Faculty of Electrical Technology and Engineering

Development of Solar-Powered Fishpond Water Filtration with an IoT-based Monitoring System

MUHAMMAD SYAFIZAL HAIKAL BIN SAMSUDIN

Bachelor of Electrical Engineering Technology with Honours

2025

Development of Solar-Powered Fishpond Water Filtration with an IoT-based Monitoring System

MUHAMMAD SYAFIZAL HAIKAL BIN SAMSUDIN

A project report submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Electrical Engineering Technology with Honours

Faculty of Electrical Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2025

DECLARATION

I declare that this project report entitled “Development of Solar-Powered Fishpond Water Filtration with an IoT-based Monitoring System” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	
Student Name	Muhammad Syafizal Haikal Bin Samsudin
Date	3/1/2025

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Electrical Engineering Technology with Honours.

Signature

Supervisor Name

Date 3/1/2025

DEDICATION

This project is dedicated to our families, whose unwavering support, encouragement, and sacrifices have been the cornerstone of our success.

To our mentors and educators, who have guided us with their knowledge and wisdom, inspiring us to strive for excellence and explore the boundaries of innovation.

To our friends and colleagues, whose camaraderie and collaboration have made this journey both enjoyable and rewarding.

Above all, we dedicate this work to the pursuit of knowledge and the passion for making a meaningful contribution to society through engineering and technology.

ABSTRACT

In this era of globalization, technology is one of the best initiatives to improve the quality of agricultural-based products. The Government of Malaysia suggests that agriculture can be implemented in residential areas without requiring large spaces. However, there are challenges in managing aquaculture and hydroponic systems, particularly in water filtration. Existing systems rely on non-renewable power sources, contributing to environmental issues, and lack advanced monitoring tools for real-time water quality checks. These limitations result in inefficiency, increased costs, environmental harm, and maintenance difficulties. Furthermore, the absence of continuous monitoring prevents these systems from adapting to changing water conditions, risking poor filtration performance and water quality. The objective of this project is to develop hardware for a water filtration system powered by solar energy with an IoT-based monitoring system, to develop software to control, optimize and monitor the operation of the water filtration system, to evaluate the performance of the water filtration system. This filtration system used Esp8266 as a microcontroller to control input and output of the system and Real-Time Firebase as a transmitter to transfer data to mobile application, respectively. Sensor used to detect water condition in fishpond. All of the system powered by solar as a supply include microcontroler and dc water pump. Observation for 4 days in every mode have been made for an automated water filtration system and normal system. It show that energy used based on automated water filtration which is mode 4 more saving compared to normal system in mode 1. This project is easy to used and user-friendly as it improve power consumption while powerd by solar of this water filtration system.

ABSTRAK

Dalam era globalisasi ini, teknologi merupakan salah satu inisiatif terbaik untuk meningkatkan kualiti produk berasaskan pertanian. Kerajaan Malaysia mencadangkan bahawa pertanian boleh dilaksanakan di kawasan perumahan tanpa memerlukan ruang yang besar. Namun, terdapat cabaran dalam menguruskan sistem akuakultur dan hidroponik, terutamanya dalam penapisan air. Sistem sedia ada bergantung kepada sumber tenaga tidak boleh diperbaharui, yang menyumbang kepada isu alam sekitar, serta kekurangan alat pemantauan canggih untuk pemeriksaan kualiti air secara masa nyata. Kelemahan ini menyebabkan ketidakcekapan, peningkatan kos, kesan negatif terhadap alam sekitar, dan kesukaran dalam penyelenggaraan. Selain itu, ketiadaan pemantauan berterusan menghalang sistem ini daripada menyesuaikan diri dengan perubahan keadaan air, sekali gus berisiko menyebabkan prestasi penapisan dan kualiti air yang rendah. Objektif projek ini adalah untuk membangunkan perkakasan bagi sistem penapisan air yang menggunakan tenaga solar serta sistem pemantauan berasaskan IoT, membangunkan perisian untuk mengawal, mengoptimumkan dan memantau operasi sistem penapisan air, serta menilai prestasi sistem penapisan tersebut. Sistem penapisan ini menggunakan ESP8266 sebagai mikropengawal untuk mengawal input dan output sistem serta Firebase Realtime sebagai pemancar untuk menghantar data ke aplikasi mudah alih. Sensor digunakan untuk mengesan keadaan air di dalam kolam ikan. Keseluruhan sistem ini dikuasakan oleh tenaga solar, termasuk mikropengawal dan pam air DC. Pemerhatian selama empat hari telah dilakukan bagi setiap mod untuk sistem penapisan air automatik dan sistem biasa. Hasil kajian menunjukkan bahawa penggunaan tenaga dalam sistem penapisan air automatik (Mod 4) adalah lebih menjimatkan berbanding sistem biasa dalam Mod 1. Projek ini mudah digunakan dan mesra pengguna kerana ia meningkatkan kecekapan penggunaan tenaga, terutama apabila dikuasakan sepenuhnya oleh tenaga solar.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Ts. Dr. Sulaiman bin Sabikan for their precious guidance, words of wisdom and patient throughout this project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) and Ministry of Higher Education for the financial support through RM200.00 which enables me to accomplish the project. Not forgetting my fellow colleague for the willingness of sharing his thoughts and ideas regarding the project.

My highest appreciation goes to my parents, parents in-law, and family members for their love and prayer during the period of my study. An honourable mention also goes to lacturer for all the motivation and understanding. And to Mugilarasan, thanks for your invaluable support, encouragement, and assistance in moments when I needed it most.

Finally, I would like to thank all the staffs at the Faculty of Electrical Engineering Technology, fellow colleagues and classmates, the Faculty members, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Ω	-	Resistor Unit

LIST OF ABBREVIATIONS

V	-	Voltage
A	-	Ampere
ppm	-	Part Per Million

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Project Background

Water filtration systems used in fish farming traditionally depend on conventional power sources, which are predominantly non-renewable and contribute significantly to environmental challenges. These systems are often rudimentary in design, relying on outdated technologies or manual processes that lack precision and fail to account for the dynamic nature of aquatic environments. This inefficiency often results in poor water quality management, which directly affects fish health, productivity, and the overall sustainability of fish farming operations. Furthermore, the reliance on non-renewable energy sources, such as fossil fuels, exacerbates environmental degradation while driving up operational costs, making such systems less viable in the long run.

One major limitation of traditional water filtration systems is their inability to continuously monitor and respond to changes in water quality parameters such as temperature, dissolved oxygen, or total dissolved solids (TDS). Inconsistent monitoring means that water quality can deteriorate rapidly without timely intervention, leading to conditions that are detrimental to aquatic life. For instance, high levels of ammonia or inadequate oxygenation can stress fish, reducing their growth rates and increasing mortality. This lack of adaptability and real-time oversight not only compromises the health of the aquatic ecosystem but also poses a risk to the livelihoods of fish farmers who depend on steady yields.

In addition to these technical shortcomings, the energy-intensive nature of traditional filtration systems often adds a significant financial burden to fish farming operations. Pumps, aerators, and other equipment operate continuously, regardless of actual need, leading to energy wastage and increased wear and tear. Frequent maintenance and repairs further inflate costs, while unscheduled downtime can disrupt operations and negatively impact fish survival rates. This inefficiency underscores the pressing need for systems that are both cost-effective and environmentally sustainable.

The broader environmental implications of traditional systems are also a cause for concern. The dependence on fossil fuels contributes to greenhouse gas emissions, accelerating climate change. Moreover, inefficient water management practices can lead to pollution in surrounding water bodies, harming local ecosystems and reducing biodiversity. As the global demand for fish continues to rise, these issues highlight the unsustainable nature of conventional fish farming practices and the urgent need for innovative solutions that address both environmental and operational challenges.

In recent years, technological advancements have opened new possibilities for addressing these longstanding challenges. The integration of advanced monitoring systems, energy-efficient components, and automated controls into water filtration setups has the potential to revolutionize aquaculture. These technologies aim to optimize water quality management while minimizing energy consumption and environmental impact. For example, sensors capable of measuring critical water parameters can provide continuous data, enabling fish farmers to identify and address potential issues before they escalate. However, the adoption of such technologies is not without its challenges, particularly for small-scale farmers who may lack the technical knowledge or financial resources to implement and maintain these systems.

Another promising development in water filtration is the use of wireless communication technologies, which enable remote monitoring and control. Devices equipped with Wi-Fi or similar connectivity options can transmit real-time data to centralized systems, reducing the need for on-site management. This can be particularly beneficial in large-scale or remote fish farming operations, where manual oversight is often impractical. By enabling seamless communication between various system components, wireless technologies can help streamline operations and improve overall efficiency.

Cloud computing further enhances the potential of modern water filtration systems by providing a platform for storing and analyzing data. Historical data on water quality trends, energy usage, and system performance can be invaluable for making informed decisions about maintenance schedules, resource allocation, and operational adjustments. However, the integration of cloud-based solutions also raises questions about data security, accessibility, and the need for reliable internet connectivity, which may be limited in rural or underdeveloped areas.

Sustainability is another critical consideration in the evolution of water filtration systems. The incorporation of renewable energy sources, such as solar or wind power, offers a pathway to reducing reliance on non-renewable resources. Solar-powered pumps and sensors, for example, can operate independently of the grid, making them ideal for off-grid or resource-constrained settings. By combining renewable energy with smart technologies, these systems can achieve greater energy efficiency, reducing both costs and environmental impact. However, the upfront investment required for renewable energy infrastructure remains a barrier to widespread adoption.

In addition to environmental benefits, the development of user-friendly interfaces has been instrumental in making advanced technologies more accessible. Mobile applications, for instance, allow fish farmers to monitor and control their systems remotely, receiving alerts and updates in real time. This democratization of technology ensures that even those without a technical background can leverage modern tools to improve their operations. Nevertheless, the effective use of such interfaces depends on adequate training and support, highlighting the need for capacity-building initiatives alongside technological innovation.

Overall, the challenges faced by traditional water filtration systems underscore the importance of transitioning to smarter, more sustainable solutions. By addressing inefficiencies in energy use, water quality management, and operational oversight, modern technologies have the potential to transform aquaculture practices. However, the successful implementation of these innovations requires careful consideration of factors such as cost, accessibility, and user education to ensure that they meet the diverse needs of fish farmers around the world.

1.2 Problem Statement

One big issue in fishpond management is the high energy use of traditional filtration systems that run all the time, even when the water is clean. This wastes energy, raises costs, and harms the environment. To fix this, I suggest using a smart filtration system that only turns on when the water gets turbid. This system uses IoT technology and solar power to monitor water quality in real-time and manage filtration efficiently. This solution can save power, reduce costs, and lower the carbon footprint of fishpond operations. It supports SDG 6: Clean Water and Sanitation by keeping water quality high, and SDG 7: Affordable and Clean Energy by using less energy and promoting renewable sources. This benefits fishpond owners, environmentalists, and communities that depend on aquaculture. It also follows regulations for energy conservation and environmental protection, helping to fight climate change and promote sustainable fishpond management.

1.3 Project Objective

The project objectives are follows :

a) To develop hardware for a water filtration system powered by solar energy with an IoT-based monitoring system.

b) To develop algorithm to control, optimize and monitor the operation of the water filtration system.

c) To evaluate the performance of optimization in the water filtration system.

1.4 Scope of Project

The fishpond water filter system project, combined with an IoT-based controller, covers various aspects of hardware and software development, renewable energy use, and aquatic management. Here's a breakdown of what the project includes:

a) Hardware Development

The hardware development for this project involves several critical components designed to ensure optimal functionality and sustainability of the fishpond filtration system. The process begins with designing the fishpond to support efficient water circulation and filtration. Selecting the appropriate water pump is essential, as it must deliver adequate flow rates while consuming minimal energy. To power the system sustainably, solar panels are integrated to supply renewable energy, making the system ideal for off-grid locations. The filtration system incorporates both biological and mechanical methods to maintain water quality. Biological filtration uses beneficial bacteria to break down harmful compounds, while mechanical filtration removes debris and solid particles. Additionally, sensors are embedded to monitor key parameters such as temperature, dissolved oxygen, and TDS levels. The ESP8266 microcontroller is chosen as the central control unit for its IoT capabilities and Wi-Fi connectivity, enabling real-time communication and system automation.

b) Software Development

The software development aspect of the project ensures seamless integration between hardware components and user interfaces. Sensors collect real-time data on water quality, which is processed and transmitted by the ESP8266 microcontroller. IoT-based control allows for automated decision-making, such as adjusting pump operations based on TDS levels or activating aerators when dissolved oxygen drops. A custom Android application, developed using MIT App Inventor, provides a user-friendly interface for monitoring and controlling the system. The app displays real-time sensor data, sends alerts for abnormal conditions, and allows manual control of pumps and other devices. Firebase serves as the cloud platform, storing data securely and enabling remote access. This integration ensures that users can monitor their fishponds from anywhere, improving efficiency and reducing the need for constant on-site supervision.

c) Solar Energy Integration

The use of solar energy in this project is a pivotal feature, making the fishpond filtration system highly sustainable and suitable for remote locations. Solar panels convert sunlight into electrical energy, which powers the pumps, sensors, and control units. By utilizing renewable energy, the system reduces its carbon footprint and minimizes dependence on grid electricity. Advanced battery storage solutions ensure continuous operation even during periods of low sunlight. The integration of solar power also lowers operational costs, making the system economically viable for small-scale and large-scale fish farmers alike. Combining solar energy with IoT technology enables intelligent energy management, where devices operate only when needed, further enhancing efficiency.

d) Testing and Deployment

The project will undergo rigorous testing to ensure reliability and effectiveness. Each hardware component, including the water pump, sensors, and ESP8266, will be individually tested for performance and accuracy. Software functionalities, such as real-time data transmission and app interface responsiveness, will also be thoroughly evaluated. Simulated scenarios will test the system's ability to respond to varying water quality conditions, ensuring that automated controls function as intended. Comprehensive documentation will accompany the development process, detailing system design, configuration, and troubleshooting steps. Strategic deployment will involve setting up the system in real-world fishpond environments, monitoring performance over time, and gathering feedback for further refinement. The ultimate goal is to implement a robust fishpond filtration and IoT monitoring system that enhances efficiency, reduces environmental impact, and provides users with a reliable, easy-to-use solution.

In conclusion, the integration of IoT, cloud computing, Wi-Fi connectivity, and user-friendly platforms like Firebase and MIT App Inventor into water filtration systems represents a significant leap forward. It addresses the inefficiencies and environmental concerns of traditional setups while empowering fish farmers with advanced tools for real-time monitoring and control. This holistic approach ensures that water filtration systems are not only sustainable but also scalable, paving the way for a smarter and greener future in aquaculture.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The rapid development of the Internet of Things (IoT) has transformed many industries by allowing devices to communicate and perform tasks independently. As the number of IoT devices grows, effective power management becomes increasingly important. Power optimization in IoT systems is a major global concern, particularly for increasing device life and ensuring sustainability. This is especially important in remote areas where frequent battery replacement is impractical. Current strategies are centered on utilizing renewable energy sources such as solar and kinetic energy, increasing battery storage capacity, and implementing low-power hardware and efficient communication techniques. Despite these advances, scaling these solutions and seamlessly integrating multiple devices remains difficult, focusing on the need for continued innovation and development.

Microcontrollers are important components in IoT applications, providing a wide range of capabilities from basic control to complex data processing and connectivity. Popular choices include the ESP8266 and ESP32, which are valued for their low cost and built-in Wi-Fi capabilities, making them suitable for a wide range of IoT applications. The Raspberry Pi, Arduino Uno, STM32, and ATtiny85 are also popular, with each offering distinct features such as processing speeds, memory capacities, and connectivity options. These microcontrollers are essential for creating prototypes and finished products, as they enable efficient data collection, processing, and communication in IoT systems.

In terms of renewable energy, solar power has emerged as an important factor in meeting the energy demands of IoT systems, particularly in off-grid and remote areas. Solar photovoltaic (PV) systems are a sustainable and cost-effective way to generate electricity, helping to conserve the environment and ensure energy security. Effective energy storage solutions, such as lithium-ion batteries, help solar power systems by providing a consistent power supply. Integrating IoT technology with solar energy systems improves their functionality by allowing for real-time monitoring and optimization of power usage. This integration is essential for developing flexible and efficient IoT systems that can operate autonomously and sustainably, establishing the way for progress in a variety of industries, including agriculture, environmental monitoring, and smart cities.

2.2 Understanding Current Issue in the Literature

Power optimization in IoT systems is a significant global issue because the number of IoT devices is growing rapidly, making efficient energy use crucial for longer device life and sustainability. This is especially important in remote areas where changing batteries often isn't possible. Current trends focus on using energy from sources like solar and kinetic energy and improving batteries to store more energy. Strategies to reduce power use include using low-power hardware, efficient communication methods, cycling devices between active and sleep states, and processing data close to where it's collected to save energy. However, challenges like making these strategies work on a large scale, ensuring different devices can work together, and using AI to manage power use dynamically still need to be solved. Addressing these challenges is key to the sustainable and efficient use of IoT systems worldwide[1].

2.3 Internet of Things (IoT)

The Internet of Things (IoT) is a system where different devices or objects can communicate over the internet without human input. It's used in many ways to make life easier. When making IoT products, the first step is to create prototypes. These prototypes include things like screens for users, hardware like sensors and processors, software for the server-side, and ways to connect everything together. We often use small computers called microcontroller units (MCUs) or development boards for these prototypes. One popular choice is the ESP8266 wifi chip, which is supported by platforms like NodeMCU and Espruino. The ESP8266 is known for being affordable and powerful, making it great for IoT projects that need to connect to networks or the internet. Studies have shown that it's widely used and effective for these purposes[2].

The ESP8266EX (Figure 2.1), officially known as ESP8266, is a chip that can connect to Wi-Fi and has a processor called Cadence Tensilica L106 32-bit RISC. It also has memory control features like SRAM and ROM built-in [3]. People like to use the ESP8266 chip in data gathering systems because it's cheap, uses little power, and is good at processing information. Plus, it has useful features like built-in Wi-Fi, lots of input/output ports, and is easy to program. They use an open-source platform called ThingSpeak to store and show the data in graphs and dashboards. This keeps the system affordable and easy for anyone to use[4].

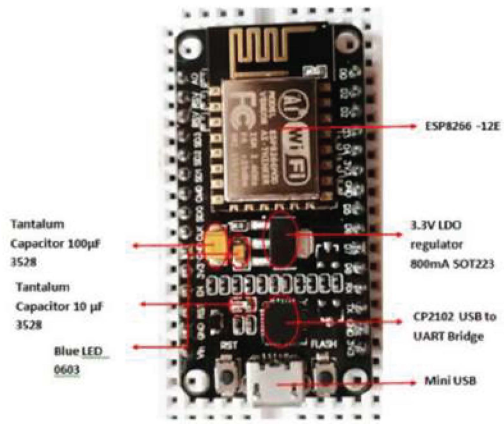


Figure 2.1 Esp8266 module

The ESP32 in Figure 2.2 is a type of microcontroller made by Espressif Systems, following on from their earlier ESP8266 model. It's designed to work with the Arduino programming environment. This microcontroller comes with built-in Wi-Fi and connects to Bluetooth Low Energy (BLE) using a separate chip. Because of this, it's very powerful and a great choice for making IoT applications. The name "ESP32" comes from "Espressif32," and it's a development board made by Espressif Systems. The ESP32 has a 32-bit processor and supports wireless networking through Wi-Fi and Bluetooth Low Energy (BLE) using the 802.11 b/g/n Wi-Fi protocol at 2.4 GHz, along with Bluetooth v4.2 technology[5].

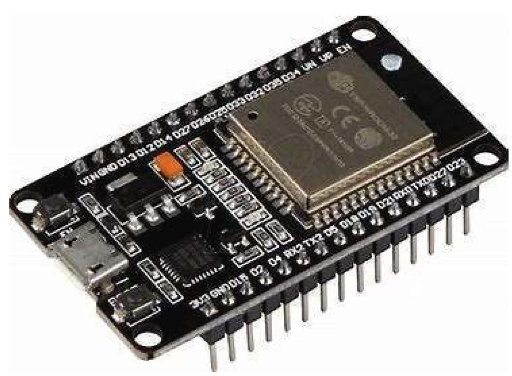


Figure 2.2 Esp32 Module

2.4 Microcontroller For IoT Applications

A microcontroller is a tiny and affordable computer that's made to do particular jobs in embedded systems, such as showing info on a microwave or getting signals from a remote. Usually, it includes a processor, memory (like RAM, ROM, and EPROM), serial ports, and extras like timers and counters. These are made for specific tasks in different kinds of machines[6]. Figure 2.3 show programme inside the microcontroller.

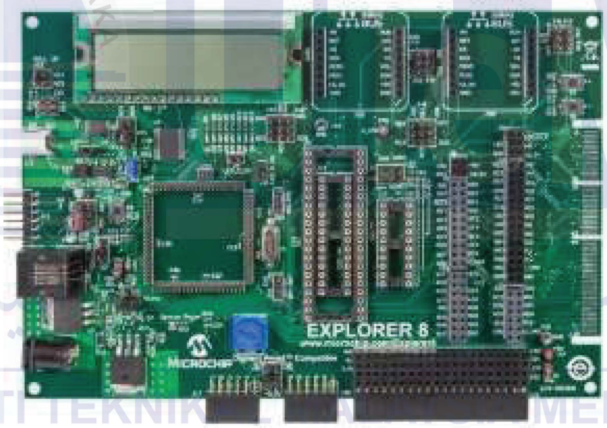


Figure 2.3 Example of Universal Microcontroller

In mechatronics, computers are used to create programs for devices like microcontroller-based programmable controllers, which are also known as programmable logic controllers (PLCs). When we pair sensors with these controllers to get input information and devices controlled by the controller (called 'actuators'), we form a simple mechatronic system. The program, typically made on a computer, is then transferred to the programmable controller[7].

2.4.1 Rasberry Pi Module

The Raspberry Pi in Figure 2.4 is a small and powerful mini-computer, roughly the size of a credit or debit card. It was created by the Raspberry Pi Foundation in the United Kingdom with the goal of inspiring and empowering learners to be more creative and effective. Unlike many video games, the Raspberry Pi is a versatile microcontroller that's not very expensive. With it, users can learn programming, including Python coding, build robots, and work on various creative projects. It can do all the things a regular computer can do, like browsing the internet, watching movies, playing games, and listening to music[8].

Figure 2.4 Respberry pi module

2.4.2 Arduino Uno Module

The Arduino microcontroller is an open-source tool that's easy to program and can be updated whenever needed. It was first introduced in 2005. Originally, Arduino was made for professionals and students to create devices that interact with their surroundings using sensors. The Arduino platform has two main parts: hardware and software. The hardware includes the Arduino development board, while the software used for coding is called the Arduino IDE (Integrated Development Environment). It's powered by 8-bit Atmel AVR microcontrollers made by Atmel, or 32-bit Atmel ARM microcontrollers. These microcontrollers can be programmed using the C or C++ language in the Arduino IDE [9]. Figure 2.5 show example Arduino microcontroller.

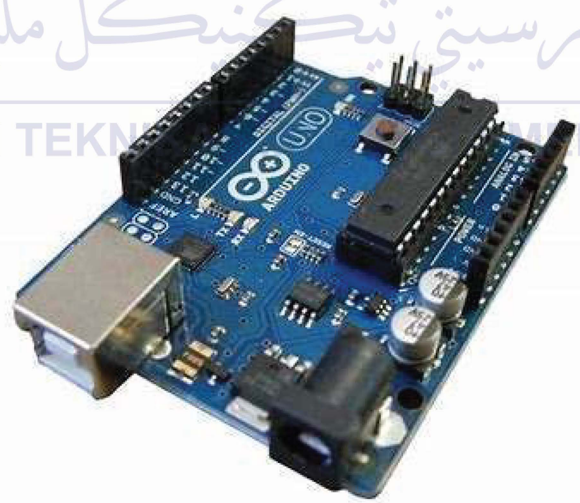


Figure 2.5 Arduino uno module

2.4.3 STM32 Module

The STM32 series in Figure 2.6 uses the ARM Cortex-M3 core, which is made for embedded applications needing good performance, affordability, and low power usage[10]. These microcontrollers are built on the ARM Cortex-M3 architecture. They bring together strong performance, real-time capabilities, digital signal processing, low power usage, and the ability to work on low voltages. In this IoT module, the specific microcontroller used is the STM32F103VET6, which is positioned as a mid-level chip within the STM32 series[11]. The STM32F446RE microcontroller, chosen by students to learn microcontroller programming, is a high-performance chip made by STMicroelectronics. It features an Arm Cortex-M4 core built specifically for digital signal processing tasks. This microcontroller was picked for the educational program because of its strong performance, DSP instruction set, and multiple ADC functions, allowing for faster AD conversion[12].

Figure 2.6 STM32 module

2.4.4 Attiny Microchip

The ATtiny85 is an 8-bit microcontroller designed for low power usage and constructed using CMOS logic. It follows the AVR enhanced RISC architecture and can perform approximately 1 million instructions per second. Despite its low power consumption, it operates at a high speed. The ATtiny85 offers 8K bytes of flash memory, EEPROM ranging from 128 to 512 bytes, 256 bytes of SRAM, six GPIO lines, 32 general-purpose registers, an 8-bit timer/counter with compare modes, both External and Internal Interrupts, a high-speed 8-bit timer, a 10-bit ADC, a watchdog timer with an internal clock oscillator, and three power-saving modes. It is compatible with the Arduino IDE environment, making it easy to work with[13]. Figure 2.7 show example of Attiny microchip.

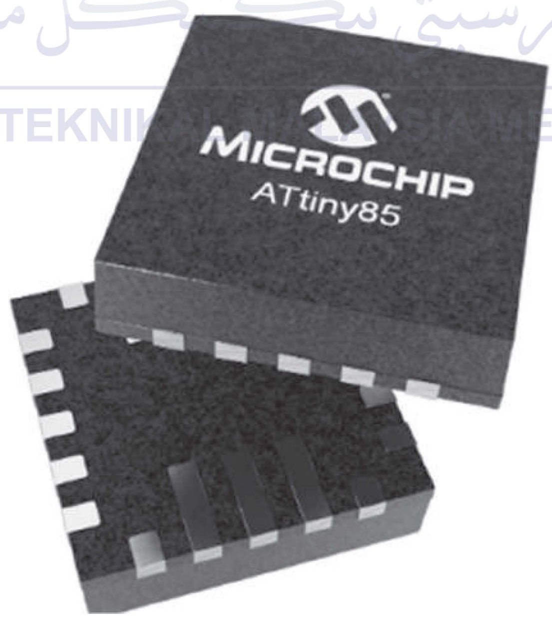


Figure 2.7 ATtiny microchip

2.4.5 Summaries of Microcontroller

Microcontrollers come in different prices and features. Cheaper ones are good for simple projects, while more expensive ones can handle more complex tasks. Some have built-in Wi -Fi and Bluetooth, making them great for smart home and IoT projects. Their processing speeds range from slow to very fast, depending on what you need. They have different numbers of input/output pins, which affects how many other devices you can connect. Power usage varies, with some using very little power and others more. Memory size also differs, assessing their data handling capacity. Some are easy to use with lots of online help, while others need more technical know-how. They can be used for anything from basic hobby projects to advanced industrial applications, with various tools available to help you program and use them. IoT devices are low-cost and energy-efficient. However, adding more transistors for complex hardware security features could raise the cost of the SoC and increase power consumption, making them unsuitable for many applications where thermal design power (TDP) is a concern[14]. Table 2.1 illustrate different between specification of microcontroller.

Table 2.1Comparison between specification of microcontroller .

