DEVELOPMENT OF A FRONTIER TECHNOLOGY-BASED SMART DIGITAL OPERATOR FOR RURAL WATER TREATMENT PLANTS

Mitigating against health issues arising from poor plant control and product water quality

Jeeten Nathoo, Dr Stephen Jones, Prof Mujahid Aziz, Prof Atanda Kamoru Raji, Ms Wendy Nkomo, Prof Dyllon Randall, Kevin Lacey

WATER RESEARCH COMMISSION

TT 958/25

DEVELOPMENT OF A FRONTIER TECHNOLOGY-BASED SMART DIGITAL OPERATOR FOR RURAL WATER TREATMENT PLANTS

Mitigating against health issues arising from poor plant control and product water quality

A Final Report

to the Water Research Commission

by

Jeeten Nathoo[1], Dr Stephen Jones[2], Prof Mujahid Aziz[3], Prof Atanda Kamoru Raji[3], Ms Wendy Nkomo[3], Prof Dyllon Randall, Kevin Lacey

[1] Saintelligent Dynamics (Pty) Ltd,
[2] Radian Consulting (Pty) Ltd
[3]Cape Peninsula University of Technology
[4]University of Cape Town

Report no. TT 958/25
ISBN 978-0-6392-0719-3

June 2025

WATER
RESEARCH
COMMISSION

This is the final report of WRC Project No. C2021/2022-01015.

EXECUTIVE SUMMARY

Background and Rationale

The increasing complexity and scale of decentralised water treatment systems present significant operational challenges, particularly in rural and resource-constrained environments. Conventional management models rely on manual interventions, periodic inspections, and delayed responses to system anomalies, exacerbating risks associated with poor water quality and plant performance failures. The Smart Water Operations Platform (SWOP) was developed to address these challenges by integrating frontier digital technologies – including Internet of Things (IoT), real-time analytics, and remote system automation – into water treatment operations.

The SWOP is designed to function as an autonomous, low-cost, plug-and-play supervisory platform capable of continuous water quality monitoring and process control. By leveraging cost-effective microcontrollers, advanced sensor networks, and cloud-based data management, the system aims to improve operational reliability, enhance decision-making, and mitigate health risks associated with suboptimal water treatment practices.

Objectives and Scope

The primary objective of this research was to design, develop, and evaluate a scalable Smart Water Operations Platform (SWOP) tailored for rural water treatment plants. The study focused on:

Hardware and Software Development – Designing a sensor-integrated monitoring platform that operates on a low-power microprocessor backbone (ESP32, Arduino Mega) for real-time water quality assessment.
Validation through Controlled Testing – Calibrating and wet-testing key parameters, including electrical conductivity, pH, oxidation-reduction potential (ORP), turbidity, pressure, and flow rate using reference instruments.
Cloud-based Remote Monitoring – Integrating the system with Blynk IoT for real-time data transmission and remote oversight.
Performance Assessment and Refinement – Evaluating system accuracy, consistency, and response time under laboratory-simulated operational conditions.

System Design and Implementation

The SWOP architecture comprises three primary subsystems:

1. **Data Collection Subsystem:** A network of analogue sensors measuring water quality parameters, interfaced with an ESP32 microcontroller and an Arduino Mega for data acquisition.
2. **Data Transmission Subsystem:** Real-time wireless communication using Wi-Fi-enabled IoT frameworks for secure cloud data logging and analytics.
3. **Data Management and Control Subsystem:** An embedded processing unit capable of autonomous decision-making, integrating historical trends and real-time alerts to facilitate predictive maintenance and anomaly detection.

Key Findings and Performance Evaluation

- **Sensor Accuracy and Calibration**: Laboratory wet-testing confirmed high correlation with commercial reference meters, with deviations remaining within acceptable tolerance limits.
- **Electrical Conductivity (EC):** The sensor demonstrated reliable performance across a dynamic range of ionic concentrations, with minor deviations at higher salinity levels due to calibration constraints.

- **pH Measurement:** The SWOP pH sensor exhibited near-zero deviation compared to reference instruments, indicating excellent linearity and stability.
- **Oxidation-Reduction Potential (ORP):** Measurements tracked expected trends with minimal deviation, confirming sensor suitability for chlorine-based disinfection monitoring.
- **Turbidity Measurement:** Sensor response was consistent with theoretical predictions, though on-site calibration is required for application-specific threshold settings.
- **Pressure and Flow Rate:** Mechanical validation showed expected trends, though a calibrated test rig is recommended for higher precision evaluation.

The comparative analysis of IoT platforms (ThingSpeak vs. Blynk) led to the selection of Blynk, which provided better real-time connectivity, security, and user-friendly visualization tools for data-driven decision-making.

Conclusions and Recommendations

The research successfully demonstrated that the Smart Water Operations Platform (SWOP) represents a feasible, cost-effective, and technically robust solution for enhancing water quality monitoring and operational efficiency in decentralised water treatment plants. The findings support the deployment of this technology for remote plant supervision, predictive fault detection, and automated process control, with direct implications for public health and water security in rural areas.

Future enhancements should focus on:

1. **Enclosure and Structural Improvements:** Developing a robust housing with translucent covers for better sensor protection.
2. **Enhanced Test Rig Calibration:** Constructing a dedicated calibration platform for flow and pressure sensor standardization.
3. **Optimised Microcontroller Integration:** Evaluating ESP32-based breakout boards with built-in voltage regulation, reducing reliance on Arduino Mega.
4. **LED-Based Status Indication:** Implementing multi-tier LED indicators for real-time status monitoring.
5. **Higher Processing Capabilities:** Investigating Raspberry Pi as a potential alternative for higher-order analytics and user-interface enhancements.

This study provides a technological foundation for next-generation smart water treatment systems, with a strong potential for real-world deployment, scalability across multiple treatment sites, and adaptation to diverse water quality challenges.

ACKNOWLEDGEMENTS

The project team wishes to thank the following people for their contributions to the project.

Reference Group	Affiliation
Dr Nonhlanhla Kalebaila	Water Research Commission (Chairman)
Dr Enos Sitabule	Sasol Technology
Prof Johannes Sekomeng Modise	Vaal University of Technology
Dr Lynette Mariah Baratta	SASOL Technology
Prof Craig Sheridan	University of the Witwatersrand
Prof Oluwasen Oyekola	Cape Peninsula University of Technology
Dr Mfandaidza Hove	Anglo Research
Mr Mluleki Mnguni	Umgeni Water
Ms Megan Dharmalingum	eThekwini Metro
Prof Masike Malatji	UNISA
Ms Rachalet Grobbelaar	DWS
Mr Mark Bannister	DWS
Mr Kevin Lacey	Alternate Water Solutions

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS & ABBREVIATIONS

AFM	Activated Glass Filter Media
AI	Artificial Intelligence
BIRM	Birm (a type of water filtration media)
BLE	Bluetooth Low Energy
BNC	Bayonet Neill‚ÄìConcelman (a type of connector)
EC	Electrical Conductivity
EEPROM	Electrically Erasable Programmable Read-Only Memory
ESP32	A low-cost microcontroller with built-in Wi-Fi and Bluetooth
IoT	Internet of Things
KCl	Potassium Chloride
LCD	Liquid Crystal Display
LoRa	Long Range (a type of low-power wireless network)
MCU	Microcontroller Unit
MQTT	Message Queuing Telemetry Transport (a messaging protocol)
NTU	Nephelometric Turbidity Units
ORP	Oxidation Reduction Potential
PLC	Programmable Logic Controller
RMSE	Root Mean Square Error
Raspberry Pi	A single-board computer used in embedded systems
SANS	South African National Standards
SWOP	Smart Digital Operator
TDS	Total Dissolved Solids
TSS	Total Suspended Solids
USB	Universal Serial Bus
USEPA	United States Environmental Protection Agency
V	Volt
Wi-Fi	Wireless Fidelity (a wireless networking technology)
mS/cm	MilliSiemens per centimeter (a unit of electrical conductivity)

GLOSSARY

Term	Definition
Activated Glass Filter Media (AFM)	A highly engineered filter media made from recycled glass that provides superior filtration compared to traditional sand filters.
Artificial Intelligence (AI)	A branch of computer science that enables machines to simulate human intelligence processes, such as learning, reasoning, and self-correction.
Bayonet Neill–Concelman (BNC) Connector	A type of coaxial connector used for RF (radio frequency) signals, instrumentation, and video applications.
BIRM	A granular filter media used to remove iron and manganese from water through catalytic oxidation.
Bluetooth Low Energy (BLE)	A wireless communication technology designed for low power consumption, commonly used in IoT devices and sensor networks.
Cloud Computing	The delivery of computing services (such as storage, processing, and networking) over the internet allows for remote data access and management.
ESP 32	A low-cost microcontroller with built-in Wi-Fi and Bluetooth, commonly used in IoT applications for data collection and transmission.
Electrical Conductivity (EC)	The ability of water to conduct an electric current, usually measured in microSiemens per centimetre (µS/cm) or milliSiemens per centimetre (mS/cm).
Electrically Erasable Programmable Read-Only Memory (EEPROM)	A type of non-volatile memory used in electronic devices to store small amounts of data that must be saved when power is removed.
Frontier Technology	Emerging and advanced technologies such as AI, IoT, blockchain, and 5G that push the boundaries of innovation and connectivity.
Industruino	An industrial-grade version of the Arduino microcontroller, designed for embedded and automation applications.
Internet of Things (IoT)	A network of interconnected devices that collect, share, and process data over the internet without human intervention.
Message Queuing Telemetry Transport (MQTT)	A lightweight messaging protocol used for communication between IoT devices and cloud platforms.
Microcontroller Unit (MCU)	A compact integrated circuit designed to perform a specific operation within an embedded system, such as sensor data processing or automation control.
MilliSiemens per centimetre (mS/cm)	A unit of electrical conductivity used to measure the ion concentration in water, commonly used in water treatment and quality assessment.
Nephelometric Turbidity Units (NTU)	A measurement unit used to quantify the cloudiness of a liquid caused by suspended solids.
Oxidation Reduction Potential (ORP)	A measure of the ability of a solution to either gain or lose electrons, indicating its oxidative or reductive strength.
Programmable Logic Controller (PLC)	An industrial computer used for the automation of electromechanical processes, such as the control of machinery or treatment plants.
Raspberry Pi	A small, single-board computer used for embedded applications, education, and industrial automation.
Smart Water Operations Platform (SWOP)	An autonomous system designed to manage and monitor rural water treatment plants by using sensors, data analytics, and cloud integration.

Total Dissolved Solids (TDS)	The combined content of all inorganic and organic substances dissolved in water, typically measured in milligrams per liter (mg/L).
Total Suspended Solids (TSS)	The total concentration of particulate matter that remains suspended in water affecting clarity and quality.
United States Environmental Protection Agency (USEPA)	The US federal agency responsible for regulating environmental policies and protecting human health through environmental laws.
Universal Serial Bus (USB)	A standard connection interface used for data transfer and power supply between devices.

CHAPTER 1: BACKGROUND

1.1 INTRODUCTION

South Africa is a water-scarce country facing increasing stress on its already limited water resources. This scarcity is projected to intensify due to the effects of climate change, which manifest in unpredictable rainfall patterns, prolonged droughts, and rising average temperatures that accelerate evaporation rates. These factors collectively exacerbate existing challenges in water availability and accessibility, particularly in rural and underserved regions.

A major constraint in the supply of bulk municipal treated water to remote rural communities forces these communities to seek alternative water sources. Consequently, the reliance on groundwater from local communal boreholes is widespread in both urban and rural areas across South Africa (Palamuleni and Akoth, 2015). However, many of these boreholes yield suboptimal quantities of water and, critically, often suffer from poor water quality. The deterioration of groundwater quality is largely attributed to various anthropogenic activities, posing significant health risks to local communities, many of whom consume this water without adequate treatment (Odiyo and Makunga, 2017).

Despite the installation of modern water treatment plants – typically incorporating iron removal, media filtration, and chlorination – there exists a widespread misconception that rural community members can independently manage these systems after receiving short-term training (Korlam et al., 2016). In reality, sustained operation and maintenance require continuous external support, which is frequently lacking due to resource constraints and inadequate planning by local authorities. Without such support, these treatment systems often fail within a few months, leading to operational breakdowns, degradation, and, in many cases, vandalism of plant components for resale. Even in cases where the plants remain operational, improper maintenance and the absence of effective disinfection protocols significantly increase health risks, such as microbiological contamination within filtration media that has not been backwashed for extended periods.

The sustainable operation of water treatment systems depends on the availability of appropriately skilled personnel, adequate financial resources, and well-structured support mechanisms to ensure that plants function within their design parameters. Ideally, these systems should incorporate some degree of automation and control, facilitated through a programmable logic controller (PLC) that regulates standard process parameters. In addition to automation, a skilled or semi-skilled plant operator or process controller is required to monitor, operate, maintain, and service the plant effectively.

The level of human oversight required depends on the complexity of the treatment processes and the capacity of the plant. Facilities with a treatment capacity exceeding 750 m^3/day generally necessitate full-time onsite personnel for daily monitoring and operation. In contrast, smaller plants with lower treatment capacities can typically function with periodic visits for monitoring and servicing.

Decentralised groundwater treatment package plants deployed in remote rural areas are relatively small, typically treating between 10 and 100 m^3/day. Despite their modest size, maintaining these plants at the required operational standard presents several challenges, particularly in ensuring that treated water consistently meets health and safety standards. The frequency of monitoring and servicing required to maintain compliance is often difficult to achieve using conventional human-operated approaches due to several key limitations:

1. The scarcity of skilled personnel available to operate, maintain, and troubleshoot these plants.
2. Limited budgets allocated for routine maintenance and operational support.
3. The overwhelming number of decentralised treatment plants, coupled with the vast geographical distances between them, relative to the availability of qualified operators and maintenance personnel.

Given these constraints, an alternative approach is necessary to enhance the operational reliability of these treatment plants, enable timely detection and resolution of system failures, and support local authorities in coordinating rapid response actions. Ensuring the continuous provision of safe potable water to these communities requires a shift toward smarter, more resilient water treatment solutions.

Frontier technologies – defined as those leveraging digitalisation and connectivity – offer significant potential in addressing these challenges. These technologies encompass artificial intelligence (AI), the Internet of Things (IoT), big data analytics, blockchain, 5G connectivity, 3D printing, robotics, drones, gene editing, nanotechnology, and solar photovoltaics (Geetha and Gouthami, 2017).

Integrating some of these advancements into decentralised water treatment systems presents an opportunity to enhance process automation, improve real-time monitoring capabilities, and facilitate predictive maintenance, ultimately supporting the provision of safe and reliable drinking water in rural South African communities.

1.2 PROJECT AIMS

This study focused on the development of an innovative, ultra-low-cost, plug-and-play, frontier-technology-based Smart Water Operations Platform (SWOP) designed for integration with rural water treatment plants. The SWOP can either be retrofitted to existing treatment facilities or supplied as part of new water treatment systems. Its primary function was to autonomously bridge the technical skills deficit and logistical challenges associated with deploying a human plant operator (process controller) to remote sites on a routine basis.

The specific objectives of this study were as follows:

1. **Development of the Smart Water Operations Platform (SWOP):** To design and develop an innovative, ultra-low-cost, plug-and-play Smart Water Operations Platform (SWOP)leveraging frontier technology. The system is based on a low-cost microprocessor backbone, such as Arduino or Industruino, and integrates cost-effective physical sensors to enable real-time monitoring and operational control of rural water treatment plants.
2. **Hardware and Software Testing:** To evaluate the performance of the system's hardware and software by testing its ability to accurately measure, convert, and interpret analogue signals from various sensors. These tests were conducted using both synthetic and actual groundwater samples sourced from a diverse range of locations to ensure robustness and adaptability.

In summary, this study aims to develop and demonstrate a novel technological advancement in water treatment systems – the Smart Water Operations Platform (SWOP) – which has the potential to make a measurable impact on public health in dispersed rural communities. By addressing the technical skills gap and logistical constraints associated with traditional human-operated water treatment systems, the SWOP represents a practical and scalable solution to improving water quality monitoring and management in remote areas.

CHAPTER 2: LITERATURE REVIEW

2.1 DECENTRALISED PACKAGE WATER TREATMENT PLANTS IN SOUTH AFRICA

Decentralised package water treatment plants are widely deployed across South Africa, particularly in rural villages that are geographically remote from centralised municipal bulk water treatment plants and reticulation networks. These systems serve as a critical solution for communities that lack access to conventional water supply infrastructure. Additionally, package plants are utilised to supplement existing municipal infrastructure, offering a means to rapidly expand treatment capacity or address acute service delivery challenges.

A comprehensive study by Korlam et al. (2016) conducted an extensive survey of the treatment technologies employed in South African package water treatment plants for the provision of potable water. The study also examined the operational and maintenance challenges associated with these systems. The key findings from this research were as follows:

1. **Widespread deployment in rural areas** – Package water treatment plants were found to be extensively used across South Africa, with the vast majority situated in rural locations where access to centralised municipal water services is limited.
2. **Deficiencies in technical skills and financial management** – A significant challenge identified was the lack of technical expertise required for the proper operation, maintenance, and servicing of these plants. Furthermore, inadequate financial management skills, particularly in handling cash flow and budget allocation, further compounded the operational difficulties.
3. **Potential benefits of automation and advanced control strategies** – The study highlighted that the implementation of sophisticated control strategies and the integration of automation equipment could improve plant performance. However, the effectiveness of such measures is contingent upon the availability of skilled personnel to maintain and troubleshoot control systems, as well as sustainable funding for the upkeep of instrumentation and automated equipment.
4. **Importance of built-in redundancy and stand-by equipment** – Given the remote locations of many rural package plants, access to maintenance workshops and technical personnel is often limited, leading to extended downtimes when repairs are required. The study emphasised the critical role of built-in redundancy and the availability of stand-by equipment in mitigating these issues. However, the risk of theft of costly spare components was also highlighted as a significant concern.
5. **Untapped potential of remote monitoring technologies** – The implementation of remote monitoring systems was identified as a promising solution to enhance process oversight and facilitate timely interventions. Despite the potential benefits, the study found that the adoption of such technologies remained largely unexplored across the extensive number of package water treatment plants surveyed.
6. **Integration of renewable energy for improved resilience and cost efficiency** – The study suggested that incorporating renewable energy technologies could help mitigate operational disruptions caused by power failures while also reducing overall operating costs.

According to Korlam et al. (2016), the surveyed package water treatment plants had treatment capacities ranging between 1 m^3/h and 250 m^3/h, with over 90% of the plants operating at capacities below 100 m^3/h. This distribution underscores the predominance of small-scale treatment facilities in rural settings, further emphasising the need for tailored operational strategies to ensure their long-term sustainability and effectiveness.

2.2 GENERAL FEED WATER CHARACTERISTICS & WATER TREATMENT UNIT PROCESSES

Feedwaters that require treatment are typically characterised as containing elevated concentrations of suspended solids and aqueous dissolved species coupled with the presence of microbiological contaminants. As a result, the treatment of these types of feed water is usually carried out in a multi-stage process before it is of a potable standard.

Large suspended particles and coarse solids present in the feed water are removed in a pre-treatment process such as screening, to avoid blocking up or damage caused to equipment in any subsequent downstream treatment processes.

Smaller sediment particles may be removed by gravity settling, flotation, filtration (both multi-media and membrane processes) or by skimming, amongst others, during the primary treatment process. The efficiency of the solid-liquid separation process can be enhanced by the addition of coagulants and flocculants. For dissolved organic compounds in the feedwater, a biological process can be used during secondary treatment to remove the organics and suspended solids. Any further constituents in the wastewater that are not removed by the secondary processes require individual treatment processes during the tertiary or advanced processes such as reverse osmosis.

When considering the treatment process for groundwater or surface water, these water sources are typically characterised as having elevated levels of suspended solids. In the case of groundwater, this is mainly due to the oxidation of dissolved iron and manganese when the water is exposed to air. In the case of surface water, the elevated suspended concentration of suspended solids is mainly attributed to clay, silica and runoff debris.

Microbiological contamination is also an issue due to the currently poor and deteriorating groundwater quality associated with diverse anthropogenic sources, leading to major health concerns for these local communities, many of whom consume this water without adequate treatment (Odiyo and Makunga, 2017).

2.3 TECHNOLOGY SCAN ON THE APPLICATION OF MICROPROCESSORS IN WATER QUALITY ONLINE

Continual technological advancements in the development of faster and more powerful microprocessors, low-cost sensors, wireless communication, cloud storage, machine learning and artificial intelligence, which are driving the broader adoption of Internet of Things (IoT) are increasingly being explored to provide disruptive solutions for smart water quality monitoring.

2.3.1 Building blocks of a smart water quality online monitoring system

The building blocks for a smart water quality system comprise three main subsystems as shown in Figure 1 (Geetha & Gouthami, 2017):

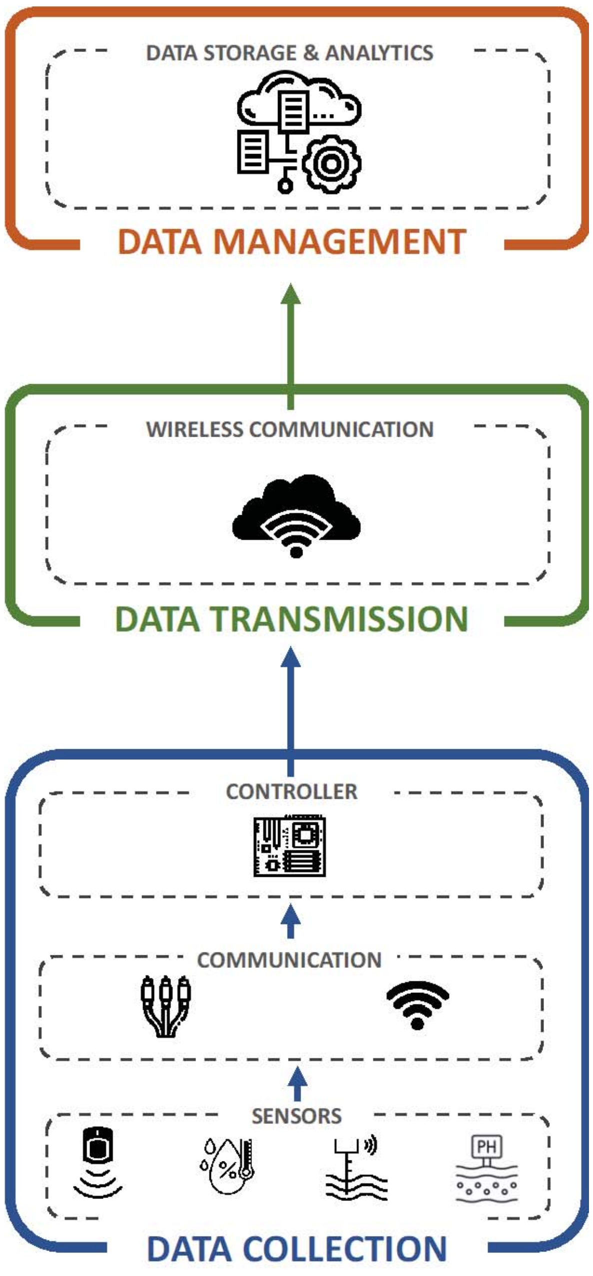


Figure 1: Building blocks for a smart water quality system (Geetha & Gouthami, 2017)

The data collection subsystem comprises a range of sensors with either direct or indirect measurements of the parameters required, which are either wired to or wirelessly connected to the controller to transfer the information from the sensor. The data transmission subsystem receives information from the data collection system and transmits this data to the data storage cloud. Finally, the data management subsystem allows remote access by the end user to the data through an application on their mobile device or desktop computer. The data management subsystem, together with the application, may or may not include data analytics, machine learning and artificial intelligence functionality.

