POLITECNICO DI BARI

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING
Master Degree in Automation Engineering - Cyber Physical System

---

Dissertation in Robotics

Modelling, Control, and Experimental Verification of a Haptic System for an Underwater Robot

Supervisor
Prof. Paolo Lino

Co-Supervisor
Prof. Vincente Feliu Batlle

Candidate
Antonio Camposeo

Academic Year 2022 - 2023

Contents

Nomenclature

(1) (x_c, y_c) = position of the mass center of the robot

(2) φ = orientation of the robot

(3) h = distance between the mass center of the robot and the joint motor

(4) (x_b, y_b) = position of the actuated joint of the flexible antenna

(5) θ = angular position of tip mass without deflection

(6) θ_t = angular position of the tip mass

(7) (x_t, y_t) = position of the tip of antenna

(8) L = length of the tip of antenna

(9) EI = stiffness of the antenna

(10) ψ = deflection angle from X_1

(11) Γ = bending moment at the base of the link

(12) $g_x(\lambda)$,$g_y(\lambda)$ = profile of the wall

(13) (X_R, Y_R) = body reference frame

(14) (X_I, Y_I) = global reference frame

(15) (X_0, Y_0) = base frame for flexible antenna

(16) (X_1, Y_1) = reference frame for tip of antenna

(17) P = point of contact

(18) v = tip velocity direction

(19) V = constant velocity

(20) ξ = orientation of the rov

(21) λ = parameter to describe the wall profile

(22) d = distance estimated from the wall and the rov

(23) F_w = drag force

(24) F_c = Force of contact

(25) $\overrightarrow{F_w}$ = drag force

(26) $\overrightarrow{F_{w,\perp}}$ = perpendicular projection of drag force along the straight line between motor position and tip of antenna

(27) $\overrightarrow{F_{w,\parallel}}$ = parallel projection of drag force along the straight line between motor position and tip of antenna

(28) $\overrightarrow{F_c}$ = contact forces

(29) $\overrightarrow{F_{c,\perp}}$ = perpendicular projection of contact force along the straight line between motor position and tip of antenna

(30) $\overrightarrow{F_{c,\parallel}}$ = parallel projection of contact force along the straight line between motor position and tip of antenna

(31) $\overrightarrow{F_{\parallel,1}}$ = parallel reaction force of revolving joint along axis the straight line between motor position and tip of antenna

(32) $\overrightarrow{F_{\perp,1}}$ = perpendicular reaction force of revolving joint along axis the straight line between motor position and tip of antenna

(33) $\overrightarrow{F_{\parallel,10}}$ = projection of $\overrightarrow{F_{\perp,1}}$ and $\overrightarrow{F_{\parallel,1}}$ along base frame axis(X_1)

(34) $\overrightarrow{F_{\perp,10}}$ = projection of $\overrightarrow{F_{\perp,1}}$ and $\overrightarrow{F_{\parallel,1}}$ along base frame axis (Y_1)

(35) $\overrightarrow{F_{\parallel,01}}$ = reaction force ROV side of $\overrightarrow{F_{\parallel,10}}$

(36) $\overrightarrow{F_{\perp,01}}$ = reaction force ROV side of $\overrightarrow{F_{\perp,10}}$

(37) $\overrightarrow{F_{x,0}}$ = projection of $\overrightarrow{F_{\perp,0,1}}$ and $\overrightarrow{F_{\parallel,0,1}}$ along base frame axis(X_0)

(38) $\overrightarrow{F_{y,0}}$ = projection of $\overrightarrow{F_{\perp,0,1}}$ and $\overrightarrow{F_{\parallel,0,1}}$ along base frame axis(Y_0)

Introduction

Underwater robot sensing technologies have recently attracted considerable attention in marine engineering and resource exploration. On the one hand, underwater robots need to sense the environment and perform autonomous navigation and obstacle avoidance. On the other hand, underwater robots also depend on the ability of sensing technology to perform a number of practical application tasks (e.g. object detection, underwater robot grasping and high-precision 3D measurement). In summary, the sensing technology of underwater robots is playing an important role in robots. Most underwater sensing technologies are based on acoustic signals (e.g. sonar), light signals, electromagnetic signals and bionic sensors. In particular, sonar estimates the position of a submerged object by measuring the travel time and phase difference of acoustic signals. Measuring the travel time and phase difference of acoustic pulses can work over a much greater distance and cannot be affected by water turbidity. Although underwater acoustic detection methods have a wide detection range, their resolution is low, which limits the practical applications of underwater sonar. Optical sensors capture light rays from the surroundings to acquire environmental information. Higher resolution and refresh rate are required to acquire information about the surroundings. However, due to the complicated conditions of underwater light (absorption and scattering), optical sensors can only achieve short-range detection. An electromagnetism-based sensor could be applied in an underwater environment to accurately estimate distance as well. However, environmental electromagnetic fields can interfere with accuracy. In addition, researchers have started to investigate underwater bionic sensing technologies (e.g. whiskers and lateral lines). However, these technologies are not mature enough and need to be improved in practice . Each of the aforementioned technologies has its own advantages and disadvantages, and researchers need to combine multiple sensing methods to perform various underwater exploration tasks in practice. [24]

The aim of the project is to be able to develop, implement and control a sensing active antenna placed on a UUV to simulate bionic underwater detection of marine animals such as lobsters. The antenna will consist of a flexible arm, attached to the end of the UUV with a rotary joint. The special feature of the flexible arm is the possibility of being able to derive the bending of the arm, so that by considering the contact with a surface we can obtain the position of the point at the moment of contact, thus obtaining the surface with which the flexible arm is in contact.

Chapter 1

System Components

The project proposal involves the revision of the underwater system consisting of the BlueROV2 underwater robot from the company BlueRobotics. An additional component is to be implemented to this system, namely a flexible rod with two underwater Dynamixle motors and a Force-Tourque 6D sensor to study the dynamics of the flexible rod.
In the following sections, the work carried out to date will be described, starting with the components of the system, the rov software update and the ROS implementation, and then moving on to the antenna implementation.

1.1 Hardware

1.1.1 BlueROV2

Figure 1.1: BlueROV2 Standard

The BlueROV2[4] is a tethered ROV. The tether is used for communication between the vehicle and a top-side unit. The vehicle can be powered either through the tether or by an onboard battery. The ROV comes with various onboard sensors, including a

pressure transmitter, leak detection sensors, a power sense module, an Inertial Measurement Unit (IMU), and a magnetometer. The IMU consists of an accelerometer and a gyroscope. The IMU and magnetometer are integrated in a Pixhawk. The open-source software architecture allows adding additional sensors such as an Underwater Acoustic Positioning System (UAPS), Doppler Velocity Log (DVL), camera vision, sonar, lasers, etc.
Software implementation of the sensors on the physical hardware can be done on the Pixhawk through ArduSub or on the Raspberry Pi using the MAVLink protocol. The main control board on the BlueROV2 is a Raspberry Pi 3, which is connected to a top-side computer through an Ethernet connection. Besides the Raspberry Pi 3, the electronic tube contains a Pixhawk. The Ethernet communication is transferred top-side through a Fathom-X Tether Interface, which provides a long-distance Ethernet connection over a single twisted pair of wires. The hardware and interfaces of the BlueROV2 is shown in the table.

