DESIGN AND DEVELOPMENT OF A MICRO UNDERWATER REMOTELY OPERATED VEHICLE (ROV) FOR MONITORING APPLICATION

CHAN YEOW CHUAN

BACHELOR OF MECHATRONICS ENGINEERING WITH HONOURS
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

DESIGN AND DEVELOPMENT OF A MICRO UNDERWATER REMOTELY OPERATED VEHICLE (ROV) FOR MONITORING APPLICATION

CHAN YEOW CHUAN

A report submitted
in partial fulfilment of the requirements for the degree of
Bachelor of Mechatronics Engineering with Honours

Faculty of Electrical Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

DECLARATION

I declare that this thesis entitled "DESIGN AND DEVELOPMENT OF A MICRO UNDERWATER REMOTELY OPERATED VEHICLE (ROV) FOR MONITORING APPLICATION is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in the candidature of any other degree.

Signature	:	
Name	:	CHAN YEOW CHUAN
Date	:	24/6/2024

APPROVAL

I hereby declare that I have checked this report entitled " DESIGN AND DEVELOPMENT OF A MICRO UNDERWATER REMOTELY OPERATED VEHICLE (ROV) FOR MONITORING APPLICATION ", and in my opinion, this thesis fulfils the partial requirement to be awarded the degree of Bachelor of Mechatronics Engineering with Honours

Signature : .....

Supervisor Name : IR. FAUZAL NAIM BIN ZOHEDI

Date : 24/6/2024

.....

DEDICATIONS

To my beloved mother and father

ACKNOWLEDGEMENTS

In preparing this report, I was in contact with many people, researchers, academicians and practitioners. They have contributed towards my understanding and thought. In particular, I wish to express my sincere appreciation to my main project supervisor, IR. Fauzal Naim Bin Zohedi, for encouragement, guidance critics and friendship. I am also very thankful to my co-supervisors I am also grateful to my project panel for their guidance, support, and motivation. Without their ongoing support and interest, this project would not have turned out as well as it accomplished.

Secondly, I would like to give special thanks to Universiti Teknikal Malaysia Melaka (UTEM) especially to Fakulti Teknologi Kejuruteraan Elektrik (FTKE) because give me chance to apply the engineering knowledge and improve skills of electrical filed in this project. Besides that, special thanks to FYP committee in providing program and preparations to complete the FYP and report.

Lastly, my gratitude goes to the family especially my father and mother who is supporting and gave me inspiration to finish my final year project. My sincere appreciation also extends to all my coursemates who have provided assistance on various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. I hope this project helps me to improve my engineering knowledge and practice as an engineer.

ABSTRACT

Underwater Remote Operated Vehicle (ROV) is a tethered marine robot that are widely employed for scientific and commercial applications. Several industries are working on underwater robots to increase the productivity, monitoring and surveillance especially in the petroleum and gas industries for periodic checking and surveillance of underwater pipelines and chains. These operations are often performed by human divers; however, the underwater environment poses hazards and pressure-related limits, making them costly and risky. As a result, ROVs have been designed to replace divers themselves. It is a tethered underwater robot that the operator controls manually using a PS2 controller. The advantage of employing this ROV lies in its capability to navigate underwater for specific duties, such as monitoring. This project is to design and develop a micro underwater ROV for monitoring applications. The ROV are designed to withstand pressure underwater by selection of suitable material for its frame and other components will be equipped including Bar02 pressure/depth sensor, MPU6050 IMU sensor and waterproof endoscope camera. The design of the ROV prototype was centered around three primary objectives: maneuverability, performance, and suitability for future industrial applications, including the capacity to carry cameras, tools and sensors. Standard testing procedures are employed to assess the ROV's performance in terms of buoyancy and control efficiency tests for the propulsion system in real environment which includes laboratory pool. The developed ROV prototype shows promising performance with achieved 90% negative buoyancy is crucial for the ROV to perform effective submerge and raise operations and also with stable velocity and acceleration in forward, backward, and submerging. The steering tests highlighted that the ROV is more flexible and faster in maneuvering concerning turning performance as the horizontal thrusters' configurations are positioned at 45° at the back of the ROV. This report illustrates the development of the ROV through mechanical, electronic, and software design. The outcomes of this project are anticipated to bring substantial advantages to industries associated with underwater applications.

ABSTRAK

Underwater Remote Operated Vehicle (ROV) ialah robot maritim yang digunakan secara meluas untuk aplikasi saintifik dan komersial. Beberapa industri sedang bekerja pada robot bawah air untuk meningkatkan produktiviti, pemantauan dan pengawasan terutamanya dalam industri minyak dan gas untuk pemeriksaan periodik dan pemeriksaan paip bawah air dan rantaian. Operasi-operasi ini sering dilakukan oleh penyelam manusia; Walau bagaimanapun, persekitaran bawah air menimbulkan bahaya dan had yang berkaitan dengan tekanan, menjadikannya mahal dan berisiko. Akibatnya, ROV telah direka untuk menggantikan penyelam itu sendiri. Ia adalah robot bawah air yang terhubung yang pengendali mengawal secara manual menggunakan pengawal PS2. Kelebihan menggunakan ROV ini terletak dalam keupayaan untuk mengemudi di bawah air untuk tugas-tugas tertentu, seperti pemantauan. Projek ini ialah untuk merancang dan membangunkan ROV bawah air mikro untuk pemantauan aplikasi. ROV direka untuk menahan tekanan di bawah air dengan memilih bahan yang sesuai untuk bingkai dan komponen lain akan dilengkapi termasuk sensor tekanan / kedalaman Bar02, sensor MPU6050 IMU dan kamera endoskop tahan air. Reka bentuk ROV difokuskan kepada tiga matlamat utama: kelenturan, prestasi, dan kelayakan untuk aplikasi industri masa depan, termasuk keupayaan untuk membawa kamera, alat dan sensor. Prosedur ujian standard digunakan untuk menilai prestasi ROV dalam hal ujian keluaran dan kawalan kecekapan untuk sistem pendorong dalam persekitaran sebenar yang termasuk kolam makmal. Prototype ROV yang dikembangkan menunjukkan prestasi yang menjanjikan dengan 90% negatif buoyancy yang dicapai adalah penting bagi ROV untuk melakukan operasi menyelam dan menaikkan yang berkesan dan juga dengan kelajuan dan kelajuan yang stabil dalam ke hadapan, ke belakang, dan menyelam. Ujian mengemudi menyoroti bahawa ROV lebih fleksibel dan lebih cepat dalam manevrasi mengenai prestasi berputar kerana konfigurasi thrusters horisontal ditempatkan pada 45° di belakang ROV. Laporan ini menggambarkan pembangunan ROV melalui reka bentuk mekanikal, elektronik dan perisian. Hasil projek ini dijangka membawa kelebihan yang besar kepada industri yang berkaitan dengan aplikasi bawah air.

TABLE OF CONTENTS

PAGE

DECLARATION

APPROVAL

DEDICATIONS

ACKNOWLEDGEMENTS **1**

ABSTRACT **2**

ABSTRAK **3**

TABLE OF CONTENTS **4**

LIST OF TABLES **7**

LIST OF FIGURES **9**

LIST OF SYMBOLS AND ABBREVIATIONS **12**

LIST OF APPENDICES **13**

CHAPTER 1 INTRODUCTION **14**
1.1 Motivation 14
1.2 Problem Statement 14
1.3 Objective 16
1.4 Project Scopes 16

CHAPTER 2 LITERATURE REVIEW **17**
2.1 Background 17
2.2 ROV Classification 17
2.2.1 Working Class ROV 18
2.2.2 Observation Class ROV 18
2.2.3 Special Use ROV 19
2.3 Material of Body Frame 20
2.4 Thruster 21
2.4.1 Brushed DC Motor 21
2.4.2 Brushless DC Motor (BLDC) 22
2.4.3 Bilge Pump 23
2.5 Buoyancy and Stability 24
2.6 Propulsion System 27
2.7 Sensors and Equipment 29
2.7.1 Pressure Sensor 30
2.7.1.1 MS-5803-14BA Pressure Sensor 31
2.7.1.2 MS5837-30BA Pressure Sensor 32
2.7.1.3 Bar02 Pressure/Depth Sensor 33
2.7.2 Inertial Measurement Unit (IMU) sensor 33

2.7.2.1 IMU Digital Combo Board – 6 degrees of freedom ITG3200/ADXL345 35
2.7.2.2 MPU6050 Module Sensor 36
2.8 Related Work 38
2.8.1 BlueROV 2 38
2.8.2 BabyROV 39
2.8.3 UTP ROV Project (Universiti Teknologi PETRONAS) 40
2.8.4 UoA ROV Project (University of Auckland) 41
2.8.5 Design Comparison Table 42
2.9 Evaluation of the Design Concept Using Pairwise Comparison & Weighted Objectively Method 43
2.10 Overall Summary 47

CHAPTER 3 METHODOLOGY 48
3.1 Introduction 48
3.2 Project Flowchart 48
3.2.1 FYP 1 Flowchart 48
3.2.2 FYP 2 Flowchart 49
3.3 Fusion360 software 50
3.4 Mechanical Design 50
3.4.1 Prototype Design Sketching 51
3.4.2 Prototype Design using Fusion 360 software 51
3.4.3 Buoyancy Control 53
3.5 Electrical Design 55
3.5.1 List of Devices and Components 55
3.5.2 Schematic Circuit Wiring Connection 58
3.5.3 Schematic Flowchart for ROV Prototype Control System 59
3.6 Software Design 59
3.6.1 Arduino IDE 60
3.6.2 Processing IDE 63
3.6.3 OBS Studio 64
3.7 Finite Element Analysis (FEA) 66
3.8 Water Pressure Estimation 68
3.9 Experiment 68
3.9.1 Experiment Setup 69
3.9.2 Experiment 3: Buoyancy of ROV 70
3.9.3 Experiment 4: Underwater Operation 72
3.9.3.1 Experiment 4.1: Moving Forward and Reverse 72
3.9.3.2 Experiment 4.2: Turn Right and Left 74
3.9.4 Experiment 4.3: Raise and Submerge 76

CHAPTER 4 RESULTS AND DISCUSSIONS 78
4.1 Introduction 78
4.2 Preliminary Result of Finite Element Analysis 78
4.3 Mechanical Prototype Design 82
4.4 Experiment Results 85
4.4.1 Buoyancy of ROV 85
4.4.2 Underwater Operation 86
4.4.2.1 Moving Forward and Reverse 86
4.4.2.2 Turn Right and Left 89
4.4.2.3 Raise and Submerge 91

4.4.2.4 Overall Performance 93
4.5 Summary 94

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS 96
5.1 Conclusion 96
5.2 Recommendation 97
5.2.1 Leakage Detector 97
5.2.2 Automated Ballast System 97
5.2.3 PID Control System 98

REFERENCE 99

APPENDICES 105

LIST OF TABLES

Table 2-1: Different Combination of the Direction of Horizontal Thruster 29

Table 2: Table of Pin Connection for Bar02 Pressure/Depth Sensor 33

Table 2-3: Design Comparison Table 42

Table 2-4: Requirements Needed 43

Table 2-5: Ratings for Pairwise Comparison 43

Table 2-6: Result for Pairwise Comparison 44

Table 2-7: Requirement Ranking 44

Table 2-8: Weighted Objectively Method 46

Table 3-1: List of Devices and Components Used in ROV Prototype 55

Table 3-2: List of Experiment 68

Table 4-1: Finite Element Analysis of the Component of ROV 79

Table 4-2: Forward Test Data 86

Table 4-3: Forward Performance of ROV 86

Table 4-4: Backward Test Data 87

Table 4-5: Backward Performance of ROV 87

Table 4-6: Turn Right Test Data 89

Table 4-7: Turn Right Performance of ROV 89

Table 4-8: Turn Left Test Data 89

Table 4-9: Turn Left Performance of ROV 89

Table 4-10: Raise Test Data 91

Table 4-11: Raise Performance of ROV 91

Table 4-12: Submerge Test Data 91

Table 4-13: Submerge Performance of ROV 92

Table 4-14: Overall Performance of ROV 93

Table 4-15: Overall Turning Performance of ROV 93

LIST OF FIGURES

Figure 2-1: Example of Working Class ROV[8] 18

Figure 2-2: Example of Observation ROV[9] 19

Figure 2-3: Example of Special Use ROV[10] 19

Figure 2-4: Variations in Materials of Body Frame. (a)Aluminum Frame ROV[13] and (b) PVC Frame ROV[14] 21

Figure 2-5: Example of Brushed DC Motor Thruster [16] 22

Figure 2-6:Example of BLDC Motor with ESC [18] 23

Figure 2-7: Example of Bilge Pump [20] 23

Figure 2-8: Types of Buoyancy. (a) Positive Buoyancy (b) Neutral Buoyancy (c) Negative Buoyancy 25

Figure 2-9: Stability of ROV [30] 26

Figure 2-10: Stability on ROV Design [30] 26

Figure 2-11: Stability on Placement of Thruster [30] 27

Figure 2-12: ROV Schematic with Corresponding Degrees of Freedom 28

Figure 2-13: Arrangement of Thrusters 28

Figure 2-14: Motion of Propulsion System of Underwater ROV 29

Figure 2-15: MS-5803-14BA Pressure Sensor [43] 31

Figure 2-16: MS-5803-14BA Pressure Sensor Schematic Circuit [43] 31

Figure 2-17: MS5837-30BA Pressure Sensor [44] 32

Figure 2-18: MS5837-30BA Pressure Sensor Schematic Circuit [44] 32

Figure 2-19: Bar02 Pressure/Depth Sensor [46] 33

Figure 2-20: IMU Digital Combo Board – 6 degrees of freedom ITG3200/ADXL345 [52] 35

Figure 2-21: IMU Digital Combo Board – 6 degrees of freedom ITG3200/ADXL345 Schematic Circuit [52] 36
Figure 2-22: MPU 6050 Module Sensor [54] 37
Figure 2-23: (a)Working Axis Details and (b)MPU 6050 Module Sensor Schematic Circuit [54] 37
Figure 2-24: BlueROV 2 [55] 38
Figure 2-25: BabyROV [56] 39
Figure 2-26: UTP ROV Project [57] 40
Figure 2-27: UoA ROV Project [58] 41
Figure 3-1: FYP 1 Flowchart 48
Figure 3-2: FYP 2 Flowchart 49
Figure 3-3: Autodesk Fusion 360 50
Figure 3-4: Design Sketching of the Prototype Underwater ROV 51
Figure 3-5: Front View of the Prototype Design of Underwater ROV 51
Figure 3-6: Top View of the Prototype Design of Underwater ROV 52
Figure 3-7: Side View of the Prototype Design of Underwater ROV 52
Figure 3-8: Isometric View of the Prototype Design of Underwater ROV 53
Figure 3-9: Upper PVC Pipe (Buoyancy Tank) 53
Figure 3-10: Bottom PVC Pipe (Ballast Tank) 54
Figure 3-11: Schematic Circuit Wiring Connection Diagram 58
Figure 3-12: Schematic Flowchart Diagram for ROV Prototype Control System 59
Figure 3-13: Arduino IDE Software 60
Figure 3-14: Arduino Program Flowchart 62
Figure 3-15: Processing Software 63

Figure 3-16: GUI Display for ROV Prototype 63

Figure 3-17: OBS Studio Software 64

Figure 3-18: ROV Screen Display in OBS Studio 65

Figure 3-19: Flowchart of Finite Element Analysis 67

Figure 3-20: Experiment Setup 69

Figure 3-21: Underwater ROV Prototype in Lab Pool 71

Figure 3-22: Underwater ROV Prototype Move Forward and Backward Process 73

Figure 3-23: Underwater ROV Prototype Turn Right and Left Process 75

Figure 3-24: Underwater ROV Prototype Raise and Submerge Process 77

Figure 4-1: Isometric View of ROV Prototype 82

Figure 4-2: Top View of ROV Prototype 83

Figure 4-3: Front View of ROV Prototype 83

Figure 4-4: Side View of ROV Prototype 83

Figure 4-5: Leakproof Food Container (Pressure Hull) 84

Figure 4-6: ROV Prototype Achieve Above 90% Negatively Buoyant 85

Figure 4-7: Graph of Forward Performance of ROV 88

Figure 4-8: Graph of Backward Performance of ROV 88

Figure 4-9: Graph of Turn Right Performance of ROV 90

Figure 4-10: Graph of Turn Left Performance of ROV 90

Figure 4-11: Graph of Raise Performance of ROV 92

Figure 4-12: Graph of Submerge Performance of ROV 92

LIST OF SYMBOLS AND ABBREVIATIONS

ROVs	-	Remote Operated Vehicles
AUVs	-	Autonomous Underwater Vehicles
UUVs	-	Unmanned Underwater Vehicles
PVC	-	Polyvinyl Chloride
ABS	-	Acrylonitrile-Butadiene-Styrene
DC	-	Direct Current
BLDC	-	Brushless Direct Current Motor
FEA	-	Finite Element Analysis
IMU	-	Inertial Measurement Unit

LIST OF APPENDICES

APPENDIX A GANTT CHART FOR FYP 1 105

APPENDIX B GANTT CHART FOR FYP 2 106

APPENDIX C SCHEMATIC DRAWING OF BASE OF ROV PROTOTYPE 107

APPENDIX D SCHEMATIC DRAWING OF HORIZONTAL CONNECTOR OF ROV PROTOTYPE 108

APPENDIX E SCHEMATIC DRAWING OF VERTICAL CONNECTOR OF ROV PROTOTYPE 109

APPENDIX F SCHEMATIC DRAWING OF MICRO UNDERWATER ROV FOR MONITORING APPLICATIONS 110

APPENDIX G BILL OF MATERIAL 111

APPENDIX H PROGRAMMING CODE IN ARDUINO IDE 112

APPENDIX I PROGRAMMING CODE IN PROCESSING SOFTWARE 123

CHAPTER 1

INTRODUCTION

1.1 Motivation

This project aims to advance the understanding and development of a small underwater remotely operated vehicle (ROV) for monitoring applications. It presents an exciting opportunity to contribute to the exploration and understanding of the underwater world. ROVs also eliminate the need for human divers, reducing the risk of accidents and promoting safer operations. They can also stay underwater much longer than a human diver, expanding the time available for exploration and monitoring. ROVs can be equipped with a variety of sensors and instruments, such as cameras, video cameras, lights, manipulators, and cutting arms, allowing for diverse monitoring and data collection tasks [1]. They can be used in various industries, including search and rescue, military, recreation and discovery, aquaculture, marine biology, oil, gas, offshore energy, shipping, and submerged infrastructure [2]. The prototype design and experiments of the ROV outlined in this project may prove beneficial to researchers involved in the development of autonomous underwater vehicles (AUVs) and remotely operated vehicles (ROVs) for a wide array of underwater assignments and missions.

1.2 Problem Statement

Aquatic environment such as harbors, dams, and coastal waters require regular inspection and monitoring to ensure operation, safety, and environmental stewardship. However, accessing infrastructure that is submerged underwater on a regular basis presents significant operational obstacles and risks for human divers undertaking manual inspections. There are many constraints that can be a handicap for divers such as the ability to withstand pressure when it's necessary for them to dive high and deep inspection. The hazardousness of the underwater environment is also a problem for divers himself. Depending on type of task, costing for equipment diving and hiring

divers itself can also be higher. For inspection and monitoring the task will take more cost and time. A key shortcoming is the non-existence of versatile and inexpensive tools that can be adapted to Malaysia's complex aquatic environments, such as rivers with various flood regimes. These vital monitoring instruments are required for environmental protection work, natural resource management activities and industrial supervision across all sectors. Therefore, a solution is required that can conduct continuous, reliable monitoring of mission-critical underwater assets over prolonged periods with minimal supervisory oversight.

The exploration and monitoring of underwater environments present significant challenges due to limited accessibility, high operational costs, and technological constraints. Most current underwater monitoring approaches can't meet the demands of comprehensive and efficient data collection, particularly in small or difficult-to-operate spaces with severe restrictions. The ROV has become an essential tool in the petroleum and gas industries. In response to the needs of the offshore industry, many different types of ROV have been developed. Their responsibilities include placing underwater manifolds and inspecting the pipes while working as observers during oil exploration. These critical limitations cry out for a small, versatile underwater ROV built with advanced technology and designed specially to monitor. Therefore, human divers can be replaced to fulfill all the underwater duties.

Today sealing the components parts to prevent water from leaking into the electronic components remains a challenging problem for current designs of ROVs by using low-cost methods[3]. Moreover, once the ROV goes down it also has to solve the problem of stability during descent and ascent. How to control is also an important part. Rarely would one be able to maneuver on the surface and beneath simultaneously by a mechanical device, so we can't let our ROVs just rely on motors but have them do their own thinking as well. Existing ROV systems are effective but they suffer many limitations in terms of maneuverability, adaptability to various environments and the ability to access complex underwater structures[4]. Their bulkier designs coupled with relatively limited sensor integration and maneuvering capabilities mean that they are rather inefficient when it comes to detailed monitoring tasks carried out in confined or difficult-to-access underwater spaces. Further, due to the high costs of implementing current ROV solutions, investment becomes prohibitive rather than economical when we consider environmental monitoring and research applications as well as industrial uses where continuous underwater evaluation is required[5].

Therefore, there exists an urgent need to develop a small underwater ROV tailored for monitoring applications that address the limitations of existing systems. This solution should encompass enhanced maneuverability, versatile sensor integration, cost-effectiveness, and adaptability to various underwater conditions. The development of this innovative ROV could alter underwater monitoring techniques by permitting accurate, affordable, and accurate data collection and observations in constricted and complex underwater spaces, supporting advances in industrial processes, environmental preservation, and scientific research.

1.3 Objective

The main objectives for this project are as follows:

i. To design and develop an affordable micro underwater ROV for monitoring applications.
ii. To optimize the structural integrity and buoyancy for underwater ROV
iii. To analyze an efficient propulsion system of underwater ROV in terms of stability, velocity and acceleration.

1.4 Project Scopes

The scopes of this project are as following:

i. Design and fabricate a prototype of an underwater ROV equipped with thrusters to control the maneuverability of the ROV.
ii. Design four degrees of freedom of inspection underwater ROV with simple electronic components such as camera and pressure sensor.
iii. The waterproof enclosure needs to be designed to prevent water from leaking into the electronic circuitry part during the operation.
iv. The depth of testing the prototype at least 5 meters.
v. Environment for testing selected is controllable for control environment at laboratory pool

CHAPTER 2

LITERATURE REVIEW

2.1 Background

Underwater vehicles are crafts designed to operate submerged beneath the surface of water to perform underwater operations via on-board or wired communication. They allow us to explore and conduct tasks in rivers, lakes, seas, and oceans where it may not be feasible or safe to send human divers. There can be classified into two types of vehicles which are manned vehicles and unmanned underwater vehicles (UUVs). Manned vehicles enclose one or more human pilots in a life-support habitat allowing direct observation and operation at depth. UUVs have no humans aboard and operate autonomously or via remote control. They can be further categorized into remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs). The primary distinction between these two types is that the AUV functions autonomously, whereas the ROV is controlled by an operator [6], [7]. This project mainly focuses on developing a small ROV for monitoring applications.

2.2 ROV Classification

ROVs have evolved into specialized tools for an increasingly diverse range of subsea applications. As a result, over the decades various frameworks have emerged for categorizing ROVs based on their size, depth ranges, and functionality. From the research, the ROVs are classified into three main categories which are working class, observation class and special use [6], [7].

2.2.1 Working Class ROV

Work Class ROVs are larger and more robust vehicles with robotic manipulator arms that use deep-sea exploration. They are often used in the offshore oil and gas industry, deep-sea exploration, salvage operations, and complex underwater construction projects. They are intended to operate on the seafloor, where pressure is at its highest due to their strength and structural integrity. They are equipped with an array of cameras, sonars, sensors and tooling skids for inspection, repairs, construction, etc. They are further classified as heavy work class or light work class based on the attachments they can sustain, the load they can carry, and the depths they can reach beneath the ocean's surface [6].

Figure 2-1: Example of Working Class ROV[8]

2.2.2 Observation Class ROV

Observation Class ROVs are low-cost, compact and lightweight vehicles. They are considerably smaller than working class ROVs. They are mostly used for observation and surveillance features. They are mostly used in smaller bodies of water, such as lakes or rivers, rather than oceans. Primarily used for visual inspections, basic surveys, and data collection in offshore infrastructure, pipelines, and relatively shallow underwater environments. They are commonly equipped with some sensors, cameras, and basic manipulator arms for carrying a moderate payload. They can be used to perform surveillance in the area, inspect hazardous regions, determine whether the water is safe for human deployment, and conduct an aquatic waste survey prior to the deployment of divers in the region [6].

Figure 2-2: Example of Observation ROV[9]

2.2.3 Special Use ROV

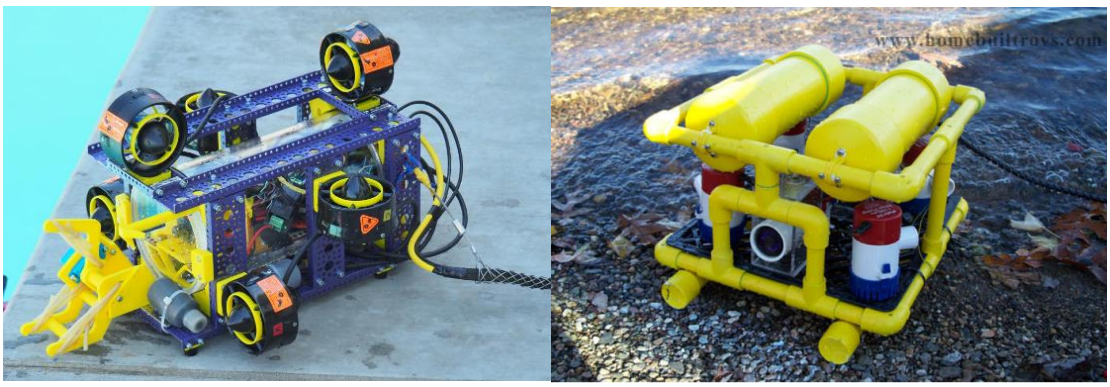
They are highly customized ROVs built for niche tasks that require special capabilities. They are less expensive than the huge working and inspection class ROVs. There are numerous permutations and combinations of attachments for these types of ROVs dependent merely on the mission specifics they serve. Because it is difficult to manage their motion in huge amounts of water, these are utilised for work in limited area such as pipelines, tanks, or submerged systems. Although they can be built to multitask, they are only designed with a specific mission in mind [6].

Figure 2-3: Example of Special Use ROV[10]

2.3 Material of Body Frame

The body frame is the main part where all the mechanical and electrical components will be mounted onto the main body. This body frame will house every component of the system and protect each component from damage in the case of a collision while operating underwater. The selection of materials used in the body frame during the fabrication of an ROV also influences the buoyancy and strength integrity of the prototype. Thus, choosing the material for the body frame properly is crucial. The body frame needs to be robust to endure underwater pressure and support the heavy components such as the thrusters. The material of the body frame needs to be lightweight to minimize the drag effect and resistant to corrosion when exposed to seawater. Metals and polymers are common materials utilized to aid the buoyancy of the ROV. Aluminum and steel are the two types of metal, with aluminum having more advantages over steel in terms of availability and weight ratio. Numerous types of polymers can be used as frames, including Polyvinyl Chloride (PVC), Acrylonitrile-Butadiene-Styrene (ABS) and acrylic [11].

The two aluminum alloys 6061 and 6063 that commonly get used in submersibles are mainly favored with very high corrosion resistivity, strong-lite weight ratio and availability. However, PVC is a common raw material for shallow-water ROV and AUV applications due to its inexpensive cost, extensive availability in hardware shops, superior corrosion resistance, and ease of construction. It is readily cut and drilled with ordinary hand and power woodworking tools. It can be attached or melted together. PVC's low melting point could be a drawback because it melts easily when used with power tools such as a lathe. They also use a box-shaped steel, aluminum, or PVC frame that houses various canisters made of different metals and polymers underwater in ROVs since most of them do not have a fairing [12]. Most of the ROVs used PVC pipe for their lower economic range frame in the majority of the articles and journals regarding the ROVs that have been studied so far.