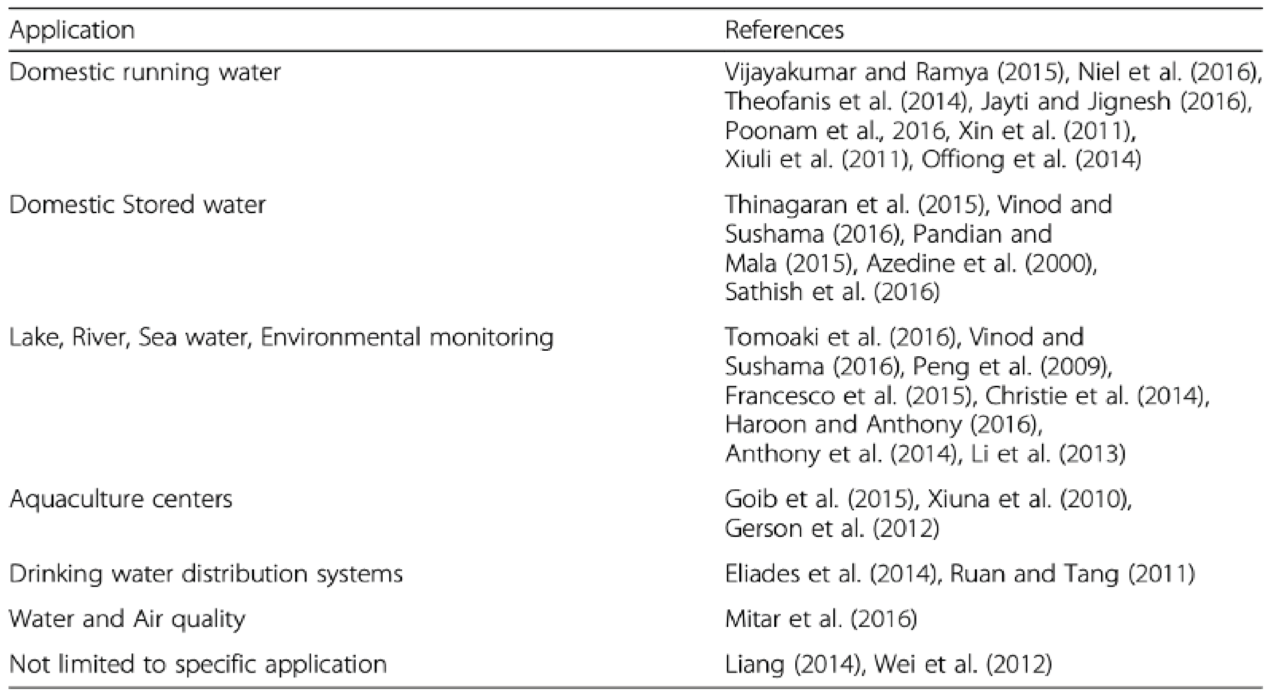
2.3.2 Application of smart water quality online monitoring systems

The use of smart water quality online monitoring systems has broad applicability in the domestic, industrial and municipal water space.

Geetha & Gouthami (2017) carried out an extensive review of the application of smart water quality monitoring.

A summary of these findings and the references are presented in Table 1:

Table 1: Application of smart water monitoring systems (Geetha & Gouthami, 2017)



Application	References
Domestic running water	Vijayakumar and Ramya (2015), Niel et al. (2016), Theofanis et al. (2014), Jayti and Jignesh (2016), Poonam et al., 2016, Xin et al. (2011), Xiuli et al. (2011), Offiong et al. (2014)
Domestic Stored water	Thinagaran et al. (2015), Vinod and Sushama (2016), Pandian and Mala (2015), Azedine et al. (2000), Sathish et al. (2016)
Lake, River, Sea water, Environmental monitoring	Tomoaki et al. (2016), Vinod and Sushama (2016), Peng et al. (2009), Francesco et al. (2015), Christie et al. (2014), Haroon and Anthony (2016), Anthony et al. (2014), Li et al. (2013)
Aquaculture centers	Goib et al. (2015), Xiuna et al. (2010), Gerson et al. (2012)
Drinking water distribution systems	Eliades et al. (2014), Ruan and Tang (2011)
Water and Air quality	Mitar et al. (2016)
Not limited to specific application	Liang (2014), Wei et al. (2012)

2.3.3 Parameters measured with smart water quality online monitoring systems

Following an extensive study, the US Environmental Protection Agency (USEPA) concluded that the presence of chemical and biological contaminants has an impact on a range of measurable onsite or inline water quality parameters including electrical conductivity (EC), pH, turbidity, oxidation and reduction potential (ORP) (Theofanis et al., 2014). As a result, the ability for real-time monitoring of these parameters would allow for early detection of deviations in the water characteristics.

Geetha & Gouthami (2017) carried out an extensive review of the key parameters measured in terms of smart water quality monitoring. A summary of these parameters and the references is presented in Table 2:

Table 2: Key parameters measured with references (Geetha & Gouthami, 2017)

Parameters monitored	References
pH	Vijayakumar and Ramya (2015), Mitar et al. (2016), Tomoaki et al. (2016), Vinod and Sushama (2016), Niel et al. (2016), Goib et al. (2015), Theofanis et al. (2014), Peng et al. (2009), Jayti and Jignesh (2016), Poonam et al., 2016, Xin Wang (2011), Gerson et al. (2012), Pandian and Mala (2015), Liang (2014), Xiuna et al. (2010), Christie et al. (2014), Azedine et al. (2000), Offiong et al. (2014), Anthony et al. (2014), Sathish et al. (2016)
Dissolved Oxygen	Vijayakumar and Ramya (2015), Goib et al. (2015), Jayti and Jignesh (2016), Gerson et al. (2012), Liang (2014), Xiuna et al. (2010), Christie et al. (2014), Offiong et al. (2014), Anthony et al. (2014)
Oxidation reduction potential	Niel et al. (2016), Theofanis et al. (2014)
Temperature	Vijayakumar and Ramya (2015), Mitar et al. (2016), Niel et al. (2016), Theofanis et al. (2014), Peng et al. (2009), Jayti and Jignesh (2016), Poonam et al., 2016, Gerson et al. (2012), Pandian and Mala (2015), Liang (2014), Xiuna et al. (2010), Francesco et al. (2015), Christie et al. (2014), Azedine et al. (2000), Anthony et al. (2014)
Turbidity	Vijayakumar and Ramya (2015), Tomoaki et al. (2016), Vinod and Sushama (2016), Theofanis et al. (2014), Jayti and Jignesh (2016), Poonam et al., 2016, Gerson et al. (2012), Pandian and Mala (2015), Francesco et al. (2015), Offiong et al. (2014), Sathish et al. (2016)
Conductivity	Vijayakumar and Ramya (2015), Niel et al. (2016), Theofanis et al. (2014), Jayti and Jignesh (2016), Gerson et al. (2012), Francesco et al. (2015), Christie et al. (2014), Azedine et al. (2000), Anthony et al. (2014), Sathish et al. (2016)
Water level sensing	Thinagaran et al. (2015)
Flow sensing	Niel et al. (2016)
Air temperature	Mitar et al. (2016)
Relative Humidity	Mitar et al. (2016)
Presence of organic compounds	Mitar et al. (2016)
Chlorine concentration	Eliades et al. (2014), Francesco et al. (2015)
Chlorophyll	Francesco et al. (2015)

2.3.4 Controllers used with smart water quality online monitoring systems

The various controllers have been used in literature, and a list of these together with references are presented in Table 3:

Table 3: Key parameters measured with references (Geetha & Gouthami, 2017)

Controller used	References
AtMega	Thinagaran et al. (2015), Mitar et al. (2016)
PIC	Theofanis et al. (2014), Niel et al. (2016), Vinod and Sushama (2016)
Raspberry pi + IOT	Vijayakumar and Ramya (2015), Jayti and Jignesh (2016), Sathish et al. (2016)
ARM LPC	Francesco et al. (2015), Poonam et al. (2016)
Arduino	Anthony et al. (2014), Christie et al. (2014), Pandian and Mala (2015), Gerson et al. (2012)
8051	Li et al. (2013)
MSP430	Peng et al. (2009)
TI CC3200	Proposed work

CHAPTER 3: SWOP DESIGN CONCEPTS AND KEY FEATURES

3.1 INTRODUCTION

This chapter lays the foundational framework for understanding the Smart Water Operations Platform (SWOP), focusing on its core design philosophies, key system characteristics, and operational constraints. It delves into the rationale behind the SWOP's development, highlighting the need for a cost-effective, resilient, and technologically advanced solution tailored to the challenges of rural water treatment systems. By addressing both technical and operational dimensions, this chapter establishes the guiding principles that shaped the SWOP's architecture and functionality.

Through an exploration of design objectives, system integration strategies, and the specific environmental and logistical demands of remote water treatment facilities, this chapter aims to provide a comprehensive understanding of how the SWOP was conceptualised and engineered. It also outlines the scope and limitations of the system within the context of the prototype development, offering insights into both the current capabilities and future potential of the SWOP.

3.2 DESIGN PHILOSOPHY AND OBJECTIVES

To ensure that the Smart Water Operations Platform (SWOP) has a measurable impact in mitigating preventable health issues in dispersed rural areas (issues often resulting from the sub-optimal operation of geographically remote package water treatment plants), the following design philosophies and objectives were established. These principles aim to address the technical skills gap and logistical challenges associated with deploying human plant operators to remote sites regularly:

a) **Cost-effectiveness:** The SWOP should have extremely low manufacturing and deployment/retrofitting costs, in the order of 1/10 to 1/100th the cost of conventional PLC automation systems.
b) **Affordability with Comprehensive Integration:** The SWOP is expected to be significantly more cost-effective than conventional PLC systems. Most entry-level PLC units offer limited analogue inputs, which are essential for processing sensor data. Accommodating additional inputs requires expansion modules, leading to a substantial increase in the initial system cost. Furthermore, this cost does not include essential components such as enclosures, sensors, communication interfaces, human-machine interfaces (HMIs), and power supplies. In contrast, the Prototype 2 SWOP integrates all necessary sensors, enclosures, microcontrollers, and display units at a cost comparable to that of a basic PLC system alone, offering a more comprehensive and economically efficient solution.
c) **Ease of Maintenance:** The SWOP is designed to be cost-effective enough to justify keeping a spare unit on-site, enabling quick replacement in the event of a system failure. Its plug-and-play functionality allows a relatively unskilled person to easily remove and replace a faulty unit without the need to dispatch a skilled technician to a remote location for troubleshooting or servicing. Incorporating aviation-type jacks and BNC connectors into the system would further enhance this ease of maintenance, facilitating quick, secure, and reliable connections for effortless replacements and minimal downtime.
d) **Energy Independence:** The SWOP should be inherently designed as an off-grid system with low energy consumption, powered primarily by solar energy and supported by a standard lithium-ion or lithium iron phosphate ($LiFePO_4$) battery for backup energy storage. These battery chemistries provide sufficient current to power the system and offer impressive recharge cycles – up to 500 and 1,500 times, respectively – translating to a battery lifespan of approximately 3 to 10 years. To optimise performance and longevity, the battery system should be housed in a separate power pack enclosure. This design minimises heat generation from charging and battery management processes while enabling simplified “hot-swappable” functionality, allowing quick and easy battery replacement without disrupting system operation.

e) **Operational Resilience:** The SWOP must be robust enough to withstand the physical and security challenges associated with the environments in which it is expected to be deployed and operated.

3.3 THEORETICAL FRAMEWORK AND TECHNOLOGICAL FOUNDATIONS

The development of the Smart Water Operations Platform (SWOP) is rooted in both theoretical principles and practical technological advancements. This section outlines the theoretical framework that guided the ideal design and development of the SWOP, as well as the technological foundations that underpin its operation. While these elements represent the aspirational goals for the SWOP, the actual prototype developed in this study was subject to constraints, which are discussed later in this chapter.

3.3.1 Theoretical Framework

The theoretical framework for the SWOP is built upon interdisciplinary concepts from control systems engineering, environmental science, and information technology. The key theoretical underpinnings include:

i) **Systems Thinking:** The SWOP is conceptualised as an integrated system where hardware, software, and human interactions are interconnected. Systems thinking allows for holistic problem-solving, ensuring that the SWOP operates efficiently within the complex environments of rural water treatment plants.
ii) **Automation and Control Theory:** Principles from control theory are employed to design feedback mechanisms that maintain system stability and optimise performance. This includes the use of sensors for real-time data acquisition, controllers for decision-making, and actuators for executing control actions.
iii) **Data-Driven Decision Making:** The SWOP leverages data analytics to support informed decision-making. Theoretical models for data processing, such as statistical analysis and machine learning algorithms, are integral to the system's ability to predict failures, optimise operations, and enhance water quality management.
iv) **Resilience Engineering:** Given the challenging environments in which the SWOP operates, resilience engineering principles are applied to ensure robustness against failures, adaptability to changing conditions, and rapid recovery from disruptions.

3.3.2 Technological Foundations

The technological foundations of the SWOP are based on the integration of low-cost microcontrollers, advanced communication protocols, and innovative data processing techniques. Key technological components include:

i) **Microcontroller Platforms:** The use of Arduino Mega and ESP32 microcontrollers provides a cost-effective and versatile backbone for the SWOP. These platforms support extensive input/output capabilities, real-time processing, and seamless integration with various sensors.
ii) **Sensor Technologies:** The SWOP employs a range of sensors to monitor critical water quality parameters, such as pH, turbidity, electrical conductivity, and oxidation-reduction potential. These sensors are selected for their accuracy, durability, and compatibility with the microcontroller platforms.
iii) **Communication Protocols:** Reliable data transmission is achieved through a combination of wired (I2C, SPI) and wireless (Wi-Fi, Bluetooth) communication protocols. This hybrid approach ensures data integrity and flexibility in diverse operational contexts.

iv) **Cloud Computing and IoT Integration:** The SWOP is designed to interface with cloud-based platforms for remote monitoring and data analytics. This integration facilitates real-time data access, centralised management, and advanced analytics through Internet of Things (IoT) technologies.

v) **Artificial Intelligence and Machine Learning:** The incorporation of AI and machine learning algorithms enables the SWOP to perform predictive analytics, anomaly detection, and adaptive control. Tools such as TensorFlow Lite are utilised for on-device AI processing, enhancing the system's autonomy and decision-making capabilities.

3.4 KEY SYSTEM CHARACTERISTICS

The SWOP incorporates the following technical characteristics designed to optimise performance and reliability in rural water treatment applications. These characteristics represent the ideal configuration and capabilities of the SWOP. However, due to project-specific constraints discussed later in this chapter, not all features were implemented in the prototype developed for this study.

a) **Low-Cost Microprocessor Backbone:** The SWOP should employ a low-cost Arduino or Industruino-type microprocessor technology backbone, coupled with low-cost physical sensors, whose functionality can be extended towards becoming virtual sensors for real-time monitoring of physicochemical water quality parameters such as turbidity, residual chlorine (ORP), pH, TDS/EC, as well as system performance parameters (e.g., feed and product flow rates, pressures across unit processes, backwashing frequency, backwashing effectiveness, chemical tank levels, etc.).

b) **Data Transmission and Local Processing:** The sensor data collected by the Arduino microcontroller boards will be transmitted to a more powerful on-site, microprocessor-based single-board computer, such as a Raspberry Pi. This intermediate processing step serves two key purposes: to pre-process the data before it is sent to the cloud, mitigating the risk of data overload, and to enable real-time, on-site data analysis. Alternatively, an ESP32 microprocessor can be utilised to reduce overall system costs, as the data volume is relatively low. This approach can be implemented using an ESP32 Devboard V1, which is already managing communications between the Arduino and the Blynk Cloud.

c) **AI Integration:** AI-based analyses on the data will be carried out locally using a range of industry-standard, open-source AI and mathematical modelling tools, such as Google TensorFlow and Keras, which are well supported by the literature. TensorFlow Lite is capable of being run on an ESP microprocessor.

d) **Local and Cloud-Based AI Communication:** This local AI system will be able to communicate with a larger cloud-based AI infrastructure that is both able to direct the learning of the local AI network as it encounters new situations in the field and perform far more complicated AI analyses, which are not possible to accomplish on-site. The local AI will also be responsible for monitoring all locally collected sensor data to check the quality of the collected data.

e) **Flexible Communication Interfaces:** The Arduino-based sensor units will interface with the ESP32 Devboard V1 through either wireless protocols, such as Wi-Fi or Bluetooth, or via shielded electrical cables utilising an industry-standard bus protocol like I2C. In addition to serving as a data hub, the Arduino will also function as a display and diagnostic panel. This setup will allow plant operators to conveniently monitor the system's operational status in real-time through an LCD panel, which will display key sensor information for quick and easy reference.

f) **Data Storage Capabilities:** The addition of an SD card module will provide significant storage space to allow for large-scale, secure storage of all sensor and diagnostic data recorded on-site, effectively acting as a short to medium-term local backup of all plant data.

g) **Signal Accuracy and Conversion:** The system should be able to accurately interpret and convert analogue signals from the various sensors to digital signals for data analytics and management over a wide range of feed water characteristics, ensuring that the accuracy of the measured parameters is consistent with conventional laboratory-based probes, meters, and sensors.

h) **Cloud Integration:** The system should be designed to integrate with a centralised remote monitoring, data logging, and remote assistance system, together with a real-time dashboard. This larger cloud-based system will perform analytics for each plant, enable the fine-tuning of the local AI, and apply AI techniques to compare information between different plants, enhancing the performance of all connected systems.

i) **Mobile App Connectivity:** The system should have the capability to communicate with an oversight mobile app with push notifications that keep community managers, district maintenance technicians, and other stakeholders up to date on the real-time operational status of the water quality.

j) **Self-Diagnostics and Fault Reporting:** The system should have the capability for self-diagnostics, with the ability to detect and report potential faults to both on-site staff and the cloud-based system. In the event of critical failure, the system should safely shut down the plant and alert relevant personnel for prompt intervention.

k) **Electrical Isolation:** To ensure that faults arising in either the sensor components or industrial machines being monitored do not cause damage to the primary computing system hardware, all sensing and communication ports on the microcontrollers should be electrically isolated from the components being measured. Where necessary, the appropriate SANS standards for electrical isolation will be employed.

3.5 DESIGN CONSTRAINTS AND OPERATIONAL SCOPE

While the preceding sections outline the comprehensive range of potential features for the final SWOP system, the deliverables for this study were limited to the aspects of prototype development defined by the approved funding, focusing on specific prototype functionalities rather than full-scale system implementation.
These constraints shaped the development of the SWOP prototype, guiding decisions on component selection, system architecture, and functional capabilities.

Despite these limitations, the prototype still successfully demonstrated the core principles and objectives outlined in this chapter, laying the groundwork for future iterations and enhancements.

3.5.1 Operational Scope

Drawing on the feedwater characteristics outlined in Section 2.2 for both groundwater and surface water sources intended for potable use, the treatment process that served as the foundation for the development of the SWOP monitoring and control system comprised the following components:

- **Stage 1:** Media filtration with activated glass filter media (AFM) for the removal of total suspended solids (TSS) above 3 mg/L but lower than 15 mg/L in the feedwater.
- **Stage 2:** Removal of heavy metals (Fe and Mn) from the feedwater using Katalox or BIRM.
- **Stage 3:** Disinfection with sodium hypochlorite dosing for the elimination of microbiological contaminants, ensuring the correct level of residual free chlorine post-treatment to provide continued disinfection of the treated water during storage and distribution to the end user.

Furthermore, the definition of package plants within the context of this study includes those with the following characteristics:

1. Package plants with treatment capacities between 10–100 m^3/day.
2. Package plants capable of treating borehole or surface water with total suspended solids, turbidity, and microbiological contaminants above SANS 241-2015 specifications, but with total dissolved solids

concentrations and aqueous species levels within SANS 241-2015 limits (i.e., systems that include pressure media filtration and disinfectant dosing post-treatment, excluding reverse osmosis).

3. Key system parameters designed to be measured and reported include:
    - **Electrical Conductivity (EC):** To measure dissolved solids levels.
    - **Turbidity:** To indicate suspended solids concentration.
    - **Oxidation-Reduction Potential (ORP):** As a proxy for residual free chlorine.
    - **pH:** To measure the acidity or basicity of the water.
    - **Pressure:** To monitor filtration system performance.
    - **Flow Rate:** To assess system throughput and efficiency.

3.6 SUMMARY

In summary, this chapter has laid the foundational principles underpinning the design and development of the Smart Water Operations Platform (SWOP). The design philosophies and objectives were established to address critical challenges in rural water treatment systems, focusing on cost-effectiveness, energy efficiency, ease of maintenance, and operational resilience. The key system characteristics highlighted the integration of low-cost microcontrollers, advanced data processing capabilities, robust communication protocols, and self-diagnostic features designed to ensure reliability and sustainability in remote environments.

The operational scope and design constraints outlined the specific conditions under which the SWOP is intended to operate, providing clarity on the system's capabilities and limitations. These insights form the basis for understanding how the SWOP can effectively contribute to improving water quality management in resource-constrained settings.

The SWOP's system architecture is described in the subsequent chapter and covers the hardware components, communication protocols, and data processing mechanisms that enable the SWOP to perform accurate data acquisition, reliable transmission, and comprehensive data management. By examining each subsystem in detail, the subsequent chapter aims to provide a comprehensive understanding of the technical framework that supports the SWOP's innovative functionality.

CHAPTER 4: SYSTEM DESIGN

4.1 INTRODUCTION

The development of the Smart Water Operations Platform (SWOP) necessitates a robust, efficient, and cost-effective system design capable of ensuring accurate data acquisition, reliable transmission, and comprehensive data management. This chapter provides an in-depth exploration of the SWOP's system architecture, detailing the hardware components, communication protocols, and data processing mechanisms that underpin its operation.

The system hardware design is strategically divided into three primary subsystems:

i) Data collection subsystem
ii) Data transmission subsystem
iii) Data management subsystem

The Data Collection Subsystem is responsible for acquiring sensor data; the Data Transmission Subsystem facilitates the secure and reliable transfer of data; and the Data Management Subsystem is tasked with processing, analysing, and managing the collected data. Each subsystem is carefully integrated to optimise system performance, scalability, and adaptability, particularly for rural water treatment applications where resource constraints and operational challenges are common.

This chapter outlines the rationale behind the selection of key components, including microcontrollers such as the Arduino Mega and ESP32, and provides detailed descriptions of their roles within the system. Additionally, it discusses future considerations for system enhancements, including potential upgrades to more advanced processors like the Raspberry Pi to support higher-level data analytics and real-time monitoring capabilities

A block flow diagram of the system is shown in Figure 2:

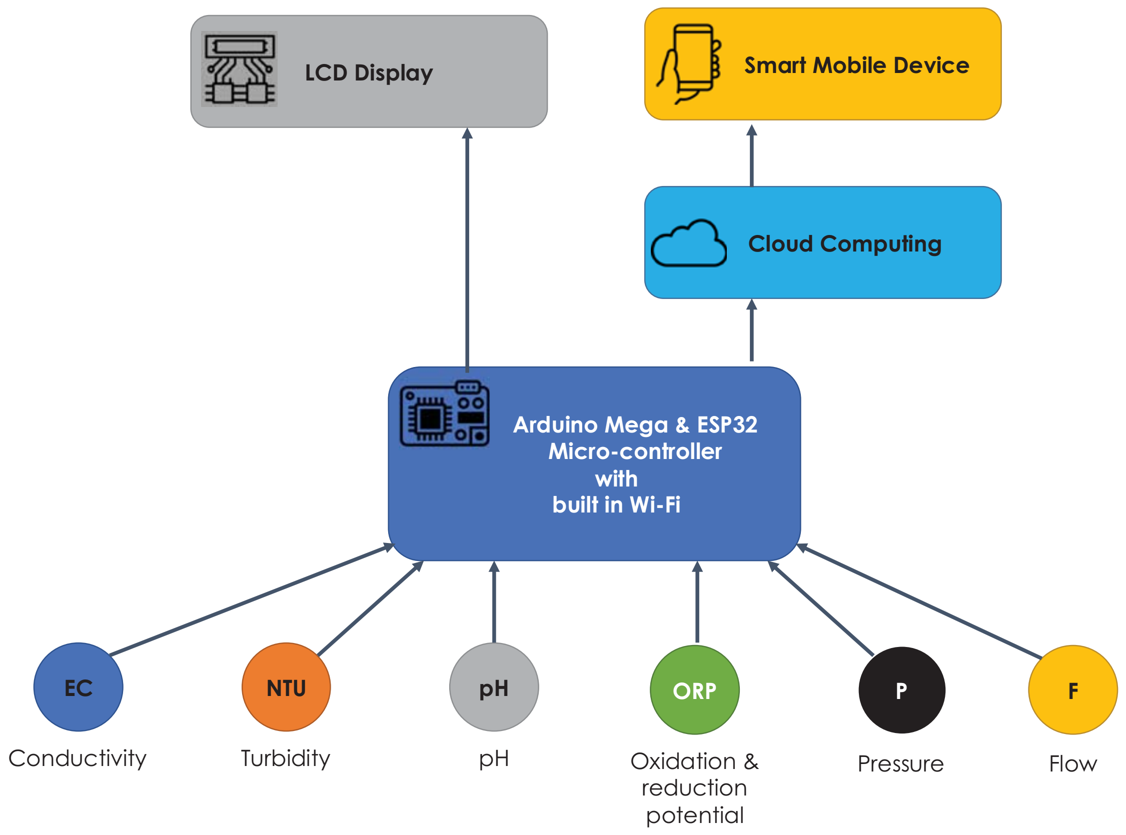


Figure 2: Block flow diagram of system

4.2 THE DATA COLLECTION SUBSYSTEM

The data collection subsystem consisted of individual sensors, including pH, electrical conductivity (EC), turbidity, ORP, pressure, and flow sensors, along with the communication technology required to transmit the collected data to the controller, and the controller itself.

Previous studies, as outlined in the literature review, primarily focused on the utilisation of low-cost microcontrollers, commonly available development boards, and higher-end boards produced by large, well-established manufacturers. While these solutions performed adequately in controlled laboratory environments, additional considerations were necessary to ensure reliable functionality within an industrial setting.

For instance, microcontrollers such as ATMega, PIC, 8051, CC3200, and MSP430 are component-based, requiring the design of custom printed circuit boards (PCBs) to support microcontroller programming, voltage regulation, and analogue front-end circuitry. This is essential to manage a wide range of impedance loads, along with other operational factors critical in industrial environments.

Alternatively, development boards like the Arduino Mega and ESP32 come equipped with integrated voltage regulation, as well as built-in capabilities for programming and debugging. However, these boards typically lack the instrumentation amplification required to support high-impedance sensors, and wireless communication features are often limited to more expensive models within their product lines.

