

FLOATING AQUACULTURE SENSOR STATION

Yong Xin Yi

UNIVERSITI TUNKU ABDUL RAHMAN

FLOATING AQUACULTURE SENSOR STATION

Yong Xin Yi


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Honours) Mechatronic Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

May 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  _____

Name : Yong Xin Yi _____


ID No. : 1903478 _____

Date : 2 May 2023 _____

APPROVAL FOR SUBMISSION


I certify that this project report entitled “**Floating Aquaculture Sensor Station**” was prepared by **Yong Xin Yi** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Ng Oon-Ee

Date : 18 May 2023

Signature : 

Co-Supervisor : Mun Hou Kit

Date : 18 May 2023

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2023, Yong Xin Yi. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Ng Oon Ee and Ir. Dr. Mun Hou Kit for their invaluable advice, guidance and their enormous patience throughout the development of the research.

ABSTRACT

Aquaculture is the production of aquatic organisms under optimal conditions, and it faces challenges due to aquatic organisms being sensitive to changes in their environment. Water quality monitoring is essential to ensure consistent survival and reproduction of aquatic organisms. Traditional manual methods are ineffective and inefficient, leading to limited sampling frequencies and human error. To prevent dangerous situations from occurring, a systematic water quality monitoring system is required. An IoT-enabled water quality monitoring system can gather real-time data and process it through cloud computing to provide the end-user with accessible information. This study developed a floating aquaculture sensor station that can collect real-time water quality parameters data and transmit it wirelessly to a designated server. The developed prototype provides accurate and accessible information of real-time water quality parameters, leading to more efficient and sustainable practices. The prototype is able to deploy and retract the sensor effectively to prevent the sensors damage from submerging in the water for too long. The developed prototype is self-sustainable in terms of power by harvesting solar energy.

TABLE OF CONTENTS

DECLARATION		i
APPROVAL FOR SUBMISSION		ii
ACKNOWLEDGEMENTS		iv
ABSTRACT		v
TABLE OF CONTENTS		vi
LIST OF TABLES		ix
LIST OF FIGURES		x
LIST OF SYMBOLS / ABBREVIATIONS		xiii
LIST OF APPENDICES		xv
CHAPTER		
1	INTRODUCTION	1
1.1	General Introduction	1
1.2	Importance of the Study	2
1.3	Problem Statement	3
1.4	Aim and Objectives	4
1.5	Scope and Limitation of the Study	4
1.6	Contribution of the Study	5
1.7	Outline of the Report	5
2	LITERATURE REVIEW	6
2.1	Introduction	6
2.2	IOT based aquaculture monitoring system	9
2.3	Water Quality Parameters and Sensors	12
2.3.1	pH 13	
2.3.2	Dissolved Oxygen	14
2.3.3	Temperature	16
2.4	Summary	16
3	METHODOLOGY AND WORK PLAN	18
3.1	Project Planning and Milestone	18

3.2	Gantt chart and Log book	20
3.3	Block diagram of floating aquaculture sensor station	25
3.4	Conceptual designs of prototype	27
	3.4.1 Sensor Reel and Junction Box	28
	3.4.2 Motor station and potentiometer holder	33
	3.4.3 Reel coupler and potentiometer	37
	3.4.4 Base structure with floating pillars and solar panel	44
3.5	Prototype electronic components	45
	3.5.1 Microcontroller	45
	3.5.2 DC motor	45
	3.5.3 Motor Driver	47
	3.5.4 Power supply	47
	3.5.5 Real Time Clock (RTC)	48
	3.5.6 Communication Device	48
	3.5.7 Temperature Sensor	49
	3.5.8 Potentiometer	50
	3.5.9 Buck converter/ current voltage regulator	51
	3.5.10 Solar Panel	52
	3.5.11 Solar Charging Controller	53
3.6	Operation flowchart	54
3.7	Software	56
	3.7.1 CAD drawing	56
	3.7.2 IDE	56
	3.7.3 Circuit design	57
	3.7.4 Data visualization	58
3.8	Summary	58
4	RESULTS AND DISCUSSION	60
4.1	Modified and improved prototype	60
	4.1.1 Sensor Reel and Junction Box	61
	4.1.2 Motor station assembly with potentiometer control	64

4.1.3	Base structure with floating pillars and solar panel	66
4.2	Operation flow and code explanation	67
4.2.1	Initialisation	67
4.2.2	Main loop	68
4.2.3	Alarm.cpp	69
4.2.4	Motor motion code	70
4.2.5	Sensor cycle	72
4.2.6	Send data to server code	72
4.2.7	Arduino sleep	73
4.3	Circuit design and schematic	75
4.4	Prototype deployment	76
4.5	Data visualisation	77
4.6	Overall performance of improved prototype	78
5	CONCLUSIONS AND RECOMMENDATIONS	81
5.1	Conclusions	81
5.2	Recommendations for future work	81
	REFERENCES	83
	APPENDICES	86

LIST OF TABLES

Table 2.1: China's fishery water quality standards.	13
Table 4.1 Challenges faced and solutions.	79

LIST OF FIGURES

Figure 1.1: Types of aquaculture operations.	1
Figure 1.2: Trend of World Capture Fisheries and Aquaculture Production (FAO, 2020).	2
Figure 2.1: Architecture concept of a wireless sensor network for aquaculture WQM (Simbeye et al., 2014).	6
Figure 2.2: Timeline of Concepts (Manrique et al., 2016).	7
Figure 2.3: Implementation of WSN into the IoT.	8
Figure 2.4. The Proposed Monitoring Architecture.	9
Figure 2.5: Fish Pond Monitoring System (Hairol et al., 2018).	10
Figure 2.6: IoT-based Monitoring System (Niswar et al., 2018).	11
Figure 2.7: IoT-based Aquaculture Monitoring and Control System Architecture (Rosaline and Sathyalakshimi, 2019).	11
Figure 3.1 Project Flowchart.	18
Figure 3.2 Gantt chart.	20
Figure 3.3 Logbook.	22
Figure 3.4 Gantt chart.	23
Figure 3.5 Logbook.	24
Figure 3.6 Project block diagram.	25
Figure 3.7 Initial design of prototype.	27
Figure 3.8 Metal hose reel.	28
Figure 3.9 Metal reel design	29
Figure 3.10 Reel holder and reel side screw.	30
Figure 3.11 Free body diagram of the reel.	30
Figure 3.12 Inside of the junction box.	31
Figure 3.13 Sensor deployed (left) and sensor retracted (right).	31

Figure 3.14 Plastic hose reel.	33
Figure 3.15 Initial motor station.	34
Figure 3.16 Power supply junction box.	35
Figure 3.17 Motor station conceptual design.	36
Figure 3.18 Window motor and built-in coupler.	37
Figure 3.19 Belt and pulley system.	39
Figure 3.20 Driven pulley and nut insert.	40
Figure 3.21 Potentiometer and coupler.	40
Figure 3.22 Broken reel coupler/driving pulleys.	41
Figure 3.23 Broken driven pulley and coupler.	41
Figure 3.24 Free body diagram.	42
Figure 3.25 Belt drive and reel coupler	43
Figure 3.26 Conceptual design of prototype.	44
Figure 3.27 Floating pillars.	44
Figure 3.28 8540-312ZY motor and bracket.	46
Figure 3.29 MD10C motor driver.	47
Figure 3.30 DS3231 RTC module.	48
Figure 3.31 SIM900a GSM/GPRS module.	49
Figure 3.32 DS18B20 temperature sensor.	50
Figure 3.33 LM2596 (left) and XL4015 (right).	52
Figure 3.34 Solar Panel.	53
Figure 3.35 Operation flowchart.	55
Figure 4.1 Floating aquaculture sensor station.	60
Figure 4.2 Modified sensor reel.	61
Figure 4.3 Bearing on sensor reel.	62

Figure 4.4 Hose Reel original design.	62
Figure 4.5 Hose connector cap.	63
Figure 4.6 Junction boxes and circuit.	64
Figure 4.7 Motor station assembly with potentiometer control.	65
Figure 4.8 Reel coupler.	66
Figure 4.9 Circuit schematic.	75
Figure 4.10 Transporting prototype and mounting.	76
Figure 4.11 Prototype deployment.	76
Figure 4.12 Data visualisation on Grafana	77

LIST OF SYMBOLS / ABBREVIATIONS

%	Percent
°C	Celsius
°F	Fahrenheit
±	plus, minus
m	meter
mm	millimetre
kb	kilobyte
Hz	Hertz
mg/L	milligrams per litre
ppm	parts per million
W	watt
V	voltage
3D	Three dimensional
ADC	Analog-to-digital convert
BOD	Biological oxygen demand
BOR	Brown-out reset
CAD	Computer aided design
CPU	Central processing unit
DO	Dissolved Oxygen
FAO	Food and Agriculture Organisation
FYP	Final year project
GSM	Global System for Mobile Communications
GPRS	General Packet Radio Services
M2M	Machine-to-machine
MEMS	Microelectronic mechanical systems
MQTT	Message Queuing Telemetry Transport
I/O	Input/Output
I ² C	Inter-Integrated Circuit
IoT	Internet of Thing
IP	Internet Protocol

ISR	Interrupt Service Routines
IDE	Integrated development environment
pH	Potential of Hydrogen
PWM	Pulse width modulation
RTC	Real time clock
SD	Secure Digital
SET	Standard environmental temperatures
USB	Universal serial bus
WSN	Wireless sensor networks
Wi-Fi	Wireless Fidelity

LIST OF APPENDICES

Appendix A: Graphs	86
Appendix B: Datasheet	90
Appendix C: Arduino program code	93

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Aquaculture can be defined as the production of aquatic organism (Li and Liu, 2019). However, Aquaculture is distinct from commercial fishing, which is the harvesting of wild fish. Aquaculture is the cultivation of aquatic organism under optimal conditions for high production. Since different species of aquatic organism thrive under different conditions and environments, there are a number of aquaculture operations that cater to each of them. Figure 1.1 shows different types of aquaculture operations.

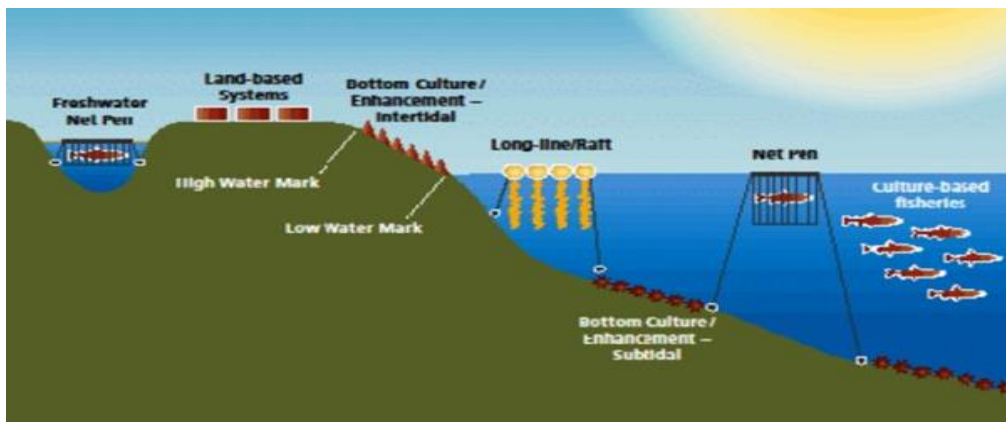


Figure 1.1: Types of aquaculture operations.

However, aquafarmers face a lot of challenges as aquatic organisms are more sensitive to changes of its surrounding compared to crops or animal farming. Water quality such as temperature, pH value, amount of dissolved oxygen (DO) and many others should be constantly monitored. Traditional methods of water quality monitoring are not able to deliver real-time data to the end-user effectively and typically have human errors and time-delay (Simbeye et al., 2014). In the last 50 years, our understanding towards aquatic ecosystem and management of aquaculture has improved tremendously with the help of scientific developments (FAO, 2020). With the advancement on technologies including sensors, wireless communications and computing devices, this is made possible.

An Internet of Things (IoT) based monitoring system can get real-time data through sensors and sending this data wireless through the internet. This data will then be processed, analysed and store in the cloud and make this information accessible for the end-user on their mobile devices. IoT also allows the possibility of automated control system that activates actuator such as fans, chemical dispenser to adjust water quality if any parameters fall below a predetermined threshold. It can even forecast potential catastrophes in the future through the analysis and learning of the water quality trend over a period of time.

1.2 Importance of the Study

One of the major turning points for aquaculture happened in 2016, when the world's production from aquaculture exceeded the captured production. As the human population grows, so does the global food supply. Figure 1.1 shows the trend of world capture fisheries and aquaculture production in terms of both inland and marine waters of aquaculture and capture fisheries.

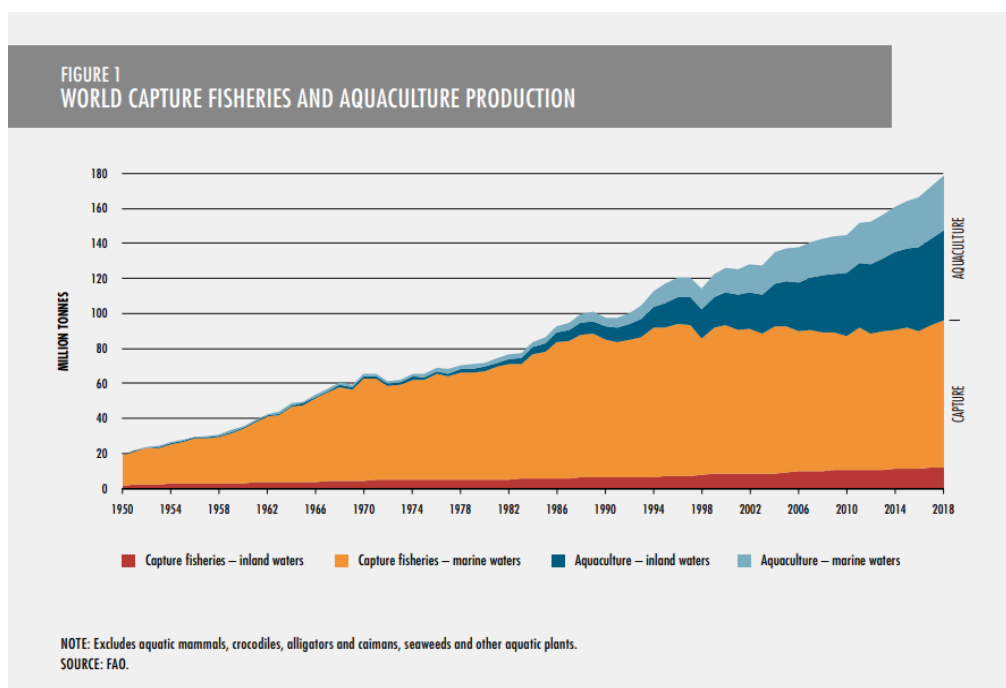


Figure 1.2: Trend of World Capture Fisheries and Aquaculture Production (FAO, 2020).

Global fish consumption increased at an annual rate of 3.1% between 1961 and 2017 (FAO, 2020). Fish consumption increased from 5.2 kg per capita in 1961 to 19.4 kg per capita in 2017. In the year of 2018, it is estimated that aquaculture produced 82 million tonnes of fish production, which is valued at USD 250 billion. 52% of the fish produced for human consumption and 46% of the total fish production came from aquaculture. Other than fish production, there are also 32.4 million tonnes of aquatic algae and 26 000 tonnes of ornamental seashells and pearls.

With aquaculture blooming in these past few decades, it is important to have a sustainable way to continue increasing aquaculture production in the future. According to research by Li and Liu (2019), the quality of the water must meet certain requirements to ensure that certain fish and plants in the water can survive and reproduce consistently. In order to ensure maximizing production, the breeding and growing of aquatic organism must be under well-controlled situations. This is where water quality monitoring comes into the scene. An efficient and effective water quality monitoring that provides long-term and real-time data are required for intensive and precise aquaculture.

1.3 Problem Statement

According to a study by Strobl and Robillard, the first appearance of WQM was using manual methods for water sampling and analysis. However, manual methods for WQM lead to an inconsistent, ineffective, and inefficient WQM model. This is caused by the following reasons:

1. Typically, the water sampling period, location, and depth are arbitrary or chosen based on other subjective factors (Strobl and Robillard, 2008).
2. Manual sampling and analysis lead to limited water sampling frequencies as well as potential human error (Simbeye et al., 2014).
3. Once the system is in place, the effectiveness of the monitoring design is typically not re-evaluated (Strobl and Robillard, 2008).
4. There is a time delay between getting the water sample from the water source and analysing it inside a laboratory (Li and Liu, 2019).

A consistent and efficient WQM model must be implemented in order to prevent dangerous situations from occurring that could potentially ruin months

or even years of laborious work (Sousa e Silva et al., 2016). A systematic WQM will need to take place on this issue to ensure that the WQM is able to stably deliver consistent real-time data. The data will then be accessible to the farmers through data visualization platforms, and this data can be used to optimize the water quality.

1.4 Aim and Objectives

This study aims to build a floating aquaculture sensor station that can collect real-time water quality parameters data such as temperature and dissolved oxygen and transmit this data wirelessly to a destined server. The floating aquaculture sensor station will be floating on top of two floating pillars. It is expected to be self-sustainable in terms of power. The objectives are as below:

1. IoT-enabled water quality monitoring system that gathers real-time water quality parameters.
2. Upload the obtained water quality parameters remotely to a server for data visualisation.
3. To be self-sufficient though harvesting solar energy

1.5 Scope and Limitation of the Study

The aim of this study is to investigate the feasibility and effectiveness of a fully automated machine for aquaculture purposes with remote data transmission capabilities. The research will focus on the design, development, and testing of the machine's automation and remote data transmission systems. The study will explore various technologies, including sensors, actuators, and communication protocols, to enable the machine's automation and data transmission capabilities. The investigation will also assess the machine's performance in different environmental conditions and analyse its potential for practical applications.

This study is limited to the evaluation of the machine's automation and remote data transmission capabilities, reliability, and functionality, and does not explore the economic or social implications of the technology.

1.6 Contribution of the Study

This study makes significant contributions to the field of automation and remote data transmission in aquaculture. The development of a fully automated machine for aquaculture purposes is a unique solution that minimizes the need for manual labour and reduces the potential for human error in aquaculture operations. The machine's ability to send data remotely provides real-time information on the performance of the aquaculture system, allowing for initiative-taking management and optimization of production processes. The study evaluates the reliability and functionality of the machine's automation and remote data transmission systems, providing valuable data for the improvement of future designs. This research advances the field of automation and remote data transmission in aquaculture and provides a basis for the development of similar technologies in other industries.

1.7 Outline of the Report

There are total of five chapters included in this report. The first chapter consist of an introduction to aquaculture and its importance, the importance of water quality monitoring for aquaculture system, problem statement, aim and objectives, scope and limitation of the study and lastly the contribution of study. The second chapter is a literature review on thesis about different approaches to water quality monitoring that are done by other researchers so that it provides a better idea and decision making on how to develop an effective water quality monitoring station. The third chapter focus on the methodology to develop the floating aquaculture sensor station. This includes project planning and milestone as well as improvement and analysis of the prototype. For the fourth chapter, it will be result and discussion where the performance or result of the prototype will be discussed. Last but not least, the fifth chapter which is conclusion summarizes the overall report and also points out all the limitation of the project with proposed future work and recommendation for this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Water quality monitoring has always been an important field in aquaculture as it directly affects the ecosystem of aquatic resources and fish. This is the reason that numerous studies, some of which date back to the 1960s, have been carried out to monitor water quality. These studies looked at the water conditions that were best for maintaining the health of plants and fish, as well as defining what was toxic or pathogenic.

Due to the limitations of conventional WQM models, which include manual sampling and wire monitoring systems, the use of Wireless Sensor Networks (WSN) is increasing. A WSN has a number of sensor nodes connected to each other to acquire and process sensor data. Each sensor node consists of one or more sensors that are connected to a routing node that is interconnected with other routing nodes (Majid et al., 2022). These interconnected routing nodes are usually connected via a closed network to an observer (user) (Kocakulak and Butun, 2017). Figure 2.1 shows the architectural concept of a wireless sensor network that is implemented in Simbeye, Zhao, and Yang's (2014) aquaculture monitoring station.

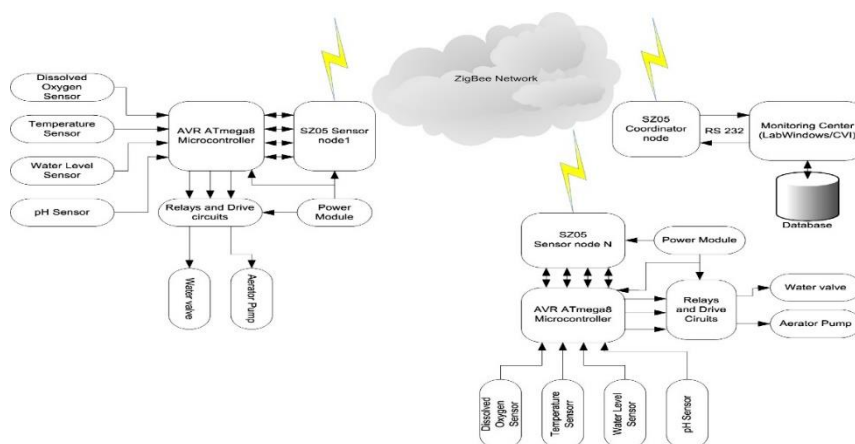


Figure 2.1: Architecture concept of a wireless sensor network for aquaculture WQM (Simbeye et al., 2014).

Simbeye, Zhao, and Yang use ZigBee network communication as their network for aquaculture WQM due to the characteristics of ZigBee being low cost and low power consumption. The overall structure of Simbeye, Zhao, and Yang is comprised of coordinator nodes, sensor nodes, and a monitoring hub. Compared to conventional methods, WSN are able to achieve real-time WQM and on a remote basis. WSN offer easier implementation, better maintainability and also are cost effective (Simbeye et al., 2014). WSN have greater adaptability for different situations such as river, lake, ponds, seas and etc (Strobl and Robillard, 2008).

On the other hand, the Internet of Things (IoT) is described as devices with computational capacity and connected to the internet (Manrique et al., 2016). In this sense, WSN and IoT are closely related, but there are still differences between these two terms. In order to show the contrast between WSN and IoT, Figure 2.2 shows a graphic timeline of when the terms WSN and IoT emerged created by Manrique and others (2016).

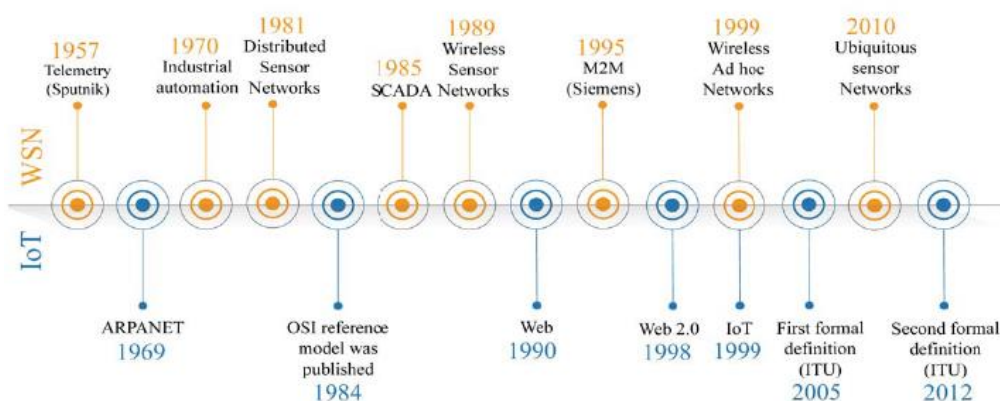


Figure 2.2: Timeline of Concepts (Manrique et al., 2016).

From Figure 2.2, we can clearly see that the term "WSN" was created before the term "IoT". This is because the concept of WSN was formed before the internet and was built on machine-to-machine (M2M) communications and computer networks. These M2M communications and computer networks are usually specialised private networks and are subjected to specific industry standards (Manrique et al., 2016). This can be observed from Figure 2.1, where the sensor nodes are connected through the ZigBee network instead of

the internet. WSN can be considered as a subset of IoT where the wireless sensor networks are able to transmit data to the internet through a gateway. A gateway is a network node that is responsible for the conversion of communications protocols between internal and external networks. WSN and IoT are not two conflicting concepts, and WSN can be implemented into the concept of IoT. Figure 2.3 shows an implementation of WSN into the IoT.