(a) (b)

Figure 2-4: Variations in Materials of Body Frame. (a)Aluminum Frame ROV[13] and (b) PVC Frame ROV[14]

2.4 Thruster

The thruster is one of the primary components that acts as a propelling force in the underwater robot's operation. This enables underwater robots' motion performance to move as demanded, such as maneuvering horizontally to move forward and backward and vertically to move up or down. According to (R. Singh et al.), maneuverability, degree of freedom (DOF), speed and to some extent depth control can be altered by the number of thrusters employed. The stability of the ROV is determined by many factors, one effect being synchronization between thrusters. Once one or more failures occur, overall device malfunction is possible and rise can be anticipated under the actuated situations [6]. Thruster in the propulsion system must be carefully chosen since the more thruster power required, the heavier the propulsion system on the ROV.[11] Three types of thrusters commonly used for underwater ROV are brushed DC motor, brushless DC motor and bilge pump.

2.4.1 Brushed DC Motor

Brushed DC motors are usually cheap and easy to control. It begins operating when attached to a suitable battery with two cables. This is because the commutator, which switches current flow through the windings, is incorporated directly into the motor[12]. They were used by ROV manufacturers in the recent past. For instance, they are integral to Deep Ocean Engineering's version III Phantom [15]. However due to advances in other DC motor technologies their use has become very rare nowadays.

Because of the mechanical limitations on brush functions, they require periodical maintenance and have a lower speed range. They aren't exactly the perfect motor to use for thrusters either; apart from having an open enclosure that has to be opened again to change brushes and low thrust output, there is a higher risk of water slipping inside through the outlet shaft seal. Despite these drawbacks, brushed DC motors have historically been used in ROVs due to their simplicity, cost-effectiveness, and ease of control.

Figure 2-5: Example of Brushed DC Motor Thruster [16]

2.4.2 Brushless DC Motor (BLDC)

The BLDC motors do not require brushes because commutation is accomplished using electronic controllers. They get rid of the brushes and commutator characteristic of brushed motors, so there is less to wear out. As Ishak et al. [17] declare, when used in extreme conditions they are the preferred option over the brushed DC motor. They have a larger speed range, less maintenance demand, higher efficiency and better heat properties. Most of these advantages revolve around the fact that brushless motors are better able to dissipate heat. Most of the heat created by a motor occurs in its windings. But this heat must be dissipated even as it is generated, or else the motor will overheat. But unlike the windings on a brushless motor, which are directly attached to the outside surface of the motor and so can more easily dissipate excess heat. As a result, the brushless motor can be pushed much harder without overheating than its brushed counterpart of physical equal size [12]. However, they do need an electronic speed controller (ESC) for speed control, and are generally more expensive. PWM signals are transmitted to the ESC using the microcontroller as the motor's speed is controlled by the PWM duty cycle. Furthermore, brushless DC motors usually require gearing to achieve optimum efficiency.

Figure 2-6:Example of BLDC Motor with ESC [18]

2.4.3 Bilge Pump

Bilge pump motors are used to remove water from the bottom of boats. They run on 12 V batteries and are already water-proofed, so naturally they've become the most used motors for small underwater vehicles. Bilge pump motors combine an enclosed direct current motor with an axial flow impeller whose underwater thrust is excellent from a very small, self-contained package. Turbines are waterproof constructed and have simple two-wire power connectivity, so they can be rapidly installed onto ROV thruster nacelles or tunnels with minimal housing fabrication. They are low-cost, distributed fairly widely in boating stores and well-sized with output power for small submarines running on calm water. But they must be altered to take a propeller. How to attach a propeller to the motor shaft is detailed in this section later [12]. Bilge pumps are an alternative to thrusters that can lower ROV costs, but also reduce ROV speed[19].

Figure 2-7: Example of Bilge Pump [20]

2.5 Buoyancy and Stability

The buoyancy and stability of ROVs are crucial for their performance. Buoyancy is the upward force exerted on an object immersed in a fluid due to the displacement of the fluid[21]. According to Archimedes' Principle, the weight of the fluid that an object displaces is equal to the buoyant force acting on it [22], [23]. Submarines and other underwater vehicles that can both float and sink in the water have been designed using this principle. Achieving neutral buoyancy is essential for a ROV to stay in place at any depth and avoid floating or sinking.

From Figure 2.8, there three types of buoyancy are encountered which can affect the ROV's ability to maneuver, control its depth, and carry out tasks effectively in underwater environments [24], [25]:

i. Positive buoyancy occurs when an object is lighter than the fluid it displaces, the object will float because the buoyant force is greater than the object's weight. This type of buoyancy makes it easy for the ROV to sink and return back easily in cases where an electrical failure has occurred, thereby improving safety and recovery.

ii. Neutral buoyancy occurs when an object's weight is equal to the fluid it displaces. In this state, the object neither floats nor sinks. ROVs' neutral buoyancy is classified as desirable because it provides an excellent environment for detailed movement and control at certain depth where inspections can be done or maintenance works carried out.

iii. Negative buoyancy occurs when an object is denser than the fluid it displaces. The object will sink because its weight is greater than the buoyant force. Negative buoyancy is a useful feature in the operation of ROV underwater as it enables descent and different depth maneuvers.

These different types of buoyancy have specific advantages and are used based on the operational requirements of the ROV, such as recovery, maneuverability, and task performance [25].

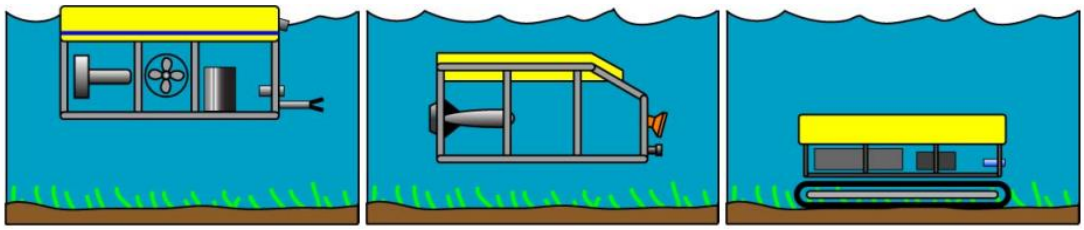


Figure 2-8: Types of Buoyancy. (a) Positive Buoyancy (b) Neutral Buoyancy (c) Negative Buoyancy

A force that acts vertically upward towards the ROV centroid is called the buoyant force which can be used to determine the weight of submerged underwater ROV. Thus, the following concept of calculation should be followed based on the Archimedes' Principle formula [26]:

$$\boldsymbol{F_b = \gamma_f V_d} \qquad 2\text{-}1$$

Where:

F_b = Buoyancy force

γ_f = Specific weight of the fluid

V_d = Displaced volume of the fluid

Underwater stability refers to an ROV's ability to return to an upright position when accidentally tipped by external disturbances like thruster forces or turbulent water from waves [27]. The placement of the Centre of Buoyancy (COB) and Centre of Gravity (COG) on an underwater object defines stability. The distance between COG and COB, known as Buoyancy Gradient (BG), significantly influences an ROV's stability. Increasing BG enhances stability but diminishes maneuverability. Conversely, reducing BG or aligning COG and COB reduces stability, enhancing maneuverability but making the ROV prone to instability without proper control. Maintaining a higher COB above the COG is crucial for an ROV's stability [28], [29], [30]. The buoyancy tank was mounted at the highest feasible point on the vehicle and placed the heavier components as low as possible maximizing the BG [31], [32]. Hence, the ROV will the maximum stability when it has a high COB and a low COG with vertical in line with each other as shown in Figure 2.9.

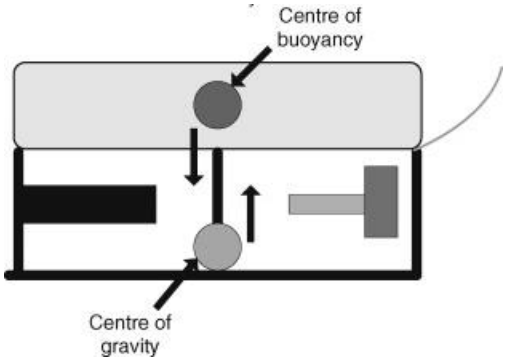


Figure 2-9: Stability of ROV [30]

A light-weight with increased buoyancy of the underwater ROV resulted in positive longitudinal and lateral stability. This led to a stable vehicle along the pitch and roll axes. An optimal design for an ROV involves maintaining a specific aspect ratio between its width and length, typically aiming for a length that is twice the width. This design choice maximizes stability to its fullest extent [33]. Figure 2.10 illustrates the design of ROV that offers the most stability. Additionally, the positioning of the thrusters holds significance. Placing the thrusters at the frame's edges ensures maximum stability. Figure 2.11 shows the ideal placement for thrusters [30].

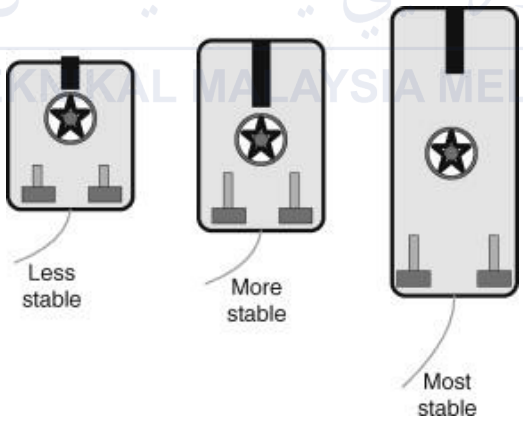


Figure 2-10: Stability on ROV Design [30]

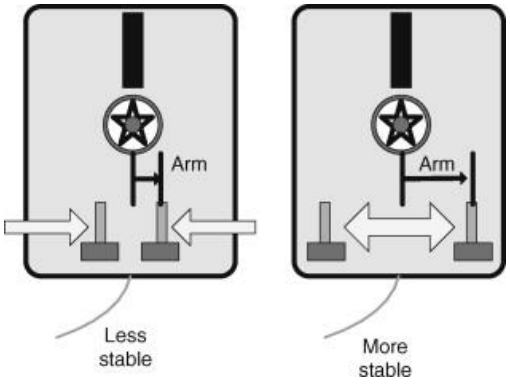


Figure 2-11: Stability on Placement of Thruster [30]

.

2.6 Propulsion System

The propulsion system of the ROV is made up of two or more thrusters that push the vehicle and allow navigation to the work side. Thrusters must be mounted on the vehicle so that the moment arm of their thrust force is parallel to the vehicle's centre of mass and allow for proper maneuverability and control. The thrust-to-physical size/drag and power input ratios of the propulsion system must be high [34]. (Joochim et al., 2016). This indicate that the propulsion system can generate a significant amount of thrust with relatively less energy input, resulting in higher efficiency [35]. Efficient propulsion systems help maximize the use of available power, which is particularly important in environments where energy resources may not be easily available [36].

The arrangement of the thrusters is the most significant aspect that will affect the ROV's propulsion system. The thruster must be appropriately arranged so that the underwater ROV may move precisely without damaging the its stability. Typically, three or four thrusters with variable location are used in the creation of small and micro class ROVs, however the work class of ROV could include more than five thrusters due to size and weight [37].

There are several arrangements for the thruster to allow for varied degrees of maneuverability. Asymmetrical thrusting based on thruster arrangement and altering thruster output is used to achieve maneuverability. The more thrusters the ROV utilises, the more precisely it can maneuver according to its degree of freedom, but this also increases the weight and expenditure of the ROV. For instance, ROVs can be

equipped with four, six, or eight thrusters to enable full six-degree-of-freedom movement, including surge, sway, heave, yaw, pitch, and roll [38].

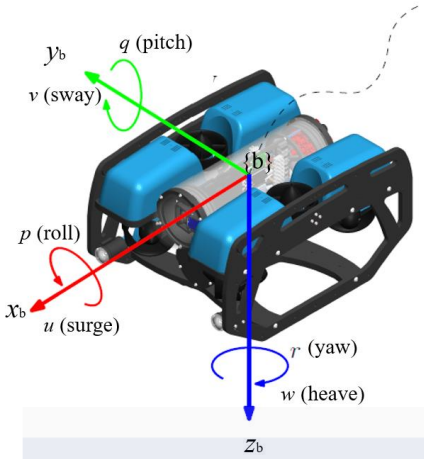


Figure 2-12: ROV Schematic with Corresponding Degrees of Freedom

In general, three thrusters ROV will have two rears and yaw motions, with two thrusters in horizontal direction and the other located centrally and oriented vertically; four thrusters ROV will have an additional lateral thrust with an additional one thruster in horizontal in x-axis; and five thrusters ROV will have four thrusters at the corner and one vertically direction [37]. Placing the thrusters off the vehicle's longitudinal axis results in a better turning moment, as illustrated in Figure 2.13 [37], [39].

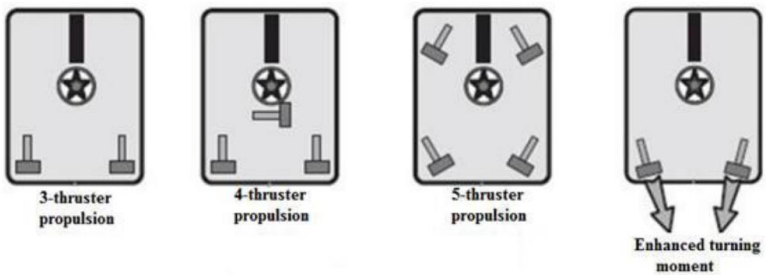
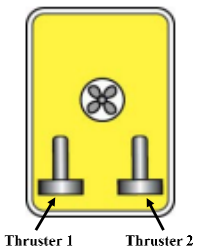


Figure 2-13: Arrangement of Thrusters

The integration of two horizontal thrusters in an underwater ROV can permit lateral and horizontal motion. By combining numerous combinations of the direction of these two thrusters, an extensive variety of horizontal vehicle motions is achievable as shown in Figure 2.14.

Table 2-1: Different Combination of the Direction of Horizontal Thruster

	Thruster 1	Thruster 2	Motion
	CW	CW	Forward
	CCW	CCW	Reverse
	CW	-	Turn Right
	-	CW	Turn Left
	CW	CCW	Spin Right
	CCW	CW	Spin Left

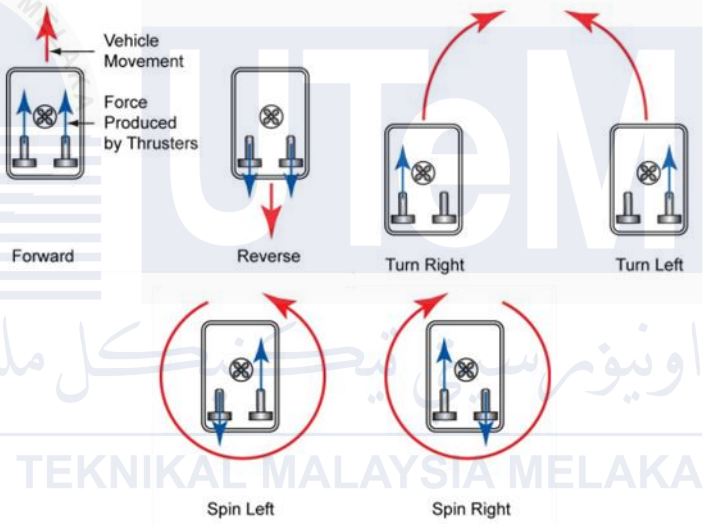


Figure 2-14: Motion of Propulsion System of Underwater ROV

2.7 Sensors and Equipment

The design and development of a micro underwater Remotely Operated Vehicle (ROV) for monitoring applications are highly reliant on the selection of sensors and equipment. This part studies the various types of sensors and equipment that are typically used in underwater ROVs, their functions, and their significance in the effective operation and data collection of the prototype. A sensor is a device that detects changes in the environment, whether physical or chemical. Typically, a sensor requires a signal filtering circuit, also known as a sensor transmitter. This sensor transmitter generates a DC electrical signal that can be either voltage or current. This

signal is then sent via an electrical conductor, the transmission range of which is determined by the conductor's quality and signal quantity. In this study, two types of sensors will be used which are IMU (Inertial Measurement Unit) sensor for detecting rotational motion and position, and a pressure sensor to measure water depth [40].

2.7.1 Pressure Sensor

Pressure sensors are essential for the precise measurement of pressure, depth, and the maintenance of operational depths in underwater ROVs. These sensors function by converting water pressure into electrical signals to determine depth, which is essential for effective navigation and prevent obstacles [41]. The depth value is calculated from the pressure value by using the hydrostatic equation given by

$$\mathbf{P = \rho g h} \qquad 2\text{-}2$$

where

P is the pressure at the given depth,

ρ is the density of the fluid ($1000 \text{kg} m^{-3}$ for water)

g is the acceleration due to gravity ($9.81 ms^{-2}$)

h is the depth below the surface of the water.

For instance, a pressure change of 1MPa equates to a depth change of 99.55 m, given the gravity rate of $g = 9.81 \text{ms}^{-2}$ and sea water density of $\rho = 1025 \text{ kgm}^{-3}$. This is especially important in efficient ROV operations because depth data can help in maintaining optimum water depth during such operations as monitoring. The sensor also helps to maintain the structural stability of the ROV and does not allow it to fall to dangerous depths. In addition, depth data is significant in the field of scientific research and investigations that take place after the mission, because it helps to explain or complement the other data accumulated and when the next mission is being prepared. Pressure sensors are indispensable for a wide range of ROV applications due to their adaptability to various underwater environments, which guarantees consistent performance [42].

2.7.1.1 MS-5803-14BA Pressure Sensor

The MS-5803-14BA sensor is a piezo-resistive pressure sensor suited to utilisation in water. Figure 2-15 depicts the physical design of the sensor. The MS-580.14BA sensor is a new generation of high-resolution pressure sensors that support both SPI and I2C bus interfaces. It is intended for depth measurement systems with a water depth resolution of 1 cm. A high-pressure linear sensor is included in the sensor module. This sensor works by using hydrostatic pressure to determine the depth of water. The pressure at a certain depth is determined by the mass of water present. Figure 2-16 depicts the schematic circuit for the MS-5803-14BA sensor, which is compatible with the microcontroller. The communication protocol is simple and does not require the device to be internally programmed with registers. The sensor is fitted with a gel protection and an anti-magnetic stainless-steel cap that can withstand pressures greater than 30 bar. The generation of this sensor module has utilised MEMS technology. The utilisation of this sensing approach results in limited hysteresis and excellent stability of both pressure and temperature signals [43].

Figure 2-15: MS-5803-14BA Pressure Sensor [43]

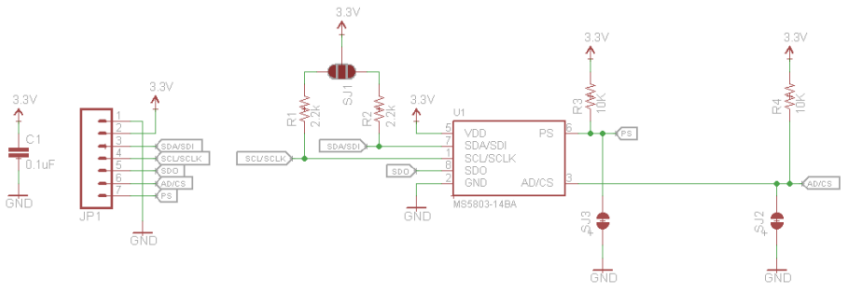


Figure 2-16: MS-5803-14BA Pressure Sensor Schematic Circuit [43]

2.7.1.2 MS5837-30BA Pressure Sensor

The MS5837-30BA is an essential component of ROVs as it is designed specifically for underwater applications. Despite its compact size and robust design, it can withstand the harsh conditions found in underwater environments. The instrument is capable of measuring depths up to approximately 300 meters with a pressure range of 0–30 bar. This sensor provides highly accurate depth measurement to ROVs through its water depth resolution of 2 mm (0.2 millibar). Water pressure is converted into electrical signals by a piezoresistive sensing element, ensuring accurate data for maintaining desired operating depths and structural integrity. I2C bus interface is another feature of the MS5837-30BA, which makes it compatible with microcontrollers such as Arduino, Raspberry Pi and autopilot systems like Pixhawk, thereby improving ROV performance and safety in underwater environments, as well as facilitating its integration into existing ROV designs [44], [45].

Figure 2-17: MS5837-30BA Pressure Sensor [44]

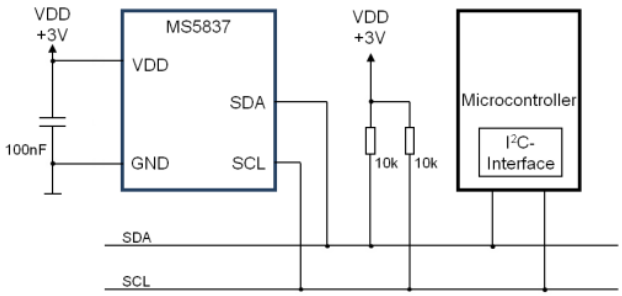


Figure 2-18: MS5837-30BA Pressure Sensor Schematic Circuit [44]

2.7.1.3 Bar02 Pressure/Depth Sensor

The Bar02 developed by BlueRobotics is pressure sensor of high accuracy that is suitable for underwater operation. Figure 2-19 depicts the physical design of the sensor. It is able to operate in pressures up to 2 Bar; or even approximately 10 meters of water head. It uses I2C communication protocol and has 4 pin DF-13 pin layout which makes it compatible and flexible in its interaction with different microcontrollers and autopilot. The sensor can measure height in air with a resolution of 13 cm and offers a remarkable 0.16 mm pressure resolution. Moreover, it has a temperature sensor with a ±2°C precision. The Bar02 can be readily integrated with a variety of systems, including Pixhawk, Arduino, and Raspberry Pi. It supports power inputs up to 5.5V and runs at a 3.3V I2C voltage. In order to provide crucial depth measurement and navigation data and to ensure safe and effective underwater operations, this sensor is frequently utilised in underwater ROVs and other applications. In spite of its superior resolution and accuracy, it is an essential tool for conducting scientific research, managing operating depths, and navigating underwater environments [46], [47].

Figure 2-19: Bar02 Pressure/Depth Sensor [46]

Table 2: Table of Pin Connection for Bar02 Pressure/Depth Sensor

Pins (Wire Colour)	Functionality	Arduino Pins
VCC (Red)	Positive pole	3.3V or 5V
GND (Black)	Negative pole	GND
SDA (White)	I2C-SDA	A4
SCL (Green)	I2C-SCL	A5

2.7.2 Inertial Measurement Unit (IMU) sensor

The Inertial Measurement Unit (IMU) sensor in ROVs combines accelerometers, gyroscopes, and sometimes magnetometers to perform precise

navigation and stability. While the accelerometers measure linear acceleration along more than one axis providing the position feedback and enable the controller to perform steady state control. It is an electrical device that are extremely sensitive and accurate over time in measuring acceleration caused by gravity or body movement because they contain sensitive parts that are free from motion mechanically. It provides the acceleration values in the form of a 3-axis vector which are A_{CC_x}, A_{CC_Y} and A_{CC_Z}.

Gyroscopes measure the rotational speed of the vehicle, which in turn contributes to assessing position and rotation, both capabilities required to ensure stability during flight heading and skid. It provides the angular velocity values in the form of a 3-axis vector which are ω_x, ω_Y and ω_z. Thus, accelerometers and gyroscope can be used to calculate the roll, roll, and pitch angles by using the following equations [48]:

$$\boldsymbol{\theta_{roll}} = \mathbf{tan^{-1}}\left(\frac{\mathbf{A_{CC_Y}}}{\sqrt{\mathbf{A_{CC_x}}^{\mathbf{2}} + \mathbf{A_{CC_Z}}^{\mathbf{2}}}}\right) \quad \text{2-3}$$

$$\boldsymbol{\theta_{pitch}} = \mathbf{tan^{-1}}\left(\frac{-\mathbf{A_{CC_X}}}{\sqrt{\mathbf{A_{CC_Y}}^{\mathbf{2}} + \mathbf{A_{CC_Z}}^{\mathbf{2}}}}\right) \quad \text{2-4}$$

$$\boldsymbol{\theta_{yaw}} = \int \boldsymbol{\omega_z}\, \boldsymbol{dt} \quad \text{2-5}$$

As a result, the system's combination of the accelerometer and gyroscope will yield an exact angle measurement because their respective shortcomings will be balanced [49].

Magnetometers measure the local magnetic field strength and direction, which help determine compass heading as well as compensating for systematic errors of the IMU (using roll, pitch and yaw corrections). This integrated sensor data enables the ROV control system to adjust thrusters and actuators in real time to maintain smooth, stable operation and precise maneuvering for tasks like inspection, data collection and obstacle navigation - all important functions in underwater missions that make the IMU a must-have component [49], [50].

2.7.2.1 IMU Digital Combo Board – 6 degrees of freedom ITG3200/ADXL345

Figure 2-20 shows The IMU Digital Combo Board – 6 degrees of freedom ITG3200/ADXL345 which is a very tiny and flexible inertial measurement unit sensor that can detect motions in six directions. The unit combines a 3-axis gyroscope with an accelerometer in order to measure angular velocity, linear acceleration, or changes in pitch, roll or yaw. The sensor communicates through a digital I2C interface which makes it quite simple to have it integrated into microcontrollers, single boards computers or any other form of digital systems. The IMU Digital Combo Board is suitable for a wide range of applications, including motion tracking and gesture recognition in consumer electronics, attitude and orientation sensing in unmanned aerial vehicles (UAVs) and robotics, vibration and tilt monitoring in industrial and structural health monitoring applications, navigation and positioning in GPS-denied environments, and activity tracking and motion analysis in sports and fitness applications. Among the documentation, example codes and guide resources provided, the IMU Digital Combo Board's 6DOF motion sensing can be quickly served through incorporation into projects but it is imperative that developers use this opportunity judiciously [51], [52].

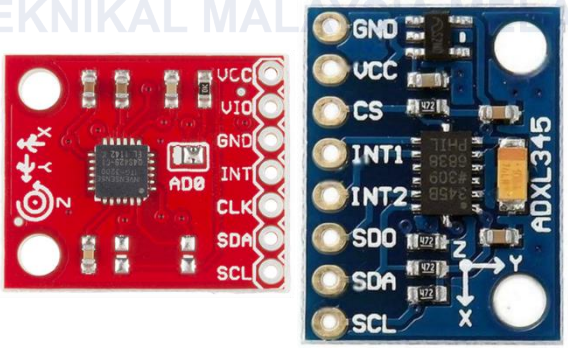


Figure 2-20: IMU Digital Combo Board – 6 degrees of freedom ITG3200/ADXL345 [52]

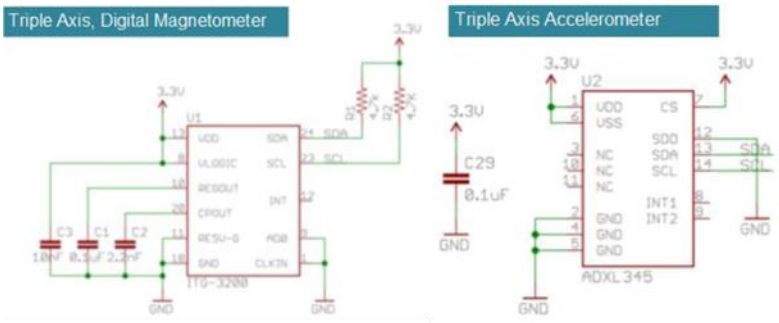


Figure 2-21: IMU Digital Combo Board – 6 degrees of freedom
ITG3200/ADXL345 Schematic Circuit [52]

2.7.2.2 MPU6050 Module Sensor

The MPU6050 is a well-known 6-axis motion tracking device that combines a 3-axis accelerometer and a 3-axis gyroscope to provide comprehensive motion data. When integrated with an external compass (magnetometer) via the I2C communication protocol, a high-speed communication at up to 400 kHz can be achieved and the sensor can be easily interfaced with a wide range of microcontrollers. The MPU6050 sensor is a popular choice for a variety of applications, including motion-based control in gaming controllers and remote controls, attitude and orientation sensing in robotics and drones, rehabilitation systems and etc due to its compact size. The sensor involves a Digital Motion Processor (DMP) responsible for sensor fusion and output processed motion data, which eases computation for the host microcontroller. Besides, it comes with configurable low pass filters for both the gyroscope and accelerometer, thereby allowing users to improve the sensor's performance based on their specific [53]. Figure 2-23(a) shows the block diagram of the MPU6050 module for a better understanding of the working of the sensor and Figure 2-23(b) depicts its circuit diagram [54].