Feature	ESP8266	ESP32	Rasberry pi	Arduino uno	STM32	ATtiny85
Processor	Tensilica Xtensa L106	Tensilica Xtensa LX6 (dual-core)	Broadcom BCM2711 (quad-core Cortex-A72)	ATmega328P	ARM Cortex-M4	AVR ATtiny85
Clock Speed	80 MHz	160/240 MHz	1.5 GHz	16 MHz	72-120 MHz	1-20 MHz
RAM	32 KB	520 KB SRAM	2-8 GB	2 KB SRAM	20-512 KB SRAM	512 B SRAM
Flash Memory	4 MB (external)	4 MB (external)	microSD card	32 KB	64-512 KB	8 KB
Wi-Fi	Yes, 2.4 GHz	Yes, 2.4/5 GHz	Yes (via external dongle)	No	Optional (STM32 Wi-Fi module)	No
Bluetooth	No	Yes, 4.2 and BLE	Yes (via external dongle)	No	Optional (STM32 Bluetooth module)	No
Operating Voltage	3.3V	3.3V	5V	5V	3.3V	2.7-5.5V
ADC Resolutio n	10-bit	12-bit	Varies (via external ADC)	10-bit	12-bit	10-bit

2.5 Solar System

Solar energy technology has become increasingly important recently. It's being seen as a viable option for generating electricity in countries like India that need more energy. Solar power generation helps with environmental issues and makes energy supplies more secure, while also reducing the carbon emissions produced by coal power plants[15]. This has several advantages both economically and environmentally. Generating electricity with solar panels can be cheaper and more sustainable over many years compared to other methods of generating electricity[16]. Figure 2.8 show example of solar system.

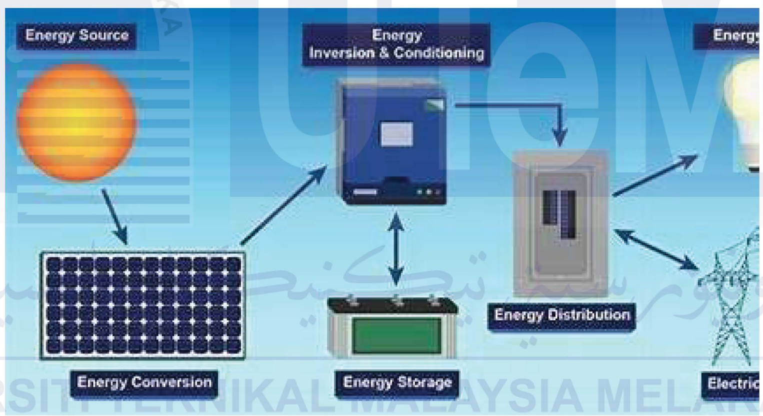


Figure 2.8 Example connection solar system

2.5.1 Solar Energy

A solar cell is a tool that captures sunlight and turns it into electricity. Sunlight has energy in tiny particles called photons. When these photons hit a solar cell, they make electrons move around, creating an electric current. This process is called a photovoltaic event or photoelectric effect. Solar cells can do this because they're made of a special material called a semiconductor, which usually contains silicon. This silicon has two layers that are sensitive to light: a positive layer (P-type) and a negative layer (N-type)[17]. Figure 2.9 show solar panel taht contain solar cell inside.

Figure 2.9 Solar panel

2.5.2 Energy Storage

Battery storage systems have been created along with power converters, control programs, and controllers to test how hybrid microgrids work. An energy management system keeps the power balanced, managing changes in renewable energy generation and the amount of power needed for different loads[18]. Using energy storage technologies like DC batteries with a PV system can greatly improve energy use and help the PV system run smoothly. Lithium-ion batteries (LIBs) have several benefits over lead-acid batteries, like being able to charge quickly, having high density, lasting a long time, and being smaller and lighter. This makes them a good choice for short-term energy storage in many situations. Batteries store electricity made by solar panels that isn't used right away by whatever's using the power. This stored energy can be used when there's not much sunlight or during the night[17]. Figure 2.10 show example type battery can used for storage.

Figure 2.10 Example of battery in energy storage

2.6 Solar Power Optimization

Solar photovoltaic energy is the world's third-largest renewable energy source, following hydro and wind power. Solar panels directly convert sunlight into electricity. In a stand alone PV system, it's crucial to maintain a steady power supply to meet energy demands. One suggested method involves analyzing energy needs hourly and determining the right sizes for battery banks and PV arrays to meet these needs throughout the day. Because solar power output changes, the battery storage must be enough to provide consistent power, no matter how many times the battery is charged or discharged. This backup ensures a continuous power supply after sunset until solar energy is available again for recharging[19]. The main goal of organizing this structure is to enhance operational and economic performance, particularly when dealing with disturbances from intermittent renewable resources[20].

2.7 Fishpond Control System

Managing a fishpond includes keeping the water clean, making sure it circulates well, providing the correct amount of food for fish, controlling how many fish are in the pond, stopping and treating diseases, taking care of where the fish live, thinking about how the water moves in and out and gets cleaned, checking on things often, and being careful about how it affects the environment. These actions all work together to keep the fish healthy and make sure they can keep living there in a sustainable way. Fish management systems play a crucial role in aquaculture. One key aspect of fish management is ensuring water quality, which encompasses factors like temperature, pH levels, oxygen content, and feeding practices. Currently, the monitoring of water quality and feeding of fish is primarily carried out manually[21].

2.7.1 Filtering System

Biological water filtration is when harmful substances are taken out of water using living things. Having the right size biological filter is really important in a system where water is reused for fish farming. In aquaculture recycling systems, the equipment for biological water filtration helps make the water from fish ponds safe to use again by getting rid of bad stuff in it[22]. Figure 2.11 show simple stage in filtration of fishpond. Filtration is crucial for purifying and decontaminating water and air, both essential for life. Growing awareness of the connection between air quality and human health has increased the demand for better personal protection against airborne pollutants and harmful microbes. Electrospun nanofibres (NFs) offer unique advantages as an active protective layer in facemasks. Compared to commonly used filters with melt-blown (MB) fibres, electrospun NFs provide superior protection against airborne particles, bacteria, and viruses[23].

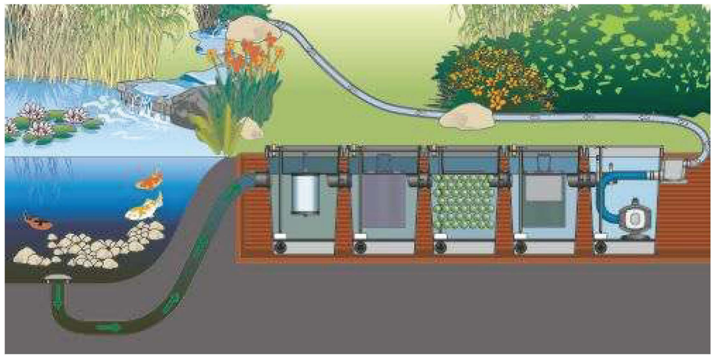


Figure 2.11 Filtration sample

2.7.2 Water Quality

Understanding water quality can be a complex topic. It's influenced by various factors, and we use several measurements together to assess it. These measurements include turbidity, total dissolved solids, and temperature [24]. Aquaculture is gaining importance in Malaysia as a key driver of economic growth. However, maintaining a healthy environment for fish production requires careful attention. Water quality plays a crucial role in this, influenced by factors like dissolved oxygen, ammonia levels, pH, temperature, salt content, nitrates, carbonates, and more. Typically, fish farmers or researchers manually test these parameters in fishponds or laboratories to ensure a stable and healthy production environment [25]. Figure 2.12 sample of water taken to check the quality.

Figure 2.12 Water quality sample

2.7.3 Household fishpond

A household fishpond is a small-scale aquatic system designed for raising fish at home, primarily for consumption, recreation, or ornamental purposes. Typically constructed using materials like concrete, plastic liners, or prefabricated tubs, these ponds require proper design to ensure adequate water depth, aeration, and filtration. A balanced ecosystem within the pond, including plants and microorganisms, helps maintain water quality and supports fish health. Popular fish species for household ponds include tilapia, koi, goldfish, and catfish, depending on the purpose. Regular maintenance, such as feeding, cleaning, and monitoring water parameters (e.g., pH, temperature, and dissolved oxygen), is crucial to keep the pond thriving and the fish healthy. Fishponds can also serve as aesthetic additions to a home, offering relaxation and promoting sustainable food practices.

2.8 Related Previous Work

To enhance my project, I have reviewed similar projects to understand their approaches and outcomes. This comparison helps me gain insights into what has worked well and what can be improved. By summarizing these previous projects, I can identify strategies and techniques that could be beneficial for my own project. This process allows me to build upon existing knowledge and make informed decisions to create a more effective and successful project.

2.8.1 Project 1 : A Self Monitoring and Analyzing System for Solar Power Station using IoT and Data Mining Algorithms

Renewable energy sources are receiving significant research attention due to their economic and sustainable features. Solar power stations, in particular, are seen as viable renewable energy systems for various locations due to their lower installation and maintenance costs compared to conventional systems, despite requiring less space. While small generating stations can be accommodated on open terraces, large-scale power stations need acres of land, posing a challenge for maintenance. Leveraging IoT and data mining techniques, the proposed algorithm aims to help human operators detect regular power generation patterns and identify faults or defective areas in solar power systems swiftly. This proactive approach enables prompt action for fault rectification, ultimately improving the efficiency of the generating station[26]. Figure 2.13 show solar connection while measure voltage, current in this project.

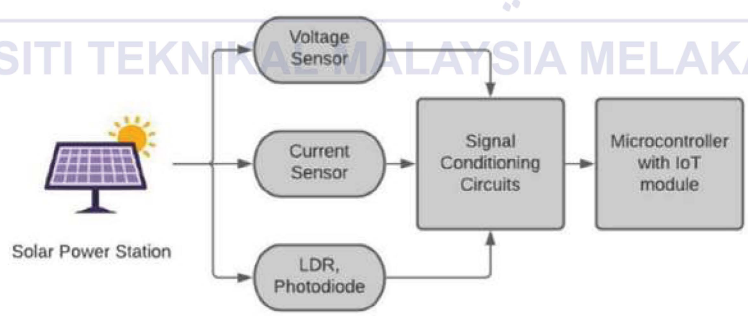


Figure 2.13 Solar flow chart

2.8.2 Project 2 : Development of IoT Based Fish Monitoring System for Aquaculture

This paper discusses the creation of an IoT-based fish monitoring system for aquaculture, aimed at improving fish production and maintaining water quality. The system uses IoT devices and an Android app to monitor pH levels, temperature, dissolved oxygen, and ammonia levels crucial for fish health. Aquaculture is vital for economy but faces challenges like water pollution and lack of water quality expertise among farmers. This system offers a cost-effective solution, using sensors and testing kits to provide real-time data via a user-friendly mobile app. The system's successful implementation demonstrates its potential to support fish farming sustainability and economic growth[27]. Figure 2.14 show equipment used in the project.

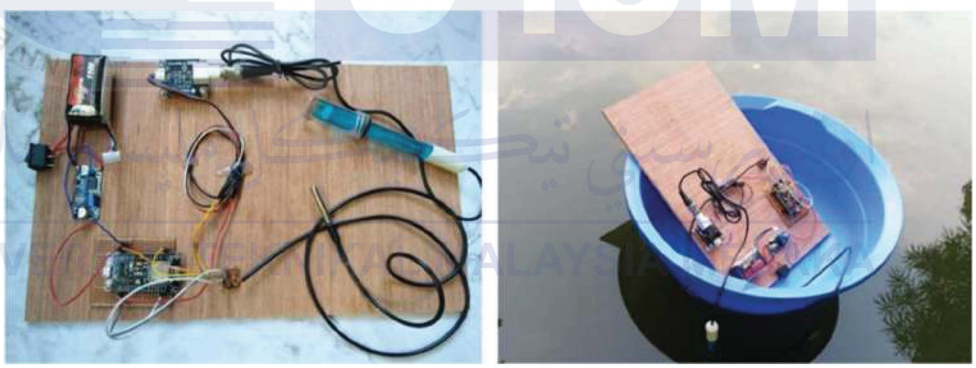


Figure 2.14 Equipment used.

2.8.3 Project 3 : IoT-Based Smart Aquaponics System Using Arduino Uno

This project suggests an aquaponics system that saves water by circulating it and using fish waste to produce nutrients for plants in a tank. The system uses sensors, actuators, and microcontrollers to watch and manage water quality. Sensors collect data transmitted through Wi-Fi to an IoT cloud platform called Thingspeak, while actuators fix any issues found by the sensors. The experiments showed that water quality and circulation were good, as seen in linear regression analysis and R-squared plots that showed a strong link between time and plant growth and fish weight. This system cuts down on water waste and boosts aquaponics yields[28]. Figure 2.15 show the hardware develop in the project with full setup.

Figure 2.15 Hardware project.

2.8.4 Project 4 : Development of Monitoring System Based on Internet of Things (IoT) for Freshwater Prawn Farming

This project focuses on integrating IoT technology into freshwater prawn farming to automate water quality monitoring, reduce labor costs, and improve operational efficiency. By utilizing sensors, microcontrollers, and communication modules, the system aims to provide real-time data on critical parameters like temperature, pH levels, and dissolved oxygen. The use of low-cost devices such as Arduino and ESP8266 Wi-Fi modules enables cost-effective solutions while supporting proactive decision-making for enhanced productivity and environmental sustainability. Overall, the project aims to contribute to smart agriculture practices in aquaculture by developing an efficient and tailored IoT-based monitoring system for freshwater prawn farming[29]. Figure 2.16 show full prototype of this project.

Figure 2.16 Project prototype.

2.8.5 Project 5 : Water IoT Monitoring System for Aquaponics Health and Fishery Applications

This paper outlines the work done to create and test a complete system for monitoring aquaponics health, focusing on the fishing industry. We found that existing monitoring systems didn't meet the needs of monitoring the entire aquaponics life cycle. The new system advances communication technology by using 5G for communication and LoRa for long-range communication between nodes and gateways, with LTE-M/NB-IoT for internet connectivity. For aquaculture, we've improved the sensing layer with more sensors to enhance performance in the fishing industry. The sensing and compensation processes take between 0.9 s to 1.5 s for digital sensors and 0.3 s for analog ones, with the whole sensing period averaging around 7.5 s with nine sensors. Information packets take about 463 ms to reach the CORE from the sensing node. We've also redesigned the hardware architecture to improve energy efficiency and reduce power consumption, extending battery life by around 70% with the inclusion of a switching IC for sensors connected to the I2C line[30]. Figure 2.17 show aquaponics cycles in this report.

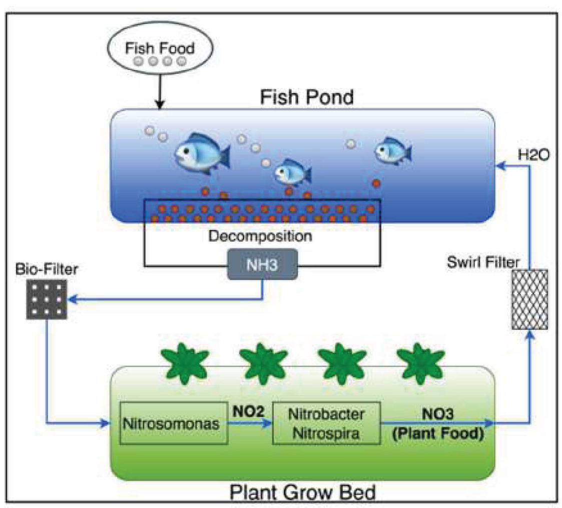


Figure 2.17 Aquaponics cycle

2.9 Summary

Table 2.2 and Table 2.3 show comparing different projects shows how they use software and hardware in various ways. Some projects use simple sensors and manual controls, while others have advanced features and better connectivity. Projects with Wi-Fi capabilities are good for IoT applications, and those with solar power systems are more energy-efficient. Using strong IoT integration helps with better remote control and monitoring, while advanced features handle complex tasks well. Solar-powered projects are more sustainable and cost-effective in the long run. These insights can help choose the best components to balance connectivity, functionality, and energy efficiency for better project results.

Table 2.2 This table show comparison between related previous project.

Previous project	**Iot Microcontroller**	**Cloude Based**	**Solar energy**	**IoT**
Project 1, 2021 By Shakya S	GSM module	6LoWPAN	**Yes**: Utilized solar energy for powering the system.	**Yes**: Enabled IoT communication over wide areas using 6LoWPAN,
Project 2, October 2021. By Tamim ABegum HShachcho SKhan MYeboah-Akowuah BMasud MAl-Amri J	Esp8266	MIT and Google Firebaase	**No**: Did not incorporate solar energy, making the system dependent on traditional power sources.	**Yes**: Leveraged ESP8266 for IoT, allowing real-time data communication with Firebase and seamless app integration via MIT.
Project 3, October 2021. By Ntulo MOwolawi PMapayi TMalele VAiyetoro GOjo J	Arduino uno	Thingspeak	**No**: Absence of solar energy integration limited its potential for off-grid applications.	**Yes**: Used Thingspeak for IoT data handling,
Project 4, 01 October 2022. By Rami M, A Majid H, Al-Fadhali N	Arduino mega 2560 and Esp8266	Arduino Ide and Thingspeak	**No**: Lacked solar power, restricting its deployment in energy-sensitive or remote areas.	**Yes**: Combined Arduino Mega's computational power with ESP8266's Wi-Fi
Project 5, 2022 by Alselek MAlcaraz-Calero JSegura-Garcia JWang Q	Raspberry pi	LoRaWan, Grafana	**No**: Did not employ solar energy, which could have enhanced its operational flexibility in remote setups.	**Yes**: Implemented LoRaWAN for long-range IoT communication and Grafana

Table 2.3Comparison In Advantages and Disadvantages

Previous Project	**Advantages**	**Disadvantages**
Project 1	- Wide area connectivity using GSM. - Energy-efficient with 6LoWPAN.	- Low data speeds and higher latency compared to Wi-Fi. - Limited scalability for real-time apps.
Project 2	- Real-time data handling with Firebase. - Easy mobile app integration using MIT.	- Relies on stable internet connectivity. - Limited offline operation capability.
Project 3	- Simple and cost-effective. - Supports basic IoT applications.	- Limited computational power. - Requires external Wi-Fi shield for connectivity.
Project 4	Increased computational power with Arduino Mega. - Reliable data handling with ESP8266. .	- Higher complexity in programming and integration. - Limited portability
Project 5	- Long-range communication with LoRaWAN. - Advanced data visualization with Grafana.	- Relatively high cost. - Higher power consumption compared to microcontrollers.

To summarize, the exponential growth of the Internet of Things (IoT) presents both opportunities and challenges, particularly in terms of power optimization. As the number of IoT devices grows, ensuring efficient energy usage is critical for increasing device longevity and achieving sustainability. The integration of renewable energy sources, such as solar power, and advancements in battery technology are critical steps toward achieving this goal. However, the complexities of scaling these solutions and ensuring seamless interoperability across diverse devices highlight the need for ongoing innovation and research.

Microcontrollers such as the ESP8266, ESP32, Raspberry Pi, Arduino Uno, STM32, and ATtiny85 are central to IoT applications, providing the processing capacity and connectivity required. These devices enable the creation of prototypes and finished products capable of efficiently collecting, processing, and transmitting data. Their diverse features and capabilities make them suitable for a wide range of applications, from basic control tasks to advanced data analysis and communication, emphasizing their critical role in the IoT ecosystem.

Moreover, combining solar energy with IoT technology provides an environmentally friendly way to satisfy these systems' energy needs, especially in isolated and off-grid locations. When paired with effective energy storage technologies, solar photovoltaic systems guarantee a consistent power source and improve the general performance of Internet of Things applications. In addition to promoting energy security and environmental preservation, this interaction between renewable energy and the Internet of Things also encourages the creation of robust, self-sufficient systems in a variety of industries. Sustained progress in this field is crucial to propel innovation and attain enduring expansion within the Internet of Things terrain.

.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This project focuses on developing a sustainable and efficient water filtration system for fishponds by integrating solar energy and IoT-based monitoring technologies. The three stages of the technique are as follows: first, a thorough literature review is conducted to obtain information about current technologies involving fishpond water filtration systems, Internet of Things-based monitoring, and solar energy integration. The following project goals—building hardware components, establishing user-friendly software for system control and optimization, and assessing system performance to guarantee efficient operation—are informed by this fundamental research.

During the hardware setup phase, a solar-powered system that controls a 12V DC pump for water filtration is designed using an ESP8266 microcontroller. A charge controller allows batteries to hold the DC electricity produced by solar panels, providing a steady power source even when there is not enough sunlight. In order to enable real-time monitoring and control, the microcontroller exchanges data with a cloud service and processes data from many sensors, including turbidity and TDS sensors. With this configuration, the system is guaranteed to be both energy-efficient and able to be operated remotely via a mobile application.

MIT App Inventor and Google Firebase are used in the software setup phase to generate a simple user interface and real-time data management system. The application shows sensor data, allows user interaction with a simple drag-and-drop interface, and enables remote monitoring and management of the water filtration system. Through the employment of these tools, the project hopes to improve the system's accessibility and usability and enable users to effectively monitor fishpond water quality from anywhere. To create a water filtration system that is dependable and sustainable, this integrated method integrates cutting-edge hardware and software technologies.

3.2 IoT Project Development Flow

The process in Figure 3.1 is developing a solution using a microcontroller begins with identifying a specific problem or challenge to address. Once the problem is clearly understood, brainstorming ideas and potential solutions leads to creative concepts for tackling the challenge. An algorithm is then developed, which serves as a step-by-step procedure outlining how the microcontroller will achieve the desired outcome. To visualize and refine this logic, a flowchart is created, using symbols like boxes, diamonds, and arrows to represent the sequence of steps, decisions, and loops involved. With the logic clearly mapped out, the next step is coding, where the algorithm and flowchart are translated into a programming language suitable for the microcontroller, such as C or C++. This code dictates the microcontroller's actions and is finally uploaded onto the device, enabling it to execute the instructions and implement the solution effectively.

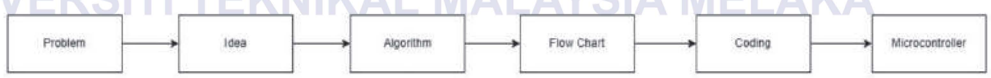


Figure 3.1 Flowchart of Development Flow Process

3.3 Overal Project Methodology Flowchart

The project as in begins with a literature review to gather information and understand existing technologies related to fishpond water filtration systems, IoT-based monitoring, and solar energy integration. The objectives of the project are then outlined: Objective 1 is to create the hardware components of the filtration system powered by solar energy and including an IoT-based monitoring system. Objective 2 involves developing software to control, optimize, and monitor the system, utilizing MIT App Inventor and Firebase for user-friendly interface and real-time data management. Objective 3 focuses on evaluating the system's performance to ensure it operates effectively and efficiently. This structured approach aims to build a sustainable and effective water filtration system for fishponds. Flowchart of Project methodology show as Figure 3.2.

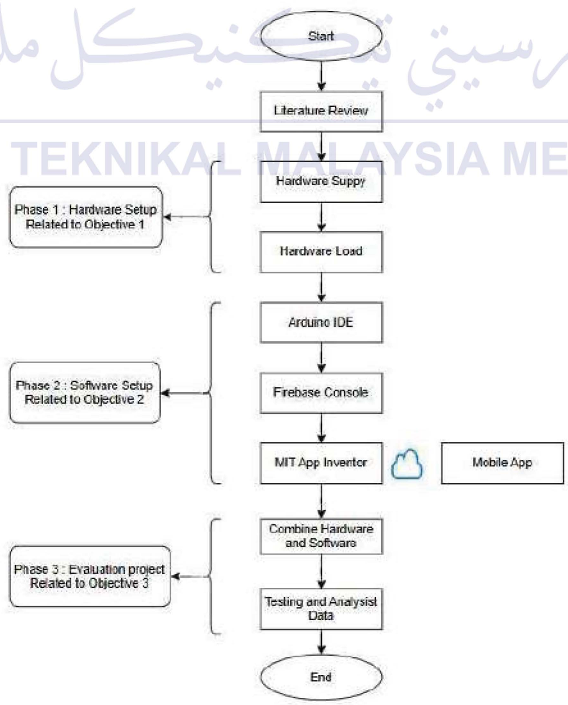


Figure 3.2 Flowchart of Project Methodology

3.4 Phase 1 : Hardware Setup

Direct current (DC) power is generated by solar panels upon receiving sunlight, the quantity of DC electricity generated is dependent upon the size, efficiency, and strength of the solar panel. The charge controller receives this DC electricity and uses it to control the electricity flow and prevent overcharging of the batteries. The charge controller lets the battery bank get its maximum current flow. After that, it reduces the current, only a small current keeps the battery fully charged. When the solar panels aren't producing energy, the DC electricity is kept in the battery bank for later use. Batteries can be used directly to power DC appliances. Figure 3.3 Power Generation and Storage Part.

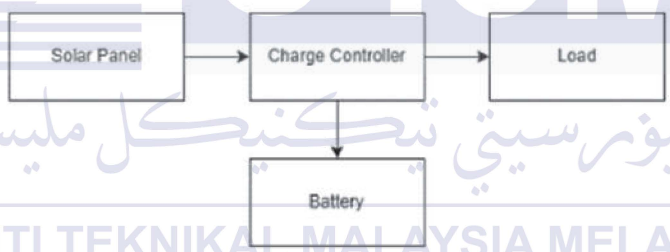


Figure 3.3 Power Generation and Storage part

This system shows the IoT system that runs on solar power and regulates a 12V DC pump in a water filtration system. The power source for the system is solar-powered. The core component of the system is an ESP8266 microcontroller, which processes information from linked sensors to monitor water turbidity or other factors. To handle water filtration, the microcontroller regulates a relay, an electrically powered switch, which in turn turns on or off a 12V DC pump. Furthermore, the microcontroller establishes a Wi -Fi connection with a cloud service, allowing for remote control and monitoring via a mobile application. This configuration provides effective and responsive system performance by enabling users to monitor and control the water filtration process in real-time from any place. Operational Control Part show as in Figure 3.4.

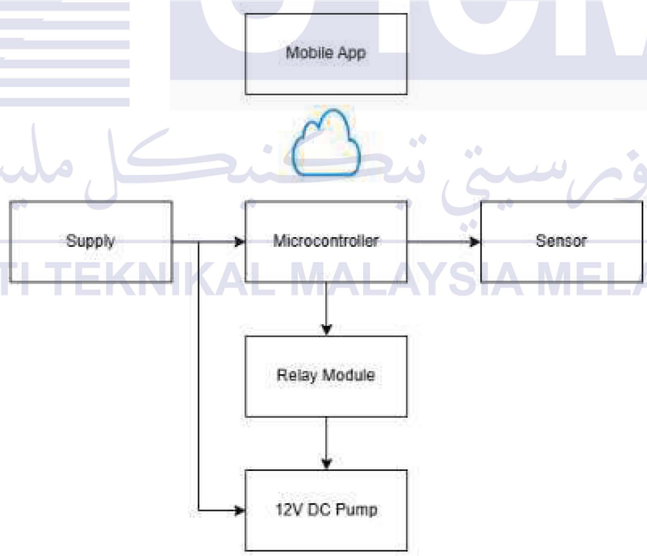


Figure 3.4 Operational Control Part

3.4.1 Experimental Equipmenet Setup

This water pump in Figure 3.5 works with a DC voltage between 6 and 12 volts and uses 400 milliamps of current, giving it a power output of about 4 to 5 watts. It can pump up to 500 liters of water per hour, which is great for keeping water moving steadily. These features make the pump a good choice for small to medium-sized projects like water filtration systems, and aquariums where a reliable flow of water is needed.

Figure 3.5 DC water pump 500L/H

This solar panel in Figure 3.6, which is of the polycrystalline type and measures 145mm x 145mm x 3mm (thickness), outputs 12V at 250mA, providing a total power output of 3W. It performs best when exposed to full sunlight, efficiently converting solar energy into electrical power. With its compact size and decent power generation, this panel is ideal for applications like small electronic devices, solar chargers, or low-power systems where space and weight are significant factors.

