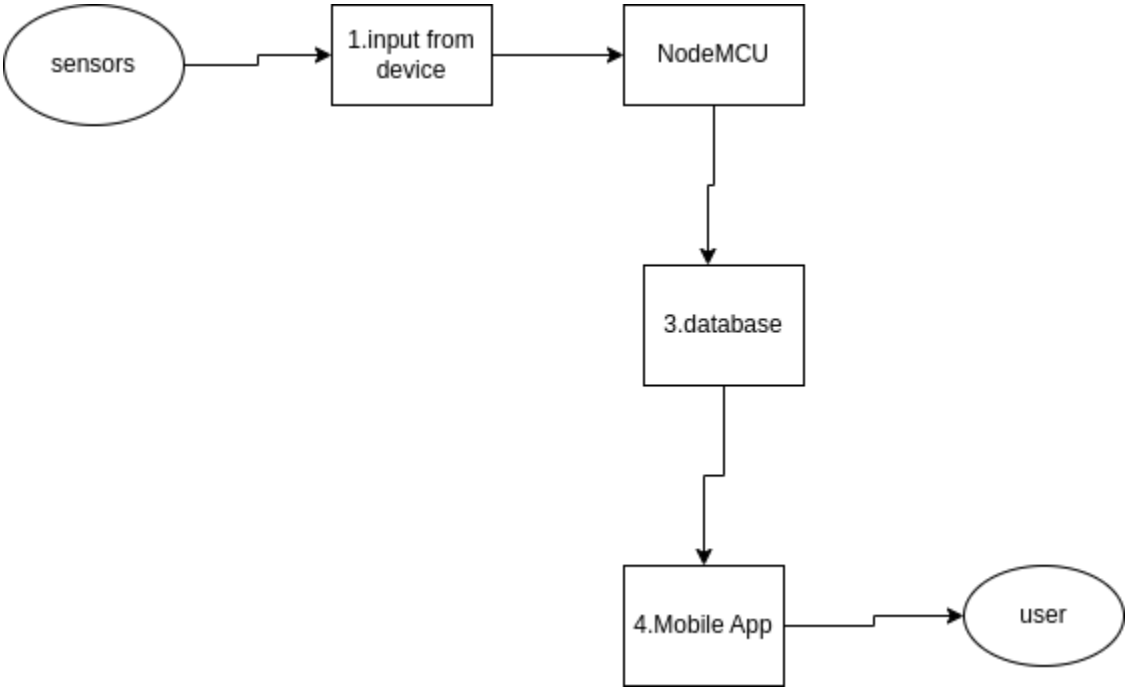


Figure 12: data flow diagram

5.1.2 sequence diagram

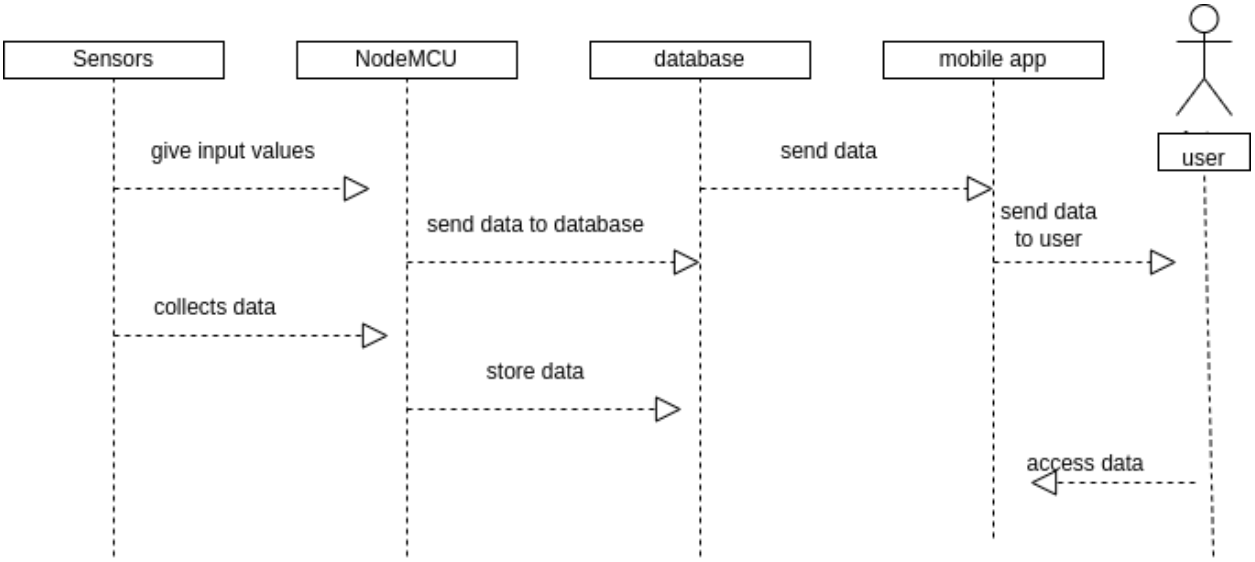


Figure 13: sequence diagram

5.2 System Architecture and Structure

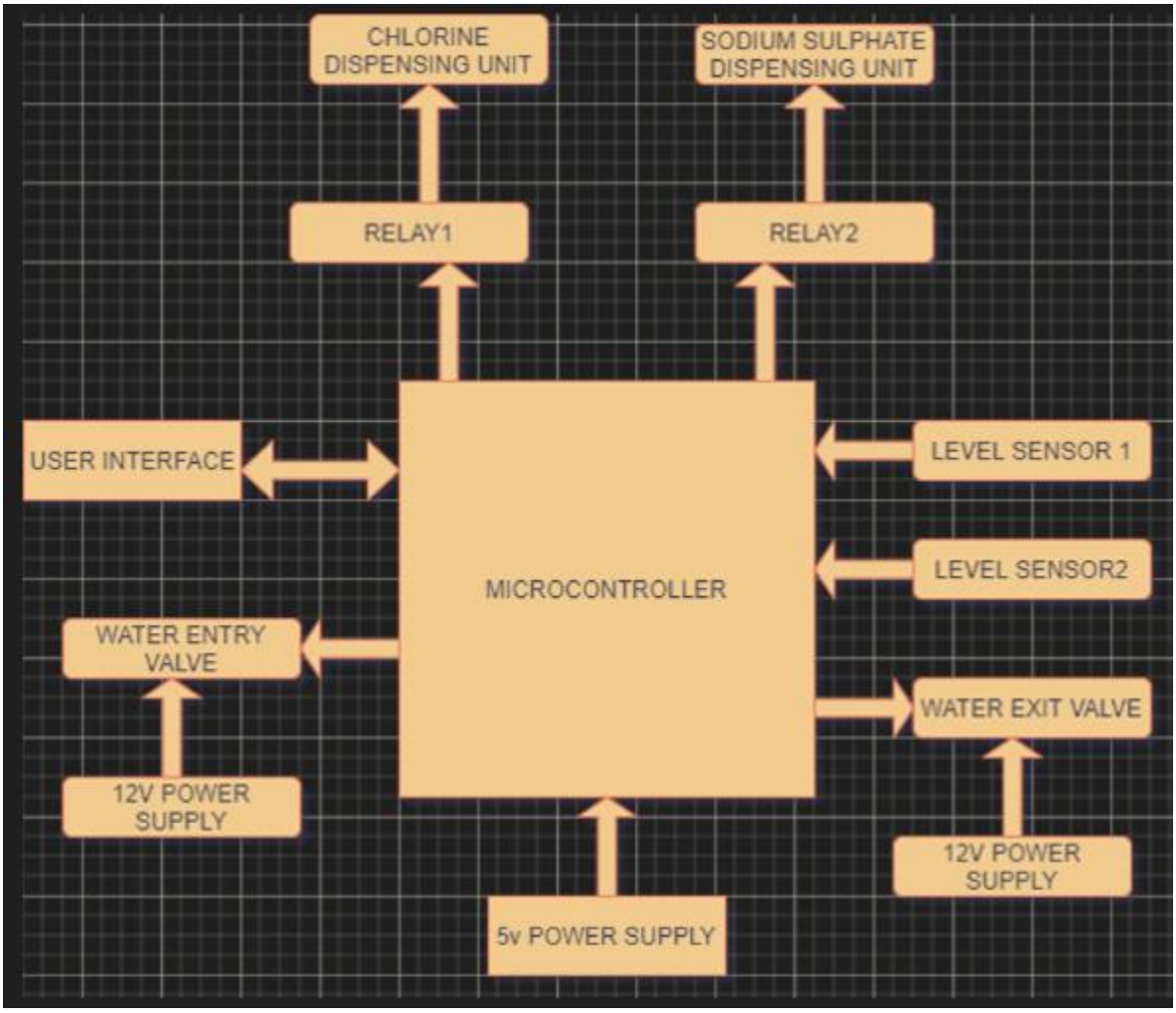


Figure 14: project block diagram

The water treatment unit is responsible for treating the water and maintaining its purity. This unit consists of a pump, pH sensor, turbidity sensor, temperature sensor, and relay. The pump is used to pump water from the tank to the treatment unit, where the water is treated and purified. The pH sensor is used to measure the pH level of the water, and the turbidity sensor is used to detect the clarity of the water. The temperature sensor is used to measure the temperature of the water, which is important for some water treatment processes. The relay is used to switch on and off the pump, ensuring that the water is only pumped when needed. The communication between the two units is established through wireless connectivity. The Node-MCU in the water level monitoring unit sends the water level data to the water treatment unit, which uses this data to determine the amount of water treatment chemicals required. The water treatment unit then sends the data back to the Node-MCU, which sends the data to the mobile app or remote monitoring system, allowing the user to monitor the water treatment process in real-time

5.3 System Data Dictionary

Here is a sample data dictionary for a water quality monitoring system:

1. User ID: A unique identifier for the User.
2. Date: The date on which the water quality parameters were measured.
3. Time: The time at which the water quality parameters were measured.
4. pH: The pH level of the water at the monitoring site.
5. Turbidity: The turbidity (cloudiness) of the water at the monitoring site.
6. Chloride Levels: The level of chloride in the water at the monitoring site.
7. Sensor ID: A unique identifier for the sensor used to measure each water quality parameter.
8. Sensor Type: The type of sensor used to measure each water quality parameter.
9. Sensor Calibration Date: The date on which each sensor was last calibrated.
10. Sensor Calibration Interval: The recommended interval between sensor calibrations.
11. Data Collection Status: A flag indicating whether the data for each monitoring site has been successfully collected.
12. Data Upload Status: A flag indicating whether the data for each monitoring site has been successfully uploaded to the central control unit or data collection server.
13. Alert Status: A flag indicating whether any alerts have been generated for each monitoring site.
14. Alert Type: The type of alert generated for each monitoring site.
15. Alert Date/Time: The date and time at which the alert was generated.
16. Alert Threshold: The threshold value used to generate the alert.