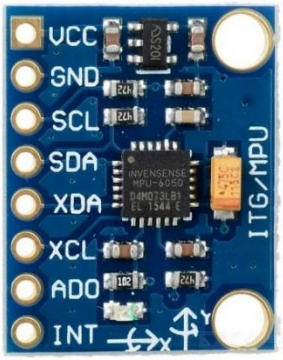


Figure 2-22: MPU 6050 Module Sensor [54]

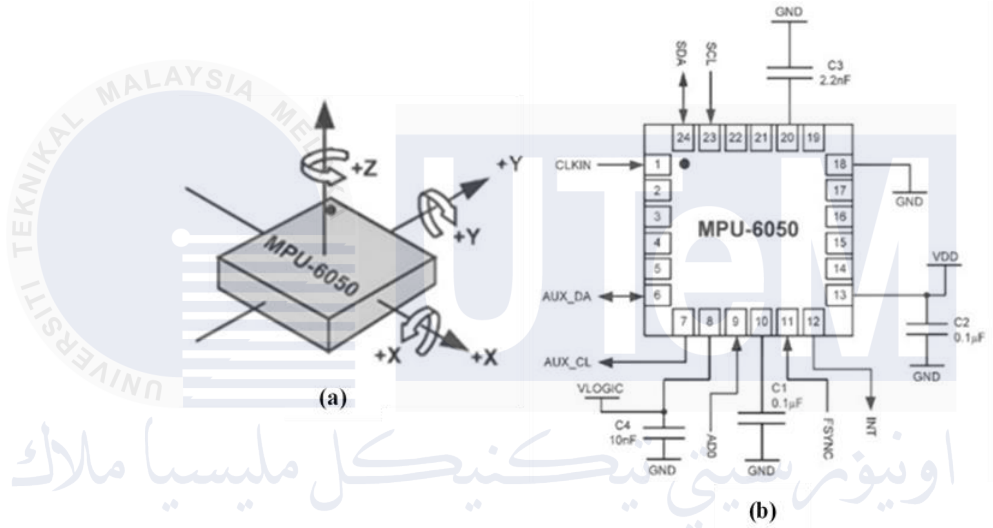


Figure 2-23: (a)Working Axis Details and (b)MPU 6050 Module Sensor Schematic Circuit [54]

2.8 Related Work

There are some relevant journals worldwide about the design and development of observation ROV after searching the research. These related journals included:

2.8.1 BlueROV 2

The Blue-ROV2 was developed in 2016 by Blue Robotics, a company specializing in marine robotics. as illustrated in Figure 2-24. The frame is built with anodized aluminum and contains High-Density Polyethylene (HDPE) panels that protect and support the vehicle's electronics and other components. It is equipped with T200 Thrusters, which are BLDC motors designed to work in a marine environment. It comes with six thrusters but can be expanded to eight thrusters with a vectored configuration, providing control in six degrees of freedom for precision navigation. Due to its open-ended structure designed, it can be exploited from its basic model to several high-end configurations. This allows the BlueROV 2 become a highly adaptive and flexible ROV. The BlueROV 2 typically suitable for a range of underwater tasks since it has a depth rating of around 100 meters. It often installed with high-resolution cameras, allowing for real-time video streaming and recording for inspection and observation. This vehicle is also available in other types of accessories, including sensors and manipulator arms that can be customized to suit different purposes [55].

Figure 2-24: BlueROV 2 [55]

2.8.2 BabyROV

In 2014, the Marine Technology Centre at Universiti Teknologi Malaysia developed the Baby ROV, which is depicted in Figure 2-25. The ROV was constructed by low cost material which are commercial grade polyvinyl chloride (PVC) pipes. It was designed in bullet-shaped framework to improve aerodynamics and was equipped with PIC controller to operate the vehicle. The dimensions of BabyROV are 35cm x 35cm x 25cm (length×breadth×height) with weighed around 8.5kg which ensure that the power of the thrusters to the weight ratio for the ROV is high. It was maneuvered by three thrusters which make from motors from the car vacuum cleaner and powered by a 12 V lithium polymer battery which allows to operate underwater for approximately 40 minutes. The BabyROV is able to up to submerge up to 20m and it tested to submerging to a depth of 2.5m in the UTM towing tank without any leaking problems in the main body, demonstrating its functionality in underwater environments. It was controlled using a PS2 joystick controller through a network cable and installed with IP CMOS Network camera which utilized LAN cable to transmit visual data to the computer. The BabyROV was cost around RM 2102. The project allowed for the employment of more efficient thrusters and propellers [56].

Figure 2-25: BabyROV [56]

2.8.3 UTP ROV Project (Universiti Teknologi PETRONAS)

The UTP ROV is developed by Ahmad Syahmi bin Salim from Universiti Teknologi PETRONAS. The design of UTP ROV was simple and cost-efficient. The frame of this ROV is made of aluminium and a waterproof enclosure is designed using Lock and Lock food container to protect the controller system. The dimension of the ROV is about 25cm×25cm×50cm (length × breadth × height). Washer and metal plates are attached to the frame to achieve neutral buoyancy when operate in underwater. There are 3 thrusters installed by using bilge pumps in this ROV to manoeuvre in underwater which two thrusters were used simultaneously for horizontal movement while only one thruster will be used for vertical movement. The UTP ROV is equipped with PIC16F877A as the main microcontroller and connected to motor driver to control the movement of the ROV. Besides, it used sewer inspection camera which build in adjustable lights for the inspection in underwater and gauge pressure sensor for depth function of the ROV. The UTP ROV is able to up to submerge up to 10m since the tether length is only 10m [57].

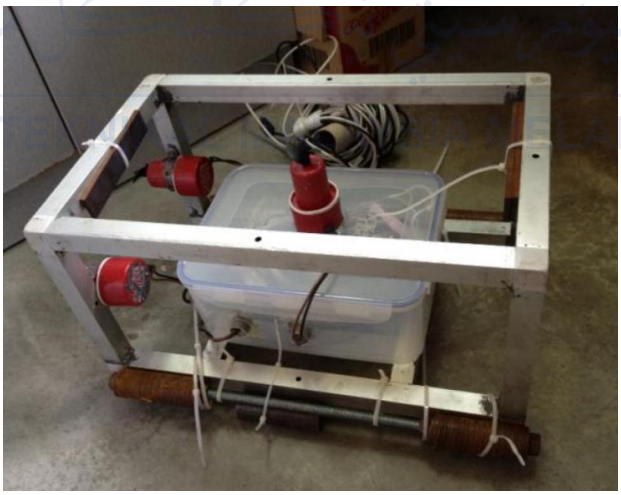


Figure 2-26: UTP ROV Project [57]

2.8.4 UoA ROV Project (University of Auckland)

The University of Auckland developed the UoA ROV, which is an open frame micro observation or inspection class ROV with six Blue Robotics T-100 thrusters. The vehicle consists of two watertight acrylic enclosures that are installed individually on an aluminium frame. One enclosure contains the control system and camera, while the other includes a battery. The dimension of UoA ROV is 54cm×34cm ×31cm.Two high-intensity luminous lights are attached at the front end of the ROV for clearer visual observations. UoA ROV with weight of 8.6 kg in the air and become slightly positively buoyant in water by mounting subsea polyurethane foam on both sides. The ROV has an average speed of 0.99 m/s and a diving depth of up to 10 m, making it suitable for shallow-water marine investigations. The ROV's electronic system is controlled by a surface station (laptop) and a joystick, with a Raspberry Pi 3 B serving as a link between the user interface and the ROV's control unit. The control unit consists of one Arduino Uno board, an ESC, and thrusters. The ROV also has several navigation sensors, including an LS20031 GPS receiver, a 30 bar pressure/depth sensor, and an MPU-9250 (3-DOF Gyroscope, 3-DOF Accelerometer, 3-DOF Magnetometer) sensor. The ROV is powered by a 14.8 V lithium-ion battery. The UoA ROV is neutrally buoyant and stable underwater, making it suitable for marine debris and ship hull inspections [58].

Figure 2-27: UoA ROV Project [58]

2.8.5 Design Comparison Table

Table 2-3: Design Comparison Table

Parameter	**Design 1**	**Design 2**	**Design 3**	**Design 4**	**My Design**
	BlueROV 2	BabyROV	UTP ROV	UoA ROV	ROV
Size (L×W×H)	62cm×45cm×35cm	35cm×35cm×25cm	25cm×25cm×50cm	54cm×34cm ×31cm	28cm×30cm ×23cm
Material	Anodized aluminium, HDPE	PVC	Aluminum	Anodized aluminum, Acrylic	Aluminum
Type of Thruster	T-200 Thruster (BLDC Motor)	Motors from Car Vacuum Cleaners	Bilge Pump	T-100 Thruster (BLDC Motor)	BLDC Motor
Number of Thruster	6-8	3	3	6	4
Microcontroller	Raspberry Pi	PIC	PIC16F877A	Raspberry Pi 3B, Arduino Uno	Arduino Uno
Tether	300m	35m	10m	90m	30m
Depth	100m	20m	10m	10m	15m
Camera	1080p USB camera	IP CMOS Cam	Video camera with build in adjustable lights	Raspberry Pi camera	Waterproof Endoscope Camera
Number of Lights	2-4	2	-	2	2
Sensor	✓ 3-DOF Gyroscope ✓ 3-DOF Accelerometer ✓ 3-DOF Magnetometer, ✓ Internal barometer, ✓ Blue Robotics Bar 30 Pressure/Depth & Temperature Sensor ✓ Current and Voltage Sensing ✓ Leak Detection Sensor	-	✓ Pressure Sensor	✓ LS20031 GPS receiver ✓ 30 bar pressure/ depth ✓ Current and Voltage Sensing ✓ Inertial MPU9250 (3-DOF Gyroscope, 3-DOF Accelerometer, 3-DOF Magnetometer) sensor	✓ Blue Robotics Bar 02 Pressure/Depth & Temperature Sensor ✓ MPU 6050 (6 DOF Accelerometer and Gyroscope) sensor

2.9 Evaluation of the Design Concept Using Pairwise Comparison & Weighted Objectively Method

Pairwise comparison and weighted objectively method are two techniques used in requirement prioritization and design concept evaluation. These two methods have been widely utilized in various fields, including architectural design, where it can be used for assessing and selecting design proposals.

Table 2-4: Requirements Needed

Requirement	**Specification**
Size	Micro
Material	Aluminium
Design	High Stability and Neutral Buoyancy
Propulsion	4 BLDC Motor Thruster
Microcontroller	Real-time applications which provide navigation tasks
Tether	30m
Depth	15m

Based on Table 2-3, we have chosen a few requirements need for design and development of the micro underwater ROV for monitoring applications. The AHP's ratings for pairwise comparison can be used to determine the important requirement on designing the ROV.

Table 2-5: Ratings for Pairwise Comparison

Rating Factor	Relative Rating of Importance of Two Selection Criteria A and B	Explanation of Rating
1	A = B	The two are the same with respect to the criterion in question
3	A is thought to be moderately superior to B	Decision maker slightly favors A over B
5	A is thought to be strongly superior to B	Decision maker slightly favors A over B
7	A is demonstrated to be superior to B	A's dominance over B has been demonstrated
9	A is demonstrated to be absolutely superior to B	There is the highest possible degree of evidence that proves A to be superior to B under appropriate conditions

By using the rating factor in Table 2-4, we rate each requirement as shown in Table 2-5 below. The requirement in the first column of Table 2-5 is considered as Requirement A and the criteria in the first row of Table 2-5 are considered as Requirement B. Reciprocal values of the rating factor are used to complete the corresponding values of each requirement. For example, if Requirement A is much more important than Requirement B and the rating factor is 9, then the corresponding value for Requirement B to Requirement A is 1/9 or 0.11.

Table 2-6: Result for Pairwise Comparison

Requirement	Size	Material	Design	Propulsion	Microcontroller	Tether	Depth	**Totals**	**Weights**
Size	1	1/3	3	1/7	5	9	9	27.48	0.19
Material	3	1	7	5	9	9	9	43.00	0.29
Design	1/3	1/7	1	1/7	5	7	7	20.62	0.14
Propulsion	7	1/5	7	1	7	5	7	34.20	0.23
Microcontroller	1/5	1/9	1/5	1/7	1	3	9	13.65	0.09
Tether	1/9	1/9	1/7	1/5	1/3	1	3	4.90	0.03
Depth	1/9	1/9	1/7	1/7	1/9	1/3	1	1.95	0.01
SUM								**145.80**	**1.00**

Table 2-7: Requirement Ranking

Requirement	**Weights**	**Rank**
Size	0.19	3
Material	0.29	1
Design	0.14	4
Propulsion	0.23	2
Microcontroller	0.09	5
Tether	0.03	6
Depth	0.01	7

From Table 2-6, it can be seen that material is the most important requirement to others for designing ROV in this project compare. The selected materials influence the ROV's weight and buoyancy, which in turn affect its manoeuvrability and overall performance. On the other hand, the second important requirement is propulsion system of the ROV. The propulsion system directly impacts the ROV's manoeuvrability and efficiency in underwater operations. Efficient propulsion systems maximize the use of available power, leading to longer mission endurance and the ability to accomplish more tasks during operation. Conversely, the depth of an ROV is the lowest ranked compared to other requirement due to its relatively lower impact on the overall design and functionality of the vehicle and it is often a specific operational requirement that can be addressed within the broader design framework. In context, the material and propulsion must prioritize compared to other criteria in design and development the underwater ROV for monitoring applications.

By using the rating system to rate each alternative in Table 2-7 below with scale of 1 to 5 where 1 indicates poor and 5 indicates excellent.

Table 2-8: Weighted Objectively Method

Requirement	**Weights**	**Design 1**		**Design 2**		**Design 3**		**Design 4**		**Design 5**	
		BlueROV 2		BabyROV		UTP ROV		UoA ROV		ROV	
Size	0.19	3	62cm×45cm×35cm	4	35cm×35cm×25cm	4	25cm×25cm×50cm	4	54cm×34cm×31cm	4	28cm×30cm×23cm
Material	0.29	5	Anodized aluminium, HDPE	2	PVC	5	Aluminum	4	Anodized aluminum, Acrylic	5	Aluminum
Design	0.14	4	Polyurethane buoyancy foam and Additional stainless steel ballast weight	1	-	4	Additional metal plates attached to frame	3	Additional subsea polyurethane foam	4	PVC pipe filled with air, additional foam and adjustable PVC pipe filled with load
Propulsion	0.23	3	6-8 T-200 Thrusters (BLDC Motor)	1	3 Motors from Car Vacuum Cleaners	2	3 Bilge Pumps	3	6 T-100 Thrusters (BLDC Motor)	4	4 BLDC Motors
Microcontroller	0.09	3	Raspberry Pi	2	PIC	2	PIC16F877A	5	Raspberry Pi 3B, Arduino Uno	3	Arduino Uno
Tether	0.03	1	300m	4	35m	2	10m	2	90m	4	30m
Depth	0.01	1	100m	4	20m	3	10m	2	10m	3	15m
Total Score	**1.00**	**3.6373**		**2.0949**		**3.5579**		**3.6237**		**4.1879**	

From the result above, design 5 shows the highest total score which indicate that it is the best concept design of ROV in this project.

2.10 Overall Summary

In conclusion, the overall dimension of underwater ROV for this project will be 28cm×30cm ×23cm, thus it is small enough to become micro ROV. Aluminum will be chosen for the frame of underwater ROV as it is more robust and high durability compared to PVC. There are 4 thrusters by using BLDC motors and equipped with ESC for the propulsion system of the underwater ROV as BLDC motor will be more efficient and provided higher speed compared to DC motor and bilge pump. Two horizontal thrusters will be located in the rear of ROV with a vectored configuration to enhanced the turning moment of underwater ROV during operation and the other two thruster will be mounted vertically on the side of the ROV. Arduino Uno will be chosen for the microcontroller of the underwater ROV as it is cost-effective compared to Raspberry Pi. The Bar02 pressure/depth sensor's integrated temperature sensor, appropriate depth range, excellent resolution, compatibility with popular microcontroller systems, and robust design are factors make it an attractive selection for underwater ROVs. Its features make it a versatile and effective option for a wide range of ROV projects, especially those that concentrate on shallow to intermediate depth investigation. The embedded MPU6050 module sensor is considered the best choice while dealing with underwater ROV tasks because it is an all-in-one solution which allows for cost-effective and reliable operations, integrates well with I2C, incorporates an onboard digital motion processor required for sensor fusion, has low power consumption rates and has been tried out in real-world applications. All this makes it a better option than any other when it comes to precision navigating and stabilizing in marine habitats. Besides, the underwater ROV in this project can be submerged in water with depth up to 15m. It will be equipped with waterproof camera, a pair of lights and pressure sensor for monitoring application.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter will provide a detailed explanation of the main idea prototype design and testing procedure while accomplish this project. The research flowchart, prototype design, and experimental setup will be thoroughly discussed. Thus, the method and materials used in the design, as well as proper planning, must be followed to ensure that this project will be success on time and fulfill all the objectives needed.

3.2 Project Flowchart

3.2.1 FYP 1 Flowchart

The diagram in Figure 3.1 illustrates the overall process that will be completed. in the project.

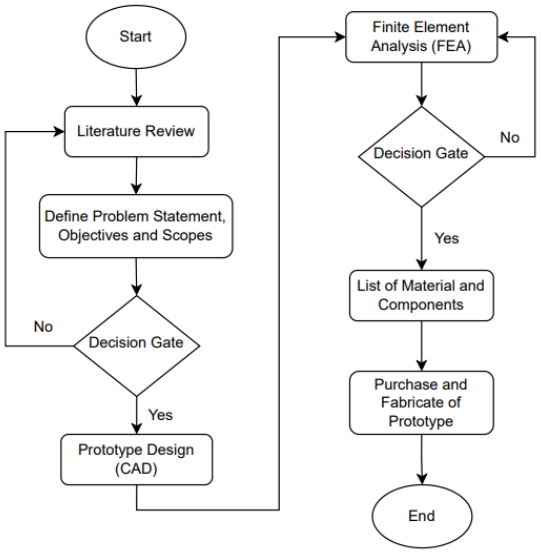


Figure 3-1: FYP 1 Flowchart

3.2.2 FYP 2 Flowchart

The diagram in Figure 3.2 illustrates the overall process that will be going to be continue in the FYP 2.

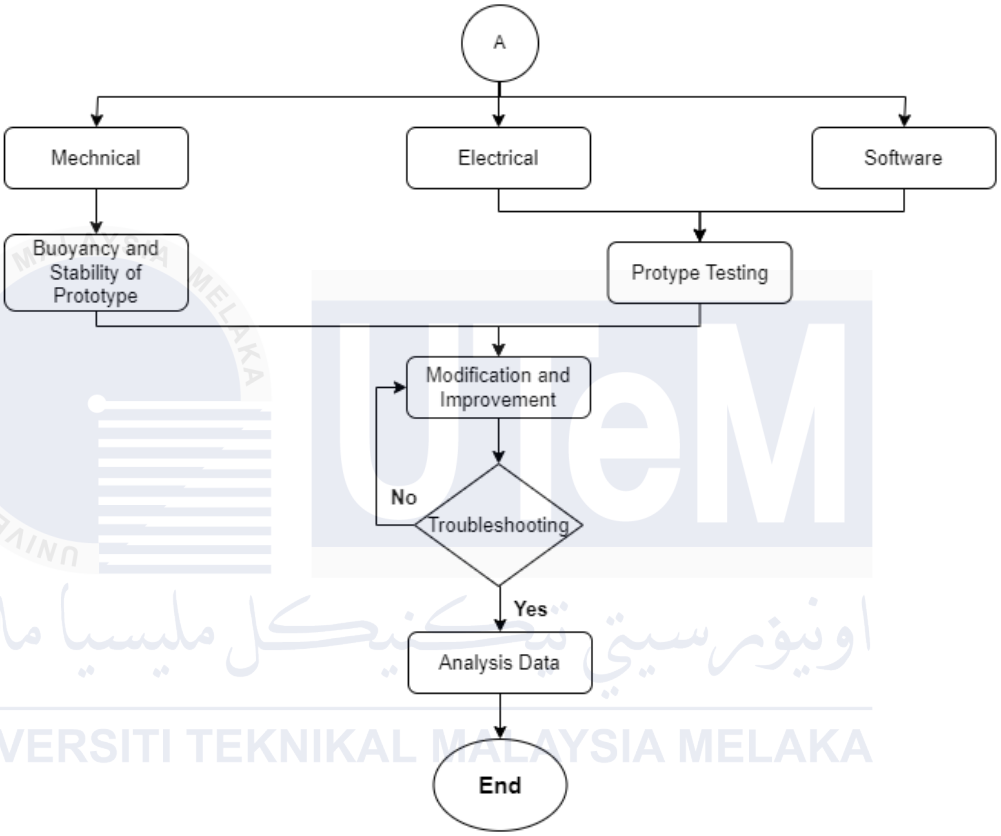


Figure 3-2: FYP 2 Flowchart

3.3 Fusion360 software

Fusion 360 is a commercially available software application that encompasses computer-aided design (CAD), computer-aided manufacturing (CAM), computer-aided engineering (CAE), and printed circuit board (PCB) design. Fusion 360 is known for its integrated approach, combining design, simulation, and manufacturing functionalities in a single platform. Fusion 360 includes generative design features, where designers can specify user constraints and goals and then run algorithms to produce optimal solutions based on such parameters.

Fusion 360, with user-friendly interface allowing convenient modelling and simulation of many features when crafting the prototype for a micro underwater ROV made possible to design important units such as body structure, propulsion system and sensors. Fusion 360 with an objective of facilitating the design process and eliminating unwanted intricacies associated with systems.

Figure 3-3: Autodesk Fusion 360

3.4 Mechanical Design

The mechanical design of the ROV includes the structural framework, buoyancy control, and propulsion system. Selection of structural framework design is crucial, concerning the framework design in order to enhance the maneuver performance in underwater operations of the ROV. The frame body can be act as the backbone of the ROV which used to hold all component parts and is usually made from strong materials such as aluminum or Polyvinyl Chloride (PVC) in consideration for the harsh underwater environment. Some of the essential features of buoyancy control include ballast system or buoyancy tanks that aids in stability and control during maneuvers. The propulsion system involves the utilization of thrusters that are

arranged such that they can facilitate movement in forward, backward, sideward and raise and submerge motions.

3.4.1 Prototype Design Sketching

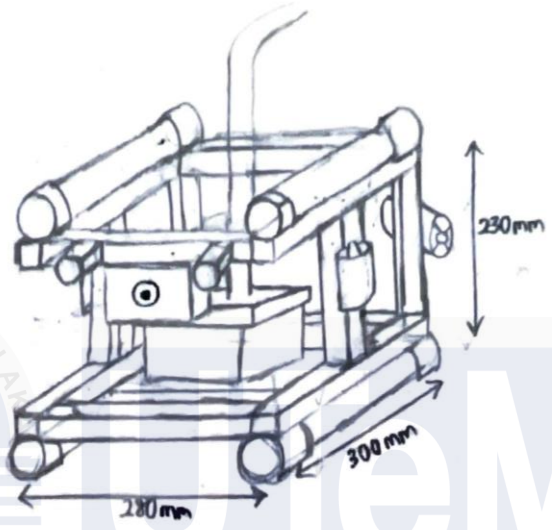


Figure 3-4: Design Sketching of the Prototype Underwater ROV

3.4.2 Prototype Design using Fusion 360 software

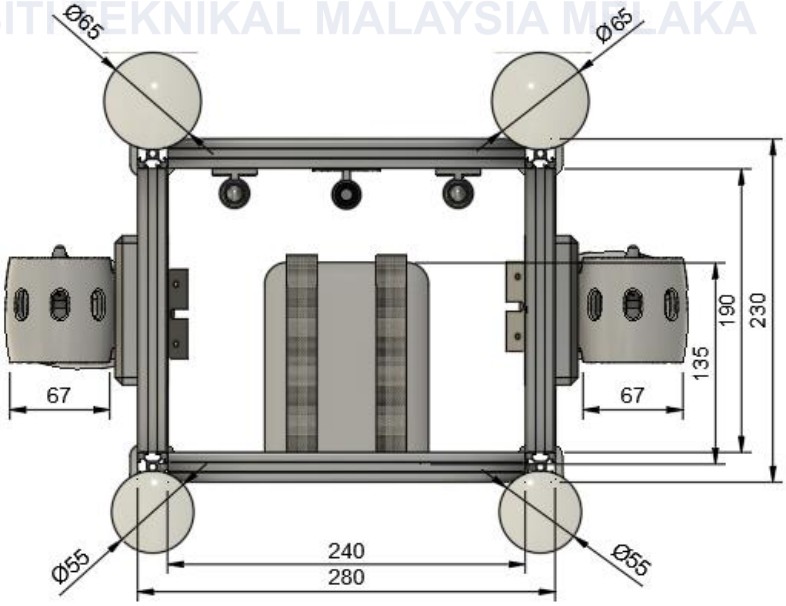


Figure 3-5: Front View of the Prototype Design of Underwater ROV

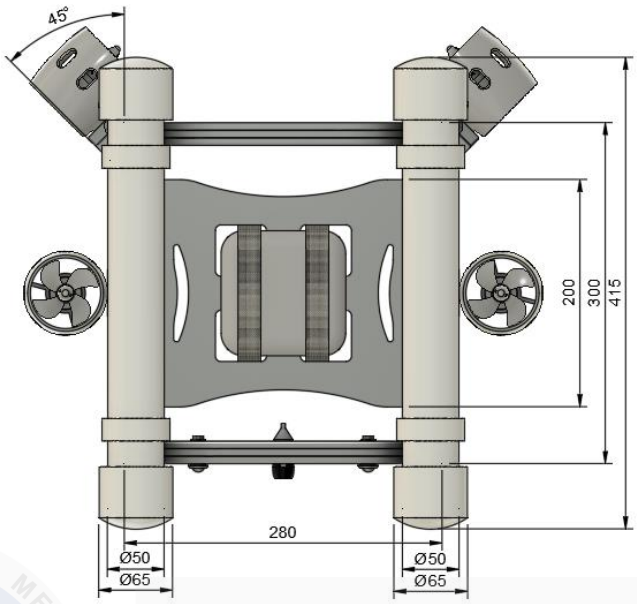


Figure 3-6: Top View of the Prototype Design of Underwater ROV

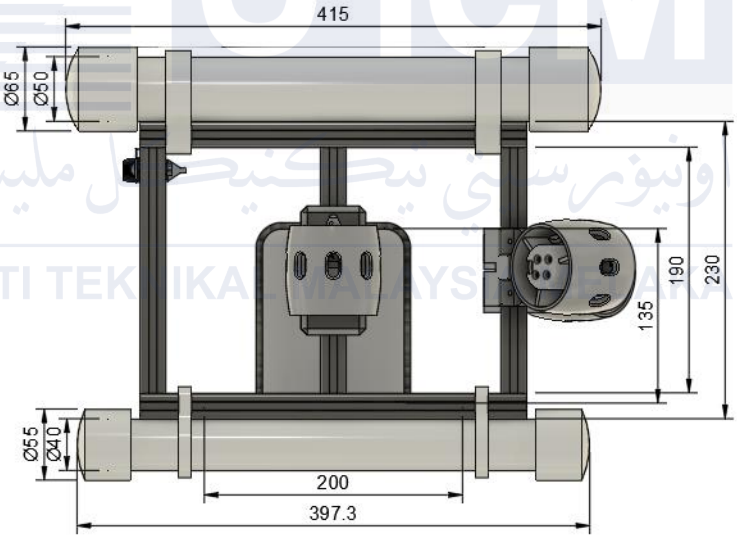


Figure 3-7: Side View of the Prototype Design of Underwater ROV

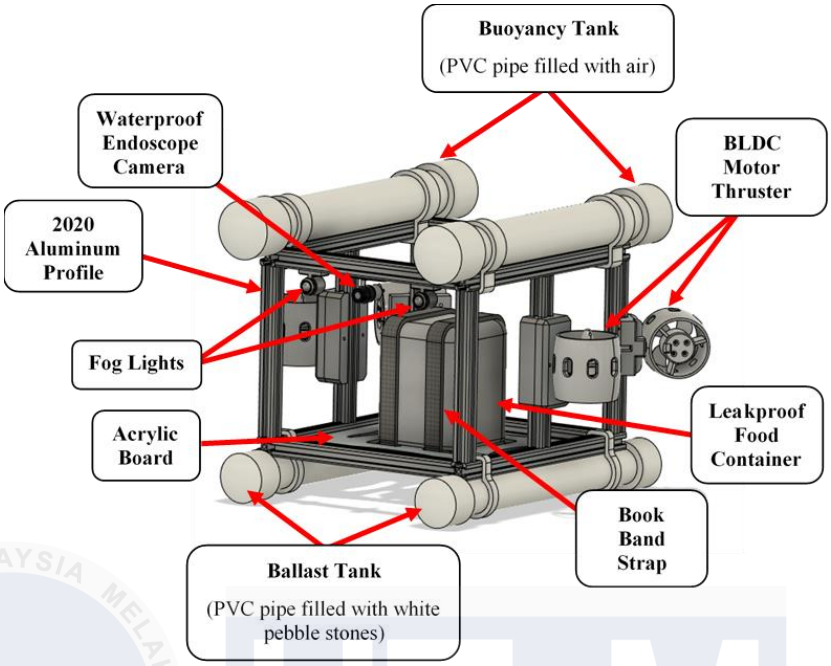


Figure 3-8: Isometric View of the Prototype Design of Underwater ROV

3.4.3 Buoyancy Control

Buoyancy control is crucial in the stability and maneuverability underwater in designing ROV prototype. This is usually done by employing ballast systems or buoyancy tanks to help the ROV prototype in achieving negatively buoyant underwater. In this specific model of the ROV, buoyancy is controlled by PVC pipes, which are mounted on top and bottom sides of the ROV.