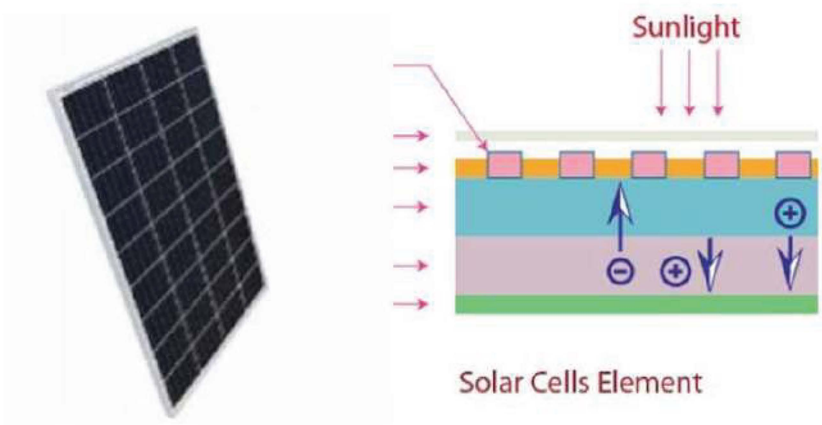


Figure 3.6 Solar panel 3w

The relay module in Figure 3.7 being used has a maximum load capacity for the normally open interface of AC 250V/10A and DC 30V/10A, with a trigger current of 5mA. It features a power indicator (green) and a relay status indicator (red). Connected to an ESP8266, this relay module is used to control the activation of a 12V DC pump. The green power indicator shows when the module is powered, while the red indicator changes based on the relay's status, providing a visual cue for monitoring the pump's operation. This setup allows for convenient and efficient control of the pump using the ESP8266 microcontroller.

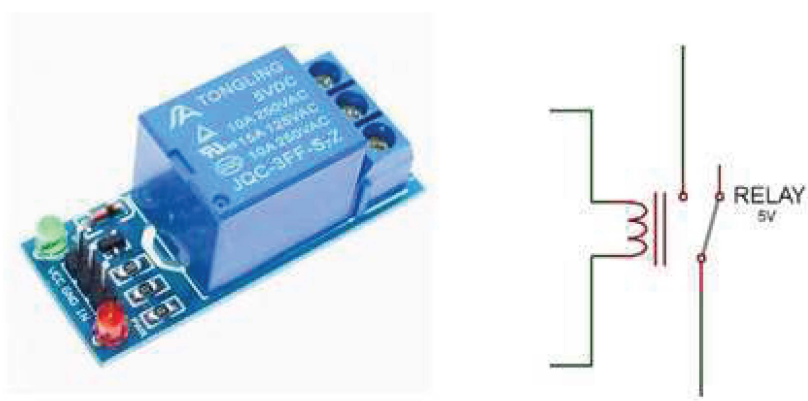


Figure 3.7 Relay module

The Figure 3.8 show solar charge controller being used in this project has several important functions to ensure efficient and safe operation. In addition to its adaptive voltage of 12V and current handling capacity of 10A, it features an LCD display for easy monitoring and control. The controller also includes a dual USB charging interface, allowing USB-powered Esp8266 directly from the solar system. It provides power and voltage display functionalities. Moreover, the controller manages the charging and discharging processes of the connected lithium-ion battery, optimizing energy utilization and prolonging battery life. When connected to a load which is dc pump.and energy powered by 3W solar panel.

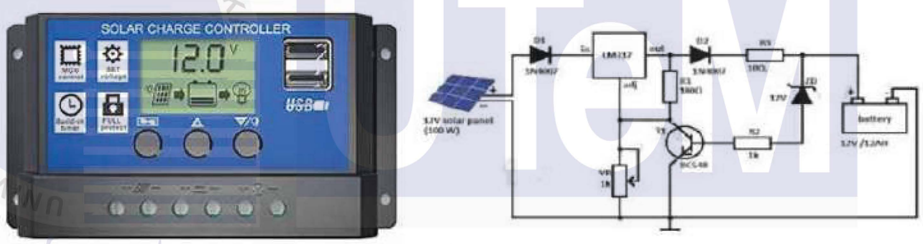


Figure 3.8 Solar charge controller

The TDS value measures how many milligrams of dissolved solids are in one liter of water. A higher TDS value usually indicates more dissolved solids, which can make the water appear murkier or more turbid in a fishpond. Therefore, the TDS value is an important indicator of water clarity or turbidity, commonly checked in fishponds. Gravity Analog TDS Sensors, which are affordable consumer-grade devices widely available in the market, were used for this project. Compared to an industrial-grade TDS sensor, the Gravity Analog TDS sensor as in Figure 3.9is significantly more budget-friendly, making it a practical choice for monitoring turbidity in fishponds.

Figure 3.9 TDS Sensor module

The ESP8266 microcontroller as Figure 2.1 plays a key role in this setup, connecting various components to streamline the fish pond system. It interfaces with the turbidity sensor to monitor water quality, activating a relay to control the DC pump as needed for filtration. Powered by solar energy, the ESP8266 ensures sustainable operation. Its Wi-Fi capability enables real-time data transmission to a database, facilitating remote monitoring via a dedicated mobile app. This integrated approach leverages technology to maintain water clarity, automate pump operations, and provide accessible monitoring for efficient fish pond management.

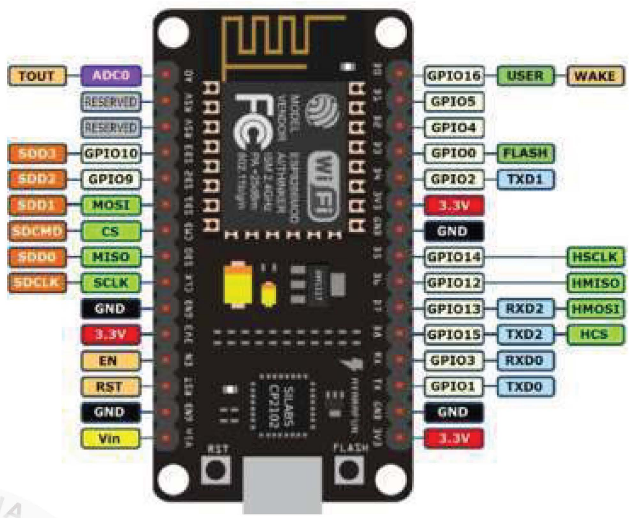


Figure 3.10 Esp8266 module

For this project, sealed lead acid (SLA) battery have selected as Figure 3.11 with a standard voltage of 12 volts and a capacity of 4500 milliampere-hours (mAh). This type of battery is well-known for its reliability and durability, making it a suitable choice for various applications. With a capacity of 4500mAh, this battery can store a significant amount of energy, providing ample power to sustain the project's operations over extended periods. Sealed lead acid batteries are also maintenance-free, meaning they do not require regular topping up with water, unlike their traditional lead-acid counterparts. Furthermore, their sealed design minimizes the risk of leaks and makes them safer to handle and use. This makes the SLA battery a practical and efficient power source for projects that require stable and long-lasting power.

Figure 3.11 Sealed lead battery

3.4.2 Hardware procedure Functionality Checking

Table 3.1 Indicator in Hardware Part

Step	Hardware Part	Indicator Functioned	Indicator Failed
1	Esp8266 and Wi-fi	Connected / LED blinking with interval 2 second	LED blinking with interval 0.5 second
2	Solar Power	Charge controllers energize / Display Solar and Bulb	No display on Charge controller
3	Input / Sensor	Show reading / Blue Light LED on module on	Illogical / LED on module off
4	Output / Actuator	Pump on / sound of water	Pump off

The process of checking in Table 3.1 and the functionality of hardware begins with a systematic verification of all components and their interactions within the system. Step 1 is to power on the system, ensuring that all components receive the necessary voltage and current. This includes observing initial LED indicators or displays to confirm that everything is operating as expected. If any indicators fail to show, it is essential to check the power supply and connections to the components.

Step 2 involves verifying the Wi-Fi and communication module, such as the ESP8266, to ensure it functions correctly. This includes confirming that it connects to the network and that the expected LED behavior, such as a solid or blinking light, is present, indicating proper connectivity. Any issues with Wi-Fi connectivity may require checking the network settings or hardware configuration.

Step 3 focuses on inspecting the solar power system, which includes the solar panels, charge controllers, and connected devices. The charge controller's display or indicators should reflect the correct operational status, ensuring that the system is properly charging and regulating power. If the charge controller shows errors or no readings, further inspection of the solar panel and wiring may be required.

Step 4 involves sensor input validation, where sensor modules are tested to ensure they provide accurate readings on the system interface. The LED indicators on the sensors should show proper functionality, signaling that the sensors are correctly measuring parameters such as temperature, humidity, or water quality. If any sensor readings are out of range or fail to show up, recalibration or hardware checks may be necessary.

Step 5 ensures the proper functioning of output devices such as pumps or actuators. These devices are activated to confirm that they perform their intended functions, with operational sounds or visible activity serving as confirmation. If any output devices fail to activate, a thorough check of wiring, power supply, and configuration is required.

Step 6, if errors or faults are detected, involves a systematic troubleshooting process. This step includes checking connections, power supplies, and configurations to resolve any issues. Once all faults are addressed, the components should be rechecked to confirm the system is fully operational.

Finally, after resolving any issues, all components are re-evaluated to ensure the system is stable and functioning correctly. Findings should be documented for future reference and verification before deployment, ensuring a reliable and fully operational system.

3.5 Phase 2 : Software Setup

It is focuses on the integration and utilization of three key software tools—Arduino IDE, Firebase Console, and MIT App Inventor—in the development of a robust and efficient IoT-based system. Arduino IDE serves as the primary platform for programming and deploying code to microcontrollers, ensuring seamless interaction with hardware components. Firebase Console provides a cloud-based database solution for real-time data storage, retrieval, and synchronization, enabling efficient communication between the hardware and other devices. Meanwhile, MIT App Inventor is employed for the design and development of a user-friendly mobile application, facilitating remote monitoring and control of the system. Together, these software tools contribute to creating a cohesive and functional system, demonstrating the power of combining programming, cloud technology, and mobile interfaces in modern IoT applications. This report outlines their individual roles, configurations, and the synergy they bring to the overall project workflow.

3.5.1 Arduino IDE

The Arduino Integrated Development Environment (IDE) in Figure 3.12 is a user-friendly software platform used for writing, editing, and uploading code to microcontrollers like Arduino boards and compatible devices such as the ESP8266. It provides a simple text editor with features like syntax highlighting, code completion, and error checking, making programming accessible to beginners. The IDE includes a built-in compiler that translates code into machine-readable instructions and offers easy access to a vast library manager for adding pre-written libraries for sensors and other hardware. It also features a Serial Monitor for real-time debugging and monitoring of data from the microcontroller. Compatible with

Windows, macOS, and Linux, the Arduino IDE is a powerful tool for developing a wide range of embedded systems and IoT projects.

Figure 3.12 Arduino IDE Interface

The Arduino IDE is used to program and upload code to microcontrollers like the ESP8266 or Arduino. To begin, download the IDE from the official Arduino website and install it following the on-screen instructions. After installation, set up the IDE for your board by navigating to File > Preferences and adding the ESP8266 board URL in the "Additional Board Manager URLs" field. Then, open Tools > Board > Boards Manager to install the necessary libraries for your specific board. Create a new sketch under File > New, write or paste your code into the editor, and connect your microcontroller to the PC using a USB cable. Configure the hardware by selecting the correct port under Tools > Port, then click the upload button to compile and upload the code. Use the Serial Monitor to observe debug messages and outputs. For best results, ensure you use appropriate libraries for sensors and modules, and regularly test and debug your code to maintain functionality.

3.5.2 Mit App Inventor

The application was created using MIT App Inventor from Massachusetts Institute of Technology (MIT). It's designed to be user-friendly, featuring a straightforward interface and seamless server-side connectivity. This software offers drag-and-drop functions that assist designers in creating the user interface. MIT App Inventor includes bars with options for buttons and designs. Its user interface consists of two main editors: the design editor for arranging UI elements using drag-and-drop, and the block editor for other functionalities. Figure 3.13 Show designer block of MIT app Inventor.

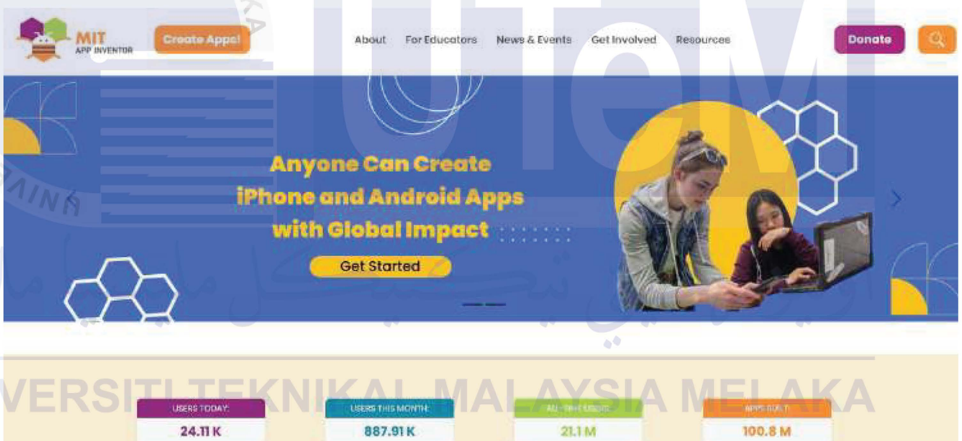


Figure 3.13 MIT App Inventor Website

MIT App Inventor is a platform for designing and building mobile applications using a user-friendly drag-and-drop interface. To get started, visit MIT App Inventor and sign in with your Google account. Create a new project by clicking "Start New Project," entering a project name, and confirming. Design the app's user interface by dragging components like buttons, labels, and text boxes from the Palette onto the Viewer, customizing their appearance and behavior through the Properties tab. Switch to the Blocks view to program the app’s functionality by using drag-and-drop blocks for tasks like reading data from Firebase or triggering actions when buttons are clicked. To connect the app to Firebase, add a Firebase Database component, then configure the Firebase URL and Token in the Properties tab. Test the app in real-time on a mobile device using the AI Companion App, making adjustments as needed. Finally, export the app as an APK file and install it on your mobile device for use.

3.5.3 Firebase Website

Using Google Firebase hostingas in Figure 3.14, we obtained an API code and connected it to the application server for storage. The Firebase Real-time Database is a cloud-based database that stores data in JSON format and keeps it updated in real-time across all connected clients. Firebase provides storage space to save data from devices and display it on the application. To access this space, we use an API key, which must be integrated into the application's server-side to display the data on the app. Figure 3.14 show Firebase console website.

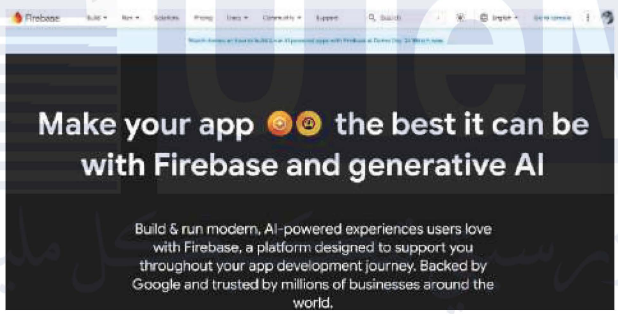


Figure 3.14 Firebase console view

Firebase is a cloud-based platform used to create a real-time database for storing and retrieving data. To begin, sign up and log into the Firebase Console using your Google account. Create a new Firebase project by clicking "Add Project," entering a project name, and enabling Google Analytics if needed. Next, navigate to the Build > Realtime Database section, create the database, select a region, and choose either Locked Mode (default) or Test Mode for initial testing. To configure Firebase for Arduino, generate a private key under Project Settings > Service Accounts in the Firebase Console and include the credentials in your Arduino sketch using Firebase libraries, such as Firebase ESP8266. Finally, monitor and test real-time data updates through the Firebase Console, allowing you to write, read, and update data directly from your hardware.

3.5.4 Software procedure

The procedure in Table 3.2 is for software checking involves multiple stages to ensure each component functions correctly. Step 1 focuses on verifying Arduino IDE functionality. First, open the Arduino IDE and load the program onto the microcontroller (e.g., ESP8266). Upload the program while monitoring the progress bar. If the upload completes successfully, the program is functioning correctly. If errors occur, check for compilation issues or upload failures. Open the Serial Monitor to verify that expected data, such as sensor readings or system logs, is displayed. If no response or incorrect data is shown, troubleshoot the code, hardware connections, or port settings to resolve the issue.

Step 2 involves verifying the Firebase Console functionality. Open the Firebase Console and monitor the real-time database for updates from the hardware system. If the database updates with the correct values in real-time, the system is functioning properly. If no updates or incorrect data are recorded, check the communication between the hardware and Firebase. If Firebase displays error messages, such as connection or authentication issues, ensure the internet connection, authentication keys, and database rules are configured correctly.

Step 3 focuses on testing MIT App Inventor functionality. Deploy the app created in MIT App Inventor to a mobile device and check the user interface for responsiveness. If the app updates data in real time and functions as expected, it is working correctly. If the app crashes or displays incorrect data, further debugging is needed. Test all buttons and controls in the app interface to ensure they respond promptly and perform their intended functions. If buttons do not respond or show delayed actions, debug the app logic and verify proper Firebase integration, then retest the app.

Table 3.2 Indicator for Software Part

Software	**Indicator Functioned**	**Indicator Failed**	**What to be observe**
Arduino IDE	Code compiles and uploads successfully. Serial Monitor shows expected output.	Compilation errors or upload failures. Serial Monitor displays unexpected or no response.	Progress bar during upload, Serial Monitor outputs. Watch for correct sensor readings or system status messages.
Firebase Console	Real-time database updates with correct values. Communication between hardware and Firebase works.	No updates or incorrect data in the database. Firebase error messages or connection issues.	Monitor data changes visually on the Firebase Console. Observe connectivity and test database rules/authentication
Mit App inventor	App interface is responsive and updates correctly. Buttons and controls function as intended.	App crashes or displays incorrect data. Buttons do not respond or there's a delay.	Check for real-time data synchronization in the app interface. Test all interactive elements manually for proper behavior.

3.5.5 Application Programme Flow

The system begins by searching for a Wi-Fi network, repeating the search if no network is found. Once a network is detected, it checks for a successful connection. If connected, the process proceeds to read data from the TDS sensor to measure the Total Dissolved Solids (TDS) level. If the connection fails, the system will continue searching for a network. The gathered sensor data, along with a timestamp, is then sent to Firebase for storage and potential access by other devices or applications. The system then reads the desired mode for the water pump from Firebase to check if the user has changed the operating mode. If a mode change is detected, the system updates the pump operation accordingly. If no change is needed, the system maintains the previous mode. Additionally, the ESP8266 retrieves the current time from a real-time clock (RTC) or an internet time server to ensure accurate timestamps. The current status of the pump (on, off, or standby) is updated in Firebase. The system continuously loops through these steps, monitoring the TDS sensor, checking for mode changes, and updating both the pump status and sensor data in Firebase. Figure 3.15 Sequent of Programme.

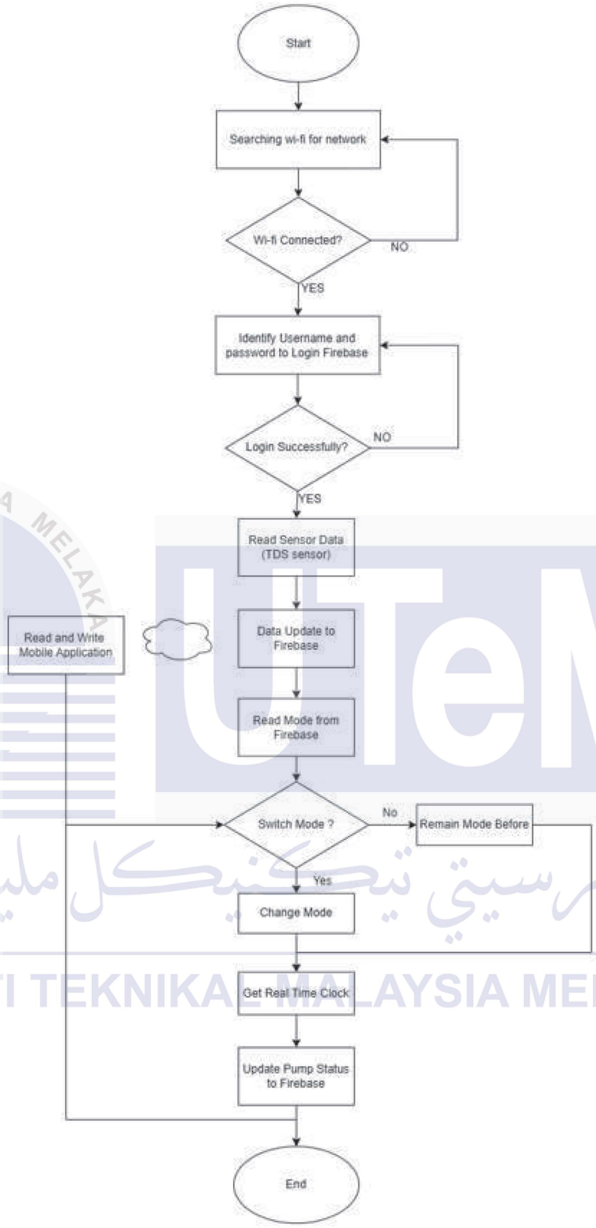


Figure 3.15 Sequent of Programme

3.5.6 Wifi Connection Login

The process in Figure 3.16 begins with the system searching for available Wi-Fi networks in the vicinity. Once a network is found, the user is prompted to enter their username and password for the selected Wi-Fi network. The system then attempts to connect to the network using the provided credentials. If the login is successful, the process continues; otherwise, it may display an error message or prompt the user to try again. Once the Wi-Fi connection is established, the system proceeds to connect to Firebase, a cloud database, and the associated mobile application. Finally, the process concludes, and the system is ready for further interactions or data exchange.

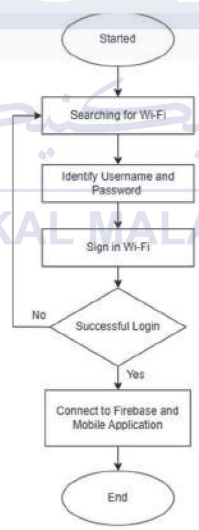


Figure 3.16Wi-fi Login Flowchart

3.5.7 Firebase Data Login

The process in Figure 3.17 begins by prompting the user to enter their username and password. The system then verifies the provided credentials against stored information, either in the ESP8266's memory or on a server. If the login is successful, the process continues; otherwise, an error message is displayed or the user is asked to try again. Next, the system checks if there is any pending data or configuration that needs updating on the ESP8266 or in the mobile application. If an update is needed, the system reads the necessary data from either the ESP8266 or the mobile application. Depending on the data transfer direction, the system then writes the required information to the mobile app or the ESP8266. Finally, the process concludes, and the system is ready for further interactions or commands.

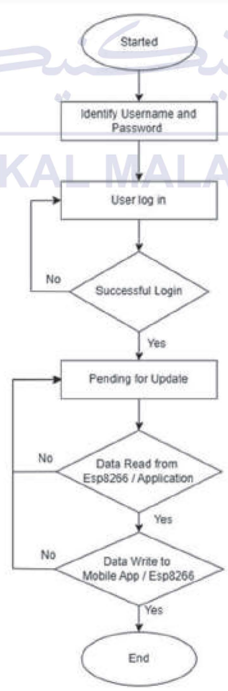


Figure 3.17 Firebase Login Flowchart

3.5.8 Switching Mode Option

The process in Figure 3.18 begins by checking if Mode 1 is selected. If Mode 1 is chosen, the system operates in "Continuous" mode, where the system remains active continuously. If Mode 1 is not selected, the system checks for Mode 2. If Mode 2 is selected, the system operates in "Repeatedly on 1 Hour" mode, where the system activates every hour. If Mode 2 is not selected, the system moves on to check for Mode 3. If Mode 3 is selected, the system operates in "Based on TDS Read" mode, adjusting its operation based on readings from a TDS sensor. If Mode 3 is not selected, the system checks for Mode 4. If Mode 4 is selected, the system operates in "Optimize" mode, adjusting settings for optimal performance. If none of the modes are selected, the system ends.

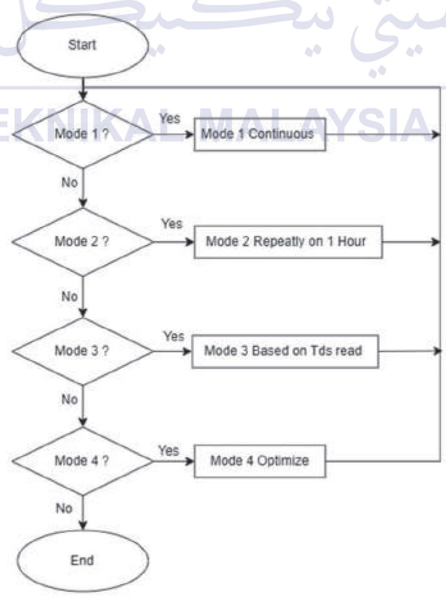


Figure 3.18 Switching Mode Operation

3.5.8.1 Mode 1 : Continuous Operation

The process in Figure 3.19 begins with the activation of the water pump, indicating that it is turned on. Once the pump is activated, the system updates the relay status to the associated mobile application, informing the user that the pump is running. The process concludes at this point, with the system remaining in this state until further instructions or commands are received.

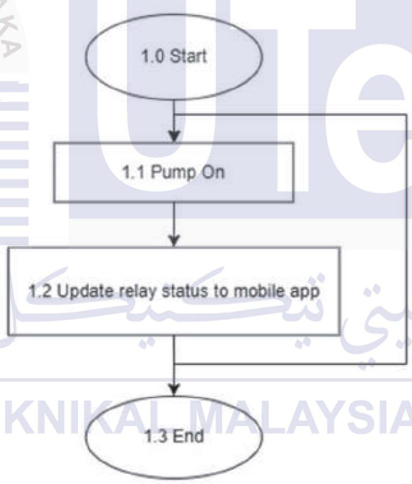


Figure 3.19 Mode 1 Operation Flow

3.5.8.2 Mode 2 : Hourly Operation

The flowchart outlines in Figure 3.20 a control sequence for a water pump system that operates based on whether the current hour is odd or even. The process begins by checking if the system can obtain the current time. If successful, it proceeds to check if the current hour is odd. If the hour is odd, the system turns the pump on and updates the status to the associated mobile app. If the hour is even, the pump is turned off, and the status is updated accordingly. After each action, the system updates the mobile app with the pump's current status (on or off). The process concludes and waits for the next hour to check the on/off condition again.