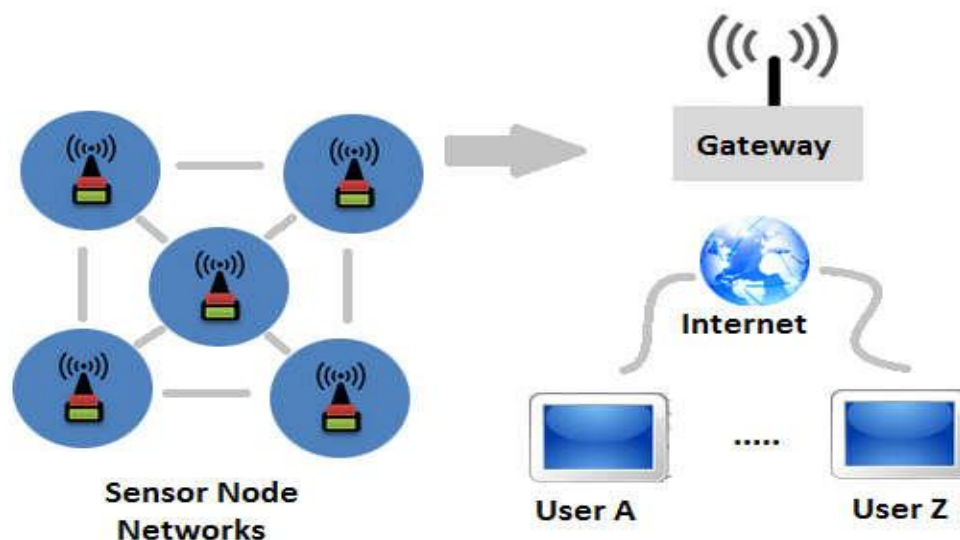


Figure 2.3: Implementation of WSN into the IoT.

Other than that, the advancement of communications devices has recently allowed for devices such as the smartphone and tablet to also be integrated into this same system, thus allowing people a more convenient way of collecting data, which will definitely be able to benefit farmers in an effective way. Recent water quality monitoring networks have shifted to concentrate on particular water quality parameters including salinization, acidification, microbial, and many others. This is made possible with newly developed sensors, including fibre optics, biosensors, and microelectronic mechanical systems (MEMS) that are made for the detection of various water quality parameters (Li and Liu, 2019).

Due to the limitation and scope of this study, which only consists of one sensor node, this literature review will be more focused on IoT based water quality monitoring systems for aquaculture. However, previous studies on aquaculture WSN for WQM will still be covered as they are closely related in terms of concept architecture. This literature review will cover the IoT

based aquaculture monitoring system architecture, different water parameters and their effects on aquaculture, and the hardware and software used in the development process. These are important aspects that will determine the direction and methodology of this project.

2.2 IOT based aquaculture monitoring system

This subsection of the literature review will be focused on the IoT architecture as well as an overview of the different components inside the architecture.

In a study that was conducted by Raposo and others (2018), they proposed an IoT monitoring system architecture that consists of four main components, as shown in Figure 2.4.

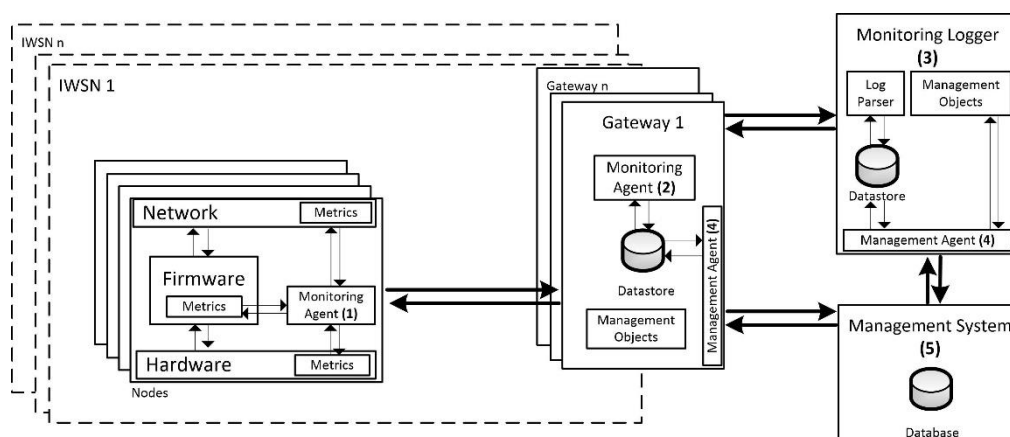


Figure 2.4. The Proposed Monitoring Architecture.

These four components include wireless sensor networks; a gateway; a monitoring logger; and a management system. Most IoT monitoring systems are structured this way, except that it doesn't necessarily be a wireless sensor network and can consist of independent sensors. The sensors are responsible for detecting water quality parameters and transferring this data to the gateway, which usually consists of a microcontroller or microprocessor. The gateway will then transmit this data to a management system. The management system will be responsible for storing and processing this data. Finally, this data will be transmitted to the monitoring logger for data visualization and can be assessed by users such as the farmers.

Hairol and others (2018) use the same monitoring architecture as above in their fishpond monitoring system. Their monitoring architecture is shown in Figure 2.5.

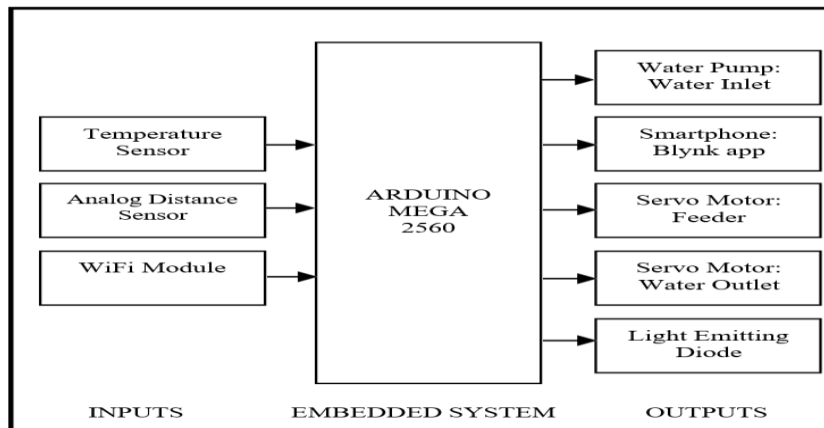


Figure 2.5: Fish Pond Monitoring System (Hairol et al., 2018).

From figure 2.5, Hairol and others use a temperature sensor and an analog distance sensor for water level detection as their sensors. The Arduino Mega with an ESP8266 Wi-Fi module act as their gateway, and the Blynk app is both their management system and monitoring logger. The Blynk app is a cross platform mobile app library that allows modification according to the project's requirements. The other outputs, such as water inlet, feeding, and water outlet, act as a control system to adjust the water source quality back to its desired range.

Niswar and others (2018) developed an IoT monitoring system architecture with the implementation of WSN. Figure 2.6 shows their IoT monitoring architecture between different IoT components.

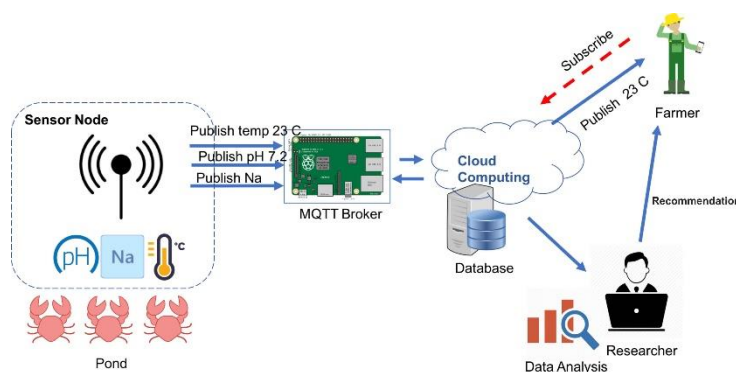


Figure 2.6: IoT-based Monitoring System (Niswar et al., 2018).

Each sensor node consists of an Arduino mega with a LoRa shield operating on a 915 MHz frequency, as well as a water temperature sensor, pH sensor, and salinity sensor. Their WSN uses the lightweight Message Queuing Telemetry Transport (MQTT) protocol to communicate between sensor nodes, which act as MQTT clients, and the Raspberry Pi, which acts as the MQTT broker. They are using the Node-RED development tool to create the monitoring logger dashboard. The monitoring logger communicates with the Raspberry Pi using MQTT as well. The MQTT broker publishes the relevant parameters to the monitoring logger and will be accessible using mobile devices.

Rosaline and Sathyalakshimi (2019) proposed an IoT based monitoring station architecture that is shown in Figure 2.7.

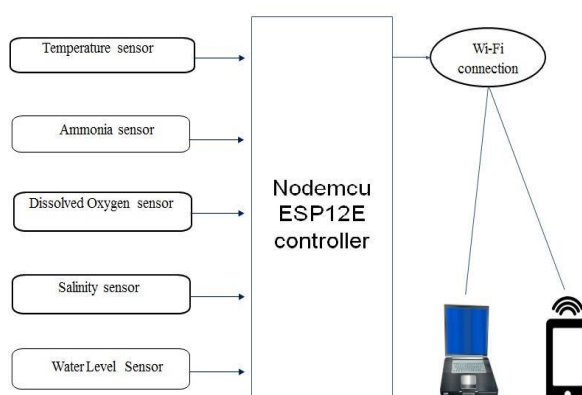


Figure 2.7: IoT-based Aquaculture Monitoring and Control System Architecture (Rosaline and Sathyalakshimi, 2019).

They used multiple sensors, including temperature, ammonia sensor, DO sensor, salinity sensor, and water level sensor. The sensors' data is processed by the NodeMCU, which is a microcontroller with built in Wi-Fi accessibility. The data is then sent to and displayed on "Ubidots", which is an IoT platform.

2.3 Water Quality Parameters and Sensors

In a thorough research of water quality parameters by Li and Liu (2019), they define water quality as the combination of its physical, chemical, biological, and radiological state. Aquaculture water quality typically refers to its temperature, dissolved oxygen, pH value, turbidity, and etc. These few parameters affect aquatic organisms the most. Li and Liu stated that since water quality has many states, there are monitoring methods targeted at the different states of the water and the methods to detect these parameters. The more commonly used monitoring methods are physical detection, chemical reaction, and electrode reaction.

Physical detection is usually targeted at water parameters that can be perceived by our senses, such as the color, odor, and turbidity of the water. For example, the color of the water indicates the presence of minerals or plant organisms such as algae in the water. The chemical reaction monitoring method involves adding a specific substance to the water sample to see if a specific chemical reaction is triggered. These chemical reactions are crucial in determining the presence of toxic organic and inorganic substances as well as microorganisms in the water. An electrode reaction involves placing an electrode into the water to initiate the transfer of electrons between the electrode and the water. It is possible to quantify changes in water parameters using the signal change on the electrode. Electrode reaction sensors are used in the detection of DO, pH, biological oxygen demand (BOD), and others.

Li and Liu listed out the testing standards for China's fishery water quality standards that were last updated in 1990.

Table 2.1: China's fishery water quality standards.

Parameter	Unit	Tap	Fresh Aquaculture	Marine Aquaculture
Hardness	mg/L	< 800	< 400	-
pH	1	6 – 9	6.5 – 8.5	7.8 – 8.5
NH ₃ -N	mg/L	< 2	< 0.2	< 0.3
Bacteriological	Per liter	< 2 000	< 5 000	< 10 000
DO	mg/L	2.0 – 7.5	> 5	> 5
Temperature	°C	Natural state	16 - 39	16 - 39

In this subsection, we will be looking into some of the parameters, including temperature, dissolved oxygen, and pH value.

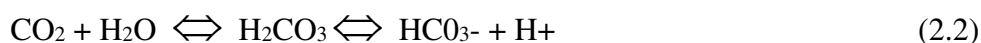
2.3.1 pH

Potential hydrogen, or pH, is the measurement of the acidity or alkalinity of substances that are soluble in water (Simbeye et al., 2014). It is approximately the base-10 logarithm of the molar concentration of hydrogen ions, expressed in moles per liter (Covington et al., 1985).

$$\text{pH} = -\log_{10}(a_{\text{H}^+}) = \log_{10}\left(\frac{1}{a_{\text{H}^+}}\right) \quad (2.1)$$

From the formula, we can know that one pH value is equivalent to an increment of 10 times the number of hydrogen ions (H⁺) in the water. The pH scale ranges from 0 to 14, with 7 as the neutral value. A pH value of less than 7 indicates acidity, while greater than 7 indicates alkalinity. It is a common misconception that pH value cannot be less than 0 or greater than 14 (Li and Liu, 2019).

In a healthy aquaculture ecosystem, the change in pH value is caused by a few things, including algal photosynthesis, the respiration of organisms, and microorganisms (Summerfelt, 2000). Summerfelt stated that the process of pH value change can be concluded with the following chemical reaction formula:



During the night, the respiration of organisms and microorganisms produces carbonic acid (H_2CO_3), which leads to bicarbonate HCO_3^- and H^+ ions. The increase of H^+ ions cause the pH value to drop. This is the same for the daytime except those plants like algae use CO_2 for photosynthesis, which causes the reaction above to go from the right to the left. This causes H^+ ions in the water to decrease, thus increasing the pH value.

pH is one of the important water quality parameters in aquaculture as it affects organisms in the water. In a study of aquaculture monitoring by Simbeye and others, they stated that the optimal pH value ranged from 6.5 to 9 in most cases. *Scylla paramamosain*, a type of mud crab, is suitable to live at a pH value of around 8.5 (Niswar et al., 2018). pH values below 4.0 cause acid deaths, and pH values above 11 cause alkaline deaths. In a low alkalinity pond, the pH value during the daytime could go up to 10, which is detrimental for hybrid striped bass and other fish. Other than that, a sudden change in pH value could cause a pH shock to the fish and end up killing the fish. This is commonly seen when fish are moved from a pond to a tank and vice versa (Summerfelt, 2000).

Jiang et al. used an analog pH electrode LE438 with a pH measurement range of 0 - 14 with ± 0.05 pH accuracy. It can also be used as a temperature sensor with a measurement range of 0 - 80 °C. It is made of polyoxymethylene and has a cable length of 1 meter.

Simbeye and Yang (2014) used a PH450 with a pH measurement range of -2 - 16 pH, a resolution of 0.01 pH, and ± 0.01 pH accuracy. The PH450 even has a wash cycle function, which ensures consistent analysis.

2.3.2 Dissolved Oxygen

Bosma and Verdegem (2011) stated that dissolved oxygen is vital in an aquatic ecosystem as it is needed for organisms' respiration and waste decomposition. DO levels in water are commonly used in parts per million (ppm), milligrams per litre (mg/L) or percent dissolved oxygen (%). It also reflects the physical

and biological aspects of the body of water (Simbeye et al., 2014). Summerfelt (2000) stated that there are three physiological facts for dissolved oxygen in aquaculture:

1. Smaller fish's oxygen consumption per unit of body weight is higher than larger fishes.
2. Swimming fish's oxygen consumption is higher than resting fish.
3. Fish's oxygen consumption is higher after they have been fed.

Summerfelt also stated that the primary sources of oxygen in ponds come from algal photosynthesis and wind mixing air and water. For tanks and raceways, the primary source of oxygen comes from the inlet water. This causes dissolved oxygen to be one of the most crucial water parameters in a pond.

Lack of dissolved oxygen causes hypoxia and anoxia and ultimately leads to the death of organisms. Other than that, low dissolved oxygen levels cause shrimp and fish to be more vulnerable to stress-related diseases and gill infections such as hamburger gill disease, limiting fish growth and appetite. Although there are different optimum DO levels for different species, the general recommended DO levels are above 5 mg/L for warm-water fish and 6 mg/L for cold-water fish. A minimum of 2 ppm is needed for mineralization and nitrification.

Simbeye and Yang (2014) used a DO3000 DO sensor with a DO measurement range of 0 - 20.00 mg/L, a resolution of 0.1% mg/L, and ± 0.2 mg/L accuracy. The DO3000 can be used as a standalone device with a battery pack as it has internal data logging capability as well as telemetry communication. It can also send serial output when connected to external processors or data loggers. The DO3000 offers low power consumption and is suitable for long term and rapid monitoring.

Shi and others (2018) used an analogue DO-954A DO sensor with a DO measurement range of 0.01 - 19.9 mg/L with ± 0.1 mg/L accuracy. The operating temperature of this sensor is 5 - 40 °C. They connected the DO sensor to a CCC2530 chip with ADC capability.

2.3.3 Temperature

Most of the fishes' body temperatures are the same as their surroundings, which is referred to as ectothermic. However, different fish have different standard environmental temperatures (SET), and they are classified as cold water, cool water, warmwater, and tropical (Summerfelt, 2000). If the standard environmental temperatures of the fishes or other organisms are not met, it will affect their appetites, growth, and make them more vulnerable to stress-related disease. Similar to pH value, a sudden change in temperature will also cause temperature shock and lead to mortality. When fish are moved from one place to another, it is important to make sure that the temperatures are similar or gradually change the water temperature. In the aquaculture industry, temperature is also crucial because it affects the chemical reactions in the water. It affects the available dissolved gases, oxygen, ammonia and many more (Simbeye et al., 2014). The increase in dissolved oxygen demand in the water affects the respiration of aquatic organisms. It also has an impact on the development and growth of aquaculture plants and animals (Li and Liu, 2019). Low temperatures lower fishes' appetites and high temperatures cause high mortality rates among shrimp (Garcia et al., 2011).

The same digital temperature sensor module DS18B20 is used by Hairol and others (2018), Fourie and others (2017), Simbeye, Zhao, and Yang (2014). The DS18B20 has a temperature measurement range of -55 to +125 °C with an accuracy of ± 0.5 °C. It has an operating voltage of 3.3 - 5v, which makes it compatible with Arduino and other popular microcontrollers. It has a 9 - 12-bit temperature reading, which means that it can have a resolution of 0.0625 °C.

2.4 Summary

The Aquaculture systems require continuous monitoring of water parameters such as pH, temperature, and dissolved oxygen to ensure the health and productivity of aquatic organisms. However, conventional methods of water quality monitoring are limited in terms of real-time monitoring and remote access. To address these challenges, the use of wireless sensor networks (WSN) and the Internet of Things (IoT) in water quality monitoring is gaining

popularity. The IoT-based aquaculture monitoring system architecture consists of wireless sensor networks, a gateway, a monitoring logger, and a management system. This monitoring approach improves water quality monitoring, facilitates early detection of potential problems, and ensures the sustainability of aquaculture systems. Water parameters such as pH, temperature, and dissolved oxygen are critical in aquaculture, affecting the growth, survival, and reproduction of aquatic organisms. Therefore, monitoring these parameters using IoT-based systems can significantly enhance the productivity and health of aquatic organisms.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Project Planning and Milestone

Figure 3.1 shows the project flowchart for Final Year Project.

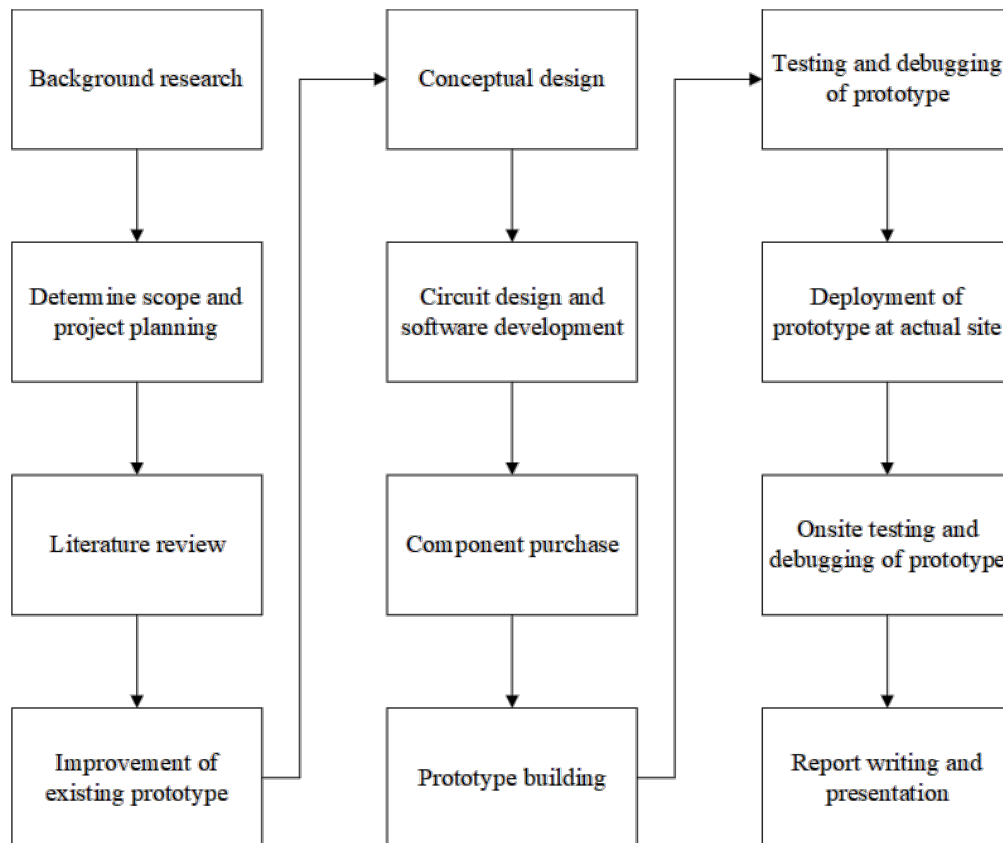


Figure 3.1 Project Flowchart.

At the start of the project, background research is done after confirmation of the final year project (FYP) title. After that, the scope of study is determined to act as a guideline for the project. Project planning was done to ensure a smooth workflow of the project as well as to keep track of project progress. A thorough literature review is done to investigate the related research and studies that have been conducted. This allows for a better understanding of the overall project and allows the decisions to be based on a strong base of information. Reviewing other researchers' work on the related

matter and determining the strengths and weaknesses of different approaches was done to formulate a roadmap that leads to the most appropriate solution.

There is already an existing prototype built by another student, Tan Wei Zhi, who was working on the same project title as me. Modifications and improvements to be made to the existing prototype are identified and sorted by importance. Next, the conceptual design of the modifications and improvements is formed. The circuit design and software development are then done on improved version of prototype.

Upon completion of conceptual design and software development, testing and debugging of the prototype is done to ensure smooth operation. Finally, the prototype will be deployed at the actual site alongside with onsite testing and debugging of the prototype to address any issue faced at the actual site.

3.2 Gantt chart and Log book

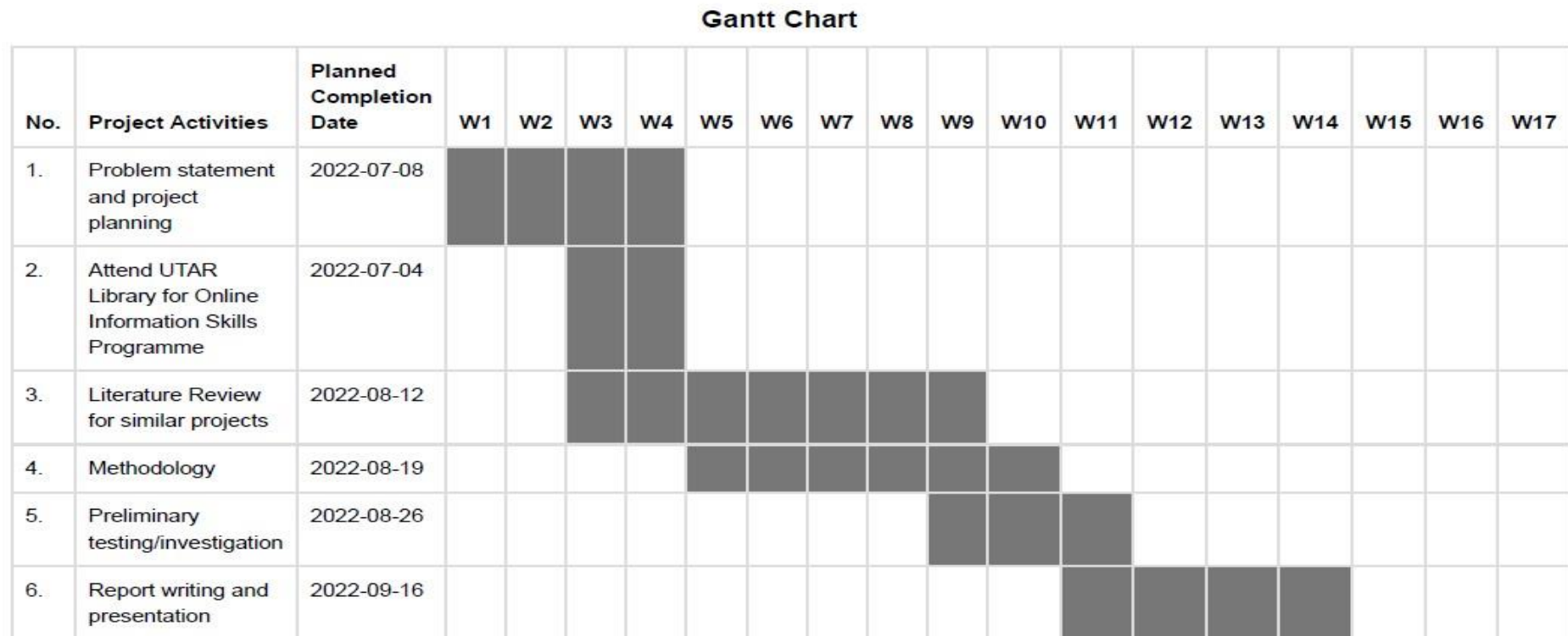


Figure 3.2 Gantt chart.