5.4 Description of System Operation (high level)

The water quality monitoring system operates as follows:

1. Sensor Data Collection: The system collects data from a variety of sensors that measure different water quality parameters, such as pH, dissolved oxygen, temperature, turbidity, electrical conductivity, and total dissolved solids.
2. Data Processing: The system processes the sensor data and analyzes it to detect any anomalies or trends in the water quality parameters. The data is also checked against predefined thresholds or limits to identify any water quality issues that require immediate attention.
3. Alert Generation: If any anomalies or trends are detected in the sensor data, or if the data falls outside the predefined thresholds, the system generates alerts and notifies the appropriate personnel, such as a water quality manager or environmental regulator.
4. Data Storage and Retrieval: The system stores the collected data in a secure database for later retrieval and analysis. The data can be accessed and analyzed by authorized

personnel to identify trends, patterns, or issues with the water quality at a particular monitoring site.

5. Data Visualization: The system provides an interface for visualizing the collected data, such as graphs or charts, to make it easier to interpret and analyze the data.
6. System Maintenance: The system undergoes regular maintenance and calibration to ensure accurate and reliable data collection. The sensors are periodically calibrated to ensure that they are measuring the water quality parameters correctly, and any malfunctions or issues are addressed promptly to minimize downtime.

5.5 Equipment Configuration

COMPONENT CONNECTIONS TO NODEMCU

1. Turbidity sensor is connected to the 5V pin on NodeMCU, the Gnd pin on NodeMCU, and any analog input pin such as A0 or A1. This is because the sensor requires power from the 5V pin and a reference to the ground to function properly. Additionally, the analog input pin is used to read the analog signal from the sensor and convert it to a digital value using an analog-to-digital converter (ADC) in the NodeMCU.
2. pH sensor is connected to the 5V pin and Gnd pin on NodeMCU, along with any analog input pin such as A0 or A1 for signal reading and another analog input pin such as A2 or A3 for temperature compensation. The temperature sensor is connected to the 3V3 pin and Gnd pin on NodeMCU, along with any digital input/output pin such as D4 or D5 for signal reading.
3. GSM module, the V_{cc} and Gnd pins are connected to the 5V and Gnd pins on NodeMCU, respectively. The Tx and Rx pins of the module are connected to the R_x and Tx pins on NodeMCU, respectively, for communication between the module and NodeMCU. The Adafruit FONA library needs to be installed in the Arduino IDE to facilitate this communication.
4. Relay module is connected to the 5V and Gnd pins on NodeMCU for power, and the control pins of the relays are connected to any digital output pins such as D6 or D7 for controlling pumps or valves based on the sensor readings.
5. Ultrasonic sensor is connected to the 5V and Gnd pins on NodeMCU for power, and the Trigger and Echo pins are connected to any digital output and input pins such as D8 and D9, respectively, for measuring the water level in a tank or reservoir. The NewPing library needs to be installed in the Arduino IDE to read the distance values from the sensor.

The longer leg (positive) of the LED is connected to a digital output pin such as D0 or D1, and the shorter leg (negative) to a current-limiting resistor, which is then connected to the Gnd pin on NodeMCU. The value of the resistor depends on the voltage of the LED and the

desired brightness. For example, a 220-ohm resistor can be used for a typical red LED. When the digital output pin is set to high (3.3V), the LED will light up, and when it is set to low (0V), the LED will turn off.
For connecting components to the NodeMCU on a breadboard, the NodeMCU is first inserted into the breadboard, and the power and ground rails of the breadboard are connected to the 5V and Gnd pins on NodeMCU, respectively. Then, the components are inserted into the breadboard and connected using jumper wires.
The positive (red) wire of the buzzer is connected to a digital output pin such as D2 or D3 through a 220-ohm resistor, and the negative (black) wire of the buzzer is connected to the Gnd pin on NodeMCU. The frequency and duration of the tone produced by the buzzer can be altered.

5.6 Implementation Languages

C++ was used for programming microcontrollers because it provided direct access to the hardware, which resulted in fast and efficient code execution. Moreover, C++ was a widely used language in the embedded systems industry, and there were several libraries and frameworks available for it, which simplified the development process.
For mobile applications, Flutter is used for the user interface(frontend) while NodeJs is used for the server side programming(backend).

Chapter 6.Module Design specifications

6.1 Sensor Module

This module is responsible for collecting water quality data from various sensors that monitor different parameters such as pH level, turbidity, etc. The module should be designed to support different types of sensors and to transmit data to the central processing unit for analysis.

6.1.1 Ultrasonic Sensor

They work by emitting high-frequency sound waves that travel through the medium being measured and bounce back to the sensor, allowing for the calculation of the distance between the sensor and the surface of the medium. This distance can then be used to determine the level of the medium in the tank or vessel.
Ultrasonic sensors are highly accurate and have a wide measurement range, making them suitable for a variety of applications. They also can handle harsh environments, such as high temperatures or corrosive substances, and can be easily integrated with a microcontroller, such as the Node-MCU, for automatic control of the water level in the tank.
The Node-MCU, a low-cost, open-source development board, is based on the ESP8266 chip and has a variety of input/output pins that can be used to connect the ultrasonic sensor and other electronic components to the microcontroller.
The Node-MCU microcontroller, combined with ultrasonic sensors, can be programmed to automatically switch on the pumping machine when the water level in the tank falls below a certain level, and to switch off the pumping machine when the water level reaches its maximum.

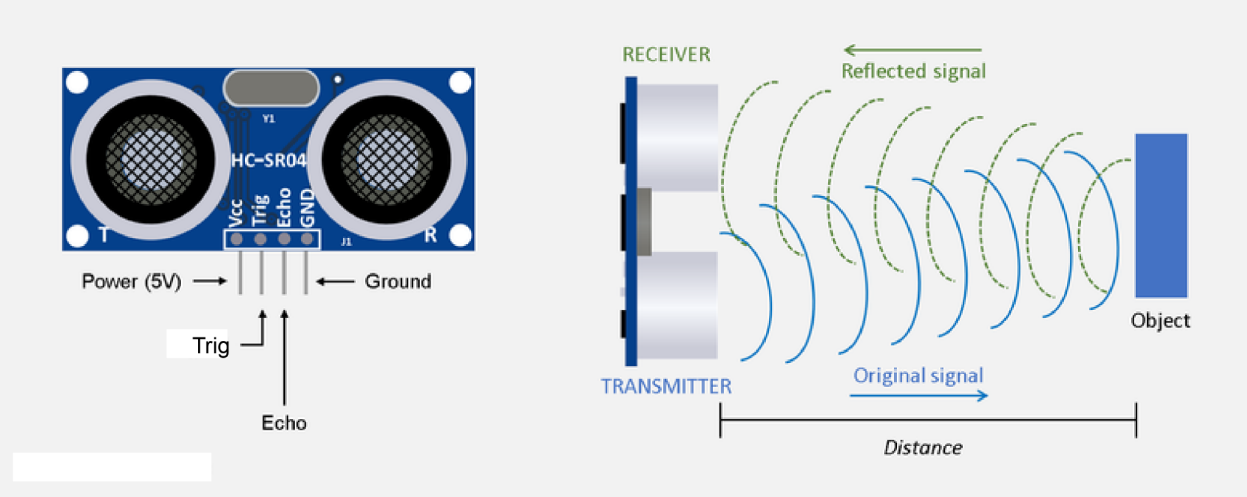


Figure 15 : Ultrasonic sensor

6.1.2 Turbidity Sensor

A turbidity sensor is a device that is used to measure the turbidity, or the degree of cloudiness or the haziness of a liquid. Turbidity is caused by the presence of suspended particles in the water, such as clay, silt, algae, or other organic matter. These particles can scatter and absorb light, making the water appear cloudy. Turbidity sensors typically measure the intensity of light that is scattered or absorbed by these particles, and use this information to calculate the turbidity of the water.

Turbidity is an important water quality parameter to measure as it can indicate the presence of pollutants, such as sewage or agricultural runoff, in the water. High turbidity can also indicate poor water treatment, leading to the spread of diseases. Turbidity sensors are commonly used in water treatment plants, wastewater treatment facilities, and other industrial and municipal settings to monitor the quality of water being treated or discharged.

Working Principle Of Turbidity Sensor

Turbidity is a key water quality parameter that indicates the presence of suspended solids, such as particles and sediments, in a liquid sample. Turbidity sensors play an essential role in the water treatment process by providing accurate and reliable measurements of turbidity levels.

These sensors are used in a wide range of applications, including drinking water, wastewater, and industrial processes, to ensure that the water is safe for consumption and meets regulatory standards.

Figure 16: The turbidity sensor

The working principle of turbidity sensors is based on the measurement of the amount of light that is scattered or absorbed by the suspended particles in the liquid. The sensor typically consists of a light source, a detector, and a chamber that holds the liquid sample. When the light beam passes through the liquid, it is scattered by the particles in the liquid, and some of the light is absorbed by the particles. The detector then measures the amount of scattered or absorbed light and generates a signal that corresponds to the turbidity of the liquid.

The output of the turbidity sensor is usually expressed in Nephelometric turbidity units (NTU), which is a standard unit used to measure turbidity. The higher the NTU value, the higher the level of turbidity in the liquid. Turbidity sensors are commonly used in combination with other water quality sensors, such as pH and conductivity sensors, to provide a comprehensive understanding of the water quality parameters. By monitoring the turbidity levels in the liquid, water treatment plants can ensure that the water is safe for consumption and meets the regulatory requirements.

6.1.3 Ph Sensor And Its Working Principle

A pH sensor is a device that measures the acidity or alkalinity of a solution. The principle behind the pH sensor is based on the properties of certain chemicals, such as indicators or electrodes, that undergo a change in color or potential when exposed to an acidic or basic solution.

The electrical potential generated by the glass membrane is converted into a pH reading by a pH meter, which is a device that measures the voltage or potential difference between the glass electrode and a reference electrode. The reference electrode is usually a silver-silver chloride electrode that is immersed in a solution of potassium chloride.

When the pH meter is turned on, it sends a small current through the glass electrode and the reference electrode. This current generates a voltage or potential difference that is proportional to the pH of the solution being measured. The pH meter then converts this voltage into a pH reading that is displayed on a digital screen or a dial.

The pH meter may also have a temperature sensor built in, which compensates for changes in temperature that can affect the accuracy of the pH reading. Some pH sensors also have automatic calibration features, which adjust the calibration of the electrode based on the characteristics of the solution being measured.

Figure 17: pH Sensor kit

6.2 Communication Module

This module is responsible for transferring data collected by the sensor module to the central processing unit. The module should be designed to support different types of communication protocols such as Wi-Fi, Ethernet, and cellular networks.

