ANERIS

UVAS

Underwater Video Acquisition System

Giorgio Salvemini s351995
Christopher Kaba s352118
Henrik Enger Reimers s352109
Benjamin Gabriel Havnen s348911

A thesis presented for Bachelor Degrees in Mechanical & Electrical Engineering

Department of Mechanical, Electronic and Chemical Engineering
Faculty of Technology, Art and Design
Oslo Metropolitan University
Oslo, Norway
2023

Abstract

This report presents the process of building a low-cost, lightweight underwater observatory. This includes an underwater camera, hydrophone, and sensor station connected by cable to a central hub stationed on land which forwards all relevant data to designated viewers via internet.

The underwater observatory is intended to be a proof-of-concept of a cheap, open-source, and easily reproducible system that has as few tool requirements as possible. This hopes to ensure that the majority of interested parties are capable of building a similar system without needing to invest sizable capital in custom part orders and/or tools.

The main contributions of this project unfolds in three parts;

1. Providing power and communication to the submerged platform through a tether cable.
2. A low-cost platform to house the camera enclosure, sensors, and lights.
3. An anti-biofouling mechanism for the camera viewport to combat marine growth and reduce maintenance.

The main objectives are to, at a minimum, successfully provide live audio, video, and exterior lighting that can be activated remotely, with possible additional telemetry including water temperature, pressure, pH level, salt levels and similar.

Acknowledgements

We would like to express our deepest gratitude to the following individuals for their invaluable support, guidance, and encouragement throughout the completion of our bachelor project.

First and foremost, we would like to extend our sincere appreciation to our supervisors, Ivar Saksvik and Rafael Borrajo, for their guidance and expertise. Their continuous support, feedback, and patience have been instrumental in shaping the direction and quality of this project. We are all grateful for the invaluable lessons we received under their supervision.

Our heartfelt thanks go to our families and friends for their unfaltering support and encouragement throughout this journey. Their belief in our abilities and their constant motivation have been a source of encouragement. We are grateful for their love and understanding.

Lastly, we would like to express our gratitude to all the authors and researchers whose work we have referenced in this thesis. Their contributions have paved the way for our own research and have shaped the broader academic discourse.

In conclusion, we are immensely grateful for the support and contributions of all those mentioned above. Their collective efforts have played a significant role in the successful completion of this bachelor project.

Contents

Glossary of Terms

[Alphabetical Order]

Acronym	Term
ABS	Acrylonitrile butadiene styrene
CSS	Cascading Style Sheets
EMUAS	Expandable Multi-imaging Underwater Acquisition System
EPS	Extracellular Polymeric Substances
FOV	Field of Vision
GPIO	General Purpose Input/Output
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
KSPS	Kilo Samples per Second
LED	Light Emitting Diode
LSB	Last Significant Byte
MBS	Methyl methacrylate-butadiene-styrene
MJPEG	Motion Joint Photographic Experts Group
MOSFET	Metal Oxide Silicon Field Effect Transistors
OMB	Operational Marine Biology
OBS	Open Broadcaster Software
PC	Polycarbonate
PE	Polyethylene
PET	Poly Ethylene Terephthalate
PLA	Polylactic Acid
POM	Polyoxymethylene
PVC	Polyvinyl Chloride
PWM	Pulse-width Modulation
SG	Specific Gravity
UI	User-Interface
UDP	User Datagram Protocol
UV	Ultraviolet
WAV	Waveform Audio File Format

Institution / Organisation	Acronym
ANERIS	operAtional seNsing lifE technologies for maRIne ecosystemS
ICM	Institut de Ciències del Mar
CSIC	Consejo Superior de Investigaciones Científicas
JAMSTEC	Japan Agency for Marine-Earth Science and Technology
UN	United Nations

Contributions

Student (Student Id)	Written Contribution	Technical contribution
Giorgio Salvemini (s351995)	Materials - Hardware, Electrical Components Price List Methods - Software - Illustration 7.13 - Software Logical Diagram, The Control software, The Web Server, The Porpoise Bit Conversion Library Methods - Final Assembly Discussion - Cost Analysis, Issues Conclusion	Developed Control Software, Video Streaming Software, Porpoise Audio Conversion Software Sourced Electrical Components Designed wiring diagram Final Device Assembly
Christopher Kaba (s352118)	Introduction - Glossary Background and Literature Review - Background, Existing Solutions, Biofouling - Securing the wiper/brush Methods - Dome Wiper Design Discussion - Results Analysis, Comparison between theory and other solutions on the market Challenges - Conclusion	Wiper Concept - Tether Strain Protection - Wiper Brush Enclosure Seat - Inner/Outer Wiper Design - Waterproofing of Parts - Ballast Distribution - Sourcing and fitting of wiper magnets Camera Mount Design - Wiper Motor Code
Henrik Enger Reimers (s352109)	Background and Literature Review - Background, Existing Solutions Materials - Intro / specifications Methods - Flowchart of the Solution, The Layer Cake Concept, Dome Wiper Design, Acrylic paint application, Final Assembly Results Discussion - Cost Analysis Conclusion	Concept for the rig Concept For Wiper Designed Motor Driver Circuit Board, deployment hook, hydrophone adapter Designed inner wiper, outer wiper, outer wiper holder Designed mounting plate, bottom flange, motor mount
Benjamin Gabriel Havnen (s348911)	Future improvements Background and Literature Review - Materials Research and Analysis Methods - Tripod Design, The Wedge Concept Software - Client Discussion - Future Improvements, Reproducibility Conclusion	Developed Client Software Software - Listing 7.1-7.7 Dome Wiper Design - Illustration 4.13 - Wiper mechanism The Wedge Concept - Illustration 4.7 7.2 - 3D-Render of V3.1 Concept For The Rig

Chapter 1

Introduction

Underwater observation platforms play a critical role in studying and understanding the ocean environment. They provide valuable data and insights into the condition of the sea, and allow scientists to monitor and analyze the behaviour of marine life and their interactions with the surrounding habitat. Current platforms however, require a substantial investment for a finished product, or are custom built on a case-by-case basis.

The intention of this bachelor thesis is to design and build an innovative solution that addresses the issues of expensive underwater observatories. The design should be durable, but above all cost-effective, reducing the barrier to entry of operating such platforms to the point where solo operators or small teams can afford to build one without needing the resources available to, for example: a large university or research company.

The question posed by this thesis is:

> *How can an underwater observation platform be designed and built at heavily reduced cost - relative to existing platforms - while still providing substantial data from approximately 100-150 meters below the surface of the sea.*

The study objectives are to:

1. Design and Build a prototype of an underwater observation platform that meets the latter requirements.
2. Evaluate the performance and functionality of the components that comprise the prototype.
3. Develop a cost-benefit analysis of the prototype platform when compared to existing observation platforms

The methodology used for this study will involve a combination of literature review, prototyping and field testing. A literature review will be used to gather information on currently available platforms, systems and technologies that may relate to platform use-cases. The majority of the time will be spent designing and developing prototype platforms.

Finally field tests will be undertaken to evaluate the performance and functionality of the platform. The findings of this study will be valuable to scientists, engineers, and other parties interested in underwater observation on a budget.

In conclusion, this bachelor's thesis aims to develop a new solution capable of addressing the issues underwater platforms deal with, but at a substantially lower cost. The objective in doing so is to contribute to the expansion of the field of underwater observation.

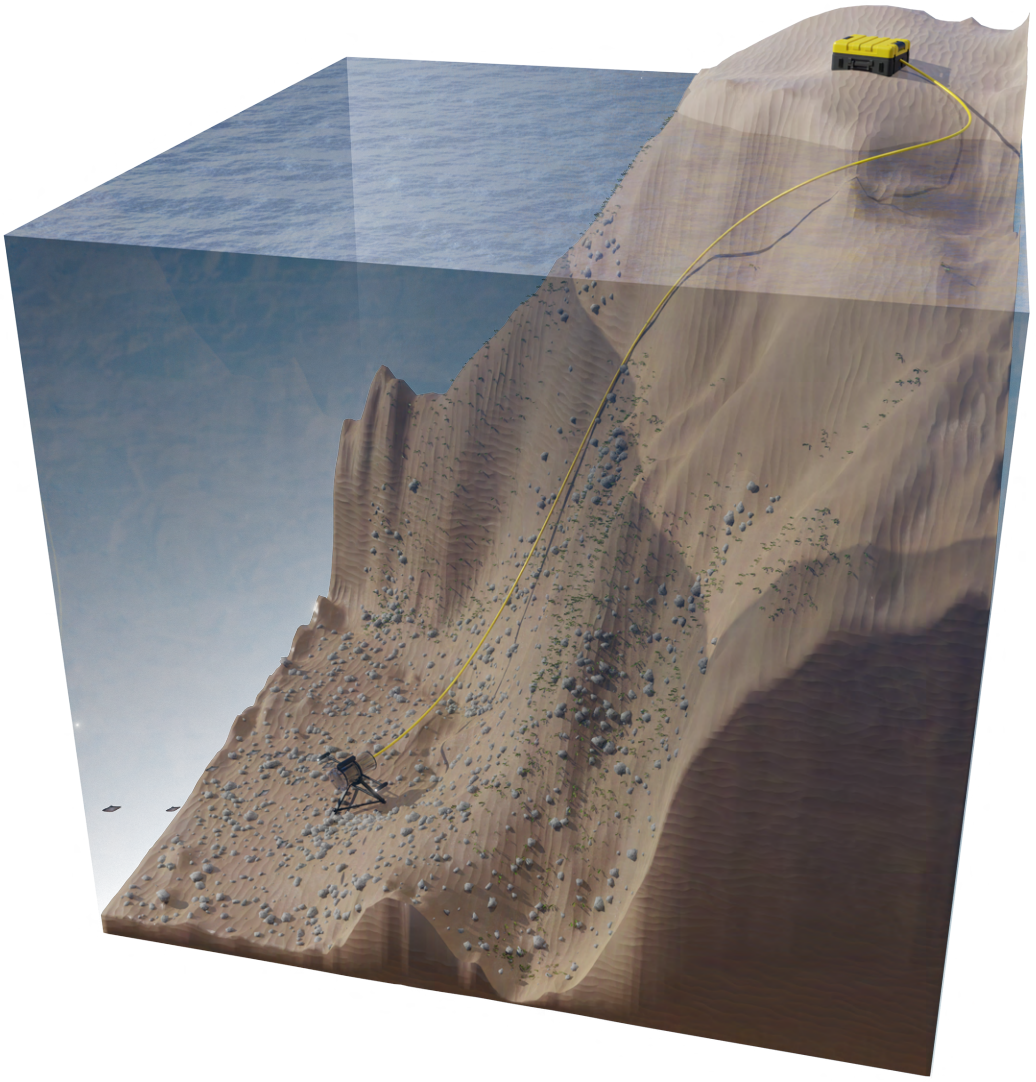


Figure 1.1: *Overview of the proposed complete solution (Pierre Boniface).*

Chapter 2

Background and Literature Research

2.1 Background

The ANERIS project (operAtional seNsing lifE technologies for maRIne ecosystemS) is led by ICM-CSIC in Spain and includes 26 international research and industrial partners. The project proposes to develop the next generation of scientific instrumentation tools and methods for sensing marine life. The design of the new instruments and methods will integrate different types of marine life-sensing technologies: genomics, imaging-bio optics, and participatory sciences (ANERIS 2023).

The technologies will be implemented in a co-design framework, involving all the interested stakeholders: academia, industry, civil society, and government. The project proposes the concept of Operational Marine Biology (OMB), understood as a biodiversity information system for systematic and long-term routine measurements of the ocean and coastal life, and their rapid interpretation and dissemination.

OsloMet is co-leading one of the project work packages and will be responsible for the development of an Expandable Multi-imaging Underwater Acquisition System (EMUAS). This is a low cost cabled underwater camera system that will obtain real-time video of underwater sites and will be used to collect marine biological and environmental data information. The system will be tested and demonstrated in OsloMet Oceanlab, and on Malmøya.

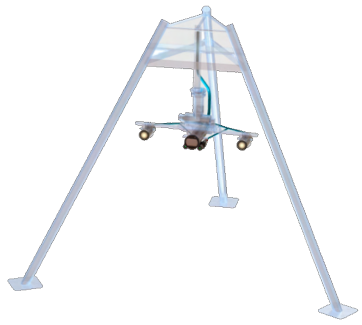
2.2 Literature Review

The use of underwater observation platforms has become increasingly important for a variety of applications in recent years. From marine biology, and oceanography, to port security, and environmental monitoring, remote autonomous platforms can perform in roles where a human simply cannot remain for extended periods of time.

To better understand the current state of the technology and research in this field, a literature review was conducted to examine existing knowledge and trends. This review aims to provide a comprehensive overview of the key findings and insights from previous studies, and to identify topics related to ANERIS design challenges.

2.2.1 Existing Solutions

SubC Imaging - Observatory Camera System

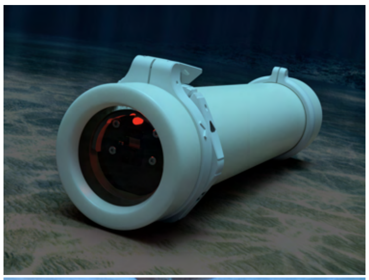


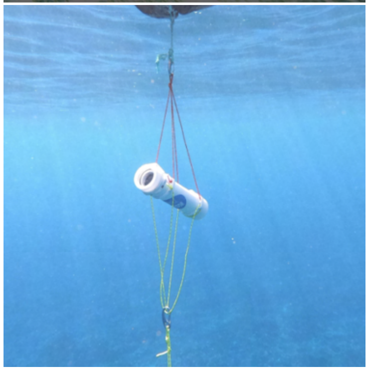
SubC Imaging is a company that specialises in underwater camera systems, software and systems integration of submersible platforms. Their clients range from research institutes, to military departments.
SubC Imaging are currently developing a streaming IP camera (named: the Viperfish) which would be suitable for the needs, however the starting price point of $10,000 is more than is intended to be spent on the full platform build (Smith 2023).
While high-tech systems such as those designed by SubC or similar manufacturers have shown great promise in fields such as deep sea surveying, aquaculture, and military applications, it is precisely because of these cutting-edge systems that are the motivation to develop a more practical and cost-effective platform that can still meet the needs of ocean floor researchers.

Hangzhou Chenquan Tech - Maritime HD Underwater Camera

Chenquan is a company that produces specialized vision systems for both low-light and underwater situations. One of their camera systems provides a 260°rotational view, and can operate anywhere from 50-2000m below sea level, depending on client demands and budget (Liu 2023).
Other models include attached lights and/or wipers to keep viewports clean, which are functionalities to be incorporate into the design however, their systems far surpass the needs and very likely overshoots the budget by a similar margin to SubC's Viperfish.

Maka Niu - A low-cost, modular imaging and sensor platform





The Maka Niu is a small, open-source, low-cost, modular imaging and sensor platform to increase observation capabilities of the deep ocean. It utilizes readily available commercial hardware while leveraging mass production to reduce the cost-per-unit, in order to provide more communities with the ability to explore regions of the deep ocean that were previously inaccessible due to the current costs of underwater imaging platforms (Novy et al. 2022).
The Maka Niu uses a mass-produced delrin housing which is rated for up to 2000m depth, and a 3D printed interior structure to house a Rasberry Pi Zero W, a Pi Cam v2 and accompanying wireless charging and communication equipment. It is an ingenious little system, however, due to the fact that it is designed for temporary deployment with a battery capacity that allows it to operate for approximately 20 hours (Novy et al. 2022), biofouling was not a major issue taken into consideration as the view port could simply be wiped down upon retrieval.

KOSMOS: An Open Source Underwater Video Lander for Monitoring Coastal Fishes and Habitats

The KOSMOS lander is a continuation of an already existing solution called STAVIRO. The STAVIRO lander is a proven lander with over 10 years of different deployments. A drawback with the STAVIRO lander is the use of proprietary parts. This makes it hard to mass- produce and it has a higher cost associated to it. The KOSMOS lander aims to be low cost, reproducible and have full open-source documentation.

The parent system consists of two watertight housings connected by a stainless-steel axis. The upper housing is a plexiglass cylinder with a flat window at one end, and at the other end, an aluminium lid secured by stainless steel crews and bolts. The lower housing encloses a motor with electronics and a battery. The motor is related to the stainless-steel axis that enables the upper housing to rotate at programmed angles and timings. The device is fixed on a three-leg aluminium stand used for deployments. The stand is equipped with an intermediate buoy that tenses the rigging to avoid mingling with the upper housing (Pelletier et al. 2021).

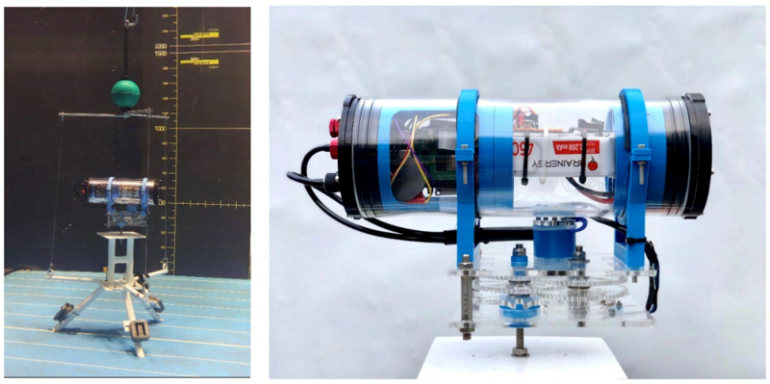


Figure 2.1: The KOSMOS prototype on it's support and with the rigging (left). Side view of the housing and reducer (right) (Pelletier et al. 2021).

Table 2.1: Comparison of existing image/video capture systems.

	SubC & Chenquan Tech	**Maka Niu**	**KOSMOS**
References	SubC-Imaging Chenquan Tech	Novy et al.	Pelletier et al.
Camera	Proprietary	8Mp RasPiCam v2	12Mp RasPiCam HQ
Resolution	4K(Fibre), HD(Eth/Coax)	1920 x 1080 pi	1600 × 1200 pi
FoV	72°x 48°	62.2°x 48.8°	60°(Rotatable)
Storage	512GB - 1TB	120GB	64GB
Runtime	Cabled	≈ 20 hrs	≈ 8 hrs
System Weight (Land/Water)	Not Specified	≈ 870g ≈ -50g	≈ 7 kg ≈ 6.2 kg
Dimensions	Variable by Usecase	∅66mm x 261mm	∅300mm x 101mm
Cost (€)	≈€9000 (Variable)	€300-900[1]	€1360
Documentation for Reproduction	Unavailable	Contact authors	Online[2]
Existing Applications	Commercial Aquaculture, Military, Port Observations	Widespread testing (See Novy et al. 2022)	Experiment in paper
Typical Use	Stand-alone Deployment Integrated to Systems	Diver deployed	Autonomous Lander
Anti-Biofouling Measures	None Specified	None	None

[1] Design intended for mass production, estimates assume a production qty. of ≈ 10000 for the lower cost and ≈ 20 for the higher. Estimates do not include cost of labor.
[2] `https://wikifactory.com/@konkarlab/kosmos30`

2.2.2 Biofouling

As seen in Table 2.1 none of the researched devices have considered any anti-biofouling measures as they are predominantly focused on short-term deployments (Supported by the fact that these systems are battery driven with commercial products being an exception.), and companies providing related systems were understandably unwilling to share measures taken to protect their systems. Due to the fact that one of the main goals of this project is to design a system for long term deployment, a low cost anti-biofouling measure must be designed and implemented.

Literature on biofouling generally recognizes two main phases that contribute to the accumulation of organisms and materials on submerged surfaces: micro- and macrofouling.

Microfouling is characterised by the development of a thin layer of micro-organisms and is commonly known as biofilm, of which extracellular polymeric substances (EPS) is considered the primary matrix material. (Approximately 50-90% of organic carbon in biofilm is composed of EPS.) The biofilm layer adheres to submerged surfaces and provides a chemical signal to organisms that this surface is a viable area of colonisation, while also acting as an adhesive to aid in attachment (Martin Wahl 1989, p. 175-176).

Macrofouling refers to the attachment and growth of larger organisms (such as barnacles, mussels, seaweed and similar marine plants and animals) on submerged surfaces. Macrofouling can have significant impact economically including but not limited to: increased drag and fuel consumption for ships, added weight causing structural damage on structures and marine infrastructure, clogging water intakes, and fouling aquaculture equipment (Martin Wahl 1989, p. 176-177).

While macrofouling may negatively affect the structure of the platform, the main concern with biofouling is the potential obscuring of the camera's FOV. Since this can begin in the first hour of the platform's deployment, prioritizion will be given to implementing preventative measures to combat microfouling with the hopes that by removing the EPS layer, the amount of macrofouling that will occur throughout the deployment can be reduced. By taking early action, significant biofouling buildup will be prevented from obscuring the camera lens, with minimal interference of the platform's primary objectives.

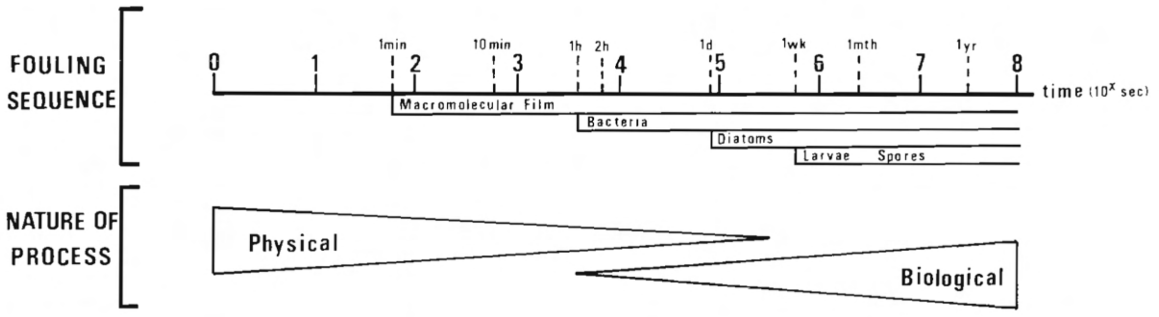


Figure 2.2: *Representation of the biofouling process showing the initial adhesion of microorganisms, followed by biofilm growth and the subsequent attachment of larger organisms (Martin Wahl 1989, p. 176).*

2.2.3 Materials Research and Analysis

Metallic Material For Underwater Use

When deploying a device underwater, especially at great depths, a cylindrical container is strongly recommended, as it can better sustain the high water pressure and density. Additionally, the greater the depth, the thicker the walls of the hull will need to be, to maintain the device's integrity. One side effect of this is an increase in weight as the hull becomes able to sustain greater depths (Hyakudome 2011). This is not ideal if the submerged device needs to be moved often, however, as this project aims to keep it stationary, the only time weight would be an issue is during setup and retrieval.

Aluminium has a specific gravity (SG) of 2.7(Solids and metals - specific gravities), making it sinkable when compared to the oceans SG between 1.02 and 1.03 (Boyd 2020). It is also resistant to low temperatures, as having a material that can handle the low temperatures at 150m depth is an essential characteristic.

When it comes to the topic of corrosion, there are eight kinds of aluminium series, from 1000 to 8000. The Japan Agency for Marine-Earth Science and Technology (JAMSTEC), use series 5000 to 7000 Aluminium alloys, each with their own characteristics and uses. It is the corrosion resistance that is of importance to the project, as aluminium alloy has a high risk of corrosion. 5000 has a higher corrosion resistance than both the 6000 and 7000 series, however the strength is lower (Hyakudome 2011).

There is also the 2000-series, which is an alloy created of Aluminium and copper, mainly used in aircraft structural aspects, where the main criteria is damage tolerance. There are different variants of the 2000-series, with the variants of the 2000 series containing magnesium being stronger than the variants without whilst also having a higher resistance to material failures like fatigue crack growth compared to other alloys (Dursun and Soutis 2014).

According to Dursun and Soutis 2014, p 862-871, Section 2: "The 2000 series alloys containing magnesium have higher strength resulting from the precipitation of Al2Cu and Al2CuMg phases and superior damage tolerance and good resistance to fatigue crack growth compared to other series of aluminium alloys."

Unfortunately, the 2000 series chemical composition gives not only a heightened strength and damage tolerance, but also reduces the corrosion resistance considerably (Dursun and Soutis 2014). Given that the project will be used on the seabed for a considerable length of time, it can be concluded that using the 2000 series of aluminium alloy is unwise.

The 6000-series Aluminium alloy is as previously stated right in the middle of the 5000 and 7000 with it's boundaries and strengths. It is stated that this alloy has excellent strength and high corrosion resistance, making it rather ideal for maritime applications. Within the 6000-series are a number of ideal material candidates, like the 6061, which is quoted to have a much greater corrosion resistance even on chafed surfaces (TABER 2023).

Using Plastic for underwater use

The use of plastic is a solution that both removes the material degradation aspect that plagues the vast majority of metals when submerged in water, as well as the need to find a waterproof and water resistant coating. Plastic is highly durable, and as is known from the rising water pollution, has a long lifespan, even when applied to certain chemicals. (see (Ltd 2019)) A problem that can occur when using plastic is it's buoyancy affecting the total bouyancy of the observatory, as the majority are either positively or neutrally buoyant in seawater. An example is Polyethylene (PE), with the exceptions of polyethylene terephthalate (PET) and polyvinyl chloride (PVC) (Andrady 2015). Given the vast majority of relevant versions of plastic, focus will be on one of the three most commonly used plastic types (PE, PVC, and Acrylonitrile Butadiene Styrene(ABS)), mainly ABS, and Acetal (polyacetal and polyoxymethylene (POM))

ABS and its recyclability

ABS plastic is an engineering thermoplastic polymer. It is amorphous, meaning it has no clearly defined shape or form (Omnexus 2023). There are two methods of creating structures or products out of ABS; injection molding, and printing, with printing being the most affordable, especially if access to a printer is already available. The cost of injection molding can be anywhere between \$3,000 to \$100,000 (Losek 2022), and the cost of an ABS spool is around 199NOK to 1399NOK (3DNet 2023), If a company were to mass-produce the structure around the enclosure, then creating a mold for the parts could be an easier and more time efficient method than printing. However, if making two or more structures, as a hobby or as a small business, then purchasing a printer that handles ABS or ordering from a 3D-print manufacturer would be equally effective, like the *Ultimaker 2+(Connect)*, which is ABS compatible (UltiMaker 2023a).

Another positive is the recyclable property of ABS plastic, as aiming towards a greener solution is considered a must if considering the UN's sustainability goals (UnitedNationsAssociationOfNorway 2023c), especially number 12 (UnitedNationsAssociationOfNorway 2023a), and 13 (UnitedNationsAssociationOfNorway 2023b). Recycled ABS can also be mixed into new material in order to produce a cheaper alternative with the same quality (Omnexus 2023, ABS Recycling and Toxicity).

A problem that can occur when recycling ABS is that the material loses is impact strength during service time as a result of oxidative degradation (Xiaodong Liu 1999). If this where to happen, it would become a problem for products and uses in non static situations. Knowing this and given that the use of ABS would primarily be in a static underwater situation, i.e posted on the seabed, should this problem occur, the odds that it would become an issue is as of now minimal. The probable cause of that impacts would be when the structure hits the seabed, and if something were to fall onto it.

According to Xiaodong Liu 1999, Introduction, para.4, a solution would be to blend the old ABS material with Polycarbonate (PC):

> "Previous work showed that the impact strength of artificially or naturally aged ABS could be improved by mixing with 20% of neat PC and 5-10phr methyl methacrylate-butadiene-styrene (MBS) core-shell modifier. The Charpy

impact strength of this system was 20 to 60% higher than those without core-shell modifier."

With the knowledge given, solutions to the possible obstacles and a method that uses ABS plastic can be prepared. In terms of production, new ABS plastic for prints and molds can be ordered, and after a while, recycle the material to be used in new prints and molds, or send it to other companies, so they can be use in their productions, maintaining a greener outcome as a bonus.

Acetal(POM)

Another thermoplastic suitable for use under water is Acetal plastic. The plastic is referred to as polyacetal and polyoxymethylene (POM) and has no specific application area. This allows the plastic to be used for a number of functions, including valves. It is most often used for parts that maintain rigidity, low friction and good dimensional stability (ISM 2023). The benefits of using an acetal plastic includes: excellent dimensional stability, resistance to friction, fatigue, and abrasion, and low moisture absorbance. (Emco 2023b).

When considering the types of acetal plastic. There are two different types available: Homopolymer and copolymers. Each of these acetal plastics has their own strengths and weaknesses. Homopolymer has a higher flex modulus and impact strength at room temperatures, with its impact strength being elevated by lower temperatures and its flex modulus being affected by higher temperatures. Copolymer has an improved stability due to its low crystallinity, and a higher chemical resistance with bases (ModernPlastics 2016). Copolymer is most commomly used for equipment in contact with either food or medicine (Emco 2023a). Due to this information, copolymers can be excused from the equation, and focus more on it's counterpart, homopolymer.

