MATE ROV competition
HydroVinci technical documentation

Marine Advanced Technology Education Remote Operated Vehicule

Sascha CAUCHON, Nathan CHOUKROUN, Mael DARNAUD, Mathurin DE CRECY, Camille FARRA, Sarah OUNES, Matthieu PECORARO, Hugo PELTIER

1 Table of Contents

2 Introduction....4

2 Abstract....4

2 Team presentation....5

3 Budget....5

4 Safety Features....7

4.1.1 Electrical Safety....7

4.1.2 Strain Reliefs....7

4.1.3 Motor Shrouds....7

5 Mechanical Design....8

5 Overview....8

5 Structure....9

5.2.1 ROV....9

5.2.2 Float....9

5 Movement....9

5.3.1 ROV....9

5.3.2 Stabilization....10

5.3.3 Float's buoyancy system....10

5 Manipulation....11

5.4.1 Arm....11

5.4.2 Control....11

5.4.3 Hand....11

5 Manufacturing....9

5.5.1 3D Printing....9

5.5.2 Machining....9

6 Electrical Design....13

6 Overview....13

6.1.1 Architecture....13

6.1.2 Safety....21

6 PCB Design....22

6.2.1 ROV....22

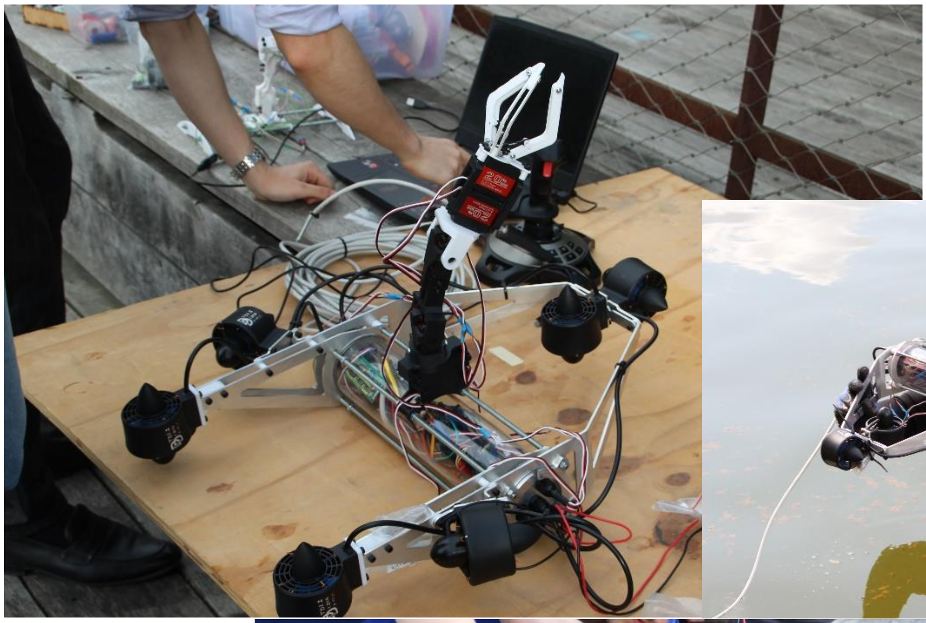
6.2.2 Float....23

6 PCB Integration 24
6.3.1 Mechanical Overview 24
6.3.2 Cable management 25
7 Software 25
7 Rust Programming 25
7.1.1 Efficiency and Memory Safety 25
7.1.2 Asynchronous Programming 26
7.1.3 Extensive Library Support 26
7.1.4 Motor and Servo Control 26
7.1.5 Stabilization System 28
7.1.6 Summary 29
7 General Architecture 29
7.2.1 Onboard Raspberry Pi 4 29
7.2.2 Control Box Laptop 29
7.2.3 Communication and Network Setup 30
7.2.4 Data Flow and Command Transmission 30
7.2.5 Summary 31
8 Conclusion 31

2 INTRODUCTION

2 ABSTRACT

This document proudly showcases HydroVinci's innovative devices for the 2024 MATE ROV Pioneer Class competition, marking the historic debut of the first French team in this prestigious event. HydroVinci, a dynamic technical club from the Engineering School Léonard De Vinci in Paris, specializes in cutting-edge sea technology. This year, the club ambitiously set out to compete in the MATE ROV World Championship. Over the past six months, eight dedicated engineering students, ranging from their second to fourth year, have meticulously designed and developed both an ROV and a Float to meet the rigorous standards of the competition. In this document, we delve into our design choices, the challenges we faced, the intricate technical details of our creations, and our remarkable journey towards bringing this project to life.



2 TEAM PRESENTATION

This project has been both a significant technical challenge and a test of our team's human and management skills. We assembled a team of 8 engineering students to handle the technical development while also managing communication and seeking partnerships for funding.

We divided the project into three categories: mechanics, electronics, and programming, each led by a team leader who excelled in their respective field. Overseeing operations, in charge of project management and mechanics is Camille Farra. His primary responsibilities included coordinating all teams and members, planning project phases, seeking funding, managing costs and budgets, and communicating about our work and activities. Concerning mechanics, its responsibilities included designing the physical structure of the ROV, such as the frame, buoyancy systems, the arm and arm controller, the Float, and waterproof enclosures. The work also focused on developing and installing the propulsion system to ensure efficient underwater navigation. Additionally, designing and implementing tools for underwater tasks. Rigorous testing was crucial to ensure reliable performance of all mechanical components. Key achievements of the team included successfully designing a lightweight yet robust frame, implementing an efficient propulsion system, and developing a mechanical arm capable of various underwater operations.

The AI&IT team, led by Mathurin de Crecy and including Sascha Cauchon, Mael Darnaud, and Sarah Ounes, was responsible for writing and maintaining the control software for the ROV, including the user interface and navigation algorithms. They also developed software for AI-powered tasks and data analysis processes, ensuring seamless integration between software and hardware components. Achievements of the team included mastering a new programming language designed for embedded systems, Rust, implementing advanced navigation algorithms, and developing reliable data processing capabilities.

The electronics team, led by Matthieu Pecoraro and supported by Hugo Peltier and Nathan Choukroun, was responsible for designing and assembling electronic circuits for power distribution, control systems, and sensor interfaces. They integrated various sensors, including cameras and environmental sensors, into the ROV system, and developed a power management system to ensure stable, safe, and efficient operation. Major achievements of the team included designing a reliable power distribution network, integrating high-performance sensors, and developing robust communication protocols.

3 BUDGET

Statistics of our budget for the project, including fabrication, components and travel expenses to the competition, travel not included.

Total actual expenses	1,252.70 €
Project budget:	7,500.00 €
Left budget:	6,247.30 €
Budget used:	16.70%

Repartition of the actual expenses between the ROV and the float.

ROV : 83.57% FLOAT : 16.43%

Details of our expenses to date.

FLOAT						
Déjà acheté / Already bought						€205.84
Eléments	Items	Prix unitaire / Unit price	Quantity	Delivery	Total	
Coupleurs d'arbres	Shaft coupler	7.88€	1	0.00€	€7.88	
Lot 5 joints toriques	Set of 5 O-rings	12.64€	1	0.00€	€12.64	
Tuyeau 80 mm	80mm hose	8.85€	1	0.00€	€8.85	
Lot écrous 6 mm	Set of 6 mm nuts	3.50€	1	0.00€	€3.50	
Tiges filetés L 1m D	Threaded rods L 1m D	0.90€	2	0.00€	€1.80	
Pack 2 batteries 9V	Pack 2 batteries 9V	11.50€	1	0.00€	€11.50	
Cylindre Acrylique 1 m	Acrylic Cylinder 1 m	31.52€	1	14.40€	€45.92	
Accu Rechargeable 9V	Accu Rechargeable 9V	14.90€	2	0.00€	€29.80	
Filament PETG	PETG filament	23.58€	1	0.00€	€23.58	
Lot 5 Joint Torique 80	Lot 5 O-ring 80	12.64€	1	0.00€	€12.64	
Lot 5 Joint Torique 80	Lot 5 O-ring 80	5.49€	1	0.00€	€5.49	
Capteur de pression	Pressure sensor	22.00€	1	0.00€	€22.00	
Batteries 9V Energizer	Batteries 9V Energizer	4.99€	2	0.00€	€9.98	
Batteries 9V G20	Batteries 9V G20	2.29€	2	0.00€	€4.58	
Cartes à trous	Hole cards	4.83€	1	0.00€	€4.83	
Gaines thermo	Gaines thermo	1.62€	1	0.00€	€1.62	
Clé multi-diamètres	Multi-diameter wrench	2.77€	1	0.00€	€2.77	
Capteur de pression	Pressure sensor	10.86€	1	0.00€	€10.86	