6.2.1 Sim 900 GSM modem

A digital cellular technology called GSM (Global System for Mobile Communications) is utilized to transmit mobile voice and data services. It works in a variety of frequency ranges, including 850MHz, 900MHz, 1800MHz, and 1900MHz. A GSM modem, more especially the SIM900 Module, is employed in the project of intelligent water level and purity control.
Like a mobile phone, this gadget accepts a SIM card and is connected to a mobile operator's plan.
UART (Universal Asynchronous Receiver/Transmitter), a hardware communication method that permits asynchronous serial transmission at a changeable speed, is used by the SIM900 Module to connect to the NodeMCU microcontroller. To synchronize the output bits of the transmitting device and the receiving end in UART, a clock signal is not present. The communication between the transmitter and receiver takes the form of packets instead, and the component that connects the two devices also creates serial packets and manages the hardware lines.

Figure 18: Sim 900 GSM modern

6.3 Data Processing Module

This module is responsible for processing the collected data from the sensor module. The module should be designed to analyze the data in real-time and provide accurate information on water quality parameters. The module should also include a data storage system to store the collected data for future analysis.

6.3.1 Microcontroller Unit

The microcontroller unit used in the intelligent water level and purity control project is the Node MCU, which is based on the ESP8266 WiFi module.
The Node MCU has several pins that are used for connecting sensors and other components. These pins can be programmed to perform different functions, such as input or output of digital or analog signals, or to support communication protocols such as I2C or SPI.
In this project, the ultrasonic sensor, turbidity sensor, pH sensor, temperature sensor, and GSM module were connected to specific pins on the Node MCU. For example, the ultrasonic sensor was connected to the Node MCU's trigger and echo pins, which were programmed to generate and receive ultrasonic waves and measure the distance to the water surface.
Similarly, the turbidity sensor was connected to the NodeMCU's analog input pin, which was programmed to read the analog voltage output of the sensor and convert it to a digital value that could be displayed on the mobile and web application.
The pH sensor and temperature sensor were also connected to the NodeMCU's analog input pins, which were programmed to read the voltage output of the sensors and convert them to digital values.
Finally, the GSM module was connected to the NodeMCU's serial pins, which were used to send and receive data via the GSM network.
Overall, the Nodemcu pins were essential in connecting the sensors and components used in the intelligent water level and purity control project and programming them to perform their respective functions.

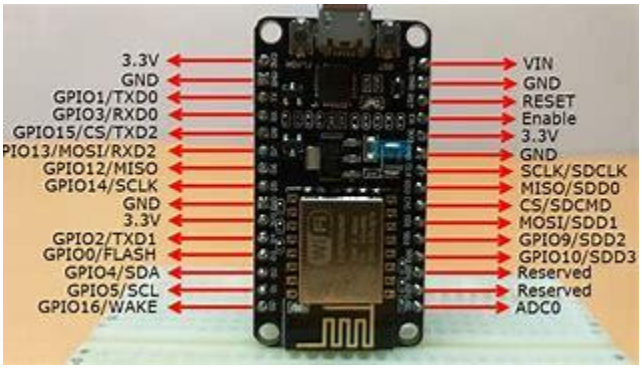


Figure 19: NodeMcu

6.4 User Interface Module

This module is responsible for providing a user-friendly interface for system users to access and interact with the system. The module is designed to provide real-time data visualization, alerts, and notifications, and support customization of the system's parameters.

6.4.1 Mobile Application

In designing a mobile application for displaying water level, water purity, and water treatment unit display in percentages, the key factors considered were a clear layout, use of graphical elements, user experience, and responsiveness.

The display unit had a clear layout with each parameter displayed in a separate section. Graphical elements such as icons and charts were used to make the display more visually appealing and easier to understand.

The application was designed with the end-users in mind, ensuring that the interface was easy to navigate, the font size and style were legible, and the colors used were easy on the eyes.

Lastly, the application was designed to be responsive to different screen sizes and resolutions, ensuring that it looked and worked well on different devices.

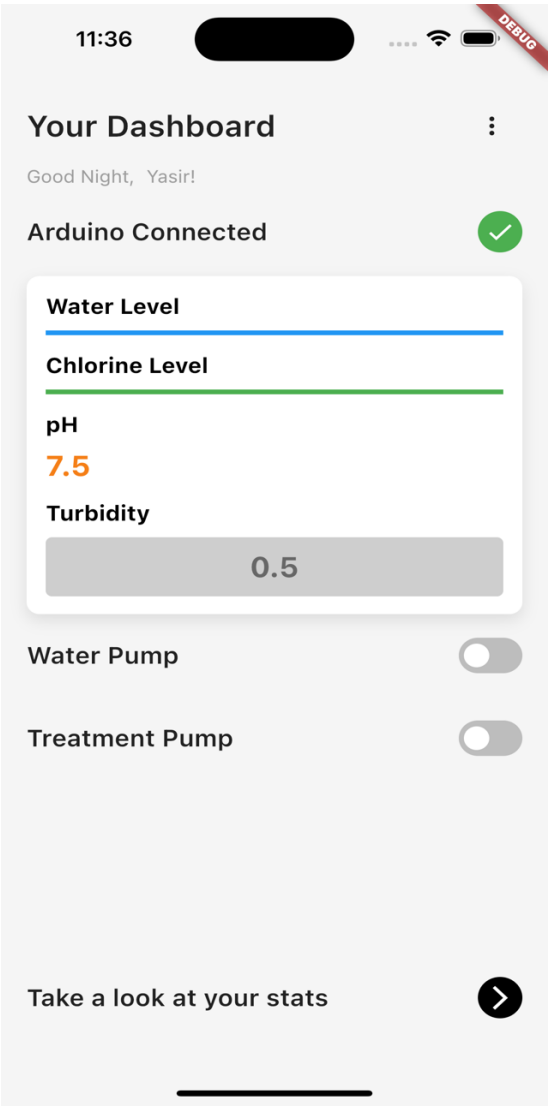


Figure 20: App Dashboard

6.5 Power Management Module

This module is responsible for managing the system's power supply. It should be designed to optimize power usage and minimize power consumption, enabling the system to operate for an extended period without requiring frequent recharging or replacement of batteries. The module should also include a backup power supply to ensure system continuity during power outages or other power-related issues.
The power source for this project is a 5V-12V AC supply, which is a type of power supply designed to convert high voltage AC mains electricity to a suitable low voltage supply for electronic circuits and other devices. A power supply typically consists of a series of blocks, each of which performs a specific function. In addition to the main power supply, the project also utilizes a USB cable for power. This allows for a versatile and convenient power source, as USB cables are readily available and can be easily connected to a variety of devices.

Ac Main 230v →Transformer →Rectifier →Smoothing →Regulator → AC Mains 5VDC

Transformer: Steps down high voltage AC mains to low voltage AC
Rectifier: Converts AC to DC but the DC output is varying diodes are the main rectifier use.
Smoothing: Smoothers the DC from varying greatly to a small ripple regulator: eliminates ripple by setting DC output to a fixed voltage.

6.6 Pump Control Module

The pump control segment of the water level and purity control project is a critical component of the system that enables automatic control of the water pump based on the water level in the tank. The system uses ultrasonic sensors to measure the water level in the tank, and a microcontroller controls the operation of the water pump based on the measured water Level.

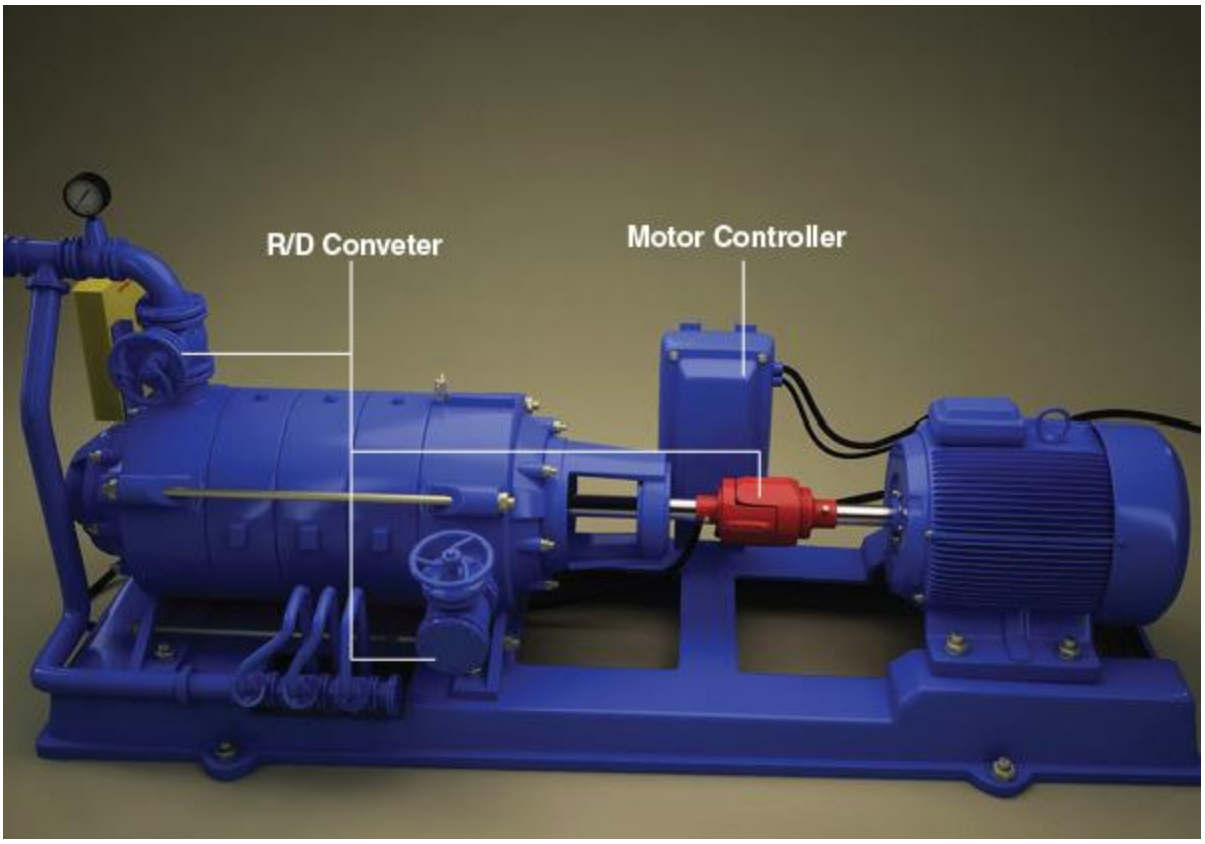


Figure 21: pump

The pump control segment is designed to turn the water pump ON when the water level in the tank falls below a certain threshold, and turn the pump OFF when the water level reaches a certain level. This ensures that the water pump operates only when it is needed, preventing water wastage and reducing energy consumption.