A common homopolymer acetal plastic is Delrin, considered to be the first commercial acetal plastic (StoreNorskeLeksikon 2017). Delrin retains the same properties of other acetal plastics, but with one main difference in it's composition. The composition of Delrin is described as a "uniform crystalline structure", and allows the plastic a high flex fatigue resistance and stiffness, as well as mechanical strength. Due to Delrins benefits, it's wildly used in a wide range of applications, and due to it's previously mentioned physical properties, Delrin, as well as other acetal plastics has the capability to replace metallic components (Emco 2023b), which makes it a contender for a suitable alternative aluminium profiles.

An example of products made out of Delrin are, amongst others the Maka Niu (Novy et al. 2022), and garden hose nozzles (Emco 2023b), which are both products that experience prolonged exposure to water. A reasonable assumption can be drawn regarding it's viability for deployment in maritime structures. Delrin's mechanical properties render it an appropriate substitute for the aluminum profiles. Furthermore, Delrin exhibits a striking resemblance to other homopolymer acetal plastics, thereby allowing for easier interchangeability between these materials, should the need or desire arise. (Emco 2023b).

Materials Research and Analysis Conclusion

Using the research, and taking cost and time into account, the decision was made to use Aluminium profiles, of the 6000-Series (RatRig 2023), as it could easily be acquired from a local hardware store, without expensive shipping. Future solutions to corrosion can be taken into account when implementing. When producing plastic parts, ABS brings several benefits thanks to its sustainability, and recyclable properties.

Chapter 3

Materials

3.1 Intro/Specifications

The stationary rig is made out of many different parts. This chapter will provide a description of the materials and hardware used in this project.

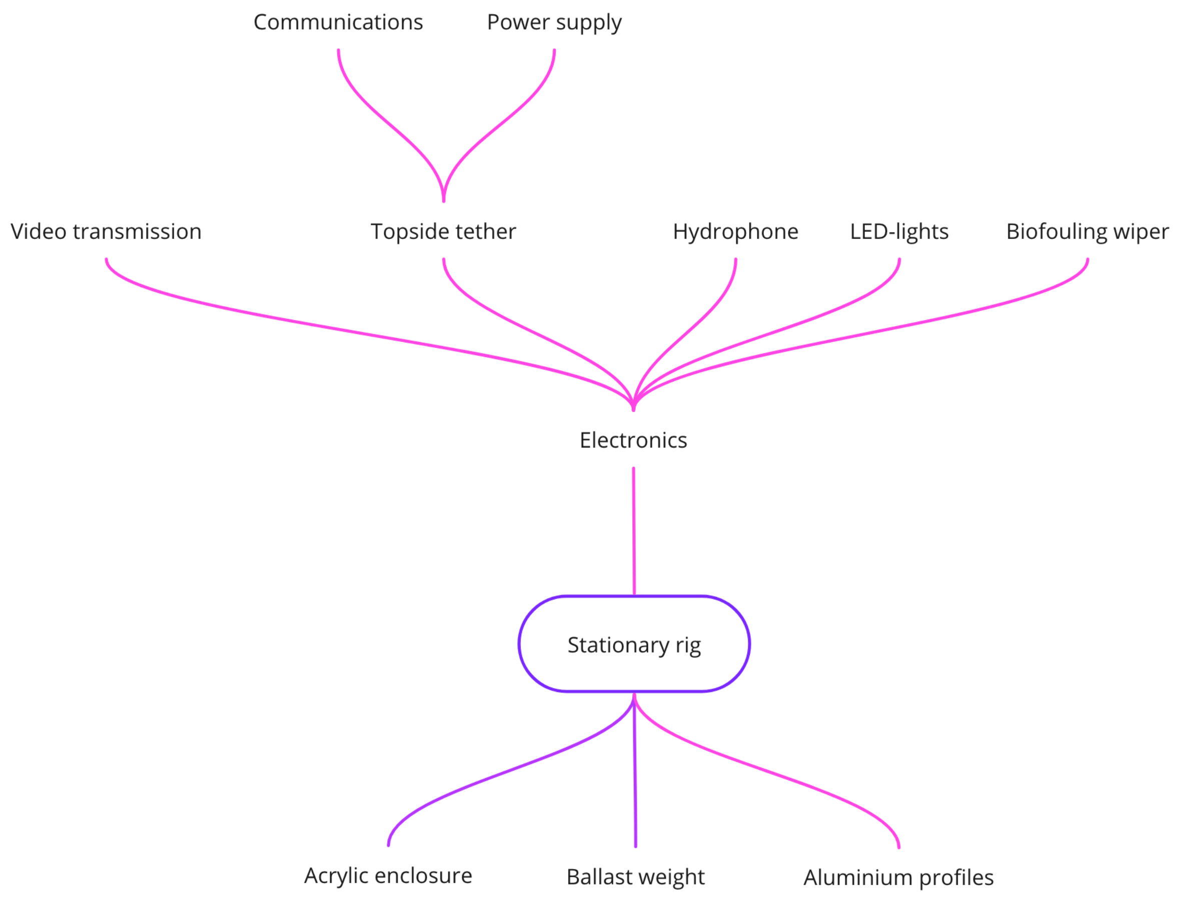


Figure 3.1: *Flowchart of the main groups of materials used in this project.*

3.2 Hardware

3.2.1 Aluminium profiles

Extruded Aluminium Profiles, 12 pcs,
20mm x 20mm x 500mm
Comprises the main structure, with rails for external attachments

3.2.2 L-brackets

L-brackets, 8 pcs, ideally made of stainless steel but can be spraycoated for some additional protection.
Used to secure each corner of the structure and possible external attachments.

3.2.3 M5-8mm screws

Low-Profile Screws, 25 pcs, ∅5mm x 8mm
Used to secure structure corners and l-brackets.

3.2.4 M2-8mm screws

Screws, 11 pcs, ∅2mm x 8mm
Used to secure the enclosure in its seat.

3.2.5 M5 T-nuts

T-nuts are designed to slide into the aluminium profile rails. Used in conjunction with the M5 screws to secure external attachments, 25 pcs.

3.2.6 Brush

The nylon brush used is sourced through IKEA's spare parts program.
Dimensions: 6.8mm x 200mm
Bristle Length: 8mm
Part Code: 121046 (ikea.com 2023).

3.2.7 Enclosure

The enclosure used in this project is a watertight enclosure made by Blue Robotics. It consists of several parts: a 200mm long acrylic cylinder with a diameter of 100mm; a dome end cap mounted to a sealing flange, which is fitted to one end of the cylinder; an aluminium end cap with five holes, allowing for cables to penetrate the cylinder, which is also mounted to a sealing flange; and a vacuum port, allowing for vacuum testing (*Enclosures, Buoyancy, and Ballast Archives* 2023).

3.2.8 Molykote

Molykote is a lubrication specially designed for use on o-rings holding a vacuum seal, this is applied to all seals on the enclosure, and on the rear connection on the porpoise hydrophone. A sachet of molycote is provided with the Blue Robotics enclosure. (Molykote 2023)

3.2.9 Hardware price list

Component	**Unit price**
Aluminium profiles	NOK 999,00
L-brackets (4 packets)	(per piece) NOK 59,90
M5-8mm screws (3 packets)	(per piece) NOK 59,90
M2-8mm screws (1 packets)	(per piece) NOK 59,90
M5-T-nuts (3 packets)	(per piece) NOK 69,90
Brush	NOK 0,00
Enclosure	NOK 3.514,83
Total	NOK 5.142,83

N.B.: prices valid as of May 25th 2023

3.3 Electrical components

To power the underwater observatory, several pieces of computer hardware are needed to perform the various required tasks, such as recording and transmitting video/audio, illuminating the surroundings, moving the wiper mechanism, and powering every internal component.

3.3.1 Communication

Ethernet switch

A 5-port TP-Link fast ethernet switch is used for communications, allowing the connection of the Raspberry Pi and the Porpoise hydrophone to the topside computer (TP-Link 2023).

Blue Robotics Fathom-X

The Fathom- X boards are used for both communication and power delivery. The boards are produced by Blue robotics, and allow for data transfer rates of up to 80Mbps, at a distance of up to 300m. These devices support the IEEE 802.3 standard, i.e. Ethernet. In combination with the Blue Robotics tether cable, the Fathom-X boards enables the use of a single wire for everything, rather than splitting data and power across multiple cables. The board is mounted inside the device itself, while the Tether interface (the blue box) stays on land and connects to a computer through usb, and a power supply (BlueRobotics 2023a), (BlueRobotics 2023b).

Tether cable

The Tether cable, produced by Blue Robotics, is a 150m long, kevlar-reinforced, underwater cable containing 4 pairs of unshielded twisted cables (UTP). This cable also has cross-talk resistance, making it ideal for data transmission and power delivery. The cable comes with a penetrator built-in on one side, allowing for easier assembly to the device, as well as a quick connect plug on the other side, allowing the user to easily connect it to the Fathom-X. The cable is also neutrally buoyant, making it well suited for underwater rovers, although less desirable for this application, as the cable won't lay on the seabed without extra weights (BlueRobotics 2023c).

3.3.2 Power

Power supply

A step-down, 48V to 24V power supply distributes power to the underwater observatory. It can output up to 15A at 24V (up to 360W), which is far more than the average load of the platform, making it upgradeable for additional components in the future, if needed. The outer enclosure has a ribbed design and is made of metal, allowing for more effective cooling of the internal components (Amazon 2023).

Buck-boost converter

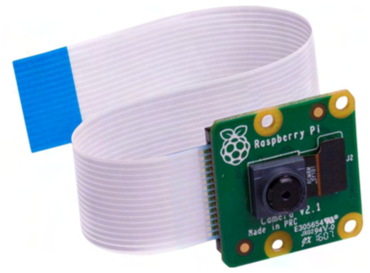
Two buck-boost converters are used to step down the 24V supply to 5V. As these modules can only output up to 2.5A (regardless of the chosen output voltage), two of them are necessary, one to supply the Raspberry Pi, and the other to supply both the Arduino Nano, and the ethernet switch (Kjell 2023).

3.3.3 Video transmission

Raspberry Pi

The main computer in the submerged system is a Raspberry Pi 3b+. It has 40 builtin GPIO pins, allowing control of components such as the lights and the wiper. It has a 1.4GHz, quad-core processor, 1GB of ram, and 4 USB ports (RaspberryPi 2023b). The RP3 does everything from acquiring and sending the video feed, to interacting with the Arduino to control the wipers and controlling the lights .

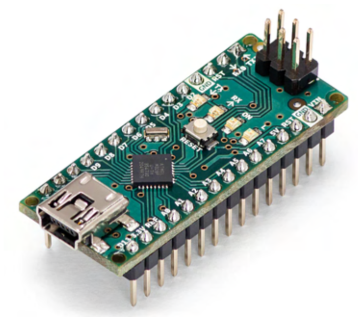
Camera



A Raspberry Pi camera V2.1 is implemented to record video. It uses a Sony IMX219 8-megapixel sensor, allowing the recording of high-definition video. It also performs reasonably well in low-light environments, an important criteria for the submerged observatory. This camera connects directly to the Raspberry Pi using a MIPI CSI-2 connector (RaspberryPi 2023a).

3.3.4 Wiper mechanism

Arduino Nano

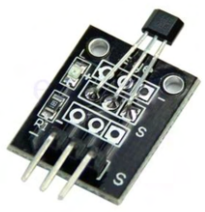


The Arduino Nano is a development board with a very small footprint. The board is driven by the ATmega328 microcontroller and offers 22 digital input/output pins as well as 8 analog pins (Arduino 2023). Its purpose in the platform is to drive the wiper.

Continuous servo motor

A continuous servo motor is used drive the wiper. The model being used is the FS90R from Feetech. It is low-cost and readily available worldwide (AdaFruit 2023b). The integrated motor driver of the FS90R enables a change in the driving direction and control the speed via PWM. The servo motor receives control signals from the Arduino Nano.

Hall effect sensors



A pair of hall effect sensors of the model KY-003 is used in the wiper mechanism (ArduinoModules 2016). These sensors are used to limit the movement of the wiper by signalling to the Arduino Nano when it has reached the end of it's rotation. This is achieved by letting the sensor come close to the wiper's magnets at the end of its rotation. Two different types of sensors are used, one which detects the presence of a magnetic field, while the other detects the polarity of said magnetic field.

3.3.5 Other components

IRF520 Mosfet

A IRF520 n-type mosfet is used to pull the reset (RST) pin on the Arduino to ground, essentially acting as a switch to turn the wiper off when desired. The gate pin is connected to a GPIO pin on the Raspberry Pi, which can to enable/disable the mosfet on demand.

Porpoise hydrophone

To record acoustic noise while underwater, a hydrophone built by Turbulent Research is integrated as an external instrument to the underwater observatory. It has multiple sampling rates, but the default is 96 kilosamples per second (KSPS), which correspond to a sampling frequency of 96 KHz. It outputs a raw stream of audio data over UDP, allowing the user to receive it using the provided TRAC application, or to develop their own application to receive and analyse it (TurbulentResearch 2023).

Lights

The Blue Robotics Lumen Subsea Lights are used as a lighting solution on the submerged platform due to attenuation of natural lighting. These lights are very bright, emitting over 1500 lumens at 15W, support a large range of voltages (10-48V), and can be controlled with a PWM signal in order to regulate their brightness (BlueRobotics 2023d).

3.3.6 Electrical Components Price List

The following table gives an overview of the total cost of the electrical components. It includes a total cost with and without the cost of the Porpoise Hydrophone, as this is an optional part.

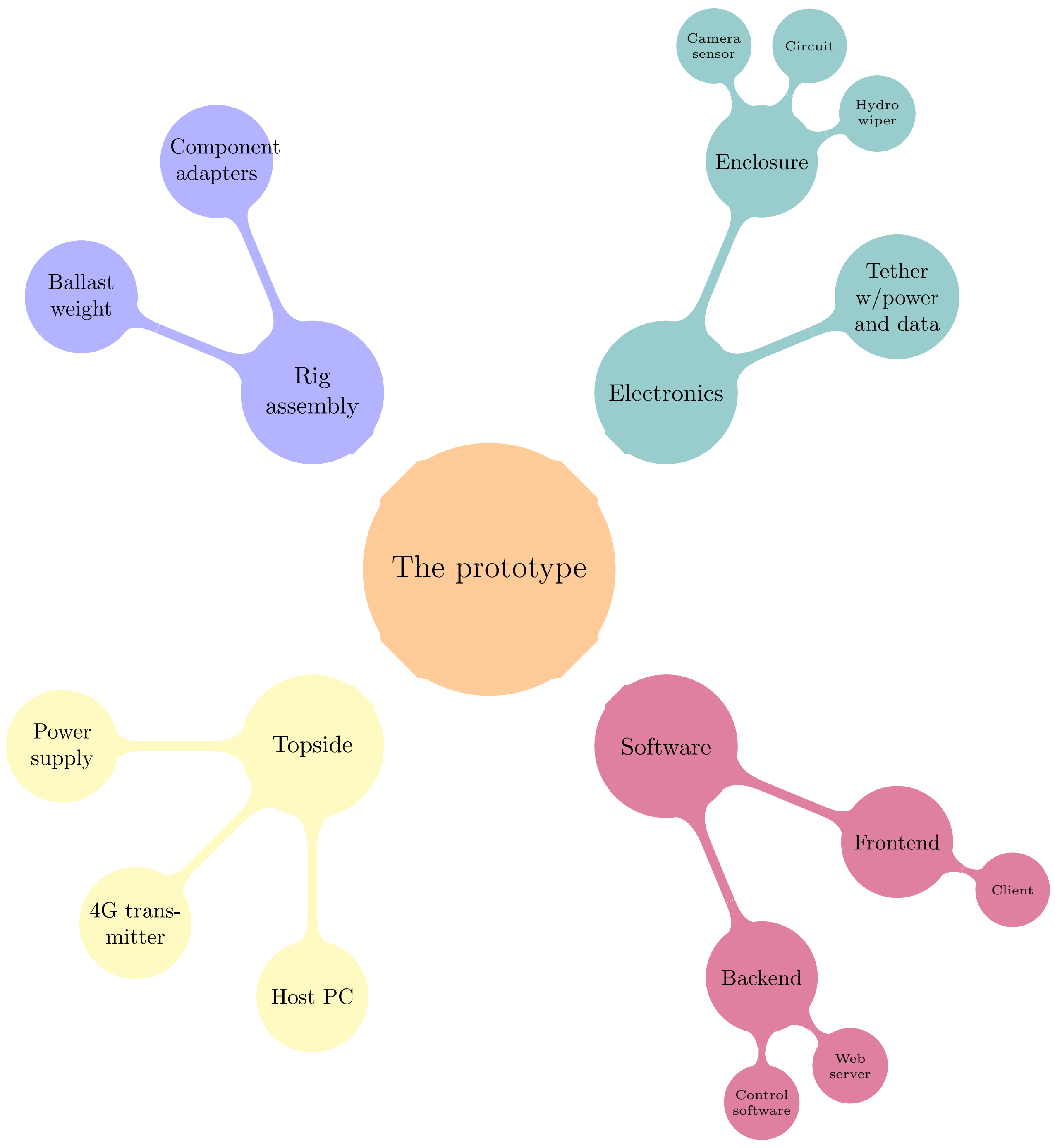
Component	**Unit price**
Communication	
Ethernet Switch	NOK 106,29
Fathom-X board	NOK 1.275,34
BR Fathom-X	NOK 2.125,74
Tether Cable	NOK 10.622,45
Power	
Power supply	NOK 280,06
Buck-boost converter (each)	NOK 99,90
Video transmission	
Raspberry Pi 3B+	NOK 507,92
Camera	NOK 283,68
Wiper mechanism	
Arduino Nano	NOK 349,00
Continuous Servo Motor	NOK 150,00
Hall effect sensor	NOK 80,90
Other components	
Porpoise hydrophone	NOK 71.833,83
Lights (each)	NOK 1.594,31
Mosfet (each)	NOK 12,96
Total (with porpoise)	NOK 89.503,18
Total (without porpoise)	NOK 17.669,35

N.B.: prices valid as of May 25th 2023.

Chapter 4

Methods

4.1 Overview of the underwater observatory



This chapter delves into the multifaceted aspects of this project, highlighting three key areas that have played a major part in the development of the prototype: construction of the stationary rig, designing an innovative wiper mechanism to combat biofouling, and the integration of comprehensive software solutions encompassing both front- and back-end systems.

4.2 Stationary rig concepts

In the process of designing a stationary rig, it is imperative to gain a thorough understanding of the environmental factors and parameters that will exert influence on the prototype. It becomes crucial to consider the aspect of stability, along with the ability to anticipate and address the necessary measures to mitigate or prevent any potential disruptions to the equilibrium of the product. This entails taking into account various elements such as ocean currents, both during descent and at the seabed, as well as the possibility of encountering marine life and debris that could obstruct, collide with, or entangle the product. Furthermore, when dealing with a heavy object, it becomes necessary to assess the soil's capacity for maintaining stability under the anticipated load. These considerations, among numerous others, form an integral part of the meticulous deliberations undertaken during the design process of the stationary rig.

4.2.1 Tripod Concept

The tripod design originated from camera tripods that are still used today in photography or film. Tripods have the ability to reposition their legs and adjust their length based on what is attached to the top of the tripod. Considering that the casing will be in a fixed position when placed, the adjusting aspect can be overlooked as of now. At each end there is a set of tripods which, when attached, would keep the enclosure stable, and if weight were pressed down, would lead the forces away from the enclosure, and further out. These six legs could stand on their own, or be affixed into a plate.

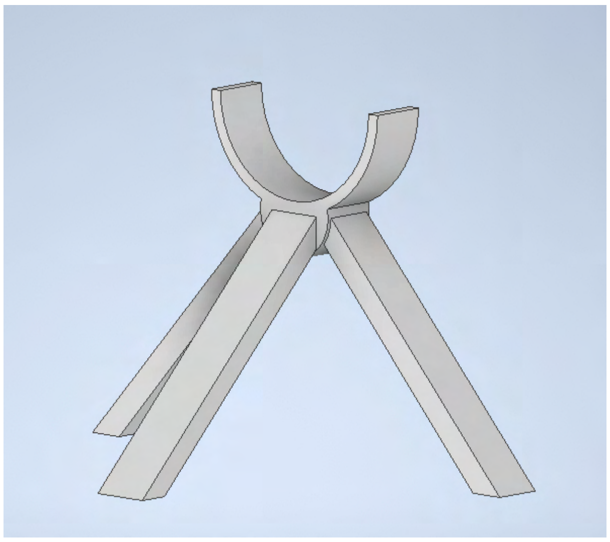


Figure 4.1: *3D-Render of Tripod legs*

Due to the inherent uncertainty associated with the seabed conditions in the oceanic environment, the specific nature of the foundation upon which a structure will be established remains unpredictable. The ground composition might vary, encompassing uneven surfaces, gravel deposits, muddy substrates, or areas with low soil density. In situations where mud and bog substrates are encountered, achieving an even distribution of weight becomes crucial and optimal for ensuring stability. Consequently, it becomes imperative to employ a configuration involving multiple posts, each possessing a small and sharp surface area. In light of this, the utilization of a plate is proposed as a viable solution, as it would effectively distribute the forces exerted by the structure, initially through the legs and subsequently across the entire surface of the plate. During concept testing a

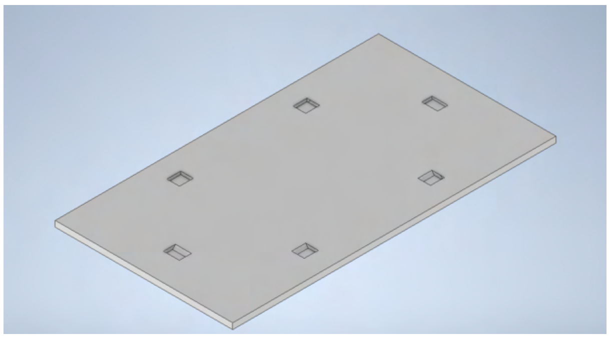


Figure 4.2: *3D-Render of plate*

slight issue was discovered. If the plate was absent, the exterior forces applied to the rear legs of each leg-component would send the front leg upward. Considering that the plates primary function was for use on muddy ocean floors, and not for other uses, it's necessity became insignificant, and was therefore removed. A secondary issue with this version was presented while testing weight tolerances, as this test would present a polar opposite problem. The front leg would be absent from the ground, supposedly due to a loose connection point, but would eventually hit the ground when weight was applied. The maximum recorded weight was 5kg, which was within the required weight tolerances, it suggests that the prototype had the opportunity to handle the forces the enclosure would exert.

Due to inconsistent results under testing, the decision was made to remove the front leg. This came with it's own problem, as the structure no longer had a supporting element that helped maintain balance. A solution was to redesign and add a bridge between the rear and front legs, that would use friction fit sockets to stay in place, and if necessary screws. This design was then named the Wedge, and would have three iterations based on this design, with V1 and V2 having the biggest fundamental changes, V3 having small but pivotal changes. It was a derivation of V3 that became the endpoint of this design, dubbed V3.1.

In the interest of printing a full scale model, a version designed with the spacial limitations of the 3d-printers length and width was made. Though printing this for deployment use, would introduce to many additional parts, which could lead to more possible points of failure. Were the Wedge concept to be used, the design would be ordered from an external manufacturer, in either metal or plastic, dependant on the requirements.

4.2.2 The Wedge Concept

First Iteration

The first iteration of this concept was simple in design, with the primary function being to test out ideas throughout the design process. This is most evident in the design of the bridge that connects the two sets of bipods. The bridge and the bipods have a semi-circle at the top to seat the enclosure, maintaining balance and stability. The bridge would be plugged into the bipod, with a large enough distance from the walls, so that pressure-based failure would not occur. In order to make the bridge and bipod easy to connect and stay fixed in the designated position, a male-female friction fit socket is used, with it's shape based on a parallelogram, due to the female end of the connection's space limitations. The socket's corners were rounded to minimize manufacturing errors due to tolerances when 3D-printing, that could make connecting the bridge to the the bipod difficult without tools.

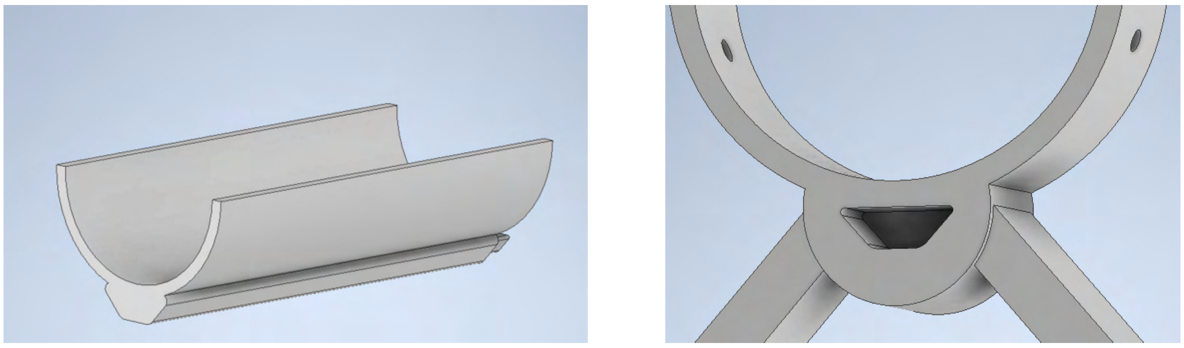

(a) *V1 bridge* ***(b)*** *V1 female socket*

Figure 4.3: *3D-Render of V1 male-female socket*

Second Iteration

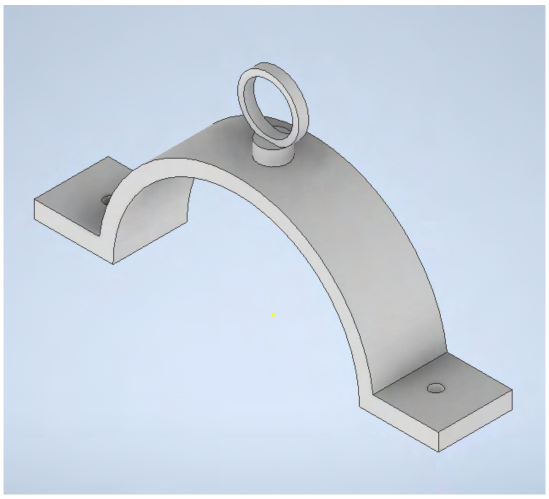
The second iteration would further develop the design with simplification, and streamlined production in mind. The bridge was simplified by offsetting it from the enclosure (initially the enclosure would rest along the bridge which required a curve to fit the enclosure on) which reduced it's complexity for printing. Given that the structure would need to be lowered into the water, the bridge was given two horizontal slots through the side, through which deployment cables could be threaded. These holes were placed as close to the bipod legs as possible, as there were no anchor points to which the deployment cable could be attached to.

To handle the displacement that the legs received when large amounts of force were applied, a strut was introduced between the bipod legs to relieve and displace the weight. Meanwhile the implementations of expanding towards the external functions, such as the lights and the hydrophone, resulting in the the third iteration.

Third Iteration

The third iteration introduces a locking bracket attached over each bipod using friction and screws to affix the enclosure in place, flanges were added to avoid having the screws coming into direct contact with the enclosure.

Working on the locking bracket, the decision to simplify the design became a focus. First the solution of the deployment of the structure became remade to be a set of anchor hooks implemented on the top of each locking bracket, eliminating the necessity of the anchor hooks that earlier were a part of the bridge. The second simplification was to flatten the curve at the top of the bridge, as this would make future development of the design and it's extensions simpler and more cost efficient.