ROV						
Déjà acheté / Already bought						€1,046.86
Eléments	Items	Prix unitaire / Unit price	Quantity	Delivery	Total	
Moteurs	Motors	54.79€	9	28.75€	€521.86	
Servomoteur	Servo motor	11.26€	1	0.00€	€11.26	
Gaine cable	Cable sheath	7.99€	1	0.00€	€7.99	
Jumper wires	Jumper wires	11.59€	1	0.00€	€11.59	
Lot 10 potentiomètres	Lot 10 potentiometers	6.99€	1	0.00€	€6.99	
Presse-étoupes	Cable glands	0.49€	5	8.33€	€10.78	
Jeu d'écrous	Nut set	11.99€	1	0.00€	€11.99	
Rangement	Storage	6.99€	1	0.00€	€6.99	
Pied à coulisses	Caliper	3.53€	1	0.00€	€3.53	
Câbles	Cables	7.87€	1	0.00€	€7.87	
Fusibles + supports	Fuses + holders	13.89€	1	0.00€	€13.89	
Lot de 3 PCA	Lot of 3 pcs	14.39€	1	0.00€	€14.39	
DC-DC converter x2	DC-DC converter x2	1.99€	1	0.00€	€1.99	
LM2596S x2	LM2596S x2	1.99€	1	0.00€	€1.99	
Monture caméra	Camera mount	1.13€	1	0.00€	€1.13	
Connecteurs	Connectors	2.73€	1	0.00€	€2.73	
Lot de sockets	Lot de sockets	2.52€	1	0.00€	€2.52	

Fusibles 1A n1	1A n1 fuses	5.99€	1	0.00€	€5.99	
Connecteurs 9V	9V connectors	1.35€	1	0.00€	€1.35	
Fusibles 1A n2	1A n2 fuses	1.61€	1	0.00€	€1.61	
Boitiers piles AA	AA battery boxes	2.84€	1	0.00€	€2.84	
Radiateurs	Radiators	2.26€	1	0.00€	€2.26	
Seringues x2	Seringues x2	2.28€	1	0.00€	€2.28	
Tube PMMA 80	Tube PMMA 80	135.74€	1	20.00€	€155.74	
Anderson PCB	Anderson PCB connectors	0.64€	20	0.00€	€12.74	
Fuse holders	Fuse holders	4.33€	4	0.00€	€17.32	
5V Fan	5V Fan	7.84€	3	0.00€	€23.52	
USBA-USBC cable	USBA-USBC cable	1.57€	2	0.00€	€3.14	
USBC-USBC cable	USBC-USBC cable	4.55€	2	0.00€	€9.10	
5V Laser	5V Laser	5.57€	3	0.00€	€16.71	
PCBs	PCBs	80.35€	1	0.00€	€80.35	
Composants PCBs	PCB components	72.42€	1	0.00€	€72.42	

4 SAFETY FEATURES

4.1.1 Electrical Safety

Ensuring electrical safety is paramount in the design and operation of our ROV system. For detailed guidelines and measures regarding electrical safety, please refer to section 6.1.2 of this document. This section outlines comprehensive strategies and precautions implemented to mitigate potential risks and hazards associated with electrical components and systems, safeguarding both equipment and personnel throughout the ROV's deployment and operation.

4.1.2 Strain Reliefs

In the design of the underwater robot's connection system, strain relief mechanisms have been incorporated to protect the integrity of the data and power cables. Specifically, springs have been utilized as strain relief components. These springs are affixed at one end to the cables and at the other end to either the robot’s structure or the ground-based generator. This configuration ensures that any tensile forces exerted on the cables are absorbed by the springs, which extend to accommodate the force. Consequently, the cables are shielded from direct stress, thereby reducing the risk of damage due to pulling or stretching. This system enhances the overall durability and reliability of the electrical connections essential for the robot’s operation.

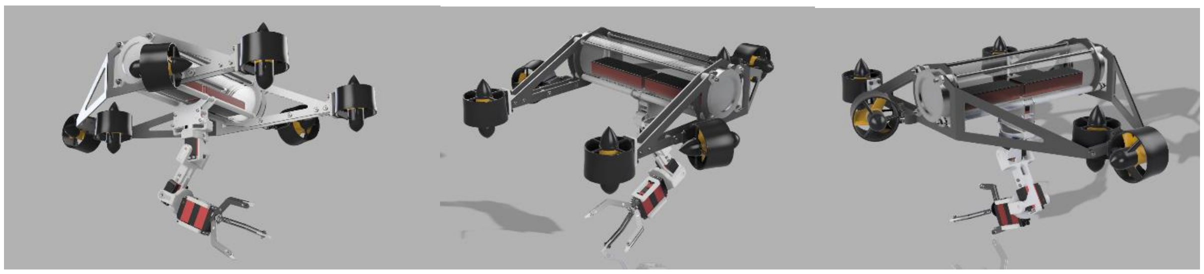
4.1.3 Motor Shrouds

To ensure the safety of personnel and prevent foreign objects from touching the propellers, the six thrusters are equipped with protective grills that were designed and 3D printed using PLA. These grills are installed on both sides of each propeller. The grills serve as a physical barrier, effectively preventing fingers or objects from accidentally entering the propeller area and causing injury or damage. This safety measure is crucial for maintaining a secure operational environment and ensuring the longevity of the propulsion system.

5 Mechanical Design

5 Overview

The ROV is a multi-purpose device aiming to move around water, manipulate objects with precision, analyze and identify objects and sounds, and deploy a probe in a defined zone.

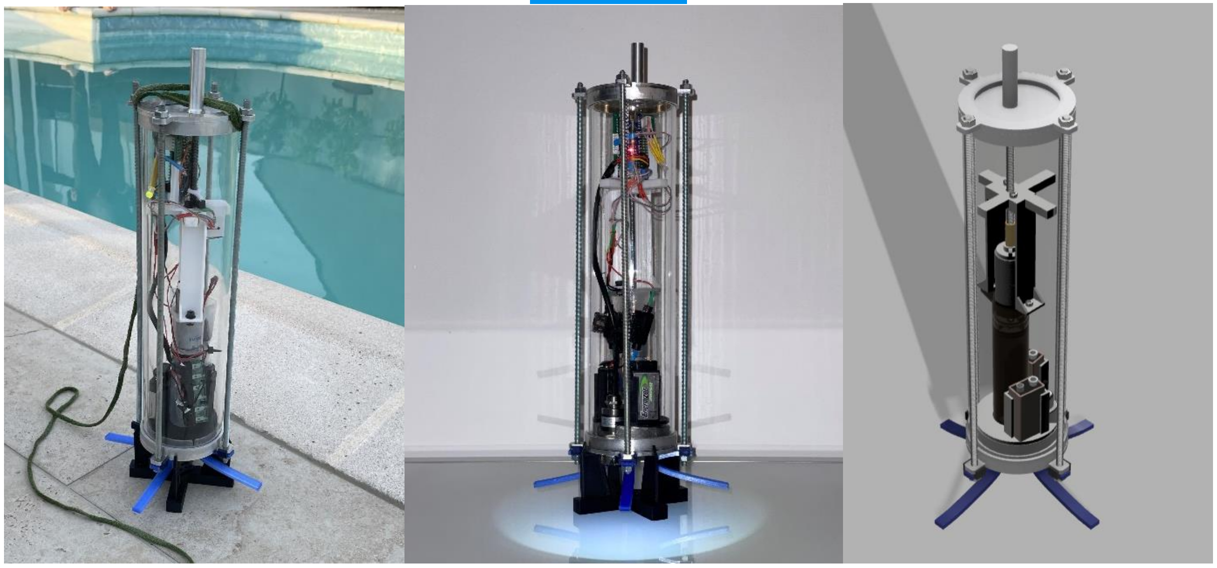


The ROV is designed as a platform for its arm. It needs to be perfectly balanced and stable. It has six thrusters, three with a vertical thrust direction and three with a horizontal thrust direction. This gives the ROV three supporting points on the horizontal plan making it able to stay perfectly stable. The other three thrusters are oriented in a way that makes the ROV capable of translating in all directions as well as rotating around itself. Most of the thrusters are away from the ROV's center giving them a lever force that easily moves the robot around.

All the electronic components are placed in the "heart" of the device that is a PMMA tube. Its orientation gives the camera a clear and large vision field of the exterior. The frame is made from aluminum and 3D printed secondary components to make the assemblage easier.

The arm is placed under the ROV's center of mass and stabilizes the robot acting as a keel. Its position limits variations of the ROV's center of mass when carrying various payloads.

The Float is a tube featuring a buoyancy system and a pressure sensor. It is powered by two 9V alkaline batteries and controlled by an Arduino Nano.



All components are displayed in the device in a way lowering the center of mass as much as possible. The electronic circuits are at the top of the Float to ease communication at the surface. There are four landing skids on the device to avoid chocs at the bottom of the pool.