Sensor Technology	Sample Rate	Delay	Variance (Filtered)
Pressure Sensor	20 Hz	-	$1.5x10^{-5}m$
IMU	100 Hz	-	$1x10^{-5}$ rad to $2.5x10^{-5}$rad
Magnetometer	100 Hz	-	-
Laser-Camera	20 Hz	-	$2x10^{-6}m$
Sonar	40 Hz	-	-
Short Baseline	4 Hz	1.2 s	-
Doppler Velocity Log	15 Hz	0.1 s	-

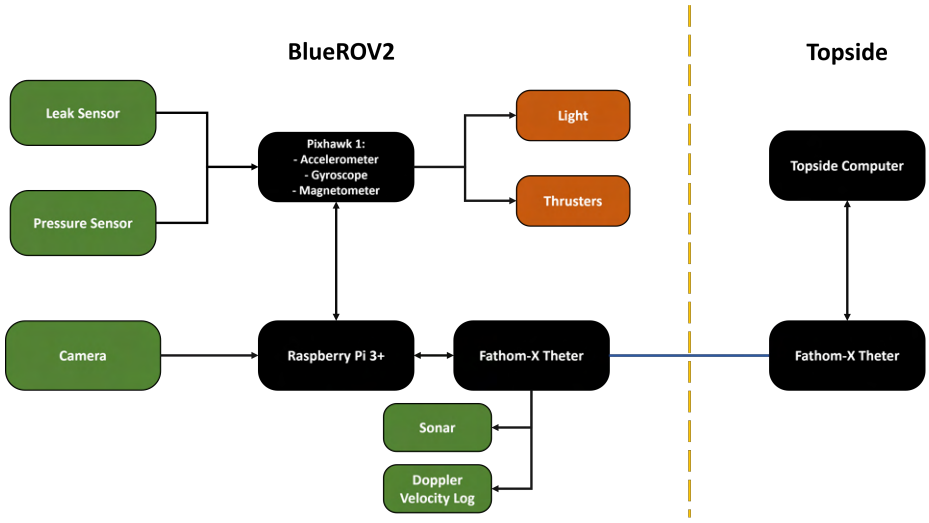

Figure 1.2: Hardware and interfaces on BlueROV2, green boxes represents sensors, black boxes represents processing units/control units, the yellows boxes represents actuators, and the dashed blue line illustrates the cut between onboard hardware and topside hardware.

Blue Robotics Inc's gives two configuration for the rov. There is the BlueROV2 Standard and BlueROV2 Heavy, the main difference is in thurster allocatin between the normal and the heavy. In the figure below is present the difference.[29]

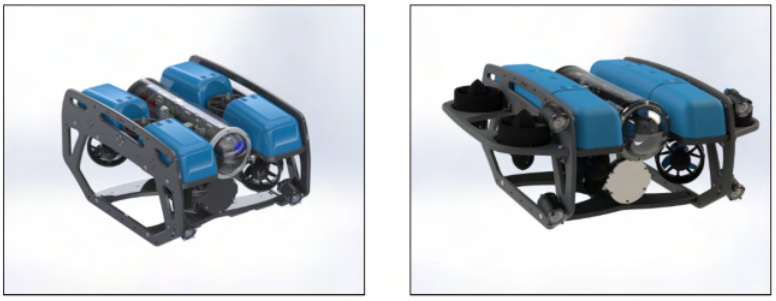

Figure 1.3: Two illustrations of the BlueROV2 with and without heavy configuration, respectively

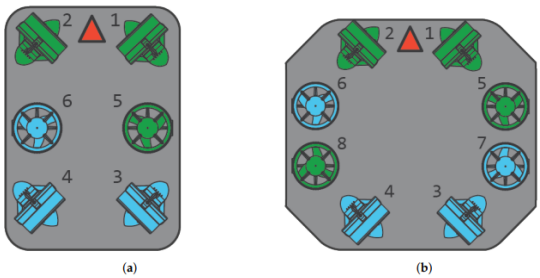

Figure 1.4: A comparison of the thruster configuration of the normal and heavy BlueROV2. The blue thrusters are the clockwise thrusters, and the green are the counterclockwise thrusters. The red arrow is the positive surge direction . (a) BlueROV2 normal configuration. (b) BlueROV2 heavy configuration

Motion	Normal BlueROV2	Heavy BlueROV2
Surge	T1, T2, T3 and T4	T1, T2, T3 and T4
Sway	T1, T2, T3 and T4	T1, T2, T3 and T4
Heave	T5 and T6	T5, T6, T7 and T8
Roll	T5 and T6	T5, T6, T7 and T8
Pitch	Uncontrollable	T5, T6, T7 and T8
Yaw	T1, T2, T3 and T4	T1, T2, T3 and T4

Table 1.1: Thurster allocation

1.1.2 Raspberry Pi 3B+

Figure 1.5: Enter Caption

The Raspberry Pi is a credit card-sized computer. The Raspberry Pi 3 Model B+ is an improved version of the Raspberry Pi 3 Model B. It is based on the BCM2837B0 system-on-chip (SoC), which includes a 1.4 GHz quad-core ARMv8 64bit processor and a powerful VideoCore IV GPU. The Raspberry Pi can run a full range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, Raspbian, Fedora, and Arch Linux, as well as Microsoft Windows 10 IoT Core. The Raspberry Pi 3 Model B+ has many performance improvements over the Model B including a faster CPU clock speed (1.4GHz vs 1.2GHz), increased Ethernet throughput, and dual-band WiFi. It also supports Power over Ethernet with a Power over Ethernet HAT (not included). The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market. [18]

Features:

- 1.4 GHz quad-core BCM2837B0 ARMv8 64bit CPU
- 1 GB RAM
- VideoCore IV 3D graphics core
- Ethernet port
- dual-band (2.4 GHz and 5 GHz) IEEE 802.11.b/g/n/ac wireless LAN (WiFi)
- Bluetooth 4.2
- Bluetooth Low Energy (BLE)
- Four USB ports
- Full-size HDMI output
- Four-pole 3.5 mm jack with audio output and composite video output
- 40-pin GPIO header with 0.1 spaced male pins that are compatible with our 2×20 stackable female headers and the female ends of our premium jumper wires.
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot

1.1.3 Pixhawk 1

Figure 1.6: Flight Controller Pixhawk1

The Pixhawk 1 autopilot is a popular general purpose flight controller based on the Pixhawk-project (FMUv2) open hardware design (it combines the functionality of the PX4FMU + PX4IO). It runs PX4 on the NuttX OS. [12]