Log Book

Week	Activities to achieve milestones	Submission Date/Status	Acknowledge By	Comments	Uploaded Documents	Action
Week 2	<ul style="list-style-type: none"> - Participated in June 2022 DMBE FYP Briefing - Confirmed FYP title and submitted Title Registration Form - Participated in General Briefing and FYP Briefing by Supervisor - Next step is to read through related Journals and Articles for literature review and liaise with FYP senior Tan Wei Zhi who is currently working on the same FYP title and decide whether to continue and enhance his prototype or to create a new prototype on my own. - Prepared a 14-week Gantt chart to track my project progress 	2022-06-24 10:30:18	Reviewed by supervisor			
Week 4	<ul style="list-style-type: none"> - completion of the problem statement and project planning, will be continuing the previous senior's prototype and making improvements to it. - completion of UTAR Library Online Information Skills Programme, had learned about effective search methods as well as the definition and consequences of plagiarism and the advantages of proper referencing. - Identifying relevant journals and literature for floating aquaculture sensor stations. - studying relevant journals and literature in detail for the next few weeks. - Setting up consistent meetings with the previous senior for discussions and exchanging ideas on the improvements of the current prototype. 	2022-07-08 18:22:59	Reviewed by supervisor			
Week 6	<ul style="list-style-type: none"> - Going to attend the FYP report writing and presentation Workshop on 22/7/2022 - Designing Potentiometer mount on the prototype - Creating Solidwork parts for every part of the prototype - Designing new version of the prototype for better center of gravity on the rotating shaft - Researching on ways to mount electronics vertically in electrical box and wiring configuration for vertical mount - Designing timing pulley coupler for potentiometer - Troubleshooting server communication using Arduino and wireless module - Designing parts for improve waterproofing of prototype - Searching for suitable Solar Panel model and controller 	2022-07-22 13:03:02	Satisfactory from Supervisor Satisfactory from co-Supervisor			

Week 8	<p>Completed:</p> <ul style="list-style-type: none"> - FYP report writing and presentation workshop - Troubleshooting server communication using Arduino and wireless module - First phase design of potentiometer mount with Wei Zhi - Solidwork parts for Rack and Shaft <p>Ongoing:</p> <ul style="list-style-type: none"> - Testing post request using JSON data after Dr. Alex set up the server - Creating functions for respective modules - Drafting flowchart for program flow - Wiring configuration and schematic diagram between Arduino and modules - Wiring Configuration for battery, motor, and junction box - Method for threading 3D printed coupler - Improving shaft spinning motion by reducing friction between shaft holder and shaft <p>Not Started:</p> <ul style="list-style-type: none"> - Designing a new version of the prototype for a better center of gravity on the rotating shaft - Researching on ways to mount electronics vertically in the electrical box and wiring configuration for vertical mount - Searching for a suitable Solar Panel model and controller - Battery mount method 	2022-08-05 13:31:36	Reviewed by supervisor			
Week 10	<p>Completed:</p> <ul style="list-style-type: none"> - Solidworks for floating pillar - Method for threading 3D printed coupler - Successfully mounted potentiometer with good belt tension using sloted hole <p>Ongoing:</p> <ul style="list-style-type: none"> - Testing sleep and wake routine with server communication with Arduino - Implementing potentiometer voltage reading with motor driver code - Testing post request using JSON data after Dr. Alex set up the server - Drafting flowchart for program flow - Wiring configuration and schematic diagram between Arduino and modules - Wiring Configuration for battery, motor driver, potentiometer and components inside junction box <p>Not Started:</p> <ul style="list-style-type: none"> - Designing a new version of the prototype for a better center of gravity on the rotating shaft - Researching on ways to mount electronics vertically in the electrical box and wiring configuration for vertical mount - Searching for a suitable Solar Panel model and controller - Improving shaft spinning motion by reducing friction between shaft holder and shaft 	2022-08-19 10:29:39	Satisfactory from Supervisor			
			Pending from co-supervisor			
Week 12	<p>Completed:</p> <ul style="list-style-type: none"> - Testing sleep and wake routine with server communication with Arduino - Implementing potentiometer voltage reading with motor driver code <p>Ongoing:</p> <ul style="list-style-type: none"> - Focusing on Report writing - Drafting flowchart for program flow - Wiring configuration and schematic diagram between Arduino and modules - Wiring Configuration for battery, motor driver, potentiometer and components inside junction box <p>Not Started:</p> <ul style="list-style-type: none"> - Testing post request using JSON data after Dr. Alex set up the server - Designing a new version of the prototype for a better center of gravity on the rotating shaft - Researching on ways to mount electronics vertically in the electrical box and wiring configuration for vertical mount - Searching for a suitable Solar Panel model and controller - Improving shaft spinning motion by reducing friction between shaft holder and shaft 	2022-09-02 09:48:19	Satisfactory from Supervisor			
			Satisfactory from co-Supervisor			

Figure 3.3 Logbook.

Gantt Chart













No.	Project Activities	Planned Completion Date	Weeks																		
			W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17		
1.	CAD design on floating station structure and motor coupler	2023-02-16	█	█	█																 
2.	software development for power shortage, data transmission and data saving. Purchasing of floating station structure and start constructing.	2023-03-02			█	█	█														 
3.	Overall prototype troubleshooting and debugging	2023-03-16					█	█	█												 
4.	Actual prototype testing and deployment onsite.	2023-03-30									█	█									 
5.	Final report writing and creating FYP poster	2023-04-27								█	█	█	█	█	█	█					 
6.	FYP presentation and FYP report submission	2023-05-05														█	█	█			 

Figure 3.4 Gantt chart.

Log Book						
Week	Activities to achieve milestones	Submission Date/Status	Acknowledge By	Comments	Uploaded Documents	Action
Week 2	<p>Completed:</p> <ul style="list-style-type: none"> - Bought new plastic reel to replace the metal reel. This reduce the stress on the 3D printed coupler as the plastic reel take less force to rotate. The components are no longer required to be mounted on top of the rotating shaft and the wire of the sensor can be inserted from the side of the rotation shaft, this makes for a better center of gravity on the rotating shaft. - 3D printed a motor coupler for the plastic reel for testing. - Finished solidworks files for plastic reel. <p>Ongoing:</p> <ul style="list-style-type: none"> - Improving motor coupler for plastic reel - Adding power shortage scenarios and program for cases of power shortage - Designing structure for solar panel and the base for the floating station - Searching for suitable solar panel controller <p>Not Started:</p> <ul style="list-style-type: none"> - Testing post request using JSON data with the new server url - Wire and component configuration for all the components in the floating station - Adding SD card module and data saving feature to the program 	2023-02-10 14:37:16	Reviewed by supervisor			
Week 12	<p>Week 4:</p> <ul style="list-style-type: none"> Tested motor coupler, adjusting dimensions for 2nd printing Added power shortage scenarios & program Completed base structure for the floating station Designing a structure for solar panel Buying aluminum & searching for a suitable solar panel controller Changing the design for the motor connection <p>Week 6:</p> <ul style="list-style-type: none"> Bought & assembled base structure Designed motor coupler to fit using Allen key Designing a structure for solar panel Wire & component configuration for the floating station Redesigned potentiometer holder <p>Week 8:</p> <ul style="list-style-type: none"> Designed structure for solar panel Bought & assembled solar panel structure Configured wire & assembled components Redesigned potentiometer holder Tested overall system & deployed prototype Monitoring data at Ooxela & ThingSpeak Report writing <p>Week 10 & 12:</p> <ul style="list-style-type: none"> Monitoring data at ThingSpeak Report writing 	2023-04-26 17:38:46 Late	Satisfactory from Supervisor			
			Satisfactory from co-Supervisor			

Figure 3.5 Logbook.

3.3 Block diagram of floating aquaculture sensor station

Figure 3.6 shows the block diagram of the floating aquaculture sensor station.

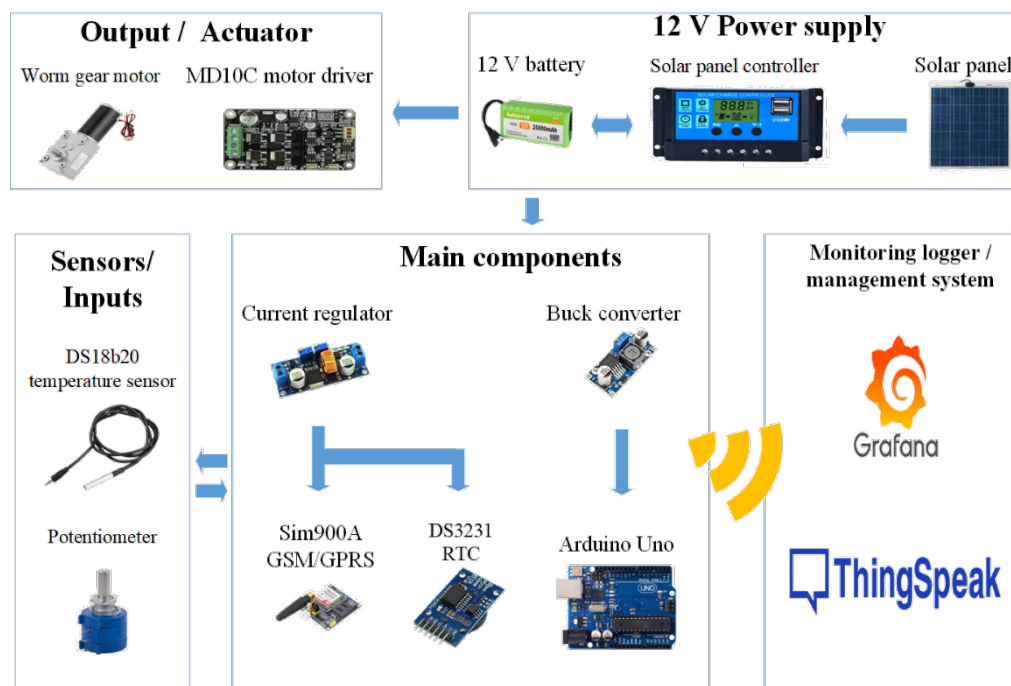


Figure 3.6 Project block diagram.

The system consists of two sensors, which are the DS18B20 temperature sensor and a potentiometer that acts as an absolute position sensor. The DS18B20 send the water temperature reading to the Arduino UNO, which acts as the main microcontroller in the embedded system. The Arduino UNO will then send out the data to the designated server which in this case is Grafana. The Arduino UNO will send the data to the designated server using a GSM/GPRS module. A real time clock act as an "alarm clock" that wakes the Arduino UNO up from sleep with a pre-set timer interval. This allows the Arduino to go into sleep mode and thus reduce power consumption. The solar panel controller is responsible for regulating the current coming from the solar panel to charge a Li-ion 12V battery. The 12 V battery is connected to the load, this includes the Arduino, every electronic module as well as the motor. The 12V battery consist of a male and female DC jack connector which acts an output and input terminal respectively. The female DC jack connector is connected to the solar panel controller to store the energy coming from the solar panel. The male DC jack is then connected to a buck converter and a voltage current regulator. The buck converter converts the 12V from the

battery to 9V and then powers the Arduino. The voltage current regulator is connected to the rest of the modules including the DS18B20 temperature sensor, SIM900A GSM/GPRS module and DS3231 RTC module. Only the potentiometer is connected to the 5V supply by the Arduino Uno. 12 V battery is also connected to the Arduino Uno and the worm gear motor through the MD10C motor driver.

The prototype of the floating aquaculture sensor station is equipped with a potentiometer which functions as an absolute angle measurement sensor. This crucial component enables the system to accurately determine the extent to which the sensor has been deployed or retracted. The potentiometer provides the necessary information to prevent the motor from under turning or over turning the reel, ensuring optimal functionality and longevity of the system. Furthermore, it serves as a vital source of position information in cases of power shortage, allowing the Arduino to retrieve the sensor and reel's position data. This feature is particularly important in maintaining the system's accuracy and reliability, ensuring consistent and precise data collection. The inclusion of the potentiometer in the design is therefore a critical aspect of the overall functionality and success of the floating aquaculture sensor station.

Originally all the modules are powered by the Arduino Uno while the Arduino Uno is connected to the output of the 12V battery directly. The output of the battery is directly connected to the Arduino Uno as well as the MD10C motor driver. However, this might harm the Arduino Uno as the current demand from the motor driver can be up to 6.5A since the stall current of the worm gear motor is 6.5A. The rated current of the worm gear motor 5480-31ZY is 1.6A which still exceed the maximum input current for the Arduino Uno. The current surge is too much for the Arduino Uno to handle as the maximum input current across Vcc and Ground that the Arduino Uno is 200mA. Figure B-1 and figure B-2 shows the datasheet of both 5480-31ZY worm gear motor and Arduino. The datasheet of 5480-31ZY is provided by the seller.

3.4 Conceptual designs of prototype

Some of the challenges faced in this project is the system architecture which consist of mechanical parts, electronic components, software development and finally the integration of all of the above. Since the mechanical parts and electronic components are interrelated making it difficult to explain one without the other. This section will focus on the mechanical parts, related electronic components, and the integration between these two. The conceptual design of the prototype is divided into four parts as listed below:

1. Sensor Reel and Junction Box
2. Motor station and Power Supply
3. Reel coupler and potentiometer
4. Base structure with floating pillars and solar panel

Since there is an initial conceptual design and prototype as well as an improved conceptual design and prototype, this section will first explain about the initial conceptual design and its limitation. After that, it will address its limitation and propose a new conceptual design. Figure 3.7 shows the overall of the initial conceptual design and prototype.

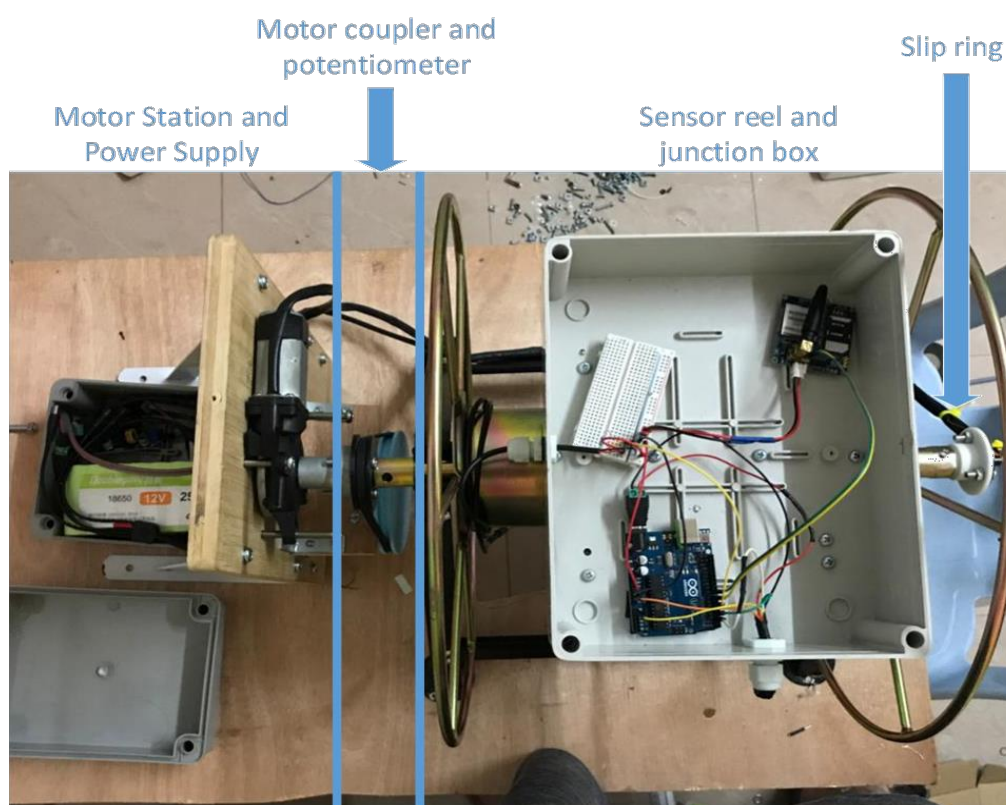


Figure 3.7 Initial design of prototype.

3.4.1 Sensor Reel and Junction Box

3.4.1.1 Initial conceptual design and prototype

In the development of the floating aquaculture sensor station, the deployment and retraction of the sensor wire were critical components for collecting data on water quality and prolonging the lifespan of the sensor. As such, selecting a suitable reel for this purpose was essential. Considering the scalability of the project such that the sensor might need to be deployed few meters deep into the water, the reel would need to have a larger diameter so that the wire can be spun around the reel preferably with lesser revolution which is related to the potentiometer. The initial selection of a metal hose reel was based on its mechanical strength properties. The first design was using a metal reel as shown in figure 3.8.



Figure 3.8 Metal hose reel.

However, during testing, it was observed that the metal hose reel created significant friction, which resulted in metal scraping sound and increase the amount of torque needed to spin the metal reel. This friction issue was caused by the lack of a bearing, which meant that the metal hose reel rested directly on the holder. Figure 3.9 shows the metal reel resting on the holder.



Figure 3.9 Metal reel design

Furthermore, the metal hose reel was not only causing friction, but it was also causing the reel holder to tilt inwards due to the weight of the reel. The reel holder was connected using two metal bars close to the base, and placing the metal reel on the holder caused the two sides of the holder to tilt inwards, making the holder's sides touch the metal screw head at the side of the reel. This additional contact between the holder and screw head increased the friction between the two, resulting in more resistance when the reel was spinning. Figure 3.10 shows the tilting resulting in the reel holder becoming in contact with the screw head at the side of the reel.



Figure 3.10 Reel holder and reel side screw.

The tilting is caused by lateral forces as the metal bar that is connecting the reel holder frame is too closed to the base of the reel holder. Since the reel holder frame may not be perfectly perpendicular and perfectly vertical causing a horizontal force. The vibration caused by the motor rotating the reel could cause the reel holder frame to tilt inwards as well. Figure 3.11 shows the free body diagram of the metal hose reel and the metal reel holder.

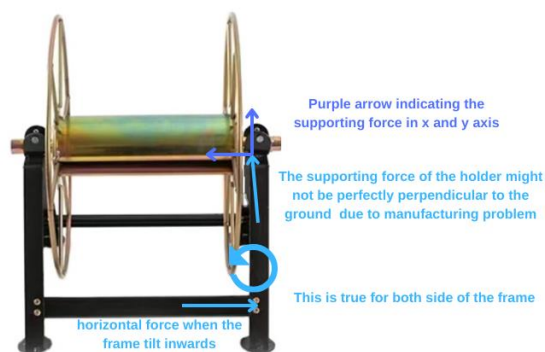


Figure 3.11 Free body diagram of the reel.

In the initial design, the junction box is mounted on the spinning reel itself. The junction box contains Arduino Uno, DS3231 timer module, SIM900A GSM/GPRS module and the DS18B20 temperature sensor module. The sensor module will exit the junction box and its wire will wrap around the reel to perform deployment and retraction of the sensor into the water. Figure 3.12 shows the junction box and its contents. Figure 3.13 shows the sensor before and after it is being deployed.

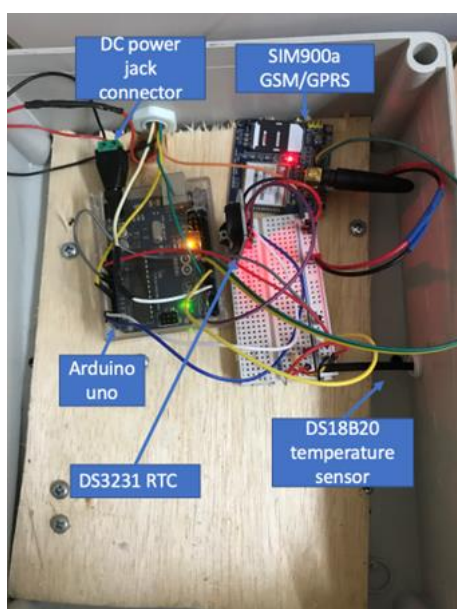


Figure 3.12 Inside of the junction box.

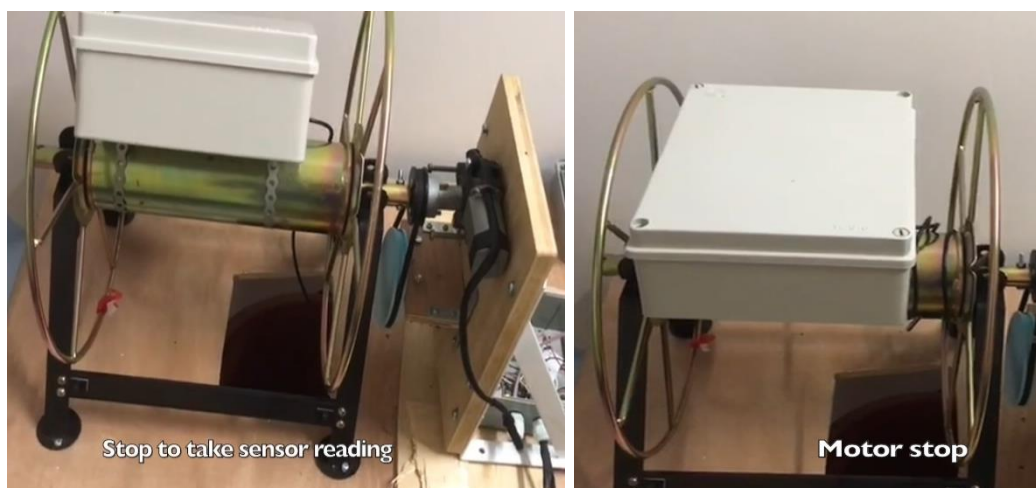


Figure 3.13 Sensor deployed (left) and sensor retracted (right).

The components are powered by the battery that is located at the behind the motor station. The wires are fed through a slip ring that is at the other side of the reel as shown in figure 3.7.

Mounting the junction box on the spinning hose reel can affect the center of weight and the rotation of the reel. The addition of the junction box to the reel will add weight to the system, and since the junction box is located off-center, it will cause an imbalance in the system. This imbalance can cause the reel to wobble or vibrate during rotation, affecting the accuracy of the data collected by the sensors. Additionally, if the slip ring is not designed properly or if it is damaged, it can cause a loss of connectivity between the sensors and the data acquisition system. This can lead to inaccurate or incomplete data being recorded, potentially impacting the success of the aquaculture operation.

Moreover, the rotation of the reel can also cause stress on the wires that are connected to the junction box. As the reel spins, the wires will twist and turn, which can cause the wires to break or fray over time. This can lead to data transmission issues and potentially cause the entire system to malfunction.

Finally, mounting the junction box on the reel can make it difficult to access and maintain. The constant rotation of the reel can make it challenging to inspect and troubleshoot the junction box if there are any issues with the wiring or the sensors. This can lead to longer downtime for the system and potentially cause delays in data collection.

3.4.1.2 Conceptual design for improved functionality

In order to overcome the challenges posed by the high friction and tilting of the reel holder, it was imperative to seek a viable solution. One approach was to modify a plastic hose reel or create a customised reel using 3D printing technology. However, considering the scalability of this project, particularly the deployment of sensors at greater depths with minimal reel revolutions would require a larger diameter of the reel. Unfortunately, producing large diameter 3D printed reels increases the cost of the project, while also raising the risk of failure due to the accumulation of weight in printed parts. One possible workaround is to divide the 3D printed reel into smaller sections, but this requires extensive design work to create sturdy connections between the

parts. So, modifying a ready-made hose reel as shown in figure 3.14 is much more cost and time efficient.



Figure 3.14 Plastic hose reel.