5 Structure

5.2.1 ROV

The frame of the ROV is built around its core and supports all external components such as the arm and thrusters. This frame, machined from aluminum using a 3-axis CNC, has a fork shape. This design provides three main points of thrust and allows the arm to approach objects closely. At the heart of the ROV is an 80x330 mm PMMA tube housing all electronic components. The tube is sealed with two aluminum covers, held together by four threaded shafts, which also support the arm.

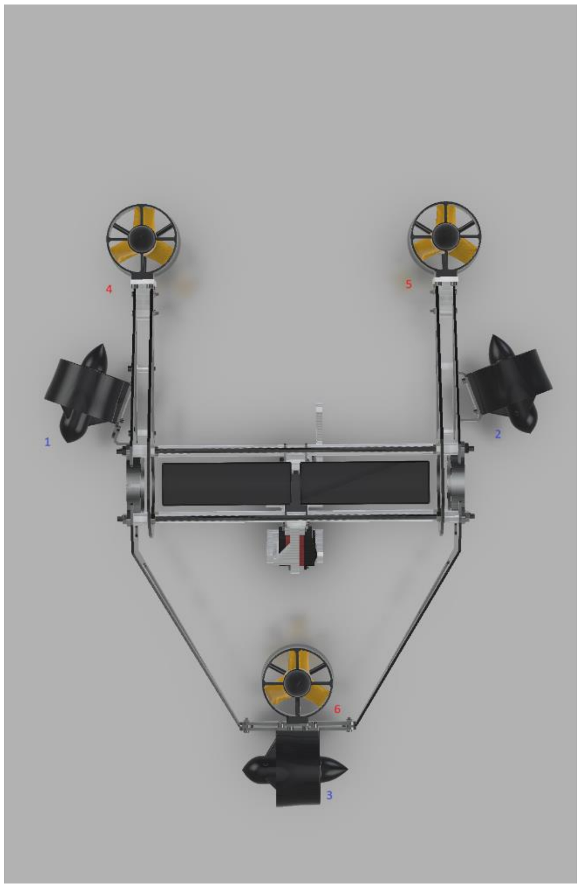
5.2.2 Float

The Float's structure is the same as the ROV's core for practical reasons. It is a 80x305 mm PMMA tube that is sealed by two aluminum covers. One specificity is that the tube was shortened to avoid a too important Archimedes' buoyancy force. In order to operate the buoyancy system that is longer than the tube, a specific section has been added to the top cover that allows just enough space for the buoyancy system to work.

5 Movement

5.3.1 ROV

The ROV moves thanks to 6 thrusters that are carefully placed and oriented around it. Three motors have a vertical thrust direction and allow vertical movement and planar stabilization. Two motors are placed on the sides of the ROV, allowing forward and backward translations. Those two motors are angled at 20° relative to the forward-aft axis of the device. Consequently, the direction perpendicular to their thrust direction passes through the center of mass. Helped by the third horizontal thruster at the back of the device, they allow the ROV to rotate around itself. Finally, with correct thrust distribution among those three same motors, the ROV can translate on the right-left axis.

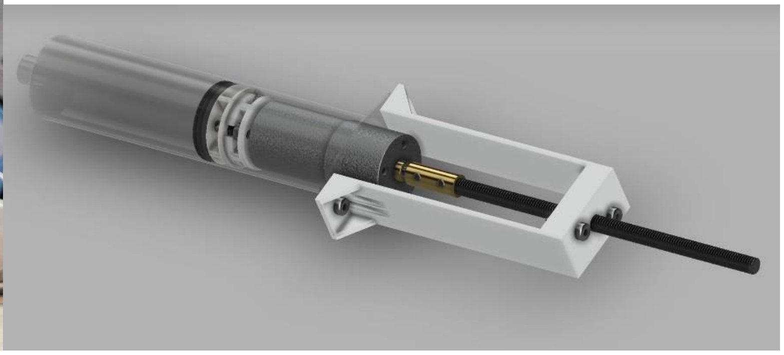


5.3.2 Stabilization

The ROV is meant to stay perfectly horizontally. This is made possible by the three support points that are the vertical thrust engines. Using a gyroscope and a PID controller, those three motors react to any perturbation to keep the ROV horizontal without any human intervention.

5.3.3 Float's buoyancy system

The Float's buoyancy system is a syringe that fills and empties with water, changing the device's density. A high torque electric motor that perfectly fits in the syringe is attached to the syringe's piston. The engine's axle is connected to a threaded shaft and rotates in a nut moving all the system with the piston upward or downward, allowing the syringe to fill or empty itself.



5 Manipulation

5.4.1 Arm

Our ROV is equipped with a 5-axis robotic arm mounted with a 3-fingers plier. Its purpose is to reach, grab and collect any kind of object. The main reason for creating such an arm, instead of a simpler craw or plier, resided in only moving the plier via a precision arm instead of moving the entire ROV less accurately. To achieve this, we added 5 servos and a controller.

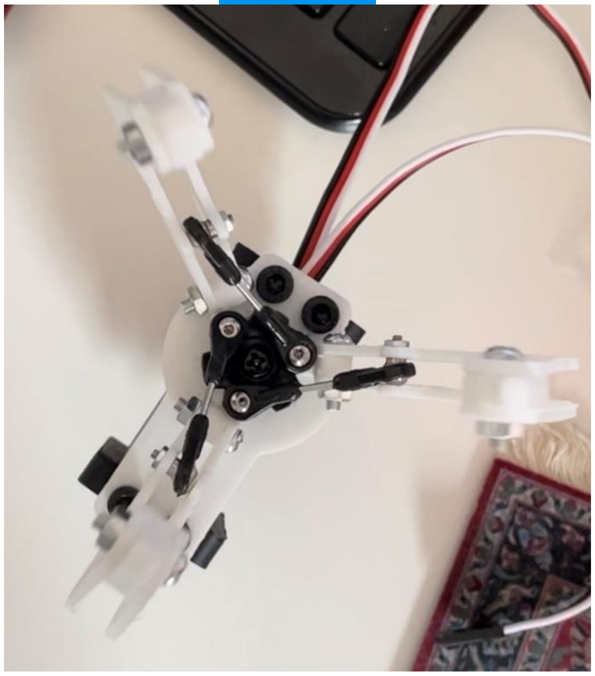
5.4.2 Control

The control of the arm is made through a “physical twin”. This twin is part of the control system and when moved, the movement is perfectly replicated on the real arm of the ROV. It is composed of 5 potentiometers, powered in series and carry the signals to an Arduino Nano. The signals are then transmitted to the ROV through a python script running on the computer.

(see Elec & soft)

5.4.3 Hand

The hand has three fingers allowing it to grab big objects as easily as tiny objects. It is operated by a servomotor and a set of connecting rods.



The hand is made of PLA for prototyping and is machined in aluminum for the final assembly.

6 ELECTRICAL DESIGN

6 OVERVIEW

In this section, we provide an overview of the electrical architecture for both our vehicles: the ROV and the float. All the electronic components are mounted on PCBs, which were entirely designed and developed by our team. Miniaturization was a crucial aspect of our design process, ensuring that all circuits are compact and efficient. This meticulous approach allowed us to optimize the performance and reliability of our systems, meeting the stringent requirements of the competition.

6.1.1 Architecture

Electronic architecture is a critical factor in the design of efficient PCBs. By carefully planning and optimizing the layout and connections of each component, we ensure that our systems perform reliably and effectively. Below, we present the different choices made by our team for the components of the system, illustrating the fundamental design before the implementation on PCBs.

6.1.1 ROV

6.1.1.1.1 Raspberry

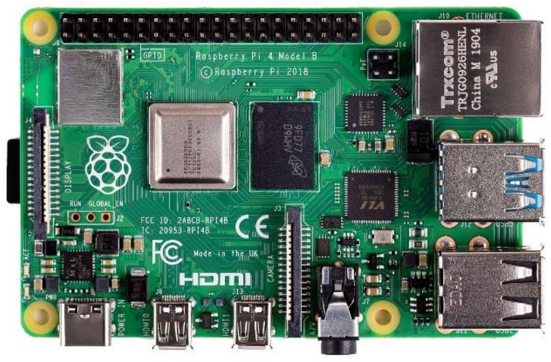


Figure 1 : Raspberry Pi 4 Model B 8Gb.

We chose to use a Raspberry Pi as the central processing unit for our ROV and float due to its versatility, compact size, and robust community support. The Raspberry Pi provides the computational power needed to handle complex tasks such as real-time data processing, sensor integration, and control algorithms. Additionally, it supports a wide range of peripherals and interfaces, making it an ideal choice for our modular design.

An essential aspect of our implementation is that all software running on the Raspberry Pi is coded in Rust. Rust offers memory safety and concurrency advantages, which are critical for the reliability and performance of our system. For detailed information on our software development process and the advantages of using Rust, please refer to section 6.1.1 of this document.