In this study, the selected sensors—ORP, pH, and EC—were equipped with their own dedicated amplification boards, effectively mitigating compatibility issues with the chosen microcontrollers. As a result, the Arduino Mega was selected for the data collection subsystem due to its 5V logic level, which is fully compatible with all the sensors used, and its extensive range of analogue input ports. To support cloud communication and enable remote command transmission via a serial monitor, the ESP32 system-on-chip was also integrated into the subsystem.

Ultimately, the combination of the Arduino Mega and ESP32 microcontrollers provided a cost-effective, reliable, and scalable solution for data collection in this research. Both microcontrollers are illustrated in Figure 3.

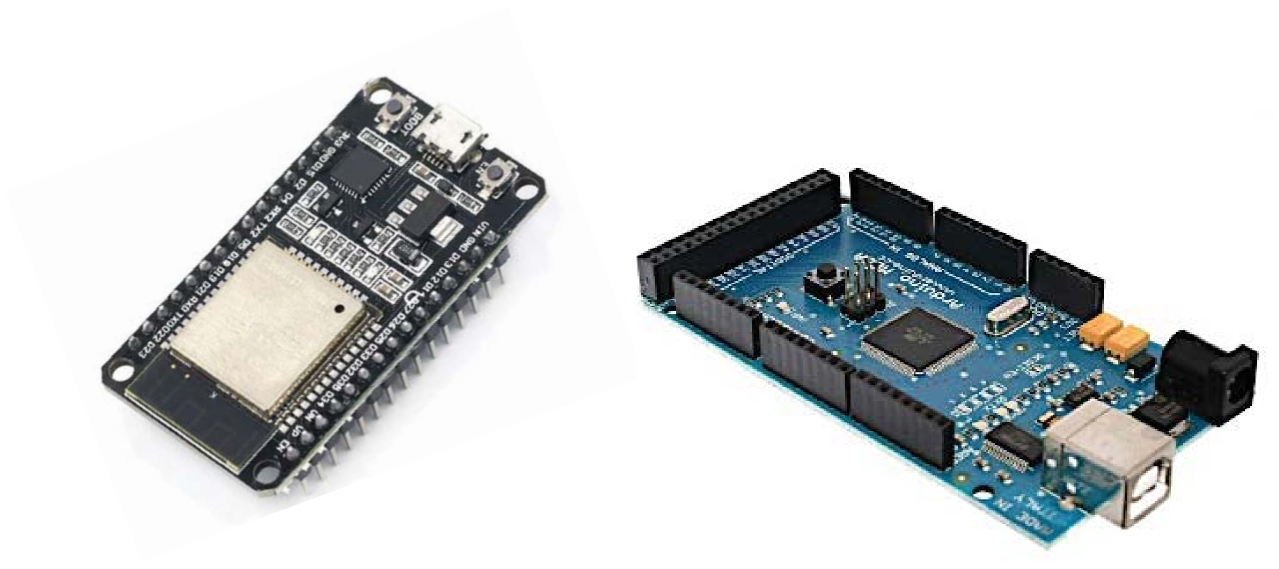


Figure 3: ESP32 Microcontroller

The ESP32 was selected for this study due to its low cost, integrated Wi-Fi and Bluetooth Low Energy (BLE) capabilities, the ability to securely store wireless security keys locally, and its capacity to run localized AI applications. Furthermore, the ESP32 has a proven track record in industrial applications, making it a reliable choice for this system.

The sensors were intentionally chosen to interface with the ESP32 through dedicated interface boards rather than direct connections. These interface boards were responsible for receiving raw sensor inputs, applying the necessary electrical corrections, and converting the data into a format compatible with the ESP32. For diagnostic purposes, the system also allowed for direct reading of raw sensor data into the ESP32, enabling more granular analysis when required.

The ESP32 board handled serial digital data inputs from the interface boards, converting these inputs into meaningful measurements, such as pH, turbidity, ORP, electrical conductivity (EC), pressure, and flow. At this stage of the system, the ESP32 employed statistical and heuristic methods to assess whether the sensor data reflected normal operating conditions. If anomalies were detected—either through comparison with predefined threshold values or significant deviations from historical data—a local alarm could be triggered. This alarm system, intended for future prototypes, could include indicators such as warning lights or audible alerts connected directly to the sensor node.

Additionally, the system enabled real-time display of live sensor data on an LCD monitor connected directly to the ESP32-powered sensor node. After acquiring and processing the sensor data, the ESP32 transmitted the calculated measurements to the data management subsystem via the transmission subsystem, as detailed in the following subsection.

4.3 THE DATA TRANSMISSION SUBSYSTEM

The data transmission subsystem comprises both the data itself and the transmission media and protocols used to transfer sensor data from individual nodes within the data collection subsystem to the data management subsystem.

As previously mentioned, all ESP32 units are equipped with built-in wireless communication capabilities, including Wi-Fi and Bluetooth Low Energy (BLE). Given the compact size of containerised water treatment plants, these wireless protocols can be effectively leveraged to eliminate the need for extensive signal cabling. However, this approach increases costs, as each sensor requires its own microprocessor and dedicated power source.

Despite the higher costs, wireless communication offers distinct advantages, particularly when retrofitting the system into existing plants, even those with pre-installed wired infrastructure. It simplifies both maintenance and fault-finding since any data discrepancies can be quickly isolated to specific sensor nodes, eliminating the potential for issues related to signal cables between nodes and the data management subsystem.

For this research project, a wired approach using shielded cables was adopted to connect the sensor nodes to the data management subsystem. This decision was driven by the goal of minimising costs, as it reduces the number of required components while still ensuring reliable data transmission.

4.4 THE DATA MANAGEMENT SUBSYSTEM

The data management subsystem consisted of the computational hardware responsible for collecting sensor data from individual nodes, applying statistical and algorithmic transformations to the data, displaying the processed information to an operator, and transmitting the data off-site to a cloud-based system for further analysis.
In addition to data processing and transmission, the data management subsystem was tasked with raising plant-wide alarms in the event of system malfunctions. This functionality could not be achieved by individual sensors alone, as it required a comprehensive understanding of data from all sensors within the system to accurately assess operational status.

For this research, the Arduino Mega and ESP32 were selected as the combined controllers for the data management subsystem. The Arduino Mega was responsible for managing data inputs and system control functions, while the ESP32 facilitated wireless communication and cloud connectivity. To ensure seamless data exchange between the 5V logic of the Arduino Mega and the 3.3V logic of the ESP32, a logic level shifter was incorporated into the system.

Images of the Arduino Mega and ESP32 microcontrollers are presented in Figure 3, with the logic level shifter depicted in Figure 4:

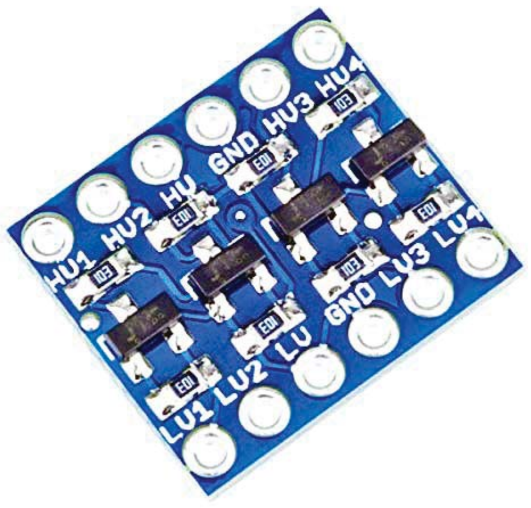


Figure 4: Logic Level Shifter

While the ESP32 was utilised for this project, future iterations of the system would greatly benefit from incorporating a more advanced processor, such as the Raspberry Pi. The Raspberry Pi offers built-in Wi-Fi and Bluetooth Low Energy (BLE) transceivers, enabling direct connectivity to all ESP32-powered sensor nodes without the need for additional hardware development.

The decision to consider the Raspberry Pi over a standard microcontroller was driven by its capability to run a full-fledged desktop operating system. This feature allows for the development of an intuitive, user-friendly interface that can be easily visualised by plant operators using conventional input devices such as a mouse, keyboard, and screen. This interface would enhance real-time monitoring and system control within the data management subsystem.

Moreover, the Raspberry Pi supports advanced mathematical and statistical software packages, including Scientific Python, Numerical Python, Google TensorFlow, Keras AI, and others. The availability of these tools would enable sophisticated local data analysis, allowing for real-time assessment of plant performance at a level that would require substantial development effort in conventional microcontroller-based systems.

The modular architecture of the Raspberry Pi also facilitates flexible data transmission to cloud-based platforms via a wide range of communication protocols. The system can be easily adapted to include transceiver modules for GSM, LoRa, general RF radio, and more. Given the variability of network infrastructure in remote locations – where reliable cellular connectivity or access to a LoRa gateway may not always be available – this flexibility in transmission protocols provides a significant advantage for ensuring consistent and reliable data communication.

Based on the system selected, a preliminary connection diagram is shown in Figure 5:

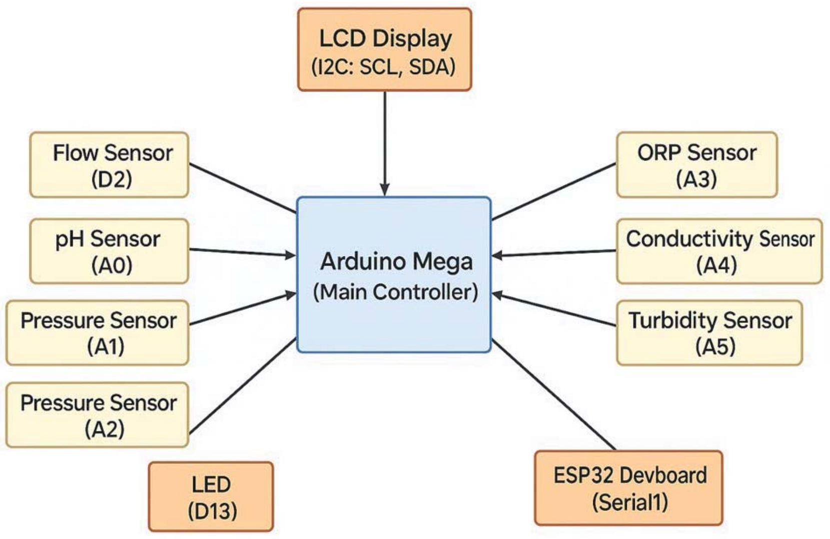


Figure 5: SWOP Prototype Connection Diagram

A summary of the basic operating program architecture is shown in Figure 6:

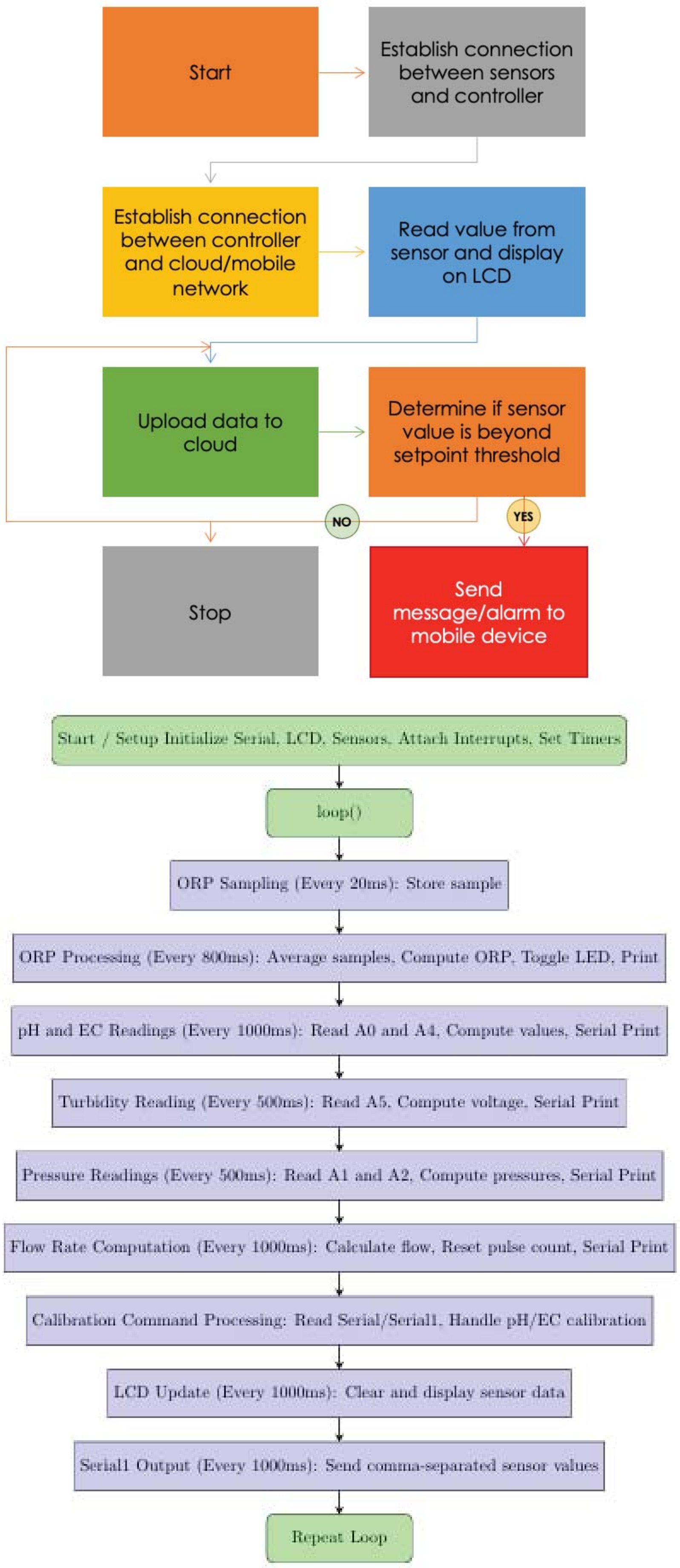

Start
Establish connection between sensors and controller
Establish connection between controller and cloud/mobile network
Read value from sensor and display on LCD
Upload data to cloud
Determine if sensor value is beyond setpoint threshold
NO
YES
Stop
Send message/alarm to mobile device
Start / Setup Initialize Serial, LCD, Sensors, Attach Interrupts, Set Timers
loop()
ORP Sampling (Every 20ms): Store sample
ORP Processing (Every 800ms): Average samples, Compute ORP, Toggle LED, Print
pH and EC Readings (Every 1000ms): Read A0 and A4, Compute values, Serial Print
Turbidity Reading (Every 500ms): Read A5, Compute voltage, Serial Print
Pressure Readings (Every 500ms): Read A1 and A2, Compute pressures, Serial Print
Flow Rate Computation (Every 1000ms): Calculate flow, Reset pulse count, Serial Print
Calibration Command Processing: Read Serial/Serial1, Handle pH/EC calibration
LCD Update (Every 1000ms): Clear and display sensor data
Serial1 Output (Every 1000ms): Send comma-separated sensor values
Repeat Loop

Figure 6: Basic operating program architecture

4.5 SUMMARY

This chapter presented a comprehensive overview of the Smart Digital Operator’s (SWOP) system design, encompassing the data collection, transmission, and management subsystems. Each subsystem was meticulously designed to ensure seamless integration, operational efficiency, and scalability. The use of Arduino Mega and ESP32 microcontrollers provided a cost-effective and reliable platform for data acquisition and processing, while the combination of wired and wireless communication protocols ensured robust data transmission, even in challenging environments.

The data management subsystem demonstrated the capability to process and analyse sensor data effectively, triggering alarms for system anomalies and supporting remote monitoring via cloud-based platforms. Moreover, the chapter highlighted potential future enhancements, including the integration of advanced processors such as the Raspberry Pi to support more sophisticated data analytics and real-time system control.

Overall, the system design reflects the SWOP’s core objectives: to deliver a low-cost, plug-and-play solution that enhances the efficiency, reliability, and sustainability of water treatment systems, particularly in resource-constrained rural areas. This foundational design supports the continued development of innovative, data-driven approaches to water quality management and operational optimisation.

With the system’s core data processing and management hardware defined, the subsequent chapter will focus on the selection of sensors that interface with the microcontrollers to enable real-time water quality monitoring and control.

CHAPTER 5: SELECTION OF SENSORS

5.1 INTRODUCTION

The effective operation of the Smart Water Operations Platform (SWOP) in rural water treatment systems relies heavily on the accuracy, reliability, and durability of the sensors integrated into the system. Sensors play a critical role in monitoring key water quality parameters, enabling real-time data acquisition for informed decision-making and system optimisation. This chapter presents a detailed overview of the key sensors selected for the SWOP, focusing on their technical specifications, functional roles, and integration within the system.

The selection criteria were based on factors such as measurement accuracy, compatibility with low-cost microcontrollers (e.g., Arduino and ESP32), power efficiency, ease of calibration, and operational stability under varying environmental conditions. The sensors described in this chapter cover essential parameters, including electrical conductivity, turbidity, oxidation-reduction potential (ORP), pH, pressure, and flow rate. Each section provides a comprehensive description of the sensor’s operating principles, key features, and technical specifications, demonstrating how they contribute to the SWOP's overall performance in water quality monitoring and control.

5.2 ANALOG ELECTRICAL CONDUCTIVITY SENSOR

The Analog Electrical Conductivity Meter V2 was specifically designed to measure the electrical conductivity of aqueous solutions. It supported a wide voltage input range of 3 to 5V, making it compatible with both 5V and 3.3V main control boards. The output signal was hardware-filtered to minimise jitter, ensuring stable and reliable readings.

The meter utilised an AC excitation source, which effectively reduced the polarisation effect, enhanced measurement accuracy, and extended the lifespan of the probe. Additionally, the accompanying software library featured a two-point calibration method and was capable of automatically recognising standard buffer solutions, making the calibration process simple, efficient, and user-friendly.

Signal Conversion Board (Transmitter) V2 Specifications:

- Supply Voltage: 3.0~5.0V
- Output Voltage: 0~3.4V
- Probe Connector: BNC
- Signal Connector: PH2.0-3Pin
- Measurement Accuracy: ±5% F.S.
- Board size: 42 x 32 mm

Electrical Conductivity Probe Specifications:

- Probe Type: Laboratory Grade
- Cell Constant: 1.0
- Support Detection Range: 0~20ms/cm
- Recommended Detection Range: 1~15ms/cm
- Temperature Range: 0~40°C
- Probe Life: >0.5 year (depending on frequency of use)
- Cable Length: 100cm

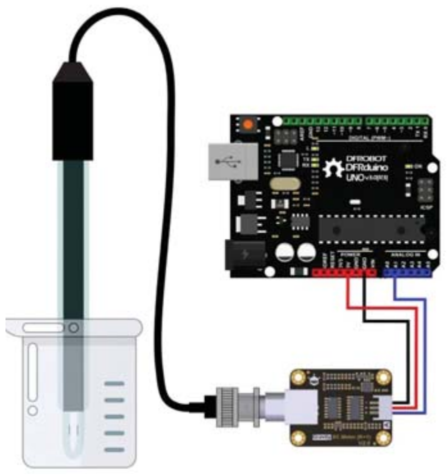


Figure 7: Analog electrical conductivity sensor

5.3 TURBIDITY SENSOR

The turbidity sensor was employed to measure turbidity levels, indicating the concentration of suspended solids in the water. The sensor operated by using light to detect suspended particles, measuring both light transmittance and scattering rates, which varied in response to changes in total suspended solids (TSS) concentration. An increase in TSS corresponded to higher turbidity levels in the water.

This sensor is commonly utilised for monitoring water quality in rivers and streams, assessing wastewater and effluent, controlling instrumentation for settling ponds, supporting sediment transport research, and conducting laboratory-based measurements. In this study, the sensor provided both analogue and digital signal output modes. The digital output mode included an adjustable threshold, allowing for flexibility based on specific measurement requirements. The appropriate output mode was selected to ensure compatibility with the microcontroller unit (MCU) used in the system.

Turbidity Sensor Specifications:

- Operating Voltage: 5V DC
- Operating Current: 40mA (MAX)
- Response Time: <500ms
- Insulation Resistance: 100M (Min)
- Analog output: 0-4.5V
- Digital Output: High/Low level signal (you can adjust the threshold value by adjusting the potentiometer)
- Operating Temperature: 5℃~90℃
- Storage Temperature: -10℃~90℃
- Weight: 30g
- Adapter Dimensions: 38mm*28mm*10mm/1.5inches *1.1inches*0.4inches

Figure 8: Turbidity sensor

5.4 ANALOG OXIDATION REDUCTION POTENTIAL SENSOR

The ORP (Oxidation-Reduction Potential) sensor was utilised to measure the oxidative and reductive capacity of aqueous solutions, expressed in millivolts (mV). ORP values indicated the solution's tendency to either gain or lose electrons, reflecting its relative oxidising or reducing potential. A positive ORP value indicated an oxidising environment, often signifying the presence of residual free chlorine, which is critical for disinfection processes.

The ORP probe employed in this study featured a platinum indicator electrode paired with a silver-silver chloride reference electrode. This configuration enabled continuous 24-hour online ORP monitoring and significantly enhanced the durability of the sensor. Additionally, a signal converter was integrated to provide a more stable voltage reference, ensuring that sensor readings remained unaffected by fluctuations in the supply voltage, thereby improving measurement accuracy and reliability.

ORP Signal Converter Specification:

- Input Voltage: 5V
- Input Signal: 0.5V ~ 4.5V
- Input Interface: 5.08mm/0.20" Pluggable Connector
- Measuring Range: -2000mV ~ +2000mV
- Output Interface: PH2.0 - 3Pin
- Module Dimensions: 42 x 32mm

Industrial ORP Probe Specification:

- Indicator Electrode: Platinum
- Reference Electrode: silver-silver chloride
- Suitable Temperature: 5-70℃
- Electrode Potential: 245mV ~ 270mV
- Reference Electrode Internal Resistance: ≤10KΩ
- Electrode Stability: ±8mV/24h

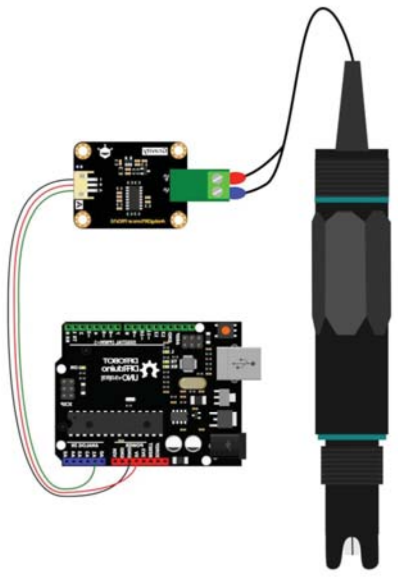


Figure 9: ORP sensor

5.5 PH SENSOR

The analogue pH sensor was utilised to measure the pH of aqueous solutions, indicating their acidity or alkalinity. The sensor's design incorporated an onboard voltage regulator chip, supporting a wide supply voltage range of 3.3 to 5.5V, making it compatible with both 5V and 3.3V main control boards.

To enhance measurement stability, the output signal was hardware-filtered to minimise jitter. The accompanying software library implemented a two-point calibration method, allowing for automatic recognition of two standard buffer solutions (pH 4.0 and 7.0). This calibration approach ensured improved accuracy and ease of use during pH measurements.
Signal Conversion Board (Transmitter) Specifications:

- Supply Voltage: 3.3~5.5V
- Output Voltage: 0~3.0V
- Probe Connector: BNC
- Signal Connector: PH2.0-3P
- Measurement Accuracy: ±0.1@25℃
- Dimension: 42 x 32mm

pH Probe Specifications:

- Probe Type: Laboratory Grade
- Detection Range: 0~14
- Temperature Range: 5~60°C
- Zero Point: 7±0.5
- Response Time: <2min
- Internal Resistance: <250MΩ
- Probe Life: >0.5 year (depending on frequency of use)
- Cable Length: 100cm

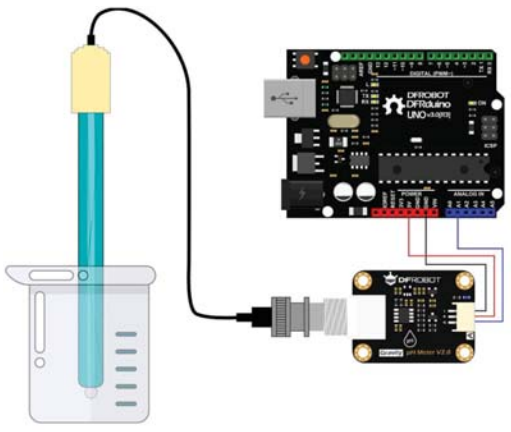


Figure 10: pH sensor

5.6 PRESSURE SENSOR

The pressure transducer used in this study was a high-performance piezo-resistive ceramic pressure sensor designed for accurate and reliable pressure measurement. It featured a specialised ASIC (Application-Specific Integrated Circuit) amplifier and a robust stainless steel housing, which contributed to its stability, durability, and exceptional sensitivity and consistency in various operating conditions.