6.7 Algorithm Specification

Flowcharts

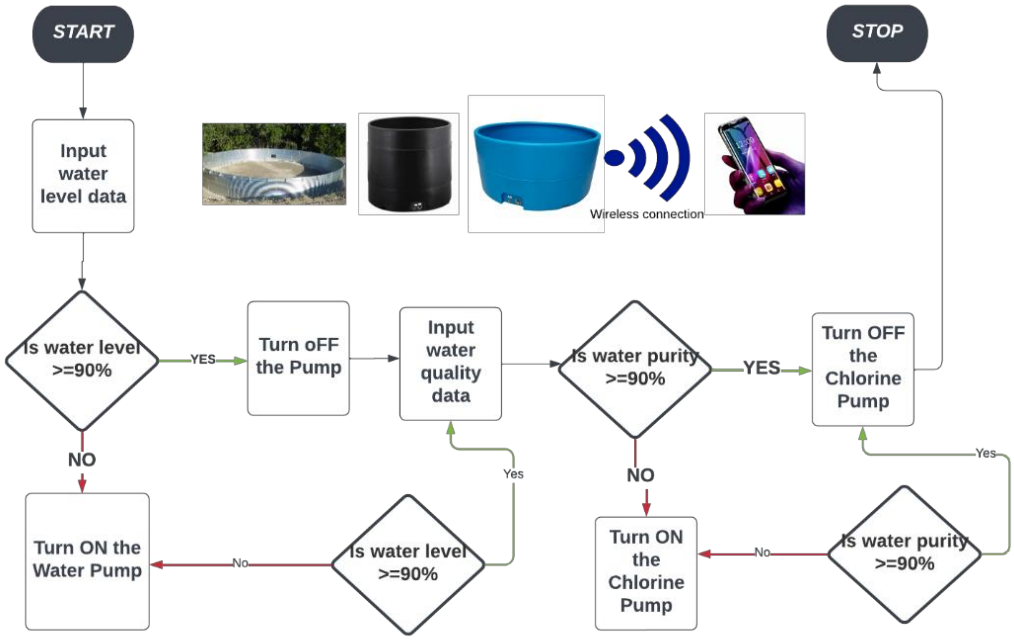


Figure 22: Project Workflow Chart

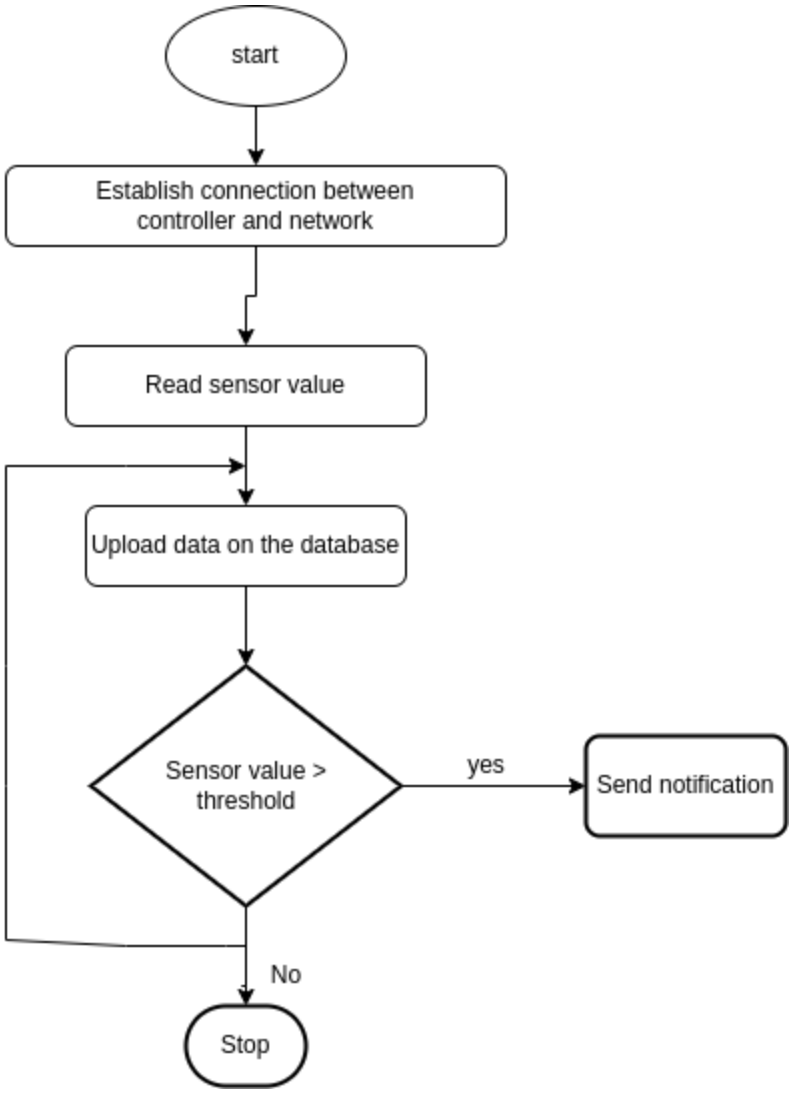


Figure 23: sensor flowchart

use cases

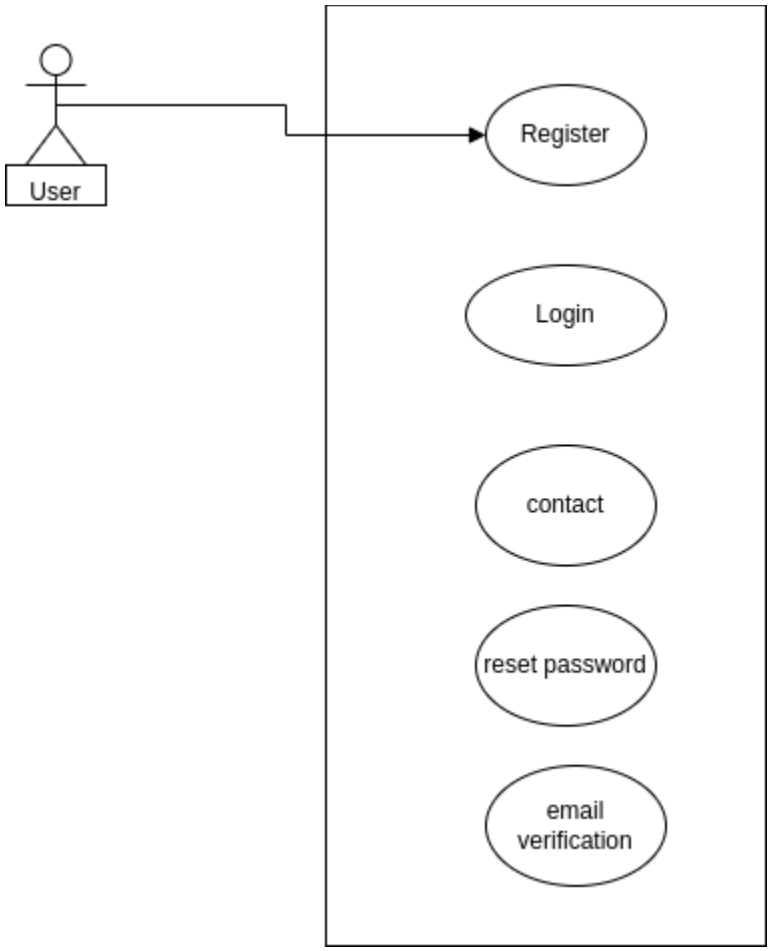


Figure 24: user login use case

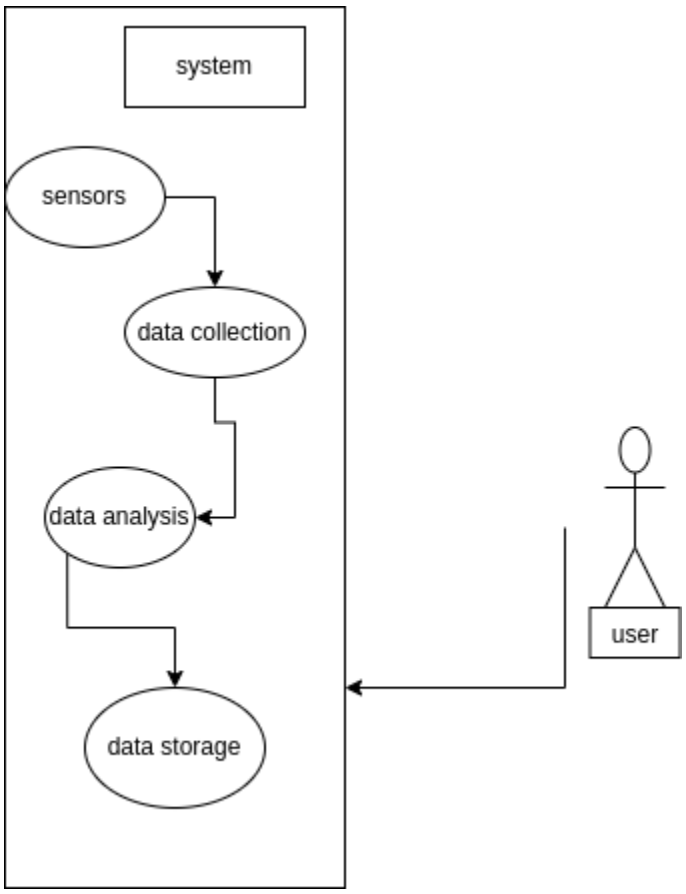


Figure 25: data use case

Chapter 7.System Verification

7.1 SYSTEM TESTING AND INTEGRATION

After the design and implementation phase, the system built must be tested for durability and effectiveness and also ascertain if there is need to modify the design of the system was first assembled using breadboard where some test was carried out at various stage .to ensure proper functioning of component expected data, the component was tested using a digital mustimeter (DMM). Resistors were tested to ensure that they were within the tolerance value.

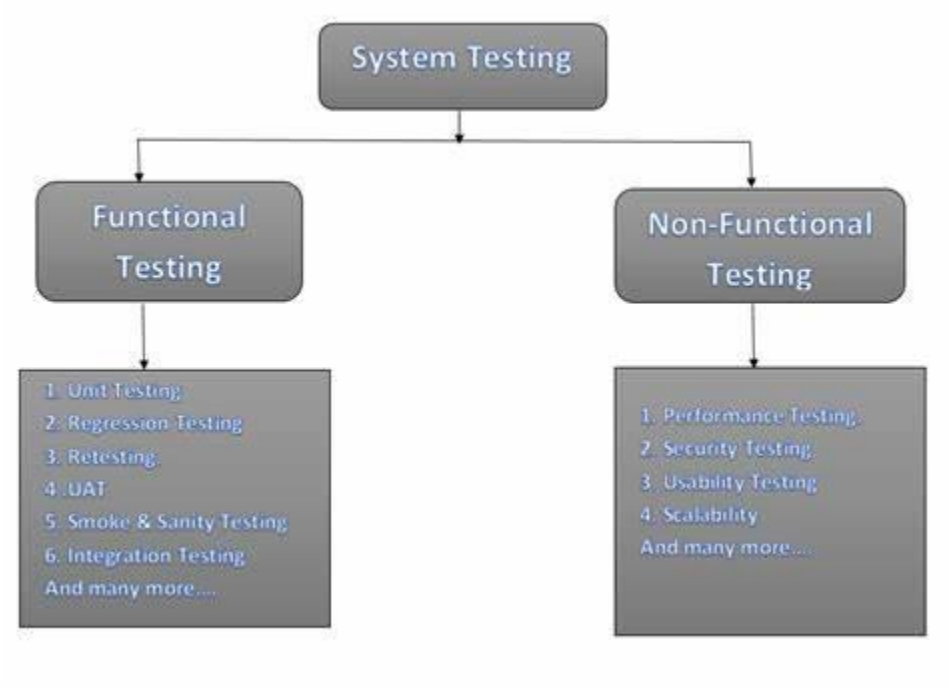


Figure 26:System testing and integration

Faulty resistor is discarded. The voltage regulator resulting output was 5.02v which is just a deviation of 0.20v from the expected result of 5.00v, the pump was also tested to ensure that it was working properly.