Features

- Main System-on-Chip: STM32F427
  - CPU: 180 MHz ARM® Cortex® M4 with single-precision FPU
  - RAM: 256 KB SRAM (L1)
- Failsafe System-on-Chip: STM32F100
  - CPU: 24 MHz ARM Cortex M3
  - RAM: 8 KB SRAM
- Wifi: ESP8266 external
- GPS: U-Blox® 7/8 (Hobbyking®) / U-Blox 6 (3D Robotics)
- Optical flow: PX4 Flow unit
- Redundant power supply inputs and automatic failover
- External safety switch
- Multicolor LED main visual indicator
- High-power, multi-tone piezo audio indicator
- microSD card for high-rate logging over extended periods of time

Connectivity

- 1x I2C
- 1x CAN (2x optional)

- 1x ADC
- 4x UART (2x with flow control)
- 1x Console
- 8x PWM with manual override
- 6x PWM / GPIO / PWM input
- S.BUS / PPM / Spektrum input
- S.BUS output

1.1.4 Fathom X Tether

Figure 1.7: Tether Interface Board

The Fathom-X Tether Interface Board Set provides a high-speed, long-distance Ethernet connection to an ROV or other remote platform. They're designed for use with the Fathom tether, standard Cat5 cable, or even a single twisted pair of wires. This board uses the Rak Wireless LX200V20 module, which leverages the robust HomePlug AV (IEEE-1901) standard for sending Ethernet through powerlines.
The Fathom-X Tether Interface Board Set is open-source and open-hardware, so the schematics and board files are free to view, modify, and reuse. [6]
Features:

- 80 Mbps Ethernet over two wires
- 300m+ tether length capability
- Plug-and-play with no setup involved
- Onboard switching power supply with 7-28V input range
- USB Mini-B connector for powering directly from a computer on the topside
- Indicator LEDs for power, link, and data
- Included 6" Ethernet cable for connection to onboard computer

1.1.5 OpenCM 9.04-C

Figure 1.8: OpenCM9.04-C

The Robotis OpenCM9.04-C is a small robot controller featuring Dynamixel XL Smart Servo connectors and a 32 bit ARM Cortex M3 processor.
The OpenCM 9.04-C has an STM32F103CB 32 bit ARM Cortex-M3 microcontroller with 128K bytes of Flash, 20K bytes of SRAM and full set of GPIOs. This robot controller features four 3 pin Dynamixel TTL Bus connectors for use with Dynamixel XL series smart servos. It also has four sets of 5 pin male header sensor ports, UART, JTAG/SWD Serial Wire Debug, Micro USB.
This robot controller offers a full set of GPIOs available via a set of solder pads. There are 26 GPIOs which include up to 10 analog inputs, 13 PWM channels, 2 USARTs, 2 SPI channels and I2C. A built in user LED and button are available to general use. A power indicator LED, reset button and power switch are also provided. Develop C/C++ programs using the Arduino IDE using OpenCM Libraries.
The OpenCM 9.04C robot controller operates at 3.3V and requires a 5V- 16V input power source. It can be powered by USB for programming and experimentation but requires a separate power supply when driving Dynamixel Smart Servos and other parts.[11]

1.1.6 Expansion Board OpenCM9.04

Figure 1.9: OpenCM 485 Expansion Board

The ROBOTIS OpenCM 485 Expansion Board was designed to control Dynamixel servomotors. Due to the powerful OpenCM9.04-A / OpenCM9.04-C development board which is based on an ARM Cortex-M3 MCU, it is possible to solve a lot of challenging task without any problems. The combination of the OpenCM 485 and the OpenCM9.04 has very small dimensions and therefore opens up for possible development aswell as redevelopment of autonomous processes.
Most of the Dynamixel controller boards that are available at the moment are provided with a TTL interface to control any servomotor of the AX/MX/X series, or an RS485 interface for the Dynamixel pro servomotors. The OpenCM9.04 development board comes with both TTL & RS485 interface allowing for a high flexibility when it comes to the usage.[10]

1.1.7 Arduino UNO

Figure 1.10: Arduino Uno

The Arduino Uno is based on the ATmega328P microprocessor, which has only 2 KB of RAM. This memory limitation is a desired characteristic, given that mTask's suitability for microcontrollers is being tested and that such devices might have extremely limited resources.

The Arduino Uno operates at a frequency of 16 MHz, has 6 analog pins, 14 digital pins, 32 KB of flash memory and a built-in LED. Its digital and analog pins are often used to interface with peripherals, including sensors and actuators.[?]

1.1.8 DC682A ADC Converter

Figure 1.11: DC682A ADC Converter

The LTC®1857/LTC1858/LTC1859 are 8-channel, low power, 12-/14-/16-bit, 100ksps, analog-to-digital converters (ADCs).
These SoftSpan™ ADCs can be softwareprogrammed for 0V to 5V, 0V to 10V, ±5V or ±10V input spans and operate from a single 5V supply. The 8-channel multiplexer can be programmed for single-ended inputs or pairs of differential inputs or combinations of both. In addition, all channels are fault protected to ±25V. A fault condition on any channel will not affect the conversion result of the selected channel.
An onboard high performance sample-and-hold and precision reference minimize external components. The low 40mW power dissipation is made even more attractive with two user selectable power shutdown modes. DC specifications include ±3LSB INL for the LTC1859, ±1.5LSB INL for the LTC1858 and ±1LSB for the LTC1857. The internal clock is trimmed for 5µs maximum conversion time and the sampling rate is guaranteed at 100ksps. A separate convert start input and data ready signal (BUSY) ease connections to FIFOs, DSPs and microprocessor.[1]

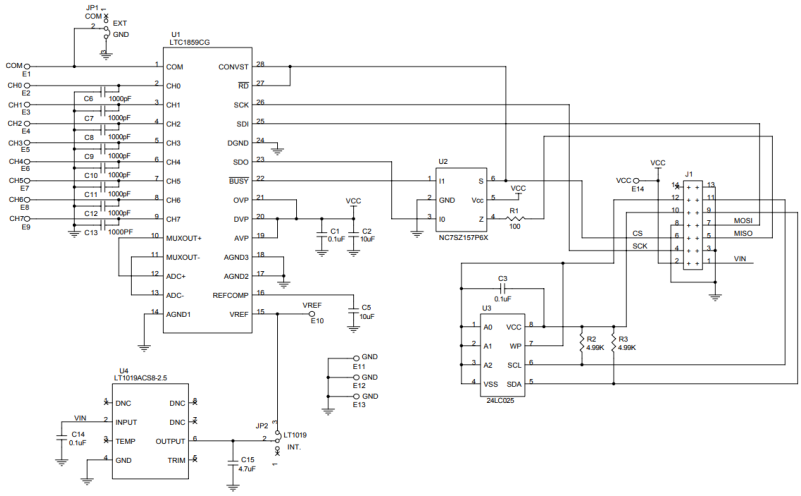

Figure 1.12: Converter Scheme

Features:

- Sample Rate: 100ksps
- 8-Channel Multiplexer with ±25V Protection
- Single 5V Supply n Software-Programmable Input Ranges:
  - 0V to 5V, 0V to 10V, ±5V or ±10V
  - Single-Ended or Differential
- ±3LSB INL for the LTC1859, ±1.5LSB INL for the LTC1858, ±1LSB INL for the LTC1857 n Power Dissipation: 40mW (Typ) n SPI/MICROWIRE™ Compatible Serial I/O n Power Shutdown: Nap and Sleep n Signal-to-Noise Ratio: 87dB (Typ) for the LTC1859 n Operates with Internal or External Reference n Internal Synchronized Clock n 28-Pin SSOP Package
- ±3LSB INL for the LTC1859, ±1.5LSB INL for the LTC1858, ±1LSB INL for the LTC1857
- Power Dissipation: 40mW (Typ)
- SPI/MICROWIRE™ Compatible Serial I/O
- Power Shutdown: Nap and Sleep
- Signal-to-Noise Ratio: 87dB (Typ) for the LTC1859
- Operates with Internal or External Reference
- Internal Synchronized Clock
- 28-Pin SSOP Package

1.1.9 Mini40 Schunk

Figure 1.13: Force Torque Sensor

Force/torque sensor FT, with water protection. [19]
Features:

- Weight [kg] 0.049
- Calibration SI-20-1
- Measuring range Fx, Fy [N] ± 20
- Measuring range Fz [N] ± 60
- Measuring range Mx, My [Nm] ± 1
- Measuring range Mz [Nm] ± 1
- Overload Fx, Fy [N] ± 810
- Overload Fz [N] ± 2400
- Overload Mx, My [Nm] ± 19
- Overload Mz [Nm] ± 20
- Resonant Frequency Fx, Fy, Mz [Hz] 3200
- Resonant Frequency Fz, Mx, My [Hz] 4900
- Resolution Fx, Fy [N] 0.005
- Resolution Fz [N] 0.01
- Resolution Mx, My 0.001
- Resolution Mz 0.001
- Diameter D [mm] 40
- Height Z [mm] 14

1.1.10 Dynamixe XM430-W350

Figure 1.14: Dynamixel Motor XM430-W350

The dynamixel motor XM430-W350 T200 is a motor created by ROBOTIS. It's is able to work in a water enviroments. DYNAMIXEL XW series are ROBOTIS' first line of IP Level models, featuring a certified IP68 (1m, 24hr) rating and are designed for use in wet environments, underwater, and any outdoor applications where a sealed servo is necessary. XW series come with separate waterproof cable and extension cable. [9]
The MX motor is composed by an ARM Cortex M3(72 MHz ,32 Bit), an contactless absolute encoder with a resolution of 4096 (pulse/rev) and a coreless motor. It is possible to change the motor baudrate from 9,600 (bps) to 4,5 (Mbps), and the control algorithm implemented is a PID, and permit to have 6 operating modes, like :

- Current Control Mode
- Velocity Control Mode
- Position Control Mode (0 $\tilde{3}$60 [°])
- Extended Position Control Mode (Multi-turn)
- Current-based Position Control Mode
- PWM Control Mode (Voltage Control Mode)

The operating temperature is from -5 to +80 °C, and is recommended to get as input voltage 12V, but can be in the range [10.0 - 14.8] [V].
To connect the single motor or a sequence of motor, are present 3 type of physical connection:

- RS485 / TTL Multidrop Bus
- TTL Half Duplex Asynchronous Serial Communication with 8bit, 1stop, No Parity
- RS485 Asynchronous Serial Communication with 8bit, 1stop, No Parity

1.2 Software

1.2.1 ROS

Figure 1.15: Ros Logo

ROS is a Linux-based framework for modular use in robotic applications. It was originally designed by Willow Garage and currently maintained by Open Source Robotics Foundation. This is a powerful tool because uses Object-Oriented Programming (OOP), which is a method of programming that organizes software design around data or objects, rather than functions and logic.

As said above, it works exactly as a regular Operating System present on a personal computer. The difference is: instead of applications ROS programs allow a user to control the mobile operations of a robot. ROS simplifies the task of creating complex and robust robot behavior due to its collection of tools, libraries and conventions [17].

Software in ROS is distributed in packages and stacks of packages. A package is a directory with a specific structure, which can contain libraries, executables, configuration files and so on. To better understand this structure, main concepts should be presented, such as nodes, messages, topics and bags.

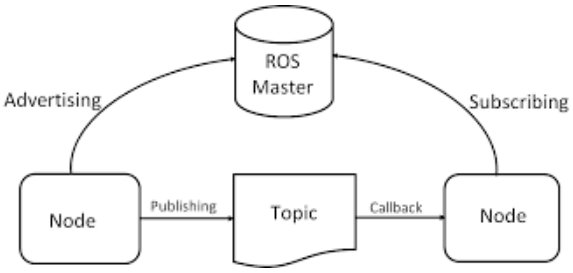

Figure 1.16: Ros Structure

Nodes

A node is basically a program (process) being executed. As defined by ROS wiki, "node is a process that performs computation". It is an executable program running inside the application. Nodes are combined into a graph and communicate with each other using ROS messages through topics and services.

Messages

ROS uses a simplified message description language for describing the data values published by a ROS node. Messages are exactly those data values. Messages descriptions are stored in files located on a sub directory of a ROS package.

Topics

The concept of topic is very simple: topics are named buses over which nodes exchange messages. In other words, topics in ROS enable the data to be exchanged between nodes. A topic can have publishers and/or subscribers, where the first send messages trough the topic and the last one receives those messages.

Bags

A bag is a file format in ROS for storing ROS message data. They offer a convenient way to store all kinds of data during an execution. The main analysis are made using ROS bags stored data.

1.2.2 QGroudControl

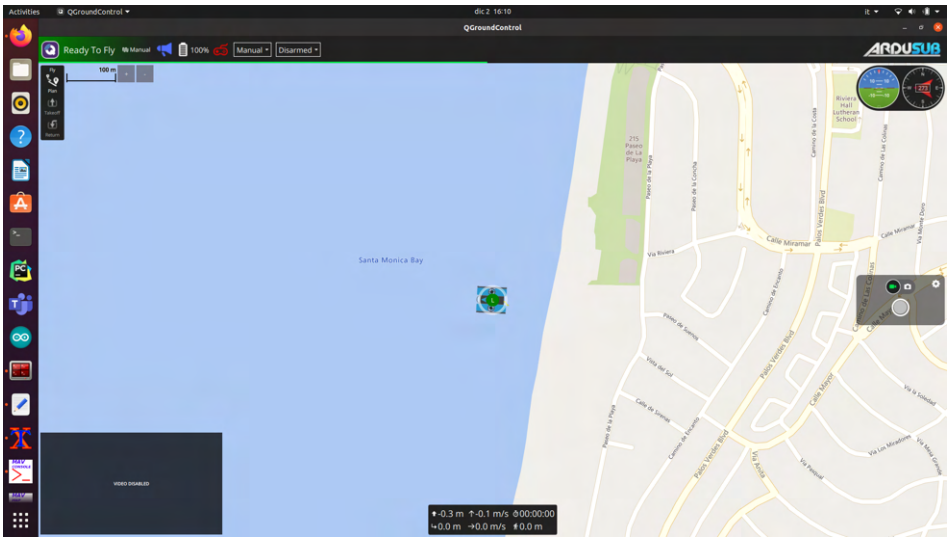

Figure 1.17: QGroundControl Interface

QGroundControl (QGC) is the Graphical User Interface (GUI) for ArduSub, which provides setup and control functionality. QGC runs on Windows, OS X, and Linux platforms.