The initial design of using a slip ring to feed the wires from the sensor into the junction box mounted on top of the spinning hose reel proved to be problematic due to the added weight and stress on the reel, as well as the potential for electrical interference and failure. Therefore, an improved design was proposed to feed the sensor wire through a hole at the side of the spinning reel, eliminating the need for a slip ring altogether.

3.4.2 Motor station and potentiometer holder

3.4.2.1 Initial conceptual design and prototype

The initial design of the motor station uses a plywood board to mount the window motor as well as the potentiometer. Figure 3.15 shows the mounting of the motor station.



Figure 3.15 Initial motor station.

However, using a wood board to mount the motor and potentiometer in a water environment is not recommended because wood is a porous material and can absorb water over time. As the wood absorbs more water, it can become swollen, warped, or even rot. This can affect the stability and durability of the motor station, potentially leading to damage or failure. In addition, wood is also susceptible to degradation from exposure to UV rays, which can cause it to become brittle and weaken over time. This can also contribute to the failure of the mount. Therefore, it is recommended to use materials that are more resistant to water and UV exposure, such as plastics, metals, or composite materials.

Other than that, using wood board causes it to be hard to adjust the positions of all the components that are mounted on the wood board. This is because the screw positions of all the components are drilled onto the wood board. To have flexibility to adjust the positions of the components would require routing a slot on the wood board. Routing a slot onto the wood can cause problems such as the slot not being straight, over cutting the slot width and wiggling screw. This is very crucial as the tension of the timing belt depends on adjusting the position between the motor and the potentiometer.

On the other hand, the junction box that was placed behind the motor station as shown in figure 3.7. The junction box contains the 12V Lithium Ion

battery, MD10C window motor driver, and XL4015 current and voltage regulator. Figure 3.16 shows the contents of the power supply junction box. The XL4015 current and voltage regulator converts the 12V from the battery to 5V and supply it to the potentiometer. XL4015 is chosen for its stability since the potentiometer is a crucial component in the design and having a fluctuating voltage might affect the overall system. The lack of wire configuration and component assembly makes it difficult to do troubleshooting in the future. The breadboard should be eliminated as well since it might cause bad connection and possible data loss.

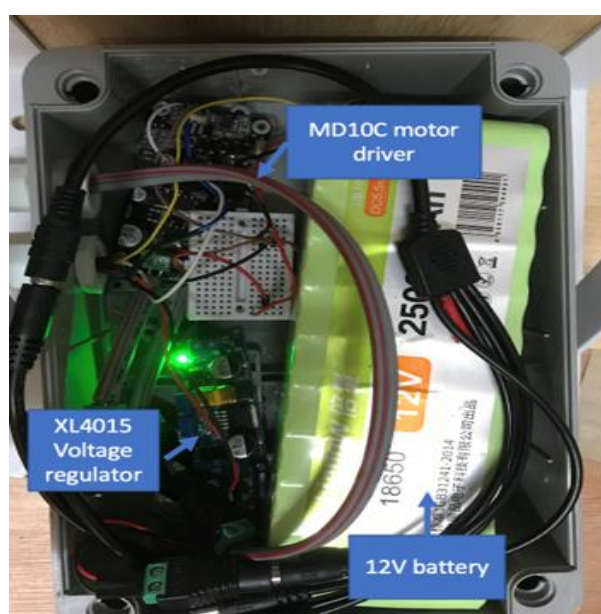


Figure 3.16 Power supply junction box.

3.4.2.2 Conceptual design for improved functionality

Aluminum is a better material choice than wood for a water environment due to its inherent properties. Unlike wood, aluminum is resistant to corrosion, which can occur when the material is exposed to water or moisture for prolonged periods. Additionally, aluminum is lightweight, making it easy to handle and install. It also has good mechanical properties, which allows it to withstand forces and impacts that can occur in water environments.

However, there are also possible issues or failures that can occur when using aluminum in a water environment. For example, if aluminum is

not coated or treated properly, it can still be subject to corrosion, particularly in saltwater environments. Additionally, if aluminum comes into contact with dissimilar metals, galvanic corrosion can occur. Figure 3.17 shows the conceptual design of the motor station using aluminum profile as structure.

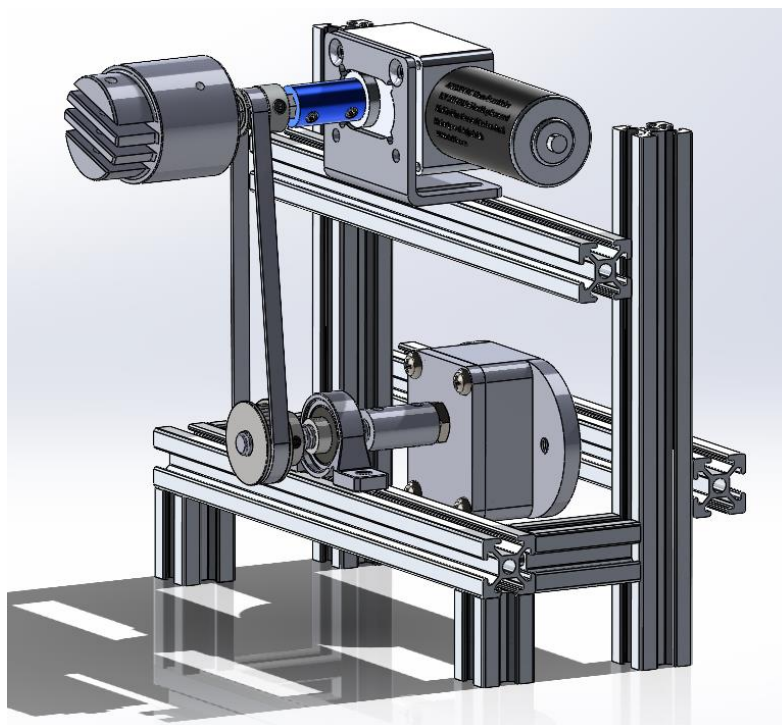


Figure 3.17 Motor station conceptual design.

This design has lighter weight compared to using the wood board. Aluminum profiles also makes adjusting positions of the components easier as it just requires moving the screw and slot nuts to the desired position. The vertical aluminium profile is taller than needed as the motor position can be adjusted according to the height of the plastic reel. The tension of the timing belt can also be adjusted by moving the potentiometer horizontally. This would reduce the risk of slippage on the timing belt.

Since the junction box that was originally mounted on the reel are now moved to the side in the improved conceptual design for sensor reel, the motor driver, current regulator can be moved to the main junction box leaving only the battery inside the power supply junction box. All the breadboard in the circuit should also be eliminated and replaced using a perf board to ensure good connection between the components.

3.4.3 Reel coupler and potentiometer

One of the challenges of this project is connecting the motor to the hose reel whether it is using metal or plastic hose reel. This is because the metal and plastic hose reel are designed to be spun by hand thus it has a handle on one side while the other side is designed to connect the water hose. Other than that, the design of the reel coupler should take into consideration that the motor needs to turn the potentiometer as well.

3.4.3.1 Initial conceptual design and prototype

The choice of window motor for the project was based on its availability and low cost. However, the design of the window motor presented challenges when it came to incorporating a gear transmission. This was due to the attached motor coupler, which featured three screw holes arranged in a circular pattern for mounting other components as shown in figure 3.18. In order to connect the motor to the reel, an additional customised reel coupler needed to be designed and 3D printed, taking into account the need to transmit rotation to a potentiometer for absolute position measurement. Given these constraints, it became clear that gear transmission would be more troublesome than belt transmission. As a result, a timing belt drive transmission was chosen as the ideal solution, offering several advantages over a gear transmission. It is more flexible, easier to install, and can transmit power with less noise and vibration. Additionally, timing belts are less susceptible to wear and tear than gears especially when it is 3D printed, resulting in a longer lifespan and decreased maintenance costs.



Figure 3.18 Window motor and built-in coupler.

The decision to use 3D printing for the reel coupler was based on several factors. First, 3D printing allows for the creation of complex and custom shapes that may not be possible with traditional manufacturing methods. This was particularly important in our case since the design of the reel coupler needs to connect the window motor to the reel while also act as a driving pulley to accommodate the potentiometer for position sensing. Second, 3D printing is a relatively fast and inexpensive process compared to other manufacturing methods such as CNC machining or injection molding. This allowed quick adjustment to be made to the reel coupler design as needed, without incurring significant time or cost penalties.

However, it is important to note that 3D printed parts may have limitations in terms of their mechanical properties. For instance, the strength and durability of 3D printed parts may not be on par with those of injection molded or machined parts. Additionally, the surface finish of 3D printed parts may not be as smooth as that of machined parts, which may affect the performance of the system.

Despite these limitations, 3D printing proved to be the most suitable manufacturing method for our reel coupler due to its convenience, speed, and customizability. Machining the reel coupler can be taken into consideration once the 3D printed parts has proved to be viable. The design of the reel coupler optimized for 3D printing, such as ensuring that it had adequate wall thickness and infill density to improve its strength and durability.

Next, we would need to put into consideration on the belt pulley ratio since the potentiometer can only turn up to ten revolutions. Assuming that the belt pulley ratio is 1:1, calculate the maximum length of the sensor.

Diameter of reel, $S_{reel} = 8.8 \text{ cm}$

Outer circumference of reel, $C_{reel} = \pi D$ (3.1)

$$= \pi \times 8.8 \text{ cm}$$

$$= 27.646 \text{ cm}$$

Sensor wire length = Number of revolution $\times C_{shaft}$ (3.2)

$$= 10 \times 27.646 \text{ cm}$$

$$= 276.46 \text{ cm}$$

$$= 2.7646 \text{ m}$$

If the belt ratio is 1:1, the maximum depth that the sensor can reach is less than 2.7646 meters since the sensor would need to connect to the main junction box that is located beside of the reel. Doubling the driven pulley ratio to 1:2 also doubles the maximum depth that the sensor can reach which in this case would be 5.5292 meters. Considering that the driving and driven pulley are 3D printed and increasing gear ration would result in bigger 3D printed parts as well as higher cost and longer printing time, a 1:2 ratio is sufficient for this application. The conceptual design and prototype of the belt pulley system as shown in figure 3.19.

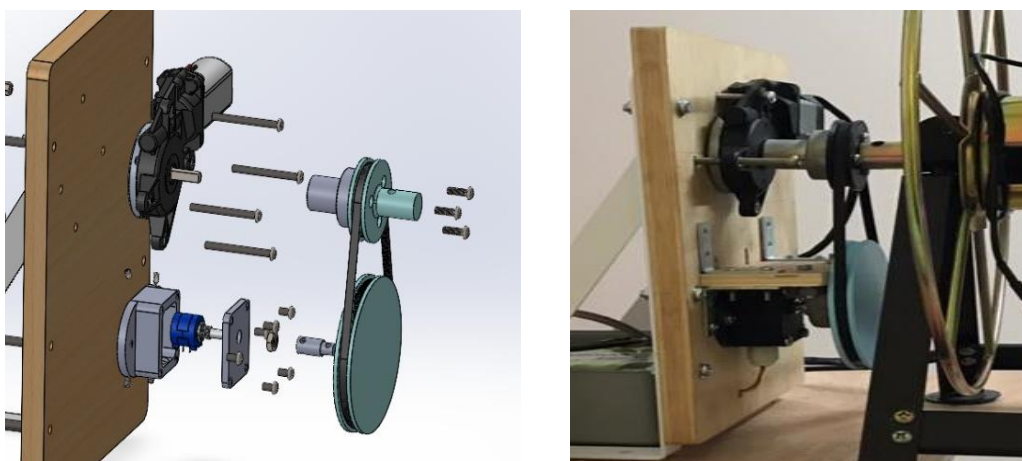


Figure 3.19 Belt and pulley system.

As seen in figure 3.19, the driven pulley connects to a customised coupler to turn the potentiometer. The driven pulley was purposely designed to have heat nut insert so that it can lock onto the potentiometer coupler using screws. Figure 3.20 shows the thread insert on the driven pulley.

The potentiometer coupler is designed to have screw holes so that the screw can go through the nut insert at the driven pulley and lock onto the potentiometer coupler. The potentiometer coupler also has a thin extruded rectangle inside to match the thin groove on the potentiometer as shown in figure 3.21.

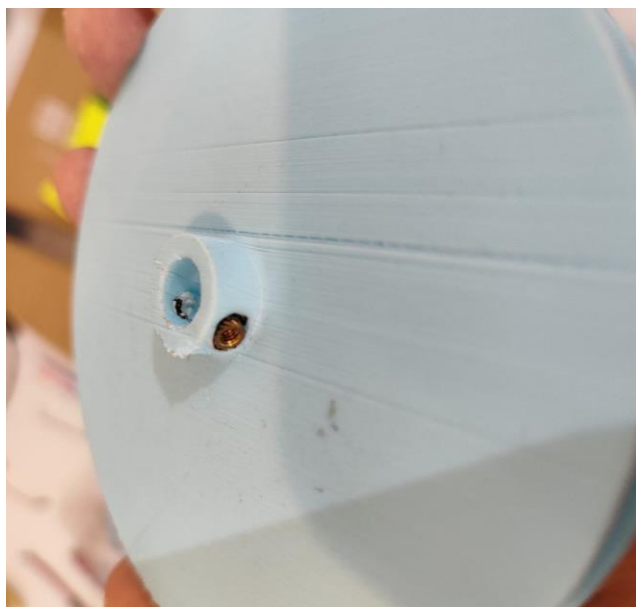


Figure 3.20 Driven pulley and nut insert.

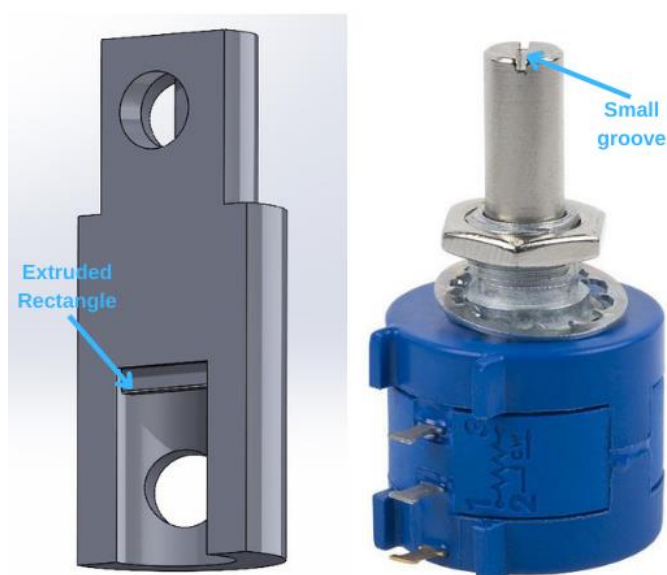


Figure 3.21 Potentiometer and coupler.

The customised reel coupler using was able to successfully drive both the metal reel and the timing belt which turns the potentiometer. Its design enables it to be mounted on the motor driver using the three screw holes that are present on the motor driver built-in coupler. Using a belt pulley of 1:2 ratio resulted in an excessively big diameter of the driven pulley. The total diameter of the driven pulley is 110mm while the diameter of the driving pulley is 55mm. The price of 3D printing the belt pulleys with 100% infill and the potentiometer mount costs a total of RM 59.49.

The 3D printed additional reel coupler, while convenient to produce, proved to be insufficient for the high torque required to rotate the metal reel. The material used in 3D printing has inherent limitations in terms of mechanical properties, such as strength and durability, which can result in premature failure when subjected to high loads or stresses. This issue was compounded by the driven pulley's big size and weight, further increasing the load on the reel coupler. Figure 3.20 shows broken driven pulleys due to high torque. All the driving pulleys' breaking spot are similar thus proving that it is due to the high torque that the driving pulley is experiencing at the connection with the metal reel as shown in figure 3.9. The driven pulley and the potentiometer coupler have also break in the process as shown in figure 3.23.



Figure 3.22 Broken reel coupler/driving pulleys.



Figure 3.23 Broken driven pulley and coupler.

The cause behind the broken driven pulley was because of the significant difference in size between the driven pulley and the potentiometer coupler. Figure 3.24 shows the free body diagram of the driven pulley.

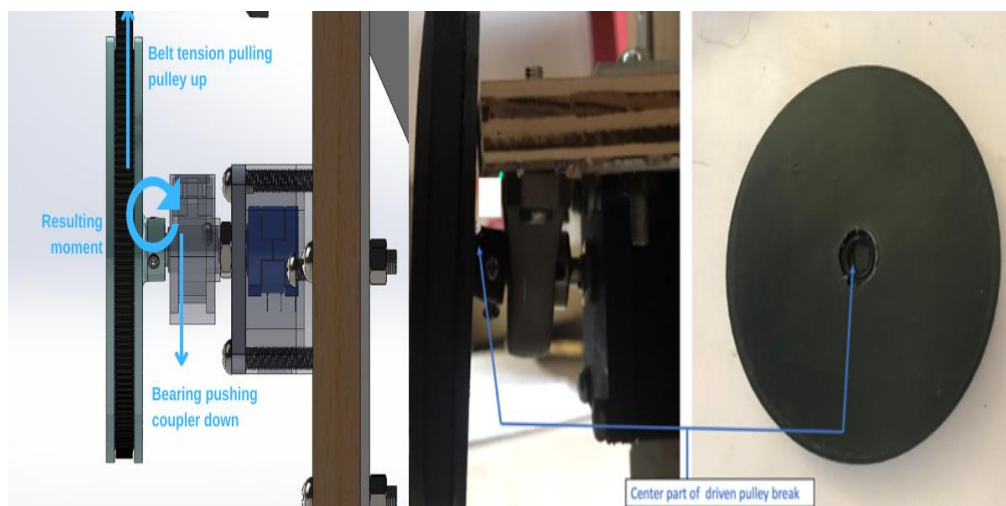


Figure 3.24 Free body diagram.

Additionally, the size of both the reel coupler and the driven pulley proved to be a challenge when it came to printing. Printing large parts with higher infill percentage can take a considerable amount of time, increasing production time and cost.

Given these limitations, it became clear that an alternative solution was needed. One possible solution could be to use a machined reel coupler and pulley, which can offer higher strength and durability compared to 3D printed parts. Machining can produce parts with greater accuracy and tolerances, which can improve performance and reduce the risk of failure. While machining may be more expensive initially, the increased durability and longer lifespan of machined parts can result in lower maintenance and replacement costs in the long run. However, this would mean that the window motor must be replaced by an alternative due to the window motor's physical configurations. Another potential solution could be to use a different type of material for 3D printing, such as carbon fiber reinforced filament, which can offer improved mechanical properties and durability compared to standard PLA or ABS filaments.

3.4.3.2 Conceptual design for improved functionality

To address the issues of breaking 3D printed reel couplers and insufficiently supported pulleys, an alternative solution was pursued. A worm gear motor with a D-shape shaft was chosen as a replacement for the window motor. This motor offers higher torque capabilities, which is necessary for rotating the reel. The motor with D-shape shaft will be coupled to an Allen key, which allows for the use of readily available, purchasable aluminium GT2 pulleys with 20 teeth, 8mm bore. The aluminum GT2 pulleys are more durable than 3D printed alternatives. The Allen key is coupled to the plastic reel through a custom 3D printed coupler, which takes advantage of the lower friction of the plastic reel compared to the previous metal reel. Figure 3.25 shows the conceptual design of the improved belt drive system.

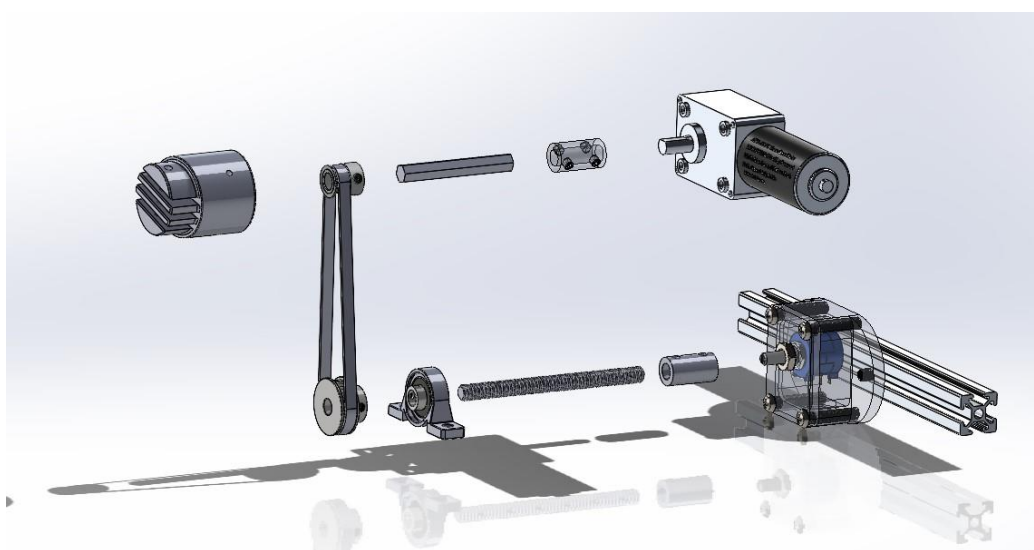


Figure 3.25 Belt drive and reel coupler

The use of a 40-tooth, 8mm bore aluminum GT2 pulley, which is directly coupled to the potentiometer, helps to ensure the accuracy and reliability of the position sensing. The potentiometer shaft will be coupled to a screw which will then be coupled to the driven pulley.

Overall, this approach reduces the number of 3D printed parts and makes use of readily available components, making production more straightforward. Additionally, the use of a worm gear motor and aluminum pulleys increases the system's durability and reliability.

3.4.4 Base structure with floating pillars and solar panel

All of the components mentioned above will be mounted on top of two floating pillars as well as having a solar panel as a roof. The base structure is using aluminum profile. Figure 3.26 shows the complete structure with all of the components mentioned above. All the drawings shown are drawn to scale and according to the dimensions of the items.

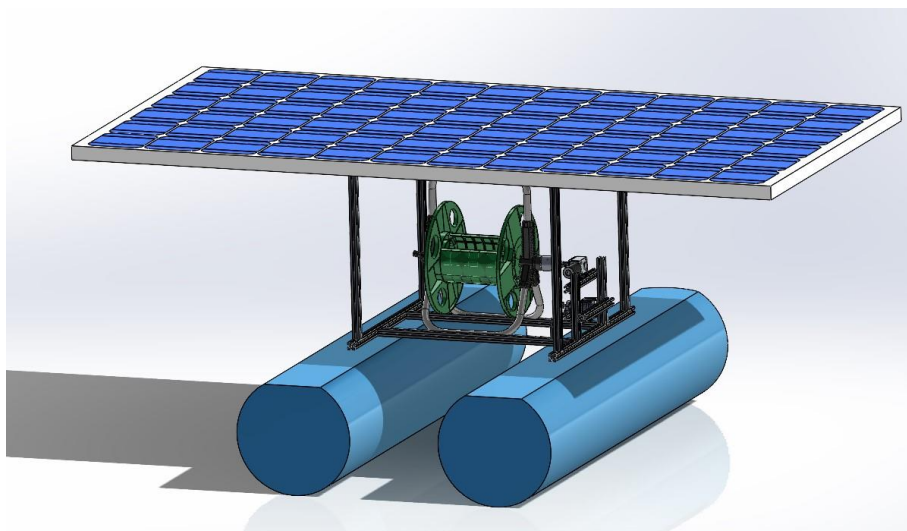


Figure 3.26 Conceptual design of prototype.

The floating pillars has a mounting bracket tied around it so the aluminium profile can be screw to the mounting bracket. Figure 3.27 shows the floating pillars with brackets tied around it.



Figure 3.27 Floating pillars.

3.5 Prototype electronic components

The block diagram is as shown in figure 3.6 and section 3.3 has explained the interaction between the electronic components. This section will be focused on the electronic components that will be used in the prototype and its respective specifications. The components of the initial prototype design and improved prototype will be discussed and compared altogether. All of the electronic components in the prototype are as shown in figure 3.4 block diagram.

3.5.1 Microcontroller

The Arduino Uno is a comprehensive microcontroller board that comes equipped with a 16 MHz quartz crystal clock, USB connectivity, power input jack, and a reset button. With 14 digital input/output pins and 6 analogue inputs, it provides a robust foundation for a wide range of projects. It is an open-source platform that allows users to easily program and develop electronic projects. With its small size, low power consumption, and easy-to-use software, the Arduino Uno is a great choice for a wide range of applications, including robotics, automation, and data logging.