The transducer was temperature-compensated and linearity-corrected, ensuring precise measurements across a range of environmental conditions. Additionally, it was engineered to effectively withstand the effects of water hammer and transient over-voltage, enhancing its resilience and long-term performance in dynamic fluid systems.

Pressure Sensor Specifications:

- Pressure Range: 0 -1mPa
- Analog Output: 0.5-4.5V (Output Signal is 90% proportional to Vdd)
- Power Supply (Vdd): 3.3-5.5V
- Connector: Aviation Type
- Pressure Port: G1/4
- Insulation Impedance (250Vdc): 50 MΩ
- Housing Material: 304 Stainless Steel

Figure 11: Pressure sensor

5.7 FLOW SENSOR

The water flow sensor utilised in this study consisted of a magnetic core, a rotating impeller, an external casing, and a Hall-effect sensor. As water flowed through the sensor, the impeller rotated, activating the magnetic core. This interaction generated pulse signals, which were detected by the Hall-effect sensor.

By measuring the frequency of these pulse signals, the flow rate of the water was accurately determined. The sensor proved suitable for monitoring flow rates in various applications, offering a reliable and straightforward method for real-time flow detection.

Flow Sensor Specifications:

- Flow range:1-60L/min
- Interior diameter: 20mm (0.81")
- Maximum current: 15 mA(DC 5V)
- Working voltage range: DC 4.5-18 V
- Load capacity: 10 mA(DC 5V)
- Operating Temp: 80°C
- Operating humidity:35%-90%RH
- Water pressure < 1.20 Mpa
- Flow = 4.8 * units of flow (L / min) * time (seconds)
- Connector Type: Female 3-Pin JST-SMP

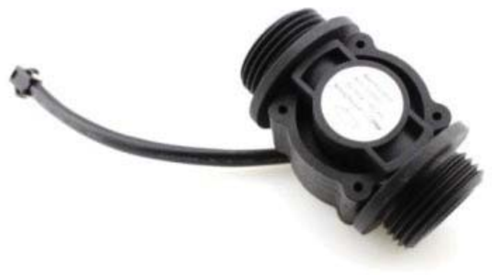


Figure 12: Flow sensor

5.8 SUMMARY

In this chapter, the selection and specification of key sensors for the Smart Water Operations Platform (SWOP) were detailed, highlighting their critical roles in monitoring and maintaining water quality within rural water treatment systems. Each sensor, ranging from electrical conductivity and turbidity to ORP, pH, pressure, and flow, was chosen based on stringent criteria that prioritised accuracy, reliability, cost-effectiveness, and compatibility with the system’s microcontroller backbone.

The integration of these sensors ensures that the SWOP can perform real-time data acquisition and environmental monitoring with high precision, supporting automated control mechanisms and facilitating timely interventions when system anomalies are detected. The combination of robust sensor technologies and efficient data management enhances the operational sustainability of water treatment facilities, particularly in resource-constrained environments. This strategic selection of sensors forms the foundation for the SWOP's ability to deliver reliable, data-driven insights, thereby improving water quality management in rural and remote areas.

CHAPTER 6: EVALUATION OF IOT PLATFORMS FOR THE SMART WATER OPERATIONS PLATFORM (SWOP) SYSTEM

6.1 INTRODUCTION

The development of the Smart Water Operations Platform (SWOP) for rural water treatment systems requires a robust, cost-effective, and scalable Internet of Things (IoT) platform to facilitate remote monitoring, control, and data analysis. This chapter focuses on the evaluation and selection of two prominent IoT platforms, ThingSpeak and Blynk, both of which were considered for integration with the SWOP. The aim was to determine the most suitable platform to aggregate, visualise, and analyse live data streams from various sensors and devices, supporting the operational needs of rural water treatment facilities. The chapter provides an in-depth analysis of each platform's architecture, data handling capabilities, advantages, limitations, and cost implications, culminating in a justified selection based on comprehensive technical criteria.

6.2 THINGSPEAK

ThingSpeak, developed by MathWorks in 2010, is a cloud-based IoT analytics platform designed to collect, visualise, and analyse live data streams from IoT devices. It allows users to aggregate data from multiple sensors, enabling comprehensive monitoring and analysis of environmental and operational parameters.

ThingSpeak's core strength lies in its seamless integration with MATLAB, providing powerful tools for advanced data analytics, predictive modelling, and trend analysis. This feature makes ThingSpeak particularly effective in prototyping and research-focused applications where complex data processing is required.

6.2.1 Architecture

The ThingSpeak architecture consists of three main components:

- **ThingSpeak Application:** Provides a user-friendly web-based interface where users can create channels for data storage and real-time visualisation. Each channel can handle up to eight data fields, allowing for the monitoring of multiple parameters simultaneously. Users can customise dashboards with widgets for graphical data representation.
- **ThingSpeak Server:** Acts as the central hub for data collection, analysis, and storage. It supports standard IoT protocols such as MQTT, HTTP, and REST APIs, enabling easy device integration. The server also facilitates data security and management, ensuring reliable performance even in high-demand applications.
- **Hardware Devices:** ThingSpeak is compatible with a wide range of IoT hardware, including Arduino boards, ESP8266, Raspberry Pi, and other microcontrollers. These devices collect sensor data and transmit it to the ThingSpeak server using secure communication protocols.

6.2.2 Data Acquisition and Storage

ThingSpeak's data acquisition process begins with sensor integration, where microcontrollers collect data from environmental sensors such as temperature, humidity, pH, and turbidity sensors. This data is then transmitted to the ThingSpeak server via secure communication channels. Data is stored in structured channels, each capable of holding multiple data streams along with metadata such as timestamps and device identifiers. This structured approach facilitates easy data retrieval, historical trend analysis, and supports advanced analytics using MATLAB.

6.2.3 Remote Monitoring and Visualisation

ThingSpeak excels in remote monitoring and data visualisation. Users can create custom dashboards with real-time graphs, gauges, and charts to monitor dynamic systems effectively. The platform supports automated alerts and notifications based on predefined data thresholds, enabling proactive system management. Integration with MATLAB further enhances its analytical capabilities, allowing for complex data processing, predictive maintenance analysis, and advanced modelling.

6.3 BLYNK

Blynk, launched in 2015, is a versatile IoT platform designed for real-time hardware control, data visualisation, and cloud-based data storage. Known for its intuitive and user-friendly interface, Blynk supports a wide range of microcontrollers, including Arduino, ESP8266, and ESP32. Its core functionality revolves around providing real-time monitoring and control, making it highly suitable for applications where immediate response and system adjustments are critical, such as water treatment processes.

6.3.1 Architecture

Blynk comprises three core components:

- **Blynk Application:** Available on both Android and iOS, the mobile application serves as the user interface, allowing operators to create custom dashboards using drag-and-drop widgets. These dashboards can display real-time data, control devices, and receive notifications, enhancing user interaction and system management.
- **Blynk Server:** Functions as the communication bridge between the hardware devices and the mobile application. It handles data processing, storage, and secure communication. Users have access to the Blynk Cloud Server for global accessibility.
- **Hardware Devices:** The platform supports various microcontrollers like Arduino, ESP32, and Raspberry Pi. These devices are connected to sensors for data acquisition and actuators for controlling system components. Data is transmitted to the Blynk server using secure protocols such as MQTT and HTTP over Wi-Fi, Ethernet, or GSM.

6.3.2 Data Acquisition and Storage

Blynk's data acquisition process involves interfacing microcontrollers with sensors to monitor critical parameters such as pH, temperature, turbidity, and flow rates. The collected data is processed by the microcontrollers and transmitted to the Blynk server. The server logs and stores the data in a structured format, enabling easy access for real-time monitoring and historical analysis. Blynk supports data export features for further analysis using external tools like Excel, Python, or MATLAB.

6.3.3 Remote Monitoring, Control and Visualisation

Blynk excels in providing real-time monitoring capabilities through its customisable dashboards. Users can visualise data trends, set threshold-based alerts, and receive notifications for critical events via push notifications, emails, or SMS. This functionality ensures that operators are promptly informed of any anomalies, facilitating quick responses to maintain optimal system performance. The system also offers the ability to actuate relays via the dashboards and virtual switches. These allow for control over certain parts of the plant, such as the ability to remotely reset a fault, or turn off the plant.

6.4 COMPARATIVE ANALYSIS OF THINGSPEAK AND BLYNK

In the process of selecting an appropriate IoT platform for the Smart Water Operations Platform (SWOP), it was essential to conduct a comparative analysis of ThingSpeak and Blynk, both of which offer distinct features tailored to different IoT applications. This comparison evaluates critical parameters such as core strengths, ease of use, real-time control capabilities, data visualisation options, scalability, advanced analytics, security measures, cost structures, and overall suitability for the SWOP.

The analysis aims to highlight how each platform aligns with the operational requirements of rural water treatment systems, particularly in terms of real-time monitoring, data-driven decision-making, and system scalability. While ThingSpeak excels in advanced data analytics and historical data processing, Blynk offers robust real-time control with an intuitive user interface, making it well-suited for applications that demand immediate response and user-friendly dashboards.

Table 4 summarises the key differences and similarities between ThingSpeak and Blynk across these parameters, providing a clear basis for determining the most suitable platform for integration with the SWOP.

Table 4: Comparison between ThingSpeak & Blynk

Feature	ThingSpeak	Blynk
Core Strength	Advanced analytics via MATLAB	Real-time monitoring and control with a user-friendly dashboard
Ease of Use	Moderate to advanced (MATLAB scripting adds complexity)	Beginner to intermediate (drag-and-drop dashboards)
Real-Time Control	Limited (designed for data logging/analytics, not reactive control loops)	Excellent (supports immediate actuation commands)
Data Visualization	Basic, mostly 2D plots and charts; lacks interactive widgets	Rich widget set (gauges, sliders, charts, notifications)
Scalability	High, but free tier limitations on data rate	High, with potential subscription for advanced or large-scale projects
Advanced Analytics	MATLAB engine enables machine learning, predictive analytics, and custom algorithms	Needs external tools (e.g., Python, R, MATLAB) for complex data analysis
Security	Supports HTTPS, MQTT over TLS, and user-authenticated channels	Provides secure connections via TLS; private server option for sensitive environments
Cost Structure	Free and paid tiers (MATLAB usage may incur additional costs if scaled)	Free and paid tiers (advanced features like white-label, extended data logging cost extra)
Suitability for SWOP	Strong data analytics but limited real-time responsiveness	Optimal real-time monitoring and control for water treatment systems

6.5 JUSTIFICATION FOR SELECTING BLYNK

Blynk was selected for integration with the SWOP due to its superior real-time monitoring and control capabilities, which are critical for maintaining optimal conditions in water treatment processes. Its user-friendly interface allows operators to respond quickly to anomalies, ensuring timely interventions. The platform's flexibility in supporting various hardware devices and communication protocols further enhances its adaptability to different operational environments.

From a cost perspective, Blynk provides a more economical solution, particularly for applications prioritising real-time control and ease of use. While ThingSpeak offers advanced analytics, its reliance on MATLAB increases costs and complexity, making it less suitable for the immediate needs of rural water treatment facilities.

Additionally, Blynk's modular design supports easy scalability, accommodating future expansions with additional sensors or control points. Its support for various communication protocols, including GSM and LoRa, ensures reliable data transmission even in remote areas with limited connectivity. This flexibility, combined with its real-time capabilities and cost-effectiveness, aligns perfectly with the SWOP's objectives of providing an innovative, low-cost, plug-and-play solution for rural water treatment systems.

6.6 SUMMARY

This chapter provided a comprehensive evaluation of ThingSpeak and Blynk as potential IoT platforms for the Smart Water Operations Platform (SWOP), focusing on technical performance, cost-effectiveness, and alignment with system requirements. Blynk emerged as the preferred platform due to its robust real-time monitoring capabilities, ease of integration with various hardware, and scalability to support expanding operational demands.

Considering the operational challenges inherent in rural water treatment-such as the necessity for prompt corrective actions and the limited technical expertise of on-site personnel-Blynk demonstrated clear advantages. Its intuitive interface reduces the learning curve for operators, while its flexibility in supporting diverse communication protocols ensures reliable data transmission even in remote locations. Furthermore, Blynk’s modular design facilitates straightforward system expansions, accommodating additional sensors and control units without significant reconfiguration.

In terms of cost, Blynk offers a more sustainable solution, balancing affordability with comprehensive functionality. Its ability to provide real-time insights and control, coupled with low deployment and maintenance costs, makes it particularly well-suited for resource-constrained environments.

Ultimately, Blynk’s combination of real-time data handling, user-friendly features, and cost efficiency aligns with the core objectives of the SWOP—enhancing the efficiency, reliability, and sustainability of rural water treatment systems. This strategic selection supports the overarching goal of improving water quality management through innovative, low-cost, and easily deployable technology solutions.

CHAPTER 7: PROTOTYPE DEVELOPMENT

7.1 INTRODUCTION

The development of the Smart Water Operations Platform (SWOP) required an iterative prototyping approach, refining system components, software architecture, and integration strategies through multiple design iterations. While this chapter presents the final iteration of the prototype, it is important to acknowledge that significant work was undertaken to develop preceding versions, each serving as a stepping stone toward achieving the functional and operational objectives of the study.

Early prototypes underwent extensive modifications to improve sensor accuracy, communication reliability, data processing capabilities, and cloud integration. Each iteration incorporated insights from testing, enabling continuous refinement of both hardware and software aspects.

This chapter provides a comprehensive overview of the final SWOP prototype, detailing its physical assembly, sensor integration, firmware development, and testing processes. It outlines the step-by-step implementation of both dry testing (validating firmware and sensor calibration) and wet testing (assessing sensor performance under real-world conditions), ensuring that the prototype meets the design specifications. The chapter also presents an analysis of the system’s real-time data acquisition capabilities and cloud-based monitoring, highlighting how the SWOP enables automated water quality assessment in remote areas.

7.2 FINAL PROTOTYPE SYSTEM ARCHITECTURE

The Smart Water Operations Platform (SWOP) was developed with a modular and scalable architecture, ensuring that all system components worked together efficiently to provide real-time monitoring and control of water treatment processes. At the core of the SWOP system was an Arduino Mega, which functioned as the primary microcontroller, managing sensor data acquisition, local data processing, and communication with external systems. An ESP32 Devboard was used for cloud connectivity.

The architecture followed a hierarchical approach, comprising three key layers:

1. Data Acquisition Layer (Sensor Network)
2. Processing and Control Layer (Arduino Mega)
3. Communication and Cloud Integration Layer (ESP32 Devboard & Display Interface)

Figure 13 illustrates the block flow diagram of the final prototype.

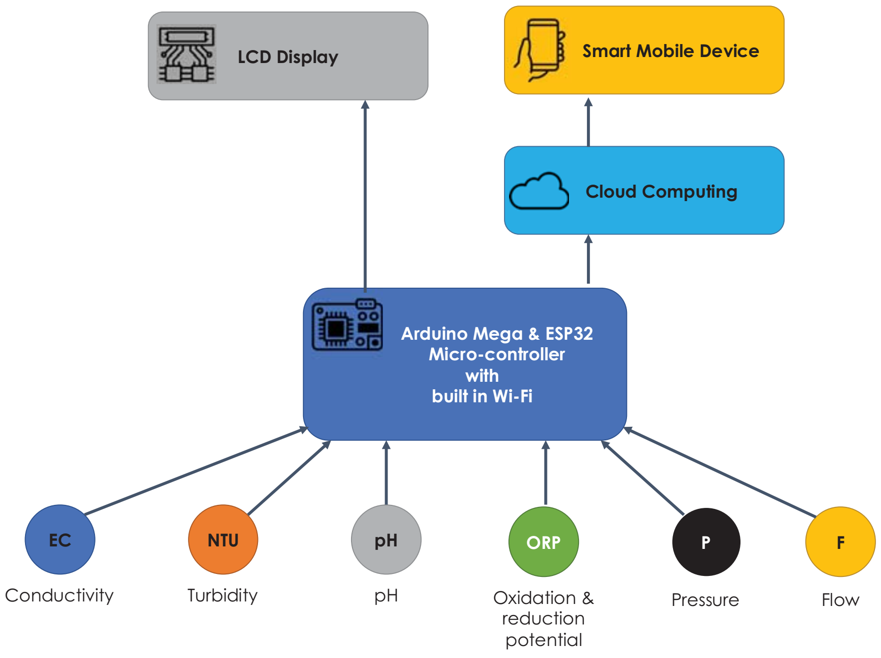


Figure 13: Block flow diagram of Prototype

7.2.1 Data Acquisition Layer (Sensor Network)

The SWOP's sensor network was responsible for collecting critical water quality and system performance parameters. Sensors were strategically deployed to monitor parameters such as pH, electrical conductivity, oxidation-reduction potential (ORP), turbidity, pressure, and flow rates.

Each sensor was interfaced with the Arduino Mega's analog or digital input pins, and their signals were processed to ensure accurate real-time data collection. The sensor readings were periodically sampled, pre-processed, and stored for transmission and further analysis.

Table 5 summarised the list of sensors used in the final prototype.

Table 5: List of sensors used in the final prototype

Sensor	Parameter Measured	Connection Type & Location
pH Sensor	Measured acidity/basicity	Analog (A0)
ORP Sensor	Assessed oxidation-reduction potential	Analog (A3)
Conductivity Sensor	Measured dissolved ion concentration	Analog (A4)
Turbidity Sensor	Determined water clarity	Analog (A5)
Pressure Sensor 1	Monitored filtration system pressure	Analog (A1)
Pressure Sensor 2	Monitored additional pressure changes	Analog (A2)
Flow Sensor	Measured volumetric water flow	Digital (D2)

7.2.2 Processing and Control Layer (Arduino Mega & Display Interface)

The Arduino Mega served as the central processing unit of the SWOP, tasked with handling sensor inputs, executing programmed logic for real-time monitoring, and managing communication with external systems. The Arduino Mega was selected for the following reasons:

- Ample I/O capabilities, which accommodated multiple sensors.
- Reliable 16 MHz processing speed, suitable for real-time environmental monitoring.

- Compatibility with external microcontrollers and communication interfaces.
- Low power consumption, enabling off-grid operation with solar energy.

The core functions of the Arduino Mega in the system were as follows:

a) **Data Sampling & Signal Processing** – Periodically collecting and filtering sensor data to ensure accuracy.
b) **Data Logging & Storage** – Storing real-time data on an SD card module as a backup to prevent data loss in the event of network disruptions.
c) **Local Alarming & Display Updates** – Displaying critical system parameters on the LCD screen and activating local alerts when thresholds were exceeded.
d) **Communication & Command Execution** – Relaying sensor data to the ESP32 Devboard for cloud integration while also receiving and executing remote commands.

To ensure stability and accuracy, the Arduino Mega utilized its built-in 10-bit ADC (analogue-to-digital converter) for most sensor readings. In cases where higher precision was required (such as electrical conductivity measurements), the system allowed for optional integration of a 16-bit ADC module.

The LCD display module provided real-time status updates to on-site operators, allowing them to monitor key parameters without requiring internet connectivity. It was connected via the I2C protocol, ensuring minimal power consumption while displaying real-time sensor readings.

7.2.3 Communication and Cloud Integration Layer (ESP32 Devboard)

The ESP32 Devboard functioned as the communication bridge between the Arduino Mega and the cloud-based monitoring platform.

Key communication roles of the ESP32 were as follows:

- Transmitting sensor data to the cloud via Wi-Fi or cellular connectivity.
- Receiving remote control commands from cloud servers or mobile applications.
- Enabling real-time system monitoring via a web-based dashboard or mobile app.

The ESP32 Devboard was chosen for the following reasons:

- Integrated Wi-Fi and Bluetooth capabilities for seamless data transmission.
- Low-cost and high-efficiency operation, making it suitable for remote deployments.
- Ability to execute lightweight AI-based local analytics for anomaly detection.

The final architecture ensured that even if cloud connectivity was lost, the Arduino Mega and ESP32 continued to function autonomously, allowing for uninterrupted local data logging and decision-making.

7.3 PROTOTYPE ASSEMBLY AND HARDWARE DEVELOPMENT

The hardware architecture of the SWOP was designed with a modular approach, allowing seamless interconnection between components while ensuring ease of maintenance and future scalability. Each subsystem was carefully designed to work in tandem, ensuring accurate data acquisition, efficient processing, and reliable communication with cloud-based platforms.

Each hardware component was selected based on technical compatibility, power efficiency, and cost-effectiveness, ensuring that the system remains affordable and deployable in resource-constrained environments.

The SWOP's physical design was optimised to accommodate real-world operational constraints, including power management, sensor reliability, and ease of maintenance.

The hardware assembly of the SWOP was completed in the following stages:

- Integration of core processing units and microcontrollers.
- Connection and calibration of multiple sensor interfaces.
- Implementation of communication protocols for data transmission.
- Housing of the system in a protective enclosure for field deployment.

Multiple iterative developments with earlier versions served as crucial stepping stones in refining the design, component selection, and system functionalities of this final prototype.

The assembly process of the final SWOP prototype involved sequential integration of hardware components, ensuring that each subsystem was individually validated before full integration.

7.3.1 Integration of core processing units and microcontrollers.

The Arduino Mega served as the central processing unit, coordinating sensor data acquisition, signal filtering, and local decision-making to ensure seamless system operation. It then transmitted data via the level shifter to the ESP32 Devboard V1. These were all housed in a protective enclosure and fixed in place with standoff's, allowing for secure placement and limited movement of the board, which is critical to operational stability, while also allowing for simple swap outs of the components should the need arise for replacement.

The selected enclosure was tall enough to allow secure access to input/output pins for interfacing with sensors and an LCD module, when open and prevent stress on the cables due to sharp bending when closed. It offers IP65 protection rating, which is critical to preventing dust and water ingress during deployment. The plastic housing also allows for the simple drilling out for installation of all the required sensor aviation connections, as well as future LED's, and the cutting out for the LCD screen.

7.3.2 Sensor Interfacing and Data Acquisition

All sensor modules were connected to the microcontroller's analogue and digital input pins, enabling efficient real-time data collection and processing. The robust architecture of the Arduino Mega supported simultaneous communication with multiple sensors while maintaining low latency and stable performance.

For sensors requiring high precision, the 10-bit ADC of the Arduino Mega, with a voltage quantization step of approximately 4.9 mV, was sufficient for most applications. However, provisions had been made for the integration of an external 16-bit ADC module, which allowed for enhanced resolution in scenarios where greater measurement sensitivity was required. These external ADC modules were included with the sensor kits, making the implementation into the system seamless. To maintain high signal fidelity, shielded connections, such as BNC or aviation jacks, were used for sensitive analogue sensors, including pH and ORP probes. Aviation jacks were also selected due to their simplicity in connecting the sensors. All sensors and interface boards are bonded to ground (0v), which is critical to providing stable readings. The use of a voltage reference pin to compare the 5v output to the system 5v supply also enhances accuracy and stability.

The sensor selections were also based on the fact that the entire SWOP would be retrofitted as a standalone unit. The reality is that there is also scope for including additional sensor information, or status information from on-site equipment. If the equipment on site has low voltage outputs or 4-20mA signal outputs, then these can be used with the inclusion of some additional interfacing boards. Current to voltage conversion boards exist for this purpose, as well as voltage to voltage step down boards, allowing the correct logic voltage to be used with the Arduino

7.3.3 Scalability and Future Expansion

The system was designed with expandability in mind, with the Arduino Mega incorporating additional analogue and digital ports to facilitate the integration of advanced analytics modules or supplementary sensors. The data processing framework was optimised to minimise noise and ensure signal integrity under varying environmental conditions, supporting real-time monitoring and adaptive control.

By effectively integrating core processing units with structured sensor interfacing, the system achieved reliable performance, robust data handling, and scalability for future enhancements, such as advanced computational algorithms and extended sensing capabilities.

7.3.4 Connection and calibration of multiple sensor interfaces

The integration of multiple sensors with the Arduino Mega was designed to ensure accurate signal acquisition, minimal interference, and reliable long-term operation. Sensors were wired using shielded connectors to reduce signal noise and improve stability in data transmission. The system incorporated Conductivity, pH and ORP sensors connected via BNC connectors, pressure sensors interfaced through aviation jacks, direct analogue input, a turbidity sensor utilizing an aviation jack to interface board to analogue input, and a flow sensor wired to digital input via aviation jack for continuous monitoring. All connections were also soldered and shrink-wrapped to avoid shorting.

To optimize measurement accuracy, a comprehensive sensor integration strategy was implemented. The electrical conductivity (EC) sensor produced a voltage output directly proportional to the conductivity level within a 0–5V range, processed by the Arduino Mega's 10-bit ADC (4.9 mV resolution). For applications requiring enhanced precision, provisions were made for an optional 16-bit ADC upgrade. The turbidity sensor was equipped with an onboard voltage converter, allowing seamless direct analogue input to the microcontroller. The pH and ORP sensors were calibrated and connected via BNC connectors, ensuring low-noise signal transmission and minimal degradation over extended use.

The SWOP system's physical construction prioritised robust component integration, ease of maintenance, and environmental protection. The Arduino Mega functioned as the central controller, managing data acquisition from the sensors, interfacing with the display unit, and handling communication modules. The prototype was enclosed in a weather-resistant casing to protect critical electronics from dust, humidity, and mechanical stressors, while sensor cables were routed through waterproof conduits to enhance long-term durability.