Figure 1.18: QGroundControl Logo

Key Features:

- Full setup/configuration of ArduSub powered vehicles.
- Video streaming with instrument display overlays.
- Map display showing vehicle position, track, and waypoints for vehicles using an underwater positioning system.

The QGC user interface is implemented using Qt QML. QML provides for hardware acceleration which is a key feature on lower powered devices such as tablets or phones. QML also provides features which allows creation of a single user interface which can adapt itself to differing screen sizes and resolution.

1.2.3 ArduSub

Figure 1.19: ArduSub Structure

The ArduSub firmware is a binary file that is loaded on to the internal memory of a compatible autopilot board. The firmware contains the necessary logic processes to control the specified vehicle type, Sub in this case. Some autopilots (eg. Pixhawk) have an SD card that is used to store data logs.
No software programming is required to operate ArduSub, settings are changed through user configurable parameters when connected to QGroundControl.
The ArduPilot project supports multiple vehicle types, and each one has it's own firmware file. You must make sure that you have programmed the autopilot with the *ArduSub* firmware to use ArduSub.[3]

1.2.4 MAVLink

Figure 1.20: Mavlink Logo

MAVLink is a very lightweight messaging protocol for communicating with drones (and between onboard drone components).
MAVLink follows a modern hybrid publish-subscribe and point-to-point design pattern: Data streams are sent / published as topics while configuration sub-protocols such as the mission protocol or parameter protocol are point-to-point with retransmission.
Messages are defined within XML files. Each XML file defines the message set supported by a particular MAVLink system. The reference message set that is implemented by *most* ground control stations and autopilots is defined in common.xml.
Code generators create software libraries for specific programming languages from these XML message definitions, which can then be used by drones, ground control stations, and other MAVLink systems to communicate. The generated libraries are typically MIT-licensed, and can therefore be *used* without limits in any closed-source application without publishing the source code of the closed-source application.[7]

1.2.5 PyMAVLink

Pymavlink is a *low level* and *general purpose* MAVLink message processing library, written in Python. It has been used to implement MAVLink communications in many types of MAVLink systems, including a GCS (MAVProxy), Developer APIs (DroneKit) and numerous companion computer MAVLink applications.
The library can be used with Python 3.5+ and supports both MAVLink 1 and MAVLink 2 versions of the protocol. This topic explains how to get and use the *Pymavlink* MAVLink Python libraries (generated using mavgen). [13]

1.2.6 MAVProxy

MAVProxy is a fully-functioning GCS for UAV's, designed as a minimalist, portable and extendable GCS for any autonomous system supporting the MAVLink protocol (such as one using ArduPilot). MAVProxy is a powerful command-line based "developer" ground station software.
It can be extended via add-on modules, or complemented with another ground station, such as Mission Planner, APM Planner 2, QGroundControl etc, to provide a graphical user interface. It has a number of key features, including the ability to forward the messages from your UAV over the network via UDP to multiple other ground station software on other devices. [8]

MAVProxy is commonly used by developers (especially with SITL) for testing new builds.

MAVProxy was first developed by CanberraUAV, to enable the use of companion computing and multiple datalinks with ArduPilot. It has grown to be one of the most versatile tools in the ArduPilot ecosystem, and many of the features users now see in other GCS tools can trace their origins to MAVProxy.

1.2.7 Qualisys Motion Capture

Figure 1.21: Qualisys Logo

The main software of the Qualisys motion capture system is called QTM - Qualisys Track Manager. This software allows the user to control, calibrate and edit measurements. The software gives the user visual feedback of what is happening in the system in real time. The different types of data that can be accessed from the cameras with QTM are

- Marker Mode - displays white dots representing markers.
- Intensity mode - see what is interpreted as markers.
- Video mode - greyscale video.

Each mode is a 2D representation of what each camera sees. QTM uses this 2D information to calculate a 3D representation of the volume. From this it can also recognise pre-defined rigid bodies called 6DOF. 6DOF bodies are a set of markers that are attached to the same object and therefore retain their relative position as the body moves. In addition to marker data, QTM can also record analogue input from devices such as pressure plates1 , audio or video recordings that can be matched to the measurements.[15]
The RT protocol (Real-Time Protocol) is part of QTM. The protocol allows network based communication with the Qualisys motion capture system via TCP/IP or UDP/IP. Various commands can be used to set various system settings and to stream data in real time. Examples of different types of information that can be streamed in real time are 2D markers, 3D markers, video, marker intensity and 6DOF (rigid bodies).

1.2.8 Arduino IDE

The Arduino IDE is an open source software used to write and upload code to the Arduino boards. The IDE application is suitable for various operating systems such as Windows, Mac OS X and Linux. It supports the C and C++ programming languages. IDE stands for Integrated Development Environment.[2]
The program or code written in the Arduino IDE is often referred to as a sketch. We need to connect the Genuino and Arduino board to the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino'.

1.2.9 VirtualHere

Figure 1.22: VirtualHere Logo

VirtualHere allows USB devices to be used remotely over a network just as if they were locally connected.[21]
The VirtualHere software is commercial quality, used by tens of thousands of customers, and multiple OEMs and is *extremely easy to setup and use*. It has a powerful API so device interaction can be scripted, as well as an intuitive user interface. There are two parts:

- *The VirtualHere USB Server* software runs on the Synology, QNAP, ASUSTOR, ReadyNAS and MyCloud NAS Devices, as well as Raspberry Pi, AMD64, x86, MIPS, ARM, PowerPC, ARM64 etc running ANY version of Linux, and now runs on OSX, Android and Windows AND WSL2.

- *The VirtualHere USB Client software* requires minimal installation and runs on each end user machine or cloud instance, for Windows, OSX and Linux.

1.3 Programming Languages

1.3.1 Python

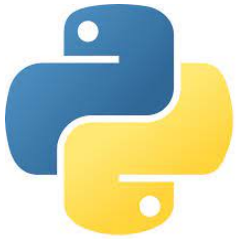

Figure 1.23: Python Logo

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. [14]

1.3.2 C++

Figure 1.24: C++ Logo

C++ is a high-level, general-purpose programming language created by Danish computer scientist Bjarne Stroustrup. First released in 1985 as an extension of the C programming language, it has since expanded significantly over time; as of 1997 C++ has object-oriented, generic, and functional features, in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Embarcadero, Oracle, and IBM.[5]
C++ was designed with systems programming and embedded, resource-constrained software and large systems in mind, with performance, efficiency, and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including

desktop applications, video games,servers e-commerce, web search, or databases), and performance-critical applications (e.g. telephone switches or space probes).