One of the key features of the Arduino Uno is its ability to be put into a power-down sleep mode, which is ideal for my application. This is achieved by disabling the brown-out reset, which ensures that the microcontroller can operate at very low power levels. This feature is particularly useful for battery-powered projects, as it allows the microcontroller to run for extended periods of time without needing a recharge. Additionally, the Arduino Uno is highly customizable, with a wide range of shields and sensors available to extend its capabilities.

Figure B-3 and B-4 shows the details of power-down sleep mode that is provided in the datasheet. It turns off most of the operations in the Arduino Uno therefore achieving very low power consumption.

3.5.2 DC motor

The initial prototype uses a window motor as it offers several advantages over other types of motors. The window motor is as shown in figure 3.18. First, the design of the worm gear allows for high torque transmission, making it an ideal choice for applications that require a lot of power. Additionally, the

worm gear is designed in such a way that it is almost impossible for the gear to turn the worm, making it a good choice for applications where back driving is a concern. This means that the worm gear motor can easily turn a gear, but it is difficult for the gear to turn the worm. Furthermore, the worm gear motor has a self-locking feature, which means that the motor can hold its position without the need for a brake or other locking mechanism. This feature is particularly useful in applications where precise positioning is required, such as robotics or automation. Finally, worm gear motors are typically more compact and efficient than other types of motors, making them a good choice for applications where space is at a premium. Figure B-5 shows the specifications of the window motor provided by the seller.

As stated above in the conceptual design, the design of the window motor coupling makes it hard to use ready-made aluminium GT2 pulleys as the aluminium GT2 pulleys usually has a bore size of 5mm or 8mm while the coupling has a diameter of 40mm.

The 5840-312ZY is a suitable alternative for the window motor. It also operates at 12V and uses worm gear reduction. The specifications for the 5840-312ZY can be referred in figure 3.5. It has an 8mm D-shape shaft which is ideal for coupling it with aluminium GT2 pulleys. The bracket for this motor is also available for it to be mounted on aluminium profiles. The rpm chosen for this prototype is 27rpm as it provides higher torque and higher rpm might causes the sensor to swing thus causing it to be entangled to the side of the reel or other foreign objects. Figure 3.29 shows the 5840-312ZY motor and its bracket.



Figure 3.28 8540-312ZY motor and bracket.

3.5.3 Motor Driver

The MD10C motor driver is a single-channel motor driver that supports bi-directional control for a single brushed DC motor. With a wide range of motor voltage from 5V to 30VDC, it provides high flexibility and can be used in a wide range of applications. It also features regenerative braking, allowing for efficient energy usage. The MD10C can handle maximum current up to 13A continuous and 30A peak, making it a robust solution for powering high current motors. The solid-state components of the MD10C provide faster response times and eliminate the wear and tear of mechanical relays. It is also fully NMOS H-bridge, allowing for better efficiency and no heat sink is required. With 3.3V and 5V logic level inputs, it is compatible with popular microcontrollers such as Arduino and Raspberry Pi. Additionally, it supports both Locked-Antiphase and Sign-Magnitude PWM operation, making it versatile for different applications. With its wide range of features, the MD10C motor driver is a reliable and efficient solution for driving brushed DC motors.

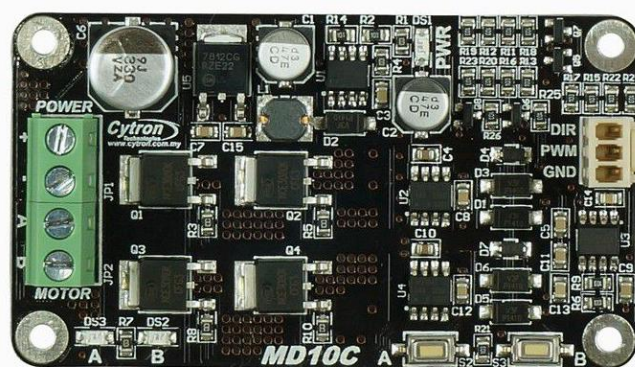


Figure 3.29 MD10C motor driver.

3.5.4 Power supply

The power supply of choice for the prototype is a 12V Lithium 18650 Rechargeable battery pack with a capacity up to 25000mAH. Figure B-6 shows the battery pack with its specification in Chinese language provided by the seller. Table B-1 shows its specification in English.

3.5.5 Real Time Clock (RTC)

The DS3231 Real Time Clock (RTC) as shown in figure 3.31 is used as an alarm clock in the prototype to notify the Arduino Uno with a pre-set time interval to collect data. The DS3231 RTC module is a highly accurate timing device commonly used in various electronic applications. It features a square wave output signal that can be used to trigger events at precise intervals, making it an ideal solution for this prototype as it can be used to interrupt the Arduino's sleep mode. This allows the Arduino to go into sleep mode during data collecting intervals. The DS3231 module offers battery backup, allowing it to continue keeping time even in the event of a power outage. The DS3231 RTC module offers a range of features that make it a reliable and convenient timekeeping solution. One of its key features is its ability to maintain accurate timekeeping over a wide temperature range, from -40°C to $+85^{\circ}\text{C}$, making it suitable for use in a variety of environments. It also includes a built-in temperature sensor that provides an accurate temperature reading.



Figure 3.30 DS3231 RTC module.

3.5.6 Communication Device

The SIM900A module as shown in figure 3.32 is a versatile device that facilitates communication between the Arduino board and the cloud platform. This module employs UART interface for two-way communication and supports the 900/1900 MHz frequency band. One of its notable features is its ability to send and receive SMS alerts provided a SIM card is inserted into the module. We could use this feature to alert farmers or adjust the system parameters using the contents of received SMS. More importantly, the SIM900A module can support GPRS (General Packet Radio Service) data transfer, making it an ideal choice for applications that require wireless

communication. GPRS is a mobile data service that allows devices to send and receive data over a cellular network. It was introduced in the 2G GSM networks and provided an always-on packet-switched data service, allowing for faster and more efficient data transmission than the circuit-switched data services used in previous mobile networks. The module has a compact design with a small form factor, allowing for easy integration into various devices. However, one important thing to note is that the SIM900A module can take up to 2A when transmitting data as shown in figure 3.35. So, it shouldn't draw current from the Arduino Uno as the maximum current output of the Arduino Uno is 40mA as shown in figure B-2.



Figure 3.31 SIM900a GSM/GPRS module.

3.5.7 Temperature Sensor

The DS18B20 as shown in figure 3.33 is a versatile, pluggable terminal waterproof temperature sensor that is commonly used in various applications including floor temperature detection and hot water tank temperature control. This sensor comes with a usable pull-up resistor that makes it easy to connect and integrate with other devices. The DS18B20 has a temperature range of -55°C to $+125^{\circ}\text{C}$, and the temperature resolution can be adjusted between 9 to 12 bits. The sensor operates with a supply voltage between 3.0V to 5.5V, making it suitable for use with various platforms such as Arduino and Raspberry Pi. Its output wires include yellow (Data), red (VCC), and black (GND), while adapter cables for DATA, VCC, and BLK are also included. It's important to note that the lead can only withstand the highest temperature of 85 degrees Celsius. With its compact and waterproof design, the DS18B20 is

an excellent choice for projects that require accurate and reliable temperature measurement in harsh environments.



Figure 3.32 DS18B20 temperature sensor.

3.5.8 Potentiometer

The 3590 as shown in figure 3.21 is a high precision multiturn rotary potentiometer, which is widely used in various applications such as audio equipment, industrial control systems, and other electronic devices that require accurate control of resistance. This wire wound potentiometer is designed to provide precise and reliable performance, with a resistance range of 500~100K and a tolerance of 5% or 2%. The 2W rated power and max operating voltage of 500V make it suitable for a wide range of applications.

The 3590 multiturn potentiometer is designed with a wire wound technology, which ensures high precision and reliability. It features a compact and durable design that makes it easy to install and use in a variety of environments. With an operating temperature range of -55°C ~ $+125^{\circ}\text{C}$, this potentiometer is suitable for use in harsh environments where temperature fluctuations are common.

The 3590 multiturn potentiometer features a rotary design, allowing users to adjust the resistance value with high precision. Its multiturn design enables it to make multiple revolutions, providing fine-tuning capabilities for precise adjustments. With its high-precision performance, this potentiometer is an ideal choice for applications that require accurate control of resistance, such as audio equipment, industrial control systems, and other electronic devices.

3.5.9 Buck converter/ current voltage regulator

The prototype uses a LM2596 buck converter module and a XL4015 current and voltage regulator as shown in figure 3.34. The LM2596 are used to convert the 12V from the battery into 9V to power up the Arduino Uno. The LM2596 acts as a safety to prevent current surge as the battery is used to power up the motor as well. The XL4015 is used to power up the DS3231 RTC module, SIM900A GSM/GPRS module and DS18B20 temperature sensor. The XL4015 also protects these modules from current surge. The SIM900A requires up to 2A when operating so it is more suitable to be powered by the battery through XL4015.

The LM2596 is a powerful and efficient buck converter switching power module designed to deliver a stable and precise output voltage. It features an adjustable output voltage range from 1.25V to 35VDC and can handle input voltages ranging from 3.2V to 40VDC, making it an ideal solution for a wide range of applications. With its on-board multiturn potentiometer, you can easily adjust the output voltage to your desired level with precision. The LM2596 is also a step-down converter, meaning the input voltage should be greater than the output voltage plus 1.5V. It can deliver a maximum peak output current of 3A and a continuous output current of 2A, making it ideal for powering various electronic devices. Additionally, it has a switching frequency of 65KHz, ensuring high power efficiency of up to 92% and minimizing output voltage ripple to less than 30mV. With its rated power of 10W and operating temperature range of -45°C to +85°C, the LM2596 is a reliable and versatile power module for your electronic projects.

The XL4015 DC/DC buck module is a highly efficient and reliable module designed to drive a 5A load with low ripple and excellent line and load regulation. With a fixed frequency PWM buck of 180 kHz, this module is capable of providing high power and efficiency with a power indicator to keep you informed. This module is ideal for various applications, including battery, power transformer, DIY adjustable regulated power supply, LCD Monitor and TV, portable instrument power supply, and telecom/networking equipment, among others.

The XL4015 DC/DC buck module features an input voltage range of 4~38VDC and an output voltage range of 1.25-36V continuously adjustable.

With an output power of 75W and an efficiency rating of up to 96%, this module delivers excellent performance while ensuring that your device stays safe with built-in thermal shutdown, current limit, and output short protection functions. However, note that the module does not come with input reverse polarity protection, so a high-current diode may need to be added in series with the input if required. Choose the XL4015 DC/DC buck module for a reliable and efficient solution for your power supply needs.

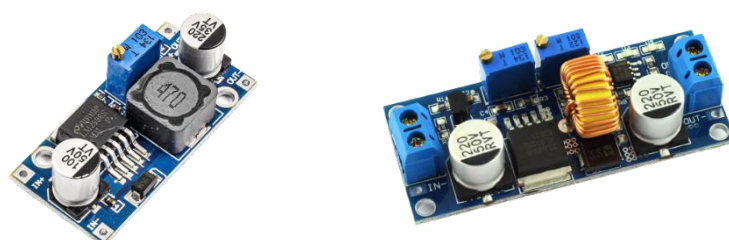


Figure 3.33 LM2596 (left) and XL4015 (right).

3.5.10 Solar Panel

The solar panel as shown in figure 3.35 is a photovoltaic (PV) module designed to generate electrical power by harnessing the energy from the sun. It is a key component of a solar power system and provides a renewable source of energy that is both environmentally friendly and sustainable. This particular solar panel is a 100-watt module with a conversion efficiency of 17% and a maximum voltage of 17.5V, making it ideal for small-scale residential or commercial applications.

With a short current of 5.98A and a maximum system current of 15A, this solar panel can generate a maximum power output of 100 watts. The MC4 socket on the panel makes it easy to connect to other components in a solar power system, such as inverters, charge controllers, and batteries.

This solar panel is built to withstand harsh outdoor conditions, with a working temperature range of -40°C to 85°C and an expected lifespan of 20 to 25 years. It also has a maximum system voltage of 1000V, making it compatible with a wide range of solar power systems.



Figure 3.34 Solar Panel.

3.5.11 Solar Charging Controller

A solar charge controller is a crucial component in a photovoltaic system, regulating the flow of electricity from the solar panels to the battery and preventing overcharging or damage to the battery. The RBL solar charge controller is a versatile and intelligent controller that can meet the requirements of various home photovoltaic systems, including home lighting systems.

One of the key features of the RBL solar charge controller is its ability to recognize 12V and 24V batteries automatically. This allows for flexible installation and operation without the need for manual configuration. Additionally, the controller has a charge current of 30A and a discharge current with a USB output of 5V/2A, making it capable of supporting moderate power needs.

The RBL solar charge controller also comes with adjustable charge/discharge control parameters, which can be set based on specific requirements, as well as settable operating modes of loads and Load Timer Setting ON/Off Hours. This flexibility allows for optimal customization and energy management.

The controller's multi-functional LCD display shows all operating data and working conditions, enabling users to conveniently switch between working modes and parameter configurations. The RBL solar charge controller uses PWM charge management, which is more efficient than conventional charging methods and can extend battery life.

To ensure safety, the RBL solar charge controller has built-in protection features, including short-circuit protection, open-circuit protection,

reverse protection, and over-load protection. These features help prevent damage to the controller and other components in the system, making it a reliable and secure solution for solar charge management.

3.6 Operation flowchart

Figure 3.36 shows the operation flowchart of the floating aquaculture sensor station. The program first initializes all the modules and input/output pins. Then it will check the time whether it is within the time range for a new data collection cycle with a 2-minute toleration for data collection. For example, if the time interval is set every 15 minutes, it will check whether the time is xx:00, xx:15, xx:30 or xx:45. The 2-minute toleration means that if the time is xx:01:00 it will still proceed to a new data collection cycle while xx:02:01 will not.

If it is within the time range, it will activate the motor to deploy the sensor while taking reading from the potentiometer. When the potentiometer reached the pre-set threshold, it will stop. Then it will take temperature readings from the temperature sensor and send this data to designated server.

After sending data to the server or if it is not within the time range for new data collection, it will activate the motor to retract back the sensor until the potentiometer reaches a pre-set threshold. This is because if the Arduino was woken up not within the time range of next interval, most probably it is because of power outage. In this case, the sensor's position might not be fully retracted and thus needs to activate the motor to retract the sensor. Then the Arduino will set an alarm through the RTC module for the next wake time according to the pre-set data collection time interval. After that the Arduino will go into power down mode until the RTC module sends a signal to interrupt the Arduino. When the Arduino wakes up, it will loop back to checking the time whether it is within the time range for a new data collection cycle.

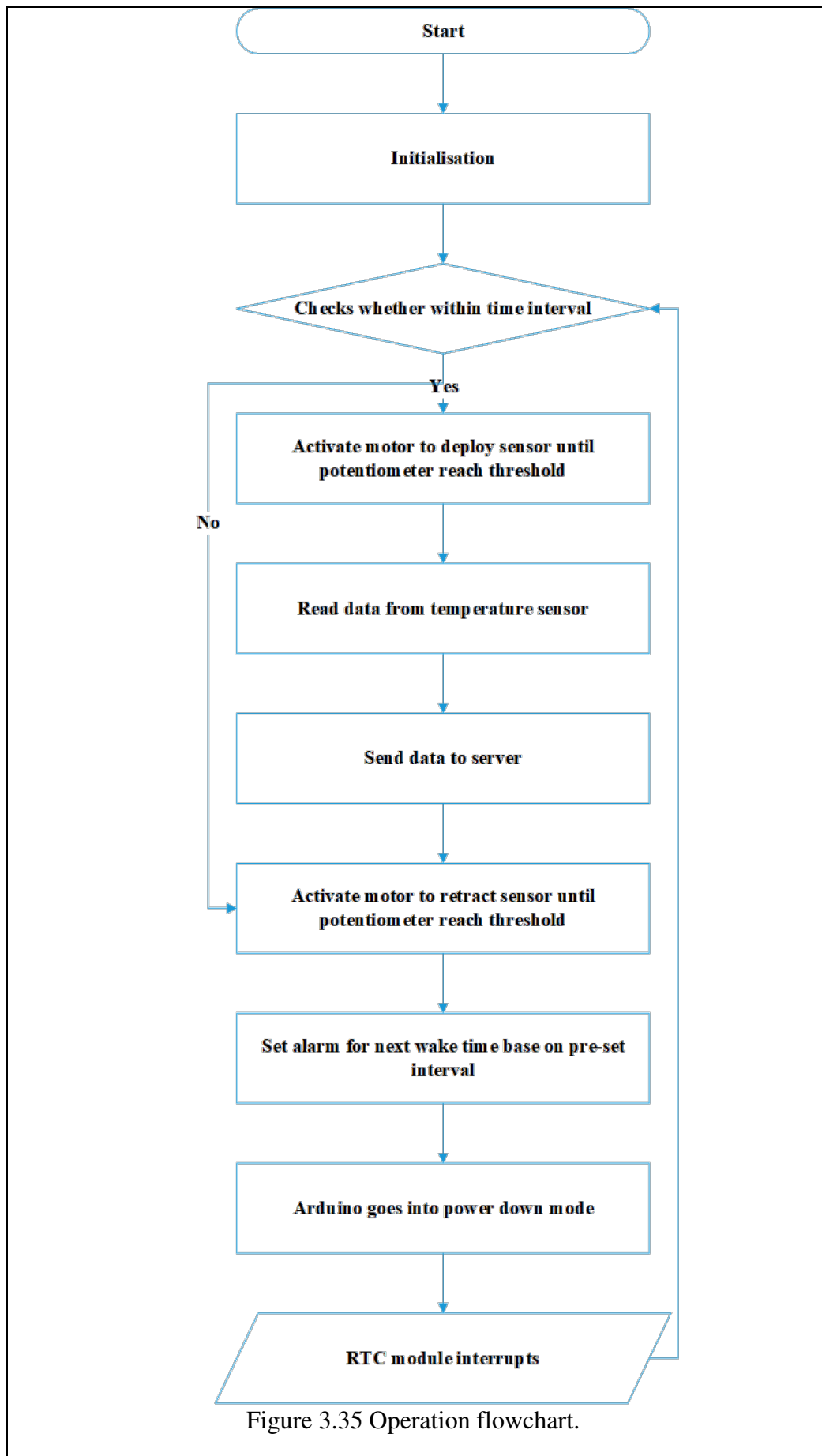


Figure 3.35 Operation flowchart.

3.7 Software

3.7.1 CAD drawing

This project uses SolidWorks to draw out the conceptual design. It allows for the creation of virtual prototypes that can be quickly and easily modified, allowing designers to experiment with different design options without the need for physical prototypes. This saves time and money by reducing the number of physical prototypes that need to be built and tested. SolidWorks is a computer-aided design (CAD) software widely used by engineers and designers to create 3D models of mechanical parts and assemblies. Developed by Dassault Systems, SolidWorks provides a comprehensive set of tools for creating, simulating, and visualizing products, from initial concept to final production.

One of the key advantages of SolidWorks is its user-friendly interface, which makes it easy for both novice and experienced users to create complex designs. The software's intuitive interface, combined with its robust functionality, enables users to quickly create detailed 3D models, simulate mechanical behavior, and test design variations. Parts created in SolidWorks can be saved as STL file format which then can be put into 3D printing software such as Ultimaker Cura or ChituBox.

SolidWorks also offers a range of advanced features that set it apart from other CAD software, including built-in finite element analysis (FEA) tools, photorealistic rendering capabilities, and integrated simulation tools for testing and optimizing product designs. Additionally, the software's ability to seamlessly integrate with other design and engineering tools makes it an ideal choice for collaborative design and manufacturing environments.

3.7.2 IDE

PlatformIO is an open-source ecosystem for embedded development that provides a unified development platform for microcontrollers. It is an extension for Visual Studio Code that supports a wide range of embedded boards, frameworks, and libraries. PlatformIO offers a more efficient and streamlined development experience for embedded systems compared to traditional development tools like the Arduino IDE.

One of the key advantages of PlatformIO is its support for a wide range of development boards, including popular boards like Arduino, ESP32, and Raspberry Pi. It also supports multiple frameworks and libraries, including Arduino, CMSIS, and FreeRTOS. This means that developers can choose the best hardware and software components for their project without worrying about compatibility issues.

Another advantage of PlatformIO is its advanced features, such as support for multiple projects in a single workspace, real-time code analysis, and debugging tools. It also provides a powerful command-line interface (CLI) that allows developers to perform tasks like building, uploading, and monitoring code directly from the terminal.

Compared to the Arduino IDE, PlatformIO offers a more modern and flexible development experience that is better suited for professional embedded systems development. It provides a more comprehensive set of features, including advanced debugging tools, project management capabilities, and support for multiple frameworks and libraries. Additionally, it has a more user-friendly interface that makes it easier to manage and navigate code projects.

3.7.3 Circuit design

This project uses Fritzing mainly for its graphics that makes the schematic colourful and easy to recognise. Fritzing is an open-source CAD software designed for creating and designing electronic circuits. It is a user-friendly program that allows users to easily create and document their electronic designs. With Fritzing, users can create schematics, PCB layouts, and even breadboard views of their designs. This allows for a more complete and intuitive representation of the electronic circuits, making it easier for users to understand and modify their designs.

Fritzing also comes with a wide variety of electronic components that can be used in designs, including popular components like resistors, capacitors, and LEDs, as well as more specialized components like sensors and microcontrollers. Additionally, Fritzing has an active community of users that create and share their own custom parts and designs, making it easy to find and use components that may not be included in the standard library.

One of the key advantages of Fritzing is its ease of use, particularly for those who are new to electronic design. The user interface is simple and intuitive, with drag-and-drop functionality for adding components to the design. Fritzing also includes helpful features like automatic routing, which simplifies the process of creating connections between components.

3.7.4 Data visualization

This project uses Grafana for data visualization set up by Dr. Alex from UTAR Kampar campus. Grafana is an open-source, web-based platform that allows users to create and display real-time data visualizations and dashboards for various applications. It is designed to be highly customizable and scalable, with support for numerous data sources and integrations with popular tools and services.

Grafana is often used in IT and DevOps contexts to monitor and visualize metrics related to infrastructure, applications, and other systems. It can also be used for business intelligence, IoT, and other data-driven applications.

One of the key features of Grafana is its support for a wide range of data sources, including time-series databases, relational databases, cloud monitoring services, and many others. This makes it a flexible and versatile tool that can be used to visualize and analyze data from a variety of sources.

In addition, Grafana offers a wide range of visualization options, including charts, graphs, tables, and more. Users can customize these visualizations to meet their specific needs and can also create interactive dashboards that allow them to drill down into data and gain deeper insights.

3.8 Summary

First step of the project is proper planning and estimating time needed to develop the prototype. After that, a literature review is done on similar topics to find out the state of the matter. Conceptual design is simulated and visualise through different software before deciding on the final design and components. After finalising the design is a series of components buying, assembly and testing before the deployment at actual site.

Preparing a block diagram that shows the interactions between different components help to create the system architecture needed for the floating aquaculture sensor station to operate smoothly and efficiently.

Based on the initial conceptual design, there has multiple improvements that needs to be implemented to ensure a smooth and reliable operation of the floating aquaculture sensor station. Most of the improvements are on the mechanical parts of the sensor station, including the sensor reel, DC motor, motor station, reel coupler, belt drive system and potentiometer mounting.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Modified and improved prototype

Several modifications and improvements that were discussed in the previous chapter are carried out on the initial prototype. Figure 4.1 shows the improved prototype equipped with solar panel and floating pillars. The photo was taken at the prototype deployment side which is inside the agriculture park in UTAR Kampar campus.



Figure 4.1 Floating aquaculture sensor station.

This section will focus on the discussion of the modifications made to the prototype and their results. The modification will be discussed based on two different parts of the prototype relating to the conceptual design as follows:

1. Sensor Reel and Junction Box
2. Motor Station and Belt Drive Assembly with Potentiometer Control
3. Base structure with floating pillars and solar panel

4.1.1 Sensor Reel and Junction Box

As stated in the conceptual design, the prototype's metal reel will be replaced by a plastic hose reel that has less friction and less weight. Figure 4.2 shows a photo of the actual plastic hose reel that is used in the prototype.