6.1.1.1.2 LT1084.5

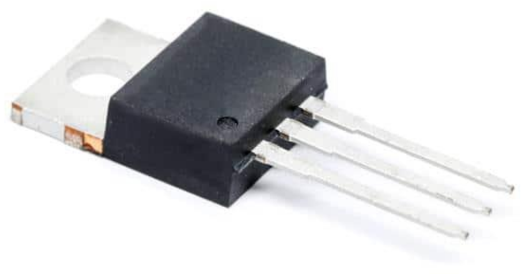


Figure 2 : LT1085CT-5

We selected the LT1084CT-5 as our current regulator for its superior performance in stepping down voltage to a stable 5V output, crucial for the reliable operation of our system. Unlike the L7805, which can only handle a 1 amp current draw, the LT1084CT-5 is capable of delivering up to 6 amps of current, making it an ideal choice for powering multiple components with high current demands.

One LT1084CT-5 regulator is dedicated to powering the Raspberry Pi, the PCA9685, and various sensors, which together require a maximum of 4 amps. This ensures that these critical components receive a consistent and sufficient power supply for optimal performance. Another LT1084CT-5 is used to power the servomotors of the robotic arm, which also require a stable 5V supply to operate effectively.

The use of the LT1084CT-5 regulators ensures that each part of our system receives the necessary power without the risk of overload or instability, contributing to the overall efficiency and reliability of our ROV and float designs.

6.1.1.1.3 MPU 6050

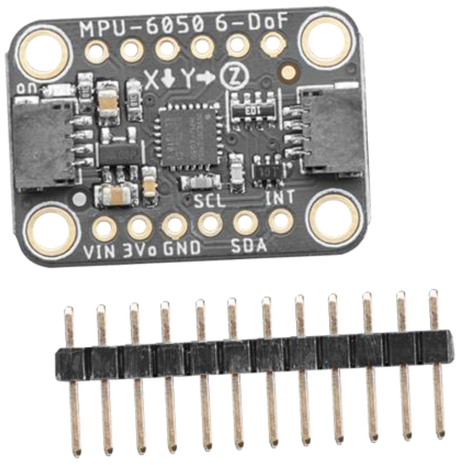


Figure 3 : MPU6050 Module

We chose the MPU6050 module for our ROV due to its integrated 3-axis gyroscope and 3-axis accelerometer, which provide precise motion tracking and orientation data. This module is essential for achieving accurate and responsive stabilization of our ROV in the underwater environment.

The MPU6050's high-performance motion sensors allow us to monitor the ROV's movements in real-time, enabling effective stabilization and control. This is particularly important for maintaining the ROV's balance and orientation, ensuring smooth and precise navigation.

For detailed information on how we read and utilize the data from the MPU6050 for the stabilization of our ROV, please refer to section 6.1.1.5 of this document. This section outlines the specific algorithms and processes we implemented to integrate the MPU6050 data into our control systems, enhancing the overall stability and performance of our ROV.

6.1.1.1.4 PCA 9685

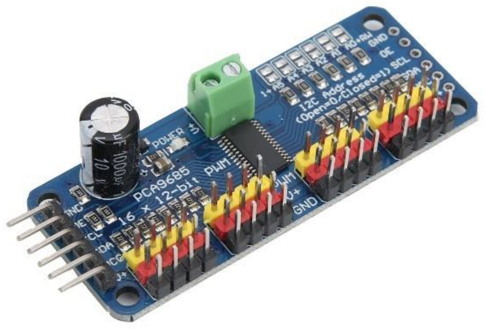


Figure 4 : PCA9685 Board

We opted for the PCA9685 to generate PWM signals for both the servos in the arm and the ESCs of the thrusters due to its versatility and capability to control multiple motors and servos simultaneously. With 6 motors mapped to ports 0-5 and 5 servos mapped to ports 8-12 of the PCA board, we efficiently utilize its channels to cater to the varied needs of our ROV system.

The PCA9685's ability to provide precise PWM signals allows for smooth and accurate control of both servos and motors, essential for tasks requiring precise movement and propulsion underwater.

It's worth noting that the PCA boards may have a frequency difference from the asked frequency on the order of +-5%. To counter this, we implement software calibration techniques to ensure that the PWM signals generated by the PCA9685 meet our exact requirements. This calibration process guarantees consistent and reliable performance across all channels, despite potential frequency variations.