(a) *3D-Render of V3 bracket. (top)*

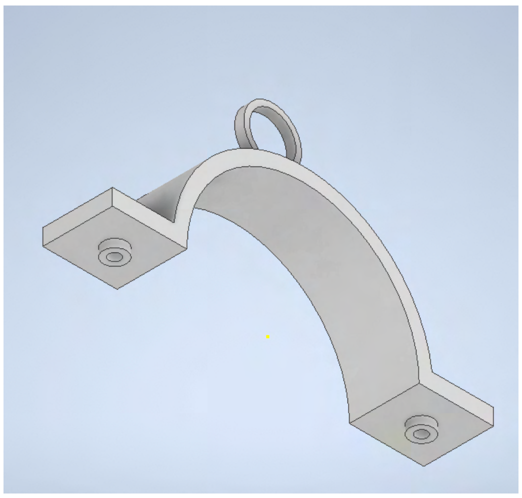

(b) *3D-Render of V3 bracket. (bottom)*

Figure 4.4: *3D-Render of V3 bracket*

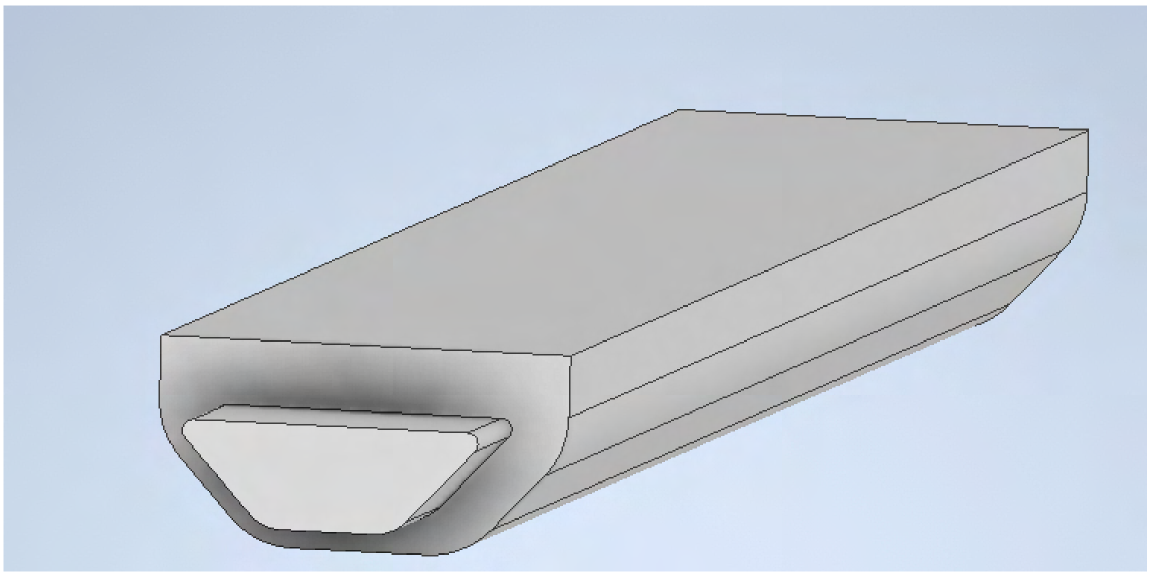


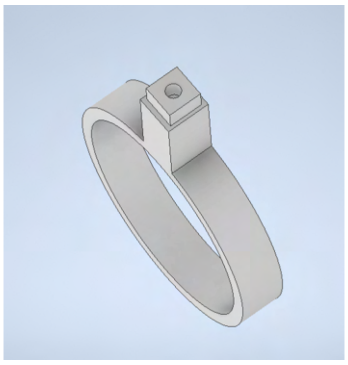
Figure 4.5: *3D-Render of flattened V3 bridge*

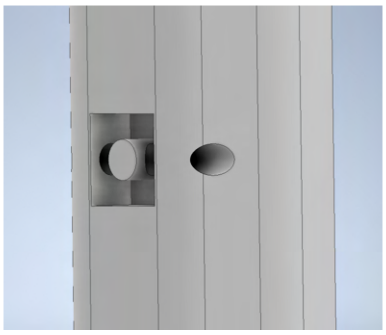
The previously mentioned flanges of the locking brackets created an easy and efficient opportunity for the design of the attachment of external parts. Each part was made with attachment points for both the front and rear of the structure. Two different attachment designs were made. The front was designed with six points of attachment along one axis, allowing flexible placement of lights. The extensions made for the rear bipods were designed with two holes on the same axis as the front, creating the possibility for further expansion. A secondary hole was designed along the perpendicular axis, for cable control. These parts would be attached with the same screws as the locking bracket.

V3.1

The third version is the rendition that has been further elaborated upon, and has been dubbed V3.1, as the V3 version is the design that is the closest to a finished prototype. The V3.1 will have an external housing for the hydrophone.

The back of the V3.1 has been adjusted with the hydrophone in mind, with a stable, sufficient distance from both the seabed and the enclosure. In order to maintain balance due to the uneven distribution of accessories both within and outside the enclosure, a connection point was added on the underside of the bridge that uses a male-female friction fit plug secured with a locking pin through the side of the bridge.


(a) *Front Hydrophone Bracket, attached to Bridge*


(b) *Bridge Socket*

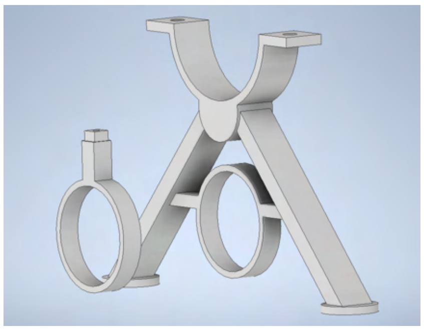

(c) *Rear Hydrophone Mount*

Figure 4.6: *Renders of V3.1 hydrophone extension.*

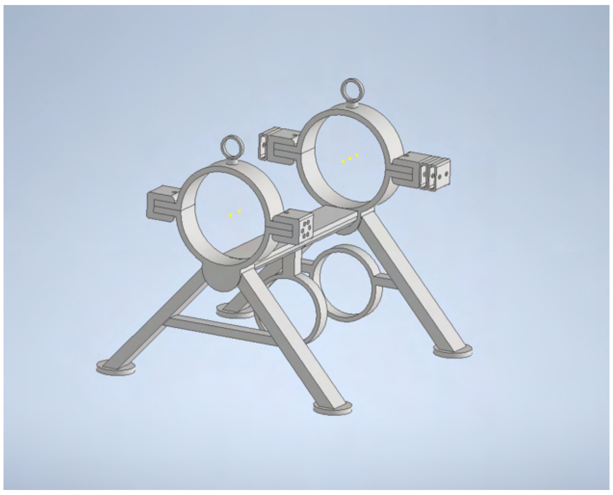
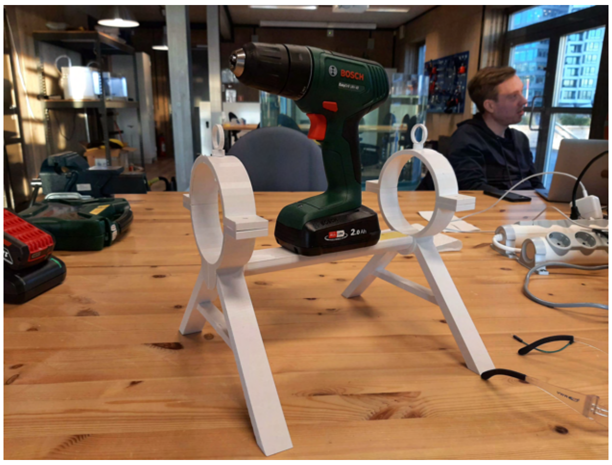


Figure 4.7: *3D-Render of V3.1*

Full Scale Model

The full-scale 3D printed prototype of the wedge design. As mentioned a full-scale model was to large for the 3d printer(Ultimaker 2+ Connect). It's build volume being 223 x 220 x 205 mm (8.7 x 8.6 x 8 inches), the bridge of the wedge design were 260mm. Given this, some of the parts had to be split in designated areas. To avoid having to glue the parts together, the model uses friction fit plug and socket joints similar to the established methods of the wedge design.

The only thing that did not have to be trimmed was the locking brackets above the bipods, as this was only half the size of the legs. This version had some stress marks and has cracked in places where it was plugged, due to variations that have occurred during the temperature changes during printing, and the fact that there was very little to no size difference for the sockets. These are errors and faults that would not occur in an actual finished prototype.



(a) *Full Scale Print of the Wedge Concept*

4.2.3 The Layer Cake Concept

In one of the initial concepts, a three-layer approach was adopted to ensure the safety of the camera cylinder rig. This design involved incorporating three distinct layers: a bottom layer equipped with ballast weight for enhanced stability, a middle layer dedicated to securing the cylinder and other external equipment to the frame, and a top layer where the anchoring points are positioned. The accompanying figure provides a visual representation of this concept.

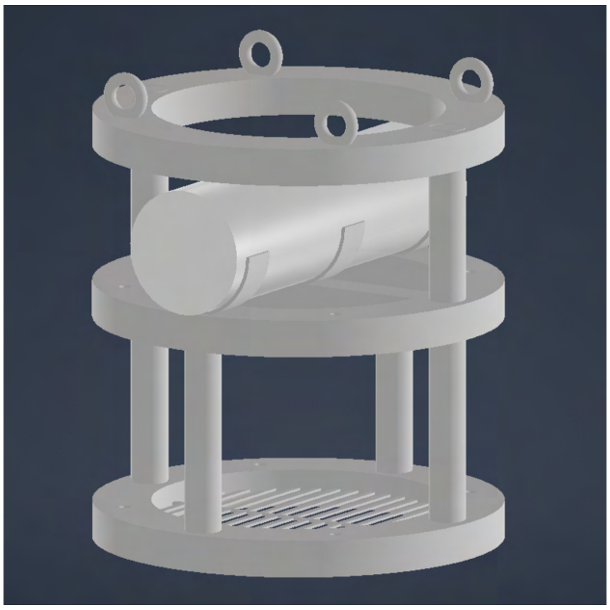


Figure 4.9: *Original three layer concept.*

In its current state, the concept would likely require a significant number of custom machined parts and welding, leading to elevated costs and added complexity. However, the project requirements specified the need for reproducibility and cost efficiency throughout all stages. Taking these requirements into consideration, other avenues to streamline and simplify the concept had to be explored.

Three layer aluminium profile concept

During the concept phase, one early idea that emerged was the incorporation of aluminum profiles. These profiles offer several benefits, such as their ability to be easily cut to the desired length, and assembled. Additionally, they are affordabile, provide an exceptional configurability, and have an impressive strength-to-weight ratio (Weerg 2023).

Due to the utilization of aluminum profiles, the original circular design of the three-layer concept was revised to a rectangular shape. Although this alteration may have implications for the overall stability of the rig on the seafloor, this was anticipated with the inclusion of adequate ballast weight that would mitigate any potential issues.

One of the initial challenges encountered during the design phase was determining the dimensions of the rig. It needed to be spacious enough to accommodate the cylinder and other components, while remaining sufficiently compact for ease of transportation. After careful consideration, it was established that a height of 0.5m would provide ample room for three floors, striking a balance between size and manageability. Additionally, a spacing of 0.25m between each corner and floor was determined to be sufficient and coincided with the length of the profiles. Below is an image of the proposed aluminium profile construction. These dimensions provided sufficient space to fit all required parts within the frame, as well as sufficient clearance so that they won't obstruct each other during deployment.

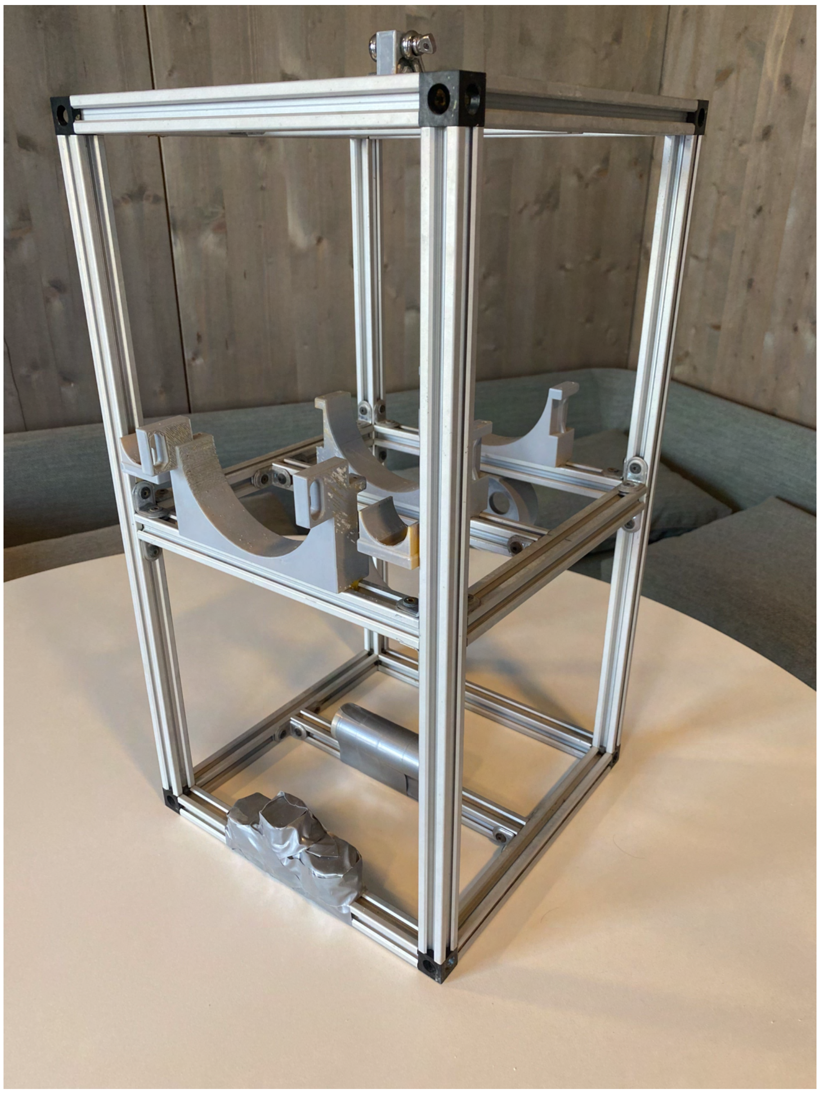


Figure 4.10: *Three layer concept with aluminium profiles.*

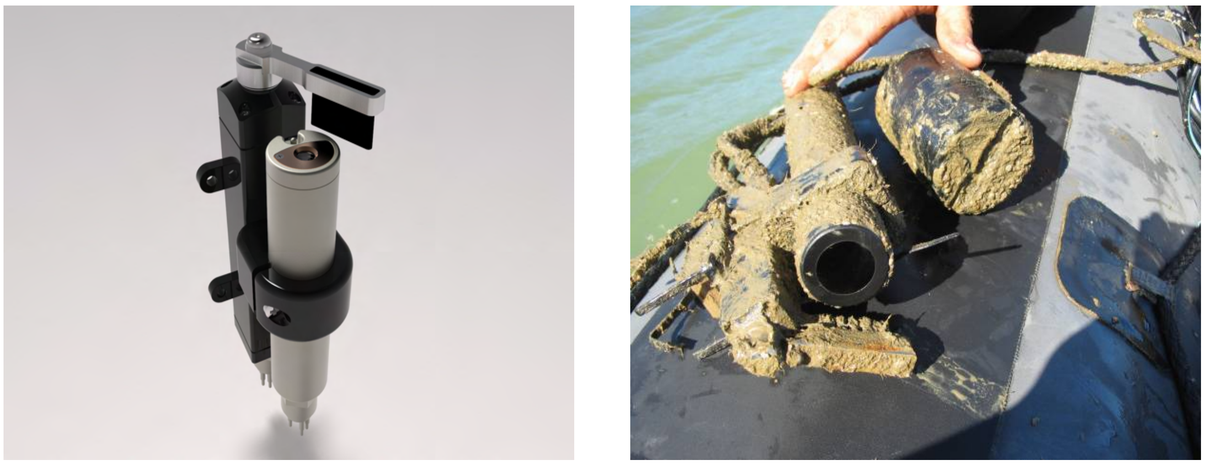
4.3 Dome Wiper Design Process

This project involves the development of a permanent underwater ocean observatory. This rig is required to stay in the water for months at a time. An inevitable challenge during long deployments is biofouling. Marine biofouling is a term describing the accumulation of organisms on man-made objects placed in the sea (Wahl 1989). Thus, it is paramount to review existing products that can limit this unwanted growth on critical instruments. Specifically, this project will look into solutions that can keep the acrylic dome on the enclosure free from biofouling. The acrylic dome houses a camera that requires clear sight to gather useful data for further research. This section explores different existing solutions to the biofouling problem and describes the design process behind the dome wiper.

4.3.1 Existing Solutions

Hydro Wiper by Zebra Tech

There are a few different proven methods to avoid biofouling. Hydro Wiper by Zebra Tech is a mechanical wiper system that is designed to keep optical instruments in a working condition. The design uses a motor-controlled brush, housed in a separate enclosure above the camera view-port, to keep organisms from growing on the surface (ZebraTech 2023). The effectiveness of the wiper is illustrated by the figure below.



(a) *Before deployment.* ***(b)*** *After deployment.*

Figure 4.11: *Product Render and Product Efficiency*

Streetlamp UV

Another solution that has been proven effective against biofouling is the use of UV light to hamper the growth conditions of organisms on critical surfaces. AML Oceanographic is a company that has developed a solution called Streetlamp UV. This solution makes use of LED modules extended on a metal rod. The Streetlamp UV can be configured in many different ways to ensure that the LED modules have a direct view of the target surface (AMLOceanographic 2023).

Figure 4.12: *Streetlamp UV.*

4.3.2 Concept Phase

In order to construct a satisfactory solution to the biofouling problem, the design requirements must be established. A common factor in the previously presented existing solutions are the high cost. These high costs are a result of the products being made for a very niche market with high quality custom parts and implementation. The top priority in this project is to keep the costs to a minimum. This will be achieved by keeping the mechanism simple and using off the shelf components. This will not only make the solution more affordable, but also make it easy for others to replicate. This project is a collaboration between many different universities, so this will enable other interested parties to recreate and iterate on what has already been done. The last key design requirement is to try and keep the number of critical external parts to a minimum. This will ideally extend the possible duration of underwater operations, and keep the cost down.

Design requirements:

- Low cost
- Easy to replicate
- Secure critical external parts

Magnetic wiper mechanism:

Two wiper arms are connected through the acrylic dome with the use of magnets. The inner arm will be actuated by motor(s) affixed within the enclosure. Figure 4.13 illustrates the idea.

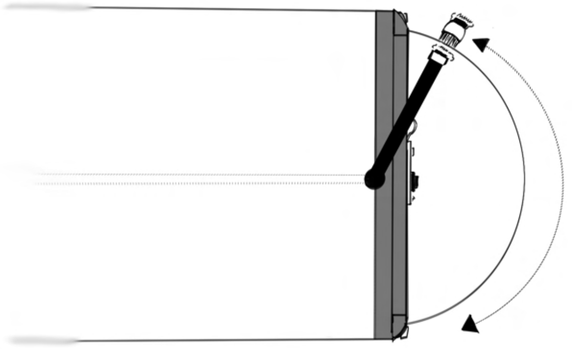


Figure 4.13: *Illustration of the wiper mechanism*

4.3.3 Construction of the Prototype

An objective with the prototype is to contain a majority of the critical components inside the enclosure. This means that the available space in the enclosure must be managed in the best manner possible. The construction inside the enclosure consists of a motor platform, a servo motor, gears, hall sensors, magnets and the inner wiper arm. All custom parts were designed using Autodesk Inventor (Autodesk 2023). An Ultimaker 2+ Connect 3D printer (UltiMaker 2023b) was used to manufacture the parts designed. All parts are made out of PLA, a biodegradable plastic made from renewable sources (RTS 2023). All exterior PLA parts were coated in an acrylic paint to increase its durability during it's extended underwater deployment. (Discussed in section 7.7) The following subsections will go into more detail of the different parts of the construction.

Motor Platform

The main purpose of the motor platform is to actuate the inner wiper arm which in turn moves the outer wiper. The platform consists of a bottom layer with mounting locations for the servo motor, side walls and hall sensors. It also has a top layer were the camera sensor is mounted. The 3D CAD models for the electronics board are available on BlueRobotics' website, which were modified to better serve this project's purposes by adding servo slots, mounting points for the hall sensors and the shelf to mount the camera to.

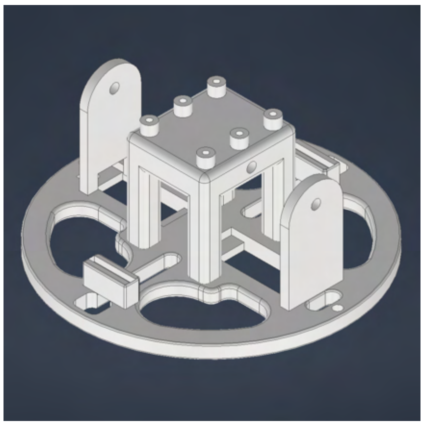


Figure 4.14: *Mounting plate for Hall Sensors, Motor(s) and Camera.*

Wiper Power Delivery

Motors

The initial plan was to use stepper motors for the power delivery to the wiper. Stepper motors offer excellent speed control, precise positioning and repeatability of movement (AdaFruit 2023a). These are all characteristics fitting for driving a wiper arm. A disadvantage with using stepper motors is the lack of positional feedback, so it is important to make sure that the start and end position of the stepper motor is consistent each time (This was later solved by the incorporation of hall sensors to trigger a direction switch in the motors.). Another disadvantage is the need for separate motor drivers which adds to the complexity of the solution.

In a lot of the early testing stepper motors were used as this was considered the most optimal solution at the time. The motors were working great and performed well initially. Problems started to arise when the motors were running for a long period of time, as a lot of unexpected behaviours started to appear. The motors receive movement instructions from an Arduino Nano, and the movements would be correct for the first couple of loops, but then suddenly the wiper would move in the same direction twice and that would cause the motors to lock up. Extensive troubleshooting was conducted, however the error would only appear at different times, making it hard to diagnose its cause. Due to this unreliability, other solutions were investigated.

Research showed that continuous rotation servo motors could be a good alternative to using stepper motors to drive the wiper mechanism. Continuous rotation servos have a lot of the same perks as a stepper motor, but they lack the precise movements and repeatability of instructions. The lack of these features will be remedied through using hall sensors to enable positional feedback. The default way to control the continuous servo motor is to specify the direction and duration measured in milliseconds. In the use case this will be very imprecise in the long run. The introduction of hall sensors simplifies the whole process, as the motor only needs to drive in one direction until it reaches a hall sensor.

Another advantage is the integrated motor driver, as this will simplify the circuit and eliminate a possible variable.

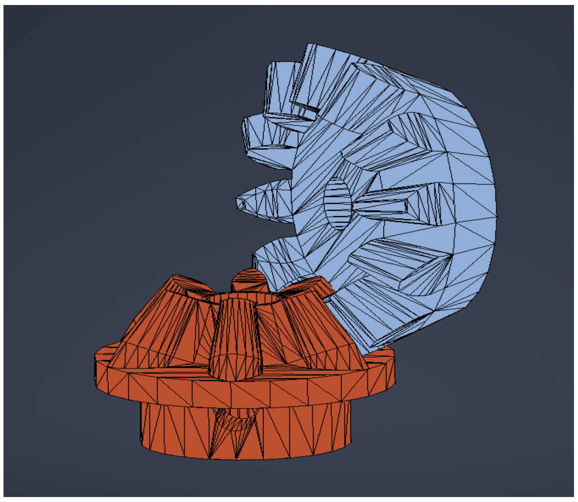
Bevel Gears

The gears connecting the servo motors to the inner wiper arm play a crucial role in delivering power to the wiper. Since the available space within the enclosure is limited, bevel gears were chosen as the solution. By using bevel gears, power can be transmitted from the servo driveshaft axis to the perpendicular wiper arm axis. This allows more freedom for the placement of the motors within the enclosure, which is highly advantageous with the limited available space. A bevel gear design from Thingiverse.com, a website where open-source designs are made available, was used (Thingiverse.com 2023). The bevel gear in question has a 1:2 gear ratio and is compact enough to fit in the enclosure.

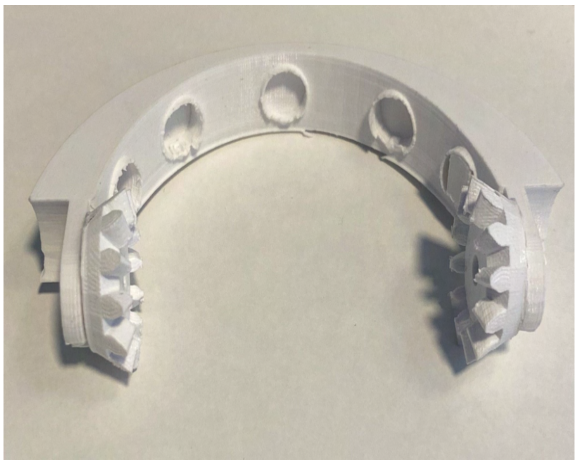
Driving the outer wiper arm with magnetic force

As stated in the introduction, the goal is to utilize magnetic force to connect the two wiper arms. For this purpose, it is crucial to keep the distance between the arms to a

(a) Continuous servo motor.


(b) Bevel gears.

minimum while allowing sufficient room to attach a brush to the outer wiper arm, and to prevent the wiper arms from contacting with the dome. To achieve this, neodymium magnets were used, which are the most powerful rare earth metal magnets commercially available (Lucas et al. 2016). Initial tests with N35 (The weakest and most common grade of neodymium magnets) proved successful but insufficient once space for a brush was included. After testing, N45 neodymium magnets were selected for the project because they provide a strong magnetic field that falls in the middle of the available range for neodymium magnets. (Higher grade magnets, such as N50 and N55, are available, but further increase the cost of production.) These magnets will be placed into pre-made slots located in both the inner and outer wiper arm. As the magnets in each of the arms are placed relatively close together, the inner magnets will be coated with a thermoplastic adhesive and the outer magnets with marine-grade silicone. This will ensure that the magnets will be kept in place over long periods of operation.


(a) Inner wiper arm.

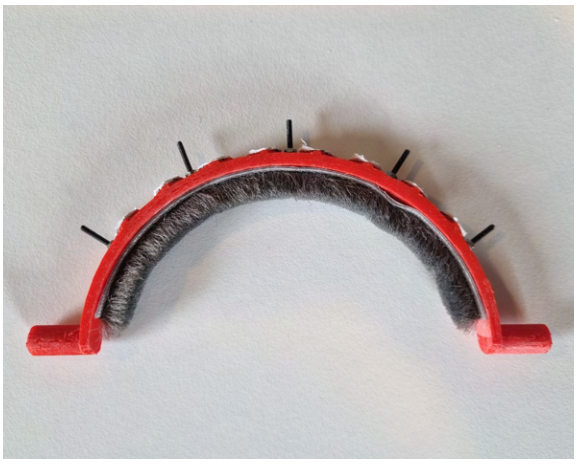

(b) Outer wiper arm.

Figure 4.16: *Final prototypes of the inner and outer wiper arms.*

Securing the Outside Wiper

The wipers' outer section is comprised of the wiping arm and a housing to prevent total loss of the wiper should the magnets disconnect. The first iteration intended to use ball bearings made of plastic and glass (chosen for underwater longevity) to secure and rotate the wiper. (Fig 4.18, left) This was unsuccessful as the exterior of the enclosure dome is not perfectly spherical and the wiper arm required some vertical leeway to be able to complete the wipe. After revisiting the problem, version 2 was made.

By removing the ball bearings and changing the circular slot for the ball bearings to a slot in the shape of a stroke cap (Figure 4.18, middle), the design relies on the magnetic attraction between the wiper arms to secure the wiper, with the slots catching the wiper in case the internal and external magnets should disengage. Should this happen the wiper will fall to rest in the bottom position and reattach on the next scheduled wipe when the inner magnets come into range again.