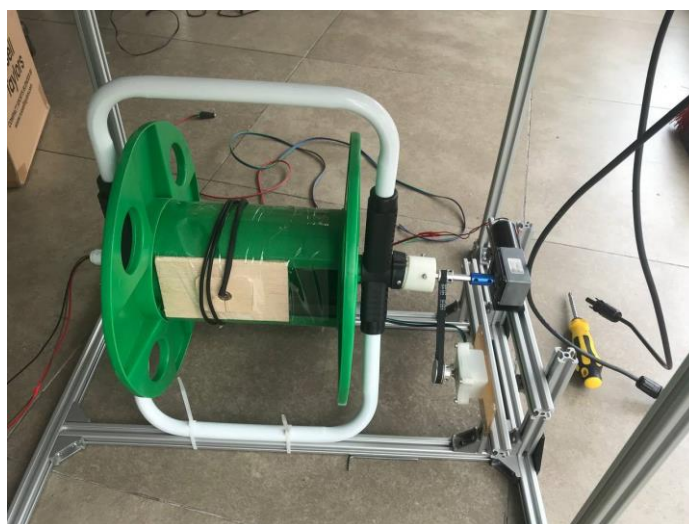


Figure 4.2 Modified sensor reel.

Although this design allows the junction box to be at the side of the sensor reel, but it would need to solve the issue of the sensor wire twisting as the reel start spinning. After some analysing on the reason behind the wire twisting, it turns out that the wire's contact with the sensor reels create enough friction to stop the wire from untwisting itself. This is because the sensor wire is made of rubber to make it water proof. To prevent the wire from twisting, a bearing was added to the design. This bearing allows the wire to untwist itself as the reel rotates, ensuring that the sensor remains in place and in proper working condition. However, the wire will still be twisted but when it is twisted too much it will start to untwist itself thus minimizing the damage to the sensor wire. The spinning of the reel will also pull the sensor wire from the junction box so the sensor wire should be fixed when it enters the sensor reel. Figure 4.3 shows the sensor wire coming out of the sensor reel through a bearing.



Figure 4.3 Bearing on sensor reel.

The original design of the hose reel has a connector on one side of the reel for connect the hose pipe and a handle on the other side of the reel as shown in figure 4.4.

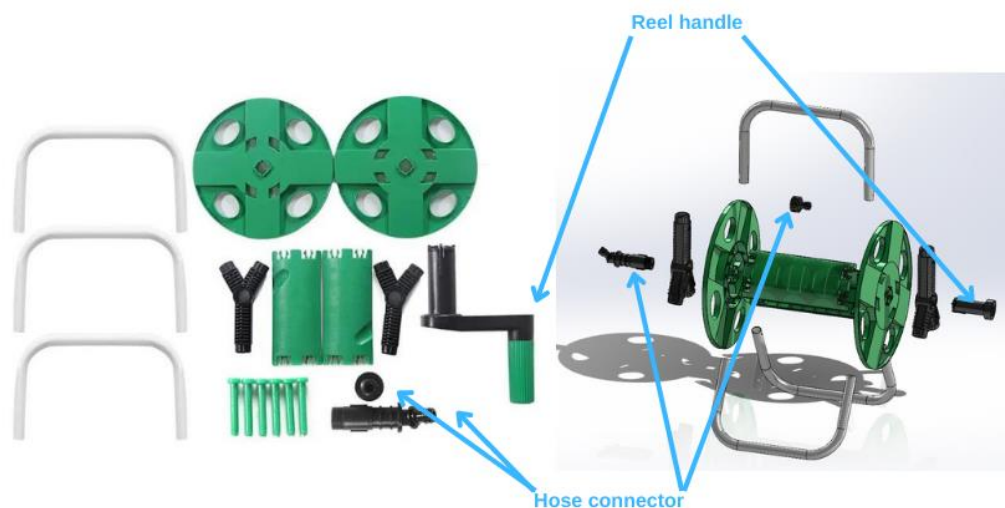


Figure 4.4 Hose Reel original design.

Both the reel handle and hose connector needed to be modified as the reel handle will obstruct the belt drive while the connector was not able to hold the sensor wire in place to prevent the spinning from pulling the sensor. To achieve this, a 3D printed screw cap is designed to replace the nozzle shape screw cap at the hose connector. The 3D printed cap should be able to fit a

cable gland that will hold the sensor wire in place. Figure 4.5 shows the 3D printed cap compared to the original hose connector nozzle and when it is installed on the prototype with cable gland.

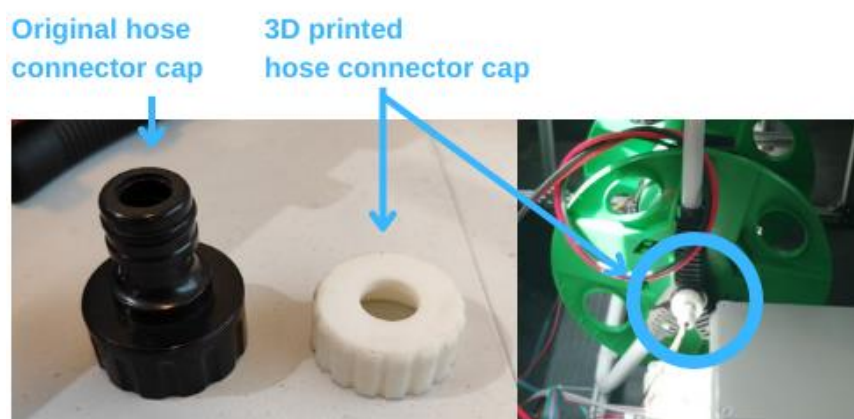


Figure 4.5 Hose connector cap.

The junction box is placed beside the sensor reel and mounted on the base structure as shown in figure 4.1. The bigger junction box that is placed at the bottom contains most of the electronic component while the smaller junction box that is placed on top contains the battery for the prototype. Figure 4.6 shows the inside of the junction boxes.

The electronics are mounted on top of a piece of acrylic using screws with nuts and washer as spacers. Then the acrylic is mounted to the junction box using screws at provided screw holes in the junction box. All of the connections between components uses either male and female DC jack or Dupont connector. This improves maintainability as the components can easily be connected and disconnected. The junction box also uses cable gland as it not only provides excellent water resistant but also hold the wires in place.

This new design improves the overall functionality and reliability of the sensor station, making it a more practical and efficient solution for floating aquaculture. By implementing this design, all the electronic components are placed beside the hose reel, minimizing the weight and stress on the reel and eliminating the potential for electrical interference and failure caused by the slip ring. The components assemble also take into consideration of improving water resistance as the prototype is exposed to water environments as well as

maintainability by using connectors to ease connecting and disconnecting components.

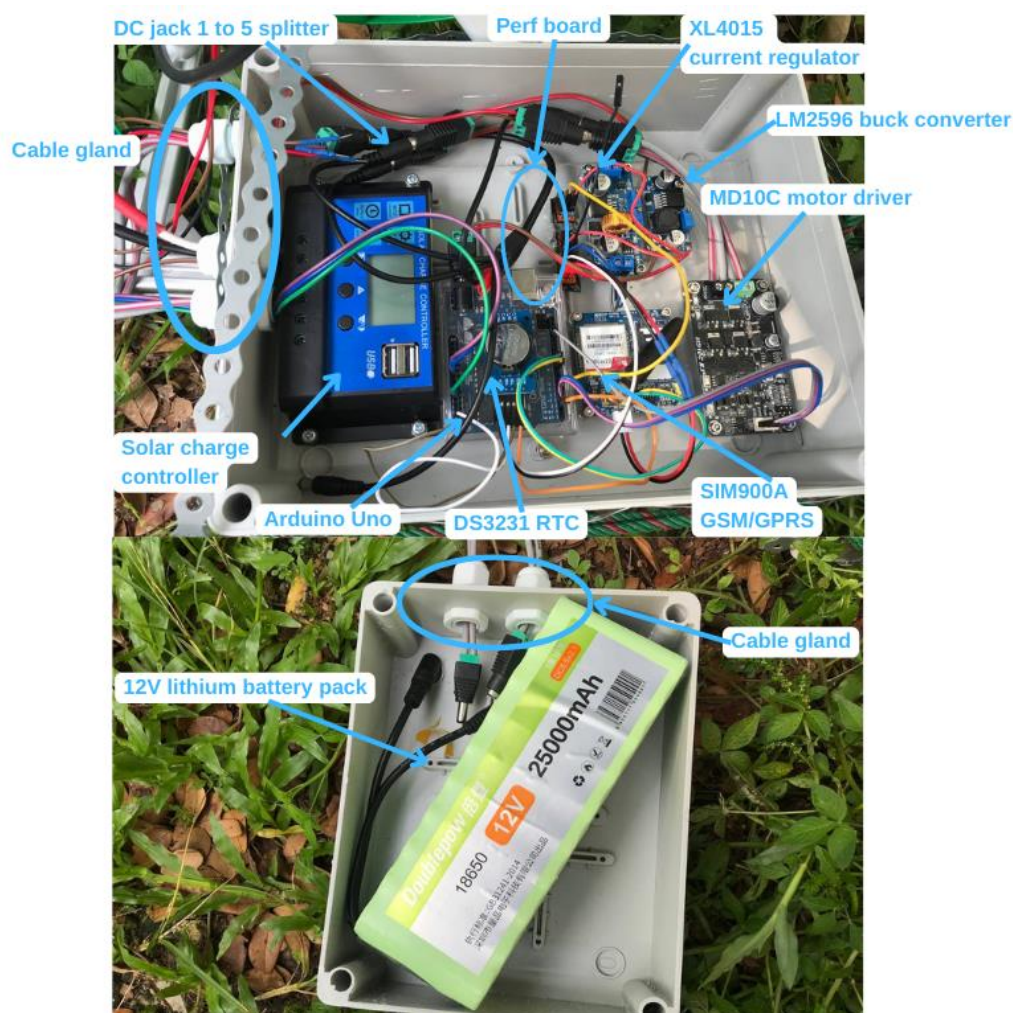


Figure 4.6 Junction boxes and circuit.

4.1.2 Motor station assembly with potentiometer control

Figure 4.7 shows the assembly of motor station, belt drive assembly and potentiometer. The original reel handle has been cut off to prevent it from obstructing the belt drive assembly. The reel is connected to the motor through a 3D printed reel coupler, Allen key with GT2 pulley and another coupler.

There are a few differences between the conceptual design and the actual prototype. The potentiometer holder designed has been improved and optimized. The previous potentiometer holder was bigger and thus increasing cost and printing time. Other than that, the potentiometer is directly coupled with the GT2 driven pulley instead of coupled to a screw and supported by a

pillow block bearing as shown in figure 3.25. This is because the timing belt minimizes the risk of slippage and thus the belt tension required is less. Other than that, the diameter of the driven pulley is 28mm which is significantly smaller than the previous 3D printed driven pulley which was 110mm. Therefore, the potentiometer can be coupled directly to the driven pulley with lesser belt tensioning and smaller driven pulley diameter. There is still a driven pulley coupler that acts as an adapter for the driven pulley and the potentiometer shaft. This is because the GT2 driven pulley has a bore size of 8mm while the shaft diameter of the potentiometer is 6.35mm. Without the driven pulley coupler, the driven pulley will become eccentric when coupled to the potentiometer shaft. Therefore, the driven pulley coupler has an inner diameter of 6.6mm and outer diameter of 7.6mm. Since the pillow block ball bearing is not longer needed, the aluminum profile that is used to mount the pillow block ball bearing is not needed as well.

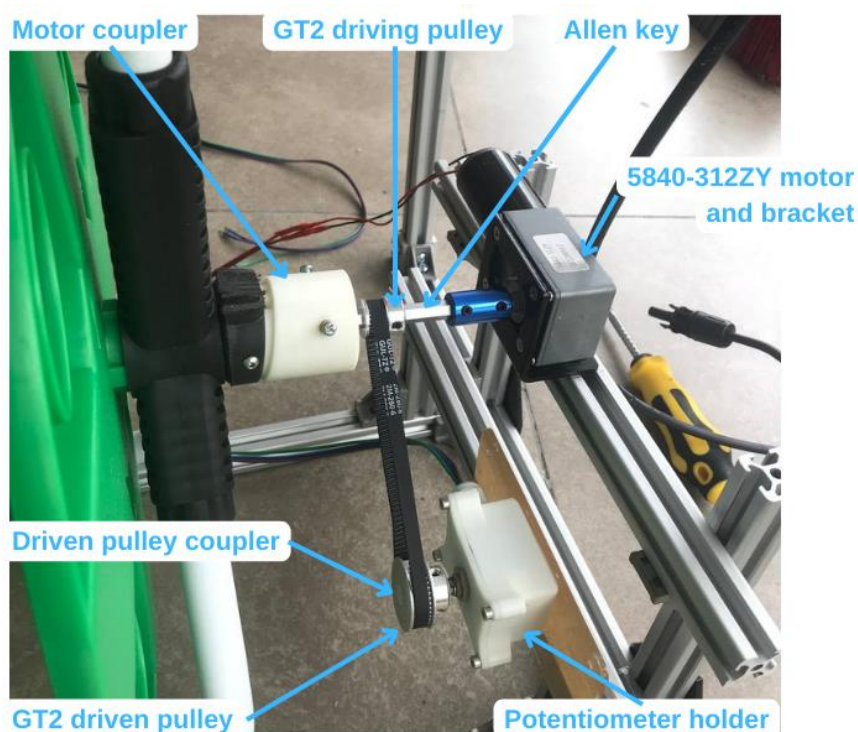


Figure 4.7 Motor station assembly with potentiometer control.

The 3D printed reel coupler is designed to match the pattern on the reel handle as shown in figure 4.8. A 3D printed reel coupler sample is printed out to make sure that it can fit into the reel handle before printing the full reel

coupler. The reel coupler has a total of three screw holes. The first screw hole at the top of the reel coupler is used to lock the reel handle to the reel handle while the other two screw are used to grip onto the Allen key as shown in figure 4.8.



Figure 4.8 Reel coupler.

4.1.3 Base structure with floating pillars and solar panel

As shown in figure 4.1, the prototype's base structure is the same as the conceptual design as shown in figure 3.26. The only addition are some ropes that are tied diagonally. The ropes act like a cross bracing to counter lateral forces that causes the vertical aluminum profile to twist. The brackets that are tied around the floating pillars are centered on the flat side of the floating pillars so that it is more balanced. The solar panel acts as a roof that protects the sensor reel, junction box and motor station assembly from raining and dry leaves.

4.2 Operation flow and code explanation

The operation flow of the floating aquaculture sensor station is as shown in figure 3.36. This section will be focused on the development of the Arduino program and explaining how each sector of code accomplishes the steps inside the operation flow.

4.2.1 Initialisation

As shown in figure C-1, the code starts by including several libraries, including the Arduino.h library for programming the Arduino board, and several additional libraries for controlling various sensors and devices.

“Wire” is a library for I2C communication protocol which is used to communicate with the DS3231 RTC module while the “ds3231” library is for controlling the DS3231 RTC such as getting time data and setting square wave trigger time.

The “AltSoftSerial” library for creating a software serial port to communicate with SIM900A. Although “SoftwareSerial” library is more common for serial communication, the main difference between the two libraries is that AltSoftSerial uses interrupts to generate the serial waveform, while SoftwareSerial uses software timing. This means that AltSoftSerial can achieve much higher baud rates than SoftwareSerial without losing accuracy. Additionally, AltSoftSerial can transmit and receive on the same pin, while SoftwareSerial requires separate transmit and receive pins. However, AltSoftSerial is limited to specific pins on certain Arduino boards and may not be compatible with all shields, while SoftwareSerial is more widely compatible but may be less reliable at higher baud rates. Therefore, the choice between AltSoftSerial and SoftwareSerial depends on the specific project requirements and hardware limitations. For the prototype, AltSoftSerial is much more suitable in this prototype as Arduino Uno is one of the boards that is compatible with the library.

The OneWire library is used for communication with OneWire devices, which use a single communication line to exchange data with the Arduino. It is particularly useful for reading data from temperature sensors like the DS18B20, which are commonly used in various projects. The DallasTemperature library is an extension of the OneWire library that

simplifies the process of reading temperature data from sensors like the DS18B20. It provides functions for getting the temperature in Celsius or Fahrenheit, setting the resolution of the sensor, and getting the device address of the sensor. These libraries make it easy to use OneWire devices with the Arduino platform and provide a lot of flexibility in terms of the types of sensors that can be used.

The Alarm library is a created purposely for this prototype as it contains methods for setting up an alarm to wake the Arduino from sleep mode specified for the prototype's usage, this will be discussed further in the following subsections.

Next, several global variables are defined, including the maximum and minimum ADC values for the potentiometer, the wake intervals for the alarm, the wake offset for the alarm, and the time tolerance for the alarm.

The pin assignments are then defined, including the pin for the potentiometer, the wake pin, and the pins for the motor controller. The OneWire bus pin is also defined.

In the setup function, the serial communication is initiated at a baud rate of 9600. The AltSoftSerial instance is also initialized at a baud rate of 9600. The Wire library is initiated for I2C communication. The wake pin and direction pin for the motor controller are defined. The current alarm is cleared using the DS3231 library. The `getCurrentTime` function is then called to set the time of the RTC to the current time. Finally, a message is printed to the serial monitor to indicate that the setup has been completed.

4.2.2 Main loop

In the main **loop** function as shown in figure C-2, the first thing that happens is that a **float** variable **temperature_value** is initialized. The **check_time** function is called, which checks whether the current time matches the set wake-up intervals and offsets within the given time tolerance. If the current time is within the time tolerance, the code inside the if statement is executed.

Inside the if statement, the **motor_motion** function is called with an argument of 1, which activates the motor to go down. The **sensor_cycle** function is then called, which reads the temperature from the DS18B20 sensor connected to the OneWire bus and passes the value to the **temperature_value**

variable. This value is then sent to the server using the **sendDataToServer** function.

After this, the **motor_motion** function is called with an argument of 0, which activates the motor to go up. Finally, the **arduino_sleep** function is called to put the Arduino into sleep mode until the next wake-up time.

4.2.3 Alarm.cpp

The Alarm library as shown in figure C-3 is a customized library that is used to store function specifically for the usage of this prototype. The functions that are related to the alarm feature is stored in this library. This is to ease software development by separating the code base into smaller chunks of code for better readability and maintainability.

The code starts by including the necessary libraries for using the DS3231 Real-Time Clock (RTC) module and the Alarm library. The DS3231 is a highly accurate and low-power RTC with an integrated temperature-compensated crystal oscillator. The Alarm library provides a simple way to set and manage alarms on the DS3231.

The code then declares a struct ts object named **t** which will be used to store the current time obtained from the RTC. It also declares two arrays of `uint8_t` variables: **wake_TIME**, which will be used to store the time at which the next alarm should trigger, and **current_TIME**, which will be used to store the current time obtained from the RTC.

Next, the code defines the wake-up pin as an interrupt pin that, when pulled low, will wake up the microcontroller from sleep mode. This pin is connected to the DS3231's interrupt output pin, which can be configured to trigger an alarm when a specific time is reached.

The **getCurrentTime()** function is then defined, which retrieves the current time from the RTC and stores it in the **current_TIME** array. This function is called every time the microcontroller wakes up from sleep mode to ensure that it has the latest time from the RTC.

The **check_time()** function is defined next. This function takes three arguments: **wake_intervals**, **wake_offset**, and **wake_tolerance**. The **wake_intervals** argument is an array of four integers that specifies the interval at which the alarm should trigger (i.e. [0, 0, 8, 0] would set the alarm to trigger

every 8 hours). The **wake_offset** argument is an array of two integers that specifies the offset from the interval at which the alarm should trigger (i.e. [0, 30] would set the alarm to trigger at 8:30 instead of 8:00). The **wake_tolerance** argument is an integer that specifies the number of minutes before or after the trigger time that the alarm should still be considered valid. The **check_time()** function calculates the time distance from the previous and next interval based on the current time obtained from the RTC and the specified wake intervals and offsets. It then checks if both the previous and next interval distances are more than the specified tolerance in seconds. If so, it returns **false**, indicating that the alarm should not trigger. Otherwise, it returns **true**, indicating that the alarm should trigger.

The **setNextAlarm()** function is defined next. This function takes two arguments: **wake_intervals** and **wake_offset**. It first sets an array named **flags** to specify which calendar components should be checked against the current time to trigger the alarm. It then calculates the time at which the next alarm should trigger by adding the specified wake intervals and offsets to the current time obtained from the RTC. The function also handles any overflow that may occur when adding the intervals and offsets.

Finally, the **setNextAlarm()** function sets the alarm time (but not yet activated) using the **DS3231_set_a1()** function and turns on the alarm using the **DS3231_set_creg()** function. The alarm will now trigger at the specified time, as long as the conditions specified by the **check_time()** function are met.

4.2.4 Motor motion code

The motor motion code as shown in figure C-4 defines a function called **motor_motion** that takes an optional argument **motor_dir** with a default value of 1. The function is responsible for deploying and retracting a sensor using a DC motor connected to MD10C driver module controlled by an Arduino. The **motor_dir** argument determines the direction of the motor's rotation, where a value of 1 rotates the motor in one direction (to deploy the sensor) and a value of 9 rotates the motor in the opposite direction (to retract the sensor).

The function starts by checking the **motor_dir** argument. If it is equal to 1, the function enters a loop that runs until the analog value read from a

potentiometer connected to an analog input pin on the Arduino is less than a defined minimum threshold **min_adc**. Inside the loop, the function reads the analog value from the potentiometer using the **analogRead** function, which returns a value between 0 and 1023 representing the voltage level at the analog input pin. This value is stored in the variable **adcValue**.

Next, the function sets the **dir** pin to a logic LOW state, which determines the direction of the motor's rotation using the H-bridge driver module. In this case, the LOW state indicates that the motor should rotate in one direction (anticlockwise) to deploy the sensor. The function then sets the **pwm** pin to a PWM value defined by the variable **pwm_value** using the **analogWrite** function. This causes the motor to rotate at a speed determined by the PWM duty cycle, which is proportional to the analog value written to the **pwm** pin.

The function continues to loop until the analog value read from the potentiometer is less than **min_adc**, at which point it prints a message to the serial monitor using the **Serial.println** function and sets the **pwm** pin to 0, which stops the motor.

If the **motor_dir** argument is not equal to 1, the function enters an else block that is similar to the previous block, except that it rotates the motor in the opposite direction by setting the **dir** pin to a logic HIGH state. This causes the motor to rotate in the opposite direction (clockwise) to retract the sensor. The loop runs until the analog value read from the potentiometer is greater than a defined maximum threshold **max_adc**. Once the loop exits, the function prints a message to the serial monitor indicating that the sensor has reached the top and sets the **pwm** pin to 0 to stop the motor.

In summary, the **motor_motion** function deploys or retracts a sensor using a DC motor controlled by an Arduino based on the value of the **motor_dir** argument. It uses a potentiometer to determine the position of the sensor and stops the motor when the sensor reaches a defined position threshold. The function also prints messages to the serial monitor indicating when the sensor has reached its top or bottom position.

4.2.5 Sensor cycle

The sensor cycle code as shown in figure C-5 defines a function called **sensor_cycle** that reads temperature data from a sensor and stores the result in an array pointed to by the argument **temperature_value**. For now, there is only one value which is the **temperature_value**, but it is expected that more sensors will be added to the prototype in the future thus creating a function named **sensor_cycle** that will contain the function for collecting sensor data.

The **sense_temperature** function starts by declaring a buffer **buf** of length 10 to store temperature values, and then calls **sensors.requestTemperatures()** to read data once and remove any residual voltage from previous readings.

Next, it reads 10 temperature values from the DS18B20 sensor using a for loop and stores them in the **buf** array. The **sort_array** function is then called to sort the values in **buf** from smallest to largest, and the **get_avg_value** function is called to calculate the average value of the middle 6 values in the sorted array. The resulting average temperature value is stored in the **avgValue** variable.

4.2.6 Send data to server code

Figure C-6 shows the code that is responsible for sending data from the device to a remote server using the SIM900A GSM/GPRS module. The data being sent is the temperature value and a random dissolved oxygen value generated using a random number generator to simulate the presence of a dissolved oxygen sensor.

The **createGetURL** function generates a string that contains the URL for the Grafana's server endpoint where the data is to be sent. The function takes a float value of the temperature as an argument and generates a random number between 50 and 100 as the dissolved oxygen value. The function creates a string that concatenates the temperature and dissolved oxygen values with the server endpoint URL.

The **createThingSpeakURL** function also generates a string that contains the URL for the ThingSpeak server endpoint where the data is to be sent. This function also takes a float value of the temperature as an argument and generates a random number between 50 and 100 as the dissolved oxygen

value. The function creates a string that concatenates the temperature and dissolved oxygen values with the ThingSpeak server endpoint URL.

The **sendGSM** function sends a string message to the GSM module and waits for a response. The function takes a string message and an optional integer wait time as arguments. The default wait time is 500 milliseconds. The function sends the message to the GSM module and then waits for a response. Once a response is received, the function prints the response to the serial monitor.