6.1.1.1.5 Circuit Schematic

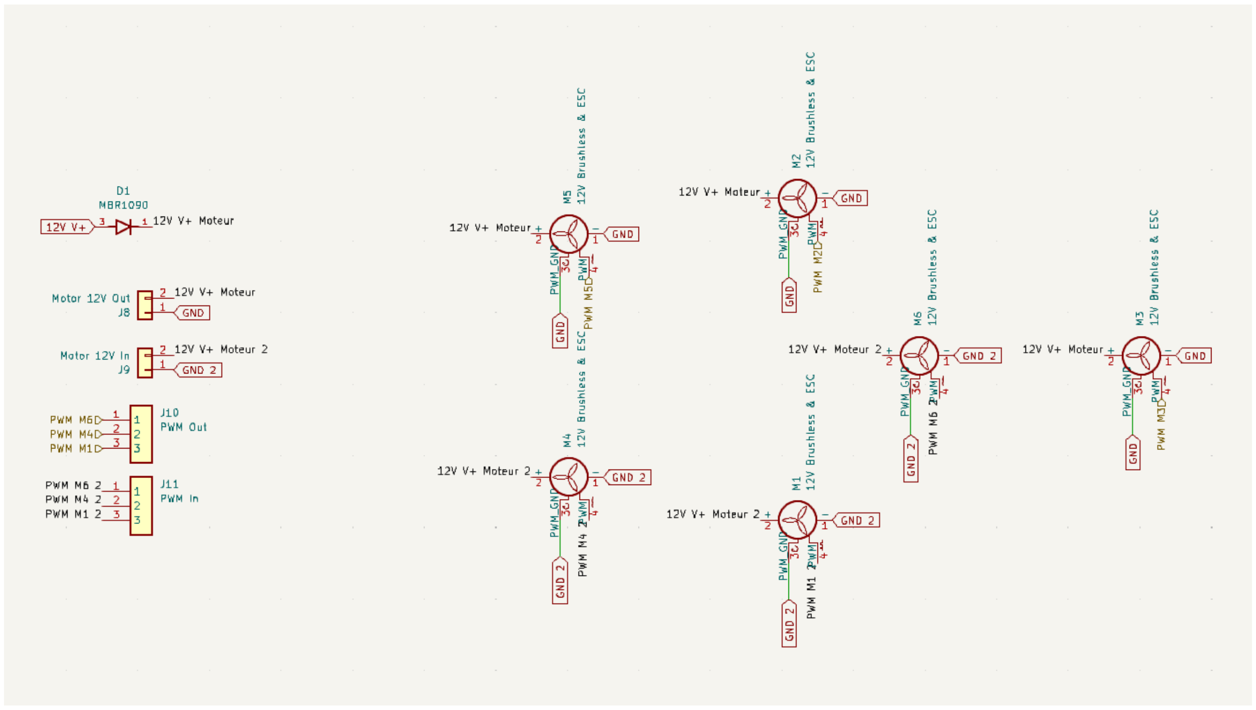


Figure 5: Thruster Connections

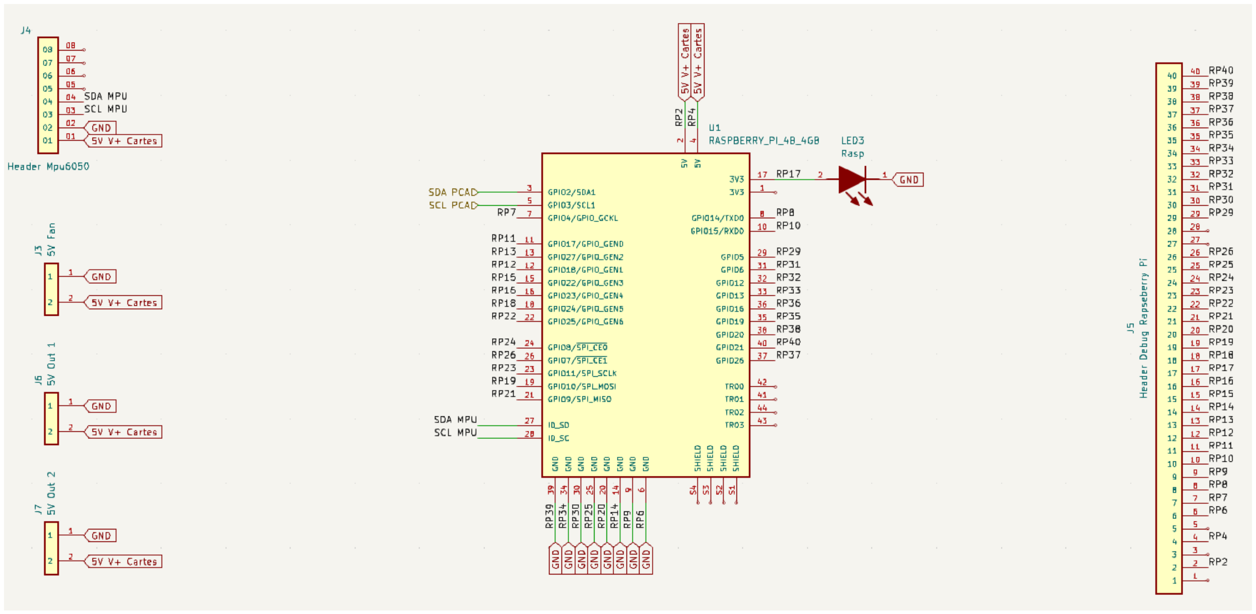


Figure 6 : Raspberry Pi 4 Connections

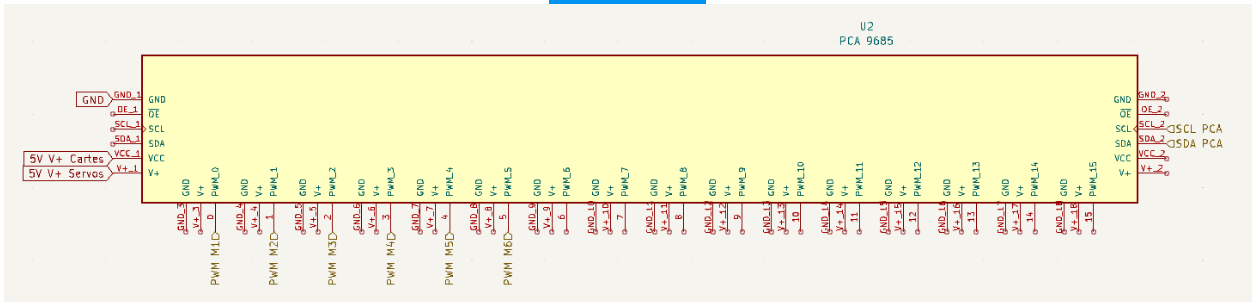


Figure 7 : PCA9685 Connections

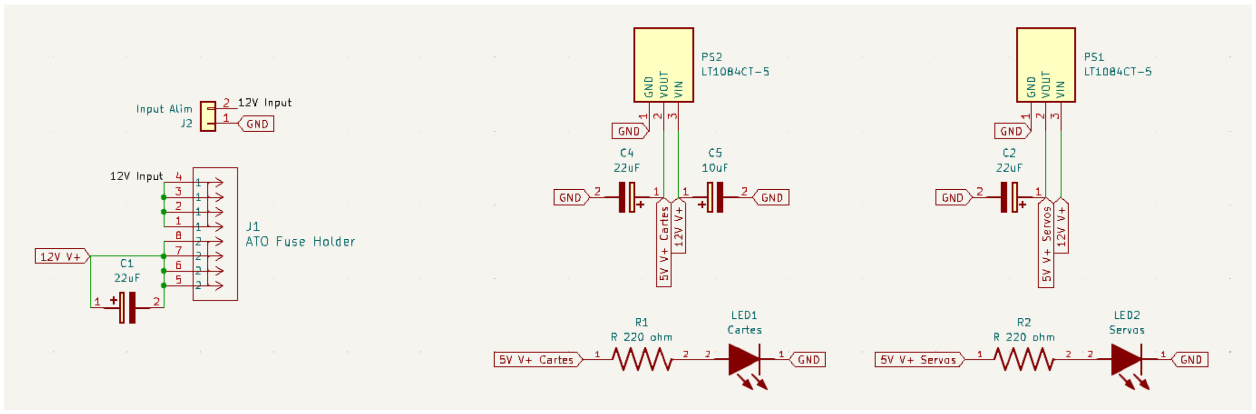


Figure 8 : Power Management

6.1.1 Float

6.1.1.2.1 Pressure sensor.

Figure 9 : SEN057 Sensor

We opted for the SEN0257 pressure sensor to be integrated into the float primarily for its ability to provide precise and continuous pressure measurements during dives. While we do not utilize the depth data for direct control purposes, it serves a critical role in providing valuable insights into the underwater environment and the float's behavior.

The pressure data collected by the SEN0257 sensor is transmitted to the control box on the shore, where it is processed and stored for later analysis. Although we do not graph the data in real-time, it serves as a valuable resource for post-dive analysis and optimization.

By analyzing the pressure data from each dive, we gain a deeper understanding of the float's performance and behavior in different underwater conditions. This information allows us to refine our designs and improve the float's efficiency and reliability for future missions.

6.1.1.2.2 Arduino Nano ESP32 & Data Transmission

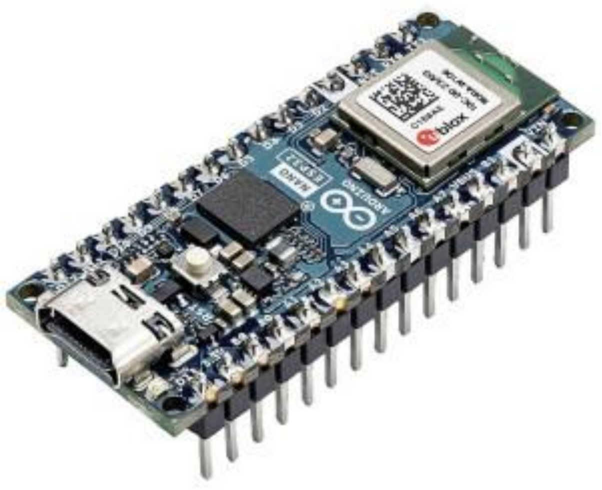


Figure 10 : Arduino Nano ESP32

We have selected the Arduino Nano ESP32 Card as the control unit for the float, considering its robust communication capabilities, ease of implementation, compact size, and cost-effectiveness.

In our search for a reliable communication protocol, we explored various options. However, concerns about frequency interference, data loss, and overall uncertainty prompted us to discard radio systems as a viable choice.

Our chosen approach involves integrating an Arduino board equipped with Bluetooth and WiFi capabilities, leveraging ESP32 modules. This solution offers a seamless management of float controls and communication protocols.

Notably, the Arduino Nano ESP32 Card's compact size, affordable price, and versatile form factor make it an ideal choice for integration into the float's design. These factors ensure that the control unit fits within the vehicle's limited space while also aligning with our budgetary constraints.

6.1.1.2.3 Circuit Schematic

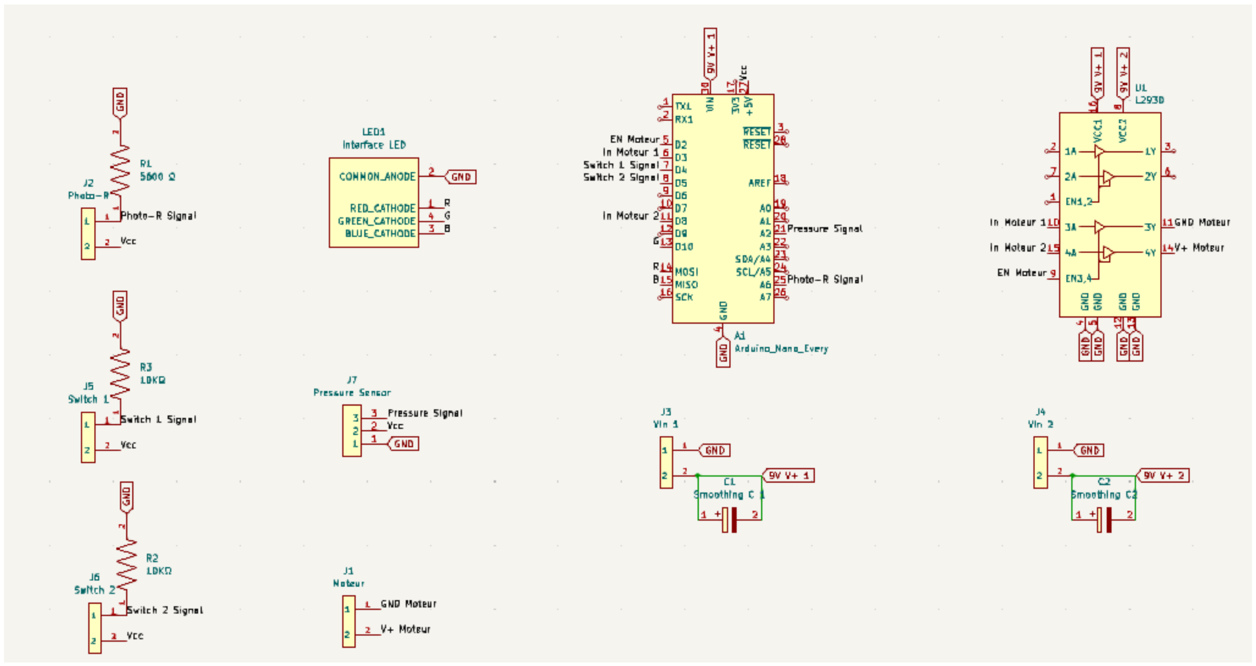


Figure 11: Float Electrical Schematic

6.1.2 Safety

6.1.2 Fuses

Fuses play a crucial role in protecting our ROV and float from overcurrent situations, ensuring the safety of both the equipment and operators. For the ROV, we employ ATO Blade Fuses, with a 25A fuse rating, strategically placed in the electrical system (refer to Figure 8 : Power Management). These fuses act as fail-safes, interrupting the circuit in case of excessive current flow, thus preventing damage to components and potential hazards. Similarly, for the float, multiple 1A fuses are installed to safeguard against overcurrent events, tailored to the specific power requirements of its components (refer to Figure 11: Float Electrical Schematic).