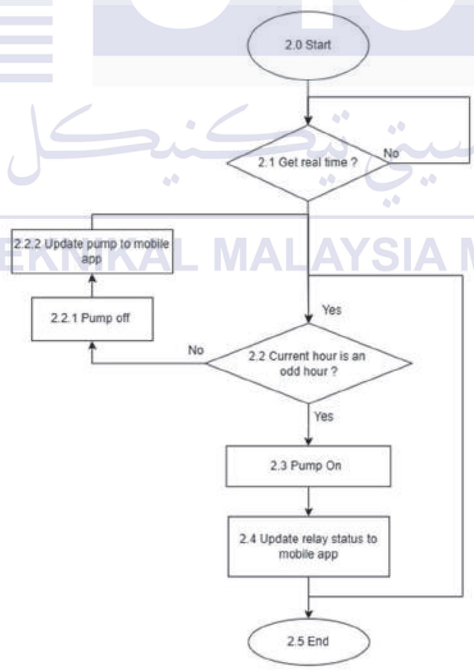


Figure 3.20 Mode 2 Operation Flow

3.5.8.3 Mode 3 : TDS Based Operation

The flowchart outlines in Figure 3.21 the control logic for a water pump system that operates based on readings from a TDS (Total Dissolved Solids) sensor and user input from a mobile application. The process begins by ensuring the pump is initially turned off. The system then enters a waiting state, monitoring for activation signals from the mobile app. Once activated, the system reads the current TDS value from the connected sensor. The TDS value is then compared to a pre-defined threshold, and if the condition is met (e.g., the TDS value exceeds the limit), the system turns the pump on. If the condition is not met, the pump remains off. After adjusting the pump's state, the system updates the mobile app with the current pump status. At the end of the cycle, the pump is turned off, and the system concludes, returning to the waiting state until the next activation signal.

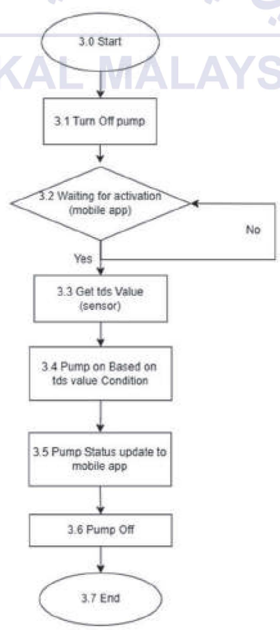


Figure 3.21 Mode 3 Operation Flow

3.5.8.4 Mode 4 : Optimize Operation

The flowchart outlines in Figure 3.22 the control logic for a water pump system that operates based on a combination of time-based scheduling and TDS (Total Dissolved Solids) sensor readings. The process begins by checking the current time and comparing it to a pre-defined schedule. If the current time matches the scheduled time, the system proceeds; otherwise, it waits for the scheduled time to arrive. Once the scheduled time is reached, the system reads the current TDS value from the connected sensor. The TDS value is then analyzed and compared to a pre-defined threshold, and if the condition is met (e.g., the TDS value exceeds the limit), the pump is turned on. If the condition is not met, the pump remains off. After adjusting the pump's state, the system updates the mobile app with the current pump status. At the end of the scheduled operation, the pump is turned off, and the process concludes, waiting for the next scheduled time to repeat the cycle.

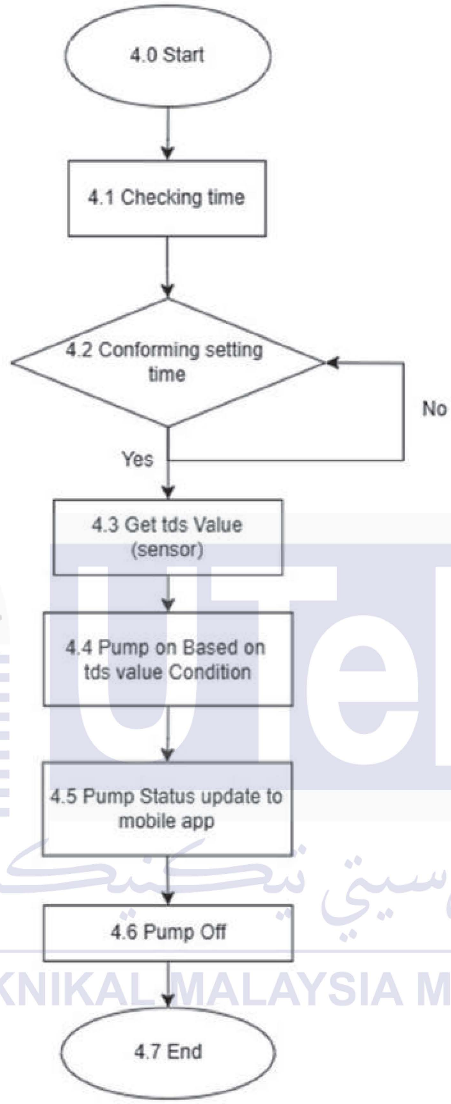


Figure 3.22 Mode 4 Operation Flow

3.6 Phase 3 : Experimental Data Analysis

In this phase, the focus is on the analysis of experimental data collected during the system's testing and operation. The objective is to evaluate the performance, reliability, and efficiency of the system under various conditions. By systematically examining the recorded data, key insights into system behavior are obtained, which aid in identifying trends, validating hypotheses, and addressing any anomalies. This analysis is essential for understanding how well the system meets its design objectives and for proposing potential improvements. The findings from this phase form the foundation for making data-driven decisions to optimize the system further.

3.6.1 Solar Panel as a Source

The solar panel serves as the primary source of renewable energy, and its performance data is collected to evaluate its efficiency and suitability for the system. A multimeter is used for this purpose as in Figure 3.23, with voltage measurements taken by connecting the multimeter in parallel with the panel's terminals and current measurements taken by connecting it in series with the load. This data collection process provides valuable insights into the solar panel's output under various conditions, aiding in performance analysis and system optimization.

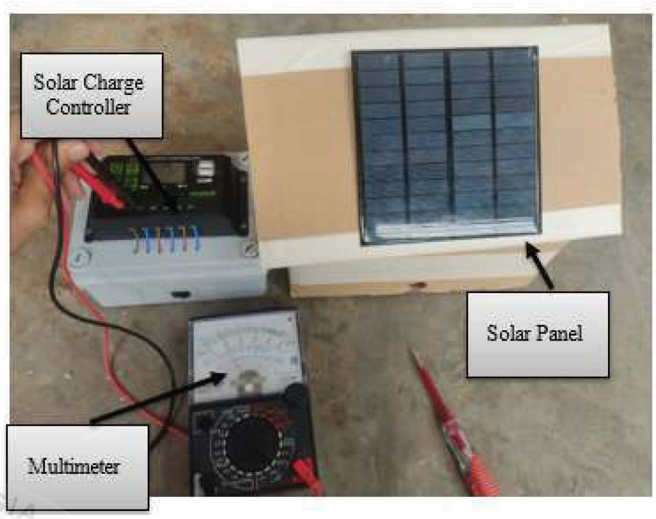


Figure 3.23 Measuring Data in Solar Panel

3.6.2 TDS Sensor for Quality of Water

The TDS (Total Dissolved Solids) sensor as in Figure 3.24 is used to measure the quality of water by detecting the concentration of dissolved solids, providing real-time data on water purity. This data is crucial for assessing the effectiveness of the tank and filtration system in maintaining clean water. The sensor operates through four different modes, allowing the system to adjust its behavior based on specific water quality parameters, such as continuous monitoring, hourly checks, optimized control, and TDS-based decision-making. Data is collected over a period of three days, allowing for a detailed analysis of how the water quality varies over time and under different operational conditions. The tank serves as a storage reservoir, where water is stored after filtration, while the filter works to remove impurities and contaminants. By combining the TDS sensor's data with the filtration system's performance, the system ensures that the water remains clean and meets the required quality standards throughout the entire process. This extended data collection period enhances the accuracy of the analysis, providing valuable insights for system optimization and maintenance.

Figure 3.24 TDS sensor Arrangement in Hardware

3.6.3 Analysist in Battery Percentages

To calculate the battery percentage over time based on the pump's power consumption, we start by determining the energy consumption rate and the battery's total capacity. The pump's power consumption is given in watts (e.g., 5W), and the battery's capacity is measured in watt-hours (Wh), which is calculated as the product of the battery's voltage (V) and ampere-hour (Ah) rating. For example, a 12V, 4.5Ah battery has a total capacity of 54 Wh. The percentage of battery consumed per hour is calculated by dividing the pump's hourly power consumption 5 Wh by the battery's total capacity 54 Wh and multiplying by 100, which gives approximately 9.26% of battery usage per hour. Starting with a full battery (100%), the battery percentage decreases by 9.26% each hour the pump operates. This formula is applied iteratively for each hour, subtracting 9.26% from the previous percentage, until the battery reaches 0%, indicating complete depletion. This method provides a straightforward way to predict the battery life for a specific device based on its power usage.

$$Batery\ Percentages$$

$$= Previous\ Battery\ Percentages - \left(\frac{Power\ Consumption\ Per\ Hour}{Battery\ Capacity} \times 100\right)$$

$$Battery\ Capacity\ (Wh) = Battery\ Voltage\ (V) \times Battery\ Capacity\ (Ah)$$

$$Battery\ capacity\ = 12\,V \times 4.5\,Ah = 54\,Wh$$

$$Power\ consumption\ per\ hour\ = 5\,Wh$$

$$Battery\ Usage\ Per\ Hour\ (\%) = \frac{5}{54} \times 100 = 9.26\%$$

3.7 Summary

In conclusion, this project integrates innovative solar energy and Internet of Things (IoT)-based monitoring technologies to create a sustainable and effective water filtration system designed for fishponds. The process was divided into three separate phases. The first phase involved a thorough analysis of the literature to extract useful information from current technologies related to solar energy integration, IoT-based monitoring, and fishpond water filtration systems. The project's primary goals were defined and accomplished with the help of this foundational research, which included the development of durable hardware components, the creation of user-friendly software for smooth system control and optimization, and the careful assessment of system performance to guarantee optimal functionality.

This device used an ESP8266 microprocessor to run a 12V DC pump that was important to the filtering of the water. Using solar panels to generate DC electricity and charging batteries with the help of a charge controller allowed for a reliable power source to be maintained even though there was no of direct sunlight. The microcontroller was essential to the data processing process because it interfaced with a variety of sensors, such as TDS and turbidity sensors, and it easily connected with a cloud service to enable real-time control and monitoring. In addition to highlighting the system's energy efficiency, this comprehensive configuration enabled remote operational capabilities via the user-friendly mobile application interface.

The software development phase was equally impactful, utilizing the capabilities of MIT App Inventor and Google Firebase to create a friendly and packed with capabilities user experience combined with a powerful real-time data management system. This dynamic application not only enabled remote monitoring and management of the water filtration system, but it also offered users with extensive insights by visualizing sensor data. The application's interactive nature, defined by its simplified simply drop interface, exemplified usability and accessibility, allowing users to efficiently regulate water quality in fishponds from any location. This combination of advanced hardware and software solutions represents a paradigm leap toward a dependable, sustainable, and technologically advanced water filtration system designed specifically for modern aquaculture techniques.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents the project's primary outcomes, with the focus on integrating the ESP8266 microcontroller with Firebase utilizing MIT App Inventor and IoT principles. The project is a hands-on exploration of real-world Internet of Things (IoT) applications, taking advantage of the ESP8266's capabilities, Firebase cloud services, and the user-friendly MIT App Inventor interface.

The successful configuration of the ESP8266 with Wi-Fi connectivity is required for stable functioning. Communication with web servers, cloud services, and other internet-connected devices is enabled by programming the microcontroller to connect to available networks and authenticate with the relevant keys. The addition of a DHT11 sensor for temperature and humidity readings improves its functionality by permitting real-time data transmission to Firebase for remote access and monitoring. Additionally, a Proteus simulation shows a water filtration system managed by the ESP8266, demonstrating its automated control and real-time monitoring capabilities.

The project also involves a water quality testing with Total Dissolved Solids (TDS) and turbidity measurements. These data aid in classifying water clarity and particle concentration levels, which are critical for assessing filtration performance. The results reveal distinct levels of water clarity and turbidity, demonstrating that the filtration system is effective in maintaining water quality.

.

4.2 Phase 1 : Result for Hardware Operation

This phase focuses on evaluating the operational performance of the hardware components within the system. The primary objective is to ensure that each hardware module functions as intended and integrates seamlessly into the overall setup. Testing procedures are conducted under various conditions to verify the reliability, stability, and efficiency of the hardware. The results from these evaluations provide critical insights into the system's physical performance, highlighting any areas that require optimization or adjustment. This phase lays the groundwork for the subsequent integration and functionality of the system.

4.2.1 TDS water Level Condition

Water TDS can be sorted into five levels. Clear water means the water is super clear, showing very high quality with hardly any dissolved solids. Slightly TDS water has a bit of a haze but is still pretty clear, indicating a low level of dissolved solids, making it generally safe for most uses. Moderately TDS water looks somewhat cloudy, with a fair amount of dissolved solids that might need some treatment. Highly TDS water is very cloudy and murky, pointing to a high level of dissolved solids that can seriously affect water quality and require significant treatment. Extremely TDS water is very muddy and opaque, with a very high level of dissolved solids, making it unsuitable for most uses without a lot of purification. Figure 4.1 shows the TDS level as read by the sensor.

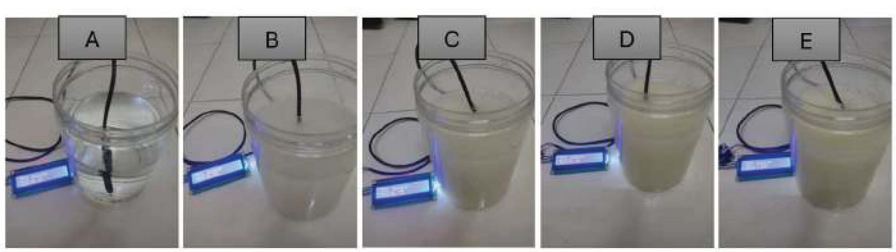


Figure 4.1 TDS Sensor read

In this scale, clear water (A) has a reading of 0.32, indicating very high clarity with few particles. Slightly turbid water (B) shows a reading of 0.53, suggesting a bit of haziness but still relatively clear. Moderately turbid water (C) measures at 0.75, indicating noticeable cloudiness and a moderate level of particles. Highly turbid water (D) has a reading of 0.94, showing very cloudy water with a high concentration of particles. Extremely turbid water (E) has the highest reading at 1.09, indicating extreme cloudiness and a very high level of particles, making it unsuitable without significant purification. All the result recorded in Table 4.1.

Table 4.1This table show result turbidity.

Water level	TDS value convered to Volt
Clear water (A)	0.32
Slightly turbis (B)	0.53
Moderately turbid (C)	0.75
Highy turbid (D)	0.94
Etremely turbid (E)	1.09

4.2.2 Testing in Filtration

The Figure 4.2 shows how water is filtered in a fish tank setup. It starts at the bottom where a pump, labeled with 12V, pushes water up to a filter at the top. This filter contains materials like wool and a biological component that acts like flour particles, representing waste. As water passes through, these "waste" particles get trapped, mimicking how impurities are removed in real water systems.

Figure 4.2 Hardware testing for result

Table 4.2 displays the TDS readings before and after a DC pump is turned on, along with the runtime of the event in minutes. TDS refers to the total dissolved solids in water, indicating the concentration of particles. The "Sensor read (before)" column shows the initial TDS level measured by the sensor. The "Run time (minute)" column indicates the duration of the process or action being observed. Finally, the "Sensor read (after)" column presents the TDS level measured after the event, allowing us to see any changes in the concentration of dissolved solids. This comparison helps assess the effectiveness of the process or action in managing water quality. The result taken with 5 different level water.

Table 4.2 Result of tds level after filtration

Water type	TDS value convered to Volt (before)	Run time (minute)	TDS value convered to Volt (after)
1	1.24	25	0.92
2	1.01	20	0.82
3	0.84	15	0.73
4	0.54	10	0.51
5	0.40	5	0.36

The bar chart in Figure 4.3 titled "TDS Sensor Read" illustrates the change in Total Dissolved Solids (TDS) values, converted to voltage, over a series of run times measured in minutes. The orange bars represent the TDS values before the run, and the green bars show the values after the run. The data indicates a consistent decrease in TDS values over time, suggesting that the filtration system becomes more effective as runtime increases. At 25 minutes, the TDS value starts at 1.24 volts and drops to 0.92 volts. This downward trend continues, with the values decreasing to 0.4 volts before and 0.36 volts after at 5 minutes. The overall reduction in voltage values signifies a successful reduction in TDS levels, highlighting the efficiency of the filtration or treatment process over time.

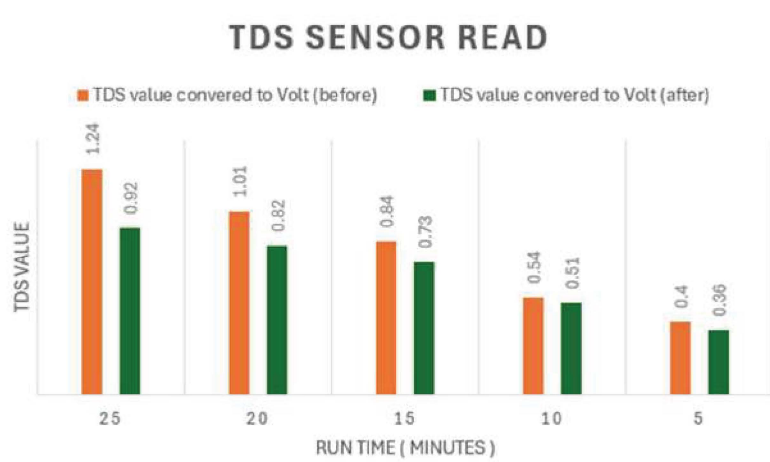


Figure 4.3 TDS sensor read graph before and after

4.2.3 Experimental Prototype Setup

The experimental prototype serves as a practical implementation of the proposed system for controlling the pump based on water quality (TDS) and operational requirements. This prototype is designed to test the functionality of the system across multiple modes of operation, ensuring its ability to meet specific performance criteria under real-world conditions.

4.2.4 Wiring Connection

The wiring diagram in Figure 4.4 depicts a solar-powered water pump system incorporating a TDS sensor and an ESP8266 module for monitoring and control. The solar panel captures sunlight, converting it to DC electricity stored in a sealed lead-acid battery via a solar charge controller, which regulates power flow and prevents overcharging. A DC-DC buck converter stabilizes the battery's 12V output, supplying appropriate voltage to the ESP8266 and other components. The ESP8266 microcontroller reads water quality data from the TDS sensor and controls a relay module to switch the 12V DC water pump on or off based on programmed logic, such as TDS thresholds or time-based schedules. A manual switch provides an option to override automatic control. The system ensures efficient solar energy utilization, continuous monitoring, and responsive water pump operation for applications like irrigation or water management.

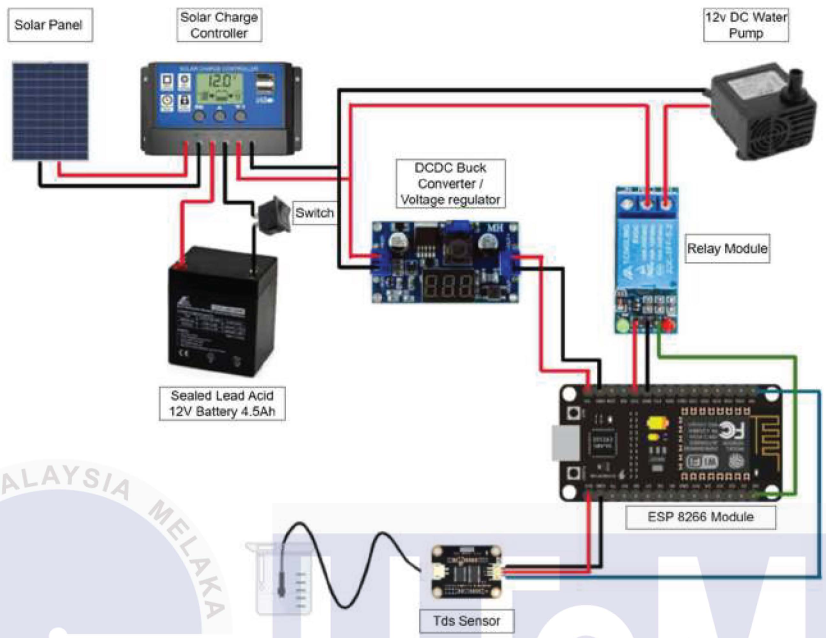


Figure 4.4 Full Wiring Diagram Operation

4.2.5 Improvement Protoype in filtration

The hardware prototype in is Figure 4.5 designed to facilitate the flow of water through the filtration system, ensuring clean water is returned to the tank. Water flows from the tank (on the left side) to the filter (on the right side) via a water pump placed in the middle tank. The pump is responsible for circulating the water through the system. Additionally, a TDS sensor is also placed in the middle tank to monitor the water quality in real-time.

Once the water passes through the filter, it moves from the bottom to the top of the filter, where it is purified through various layers (e.g., activated carbon, wool, and biological filters). After filtering, the purified water flows back into the tank, completing the cycle. This setup ensures that the water is constantly circulated, filtered, and monitored for quality, maintaining an efficient water management system.

Figure 4.5 Improvement Prototype

4.2.5.1 Tank and Pump System

A mechanisms in Figure 4.6 for collecting data on water quality. A TDS sensor is used to measure the Total Dissolved Solids (TDS) in the water. The sensor is integrated with the system to provide real-time data, which is essential for making informed decisions about pump operation. This section focuses on the storage and management of water. The tank serves as a reservoir, while the pump operates based on the selected mode. The system uses TDS readings from the source to adjust the pump's operation, ensuring water quality is maintained while optimizing energy usage.

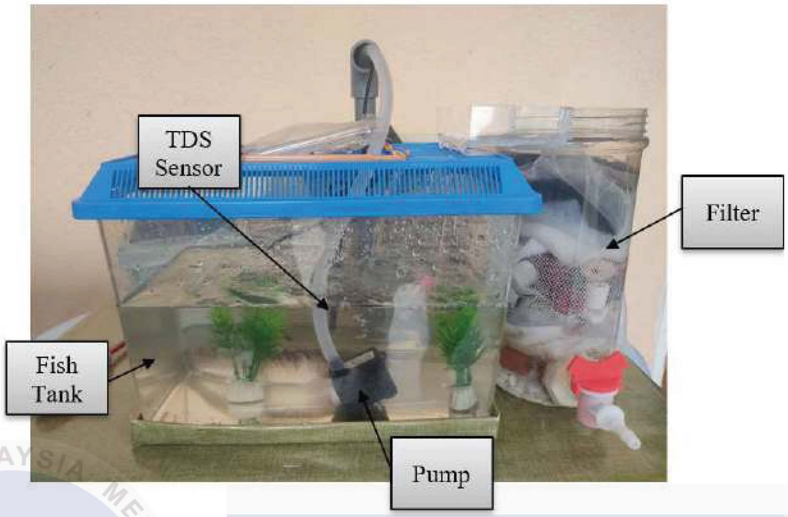


Figure 4.6 Arrangement Hardware in Operation

4.2.5.2 Solar Power Source and Energy Storage

This component in Figure 4.7 includes a solar charge controller and a battery system. The solar charge controller manages the energy harvested from solar panels, ensuring efficient charging of the battery. The battery stores the energy, providing a reliable power source for the pump and associated electronics, even during periods of low sunlight. This renewable energy setup ensures the system's sustainability and independence from conventional power sources.

Figure 4.7 Arrangement in Power Generation and Storage

4.2.5.3 Filter to Clean Up Water

The water filtration system Figure 4.8 is designed for upward flow, starting from the bottom and passing through multiple layers for comprehensive purification. At the bottom, a layer of small white stones provides structural support and facilitates initial water flow. Above this, a biological filter promotes the growth of beneficial microorganisms to break down harmful substances. Next, a combination of wool and a net traps finer particles and debris. The final layer consists of activated carbon, which removes odors, chlorine, and organic impurities, ensuring the water is clean and safe as it exits the filter. This arrangement maximizes filtration efficiency by sequentially addressing different types of impurities.

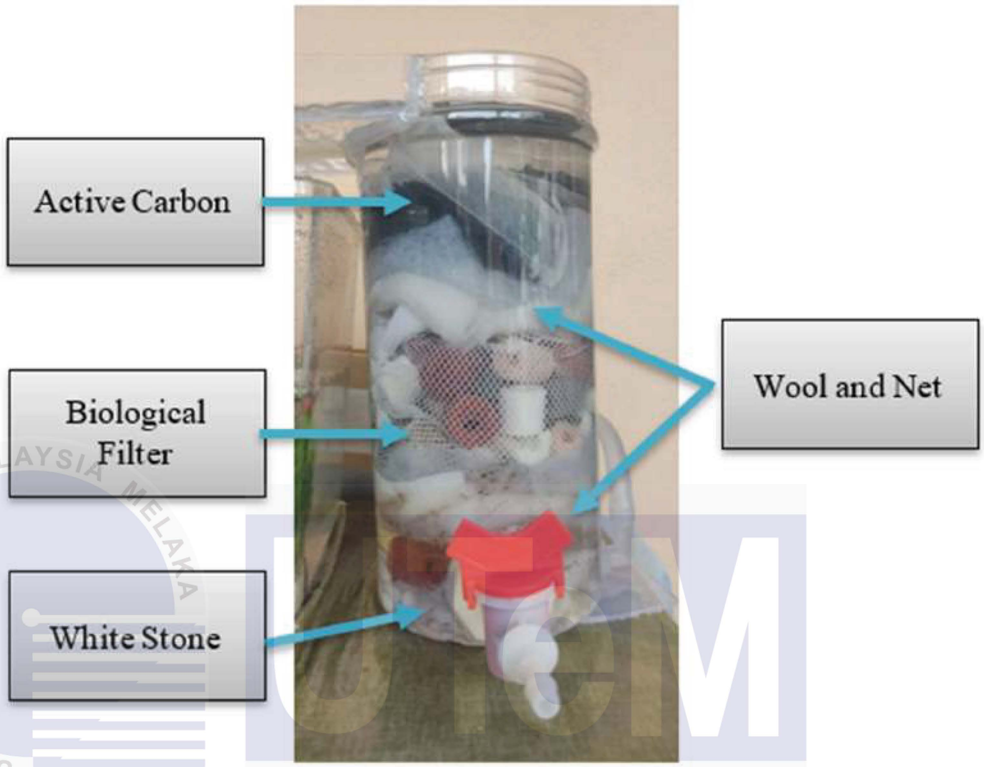


Figure 4.8 Filter Arrangement

4.2.5.4 Waste Trapped at Bottom Filter

The filtration system is designed to efficiently trap and collect waste at the bottom of the filter as in Figure 4.9. As water flows upward, larger debris and sediments settle at the bottom, where a valve is strategically installed. This valve allows for easy removal of accumulated waste, ensuring the filter remains functional and preventing clogging. Regular draining of the waste through the valve enhances the system's efficiency and prolongs the lifespan of the filtration materials. This simple yet effective mechanism simplifies maintenance and ensures consistent performance.

Figure 4.9 Disposal Outlet Flow

4.3 Phase 2 : Result for Software Integration

In order to read temperature and humidity data from the DHT11 sensor using the Internet of Things, it can be integrated with the ESP8266. This data is then sent to a cloud service for remote access. Through the use of MIT App Inventor, it can communicate with the Firebase console. A strong Google platform that serves as a real-time database and synchronization tool for the microcontroller and mobile application.

The setup makes sure efficient management and prompt reaction to any temperature changes by allowing continuous monitoring of temperature and humidity as well as offering an interactive interface for users to track conditions from any location. Figure 4.10 show data read from sensor displayed in Arduino IDE and Firebase Console.