The LCD display was positioned on the front panel, enabling real-time offline data access for field operators. With all sensor and communication components successfully integrated, the next phase focused on firmware implementation, dry testing, and calibration, key steps to validate accuracy, stability, and system reliability before full-scale deployment. Figures 14 and 15 illustrate the sensor interfacing and electrical connections within the SWOP system.

Figure 14 and Figure 15 illustrate the sensor interfacing and electrical connections. With all sensor and communication components successfully integrated, the next phase involved firmware implementation, dry testing, and calibration—critical steps to ensure accuracy, stability, and reliability before full-scale deployment.

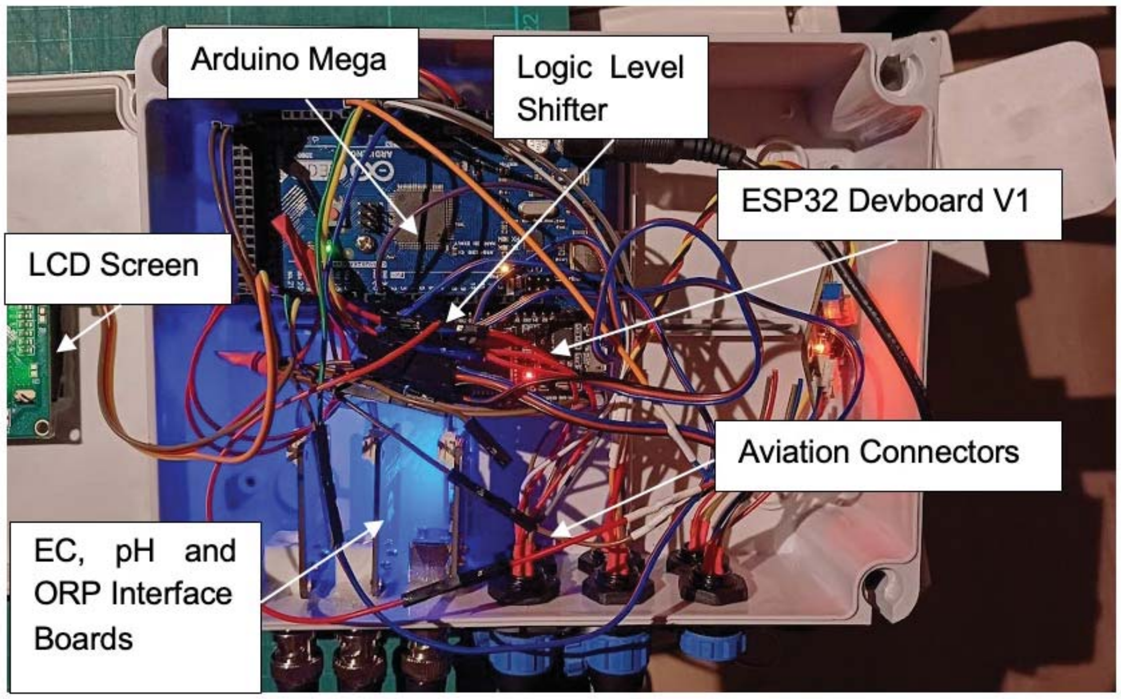


Figure 14:Sensors and LCD connected to Arduino

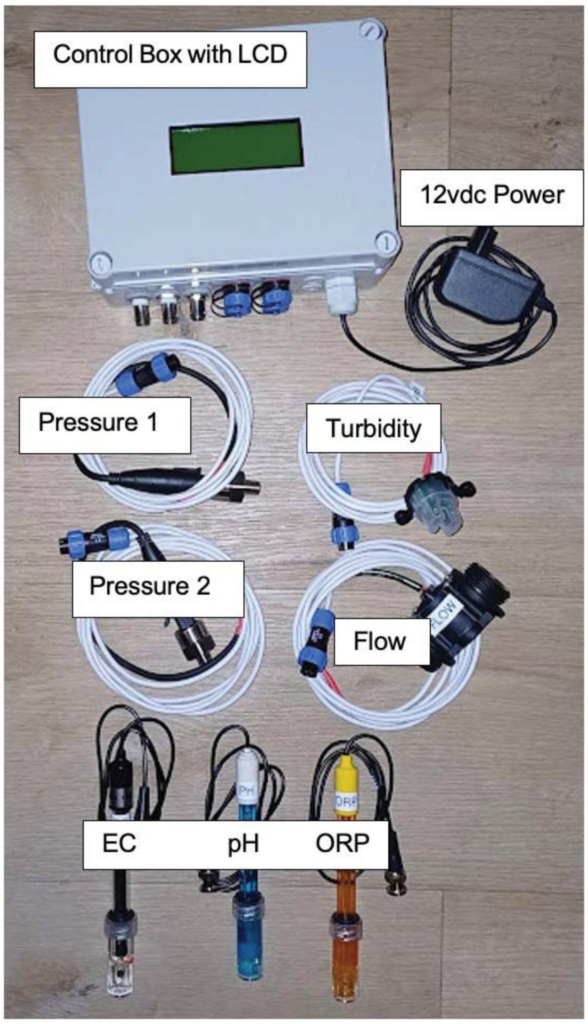


Figure 15: Prototype sensors (disconnected) and enclosure

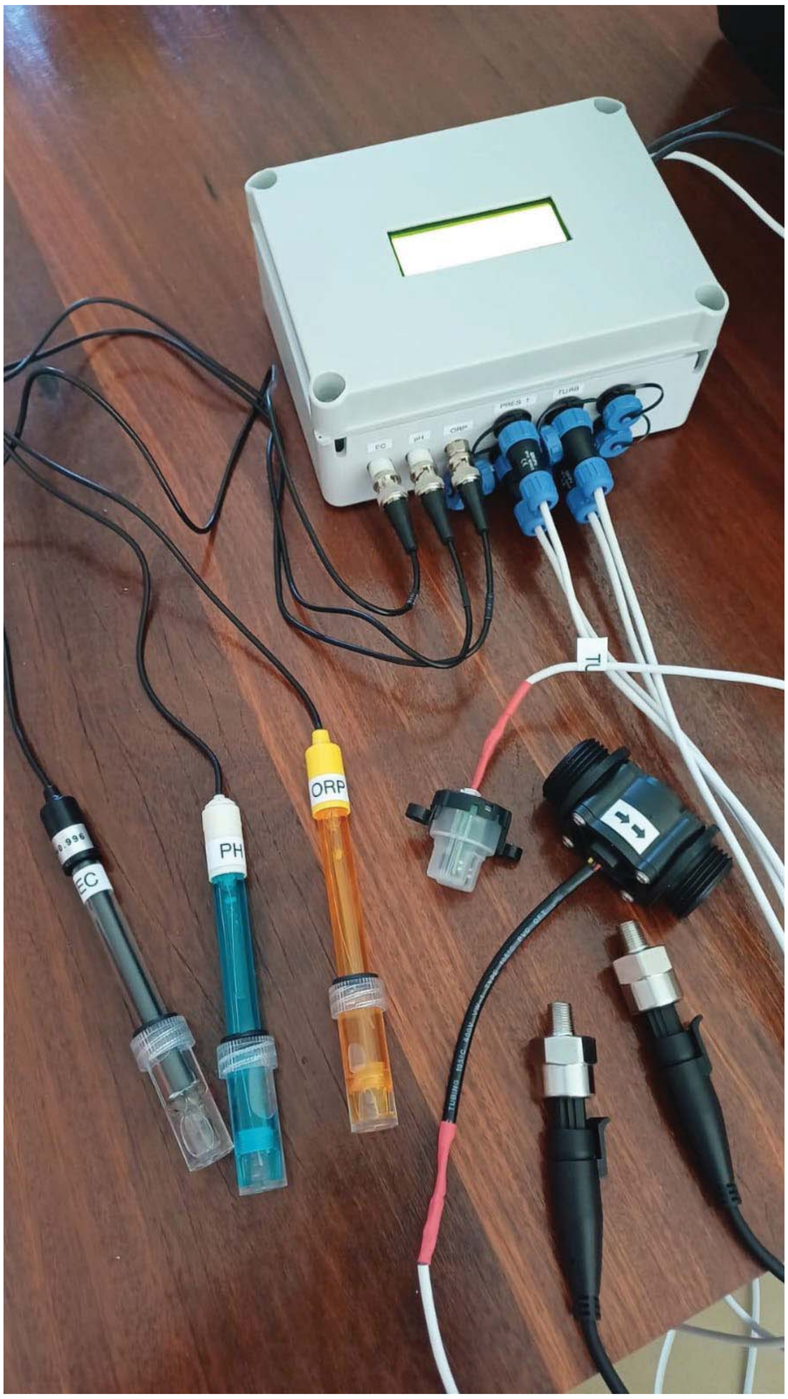


Figure 16: Prototype sensors (connected) and enclosure

7.3.5 Validation of Hardware Integration

After assembling the final prototype, a comprehensive validation process was undertaken to ensure all hardware components functioned correctly:

1. **Power-On Tests** – Verified that the microcontrollers, sensors, and communication modules received a stable power supply.
2. **Sensor Connectivity Tests** – Ensured that each sensor delivered expected readings when exposed to reference solutions.
3. **Data Transmission Tests** – Assessed the ESP32's ability to relay data to cloud servers and receive commands remotely.
4. **Local Display Functionality** – Checked that the LCD screen correctly displayed real-time sensor data for on-site users.

The successful validation of the hardware integration confirmed that the prototype was fully operational and ready for firmware deployment and testing.

7.3.6 Final Assembly and Component Integration

Following individual validations, all components were merged into a unified system:

- Central Controller: Arduino Mega managing sensor data collection, local storage, and LCD display updates.
- Communication Module: ESP32 Devboard for Wi-Fi/Bluetooth-based data upload and remote access.
- Protective Enclosure: Weather-resistant housing to shield electronics from environmental stressors.

Table 6 (an example) outlines the final prototype configuration.

Component	Function	Connection Type & Location
LCD Display	Shows real-time sensor readings	I2C (SCL, SDA)
ESP32 Devboard	Provides cloud-based monitoring & remote access	Serial1
pH Sensor	Measures acidity/basicity	Analog (A0)
ORP Sensor	Monitors oxidation-reduction potential	Analog (A3)
Conductivity Sensor	Assesses dissolved ion content	Analog (A4)
Turbidity Sensor	Evaluates clarity/particulate content	Analog (A5)
Pressure Sensors	Monitors filtration system pressures	Analog (A1, A2)
Flow Sensor	Monitors volumetric flow rate	Digital (D2)
LED Indicator	Displays system status	Digital (D13)

7.3.7 Electrical Design and Sensor Integration

The sensor integration strategy was developed to ensure efficient signal acquisition and minimal interference.

- The electrical conductivity (EC) sensor outputs a voltage directly proportional to the conductivity level, operating within a 0–5V range. The Arduino Mega's 10-bit ADC resolution (4.9 mV per unit) was used for initial processing, with an option for a 16-bit ADC upgrade if higher precision was required.
- The turbidity sensor was integrated with an onboard voltage converter, allowing seamless direct analogue input to the microcontroller.
- The pH and ORP sensors were calibrated and interfaced using industry-standard BNC connectors, ensuring minimal signal degradation during operation.

7.3.8 Completion of Hardware Assembly

With all sensor and communication components successfully integrated, the next phase of the development process focused on firmware implementation, dry testing, and calibration. These steps were critical in ensuring the accuracy, stability, and reliability of the sensor readings before full-scale deployment.

7.4 DRY TESTING

The dry testing phase comprised:

a) Sensor firmware validation
b) Sensor calibration

7.4.1 Sensor Firmware Validation

Validation of the sensor firmware and functionality was carried out during the dry testing phase. Establishing the sensors' ability to reliably measure and store both calibration data and alarm conditions in non-volatile memory in the event of a loss of power is crucial to ensuring the integrity of the collected data and the long-term performance of the sensor. Preliminary testing was completed in this phase to demonstrate recording of data from the sensor and further refinement testing is in progress in preparation for the subsequent wet testing campaign.

7.4.2 Sensor Calibration

Once the firmware validation was completed, the sensors underwent basic calibration procedures to establish measurement capabilities. However, further detailed calibration tests will be performed during the wet testing phase to acquire data, including offsets and coefficients specific to each sensor. Using these results, calibration curves will be generated based on known reference values, allowing accurate measurements to be obtained from the sensors. The performance of each sensor is to be evaluated by comparing the measured values with the expected readings.

7.5 PROTOTYPE TEST CODE

The following section of code is loaded into the Arduino Mega. All source code was available from the sensor supplier's website or as examples within the Arduino IDE example library. The code has been modified to specifically align with the hardware setup being used in Prototype 2. This code combines a multitude of sensor codes and logic into a single complete sketch.

The code has been commented to allow for ease of reference and to assist with simple troubleshooting by less experienced personnel.

```
#include <LiquidCrystal_I2C.h>
#include <DFRobot_PH.h>
#include <DFRobot_EC.h>
#include <EEPROM.h>

//================== PIN DEFINITIONS ===================
#define FLOW_SENSOR_PIN        2     // digital input for water flow sensor (attachInterrupt will be used)
#define PH_SENSOR_PIN          A0    // analog pin for pH sensor
#define PRESSURE_SENSOR1_PIN   A1    // analog pin for pressure sensor 1
```

```
#define PRESSURE_SENSOR2_PIN    A2     // analog pin for pressure sensor 2
#define ORP_SENSOR_PIN         A3     // analog pin for ORP sensor
#define CONDUCTIVITY_SENSOR_PIN A4     // analog pin for EC sensor
#define TURBIDITY_SENSOR_PIN    A5     // analog pin for turbidity sensor

#define LED_PIN                13     // an LED (used in the ORP demo)

//================== ORP SENSOR VARIABLES ===================
#define SYSTEM_VOLTAGE         5.00
#define ORP_ZERO_OFFSET         7        // sensor-specific zero drift offset
#define ORP_ARRAY_LENGTH       40       // number of samples for averaging

int orpArray[ORP_ARRAY_LENGTH];        // array to store samples
int orpArrayIndex = 0;                 // current index in the sample array
double orpValue = 0;                   // computed ORP value

//================== PH & EC SENSOR OBJECTS ===================
DFRobot_PH ph;
DFRobot_EC ec;
float voltagePH, pH;
float voltageEC, conductivity;
float temperature = 25.0;  // default temperature in °C (replace with sensor reading if available)

//================== TURBIDITY SENSOR VARIABLE ===================
float turbidity;  // for now, we simply use the voltage as the “turbidity” reading

//================== FLOW METER VARIABLES ===================
volatile unsigned long pulseCount = 0;  // counts pulses from the flow sensor interrupt
float flowRate = 0;                    // flow rate in L/min

//================== PRESSURE SENSOR VARIABLES ===================
const float PressureOffSet = 0.483;  // sensor calibration offset (obtained during calibration)
float pressure1 = 0;  // in Bar
float pressure2 = 0;  // in Bar

//================== LCD INITIALIZATION ===================
// Adjust the I2C address (0x27) and dimensions (20 columns x 4 rows) as needed.
LiquidCrystal_I2C lcd(0x27, 20, 4);

//================== TIMER VARIABLES ===================
// These variables hold the last time (in ms) each sensor section was updated.
unsigned long previousORPSampleMillis   = 0;
unsigned long previousORPPrintMillis    = 0;
unsigned long previousPH_EC_Millis      = 0;
unsigned long previousTurbidityMillis   = 0;
unsigned long previousPressureMillis    = 0;
unsigned long previousFlowMillis        = 0;
unsigned long lcdUpdateMillis           = 0;
unsigned long serial1UpdateMillis       = 0;

//================== SERIAL COMMAND BUFFER ===================
char cmdBuffer[20];   // buffer to hold an incoming command string
int cmdIndex = 0;     // index into the command buffer

//----------------- Helper: Array Average (for ORP) -----------------
// This function calculates an average that excludes the minimum and maximum
// values if the number of samples is 5 or more.
```

```
double averageArray(int *arr, int number) {
  int i, min, max;
  long total = 0;
  double avg = 0;

  if(number <= 0) return 0;

  if(number < 5) {
    for (i = 0; i < number; i++) {
      total += arr[i];
    }
    avg = total / number;
    return avg;
  } else {
    // Initialize min and max with first two values
    if (arr[0] < arr[1]) {
      min = arr[0];
      max = arr[1];
    } else {
      min = arr[1];
      max = arr[0];
    }
    // Process remaining samples
    for (i = 2; i < number; i++) {
      if (arr[i] < min) {
        total += min;
        min = arr[i];
      } else if (arr[i] > max) {
        total += max;
        max = arr[i];
      } else {
        total += arr[i];
      }
    }
    avg = (double) total / (number - 2);
    return avg;
  }
}

//------------------ Flow Meter Interrupt Service Routine ------------------
// Every pulse (rising edge) increments the pulse count.
void pulse() {
  pulseCount++;
}

//------------------ Helper: Read Serial Command ------------------
// This function reads from a given HardwareSerial port until a newline ('\n')
// is encountered. It stores the command in "result" and returns true when complete.
bool readSerialCommand(HardwareSerial &serialPort, char *result, int maxLength) {
  while(serialPort.available() > 0) {
    char inChar = serialPort.read();
    if (inChar == '\n') {
      result[cmdIndex] = '\0';
      cmdIndex = 0;
      return true;
    }
    if (inChar != '\r') {
      if (cmdIndex < maxLength - 1) {
        result[cmdIndex++] = inChar;
      }
    }
  }
  return false;
}

void setup() {
  // Initialize Serial ports
  Serial.begin(115200);   // for debugging and calibration commands
  Serial1.begin(9600);  // for communication with the ESP32

  // Initialize the LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Initializing...");
  delay(2000);
  lcd.clear();

  // Initialize the pH and EC sensor objects
  ph.begin();
  ec.begin();

  // Set the LED pin mode
  pinMode(LED_PIN, OUTPUT);

  // Attach the interrupt for the flow sensor.
  // (Using digitalPinToInterrupt ensures portability.)
  attachInterrupt(digitalPinToInterrupt(FLOW_SENSOR_PIN), pulse, RISING);

  // Initialize timer variables with current time
  unsigned long currentMillis = millis();
  previousORPSampleMillis = currentMillis;
  previousORPPrintMillis  = currentMillis;
  previousPH_EC_Millis    = currentMillis;
  previousTurbidityMillis = currentMillis;
  previousPressureMillis  = currentMillis;
  previousFlowMillis      = currentMillis;
  lcdUpdateMillis         = currentMillis;
  serial1UpdateMillis     = currentMillis;

  lcd.print("System Ready");
  delay(2000);
  lcd.clear();
}

void loop() {
  unsigned long currentMillis = millis();

  // ------------------- ORP Sensor Section -------------------
  // Sample the ORP sensor every 20ms
  if (currentMillis - previousORPSampleMillis >= 20) {
    previousORPSampleMillis = currentMillis;
    orpArray[orpArrayIndex++] = analogRead(ORP_SENSOR_PIN);
    if (orpArrayIndex >= ORP_ARRAY_LENGTH) {
```

```
    orpArrayIndex = 0;
  }
}
// Every 800ms, compute the ORP value using the averaged samples.
if (currentMillis - previousORPPrintMillis >= 800) {
  previousORPPrintMillis = currentMillis;
  // Calculation as provided in the sensor code:
  orpValue = ((30 * SYSTEM_VOLTAGE * 1000) -
            (75 * averageArray(orpArray, ORP_ARRAY_LENGTH) * SYSTEM_VOLTAGE * 1000 / 1024))
            / 75 - ORP_ZERO_OFFSET;
  // (Optional) Print to Serial for debugging:
  Serial.print("ORP: ");
  Serial.print((int)orpValue);
  Serial.println(" mV");
  // Optionally toggle an LED to indicate a new reading:
  digitalWrite(LED_PIN, !digitalRead(LED_PIN));
}

// ------------------- pH and EC Sensor Section -------------------
// Read pH and EC values every 1000ms (1 second)
if (currentMillis - previousPH_EC_Millis >= 1000) {
  previousPH_EC_Millis = currentMillis;

  // pH sensor reading
  int rawPH = analogRead(PH_SENSOR_PIN);
  voltagePH = rawPH / 1024.0 * 5000;   // convert reading to mV
  pH = ph.readPH(voltagePH, temperature);

  // EC sensor reading
  int rawEC = analogRead(CONDUCTIVITY_SENSOR_PIN);
  voltageEC = rawEC / 1024.0 * 5000;   // convert reading to mV
  conductivity = ec.readEC(voltageEC, temperature);

  Serial.print("pH: ");
  Serial.print(pH, 2);
  Serial.print(" | EC: ");
  Serial.print(conductivity*1000, 2);
  Serial.println(" us/cm");
}

// ------------------- Turbidity Sensor Section -------------------
// Read the turbidity sensor every 500ms
if (currentMillis - previousTurbidityMillis >= 500) {
  previousTurbidityMillis = currentMillis;
  int rawTurbidity = analogRead(TURBIDITY_SENSOR_PIN);
  float voltageTurbidity = rawTurbidity * (5.0 / 1024.0);
  // For this example we use the voltage as the "turbidity" value.
  turbidity = voltageTurbidity;
  Serial.print("Turbidity: ");
  Serial.println(turbidity, 2);
}

// ------------------- Pressure Sensors Section -------------------
// Read both pressure sensors every 500ms.
if (currentMillis - previousPressureMillis >= 500) {
  previousPressureMillis = currentMillis;
  // Pressure Sensor 1 (using A1)
  float voltageP1 = analogRead(PRESSURE_SENSOR1_PIN) * (5.0 / 1024.0);
  float pressureKPa1 = (voltageP1 - PressureOffSet) * 250;  // in KPa
  pressure1 = pressureKPa1 / 100.0;  // convert to Bar (1 Bar ~ 100 KPa)

  // Pressure Sensor 2 (using A2)
  float voltageP2 = analogRead(PRESSURE_SENSOR2_PIN) * (5.0 / 1024.0);
  float pressureKPa2 = (voltageP2 - PressureOffSet) * 250;
  pressure2 = pressureKPa2 / 100.0;

  Serial.print("Pressure1: ");
  Serial.print(pressure1, 2);
  Serial.print(" Bar, Pressure2: ");
  Serial.print(pressure2, 2);
  Serial.println(" Bar");
}

// ------------------- Flow Sensor Section -------------------
// Every 1000ms, compute the flow rate (L/min) based on the pulse count.
if (currentMillis - previousFlowMillis >= 1000) {
  previousFlowMillis = currentMillis;
  // Each pulse corresponds to 1/288 L; so for pulses counted over 1 sec,
  // the flow rate in L/min is: (pulseCount * 60) / 288.
  flowRate = (pulseCount * 60.0) / 288.0;
  pulseCount = 0;  // reset count for the next interval

  Serial.print("Flow Rate: ");
  Serial.print(flowRate, 2);
  Serial.println(" L/min");
}

// ------------------- Calibration Commands Processing -------------------
// Check for commands from either Serial (USB) or Serial1 (from the ESP32).
// The pH and EC libraries use command strings (e.g., "enterph", "calph", "exitph",
// "enterec", "calec", "exitec") to perform calibration.
if ( readSerialCommand(Serial, cmdBuffer, sizeof(cmdBuffer)) ||
     readSerialCommand(Serial1, cmdBuffer, sizeof(cmdBuffer)) ) {
  // Convert the command to uppercase for case-insensitive matching.
  for (int i = 0; cmdBuffer[i] != '\0'; i++) {
    cmdBuffer[i] = toupper(cmdBuffer[i]);
  }
  // If the command contains "PH", pass it to the pH calibration routine.
  if (strstr(cmdBuffer, "PH") != NULL) {
    ph.calibration(voltagePH, temperature, cmdBuffer);
  }
  // If the command contains "EC", pass it to the EC calibration routine.
  if (strstr(cmdBuffer, "EC") != NULL) {
    ec.calibration(voltageEC, temperature, cmdBuffer);
  }
}
  // ------------------- LCD Display Update -------------------
// Update the LCD every 1000ms with all sensor readings.
if (currentMillis - lcdUpdateMillis >= 1000) {
  lcdUpdateMillis = currentMillis;
  lcd.clear();

  // Line 0: Flow rate and pH
  lcd.setCursor(0, 0);
  lcd.print("Flow ");
  lcd.print(flowRate, 1);
  lcd.print("LPM pH ");
  lcd.print(pH, 2);

  // Line 1: Pressure sensors
  lcd.setCursor(0, 1);
  lcd.print("P1 ");
  lcd.print(pressure1, 1);
  lcd.print("Bar P2 ");
  lcd.print(pressure2, 1);
  lcd.print("Bar");

  // Line 2: ORP and Turbidity
  lcd.setCursor(0, 2);
  lcd.print("ORP ");
  lcd.print(orpValue, 1);
  lcd.print("mV NTU ");
  lcd.print(turbidity, 1);

  // Line 3: EC value
  lcd.setCursor(0, 3);
  lcd.print("EC ");
  lcd.print(conductivity*1000);
  lcd.print("uS/cm");
 }
  // ------------------- Serial1 (ESP32) Output -------------------
 // Every 1000ms, send a comma-separated string of sensor values to Serial1.
 if (currentMillis - serial1UpdateMillis >= 1000) {
  serial1UpdateMillis = currentMillis;
  Serial1.print(flowRate, 1);
  Serial1.print(",");
  Serial1.print(pH, 2);
  Serial1.print(",");
  Serial1.print(pressure1, 1);
  Serial1.print(",");
  Serial1.print(pressure2, 1);
  Serial1.print(",");
  Serial1.print(orpValue, 1);
  Serial1.print(",");
  Serial1.print(turbidity, 1);
  Serial1.print(",");
  Serial1.println((int)(conductivity * 1000));
 }

  // (The loop then repeats; each sensor section runs on its own schedule.)
```

The DFRobot_PH.h and DFRobot_PH.cpp files were modified to allow for 3 point calibration instead of the current 2 point. The system will autodetect pH 4, pH 7 and pH 10 calibration solutions.