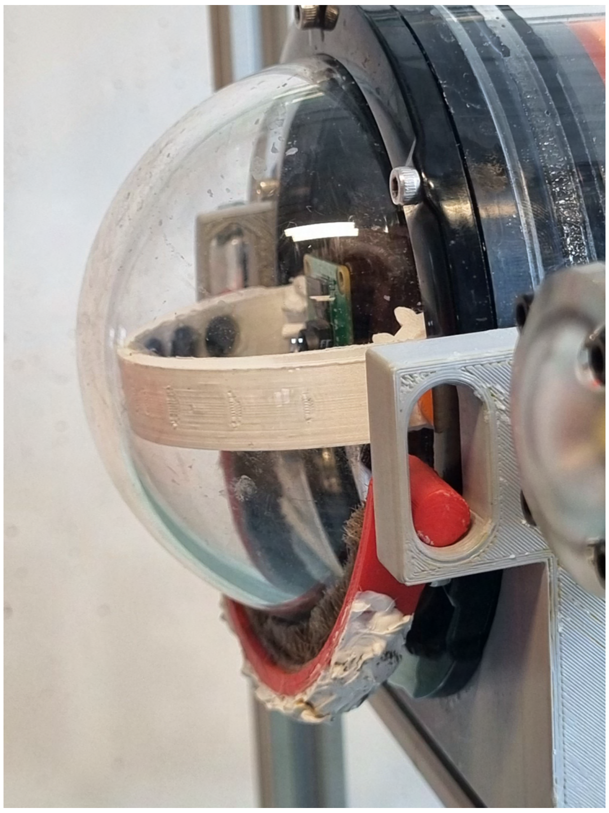


Figure 4.17: *Image showing the outer wiper disengaged from the inner arm.*

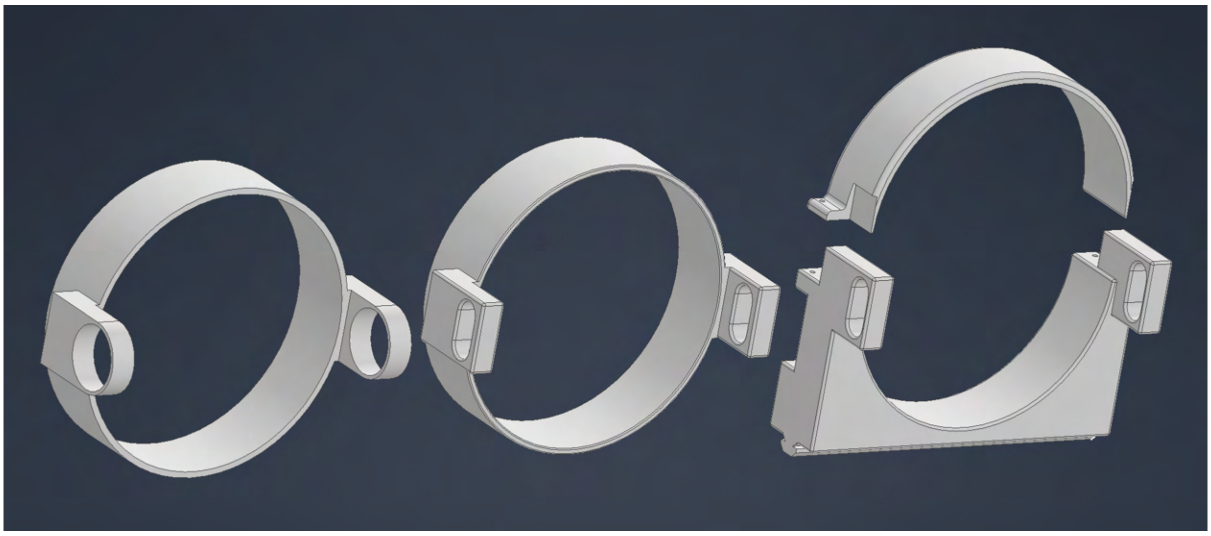


Figure 4.18: *1st (left), 2nd (middle), and 3rd (right) iterations of the outside wiper holder.*

Version 2 worked well, however, as the rest of the design was developed it became necessary to incorporate the wiper holder into the enclosure seat as the two parts would occupy the same space. To do this the wiper holder is split into a main lower section with the top section connected with two nuts and bolts. (Fig. 13: right) The final design would include rubber bumpers on the faces that come into contact with the enclosure. An added benefit is some increased structural support as the seat holder is affixed to the frame and no longer has any possibility to rotate around the enclosure, something that could possibly happen with the previous friction fit designs.

Securing the Brush

To attach the brush to the outer wiper, a 140mm long section is cut from the roll. Using the pre-applied glue, it is attached to the wiper and holes are marked using the pilot holes in the wiper, which are then drilled out using a 2mm drill bit. The brush is then affixed with M2 screws, washers and nuts, which are clipped as close to flush as possible. Finally a thick layer of marine silicon is applied over the screws and magnets to secure them in place. (See figure 4.19.)

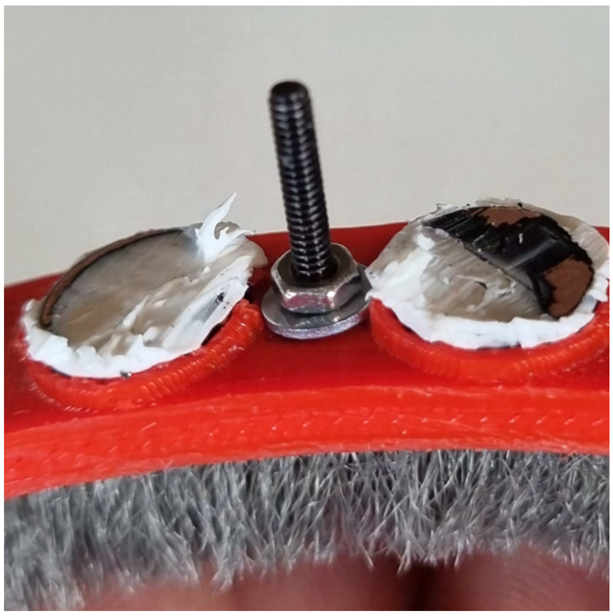


Figure 4.19: *Close-up of top of the outer wiper, showing brush attachment method.*

4.4 Acrylic paint application on PLA-parts

In this project, a lot of custom designed parts made from PLA were used. This material choice keeps the cost down and makes different iterations easy to produce via the use of a 3D- printer. The material is perfect for making parts that sits inside the watertight- enclosure, but for parts exposed to the sea water there are some unfavourable characteristics that must be addressed.

The hydrophobic trait of PLA has potential to alter the expected mechanical properties. In a study conducted by Ecker et al. a 3D- printed PLA part was submerged over seven days. The study determined that the tensile strength and stiffness were significantly reduced. The water had penetrated between the layers of the PLA-part and had a softening effect (Ecker et al. 2019). The conclusion that can be drawn from this study is that a solution needs to be found that prevents the water from compromising the different layers of the PLA-part.

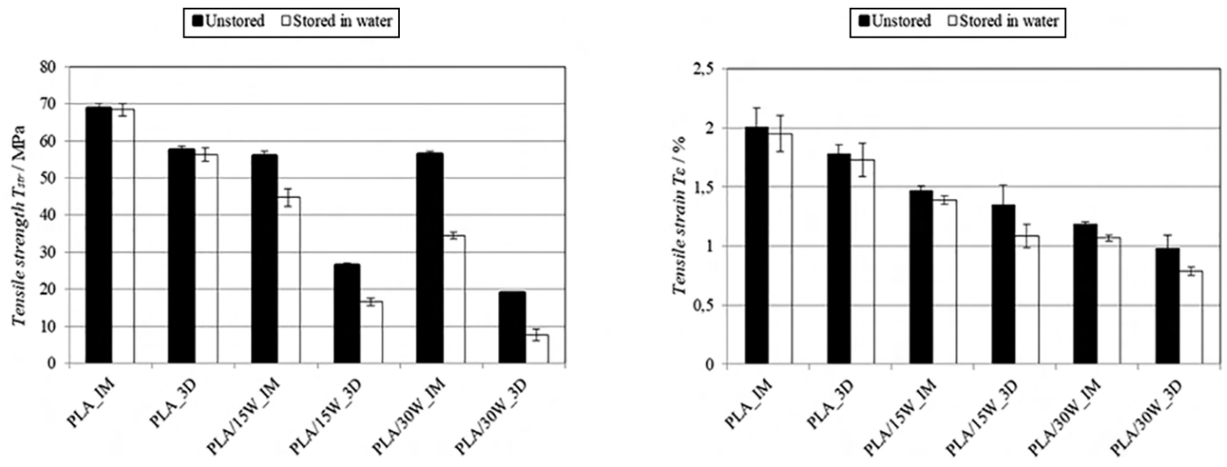


Figure 4.20: *Tensile strength and tensile strain of different types of PLA submersed over a period of 7 days. (Ecker et al. 2019)*

The solution contrived to solve this problem was to use acrylic- paint. The paint will act as a protective layer between the PLA-material and the seawater. It is also expected that the paint will fill in the gaps between each layer of the PLA- part. To achieve satisfactory paint coverage, two paint layers were applied. This solution will hopefully make the parts more resistant to the seawater and hinder the degradation of mechanical properties.

4.5 Software

As the basis of this project is streaming audio and video back to land, a good amount of software is required. The software needs to make both an audio and video stream available to the user, and it also needs to allow the user to control the device itself. The project is built using two languages, C++14 to control the device and Python3.10.8 for the audio and video streaming server. In addition, it uses HTML, CSS and JavaScript to create a user guide made available through the web server, and Makefile to help build the project.

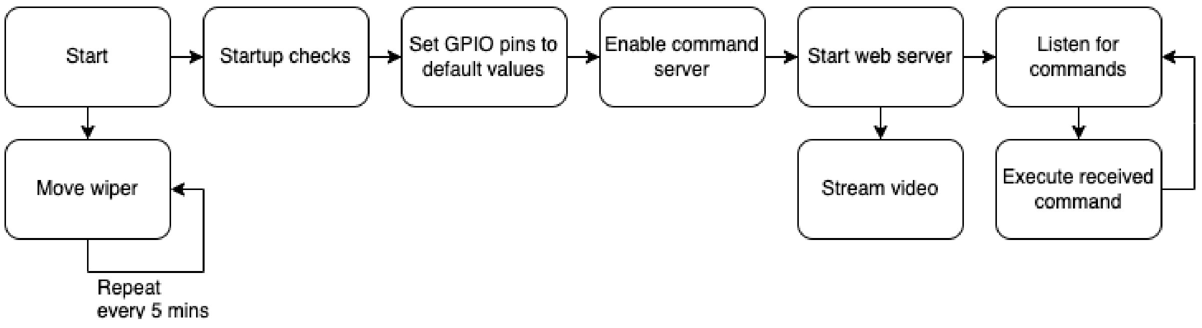


Figure 4.21: *Software Logic Diagram*

4.5.1 The Control Software

This software helps the user control the Pi in a more user friendly way. It's divided in two parts: a linux service which runs at startup, and a client which interacts with the service. This service does some basic checks at boot (that the user has the appropriate permissions, that the GPIO pins are available and controllable, that it is successfully connected to land, and so on) and logs both successes and fails. Three libraries have been written to accomplish this: a Logger library to log any events, allowing the user to monitor the device's operation; a GPIO library, which allows the program to interact with the GPIO pins; and a CPython library, permitting the import of Python code to C++ to run it as a native function. This last library was written to allow the program to more closely interact with the python web server, allowing it to start/terminate it on demand as well as receive a return code.

Logger Library

This library was written to allow the service to log events, and make them available to the user for troubleshooting. It has four log levels (Info, Warning, Error and Fatal), and logs it together with the current time and a message given from the user.

GPIO library

This library consists of two parts, a GPIO class which provides a layer of abstraction, simplifying the process of reading a GPIO pin or setting the state, and a custom exception to help with logging. The way to interact with GPIO pins on a Raspberry Pi is through the "SYSFS" method, which essentially requires the user to write to specific files within the GPIO directory folder: the user must first select the desired pin, then choose whether to use it as an input or an output, and then they can read from/write to it. This can be cumbersome, as the folder structure is relatively complex, which is why the GPIO class was written: by creating a layer of abstraction, interacting with the GPIO pins becomes a lot simpler, meaning that the amount of code needed also becomes less. One issue with the implementation is that through the "SYSFS" method, only one pin can be selected/used at the time. This was solved by implementing the "deselection" in the deconstructor, and using pointers instead of variables on the stack: whenever a pointer is deleted, it has "gone out of scope", and the deconstructor will be called, essentially freeing the GPIO pin.

Web Server library

This library was written to import and use Python code natively in C++. It uses CPython to accomplish this, importing the python function needed, and calling it. This process is run on a different thread from the main application, taking advantage of the four cores of the Raspberry Pi: this lets both the web server and the command listener run simultaneously.

4.5.2 The Web Server

The web server is a core part of the software, letting the user send commands to the raspberry pi, show the log file, publish a video and audio stream, and show relevant documentation to the user. The web server is written in Python, and is based on the http.server.BaseHTTPRequestHandler class, implementing a simple http server from scratch. While it is not well suited for larger scale deployments, it's flexibility is perfectly suited in this project. Additionally, the PiCamera class is used to receive the video stream from the camera.

The StreamingHandler class

The StreamingHandler class inherits from the aforementioned BaseHTTPRequestHandler, enabling the definition of several endpoints that the user can interact with to manage the device. These are better discussed in the FrontEnd section. This class defines two HTTP methods, a GET method which the user can use to receive the video stream, read the logs, and so on, and a POST method, which the user can use to send commands to the device.

The VideoStreamOutput class

This class was defined to receive the video stream from the camera and make it available to the web server. Whenever it reads two bytes (0xFF,0xD8) that signal the start of a new video frame, it empties it's buffer, reads and stores the camera's output, and returns it to the web server for streaming.

4.5.3 The Porpoise Bit Conversion Library

While a mostly prepackaged solution, the Porpoise hydrophone required a bit of software developed to properly receive and play back audio. The Porpoise provides a live, 24bit, WAV audio stream over port 5453/UDP, however, each sample is encoded big-endian, while WAV format requires little endian. (endianness indicates the order of the bytes in a sample; big-endian data has it's most significant byte (MSB) first, while little-endian has it's least significant byte (LSB) first (Bolanakis, Kotsis, and Laopoulos 2009).) This means that the order of the bytes in each sample had to be "flipped" for the audio to be played back to the user.A small library was written in C++ that allows the user to convert binary data from 24bit to either 32bit or 16bit, as well as change it's endianness. Through the use of macros, the user can then compile it as an ".o" file, that can be included in other C++ projects, or as a CPython file, allowing them to import it and associated functions in a Python program.

4.5.4 Client

Another software component of this project is a client that can receive and run image analysis on the generated video stream. This software is written in Python and uses OpenCV to receive and process the video.

FrontEnd - Console

Aiming to give control towards the user, the console User-Interface (UI) is designed to maintain ease of use when controlling the functions of the lights, the wipers, and cleaning up log files.

In the first iteration of the interface it used a simple brute solution written in HTML, to make sure that everything worked as desired when the requested signal was received.

```
<form method="POST">
    <button name="command" value=3 type="submit"
    action="/console">
    Lights On
    </button>
 </form>
```

Listing 4.1: *Html brute solution*

The method worked by sending an array containing *command : 3*, which the back-end would receive before turning on the lights as requested. This method would then be repeated amongst the other function.

The problem with this solution was that whenever the user would *post the command* the page would redirect itself to *itself*, refreshing and effectively erase the network activity and console log with every post. This made testing the functions near impossible, as there was no feedback for remote testing. A minute problem was that each button contained practically the same lines of code, making the code unnecessarily repetitive. A solution to this is to write the post function in JavaScript (JS) and a one line in HTML calling the function.

```
<button onclick="addCom(3)">Lights On</button>
```

Listing 4.2: *Calling the addCom() function in HTML*

```
function addCom(buttonvalue) {
    let testUrl = 'https://httpbin.org/post';
    let url = 'http://10.44.6.51/api';
    let buttonInfo = {
        command: buttonvalue
    }
    fetch(url, {
        method: 'POST',
        headers: { 'Content-type': 'application/json;' },
        body: JSON.stringify(buttonInfo),
    })
    .then(res => res.json())
    .then(json => console.log(buttonvalue));
}
```

Listing 4.3: *A unified clean script*

Using a *fetch()* function, an array can be posted (Mozilla 2023b), converting it into json, witch in turn can be read and posted back towards the user through the console log. Since there were no longer a problem with the function redirecting itself and refreshing, the user can see what the function posted, and write a solution according to the result. This function only required an assigned value, witch in turn would be applied to a *buttonInfo* array, containing the required tag and value.

FrontEnd - Index

The index page is meant to be clear and concise for the user to navigate. Containing four buttons, each with their own modal popups, allowing the user to change whether they want to control the functions in console, check the system logs, read the docs or view the video stream.

```
function genModal(ModalId,i) {
    var span = document.getElementsByClassName("close")[i];
    ModalId.style.display = "block";
    span.onclick = function () {
        ModalId.style.display = "none";
    }
    window.onclick = function (event) {
        if (event.target == ModalId) {
            ModalId.style.display = "none";
        }
    }
}

function openConsoleModal() {
    var modal = document.getElementById("consoleModal");
    var id = 3;
    genModal(modal, id);

}
```

Listing 4.4: *The modal pop-up script*

Considering that every modal and id has to be unique, making the function for each button was executed similar to the first iteration of the console page. It's possible to make it clean and uniform by creating a simple *openModal* function in the script, and passing the name of the modal and it's value through the button in HTML:

```
function openModal(string ModalName, int Value) {
    var modal = document.getElementById("ModalName");
    var id = Value;
    genModal(modal, id);
}
```

Listing 4.5: *Example of an openModal script*

When the *OpenConsoleModal()* function is called, it creates two variables of modal and id, assigning the desired values. Considering that the desired result is a popup modal, it is useful to use a *getElementById()* method to return an object represented by the matching id (Mozilla 2023a). This bridges the desired result to a function. In this case a pop-up modal. The function is then called within the *genModal()* function that displays the modal according to the style, and allows the modal to be closed, depending on where the user clicks.

```
<div>
    <button onclick="openConsoleModal()">Get Console</button>

    <div id="consoleModal" class="modal">
        <div class="modal-content">
            <span class="close">&times;</span>
            <iframe src="console.html" title="Console" class="modal-
   child"></iframe>
            <p>
                <a href="console.html" style="text-align:center;">
   Console</a>
            </p>
        </div>
     </div>
</div>
```

Listing 4.6: *Modal pop-up in HTML*

In the HTML-code, the function is bound to their buttons and modal. It's then given classes that allows the popup to work as designed. Most importantly the .modal. stylesheet.

```
<style>
    .modal {
        display: none; /* Hidden by default */
        position: fixed; /* Stay in place */
        z-index: 1; /* Sit on top */
        padding-top: 100px; /* Location of the box */
        left: 0;
        top: 0;
        width: 100%; /* Full width */
        height: 100%; /* Full height */
        overflow: auto; /* Enable scroll if needed */
        background-color: rgb(0,0,0); /* Fallback color */
        background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
    }
</style>
```

Listing 4.7: *Stylesheet*

4.6 Final assembly

4.6.1 Assembly of the Rig

To assemble the complete rig, an assortment of aluminum profiles, L- brackets, M5 screws, T- nuts and 3- way connectors are needed. The complete parts list can be seen below.

- 4x 0,5m aluminum profiles
- 15x 0,25m aluminum profiles
- 24x L- brackets
- 72x 0,8mm M5 screws
- 48x T- nuts
- 8x 3- way connectors

The assembly process is very straightforward. The 3- way connectors get attached to both ends of the 0,5m aluminum profiles. As each of these profiles represent a corner in the construction, the 0,25m profiles can be attached between them. This creates the outer skeleton of the rig.

To assemble the mid- floor, the L- brackets and T- nuts secure the 0,25M profiles to each corner. These parts are also used to attach the brace at every floor of the construction. In the end what is left is a lightweight, sturdy and easily configurable rig.

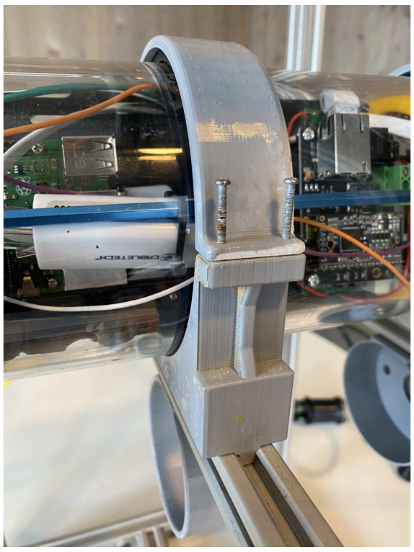
4.6.2 Mounting of Components

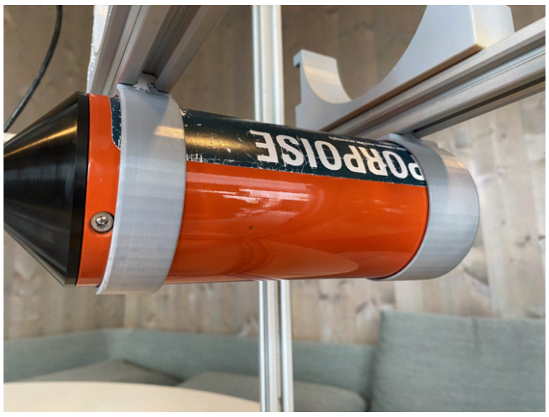
The aluminum profiles provide a useful rail-mounting system, that makes the mounting of external parts very simple. The printed mounts are slid onto the profiles to secure them. It is also possible to use additional l-brackets on either side to further secure them.

4.6.3 Electronics Assembly

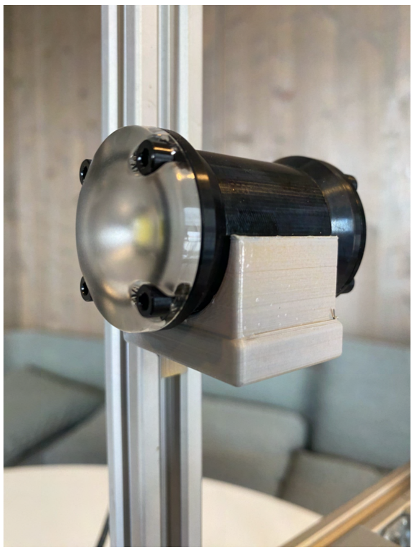
In order to mount everything inside the enclosure, a platform was designed to sit inside of it. The platform was 3D-printed to be compatible with the BlueRobotics electronics board, and proceeded to mark where everything was to be mounted. Once that was done, holes were drilled through the platform and affixed everything to it using nuts and bolts.

The next step is to wire everything together. The power supply is connected to bus bars to distribute 24V power to all devices inside and outside the enclosure, such as the fathom-X board, the lights, and the porpoise hydrophone, as well as to the two buck boost converters, providing the needed 5V for the wiper motor, ethernet switch, arduino nano and raspberry pi.


(a) Acrylic cylinder housing adapter.



(b) Porpoise hydrophone adapter.


(c) Lumen light adapter.

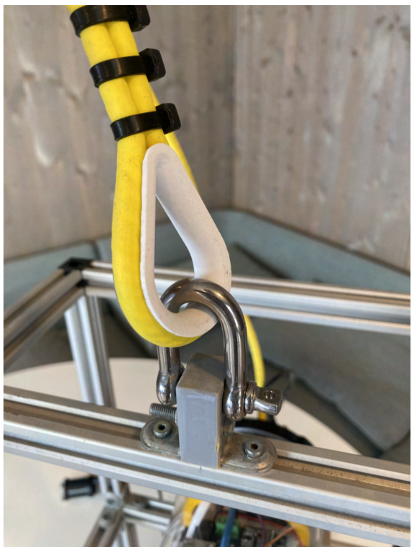

(d) Lowering hook adapter.

Figure 4.22: *PLA-adapters mounted on the rig. Technical drawings of these adapters can be found in the appendix.*

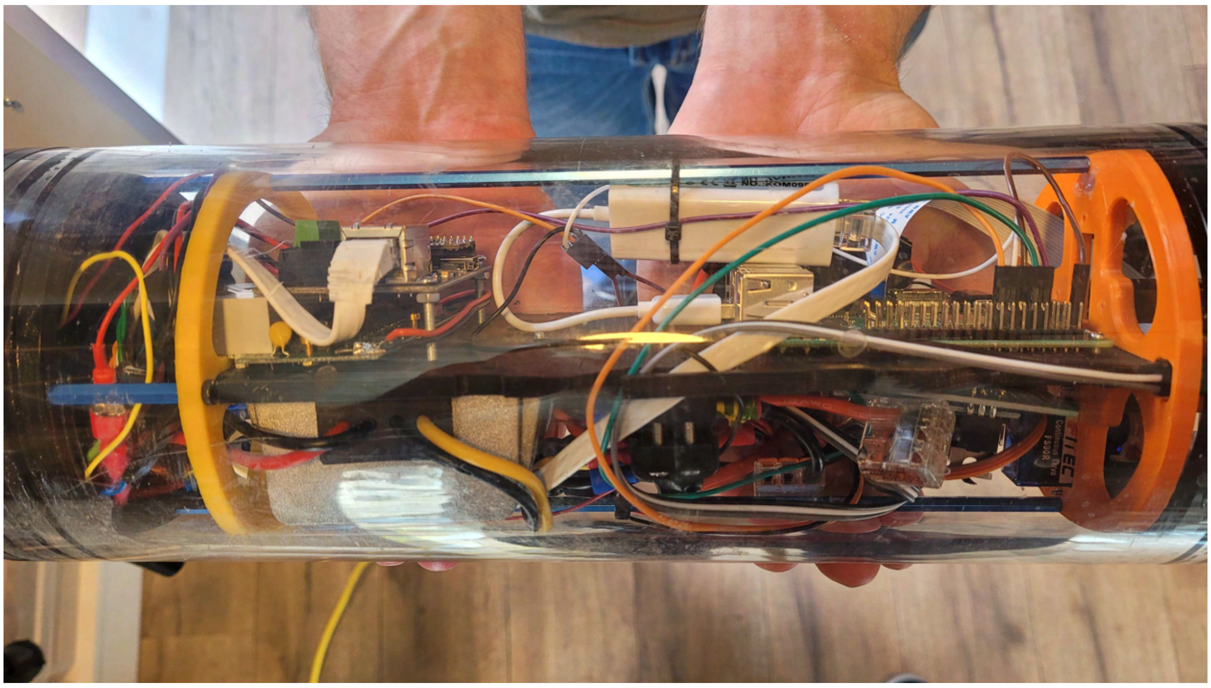


Figure 4.23: *Overview of the assembled electronics.*

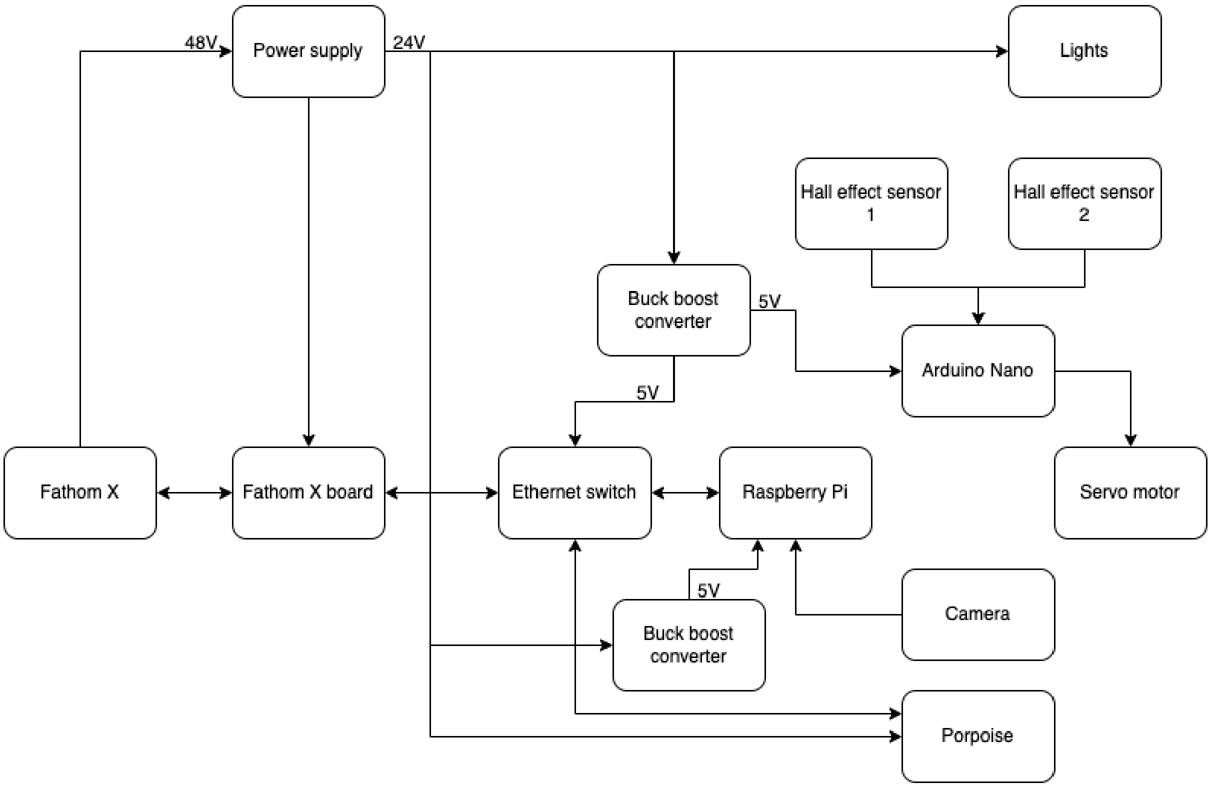


Figure 4.24: *The wiring diagram of the device*

Chapter 5

Results

5.1 Test Planning

The original plan was to test the finished rig in the ocean right outside the workspace, as there is easy access to internet and power. Areas that would meet the needs of a deployment test were investigated for utilities access. After surveying applicable locations for the testing, an appropriate testing site was found of the coast of the island of Malmøya in the Oslo fjord. The place was private and most importantly had access to power. But before the rig could be lowered and testing could begin, it was necessary to survey the seabed and document conditions.