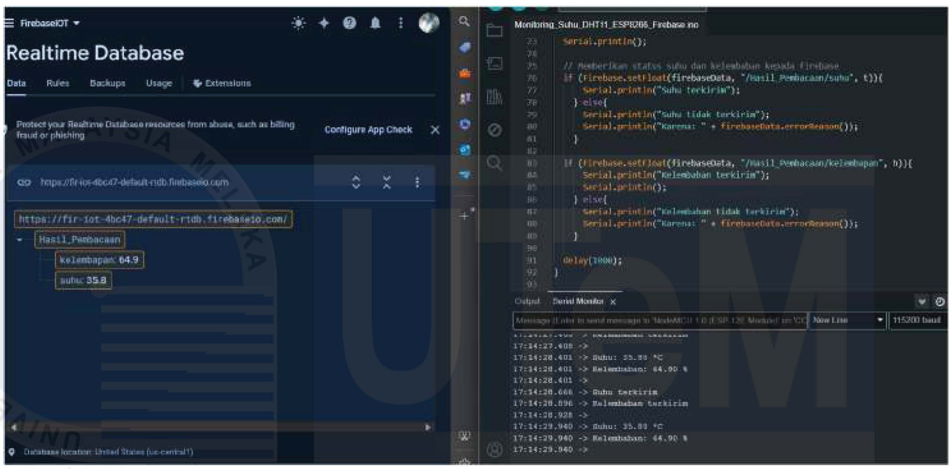


Figure 4.10 Communication arduino with realtime firebase

4.3.1 Arduino IDE Software

It is important to correctly set up the ESP8266 with the correct network information in order ensure a steady Wi-Fi connection. In order to do this, programed the ESP8266 to look for available networks. Authenticate with the SSID and password. After it is connected, the ESP8266 can communicate with web servers, cloud services, and other devices by sending and receiving data over the internet. Figure 4.11 show Esp8266 successfully connect to wi-fi.

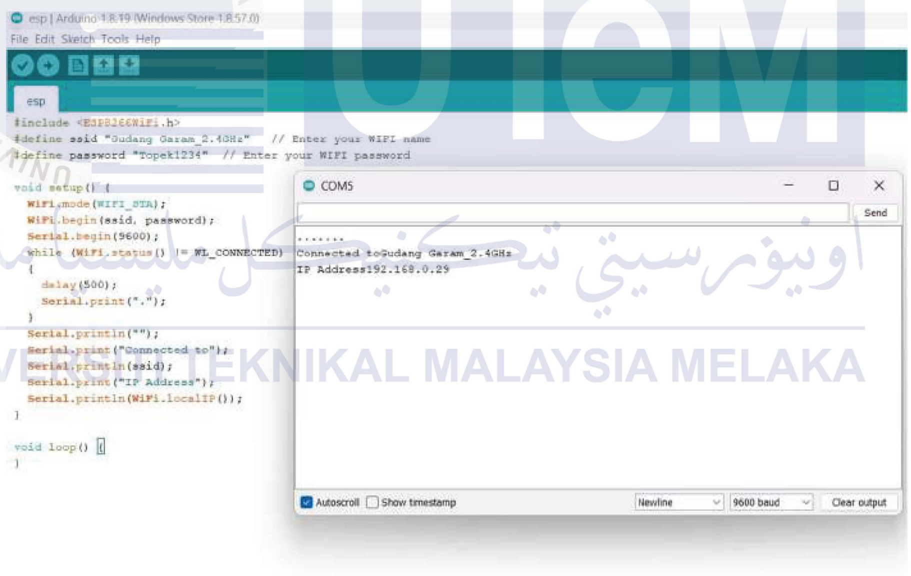


Figure 4.11 Esp 8266 connecting with wi-fi

The Arduino acts as the central controller for managing various system components, including the TDS sensor, pump, clock, WiFi, and Firebase integration. It reads data from the TDS sensor to monitor water quality, updates the pump status, and logs the data to Firebase via WiFi. The system's status and data are displayed on the Serial Monitor for real-time tracking. In the Arduino IDE, the libraries used are shown at the top of the screen, and the "Add Library" button, located at the left-middle of the interface, allows users to include new libraries. The code editor in Figure 4.12 appears in the center, where the user can write and edit the program. The "Compile" button is at the top left, and the "Upload" button is beside it, enabling the user to compile and upload the code to the Arduino board. Once uploaded, the system operates by continuously reading the TDS sensor, updating the pump status, and logging data to Firebase, all while providing real-time feedback on the Serial Monitor.

Figure 4.12 Arduino IDE Result Interface

4.3.2 Firebase Console

The Firebase integration in Figure 4.13 plays a critical role in remotely managing and storing real-time data from the system, such as sensor values, pump status, and the manual switch settings. The real-time database updates continuously, storing the latest TDS sensor readings, providing updates on the pump status (whether it's running or idle), and logging the status of the manual switch for manual control. Firebase also updates the operating mode (Mode 1, 2, 3, or 4) to reflect the current configuration of the pump. Additionally, on the Firebase console, the Realtime Database section is accessible via a button on the left side of the interface, which allows users to monitor and manage data in real-time. In the project settings, users can configure the Firebase settings, manage the authentication methods, and adjust parameters such as database rules and security, ensuring that the data is secure and accessible according to the desired permissions. This integration facilitates seamless control and monitoring, making it easy to track the performance of the system and remotely adjust settings.

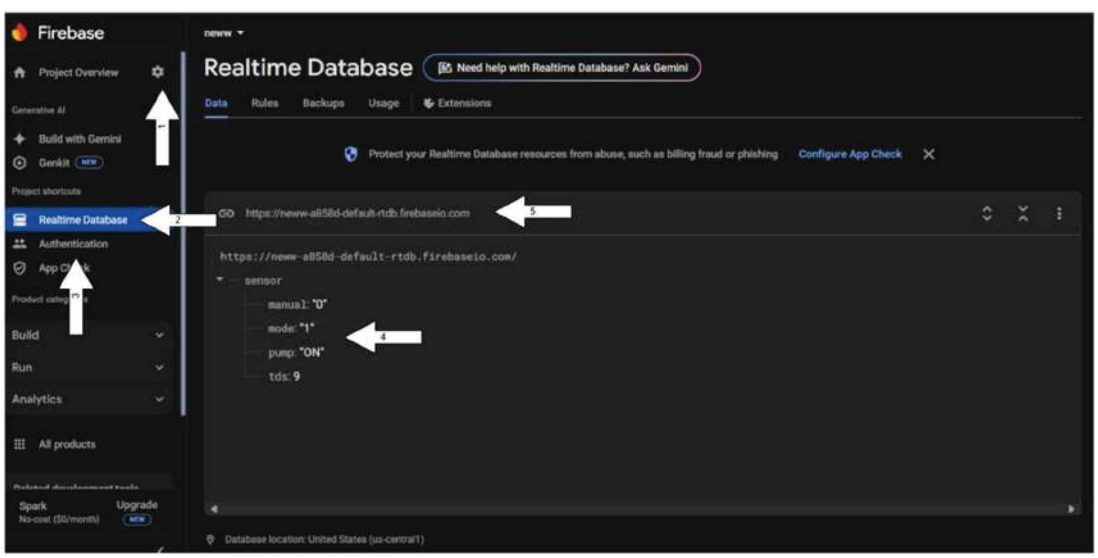


Figure 4.13 Firebase Result Interface

4.3.3 Mit App Inventor

The **MIT App Inventor** platform in Figure 4.14 enables the creation of a mobile app interface for controlling and monitoring the system effectively. The app interface consists of several key sections: At the **top**, the **Build** and **Connect** options allow for compiling the app and connecting it to a mobile device for testing. Users can also easily switch between **screens**, and toggle between **Blocks** and **Designer** views. The **Designer** section on the left allows for the visual arrangement of UI components, such as buttons, labels, and sliders, to create the app layout. The **Blocks** section on the right allows users to define the app's functionality and logic using visual programming blocks. The center area displays the live preview of the app interface, simulating how the app will look and behave on the user's mobile device. This structure provides an intuitive and efficient environment for building, testing, and deploying the app, allowing users to remotely control the pump and monitor water quality through a simple and interactive interface.

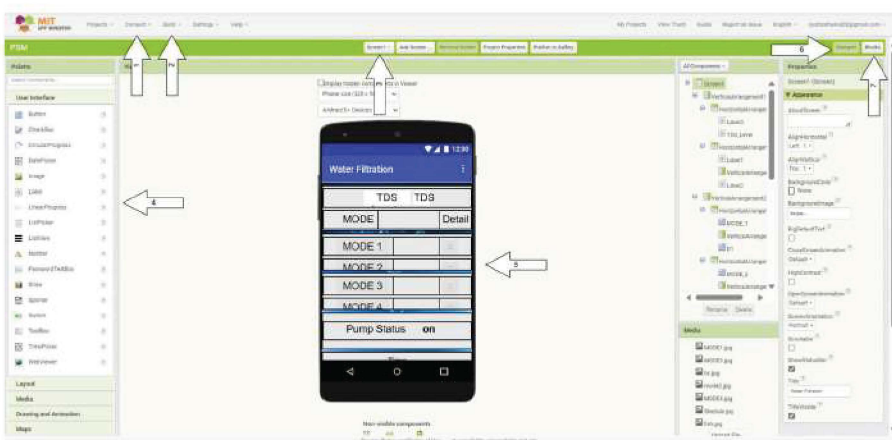


Figure 4.14 MIT App Inventor Result Interface

4.3.3.1 Mobile Application

The mobile app interface in Figure 4.15 is designed for intuitive control and monitoring of the system. At the top of the screen, the TDS value is prominently displayed, showing the real-time reading of the water quality. Below this, there are buttons to change between the four modes of operation (Mode 1, Mode 2, Mode 3, Mode 4), each accompanied by a brief description of the mode's functionality, allowing the user to select the desired operation for the pump. Next, the pump status is shown, indicating whether the pump is active or idle. At the bottom of the screen, a time clock is displayed, providing the current time to assist with scheduling and monitoring pump operation. This layout ensures the user can easily view key data, control the pump, and monitor water quality with a user-friendly interface.

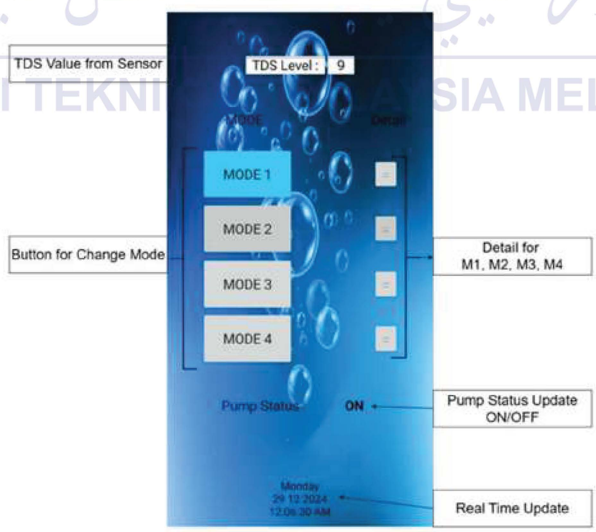


Figure 4.15 Mobile Application View

4.3.3.2 MIT App Inventor Blocks - Firebase Integration and Navigation

In MIT App Inventor, the blocks as in Figure 4.16 are used to define the logic for the mobile app's interaction with Firebase and screen navigation. When the user clicks any of the mode buttons (1, 2, 3, or 4), the app stores the selected mode value to Firebase using the appropriate blocks that send the data to the Firebase database. Each button is associated with an event handler that triggers when clicked, storing the mode value (e.g., "Mode 1," "Mode 2") to the Firebase real-time database.

Additionally, each button (1, 2, 3, or 4) is connected to a block that opens a new screen, displaying detailed information about that specific mode. For example, when the user clicks Button 1, the app navigates to Screen 1 with detailed information about Mode 1. Similarly, clicking on Buttons 2, 3, or 4 opens Screen 2, Screen 3, or Screen 4, respectively, each showing detailed information for the corresponding mode. This setup allows the user to interact with the app, view detailed mode information, and store relevant data to Firebase for remote monitoring and control.

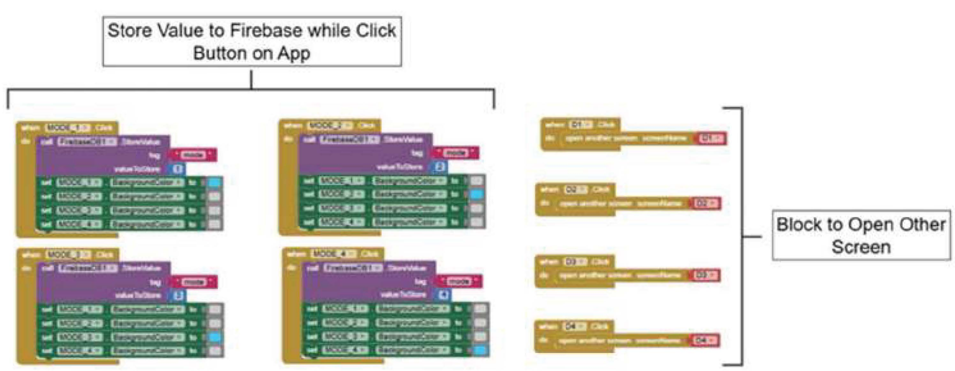


Figure 4.16 Part 1 Block MIT App Inventor

4.3.3.3 MIT App Inventor Blocks - Initialization, Storing Date & Time

To enhance functionality, MIT App Inventor blocks are used to initialize the app and store date and time data as in Figure 4.17.

Initialization Block: When the app is launched, the initialize block is triggered to set up variables and prepare the app for use. This block may set default values for various parameters (e.g., default mode, pump status), and initialize connections to Firebase and the time component.List Day and

Time Format: The app stores the current date and time in a formatted list for logging or display. A list block is used to create a list of values that includes the day, time, and TDS sensor reading. This list can then be sent to Firebase to store the data for each interaction with the system.

Calling Time and Date: To format and display the current date and time, a block is used to call the Clock component, which retrieves the system's date and time. The app then formats this data as needed (e.g., in "dd/mm/yyyy hh:mm:ss" format). This formatted time and date can then be logged in Firebase and displayed on the app interface, ensuring that each action (such as changing the mode or updating the pump status) is recorded with the correct timestamp.

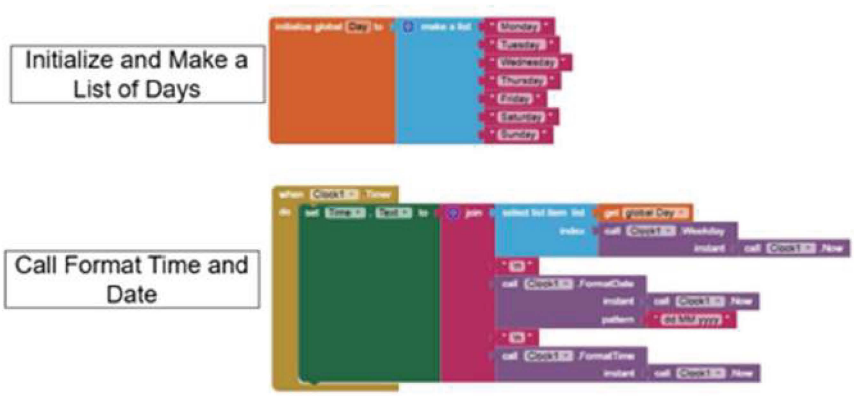


Figure 4.17 Part 2 Block MIT App Inventor

4.3.3.4 MIT App Inventor Blocks - Storing Initial Values and Firebase Integration

To initialize and store the initial values, and to ensure real-time data updating in Firebase, the following blocks as in Figure 4.18 can be used:

Store Initial Values (1, 2, 3, 4): When the app starts, the initialize block is triggered to store the initial values of the system modes. The values 1, 2, 3, or 4 (representing the different operational modes) are stored in global variables or lists. These values are linked to the mode buttons in the app, allowing users to select and update the mode.

Calling Firebase on Screen Initialize: When the screen is initialized (when the user first opens the app or a new screen), a Firebase call block is triggered. This block connects the app to the Firebase database, allowing it to retrieve and store data in real-time. For example, it might retrieve the current mode value and display it on the interface. The initialize block can also call Firebase to check and update the current status of the pump, TDS readings, or mode selection.

Data Change and Real-time Updates: The app continuously monitors any changes in data, such as the user selecting a different mode or updating the pump status. When a change occurs (e.g., selecting Mode 1, 2, 3, or 4), the corresponding button block updates the Firebase database with the new value. The "When Button Clicked" block updates the value stored in Firebase with the new mode and can also update the UI elements accordingly.

WriteValue to Firebase : The "Firebase store" block is used to write the updated value (e.g., the new mode or pump status) back into the Firebase real-time database. This ensures that the data is updated instantly and remains synchronized across all connected devices. For instance, when Mode 1 is selected, the Firebase entry for current_mode will be updated to "Mode 1," and this will reflect across all devices connected to the system in real-time.

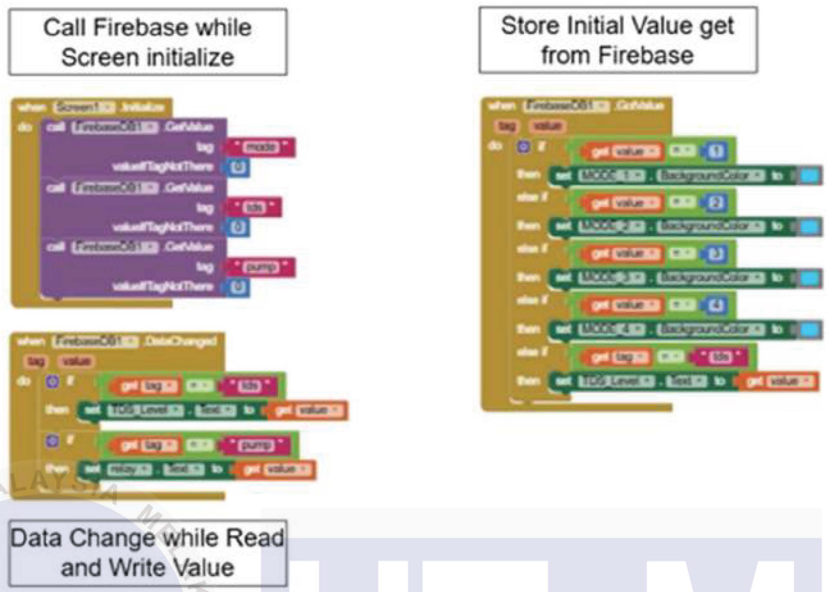


Figure 4.18 Part 3 Block MIT App Inventor

4.4 Phase 3 : Analysis of Resut for the System

The bar chart show in Figure 4.19 titled "Power VS Time" shows the energy used by a pump that uses 5 watts of power over different durations. When the pump runs for 24 hours, it uses 120 watts. If it runs for 10 hours, it uses 50 watts. For 5 hours, it consumes 25 watts, and for 2 hours, it uses 10 watts. This relationship is straightforward: the longer the pump runs, the more energy it consumes, with the energy used being the product of the power and the time the pump is on.

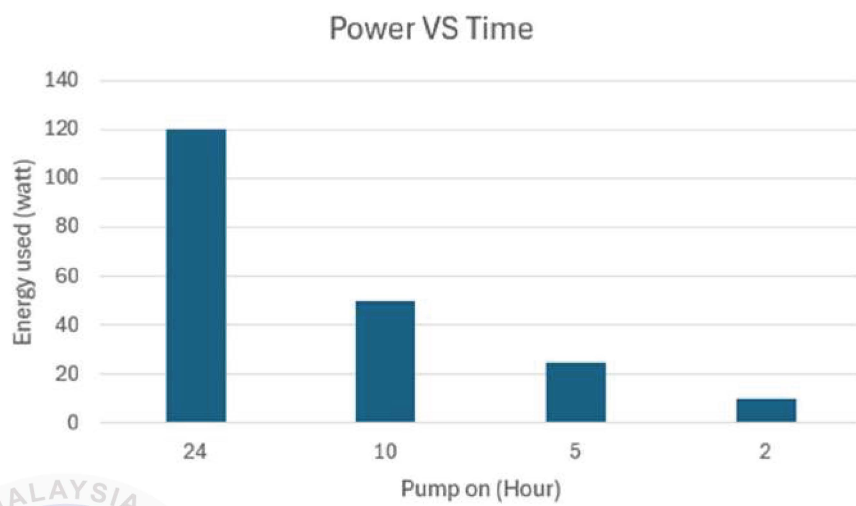


Figure 4.19 Power consumption graph

4.4.1 Solar Charging

The data Table 4.3 presented shows the voltage (Voc) and current (Imp) readings over a 12-hour period, likely from a solar panel system. From 0700 to 1900, the system displays changes in voltage and current that follow a typical daily solar cycle:

Voltage (Voc) starts at 8.0V at 0700 and gradually increases, reaching a peak of 13.2V at 1400, before decreasing toward the evening, ending at 9.5V at 1900. This reflects the natural increase in solar panel voltage due to sunlight intensity during the day, followed by a decrease as sunlight wanes in the evening.

Current (Imp) starts at 0A at 0700, increases to a peak of 240mA (0.24A) between 1100 and 1400, and then gradually decreases back to 0A by 1900. This shows the typical performance of a solar panel, where current output is highest around midday when sunlight is strongest and falls off in the evening as sunlight decreases.

Table 4.3 Measuring in Solar Data

Time	Voc/Volt	Imp/Ampere
0700	8.0	0
0800	10.5	150m
0900	12.0	200m
1000	12.2	230m
1100	13.00	240m
1200	12.50	240m
1300	12.50	240m
1400	13.20	240m
1500	12.30	240m
1600	11.5	230m
1700	10.4	200m
1800	9.9	220m
1900	9.5	0

Overall, the system operates most efficiently during midday hours (1100-1400) as in Figure 4.20, with optimal current generation around 240mA. The system appears to generate no current before 0700 and after 1900, likely due to insufficient sunlight. The voltage consistently follows the expected pattern of a solar panel, peaking during the day and dropping off as the sun sets. This data could be used to optimize energy storage and power management for applications relying on solar energy.

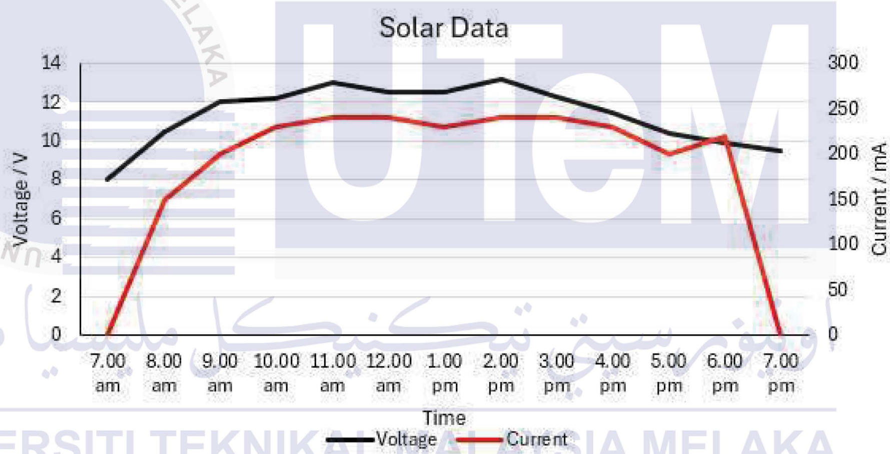


Figure 4.20 Graph of Solar Voltage and Current

4.4.2 Mode 1 Analysisist

In Mode 1, the pump operates continuously, running 24 hours a day without interruption. Unlike the on/off switching mechanism in Mode 2, Mode 1 represents a constant state of operation where the pump remains in the "on" state throughout the entire period. This is achieved using a binary switching mechanism, where the pump's status is controlled digitally. A value of **1** corresponds to the pump being "on," while a value of **0** represents the pump being "off."

In Mode 1, the digital signal in Figure 4.21 is consistently set to **1**, ensuring the pump is always active. When visualized on a digital graph, the y-axis (representing the pump's state) consistently stays at **1** across all points on the x-axis (time), creating a flat horizontal line at the top of the graph. This uninterrupted "on" state ensures there are no periods of inactivity.

This mode is particularly useful in systems that require uninterrupted water flow, such as in critical applications where a continuous supply is essential for maintaining operational stability. Examples include agricultural irrigation systems, industrial cooling systems, or any scenario where downtime could result in significant disruption or damage.

By running the pump continuously, Mode 1 eliminates the complexity of scheduling or monitoring on/off cycles, as the system does not need to account for downtime. However, this constant operation mode requires careful attention to energy management and maintenance. Since the pump consumes power continuously, it places a higher demand on the energy source, such as a battery or mains power, and may require larger capacity batteries or more robust energy systems to sustain 24/7 operation. Additionally, continuous operation can lead to increased wear and tear on the pump, necessitating regular maintenance to ensure reliability and prevent unexpected breakdowns.

The binary switching mechanism used in this mode simplifies the control system while ensuring reliability. By maintaining the switch at **1**, the pump operates without interruptions, making Mode 1 ideal for applications that prioritize consistent and predictable performance over energy efficiency.

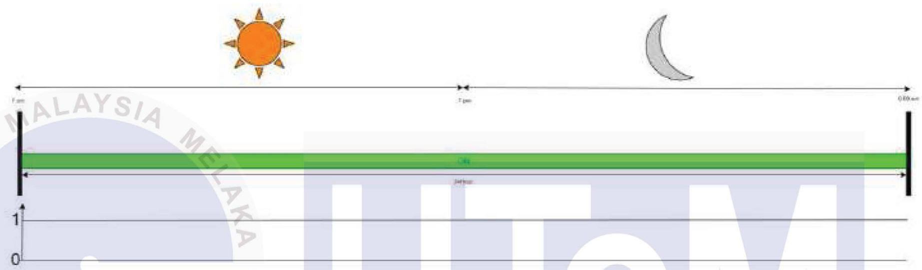


Figure 4.21 Mode 1 Switching Operation

4.4.2.1 TDS Value Mode 1

The TDS values recorded in Table 4.4 over the course of Day 1 in Mode 1 show fluctuations in water quality throughout the day. The TDS values vary between 19 ppm and 25 ppm, with lower values indicating better water quality. Based on the readings:

The lowest TDS value of 19 ppm is recorded consistently during the early morning (from 5:00 am to 7:00 am) and evening hours (from 3:00 pm to 6:00 pm), indicating periods of better water quality.

Higher TDS values (24-25 ppm) are observed during midday and nighttime (from 12:00 pm to 12:00 am), suggesting that the water quality slightly deteriorates during these times.

Table 4.4 TDS Value in Mode 1 by 3 Days

Mode 1	Day 1	Day 2	Day 3
Time	TDS Value (PPM)	TDS Value(PPM)	TDS Value(PPM)
7.00 am	20	19	19
8.00 am	22	19	19
9.00 am	23	25	20
10.00 am	20	23	19
11.00 am	19	24	23
12.00 am	24	24	23
1.00 pm	24	21	21
2.00 pm	21	19	24
3.00 pm	19	20	24
4.00 pm	20	19	21
5.00 pm	19	19	19
6.00 pm	19	19	20
7.00 pm	25	19	19
8.00 pm	23	25	19
9.00 pm	24	23	25
10.00 pm	19	24	23
11.00 pm	19	19	24
12.00 pm	25	19	23

In conclusion, the lower TDS levels (around 19 ppm) as in Figure 4.22 are associated with better water quality, which is ideal for maintaining the pump's efficiency and overall system performance. High TDS levels, such as those seen at 12:00 pm and 7:00 pm, could indicate higher dissolved solids, which may reduce the water's quality. Regular monitoring and filtration are needed to maintain the water quality within an optimal range (close to 19 ppm).