Chapter 2

BlueROV2 and Antenna System

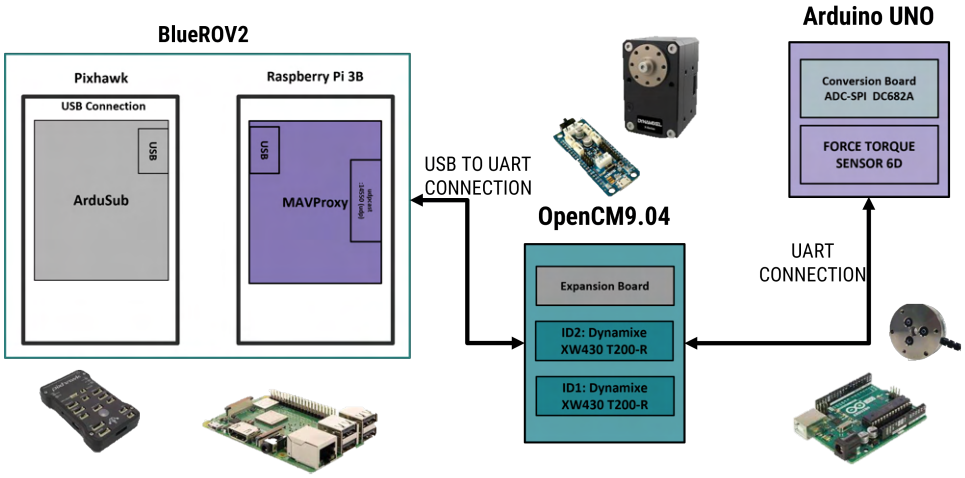

Figure 2.1: Haptic System Scheme

The system description starts with the implementation of the antenna on the existing BlueROV2 hardware. Therefore, the components already present on the ROV and in particular the communication and data transmission system have been taken into account in the development of this system. It should be noted that the system will be operating in a different environment, i.e. in a swimming pool or at sea.
The presence of an existing system severely limited the development and addition of another component to the ROV, as the main board does not have a great capacity to manage resources, and the implementation time also influenced the development and design choices.
After a brief introduction, we move on to the explanation of the system shown in the figure below. The main system to be implemented was the addition of 3 basic components that characterise the haptic system. This system consists of two Dynamixel motors, a force-torque sensor and the flexible antenna placed on the sensor. The components of this system are waterproof and were therefore placed on the front of the Rov. All the wiring, on the other hand, was placed in special hermetically sealed containers inside and on the underside of the robot.

To connect the whole system to the ROV, a single board had to be used to manage all the components such as motors and sensors. The choice of a single board was due to

the limited number of USB ports available on the Raspberry Pi3+, as the other ports are occupied by the Pixhawk 1 and the Fathom X tether for connection to the ground control station.
As a result, the main haptic system board is the Open CM 904 with its expansion board to manage the two motors and the FT sensor.
The whole system is placed one tube below the ROV, allowing us to place all the system electronics without modifying the existing ROV.

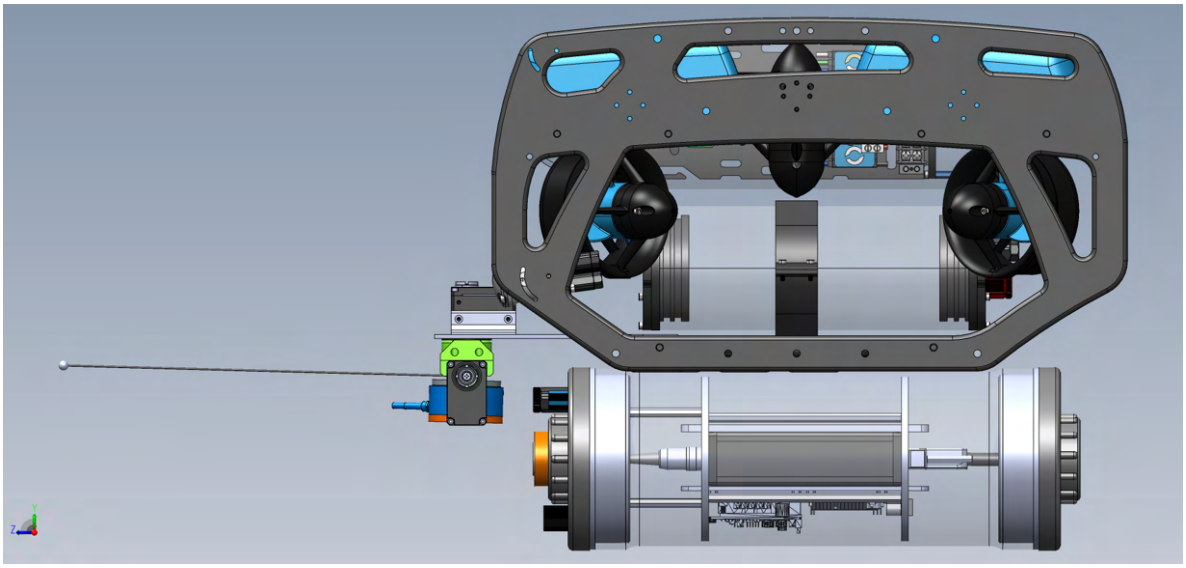

Figure 2.2: Lateral Prospective of CAD Model

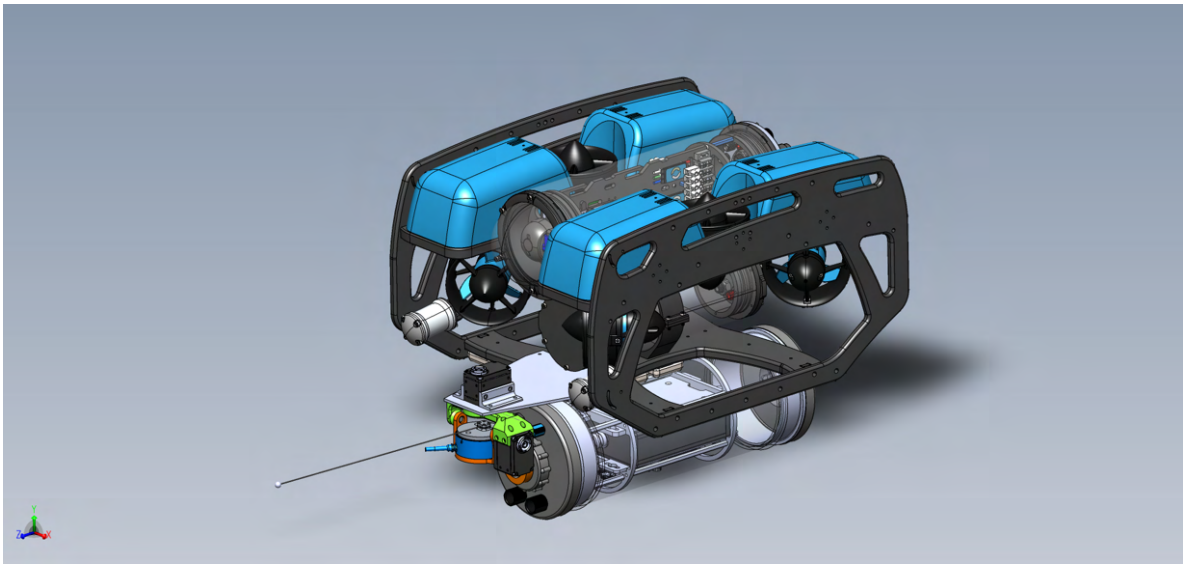

Figure 2.3: CAD Model of Rov + Antenna System

2.1 Hardware

Going into the details of the implementation for the main board is connected to the RPi3B+ via USB protocol, whereas the Arduino board, which serves mainly as a bridge

to obtain data from the sensor, more precisely data from the conversion board connected to the force-torque sensor, is connected directly to the opencm 904 via UART/USART protocol. As can be seen from the following figure.