7.2 TEST PLANE AND TEST DATA

This chapter entail the overall system testing of integrated design of voltage measurement device, the testing, and integration is done to ensure that the design is functioning properly as expected there by enabling one or even intended users for which the project was targeted for, appreciate its implementation and equally approaches used in the design and integration of various modules of the project, however, this involves checks made to ensure that all the various units and subsystems function adequately also there must be a good interface existing between

the output /input unit subsystem. When the totality of the modules was integrated together, the system was created, and all modules and sections responded to as specified in the design through the power supply delivering into the system design.

7.3 COMPONENT TEST

Similar components like resistors were packed together. The other component include capacitor, switch, transformer, resistor, Diodes (rectifier) LED, transistor, voltage regulators etc.
Reference was made to the color-coding data sheet to ascertain the expected value of resistors used.
Each resistor was tested, and the value read and recorded. Also, for transistor test the DIMM was switched to the diode range. The collector, base, emitter junctions were tested in the following order. The collector, emitter and base pins were gotten from the data analysis on. power transistor.

7.3.1 TEST FOR TRANSISTORS

Table 1: Test for transistors

	Black probe	Red probe
1st test on pins	Collector	Base
2nd test on pins	Emitter	Base

7.3.2 TRANSFORMER TEST (step down)

Expectedly the transformer was rated 220v/12v, 1000mA. from the mains power supply, the primary coil receives 220v input, the output was measured to be 16.75v using a DMM. Test data on the transformer has it that the resistance of the primary windings for the step-down transformer is higher than that of the secondary side this was ascertained.

7.4 SYSTEM TEST

The system was powered and operated upon using several possibilities. They include making sure that the pump only starts when the water level has gone below the mark and stops when the

water level has reached maximum. The seven-segment display was also tested to make sure the correct level was displayed on the seven segment display screen. The sensors were also tested.

7.4.1 OTHER TEST

The bucket used as tank in my project was tested in other to make sure there was no leakage, the hose or pipe conveying the water from the lower tank to the upper tank was tested or checked for any kind of breakage or leakage.

7.5 PERFORMANCE EVALUATION

From the table above, it shows that range between the expected value and the actual can be tolerated. As a result of this the drift in expected value has no critical effect on the system design since the result current range was also exceeded, also the operational voltage was not exceeded.

Table 2: Actual measured results

Sensor	Sensor reading for pure water	Sensor reading for dirty water	Sensor reading after adding treatment chemicals
PH sensor	6.0	7.0	6.9
Ultrasonic sensor	60	60	60
Turbidity sensor	4.0	1.0	2.0

Description of the values

A pH sensor reading of 6.0 indicates that the water is slightly acidic. However, after the addition of appropriate pH adjusters, the pH of the water increases to 6.9, which is almost neutral with a pH of 7.

The ultrasonic sensor is used to measure the level of water, and it has been observed that the water level slightly increases after the addition of water treatment chemicals. However, this increase is very small, as the water treatment chemicals dissolve in the water.

A turbidity value of 1.0 indicates that the water is very clear, while a turbidity value of 4.0 indicates some degree of cloudiness or haziness. Following the addition of appropriate water treatment chemicals, the turbidity of the water reduces to 2.0. This indicates that the water has

been treated, and is now relatively clear.

7.6 ACTUAL RESULTS AND DISCUSSION

7.6.1 ALERTS

After the design and testing phase was completed, the app successfully notified the owner of the water tank about the level of water in the tank, ensuring that they were aware of any fluctuations in the water level.

The app was programmed to send alerts when the water level in the tank reached a certain threshold, such as when it was running low or when it was full. This notification was sent directly to the owner's smartphone, ensuring that they were always up to date on the status of their water supply.

By receiving real-time updates on the water level in the tank, the owner was able to take proactive measures to manage their water supply effectively. For example, if the water level was running low, they could schedule a water delivery or take steps to conserve water until the next delivery arrived.

The app also provided valuable insights into water usage patterns, enabling the owner to make informed decisions about their water consumption. By monitoring water levels over time, the owner was able to identify trends and patterns in water usage, which helped them to optimize their water consumption and reduce waste.

7.6.2 REMOTE TURN ON/OFF OF THE PUMP

The project allowed users to remotely turn ON/OFF the pump. This feature provided the user with greater control over their water supply, enabling them to manage their water resources more efficiently.

The remote pump control feature was integrated into the app, which allowed users to easily turn the pump ON/OFF from their smartphone. This feature was found to be reliable and responsive, allowing users to control their water pump quickly and easily from anywhere, at any time.

One of the main benefits of the remote pump control feature was that it allowed users to conserve water and energy. For example, if the user was away from home and did not need to use water, they could turn the pump OFF remotely, which would prevent unnecessary water usage and reduce energy consumption.

The remote pump control feature also provided users with greater flexibility in managing their water supply. If the user needed more water than usual, they could easily turn the pump ON remotely, ensuring that they had access to the water they needed when they needed it.

7.6.3 ACCURATE WATER PURIFICATION

After the design and testing phase was completed, this project was highly effective in ensuring that the right volume of water was in the tank before the addition of treatment chemicals. This feature provided the user with greater control over their water treatment processes, enabling them to manage their water resources more efficiently.

The water quality sensors were integrated into the tanks, which allowed the system to detect the level of water in the tank as well as the quality of the water. The sensors were designed to provide accurate and reliable readings, ensuring that the system could detect any changes in the water quality.

Water quality sensors allowed users to add treatment chemicals at the right time and in the right volume. For example, if the water quality was poor, the system would not add treatment chemicals until the right volume of water was in the tank. This ensured that the treatment chemicals were used efficiently and effectively, reducing wastage and promoting sustainable water management practices.

Another benefit of using water quality sensors was that it provided users with greater control over their water treatment processes. If the user needed to adjust the amount of treatment chemicals added to the water, they could do so easily and quickly, ensuring that the water was treated to the right standards.

7.6.4 LEVEL MONITORING AND CONTROL

The ultrasonic sensors in this project were highly effective in preventing water overflow and minimizing water wastage. This feature provided the user with greater control over their water supply, enabling them to manage their water resources more efficiently.

The ultrasonic sensors were integrated into the tanks allowing the system to detect the level of water in the tank. The sensors were designed to provide accurate and reliable readings, ensuring that the system could detect any changes in the water level.

One of the main benefits of using ultrasonic sensors was that it prevented water overflow, which minimized water wastage. For example, if the water level in the tank was getting too high, the system would automatically turn off the water supply, preventing any overflow and conserving water.

Also ultrasonic sensors were that it provided users with greater control over their water supply. If the user needed to monitor the water level in the tank, they could do so easily and quickly, ensuring that they had a reliable and accurate picture of their water supply.

7.7 PACKAGING

The packaging of electronic equipment is a crucial aspect of any project, and the intelligent real-time water level and purity control project was no exception. To ensure the protection of

the circuitry, an enclosure was created to safeguard against any potential damage. This protective measure not only prevents damage but also adds aesthetic value to the overall design.

In determining the appropriate size of the tanks to be used for packaging, various factors were considered. These factors included the ease of input and removal of water from the tanks, the positions of the tanks, space for future modifications, easy accessibility to the circuit board, and the mobility of the tanks. By taking these factors into account, the packaging was optimized to ensure that the system was easy to use and maintain.

Chapter 8. Conclusions

8.1 Achievement of Objectives

The real-time water level and quality control project has successfully achieved its main objectives of designing and developing an automated system for monitoring and controlling water levels and purity. The system is user-friendly and provides real-time information on water quality and quantity using sensors and automated control mechanisms.
The project also includes an SMS notification system and mobile application that allows users to control the water level and automatically purify the water.

8.2 Limitations and Challenges

Throughout the system development process various challenges were encountered. This was experienced from the conception, the requirements elicitation stage all the way to system implementation and testing. The constraint encountered were;

- The sourcing of materials and components for the project. I had a hard time finding the right pump and buffer for programming that would be suitable for the project. This required significant time and effort to find the right materials and components that would meet the project's needs and budget.
- Budget was also a constraint, I had to work within a limited budget (22K) and had to be mindful of the costs of materials and components while designing the system.
- One of the main limitations is that the design is only intended for use with 12V, 5 amps electric pumps and cannot be used to control industrial water pumps that require more than 5 amps of power.

8.3 RECOMMENDATIONS

The following are some recommendations that can be used towards the improvement of this system;

To start with, the government should invest in establishing industries to produce basic electronic components locally and research centers in universities. This will not only boost the local economy but also provide students with practical knowledge on electronics components and their operation, which is essential for the successful implementation of this project.

Technical skill acquisition is another critical area that requires attention to ensure that the project's stakeholders possess the necessary skills to design and develop the system. Incorporating machine learning algorithms into the system can improve its ability to detect irregularities and predict future changes in water quality.

Adding more sensors to monitor other parameters such as dissolved oxygen or conductivity could provide even more detailed and comprehensive water quality information.

8.4 Conclusion

In conclusion, the real-time water quality and level monitoring project successfully developed an automated system for monitoring and controlling water levels and purity at residential houses. This system used sensors and automated control mechanisms to provide real-time information on water quality and quantity, making it user-friendly and easy to operate.

In addition, the project included an SMS notification system and mobile application that allowed users to control the water level and automatically purify the water. This feature made the system more accessible to users and enabled them to manage their water supply more efficiently.

Bibliography

A.T. Demetillo and E.B. Taboada, (2019). Real Time Water Quality Monitoring for Small Aquatic Area Using Unmanned Surface Vehicle Engineering, Technology & Applied Science Research,9(2), pp. 3959-3964 https://etasr.com/index.php/ETASR/article/view/2661.

A. Kumar and N.P. Pathak, (2018). Wireless monitoring of volatile organic compounds/ Water vapour/gas pressure/temperature using RF transceiver. IEEE Transactions on Instrumentation and Measurement, 67(9), pp. 2223-2234 https://www.semanticscholar.org/paper/Wireless-Monitoring-of-Volatile-OrganicCompounds-Kumar-Pathak/097fc263e1c74644969183168a43e473a98bd629

Byrne, L., Lau, K. T., & Diamond, D. (2002). Monitoring of Headspace Total Volatile Basic Nitrogen from Selected fish Species using Reflectance Spectrosc optic Measurements of pH Sensitive films, The Analyst, vol. 127, https://pubmed.ncbi.nlm.nih.gov/12430606/

C. I. Onah, Federal University of Technology Owerri (2016). Design and Construction of Water Tank Monitoring and Control System with Digital Display, https://www.researchgate.net/publication/350189792_Design_and_Construction_of_Water_Tank_Monitoring_and_Control_System_with_Digital_Display.