6.1.2 Andersons

In compliance with competition regulations, we utilize Andersons PowerPole Connectors for the 12V connections in our ROV and float. These connectors offer a reliable and standardized interface, facilitating quick and secure connections between various components. Their robust design ensures efficient power transmission while minimizing the risk of accidental disconnections, contributing to the overall safety and reliability of our electrical system.

6.1.2 Circuit protection and power smoothing

To enhance the stability and reliability of our electrical system, we implement various circuit protection and power smoothing techniques. Diodes are strategically placed to prevent reverse current flow, safeguarding sensitive components from potential damage. Additionally, capacitors are employed to smooth out input voltage fluctuations and stabilize the output voltage of the LT1084 regulators mentioned earlier. These measures not only protect our equipment from electrical anomalies but also contribute to smoother operation and increased longevity of our ROV and float systems.

6 PCB Design

6.2.1 ROV

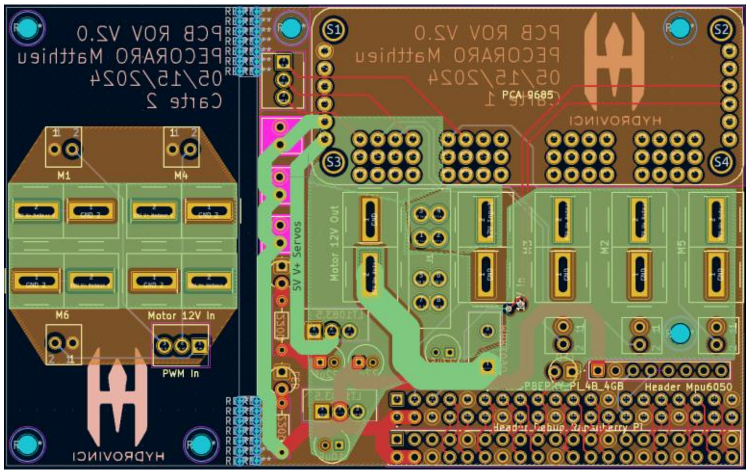


Figure 12 : ROV General PCB Layout

6.2.1 *Spatial Constraint*

The design of the ROV's PCB was governed by a stringent spatial constraint, driven by the need to accommodate Anderson Powerpole connectors within a narrow 74mm inner diameter tube. This requirement necessitated the PCB to be compact, allowing seamless integration with the connectors while maximizing space efficiency within the confined enclosure.

6.2.1 *2 Signal Layers*

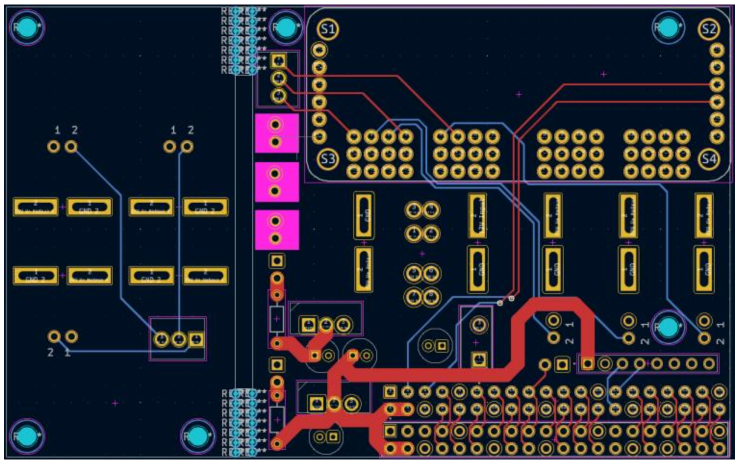


Figure 13 : ROV PCB Signal Planes

The ROV's PCB layout comprises four distinct layers, meticulously organized to optimize functionality and performance. Each layer serves a specific purpose, from signal routing to power distribution, ensuring efficient operation of the vehicle's electronic systems.

In particular, we use the outer most layers for signal routing to all of the diverse components on the board.

6.2.1 *2 Power Planes*

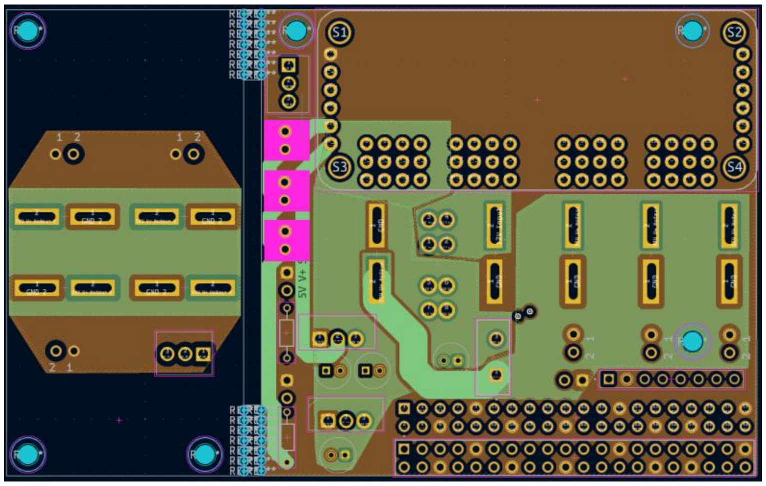


Figure 14 : ROV PCB Power Planes

The power distribution within the ROV's PCB is carefully planned to minimize power loss and heating while maximizing efficiency. A general ground plane is situated on the inner back layer, providing a stable reference for signal integrity and noise reduction. Meanwhile, the inner front layer houses all 12V 25Amp connections, employing adapted trace widths to mitigate power loss and ensure reliable operation under demanding conditions.

6.2.2 Float

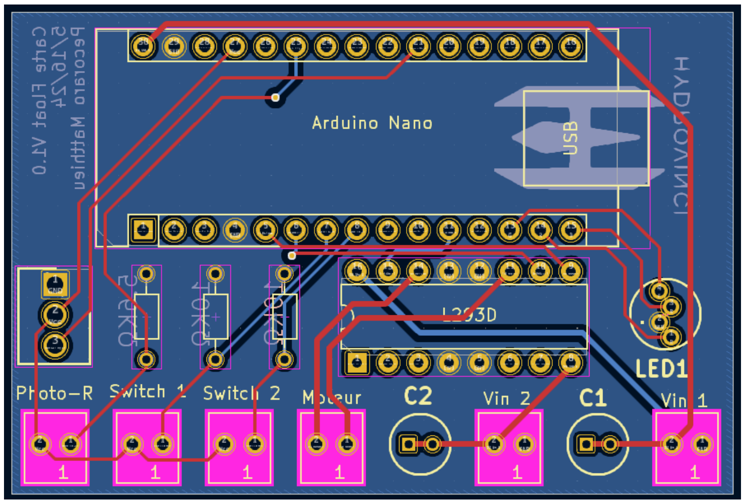


Figure 15 : PCB Layout of the Float

The PCB design for the float serves as the backbone of its electronic system, facilitating efficient operation and communication in underwater environments. Tailored to meet the unique requirements of the float, the PCB layout is optimized for compactness and functionality.

With a focus on reliability and performance, the float's PCB incorporates essential components and connectors necessary for seamless integration and operation. Special attention is paid to spatial constraints, ensuring compatibility with the vehicle's structure and components.