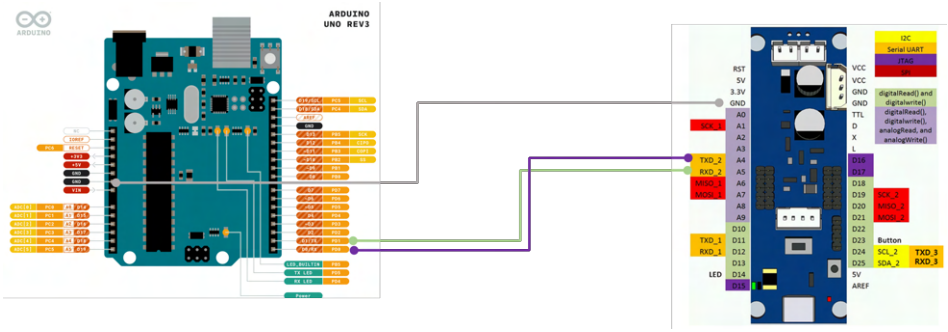

Figure 2.4: Arduino Connection with OpenCM9.04

Instead, the sensor is connected to Arduino via a conversion board, which converts the analogue signal coming in from the sensor into a digital signal. This digital signal is read by the Arduino via the SPI serial communication protocol. Below we see how the protocol is structured.[20]

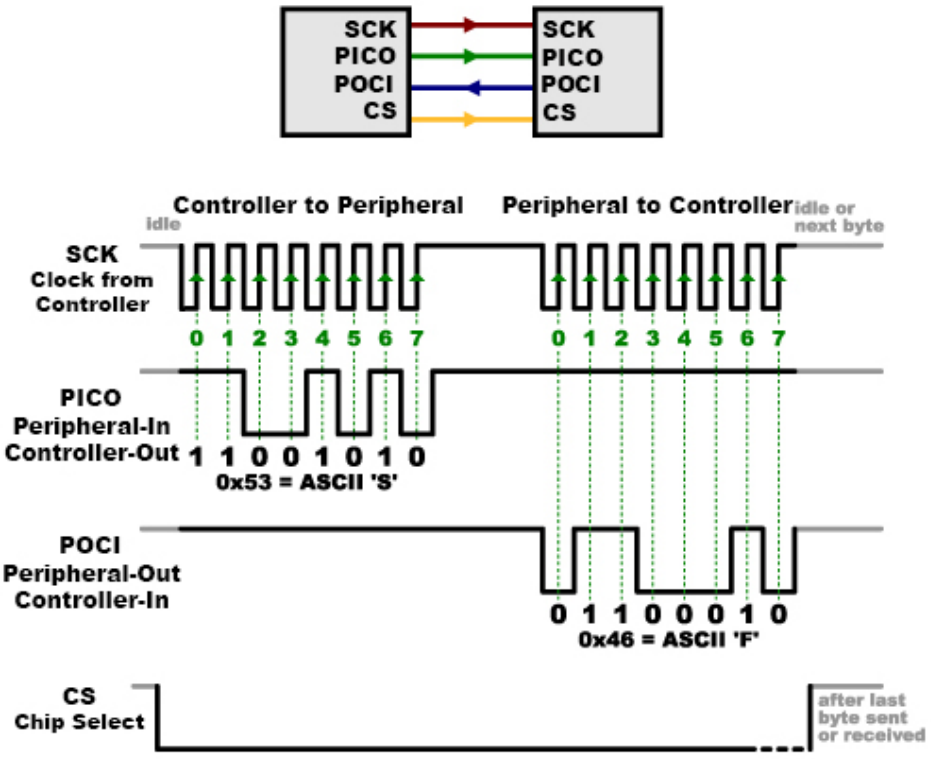

Figure 2.5: SPI Protocol

Instead, the following figure shows the connection between sensor-converter and converter-Arduino.

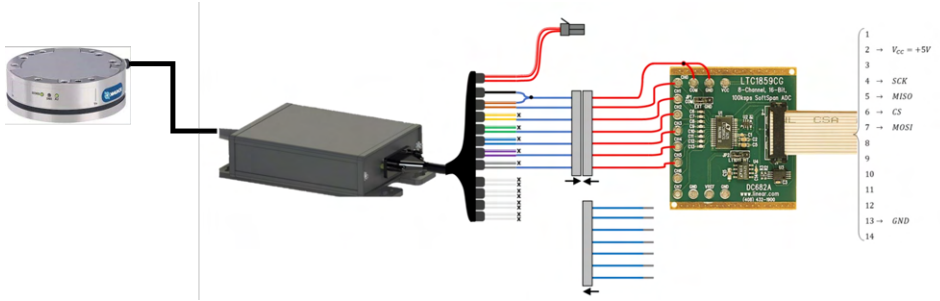

Figure 2.6: DAQ Sensor Connected to ADC Converter

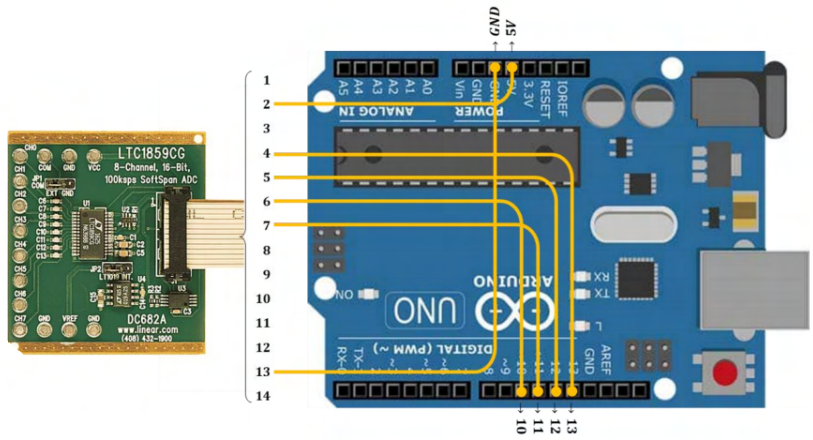

Figure 2.7: ADC Converter Connected to

2.2 Firmware

The firmware for both boards was written in C/C++.

2.2.1 Arduino

The purpose of the Arduino code is to manage the data read by the converter via the SPI communication protocol and to send it via the UART protocol to the OpenCM9.04 board. This firmware is intended to be much faster, and therefore to always have the data available during the request, otherwise there would be problems during data input/output and its management would not be optimal.
In order for both boards to be synchronised with the exchange of data, we have structured the sending of data only in the event of a request from the opencm board. In the event of a non-request, the Arduino continues to process the data read from the sensor and during the data request only the last data read will be returned. This reduces the latency of the communication and therefore there is no delay from reading the data at the time of the request. In addition, has been implemented a low pass filter, to reduce the noise at high frequency. The filter can be active or disactive by a flag in the code, the activation does not impact the data transmission. Below there the flow chart of Arduino firmware.