Figure 3-9: Upper PVC Pipe (Buoyancy Tank)

Figure 3-9 shows the upper PVC pipes are filled with air which act as a buoyancy tank. The air-filled PVC pipe produces an upthrust buoyancy force to

counteract the weight of the ROV making it float and maintain at a stable position during underwater operation. The positive buoyancy helps in balancing the weight of ROV to avoid sinking uncontrollably.

Figure 3-10: Bottom PVC Pipe (Ballast Tank)

The bottom PVC pipes are drilled with some small holes and filled with white pebble stones that serve as ballast as shown in Figure 3-10. The small holes at the PVC pipes allows air bubbles to escape so that the buoyancy of ROV prototype will not be affected. These ballast tank enable the provision of negative buoyancy which in this case is a necessity as it helps to offer weight to the ROV to counteract the buoyantly forces produced by the top of PVC pipes that filled with air. In addition, the overall weight from the pebbles aids in lowering the center of gravity of the ROV prototype and reduces the probability of the ROV tipping or rolling.

The combination of the air-filled upper PVC pipe and the weighted bottom PVC pipe forms a balanced buoyancy system, ensuring that the ROV prototype remains negatively buoyant during underwater operations. The upward force from the buoyancy tank and the downward force from the ballast provides a stabilizing effect, reducing undesirable tilting or drifting.

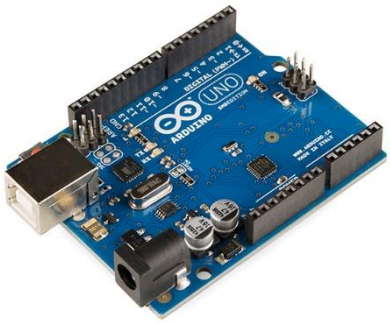
3.5 Electrical Design

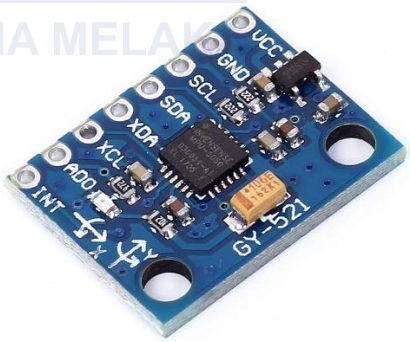
The electrical design of the ROV prototype involves the function of devices and components used in the ROV prototype, circuit connection and flowchart of the ROV prototype control system. The Arduino Uno controls all the operations of ROV including sensors, motor thrusters, and control inputs while the circuit connection guarantees that all the components are correctly powered and connected. The flow chart of the control system reveals the logical process of operation which enhances the execution of the ROV's task.

3.5.1 List of Devices and Components

This section explains the electrical devices and components which are installed in the ROV prototype. From Table 3-1 shows the devices and components used to suffice the needs in design and development of a micro underwater ROV for monitoring applications.

Table 3-1: List of Devices and Components Used in ROV Prototype

Devices/Components	Function	Picture
Arduino Uno	• serves as the main control unit for the ROV • manages the overall operation, executing programmed instructions to control the ROV's movement, depth, orientation and	

	other functionalities.	
Brushless Motor Thrusters	• to maneuver the ROV during underwater operation	
ZMR 30A Bidirectional ESC	• control the speed and direction of the thrusters based on the commands received from the microcontroller	
Bar02 Pressure/Depth Sensor	• to detect real-time depth, pressure and temperature during underwater operation	
MPU6050 Accelerometer and Gyroscope Sensor	• to determine the ROV's orientation and motion dynamics, allowing for stable and controlled movements.	
Waterproof Endoscope Camera	• captures video footage of the underwater environment • provides a visual feed to the operator, allowing for real-	

	time monitoring and navigation	
Fog Lights	• illuminate the area around the ROV in dark underwater environments	
PS2 Controller	• to control the ROV's movements, depth, and lights. • to send commands to the Arduino Uno for maneuvering the ROV	
12V/8AH Sealed Lead Acid Battery	• provide power to all the ROV's components including lights and motor thrusters.	

3.5.2 Schematic Circuit Wiring Connection

All electronic devices and components mentioned previously were connected as shown in Figure 3-11.

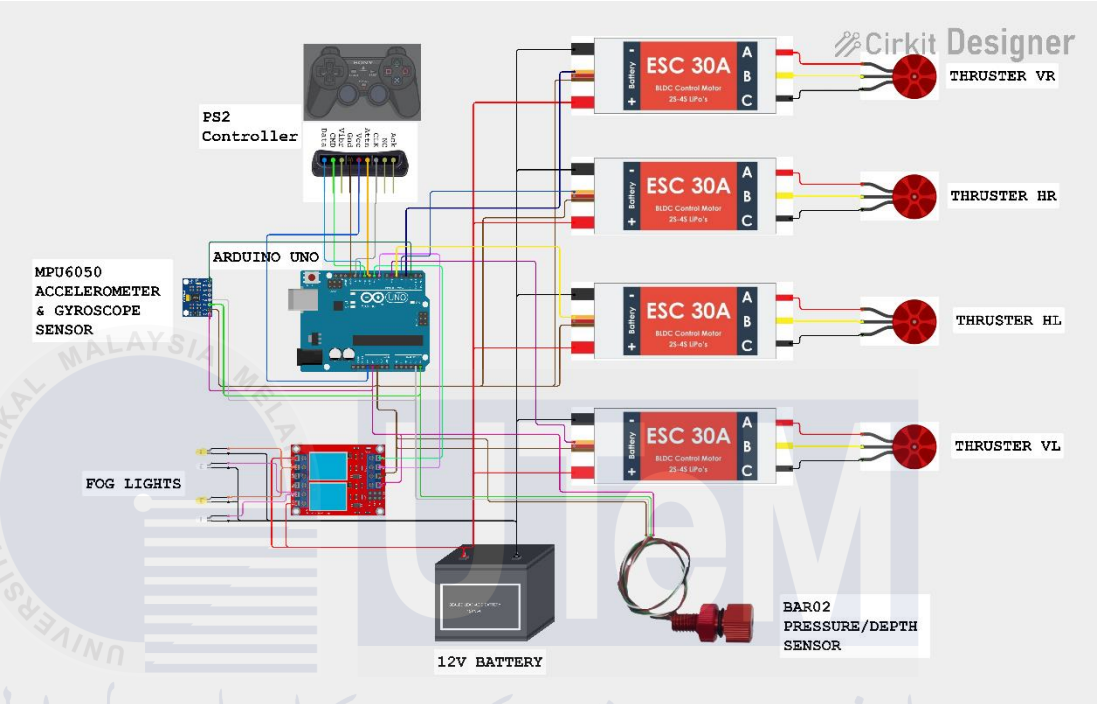


Figure 3-11: Schematic Circuit Wiring Connection Diagram

3.5.3 Schematic Flowchart for ROV Prototype Control System

The detailed description of the ROV prototype control system schematic flowchart is presented in Figure 3-12.

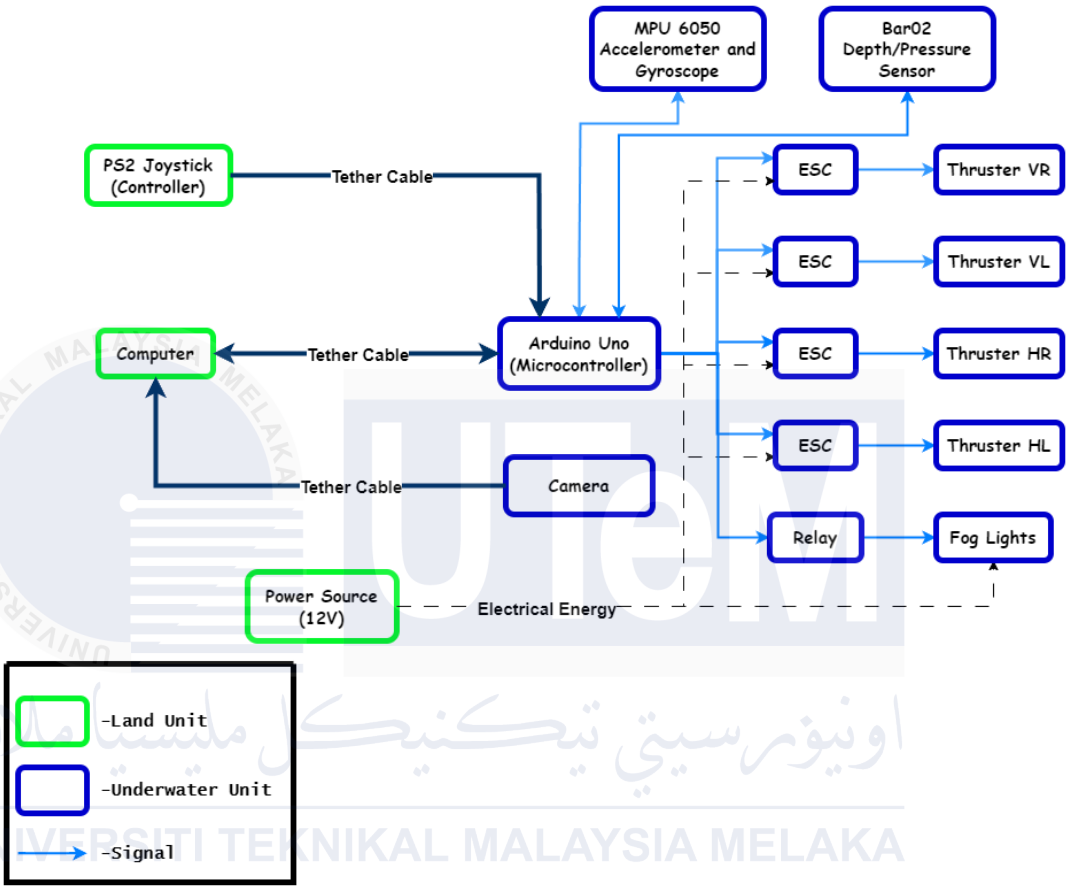


Figure 3-12: Schematic Flowchart Diagram for ROV Prototype Control System

3.6 Software Design

The design of the software involves the control algorithms for the execution of operations, the graphic user interface (GUI) and data analysis management. The control algorithms are responsible for the ROV's maneuverability and balance, as processing input from sensors to make real-time adjustments. The GUI of the ROV enables the operators to observe the operation of the subsea equipment, usually with added features like video streaming and data transmission. Data processing is essential in analyzing data obtained from sensors, making decisions, and storing information for later use in the mission.

3.6.1 Arduino IDE

The software design of the developed ROV prototype entails programming of the control algorithms using Arduino Integrated Development Environment (IDE) as shown in Figure 3-13. Arduino IDE is a user-friendly platform for programming Arduino Uno microcontroller using C/C++ languages that is best suited for real-time control systems and can easily allow users to type and upload instructions to the devices. It also has several libraries and utilities that enable the controllers, sensors and other equipment to be interfaced with the ROV control system.

Figure 3-13: Arduino IDE Software

Control algorithms are the core of the software design. These algorithms involve control of the sensors through data acquisition from the Bar02 pressure/depth sensor and the MPU6050 accelerometer and gyroscope sensor to determine the depth, temperature, position and orientation of the ROV prototype. This raw data is then further processed by the algorithms through filtering, calibration and conversion of measured readings into usable units to provide meaningful information like the current depth, temperature and orientation of the ROV prototype. Functions within the software will also transmit the correct signals to the ZMR 30A bidirectional ESCs and relay to control the brushless motor thrusters and lights respectively. Furthermore, the algorithms will manage input from the PS2 controller, such as the buttons and a joystick that would signal actions such as moving forward, backward, raising, submerging, and turning around. Safety features and error handling mechanisms are integrated to ensure reliable operation such as setting of failsafe boundaries that snap off the motors if some of the critical sensors fail or if the ROV is operated beyond its expected limitations.

There are several phases involved in the process of developing the software. First, the control algorithms are programmed and tested on the Arduino platform within the Arduino IDE, including declarations, definitions of sensors and motors, and the control strategies. The code is then written and uploaded into the Arduino Uno to check on their performance. This phase involves testing to ensure they are performing in the right manner, correcting their faults and improving their efficiency. After basic debugging, the algorithms are put in actual and emulated conditions for further debugging considering various issues that might not have been observed earlier and fine tuning in the environment for which they have been designed. From the results obtained from the testing process, the code is revised to gain efficiency, to accommodate for other features or to increase its reliability. This process is carried out until the control system is well designed to fit its design requirements or until it is fully functional.

In conclusion, the development of the software used to control the ROV prototype includes algorithms to be programmed in the Arduino IDE that consist of sensor acquisition, signal processing, motor operation, input from the operator and safety systems. Editing and recompiling of the code guarantees that the ROV functions optimally in the underwater environment due to the checking and correcting of the program as it is implemented.

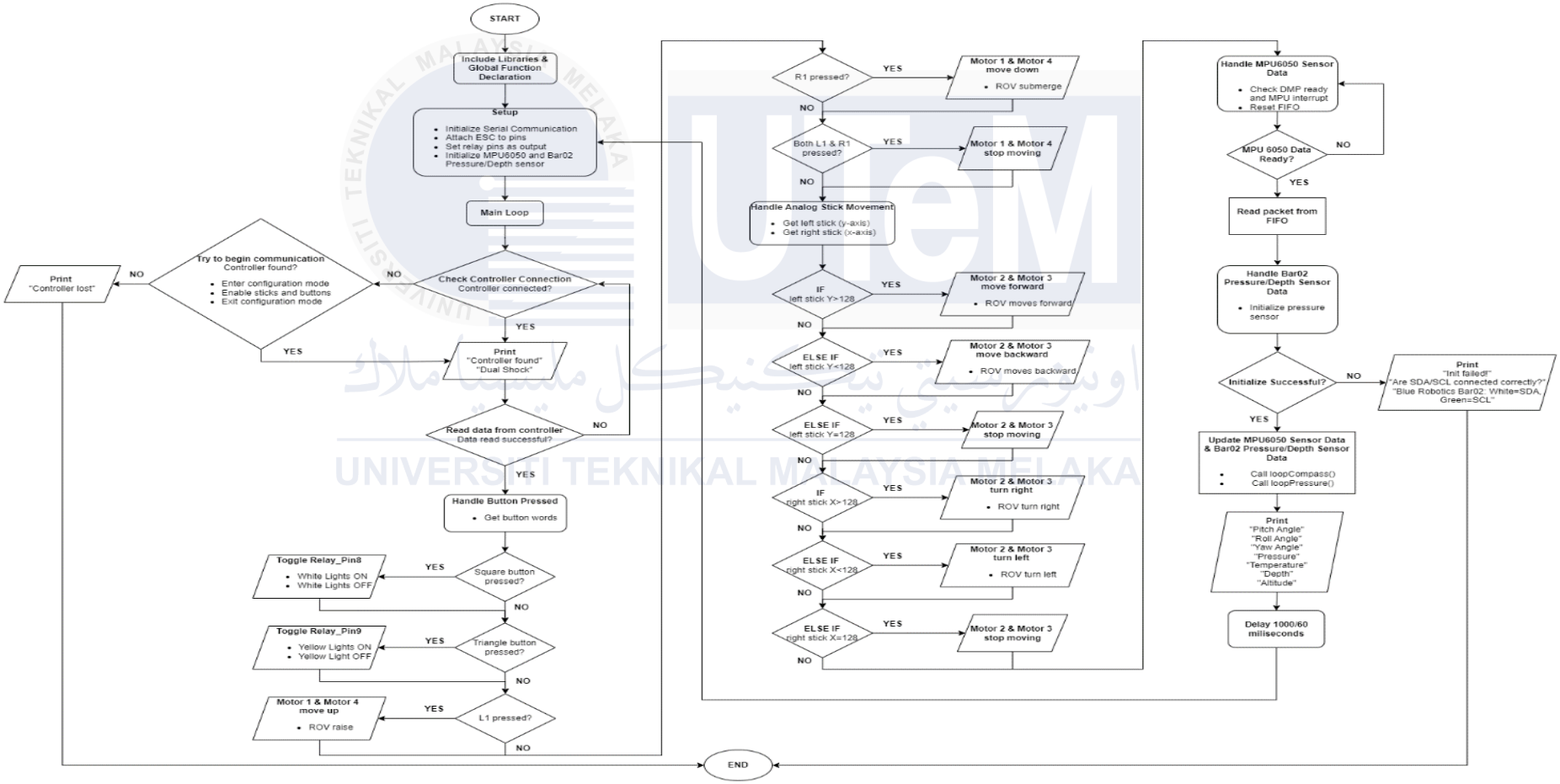


Figure 3-14: Arduino Program Flowchart

3.6.2 Processing IDE

The GUI is designed using the Processing software shown in Figure 3-15. Processing is open-source and free to download. It is useful for developing a GUI for ROV prototype. This study uses processing to communicate data between a microcontroller and an operator. Processing receives input data from PS2 controller and then sends commands to the microcontroller. The microcontroller sends sensor data to the processing, including rotation angle, compass, pressure, depth, temperature and altitude and then display the data to monitor. The operator can observe the ROV prototype's behaviour. Figure 3-16 displays the horizon, compass, rotation angle, depth, temperature, etc. The user can observe the movement and surrounding environment of ROV prototype in real-time.

Figure 3-15: Processing Software

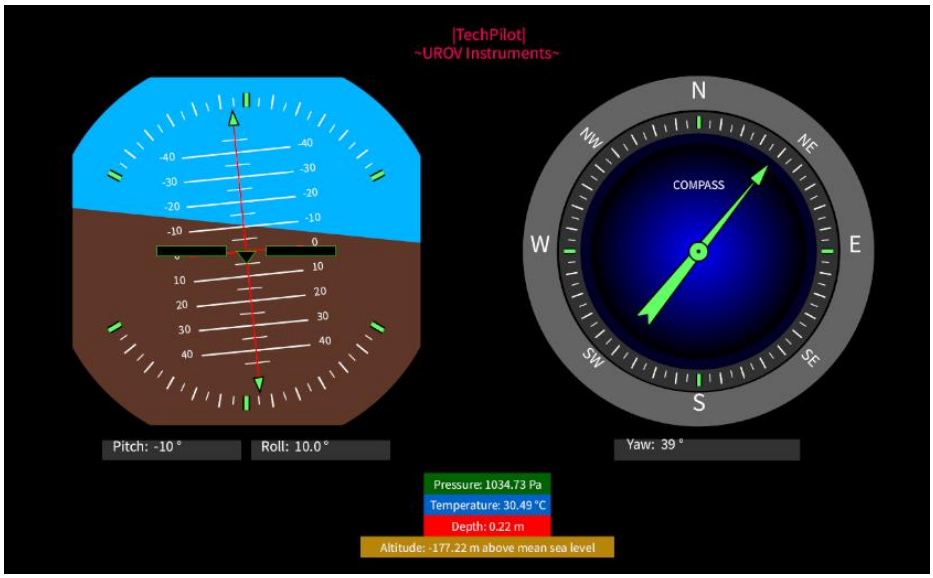


Figure 3-16: GUI Display for ROV Prototype

3.6.3 OBS Studio

OBS Studio is the most efficient open-source video recording and live stream program that can be used to add live video with graphical data. This integration helps in enabling efficient and accurate operations because it provides an all-round view of the surrounding environment, as well as overall condition of the ROV.

Figure 3-17: OBS Studio Software

One major consideration when operating an ROV is the video feed from the camera mounted on the vehicle. A waterproof endoscope camera is fixed on the ROV prototype to provide live video images which are useful in maneuvering the ROV prototype, inspection and implementation of tasks. OBS Studio receives this video feed from the camera directly and ensures that the operator is given a very good view of the feed without blurring or lagging. This configuration makes it possible for the operator to view what the ROV is viewing, helping to maneuver the vehicle and accomplish complex tasks in the water. The high flexibility in OBS Studio in managing multiple video inputs guarantees that the camera feed is transmitted frequently, and the quality is optimal for precise operations.

Processing, a sketchbook and coding language within the context of the visual arts, can be used to design a GUI specifically for the underwater ROV. This GUI displays orientation, depth, temperature, pressure and related data that are crucial for navigation. This GUI is integrated alongside the real-time video feed that is captured by the camera in the ROV prototype and OBS Studio. Thereby, using the Processing window as a source, OBS enables the operator to view both the camera feed and the data about the ROV underwater operations at the same time. This integration improves

the situation awareness and the operator is able to make decisions based on visual and data feedback.

There are several benefits of employing OBS Studio to merge the ROV camera feed with the Processing GUI. The GUI can be displayed on the same screen together with the live video feed, which means the operator does not need multiple screens to do their work. The software has different layout settings which allow the resizing and repositioning of the video and graphical user interface to suit the operator's input. Moreover, OBS Studio offers functionalities for recording processes for future review and broadcasting the video stream and GUI to other locations, which makes it possible to collaborate and monitor the process in real-time. Additionally, OBS Studio provides capabilities for recording operations for later analysis and streaming the video feed and GUI to remote locations, facilitating real-time collaboration and supervision. Overall, the simplicity of the OBS Studio is its main feature so that operators, no matter how technically competent or incompetent they are, can quickly switch to the necessary points and set up the display.

The combination of OBS Studio to show the ROV's camera feed and the Processing GUI brings an efficient, reliable, and friendly UX/UI to control the operations underwater. This setup improves the operator real-time control and manipulation of the ROV since important operational information is displayed at the operator's fingertips in a single intuitive interface. The functions of OBS Studio like recording and streaming enhance the facility of using the ROV system in real-time as well as in post operation use.

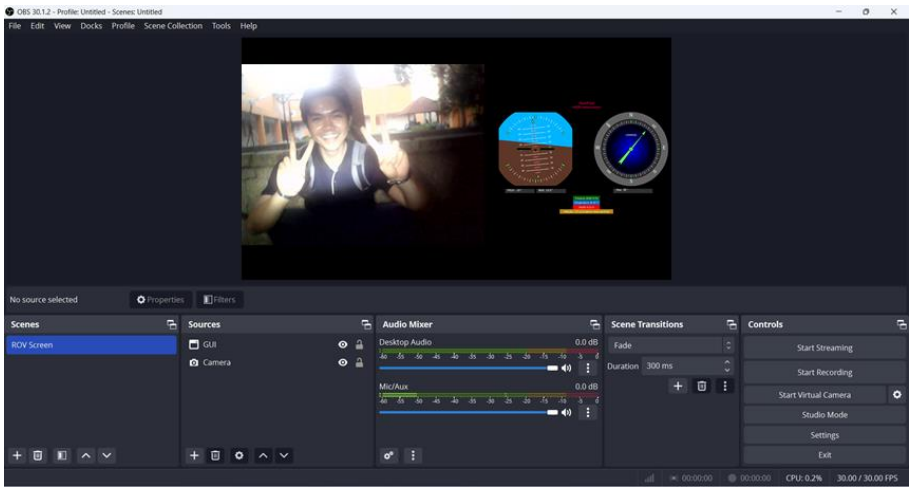


Figure 3-18: ROV Screen Display in OBS Studio

3.7 Finite Element Analysis (FEA)

Finite Element Analysis (FEA) is a computational tool designed to identify the behavior of physical systems or components under certain conditions or loads. It is a numerical technique that has been used in the simulation and analysis of structural or component behavior due to diverse load input situations as well as boundary conditions.

There are some main objectives of FEA include:

1. To predict the response and behavior of a physical system or component when subjected to specific conditions, loads or external forces.
2. To test the designs and materials to reduce costs and improve performance before the design of a component or structure is built.
3. To determine potential regions of stress concentration, failure, or deformation that may occurs in a structure or component.
4. To optimize the design structure of component in order to reduce weight, improve efficiency and reliability.
5. To evaluate the performance of a design under various loads, temperatures, and other environmental factors.

In this experiment, FEA will be used to determine the behaviour of design prototype as shown in Figure 3.9 under a applied working pressure when submerged at a desired depth in water.

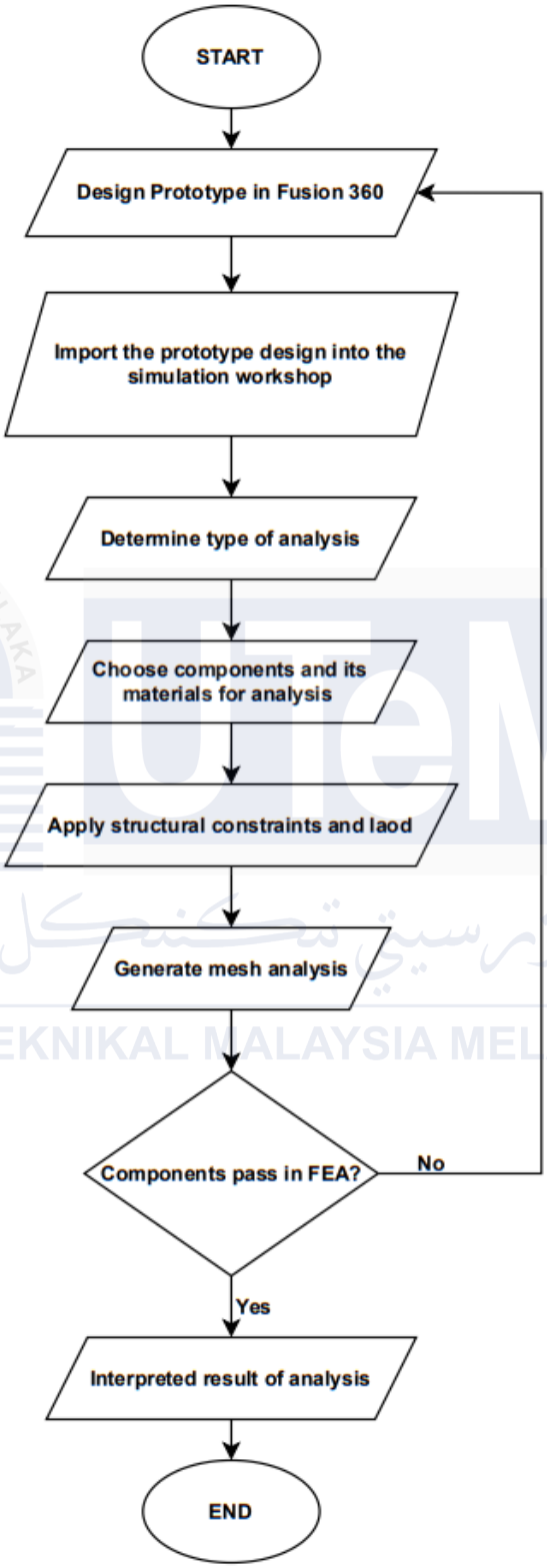


Figure 3-19: Flowchart of Finite Element Analysis

3.8 Water Pressure Estimation

The water pressure exerted on the ROV at a certain depth can be calculated by using Pascal's Law. This principle states that a pressure change at any point in a confined incompressible fluid is transmitted equally in all directions. The equation to calculate pressure exerted at certain depth by using Equation 2-2.

Assume the underwater ROV is submerged into water with depth of 15m, $\rho = 1000\text{kg}m^{-3}$

$$\mathbf{P = \rho gh}$$
$$= (1000\text{kg}m^{-3})(9.81ms^{-2})(15m)$$
$$= 147.15kPa$$
$$\approx 150kPa$$

Thus, the water pressure exerted on the underwater ROV is about 150kPa.

3.9 Experiment

Several experiments have been devised to accomplish the project's objectives. Experiment 1 will be utilized to achieve objective 1, which is to design a micro underwater ROV for inspection using Fusion 360 platform. Experiment 2 will be discussed in this chapter. Experiments 3 is to determine the state of ROV while Experiment 4 is to analyze the propulsion system performance of the ROV. These both experiments will be done after prototype is completed assembled during FYP 2.