Below are the updated DFRobot_PH.h and DFRobot_PH.cpp codes respectively.

```
* @file DFRobot_PH.h
 * @brief Arduino library for Gravity: Analog pH Sensor / Meter Kit V2, SKU: SEN0161-V2
 *
 * @details
 * Updated for three-point calibration using standard buffer solutions at pH 4.0, 7.0, and 10.0.
 * The calibration commands remain similar (e.g., "enterph", "calph", "exitph") but the "calph"
 * command will now automatically recognize the three calibration points.
 *
 * @copyright
 * Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
 * @license
 * The MIT License (MIT)
 * @author
 * [Jiawei Zhang](jiawei.zhang@dfrobot.com)
 * @version
 * V1.1  (Modified for 3-point calibration by Kevin Lacey – info@redantprojects.co.za)
 * @date
 * 2018-11-06 (modified 2025-01-31 for 3-point calibration)
 * @url
 * https://github.com/DFRobot/DFRobot_PH
 */

#ifndef _DFROBOT_PH_H_
#define _DFROBOT_PH_H_

#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

#define ReceivedBufferLength 10  // length of the Serial CMD buffer

class DFRobot_PH
{
public:
  DFRobot_PH();
  ~DFRobot_PH();

  /**
   * @fn calibration
   * @brief Calibrate the sensor using three standard buffer solutions.
   *
   * @param voltage     : Voltage value read from the sensor
   * @param temperature : Ambient temperature
   * @param cmd         : Calibration command received via Serial
   *                      Commands:
   *                        "enterph" -> enter pH calibration mode
   *                        "calph"   -> calibrate with the standard buffer solution;
   *                                     now supports three points (pH 4.0, 7.0, and 10.0)
   *                        "exitph"  -> compute and save the calibration parameters and exit calibration mode
   */
  void calibration(float voltage, float temperature, char* cmd);  // calibration via Serial CMD

  /**
   * @fn calibration
   * @brief Calibrate the sensor without using Serial commands.
   *
   * @param voltage     : Voltage value read from the sensor
   * @param temperature : Ambient temperature
   */
  void calibration(float voltage, float temperature);

  /**
   * @fn readPH
   * @brief Convert voltage to pH with temperature compensation using a quadratic fit.
   *
   * @param voltage     : Voltage value read from the sensor
   * @param temperature : Ambient temperature
   * @return The pH value calculated from the quadratic calibration curve
   */
  float readPH(float voltage, float temperature);

  /**
   * @fn begin
   * @brief Initialize the Analog pH Sensor.
   */
  void begin();

private:
    float  _phValue;          // Latest computed pH value
    float  _acidVoltage;       // Calibration voltage for acidic buffer (pH 4.0)
    float  _neutralVoltage;    // Calibration voltage for neutral buffer (pH 7.0)
    float  _alkalineVoltage;   // Calibration voltage for alkaline buffer (pH 10.0)
    float  _voltage;           // Latest sensor voltage reading
    float  _temperature;       // Current temperature

    // Calibration coefficients for the quadratic fit: pH = a*V^2 + b*V + c
    float  _a;
    float  _b;
    float  _c;

    char   _cmdReceivedBuffer[ReceivedBufferLength];  // Buffer to store the Serial command
    byte   _cmdReceivedBufferIndex;

private:
    boolean cmdSerialDataAvailable();
    byte    cmdParse(const char* cmd);
    byte    cmdParse();

    /**
     * @fn phCalibration
     * @brief Internal calibration process that writes key parameters to EEPROM.
     *        (You will need to update this function in the .cpp file so that it stores three calibration points.)
     *
     * @param mode : Calibration mode or step.
     */
    void phCalibration(byte mode);

    /**
     * @fn calculateCoefficients
     * @brief Compute the quadratic calibration coefficients (_a, _b, _c) based on the three calibration voltages.
     *
     * Uses the known standard pH values:
     *    pH 4.0 for _acidVoltage, pH 7.0 for _neutralVoltage, and pH 10.0 for _alkalineVoltage.
     *
     * The coefficients are calculated so that:
     *    4.0 = _a * (_acidVoltage)^2 + _b * (_acidVoltage) + _c
     *    7.0 = _a * (_neutralVoltage)^2 + _b * (_neutralVoltage) + _c
     *    10.0 = _a * (_alkalineVoltage)^2 + _b * (_alkalineVoltage) + _c
     */
    void calculateCoefficients();
};

#endif


/*!
 * @file DFRobot_PH.cpp
 * @brief Arduino library for Gravity: Analog pH Sensor / Meter Kit V2, SKU: SEN0161-V2
 *
 * Updated for three-point calibration (pH 4.0, 7.0, and 10.0).
 *
 * @copyright   Copyright (c) 2010 DFRobot Co.Ltd
 * @license     The MIT License (MIT)
 * @author
 * [Jiawei Zhang](jiawei.zhang@dfrobot.com) (original), modified 2025-01-31 for 3-point calibration by Kevin Lacey -
info@redantprojects.co.za
 * @version     V1.1
 * @url         https://github.com/DFRobot/DFRobot_PH
 */

#if ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif

#include "DFRobot_PH.h"
#include <EEPROM.h>

#define EEPROM_write(address, p) { int i = 0; byte *pp = (byte*)&(p); for(; i < sizeof(p); i++) EEPROM.write(address+i,
pp[i]); }
#define EEPROM_read(address, p) { int i = 0; byte *pp = (byte*)&(p); for(; i < sizeof(p); i++) pp[i] =
EEPROM.read(address+i); }

#define PHVALUEADDR 0x00    // Start address for pH calibration parameters stored in EEPROM

// ------------------------------------------------------------------

DFRobot_PH::DFRobot_PH()
{
    this->_temperature     = 25.0;
    this->_phValue         = 7.0;
    this->_acidVoltage     = 2032.44;   // default voltage for pH 4.0 at 25°C
    this->_neutralVoltage  = 1500.0;    // default voltage for pH 7.0 at 25°C
    this->_alkalineVoltage = 1000.0;    // default voltage for pH 10.0 (new)
    this->_voltage         = 1500.0;
}

DFRobot_PH::~DFRobot_PH()
{
}

void DFRobot_PH::begin()
```

```
{
    // Load stored calibration values from EEPROM for pH 7.0, 4.0, and 10.0
    EEPROM_read(PHVALUEADDR, this->_neutralVoltage);
    Serial.print("_neutralVoltage:");
    Serial.println(this->_neutralVoltage);
    if(EEPROM.read(PHVALUEADDR)==0xFF && EEPROM.read(PHVALUEADDR+1)==0xFF &&
       EEPROM.read(PHVALUEADDR+2)==0xFF && EEPROM.read(PHVALUEADDR+3)==0xFF){
        this->_neutralVoltage = 1500.0;  // New EEPROM, use typical value
        EEPROM_write(PHVALUEADDR, this->_neutralVoltage);
    }
    EEPROM_read(PHVALUEADDR+4, this->_acidVoltage);
    Serial.print("_acidVoltage:");
    Serial.println(this->_acidVoltage);
    if(EEPROM.read(PHVALUEADDR+4)==0xFF && EEPROM.read(PHVALUEADDR+5)==0xFF &&
       EEPROM.read(PHVALUEADDR+6)==0xFF && EEPROM.read(PHVALUEADDR+7)==0xFF){
        this->_acidVoltage = 2032.44;  // New EEPROM, use typical value
        EEPROM_write(PHVALUEADDR+4, this->_acidVoltage);
    }
    EEPROM_read(PHVALUEADDR+8, this->_alkalineVoltage);
    Serial.print("_alkalineVoltage:");
    Serial.println(this->_alkalineVoltage);
    if(EEPROM.read(PHVALUEADDR+8)==0xFF && EEPROM.read(PHVALUEADDR+9)==0xFF &&
       EEPROM.read(PHVALUEADDR+10)==0xFF && EEPROM.read(PHVALUEADDR+11)==0xFF){
        this->_alkalineVoltage = 1000.0;  // New EEPROM, use typical value for pH 10.0
        EEPROM_write(PHVALUEADDR+8, this->_alkalineVoltage);
    }
}

float DFRobot_PH::readPH(float voltage, float temperature)
{
    // Compute quadratic calibration coefficients based on the three calibration points.
    calculateCoefficients();
    // Convert the measured voltage to pH using the quadratic equation:
    // pH = a*voltage^2 + b*voltage + c
    this->_phValue = this->_a * voltage * voltage + this->_b * voltage + this->_c;
    return this->_phValue;
}

void DFRobot_PH::calibration(float voltage, float temperature, char* cmd)
{
    this->_voltage = voltage;
    this->_temperature = temperature;
    strupr(cmd);
    phCalibration(cmdParse(cmd));  // Process calibration command from Serial CMD
}

void DFRobot_PH::calibration(float voltage, float temperature)
{
    this->_voltage = voltage;
    this->_temperature = temperature;
    if(cmdSerialDataAvailable() > 0){
        phCalibration(cmdParse());  // Process calibration command from Serial
    }
}

boolean DFRobot_PH::cmdSerialDataAvailable()
{
    char cmdReceivedChar;
    static unsigned long cmdReceivedTimeOut = millis();
    while(Serial.available() > 0){
        if(millis() - cmdReceivedTimeOut > 500U){
            this->_cmdReceivedBufferIndex = 0;
            memset(this->_cmdReceivedBuffer, 0, (ReceivedBufferLength));
        }
        cmdReceivedTimeOut = millis();
        cmdReceivedChar = Serial.read();
        if (cmdReceivedChar == '\n' || this->_cmdReceivedBufferIndex == ReceivedBufferLength-1){
            this->_cmdReceivedBufferIndex = 0;
            strupr(this->_cmdReceivedBuffer);
            return true;
        } else {
            this->_cmdReceivedBuffer[this->_cmdReceivedBufferIndex] = cmdReceivedChar;
            this->_cmdReceivedBufferIndex++;
        }
    }
    return false;
}

byte DFRobot_PH::cmdParse(const char* cmd)
{
    byte modeIndex = 0;
    if(strstr(cmd, "ENTERPH") != NULL){
        modeIndex = 1;
    } else if(strstr(cmd, "EXITPH") != NULL){
        modeIndex = 3;
    } else if(strstr(cmd, "CALPH") != NULL){
        modeIndex = 2;
    }
    return modeIndex;
}

byte DFRobot_PH::cmdParse()
{
    byte modeIndex = 0;
    if(strstr(this->_cmdReceivedBuffer, "ENTERPH") != NULL){
        modeIndex = 1;
    } else if(strstr(this->_cmdReceivedBuffer, "EXITPH") != NULL){
        modeIndex = 3;
    } else if(strstr(this->_cmdReceivedBuffer, "CALPH") != NULL){
        modeIndex = 2;
    }
    return modeIndex;
}

void DFRobot_PH::phCalibration(byte mode)
{
    // Use static flags to track calibration state.
    static boolean phCalibrationFinish  = 0;
    static boolean enterCalibrationFlag = 0;

    switch(mode){
        case 0:
            if(enterCalibrationFlag){
                Serial.println(F(">>>Command Error<<<"));
            }
            break;

        case 1:
            // Enter calibration mode.
            enterCalibrationFlag = 1;
            phCalibrationFinish  = 0;
            Serial.println();
            Serial.println(F(">>>Enter PH Calibration Mode<<<"));
            Serial.println(F(">>>Please put the probe into the standard buffer solution (pH 4.0, 7.0, or 10.0)<<<"));
            Serial.println();
            break;

        case 2:
            if(enterCalibrationFlag){
                // Check which buffer solution the voltage falls into.
                // pH 10 (basic): assume voltage between 300 and 1100.
                if((this->_voltage > 300) && (this->_voltage < 1100)){
                  Serial.println();
                  Serial.print(F(">>>Buffer Solution:10.0"));
                  this->_alkalineVoltage = this->_voltage;
                  Serial.println(F(", Send EXITPH to Save and Exit<<<"));
                  Serial.println();
                  phCalibrationFinish = 1;
                }
                // pH 7 (neutral): voltage between 1322 and 1678.
                else if((this->_voltage > 1322) && (this->_voltage < 1678)){
                  Serial.println();
                  Serial.print(F(">>>Buffer Solution:7.0"));
                  this->_neutralVoltage = this->_voltage;
                  Serial.println(F(", Send EXITPH to Save and Exit<<<"));
                  Serial.println();
                  phCalibrationFinish = 1;
                }
                // pH 4 (acidic): voltage between 1854 and 2210.
                else if((this->_voltage > 1854) && (this->_voltage < 2210)){
                  Serial.println();
                  Serial.print(F(">>>Buffer Solution:4.0"));
                  this->_acidVoltage = this->_voltage;
                  Serial.println(F(", Send EXITPH to Save and Exit<<<"));
                  Serial.println();
                  phCalibrationFinish = 1;
                }
                else{
                  Serial.println();
                  Serial.print(F(">>>Buffer Solution Error, Try Again<<<"));
                  Serial.println();
                  phCalibrationFinish = 0;
                }
            }
            break;

        case 3:
            if(enterCalibrationFlag){
                Serial.println();
                if(phCalibrationFinish){
                  // Save all three calibration voltages to EEPROM.
```

```
                EEPROM_write(PHVALUEADDR, this->_neutralVoltage);
                EEPROM_write(PHVALUEADDR+4, this->_acidVoltage);
                EEPROM_write(PHVALUEADDR+8, this->_alkalineVoltage);
                Serial.print(F(">>>Calibration Successful"));
                Serial.println(F(", Exit PH Calibration Mode<<<"));
            } else {
                Serial.print(F(">>>Calibration Failed"));
                Serial.println(F(", Exit PH Calibration Mode<<<"));
            }
            Serial.println();
            phCalibrationFinish  = 0;
            enterCalibrationFlag = 0;
        }
        break;
    }
}

/*!
 * @brief Compute the quadratic calibration coefficients (_a, _b, _c) based on three calibration points.
 *
 * The three calibration points are:
 *   (_acidVoltage, 4.0), (_neutralVoltage, 7.0), and (_alkalineVoltage, 10.0).
 *
 * We solve the following equations:
 *   a*(x1)^2 + b*(x1) + c = 4.0
 *   a*(x2)^2 + b*(x2) + c = 7.0
 *   a*(x3)^2 + b*(x3) + c = 10.0
 *
 * This implementation uses elimination.
 */
void DFRobot_PH::calculateCoefficients()
{
    float x1 = this->_acidVoltage;     // pH 4.0 calibration voltage
    float x2 = this->_neutralVoltage;  // pH 7.0 calibration voltage
    float x3 = this->_alkalineVoltage; // pH 10.0 calibration voltage
    float y1 = 4.0;
    float y2 = 7.0;
    float y3 = 10.0;

    // Compute differences for elimination.
    float a1 = (x2 * x2) - (x1 * x1);
    float b1 = x2 - x1;
    float r1 = y2 - y1;

    float a2 = (x3 * x3) - (x2 * x2);
    float b2 = x3 - x2;
    float r2 = y3 - y2;

    float denominator = a1 * b2 - a2 * b1;
    if (denominator == 0) {
        // Fallback to linear calibration if degenerate.
        this->_a = 0;
        this->_b = (y2 - y1) / (x2 - x1);
        this->_c = y1 - this->_b * x1;
    } else {
        this->_a = (r1 * b2 - r2 * b1) / denominator;
        this->_b = (r1 - this->_a * a1) / b1;
        this->_c = y1 - this->_a * x1 * x1 - this->_b * x1;
    }
}
```

The following code runs on the ESP32 and communicates with the Blynk Cloud. It provides the interface between the Arduino Mega and the Cloud, and can also receive user commands from the Blynk Cloud for sending calibration requests to the Arduino Mega. This eliminates the need to use a USB cable to calibrate the ORP, pH and EC meters.

In order to access the Blynk network, a profile needs to be created and the device added to the list of devices available to monitor. A secure Template ID and Authentication Token are created that can only be used by this device, ensuring the security of the data transmission. Detailed lists of instructions are available on the Blynk website to assist with additions or modifications that would need to be made if adding or changing sensors.

The WiFi Credentials need to be programmed to the ESP32 on initial installation in order for the system to run properly. For proof of concept in this Prototype 2, the WiFi option has been used with a home network, and as such, the credentials have been redacted. The simplest and most cost-effective way of deploying a network in a remote area is to use a mini Wifi, or MiFi router and create the wireless network within the existing cellular network.

```
#define BLYNK_TEMPLATE_ID "TMPL2ZU15r4kv"
#define BLYNK_TEMPLATE_NAME "WaterPlant"
#define BLYNK_AUTH_TOKEN "eio82a-YhKv82KK934l-7ddDDejzL5s7"

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

// WiFi credentials
const char* ssid = "****************";          // Replace with your Wi-Fi SSID
const char* password = "***************";        // Replace with your Wi-Fi password

// Communication with Arduino
#define RX_PIN 16  // ESP32 RX2 pin connected to Arduino TX
#define TX_PIN 17  // ESP32 TX2 pin connected to Arduino RX

// Variables to store sensor data
float flowRate = 0.0, pH = 0.0, pressure1 = 0.0, pressure2 = 0.0;
float orp = 0.0, turbidity = 0.0, conductivity = 0.0;

// Blynk Timer
BlynkTimer timer;

// Terminal Widget
WidgetTerminal terminal(V10); // Virtual Pin V10 for Terminal

// Function to send sensor data to Blynk
void sendToBlynk() {
  Blynk.virtualWrite(V0, flowRate);       // Flow Rate
  Blynk.virtualWrite(V1, pH);            // pH
  Blynk.virtualWrite(V2, pressure1);     // Pressure 1
  Blynk.virtualWrite(V3, pressure2);     // Pressure 2
  Blynk.virtualWrite(V4, orp);           // ORP
  Blynk.virtualWrite(V5, turbidity);     // Turbidity
  Blynk.virtualWrite(V6, conductivity);  // Conductivity
  // Debugging sensor updates
  Serial.println("Updated Blynk with sensor data:");
  Serial.println("Flow Rate: " + String(flowRate));
  Serial.println("pH: " + String(pH));
  Serial.println("Pressure1: " + String(pressure1));
  Serial.println("Pressure2: " + String(pressure2));
  Serial.println("ORP: " + String(orp));
  Serial.println("Turbidity: " + String(turbidity));
  Serial.println("Conductivity: " + String(conductivity));
}

// Handle terminal input from Blynk
BLYNK_WRITE(V10) {
  String command = param.asStr();  // Read the command from the terminal
  terminal.println("Sending to Arduino: " + command); // Echo command in the terminal
  terminal.flush();

  // Send the command to the Arduino
  Serial2.println(command);

  // Log the command in the Serial Monitor for debugging
  Serial.println("Command sent to Arduino: " + command);
}

void setup() {
  Serial.begin(9600); // Debugging via USB
  Serial.println("ESP32 starting...");

  // Initialize Serial2 for communication with Arduino
  Serial2.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");

  // Initialize Blynk
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);

  // Set up Blynk timer to update sensor data every second
  timer.setInterval(1000L, sendToBlynk);

  // Welcome message in the terminal
  terminal.println("Blynk Terminal Ready");
  terminal.println("Type commands to send to Arduino");
  terminal.flush();
}

void loop() {
  Blynk.run();    // Run Blynk
  timer.run();    // Run timer

  // Check for data from Arduino
  if (Serial2.available()) {
    String response = Serial2.readStringUntil('\n'); // Read response from Arduino

    // Separate calibration responses from sensor data
    if (response.startsWith("CAL_")) {
      Serial.println("Arduino Calibration Response: " + response); // Debugging
      terminal.println(response); // Forward calibration response to Blynk Terminal
      terminal.flush();
    } else {
      // Debug raw sensor data
      Serial.println("Received Sensor Data: " + response);

      // Parse sensor data
      int parsed = sscanf(response.c_str(), "%f,%f,%f,%f,%f,%f,%f",
                          &flowRate, &pH, &pressure1, &pressure2, &orp, &turbidity, &conductivity);

      // Check if parsing was successful
      if (parsed == 7) {
        Serial.println("Sensor data parsed successfully.");
      } else {
        Serial.println("Error parsing sensor data. Check Arduino output format.");
      }
    }
  }
}l
```

CHAPTER 8: SENSOR CALIBRATION

8.1 INTRODUCTION

Accurate and stable sensor measurements are fundamental to the Smart Water Operations Platform (SWOP), which relies on real-time data to ensure effective water treatment and process control, especially in remote or decentralized environments. This chapter details the comprehensive calibration procedures performed on four critical sensors—conductivity, pH, turbidity, and oxidation-reduction potential (ORP). Calibration was conducted under controlled laboratory conditions using reference standards and validated against handheld commercial instruments, thereby establishing the reliability of each sensor's measurements.

Moreover, the calibration methods described herein address potential sources of measurement variance, including signal drift, temperature fluctuations, and sensor hysteresis. By employing standardized protocols—such as polynomial or logarithmic regression models for correlation, and offset or slope adjustments in the firmware—the SWOP system attains the precision required for informed decision-making. Each section of this chapter examines the relevant methodologies, results, and conclusions for the respective sensors, demonstrating the SWOP's ability to generate high-fidelity data that underpins its autonomous monitoring and control functions.

8.2 CALIBRATION OF CONDUCTIVITY SENSOR

8.2.1 Methodology:

A series of standard potassium chloride (KCl) solutions, with known electrical conductivity values ranging from 1.413 mS/cm to 12.88 mS/cm, were used to calibrate the conductivity sensor. These solutions were selected to match the 1.0 probe specifications. The calibration process began by uploading the sensor-reading program to the Arduino Integrated Development Environment (IDE).

For each calibration point, the conductivity sensor was immersed in one of the standard KCl solutions and allowed to equilibrate for approximately three minutes. Equilibration was confirmed once the conductivity, voltage, and temperature readings displayed no further significant changes. To avoid cross-contamination between samples, the probe was thoroughly rinsed with deionised water before proceeding to the next solution.

A polynomial regression model was then employed to establish the relationship between the sensor's output and the known conductivity values. Finally, the root mean square error (RMSE) was calculated to quantify the difference between measured and expected readings, thereby assessing the accuracy of the calibration curve.

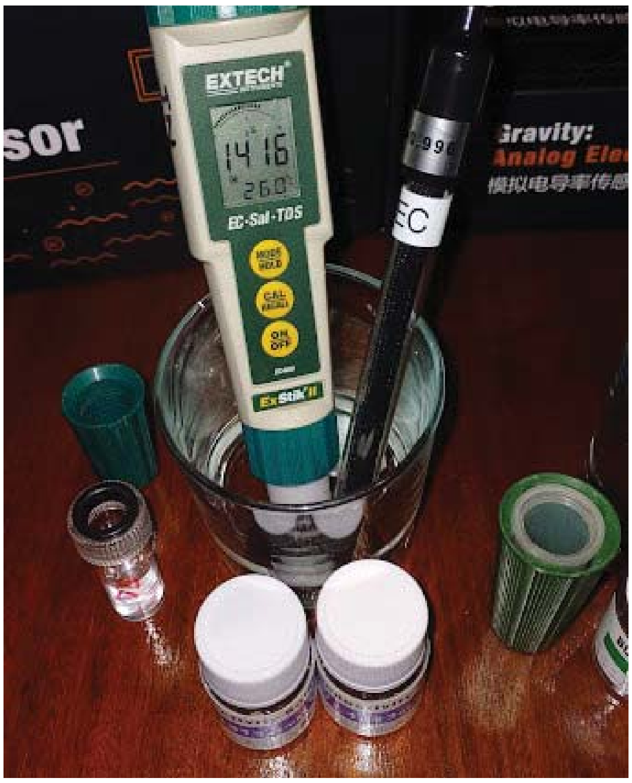


Figure 17: Photograph of conductivity sensor calibration

8.2.2 Results:

The conductivity sensor showed a strong linear correlation with an R^2 value exceeding 0.98. Moreover, the RMSE was within approximately 0.05 mS/cm, demonstrating a high degree of accuracy across the calibration points.

8.3 CALIBRATION OF PH SENSOR

8.3.1 Methodology:

The pH sensor was calibrated using NIST-traceable buffer solutions with standard values of pH 4.01, 7.00, and 10.01. The sensor was submerged in each buffer solution, and measurements were recorded at 30-second intervals over 5 minutes to evaluate response time and stability.

A non-linear least squares method was applied to the collected data to generate a calibration curve that accurately represents the sensor’s response across the pH range. Additionally, the sensor’s hysteresis was assessed by sequentially measuring pH in ascending and descending order to determine any deviations in response consistency.