5.2 Surveying the Seafloor

The purpose of surveying the seafloor was to find the optimal location for the rig. The objective was to find a mostly level seabed, which is free of large rocks that could make the placement of the rig unstable. The surveying equipment used was the Chasing Dory underwater drone (*hasing* 2023). The drones capabilities are as followed:

- ✓ Depth lock
- ✓ 45° tilt lock
- ✓ Camera sensor: 100° FOV and 1080p resolution

Figure 5.1: *Chasing Dory underwater drone (*hasing *2023).*

The underwater drone was very helpful in collecting data from the seabed. The seafloor was mapped with multiple passes to locate an appropriate location. To achieve this the depth lock and 45° tilt lock of the dory was used while scanning the seafloor. Data showed that the seafloor at the potential testing location was comprised of mostly mud and silt, and was generally free of any large objects that could interfere with the rig, however the visibility was generally poor due to cloudy water. The frame below is gathered from the camera recording during the mapping process.

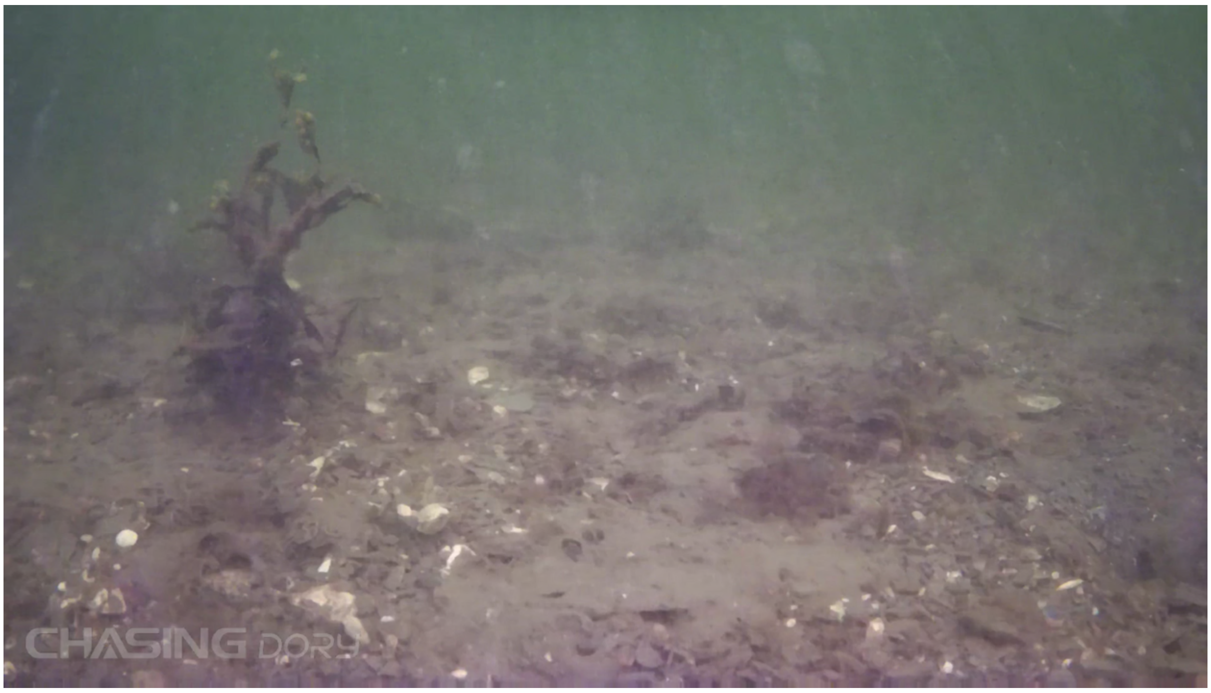


Figure 5.2: *Seafloor mapping 22.04.2023.*

5.3 First Test Attempt

After successfully finding a proper deployment location, the rig was prepared. The critical parts of the rig such as the wiper, electronics and software were tested, and initially all seemed to be working as expected. Once the assembly was completed, all of the equipment was transported to the testing location, and a final dry run was performed before deployment.

During the dry run, a problem with the wiper mechanism was discovered. The inner wiper arm came into contact with the dome, and caused the whole mechanism to seize up. This was caused by the servo motors not running in sync, causing the arm to leave the planned arc of travel and push against the dome. This problem was exaggerated by the small margin between the wiper arm and the dome. As this was not a problem that could be fixed on location, the planned test had to be scrapped. To ensure such a problem would not happen again, more tests will be performed before the next deployment, to ensure all edge cases are covered.

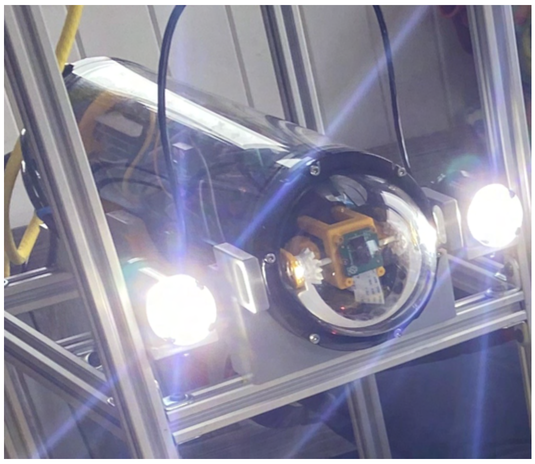


Figure 5.3: *Stalled wiper mechanism.*

5.4 Deployment of the rig and preparations

5.4.1 Changes before deployment

Multiple aspects of the wiper mechanism were changed in preparations for the next deployment of the rig. The first thing that was changed was the axis of rotation for the inner wiper arm, which was lowered so that the clearance between the arm and the dome feature would become less prone to changes in the planned arc of travel. The axis of rotation was lowered by 3mm to increase the distance between the internal wiper arm and the dome, without lowering the pull force between the two wiper arms too much.

Another change that was implemented was the removal of one of the servo motors. It was discovered during testing that the wiper mechanism was almost as effective with one motor compared to two. This change eliminated the syncing errors encountered during the first planned deployment.

5.4.2 Pool Testing

A lesson learnt from the previous failed deployment was to test extensively. As pretty drastic changes were made to the wiper mechanism the decision was made to perform in-lab testing that simulated deployment in the sea. A small tank with transparent walls was used that allowed monitoring of the wiper arm in action. After several runs, it was confirmed that the changes made to the wiper were working as they should.

The pool testing also allowed examining of the angle of the rig while being lowered into a body of water. This showed that the rig was leaning heavily to one side, which would cause the rig to nearly tip over once it hit the bottom of the pool. This provides insight on how the ballast weights should be distributed during the real deployment.

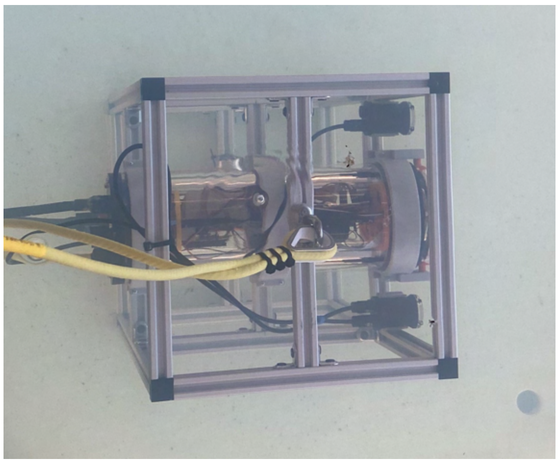
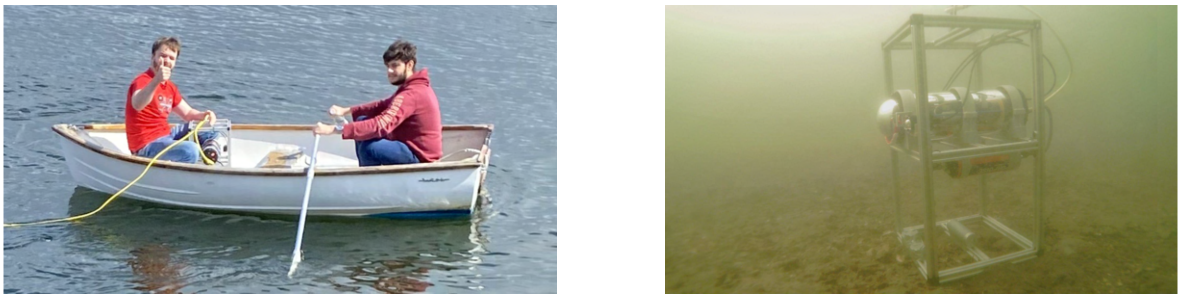


Figure 5.4: *Rig during pool testing.*

5.4.3 Deployment

After the successful completion of pre-deployment testing, it was time to put the rig out in the ocean. An additional dry run was performed on location just before deployment. A small rowboat was used to transport the prototype to the area determined to be acceptable from the previously conducted survey. The rig was manually lowered to the seafloor, and to confirm the successful placement of the stationary rig, a BlueEye underwater drone was used for observation (Blueye 2023).



(a) *Preparing for deployment.* ***(b)*** *Rig stationed on the seafloor.*

Figure 5.5: *Images from deployment 2, 02.05.2023*

5.4.4 Livestream with OBS and Youtube

In the project, a primary objective was to enable a live stream showcasing real-time underwater conditions. To achieve this, a widely recognized open-source broadcasting software, OBS, was used to process the video feed from the camera sensor. Subsequently, YouTube was leveraged as the streaming platform to host and broadcast the live stream.

To initiate the process, OBS was integrated into the setup, allowing for efficient processing of the incoming video feed from the camera sensor, ensuring optimal quality and performance. By using OBS, various overlays such as time and date could be applied, making the live stream more informative (OpenBroadcasterSoftware 2023).

Once the video stream was processed, the next step was to host and share the content with a broader audience. Youtube, a renowned video hosting and streaming platform, was selected for this purpose, as it offered the necessary infrastructure and audience reach. YouTube's robust streaming capabilities provided a toolkit to seamlessly stream the processed video feed from OBS (YouTube 2023).

Poor visibility was experienced during the livestream due to algae, dirt and pollution, which made it difficult to classify objects that were further away from the rig. Even with these tough conditions, the camera managed to detect various kinds of sea life. With a better camera sensor it could be possible for researchers to use machine-learning to classify different types of marine life. The picture below shows an example of the type of sea life that could be observed during deployment.

Figure 5.6: *Flounder seen on the live stream.*

5.5 Retrieval of the Rig

The rig was submerged in water for a duration of 9 days with the objective of assessing the efficiency of the wiper and testing the prototype's capabilities. Originally, the plan was to keep the rig in the ocean for 14 days, however a significant malfunction happened in the wiper mechanism. Due to concerns about the potential risk of damage to the rig's systems and components, the decision was made to retrieve the rig earlier than previously planned. Once the rig was back on shore, two main points were evaluated: the wiper's effectiveness in combating biofouling was assessed and any potential degradation of the PLA-parts was evaluated.

5.5.1 Biofouling Results, Effectiveness of the Wiper

Biofouling was observed on various components of the rig, despite it's relatively short 9-day deployment. The presence of microfouling on the exposed parts was localised to surfaces other than the camera dome. Additionally, sediment had accumulated on the rig's surface. However, despite these challenging conditions, the camera dome equipped with the biofouling wiper exhibited significantly less fouling compared to the rest of the rig. This can be attributed to the effectiveness of the brush on the outside-wiper, which notably reduced the accumulation of microfouling. The image below illustrates the contrast between the wiping area and the non-wiping area in terms of microfouling.

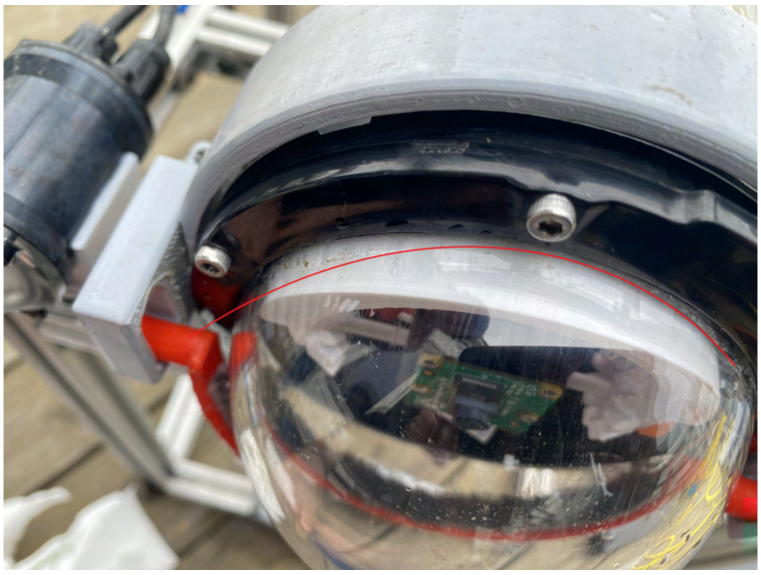


Figure 5.7: *Biofouling on the camera dome.*

Chapter 6

Discussion

6.1 Results Analysis

The project's aim was to design an underwater observatory that would enable researchers to observe marine ecosystems in an accessible and cost-effective manner. the analysis of the results sheds light on various aspects of the prototype´s cost, functionality, and potential for scientific research. This discussion section will present a comprehensive analysis of the obtained results and their implications for the design of the underwater observatory.

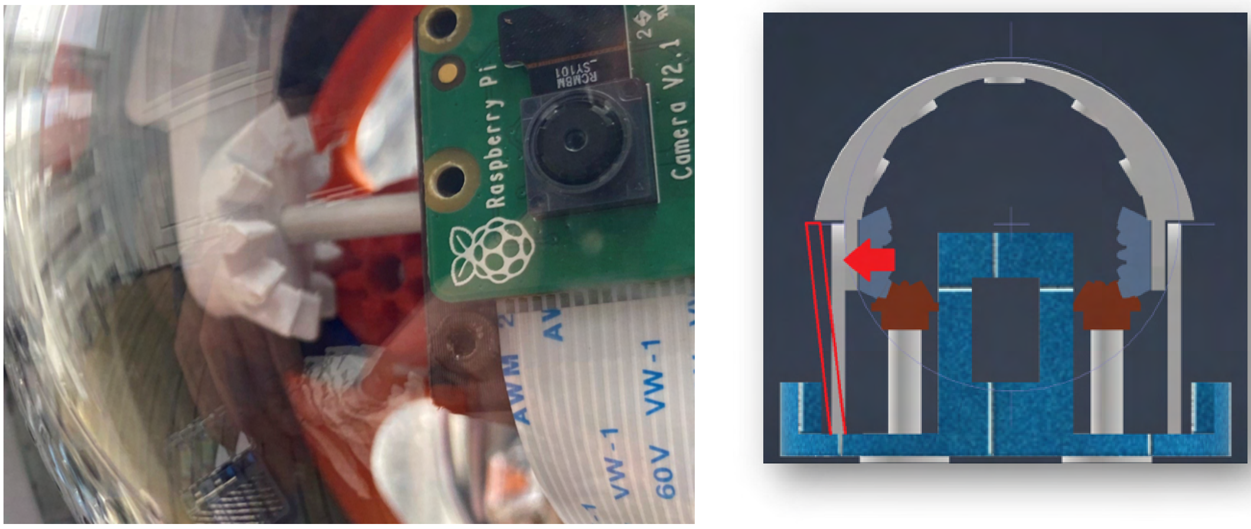
After retrieval of the rig from the testing location observations were made on the three different systems; the wiper, the stream, and the rig structure, and how they were expected to perform in practice.

6.1.1 Wiper Results

Over the 9 days of deployment, the wiper ran successfully for seven days until the wiper wall failed: this part was a separate piece glued on to allow for the insertion of the second servo motor that was removed before deployment. This was a last minute modification after one motor began to show signs of failure, as it was determined that the wiper could be driven with one servo, so the other was removed. There was not enough time to reprint the camera/motor mount and so the test deployment ran with the original mount designed, which was the eventual point of failure for the wiper system.

Once the rig was retrieved, deposits of sediment and initial stages of biofouling at the edges of the dome where the wiper was not able to reach can be seen: this was a clear sign of the brush keeping the dome clear, providing some form of biofouling defense.

Shown in figure 6.1, at some point during the test, the glue holding the wiper wall loosened and allowed the wiper arm to be displaced just enough for the gear to disengage. This resulted in the inner wiper locking up, and the servo running continuously until approximately 6AM on day nine just before retrieval, when the sound finally stopped. This sound was initially observed once the YouTube stream was set up with the hydrophone audio. Once the inner wiper locked up the hall sensors were no longer being triggered which resulted in the servo never receiving a stop trigger, and running until it eventually burnt out.



(a) Image showing the dislocation of the gears driving the wiper arms.

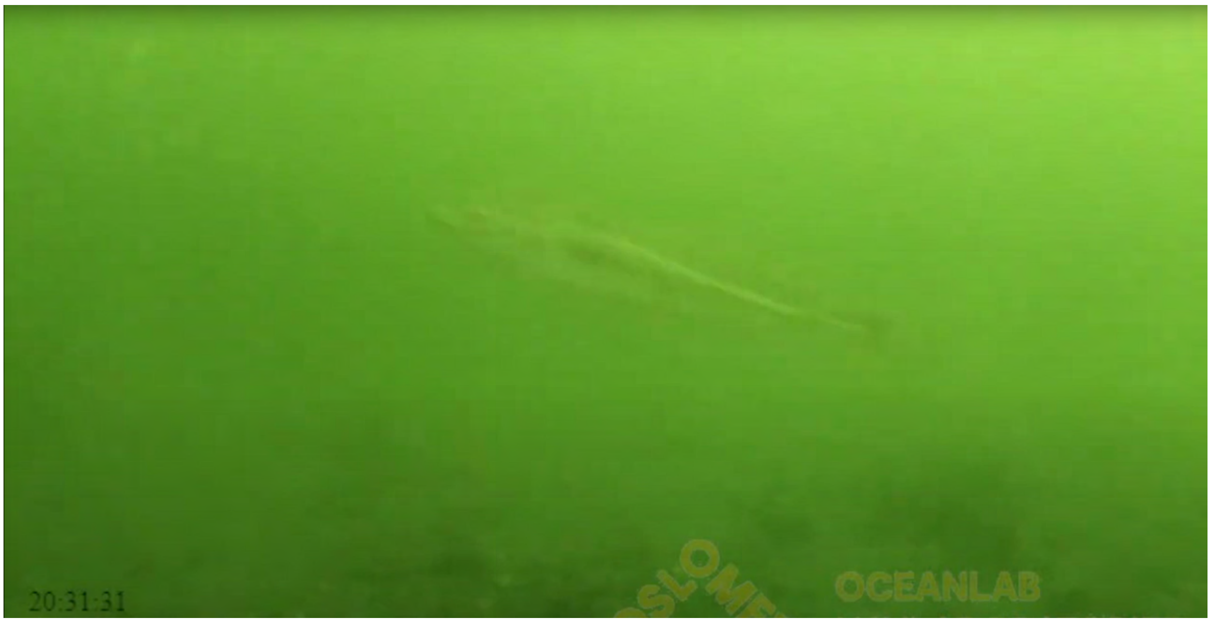
(b) Diagram showing the failure point of the gear system.

Figure 6.1: *Images depicting gear point of failure.*

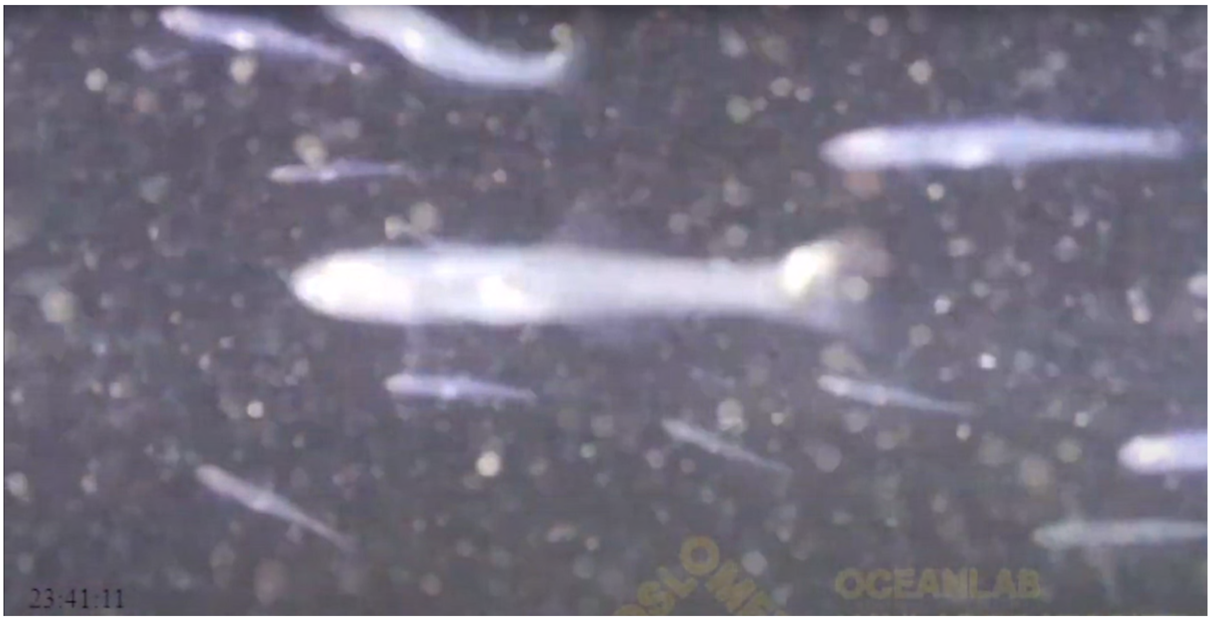
In hindsight, reprinting the mount with both wiper walls fixed as part of the mount print, as opposed to separate parts that were glued on, would likely have provided some increased rigidity and could have prevented the failure. This also provides the possibility of adding struts to support the walls, further reducing the probability of encountering this failure again.

6.1.2 The stream

The stream was initially set up on a private domain that only had video, but was then moved to YouTube when the audio stream was included. The camera used was limited in functionality and this unfortunately shows in the video quality, however this was an accepted limitation in the interest of providing a proof of concept prototype as opposed to a finished product. Additionally, a portable wi-fi router was used which resulted in mild buffering issues due to maximum possible bitrate transfer, though these were largely fixed by the compression on YouTube streams, at a cost of slightly lower resolution (support.google.com n.d.). In a further iteration it would be recommended to use a higher quality camera to provide clearer image and a variable focus camera to ensure better focus on subjects closer to the camera. Vision was limited to 1-2 meters due to the cloudiness of the water. It might also be prudent to include some storage on the topside unit to allow video to be collected and distributed in batches to reduce buffering issues, and to privately host the stream to maintain the high resolution of the feed which would be important if the video data is to be used for any further manipulation or analysis. Having said that, we were fortunate enough to get a lucky capture of a fish swimming across the shot during one day, and a large amount of activity (likely due to the lights) around the rig at night.



(a) *Screenshot of the feed during the day.*



(b) *Screenshot of the feed during the night.*

Figure 6.2: *Day and Night shots from the live feed.*

Additionally the positioning of the lights could be improved by moving them away from the camera and either below or above the plane the camera sits on to reduce the amount of light reflecting off of sediment and other particulates in the water, very strong reflections were observed when the lights were on, though this can be partly attributed to the saturation of particulates and general cloudiness of the Oslo fjord.

6.1.3 Aluminium Structure

When observing the effect of being submerged on the aluminium structure of the rig, and steel components (brackets, screws etc.); the profiles themselves, while having a layer of sediment on and in the rails which could cause issues should the structure need to be dismantled, not much physical damage or corrosion was observed on the structure itself. The steel components however, displayed both rust and mineral deposits once dry as seen in figure 6.3, this could be combated by spraying an acrylic or epoxy coating on them before deployment.

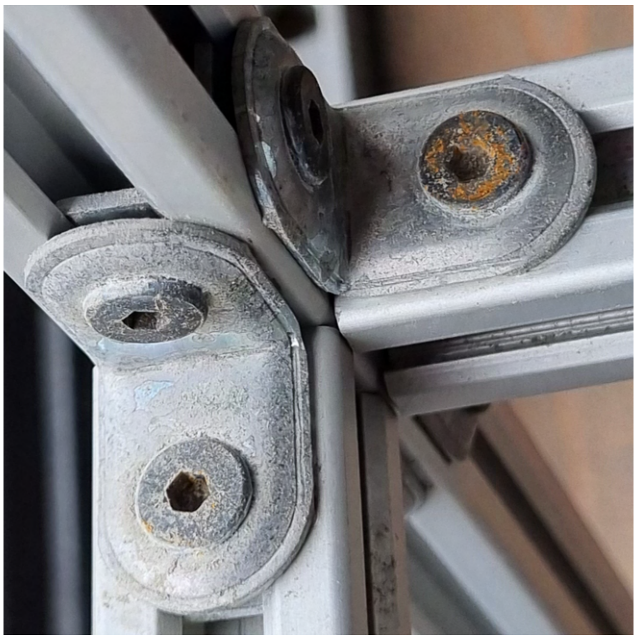


Figure 6.3: *Image showing rust and salt build up on l-bracket and screw.*

6.2 Comparisons to the Theory and Existing Solutions

One major observation that departed from expectations was the speed of biofouling buildup, while there was a noticeable difference between the area disturbed by the wiper and the rest of the enclosure's surface, it was far less than expected after being submerged 9 days, this suggests that while there is a possibility that even stronger magnets would be necessary at 150m depth, and definitely a more robust connection from motor to wiper arm than 3D printed PLA, the wipe frequency can be reduced to once or twice a day and still prolong functionality of the camera for a significant amount of time.

In addition, further research into the topic suggests that as depth increases, the abundance of sealife and biomass decreases due to reduced quantity of light, food, and oxygen (Wishner 2000). This has provided some insight as to why the commercial entities that provide deep water solutions do not typically have a brush attachment or other form of anti-biofouling measure. The viewport would remain functional for far longer at 500+ meters deep, requiring very infrequent, if any, maintenance.

The coated PLA external parts came out far better than expected, and from some imperfections in the acrylic spray coat it could be seen that where the PLA was fully coated there was little to no wear, with very minor softening occurring where the paint layer was thinnest. While these would normally work well for a prolonged deployment, and in the interest of not further polluting the oceans it would be advisable to have these manufactured from a non-plastic material to reduce microplastic pollution. This is due to how non-biodegradable and biodegradable plastic produce microplastic (Wang et al. 2021).

6.3 Cost Analysis

The current configuration of the prototype offers a significant cost advantage compared to existing counterparts. A balance was struck between cost and reliability, however, certain components used in this project did not meet the desired level of longevity.

As mentioned earlier, the utilization of custom PLA parts proved to be a cost-effective alternative to purchasing expensive pre-built solutions. While this approach yielded considerable cost savings, there were drawbacks as well. The tolerances of the 3D-printed parts had a negative impact on the connection of the bevel gears between the servo motor and the inner wiper-arm. An alternative option could have involved outsourcing the manufacturing of the components to a company capable of injection molding ABS parts. Although this would have resulted in slightly increased costs, it would have ensured better tolerances and improved power delivery. A work-around could be to mass-produce or batch-produce parts to reduce the price per unit.

Furthermore, investing in a higher quality motor for this project would have extended the rig's runtime. The gears on the current servo motor were made of plastic, which could be considered a notable weak point considering the planned runtime of several months. Purchasing a motor with metal gears would have increased the overall solution cost but would have provided the added benefit of a longer lifetime.

Additionally, another good investment would have been a better camera: while the Pi Camera was adequate for the job (i.e. demonstrating it was possible to build such a device), in reality this kind of project would benefit immensely from a higher quality camera, providing the user with a higher quality video feed that can then be analyzed.

Finally, several components used, while adequate for the job, were relatively expensive, such as the tether cable, the Porpoise hydrophone, and the blue robotics lights. These all had a relatively hefty price tag, especially the blue robotics tether cable, coming in at well over NOK 10 000. These parts were used to facilitate the development of the prototype, but could be substituted with more generic components if so desired. While also expensive, the Fathom-X unit and board were necessary, as equivalent products are quite hard to find, especially that can perform as well as these; the acrylic cylinder was also relatively expensive, however, like the Fathom-X board, it was a critical purchase, as it provided a pre-tested enclosure for the device.