D Sekhwela, PA Owolawi Et Al, (2020). Water quality Monitoring with notification system. https://ieeexplore.ieee.org/abstract/document/9334095/

F Jan, N Min-Allah Et Al, (2021). IoT based smart water quality monitoring: Recent techniques, trends, and challenges for domestic applications https://www.mdpi.com/2073-4441/13/13/1729

H Khaleeq, A Abou-ElNour Et Al, (2016). A Reliable Wireless System for Water Quality Monitoring and Level Control, https://www.researchgate.net/publication/312025554_A_Reliable_Wireless_System_for_Water_Quality_Monitoring_and_Level_Control

M. O. Faruq, I. H. Emu, M. N. Haque, M. Dey, N. K. Das and M. Dey, (2017). Design and implementation of cost-effective water quality evaluation system, 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC) pp. 860-863, https://www.researchgate.net/publication/323208748_Design_and_implementation_of_cost_effective_water_quality_evaluation_system

N. Vijayakumar, R. Ramya, (2015). The Real Time Monitoring of Water Quality in IoT Environment, International Conference on Circuit, Power, and Computing Technologies, https://ieeexplore.ieee.org/document/7159459

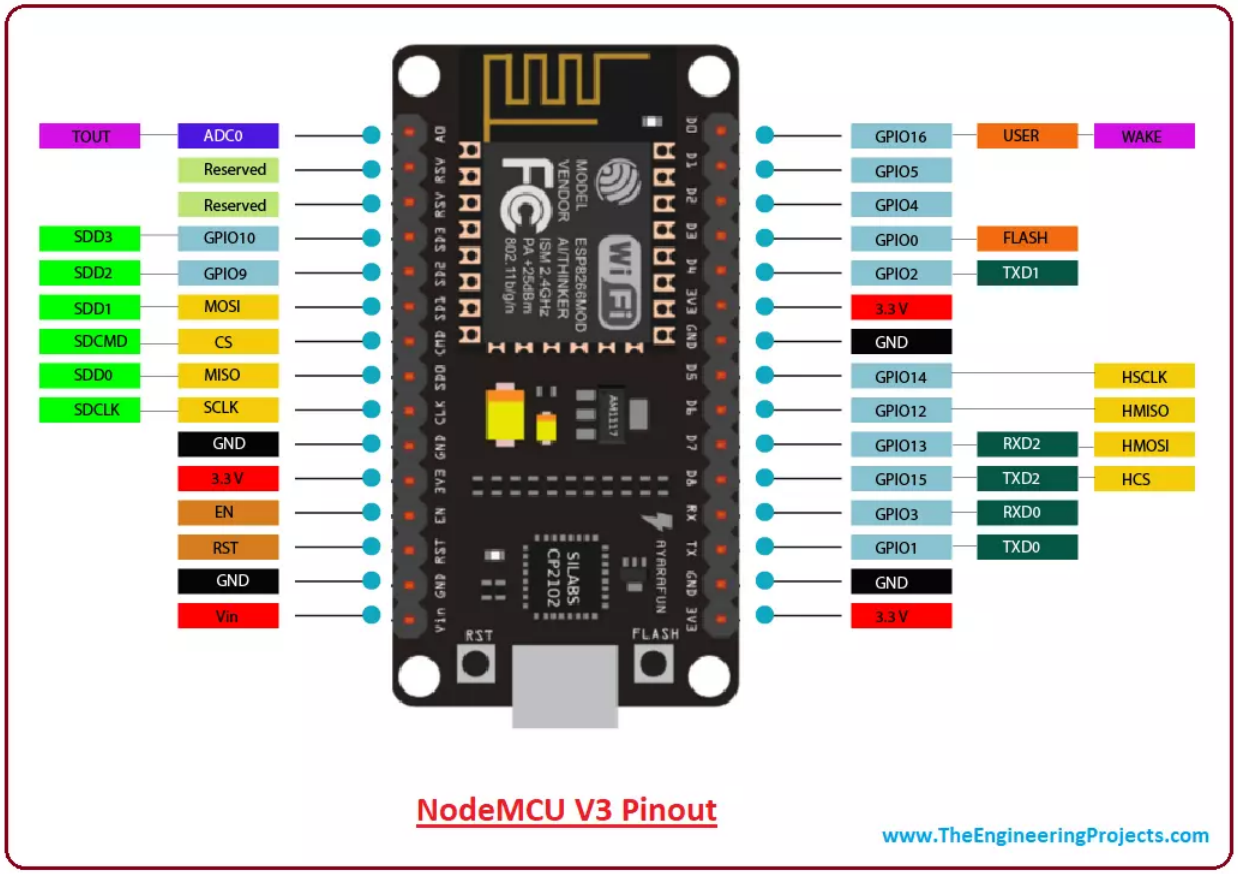
Appendices

SYSTEM COMPONENT LIST

1. Dc water pump
2. Lm7805 voltage regulator
3. Vero board
4. Connecting wire
5. Seven segment display
6. 240/12v, 1000ma transformer
7. Soldering iron
8. 30ph capacitor
9. 10uf 16v capacitor
10. Rectifier diodes
11. 2200uf/25v capacity
12. Connecting wire
13. Seven segment display
14. 240/12v, 1000ma transformer
15. Soldering iron
16. 2200uf/25v capacity
17. Connecting wire
18. Seven segment display
19. 240/12v, 1000ma transformer
20. Soldering iron
21. Soldering iron
22. 2200uf/25v capacity
23. Connecting wire
24. Seven segment display
25. 240/12v, 1000ma transformer
26. Soldering iron
27. Relay
28. Water pipes

NODEMCU PIN CONFIGURATION DATASHEET

Category	Name	Description
Power	Micro-USB, 3.3V, GND Vin	**Micro-USB:** NodeMCU can be powered through the USB port pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	 of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but you have to find which pin is I2C.
Inbuilt LED	13	Turns on the inbuilt LED
TWI	SDA, SCA	Used for TWI communication



NodeMCU V3 Pinout

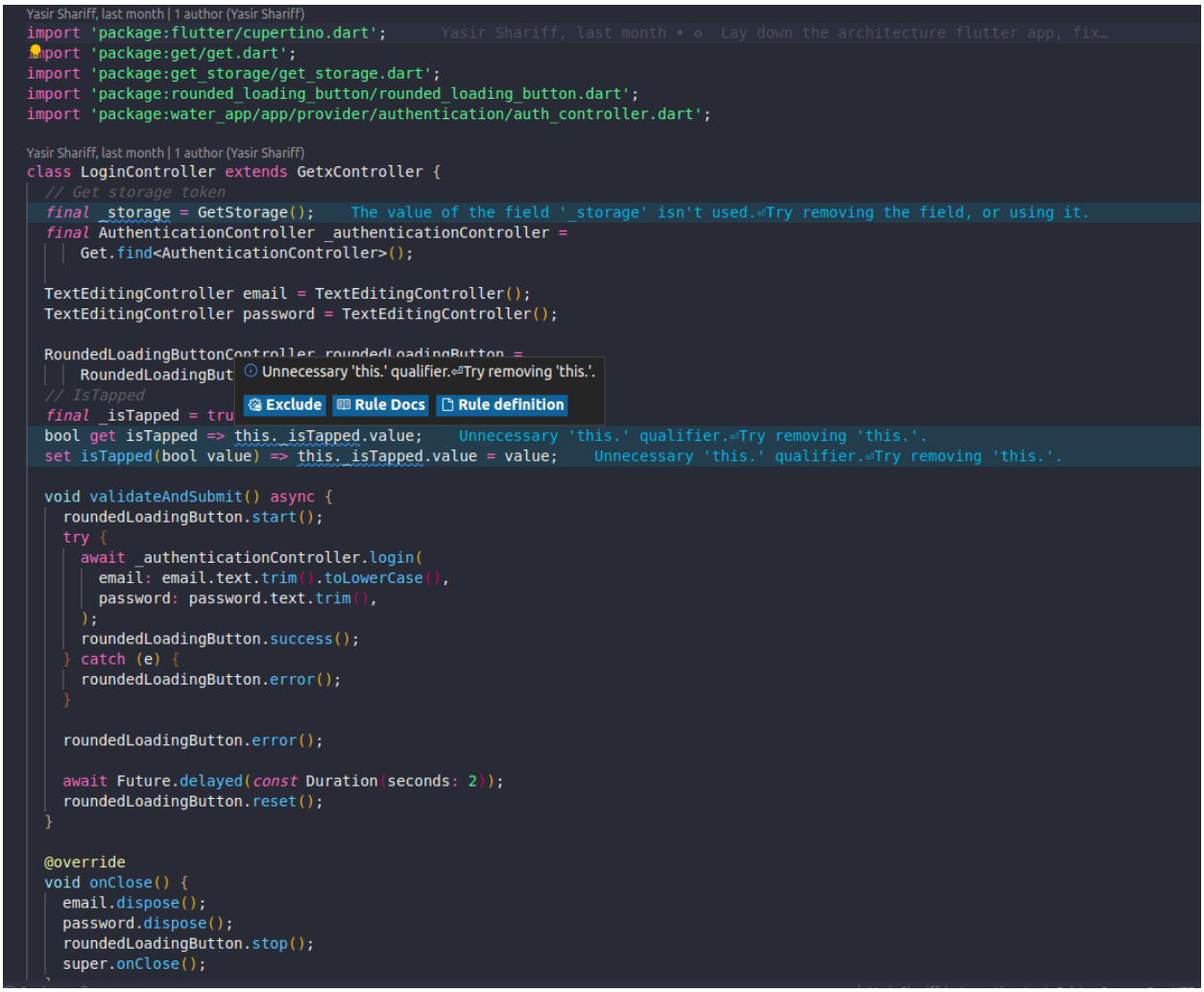
www.TheEngineeringProjects.com

Program Listings

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <HttpClient.h>
#include <NewPing.h>

// Define the WiFi credentials
const char *ssid = "Alpha";
const char *password = "Banshee42";

// Define the IP address and port of the server
const char *serverAddress = "172.16.0.103";
int serverPort = 3000;

// Define User credentials
const String email = "yasir@gmail.com";
const String userPassword = "123456";

// Define the pins used for the water and chlorine level sensors, pumps, and LEDs
const int waterTrigPin = 14;
const int waterEchoPin = 12;
const int chlorineTrigPin = 15;
const int chlorineEchoPin = 10;
int waterPumpInput = 2;
int treatmentPumpInput = 13;
int LedRed = 16;
int LedGreen = 9;
const int turbidityPin = A0;