The **sendDataToServer** function is responsible for sending the data to the server. The function takes a float value of the temperature as an argument. The function first calls the **createGetURL** and **createThingSpeakURL** functions to generate the URLs for the server endpoints. It then sends a series of AT commands to the GSM module to initialize the HTTP connection with the server. The function then sends the data to the server by sending the URLs generated by the **createGetURL** and **createThingSpeakURL** functions using the **sendGSM** function. The function waits for a response from the server after each URL is sent. Finally, the function closes the HTTP connection and ends the GSM session.

In summary, this code sends temperature data from a device to a remote server using a GSM module. The data is sent to two different server endpoints using two different URLs generated by the **createGetURL** and **createThingSpeakURL** functions. The code uses the **sendGSM** function to send the data and the **sendDataToServer** function to orchestrate the whole process.

4.2.7 Arduino sleep

Figure C-7 shows the code that implements a function **arduino_sleep()** that puts the microcontroller into a low-power sleep mode until it is woken up by a specified interrupt.

The first thing the function does is to call **setNextAlarm()** to set the alarm on a DS3231 real-time clock, which will wake up the microcontroller after a specified number of seconds as explained in subsection 4.2.3.

Next, the code disables the Analog to Digital Converter (ADC) by setting **ADCSRA** to 0. This saves power because the ADC consumes a

significant amount of power. The code then sets the type of sleep mode to **SLEEP_MODE_PWR_DOWN**, which is the lowest-power sleep mode.

Next, the code turns off the Brown Out Detection (BOD) feature temporarily using software. BOD is a feature that protects the microcontroller from low voltage conditions, but it consumes power. By turning it off temporarily, power consumption is reduced.

The code then disables interrupts temporarily, attaches an interrupt to the specified pin **wakePin**, and defines an interrupt service routine **sleepISR()** that will run when the interrupt is triggered.

Before entering sleep mode, the code sends a message to the serial port to indicate that the microcontroller is about to go to sleep. It then allows interrupts and enters sleep mode using **sleep_cpu()**.

While the microcontroller is asleep, it can only be woken up by the specified interrupt on **wakePin** which is triggered by the DS3231 RTC. When the interrupt is triggered, the interrupt service routine **sleepISR()** is called, which first disables sleep mode and then detaches the interrupt that brought the microcontroller out of sleep mode. The main loop is then resumed.

Finally, the code clears the alarm on the DS3231, re-enables the ADC if it was previously running, and returns from the **arduino_sleep()** function and the main loop continues as explained in subsection 4.2.2.

4.3 Circuit design and schematic

Figure 4.9 shows the complete schematic diagram for the prototype which is based on the block diagram in section 3.3 and figure 4.6. The DS3231 RTC uses I2C communication protocol thus it is connected to analogue pin A4 and A5 as they are the SDA and SCL pin respectively that is used in I2C communication. The DS3231 has a SQW output pin that is able to send a square wave on a pre-set time which is used to wake the Arduino Uno up from its power down mode. The SQW output pin is connected to digital pin D2 as it is one of the two interrupt pins of the Arduino Uno. The GSM900A GSM/GPRS module has a TX and RX pin which represents its transmit and receive pin needed for serial communication. The TX and RX pin from GSM900A is connected to digital pin D9 and D8 respectively as it is set inside the AltSoftSerial library. The motor driver MD10C has a PWM, DIR and GND pin that is connected to digital pin D3, digital pin D4 and ground of the Arduino Uno respectively. Digital pin D3 are one of the six digital pins that can generate PWM. The motor will then drive the worm gear motor. The buck converter LM2596 is used to power the Arduino Uno while the current regulator XL4015 is used to power SIM900A, DS3231, DS18B20. The motor driver draws the needed current from the battery and the potentiometer draws current from the Arduino Uno. The solar panel charges the battery through the solar charger controller.

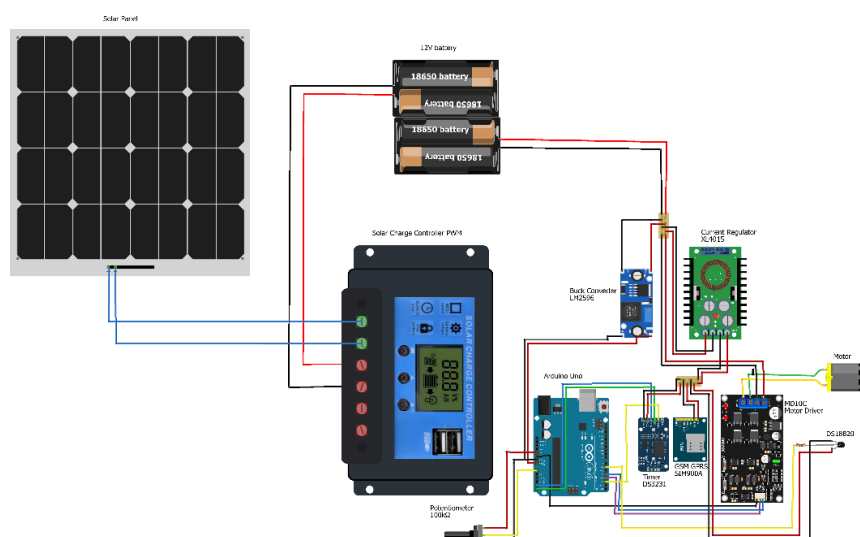


Figure 4.9 Circuit schematic.

4.4 Prototype deployment

As the prototype is quite big in size and is quite difficult to transport, the floating pillar is separated from the prototype when transporting to UTAR Kampar campus. Figure 4.10 shows the prototype being transported to the actual deployment site and that the prototype is mounted on top of the floating pillar onsite.



Figure 4.10 Transporting prototype and mounting.

The prototype is then wrapped with transparent clear stretch film roll to improve its water and weather resistant. The solar panel and film will block off the rain as well as potential water splashes. A metal rod is also hammered near the shoreline of the lake and the prototype is tied to the metal rod with ropes to prevent the prototype from floating too far away. This is because it will be hard to retrieve the prototype if it is too far away from the shoreline. Figure 4.11 shows the prototype being deployed on the lake as well as the metal rod at the shoreline.



Figure 4.11 Prototype deployment.

4.5 Data visualisation

Grafana has successfully receive data sent from the prototype for a total of 5 days from 23rd of March to 27th of March. This is because the server went down after 27th of March and therefore even if the prototype is still sending data, it is not viewable. Figure 4.12 shows the visualisation of the data sent by the prototype. Other graphs can be view in Appendix A.



Figure 4.12 Data visualisation on Grafana

From the graph, it is clear that there is 4 data points within every one hour range. This is because the time interval between data collection are 15 minutes apart. For these five days, Grafana shows that the prototype is sending data to the Grafana server consistently which is one of the crucial objectives of this prototype.

The graph presented in figure 4.12 is coherent and logical, as the temperature slowly increases during the day and decreases during the night. However, in the temperature graph obtained from the floating aquaculture sensor station, there were observed sudden spikes which could be attributed to variations in the sensor data collection location. It is hypothesized that when the floating station is positioned closer to the shoreline where the water is shallower, the temperature in shallow waters is higher. Conversely, when the floating station is positioned further away from the shoreline where the water is deeper, the water temperature is lower. This is because the sensor might reach the floor bed at shallow waters before even being fully deployed causing

it to not reach the same depth of water as when the floating station is further from the shoreline. This phenomenon can also be caused by the thermal stratification of water bodies, where water temperature varies depending on depth and location, resulting in localized temperature variations in the collected data. Further investigation and analysis are required to validate this hypothesis and identify potential solutions to minimize or correct the observed temperature spikes.

4.6 Overall performance of improved prototype

This section will be focusing on the overall performance of the improved prototype with some comparison to the initial prototype. Table 4.1 shows the challenges faced by the initial prototype and their respective solutions presented in the improved prototype. The total time needed for the plastic reel to fully deploy or fully retract the sensor is approximately 10 seconds after timing it. The total time for data collection and data sending is approximately 10 seconds as well. One cycle of data collection is approximately 30 seconds in total.

The improved prototype of the floating aquaculture sensor station has shown promising results. The successful deployment of the sensor station on top of the lake indicates that the design and engineering of the prototype are effective and efficient. The consistent data sending to the server every 15 minutes with minimum time delay suggests that the prototype is reliable and accurate in measuring and transmitting data.

Moreover, the sensor station has proved to be self-sufficient as it has been sending data consistently for 5 days without any interruption. The fact that the sensor station is self-sufficient, fully automated, and weather-proof highlights its resilience and durability. The retracting reel feature is another innovative aspect of the prototype that prevents sensor damage by submerging it in water too long, indicating a well-thought-out design. The use of a potentiometer and worm gear motor in the retracting reel mechanism ensures accurate retraction and prevents any slippage errors.

Table 4.1 Challenges faced and solutions.

Challenges faced by initial prototype	Solution in improved prototype
Design of metal reel causes friction during rotation due to metal contact and reel holder frame tilting inwards	Plastic reel frame is fixed with three metal bars to prevent frame tilting inwards. Plastic contact minimizes friction during rotation
Junction box mounted on metal reel causes imbalance centre of weight	Junction box placed beside plastic reel and uses cable gland and bearing to prevent wire twisting
Slip ring is susceptible to wear and tear, corrosion and contamination. Causing loss of power or data.	
Window motor physical design causes difficulty in designing belt drive assembly due to attached motor coupler	Replace window motor with worm gear motor that has 8mm D-shape shaft
3D printed belt pulley has big size and heavy weight therefore making them high cost and long printing time.	Replaced belt pulleys with ready-made GT2 aluminium pulleys that is more durable, lightweight and smaller in size.
3D printed belt pulley breaks easily in high torque usage	
Potentiometer holder has big size and heavy weight therefore making them high cost and has longer printing time	Improved and optimised potentiometer holder design for lower cost and faster printing time

The circumference of the plastic reel is 40.35 cm which is larger than the circumference of the metal reel which is 27.646cm. Since the plastic reel's center is not a circle, the circumference is measured using a rope tied around the plastic reel and marking the place where they intersect. With a belt pulley ratio of 1:2, the total depth that the sensor can reach is calculated as below.

$$\begin{aligned}
 \text{Sensor wire length} &= \text{Number of revolution} \times C_{reel} & (4.1) \\
 &= 20 \times 40.35 \text{ cm} \\
 &= 807 \text{ cm} \\
 &= 8.07 \text{ m}
 \end{aligned}$$

Assuming that the sensor is retracted 15cm above the water level, the sensor can reach the depth of 7.92 meters.

Since the driven pulley that is used in this prototype is a ready-made GT2 aluminum pulley, it can be easily replaced with a 60 tooth, GT2 aluminum pulley and thus increasing the ratio to 1:3. This means that the total depth that the sensor can reach is calculated as below:

$$\begin{aligned}
 \text{Sensor wire length} &= \text{Number of revolution} \times C_{reel} & (4.2) \\
 &= 30 \times 40.35 \text{ cm} \\
 &= 1210.5 \text{ cm} \\
 &= 12.105 \text{ m}
 \end{aligned}$$

Assuming that the sensor is retracted 15cm above the water level, the sensor can reach the depth of 11.955 meters.

In summary, the improved prototype of the floating aquaculture sensor station has demonstrated remarkable overall performance in terms of functionality, reliability, and durability. It is capable of measuring and transmitting accurate data while remaining resilient to harsh weather conditions, making it an excellent tool for monitoring aquaculture activities in lakes.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The enhanced floating aquaculture sensor station has effectively fulfilled its objectives. It has enabled an IoT-enabled water quality monitoring system to gather real-time water quality parameters as it has been successfully deployed on top of the lake, allowing it to collect data from the surrounding area. The station can transmit data to a remote server with a pre-set data collection interval and minimal time delay, enabling real-time visualization of collected data. Moreover, the retracting reel sensor is able to deploy and retract the sensor. The sensor reel is attached to a potentiometer and uses a worm gear motor to prevent slippage error and sensor damage caused by prolonged water immersion. The potentiometer also allows it to know the sensor's absolute position in cases of power shortage. The floating aquaculture sensor station provides a reliable and efficient solution for collecting valuable data on aquatic environments, allowing researchers to better understand and manage these ecosystems.

5.2 Recommendations for future work

The floating aquaculture sensor station prototype can be improved in the future by adding several features. One of the features is to measure the power consumption of the overall system, which can help optimize the energy usage and improve the efficiency of the device. Another feature is to add a message feature that can adjust data collection intervals based on the SMS received, which will make it more flexible and adaptable to different conditions. This feature can be used to notify related parties of potential hazards or sending failure reports to related parties. This is crucial in cases where the battery capacity starts to drop low, potentiometer reading not changing even if motor is activated, not receiving sensible data from sensors, unable to send data to server and many others.

Moreover, adding an SD card to store data as a backup in case of failing to send data to the server is also a crucial improvement. This feature can ensure that the data collected by the device will not be lost even if the connection to the server is interrupted or unstable. The data stored in the SD card can be retrieved later and transmitted to the server when the connection is restored. This feature can improve the reliability and data integrity of the device, which is essential for accurate data analysis and decision-making in aquaculture management. Overall, these improvements can enhance the performance and functionality of the floating aquaculture sensor station prototype and make it more efficient and reliable for aquaculture farmers.

Finally, to address the temperature spikes that were observed from the graph, GPS module can be included to indicate whether the floating station is close to the shoreline or not at the time of data collection. The floating station could also be anchored at the middle of the lake. Even though the floating station will still move within the anchor swinging radius, but it will ensure that the sensor will reach the same depth of water level at the moment of data collection. This method will require a boat to move the prototype to the center of the lake as well as an anchor.

REFERENCES

- Bosma, R.H. and Verdegem, M.C.J., 2011. Sustainable aquaculture in ponds: Principles, practices and limits. *Livestock Science*, 139(1–2), pp.58–68.
- Covington, A.K., Bates, R.G. and Durst, R.A., 1985. Definition of pH scales, standard reference values, measurement of pH and related terminology (Recommendations 1984). *Pure and Applied Chemistry*, 57(3), pp.531–542.
- Fourie, C.M. et al., 2017. A solar-powered fish pond management system for fish farming conservation. *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. June 2017 IEEE, pp. 2021–2026.
- Garcia, M., Sendra, S., Lloret, G. and Lloret, J., 2011. Monitoring and control sensor system for fish feeding in marine fish farms. *IET Communications*, 5(12), pp.1682–1690.
- Hairol, K.N., Adnan, R., Samad, A.M. and Ahmat Ruslan, F., 2018. Aquaculture Monitoring System using Arduino Mega for Automated Fish Pond System Application. *2018 IEEE Conference on Systems, Process and Control (ICSPC)*. December 2018 IEEE, pp. 218–223.
- Kocakulak, M. and Butun, I., 2017. An overview of Wireless Sensor Networks towards internet of things. *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. January 2017 IEEE, pp. 1–6.
- Li, D. and Liu, S., 2019. Water Quality Monitoring in Aquaculture. In: *Water Quality Monitoring and Management*. Elsevier, pp. 303–328.
- Majid, M. et al., 2022. Applications of Wireless Sensor Networks and Internet of Things Frameworks in the Industry Revolution 4.0: A Systematic Literature Review. *Sensors*, 22(6), p.2087.

- Manrique, J.A., Rueda-Rueda, J.S. and Portocarrero, J.M.T., 2016. Contrasting Internet of Things and Wireless Sensor Network from a Conceptual Overview. *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. December 2016 IEEE, pp. 252–257.
- Niswar, M. et al., 2018. IoT-based Water Quality Monitoring System for Soft-Shell Crab Farming. *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*. November 2018 IEEE, pp. 6–9.
- Raposo, D. et al., 2018. Industrial IoT Monitoring: Technologies and Architecture Proposal. *Sensors*, 18(10), p.3568.
- Rosaline, N. and Sathyalakshimi, S., 2019. IoT Based Aquaculture Monitoring and Control System. *Journal of Physics: Conference Series*, 1362, p.012071.
- Shi, B. et al., 2018. A wireless sensor network-based monitoring system for freshwater fishpond aquaculture. *Biosystems Engineering*, 172, pp.57–66.
- Simbeye, D.S., Zhao, J. and Yang, S., 2014. Design and deployment of wireless sensor networks for aquaculture monitoring and control based on virtual instruments. *Computers and Electronics in Agriculture*, 102, pp.31–42.
- Sousa e Silva, M., Cruz, N.A. and Lima, F.P., 2016. Remote Supervision System for Aquaculture Platforms. *OCEANS 2016 MTS/IEEE Monterey*. September 2016 IEEE, pp. 1–6.
- Strobl, R.O. and Robillard, P.D., 2008. Network design for water quality monitoring of surface freshwaters: A review. *Journal of Environmental Management*, 87(4), pp.639–648.

Summerfelt, R.C., 2000. Water quality considerations for aquaculture. *Department of Animal Ecology*, pp.2–7.

Zhao, J., 2014. Research on Wireless Sensor Network in Aquaculture. *Applied Mechanics and Materials*, 686, pp.397–401.

APPENDICES

Appendix A: Graphs



Figure A- 2: Grafana graph for temperature from 20th to 26th March.

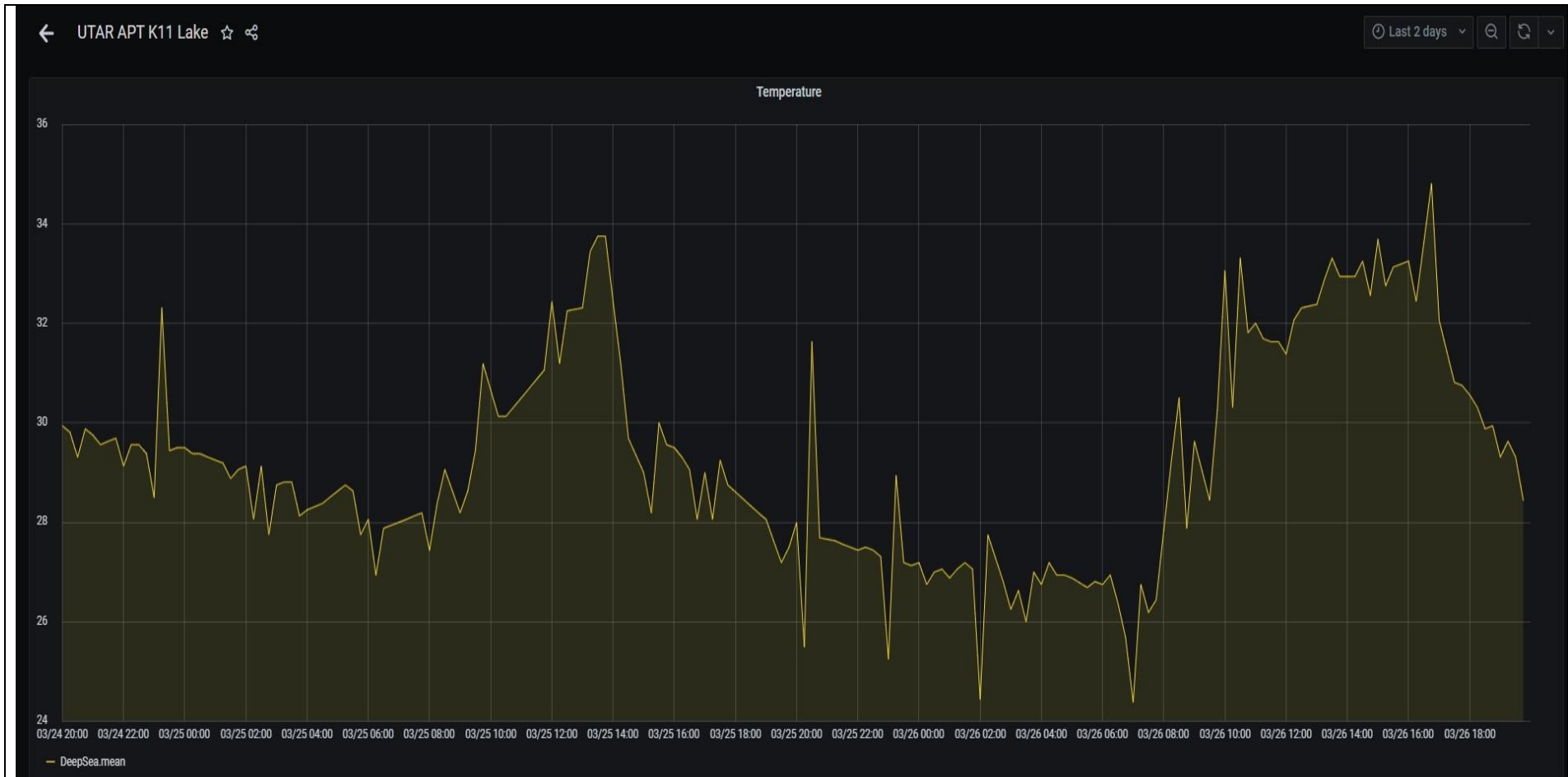
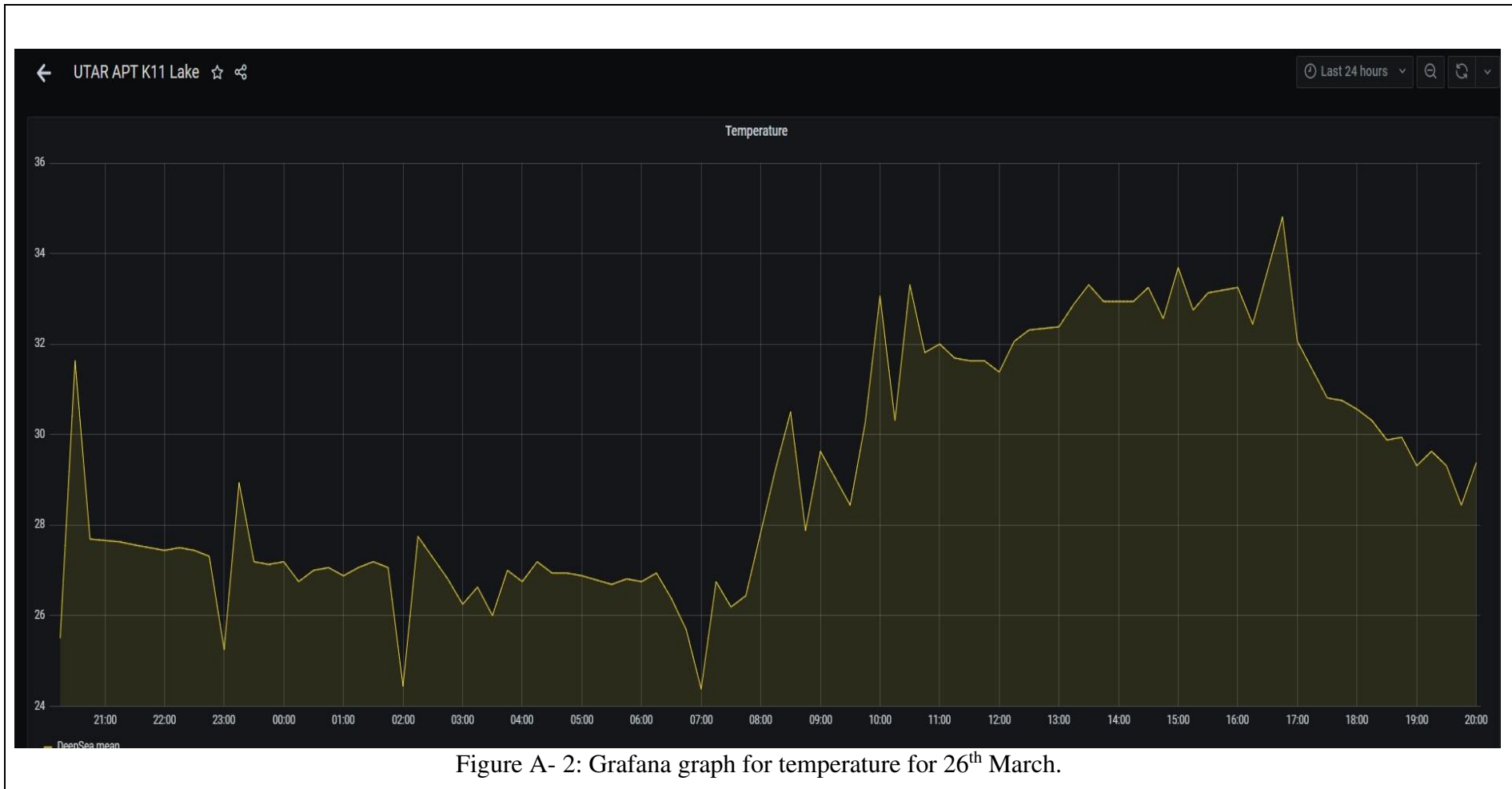


Figure A- 2: Grafana graph for temperature from 24th to 26th March.