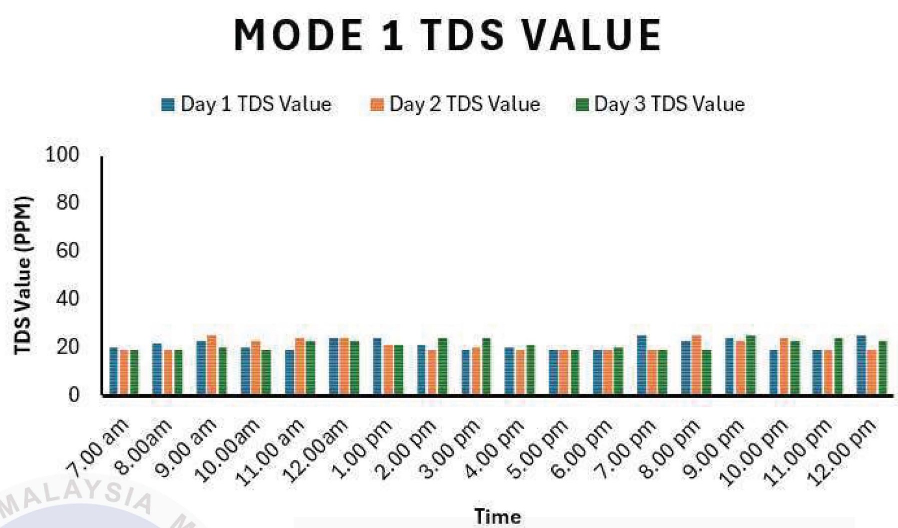


Figure 4.22TDS Value in Mode 1 Graph by 3 Day

4.4.2.2 Power Consumption Mode 1

The power consumption data recorded in Table 4.5 over Day 1 in Mode 1 shows a consistent usage of 5 units throughout the day, with no variation in the recorded power consumption. Each hour, from 7:00 am to 6:00 am the next day, consumes exactly 5 units of power. This indicates that the system operates at a constant power consumption rate throughout the entire 24-hour period.

Since the power usage remains the same across all hours, the cumulative power consumption will increase linearly over time. After 24 hours, the total power consumed will be 120 units (5 units x 24 hours). This consistent power usage suggests that the system operates at a fixed power demand, irrespective of the time of day or other variables. Thus, the system is designed to function at a stable energy requirement, making it predictable in terms of energy consumption.

Table 4.5 Power Consumption In Mode 1

Mode 1	Day 1	Day 2	Day 3	Cumulatif Power	Cumulatif Power	Cumulatif Power
Time	Power consumption	Power consumption	Power consumption			
7.00 am	5	5	5	5	125	245
8.00am	5	5	5	10	130	250
9.00 am	5	5	5	15	135	255
10.00am	5	5	5	20	140	260
11.00 am	5	5	5	25	145	265
12.00am	5	5	5	30	150	270
1.00 pm	5	5	5	35	155	275
2.00 pm	5	5	5	40	160	280
3.00 pm	5	5	5	45	165	285
4.00 pm	5	5	5	50	170	290
5.00 pm	5	5	5	55	175	295
6.00 pm	5	5	5	60	180	300
7.00 pm	5	5	5	65	185	305
8.00 pm	5	5	5	70	190	310
9.00 pm	5	5	5	75	195	315
10.00 pm	5	5	5	80	200	320
11.00 pm	5	5	5	85	205	325
12.00 pm	5	5	5	90	210	330
1.00 am	5	5	5	95	215	335
2.00 am	5	5	5	100	220	340
3.00 am	5	5	5	105	225	345
4.00 am	5	5	5	110	230	350
5.00 am	5	5	5	115	235	355
6.00am	5	5	5	120	240	360

Over the course of 3 days, the cumulative power consumption will continue to follow the same pattern as Figure 4.23, as the system operates at a constant rate of 5 units per hour. For each day, the system consumes 120 units of power (5 units x 24 hours). Therefore, after 3 days, the total power consumption will be 360 units (120 units x 3 days). This linear increase in power consumption highlights that the system's energy requirements remain consistent over multiple days, making it easy to predict and manage energy usage over an extended period.

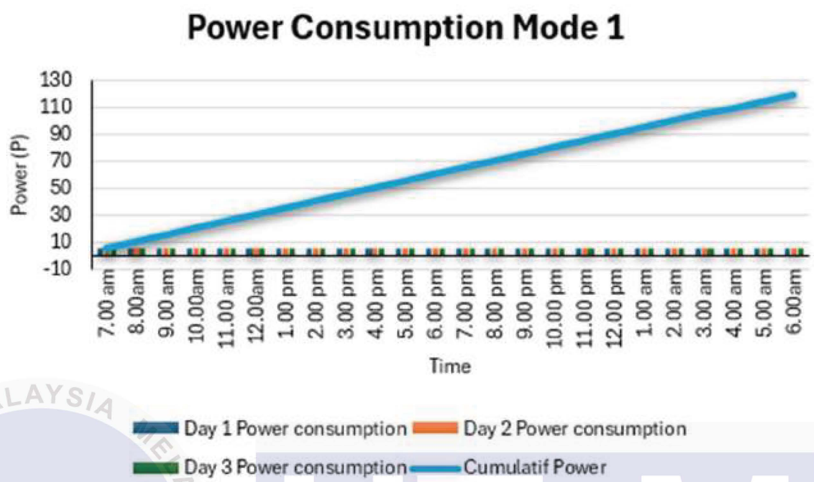


Figure 4.23 Power Consumption In Mode 1 Graph

Table 4.6 Battery Percentages Mode 1

Time	Power Consumption (W)	Cumulative Power (Wh)	Battery Percentage Remaining
7:00 AM	5	5	100%
8:00 AM	5	10	90.70%
9:00 AM	5	15	81.40%
10:00 AM	5	20	72.10%
11:00 AM	5	25	62.80%
12:00 PM	5	30	53.50%
1:00 PM	5	35	44.20%
2:00 PM	5	40	34.90%
3:00 PM	5	45	25.60%
4:00 PM	5	50	16.30%
5:00 PM	5	55	7.00%

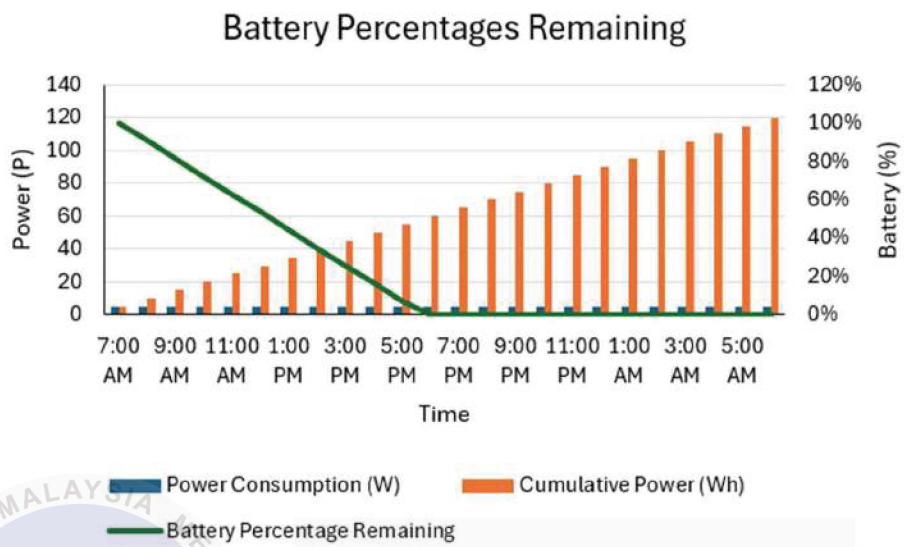


Figure 4.24 Battery Percentages Mode 1 Graph

With a 5W pump running continuously for 24 hours, the system will consume 120 Wh of power, which exceeds the 54 Wh capacity of the 12V, 4.5Ah battery. By 6:00 pm, the battery will be depleted as in Table 4.6, and by the following hours, it will be significantly overdrawn shown in Figure 4.24, resulting in negative percentages for battery capacity. To prevent battery depletion, the system would need to either reduce the pump's operational hours, utilize a larger battery, or employ a charging mechanism to maintain sufficient power.

4.4.3 Mode 2 Analysis

The pump in the system in Figure 4.25 operates using a binary switching mechanism, where the pump is controlled with digital signals. A value of 1 represents the pump being turned on, while a value of 0 represents the pump being turned off. This on/off control method follows a digital logic, ensuring the pump's operation is easy to monitor and manage.

When visualized as a digital graph, the y-axis corresponds to the pump's state (with values of 1 for "on" and 0 for "off"), while the x-axis represents time. As the system operates, the graph will show alternating periods of 1s and 0s, indicating the periods during which the pump is active or inactive. For example, if the pump is programmed to run for 1 hour and then turn off for the next hour, the graph will show 1 for the hour the pump is on, followed by **0** for the hour it is off.

This binary switching method ensures that the pump operates according to a predefined schedule, maximizing energy efficiency by allowing it to run only when needed. It also simplifies the automation process, as the pump's status can be easily monitored and adjusted based on the system's requirements, leading to improved management of water flow and overall system performance.

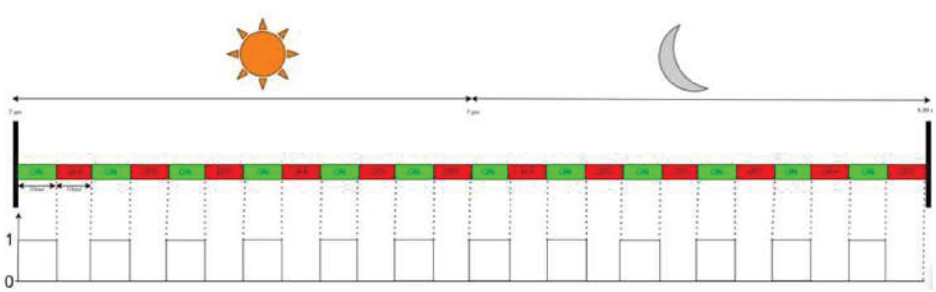


Figure 4.25 Mode 2 Switching Operation

4.4.3.1 TDS Value in Mode 2

The Table 4.7 provides an overview of the TDS (Total Dissolved Solids) values measured in Mode 2 over three consecutive days at different times. In Mode 2, the pump operates in an on/off cycle, running for one hour and turning off for the next, allowing the system to optimize water flow based on periodic control and monitoring. This schedule ensures consistent operation while conserving energy and maintaining water quality by responding to changes in TDS values.

Table 4.7 Mode 2 TDS Value Data by 3 Days

Mode 2	Day 1	Day 2	Day 3
Time	TDS Value	TDS Value	TDS Value
7.00 am	20	21	22
8.00am	22	21	23
9.00 am	23	24	25
10.00am	20	21	20
11.00 am	19	18	19
12.00am	20	21	22
1.00 pm	19	20	21
2.00 pm	19	19	18
3.00 pm	23	22	21
4.00 pm	21	22	21
5.00 pm	20	19	18
6.00 pm	19	18	19
7.00 pm	22	23	24
8.00 pm	24	25	26
9.00 pm	25	24	23
10.00 pm	23	22	21
11.00 pm	22	23	22
12.00 pm	21	20	21

Day 1: The TDS values show fluctuations throughout the day, starting at 20 ppm at 7:00 am, peaking at 25 ppm at 9:00 pm, and then gradually decreasing. There are noticeable dips during late morning and early afternoon (e.g., 19 ppm at 11:00 am), indicating lower

dissolved solids, potentially due to system adjustments or reduced water usage during these hours.

Day 2: The values generally follow a similar trend to Day 1 but exhibit slight variations. For example, the TDS peaks occur earlier in the evening (e.g., 24 ppm at 8:00 pm) and are slightly higher during the day compared to Day 1. The values return to a balanced range (around 20–23 ppm) by midnight, suggesting stable water conditions.

Day 3: TDS levels remain relatively higher compared to the first two days, peaking at 26 ppm at 8:00 pm and maintaining a steady range in the late evening. The fluctuations during the daytime (e.g., 19–25 ppm) reflect the natural variations in water quality that the pump system addresses. It is shown in Figure 4.26 Mode 2 TDS Value Reading.

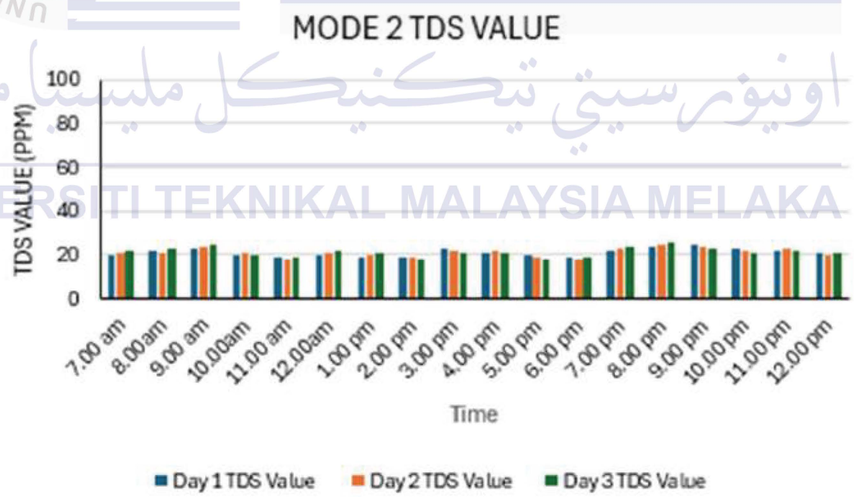


Figure 4.26 Mode 2 TDS Value Reading

4.4.3.2 Power Consumption in Mode 2

The input time settings in Table 4.8 indicate that the system is set to operate in 1-hour intervals throughout the entire 24-hour period. The output, shown as 1 hour for each time slot, suggests that the system is configured to function for 1 hour at each scheduled time. This consistent operation schedule ensures that the system will be active at regular intervals, promoting a predictable and stable operation cycle. With this setup, the system is likely designed to provide controlled, periodic functionality, which can help with energy management and efficiency over the course of the day and night. The system will repeat this 1-hour operation every day, maintaining the same pattern.

Table 4.8 Mode 2 Power Consumption

Condition	Input (Time Setting)	Output	Power Consumption
1	7.00 am	1hour	5Wh
2	9.00 am	1hour	5Wh
3	11.00 am	1hour	5Wh
4	1.00 pm	1hour	5Wh
5	3.00 pm	1hour	5Wh
6	5.00 pm	1hour	5Wh
7	7.00 pm	1hour	5Wh
8	9.00 pm	1hour	5Wh
9	11.00 pm	1hour	5Wh
10	1.00 am	1hour	5Wh
11	3.00 am	1hour	5Wh
12	5.00 am	1hour	5Wh
		Total per day	60Wh

The power consumption as Figure 4.27 data for Mode 2 on Day 1 shows a cyclic pattern of power usage, where the system consumes 5 units of power during specific hours and uses 0 units during others. The system operates for 1 hour at regular intervals throughout the day, alternating between 5 units of power consumption and no power consumption. This indicates that the pump or system is active for 1 hour at scheduled times and inactive during the following hours.

The alternating pattern of 5 units of power followed by 0 units suggests that the system is operating in a way that optimizes energy usage, only consuming power during its active phases. After 24 hours, the total power consumption will be 60 units (5 units x 12 active hours). This strategy allows for controlled, energy-efficient operation, ensuring that the system only draws power when necessary while maintaining a consistent operation cycle.

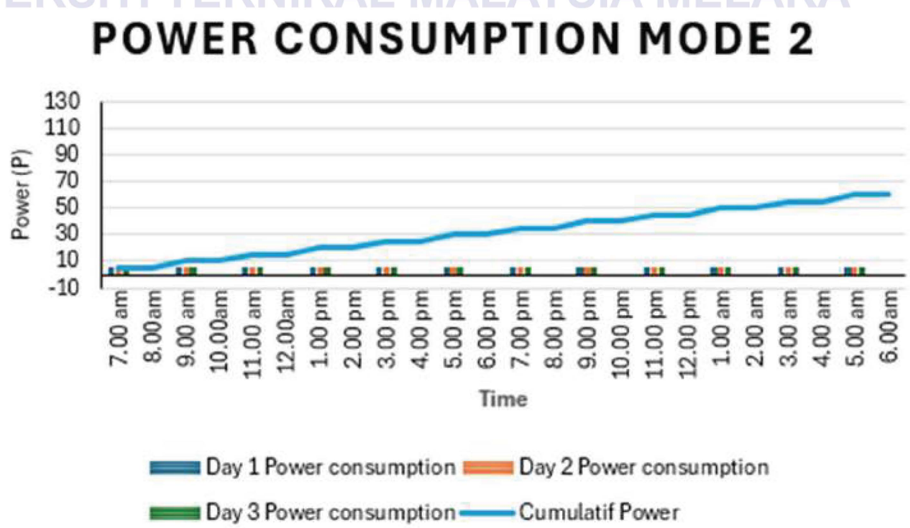


Figure 4.27 Power Consumption In Mode 2 Graph

Table 4.9 Battery Percentages Mode 2

Time	Power Consumption (W)	Cumulative Power (Wh)	Battery Percentage Remaining (%)
7:00 AM	5	5	90.74%
8:00 AM	0	5	90.74%
9:00 AM	5	10	81.48%
10:00 AM	0	10	81.48%
11:00 AM	5	15	72.22%
12:00 PM	0	15	72.22%
1:00 PM	5	20	62.96%
2:00 PM	0	20	62.96%
3:00 PM	5	25	53.70%
4:00 PM	0	25	53.70%
5:00 PM	5	30	44.44%
6:00 PM	0	30	44.44%
7:00 PM	5	35	35.19%
8:00 PM	0	35	35.19%
9:00 PM	5	40	25.93%
10:00 PM	0	40	25.93%
11:00 PM	5	45	16.67%
12:00 AM	0	45	16.67%
1:00 AM	5	50	7.41%
2:00 AM	0	50	7.41%

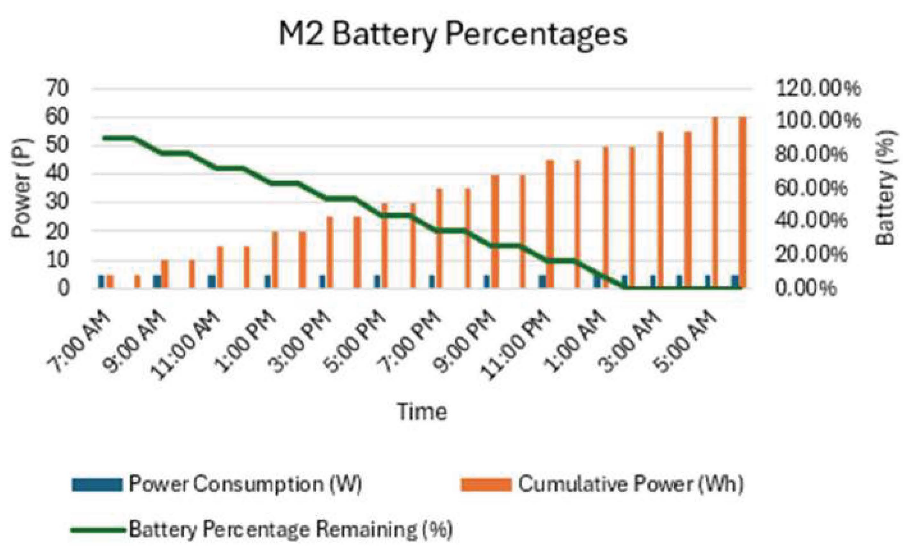


Figure 4.28 Battery Percentages Mode 2 Graph

The analysis of Mode 2 in Table 4.9 reveals that using a 12V 4.5Ah battery, which has an energy capacity of 54Wh, is insufficient to sustain the pump's operation for a full 24-hour cycle. The pump, consuming 5W/hour, depletes the battery completely by 3:00 AM, providing a total power consumption of 55Wh before the battery reaches 0%. The battery's percentage declines steadily due to the constant power draw, with 35.19% remaining at 7:00 PM, 16.67% at 11:00 PM, and only 7.41% at 1:00 AM. This Figure 4.28 indicates that the current battery cannot meet the pump's daily power requirement of 60Wh. To ensure uninterrupted operation, a higher-capacity battery, such as a 12V 7.2Ah battery (86.4Wh), is recommended. Alternatively, optimization strategies, such as reducing the pump's operating hours or incorporating a solar charging system, can be employed to enhance the system's efficiency and ensure reliable operation throughout the day.

4.4.4 Mode 3 Analysis

In Figure 4.29 Mode 3 Switching Operation, the pump operates based on sensor values in specific ranges, adjusting its run time accordingly. If the sensor value is between 0 and 30, the pump runs for 30 minutes. For values between 31 and 60, the pump runs for 60 minutes, and when the value is between 61 and 100, the pump operates for 90 minutes. This ensures that the pump's operation is proportional to the sensor readings, optimizing its efficiency. A switching graph can be used to represent this logic, where a "1" indicates the pump is on and a "0" means the pump is off. For each time interval, the graph would display a sequence of 1s for the duration the pump is active and 0s when it's inactive, allowing for a clear visualization of the pump's operation in real-time.

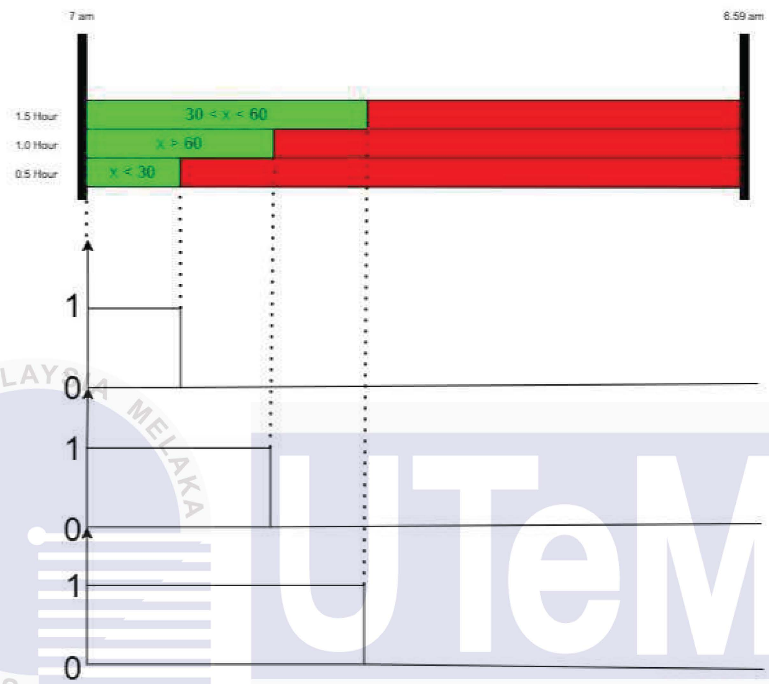


Figure 4.29 Mode 3 Switching Operation

4.4.4.1 Power Consumption in Mode 3

The pump's energy consumption is determined by its operation time as in Figure 4.30 based on sensor values. If the sensor value is less than 30, the pump runs for 0.5 hours, consuming 2.5Wh of energy. For values between 30 and 60, the pump operates for 1 hour, using 5Wh. When the sensor value exceeds 60, the pump runs for 1.5 hours and consumes 7.5Wh. Based on these patterns in Table 4.10Table 4.10 Power Consumption Mode 3, the total predicted energy consumption per day is 15Wh. This prediction assumes the pump runs within these ranges throughout the day, with each operation period contributing to the total energy usage based on the duration and energy consumption per hour.

Table 4.10 Power Consumption Mode 3

Condition	Input (sensor)	Output	Power Consumption
1	Less than 30	0.5 hour	2.5Wh
2	More than 30, below than 60	1 hour	5Wh
3	More than 60	1.5 hour	7.5Wh
		Total per day (prediction)	15Wh

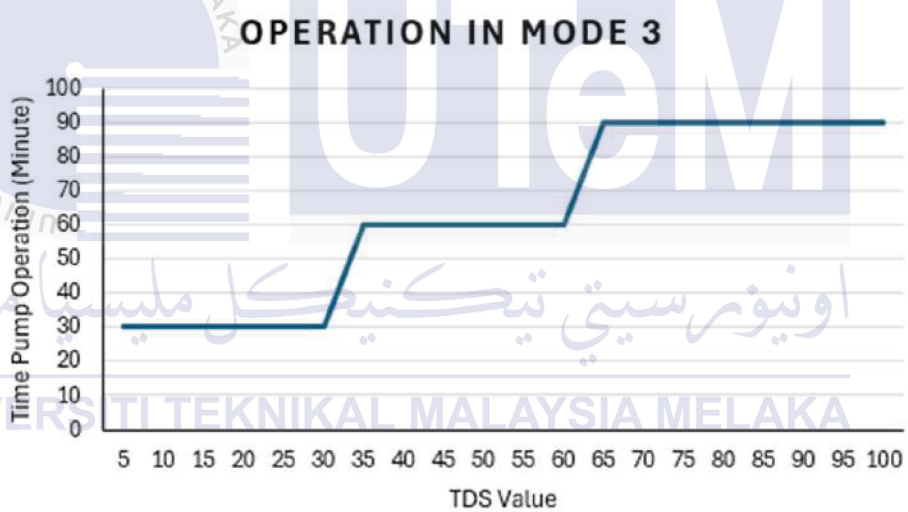


Figure 4.30 Mode 3 Operation Time

4.4.5 Mode 4 Analysis

The Figure 4.31 show pump's operation schedule, fish feeding times are integrated into the day's routine. The pump operates based on the sensor values, turning on when the value is 3.75, and off when the value is 0, as indicated by the switching graph with "1" (green) for on and "0" (red) for off. The feeding schedule for the fish occurs at specific times: 8:00 AM, 2:00 PM, 8:00 PM, and 1:00 AM. During these feeding times, the pump is on to ensure proper circulation or other related tasks for the fish, which aligns with the times when the value reaches 3.75. Therefore, the switching graph will show green at 8:00 AM, 2:00 PM, 8:00 PM, and 1:00 AM, representing both the pump's operation and the feeding times for the fish. At all other times, when the value is 0, the graph will show red, indicating that both the pump and feeding activity are off. This setup ensures that the pump operates in sync with the feeding schedule, providing a clear and efficient system for managing both tasks.

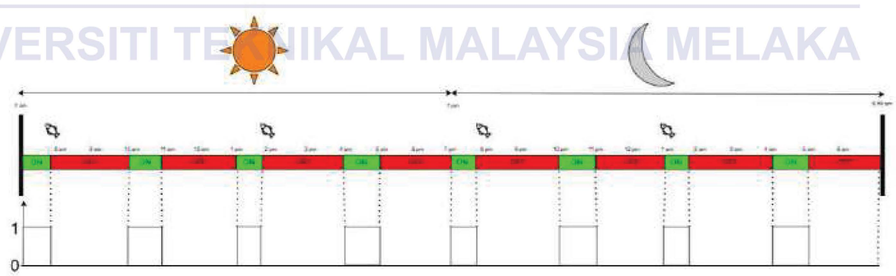


Figure 4.31 Mode 4 Switching Operation

4.4.5.1 TDS Value in Mode 4

The

Table 4.11 TDS Value Mode 4 provides an overview of TDS (Total Dissolved Solids) values in Mode 4 over three days, captured hourly from 7:00 am to 12:00 am (midnight). In Mode 4, the pump operates based on specific TDS thresholds, activating only when TDS levels exceed a predefined value. This mode is designed for dynamic water quality management, ensuring that the pump operates only when necessary, maximizing energy efficiency and maintaining optimal TDS levels.

Table 4.11 TDS Value Mode 4

Mode 4	Day 1	Day 2	Day 3
Time	TDS Value	TDS Value	TDS Value
7.00 am	22	24	25
8.00am	23	37	38
9.00 am	30	34	42
10.00am	32	20	19
11.00 am	20	22	21
12.00am	32	26	27
1.00 pm	32	23	27
2.00 pm	30	35	32
3.00 pm	20	33	37
4.00 pm	34	30	36
5.00 pm	22	31	32
6.00 pm	30	31	32
7.00 pm	22	23	24
8.00 pm	42	25	39
9.00 pm	35	39	40
10.00 pm	35	38	37
11.00 pm	29	30	31
12.00 pm	27	28	29

Figure 4.32 TSD Value Mode 4 Graph, on Day 1, the TDS values show considerable variation throughout the day, starting at 22 ppm at 7:00 am and peaking at 42 ppm at 8:00 pm. The values are relatively stable in the morning, with moderate increases reaching 30 ppm at 9:00 am and 32 ppm at 10:00 am. A significant dip is observed at 11:00 am (20 ppm), suggesting improved water quality during this period. By the afternoon, TDS levels rise again, fluctuating between 30 ppm and 34 ppm, before reaching the day's peak in the evening. Afterward, the values begin to stabilize, dropping to 29 ppm at 11:00 pm and ending the day at 27 ppm at midnight. This variability highlights the need for the pump to adjust dynamically based on real-time TDS levels.