Table 3-2: List of Experiment

Experiment	Objective 1	Objective 2	Objective 3
Experiment 1: Design prototype of micro underwater ROV	★		

Experiment 2: Finite Element Analysis	★		
Experiment 3: Buoyancy of underwater ROV		★	
Experiment 4: Underwater Operation			★

3.9.1 Experiment Setup

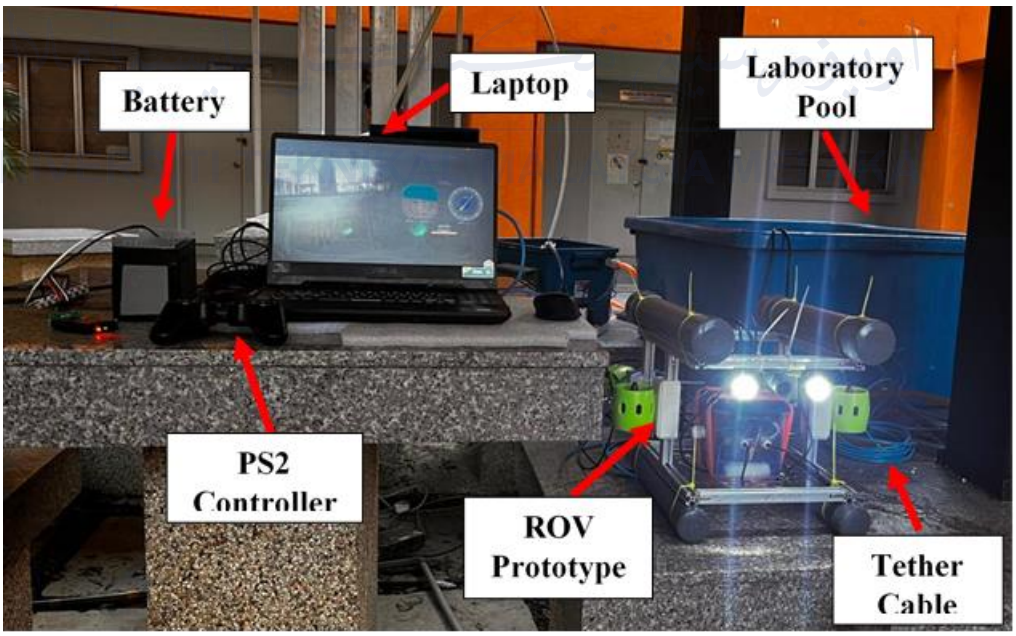


Figure 3-20: Experiment Setup

3.9.2 Experiment 3: Buoyancy of ROV

Objective:

To determine the state of ROV buoyancy

Hypothesis:

a) ROV exist positive buoyancy when ROV is float at surface water,
b) ROV exist neutral buoyancy when ROV is float in water,
c) ROV exist negatively buoyancy when ROV is sinking into water

Apparatus:

a) Massed white pebble stones
b) ROV Prototype
c) Lab Pool
d) Measuring Tape
e) Weighing Machine

Procedure:

1. The weight for ROV is measured without inserted any white pebble stones
2. The body frame of ROV was tape to get a line.
3. ROV is put in the lab pool for 3 minutes. The ROV position was observed and recorded.
4. The weight of ROV is increased by inserting massed white pebble stones into PVC pipe and the ROV position is observed.
5. The buoyancy types of ROV is observed either negatively buoyant, neutral buoyant or positively buoyant when the ROV lay float on the surface of water.
6. The massed white pebble stone is adjusted to make the ROV position until it reached above 90% of negatively buoyant. The observation is then being recorded.

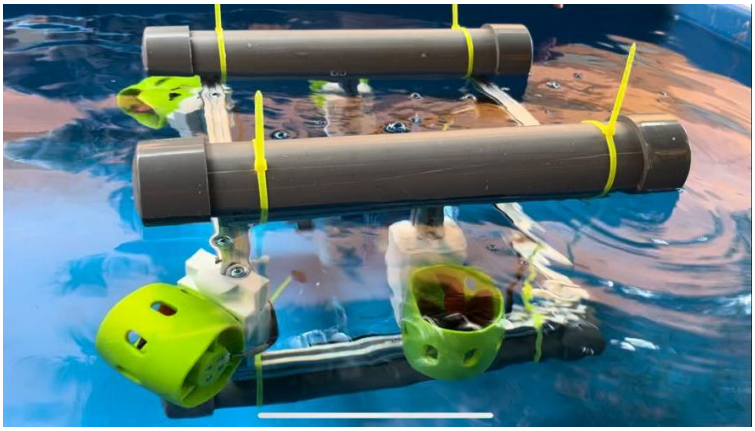


Figure 3-21: Underwater ROV Prototype in Lab Pool

3.9.3 Experiment 4: Underwater Operation

There are several tests been done to analyse the propulsion system performance of the ROV. These experiments allow to perform the ROV manoeuvrability underwater.

3.9.3.1 Experiment 4.1: Moving Forward and Reverse

Objective:

To determine the speed for ROV for moving forward and reverse

Hypothesis:

Velocity of ROV is increase when accelerate and vice versa while decelerate

Apparatus:

a) ROV Prototype
b) ROV Controller
c) Lab Pool
d) Stopwatch
e) Measuring Tape

Procedure:

1. ROV prototype is prepared and make sure the condition is ready for launch.
2. A point is measured and set at length of 2.4 meter in the lab pool.
3. The ROV prototype is placed inside the lab pool and make sure it is in correct buoyancy (negative buoyancy).
4. The power of ROV prototype is switched on.
5. The horizontal thruster of the ROV is controlled using PS2 joystick controller to move the ROV prototype forward.
6. The data for forward movement is recorded.
7. The experiment is repeated three times to ensure the data obtained is accuracy and consistency.
8. The velocity and acceleration can be calculated by the formula as shown below:

$$\boldsymbol{Velocity} = \frac{\boldsymbol{Distance}\ (\boldsymbol{m})}{\boldsymbol{Average\ Time\ Taken}\ (\boldsymbol{s})} \quad \text{3-1}$$

$$\boldsymbol{Acceleration} = \frac{\boldsymbol{Velocity}\ (\frac{\boldsymbol{m}}{\boldsymbol{s}})}{\boldsymbol{Average\ Time\ Taken}\ (\boldsymbol{s})} \quad \text{3-2}$$

9. Step 3 to 7 is repeated for reverse movement.

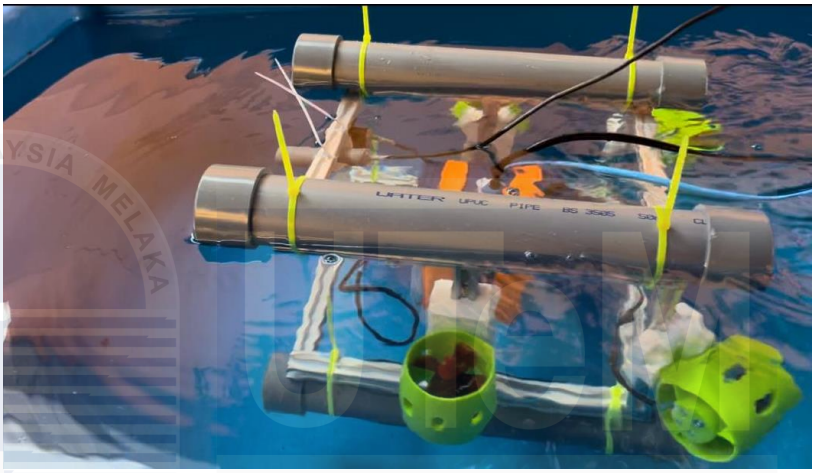


Figure 3-22: Underwater ROV Prototype Move Forward and Backward Process

3.9.3.2 Experiment 4.2: Turn Right and Left

Objective:

To determine the speed for ROV for turning right and left

Hypothesis:

Angular velocity of ROV turning is increased when it begins to turn and the angular velocity decreased when it reached 180º to stop.

Apparatus:

a) ROV Prototype
b) ROV Controller
c) Lab Pool
d) Stopwatch

Procedure:

1. ROV prototype is prepared and make sure the condition is ready for launch
2. A point is measured and set at depth of 1.5 meter in the lab pool.
3. The ROV prototype is placed inside the lab pool and make sure it is in correct buoyancy (negative buoyancy).
4. The power of ROV prototype is switched on.
5. The horizontal thruster of the ROV is controlled using PS2 joystick controller to the turn ROV prototype to the right.
6. The data for turning right condition of ROV until 180º is recorded.
7. The experiment is repeated three times to ensure the data obtained is accuracy and consistency.
8. The angular velocity and angular acceleration can be calculated by the formula as shown below:

$$\boldsymbol{Angular\ Velocity} = \frac{\boldsymbol{Angle\ Rotation}\ (\boldsymbol{rad})}{\boldsymbol{Average\ Time\ Taken}\ (\boldsymbol{s})} \qquad 3\text{-}3$$

$$\boldsymbol{Angular\ Acceleration} = \frac{\boldsymbol{Angular\ Velocity}\ (\frac{\boldsymbol{rad}}{\boldsymbol{s}})}{\boldsymbol{Average\ Time\ Taken}\ (\boldsymbol{s})} \qquad 3\text{-}4$$

9. Step 3 to 7 is repeated turning left.

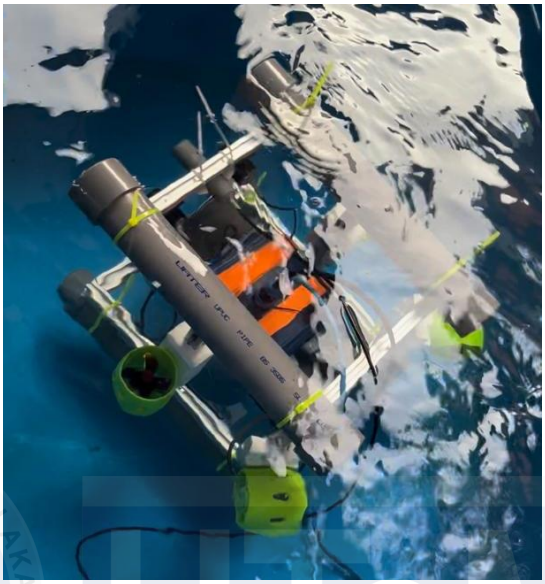


Figure 3-23: Underwater ROV Prototype Turn Right and Left Process

3.9.4 Experiment 4.3: Raise and Submerge

Objective:

To determine the speed for ROV for raise and submerge

Hypothesis:

Speed of ROV turning is increased with time when it starts to submerge and raise

Apparatus:

a) ROV Prototype
b) ROV Controller
c) Lab Pool
d) Stopwatch
e) Massed iron column

Procedure:

1. ROV prototype is prepared and make sure the condition is ready for launch
2. A point is measured and set at depth of 0.45 meter in the lab pool.
3. The ROV prototype is placed inside the lab pool and make sure it is in correct buoyancy (negative buoyancy).
4. The power of ROV prototype is switched on.
5. The vertical thruster of the ROV is controlled using PS2 joystick controller to the ROV prototype raise on the surface of water.
6. The data for the ROV prototype to raise on the surface of water is recorded.
7. The experiment is repeated three times to ensure the data obtained is accuracy and consistency.
8. The velocity and acceleration can be calculated by the formula as shown below:

$$\boldsymbol{Velocity} = \frac{\boldsymbol{Distance}\ (\boldsymbol{m})}{\boldsymbol{Average\ Time\ Taken}\ (\boldsymbol{s})} \quad \text{3-5}$$

$$\boldsymbol{Acceleration} = \frac{\boldsymbol{Velocity}\ (\frac{\boldsymbol{m}}{\boldsymbol{s}})}{\boldsymbol{Average\ Time\ Taken}\ (\boldsymbol{s})} \quad \text{3-6}$$

9. Step 3 to 7 is repeated by submerging the underwater ROV.

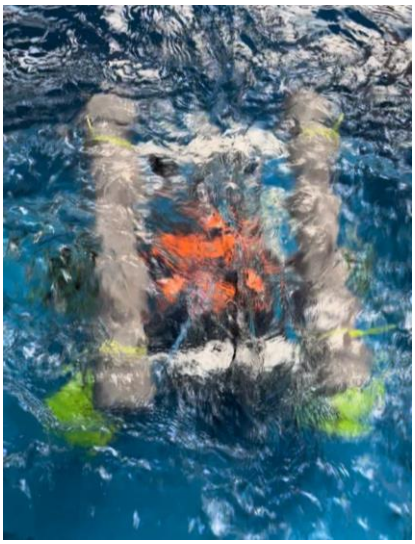


Figure 3-24: Underwater ROV Prototype Raise and Submerge Process

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

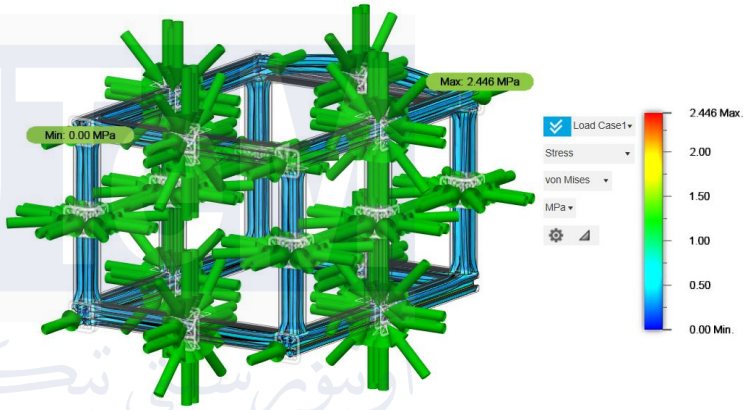
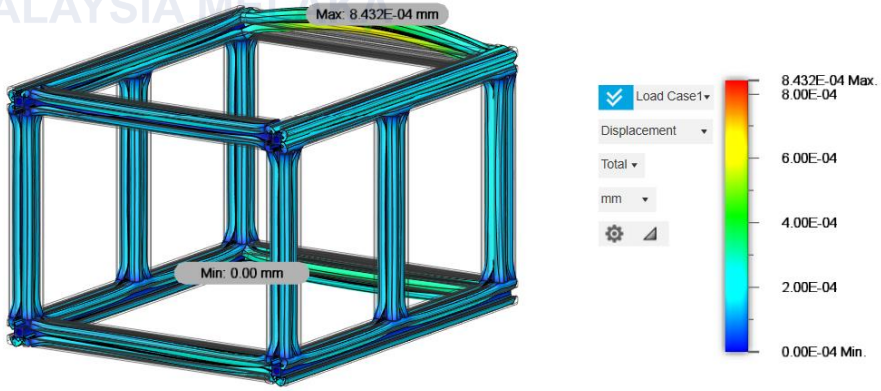
In this chapter, we will evaluate the preliminary result of finite element analysis of the design prototype of the micro underwater ROV using Fusion 360 and overall construction of the ROV prototype. After that, we will discuss the result of design and development of the micro underwater ROV prototype for monitoring applications by conducting experiments to test its performance in underwater conditions. The procedure includes deploying the ROV prototype in underwater environment, observing its buoyancy, maneuverability, stability, and data collection capabilities. The results will be recorded and analyzed to make necessary adjustments to improve the system's performance, and the experiment will be repeated periodically to ensure the ROV prototype operates efficiently in real-world conditions.

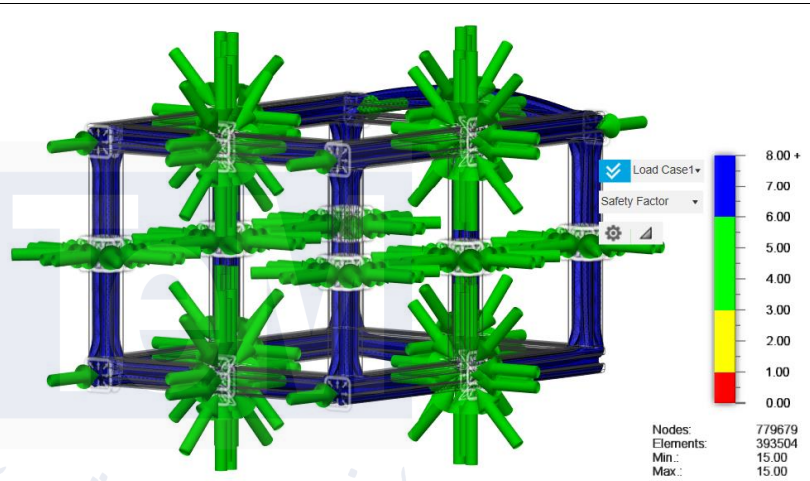
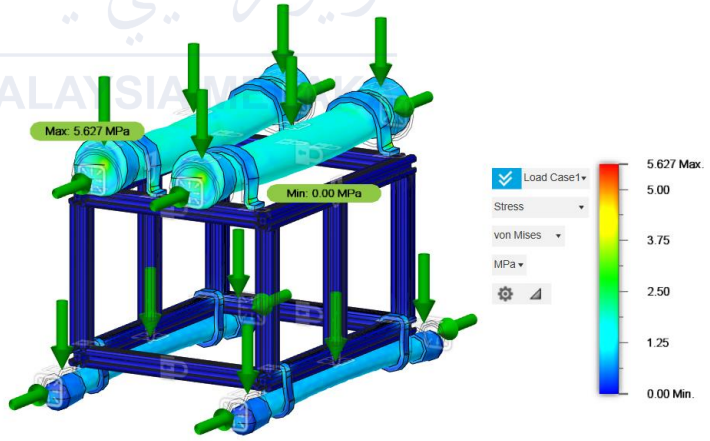
4.2 Preliminary Result of Finite Element Analysis

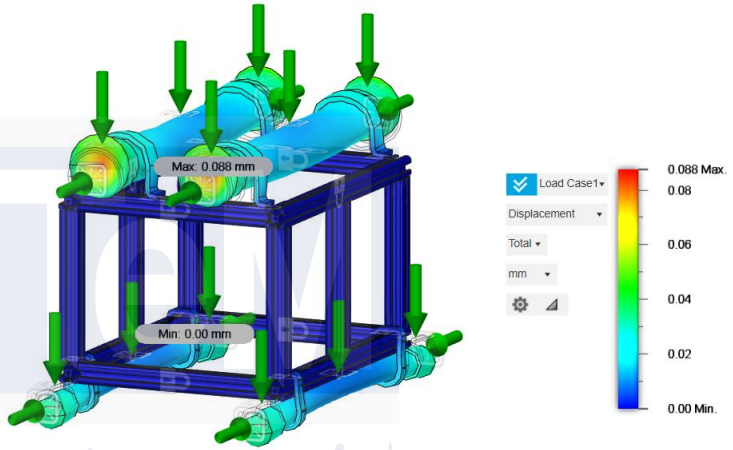
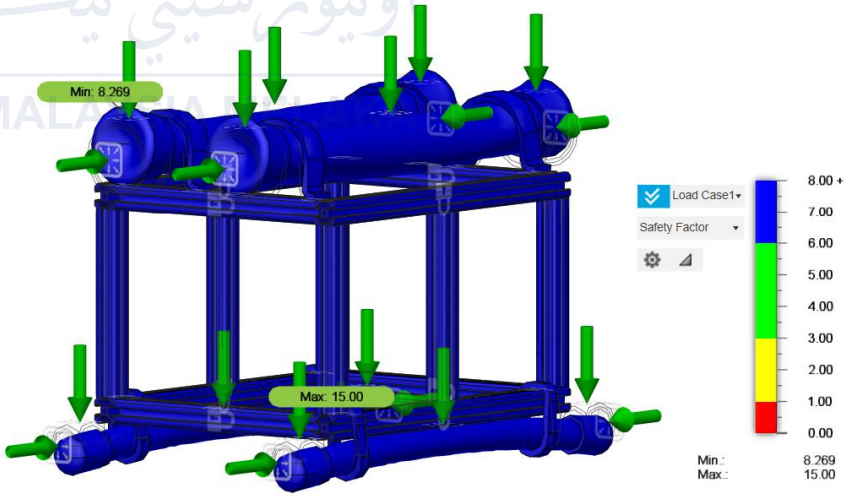
In this chapter, the preliminary result will be discussed the FEA for the design prototype of the micro underwater ROV. FEA enables the simulation of water pressure on the ROV's structure by analyzing how the components, such as the buoyancy tank and frame, respond to the static pressure exerted by water at required depths. The static stress study, safety factor and displacement of deformation analysis for the essential components of the design prototype of the ROV will be conducted.

2020 aluminum profile was chosen as the material for the underwater ROV's frame. It has a yield strength of 240 MPa. If the greatest pressure applied to the component is less than its yield strength of 240 MPa, the prototype design component is safe and durable. Furthermore, if the number for the safety analysis factor is much greater than one, it indicates that the component is resistant to failure. This analysis spectrum resembles a rainbow, with blue being the lowest value and red being the highest. Based on the Equation 2-2, the pressure exerted on the underwater ROV is about 150kPa when submerged into water at depth of 15m.

Table 4-1: Finite Element Analysis of the Component of ROV

Component	Type of Analysis	Result	Figure
Frame	Stress Analysis	Applied pressure = 150 kPa Max. stress value: 2.446 MPa Min. stress value: 0 MPa	
	Displacement of Deformation Analysis	Max. displacement 8.432×10^{-4} mm Min.displacement: 0 mm	

	Safety Factor Analysis	Max safety factor: 15	
Frame with Buoyancy Tank and Ballast Tank	Stress Analysis	Applied pressure = 150 kPa Max. stress value: 5.627 MPa Min. stress value: 0 MPa	

	Displacement of Deformation Analysis	Max. displacement: 0.088 mm Min.displacement: 0 mm	
	Safety Factor Analysis	Max safety factor: 8.27	

In summary, the finite element analysis results show that the forces applied to the material do not exceed its yield strength, indicating that the material can withstand the forces without deformation or failure. Furthermore, the factor of safety analysis revealed that all of the materials utilised in the design have a factor of safety greater than one, suggesting that they can sustain the applied forces without breaking or causing damage. This is a good conclusion since it confirms that the materials used in the design are appropriate for the intended use and will operate as expected under the given conditions.

4.3 Mechanical Prototype Design

The mechanical aspect plays a vital role in developing an underwater ROV with key considerations including size, stability, material, and buoyancy to ensure efficient underwater performance. The ROV prototype design was initially designed by drawing a sketch first as shown in Figure 3-8 with dimensions of frame 280mm(L)×300mm(W)×230mm(H). This preliminary design was then refined using Fusion 360 software. The main purpose of the frame is to give support to the water-proof enclosure, thrusters and any trimming weights. Aluminum will be selected for the frame material of ROV due to its robustness and toughness compared to PVC pipes and ease of construction. The ROV prototype body structure is shown as Figure 4-1.

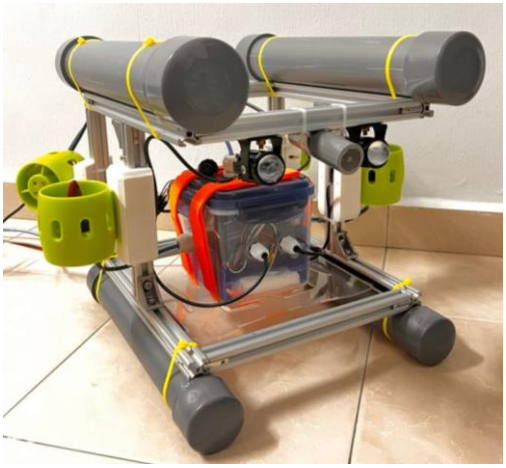


Figure 4-1: Isometric View of ROV Prototype

Figure 4-2: Top View of ROV Prototype

Figure 4-3: Front View of ROV Prototype

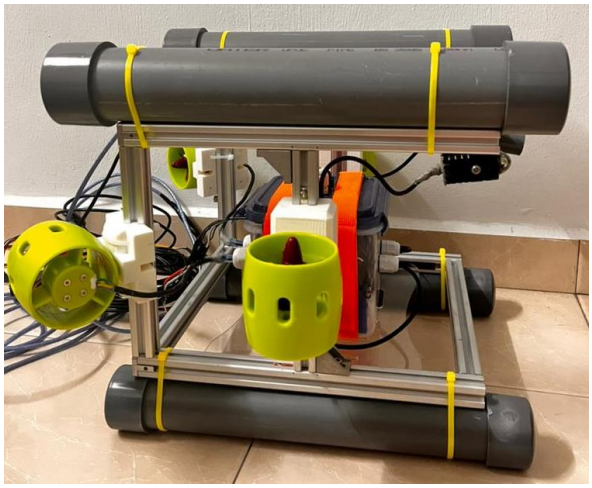


Figure 4-4: Side View of ROV Prototype

Figure 4-5 shows the leakproof container has been chosen to be used as the pressure hull of the ROV prototype due to its waterproof, accessible, modifiable and uncompressible design structure. All electronics components and wire will be stored inside the pressure hull. This type of food container used silicone ring to hold the cover and body tight by interlocking at 4 sides the container waterproof. Few holes have been drilled at the food container for the thrusters wires and tethered cable wires. The nylon plastic IP 68 waterproof cable glands will be installed to the holes on the container allowing the wire cables pass through while protecting the electronics components from water invasion. The most challenging aspect of this part is how to keep water out from this container at all time. Thus, a highly absorbent pads and several packs of silica gel will be placed inside the container. These materials will assist absorb water that might enter and offering an extra layer of protection to the electronic components. This technique seeks to keep the pressure hull dry, which improves the reliability and long-term of the ROV's internal systems.

Figure 4-5: Leakproof Food Container (Pressure Hull)

4.4 Experiment Results

4.4.1 Buoyancy of ROV

Total height of ROV prototype = 335mm

$$\textit{Percentage of ROV prototype immersed} = \frac{\textit{Height of ROV prototype immersed}}{\textit{Total height of ROV body}} \times 100\% \quad \text{4-1}$$

Mass of White Pebble Stones (g)	First Trial (Negative buoyancy)		Second Trial (Negative buoyancy)	
	Height of ROV prototype immersed (mm)	Percentage of ROV body immersed (%)	Height of body immersed (mm)	Percentage of body immersed (%)
0	278	82.99	278	82.99
200	284	84.78	288	85.97
400	293	87.46	297	88.66
600	301	89.85	306	91.34
800	310	92.54	315	94.03

In this experiment, achieving above 90% negative buoyancy is crucial for the ROV to perform effective submerge and raise operations. To accomplish this, additional external load using white pebble stones is required. It was determined 800g of white pebble stones is filled into bottom PVC pipe to achieve above 90% negative buoyancy. In the first trial, attaching only 200g of white pebble stones resulted in the ROV achieving 84.78% negative buoyancy. As more white pebble stones are added, the percentage of negative buoyancy increased until it reached the target above 90%. The second trial yielded similar results, indicating consistency and reliability in the findings. Therefore, it can be concluded that 800g of white pebble stones are necessary to achieve above 90% negative buoyancy for optimal ROV performance as shown in Figure 4-6.

Figure 4-6: ROV Prototype Achieve Above 90% Negatively Buoyant

4.4.2 Underwater Operation

4.4.2.1 Moving Forward and Reverse

To manoeuvre the ROV in horizontal movement, two horizontal thrusters at the back of the ROV prototype were used simultaneously. Both propellers of the horizontal thrusters must be turn clockwise to move the ROV forward and vice versa for moving backward. Therefore, the speed of ROV moves forward and backward will be determined.