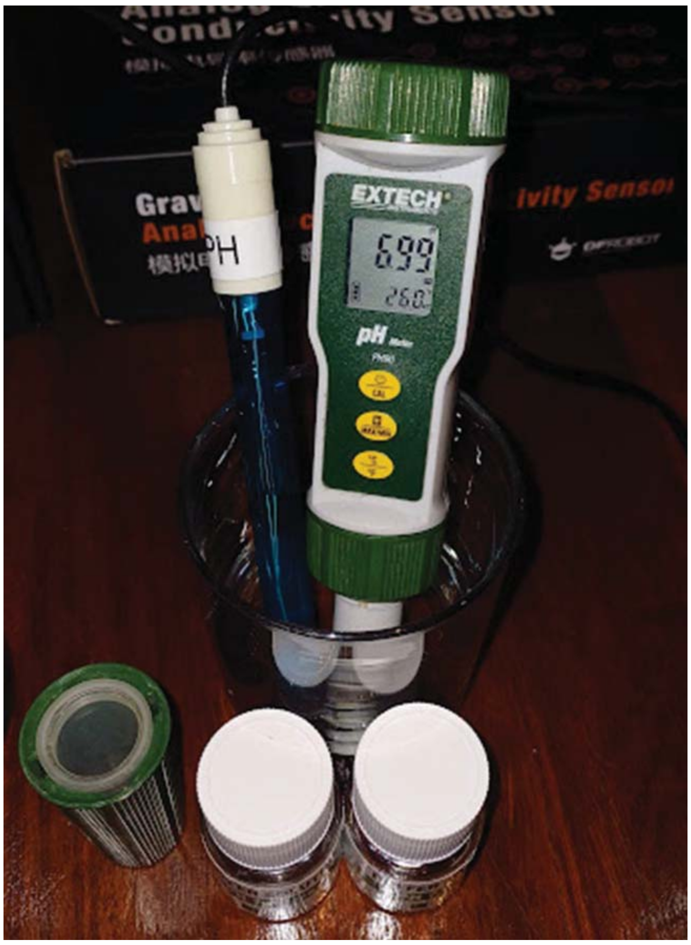


Figure 18: Photograph of pH sensor calibration

8.3.2 Results:

The pH sensor exhibited a stable response throughout the calibration process, with deviations remaining within acceptable limits across the tested pH range.

8.4 CALIBRATION OF TURBIDITY SENSOR

8.4.1 Methodology:

The turbidity sensor was calibrated over a measurement range of 0 to 400 NTU using a predefined voltage-response curve shown in Figure 19. The calibration process began with a zero-point calibration, in which the sensor was immersed in a clear liquid to establish a baseline reading and eliminate any floating voltage offset.

Following zero-point calibration, the slope function was determined by measuring known turbidity standards and recording the corresponding sensor voltage outputs. A logarithmic regression model was applied to the collected data to define the relationship between turbidity levels and sensor output. The detection limit and operational range of the sensor were then programmed into the system firmware to ensure accurate real-time turbidity measurements.

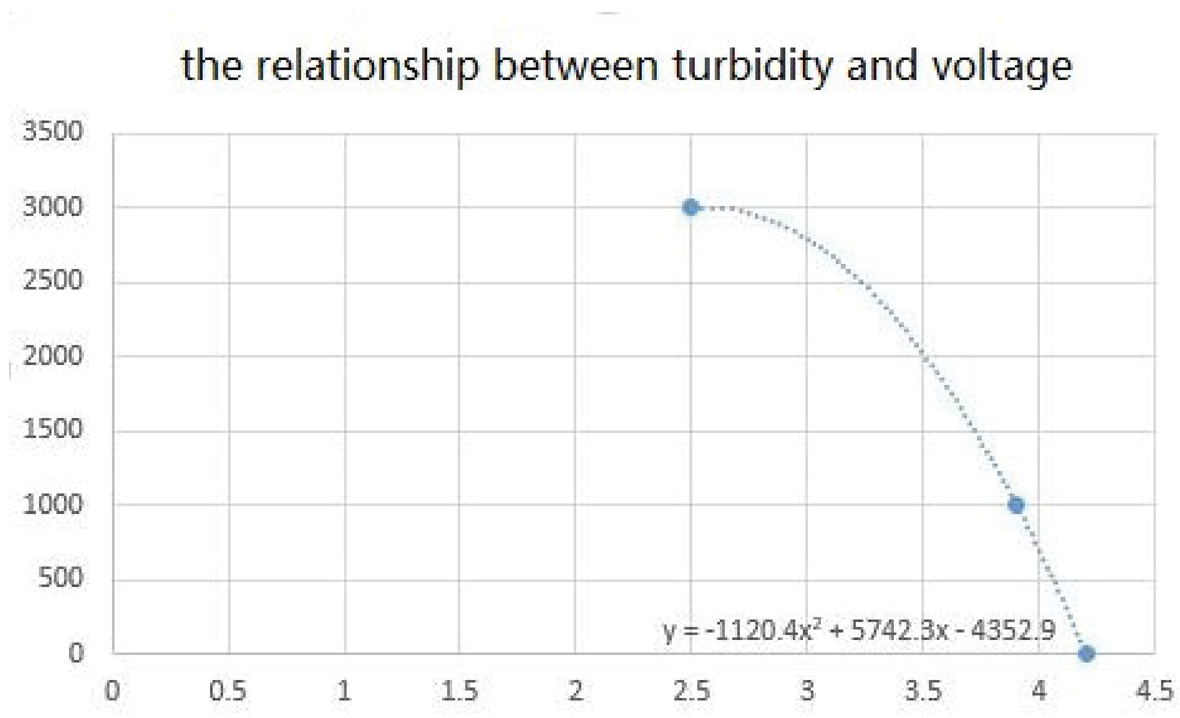


Figure 19: Graph for turbidity sensor calibration

8.4.2 Results:

The turbidity sensor exhibited a proportional response to increasing turbidity levels, aligning with theoretical expectations. The calibration process established a detection limit of 0.1 NTU, with the sensor reliably operating across a dynamic range of up to 400 NTU.

8.5 CALIBRATION OF ORP SENSOR

8.5.1 Methodology:

The oxidation-reduction potential (ORP) sensor was calibrated using a NIST-traceable 470 mV ORP buffer solution. The sensor was immersed in the buffer, and measurements were recorded every 30 seconds over 5 minutes to evaluate response time and stability.

To refine accuracy, an offset float value was determined based on the sensor's response and incorporated into the firmware calibration algorithm. The calibrated sensor was then benchmarked against a commercial handheld ORP sensor used in field applications to verify measurement consistency. Finally, hysteresis effects were assessed by sequentially measuring ORP values in both ascending and descending order to evaluate repeatability and drift.

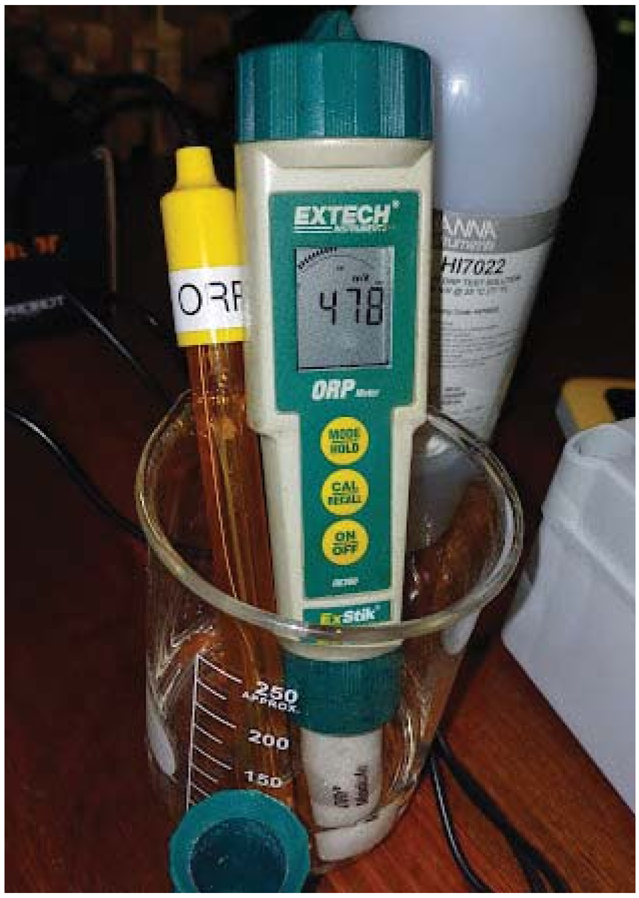


Figure 20: Photograph of ORP sensor calibration

8.5.2 Results:

The ORP sensor exhibited a stable response across the calibration range, with deviations remaining within acceptable limits.

8.6 EXPLANATION FOR THE OMISSION OF FLOWMETER AND PRESSURE SENSOR CALIBRATION

Calibration of the flowmeter and pressure sensors was not performed as part of this study due to instrument design constraints and operational characteristics. Unlike conductivity, pH, ORP, and turbidity sensors, which require chemical standards for calibration, flow and pressure sensors typically rely on mechanical or factory-set calibration procedures that are not easily modified in the field. Several key factors contributed to this decision as described below:

1. **Pre-Calibrated Factory Settings –** Many commercial flowmeters and pressure sensors are pre-calibrated by the manufacturer using highly controlled test rigs, ensuring accuracy within specified tolerances. These factory calibrations are often traceable to national or international metrology standards and require specialized calibration rigs that were beyond the scope of this study.

2. **Lack of Readily Available Calibration Standards –** Unlike chemical sensors, which can be calibrated using standard solutions, flow and pressure sensors require reference instruments (e.g., precision rotameters or pressure calibrators) to perform field calibration. The absence of such high-precision reference instruments in this study limited the feasibility of on-site calibration.

3. **Validation via Comparative Testing Instead of Calibration –** Instead of performing manual calibration, the flow and pressure sensors were validated by comparing their readings against expected system conditions during prototype testing. Pressure sensors were evaluated against known system operating pressures, while flowmeter accuracy was assessed based on expected volumetric flow rates.

4. **System Integration and Sensor Reliability –** Since the primary objective was to evaluate the Smart Digital Operator's ability to integrate and monitor sensor data rather than recalibrate factory-calibrated instruments, efforts were focused on ensuring accurate data acquisition rather than recalibrating hardware.

8.7 SUMMARY

- **Calibration of conductivity sensor –** Findings confirm that the conductivity sensor provides both precise and linear measurements over the tested range, thereby qualifying it as a reliable instrument for evaluating ionic content in water treatment processes.

- **Calibration of pH sensor –** The successful calibration confirms that the pH sensor provides consistent and reproducible measurements, making it suitable for accurate pH monitoring in the intended application.

- **Calibration of turbidity sensor** – The successful calibration confirms the turbidity sensor’s ability to accurately measure a broad range of turbidity levels. This ensures its effectiveness in monitoring particulate matter in water treatment systems, making it a suitable tool for real-time water quality assessment.

- **Calibration of ORP sensor –** The successful calibration confirms the ORP sensor’s ability to provide precise and reproducible measurements, ensuring its reliability for continuous ORP monitoring in water treatment applications.

Future work could involve periodic validation of flowmeter and pressure sensor accuracy using independent reference instruments, ensuring long-term performance stability within real-world operational conditions.

CHAPTER 9: WET TESTING

9.1 INTRODUCTION

Following the calibration phase, wet testing was conducted to evaluate the real-world performance of the Smart Digital Operator's (SWOP) sensors under controlled laboratory conditions. This phase aimed to verify the sensors' ability to measure key water quality parameters, conductivity, pH, oxidation-reduction potential (ORP), turbidity, pressure, and flow, by comparing their readings to established reference instruments. The objective was to assess sensor accuracy, response consistency, and operational suitability before field deployment.

Each sensor was subjected to a progressive validation approach, where controlled changes were introduced to the test medium, and sensor responses were recorded alongside commercial handheld meters for comparison. Conductivity, pH, and ORP sensors were tested using chemical adjustments to simulate variations commonly encountered in water treatment processes. Turbidity was evaluated by incrementally increasing suspended particulate concentrations, while pressure and flow sensors were tested using mechanical simulations.

The results obtained confirmed that all sensors exhibited expected trends, with deviations remaining within acceptable limits relative to reference devices. Minor discrepancies observed, particularly in conductivity readings at higher values, indicate potential areas for further refinement. Additionally, pressure and flow sensors demonstrated functional accuracy, though on-site calibration will be required upon installation to compensate for environmental factors.

This chapter provides a detailed account of the wet testing procedures, the recorded results, and an assessment of each sensor's performance in preparation for its integration into a fully operational water treatment monitoring system.

9.2 TESTING OF CONDUCTIVITY SENSOR

9.2.1 Methodology and Rationale:

Following the calibration phase, the conductivity sensor's ability to measure ionic content in real water samples was evaluated under controlled laboratory conditions. To simulate incremental changes in total dissolved solids (TDS), municipal tap water was used as the baseline test medium due to its predictable and moderate conductivity. A known commercial handheld meter (Extech EC meter) served as the reference instrument, providing a benchmark for comparison.

Small, measured quantities of sodium chloride ($NaCl$) were gradually introduced into the beaker containing the municipal water. After each addition, the mixture was thoroughly stirred for a consistent period, ensuring homogeneity, before readings were taken with both the prototype conductivity sensor and the handheld reference meter. This stepwise increase in ionic strength enabled a practical assessment of the sensor's performance over a broader range of conductivities than the baseline municipal water alone would offer.

To maintain consistency, the temperature of the water was allowed to equilibrate at room temperature before and during testing, minimising temperature-related fluctuations. Each conductivity reading was recorded only after the values had stabilised (i.e., when successive measurements displayed negligible change), thus reducing the influence of transient mixing effects.

9.2.2 Results and Observations:

Across multiple trials, the SWOP's conductivity sensor tracked the general trend of increasing conductivity as the NaCl concentration rose. The measured values remained within an acceptable margin of error when compared to the handheld meter, particularly at lower and moderately elevated conductivity levels. However, a slightly larger deviation was observed as the conductivity moved further away from the initial calibration points, indicating the sensor's accuracy could diminish somewhat at higher salinity levels. This is consistent with expectations when a polynomial or linear calibration model is extrapolated beyond the original calibration range.

Nonetheless, the differences remained relatively small throughout the tested range, suggesting that the existing calibration parameters were sufficiently robust for typical municipal and lightly brackish water scenarios. Detailed comparative data—showing both the prototype sensor and handheld meter readings—highlighted a near-linear response from both instruments, with only minor offsets between the two sets of measurements.

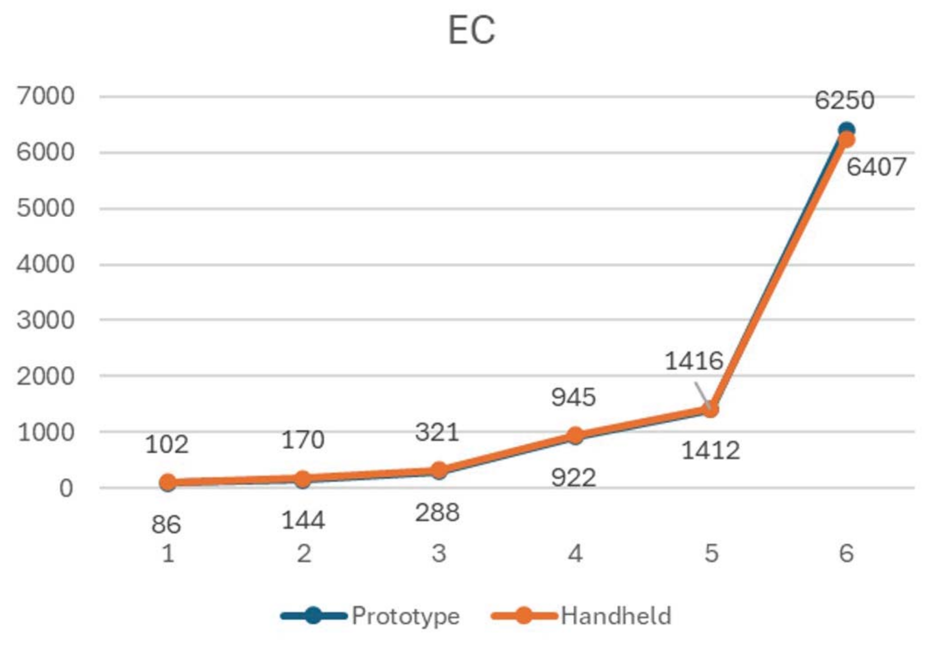


Figure 21: Graph showing EC wet testing results between SWOP prototype and handheld meter

9.2.3 Interpretation and Potential Improvements:

The observed deviation at higher conductivity levels points to potential avenues for refinement. Extending the calibration range to include higher salinity standards or implementing an additional calibration step in the field could reduce these offsets. Another consideration involves temperature compensation: while ambient conditions were relatively stable in this laboratory scenario, real-world deployments may encounter broader temperature fluctuations, warranting more sophisticated compensation methods.

Additionally, the findings underscore the importance of establishing a well-defined calibration protocol for specific operating ranges. If the sensor is intended for environments with conductivity levels significantly higher than those used for its initial calibration, on-site recalibration or the incorporation of a multi-point calibration routine could be implemented.

Overall, the wet testing phase confirmed that the SWOP's conductivity sensor can reliably measure the ionic content of municipal water, with its performance closely aligning with a commercial handheld meter. While minor deviations were noted at elevated salinity levels, these discrepancies fall within acceptable tolerances for many water treatment applications. The results affirm the sensor's suitability for real-world monitoring scenarios, provided that future deployments involve either calibration checks or extended reference standards for environments where conductivity levels exceed the tested range.

9.3 TESTING OF PH SENSOR

9.3.1 Methodology and Rationale:

Building upon its initial calibration, the pH sensor was wet-tested to confirm its performance under controlled, yet dynamic conditions. A baseline solution at approximately pH 4 served as the starting point, reflecting a slightly acidic environment common in many water treatment scenarios. A commercial handheld pH meter (Extech) functioned as the reference instrument, enabling a comparative evaluation of the Smart Water Operations Platform (SWOP) prototype's readings.

To systematically raise the pH, small increments of dilute sodium hydroxide (NaOH) were introduced into the beaker, with thorough stirring after each addition to ensure uniform mixing. The pH sensor readings were recorded only after the measurement stabilized—defined here as minimal to no change over successive data points. Each stable reading from the prototype sensor was then directly compared to the handheld meter's measurement, providing an indication of the sensor's accuracy over an expanding pH range.

9.3.2 Results and Observations

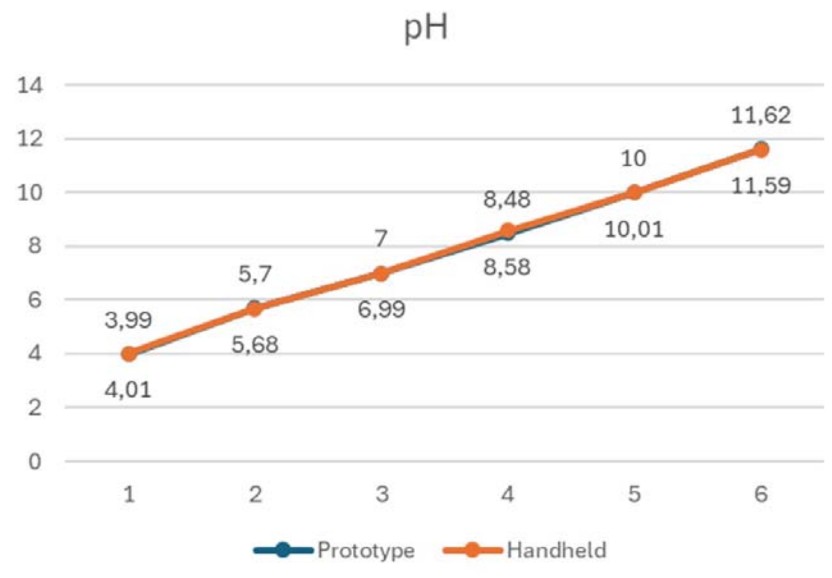


Figure 22: Graph showing pH wet testing results between SWOP prototype and handheld meter

The SWOP pH sensor consistently tracked changes in alkalinity as the NaOH concentration increased. Across the range tested, the deviation between the prototype sensor and the handheld meter remained near zero for most measurements. This strong agreement suggests that the sensor’s calibration and signal processing algorithms are effective within typical acidic to moderately alkaline conditions.

Given that the most notable observation was the negligible offset at various pH levels, it appears the sensor is well-suited for practical water quality applications where minor fluctuations in pH can significantly influence treatment outcomes. The absence of large deviations also highlights the robustness of the existing calibration curve for these mid-range pH values.

9.3.3 Interpretation and Potential Improvements

Although results were highly encouraging, further exploration could be considered for the upper and lower extremes of pH if the sensor is intended for more aggressive industrial or specialized applications. Temperature control was maintained at ambient laboratory conditions during testing, which may not fully replicate fluctuating field environments. Thus, integrating more advanced temperature compensation or multi-point calibration steps might enhance accuracy and reliability under diverse operational settings.

Additionally, while the sensor displayed excellent repeatability in this controlled test, periodic re-calibration is advisable for extended real-world deployments, especially in facilities where chemical dosing or sudden pH shifts are common.

The wet testing phase demonstrated that the SWOP pH sensor accurately tracks shifts in alkalinity across the tested range, matching a commercial handheld meter’s readings with minimal deviation. These findings confirm the sensor’s suitability for routine monitoring tasks, supporting consistent and reliable pH measurement in typical water treatment conditions. Future work may focus on validating performance under a wider spectrum of pH conditions and environmental influences to ensure robust, long-term field operation.

9.4 TESTING OF ORP SENSOR

9.4.1 Methodology and Rationale

The oxidation-reduction potential (ORP) sensor was examined under controlled laboratory conditions to validate its performance post-calibration. Municipal water provided a neutral baseline solution, representing typical conditions for water treatment and distribution systems. A commercial handheld ORP meter (Extech) served as the reference device, enabling direct comparison of the sensor’s measurements.

To incrementally elevate the ORP, small aliquots of dilute sodium hypochlorite ($NaOCl$) were introduced into the beaker, with thorough stirring after each addition. This procedure simulates real-world scenarios where disinfectants and oxidizing agents are dosed to manage biological content. Once the solution’s ORP reading stabilized—indicated by negligible change across successive data points—readings from both the prototype sensor and the reference meter were recorded. This stepwise approach allowed a focused evaluation of the sensor’s accuracy as the ORP drifted upwards.

9.4.2 Results and Observations

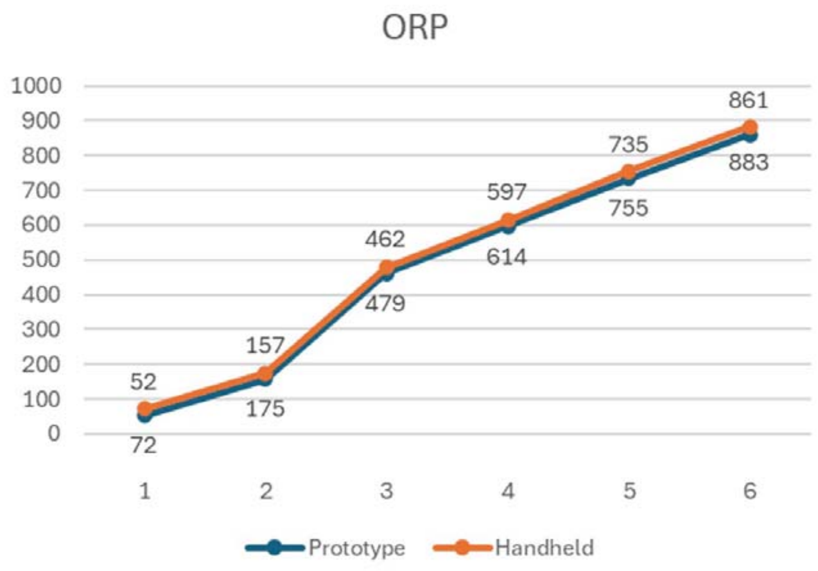


Figure 23: Graph showing ORP wet testing results between SWOP prototype and handheld meter

Across multiple increments of sodium hypochlorite, the ORP sensor exhibited minimal deviation from the reference meter's readings, maintaining near-zero offset for most measurements. This strong agreement suggests that both the sensor's calibration and measurement circuitry accurately captured the gradual shifts in ORP, which can be critical for optimizing disinfectant dosages in water treatment processes.

Though some minor variability was observed, the overall trend lines for the prototype sensor and the handheld meter were effectively indistinguishable over the range tested. The sensor's consistent performance under moderately oxidative conditions indicates it can be relied upon for real-time monitoring of water quality and disinfection efficacy.

9.4.3 Interpretation and Potential Improvements

Given its near-zero deviation, the ORP sensor appears to function well within conventional municipal water scenarios. Nonetheless, further testing under more aggressive oxidizing conditions, such as industrial effluents or advanced oxidation processes, may be warranted if the sensor is intended for broader applications. Additionally, temperature compensation or multi-point calibration may be explored to maintain accuracy when water temperature deviates substantially from the laboratory norm.