6.4 Challenges

To summarise the issues encountered during the project, the major limitations when designing and testing the wiper included:

1. **Margins of Error with the 3D-Printer:**
   When printing components for the wiper, small details such as pilot holes for screws and bolts would often need to be drilled to the specified size due to expansion and shrinking of the PLA. Additionally, circular or curved surfaces would display rough edges that could be obstructed when rotating, which need to be sanded down.

2. **Attaching the brush to the wiper arm:**
   The initial attempts of attaching the brush to the wiper involved the use of a water-resistant epoxy glue. Initially, this appeared to be effective, however, once submerged the bond broke apart. Eventually it was decided to opt for a method of mechanically attaching the brush to the wiper.

3. **Limited space in the enclosure:**
   The limited available space in the enclosure dome posed a significant challenge when designing a mechanical wiper system. Combined with the margins of error due to the 3D-Printing, ensuring full functionality of the system without the wiper potentially damaging the inner surface of the dome or getting caught on the flange was challenging.

4. **The relay board:**
   Due to the elevated power draw of the original stepper motors, a relay board was added in series to their drivers, to cut power to them whenever they weren't doing work. This solution worked well for a period of time, however the relay board eventually stopped working. This is likely due to the age and storage of the board itself, as it was stored in sub-optimal conditions.

6.5 Future Improvements

6.5.1 Corrosion resistance

As the structure will be submerged in seawater for the majority of time, corrosion-proofing the aluminium is key to minimizing the potential of degradation and maximizing it's longevity (Shaw and Kelly 2006). The corrosion of aluminium can be minimized by placing a material in the vicinity that has a higher rate of corrosion. An example of a suitable material is Zinc (J. Piippo 1997). A method of isolation would be to apply an epoxy coating designed for underwater use.

Epoxy Coating

Resimac Ltd. produces a epoxy coating manufactured to protect metallic structures submerged in saltwater. The typical applications include: "Subsea structures, pipelines, risers, splash zones, sheet and bearing piles, other land and marine structures" Other advantages include chemical resistance to salts, alkalies and organic media (ResimacLtd 2023). A downside that follows the use of epoxy coating, is that for maximum waterproofing, certain parts may need trimming to account for the added layer. These layers may vary on how well and equally distributed the epoxy coating is. The *Resimac 208 Ceramic UW* recommends a 1mm layer of coating over the surface area (ResimacLtd 2023). This would mean that the anchor points of the plastic components need to trimmed by 1 mm for each area in direct contact with the profiles.

Figure 6.4: *Resimac 208 Ceramic UW epoxy coating applied on underwater structure (U-CoatThecnologies 2023).*

Using an epoxy like *Resimac 208 Ceramic UW* is a applicable solution when working with metallic materials such as aluminium or steel, however the product is expensive, being sold for $285.00 USD (U-CoatThecnologies 2023). For those that plan on mass-producing similar structures; purchasing a container of epoxy might prove to be more cost efficient. However for those aiming to produce a single or two of structures might find the best cause of action, both economically and timely, to abstain the use of metallic materials in a finished product, and instead go for a plastic solution, such as ABS and Acetals like Delrin mentioned in *Materials Research and Analysis.*

Zinc anodes

Another effective solution is to use Zinc anodes, equally distributed on the aluminium profile construction. Zinc is the most commonly used metal for handling corrosion, and is most often found on steel boats with aluminium propellers. The rate that aluminium corrodes is less than that of Zinc in water, as long as the PH level is less than 11 (J. Piippo 1997).

Having the zinc as anodes equally distributed will also help to maintain the stability of the structure. This will make it easier to lower the structure and zinc into water, compared to lowering the structure and the zinc as two separate entities. If there is a strong current, separate deployments may prove difficult.

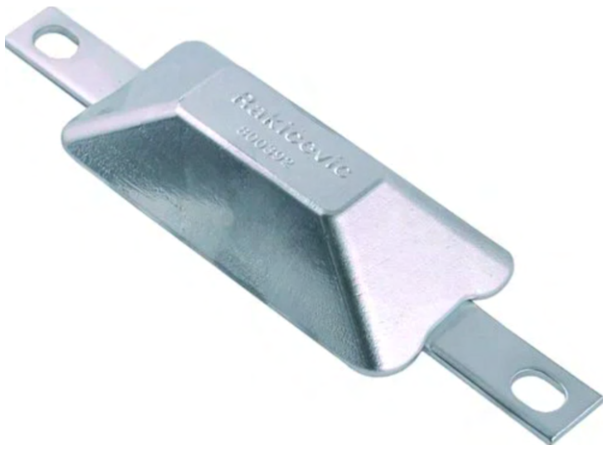


Figure 6.5: *Zinc anode with mounting capabilities(MarineshopAS 2023).*

6.5.2 Plastic Component Alternatives

The custom made PLA parts should be replaced with either ABS or Acetal Plastic, for reasons mentioned in *Materials Research and Analysis, Acetal(POM).* Unfortunately there are few clear specifications of ABS plastic's water resistance, although there is a large library of examples from which educated guesses can be made. Such examples are kitchen utensils, and the fact that the plastic can withstand disadvantageous environmental conditions (plastics 2023). Given the properties and uses of Acetal plastics, Delrin looks to be a solid choice to use for part productions, as it is already being used for underwater applications, like the Maka Niu (Novy et al. 2022).

6.6 Reproducibility

When recreating the physical prototype and to ensure that the results are as close to those that were gathered in this report, certain requirements must be met. Considering the results of the physical prototype, and what was observed, certain changes to the requirements with *Future Improvements* in mind, can be taken into consideration. Since there is no *one* solution, and all the solutions are dependent on the factors mentioned in *Solution to Corrosion*, future renditions are free to chose either of these solutions when reproducing. With that in mind, the prototype is rather easy to assemble, and will by the end be open-source, allowing others to reproduce this work.

1. Observation
   When observing the prototype, a dry run is extremely useful to check for any errors or oversights. When deployed, make use of the hydrophone and the camera within the enclosure to check for problems that were absent during the dry run, such as those observed in *Results*.

2. Location
   Given the nature of the experiment, any large bodies of water could be found useful. For results similar to those found in this report, large bodies of salt-water, such as the Oslofjord, should be considered due to the corrosion/-rust rate being higher than that of freshwater.

3. Power Source
   In order to operate the prototype, access to power is paramount. 48V is the required input voltage which would require a variable laboratory power supply or a transformer to convert the voltage available from a wall socket to the desired voltage.

4. Vacuum pump
   In order to ensure a tight seal on the enclosure, and that there will be no leaks or entry of liquids, it is recommended that a vacuum pump is used prior to deployment. Connect the pump to the rear vent and pump up to 15 psi, check every 5 minutes for 15 to 30 minutes. It is expected to observe a minor drop in pressure, as it needs to stabilize, however if it has decreased by more than 5 psi, it is recommended to reapply lubrication to the o-rings before resealing the enclosure and repeating the vacuum test. If by the end the enclosure has maintained a stable level of pressure for at least 5 minutes, it is safe to conclude that the enclosure is sealed.

5. Profiles and joints
   The aluminium profiles, screws, and l-brackets are all widely available in most hardware stores both physical and online.

6. Custom-made PLA parts
   Each of the parts were originally made out of PLA plastic, I.E the Wipers, and the mounting adapters for holding the enclosure, lights, and hydrophone in place. These pieces can be 3D-printed or made with custom molds.

Chapter 7

Conclusion

In conclusion, this bachelor thesis has explored the possibility of developing a low-cost underwater observatory. Throughout the thesis, a comprehensive analysis of existing solutions has been conducted, leading to the discovery of current limitations and possible improvements. The primary objective of this project was to develop a proof of concept for an affordable stationary rig, and a solution to deal with the applicable biofouling problem during long term deployments in the sea.

Through extensive literature review, development, and experimentation this thesis has successfully addressed the research questions and achieved the set objectives. The findings of this study have provided valuable contributions to the field of underwater observatories, and a potential solution to biofouling accumulation for a significantly reduced price.

One of the key contributions of this thesis lies in the development and implementation of the dome wiper to combat biofouling. The effectiveness and efficiency of the proposed wiper have been rigorously tested, demonstrating it's potential for real-world applications and as an alternative to costlier solutions.

It is worth noting that, like any research, this thesis has certain limitations. The time constraint of the testing period might have had implications on the results gathered in the report. Additionally, as several of the parts used were sourced from the lab, their state or age couldn't be easily determined, increasing their risk of failure when deployed. This caused problems with the first iteration of the wiper mechanism, as the motors' excessive power draw caused their drivers (and connected arduino) to burn out. These limitations should be considered when interpreting the findings and recommendations presented in this thesis, and they provide opportunities for future research and improvement.

In summary, throughout the five months of research, design and work, we have been able to create a functioning proof of concept. The prototype is far cheaper, and has the potential to compete with existing solutions on the market.

Chapter 8

Appendixes

8.1 Design Schematics

All final designs are available for download in .ipt or .stl format at:
https://github.com/OsloMet-OceanLab/aneris

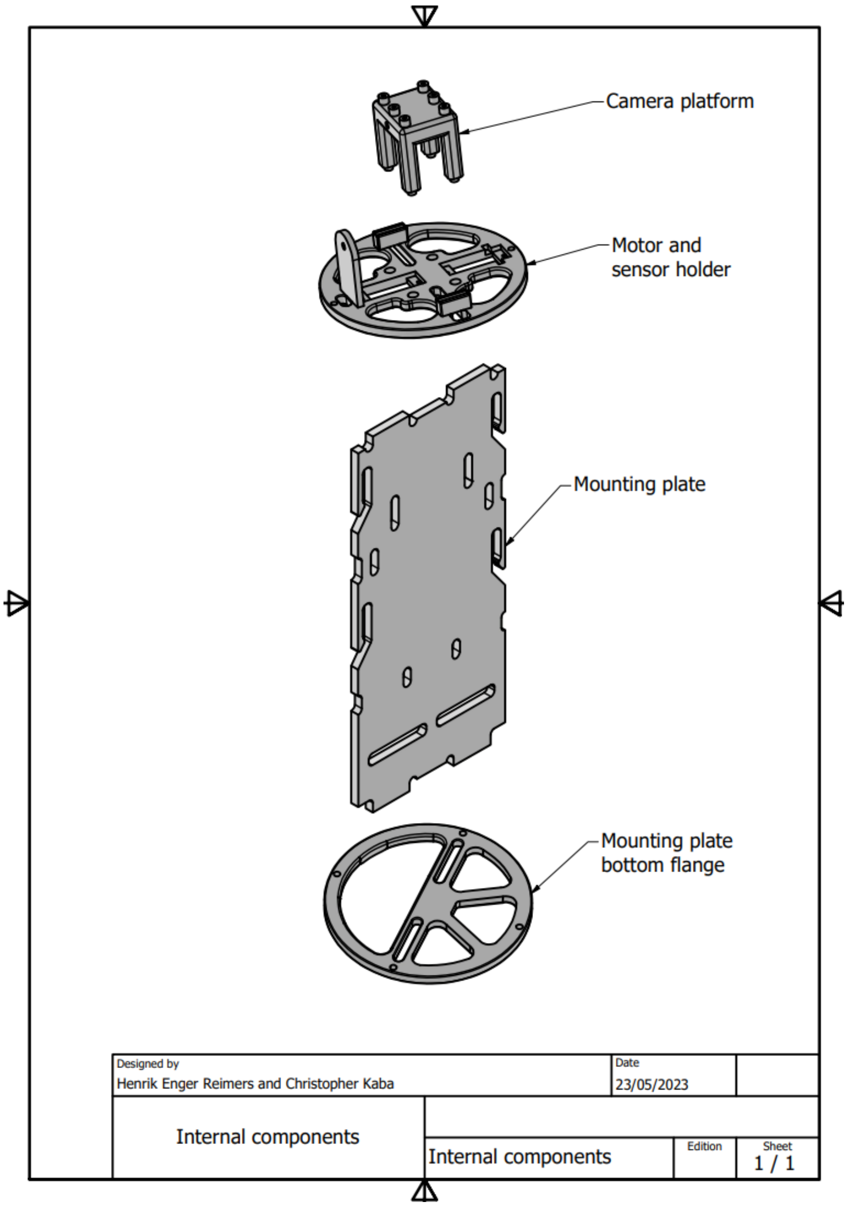


Figure 8.1: *Internal components of the enclosure.*

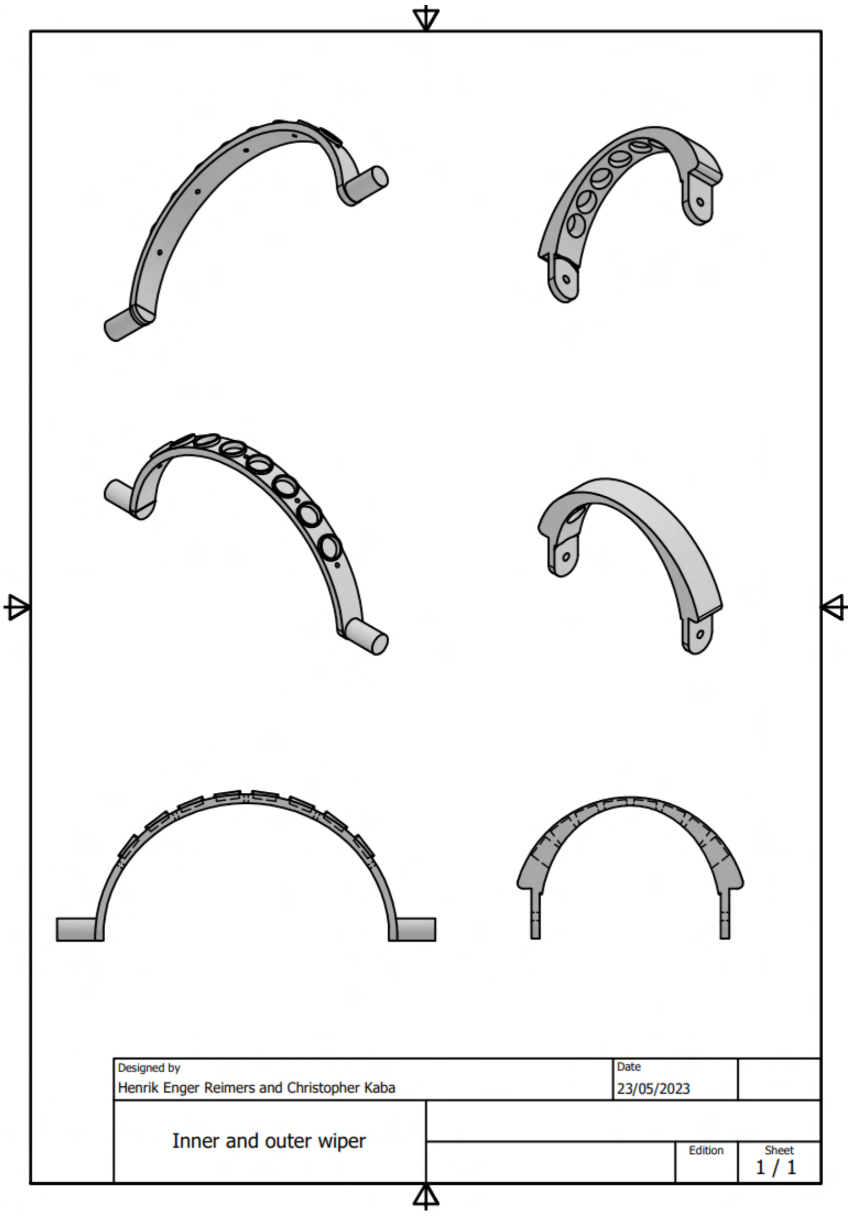


Figure 8.2: *Inner and outer wiper.*

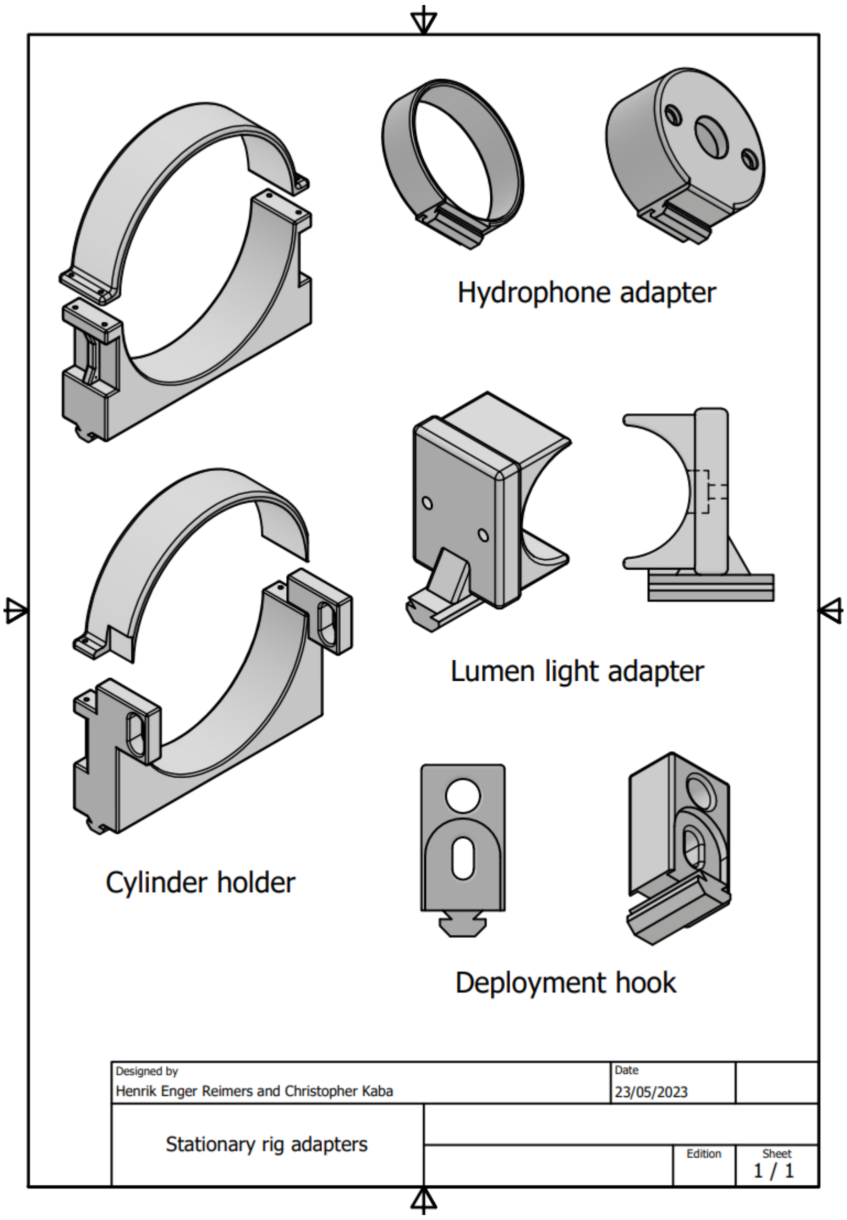


Figure 8.3: *Stationary rig adapters.*

8.2 Source code

All source code is also available at: https://github.com/OsloMet-OceanLab/aneris
N.B. all code retrieved 21.5.2023

8.2.1 aneris-ctl.cpp

```
#include <iostream>
#include <unistd.h>
#include <cstring>
#include <sys/socket.h>
#include <sys/un.h>

#define SOCKET_PATH "/var/run/aneris.sock"
#define COMMAND_SIZE 2

void printHelp(char *pName);

int main(int argc, char **argv)
{
  if (argc != 2)
  {
    printHelp(argv[0]);
    exit(1);
  }

  if (!strncmp("--help", argv[1], 6) || !strncmp("-h", argv[1], 2))
  {
    printHelp(argv[0]);
    exit(0);
  }

  int input = atoi(argv[1]);

  switch(input)
  {
    case 1:
    {
      printf("Powering off device\n");
      break;
    }
    case 2:
    {
      printf("Rebooting device\n");
      break;
    }
    case 3:
    {
      printf("Turning lights on\n");
      break;
    }
    case 4:
    {
      printf("Turning lights off\n");
      break;
    }
    case 5:
    {
      printf("Turning wipers on\n");
      break;
    }
    case 6:
    {
      printf("Turning wipers off\n");
      break;
    }
    case 7:
    {
      printf("Turning porpoise on\n");
      break;
    }
    case 8:
    {
      printf("Turning porpoise off\n");
      break;
    }
    case 9:
    {
      printf("Clearing log file\n");
      break;
    }
    case 10:
    {
      printf("Starting web server\n");
      break;
    }
    case 11:
    {
      printf("Terminating web server\n");
      break;
    }
    default:
    {
      printf("Invalid option\n");
      printHelp(argv[0]);
      exit(3);
    }
  }

  int sock;
  struct sockaddr_un serv;
  char buf[COMMAND_SIZE];
  memset(&serv, 0, sizeof(struct sockaddr_un));

  sock = socket(AF_UNIX, SOCK_DGRAM, 0);
  serv.sun_family = AF_UNIX;
  memcpy(serv.sun_path, SOCKET_PATH, strlen(SOCKET_PATH));

  memcpy(buf, argv[1], COMMAND_SIZE);
  sendto(sock, buf, strlen(buf), 0, (struct sockaddr *) &serv, sizeof(
    serv));
  close(sock);

  return 0;
}

```

```
void printHelp(char *pName)
{
  printf("Usage: %s <option>\n", pName);
  printf("Available options:\n");
  printf("1\tPower device off\n");
  printf("2\tReboot device\n");
  printf("3\tTurn lights on\n");
  printf("4\tTurn lights off\n");
  printf("5\tTurn wipers on\n");
  printf("6\tTurn wipers off\n");
  printf("7\tTurn porpoise on\n");
  printf("8\tTurn wipers off\n");
  printf("9\tClear log file\n");
  printf("10\tStart web server\n");
  printf("11\tTerminate web server\n");
}
```