6 PCB INTEGRATION

6.3.1 Mechanical Overview

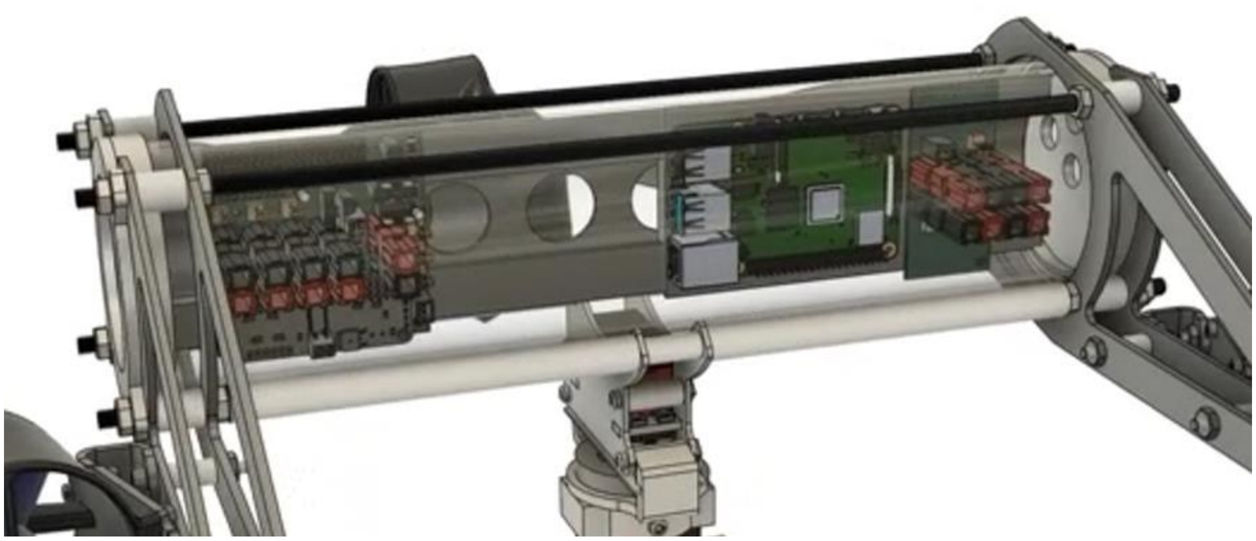


Figure 16 : Mechanical Integration of the PCBs

The mechanical integration of the PCBs is facilitated by a robust and versatile 3D printed internal structure. This structure incorporates a rail system that allows for secure attachment and alignment of the PCBs within the ROV. By providing a stable mounting platform, this design ensures optimal positioning and functionality of the electronic components, contributing to the overall efficiency and reliability of the system.

6.3.2 Cable management

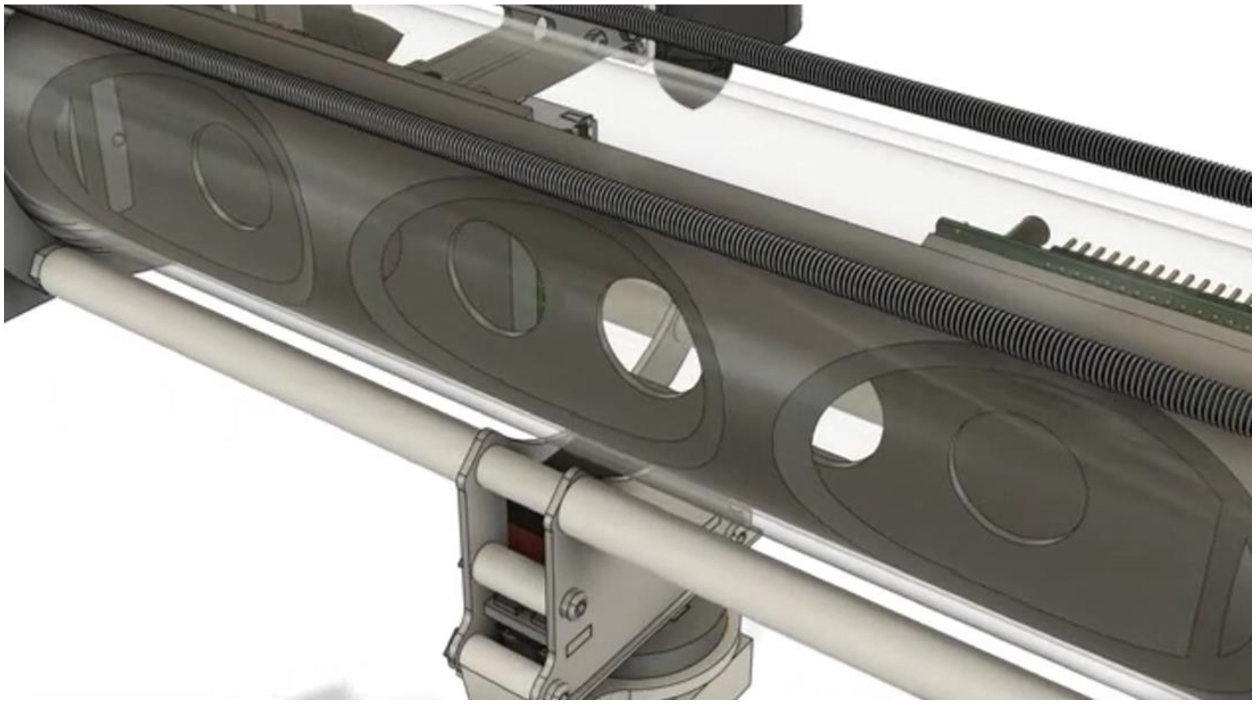


Figure 17 : Cable Management Channel

Effective cable management is essential for maintaining the integrity and organization of the ROV's electrical system. To address this, the 3D printed structure housing the PCBs features a hollow design with a wide channel, allowing cables to pass through smoothly. This design not only ensures neat and tidy cable routing but also minimizes the risk of tangling or interference, promoting seamless operation and maintenance of the ROV system.

7 SOFTWARE

7 RUST PROGRAMMING

For this project, we chose the Rust language for the embedded computer unit. This choice was motivated by several key advantages that Rust offers, particularly in the context of embedded systems and robotics.

7.1.1 Efficiency and Memory Safety

Rust is renowned for its efficiency in memory allocation and computation. Its ownership system ensures memory safety without the need for a garbage collector, which is crucial for real-time applications like controlling a submarine drone. This guarantees that the system remains performant and free of common programming errors such as null pointer dereferencing and buffer overflows.

7.1.2 Asynchronous Programming

Rust's powerful asynchronous capabilities were another reason for our choice. The async/await syntax in Rust allows for efficient non-blocking operations, which is essential when handling multiple tasks such as motor control, sensor data processing, and communication simultaneously. This ensures that the drone can perform complex operations without delays or interruptions.

7.1.3 Extensive Library Support

Rust boasts a rich ecosystem of libraries (crates) that facilitate development for embedded electronics. We leveraged several of these libraries to interface with various components of the drone. For instance, we used the `embedded-hal` and `rppal` crates to interact with GPIO pins and control hardware peripherals.

7.1.4 Motor and Servo Control

The Rust code is responsible for controlling the motors and servomotors of the drone. By receiving commands from the control station via an Ethernet cable using the TCP protocol, the embedded system can adjust the speed and direction of the motors. This is achieved through precise PWM (Pulse Width Modulation) signals generated by the Rust program.

```rust
use tokio::net::TcpListener;
use tokio::io::AsyncReadExt;

use linux_embedded_hal::{Delay,I2cdev};
use pwm_pca9685::{Channel, Pca9685, Address};

use std::error::Error;
use std::thread;
use std::time::Duration;


#[tokio::main]
async fn main() -> Result<(),Box<dyn Error>> {
        let listener = TcpListener::bind("0.0.0.0:12345").await?;
        println!("connected");

        let i2c_pca = I2cdev::new("/dev/i2c-1")
        .map_err(|e| format!("Failed to open I2C device: {:?}", e))?;
        //let mut delay = Delay;

    // Attempt to initialize PCA9685
        let pca_address = Address::default(); // default I2C address for PCA9685
        let mut pca= Pca9685::new(i2c_pca, pca_address).unwrap();
        //pca.init(&mut delay);

        pca.set_prescale(127).unwrap(); // Set frequency of PWM outputs to 50Hz

        pca.enable().unwrap();
        pca.set_channel_on_off(Channel::C2,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C1,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C0,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C5,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C4,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C3,0, 307).unwrap();
        thread::sleep(Duration::from_millis(5000));

        pca.set_channel_on_off(Channel::C8,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C9,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C10,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C11,0, 307).unwrap();
        pca.set_channel_on_off(Channel::C12,0, 307).unwrap();
        println!("everything is initialized");


        while let Ok((mut socket, addr))=listener.accept().await{
            println!("connection from {}",addr);

            let mut buf=[0u8;44];
            while let Ok(_)=socket.read_exact(&mut buf).await{
                let data_points=[
                f32::from_le_bytes(buf[0..4].try_into().unwrap()),//x
                f32::from_le_bytes(buf[4..8].try_into().unwrap()),//y
                f32::from_le_bytes(buf[8..12].try_into().unwrap()),//z
                f32::from_le_bytes(buf[12..16].try_into().unwrap()),//rot
                f32::from_le_bytes(buf[16..20].try_into().unwrap()),
                f32::from_le_bytes(buf[20..24].try_into().unwrap()),
                f32::from_le_bytes(buf[24..28].try_into().unwrap()),
                f32::from_le_bytes(buf[28..32].try_into().unwrap()),
                f32::from_le_bytes(buf[32..36].try_into().unwrap()),
                f32::from_le_bytes(buf[36..40].try_into().unwrap()),
                f32::from_le_bytes(buf[40..44].try_into().unwrap())
                ];
                println!("received data: {:?}", data_points);


        let b0=data_points[6];
        let b1=data_points[7];
        let b2=data_points[8];
        let b3=data_points[9];
        let b4=data_points[10];

        let h2=307.0-data_points[1]*60.0;//avant gauche
        let h1=307.0-data_points[1]*60.0;//avant droit
        let h3=307.0+data_points[0]*60.0;//arrière
        let h4=307.0-data_points[2]*25.0+data_points[4]*20.0+data_points[5]*20.0;//vertical avg

        let h5=307.0-data_points[2]*25.0-data_points[4]*20.0+data_points[5]*20.0;// vertical avd

        let h6=307.0-data_points[2]*50.0-data_points[5]*30.0; // vertical ar


        pca.set_channel_on_off(Channel::C2, 0,h1.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C3, 0,h2.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C1, 0,h3.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C0,0, h5.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C4,0, h4.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C5,0, h6.round() as u16).unwrap();


        pca.set_channel_on_off(Channel::C8, 0,b0.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C9, 0,b1.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C10, 0,b2.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C11, 0,b3.round() as u16).unwrap();
        pca.set_channel_on_off(Channel::C12, 0,b4.round() as u16).unwrap();

        thread::sleep(Duration::from_millis(100));
        }
    }
    Ok(())
```

```
}