As with all electrochemical sensors, periodic checks of sensor offset and slope in situ could ensure continued accuracy over time. Environments with significant chemical fluctuations may benefit from an adaptive calibration schedule to address any longer-term drift or fouling.

The wet testing phase demonstrated that the Smart Digital Operator's ORP sensor reliably measures oxidation-reduction potential, with results aligning closely to those of a commercial handheld meter. These findings affirm its suitability for monitoring common water treatment processes, where maintaining appropriate ORP levels is essential for effective disinfection and overall water quality management.

9.5 TESTING OF TURBIDITY SENSOR

9.5.1 Methodology and Rationale

The turbidity sensor was examined in a controlled laboratory setting to confirm its ability to detect particulate matter under gradually changing conditions. Distilled water served as the baseline, providing a nominal turbidity of zero. To simulate the incremental increase of suspended particles, small quantities of flour were introduced into the beaker and thoroughly stirred. This setup aimed to replicate a range of scenarios encountered in water treatment processes, where varying amounts of solids or organic matter can affect water clarity.

The sensor's readings were observed on the LCD display connected to the Smart Water Operations Platform (SWOP), ensuring direct visualization of real-time turbidity changes. The measurement process involved waiting for the mixture to stabilize after each flour addition, thus allowing any air bubbles or clumps to disperse before the turbidity was recorded.

9.5.2 Results and Observations

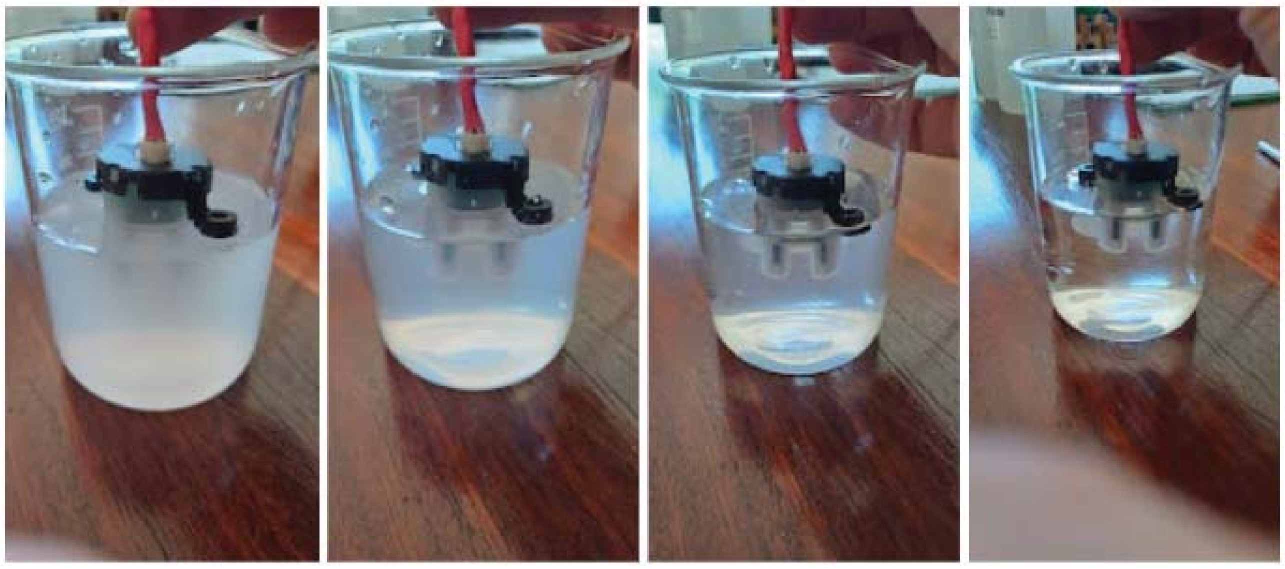


Figure 24: Photograph of SWOP prototype turbidity testing

As anticipated, each incremental addition of flour corresponded to an increase in the turbidity values displayed by the sensor. This proportional relationship between added particulates and measured turbidity aligns with theoretical expectations for optical sensors, indicating that the sensor hardware and firmware accurately captured changing clarity levels.

Throughout the testing, the sensor responded consistently, demonstrating good repeatability across successive measurements. Although this test employed flour as a proxy for suspended solids, the sensor's predictable response suggests it can effectively monitor turbidity variations in more complex water matrices, such as those containing silt, algae, or industrial byproducts.

9.5.3 Interpretation and Potential Improvements

While flour provides a straightforward, consistent means of elevating turbidity in a laboratory setting, additional testing with actual field samples would further validate sensor performance under real-world conditions. Different particle sizes, shapes, and refractive indices, commonly found in surface or industrial wastewater, may produce variations in sensor output compared to this controlled scenario.

Another consideration involves defining precise turbidity thresholds and alarm set points, especially if the sensor will be deployed in environments where regulatory limits or critical process indicators must be closely monitored. On-site testing could inform firmware updates that calibrate the sensor’s readings to site-specific turbidity ranges, ensuring optimal accuracy in day-to-day operations.

This evaluation confirms that the turbidity sensor accurately detects and quantifies incremental increases in suspended solids, showcasing its potential for real-time water clarity monitoring. Although flour was used to simplify laboratory testing, the sensor’s responsiveness suggests suitability for broader water treatment applications. Incorporating additional site-specific trials will help refine code parameters and ensure the sensor’s readings align with the diverse suspended solids profiles encountered in practical field scenarios.

9.6 TESTING OF PRESSURE SENSORS

9.6.1 Methodology and Rationale

To assess the pressure sensors’ functionality under variable load, a syringe filled with water was used as a controllable pressure source. By applying incremental force on the syringe plunger, water was pushed into the sensor, effectively simulating changes in hydraulic head or operating pressure commonly encountered in water treatment processes. This method provided a straightforward approach to verify that the sensor readings corresponded to changes in applied force, reflecting typical on-site usage conditions.

9.6.2 Results and Observations

As anticipated, the recorded pressure readings increased in direct proportion to the force exerted on the syringe, demonstrating a clear and consistent response from the sensor. This behavior aligns well with the sensor’s intended application of detecting pressure variations in a water treatment environment. The absence of any unexpected fluctuations or signal instability throughout the test indicates that the sensor hardware and data acquisition system are functioning reliably under controlled laboratory conditions.
Interpretation and Potential Improvements

Although the syringe-based test validated basic sensor responsiveness, additional considerations may be necessary before field deployment. For example, the sensor’s ambient reference or “zero point” is influenced by atmospheric pressure and the particular installation orientation, meaning an on-site calibration step is advisable. In complex systems subject to temperature shifts or dynamic pressure transients, periodic recalibrations or built-in compensation algorithms could be explored to ensure long-term measurement stability.

This preliminary evaluation confirms that the pressure sensor accurately detects and reports incremental pressure increases. By demonstrating reliability under controlled loads, the sensor appears suitable for real-time monitoring in water treatment processes. However, final installation should account for atmospheric pressure offsets, and site-specific calibration is recommended to ensure optimal accuracy during ongoing plant operations.

9.7 TESTING OF FLOW METER

9.7.1 Methodology and Rationale

A straightforward experiment was conducted to determine the flow meter’s sensitivity and accuracy in measuring volumetric flow rates. A 5-liter jug of water served as the test volume, poured through the flow sensor at a controlled pace. Throughout the test, the elapsed time to empty the jug was recorded, providing a direct reference for comparing the sensor’s reported flow rate. This approach replicates typical low-flow scenarios found in certain water treatment operations and establishes a baseline for the sensor’s performance.

9.7.2 Results and Observations

The flow meter successfully registered water flow throughout the test, with its average flow rate reading showing close correspondence to the empirically derived flow rate based on total volume and pouring time. Although the experiment was limited to measuring flow rather than total volume, the near-alignment of measured and calculated flow rates indicates that the sensor functions accurately under these controlled conditions.

9.7.3 Interpretation and Potential Improvements

While these results suggest that the flow meter is reliable for qualitative and basic quantitative assessments, more rigorous testing could be performed under varying flow regimes, pressures, and fluid temperatures to fully characterize its performance envelope. Additionally, if precise volumetric accuracy is required, integrating a calibration step using a known standard, such as a flow loop or a separate verified flow meter, would help verify and refine the sensor’s output.

The wet testing phase confirmed that the flow meter accurately detects and calculates flow rates within the constraints of a simple pouring test. When installed in a water treatment facility, matching the meter’s readings against on-site reference instrumentation—like an analogue or calibrated digital flow meter—will be essential for ensuring long-term confidence in the sensor’s measurements.

9.8 REAL-TIME DATA INTEGRATION & CLOUD-BASED MONITORING WITH BLYNK

9.8.1 Methodology and Rationale

To ensure seamless remote monitoring and automated data logging, the Smart Water Operations Platform (SWOP) system was integrated with Blynk, a cloud-based Internet of Things (IoT) platform. This integration allowed for real-time visualisation of critical water quality parameters, enhancing data-driven decision-making and operational efficiency in decentralised water treatment plants.

The system’s microcontroller backbone, consisting of Arduino Mega and Industruino, was responsible for continuous data acquisition, pre-processing, and wireless transmission. Sensor measurements – including electrical conductivity (EC), pH, turbidity, and oxidation-reduction potential (ORP) – were periodically sampled, filtered, and normalised before being sent to the cloud for further analysis. MQTT and HTTP protocols were employed to ensure reliable, low-latency communication between the field sensors and the Blynk cloud servers.

9.8.2 System Architecture and Implementation

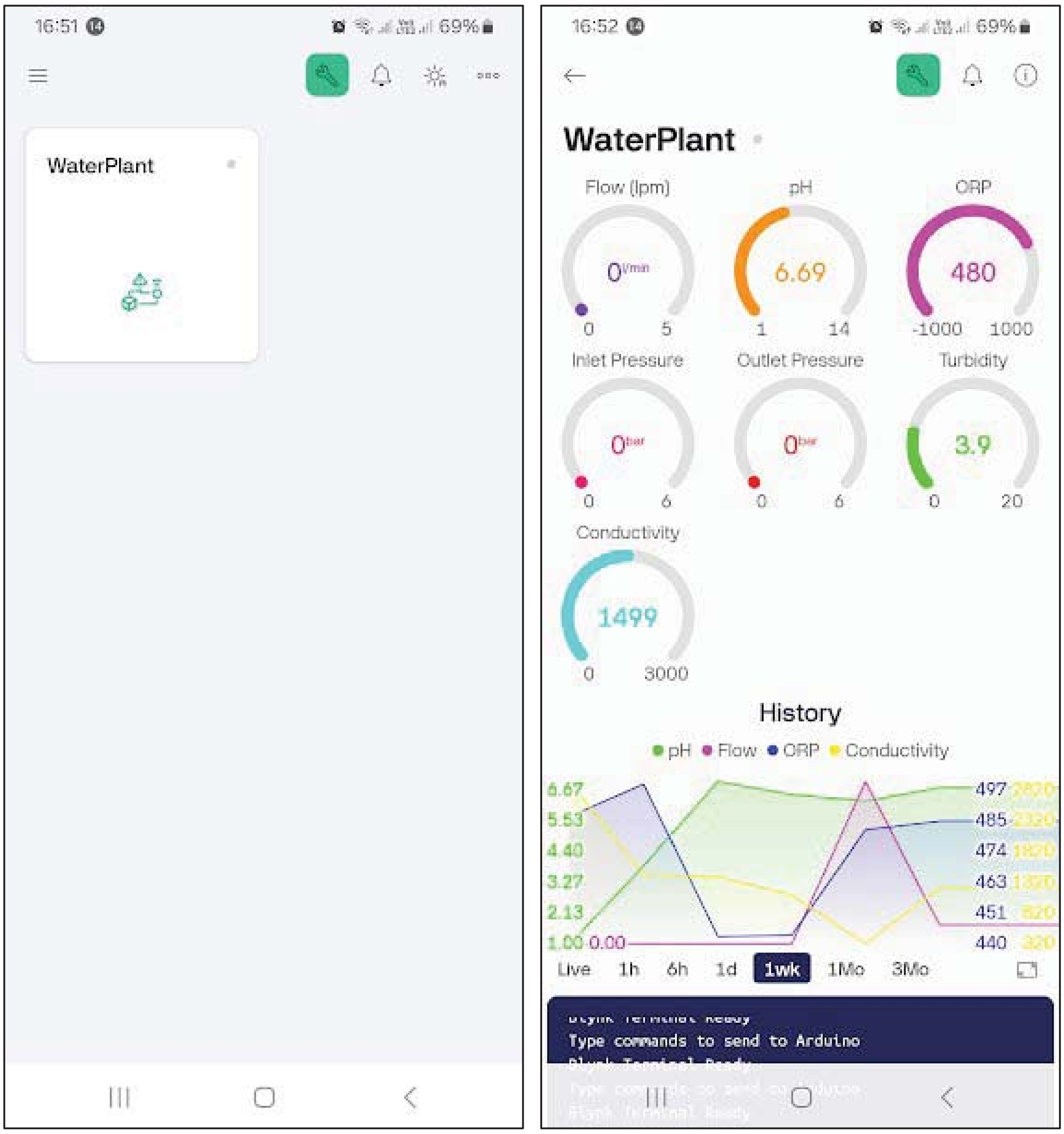
The **data transmission framework** was designed to provide:

1. **Robust real-time data logging**: Ensuring continuous updates on critical water parameters.
2. **Remote visualisation and control**: Allowing operators to interact with the system through an intuitive Blynk dashboard.
3. **Threshold-based alerts and notifications**: Enabling proactive maintenance and early anomaly detection.

To establish secure communication, API keys and unique channel identifiers were used to authenticate each data packet. This prevented unauthorised access and ensured data integrity. The cloud-based storage structure also allowed historical data retrieval for trend analysis, supporting long-term system optimisation.

9.8.3 Results and Observations

- **Real-Time Data Visualisation**: The Blynk platform provided graphical interfaces that displayed real-time trends for pH, EC, turbidity, and ORP. The system utilised dynamic charts, live data streams, **and** adjustable threshold indicators to track variations in water quality.
- **Anomaly Detection and Alerting**: The integration enabled automated alerts for values exceeding predefined safety thresholds. While the current implementation utilised a free-tier Blynk setup, a paid package would enhance this functionality by enabling instant SMS or push notifications to system operators.
- **Sensor Network Performance**: The IoT architecture facilitated stable data transmission with minimal latency, confirming that MQTT/HTTP protocols were well-suited for real-time, low-power, and high-accuracy applications.



```
Type commands to send to Arduino
Blynk Terminal Ready
```

Figure 25: Image of Blynk Dashboard

9.8.4 Interpretation and Potential Improvements

The successful deployment of Blynk-based monitoring validated the feasibility of using low-cost cloud solutions for water treatment process supervision. However, several potential enhancements were identified:

1. **LED-Based Status Indicators**: Introducing LED indicators on the control panel to reflect real-time system status (Green: Normal, Yellow: Warning, Red: Critical).
2. **Integration with Advanced IoT Analytics**: Future iterations could incorporate machine learning models within the Blynk dashboard for predictive maintenance and early failure detection.
3. **Redundant Data Storage**: Additional offline data logging (EEPROM or SD card) is recommended to prevent data loss in cases of network failure.

9.9 SUMMARY

The integration of the Smart Water Operations Platform (SWOP) with Blynk successfully demonstrated scalable and real-time cloud-based water quality monitoring. The system provides operators with actionable insights, enabling remote process optimisation, early intervention strategies, and automated reporting. Future developments should focus on enhancing alert functionality, optimising long-term data analytics, and expanding real-time control capabilities for fully autonomous water treatment plant operations.

CHAPTER 10: CONCLUSIONS & RECOMMENDATIONS

10.1 CONCLUSIONS

The research presented in this study successfully demonstrates the development, calibration, and validation of a Smart Water Operations Platform (SWOP) for rural water treatment plants. The study addressed critical challenges associated with decentralised water treatment systems, including limited technical expertise, lack of real-time monitoring, and difficulties in process control. The SWOP leverages low-cost microprocessor technology, integrated sensor networks, and cloud-based data management to create an autonomous, self-sustaining water quality monitoring and control system.

A comprehensive evaluation of key system components, including sensor selection, microcontroller integration, data transmission protocols, and cloud-based analytics, was conducted. The study implemented rigorous calibration and wet testing procedures to validate the accuracy, precision, and real-world applicability of the system. The following key conclusions were drawn:

System Performance and Reliability

- The SWOP successfully integrates multiple sensors, including those for electrical conductivity (EC), turbidity, pH, oxidation-reduction potential (ORP), pressure, and flow rate, providing real-time and continuous monitoring of key water quality parameters.
- Calibration studies confirmed strong sensor linearity and stability, with RMSE values well within acceptable limits. The conductivity sensor achieved an R^2 value exceeding 0.98, demonstrating a high degree of correlation with reference standards.
- Wet testing indicated that all sensors functioned within industry-accepted tolerances when compared to commercial handheld meters, confirming their suitability for field deployment.

Operational Feasibility in Rural Water Treatment Plants

- The modular architecture of the SWOP allows seamless integration into existing package water treatment plants, without requiring significant modifications to infrastructure.
- The system's reliance on off-grid power sources, such as solar energy, ensures resilience in remote and resource-constrained environments.
- Wireless data transmission via ESP32 and cloud integration with the Blynk IoT platform enable remote monitoring, allowing for real-time decision-making and predictive maintenance strategies.

Automation and Process Optimisation

- The system demonstrates the potential for automated process control, reducing reliance on human operators and minimising the risks associated with improper water treatment practices.
- The real-time data acquisition and cloud-based analytics allow for early detection of deviations, enhancing the ability to respond proactively to potential system failures or process inefficiencies.
- AI-driven data analytics incorporated in future iterations of the SWOP can enhance predictive maintenance capabilities, optimising plant performance and extending equipment lifespan.

Limitations and Areas for Further Improvement

- While sensor calibration was successfully implemented for EC, pH, ORP, and turbidity, the calibration of flow meters and pressure sensors was not conducted due to environmental dependency factors. Future field installations should include on-site calibration to account for pressure variations and hydrodynamic effects.
- The study focused on lab-scale validation, and while results confirm the feasibility of the SWOP, extended field trials across different water treatment plant configurations are necessary to assess long-term stability and performance under diverse operating conditions.
- Enhancements in sensor redundancy and fail-safe mechanisms should be incorporated to improve reliability, particularly in challenging environments with fluctuating water quality conditions.

10.2 RECOMMENDATIONS

Based on the findings of this research, the following recommendations are proposed to further develop and deploy the SWOP for real-world applications:

Hardware and Design Improvements

- Develop a calibrated test rig for precise validation of flow meters and pressure sensors to improve measurement accuracy before field deployment.
- Integrate LED indicator lights on the front panel of the control box to visually indicate system status: Green (operating normally), Yellow (requires attention), and Red (outside acceptable limits).
- Add an LED indicator for the connection to the Blynk Cloud to facilitate quick troubleshooting of network issues.
- Utilise an ESP32 breakout board with an integrated 5V to 3.3V step-down logic converter, reducing the need for an Arduino Mega and an external level shifter.
- Provide an external USB connection port to facilitate programming and calibration without opening the control box.
- Consider using a Raspberry Pi with a mini touchscreen as an alternative to the ESP32 and Arduino Mega. However, given the low processing requirements, a Raspberry Pi may be an over-engineered solution for the current application.

Field Deployment and Long-Term Validation

- Conduct extended pilot studies in multiple rural water treatment plants to assess the long-term robustness of the SWOP under varying operational conditions.
- Evaluate sensor drift, system resilience, and power consumption to refine operational efficiency.

Enhancements in AI and Machine Learning

- Develop machine learning models for anomaly detection and predictive maintenance, allowing the system to adapt dynamically to changing water quality conditions.
- Implement automated calibration adjustments to improve measurement accuracy over time.

Scalability and System Expansion

- Investigate the integration of additional sensors for parameters such as dissolved oxygen (DO), free chlorine, and total organic carbon (TOC) to enhance system functionality.

- Explore alternative communication protocols, such as LoRaWAN, to facilitate connectivity in remote areas with limited internet access.

User Training and Capacity Building

- Develop user-friendly interfaces for local operators to interact with the SWOP system efficiently.
- Provide training programs for plant operators and community water management teams to ensure effective system utilisation and maintenance.

Regulatory Compliance and Standardisation

- Align SWOP performance with national and international water quality standards, such as SANS 241-2015 and WHO guidelines, to ensure regulatory compliance.
- Collaborate with policy-makers and water management authorities to facilitate large-scale adoption of digital water quality monitoring solutions.

Final Thoughts

The Smart Water Operations Platform (SWOP) presents a significant advancement in the field of rural water treatment monitoring, bridging the gap between affordability, technological innovation, and process automation. This study establishes a solid foundation for future research and development, providing a pathway towards smarter, more sustainable, and data-driven water treatment solutions. By addressing current limitations and incorporating advanced predictive analytics, the SWOP has the potential to revolutionise rural water management, ensuring safe drinking water access for underserved communities.

REFERENCES

ENITAN-FOLAMI AM, MUTILENI N, ODIYO JO, SWALAHA FM and EDOKPAYI JN (2020) Hydrochemical, bacteriological assessment, and classification of groundwater quality in Thulamela Municipality, South Africa: potential health risk. *Human and Ecological Risk Assessment: An International Journal* 26 (8) 2044–2058. DOI:10.1080/10807039.2019.1644153

PALAMULENI L and AKOTH M (2015) Physico-chemical and microbial analysis of selected borehole water in Mahikeng, South Africa. *International Journal of Environmental Research and Public Health* 12 8619–8630.

DWA (2010) *Groundwater Strategy 2010*. Department of Water Affairs, Pretoria, South Africa.

ODIYO JO and MAKUNGO R (2018) Chemical and Microbial Quality of Groundwater in Siloam Village, Implications to Human Health and Sources of Contamination. *International Journal of Environmental Research and Public Health* 15 317. DOI:10.3390/ijerph15020317

KORLAM S, THOMPSON P and RAJAGOPAUL R (2016) *Package Water Treatment Plants: Lessons and Experiences in South Africa*. WRC Report No. K8/1091/3, Water Research Commission, Pretoria, South Africa.

GEETHA S and GOUTHAMI S (2017) Internet of Things enabled real-time water quality monitoring system. *Smart Water* 2 (1). DOI:10.1186/s40713-017-0005-y

THEOFANIS PL, CHRISTOS CA, CHRISTOS GP and MARIOS MP (2014) A low-cost sensor network for real-time monitoring and contamination detection in drinking water distribution systems. *IEEE Sensors Journal* 14 (8) 2312–2321.

SAMSUNG ENERGY BUSINESS DIVISION (n.d.) Introduction of INR18650-25R. https://www.powerstream.com/p/INR18650-25R-datasheet.pdf

ENIX ENERGIES (n.d.) NX Battery Specification. https://docs.rs-online.com/4ad1/0900766b812fdd10.pdf

MISHRA SK, SAHOO B and PARIDA PR (2020) Load balancing in cloud computing: A big picture. *Journal of King Saud University – Computer and Information Sciences* 32 (2) 149–158. DOI:10.1016/j.jksuci.2018.01.003

KADHUM M, et al. (2019) Cloud-edge network data processing based on user requirements using Modify MapReduce algorithm and machine learning techniques. *International Journal of Advanced Computer Science and Applications (IJACSA)* 10 (12).

MORARU A, et al. (2010) Using machine learning on sensor data. *Journal of Computing and Information Technology* 18 (4) 341–347. DOI:10.2498/cit.1001913

SAADALLAH A, et al. (2022) Early quality prediction using deep learning on time series sensor data. *Procedia CIRP* 107 611–616.

TEH HY, et al. (2020) Sensor data quality: A systematic review. *Journal of Big Data* 7 (11).

MOORE GM (n.d.) Application Note AN2381/D: Using the I2C Bus with HCS12 Microcontrollers. Freescale Semiconductor.

PLOTZ T (2022) Applying machine learning for sensor data analysis in interactive systems: Common pitfalls of pragmatic use and ways to avoid them. *ACM Computing Surveys* 54 (6) Article No.: 134, 1–25. DOI:10.1145/3459666

AGHAEI M, et al. (2022) Review of degradation and failure phenomena in photovoltaic modules. *Renewable and Sustainable Energy Reviews* 159 112160.

LYU D, et al. (2019) Failure modes and mechanisms for rechargeable lithium-based batteries: A state-of-the-art review. *Acta Mechanica* 230 (3).

SHARMA A, et al. (n.d.) Sensor faults detection methods and prevalence in real-world datasets. University of Southern California. https://www.researchgate.net/publication/220443137

GHAZI S, et al. (2014) The effect of weather conditions on the efficiency of PV panels in southeast of UK. *Renewable Energy* 69 50–59. DOI:10.1016/j.renene.2014.03.018

INDEPENDENT COMMUNICATIONS AUTHORITY OF SOUTH AFRICA (n.d.) Database of Type Approved Equipment. https://www.icasa.org.za/legislation-and-regulations/equipment-type-approval/database-of-type-approved-equipment

JIAN C, et al. (2020) A cloud edge-based two-level hybrid scheduling learning model in cloud manufacturing. *International Journal of Production Research* 58 (16) 4836–4850.

CALO SB (2017) Edge computing architecture for applying AI to IoT. In: *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, 11–14 December 2017. IEEE, Piscataway, NJ, USA.