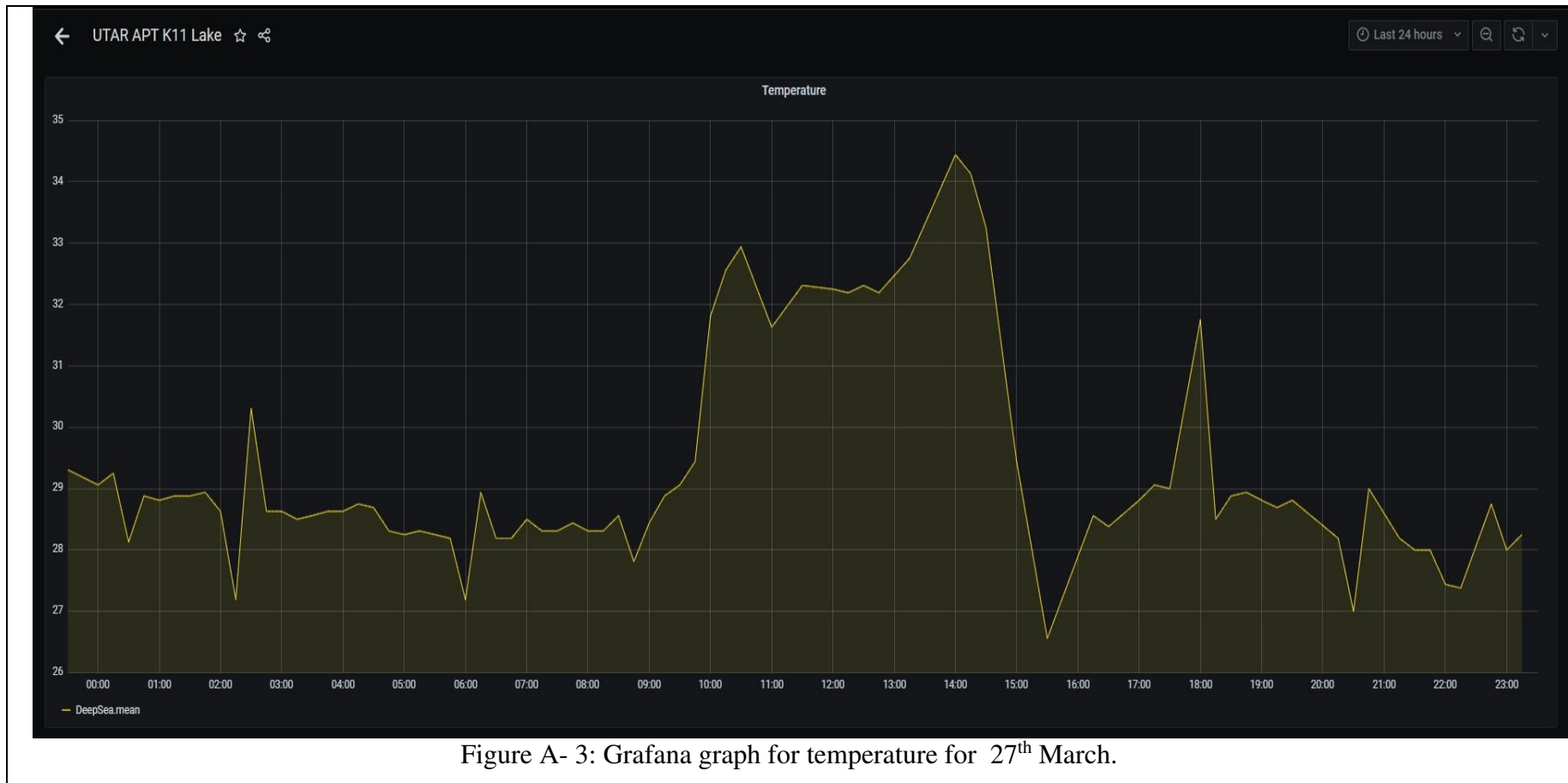


Figure A- 3: Grafana graph for temperature for 27th March.

Appendix B: Datasheet

1	Motor Type/Model	Rated Voltage	RPM/Speed 10%±	Rated Torque	Stall Torque	Rated Current	Stall Current
2	JGY-370	6V	10 RPM±	16 kg.cm±	50 kg.cm±	0.45A±	1.8A±
3	JGY-370	6V	40 RPM±	4.2 kg.cm±	14 kg.cm±	0.45A±	1.8A±
4	JGY-370	6V	150 RPM±	1 kg.cm±	3.6 kg.cm±	0.45A±	1.8A±
5							
6	A58-555	12V	27 RPM±	70 kg.cm±	70 kg.cm±	1.6A±	5A±
7	A58-555	12V	80 RPM±	30 kg.cm±	70 kg.cm±	1.6A±	5A±
8	A58-555	12V	160 RPM±	15 kg.cm±	50 kg.cm±	1.6A±	5A±
9	A58-555	12V	260 RPM±	9 kg.cm±	30 kg.cm±	1.6A±	5A±
10	A58-555	12V	470 RPM±	5 kg.cm±	16 kg.cm±	1.6A±	5A±
11							
12	5840-31ZY	12V	27 RPM±	40 kg.cm±	40 kg.cm±	1.6A±	6.5A±
13	5840-31ZY	12V	80 RPM±	16 kg.cm±	16 kg.cm±	1.6A±	6.5A±
14	5840-31ZY	12V	160 RPM±	8 kg.cm±	25 kg.cm±	1.6A±	6.5A±
15	5840-31ZY	12V	260 RPM±	5 kg.cm±	20 kg.cm±	1.6A±	6.5A±
16	5840-31ZY	12V	470 RPM±	2.6 kg.cm±	13 kg.cm±	1.6A±	6.5A±

Figure B- 1: Motor types and their specifications.

Electrical Characteristics

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to $V_{CC}+0.5V$
Voltage on $\overline{\text{RESET}}$ with respect to Ground.....	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0mA
DC Current V_{CC} and GND Pins.....	200.0mA and 400.0mA TQFP/MLF

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Figure B- 2: Arduino Uno electrical characteristics.

Power-down Mode When the SM2..0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the External Oscillator is stopped, while the External interrupts, the Two-wire Serial Interface address watch, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, a Two-wire Serial Interface address match interrupt, an External level interrupt on INT0 or INT1, or an External interrupt on INT2 can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to "External Interrupts" on page 66 for details.

When waking up from Power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL fuses that define the reset time-out period, as described in "Clock Sources" on page 25.

Figure B- 3: Power down sleep mode.

Table 14. Active Clock Domains and Wake Up Sources in the Different Sleep Modes

Sleep Mode	Active Clock domains					Oscillators		Wake-up Sources					
	clk _{CPU}	clk _{FLASH}	clk _{IO}	clk _{ADC}	clk _{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT2 INT1 INT0	TWI Address Match	Timer 2	SPM / EEPROM Ready	ADC	Other I/O
Idle			X	X	X	X	X ⁽²⁾	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X ⁽²⁾	X ⁽³⁾	X	X	X	X	
Power-down								X ⁽³⁾	X				
Power-save					X ⁽²⁾		X ⁽²⁾	X ⁽³⁾	X	X ⁽²⁾			
Standby ⁽¹⁾						X		X ⁽³⁾	X				
Extended Standby ⁽¹⁾					X ⁽²⁾	X	X ⁽²⁾	X ⁽³⁾	X	X ⁽²⁾			

Notes: 1. External Crystal or resonator selected as clock source.
2. If AS2 bit in ASSR is set.
3. Only INT2 or level interrupt INT1 and INT0.

Figure B- 4: Arduino Uno sleep modes.

Specifications:

- Left motor
- Voltage Rating: 12VDC
- Rated Speed: 60 RPM
- Rated Torque: 2.9N.m (30kgf.cm)
- Rated Current: less than 15A at 12V
- Stall Torque (Locked): 9.8N.m (100kg.cm)
- Stall Current (Locked): less than 28A at 12V
- This is motor meant for driving car's window as the name says
- It is being tested to drive car's window:
 - Up and Down for 8 seconds, survive 10,000 times
 - Not meant for continuous driving without stopping

Figure B- 5: Window motor specifications.



适用工作电流24A以内 持续工作电流 \leq 24A 瞬间启动电流 \leq 50A的用电器

型号:	12V 25000mAh锂电池	电芯:	30节18650芯
电压:	12V	容量:	25000mAh
接头:	DC5.5-2.1公头 母头各一条	尺寸:	195*55*68mm
可持续电流:	24A	重量:	约1302g
过流保护值:	50A		

Figure B- 6: Lithium battery pack.

Table B- 1: Battery specifications.

Model	12V 25000mAH lithium battery pack
Operating voltage	12V
Connector	DC 5.5-2.1 DC jack male and female
Max uninterrupted current	24A
Overcurrent protection threshold	50A
Number of 18650	30
Capacity	25000mAH
Dimensions	195 x 55 x 68 mm
Weight	Approximately 1302g

Appendix C: Arduino program code

```

1  #include <AltSoftSerial.h>
2  #include <OneWire.h>
3  #include <DallasTemperature.h>
4  #include <String.h>
5  #include <Alarm.h>
6
7  #define pwm_value 200 // make sure in the range of 0 to 255 for motor
8  unsigned long long max_adc = 440, min_adc = 290;
9
10 // Set wake up intervals using Second, Minute, Hour and Day
11 // if alarm wake for every hour, set wake_intervals[4] = {0, 0, 1, 0}
12 // if alarm wake for every 15 minutes, set wake_intervals[4] = {0, 15, 0, 0}
13 uint8_t wake_intervals[4] = {0, 15, 0, 0};
14
15 // Set wake up offset using Minute and Second
16 // To wake every 15 minutes start from 12:05, set wake_intervals[4] = {0, 0, 15, 0}, wake_offset = {0, 5}
17 // will wake 12:20, 12:35, 12:50, 1:05, 1:20
18 uint8_t wake_offset[2] = {0, 0};
19
20 // Set time tolerance which if the time that Arduino wakes up doesn't match the time, it will not initiate a new cycle
21 // Using minutes
22 uint8_t tolerance_minutes = 3;
23
24 #define potentiometer_pin A0
25 #define wakePin 2 // when low, makes 328P wake up, must be an interrupt pin (2 or 3 on ATMEGA328P)
26 #define pwm 3
27 #define dir 4
28 #define ONE_WIRE_BUS 6 // define the pin for Temperature sensor (DS18B20)
29
30 #define tempUpperLimit 50 // Change the temperature limit
31 #define tempLowerLimit 20
32
33 unsigned long adcValue = 0;
34
35 // Setup a softserial instance
36 AltSoftSerial GSM;
37
38 // Setup a oneWire instance to communicate with any OneWire devices
39 OneWire oneWire(ONE_WIRE_BUS);
40 // Pass our oneWire reference to Dallas Temperature.
41 DallasTemperature sensors(&oneWire);
42
43 // Standard setup( ) function
44 void setup()
45 {
46   Serial.begin(9600);
47   GSM.begin(9600); /* Define baud rate for software serial communication */
48   Wire.begin();
49
50   // Keep pins high until we ground them
51   pinMode(wakePin, INPUT_PULLUP);
52   pinMode(dir, OUTPUT);
53
54   // Clear the current alarm (puts DS3231 INT high)
55   DS3231_init(DS3231_CONTROL_INTCN);
56   DS3231_clear_alf();
57
58   getcurrentTime();
59
60   Serial.println("Setup completed.");
61 }

```

Figure C- 1: Program initialisation.

```
1 void loop()
2 {
3   float temperature_value;
4
5   if (check_time(wake_intervals, wake_offset, tolerance_minutes)){
6     Serial.println("Within time tolerance");
7     // Activate motor go down
8     motor_motion(1);
9
10    // sensor cycle code
11    sensor_cycle(&temperature_value);
12    // randomSeed(analogRead(A2));
13    // temperature_value = random(25,40);
14    sendDataToServer(temperature_value);
15  }
16
17  // Activate motor go up
18  motor_motion(0);
19
20  arduino_sleep();
21 }
```

Figure C- 2: Main loop.

```

1 #include <Wire.h>
2 #include <Alarm.h>
3 #include <ds3231.h>
4
5 struct ts t;
6
7 // DS3231 alarm time
8 uint8_t wake_TIME[4] = {0, 0, 0, 0};
9
10 uint8_t current_TIME[4] = {0, 0, 0, 0};
11
12 #define wakePin 2 // when low, makes 328P wake up, must be an interrupt pin (2 or 3 on ATMEGA328P)
13
14 void getCurrentTime(){
15     DS3231_get(&t);
16     current_TIME[0] = t.sec;
17     current_TIME[1] = t.min;
18     current_TIME[2] = t.hour;
19 }
20
21 boolean check_time(uint8_t wake_intervals[4], uint8_t wake_offset[2], uint8_t wake_tolerance)
22 {
23     // put your main code here, to run repeatedly;
24     // since tolerance means +-, so this is calculate the time distance from the previous and next interval
25
26     long tolerance_seconds = wake_tolerance * 60;
27     long previous_interval = 0;
28     long next_interval = 0;
29
30     for (int i = 2; i >= 0; i--)
31     {
32         int time_modulus = 0;
33         int time_diff = 0;
34
35         time_modulus = current_TIME[i] % wake_intervals[i];
36         time_diff = wake_intervals[i] - time_modulus;
37
38         previous_interval = previous_interval * 60 + time_modulus;
39         next_interval = next_interval * 60 + time_diff;
40     }
41
42     // if both previous and next interval distance is more than tolerance seconds means do not deploy
43     if (previous_interval > tolerance_seconds && next_interval > tolerance_seconds)
44     {
45         return false;
46     }
47     else
48     {
49         return true;
50     }
51 }
52
53 void setNextAlarm(uint8_t wake_intervals[4], uint8_t wake_offset[2])
54 {
55     // flags define what calendar component to be checked against the current time in order
56     // to trigger the alarm - see datasheet
57     // A1M1 (seconds) (0 to enable, 1 to disable)
58     // A1M2 (minutes) (0 to enable, 1 to disable)
59     // A1M3 (hour) (0 to enable, 1 to disable)
60     // A1M4 (day) (0 to enable, 1 to disable)
61     // DY/DT (dayofweek == 1/dayofmonth == 0)
62     uint8_t flags[5] = {0, 0, 0, 1, 1};
63
64     for (int i = 0; i < 4; i++)
65     {
66         wake_TIME[i] = current_TIME[i];
67         wake_TIME[i] = wake_TIME[i] + wake_intervals[i];
68     }
69
70     for (int i = 0; i < 2; i++)
71     {
72
73         // calculate second and hour offset
74         if (wake_intervals[i] != 0)
75         {
76             wake_TIME[i] = wake_TIME[i] - wake_TIME[i] % wake_intervals[i];
77             wake_TIME[i] = wake_TIME[i] + wake_offset[i];
78         }
79         else
80         {
81             wake_TIME[i] = wake_offset[i];
82         }
83
84         // overflow increment for seconds and minutes
85         if (wake_TIME[i] > 59)
86         {
87             wake_TIME[i + 1] = wake_TIME[i + 1] + 1;
88             wake_TIME[i] = wake_TIME[i] - 60;
89         }
90     }
91
92     // overflow increment for hours
93     if (wake_TIME[2] > 23)
94     {
95         wake_TIME[2] = wake_TIME[2] - 24;
96     }
97
98     for (int i = 0; i < 4; i++)
99     {
100         current_TIME[i] = wake_TIME[i];
101     }
102
103     Serial.println("Next wake time");
104     Serial.print(wake_TIME[2], DEC);
105     Serial.print(':');
106     Serial.print(wake_TIME[1], DEC);
107     Serial.print(':');
108     Serial.print(wake_TIME[0], DEC);
109     Serial.println();
110
111     delay(5000);
112
113     // Set the alarm time (but not yet activated)
114     DS3231_set_a1(wake_TIME[0], wake_TIME[1], wake_TIME[2], 0, flags);
115
116     // Turn the alarm on
117     DS3231_set_creg(DS3231_CONTROL_INTCN | DS3231_CONTROL_A1IE);
118 }

```

Figure C- 3: Alarm.cpp.


```
1 void motor_motion(uint8_t motor_dir = 1)
2 {
3   // deploy sensor when motor_dir = 1 and retract sensor when motor_dir = 9
4
5   if (motor_dir == 1)
6   {
7     while (adcValue >= min_adc)
8     {
9       analogRead(potentiometer_pin);
10
11       // read the input on analog pin 0:
12       adcValue = analogRead(potentiometer_pin);
13
14       digitalWrite(dir, LOW); // rotate anticlockwise to drop the sensor
15       analogWrite(pwm, pwm_value); // Send PWM to output pin
16     }
17
18     Serial.println("Reached the bottom");
19     analogWrite(pwm, 0);
20   }
21   else
22   {
23     while (adcValue <= max_adc)
24     {
25       analogRead(potentiometer_pin);
26
27       // read the input on analog pin 0:
28       adcValue = analogRead(potentiometer_pin);
29
30       digitalWrite(dir, HIGH); // rotate anticlockwise to drop the sensor
31       analogWrite(pwm, pwm_value); // Send PWM to output pin
32     }
33
34     Serial.println("Reached the top!");
35     analogWrite(pwm, 0);
36   }
37 }
```

Figure C- 4: Motor motion code.

```

1 void sort_array(float *array_value, size_t array_size)
2 {
3
4     uint8_t temp;
5
6     for (uint8_t i = 0; i < array_size - 1; i++) // sort the analog from small to large
7     {
8         for (uint8_t j = i + 1; j < array_size; j++)
9         {
10            if (array_value[i] > array_value[j])
11            {
12                temp = array_value[i];
13                array_value[i] = array_value[j];
14                array_value[j] = temp;
15            }
16        }
17    }
18
19    return;
20 }
21
22 float get_avg_value(float *array_value, size_t array_size)
23 {
24     float avgValue = 0;
25     for (int i = 2; i < 8; i++) // take the average value of 6 center sample
26         avgValue += array_value[i];
27     avgValue = (float)avgValue / 6; // convert the analog into millivolt
28
29     return avgValue;
30 }
31
32 float sense_temperature()
33 {
34
35     float avgValue; // Store the average value of the sensor feedback
36     const int buf_length = 10;
37     float buf[buf_length];
38
39     sensors.requestTemperatures(); // read data once to remove residual voltage from previous analog readings
40
41     for (int i = 0; i < 10; i++) // Get 10 sample value from the sensor for smooth the value
42     {
43         buf[i] = sensors.getTempCByIndex(0);
44         delay(100);
45     }
46     sort_array(buf, sizeof(buf) / sizeof(buf[0]));
47     avgValue = get_avg_value(buf, sizeof(buf) / sizeof(buf[0]));
48
49     Serial.print("temperature value:");
50     Serial.print(avgValue);
51     Serial.println(" ");
52
53     return avgValue;
54 }
55
56 void sensor_cycle(float *temperature_value)
57 {
58     *temperature_value = sense_temperature();
59
60     // Save to sd card code below:
61 }

```

Figure C- 5: Sensor cycle.

```

1  String createGetURL(float temperature_value)
2  {
3      randomSeed(analogRead(0));
4      // int dioxy = random(50, 100);
5      int dioxy = 0;
6
7      String stringOne = "AT+HTTTPARA=\"URL\", \"http://45.127.4.18:60288/endpoint2?temp=";
8      String stringTwo = "&dioxy=";
9      String stringThree = "\"";
10     String stringFull = stringOne + temperature_value + stringTwo + dioxy + stringThree;
11
12     return stringFull;
13 }
14
15 String createThingSpeakURL(float temperature_value)
16 {
17     randomSeed(analogRead(0));
18     int dioxy = random(50, 100);
19     // int temp = random(20, 35);
20
21     String stringOne = "AT+HTTTPARA=\"URL\", \"http://api.thingspeak.com/update?api_key=LMDKI99A62C3F77X&field1=";
22     String stringTwo = "&field2=";
23     String stringThree = "\"";
24     String stringFull = stringOne + temperature_value + stringTwo + dioxy + stringThree;
25
26     return stringFull;
27 }
28
29 void sendGSM(const char *msg, int waitMs = 500)
30 {
31     GSM.println(msg);
32     while (GSM.available())
33     {
34         Serial.write(char(GSM.read())); /* Print character received on to the serial monitor */
35         // Serial.println(GSM.read());
36     }
37     delay(waitMs);
38 }
39
40 void sendDataToServer(float temperature_value)
41 {
42     String getURL = createGetURL(temperature_value);
43     String ThingSpeak = createThingSpeakURL(temperature_value);
44
45     sendGSM("AT+SAPBR=3,1,\"APN\", \"yoodo\"");
46     sendGSM("AT+SAPBR=1,1", 3000);
47     sendGSM("AT+SAPBR=2,1", 3000);
48     sendGSM("AT+HTTPIINIT");
49     sendGSM("AT+HTTTPARA=\"CID\",1");
50
51     GSM.println(getURL);
52     while (GSM.available())
53     {
54         GSM.read();
55     }
56     delay(2000);
57
58     sendGSM("AT+HTTTPACTION=0");
59     GSM.read();
60     delay(2000);
61
62     GSM.println(ThingSpeak);
63     while (GSM.available())
64     {
65         GSM.read();
66     }
67     delay(2000);
68
69     sendGSM("AT+HTTTPACTION=0");
70     GSM.read();
71     delay(2000);
72
73     sendGSM("AT+HTTTPACTION=0");
74     GSM.read();
75
76     sendGSM("AT+HTTPTERM");
77     GSM.read();
78
79     sendGSM("AT+SAPBR=0,1");
80     GSM.read();
81 }

```

Figure C- 6: Send data to server.

```

1 // When wakePin is brought LOW this interrupt is triggered FIRST (even in PWR_DOWN sleep)
2 void sleepISR()
3 {
4     // Prevent sleep mode, so we don't enter it again, except deliberately, by code
5     sleep_disable();
6
7     // Detach the interrupt that brought us out of sleep
8     detachInterrupt(digitalPinToInterrupt(wakePin));
9
10    // Now we continue running the main Loop() just after we went to sleep
11 }
12
13 void arduino_sleep()
14 {
15     // Set the DS3231 alarm to wake up in X seconds
16     setNextAlarm(wake_intervals, wake_offset);
17
18     // Disable the ADC (Analog to digital converter, pins A0 [14] to A5 [19])
19     static byte prevADCSRA = ADCSRA;
20     ADCSRA = 0;
21
22     /* Set the type of sleep mode we want. Can be one of (in order of power saving):
23
24     SLEEP_MODE_IDLE (Timer 0 will wake up every millisecond to keep millis running)
25     SLEEP_MODE_ADC
26     SLEEP_MODE_PWR_SAVE (TIMER 2 keeps running)
27     SLEEP_MODE_EXT_STANDBY
28     SLEEP_MODE_STANDBY (Oscillator keeps running, makes for faster wake-up)
29     SLEEP_MODE_PWR_DOWN (Deep sleep)
30     */
31     set_sleep_mode(SLEEP_MODE_PWR_DOWN);
32     sleep_enable();
33
34     // Turn of Brown Out Detection (low voltage)
35     // Thanks to Nick Gammon for how to do this (temporarily) in software rather than
36     // permanently using an avrdude command line.
37     //
38     // Note: Microchip state: BODS and BODSE only available for picoPower devices ATmega48PA/88PA/168PA/328P
39     //
40     // BODS must be set to one and BODSE must be set to zero within four clock cycles. This sets
41     // the MCU Control Register (MCUCR)
42     MCUCR = bit(BODS) | bit(BODSE);
43
44     // The BODS bit is automatically cleared after three clock cycles so we better get on with it
45     MCUCR = bit(BODS);
46
47     // Ensure we can wake up again by first disabling interrupts (temporarily) so
48     // the wakeISR does not run before we are asleep and then prevent interrupts,
49     // and then defining the ISR (Interrupt Service Routine) to run when poked awake
50     noInterrupts();
51     attachInterrupt(digitalPinToInterrupt(wakePin), sleepISR, LOW);
52
53     // Send a message just to show we are about to sleep
54     Serial.println("Good night!");
55     Serial.flush();
56
57     // Allow interrupts now
58     interrupts();
59
60     // And enter sleep mode as set above
61     sleep_cpu();
62
63     // -----
64     // Controller is now asleep until woken up by an interrupt
65     // -----
66
67     // Wakes up at this point when wakePin is brought LOW - interrupt routine is run first
68     Serial.println("I'm awake!");
69
70     // Clear existing alarm so int pin goes high again
71     DS3231_clear_alf();
72
73     // Re-enable ADC if it was previously running
74     ADCSRA = prevADCSRA;
75
76     return;
77 }

```

Figure C- 7: Arduino sleep.