```

7.1.5 Stabilization System

One of the critical aspects of our project is the stabilization system, which ensures the drone maintains the desired orientation underwater. We implemented this using the MPU6050 sensor, a combined accelerometer and gyroscope. The Rust code processes the sensor data to compute the pitch and roll angles. This is done using quaternions, which are calculated from the MPU6050's Digital Motion Processor (DMP) data. Quaternions provide a robust method for representing 3D orientations and rotations, avoiding issues like gimbal lock that can occur with Euler angles.

```
        let i2c_mpu=I2cMPU::new("/dev/i2c-1").map_err(|e| format!("Failed to open I2C device: {:?}", e)).unwrap();
        let mut mpu=Mpu6050::new(i2c_mpu);
        let mut delay=DelayMPU;
        mpu.init(&mut delay).expect("erreure d'initialisation");
        let mut accel;
        let mut roll;
        let mut pitch;

        let mut mean_roll=0.0;
        let mut mean_pitch=0.0;

        let mut pid_pitch=Pid::new(0.0,90.0);
        let mut pid_roll=Pid::new(0.0,90.0);
        pid_pitch.p(1.0,90.0).i(0.1,90.0).d(0.01,90.0);
        pid_roll.p(1.0,90.0).i(0.1,90.0).d(0.01,90.0);

        let start = Instant::now();
        while start.elapsed() < Duration::new(10, 0){
            accel=mpu.get_acc_angles().unwrap();
            mean_roll+=accel[0].to_degrees();
            mean_pitch+=accel[1].to_degrees();
            thread::sleep(Duration::from_millis(50));
        }
        mean_pitch/=200.0 as f32;
        mean_roll/=200.0 as f32;

        println!("mpu calibrated");

                accel=mpu.get_acc_angles().unwrap();
                roll=accel[0];
                pitch=accel[1];

                roll=roll.to_degrees() as f32;
                roll-=mean_roll;
                if roll.abs()<3.0{
                    roll=0.0;
                }
                pitch=pitch.to_degrees() as f32;
                pitch-=mean_pitch;
                if pitch.abs()<3.0{
                    pitch=0.0;
                }
                let correction_pitch=pid_pitch.next_control_output(pitch).output;
                let correction_roll=pid_roll.next_control_output(roll).output;

```

```
                    let new_correction_pitch=(correction_pitch+90.0)*2.0/180.0-1.0;
                    let new_correction_roll=(correction_roll+90.0)*2.0/180.0-1.0;

                    println!("correction pitch: {}, correction roll: 
{}",new_correction_pitch,new_correction_roll);
```

7.1.6 Summary

In summary, Rust's efficiency, memory safety, asynchronous programming capabilities, and extensive library support made it an ideal choice for the embedded system of our submarine drone. By leveraging these features, we were able to create a reliable, performant, and safe control system that meets the demanding requirements of underwater navigation and stabilization.

7 GENERAL ARCHITECTURE

The system architecture of our project involves two main computing units: an onboard Raspberry Pi 4 and a control box laptop. Each plays a crucial role in the operation and control of the ROV.

7.2.1 Onboard Raspberry Pi 4

The onboard Raspberry Pi 4 is the core of the ROV's control system. It hosts all the Rust code responsible for the real-time control of the vehicle. This includes managing motor functions, servomotors, and the stabilization system. The Raspberry Pi 4 was chosen for its powerful processing capabilities, making it suitable for handling the complex computations required for real-time control and sensor data processing.

7.2.2 Control Box Laptop

The control box contains a laptop that serves as the user interface and primary command center for the ROV. This laptop runs Python code to interface with user input devices, such as a joystick and a twin arm controller. The joystick provides directional commands, while the twin arm controller, interfaced via an Arduino, transmits the potentiometer values to the laptop. These inputs are then translated into movement commands and sent to the Raspberry Pi.

```python
import socket
import time
import pygame
import sys
import struct
import serial

pygame.init()
pygame.joystick.init()
joystick = pygame.joystick.Joystick(0)
joystick.init()
axis_indices = [0, 1, 2, 4]  # indices for X, Y, Z, RZ axes

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect(('169.254.245.11', 12345))

        time.sleep(1)
        arduino=serial.Serial(port='COM3',baudrate=9600, timeout=1)
        print("fully connected")
        try:
            arduino.write(b"test")
            while True:
                val=0.0
                power=0.0
                pygame.event.pump()
                data = bytearray()
                pwm=arduino.read(100)
                for index in axis_indices:
                    axis_value = joystick.get_axis(index)
                    #if abs(axis_value) < 0.1:
                    #    axis_value = 0.0
                    #print(index, axis_value)
                    data.extend(struct.pack('f', axis_value))

                for i in pwm:

                    if (i == 59):
                        print(";")
                        data.extend(struct.pack('f',val))
                        val = 0.0
                        power = 0
                    elif (chr(i).isnumeric()) :
                        val+=float(chr(i)) * pow(10, power)
                        power += 1
                        print(val)

                s.sendall(data)
                time.sleep(0.1)

        except Exception as e:
                print(f"Error sending data: {e}")

pygame.quit()
sys.exit()

```

7.2.3 Communication and Network Setup

To facilitate communication between the control box laptop and the onboard Raspberry Pi, we established a local network. Instructions are transmitted over this network using the TCP protocol, ensuring reliable and ordered delivery of commands.

The control box laptop connects to the onboard Raspberry Pi using a secure SSH (Secure Shell) connection. This allows for remote control and monitoring of the ROV's systems, enabling the operator to execute commands and retrieve system status updates in real-time.

7.2.4 Data Flow and Command Transmission

User Input: The operator uses the joystick and twin arm controller to generate movement commands.

Python Interface: The control box laptop, running Python code, retrieves these commands and processes them.

Arduino Integration: For the twin arm controller, an Arduino collects the potentiometer values and sends them to the laptop.
Command Transmission: The processed commands are sent from the laptop to the onboard Raspberry Pi over the local network using the TCP protocol.
Rust Control System: The Raspberry Pi, running Rust code, receives these commands and executes the necessary actions to control the ROV's motors, servomotors, and stabilization mechanisms.

7.2.5 Summary

The dual-computer architecture, with the Raspberry Pi 4 onboard the ROV and the control box laptop, provides a robust and efficient control system. By leveraging a combination of Rust and Python, along with an Arduino for analog input, we created a seamless integration of hardware and software. This architecture ensures precise control, real-time response, and reliable communication, all critical for the successful operation of the underwater drone.

8 Conclusion

In conclusion, our technical report encapsulates the comprehensive design, development, and implementation of our ROV and float systems, poised at the forefront of underwater exploration. Through meticulous planning and innovation, we have overcome numerous challenges, from spatial constraints to electrical safety considerations, to create a robust and efficient underwater platform. Leveraging advanced techniques such as Rust embedded software programming for precise control and integration of a 5-axis robotic arm for versatile operations, we have crafted systems that excel in both functionality and reliability. Our commitment to excellence is evident in every aspect of our design, from the compact layout of PCBs to the seamless cable management solutions. As we navigate the depths, our systems stand as a testament to the power of collaboration, ingenuity, and perseverance in pushing the boundaries of underwater exploration.