Table 4-2: Forward Test Data

Distance (m)	Time Taken (s)			
	Test 1	Test 2	Test 3	Average
0	0	0	0	0
0.2	0.5	0.83	0.96	0.7633
0.4	1.16	1.63	1.88	1.5567
0.6	1.88	2.52	2.74	2.3800
0.8	2.54	3.43	3.67	3.2133
1	2.97	4.36	4.53	3.9533
1.2	3.21	5.16	5.29	4.5533
1.4	3.43	5.73	5.78	4.9800
1.6	3.95	6.29	6.21	5.4833
1.8	4.56	6.92	6.54	6.0067
2	4.98	7.59	6.91	6.4933
2.2	5.1	7.81	7.53	6.8133
2.4	5.21	8.17	7.83	7.0700

Table 4-3: Forward Performance of ROV

Distance (m)	Average Time Taken (s)	Velocity (m/s)	Acceleration (m/s²)
0	0	0	0
0.2	0.7633	0.2620	0.3432
0.4	1.5567	0.2570	0.1651
0.6	2.3800	0.2521	0.1059
0.8	3.2133	0.2490	0.0775
1	3.9533	0.2530	0.0640
1.2	4.5533	0.2635	0.0579
1.4	4.9800	0.2811	0.0565
1.6	5.4833	0.2918	0.0532
1.8	6.0067	0.2997	0.0499
2	6.4933	0.3080	0.0474
2.2	6.8133	0.3229	0.0474
2.4	7.0700	0.3395	0.0480

Table 4-4: Backward Test Data

Distance (m)	Time Taken (s)			
	Test 1	Test 2	Test 3	Average
0	0	0	0	0
0.2	0.54	0.84	0.95	0.7767
0.4	1.26	1.75	1.86	1.6233
0.6	2.13	2.69	2.87	2.5633
0.8	2.96	3.47	3.98	3.4700
1	3.47	4.12	4.88	4.1567
1.2	3.83	4.76	5.62	4.7367
1.4	4.33	5.17	6.31	5.2700
1.6	4.55	5.82	6.87	5.7467
1.8	4.78	6.25	7.31	6.1133
2	4.98	6.98	7.76	6.5733
2.2	5.07	7.21	8.12	6.8000
2.4	5.28	7.3	8.47	7.0167

Table 4-5: Backward Performance of ROV

Distance (m)	Average Time Taken (s)	Velocity (m/s)	Acceleration (m/s²)
0	0	0	0
0.2	0.7767	0.2575	0.3316
0.4	1.6233	0.2464	0.1518
0.6	2.5633	0.2341	0.0913
0.8	3.4700	0.2305	0.0664
1	4.1567	0.2406	0.0579
1.2	4.7367	0.2533	0.0535
1.4	5.2700	0.2657	0.0504
1.6	5.7467	0.2784	0.0484
1.8	6.1133	0.2944	0.0482
2	6.5733	0.3043	0.0463
2.2	6.8000	0.3235	0.0476
2.4	7.0167	0.3420	0.0487

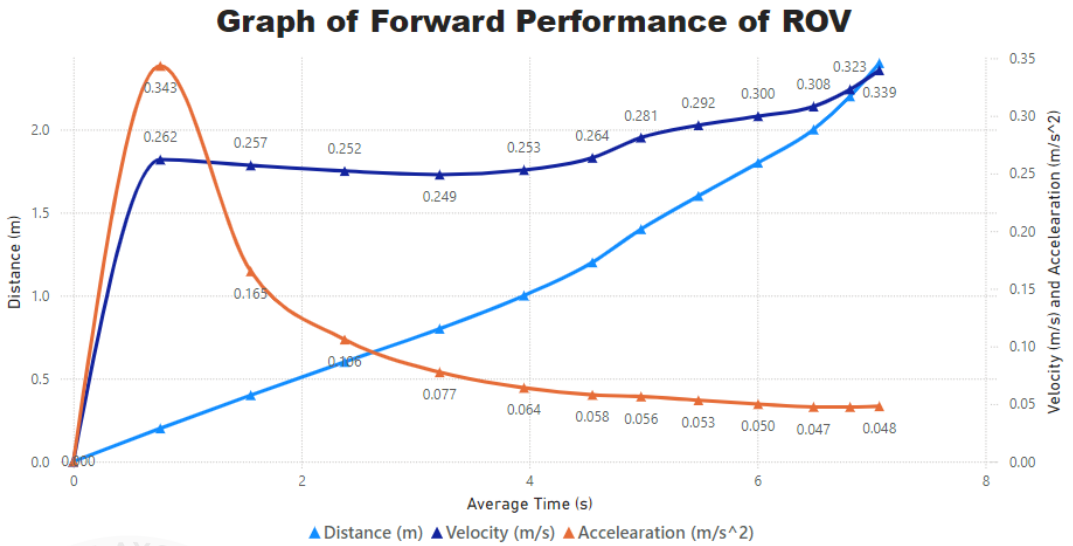


Figure 4-7: Graph of Forward Performance of ROV

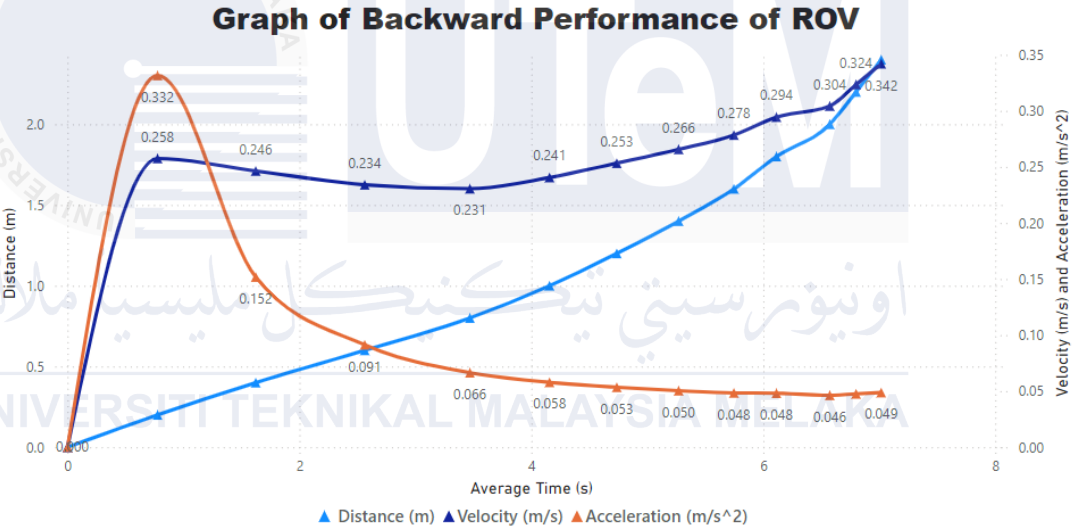


Figure 4-8: Graph of Backward Performance of ROV

From the results obtained while testing the performance of the ROV, it is observed that the average time taken when moving forward and backward is directly proportional to distance. In the forward test, velocity started at 0.2620 m/s at 0.2m and slightly decreased to 0.3395 m/s at 2.4m, while acceleration began at 0.3432 m/s² and dropped to 0.0480 m/s², showing the ROV's propulsion system stabilizes speed over longer distances. In the backward test, velocity ranged from 0.2575 m/s to 0.3420 m/s, and acceleration declined from 0.3316 m/s² to 0.0487 m/s². These both forward and backward performance trend indicates that the ROV maintains a consistent speed initially, but slight variations in velocity may occur due to underwater resistance or

changes in buoyancy. Furthermore, the ROV performs slightly better in forward direction compared to backward direction.

4.4.2.2 Turn Right and Left

For horizontal movement, both propellers of the horizontal thrusters rotate in the opposite direction to allows ROV steer right or left.

Table 4-6: Turn Right Test Data

Degree (°)	Time Taken (s)			
	Test 1	Test 2	Test 3	Average
0	0	0	0	0
45	1.13	1.39	1.49	1.3367
90	1.61	1.86	1.91	1.7933
135	2.09	2.16	2.2	2.1500
180	2.5	2.61	2.85	2.6533

Table 4-7: Turn Right Performance of ROV

Degree (°)	Average Time Taken (s)	Angular Velocity (rad/s)	Angular Acceleration (rad/s²)
0	0	0	0
45	1.3367	0.5876	0.4396
90	1.7933	0.8759	0.4884
135	2.1500	1.0959	0.5097
180	2.6533	1.1840	0.4462

Table 4-8: Turn Left Test Data

Degree (°)	Time Taken (s)			
	Test 1	Test 2	Test 3	Average
0	0	0	0	0
45	1.23	0.94	1.03	1.0667
90	1.53	1.34	1.51	1.4600
135	2.13	1.87	1.85	1.9500
180	2.45	2.2	2.53	2.3933

Table 4-9: Turn Left Performance of ROV

Degree (°)	Average Time Taken (s)	Angular Velocity (rad/s)	Angular Acceleration (rad/s²)
0	0	0	0
45	1.0667	0.7363	0.6903
90	1.4600	1.0759	0.7369
135	1.9500	1.2083	0.6196
180	2.3933	1.3126	0.5485

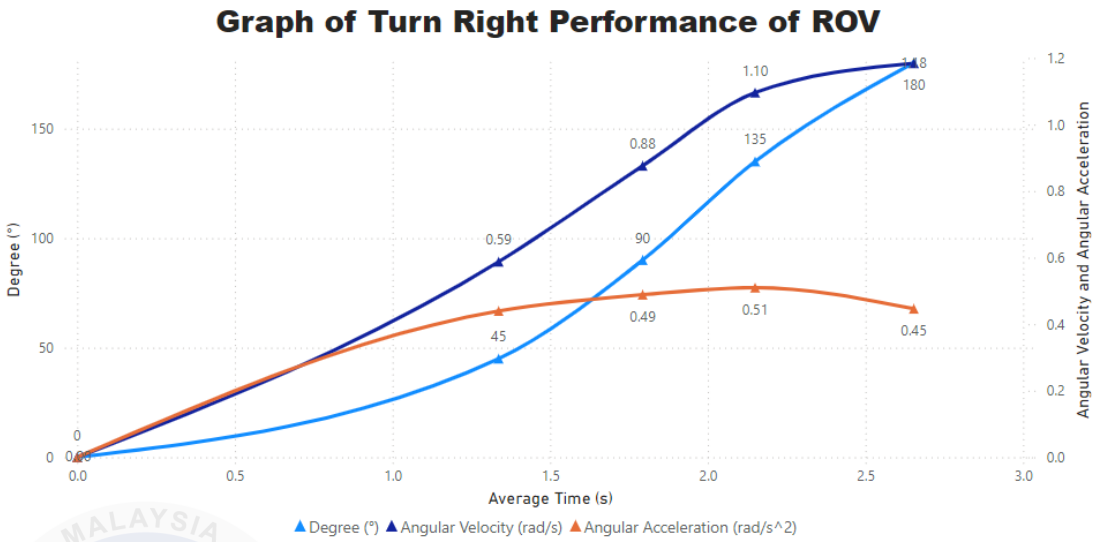


Figure 4-9: Graph of Turn Right Performance of ROV

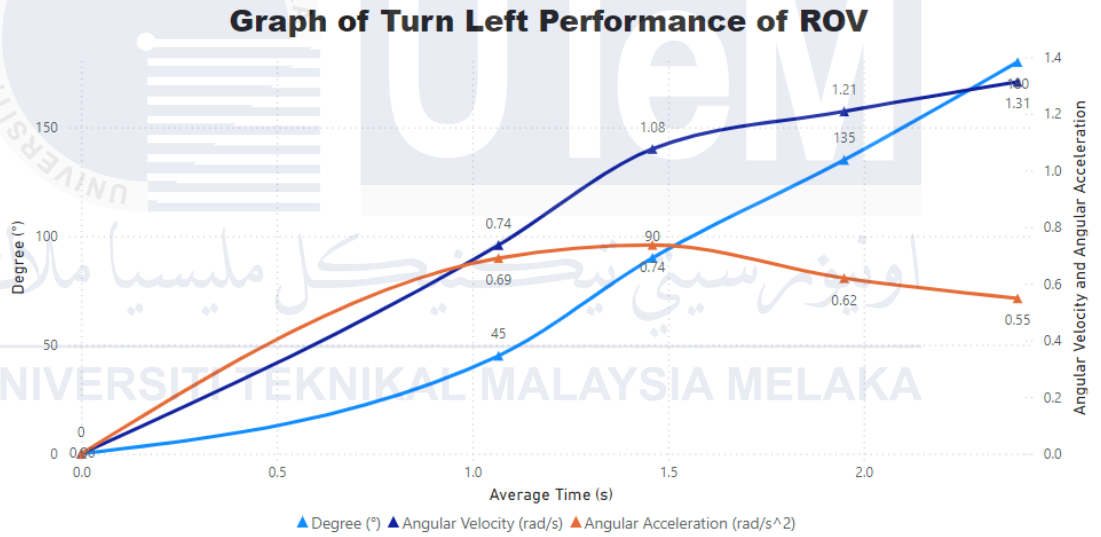


Figure 4-10: Graph of Turn Left Performance of ROV

The ROV's steering performance shows that it takes longer to complete turns as the angle increases. For example, a 45-degree right turn takes an average of 1.34 seconds with an angular velocity of 0.59 rad/s, while the same turn to the left takes only 1.07 seconds with an angular velocity of 0.74 rad/s. This trend continues with increasing angles, where the ROV shows higher angular velocities and accelerations for right and left turn compared to forward and backward, indicating better maneuverability due to asymmetries in the thruster configuration.

4.4.2.3 Raise and Submerge

Table 4-10: Raise Test Data

Distance (m)	Time Taken (s)			
	Test 1	Test 2	Test 3	Average
0	0	0	0	0
0.05	0.23	0.31	0.55	0.3633
0.1	0.77	0.86	1.03	0.8867
0.15	1.05	1.18	1.43	1.2200
0.2	1.12	1.35	1.72	1.3967
0.25	1.54	1.49	1.94	1.6567
0.3	1.75	1.67	2.17	1.8633
0.35	1.98	1.84	2.33	2.0500
0.4	2.24	2.06	2.51	2.2700
0.45	2.53	2.34	2.65	2.5067

Table 4-11: Raise Performance of ROV

Distance (m)	Average Time Taken (s)	Velocity (m/s)	Acceleration (m/s²)
0	0	0	0
0.05	0.3633	0.1376	0.3788
0.1	0.8867	0.1128	0.1272
0.15	1.2200	0.1230	0.1008
0.2	1.3967	0.1432	0.1025
0.25	1.6567	0.1509	0.0911
0.3	1.8633	0.1610	0.0864
0.35	2.0500	0.1707	0.0833
0.4	2.2700	0.1762	0.0776
0.45	2.5067	0.1795	0.0716

Table 4-12: Submerge Test Data

Distance (m)	Time Taken (s)			
	Test 1	Test 2	Test 3	Average
0	0	0	0	0
0.05	0.16	0.25	0.13	0.1800
0.1	0.77	0.94	0.61	0.7733
0.15	1.34	1.47	1.25	1.3533
0.2	1.49	1.89	1.56	1.6467
0.25	1.63	2.16	1.94	1.9100
0.3	1.88	2.35	2.28	2.1700
0.35	2.17	2.64	2.39	2.4000
0.4	2.68	2.92	2.55	2.7167
0.45	2.83	3.15	2.70	2.8933

Table 4-13: Submerge Performance of ROV

Distance (m)	Average Time Taken (s)	Velocity (m/s)	Acceleration (m/s²)
0	0	0	0
0.05	0.1800	0.2778	1.5432
0.1	0.7733	0.1293	0.1672
0.15	1.3533	0.1108	0.0819
0.2	1.6467	0.1215	0.0738
0.25	1.9100	0.1309	0.0685
0.3	2.1700	0.1382	0.0637
0.35	2.4000	0.1458	0.0608
0.4	2.7167	0.1472	0.0542
0.45	2.8933	0.1555	0.0538

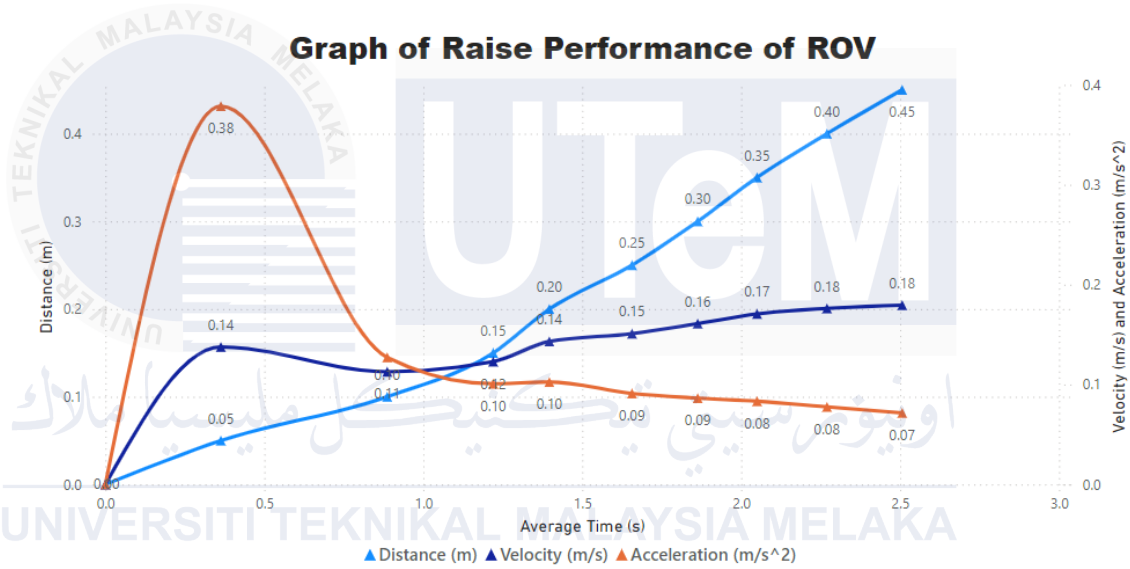


Figure 4-11: Graph of Raise Performance of ROV

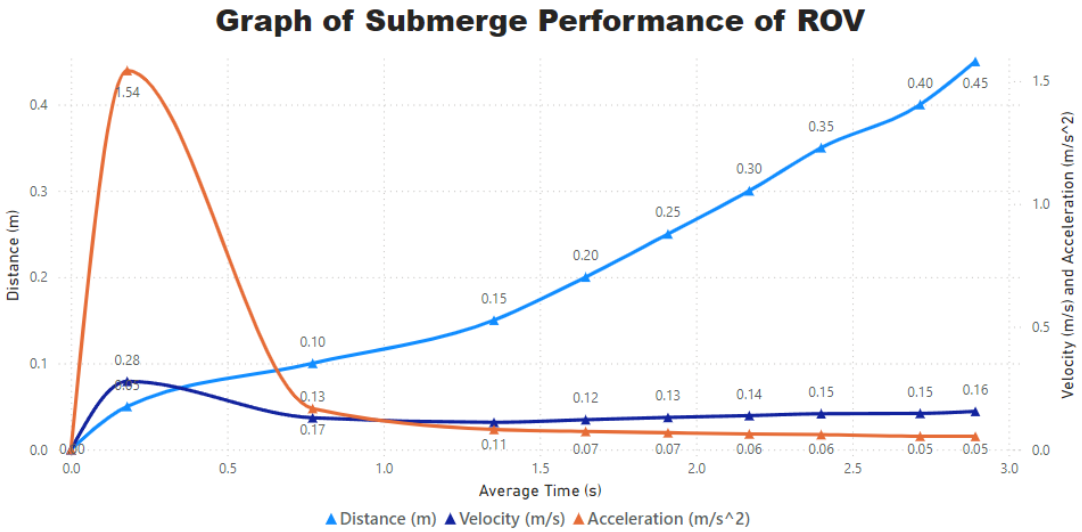


Figure 4-12: Graph of Submerge Performance of ROV

From Figure 4-11 and Figure 4-12, the ROV's raise and submerge performance shows expected trends with increasing average time and varying velocities over distance. During the raise test, velocity starts at 0.1376 m/s at 0.05 meters and increases to 0.1795 m/s at 0.45 meters, while acceleration decreases from 0.3788 m/s² to 0.0716 m/s², indicating initial acceleration and subsequent stabilization. In the submerge test, velocity starts higher at 0.2778 m/s at 0.05 meters but decreases to 0.1555 m/s at 0.45 meters, with acceleration dropping from 1.5432 m/s² to 0.0538 m/s², reflecting strong initial propulsion and later stabilization. In contrast, acceleration and velocity at the start of the submerging are higher at the ROV due to buoyancy and gravity forces. In summary, the use of the ROV is characterized by well-coordinated and stable vertical movements, the provision of which is mandatory when working in the underwater environment.

4.4.2.4 Overall Performance

Table 4-14: Overall Performance of ROV

Direction	Average Velocity (m/s)	Average Acceleration (m/s²)
Forward	0.2600	0.0858
Backward	0.2516	0.0801
Submerge	0.1357	0.2167
Raise	0.1355	0.1112

Table 4-15: Overall Turning Performance of ROV

Direction	Average Angular Velocity (rad/s)	Average Angular Acceleration (rad/s²)
Left	0.8666	0.7487
Right	0.5191	0.3768

It is depicted from Table 4-14 that the forward and backward movements have comparatively significantly higher values of average velocities that is, 0.2600 m/s and 0.2516 m/s respectively compared to vertical movements which submerge and elevate having approximately 0. 1357 m/s and 0. 1355 m/s. This implies that the design of the ROV suggests that it is designed for horizontal movement rather than vertical

movement. The average accelerations adhere to this, with the maximum acceleration measured during sinking at 0.2167 m/s², implying a higher initial propulsion force against buoyancy. The forward and backward accelerations are almost similar, at 0.0858 m/s² and 0.0801 m/s², demonstrating consistent propulsion power in horizontal movements. The raise operation has a moderate acceleration of 0.1112 m/s², ensuring a calm ascent.

Referring to Table 4-15, the turning performance of the ROV for both the left and the right shows that the average velocity and acceleration are different as compared to Table 4-14. This could imply that the ROV is more flexible and faster in maneuvering concerning turning performance as the horizontal thrusters' configurations are positioned at 45° at the back of the ROV.

In conclusion, the ROV has good horizontal control and is balanced vertically; the maneuverability demonstrated by how the ROV turns is impressive, making it useful in all sorts of underwater operations.

4.5 Summary

Several finite element analysis and experimental tests conducted on the developed ROV prototype shows that the performance is promising. It is verified from the finite element analysis that the materials used for the construction of the ROV are capable of withstanding the applied force without deformation or failure. All of the materials have a factor of safety larger than one, suggesting that they are appropriate for the intended use.

In the buoyancy experiment, the ROV prototype was critical to attain above 90% negative buoyancy to effectively submerge and rise. Thus, the goal was achieved by adding 800g of white pebble stones to the bottom PVC pipe for the best performance of the ROV. There were the several experiments being done for the purpose of credibility of all the information that were obtained.

According to Experiment 4.1 and Experiment 4.3, the forward and backward movements of the ROV had significantly greater average velocities (0.2600 m/s and 0.2516 m/s respectively) than the vertical motions of sinking and elevating (0.1357 m/s and 0.1355 m/s respectively). The effective propulsion force against buoyancy was highlighted by the greatest acceleration of 0.2167 m/s2 that was recorded during

submerging. The raise operation experienced a moderate acceleration of 0.1112 m/s², whereas the forward and backward accelerations were consistent at 0.0858 m/s² and 0.0801 m/s² respectively.

The ROV's turning performance data showed that its 45°-angled horizontal thrusters at the rear of the vehicle enable more efficient and rapid turning manoeuvres. Overall, the ROV has outstanding horizontal control, balanced vertical movement, and impressive manoeuvrability, making it ideal for a variety of underwater applications.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, the goal of this project is to design and develop an affordable micro underwater ROV for monitoring application, optimize the structural integrity and buoyancy for underwater ROV and analyze an performance of underwater ROV in terms of stability, velocity and acceleration. This project had successfully achieved in design and develop a low-cost underwater ROV for monitoring application as it only required les than RM 1000 to build the whole ROV prototype. This design of ROV prototype is capable to do monitoring and surveillance application. Otherwise, this design also waterproof body structure with optimal size which built by aluminum.

The ROV prototype can successfully move in four degrees of freedom controlled by PS2 controller. The design can perform to submerge less than 15m depth and perfectly stable at neutral buoyancy. The Arduino Uno which act as the microcontroller that been attach to the electronic circuit capable to send command from the operator to control either forward, reverse, turn right, left, submerge and raise. The challenging part in assembly procedure are to make all component waterproof to cover the electronic part inside the pressure hull.

Furthermore, the BLDC motor thruster was required ZMR 30A ESC to run, therefore the circuit must capable to withstand high current flow without burn or short circuit. The horizontal thruster was attached 45° to the body frame in order to enhanced the turning moment of underwater ROV prototype during operation and the other two thruster will be mounted vertically on the side of the ROV prototype. Therefore, the SMART ROV has ability of better maneuver controlling system.

For the sensors and equipment of the ROV prototype, the Bar02 pressure/depth sensor and MPU6050 IMUs were selected due to their high performance and suitability for use in the ROV's environment. The Bar02 pressure/depth sensor provided accurate measures of the pressure, temperature, depth and altitude during the operation of the ROV prototype, while the MPU6050 accelerometer and gyroscope sensors offered

stability and control of the orientation of the underwater robot. The performance of the system was confirmed over the multiple tests due to its increased reliability and ability to provide the proper depth and maneuvering range for the operation. Moreover, integration with OBS Studio for streaming the video feed of ROV prototype from waterproof endoscope camera and GUI from Processing which assists the enhancement of the instantaneous control for optimum user interface and operation control.

Regardless, there is a little issue that occurs during underwater operation due to a water leak inside the pressure hull that affects the electronic components, thus an appropriate technique is required by inserting highly absorbent pads and several packs of silica gel to keep the pressure hull dry, which improves the reliability and long-term of the ROV's internal systems.

5.2 Recommendation

5.2.1 Leakage Detector

One of the most difficult aspects of this project is ensuring that the inside electronics is protected from water penetration into the container. The container's inside is connected to several exterior wires that are exposed to water, therefore any unprotected gaps between these wires could cause catastrophic damage to the internal electronics. Leakage sensor should be strategically placed in key areas of the ROV to reduce this risk. These sensors will detect any water intrusion, allowing quick preventive steps to be implemented to safeguard the internal components from damage.

5.2.2 Automated Ballast System

Future development should include an additional automated ballast system that operates during emergencies. This feature is required for any underwater vehicle since it aids in the retrieval of a ROV if it goes missing underwater. The automatic ballast system comprises of a container filled with a liquid that has the same density as the surrounding fluid. In an emergency, the ballast mechanism will automatically remove

the liquid and replace it with air. This will provide the ROV positive buoyancy, allowing it to float to the surface of the ocean.

5.2.3 PID Control System

It is recommended to incorporate an improved type of controller known as Proportional-Integral-Derivative (PID) controller rather than the current steering based on the range to target figures which is rather ineffective and faulty for improving the ROV control dynamics. The PID controller will work on targets individually, and so there will always be constant adjustments to the thrusters and this will make the operation more stable. This controller will help enhancing the accuracy of the ROV by enabling it to perform desired position and orientation control without relying on range measurements. PID controller ensures the ROV to perform well in varying underwater conditions and this modularity of the control will make the ROV to be more reliable. The PID controller is designed by setting the values of the proportional, integral, and derivative values, integrating it with the existing systems, and testing the performance for checking the validity of the changes.

REFERENCE

[1] NOAA, "What is an ROV? : Ocean Exploration Facts: NOAA Office of Ocean Exploration and Research." 2019. [Online]. Available: https://oceanexplorer.noaa.gov/facts/rov.html

[2] D. Trekker, "What Are Underwater ROVs & What Are They Used For? | Deep Trekker." 2019. [Online]. Available: https://www.deeptrekker.com/resources/underwater-rovs

[3] J. Linowes *et al.*, "UNH ROV," 2015.

[4] S. C. Yu, J. Yuh, and J. Kim, "Armless underwater manipulation using a small deployable agent vehicle connected by a smart cable," *Ocean Engineering*, vol. 70, pp. 149–159, 2013, doi: 10.1016/j.oceaneng.2013.06.006.

[5] Z. Smolder and J. Yi, "Cost-Effective Remote Operated Vehicle," *Aresty Rutgers Undergraduate Research Journal*, vol. 1, Dec. 2021, doi: 10.14713/arestyrurj.v1i3.167.

[6] R. Singh, P. Sarkar, V. Goswami, and R. Yadav, "Review of low cost micro remotely operated underwater vehicle," *Ocean Engineering*, vol. 266. Elsevier Ltd, Dec. 15, 2022. doi: 10.1016/j.oceaneng.2022.112796.

[7] Y. He, D. B. Wang, and Z. A. Ali, "A review of different designs and control models of remotely operated underwater vehicle," *Measurement and Control (United Kingdom)*, vol. 53, no. 9–10, pp. 1561–1570, Nov. 2020, doi: 10.1177/0020294020952483.