Day 2 shows greater variability compared to Day 1, with TDS values starting slightly higher at 24 ppm at 7:00 am and rising sharply to 37 ppm at 8:00 am. The levels remain elevated throughout the morning, peaking at 39 ppm at 9:00 pm. Interestingly, there is a notable dip at 10:00 am (20 ppm) and 11:00 am (22 ppm), mirroring the stabilization pattern observed on Day 1. Afternoon values exhibit moderate fluctuations, ranging from 23 ppm at 1:00 pm to 35 ppm at 2:00 pm, followed by a brief decline. By the evening, TDS values climb again, reaching 39 ppm at 9:00 pm, before gradually decreasing to 30 ppm at 11:00 pm and 28 ppm at midnight. These patterns indicate dynamic water quality changes requiring consistent pump intervention.

Day 3 exhibits the highest overall TDS levels among the three days. It begins at 25 ppm at 7:00 am, with a steep rise to 42 ppm by 9:00 am, the highest reading of the day. Unlike the previous days, the late morning hours show relatively stable values, such as 20 ppm at 10:00 am and 21 ppm at 11:00 am, suggesting improved water quality. In the afternoon, TDS levels fluctuate significantly, with peaks of 37 ppm at 3:00 pm and 36 ppm at 4:00 pm, before stabilizing at 32 ppm in the late afternoon and evening. Evening readings, such as 39 ppm at 8:00 pm, remain consistently high but start to decline toward the end of the day, ending at 29 ppm at midnight. The higher TDS levels observed throughout Day 3 reflect increasing dissolved solids in the water, necessitating frequent pump operation.

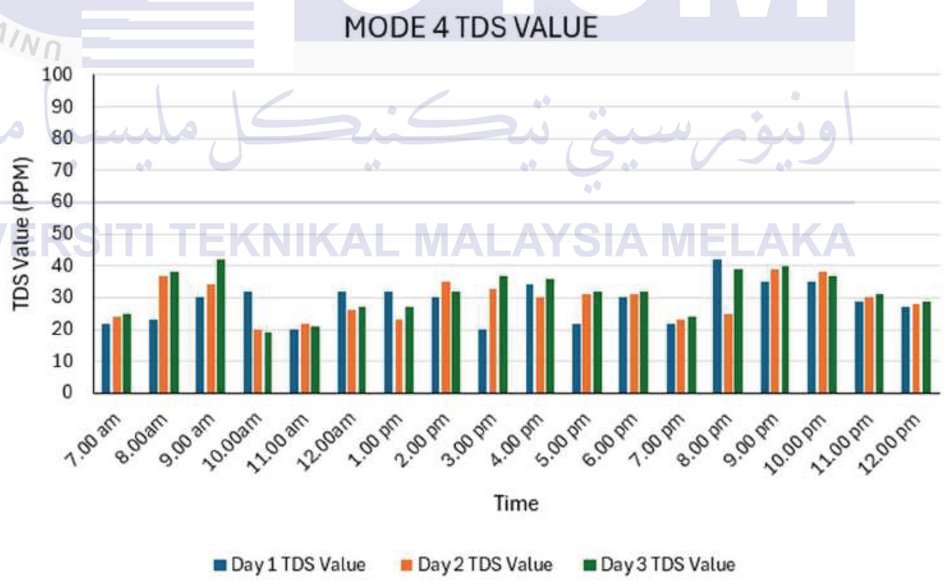


Figure 4.32 TSD Value Mode 4 Graph

4.4.5.2 Power Consumption in Mode 4

Table 4.12 show the system operates based on specific conditions related to the Time, TDS (Total Dissolved Solids) levels, and the corresponding actions taken. When the TDS is less than 30, the system operates for a shorter duration, consuming less power. As the TDS increases to a range between 31 and 60, the operation duration and power consumption increase accordingly. For TDS values greater than 61, the system runs for a longer period, resulting in higher power consumption. These conditions ensure that the system operates efficiently, with power consumption directly tied to the duration and level of activity based on the TDS readings. The overall power consumption for each range is calculated to ensure optimal operation, with energy usage proportional to the sensor readings and time intervals. It is shown in Figure 4.33Power Consumption In Mode 4 Graph.

Table 4.12 Operating Data in Mode 4

Condition	Time	TDS			Action	Power Consumption
		<30	31-60	>61		
1	7.00 am	1	0	0	Pump On,15	3.75Wh
2	7.00 am	0	1	0	Pump On,30	
3	7.00 am	0	0	1	Pump On,45	
4	10.00 am	1	0	0	Pump On,15	3.75Wh
5	10.00 am	0	1	0	Pump On,30	
6	10.00 am	0	0	1	Pump On,45	
7	1.00 pm	1	0	0	Pump On,15	3.75Wh
8	1.00 pm	0	1	0	Pump On,30	
9	1.00 pm	0	0	1	Pump On,45	
10	3.00 pm	1	0	0	Pump On,15	3.75Wh
11	3.00 pm	0	1	0	Pump On,30	
12	3.00 pm	0	0	1	Pump On,45	
13	7.00 pm	1	0	0	Pump On,15	3.75Wh
14	7.00 pm	0	1	0	Pump On,30	
15	7.00 pm	0	0	1	Pump On,45	
16	10.00 pm	1	0	0	Pump On,15	3.75Wh
17	10.00 pm	0	1	0	Pump On,30	
18	10.00 pm	0	0	1	Pump On,45	
19	1.00 am	1	0	0	Pump On,15	3.75Wh
20	1.00 am	0	1	0	Pump On,30	
21	1.00 am	0	0	1	Pump On,45	
22	4.00 am	1	0	0	Pump On,15	3.75Wh
23	4.00 pm	0	1	0	Pump On,30	
24	4.00 pm	0	0	1	Pump On,45	
					Total per day	30 Wh

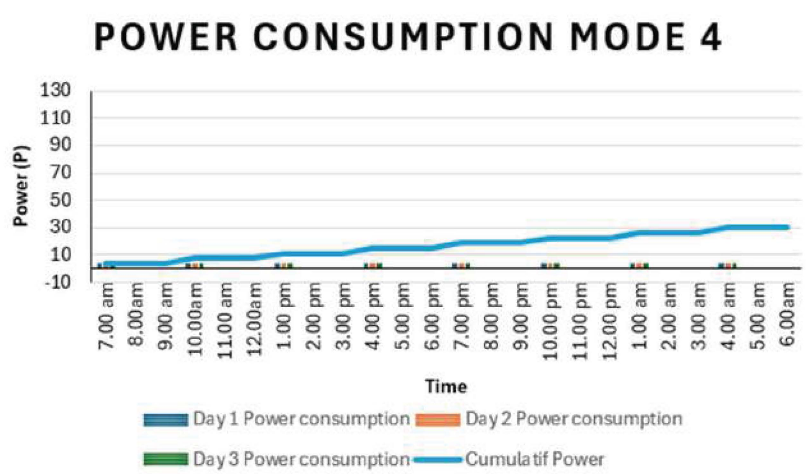


Figure 4.33Power Consumption In Mode 4 Graph

Table 4.13Battery Percentages Mode 4 Data

Time	Power Consumption (W)	Cumulative Power (Wh)	Battery Percentage Remaining
7:00 AM	3.75	3.75	93.06%
8:00 AM	0	3.75	93.06%
9:00 AM	0	3.75	93.06%
10:00 AM	3.75	7.5	86.11%
11:00 AM	0	7.5	86.11%
12:00 PM	0	7.5	86.11%
1:00 PM	3.75	11.25	79.17%
2:00 PM	0	11.25	79.17%
3:00 PM	0	11.25	79.17%
4:00 PM	3.75	15	72.22%
5:00 PM	0	15	72.22%
6:00 PM	0	15	72.22%
7:00 PM	3.75	18.75	65.28%
8:00 PM	0	18.75	65.28%
9:00 PM	0	18.75	65.28%
10:00 PM	3.75	22.5	58.33%
11:00 PM	0	22.5	58.33%
12:00 AM	0	22.5	58.33%
1:00 AM	3.75	26.25	51.39%
2:00 AM	0	26.25	51.39%
3:00 AM	0	26.25	51.39%
4:00 AM	3.75	30	44.44%
5:00 AM	0	30	44.44%
6:00 AM	0	30	44.44%

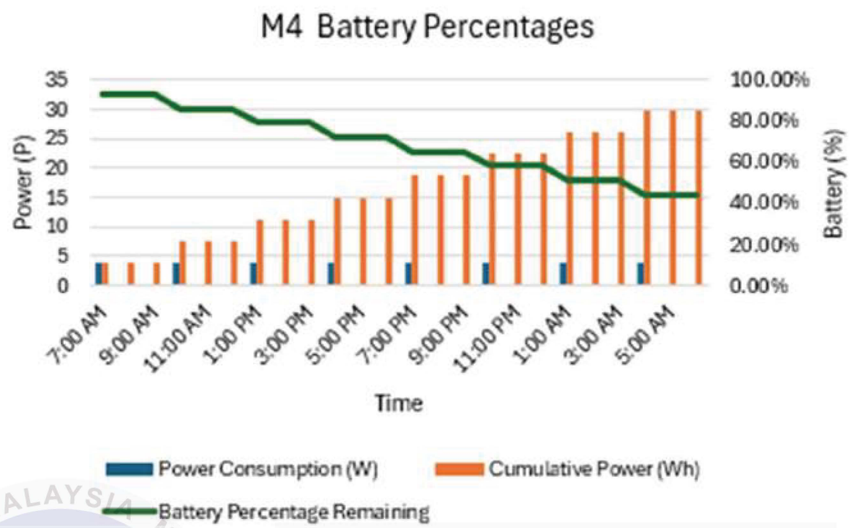


Figure 4.34 Battery Percentages Mode 4 Graph

In Figure 4.34 Battery Percentages Mode 4 Graph, the pump operates intermittently throughout the day, consuming 3.75W during its active periods. Starting at 7:00 AM with a fully charged 12V 4.5Ah battery (54Wh), the cumulative power consumption gradually increases, reaching 30Wh by 6:00 AM the next day. This results in a remaining battery percentage of 44.44% as in Table 4.13Battery Percentages Mode 4 Data. The intermittent operation of the pump allows for efficient power usage, preserving battery capacity during inactive periods. However, if the pattern continues without recharging, the battery would be depleted within two days of similar operation. This highlights the importance of incorporating recharge cycles or extending the battery capacity for prolonged usage in Mode 4. The mode demonstrates a balance between operational efficiency and power conservation, making it suitable for applications requiring periodic activation based on specific conditions.

4.4.6 Overal Analysis

Table 4.14 Summary of Input and Output Operation and Figure 4.35 Input Output Summary Ilustration. When C1 = 0 and C2 = 0, the pump operates continuously, ignoring both time and sensor inputs. In the case of C1 = 0 and C2 = 1, the pump runs only when triggered by sensor input, such as reaching a specific TDS threshold. If C1 = 1 and C2 = 0, the pump follows a predefined schedule and operates independently of sensor readings. Finally, when both C1 and C2 are set to 1, the pump optimizes its operation based on both time and sensor conditions, ensuring efficient and context-specific performance.

Table 4.14 Summary of Input and Output Operation

Mode	C1	C2	O1
1	0	0	Pump run all the time
2	0	1	Operate on reading manually based on C2
3	1	0	Operate on Timer manually based on C1
4	1	1	Optimize because base on C1 & C2

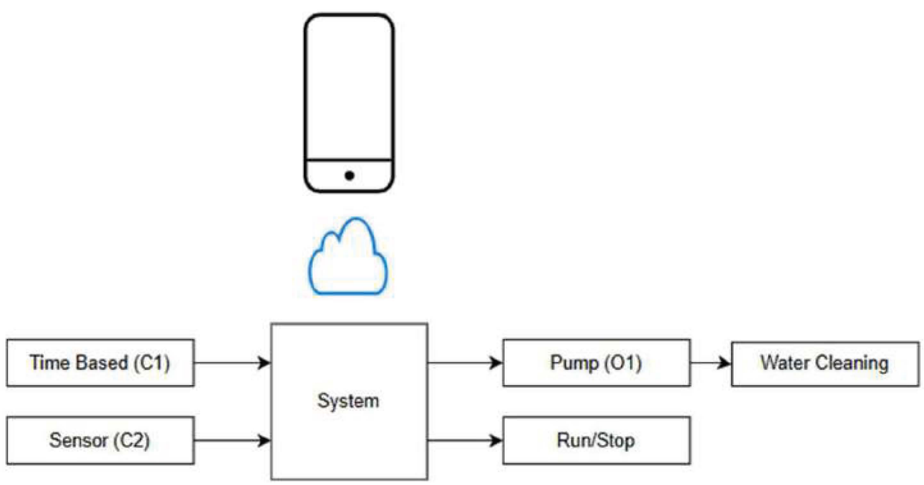


Figure 4.35 Input Output Summary Ilustration

4.4.7 Analysis of Power Consumption Comparison Across Modes

The Figure 4.36 illustrates the battery percentage over time for three operational modes: Mode 1, Mode 2, and Mode 4. The following analysis highlights the differences in power consumption patterns.

Mode 1 shows the steepest decline in battery percentage, indicating the highest rate of power consumption. The battery depletes entirely by approximately 9:00 PM, suggesting that continuous operation puts significant demand on the power source. This mode is likely the least efficient in terms of energy usage.

Mode 2 demonstrates a slower depletion rate compared to Mode 1. The battery lasts until around 3:00 AM, reflecting moderate power consumption. This mode balances operational functionality with improved energy efficiency, as the pump operates intermittently rather than continuously.

Mode 4 exhibits the most efficient power consumption, with the battery maintaining a relatively steady decline and retaining a significant charge even at 6:00 AM. This suggests that the TDS-based control optimizes energy usage by activating the pump only when specific conditions are met, minimizing unnecessary power drain.

The comparison highlights that Mode 4 is the most energy-efficient, making it ideal for applications where battery longevity is crucial. Mode 2 provides a balance between functionality and energy efficiency, while Mode 1, due to its continuous operation, consumes power rapidly and is less suitable for scenarios requiring prolonged use.

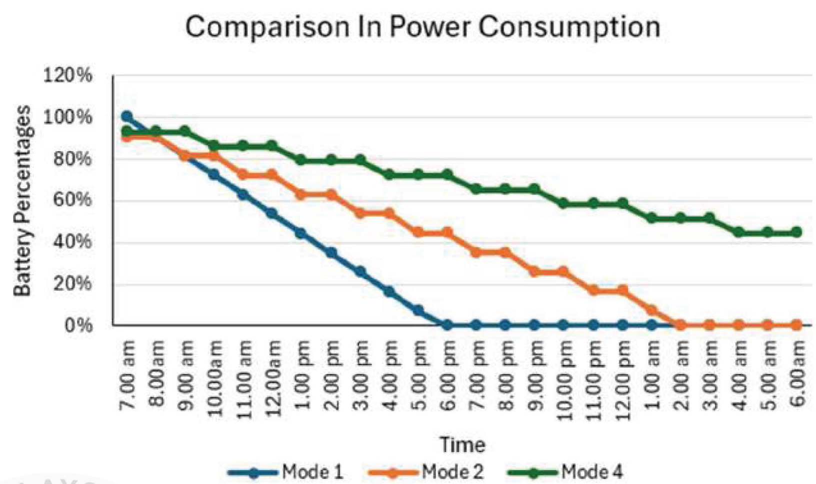


Figure 4.36 Comparison Across Mode Graph

4.5 Summary

The development of the pump control system followed a structured and methodical approach, divided into three distinct phases: hardware design and implementation, software development and integration, and performance analysis and evaluation. Each phase contributed significantly to the system's overall functionality, ensuring its efficiency, adaptability, and reliability in managing water quality and optimizing energy consumption. The project's ultimate aim was to provide an energy-efficient, automated solution for dynamic water management in various conditions, with a particular emphasis on sustainability.

The first phase of the project focused on the hardware design and assembly, which provided the foundation for the entire system. Key hardware components, such as the water pump, sensors for measuring Total Dissolved Solids (TDS), solar panels for energy harvesting, and a battery for energy storage, were integrated into the design to ensure the system's robustness and efficiency. The pump selected was capable of handling both continuous and intermittent operations, which was essential for the flexibility required in the system's four operational modes. The energy harvesting system was powered by solar panels, which were carefully sized to meet the energy consumption needs of the system under typical sunlight conditions. The battery was included to allow for uninterrupted operation during periods of low sunlight or when the energy demands exceeded the solar generation. The TDS sensor played a critical role in monitoring water quality, providing real-time data that would guide the pump's operational decisions in more advanced modes. Careful attention was paid to the interconnections between the components, ensuring that communication between the sensors, pump, and control unit was both reliable and efficient.

The second phase of the project was centered around the software development and integration, where the logic and intelligence of the system were built. The software was responsible for controlling the pump's operation across the four modes, allowing for flexible and efficient management of water flow and quality. The four modes included: Mode 1 (continuous pump operation), Mode 2 (hourly cycling for energy conservation), Mode 3 (dynamic control based on real-time TDS readings), and Mode 4 (scheduled operation tied to TDS thresholds and feeding times). A microcontroller was programmed to manage the operation, integrating sensor data and initiating appropriate actions based on real-time inputs. Algorithms for decision-making were developed to enable dynamic adjustments of the pump's operation in response to the TDS readings. Modes 3 and 4 were particularly sophisticated, with TDS-based decision algorithms allowing for adaptive pump operation that ensured optimal water quality with minimal energy use.

Additionally, the software monitored solar charging and battery status, allowing the system to adapt its operation depending on energy availability. For example, the software could adjust the frequency of pump operation to prevent battery depletion during low-sunlight periods. A user interface was created using MIT App Inventor, which allowed for remote monitoring and control of the system. This interface provided users with the ability to switch between operational modes, view system status, and customize various settings to meet their specific needs. This phase thus provided the software intelligence that allowed the system to function autonomously, with real-time adaptability to changing conditions.

The final phase of the project was dedicated to evaluating the system's performance through comprehensive analysis and testing. This phase provided valuable insights into the system's energy consumption, water quality management capabilities, and overall operational efficiency under different conditions. Data was collected for each mode, enabling a comparison of energy usage, pump operation frequency, and TDS control performance.

Mode 1 was tested for continuous pump operation, proving reliable but energy-intensive. This mode quickly drained the battery, exceeding the capacity of a standard 12V, 4.5Ah battery. Mode 2, which operated on an hourly cycling schedule, was more energy-efficient and reduced power consumption by approximately 50%. However, it required careful scheduling to balance water flow needs effectively. Mode 3, with its dynamic control based on real-time TDS readings, was the most energy-efficient, using only around 15Wh of power per day. This mode successfully minimized energy consumption while ensuring effective water quality management, demonstrating its advantage in sustainable operations. Mode 4, which synchronized operation with feeding schedules and TDS thresholds, also proved to be highly efficient. This mode provided an optimal balance between energy conservation and the need for time-sensitive operations, catering to specific requirements while achieving similar efficiency to Mode 3.

The solar charging performance was also evaluated, revealing that optimal charging occurred between 11:00 am and 2:00 pm. However, it was noted that the battery size would need to be increased or supplemented with additional energy sources to support continuous operations in Modes 1 and 2. The analysis highlighted the adaptability and energy-saving potential of Modes 3 and 4, particularly in scenarios requiring dynamic water quality control, which made them the preferred choices for energy-efficient operation.

Mode 4 stood out as the most energy-efficient and effective mode, striking an optimal balance between maintaining water quality and minimizing energy consumption. This mode utilized real-time TDS measurements and scheduled operations, which ensured the pump operated only when needed, thus eliminating unnecessary energy use. Based on calculations, Mode 4 saved up to 80% of battery power compared to Mode 1 and 70% compared to Mode 2. Its ability to synchronize operation with feeding schedules and TDS thresholds resulted in precise control over pump activity, reducing waste and ensuring water quality. The low energy consumption of Mode 4, approximately 15Wh per day, demonstrated its compatibility with renewable energy sources such as solar panels, supporting long-term sustainability even in resource-limited environments.

In conclusion, the development of the pump control system was a successful endeavor, demonstrating the feasibility and benefits of integrating advanced control systems, real-time data analysis, and energy-efficient technologies in water management. Mode 4, in particular, set a benchmark for future systems by combining sensor-based automation with time-sensitive scheduling, providing an ideal balance between functionality, efficiency, and sustainability. The project highlights the importance of intelligent system design in achieving energy conservation and environmental stewardship in modern water management systems. Through careful design, software integration, and performance evaluation, the system has proven to be a reliable and sustainable solution for managing water quality with minimal energy use.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

This report presents the development of a sustainable and efficient water filtration and management system designed specifically for fish farming applications. The project has successfully achieved its three primary objectives: the hardware for the water filtration system powered by solar energy with an IoT-based monitoring system has been fully developed (Objective 1: 100% completion), the algorithm to control, optimize, and monitor the operation of the water filtration system has been successfully created (Objective 2: 100% completion), and the performance of the water filtration system has been thoroughly evaluated (Objective 3: 100% completion). This integrated system leverages solar energy and IoT-based monitoring technologies, making it a forward-thinking solution that addresses both environmental and operational challenges in aquaculture. By utilizing solar energy, the system ensures a renewable and cost-effective power source, significantly reducing dependence on traditional energy supplies, making it suitable for remote and off-grid locations. The incorporation of IoT technology allows for real-time monitoring and control, enabling the system to respond dynamically to changing water quality conditions, which is crucial for maintaining the health and productivity of the fish stock.

The system offers multiple operational modes, each tailored to specific needs. Mode 1 ensures continuous operation for applications requiring uninterrupted water flow, while Mode 2 employs a scheduled on/off cycle to balance energy efficiency with water quality management. Mode 3 introduces a hybrid approach, allowing manual and automated TDS-based control for dynamic adaptability. Mode 4 provides a sophisticated threshold-based operation, activating the pump only when TDS levels exceed predefined limits. These modes provide flexibility, ensuring the system can adapt to diverse scenarios and optimize resource utilization.

A key focus of the project has been on energy efficiency and performance optimization. The pump's energy consumption is carefully monitored and managed to align with the available battery capacity and solar power input. Detailed calculations of power consumption, battery usage, and cumulative operation times have informed the system's design, ensuring its reliability over extended periods. Additionally, the implementation of a binary switching mechanism simplifies the system's control logic, allowing seamless operation in each mode and ensuring ease of maintenance.

The project also emphasizes learning from past experiences and similar initiatives, applying these insights to refine the system further. Suggestions for future improvements include integrating additional sensors, such as pH and turbidity, to monitor a broader range of water quality parameters, incorporating oxygenation mechanisms to enhance dissolved oxygen levels, and using predictive analytics to optimize pump operations. Expanding the system's scalability to support multiple zones, improving its user interface, and exploring renewable energy options like solar tracking are also potential enhancements.

In conclusion, this water filtration and management system represents a significant step forward in sustainable fish farming practices. By addressing critical challenges such as energy efficiency, adaptability, and water quality management, it provides a practical, scalable, and environmentally conscious solution for the aquaculture industry. Its robust design and potential for future improvements make it a promising candidate for commercialization, offering a pathway to more sustainable and productive fish farming operations worldwide.

5.2 Potential for Commercialization

The project on weather data collection and system efficiency shows considerable promise for commercialization, thanks to its significant progress in three core areas: hardware, software, and testing. The hardware component of the system is fully functional, having successfully integrated a microcontroller to manage data, a reliable water filter to maintain the sensors, and a storage tank that ensures a steady water supply. These essential components provide the foundation for the system's performance, guaranteeing that the sensors operate smoothly and without interruptions. Additionally, the software development is 100% complete, ensuring that the system can process and communicate data effectively and accurately. This allows for seamless operation and data transfer, which is crucial for real-time monitoring and system adjustments. The testing phase has been thoroughly conducted and is fully completed, with the system having been rigorously tested in real-world conditions to ensure that it functions accurately and reliably. These advancements not only demonstrate the system's readiness but also make it an appealing solution for potential commercial use, offering a dependable and efficient means of weather data collection and environmental monitoring.

Given the project's success in hardware, software, and testing, it is well-positioned for commercialization. The system's ability to collect, store, and process data efficiently provides valuable insights, which can be leveraged across various industries such as agriculture, weather forecasting, environmental monitoring, and smart city initiatives. The use of reliable and efficient hardware and software, paired with rigorous real-world testing, ensures that the system can be scaled and adapted for diverse commercial applications. Furthermore, the focus on energy-efficient operation, such as integrating solar power for sustainability, makes the system particularly attractive to markets where energy efficiency and cost reduction are key priorities.

5.3 Future Works

For future work, the report suggests the possibility of commercializing the system and highlights the importance of ongoing improvements and advancements in this area. The project also aims to assess the performance of the water filtration system and enhance its control and monitoring features. Future work on the system could focus on incorporating oxygenation mechanisms, such as aerators, to enhance dissolved oxygen levels and overall water quality, particularly for applications like aquaculture or irrigation. The integration of advanced sensors, such as pH, turbidity, and dissolved oxygen sensors, would enable comprehensive water quality monitoring. Leveraging machine learning and predictive analytics could optimize pump operation by forecasting TDS trends based on historical data. Exploring renewable energy sources, such as solar power, could improve sustainability and reduce reliance on batteries. System scalability to support multiple pumps or zones, automated cleaning mechanisms for sensors and pipelines, and a more user-friendly app interface with real-time monitoring and control options could significantly enhance usability and efficiency. Additionally, adding more operation modes, such as seasonal or event-based schedules, would increase adaptability to varying water quality and usage patterns, making the system more robust and versatile for diverse applications. Additionally, the report emphasizes the value of reviewing similar projects to understand what has been successful and what needs improvement. This will help build on existing knowledge and make informed decisions to create a more efficient and successful project.

REFERENCES

[1] M. Ali and M. K. Paracha, "AN IOT BASED APPROACH FOR MONITORING SOLAR POWER CONSUMPTION WITH ADAFRUIT CLOUD," 2020. [Online]. Available: http://www.ijeast.com

[2] Y. Singh Parihar and Y. S. Parihar, "Internet of Things and Nodemcu A review of use of Nodemcu ESP8266 in IoT products," JETIR, 2019. [Online]. Available: www.jetir.org

[3] S. Kocer, O. Dundar, and R. Butuner, *Programmable Smart Microcontroller Cards EDITORS*. 2021. [Online]. Available: www.isres.org

[4] T. Sutikno, H. S. Purnama, A. Pamungkas, A. Fadlil, I. M. Alsofyani, and M. H. Jopri, "Internet of things-based photovoltaics parameter monitoring system using NodeMCU ESP8266," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 6, pp. 5578–5587, Dec. 2021, doi: 10.11591/ijece.v11i6.pp5578-5587.

[5] E. W. Pratama and A. Kiswantono, "Electrical Analysis Using ESP-32 Module In Realtime," *JEECS (Journal of Electrical Engineering and Computer Sciences)*, vol. 7, no. 2, pp. 1273–1284, Jan. 2023, doi: 10.54732/jeecs.v7i2.21.

[6] K. A. Moharkar, "Review on Different Microcontroller Boards Used in IoT," *Int J Res Appl Sci Eng Technol*, vol. 10, no. 1, pp. 234–242, Jan. 2022, doi: 10.22214/ijraset.2022.39778.