8.2.2 aneris.cpp

```
/*************************************/
/* aneris project                    */
/* insert information on license     */
/* and authors here                  */
/*                                   */
/* brief description of the program */
/* as well                           */
/*************************************/

#include <iostream>
#include <unistd.h>
#include <cstdlib>
#include <csignal>
#include <cstring>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <pthread.h>

#include "gpio/GPIO.hpp"
#include "Logger/Logger.hpp"
#include "Web_Server/Web_Server.hpp"

#define WEB_SERVER_PORT 80

#define SOCKET_PATH "/var/run/aneris.sock"
#define COMMAND_SIZE 2

#define GPIO_LIGHTS 4
#define GPIO_TEST 17
#define GPIO_HYDROPHONE 27
#define GPIO_WIPER 22

void sig_handler(int signum);

int main(void)
{
  /***************************/
  /* register signal handler */
```

```
/**************************/
std::signal(SIGINT, sig_handler);
std::signal(SIGTERM, sig_handler);
std::signal(SIGABRT, sig_handler);
std::signal(SIGSEGV, sig_handler);
std::signal(SIGFPE, sig_handler);
std::signal(SIGILL, sig_handler);

/*********************/
/* verify user is root */
/*********************/
if(geteuid())
{
  Logger::log(Logger::LOG_FATAL, "This program needs to be run as
 root");
  exit(1);
} else Logger::log(Logger::LOG_INFO, "Verified user permission");

/**************************/
/* verify gpio is available */
/**************************/
try
{
  gpio::GPIO *test_gpio = new gpio::GPIO(GPIO_TEST, gpio::GPIO_INPUT)
 ;
  if(!test_gpio) throw gpio::GPIOError("Couldn't allocate memory for
 'test_gpio' variable");
  if(test_gpio->getval()) Logger::log(Logger::LOG_INFO, "Verified
 GPIO is available");
  else Logger::log(Logger::LOG_ERROR, "GPIO unavailable");
  delete test_gpio;
}
catch(gpio::GPIOError& e)
{
  Logger::log(Logger::LOG_ERROR, e.what());
  Logger::log(Logger::LOG_ERROR, "GPIO unavailable");
}

/***************************************/
/* initialize GPIO pins to default value */
/***************************************/
gpio::GPIO *lights_tmp = nullptr;
try
{
  lights_tmp = new gpio::GPIO(GPIO_LIGHTS, gpio::GPIO_OUTPUT);
  if(!lights_tmp) throw gpio::GPIOError("Couldn't allocate memory for
  'lights_tmp' variable");
  lights_tmp->setval(gpio::GPIO_HIGH);
  Logger::log(Logger::LOG_INFO, "Enabled lights");
}
catch(gpio::GPIOError& e)
{
  Logger::log(Logger::LOG_ERROR, e.what());
}
delete lights_tmp;

gpio::GPIO *wiper_tmp = nullptr;
try
```

```
  {
    wiper_tmp = new gpio::GPIO(GPIO_WIPER, gpio::GPIO_OUTPUT);
    if(!wiper_tmp) throw gpio::GPIOError("Couldn't allocate memory for
   'wiper_tmp' variable");
    wiper_tmp->setval(gpio::GPIO_LOW);
    Logger::log(Logger::LOG_INFO, "Enabled wiper");
  }
  catch(gpio::GPIOError& e)
  {
    Logger::log(Logger::LOG_ERROR, e.what());
  }
  delete wiper_tmp;

  gpio::GPIO *hydrophone_tmp = nullptr;
  try
  {
    hydrophone_tmp = new gpio::GPIO(GPIO_HYDROPHONE, gpio::GPIO_OUTPUT)
   ;
    if(!hydrophone_tmp) throw gpio::GPIOError("Couldn't allocate memory
    for 'hydrophone_tmp' variable");
    hydrophone_tmp->setval(gpio::GPIO_LOW);
    Logger::log(Logger::LOG_INFO, "Enabled hydrophone");
  }
  catch(gpio::GPIOError& e)
  {
    Logger::log(Logger::LOG_ERROR, e.what());
  }
  delete hydrophone_tmp;

  /****************************************/
  /* test uplink to fathom tether interface */
  /*                                      */
  /* 0 on successful ping, any other      */
  /* integer otherwise                    */
  /****************************************/
  int ping_counter = 0, tether_up = 1;
  do
  {
    Logger::logf(Logger::LOG_INFO, "Testing connection to land, attempt
    %d", ping_counter + 1);
    tether_up = system("ping -c 1 192.168.2.1");

    if(tether_up && ping_counter < 2)
    {
      ++ping_counter;
      sleep(3);
    } else break;
  } while(true);

  if(tether_up) Logger::log(Logger::LOG_WARN, "No connection to land");
  else Logger::log(Logger::LOG_INFO, "Verified connection to land is
   available");

  /****************************************/
  /* test hydrophone is connected         */
  /*                                      */
  /* currently checks that led pin is     */
  /* enabled, but could also check that   */
```

```
/* network card is up? TBD                 */
/* N.B. most likely unnecessary, can       */
/* probably be deleted                     */
/*****************************************/
/*try
{
  gpio::GPIO *hydrophone = new gpio::GPIO(GPIO_HYDROPHONE, gpio::
 GPIO_INPUT);
  if(!hydrophone) throw gpio::GPIOError("Couldn't allocate memory for
  'hydrophone' variable");
  if(hydrophone->getval()) Logger::log(Logger::LOG_INFO, "Verified
 Hydrophone is available");
  else Logger::log(Logger::LOG_ERROR, "Hydrophone is not available");
  delete hydrophone;
}
catch(gpio::GPIOError& e)
{
  Logger::log(Logger::LOG_ERROR, e.what());
  Logger::log(Logger::LOG_ERROR, "Hydrophone is not available");
}*/

/************************/
/* start command listener */
/************************/
int sock, len;
struct sockaddr_un server_sockaddr, peer_sock;
char buf[COMMAND_SIZE];
memset(&server_sockaddr, 0, sizeof(struct sockaddr_un));

sock = socket(AF_UNIX, SOCK_DGRAM, 0);
if (sock < 0)
{
  Logger::log(Logger::LOG_FATAL, "Couldn't create socket");
  exit(2);
}

server_sockaddr.sun_family = AF_UNIX;
memcpy(server_sockaddr.sun_path, SOCKET_PATH, strlen(SOCKET_PATH));
len = sizeof(server_sockaddr);
unlink(SOCKET_PATH);

if (bind(sock, (struct sockaddr*) &server_sockaddr, len) < 0)
{
  Logger::log(Logger::LOG_FATAL, "Couldn't bind to socket");
  close(sock);
  exit(2);
}

/************************************************/
/* start python web server                        */
/*                                                */
/* N.B. do NOT modify the 'PYTHONPATH' environment */
/* variable, it WILL break the program            */
/************************************************/
setenv("PYTHONPATH", ".", 1);
pthread_t ws_thread;
bool thread_running;
int ws_port = WEB_SERVER_PORT;
```

```
if (pthread_create(&ws_thread, NULL, &Web_Server::serve, &ws_port))
{
  Logger::log(Logger::LOG_FATAL, "Couldn't start web server process");
  close(sock);
  exit(3); //system("reboot");
}
Logger::log(Logger::LOG_INFO, "Started web server process");
thread_running = true;

/******************/
/* begin main loop */
/******************/
Logger::log(Logger::LOG_INFO, "Listening to socket");
while (true)
{
  memset(buf, 0, COMMAND_SIZE);
  if (recvfrom(sock, buf, COMMAND_SIZE, 0, (struct sockaddr*) &peer_sock, (socklen_t*) &len) < 0)
  {
    Logger::log(Logger::LOG_FATAL, "Couldn't receive data");
    exit(2);
  }

  switch(atoi(buf))
  {
    case 0: // handle received commands that are not numerical and/or atoi() errors
    {
      Logger::log(Logger::LOG_ERROR, "Non numerical command received");
      break;
    }
    case 1: // shutdown
    {
      Logger::log(Logger::LOG_INFO, "Received shutdown command");
      system("poweroff");
    }
    case 2: // reboot
    {
      Logger::log(Logger::LOG_INFO, "Received reboot command");
      system("reboot");
    }
    case 3: // turn lights on
    {
      gpio::GPIO *lights = nullptr;
      try
      {
        lights = new gpio::GPIO(GPIO_LIGHTS, gpio::GPIO_OUTPUT);
        if(!lights) throw gpio::GPIOError("Couldn't allocate memory for 'lights' variable");
        lights->setval(gpio::GPIO_HIGH);
        Logger::log(Logger::LOG_INFO, "Enabled lights");
      }
      catch(gpio::GPIOError& e)
      {
        Logger::log(Logger::LOG_ERROR, e.what());
      }
```

```
        delete lights;
        break;
      }
      case 4: // turn lights off
      {
        gpio::GPIO *lights = nullptr;
        try
        {
          lights = new gpio::GPIO(GPIO_LIGHTS, gpio::GPIO_OUTPUT);
          if(!lights) throw gpio::GPIOError("Couldn't allocate memory
    for 'lights' variable");
          lights->setval(gpio::GPIO_LOW);
          Logger::log(Logger::LOG_INFO, "Disabled lights");
        }
        catch(gpio::GPIOError& e)
        {
          Logger::log(Logger::LOG_ERROR, e.what());
        }
        delete lights;
        break;
      }
      case 5: // turn wipers on
      {
        gpio::GPIO *wiper = nullptr;
        try
        {
          wiper = new gpio::GPIO(GPIO_WIPER, gpio::GPIO_OUTPUT);
          if(!wiper) throw gpio::GPIOError("Couldn't allocate memory
    for 'wiper' variable");
          wiper->setval(gpio::GPIO_LOW);
          Logger::log(Logger::LOG_INFO, "Enabled wiper");
        }
        catch(gpio::GPIOError& e)
        {
          Logger::log(Logger::LOG_ERROR, e.what());
        }
        delete wiper;
        break;
      }
      case 6: // turn wipers off
      {
        gpio::GPIO *wiper = nullptr;
        try
        {
          wiper = new gpio::GPIO(GPIO_WIPER, gpio::GPIO_OUTPUT);
          if(!wiper) throw gpio::GPIOError("Couldn't allocate memory
    for 'wiper' variable");
          wiper->setval(gpio::GPIO_HIGH);
          Logger::log(Logger::LOG_INFO, "Disabled wiper");
        }
        catch(gpio::GPIOError& e)
        {
          Logger::log(Logger::LOG_ERROR, e.what());
        }
        delete wiper;
        break;
      }
      case 7: // turn porpoise on
      {
        gpio::GPIO *hydrophone = nullptr;
        try
        {
          hydrophone = new gpio::GPIO(GPIO_HYDROPHONE, gpio::GPIO_OUTPUT);
          if(!hydrophone) throw gpio::GPIOError("Couldn't allocate memory for 'hydrophone' variable");
          hydrophone->setval(gpio::GPIO_LOW);
          Logger::log(Logger::LOG_INFO, "Enabled hydrophone");
        }
        catch(gpio::GPIOError& e)
        {
          Logger::log(Logger::LOG_ERROR, e.what());
        }
        delete hydrophone;
        break;
      }
      case 8: // turn porpoise off
      {
        gpio::GPIO *hydrophone = nullptr;
        try
        {
          hydrophone = new gpio::GPIO(GPIO_HYDROPHONE, gpio::GPIO_OUTPUT);
          if(!hydrophone) throw gpio::GPIOError("Couldn't allocate memory for 'hydrophone' variable");
          hydrophone->setval(gpio::GPIO_HIGH);
          Logger::log(Logger::LOG_INFO, "Disabled hydrophone");
        }
        catch(gpio::GPIOError& e)
        {
          Logger::log(Logger::LOG_ERROR, e.what());
        }
        delete hydrophone;
        break;
      }
      case 9: // clean up log file
      {
        Logger::clearLog();
        break;
      }
      case 10: // start web server process
      {
        if(!thread_running)
        {
          if (pthread_create(&ws_thread, NULL, &Web_Server::serve, &ws_port))
          {
            Logger::log(Logger::LOG_FATAL, "Couldn't start web server process");
            exit(3);
          }
          Logger::log(Logger::LOG_INFO, "Started web server process");
          thread_running = true;
        }
        break;
      }
```

```
      case 11: // end web server process
      {
        if(thread_running)
        {
          Logger::log(Logger::LOG_INFO, "Ending web server process");
          if (pthread_cancel(ws_thread)) Logger::log(Logger::LOG_ERROR, "Couldn't stop web server process");
          else Logger::log(Logger::LOG_INFO, "Ended web server process");
          thread_running = false;
        }
        break;
      }
      default: // return that command is invalid
      {
        Logger::log(Logger::LOG_INFO, "Received an invalid command");
        close(sock);
        exit(0);
      }
    }
  }
}

void sig_handler(int signum)
{
  switch(signum)
  {
    case 2:
    {
      Logger::log(Logger::LOG_INFO, "Received SIGINT signal");
      break;
    }
    case 3:
    {
      Logger::log(Logger::LOG_INFO, "Received SIGILL signal");
      break;
    }
    case 6:
    {
      Logger::log(Logger::LOG_INFO, "Received SIGABRT signal");
      break;
    }
    case 8:
    {
      Logger::log(Logger::LOG_INFO, "Received SIGFPE signal");
      break;
    }
    case 11:
    {
      Logger::log(Logger::LOG_INFO, "Received SIGSEGV signal");
      break;
    }
    case 15:
    {
      Logger::log(Logger::LOG_INFO, "Received SIGTERM signal");
      break;
    }
    default:
```

```
    {
      Logger::log(Logger::LOG_INFO, "Received unknown signal");
      break;
    }
  }

  exit(signum);
}
```

8.2.3 aneris.service

```
[Unit]
Description=Aneris streaming service

[Service]
User=root
WorkingDirectory=/etc/aneris
ExecStart=/etc/aneris/aneris
Restart=always

[Install]
WantedBy=multi-user.target
```

8.2.4 audio_recv.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 22 10:11:51 2023

@author: salve
"""

from socket import socket, AF_INET, SOCK_DGRAM
import sys
from bit_converter import bytes_to_16

def genHeader(sampleRate, bitsPerSample, channels, samples):
    datasize = samples #* channels * bitsPerSample // 8
    endian = 'little'
    sign = True
    o = bytes("RIFF", 'ascii')
    o += (datasize + 36).to_bytes(4, endian, signed = sign)
    o += bytes("WAVE", 'ascii')
    o += bytes("fmt ", 'ascii')
    o += (16).to_bytes(4, endian, signed = sign)
    o += (1).to_bytes(2, endian, signed = sign)
    o += (channels).to_bytes(2, endian, signed = sign)
    o += (sampleRate).to_bytes(4, endian, signed = sign)
    o += (sampleRate * channels * bitsPerSample // 8).to_bytes(4,
endian, signed = sign)
    o += (channels * bitsPerSample // 8).to_bytes(2, endian, signed =
sign)
    o += (bitsPerSample).to_bytes(2, endian, signed = sign)
    o += bytes("data", 'ascii')
    o += (datasize).to_bytes(4, endian, signed = sign)
    return o

HOST, PORT = '', 5453
```

```python

d = b''
i = 0
raw_size = 0

with socket(AF_INET, SOCK_DGRAM) as sock:
    sock.bind((HOST, PORT))
    print("Bound")
    try:
        while True:
            message, _ = sock.recvfrom(1536)
            sync = message.hex()[0:4]
            print(f"Sync: {sync}")
            size = int(message.hex()[4:8], 16)
            print(f"Size: {size}")
            if message.hex()[8:10] == '00':
                dtype = 'Unknown'
            elif message.hex()[8:10] == '01':
                dtype = 'DAQ'
            elif message.hex()[8:10] == '02':
                dtype = 'Noise'
            print(f"Data type: {dtype}")

            data = message[5:size]

            if data.hex()[0:2] == '00':
                dformat = 'Unknown'
            elif data.hex()[0:2] == '01':
                dformat = 'PCM16'
            elif data.hex()[0:2] == '02':
                dformat = 'PCM24'
            print(f"Data format: {dformat}")
            seq = data.hex()[2:6]
            print(f"Sequence number: {seq}")
            sr = int(data.hex()[6:14], 16)
            print(f"Sample rate: {sr}")
            chmap = int(data.hex()[14:22], 16)
            print(f"Enabled channels: {chmap}")
            scnt = int(data.hex()[22:26], 16)
            print(f"Samples per channel: {scnt}")
            raw = data[13:]#(3*scnt*chmap)+13]
            print(raw.hex())

            d += bytes_to_16(raw)
            raw_size += scnt * 3
            i += 1

            if i % 600 == 0:
                import matplotlib.pyplot as plt, numpy as np

                plt.figure()
                d2 = np.frombuffer(d, dtype=np.int32)
                plt.plot(d2)
                print(d2)
                print(max(d2))
                print(min(d2))
                print(min(abs(d2)))
                plt.xlabel('Sample index')
                plt.ylabel('Amplitude')
                plt.title('Waveform')
                plt.show()
                sys.exit(0)

    except KeyboardInterrupt:
        print("Done")
```

8.2.5 bit_converter.hpp

```
#ifndef BIT_CONVERTER_HPP
#define BIT_CONVERTER_HPP

#include <cstdint>

#ifdef CPYTHON

#include <pybind11/pybind11.h>
namespace py = pybind11;

#endif

namespace bit_converter
{

#ifdef CPYTHON
py::bytes
#else
char *
#endif
bytes_to_16(const char *buf = nullptr, size_t len = 0, bool
    invertEndianness = true);

#ifdef CPYTHON
py::bytes
#else
char *
#endif
bytes_to_32(const char *buf = nullptr, size_t len = 0, bool
    invertEndianness = true);

#ifdef CPYTHON
py::bytes
#else
char *
#endif
invert_endianness_24(const char *buf = nullptr, size_t len = 0);

} // end namespace bit_converter

#ifdef CPYTHON

PYBIND11_MODULE(bit_converter, m)
{
        m.doc() = "Convert a 24 bit signed buffer to either 16 or 32
    bit";

        m.def("bytes_to_16", &bit_converter::bytes_to_16, "Convert a 24
 bit sample to 16 bit",
                py::arg("buf") = nullptr, py::arg("len") = 0, py::arg("
    invertEndianness") = true);
        m.def("bytes_to_32", &bit_converter::bytes_to_32, "Convert a 24
 bit sample to 32 bit",
                py::arg("buf") = nullptr, py::arg("len") = 0, py::arg("
    invertEndianness") = true);
        m.def("invert_endianness_24", &bit_converter::
    invert_endianness_24, "Change the endianness of the sample",
                py::arg("buf") = nullptr, py::arg("len") = 0);
}

#endif

#endif
```

8.2.6 bit_converter.cpp

```
#include "bit_converter.hpp"

#include <array>
#include <cstdint>
#include <string>

namespace bit_converter
{

static inline int16_t _24to16(const char byteArray[])
{
    return (((uint16_t) (byteArray[1] << 8)) | (uint8_t) byteArray[2]);
}

#ifdef CPYTHON
py::bytes
#else
char *
#endif
bytes_to_16(const char *buf, size_t len, bool invertEndianness)
{
    if (!buf || !len) return nullptr;

    int16_t *tmp_arr = new int16_t[len/3];
    std::string newbuf;

    for (size_t i = 0; i < len; i += 3)
        *(tmp_arr + (i/3)) = _24to16(buf+i);

    for (size_t i = 0; i < 2*len/3; i += 2)
    {
        newbuf += 0xFF & (tmp_arr[i/2] >> (invertEndianness ? 0 : 8));
        newbuf += 0xFF & (tmp_arr[i/2] >> (invertEndianness ? 8 : 0));
    }

    delete[] tmp_arr;

#ifdef CPYTHON
    return py::bytes(newbuf);
#else
```

```
    return newbuf.c_str();
#endif
}

static inline int32_t _24to32(std::array<uint8_t, 3> byteArray)
{
    return (((int32_t)(byteArray[0]) << 24) |
            ((int32_t)(byteArray[1]) << 16) |
            ((int32_t)(byteArray[2]) << 8)) >> 8;
}

#ifdef CPYTHON
py::bytes
#else
char *
#endif
bytes_to_32(const char *buf, size_t len, bool invertEndianness)
{
    if (!buf || !len) return nullptr;

    int32_t *tmp_int_arr = new int32_t[len/3];
    std::array<uint8_t, 3> tmparr;
    std::string newbuf;

    for (size_t i = 0; i < len; i += 3)
    {
        tmparr[0] = (uint8_t) buf[i];
        tmparr[1] = (uint8_t) buf[i+1];
        tmparr[2] = (uint8_t) buf[i+2];

        *(tmp_int_arr + (i/3)) = _24to32(tmparr);
    }

    for (size_t i = 0; i < 4*len/3; i += 4)
    {
        newbuf += 0xFF & (tmp_int_arr[i/4] >> (invertEndianness ? 0 : 24));
        newbuf += 0xFF & (tmp_int_arr[i/4] >> (invertEndianness ? 8 : 16));
        newbuf += 0xFF & (tmp_int_arr[i/4] >> (invertEndianness ? 16 : 8));
        newbuf += 0xFF & (tmp_int_arr[i/4] >> (invertEndianness ? 24 : 0));
    }

    delete[] tmp_int_arr;

#ifdef CPYTHON
    return py::bytes(newbuf);
#else
    return newbuf.c_str();
#endif
}

#ifdef CPYTHON
py::bytes
#else
char *
```

```
#endif
invert_endianness_24(const char *buf, size_t len)
{
    if (!buf || !len) return nullptr;

    std::string newbuf;

    for (size_t i = 0; i < len; i += 3)
    {
        newbuf += (uint8_t) *(buf + i + 2);
        newbuf += (uint8_t) *(buf + i + 1);
        newbuf += (uint8_t) *(buf + i);
    }

#ifdef CPYTHON
    return py::bytes(newbuf);
#else
    return newbuf.c_str();
#endif
}

} // end namespace bit_converter
```

8.2.7 console.html

```
<html>
<body>

    <div style="text-align:center;">
        <h1>Console page</h1>
        <p>Use this page to control the device</p>
        <div class="flex-container">
          <!--  <form method="POST">
                <button name="command" value=3 type="submit"> Lights On </button>
            </form>

            <form method="POST">
                <button name="command" value=4 type="submit"> Lights Off</button>
            </form>

             <form method="POST">
                <button name="wipersOn" value=5 type="submit"> Wipers on</button>
            </form>

              <form method="POST">
                <button name="wipersOff" value=6 type="submit"> Wipers off</button>
            </form>

  <form method="POST">
                <button name="CleanLog" value=7 type="submit"> Clean up log files</button>
            </form>
            -->
            <button onclick="addCom(3)">Lights On</button>
            <button onclick="addCom(4)">Lights Off</button>
            <button onclick="addCom(5)">Wipers on</button>
            <button onclick="addCom(6)">Wipers Off</button>
            <button onclick="addCom(7)">Clean up log files</button>

        </div>


            <!--
             <form method="POST" action="/console">
            <input name="command" type="text" placeholder="Write a
    command here" />
            <input type="submit" value="Submit" />
        </form>
        -->


    </div>
    <style>
        .flex-container {
            display: flex;
            flex-wrap: nowrap;
            justify-content: center;
        }

            .flex-container > form {
                margin: 0.1em;
                text-align: center;
                line-height: 75px;
                font-size: 30px;
            }
    </style>
    <script>

        function addCom(buttonvalue) {
            let testUrl = 'https://httpbin.org/post';
            let url = 'http://10.44.6.51/api';
            //let url = 'api';
            let buttonInfo = {
                command: buttonvalue
            }
            fetch(url, {
                method: 'POST',
                headers: { 'Content-type': 'application/json;' },
                body: JSON.stringify(buttonInfo),
            })
                    .then(res => res.json())
                .then(json => console.log(buttonvalue));
            }

    </script>
</body>
</html>
```

8.2.8 GPIO.hpp

```
#ifndef GPIO_HPP
#define GPIO_HPP
```

```

#include <exception>

namespace gpio
{

enum
{
    GPIO_LOW =      0x1,
    GPIO_HIGH =     0x2,

    GPIO_INPUT =    0x4,
    GPIO_OUTPUT =   0x8
};

class GPIO
{
public:
    explicit GPIO(int pin = 4, long dir = GPIO_INPUT);
    virtual ~GPIO() noexcept(false); // N.B. allowing the dtor to throw exceptions is not recommended, but should be fine as they're handled
    void setval(long val); // Set GPIO Value (high/low)
    int getval(); // Get GPIO Value
    int get_gpionum();
    long get_direction();

private:
    int gpionum; // GPIO number associated with the instance of an object
    long direction;
};

class GPIOError : std::exception
{
public:
    GPIOError(const char* message)
    {
        this->msg = message;
    }
    virtual const char* what()
    {
        return this->msg;
    }
private:
    const char* msg;
};

} // end namespace gpio

#endif
```

8.2.9 GPIO.cpp

```
#include <fstream>
#include <string>
//#include <iostream>

```

```cpp
#include "GPIO.hpp"

namespace gpio
{

GPIO::GPIO(int pin, long dir)
{
    this->gpionum = pin; // default is GPIO4
    this->direction = dir;

    // Export the pin
    std::ofstream export_pin("/sys/class/gpio/export", std::ofstream::out);
    if (!export_pin.is_open()) throw GPIOError("Unable to export GPIO");
    // Write the GPIO number to export
    export_pin << this->gpionum;
    export_pin.close();

    // Open the direction file for gpio
    std::string setdir_str = "/sys/class/gpio/gpio" + std::to_string(pin) + "/direction";
    std::ofstream setdir(setdir_str.c_str(), std::ofstream::out);
    if (!setdir.is_open()) throw GPIOError("Unable to set the direction of GPIO");
    // Write the direction to direction file
    if (this->direction & GPIO_INPUT) setdir << "in";
    else if (this->direction & GPIO_OUTPUT) setdir << "out";
    else throw GPIOError("Invalid direction chosen");
    setdir.close();
}

GPIO::~GPIO() noexcept(false)
{
    // Unexport the pin
    std::ofstream ofs("/sys/class/gpio/unexport", std::ofstream::out);
    if (!ofs.is_open()) throw GPIOError("Unable to unexport GPIO");
    // Write GPIO number to unexport
    ofs << this->gpionum;
    ofs.close();
}

void GPIO::setval(long val)
{
    if (this->direction & GPIO_INPUT) throw GPIOError("Trying to set an output value on an input");

    // Open the value file for gpio
    std::string setval_str = "/sys/class/gpio/gpio" + std::to_string(this->gpionum) + "/value";
    std::ofstream ofs(setval_str.c_str());
    if (!ofs.is_open()) throw GPIOError("Unable to set the value of GPIO");
    // Write the value to value file
    if (val & GPIO_HIGH) ofs << 1;
    else if (val & GPIO_LOW) ofs << 0;
    else throw GPIOError("Invalid value chosen");
    ofs.close();
}

int GPIO::getval()
{
    if (this->direction & GPIO_OUTPUT) throw GPIOError("Trying to read
 an input value on an output");

    // Open the value file for gpio
    std::string getval_str = "/sys/class/gpio/gpio" + std::to_string(
 this->gpionum) + "/value";
    std::ifstream ifs(getval_str.c_str());
    if (!ifs.is_open()) throw GPIOError("Unable to read the value of
 GPIO");
    // Read the current gpio value
    int val;
    ifs >> val;
    ifs.close();

    if(val) val = 1;
    return val;
}

int GPIO::get_gpionum()
{
    return this->gpionum;
}

long GPIO::get_direction()
{
    return this->direction;
}

} // end namespace gpio
```

8.2.10 index.html

```
<html>
<head>
    <title>Aneris Bachelor project web server</title>
</head>
<body>
    <div id="header">
        <h1 style="text-align:center;">This is the index page</h1>
        <h2 style="text-align:center;">Find your way to another page:</
 h2>
    </div>
    <div id="popups" class="container">
            <div>
                <button onclick="openStreamModal()">Video stream</
 button>
                <div id="streamModal" class="modal">
                    <div class="modal-content">
                        <video autoplay id="stream">
                            <source src="/stream" type="video/mpeg" />
                        </video>
                        <span class="close">&times;</span>
                        <p>
                            <a href="/stream" style="text-align:center;
```

```
">Video stream</a>
                     </p>
                 </div>
             </div>
         </div>

         <div>
             <button onclick="openLogModal()">System logs</button>
             <div id="logsModal" class="modal">
                 <div class="modal-content">
                     <span class="close">&times;</span>
                     <iframe src="/logs" title="logs" class="modal-child"></iframe>
                     <p>
                         <a href="/logs" style="text-align:center;"> System logs</a>
                     </p>
                 </div>
             </div>
         </div>
         <div>
             <button onclick="openDocumentModal()">Helpful docs</button>
             <div id="docsModal" class="modal">
                 <div class="modal-content">
                     <span class="close">&times;</span>
                     <iframe src="/docs" title="docs" class="modal-child"></iframe>
                     <p>
                         <a href="/docs" style="text-align:center;"> Helpful documents</a>
                     </p>
                 </div>
             </div>
         </div>
         <div>
             <button onclick="openConsoleModal()">Get Console</button>

             <div id="consoleModal" class="modal">
                 <div class="modal-content">
                     <span class="close">&times;</span>
                     <iframe src="console.html" title="Console" class="modal-child"></iframe>
                     <p>
                         <a href="console.html" style="text-align:center;">Console</a>
                     </p>
                 </div>
             </div>
         </div>

 </div>
 <style>
     .container {
         display: flex;
         flex-direction: column;

```

```
            justify-content: center;
        }

            .container > div {
                margin: 0.1em;
                text-align: center;
                line-height: 75px;
                font-size: 30px;
            }

        .modal {
            display: none; /* Hidden by default */
            position: fixed; /* Stay in place */
            z-index: 1; /* Sit on top */
            padding-top: 100px; /* Location of the box */
            left: 0;
            top: 0;
            width: 100%; /* Full width */
            height: 100%; /* Full height */
            overflow: auto; /* Enable scroll if needed */
            background-color: rgb(0,0,0); /* Fallback color */
            background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
        }

        /* Modal Content */
        .modal-content {
            background-color: #fefefe;
            margin: auto;
            padding: 20px;
            border: 1px solid #888;
            width: 80%;
            height: 50%; /* Full height */
        }

        /* The Close Button */
        .close {
            color: #aaaaaa;
            float: right;
            font-size: 28px;
            font-weight: bold;
        }

            .close:hover,
            .close:focus {
                color: #000;
                text-decoration: none;
                cursor: pointer;
            }
        .modal-child {
            width:100%;
            height:80%;
        }
    </style>
    <script type="text/javascript" src="https://code.jquery.com/jquery
-3.6.0.min.js"></script>

    <script>
        function genModal(ModalId,i) {
            var span = document.getElementsByClassName("close")[i];
            ModalId.style.display = "block";
            span.onclick = function () {
                ModalId.style.display = "none";
            }
            window.onclick = function (event) {
                if (event.target == ModalId) {
                    ModalId.style.display = "none";
                }
            }
        }

        function openStreamModal() {//Remember, every modal and id has
    to be unique.
            var modal = document.getElementById("streamModal");
            var id = 0;
            genModal(modal, id);
        }
        function openLogModal() {
            var modal = document.getElementById("logsModal");
            var id = 1;
            genModal(modal, id);
        }


        function openDocumentModal() {
            var modal = document.getElementById("docsModal");
            var id = 2;
            genModal(modal, id);

        }
        function openConsoleModal() {
            var modal = document.getElementById("consoleModal");
            var id = 3;
            genModal(modal, id);


        }
    </script>
</body>
</html>
```

8.2.11 Logger.hpp

```
#ifndef LOG_HPP
#define LOG_HPP

#include <string>
#include <cstdarg>

namespace Logger
{

enum
{
  LOG_INFO =    0x1, // general information (e.g. everything works)
  LOG_WARN =    0x2, // can cause oddities (e.g. can't ping client on
    land)
  LOG_ERROR =   0x4, // fatal to an operation but not a program (e.g.
   gpio is inaccessible)
  LOG_FATAL =   0x8  // fatal to the whole program (e.g. can't start
   the web server)
};

void log(long level, std::string logMsg);
void logf(long level, const char *fmt, ...);
void clearLog();

} // end namespace Logger

#endif
```

8.2.12 Logger.cpp

```
#include "Logger.hpp"
#include <fstream>
#include <ctime>
#include <string>
#include <cstdarg>

#define FILEPATH "/var/log/aneris/aneris.log"
#define F_BUFSIZE 256

namespace Logger
{

static std::string getCurrentTime()
{
  time_t now = time(0);
  struct tm tstruct = *localtime(&now);
  char buf[40];
  strftime(buf, sizeof(buf), "%Y-%m-%d %X", &tstruct);
  return std::string(buf);
}

void log(long level, std::string logMsg)
{
  std::string strLevel = "[]";

  if(level & LOG_INFO) strLevel = "[INFO]";
  else if(level & LOG_WARN) strLevel = "[WARN]";
  else if(level & LOG_ERROR) strLevel = "[ERROR]";
  else if(level & LOG_FATAL) strLevel = "[FATAL]";

  // out flag allows to write, app flag allows to not overwrite file
  std::ofstream ofs(FILEPATH, std::ios_base::out | std::ios_base::app);
  ofs << strLevel << '\t' << getCurrentTime() << '\t' << logMsg << '\n'
   ;
  ofs.close();
}

void logf(long level, const char *fmt, ...)
{
  std::string strLevel = "[]";
  char logMsg[F_BUFSIZE];

  if(level & LOG_INFO) strLevel = "[INFO]";
  else if(level & LOG_WARN) strLevel = "[WARN]";
  else if(level & LOG_ERROR) strLevel = "[ERROR]";
  else if(level & LOG_FATAL) strLevel = "[FATAL]";

  va_list args;
  va_start(args, fmt);
  vsnprintf(logMsg, F_BUFSIZE, fmt, args);

  std::ofstream ofs(FILEPATH, std::ios_base::out | std::ios_base::app);
  ofs << strLevel << '\t' << getCurrentTime() << '\t' << logMsg << '\n'
   ;
  ofs.close();

  va_end(args);
}

void clearLog()
{
  std::ofstream ofs(FILEPATH, std::ios_base::out | std::ios_base::trunc
   );
  ofs.close();
  log(LOG_INFO, "Cleared out log file");
}

} // end namespace Logger
```

8.2.13 main.ino

```
#include <Servo.h>

Servo servo[2];

#define HALL_1 A7 // has led
#define HALL_2 A6 // does not have led

#define SERVO_1 11
#define SERVO_2 12

#define PI_OUT 10

void setup() {
  Serial.begin(9600);
  // The servo control wires are connected to Arduino D11-D12 pins.
  servo[0].attach(SERVO_1);
  servo[1].attach(SERVO_2);

  // Servos are stationary.
  servo[0].write(90);
  servo[1].write(90);