[8] www.oceaneering.com, "ROV Systems | Oceaneering." Dec. 2019. [Online]. Available: https://www.oceaneering.com/rov-services/rov-systems/

[9] "Electric Underwater Robotics (ROVs) | Saab Seaeye." Dec. 2015. [Online]. Available: https://www.saabseaeye.com/news/oceanica-s-petrobras-operations-get-enhanced-lynx-rovs

[10] "UTILITY CRAWLER BASE PACKAGE | Deep Trekker." [Online]. Available: https://www.deeptrekker.com/shop/products/dt640-utility-crawler

[11] M. N. Abdul Rahim, D. T. Awg Hamid, M. F. Muhamad Said, M. N. Jamaludin, and H. H. Rozalan, "DESIGN AND DEVELOPMENT OF

REMOTELY OPERATED VEHICLE TO ASSESS THE WATER QUALITY IN AQUACULTURE AREA," *Asian People Journal (APJ)*, vol. 5, no. 2, pp. 50–60, Oct. 2022, doi: 10.37231/apj.2022.5.2.409.

[12] S. W. Moore, H. Bohm, and V. Jensen, *Underwater robotics : science, design & fabrication*. Marine Advanced Technology Education (MATE) Center, 2010.

[13] "Past ROVs." Dec. 2022. [Online]. Available: https://uwrov.org/past-rovs/

[14] Z. Smolder and J. Yi, "Cost-Effective Remote Operated Vehicle," *Aresty Rutgers Undergraduate Research Journal*, vol. 1, no. 3, Oct. 2021, doi: 10.14713/arestyrurj.v1i3.167.

[15] D. Rosen and P. Ballou, "Development of the Phantom III Remotely Operated Vehicle."

[16] "12V DC Brushed Underwater Thruster Motor Propeller for ROV Robot RC Bait Boat." [Online]. Available: https://www.ebay.com.my/itm/194221780696

[17] D. Ishak, N. A. A. Manap, M. S. Ahmad, and M. R. Arshad, "Electrically actuated thrusters for autonomous underwater vehicle," in *International Workshop on Advanced Motion Control, AMC*, 2010, pp. 619–624. doi: 10.1109/AMC.2010.5464062.

[18] "Control the Basic ESC with the Arduino Serial Monitor." [Online]. Available: https://bluerobotics.com/learn/controlling-basic-esc-with-the-arduino-serial-monitor/

[19] M. Joordens, M. M. Jamshidi, S. Eega, S. Member, M. A. Joordens, and M. Jamshidi, "Design of a low cost Thruster for an Autonomous Underwater Vehicle," 2009. [Online]. Available: https://www.researchgate.net/publication/265991723

[20] "Homebuilt Rovs." [Online]. Available: https://www.homebuiltrovs.com/seafoxretrofitthrusters.html

[21] T. Love, D. Toal, and C. Flanagan, "Buoyancy Control for an Autonomous Underwater Vehicle," *IFAC Proceedings Volumes*, vol. 36, pp. 199–204, Jan. 2003, doi: 10.1016/s1474-6670(17)36681-8.

[22] F. M. White, *Fluid Mechanics*, 8th ed. Mcgraw-Hill Education, 2016.

[23] Benny. Lautrup, *Physics of continuous matter : exotic and everyday phenomena in the macroscopic world*. Institute of Physics, 2005.

[24] M. Shahrieel Mohd Aras and F. Abdul Azis, “ROV Trainer Kit for Education Purposes,” *International Journal of Science and Research*, [Online]. Available: www.ijsr.net

[25] “Buoyancy - Underwater Robotics at RMSST.” [Online]. Available: http://underwaterrobotics.wikidot.com/buoyancy

[26] M. S. M. Aras, M. K. M. Zambri, F. A. Azis, S. S. Abdullah, and A. M. Kassim, “Design and Development of Underwater Robot for Cleaning Process”.

[27] N. Ehrich Leonard, “Stability of a Bottom-heavy Underwater Vehicle*,” 1997.

[28] C. C. Eriksen *et al.*, “Seaglider: A Long-Range Autonomous Underwater Vehicle for Oceanographic Research,” 2001.

[29] A. F. Molland, Ed., “Chapter 3 - Flotation and stability,” in *The Maritime Engineering Reference Book*, Oxford: Butterworth-Heinemann, 2008, pp. 75–115. doi: https://doi.org/10.1016/B978-0-7506-8987-8.00003-2.

[30] A. F. Molland, Ed., “Chapter 10 - Underwater vehicles,” in *The Maritime Engineering Reference Book*, Oxford: Butterworth-Heinemann, 2008, pp. 728–783. doi: https://doi.org/10.1016/B978-0-7506-8987-8.00010-X.

[31] A. R. Frost, A. P. Mcmaster, K. G. Saunders, and S. R. Lee, “The Development of a Remotely Operated Vehicle (ROV) for Aquacul ture,” 1996.

[32] “ROV design—— ballast buoyancy control.” Jan. 2023. [Online]. Available: https://www.underwaterthruster.com/blogs/knowledge/rov-design-ballast-buoyancy-control

[33] J. E. Moore and R. Compton-Hall, *Submarine Warfare: Today and Tomorrow*. Adler & Adler Publishers, 1987.

[34] C. Joochim, R. Phadungthin, and S. Srikitsuwan, “Design and development of a Remotely Operated Underwater Vehicle,” in *2015 16th International Conference on Research and Education in Mechatronics (REM)*, IEEE, Nov. 2015, pp. 148–153. doi: 10.1109/REM.2015.7380385.

[35] K. I. Parker and D. C. Folta, “Propulsion system,” *CubeSat Handbook: From Mission Design to Operations*, pp. 283–301, Jan. 2021, doi: 10.1016/B978-0-12-817884-3.00015-1.

[36] "ROV Propulsion Efficiency." Jan. 2023. [Online]. Available: https://seamor.com/rov-propulsion-efficiency/

[37] R. D. Christ and R. Wernli, "The ROV Manual: A User Guide for Remotely Operated Vehicles: Second Edition," *The ROV Manual: A User Guide for Remotely Operated Vehicles: Second Edition*, pp. 1–679, Jan. 2013.

[38] D. G. Walker, "Design and Control of an High Maneuverability Remotely Operated Vehicle with Multi-Degree of Freedom Thrusters," 2005.

[39] J. H. Li *et al.*, "Conceptual design of optimal thrust system for efficient cable burying of ROV threncher," in *2014 Oceans - St. John's*, IEEE, Sep. 2014, pp. 1–5. doi: 10.1109/OCEANS.2014.7003153.

[40] I. S. Roslan, M. Said, and A. Bakar, "Conceptual Design of Remotely Operated Underwater Vehicle," 2015.

[41] M. Shahrieel Mohd Aras, S. S. Abdullah, and A. Msm, "Investigation and evaluation of low cost depth sensor system using pressure sensor for unmanned underwater vehicle," 2012. [Online]. Available: https://www.researchgate.net/publication/267752434

[42] D. B. Duraibabu *et al.*, "An optical fibre depth (pressure) sensor for remote operated vehicles in underwater applications," *Sensors (Switzerland)*, vol. 17, no. 2, Feb. 2017, doi: 10.3390/s17020406.

[43] S. M., "Pressure Sensor - MS5803-14BA." Measurement Specialties, 2014. [Online]. Available: https://my.mouser.com/ProductDetail/Measurement-Specialties/MS580314BA01-00?qs=oFx6pF86PmChyzStAtLLQA%3D%3D

[44] C. TE, "0-30 Bar Digital Pressure Sensor | MS5837." 2017. [Online]. Available: https://www.te.com/en/product-CAT-BLPS0017.html

[45] J. Betancourt, W. Coral, and J. Colorado, "An integrated ROV solution for underwater net-cage inspection in fish farms using computer vision," *SN Appl Sci*, vol. 2, no. 12, Dec. 2020, doi: 10.1007/s42452-020-03623-z.

[46] B. Robotics, "Bar02 Ultra High Resolution 10m Depth/Pressure Sensor." [Online]. Available: https://bluerobotics.com/store/sensors-sonars-cameras/sensors/bar02-sensor-r1-rp/

[47] T. Bräunl, "Low-Cost Sensor Based Orientation Detection System for Hydrofoil Jet Ski." 2020.

[48] Y. TAWIL, “Towards understanding IMU: Basics of Accelerometer and Gyroscope Sensors and How to Compute Pitch, Roll and Yaw Angles - Atadiat.” Jun. 2023. [Online]. Available: https://atadiat.com/en/e-towards-understanding-imu-basics-of-accelerometer-and-gyroscope-sensors/

[49] M. BINTI MAT ALI, “DEVELOPMENT OF SELF BALANCING PLATFORM ON MOBILE ROBOT USING PID CONTROLLER.” 2013.

[50] B. Frazer, R. Lueg, R. Borden, and J. D. Howard, “Dronenet : The Quad Chronicles,” 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:35721114

[51] W. Geeetech, “10DOF ITG3200/ADXL345/HMC5883L/BMP085 sensor breakout - Geeetech Wiki.” 2012. [Online]. Available: https://www.geeetech.com/wiki/index.php/10DOF_ITG3200/ADXL345/HMC5883L/BMP085_sensor_breakout

[52] P. 감사 합니다 at Permalink, “Wiring the ITG3200 / ADXL345 / HMC5883L 9DOF RAZOR IMU AHRH UART SPI i2C Communication.” Jun. 2017. [Online]. Available: https://www.14core.com/wiring-the-itg3200-adxl345-hmc5883l-9dof-razor-imu-ahrh-uart-spi-i2c-communication/#google_vignette

[53] A. S. Daddi, P. P. Gundewar, and G. Mulay, “MIMO Model Development of the Navigational System of an Underwater ROV,” in *2022 International Conference for Advancement in Technology, ICONAT 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICONAT53423.2022.9725952.

[54] A. P. Paswan and A. K. Shrivastava, “Evaluation of a Tilt-Based Monitoring System for Rainfall-Induced Landslides: Development and Physical Modelling,” *Water (Switzerland)*, vol. 15, no. 10, May 2023, doi: 10.3390/w15101862.

[55] BlueRobotics, “BlueROV2 - Affordable and Capable Underwater ROV.” [Online]. Available: https://bluerobotics.com/store/rov/bluerov2/

[56] Y. M. Ahmed, O. Yaakob, and B. K. Sun, “Design of a New Low Cost ROV Vehicle,” *J Teknol*, vol. 69, no. 7, Jul. 2014, doi: 10.11113/jt.v69.3262.

[57] A. SYAHMI BIN SALIM, “DESIGNING AN ROV FOR UNDERWATER INSPECTION.” Jan. 2012. [Online]. Available: https://utpedia.utp.edu.my/id/eprint/6395/1/14006_FinRep.pdf

[58] P. Tarwadi, Y. Shiraki, O. Ganoni, S. Wei, H. S. Ahn, and B. MacDonald, “Design and Development of a Robotic Vehicle for Shallow-Water Marine Inspections.” Jan. 2020. doi: 10.48550/arXiv.2007.04563.

APPENDICES

APPENDIX A GANTT CHART FOR FYP 1

Duration	OCT				NOV				DIS				JAN		
Project Activities	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
Title confirmation															
Literature Review															
i) Search & review journals															
ii) Summarize journals															
Introduction															
i) Problem statements, objectives and scopes															
Methodology															
i) Selection of conventional design parameters															
ii) Preliminary Result (Finite Element Analysis)															
ii) Finalize materials list															
iv) Materials Order															
FYP 1 Seminar															
Progress report FYP 1															
Report FYP 1 submission															

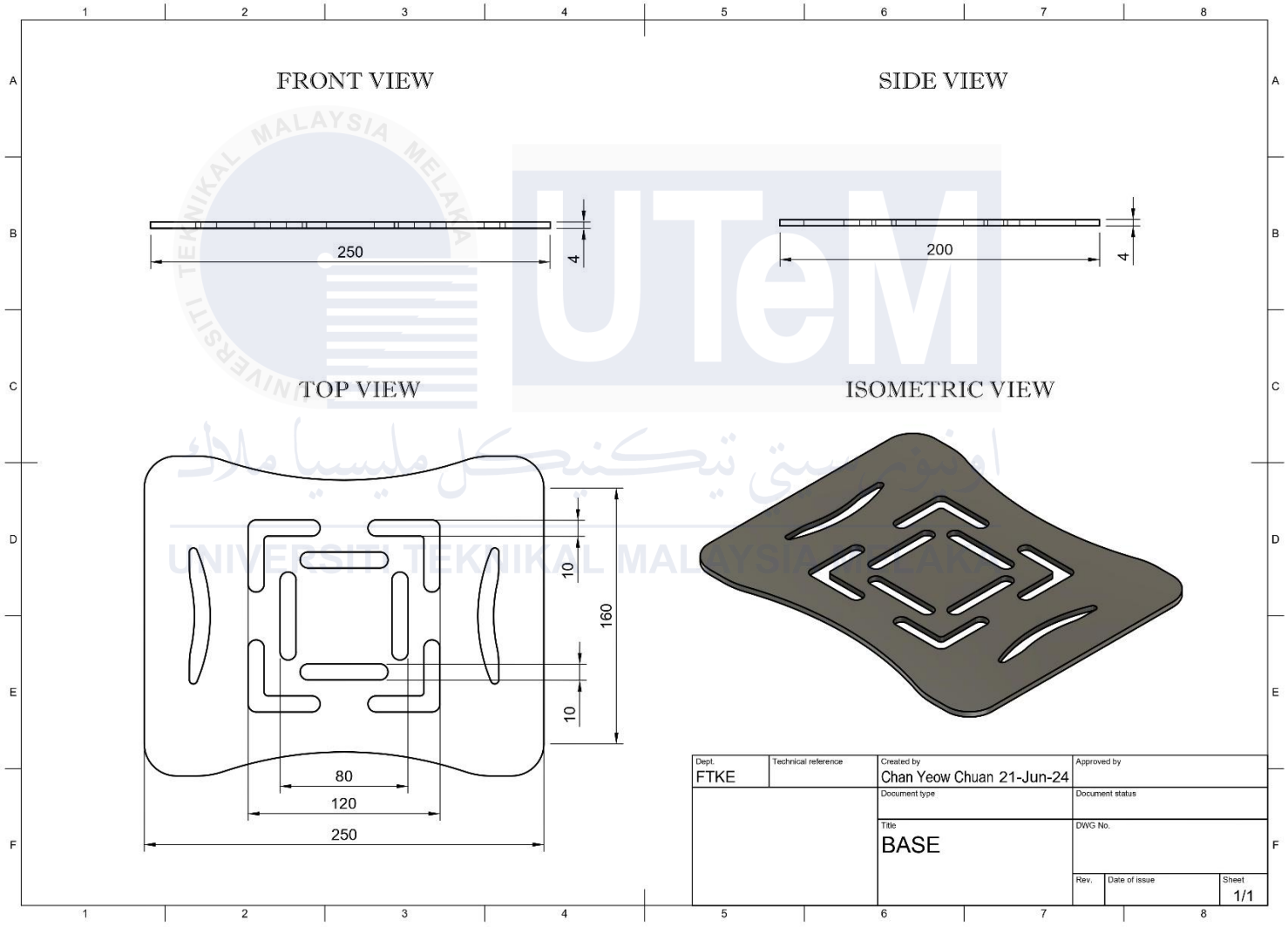
	Completed
	In Progress
	Delayed

APPENDIX B GANTT CHART FOR FYP 2

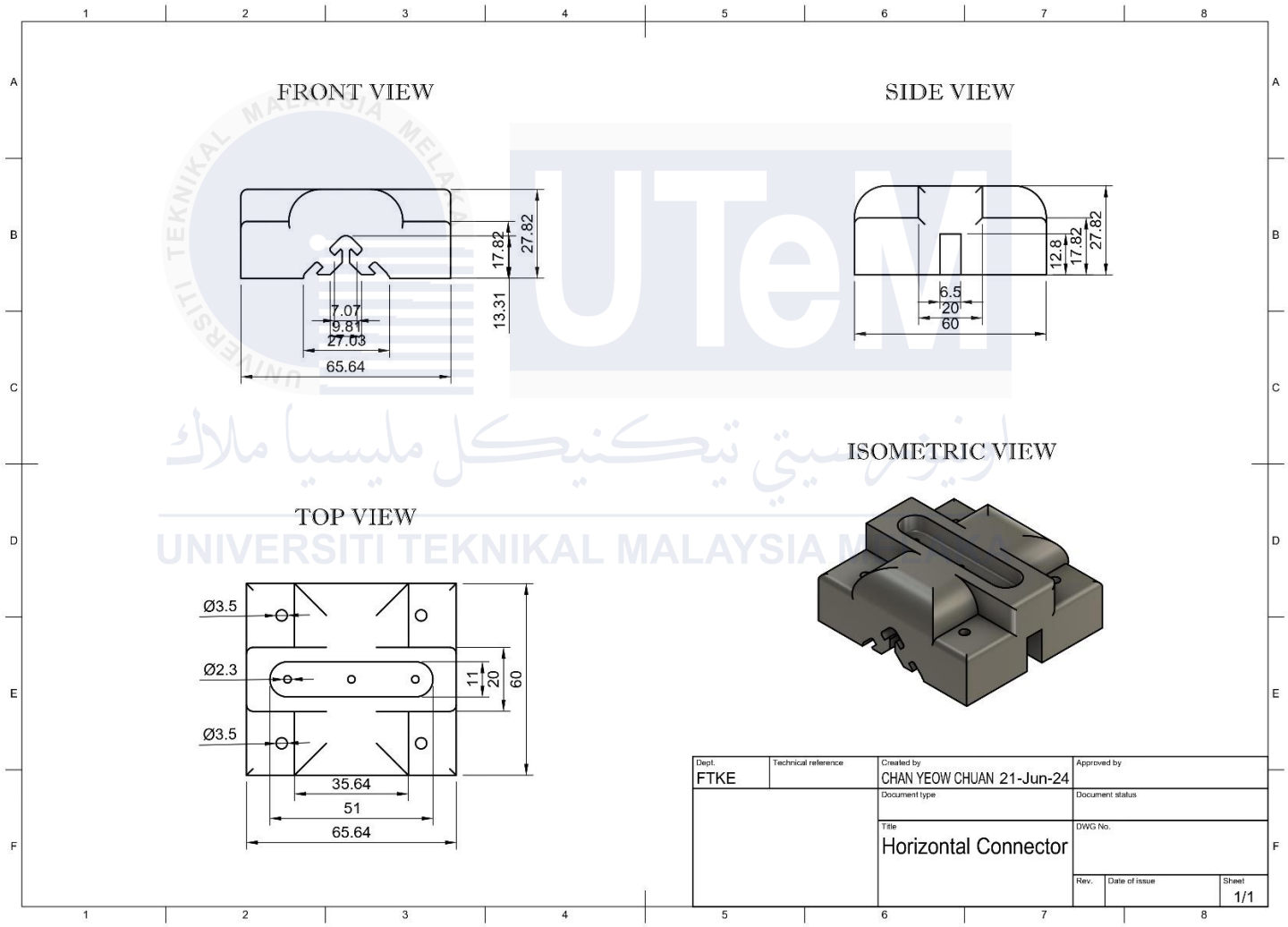
Duration	MAC				APR				MAY				JUN		
Project Activities	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
Research Methodology															
i) Design Software Coding															
ii)Design Electrical Circuit															
Assemble Prototype															
i) Trobleshooting Software Part															
i) Trobleshooting Electrical Part															
ii) Troubleshooting Hardware Part															
iii) Testing of Prototype															
iv) Modification and Improvement															
Result and Discussion															
i) Finalize Data															
Radiate 2024 (FYP 2 Presentation)															
Progress report FYP 2															
Report FYP 2 Submission															

	Completed
	In Progress
	Delayed

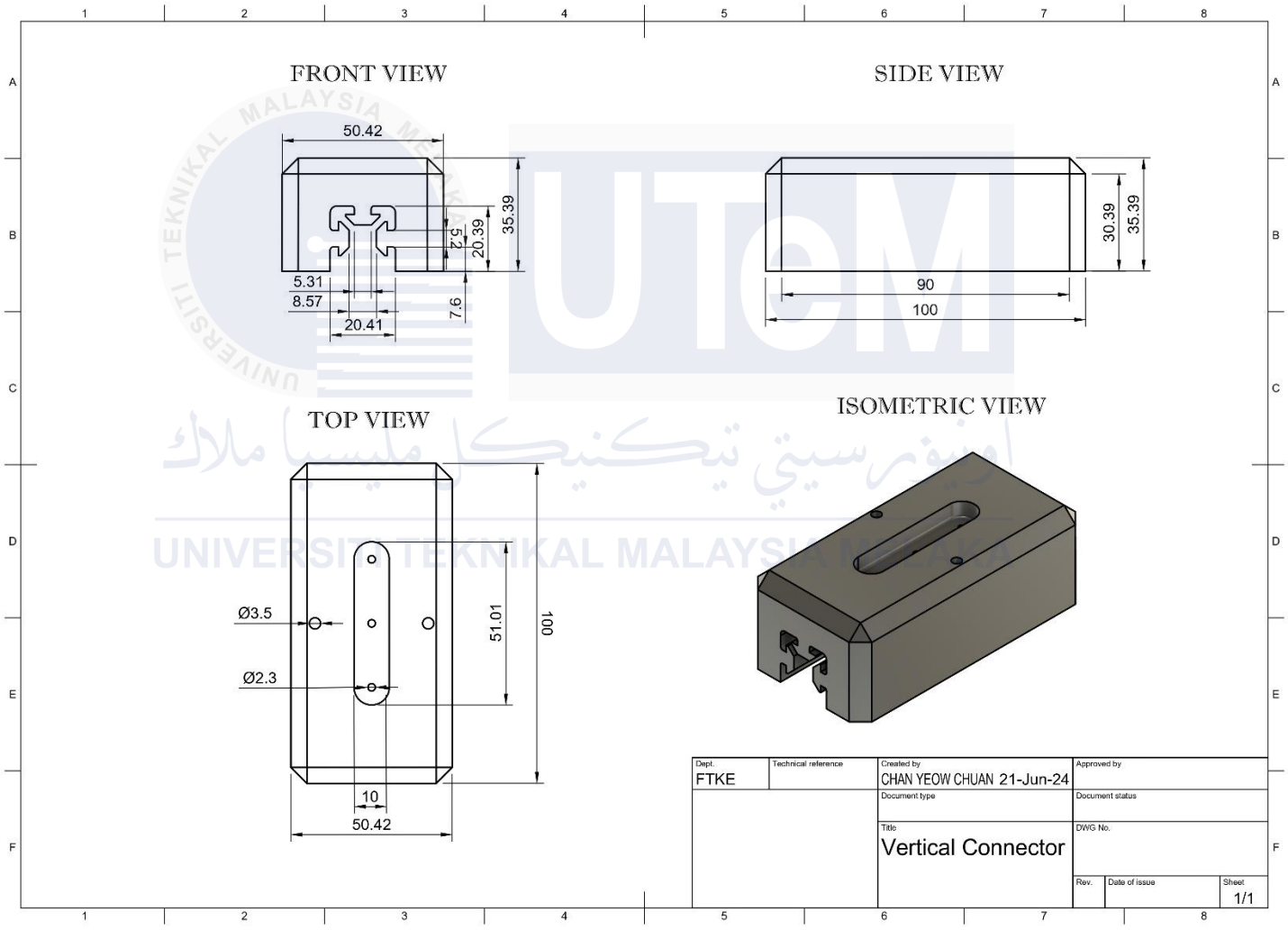
APPENDIX C SCHEMATIC DRAWING OF BASE OF ROV PROTOTYPE



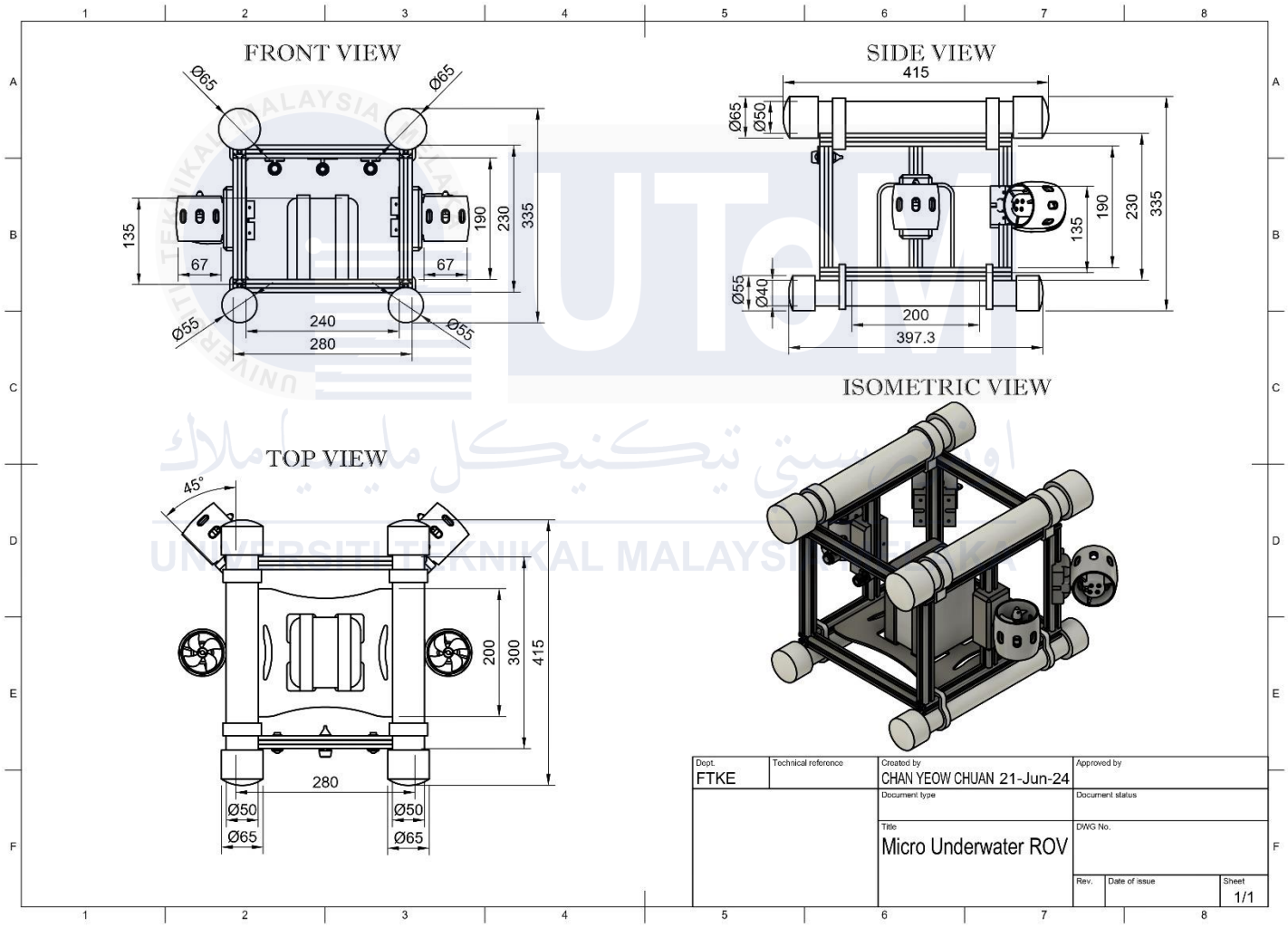
APPENDIX D SCHEMATIC DRAWING OF HORIZONTAL CONNECTOR OF ROV PROTOTYPE



APPENDIX E SCHEMATIC DRAWING OF VERTICAL CONNECTOR OF ROV PROTOTYPE



APPENDIX F SCHEMATIC DRAWING OF MICRO UNDERWATER ROV FOR MONITORING APPLICATIONS



APPENDIX G BILL OF MATERIAL

MATERIAL	QTY	UNIT PRICE (RM)	AMOUNT (RM)
190mm 2020 Aluminium Profile	4	4.05	16.20
240mm 2020 Aluminium Profile	4	5.31	21.24
300mm 2020 Aluminium Profile	4	5.94	23.76
3-Way Inner L Bracket	4	7.60	30.40
20 Series M3 Ball Spring Nut	4	0.70	2.80
T Bolt Connector	2	0.40	0.80
20 Series T Slot L Bracket	8	1.80	14.40
T Nut Rhombus Hammer Head Nut	8	0.60	4.80
M5 316 Stainless Steel Gasket (3pcs)	1	0.44	0.44
PS2 Wireless Dualshock 2 Controller	1	22.90	22.90
1500ML Food Container	1	7.90	7.90
BLDC Motor Underwater Thruster	4	59.98	239.92
ZMR 30A Bidirectional ESC	4	28.60	114.40
MPU6050 Module Sensor	1	9.90	9.90
Bar02 Ultra High Resolution 10m Depth/Pressure Sensor	1	350.00	350.00
400 Holes Breadboard	1	2.30	2.30
1M AWG20 8 Core Cable	15	5.14	77.10
9Pin Waterproof Aviation Plug Socket Connector	1	16.86	16.86
2 Channel 5V Relay Module	1	6.60	6.60
Waterproof LED Headlight	1	13.69	13.69
PG11 IP68 Nylon Cable Glands (10pcs)	1	4.89	4.89
PG7 IP68 Nylon Cable Glands (10pcs)	1	3.91	3.91
15M CAT 6 Ethernet Cable	1	7.99	7.99
		TOTAL	**993.20**