[7] F. Vrbančič and S. Kocijančič, "Strategy for learning microcontroller programming—a graphical or a textual start?," *Educ Inf Technol (Dordr)*, vol. 29, no. 4, pp. 5115–5137, Mar. 2024, doi: 10.1007/s10639-023-12024-9.

[8] A. Irfan Ghaffar, M. Aiman, A. Ahmad, and M. Faizal Ismail, "Raspberry Pi Basic Tutorial using Android Application for Beginner User MALAYSIA *Corresponding Author Designation," *Multidisciplinary Applied Research and Innovation*, vol. 3, no. 1, pp. 510–516, 2022, doi: 10.30880/mari.2022.03.01.061.

[9] A. Shakirovich Ismailov Zafar Botirovich Jo, "Study of arduino microcontroller board." [Online]. Available: www.openscience.uz

[10] F. Yuan, X. Zhao, H. Chen, Y. Ming, and J. Zhang, "Design and Implementation of an Indoor light supplement control system based on STM32 microcontroller," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Aug. 2020. doi: 10.1088/1742-6596/1601/2/022053.

[11] H. Guo, E. Magid, K. H. Hsia, and K. L. Su, "Development of IoT module with AI function using STM32 chip," *Journal of Robotics, Networking and Artificial Life*, vol. 7, no. 4, pp. 253–257, Mar. 2021, doi: 10.2991/jrnal.k.201215.009.

[12] P. Jacko *et al.*, "Remote IoT Education Laboratory for Microcontrollers Based on the STM32 Chips," *Sensors*, vol. 22, no. 4, Feb. 2022, doi: 10.3390/s22041440.

[13] G. Chamundeeswari, S. Srinivasan, and S. P. Bharathi, "Low cost Ecg monitoring using internet of things," *International Journal of Recent Technology and Engineering*, vol. 8, no. 3, pp. 351–354, Sep. 2019, doi: 10.35940/ijrte.C4170.098319.

[14] W. Zhou, Z. Jiang, and L. Guan, "Understanding MPU Usage in Microcontroller-based Systems in the Wild," Internet Society, Mar. 2023. doi: 10.14722/bar.2023.23007.

[15] I. Akhtar, S. Kirmani, M. Jameel, and F. Alam, "Feasibility Analysis of Solar Technology Implementation in Restructured Power Sector with Reduced Carbon Footprints," *IEEE Access*, vol. 9, pp. 30306–30320, 2021, doi: 10.1109/ACCESS.2021.3059297.

[16] L. Ahmad, N. Khordehgah, J. Malinauskaite, and H. Jouhara, "Recent advances and applications of solar photovoltaics and thermal technologies," *Energy*, vol. 207, Sep. 2020, doi: 10.1016/j.energy.2020.118254.

[17] S. Dadi Riskiono, L. Oktaviani, and F. Mulya Sari, "IMPLEMENTATION OF THE SCHOOL SOLAR PANEL SYSTEM TO SUPPORT THE AVAILABILITY OF ELECTRICITY SUPPLY AT SDN 4 MESUJI TIMUR."

[18] P. S. Kumar, R. P. S. Chandrasena, V. Ramu, G. N. Srinivas, and K. V. S. M. Babu, "Energy Management System for Small Scale Hybrid Wind Solar Battery Based Microgrid," *IEEE Access*, vol. 8, pp. 8336–8345, 2020, doi: 10.1109/ACCESS.2020.2964052.

[19] A. M. Soomar, A. Hakeem, M. Messaoudi, P. Musznicki, A. Iqbal, and S. Czapp, "Solar Photovoltaic Energy Optimization and Challenges," *Frontiers in Energy Research*, vol. 10. Frontiers Media S.A., May 30, 2022. doi: 10.3389/fenrg.2022.879985.

[20] S. Ahmad, M. Shafiullah, C. B. Ahmed, and M. Alowaifeer, "A Review of Microgrid Energy Management and Control Strategies," *IEEE Access*, vol. 11. Institute of Electrical and Electronics Engineers Inc., pp. 21729–21757, 2023. doi: 10.1109/ACCESS.2023.3248511.

[21] J. Kustija and F. Andika, "Control - Monitoring System Of Oxygen Level, Ph, Temperature And Feeding in Pond Based on Iot," *REKA ELKOMIKA: Jurnal Pengabdian kepada Masyarakat*, vol. 2, no. 1, pp. 1–10, Jul. 2021, doi: 10.26760/rekaelkomika.v2i1.1-10.

[22] E. A. Laza, I. L. Caba, M. Olan, and V. Vladut, "Biological water treatment in a recirculating aquaculture system," in *E3S Web of Conferences*, EDP Sciences, Jul. 2021. doi: 10.1051/e3sconf/202128603013.

[23] F. N. H. Karabulut, G. Höfler, N. A. Chand, and G. W. Beckermann, "Electrospun nanofibre filtration media to protect against biological or nonbiological airborne particles," *Polymers (Basel)*, vol. 13, no. 19, Oct. 2021, doi: 10.3390/polym13193257.

[24] J. Duangwongsa, T. Ungsethaphand, P. Akaboot, S. Khamjai, and S. Unankard, "Real-Time Water Quality Monitoring and Notification System for Aquaculture," in *2021 Joint 6th International Conference on Digital Arts, Media and Technology with 4th ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering, ECTI DAMT and NCON 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 9–13. doi: 10.1109/ECTIDAMTNCON51128.2021.9425744.

[25] R. Ismail, K. Shafinah, and K. Latif, "A Proposed Model of Fishpond Water Quality Measurement and Monitoring System based on Internet of Things (IoT)," in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing Ltd, Aug. 2020. doi: 10.1088/1755-1315/494/1/012016.

[26] S. Shakya, "A Self Monitoring and Analyzing System for Solar Power Station using IoT and Data Mining Algorithms," *Journal of Soft Computing Paradigm*, vol. 3, no. 2, pp. 96–109, Jun. 2021, doi: 10.36548/jscp.2021.2.004.

[27] A. T. Tamim *et al.*, "Development of IoT Based Fish Monitoring System for Aquaculture," *Intelligent Automation and Soft Computing*, vol. 32, no. 1, pp. 55–71, 2022, doi: 10.32604/IASC.2022.021559.

[28] M. P. Ntulo, P. A. Owolawi, T. Mapayi, V. Malele, G. Aiyetoro, and J. S. Ojo, "IoT-Based smart aquaponics system using arduino uno," in *International Conference on Electrical, Computer, Communications and Mechatronics Engineering, ICECCME 2021*, Institute of Electrical and Electronics Engineers Inc., Oct. 2021. doi: 10.1109/ICECCME52200.2021.9590982.

[29] M. I. Rami, H. A Majid, and N. Al-Fadhali, "Development of Monitoring System Based on Internet of Things (IoT) for Freshwater Prawn Farming," *Journal of Advanced Industrial Technology and Application*, vol. 3, no. 2, Dec. 2022, doi: 10.30880/jaita.2022.03.02.002.

[30] M. Alselek, J. M. Alcaraz-Calero, J. Segura-Garcia, and Q. Wang, "Water IoT Monitoring System for Aquaponics Health and Fishery Applications," *Sensors*, vol. 22, no. 19, Oct. 2022, doi: 10.3390/s22197679.

[31] Espressif Systems, "About This Guide," 2023. Accessed: Jun. 09, 2024. [Online]. Available: https://www.espressif.com/

[32] An Avnet Company, "A000066," 2020. Accessed: Jun. 09, 2024. [Online]. Available: https://www.digikey.com/en/products/detail/arduino/A000066/2784006

[33] DFROBOT, "Turbidity sensor SKU: SEN0189," 2017. Accessed: Jun. 09, 2024. [Online]. Available: https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189

[34] DFROBOT, "Gravity: Analog TDS Sensor / Meter For Arduino SKU: SEN0244." Accessed: Jun. 09, 2024. [Online]. Available: https://wiki.dfrobot.com/Gravity__Analog_TDS_Sensor___Meter_For_Arduino_SKU__SEN0244

[35] DFROBOT, "DC Mini Immersible Water Pump (6V~12V)," 2015. Accessed: Jun. 09, 2024. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/FIT0563_Web.pdf

APPENDICES

Appendix A Specification of ESP8266

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4 GHz ~ 2.5 GHz (2400 MHz ~ 2483.5 MHz)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5 V ~ 3.6 V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	-40 °C ~ 125 °C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

[31]

Appendix B Arduino Specification

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.**This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

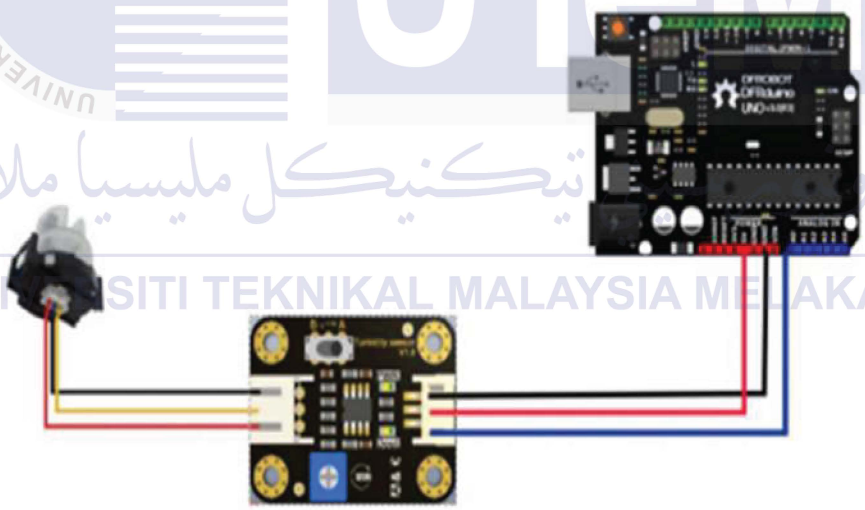
[32]

Appendix C Turbidity Sensor Specification

Specification

- Operating Voltage: 5V DC
- Operating Current: 40mA (MAX)
- Response Time : <500ms
- Insulation Resistance: 100M (Min)
- Output Method:
  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

Connection Diagram



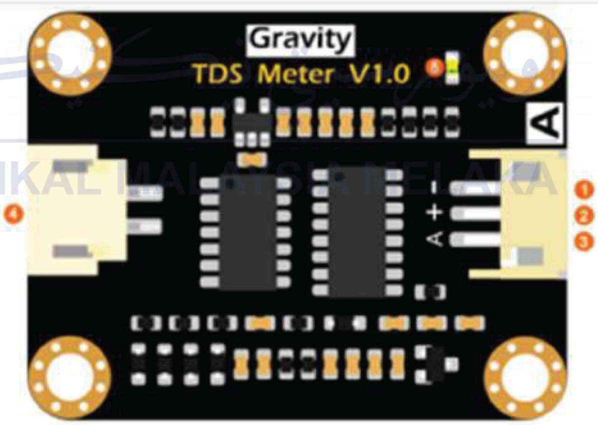
[33]

Appendix D TDS Sensor Specification

Specification

- **Signal Transmitter Board**
  Input Voltage: 3.3 ~ 5.5V
  Output Voltage: 0 ~ 2.3V
  Working Current: 3 ~ 6mA
  TDS Measurement Range: 0 ~ 1000ppm
  TDS Measurement Accuracy: ± 10% F.S. (25 ℃)
  Module Size: 42 * 32mm
  Module Interface: PH2.0-3P
  Electrode Interface: XH2.54-2P
- **TDS probe**
  Number of Needle: 2
  Total Length: 83cm
  Connection Interface: XH2.54-2P
  Colour: Black
  Other: Waterproof Probe

Board Overview



Analog TDS Sensor / Meter For Arduino

[34]

Appendix E Specification of 12V DC pump

SPECIFICATION

- Power supply: 6~12V DC,65mA-500mA
- Interface: DC 5.5-2.1
- Pumping head: 0-200cm
- Capacity: 0~550L/H
- Power range: 4~5W
- Dimensions: 45x43x30mm(1.77x1.69x1.18")
- Weight: 300g
- Cable length: 1m (39.37")

SHIPPING LIST

- Immersible Water Pump (6V~12V) x1
- water pipe x1
- DC2.1 Connector (Red: + Black: -) x1

[35]

Appendix F Coding Arduino IDE

```
#include <Arduino.h>  // Standard Arduino library
#include <ESP8266WiFi.h>  // Wi-Fi functionality for ESP8266
#include <Firebase_ESP_Client.h>  // Firebase library for ESP8266
#include "addons/TokenHelper.h"  // Token generation and debugging
#include "addons/RTDBHelper.h"  // Firebase Realtime Database helper
functions
#include <NTPClient.h>  // Library for network time synchronization
#include <WiFiUdp.h>  // Library for UDP communication, used by NTPClient

// Wi-Fi credentials
#define WIFI_SSID "Gudang Garam_2.4GHz"  // Your Wi-Fi SSID
#define WIFI_PASSWORD "Topek1234"  // Your Wi-Fi password

// Firebase credentials
#define API_KEY "AIzaSyBtCpEawu1iMM4IlJoXYhcIYPGxGf_4zzE"  // Firebase
project API key
#define DATABASE_URL "https://neww-a858d-default-rtdb.firebaseio.com/"  //
Firebase Realtime Database URL

// Firebase objects
FirebaseData fbdo;  // Firebase data object for handling requests
FirebaseAuth auth;  // Firebase authentication object
FirebaseConfig config;  // Firebase configuration object

// NTP Client for time
WiFiUDP ntpUDP;  // UDP instance for NTP
NTPClient timeClient(ntpUDP, "pool.ntp.org", 8 * 3600, 600000);  //
Initialize NTP client with time zone offset and update interval

// Pin definitions
#define TDS_PIN A0  // Analog pin for TDS sensor
#define RELAY_PIN 4  // Digital pin for relay control

// TDS sensor calibration constants
#define VREF 3.3  // Reference voltage for ADC
#define ADC_MAX 1024  // Maximum ADC value
#define K_VALUE 0.5  // Calibration constant for TDS calculation

// Variables
int mode = 1;  // Default mode set to 1
String relayStatus = "OFF";  // Initial relay status
String manualRelayStatus = "OFF";  // Initial manual relay status
String cooldownStatus = "00:00";  // Initial cooldown status

bool isMode2Active = false;  // Flag to track if Mode 2 is active
```

```
unsigned long lastActionTime = 0;  // Timestamp of the last action
unsigned long lastCooldownUpdate = 0;  // Timestamp of the last cooldown
update

int tdsValue = 0;  // TDS sensor value
bool signupOK = false;  // Flag for Firebase signup status
unsigned long lastRun = 0;  // Tracks pump runtime

int currentMode = 0;  // Current mode of operation

// Function Prototypes
void setupWiFi();  // Function to set up Wi-Fi connection
void setupFirebase();  // Function to configure Firebase
void readTDS();  // Function to read TDS sensor data
void updateFirebase();  // Function to update data on Firebase
void handleMode();  // Function to handle pump modes
void mode1Continuous();  // Mode 1: Continuous pump operation
void mode2Hourly();  // Mode 2: Hourly pump operation
void mode3TDSBased();  // Mode 3: TDS-based pump control
void mode4Optimized();  // Mode 4: Optimized pump control
void handleManualRelay();  // Function to handle manual relay operations
void updateRelayStatus(String status);  // Function to update relay status
in Firebase

void setup() {
  Serial.begin(115200);  // -------------------------------------------
---------Initialize serial communication at 115200 baud rate

  setupWiFi();  //-----------------------------------------------------
Connect to Wi-Fi
  setupFirebase();  // ------------------------------------------------
----Set up Firebase configuration

  timeClient.begin();  // ---------------------------------------------
-------Start the NTP client
  timeClient.update();  //---------------------------------------------
------- Force an immediate time update

  pinMode(RELAY_PIN, OUTPUT);  // Configure relay pin as output
  digitalWrite(RELAY_PIN, HIGH);  // ------------------------------------
------------------Ensure relay is initially off

  // Additional setup (e.g., sensor initialization) can be added here
}

void loop() {
  readTDS();  //-------------------------------------------------------
Read TDS sensor data
```

```
  updateFirebase();  // --------------------------------------------------
-----Sync data with Firebase
  handleMode();  //-------------------------------------------------------
Execute the pump control logic based on the mode

  if (!timeClient.update()) {  // ----------------------------------------
---------------Update NTP time; retry on failure
    Serial.println("Time sync failed. Forcing update...");
    ntpUDP.stop();  //----------------------------------------------------
--- Stop the UDP client
    ntpUDP.begin(123);  //------------------------------------------------
------- Restart UDP communication
    timeClient.forceUpdate();  //-----------------------------------------
-------------- Force a time update
  }

  // Print the current time
  Serial.printf("Current Time: %02d:%02d:%02d\n", timeClient.getHours(),
timeClient.getMinutes(), timeClient.getSeconds());

  delay(1000);  //-------------------------------------------------------
Delay for 1 second before the next loop iteration
}

void setupWiFi() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  // ------------------------------
-------------------------Connect to Wi-Fi network
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {  //-----------------------------
-------------------------- Wait until connected
    Serial.print(".");
    delay(300);  //-------------------------------------------------------
Retry every 300 milliseconds
  }
  Serial.println("\nConnected to Wi-Fi");
  Serial.println("IP Address: " + WiFi.localIP().toString());  // --------
-----------------------------------------------Print device IP address
}

void setupFirebase() {
  config.api_key = API_KEY;  // ------------------------------------------
-------------Set API key
  config.database_url = DATABASE_URL;  // --------------------------------
------------------------Set database URL

  if (Firebase.signUp(&config, &auth, "", "")) {  // ---------------------
----------------------------------Sign up for Firebase
    Serial.println("Firebase sign up successful");
```

```
    signupOK = true;  //--------------------------------------------------
----- Mark signup as successful
  } else {
    Serial.printf("Firebase sign up failed: %s\n",
config.signer.signupError.message.c_str());  // Print error message
  }

  config.token_status_callback = tokenStatusCallback;  // ----------------
---------------------------------------Set token status callback
  Firebase.begin(&config, &auth);  // ------------------------------------
------------------Initialize Firebase with configuration
  Firebase.reconnectWiFi(true);  // --------------------------------------
-----------------Enable automatic Wi-Fi reconnection
}

void readTDS() {
  int sensorValue = analogRead(TDS_PIN);  // -----------------------------
--------------------------Read raw ADC value from TDS sensor
  float voltage = (sensorValue / (float)ADC_MAX) * VREF;  // -------------
------------------------------------------Calculate voltage
  tdsValue = (voltage / VREF) * 1000 * K_VALUE;  //-----------------------
------------------------------- Calculate TDS value based on calibration
  Serial.printf("TDS Value: %d\n", tdsValue);  //-------------------------
----------------------------- Print the TDS value
}

void updateFirebase() {
  if (Firebase.RTDB.setInt(&fbdo, "sensor/tds", tdsValue)) {  //----------
--------------------------------------------- Update TDS value in Firebase
    Serial.println("TDS value updated to Firebase");
  } else {
    Serial.println("Failed to update TDS value to Firebase");
    Serial.println(fbdo.errorReason());  //-------------------------------
----------------------- Print error reason
  }

  if (Firebase.RTDB.getString(&fbdo, "sensor/mode")) {  //----------------
---------------------------------------- Retrieve mode from Firebase
    mode = fbdo.stringData().toInt();  //---------------------------------
--------------------- Convert retrieved string to integer
    Serial.printf("Mode updated to: %d\n", mode);
  } else {
    Serial.println("Failed to retrieve mode from Firebase");
  }

  if (Firebase.RTDB.getString(&fbdo, "sensor/manual")) {  // -------------
------------------------------------------Retrieve manual relay status
from Firebase
```

```
    manualRelayStatus = fbdo.stringData();  // ---------------------------
----------------------------Store retrieved value
    Serial.printf("Manual pump status fetched: %s\n",
manualRelayStatus.c_str());
  } else {
    Serial.println("Failed to retrieve manual pump status from Firebase");
    Serial.println(fbdo.errorReason());  // ------------------------------
-------------------------Print error reason
  }

  if (Firebase.RTDB.getString(&fbdo, "sensor/pump")) {  //----------------
--------------------------------------- Retrieve pump status from Firebase
    relayStatus = fbdo.stringData();  //----------------------------------
--------------------- Store retrieved value
    Serial.printf("Pump status fetched: %s\n", relayStatus.c_str());
  } else {
    Serial.println("Failed to retrieve pump status from Firebase");
    Serial.println(fbdo.errorReason());  //-------------------------------
------------------------ Print error reason
  }
}

void handleMode() {
  switch (mode) {  // ----------------------------------------------------
---Select operation mode based on current mode value
    case 1:
      mode1Continuous();  //----------------------------------------------
--------- Execute Mode 1 logic
      break;
    case 2:
      mode2Hourly();  // -------------------------------------------------
------Execute Mode 2 logic
      break;
    case 3:
      mode3TDSBased();  // -----------------------------------------------
------------Execute Mode 3 logic
      break;
    case 4:
      mode4Optimized();  // ----------------------------------------------
---Execute Mode 4 logic
      break;
    default:
      Serial.println("Invalid mode selected");  // -----------------------
------------------------------------------Handle invalid mode
  }
}

void mode1Continuous() {
```

```
  Serial.println("Mode 1: Continuous Operation");
  digitalWrite(RELAY_PIN, LOW);  // -------------------------------------
--------------Turn relay ON
  updateRelayStatus("ON");  // ------------------------------------------
----Update relay status in Firebase
}

void mode2Hourly() {
  if (!timeClient.update()) {  // ---------------------------------------
--------------Synchronize time; retry on failure
    Serial.println("Failed to synchronize time with NTP server.
Retrying...");
    timeClient.forceUpdate();  //----------------------------------------
---------------------- Force time update
  }

  String formattedTime = timeClient.getFormattedTime();  //--------------
------------------------------ Get formatted time string
  int hour = timeClient.getHours();  //----------------------------------
---------- Extract current hour
  int minute = timeClient.getMinutes();  //------------------------------
------------------- Extract current minute

  Serial.print("Current Time: ");
  Serial.println(formattedTime);  // ------------------------------------
------Print current time

  if (hour % 2 == 1) {  // ----------------------------------------------
---Check if hour is odd (ON state)
    digitalWrite(RELAY_PIN, LOW);  // -----------------------------------
----------------Turn relay ON
    updateRelayStatus("ON");  //-----------------------------------------
---------- Update relay status
    Serial.println("Pump ON: Running during odd hour");
  } else {
    digitalWrite(RELAY_PIN, HIGH);  //-----------------------------------
------------------- Turn relay OFF
    updateRelayStatus("OFF");  //----------------------------------------
-- Update relay status
    Serial.println("Pump OFF: Resting during even hour");
  }
}

void mode3TDSBased() {
  handleManualRelay();  //-----------------------------------------------
------ Update manual relay status from Firebase
```

```
  if (manualRelayStatus == "1") {  //------------------------------------
----------------- Check if manual relay is ON
    unsigned long pumpDuration;  // -----------------------------------
-------Variable to hold pump runtime
    if (tdsValue < 30) {
      pumpDuration = 10 * 100 * 10;  //-----------------------------------
--------- Set duration for TDS < 30
    } else if (tdsValue >= 30 && tdsValue <= 60) {
      pumpDuration = 20 * 100 * 10;  // ----------------------------------
------------------Set duration for 30 <= TDS <= 60
    } else {
      pumpDuration = 30 * 100 * 10;  //-----------------------------------
---- Set duration for TDS > 60
    }

    digitalWrite(RELAY_PIN, LOW);  // ---------------------------------
Turn relay ON
    updateRelayStatus("ON");  // ----------------------------------------
--------Update relay status
    Serial.printf("Mode 3: Pump running for %lu milliseconds\n",
pumpDuration);

    delay(pumpDuration);  // --------------------------------------------
------------Wait for the specified duration

    digitalWrite(RELAY_PIN, HIGH);  // ----------------------------------
---------------Turn relay OFF
    updateRelayStatus("OFF");  //----------------------------------------
------------------ Update relay status
    Serial.println("Mode 3: Pump operation completed");
  } else {
    Serial.println("Mode 3: Waiting for manual activation...");  // ------
---------------Wait for manual relay activation
    updateRelayStatus("OFF");  // ---------------------------------------
-----------Ensure relay status is OFF
  }
}

void mode4Optimized() {
  if (!timeClient.update()) {  // ---------------------------------------
----Synchronize time; retry on failure
    Serial.println("Failed to synchronize time with NTP server.
Retrying...");
    timeClient.forceUpdate();  // ---------------------------------------
---------------Force time update
  }
```

```
  String formattedTime = timeClient.getFormattedTime();  // --------------
-----------------------Get formatted time string
  int hour = timeClient.getHours();  //-----------------------------------
-------------------- Extract current hour
  int minute = timeClient.getMinutes();  // -----------------------------
--------------------Extract current minute

  Serial.print("Current Time: ");
  Serial.println(formattedTime);  // ------------------------------------
-------Print current time

  if ((hour == 7 && minute == 5) || (hour == 10 && minute == 5) || (hour
== 13 && minute == 5)
   || (hour == 16 && minute == 5) || (hour == 19 && minute == 5) || (hour
== 22 && minute == 5)
   || (hour == 1 && minute == 5) || (hour == 4 && minute == 5)) {  // ----
------------------------------Specific times for operation
    if (tdsValue < 30) {
      digitalWrite(RELAY_PIN, LOW);  //-----------------------------------
-------------- Turn relay ON
      updateRelayStatus("ON");  //----------------------------------------
-- Update relay status
      Serial.println("Pump ON: Running for 10 minutes");
      delay(10 * 10 * 100);  //----------------------------------------
Run for 10 seconds (simulated)
    } else if (tdsValue >= 31 && tdsValue <= 60) {
      digitalWrite(RELAY_PIN, LOW);  // ----------------------------------
-----Turn relay ON
      updateRelayStatus("ON");  // ---------------------------------------
--Update relay status
      Serial.println("Pump ON: Running for 20 minutes");
      delay(20 * 10 * 100);  // ------------------------------------------
--------Run for 20 seconds (simulated)
    } else if (tdsValue > 61) {
      digitalWrite(RELAY_PIN, LOW);  //-----------------------------------
- Turn relay ON
      updateRelayStatus("ON");  //----------------------------------------
- Update relay status
      Serial.println("Pump ON: Running for 30 minutes");
      delay(30 * 10 * 100);  //--------------------------------------- Run
for 30 seconds (simulated)
    }
  } else {
    digitalWrite(RELAY_PIN, HIGH);  // -----------------------------------
------Turn relay OFF
    updateRelayStatus("OFF");  // ----------------------------------------
----Update relay status
    Serial.println("Pump OFF");
```

```
    Serial.println("Current time not in the specified range for Mode 4
operation");
  }
}

void updateRelayStatus(String status) {
  relayStatus = status;  // -------------------------------------------
Update local relay status variable
  Serial.printf("Pump Status: %s\n", relayStatus.c_str());

  if (Firebase.RTDB.setString(&fbdo, "sensor/pump", relayStatus))
{     //---------------------------------- Update relay status in Firebase
    Serial.println("Pump status updated to Firebase");
  } else {
    Serial.println("Failed to update Pump status to Firebase");
    Serial.println(fbdo.errorReason());   // ---------------------------
------------------Print error reason
  }
}

void handleManualRelay() {
  if (Firebase.RTDB.getString(&fbdo, "sensor/manual"))
{                                  // Retrieve manual relay status from
Firebase
    manualRelayStatus =
fbdo.stringData();                                              // Store
retrieved value
    Serial.printf("Manual Pump status fetched: %s\n",
manualRelayStatus.c_str());
  } else {
    Serial.println("Failed to retrieve manual Pump status from Firebase");
    Serial.println(fbdo.errorReason());
                // Print error reason
  }
}
```