  // Enabling the digital pins on the hall sensor
  pinMode(HALL_1, INPUT);
  pinMode(HALL_2, INPUT);

  digitalWrite(PI_OUT, LOW);
  pinMode(PI_OUT, OUTPUT);
}
```

```

/** N.B. the values read by the hall effect sensors need to be adjusted based on the specific sensors used
 * unless you use the exact ones we did, these values may not work and cause the program to never run/end
 * also good idea to adjust the servo speed based on your specific motor/setup
 */

void loop() {
  digitalWrite(PI_OUT, HIGH);
  servo[0].write(83);
  servo[1].write(97);

  int hall_2_read = analogRead(HALL_2);
  while(250 < hall_2_read && hall_2_read < 900) hall_2_read = analogRead(HALL_2);

  servo[0].write(90);
  servo[1].write(90);

  delay(1000);

  servo[0].write(97);
  servo[1].write(83);

  while(analogRead(HALL_1) > 100);

  servo[0].write(90);
  servo[1].write(90);

  digitalWrite(PI_OUT, LOW);

  delay(300000);
}
```

8.2.14 mjpeg_recv.py

```python
from cv2 import VideoCapture, imshow, destroyAllWindows, waitKey

import cv2 # temporary, better to just import individual classes/functions

class Video(cv2.VideoCapture):
  def __enter__(self):
    return self
  def __exit__(self, *args):
    self.release()

def main():
  try:
    with Video("http://192.168.2.2/stream") as cap:
      if not cap.isOpened():
        raise Exception('Unable to open stream')

      while True:
        _, frame = cap.read()

        # below is a code example to find squares from https://www.tutorialspoint.com/how-to-detect-a-rectangle-and-square-in-an-image-using-opencv-python

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        ret, thresh = cv2.threshold(gray, 50, 255, 0)
        contours, hierarchy = cv2.findContours(thresh, 1, 2)

        print("Number of contours detected:", len(contours))

        for cnt in contours:
          x1, y1 = cnt[0][0]
          approx = cv2.approxPolyDP(cnt, 0.01 * cv2.arcLength(cnt, True), True)

          if len(approx) == 4:
            x, y, w, h = cv2.boundingRect(cnt)
            ratio = float(w)/h

            if 0.9 <= ratio <= 1.1:
              frame = cv2.drawContours(frame, [cnt], -1, (0, 255, 255), 3)
              cv2.putText(frame, 'Square', (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 0), 2)

            else:
              cv2.putText(frame, 'Rectangle', (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)
              frame = cv2.drawContours(frame, [cnt], -1, (0,255,0), 3)

        imshow('frame', frame)
        if waitKey(10) == 27:
          break

  except Exception as e:
    print(f'Exception: {str(e)}')
  finally:
    destroyAllWindows()
    print('Done')

if __name__ == "__main__":
  main()
```

8.2.15 Makefile (aneris program)

```
CPP := g++
CFLAGS := -O3 -Wpedantic -Wall -Wextra -Wconversion -Wshadow -Werror
#PYTHON_LIB := -I/usr/include/python3.9 -lpython3.9
PYTHON_LIB := $(shell python3-config --includes) -lpython3.9
THREAD_LIB := -lpthread

#PYBIND_LIB := $(shell python3 -m pybind11 --includes)
#CPYTHON_EXTENSION := $(shell python3-config --extension-suffix)

TARGET := aneris
INSTALL_DIR := /etc/$(TARGET)/

default: $(TARGET) $(TARGET)-ctl
```

```

$(TARGET): $(TARGET).o GPIO.o Logger.o Web_Server.o
	$(CPP) -o $@ $? $(CFLAGS) $(THREAD_LIB) $(PYTHON_LIB)

$(TARGET).o: $(TARGET).cpp
	$(CPP) -c $< $(CFLAGS) $(THREAD_LIB)

GPIO.o: gpio/GPIO.cpp
	$(CPP) -c $< $(CFLAGS)

Logger.o: Logger/Logger.cpp
	$(CPP) -c $< $(CFLAGS)

Web_Server.o: Web_Server/Web_Server.cpp
	$(CPP) -c $< $(CFLAGS) $(PYTHON_LIB)

Cpython: bit_converter/bit_converter.cpp
	$(CPP) -shared -O3 -Wall -fPIC $(PYBIND_LIB) $< -o bit_converter$(
	  CPYTHON_EXTENSION)

$(TARGET)-ctl: $(TARGET)-ctl.cpp
	$(CPP) -o $@ $< $(CFLAGS)

install:
	mkdir -p $(INSTALL_DIR)
	mkdir -p /var/log/aneris/
	cp $(TARGET) web_server.py $(INSTALL_DIR)
	cp -r web/ $(INSTALL_DIR)
	cp $(TARGET).service /etc/systemd/system/
	cp $(TARGET)-ctl /usr/bin/
	systemctl daemon-reload
	systemctl enable aneris
	systemctl start aneris

clean:
	$(RM) $(TARGET) $(TARGET)-ctl *.o

remake: clean $(TARGET)
```

8.2.16 Makefile (bit_converter)

```
CPP := g++
CFLAGS := -shared -O3 -Wall -fPIC
PYBIND := $(shell python3 -m pybind11 --includes)
TARGET := bit_converter
EXTENSION := $(shell python3-config --extension-suffix)

default: $(TARGET)

$(TARGET):
	$(CPP) $(CFLAGS) -c $(TARGET).cpp

cpython:
	$(CPP) $(CFLAGS) $(PYBIND) $(TARGET).cpp -o $(TARGET)$(EXTENSION) -
	  DCPYTHON

clean:
	$(RM) $(TARGET)$(EXTENSION)
```

```

remake: clean $(TARGET)
```

8.2.17 Web_Server.hpp

```
#ifndef WEB_SERVER_HPP
#define WEB_SERVER_HPP

#include <exception>

namespace Web_Server
{

void *serve(void *port);

class WS_Err : std::exception
{
public:
    WS_Err(const char* message)
    {
        this->msg = message;
    }
    virtual const char* what()
    {
        return this->msg;
    }
private:
    const char* msg;
};

} // end namespace Web_Server

#endif
```

8.2.18 Web_Server.cpp

```
#include <Python.h>
#include "Web_Server.hpp"
#include "../Logger/Logger.hpp"

#define PY_SSIZE_T_CLEAN
#define PY_MODULE_NAME "web_server"
#define PY_FUNC_NAME "serve"

namespace Web_Server
{

static void cleanup_handler(void *ret);

PyObject *pName, *pModule, *pFunc, *pArgs, *pValue;

void *serve(void *port)
{
  long ret = 0;

  pthread_cleanup_push(&cleanup_handler, &ret);

  Py_Initialize();
  pName = PyUnicode_DecodeFSDefault(PY_MODULE_NAME);
  pModule = PyImport_Import(pName);
  Py_DECREF(pName);

  try
  {
    if(pModule == NULL) throw WS_Err("Couldn't import python module");

    pFunc = PyObject_GetAttrString(pModule, PY_FUNC_NAME);

    if (!pFunc || !PyCallable_Check(pFunc)) throw WS_Err("Couldn't find
    function");
    {
      pArgs = PyTuple_New(1);
      PyTuple_SetItem(pArgs, 0, PyLong_FromLong(*(int*)port));

      pValue = PyObject_CallObject(pFunc, pArgs);
      if (pValue == NULL) throw WS_Err("Call to function failed");
      ret = PyLong_AsLong(pValue);
    }
  }
  catch(WS_Err& e)
  {
    if(PyErr_Occurred()) PyErr_Print();
    Logger::log(Logger::LOG_ERROR, e.what());
    ret = -1;
  }

  pthread_cleanup_pop(1);

  pthread_exit(&ret);
}

static void cleanup_handler(void *ret)
{
  switch(*(int*)ret)
  {
    case 0:
    {
      Logger::log(Logger::LOG_INFO, "Web server process ended correctly
  ");
      break;
    }
    case 1:
    {
      Logger::log(Logger::LOG_ERROR, "Port parameter is invalid");
      break;
    }
    case 2:
    {
      Logger::log(Logger::LOG_ERROR, "Exception raised in web server");
      break;
    }
    default:
    {
      Logger::log(Logger::LOG_WARN, "Unknown value returned by process"
  );
    }
  }
  Py_DECREF(pArgs);
  Py_DECREF(pValue);
  Py_XDECREF(pFunc);
  Py_DECREF(pModule);
  Py_FinalizeEx();

  return;
}

} // end namespace Web_Server
```

8.2.19 web_server.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 22 10:11:51 2023

@author: salve
"""

try:
    # camera imports
    from io import BytesIO
    from threading import Condition
    from picamera import PiCamera

    # web server imports
    from logging import warning
    from socketserver import ThreadingMixIn
    from http.server import BaseHTTPRequestHandler, HTTPServer
    from urllib.parse import urlsplit
    from socket import socket, AF_UNIX, SOCK_DGRAM
    #from datetime import datetime
except ImportError as e:
    print('Couldn\'t import all required modules')
    print(f'Additional information: {str(e)}')

__version__ = '1.0.0'

HOME_DIR =  '/etc/aneris/'
WEB_DIR = HOME_DIR + 'web/'
DOCS_DIR =  WEB_DIR + 'docs/'
LOG_FILE =  '/var/log/aneris/aneris.log'
SOCKET =  '/var/run/aneris.sock'

class VideoStreamBuffer:
    def __init__(self):
        self.frame = None
        self.buffer = BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)

    def write2(self, buf): #h.264 encoding
        if buf.startswith(b'\x00\x00\x00\x01'):
            with self.condition:
                self.buffer.seek(0)
                self.buffer.write(buf)
                self.buffer.truncate()
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()

class StreamingHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        path, query = urlsplit(self.path).path, urlsplit(self.path).
query
        try:
            get_params = dict(query.split('=') for query in query.split
('&'))
        except ValueError:
            get_params = dict()

        if path == '/' or path == '/index.html' or path == '/index':
            try:
                with open(WEB_DIR + 'index.html', 'rb') as index:
                    self.send_response(200)
                    self.send_header('Content-Type', 'text/html')
                    self.end_headers()
                    self.wfile.write(index.read())
            except FileNotFoundError:
                self.send_error(500,
                                'Error: index file does not exist',
                                f'Check that the index file exists in
directory \'{WEB_DIR}\'')

        elif path == '/stream':
            self.serve_video()

        elif path == '/logs':
            try:
                with open(LOG_FILE, 'rb') as log:
                    self.send_response(200)
                    self.send_header('Content-Type', 'text/plain')
                    self.end_headers()
                    self.wfile.write(log.read())
            except FileNotFoundError:
                self.send_error(500,
                                'Error: log file does not exist',
                                f'Check that the log file exists in
directory \'{LOG_FILE}\'')

        elif path == '/console':
            self.serve_console();

        elif path == '/docs' or path == '/docs/index.html':
            self.serve_docs(path, query)

        else:
            self.send_error(404)
            self.end_headers()

    def do_POST(self):
        if self.path == '/api':
            try:
                content_length = int(self.headers['Content-Length'])
                post = dict(x.split(b'=') for x in self.rfile.read(content_length).split(b';'))
            except:
                self.send_response(400)
                self.send_header('Content-type', 'application/json')
                self.end_headers()
                response = '{\
                                "error": {\
                                "code": "InvalidPOST",\
                                "message": "The POST request sent to the server did not include a \'command\' parameter"\
                                }\
                            }'
                self.wfile.write(response.encode())
                return

            ### remove after, just for debug
#            self.wfile.write('This is POST request. '.encode())
#            self.wfile.write('Received: '.encode())
#            for x in post:
#                self.wfile.write(f'{x}: {post[x]}'.encode())
            ###

            if b'command' in post:
                try:
                    with socket(AF_UNIX, SOCK_DGRAM) as sock:
                        sock.connect(SOCKET)
                        sock.sendall(post[b'command'])
                    self.send_response(200)
                    self.send_header('Content-type', 'application/json')
                    self.end_headers()
                    response = '{\
                                    "result": {\
                                    "code": "Success",\
                                    "message": "The operation was successful"\
                                    }\
                                }'
                    self.wfile.write(response.encode())
                except:
                    self.send_response(500)
                    self.send_header('Content-type', 'application/json')
                    self.end_headers()
                    response = '{\
                                    "error": {\
                                    "code": "SocketError",\
                                    "message": "The server couldn\'t connect to the socket"\
                                    }\
                                }'
                    self.wfile.write(response.encode())

            else:
                self.send_response(400)
                self.send_header('Content-type', 'application/json')
                self.end_headers()
                response = '{\
                                "error": {\
                                "code": "InvalidPOST",\
                                "message": "The POST request sent to the server did not include a \'command\' parameter"\
                                }\
                            }'
                self.wfile.write(response.encode())



        else:
            self.send_error(404)
            self.end_headers()

    # Get params for stream:
    # content: video, audio, videoaudio, default: videoaudio
    # timestamp: true, false (ignored when content: audio), default: true
    # N.B. timestamp has been temporarily disabled

    def serve_video(self):
        global videoBuffer
        self.send_response(200)
        self.send_header('Age', 0)
        self.send_header('Cache-Control', 'no-cache, private')
        self.send_header('Pragma', 'no-cache')
        self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
        self.end_headers()
        try:
            while True:
                # lets us annotate time on top of the frame, but
                # latest choice is forced on all users
                #if get_params['timestamp'] == 'true':
                    #camera.annotate_text = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                #else:
                    #camera.annotate_text = ""
                with videoBuffer.condition:
                    videoBuffer.condition.wait()
                    frame = videoBuffer.frame
                self.wfile.write(b'--FRAME\r\n')
                self.send_header('Content-Type', 'image/jpeg')
                self.send_header('Content-Length', len(frame))
                self.end_headers()
                self.wfile.write(frame)
                self.wfile.write(b'\r\n')
        except Exception as e:
            warning('Removed streaming client %s: %s', self.client_address, str(e)) # probaably unneeded
```

```python

    def serve_video_2(self):
        global videoBuffer
        self.send_response(200)
        self.send_header('Age', 0)
        self.send_header('Cache-Control', 'no-cache, private')
        self.send_header('Pragma', 'no-cache')
        self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
        self.end_headers()
        try:
            while True:
                with videoBuffer.condition:
                    videoBuffer.condition.wait()
                    frame = videoBuffer.frame
                self.wfile.write(frame)
        except Exception as e:
            warning('Removed streaming client %s: %s', self.client_address, str(e)) # probably unneeded

    def serve_docs(self, docs, query):
        try:
            with open(DOCS_DIR + 'index.html', 'rb') as index:
                self.send_response(200)
                self.send_header('Content-Type', 'text/html')
                self.end_headers()
                self.wfile.write(index.read())
        except FileNotFoundError:
            self.send_response(404)
            self.send_header('Content-type', 'text/plain')
            self.end_headers()
            self.wfile.write('Error: index.html does not exist'.encode())

    def serve_console(self):

        try:
            with open(WEB_DIR + 'console.html', 'rb') as index:
                self.send_response(200)
                self.send_header('Content-Type', 'text/html')
                self.end_headers()
                self.wfile.write(index.read())
        except FileNotFoundError:
            self.send_response(404)
            self.send_header('Content-type', 'text/plain')
            self.end_headers()
            self.wfile.write('Error: console.html does not exist'.encode())

class StreamingServer(ThreadingMixIn, HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

videoBuffer = VideoStreamBuffer()

def serve(port = 0):
    if port <= 0 and not port.isdigit():
        return 1

    global videoOutput
    with PiCamera(resolution='1280x720', framerate=30) as camera:
        camera.start_recording(videoBuffer, format='mjpeg') # this type of format uses lossy compression so it might or might not be suited to opencv image analysis
        #camera.start_recording(videoBuffer, format='h264', profile='baseline') # h.264 option
        try:
            return_int = 0
            address = ('', int(port)) # ip, port
            server = StreamingServer(address, StreamingHandler)
            print('Stream started successfully')
            server.serve_forever()
        except Exception as e:
            print('Couldn\'t start stream')
            print(f'Additional information: {str(e)}')
            return_int = 2
        finally:
            camera.stop_recording()
            return return_int

if __name__ == "__main__":
    serve(5000)
```

Bibliography

3DNet (May 2023). No. URL: https://3dnet.no/search?type=product&q=ABS* (visited on 05/24/2023).

AdaFruit (2023a). *All About Stepper Motors*. en-US. URL: https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor (visited on 05/04/2023).

– (2023b). *Continuous Rotation Micro Servo*. en-US. URL: https://www.adafruit.com/product/2442 (visited on 05/12/2023).

Amazon (May 2023). En. URL: https://www.amazon.com/Stayhome-Converter-48VDC-24VDC-Voltage/dp/B07SSXTWMS (visited on 05/25/2023).

AMLOceanographic (2023). *UV Biofouling Control*. URL: https://amloceanographic.com/uv-biofouling-control (visited on 05/10/2023).

Andrady, Anthony L. (2015). “Persistence of Plastic Litter in the Oceans”. In: *Marine Anthropogenic Litter*. Ed. by Melanie Bergmann, Lars Gutow, and Michael Klages. Cham: Springer International Publishing, pp. 57–72. ISBN: 978-3-319-16510-3. DOI: 10.1007/978-3-319-16510-3_3. URL: https://doi.org/10.1007/978-3-319-16510-3_3.

ANERIS (Mar. 29, 2023). *aneris.eu*. https://www.aneris.eu/about.

Arduino (2023). *Arduino Nano*. en. URL: https://store.arduino.cc/products/arduino-nano (visited on 05/12/2023).

ArduinoModules (May 2016). *KY-003 Hall Magnetic Sensor Module*. en-US. URL: https://arduinomodules.info/ky-003-hall-magnetic-sensor-module/ (visited on 05/25/2023).

Autodesk (2023). *Autodesk Inventor Software — Get Prices & Buy Official Inventor 2024*. en-US. URL: https://www.autodesk.com/products/inventor/overview (visited on 05/04/2023).

BlueRobotics (May 2023a). En. URL: https://bluerobotics.com/store/comm-control-power/tether-interface/fathom-x-tether-interface-board-set-copy/ (visited on 05/25/2023).

– (May 2023b). En. URL: https://bluerobotics.com/store/rov/bluerov2-accessories/fxti-asm-r1-rp/ (visited on 05/25/2023).

– (May 2023c). En. URL: https://bluerobotics.com/store/cables-connectors/cables/fathom-rov-tether-rov-ready/ (visited on 05/25/2023).

– (May 2023d). En. URL: https://bluerobotics.com/store/thrusters/lights/lumen-r2-rp/ (visited on 05/25/2023).

Blueye (2023). *Blueye Pioneer — Underwater drone with camera*. en. URL: https://www.blueyerobotics.com/products/pioneer (visited on 05/11/2023).

Bolanakis, Dimosthenis E., Konstantinos T. Kotsis, and Theodore Laopoulos (2009). “Arithmetic operations in assembly language: Educators’ perspective on endianness learning using 8-bit microcontrollers”. In: *2009 IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pp. 600–604. DOI: 10.1109/IDAACS.2009.5342909.

Boyd, Claude E. (2020). *Typical chemical characteristics of full-strength seawater*. URL: https://www.globalseafood.org/advocate/typical-chemical-characteristics-of-full-strength-seawater/#:~:text=The%20density%20%28specific%

20gravity%29%20of%20seawater%20ranges%20from,%C2%B5S%2Fcm%29%20with%20an%20average%20of%20about%2050%2C000%20%C2%B5mhos%2Fcm. (visited on 02/10/2023).

Chenquan Tech, Hangzhou (2020). *Rotating HD Video Underwater Camera.* URL: https://www.hzcqtech.com/260-Degree-Rotating-Platform-Maritime-Hd-Video-Underwater-Camera-pd43636433.html.

Dursun, Tolga and Costas Soutis (2014). "Recent developments in advanced aircraft aluminium alloys". In: *Materials & Design (1980-2015)* 56, pp. 862–871. ISSN: 0261-3069. DOI: https://doi.org/10.1016/j.matdes.2013.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0261306913011357.

Ecker, Josef Valentin et al. (Jan. 2019). "Mechanical properties and water absorption behaviour of PLA and PLA/wood composites prepared by 3D printing and injection moulding". In: *Rapid Prototyping Journal* 25.4. Publisher: Emerald Publishing Limited, pp. 672–678. ISSN: 1355-2546. DOI: 10.1108/RPJ-06-2018-0149. URL: https://doi.org/10.1108/RPJ-06-2018-0149 (visited on 05/11/2023).

Emco (2023a). *POM Acetal copolymer.* URL: https://www.emcoplastics.com/unfilled-copolymer/.

– (2023b). *What is the difference between acetal plastic and delrin?* URL: https://www.emcoplastics.com/acetal-vs-delrin/.

Enclosures, Buoyancy, and Ballast Archives (2023). en-US. URL: https://bluerobotics.com/product-category/watertight-enclosures/ (visited on 05/25/2023).

hasing (2023). en. URL: https://www.chasing.com/en/chasing-dory.html (visited on 05/26/2023).

Hyakudome, Tadahiro (2011). "Design of autonomous underwater vehicle". In: *International Journal of Advanced Robotic Systems* 8.1, p. 9.

ikea.com (2023). *IKEA Spare Parts.* URL: https://www.ikea.com/no/no/customer-service/returns-claims/spareparts/.

ISM (Feb. 2023). *What is Acetal Plastic? — POM Plastics for Flow Control Parts.* URL: https://www.industrialspec.com/about-us/blog/detail/what-is-acetal-plastic-pom-polyacetal-polyoxymethylene.

J. Piippo T. Laitinen, P. Sirkiae (Feb. 1997). "Corrosion behaviour of zinc and aluminium in simulated nuclear accident environments". In: URL: https://www.osti.gov/etdeweb/servlets/purl/480952.

Kjell (May 2023). En. URL: https://www.kjell.com/no/produkter/elektro-og-verktoy/arduino/stromforsyning/luxorparts-variabel-spenningsregulator-switchet-p87049 (visited on 05/25/2023).

TP-Link (May 2023). En. URL: https://www.tp-link.com/us/home-networking/5-port-switch/tl-sf1005d/v15/ (visited on 05/25/2023).

Liu, Louis (Feb. 10, 2023). Personal Communication. General Manager.

Losek, Jeremy (Apr. 2022). *Injection Molding Price.* en. URL: https://icomold.com/much-injection-molding-cost/ (visited on 05/24/2023).

Ltd, ProSciTech Pty (July 2019). *Plastic Properties Table.* URL: https://laboratoryresource.com.au/?navaction=getitem&id=39.

Lucas, Jacques et al. (2016). *Rare earths: Science, technology, production and use.* Elsevier.

MarineshopAS (2023). *Zinc anode 0,5 kg - Marineshop AS.* URL: https://www.marineshop.no/universal/114471/sinkanode-0-5-kg (visited on 05/25/2023).

ModernPlastics (Jan. 2016). *Differences Between Acetal Products.* URL: https://modernplastics.com/blog/differences-between-acetal-products/.

Molykote (2023). *Molykote.* URL: https://www.dupont.com/molykote.html.

Mozilla (2023a). *Document: getElementById() method.* URL: https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById.

– (2023b). *Using the Fetch API.* URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch.

Novy, Dan et al. (2022). "Maka Niu: A low-cost, modular imaging and sensor platform to increase observation capabilities of the deep ocean". In.

Omnexus (Mar. 2023). *Comprehensive Guide on Acrylonitrile Butadiene Styrene (ABS).* URL: https://omnexus.specialchem.com/selection-guide/acrylonitrile-butadiene-styrene-abs-plastic.

OpenBroadcasterSoftware (2023). *Open Broadcaster Software — OBS.* URL: https://obsproject.com/ (visited on 05/23/2023).

Pelletier, Dominique et al. (2021). "KOSMOS: An Open Source Underwater Video Lander for Monitoring Coastal Fishes and Habitats". In: *Sensors* 21.22. ISSN: 1424-8220. DOI: 10.3390/s21227724. URL: https://www.mdpi.com/1424-8220/21/22/7724.

plastics, Adreco (2023). *ABS plastic properties.* en. URL: https://adrecoplastics.co.uk/abs-plastic-properties/ (visited on 05/22/2023).

RaspberryPi (May 2023a). En. URL: https://www.raspberrypi.com/products/camera-module-v2/ (visited on 05/25/2023).

– (2023b). *Raspberry Pi.* en-GB. URL: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/ (visited on 05/12/2023).

RatRig (Feb. 2023). *V-slot 2020.* URL: https://ratrig.com/aluminium-profiles/v-slot-2020-standard.html.

ResimacLtd (Mar. 2023). *Product Specification of 208 Ceramic UW.* URL: https://www.resimacsolutions.com/uploads/4/5/0/7/45072315/208_uw_ps.rv6.21062022.pdf.

RTS (2023). *PLA vs Plastic: What's the difference? — RTS.* URL: https://www.rts.com/blog/pla-vs-plastic-whats-the-difference/ (visited on 05/04/2023).

Shaw, Barbara and Robert Kelly (Apr. 2006). "What is Corrosion?" In: *The Electrochemical Society Interface* 15.1, p. 24. DOI: 10.1149/2.F06061IF. URL: https://dx.doi.org/10.1149/2.F06061IF.

Smith, Paul (Feb. 21, 2023). Personal Communication. Sales Representative.

StoreNorskeLeksikon (Dec. 2017). *Delrin.* URL: https://snl.no/Delrin.

SubC-Imaging (Jan. 2023). *SUBC camera comparison guide.* URL: https://www.subcimaging.com/case-studies/subsea-camera-comparison-guide/#comparison-chart.

support.google.com (n.d.). *YouTube live encoder settings, bitrates, and resolutions - youtube help.* URL: https://support.google.com/youtube/answer/2853702?hl=en.

TABER (2023). *6000 Series Aluminum Alloys.* en. URL: https://taberextrusions.com/6000-series-aluminum-alloys/ (visited on 05/24/2023).

Thingiverse.com (2023). *Bevel gear for nema17.* en. URL: https://www.thingiverse.com/thing:1637765 (visited on 05/04/2023).

TurbulentResearch (May 2023). En. URL: https://turbulentresearch.com/porpoise (visited on 05/25/2023).

U-CoatThecnologies (2023). *Resimac 208 Ceramic UW.* URL: https://u-coat.com/product/resimac-208-ceramic-uw/.

UltiMaker (Jan. 2023a). *How to print with Ultimaker ABS.* URL: https://support.makerbot.com/s/article/1667337602935?_gl=1*1qact9t*_ga*MTU1OTgxNTcyOC4xNjc4MjgOMjQ0
_ga_JHX8W909G8*MTY3ODI4NDI0My4xLjEuMTY3ODI4NDQ3Ny4wLjAuMA..*_ga_CJM2DTBWYF*MTY3ODI4NDI1Mi4xLjEuMTY3ODI4NDQ3Ny4wLjAuMA...

– (2023b). *UltiMaker 2+ Connect.* en. URL: https://ultimaker.com/3d-printers/s-series/ultimaker-2-connect/ (visited on 05/04/2023).

UnitedNationsAssociationOfNorway (Feb. 2023a). *12 Ansvarlig forbruk og produksjon.* URL: https://www.fn.no/om-fn/fns-baerekraftsmaal/ansvarlig-forbruk-og-produksjon.

– (Feb. 2023b). *13 Stoppe klimaendringene.* URL: https://www.fn.no/om-fn/fns-baerekraftsmaal/stoppe-klimaendringene.

– (Jan. 2023c). *FNs bærekraftsmål.* URL: https://www.fn.no/om-fn/fns-baerekraftsmaal.

Wahl, M (1989). "Marine epibiosis. I. Fouling and antifouling: some basic aspects". en. In: *Marine Ecology Progress Series* 58, pp. 175–189. ISSN: 0171-8630, 1616-1599. DOI: 10.3354/meps058175. URL: http://www.int-res.com/articles/meps/58/m058p175.pdf (visited on 02/20/2023).

Wahl, Martin (1989). "Marine epibiosis. I. Fouling and antifouling: some basic aspects". In: *Marine ecology progress series*, pp. 175–189.

Wang, Can et al. (Sept. 2021). "Biodegradable microplastics (BMPs): a new cause for concern?" In: *Environmental Science and Pollution Research* 28. DOI: https://doi.org/10.1007/s11356-021-16435-4. URL: https://link.springer.com/article/10.1007/s11356-021-16435-4#citeas.

Weerg (2023). *Aluminium: Properties and Advantages.* en. URL: https://www.weerg.com/guides/aluminum-properties-and-advantages-of-aluminum (visited on 05/19/2023).

Wishner, Karen (2000). *Zooplankton in the Deep Sea.* URL: https://www.gso.uri.edu/maritimes/Back_Issues/00%5C%20Summer/Text(htm)/zooplankton.htm#wfig1.

Xiaodong Liu, Hans Bertilsson (Aug. 1999). *Recycling of ABS and ABS/PC Blends.* URL: https://doi.org/10.1002/(SICI)1097-4628(19991017)74:3%3C510::AID-APP5%3E3.0.CO;2-6.

YouTube (2023). *YouTube.* nb-NO. URL: https://www.youtube.com/ (visited on 05/23/2023).

ZebraTech (2023). *Hydro-Wiper.* en-NZ. URL: https://www.zebra-tech.co.nz/hydro-wiper/ (visited on 05/10/2023).