// Define some global variables used for timing
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 1000 * 60 * 2; // 120 seconds

// Define some variables used to store data
long tankDuration;
long chlorineDuration;
long waterLevelPercentage;
long chlorineLevelPercentage;
int tankLevel;
int chlorineLevel;
float turbidity;
String token;

// Define the HTTP client object and the WiFi client object
WiFiClient client;
HttpClient httpClient = HttpClient(client, serverAddress, serverPort);

// Define the LCD object
```

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <HttpClient.h>
#include <NewPing.h>

// Define the WiFi credentials
const char *ssid = "Alpha";
const char *password = "Banshee42";

// Define the IP address and port of the server
const char *serverAddress = "172.16.0.103";
int serverPort = 3000;

// Define User credentials
const String email = "yasir@gmail.com";
const String userPassword = "123456";

// Define the pins used for the water and chlorine level sensors, pumps, and LEDs
const int waterTrigPin = 14;
const int waterEchoPin = 12;
const int chlorineTrigPin = 15;
const int chlorineEchoPin = 10;
int waterPumpInput = 2;
int treatmentPumpInput = 13;
int LedRed = 16;
int LedGreen = 9;
const int turbidityPin = A0;

// Define some global variables used for timing
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 1000 * 60 * 2; // 120 seconds

// Define some variables used to store data
long tankDuration;
long chlorineDuration;
long waterLevelPercentage;
long chlorineLevelPercentage;
int tankLevel;
int chlorineLevel;
float turbidity;
String token;

// Define the HTTP client object and the WiFi client object
WiFiClient client;
HttpClient httpClient = HttpClient(client, serverAddress, serverPort);

// Define the LCD object
```

```dart
import 'package:water_app/app/provider/authentication/auth_controller.dart';    Unused import: 'package:water_app/app/provider/aut
import 'package:water_app/app/provider/pumps/pumps_controller.dart';

Yasir Shariff, 3 weeks ago | 1 author (Yasir Shariff)
class HomeController extends GetxController {
  @override
  void onInit() {
    super.onInit();
    setWaterPumpStatus();
    setTreatmentPumpStatus();
  }

  PumpsProvider pumpsProvider = PumpsProvider();

  //arduinoConnected
  var treatmentPump = false.obs;

  // Water pump
  var waterPump = false.obs;

  void toggleWaterPump() async {
    await pumpsProvider.toggleWaterPump();
    waterPump.value = !waterPump.value;
  }

  void toggleTreatmentPump() async {
    await pumpsProvider.toggleTreatmentPump();
    treatmentPump.value = !treatmentPump.value;
  }

  void setWaterPumpStatus() async {
    final waterStatus = await pumpsProvider.getWaterStatus();
    waterPump.value = waterStatus;
  }

  void setTreatmentPumpStatus() async {
    final treatmentStatus = await pumpsProvider.getTreatmentStatus();
    treatmentPump.value = treatmentStatus;
  }

  bool getWaterPumpStatus() {
    return waterPump.value;
  }

  bool getTreatmentPumpStatus() {
    return treatmentPump.value;
  }
}
```