APPENDIX H PROGRAMMING CODE IN ARDUINO IDE

```
/*****************************************************************************
 * This file is part of PsxNewLib.                                           *
 *                                                                           *
 * Copyright (C) 2019-2020 by SukkoPera <software@sukkology.net>             *
 *                                                                           *
 * PsxNewLib is free software: you can redistribute it and/or                *
 * modify it under the terms of the GNU General Public License as published by *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * PsxNewLib is distributed in the hope that it will be useful,              *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with PsxNewLib. If not, see http://www.gnu.org/licenses.            *
 *****************************************************************************
 *
 * This sketch will dump to serial whatever is done on a PSX controller. It is
 * an excellent way to test that all buttons/sticks are read correctly.
 *
 * It's missing support for analog buttons, that will come in the future.
 *
 * This example drives the controller through the hardware SPI port, so pins are
 * fixed and depend on the board/microcontroller being used. For instance, on an
 * Arduino Uno connections must be as follows:
 *
 * CMD: Pin 11
 * DATA: Pin 12
 * CLK: Pin 13
 *
 * Any pin can be used for ATTN, but please note that most 8-bit AVRs require
 * the HW SPI SS pin to be kept as an output for HW SPI to be in master mode, so
 * using that pin for ATTN is a natural choice. On the Uno this would be pin 10.
 *
 * It also works perfectly on OpenPSX2AmigaPadAdapter boards (as it's basically
 * a modified Uno).
 *
 * There is another similar one using a bitbanged protocol implementation that
 * can be used on any pins/board.
 */

#include <DigitalIO.h>
#include <PsxControllerHwSpi.h>
#include <Wire.h>
#include "MS5837.h"

#include <Servo.h>

#include <avr/pgmspace.h>

#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
```

```
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

MS5837 sensor;
MPU6050 mpu;

#define OUTPUT_READABLE_YAWPITCHROLL
#define LED_PIN 13
bool blinkState = false;

typedef const __FlashStringHelper * FlashStr;
typedef const byte* PGM_BYTES_P;
#define PSTR_TO_F(s) reinterpret_cast<const __FlashStringHelper *> (s)

Servo esc1;         // create servo object to control servo
Servo esc2;
Servo esc3;
Servo esc4;

int leftspd = 0;          // variable to define speed sent to motor/servo
int rightspd = 0;         // variable to define speed sent to motor/servo
int ly = 0;               // Variable for fixed values based on Left Stick Y-axis
int ry = 0;               // Variable for fixed values based on Left Stick Y-axis
int threshold = 128;      // Assuming center value is 128

const int ESC_PIN3 = 3;    // Define ESC control pin
const int ESC_PIN4 = 4;    // Define ESC control pin
const int ESC_PIN5 = 5;    // Define ESC control pin
const int ESC_PIN6 = 6;    // Define ESC control pin

const int RELAY_PIN8 = 8;   // Pin connected to the relay
const int RELAY_PIN9 = 9;   // Pin connected to the relay

// This can be changed freely but please see above
const byte PIN_PS2_ATT = 10;

const byte PIN_BUTTONPRESS = A0;
const byte PIN_HAVECONTROLLER = A1;

const char buttonSelectName[] PROGMEM = "Select";
const char buttonL3Name[] PROGMEM = "L3";
const char buttonR3Name[] PROGMEM = "R3";
const char buttonStartName[] PROGMEM = "Start";
const char buttonUpName[] PROGMEM = "Up";
const char buttonRightName[] PROGMEM = "Right";
const char buttonDownName[] PROGMEM = "Down";
const char buttonLeftName[] PROGMEM = "Left";
const char buttonL2Name[] PROGMEM = "L2";
const char buttonR2Name[] PROGMEM = "R2";
const char buttonL1Name[] PROGMEM = "L1";
const char buttonR1Name[] PROGMEM = "R1";
const char buttonTriangleName[] PROGMEM = "Triangle";
```

```
const char buttonCircleName[] PROGMEM = "Circle";
const char buttonCrossName[] PROGMEM = "Cross";
const char buttonSquareName[] PROGMEM = "Square";

const char* const psxButtonNames[PSX_BUTTONS_NO] PROGMEM = {
  buttonSelectName,
  buttonL3Name,
  buttonR3Name,
  buttonStartName,
  buttonUpName,
  buttonRightName,
  buttonDownName,
  buttonLeftName,
  buttonL2Name,
  buttonR2Name,
  buttonL1Name,
  buttonR1Name,
  buttonTriangleName,
  buttonCircleName,
  buttonCrossName,
  buttonSquareName
};

byte psxButtonToIndex (PsxButtons psxButtons) {
  byte i;

  for (i = 0; i < PSX_BUTTONS_NO; ++i) {
    if (psxButtons & 0x01) {
      break;
    }

    psxButtons >>= 1U;
  }

  return i;
}

FlashStr getButtonName (PsxButtons psxButton) {
  FlashStr ret = F("");

  byte b = psxButtonToIndex (psxButton);
  if (b < PSX_BUTTONS_NO) {
    PGM_BYTES_P bName = reinterpret_cast<PGM_BYTES_P> (pgm_read_ptr
(&(psxButtonNames[b])));
    ret = PSTR_TO_F (bName);
  }

  return ret;
}

void dumpButtons (PsxButtons psxButtons) {
  static PsxButtons lastB = 0;

  if (psxButtons != lastB) {
```

```
    lastB = psxButtons;     // Save it before we alter it

    Serial.print (F("Pressed: "));

    for (byte i = 0; i < PSX_BUTTONS_NO; ++i) {
      byte b = psxButtonToIndex (psxButtons);
      if (b < PSX_BUTTONS_NO) {
        PGM_BYTES_P bName = reinterpret_cast<PGM_BYTES_P> (pgm_read_ptr
(&(psxButtonNames[b])));
        //Serial.print (PSTR_TO_F (bName));
      }

      psxButtons &= ~(1 << b);

      if (psxButtons != 0) {
        //Serial.print (F(", "));
      }
    }

    //Serial.println ();
  }
}

void dumpAnalog (const char *str, const byte x, const byte y) {
  Serial.print (str);
  Serial.print (F(" analog: x = "));
  Serial.print (x);
  Serial.print (F(", y = "));
  Serial.println (y);
}
```



```
const char ctrlTypeUnknown[] PROGMEM = "Unknown";
const char ctrlTypeDualShock[] PROGMEM = "Dual Shock";
const char ctrlTypeDsWireless[] PROGMEM = "Dual Shock Wireless";
const char ctrlTypeGuitHero[] PROGMEM = "Guitar Hero";
const char ctrlTypeOutOfBounds[] PROGMEM = "(Out of bounds)";

const char* const controllerTypeStrings[PSCTRL_MAX + 1] PROGMEM = {
  ctrlTypeUnknown,
  ctrlTypeDualShock,
  ctrlTypeDsWireless,
  ctrlTypeGuitHero,
  ctrlTypeOutOfBounds
};

PsxControllerHwSpi<PIN_PS2_ATT> psx;

boolean haveController = false;
boolean relayState = false; // Initial state of the relay

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
```

```
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 =
error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity
vector

// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };

// ================================================================
// ===               INTERRUPT DETECTION ROUTINE                ===
// ================================================================

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone
high
void dmpDataReady() {
    mpuInterrupt = true;
}

void setup () {

  fastPinMode (PIN_BUTTONPRESS, OUTPUT);
  fastPinMode (PIN_HAVECONTROLLER, OUTPUT);

  delay (300);

  Serial.begin (9600);
  while(!Serial);
  Serial.println (F("Ready!"));

  esc1.attach(ESC_PIN3, 1000, 2000);          // Initialize ESC1
  esc2.attach(ESC_PIN4, 1000, 2000);          // Initialize ESC2
  esc3.attach(ESC_PIN5, 1000, 2000);          // Initialize ESC3
  esc4.attach(ESC_PIN6, 1000, 2000);          // Initialize ESC3

  pinMode(RELAY_PIN8,OUTPUT); // Set relay pin as output
  pinMode(RELAY_PIN9,OUTPUT); // Set relay pin as output

  setupCompass();

  setupPressure();

}
```

```
void loop () {

  static byte slx, sly, srx, sry;

  fastDigitalWrite (PIN_HAVECONTROLLER, haveController);

  if (!haveController) {
    if (psx.begin ()) {
      Serial.println (F("Controller found!"));
      delay (300);
      if (!psx.enterConfigMode ()) {
        Serial.println (F("Cannot enter config mode"));
      } else {
        PsxControllerType ctype = psx.getControllerType ();
        PGM_BYTES_P cname = reinterpret_cast<PGM_BYTES_P> (pgm_read_ptr
(&(controllerTypeStrings[ctype < PSCTRL_MAX ? static_cast<byte> (ctype) : PSCTRL_MAX])));
        Serial.print (F("Controller Type is: "));
        Serial.println (PSTR_TO_F (cname));

        if (!psx.enableAnalogSticks ()) {
          Serial.println (F("Cannot enable analog sticks"));
        }

        //~ if (!psx.setAnalogMode (false)) {
          //~ Serial.println (F("Cannot disable analog mode"));
        //~ }
        //~ delay (10);

        if (!psx.enableAnalogButtons ()) {
          Serial.println (F("Cannot enable analog buttons"));
        }

        if (!psx.exitConfigMode ()) {
          Serial.println (F("Cannot exit config mode"));
        }
      }

      haveController = true;
    }
  } else {
    if (!psx.read ()) {
      Serial.println (F("Controller lost :("));
      haveController = false;
    } else {
      fastDigitalWrite (PIN_BUTTONPRESS, !!psx.getButtonWord ());
      dumpButtons (psx.getButtonWord ());
      boolean l1Pressed = psx.getButtonWord() & (1 << 10); // Check L1 (bit position 10)
      boolean r1Pressed = psx.getButtonWord() & (1 << 11); // Check R1 (bit position 11)
      boolean squarePressed = psx.getButtonWord() & (1 << 15); // Check Square button (bit
position 15)

      if (squarePressed) {
          // Toggle relay state
```

```
        relayState = !relayState;
        digitalWrite(RELAY_PIN8 , relayState ? HIGH : LOW);
        digitalWrite(RELAY_PIN9 , relayState ? HIGH : LOW);
        Serial.println(relayState ? F("WhiteLight ON") : F("YellowLight ON"));
        delay(10); // Debounce delay
    }

    if (l1Pressed && r1Pressed) {
        // Both L1 and R1 pressed, stop motor
        // You can add code here to stop the motor
        esc1.write(90);
        esc4.write(90);
        //Serial.println(F("Motor stopped"));
      } else if (l1Pressed && !r1Pressed) {
        // Only L1 pressed, move motor down
        // You can add code here to move the motor down
        esc1.write(0);
        esc4.write(0);
        //Serial.println(F("Motor moving down"));
      } else if (!l1Pressed && r1Pressed) {
        // Only R1 pressed, move motor up
        //You can add code here to move the motor up
        esc1.write(180);
        esc4.write(180);
        //Serial.println(F("Motor moving up"));
      } else{
        // Neither L1 nor R1 pressed, stop motor
        // You can add code here to stop the motor
        esc1.write(90);
        esc4.write(90);
      }

    byte lx, ly;
    psx.getLeftAnalog (lx, ly);
    // Determine direction based on thresholds
    if (ly > threshold) {
      // Move clockwise
      int forward = map(ly, threshold, 255, 90, 180); // Map Y-axis value to speed range
(0 to 180)
      esc2.write(forward); // Adjust center value accordingly
      esc3.write(forward); // Adjust center value accordingly
    } else if (ly < threshold) {
      // Move anticlockwise
      int backward = map(ly, threshold,0,  90, 0); // Map Y-axis value to speed range (0
to -180)
      esc2.write(backward); // Adjust center value accordingly
      esc3.write(backward); // Adjust center value accordingly
    } else {
      // Stop if stick is at center
      esc2.write(90);
      esc3.write(90);
    }

    byte rx, ry;
```

```
      psx.getRightAnalog (rx, ry);
      // Determine direction based on thresholds
      if (rx > threshold) {
        // Move clockwise
        int right = map(rx, threshold, 255, 90, 180); // Map Y-axis value to speed range
(0 to 180)
        int left = map(rx, threshold, 255, 90, 0); // Map Y-axis value to speed range (0
to 180)
        esc2.write(right); // Adjust center value accordingly
        esc3.write(left); // Adjust center value accordingly
      } else if (rx < threshold) {
        // Move anticlockwise
        int right = map(rx, threshold,0,  90, 0); // Map Y-axis value to speed range (0 to
-180)
        int left = map(rx, threshold,0,  90, 180); // Map Y-axis value to speed range (0
to -180)
        esc2.write(right); // Adjust center value accordingly
        esc3.write(left); // Adjust center value accordingly
      } else {
        // Stop if stick is at center
        esc2.write(90);
        esc3.write(90);
      }
    }

  }
  loopCompass();

  delay (1000 / 60);
}

void setupCompass(){
  Wire.begin();
  mpu.initialize();
  devStatus = mpu.dmpInitialize();

  // supply your own gyro offsets here, scaled for min sensitivity
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

  // make sure it worked (returns 0 if so)
  if (devStatus == 0) {
      mpu.setDMPEnabled(true);

      // enable Arduino interrupt detection
      //Serial.println(F("Enabling interrupt detection (Arduino external interrupt
0)..."));
      attachInterrupt(0, dmpDataReady, RISING);
      mpuIntStatus = mpu.getIntStatus();

      // set our DMP Ready flag so the main loop() function knows it's okay to use it
      //Serial.println(F("DMP ready! Waiting for first interrupt..."));
```

```
        dmpReady = true;

        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {

        Serial.print(devStatus);
        Serial.println(F(")"));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}

void setupPressure(){

    Wire.begin();

    // Initialize pressure sensor
    // Returns true if initialization was successful
    // We can't continue with the rest of the program unless we can initialize the sensor
    while (!sensor.init()) {
      Serial.println("Init failed!");
      Serial.println("Are SDA/SCL connected correctly?");
      Serial.println("Blue Robotics Bar30: White=SDA, Green=SCL");
      Serial.println("\n\n\n");
      delay(5000);
    }

    sensor.setModel(MS5837::MS5837_02BA);
    sensor.setFluidDensity(997); // kg/m^3 (freshwater, 1029 for seawater)
}

void loopCompass() {
    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize) {

    }

    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code is too inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
```

```
// otherwise, check for DMP data ready interrupt (this should happen frequently)
} else if (mpuIntStatus & 0x02) {
    // wait for correct available data length, should be a VERY short wait
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

    // read a packet from FIFO
    mpu.getFIFOBytes(fifoBuffer, packetSize);

    // track FIFO count here in case there is > 1 packet available
    // (this lets us immediately read more without waiting for an interrupt)
    fifoCount -= packetSize;

    #ifdef OUTPUT_READABLE_QUATERNION
        // quaternion values in easy matrix form: w x y z
        mpu.dmpGetQuaternion(&q, fifoBuffer);
    #endif

    #ifdef OUTPUT_READABLE_EULER
        // Euler angles in degrees
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetEuler(euler, &q);
    #endif

    #ifdef OUTPUT_READABLE_YAWPITCHROLL
        //display Euler angles in degrees
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

        sensor.read();

        //Serial.print("Phi: ");
        Serial.print(ypr[2] * 18/M_PI);
        //Serial.print("\t theta: ");
        Serial.print(" ");
        Serial.print(ypr[1] * 180/M_PI);
        //Serial.print("\t Psi: ");
        Serial.print(" ");
        Serial.print(ypr[0] * 180/M_PI);
        Serial.print(" ");
        Serial.print(sensor.pressure());      // Pressure
        Serial.print(" ");
        Serial.print(sensor.temperature());   // Temperature
        Serial.print(" ");
        Serial.print(sensor.depth());         // Depth
        Serial.print(" ");
        Serial.println(sensor.altitude());    // Altitude
        //delay(100);
  #endif

    #ifdef OUTPUT_READABLE_REALACCEL
        // real acceleration, adjusted to remove gravity
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetAccel(&aa, fifoBuffer);
```

```
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
        #endif

        #ifdef OUTPUT_READABLE_WORLDACCEL
            // initial world-frame acceleration, adjusted to remove gravity
            // and rotated based on known orientation from quaternion
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetAccel(&aa, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
            mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
        #endif

        #ifdef OUTPUT_TEAPOT
            // display quaternion values in InvenSense Teapot demo format:
            teapotPacket[2] = fifoBuffer[0];
            teapotPacket[3] = fifoBuffer[1];
            teapotPacket[4] = fifoBuffer[4];
            teapotPacket[5] = fifoBuffer[5];
            teapotPacket[6] = fifoBuffer[8];
            teapotPacket[7] = fifoBuffer[9];
            teapotPacket[8] = fifoBuffer[12];
            teapotPacket[9] = fifoBuffer[13];
            teapotPacket[11]++; // packetCount, loops at 0xFF on purpose
        #endif

        // blink LED to indicate activity
        blinkState = !blinkState;
        digitalWrite(LED_PIN, blinkState);
    }
}
```

APPENDIX I PROGRAMMING CODE IN PROCESSING SOFTWARE

```
import org.firmata.*;
import cc.arduino.*;

import processing.serial.*;
import cc.arduino.*;

float Pressure;
float Temperature;
float Depth;
float Altitude;

float Pitch;
float Roll;
float Yaw;
float ArtificialHoizonMagnificationFactor=0.7;
float CompassMagnificationFactor=0.85;
float SpanAngle=120;
int NumberOfScaleMajorDivisions;
int NumberOfScaleMinorDivisions;
PVector v1, v2;


Serial port;
float Phi;
float Theta;
float Psi;

void setup()
{
  size(1366, 800);
  rectMode(CENTER);
  smooth();
  strokeCap(SQUARE);//Optional

  println(Serial.list());
  port = new Serial(this, Serial.list()[1], 9600);
  port.bufferUntil(&apos;\n&apos;);
}
void draw()
{
  background(0);
  translate(1366/4, 768/2.1);
  MakeAnglesDependentOnMPU6050();
  Horizon();
  rotate(-Roll);
  PitchScale();
  Axis();
  rotate(Roll);
  Borders();
  Plane();
  ShowAngles();
  Compass();
  ShowYaw();
  ShowPressure();  // Add this line to call the pressure display function
  ShowTemperature(); // Add this line to call the temperature display function
  ShowDepth();      // Add this line to call the depth display function
  ShowAltitude();   // Add this line to call the altitude display function
}
void serialEvent(Serial port)
{
  while(port.available()>0){
    String input = port.readStringUntil(&apos;\n&apos;);
    if(input != null){
       input = trim(input);
       String[] values = split(input, " ");
       if(values.length == 7){
```

```
            float phi = float(values[0]);
            float theta = float(values[1]);
            float psi = float(values[2]);
            float pressure = float(values[3]);
            float temperature = float(values[4]);
            float depth = float(values[5]);
            float altitude =  float(values[6]);
             print(phi);
             print(theta);
             println(psi);
             println(pressure);
             Phi = phi;
             Theta = theta;
             Psi = psi;
             Pressure = pressure;  // Store the pressure data
             Temperature = temperature;
             Depth = depth;
             Altitude = altitude;
        }
     }
   }
}
void MakeAnglesDependentOnMPU6050()
{
  Roll =-Phi/5;
  Pitch=Theta*10;
  Yaw=Psi;
}
void Horizon()
{
  scale(ArtificialHoizonMagnificationFactor);
  noStroke();
  fill(0, 180, 255);
  rect(0, -100, 900, 1000);
  fill(95, 55, 40);
  rotate(Roll);
  rect(0, 400+Pitch, 900, 800);
  rotate(-Roll);
  rotate(-PI-PI/6);
  SpanAngle=120;
  NumberOfScaleMajorDivisions=12;
  NumberOfScaleMinorDivisions=24;
  CircularScale();
  rotate(PI+PI/6);
  rotate(-PI/6);
  CircularScale();
  rotate(PI/6);
}
void ShowYaw()
{
  fill(50);
  noStroke();
  rect(20, 470, 440, 50);
  int Yaw1=round(Yaw);
  textAlign(CORNER);
  textSize(35);
  fill(255);
  text("Yaw:  "+Yaw1+" °", 80, 477, 500, 70);
  textSize(40);
  fill(255,0,95);
  text("|TechPilot|", -335, -500, 500, 60);
  text("~UROV Instruments~", -425, -450, 500, 70);
}
void Compass()
{
  translate(2*1366/3, 0);
  scale(CompassMagnificationFactor);
```

```
  noFill();
  stroke(100);
  strokeWeight(80);
  ellipse(0, 0, 750, 750);
  strokeWeight(50);
  stroke(50);
  fill(0, 0, 40);
  ellipse(0, 0, 610, 610);
  for (int k=255;k>0;k=k-5)
  {
    noStroke();
    fill(0, 0, 255-k);
    ellipse(0, 0, 2*k, 2*k);
  }
  strokeWeight(20);
  NumberOfScaleMajorDivisions=18;
  NumberOfScaleMinorDivisions=36;
  SpanAngle=180;
  CircularScale();
  rotate(PI);
  SpanAngle=180;
  CircularScale();
  rotate(-PI);
  fill(255);
  textSize(60);
  textAlign(CENTER);
  text("W", -375, 0, 100, 80);
  text("E", 370, 0, 100, 80);
  text("N", 0, -365, 100, 80);
  text("S", 0, 375, 100, 80);
  textSize(30);
  text("COMPASS", 0, -130, 500, 80);
  rotate(PI/4);
  textSize(40);
  text("NW", -370, 0, 100, 50);
  text("SE", 365, 0, 100, 50);
  text("NE", 0, -355, 100, 50);
  text("SW", 0, 365, 100, 50);
  rotate(-PI/4);
  CompassPointer();
}
void CompassPointer()
{
  rotate(PI+radians(Yaw));
  stroke(0);
  strokeWeight(4);
  fill(100, 255, 100);
  triangle(-20, -210, 20, -210, 0, 270);
  triangle(-15, 210, 15, 210, 0, 270);
  ellipse(0, 0, 45, 45);
  fill(0, 0, 50);
  noStroke();
  ellipse(0, 0, 10, 10);
  triangle(-20, -213, 20, -213, 0, -190);
  triangle(-15, -215, 15, -215, 0, -200);
  rotate(-PI-radians(Yaw));
}
void Plane()
{
  fill(0);
  strokeWeight(1);
  stroke(0, 255, 0);
  triangle(-20, 0, 20, 0, 0, 25);
  rect(110, 0, 140, 20);
  rect(-110, 0, 140, 20);
}
void CircularScale()
```

```
  rect(150, 400, 280, 40);
  fill(255);
  Pitch=Pitch/5;
  int Pitch1=round(Pitch);
  text("Pitch:  "+Pitch1+" °", -20, 411, 500, 60);
  text("Roll:  "+Roll*100+" °", 280, 411, 500, 60);
}
void Borders()
{
  noFill();
  stroke(0);
  strokeWeight(400);
  rect(0, 0, 1100, 1100);
  strokeWeight(200);
  ellipse(0, 0, 1000, 1000);
  fill(0);
  noStroke();
  rect(4*1366/5, 0,1366, 2*768);
  rect(-4*1366/5, 0, 1366, 2*768);
}
void PitchScale()
{
  stroke(255);
  fill(255);
  strokeWeight(3);
  textSize(24);
  textAlign(CENTER);
  for (int i=-4;i<5;i++)
  {
    if ((i==0)==false)
    {
      line(110, 50*i, -110, 50*i);
    }
    text(""+i*10, 140, 50*i, 100, 30);
    text(""+i*10, -140, 50*i, 100, 30);
  }
  textAlign(CORNER);
  strokeWeight(2);
  for (int i=-9;i<10;i++)
  {
    if ((i==0)==false)
    {
      line(25, 25*i, -25, 25*i);
    }
  }
}
void ShowPressure() {
   fill(0,100,0);
   noStroke();
   rect(-500, 550, 300, 50);  // Adjust the position and size as needed
   fill(255);
   textSize(30);
   textAlign(CENTER,CENTER);
   text("Pressure: " + Pressure + " Pa", -500, 550, 300, 50);  // Adjust the position
as needed
}
void ShowTemperature() {
   fill(0,100,200);
   noStroke();
   rect(-500, 600, 300, 50);  // Adjust the position and size as needed
   fill(255);
   textSize(30);
   textAlign(CENTER,CENTER);
   text("Temperature: " + Temperature + " °C", -500, 600, 300, 50);  // Adjust the
position as needed
}
```

```
{
  float GaugeWidth=800;
  textSize(GaugeWidth/30);
  float StrokeWidth=1;
  float an;
  float DivxPhasorCloser;
  float DivxPhasorDistal;
  float DivyPhasorCloser;
  float DivyPhasorDistal;
  strokeWeight(2*StrokeWidth);
  stroke(255);
  float DivCloserPhasorLenght=GaugeWidth/2-GaugeWidth/9-StrokeWidth;
  float DivDistalPhasorLenght=GaugeWidth/2-GaugeWidth/7.5-StrokeWidth;
  for (int Division=0;Division<NumberOfScaleMinorDivisions+1;Division++)
  {
    an=SpanAngle/2+Division*SpanAngle/NumberOfScaleMinorDivisions;
    DivxPhasorCloser=DivCloserPhasorLenght*cos(radians(an));
    DivxPhasorDistal=DivDistalPhasorLenght*cos(radians(an));
    DivyPhasorCloser=DivCloserPhasorLenght*sin(radians(an));
    DivyPhasorDistal=DivDistalPhasorLenght*sin(radians(an));
    line(DivxPhasorCloser, DivyPhasorCloser, DivxPhasorDistal, DivyPhasorDistal);
  }
  DivCloserPhasorLenght=GaugeWidth/2-GaugeWidth/10-StrokeWidth;
  DivDistalPhasorLenght=GaugeWidth/2-GaugeWidth/7.4-StrokeWidth;
  for (int Division=0;Division<NumberOfScaleMajorDivisions+1;Division++)
  {
    an=SpanAngle/2+Division*SpanAngle/NumberOfScaleMajorDivisions;
    DivxPhasorCloser=DivCloserPhasorLenght*cos(radians(an));
    DivxPhasorDistal=DivDistalPhasorLenght*cos(radians(an));
    DivyPhasorCloser=DivCloserPhasorLenght*sin(radians(an));
    DivyPhasorDistal=DivDistalPhasorLenght*sin(radians(an));
    if
(Division==NumberOfScaleMajorDivisions/2|Division==0|Division==NumberOfScaleMajorDivisi
ons)
    {
      strokeWeight(15);
      stroke(0);
      line(DivxPhasorCloser, DivyPhasorCloser, DivxPhasorDistal, DivyPhasorDistal);
      strokeWeight(8);
      stroke(100, 255, 100);
      line(DivxPhasorCloser, DivyPhasorCloser, DivxPhasorDistal, DivyPhasorDistal);
    }
    else
    {
      strokeWeight(3);
      stroke(255);
      line(DivxPhasorCloser, DivyPhasorCloser, DivxPhasorDistal, DivyPhasorDistal);
    }
  }
}
void Axis()
{
  stroke(255, 0, 0);
  strokeWeight(3);
  line(-115, 0, 115, 0);
  line(0, 280, 0, -280);
  fill(100, 255, 100);
  stroke(0);
  triangle(0, -285, -10, -255, 10, -255);
  triangle(0, 285, -10, 255, 10, 255);
}
void ShowAngles()
{
  textSize(30);
  fill(50);
  noStroke();
  rect(-150, 400, 280, 40);
```

```
void ShowDepth() {
   fill(255,0,0);
   noStroke();
   rect(-500, 650, 300, 50);  // Adjust the position and size as needed
   fill(255);
   textSize(30);
   textAlign(CENTER,CENTER);
   text("Depth: " + Depth + " m", -500, 650, 300, 50);  // Adjust the position as
needed
}

void ShowAltitude() {
   fill(184,134,11);
   noStroke();
   rect(-500, 700, 600, 50);  // Adjust the position and size as needed
   fill(255);
   textSize(30);
   textAlign(CENTER,CENTER);
   text("Altitude: " + Altitude + " m above mean sea level", -500, 700, 700, 50);  //
Adjust the position as needed
}
```