Yasir Shariff, last month | 1 author (Yasir Shariff)
import 'package:flutter/cupertino.dart';
Yasir Shariff, last month • Lay down the architecture flutter app, fix…
import 'package:get/get.dart';
import 'package:get_storage/get_storage.dart';
import 'package:rounded_loading_button/rounded_loading_button.dart';
import 'package:water_app/app/provider/authentication/auth_controller.dart';
Yasir Shariff, last month | 1 author (Yasir Shariff)
class LoginController extends GetxController {
// Get storage token
final _storage = GetStorage();
The value of the field '_storage' isn't used.Try removing the field, or using it.
final AuthenticationController _authenticationController =
Get.find<AuthenticationController>();
TextEditingController email = TextEditingController();
TextEditingController password = TextEditingController();
RoundedLoadingButtonController roundedLoadingButton =
RoundedLoadingBut
Unnecessary 'this.' qualifier.Try removing 'this.'.
Exclude
Rule Docs
Rule definition
// IsTapped
final _isTapped = tru
bool get isTapped => this._isTapped.value;
Unnecessary 'this.' qualifier.Try removing 'this.'.
set isTapped(bool value) => this._isTapped.value = value;
Unnecessary 'this.' qualifier.Try removing 'this.'.
void validateAndSubmit() async {
roundedLoadingButton.start();
try {
await _authenticationController.login(
email: email.text.trim().toLowerCase(),
password: password.text.trim(),
);
roundedLoadingButton.success();
} catch (e) {
roundedLoadingButton.error();
}
roundedLoadingButton.error();
await Future.delayed(const Duration(seconds: 2));
roundedLoadingButton.reset();
}
@override
void onClose() {
email.dispose();
password.dispose();
roundedLoadingButton.stop();
super.onClose();

```
            password: password.text.trim(),
            confirmPassword: confirmPassword.text,
            firstName: firstName.text.trim(),
            lastName: lastName.text.trim(),
          );

          loadingButtonController.success();
        } catch (e) {
          loadingButtonController.error();

          await Future.delayed(const Duration(seconds: 2));
          loadingButtonController.reset();
        }
      }
    }

    @override
    void onClose() {
      email.dispose();
      password.dispose();
      firstName.dispose();
      lastName.dispose();
      confirmPassword.dispose();
      loadingButtonController.stop();
      super.onClose();
    }
  }
```

```dart
Yasir Shariff, last month | 1 author (Yasir Shariff)
import 'package:flutter/material.dart';      Yasir Shariff, last month • Add Register Page …
import 'package:get/get.dart';
import 'package:rounded_loading_button/rounded_loading_button.dart';

import '../../../provider/authentication/auth_controller.dart';

Yasir Shariff, last month | 1 author (Yasir Shariff)
class RegisterController extends GetxController {
  final AuthenticationController _authenticationController =
      Get.find<AuthenticationController>();
  GlobalKey<FormState> saveFormKey = GlobalKey<FormState>();

  RoundedLoadingButtonController loadingButtonController =
      RoundedLoadingButtonController();

// TextEditing Controllers
  TextEditingController email = TextEditingController();
  TextEditingController password = TextEditingController();
  TextEditingController confirmPassword = TextEditingController();
  TextEditingController firstName = TextEditingController();
  TextEditingController lastName = TextEditingController();

  // IsTapped
  final _isTapped = true.obs;
  bool get isTapped => this._isTapped.value;    Unnecessary 'this.' qualifier.⏎Try removing 'this.'.
  set isTapped(bool value) => this._isTapped.value = value;    Unnecessary 'this.' qualifier.⏎Try removing 'this.'.

  void validateAndSave() async {
    if (saveFormKey.currentState!.validate()) {
      loadingButtonController.start();
      try {
        await _authenticationController.register(
          email: email.text.trim().toLowerCase(),
          password: password.text.trim(),
          confirmPassword: confirmPassword.text,
          firstName: firstName.text.trim(),
          lastName: lastName.text.trim(),
        );

        loadingButtonController.success();
      } catch (e) {
        loadingButtonController.error();

        await Future.delayed(const Duration(seconds: 2));
        loadingButtonController.reset();
      }
    }
```

```
      const newPump = await Pumps.create({ userId: req.user.id });
      newPump.treatmentStatus = !newPump.treatmentStatus;
      await newPump.save();
      return successResponse(req, res, 'Pump toggled successfully', 200);
    }

    // Toggle the pump
    pump.treatmentStatus = !pump.treatmentStatus;

    // Save the pump
    await pump.save();

    return successResponse(req, res, 'Pump toggled successfully', 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
};

export const getWaterPumpStatus = async (req, res) => {
  logger.info('Getting water pump status');
  try {
    // Get the pump
    const pump = await Pumps.findOne({ where: { userId: req.user.id } });

    return successResponse(req, res, pump.waterStatus, 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
};

export const getTreatmentPumpStatus = async (req, res) => {
  logger.info('Getting treatment pump status');
  try {
    // Get the pump
    const pump = await Pumps.findOne({ where: { userId: req.user.id } });

    return successResponse(req, res, pump.treatmentStatus, 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
};
```

```
Yasir Shariff, 3 weeks ago | 1 author (Yasir Shariff)
import logger from '../../utils/logger';      Yasir Shariff, 3 weeks ago • Add pump toggle …
import { errorResponse, successResponse } from '../../utils';
// Import the pump model
const { Pumps } = require('../../db/models');

// Import the pump service
export const toggleWaterPump = async (req, res) => {
  logger.info('Toggling water pump');
  try {
    // Get the pump
    const pump = await Pumps.findOne({ where: { userId: req.user.id } });

    //   If it doesn't exist, create it
    if (!pump) {
      const newPump = await Pumps.create({ userId: req.user.id });
      newPump.waterStatus = !newPump.waterStatus;
      await newPump.save();
      return successResponse(req, res, 'Pump toggled successfully', 200);
    }

    // Toggle the pump
    pump.waterStatus = !pump.waterStatus;

    // Save the pump
    await pump.save();

    return successResponse(req, res, 'Pump toggled successfully', 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
};

export const toggleTreatmentPump = async (req, res) => {
  logger.info('Toggling treatment pump');
  try {
    // Get the pump
    const pump = await Pumps.findOne({ where: { userId: req.user.id } });

    //   If it doesn't exist, create it
    if (!pump) {
      const newPump = await Pumps.create({ userId: req.user.id });
      newPump.treatmentStatus = !newPump.treatmentStatus;
      await newPump.save();
      return successResponse(req, res, 'Pump toggled successfully', 200);
    }
```

```
};

export const getStatsRange = async (req, res) => {
  logger.info('Getting stats range');
  try {
    const { start, end } = req.query;

    const stats = await Stats.findAll({
      where: {
        userId: req.user.id,
        createdAt: {
          [Op.between]: [start, end],
        },
      },
    });
    return successResponse(req, res, stats, 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
};

export const getLatestStats = async (req, res) => {
  logger.info('Getting latest stats');
  try {
    const stats = await Stats.findAll({
      where: {
        userId: req.user.id,
      },
      order: [['createdAt', 'DESC']],
      limit: 1,
    });
    return successResponse(req, res, stats, 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
};
```

```
Yasir Shariff, 2 weeks ago | 1 author (Yasir Shariff)
import { Op } from 'sequelize';      Yasir Shariff, 2 months ago • Add stats api …
import logger from '../../utils/logger';
import { errorResponse, successResponse } from '../../utils';

const jwt = require('jsonwebtoken');

const { Stats } = require('../../db/models');

export const postStats = async (req, res) => {
  logger.info('Posting stats');
  try {
    // Workaround for now

    const { token } = req.body;

    // Decode token
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    const userId = decoded.id;

    const stats = await Stats.create({
      userId,
      chlorineLevel: req.body.chlorineLevel,
      ph: req.body.ph,
      waterLevel: req.body.waterLevel,
      turbidity: req.body.turbidity,
    });

    return successResponse(req, res, stats, 201);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
};

export const getStats = async (req, res) => {
  logger.info('Getting stats');
  try {
    const stats = await Stats.findAll({
      where: {
        userId: req.user.id,
      },
    });
    return successResponse(req, res, stats, 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
```

```
    return successResponse(req, res, 'User logged out successfully', null);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 400);
  }
};

export const getCurrentUser = async (req, res) => {
  logger.info('Getting current user');
  try {
    const user = await User.findOne({ where: { id: req.user.id } });
    return successResponse(req, res, user, 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 400);
  }
};
```

```
export const login = async (req, res) => {
  logger.info('Logging in user');
  try {
    const { email, password } = req.body;

    // Check if user exists
    const user = await User.findOne({ where: { email } });

    if (!user) {
      throw new Error('User does not exist');
    }

    // Check if password is correct
    const isPasswordCorrect = await Password.compare(user.password, password);

    if (!isPasswordCorrect) {
      throw new Error('Incorrect password');
    }

    // Generate token
    // eslint-disable-next-line max-len
    const token = jwt.sign({ id: user.id, email: user.email, createdAt: new Date() }, process.env.JWT_SECRET);

    const data = {
      token,
      user: {
        id: user.id,
        email: user.email,
      },
    };

    return successResponse(req, res, data, 200);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 400);
  }
};

export const logout = async (req, res) => {
  logger.info('Logging out user');
  try {
    req.headers['x-token'] = null;
    req.headers = {};
    return successResponse(req, res, 'User logged out successfully', null);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 400);
  }
};
```

```
Yasir Shariff, last month | 1 author (Yasir Shariff)
import jwt from 'jsonwebtoken';      Yasir Shariff, 2 months ago • Move to js, fix database …
import logger from '../../utils/logger';
import Password from '../../services/password';
import { errorResponse, successResponse } from '../../utils';

const { User } = require('../../db/models');

export const register = async (req, res) => {
  logger.info('Registering user');
  try {
    const {
      firstName, lastName, email, password, confirmPassword,
    } = req.body;

    // Check if passwords match
    if (password !== confirmPassword) {
      throw new Error('Passwords do not match');
    }

    // Check if user already exists
    const user = await User.findOne({ where: { email } });

    if (user) {
      throw new Error('User already exists');
    }

    // Hash password
    const hashedPassword = await Password.toHash(password);

    // Create user
    try {
      await User.create({
        email, password: hashedPassword, firstName, lastName,
      });
    } catch (error) {
      logger.error(error);
      throw new Error('Error creating user');
    }

    return successResponse(req, res, 'User created successfully', 201);
  } catch (error) {
    logger.error(error);
    return errorResponse(req, res, error.message, 'Failed', 500);
  }
};

export const login = async (req, res) => {
```

```
  httpClient.beginRequest();
  httpClient.get("/api/v1/pumps/treatmentStatus"); // get the status of the treatment pump
  httpClient.sendHeader("x-token", token);         // add the token to the header
  httpClient.endRequest();

  int statusCode = httpClient.responseStatusCode();
  String responseBody = httpClient.responseBody();

  // Parse the response JSON to get the token
  size_t tokenCapacity = JSON_OBJECT_SIZE(3);
  DynamicJsonBuffer tokenJsonBuffer(tokenCapacity);
  JsonObject &tokenRoot = tokenJsonBuffer.parseObject(responseBody);
  bool status = tokenRoot["data"].as<bool>();

  if (status)
  {
    digitalWrite(treatmentPumpInput, HIGH); // turn on the treatment pump
  }
  else
  {
    digitalWrite(treatmentPumpInput, LOW); // turn off the treatment pump
  }
}
```

```
  else
  {
    Serial.println("Error sending data");
  }
}

/// @brief This function toggle the water pump
/// @param void
/// @return void
/// @note This function is called in the loop function
void toggleWaterPump()
{
  // We are getting the status of the water pump from the server to toggle the relay
  httpClient.beginRequest();
  httpClient.get("/api/v1/pumps/waterStatus"); // get the status of the water pump
  httpClient.sendHeader("x-token", token);     // add the token to the header
  httpClient.endRequest();

  int statusCode = httpClient.responseStatusCode();
  String responseBody = httpClient.responseBody();

  size_t tokenCapacity = JSON_OBJECT_SIZE(3);
  DynamicJsonBuffer tokenJsonBuffer(tokenCapacity);
  JsonObject &tokenRoot = tokenJsonBuffer.parseObject(responseBody);
  bool status = tokenRoot["data"].as<bool>();

  if (status)
  {
    digitalWrite(waterPumpInput, HIGH); // turn on the water pump
  }
  else
  {
    digitalWrite(waterPumpInput, LOW); // turn off the water pump
  }
}

/// @brief This function toggle the treatment pump
/// @param void
/// @return void
/// @note This function is called in the loop function
void toggleTreatmentPump()
{

  httpClient.beginRequest();
  httpClient.get("/api/v1/pumps/treatmentStatus"); // get the status of the treatment pump
  httpClient.sendHeader("x-token", token);         // add the token to the header
  httpClient.endRequest();
```

```
  httpClient.endRequest();

  int statusCode = httpClient.responseStatusCode();
  String responseBody = httpClient.responseBody();

  // check the status code
  if (statusCode != 200)
  {
    Serial.println("Error logging in");
    return "";
  }

  // Parse the response JSON to get the token
  const size_t capacity1 = JSON_OBJECT_SIZE(3) + JSON_OBJECT_SIZE(2) + 2 * JSON_OBJECT_SIZE(1) + 200;
  DynamicJsonBuffer jsonBuffer1(capacity1);
  JsonObject &root1 = jsonBuffer1.parseObject(responseBody);
  Serial.println("Logged in successfully");

  return root1["data"]["token"];
}

/// @brief Send the data to the server
/// @param void
/// @info The data should be a JSON object
void sendData()
{
  // Create a JSON object to hold the data
  const size_t capacity = JSON_OBJECT_SIZE(5);
  DynamicJsonBuffer jsonBuffer(capacity);
  JsonObject &root = jsonBuffer.createObject();

  root["chlorineLevel"] = chlorineLevelPercentage;
  root["ph"] = 7.5;
  root["turbidity"] = turbidity;
  root["waterLevel"] = waterLevelPercentage;
  root["token"] = token;

  String requestBody;
  root.printTo(requestBody);
  httpClient.post("/api/v1/stats/new", "application/json", requestBody); //
  httpClient.endRequest();
  int statusCode = httpClient.responseStatusCode();
  String responseBody = httpClient.responseBody();

  if (statusCode == 201)
  {
    Serial.println("Data sent successfully");
  }
```

```
  if (chlorineLevel < 8)
  {
    digitalWrite(LedRed, HIGH);
    digitalWrite(LedGreen, LOW);
  }
  else
  {
    digitalWrite(LedGreen, HIGH);
    digitalWrite(LedRed, LOW);
  }

  toggleTreatmentPump();
  toggleWaterPump();

  Serial.println("Tank Level: " + String(waterLevelPercentage));
  Serial.println("CHL Level: " + String(chlorineLevelPercentage));

  // Send data to the server every 2 minutes
  if (currentMillis - startMillis >= period)
  {
    sendData();
    startMillis = currentMillis;
  }
}

/// @brief Login to the server and get the token
/// @return The token
/// @info The token is used to authenticate the user for subsequent requests
String login()
{
  // Create a JSON object to hold the login credentials
  const size_t capacity = JSON_OBJECT_SIZE(2);
  DynamicJsonBuffer jsonBuffer(capacity);
  JsonObject &root = jsonBuffer.createObject();
  root["email"] = email;
  root["password"] = userPassword;

  // Serialize the JSON object to a String
  String requestBody;
  root.printTo(requestBody);

  // Send a POST request to the login endpoint with the JSON data in the body
  httpClient.beginRequest();
  httpClient.post("/api/v1/auth/login", "application/json", requestBody);
  httpClient.endRequest();

  int statusCode = httpClient.responseStatusCode();
```

```ino
void loop()
{

  currentMillis = millis(); // get the current "time" (actually the number of milliseconds since the program started)

  tankLevel = waterSensor.ping_cm();

  chlorineLevel = chlorineSensor.ping_cm();

  waterLevelPercentage = map(tankLevel, 23, 3, 0, 100);

  chlorineLevelPercentage = map(chlorineLevel, 17, 2, 0, 100);

  turbidity = analogRead(turbidityPin);
  turbidity = map(turbidityPin, 0, 640, 100, 0);

  Serial.print("Turbidity : ");
  Serial.println(turbidity);

  // Prints the tankLevel on the Serial Monitor
  Serial.print("Tank : ");
  Serial.println(tankLevel);

  lcd.setCursor(14, 0);
  lcd.print(" ");
  lcd.setCursor(12, 0);
  lcd.print(waterLevelPercentage);
  lcd.setCursor(15, 0);
  lcd.print("%");

  lcd.setCursor(14, 1);
  lcd.print(" ");
  lcd.setCursor(12, 1);
  lcd.print(chlorineLevelPercentage);
  lcd.setCursor(15, 1);
  lcd.print("%");

  Serial.print("CHL : ");
  Serial.println(chlorineLevel);

  if (chlorineLevel < 8)
  {
    digitalWrite(LedRed, HIGH);
    digitalWrite(LedGreen, LOW);
  }
  else
  {
    digitalWrite(LedGreen, HIGH);
```

User Manual

Water Monitoring App

This is the user manual for the water monitoring system application! This app is designed to help you monitor the levels of water and treatment in your home or office. This manual will guide you through the app's features, including login, registration, dashboard, turbidity and pH sensors.

Getting Started

First, download and install the app from the App Store or Google Play Store. After installation, open the app and you will be prompted to either login or register.

Login

If you have already registered, you can log in to the app using your email address and password. Once you have successfully logged in, you will be taken to the dashboard.

Register

If you are a new user, you will need to register for an account. To register, tap on the register button and fill in the required information, including your name, email address, and password. Once you have successfully registered, you can log in to the app using your email address and password.

Dashboard

The dashboard is the main screen of the app. It provides you with an overview of the water and treatment levels in your home or office. The dashboard also displays the current pH and turbidity levels of your water.

Water and Treatment Levels

The water and treatment levels are displayed in a simple graphical format on the dashboard. You can see the current levels and track the changes over time. You can also set up notifications to alert you when the levels are too high or too low.

Turbidity Sensor

The turbidity sensor measures the clarity of the water. It helps you to detect any dirt or impurities in the water. The turbidity level is displayed on the dashboard in real-time.

pH Sensor

The pH sensor measures the acidity or alkalinity of the water. It helps you to maintain the pH levels of the water at an optimal level. The pH level is displayed on the dashboard in real-time.

Settings

The settings menu allows you to customize the app according to your preferences. You can set up notifications, change the display units, and update your account information.

Conclusion

This is the overall menu for the water app. With this app, you can easily monitor the levels of water and treatment in your home or office. You can also track the turbidity and pH levels of your water.