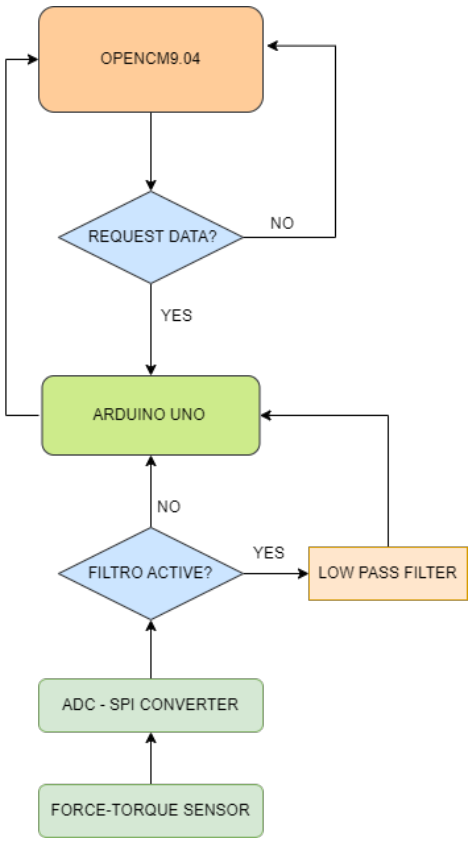

Figure 2.8: Flowchart Arduino Code

2.2.2 OpenCM9.04

The firmware of the opencm board, is the one that handles both the torque force sensor data and input output data in the connection with the two Dynamixel motors via the R459 bus. The objective of the board is to read the sensor data then position of the two motors, the 37-character string from Arduino, and send it via rosserial to the rosmaster, it will also have to receive the position reference to send to the two motors.
Using a single device to manage all the input/output components can lead to poor data management if the firmware is not written well, as there can be delays when accessing the various connected devices, which can compromise the logic of the firmware. To avoid such problems and to ensure the reception and sending of data, the code structure was represented by means of a flowchart.

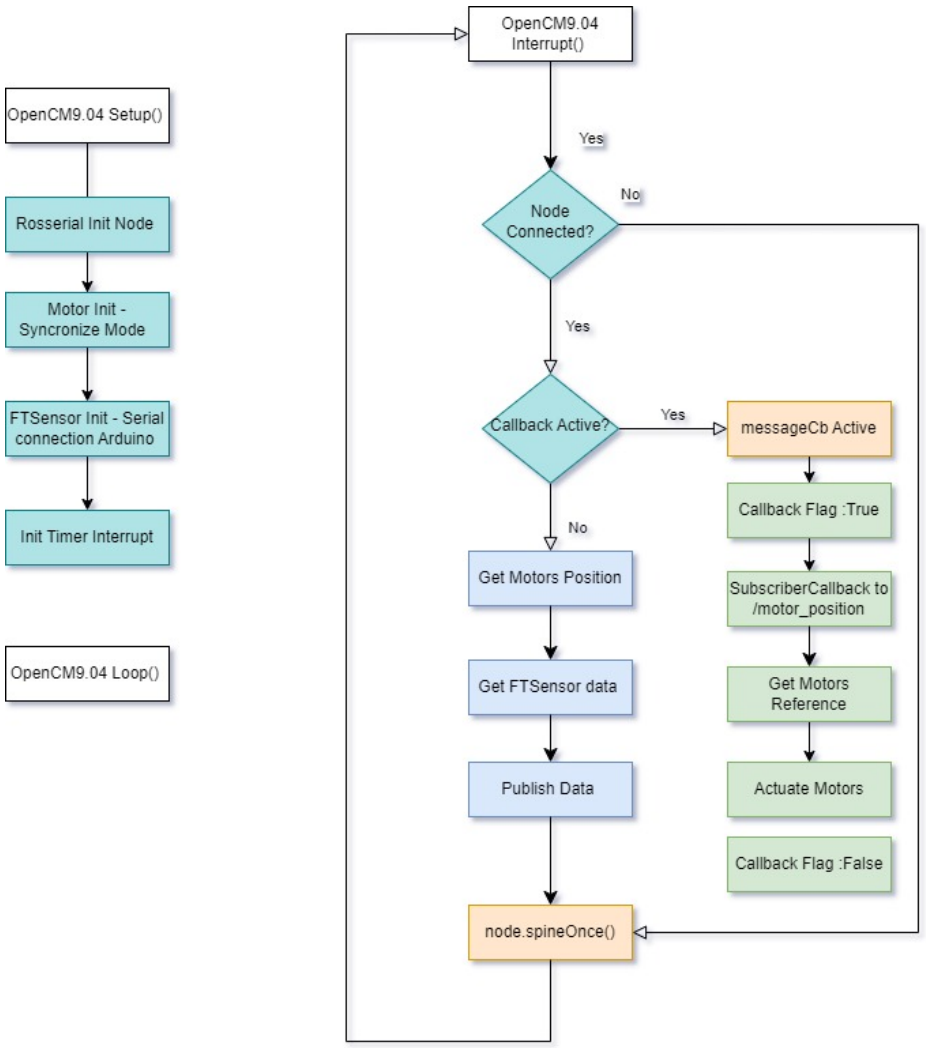

Figure 2.9: Flowchart OpenCM Code

Finally, we obtain a ros node connected to the master ros, this node then sends the motor position data via the encoders, and the force sensor channel string. The frequency at which the data is sent is around 77hz, increasing the rate at which the data is sent causes synchronisation problems with the device it is connected to, as the firmware needs time to acquire and invoke the data, so it is much slower than the ground station. Therefore, if we send data at a higher frequency, the data will be stored in the ros buffer, but if this buffer fills up, we will lose synchronisation and have to reboot the device.
The board gets the data from the topic '/motor_position' and sends the sensor and motor data on a single string to the topic '/data_opencm'. Finally, we can see that the frequency at which data is sent is not constant as there are latencies given by the code that we cannot consider as precise latencies.

2.3 Qualisys Motion Capture System

A motion capture system was used to obtain the position, speed and acceleration of the ROV in the pool. This system keeps track of the markers on the object to be identified

and tracked and monitors its position in real time, thanks to four cameras and the qualisys software that allows us to record and display all the data.
The problem we encountered with this system was with the development environment of this software, as it could not be used in a linux/ubuntu environment. The ubuntu environment would have allowed us to use ros and publish all the data in that environment and not use other types of data transmission that were much slower.
The solution to this problem was the use of a virtual machine containing Ubuntu 20.04, through Qualisys Realtime SDK we can obtain the software data on ubuntu and then publish it on the ros system. [16]

Figure 2.10: Qualisys Camera

Figure 2.11: Qualisys System Capture

In addition, it was planned to place a structure on the Rov in order to keep track with the motion capture system. This structure was designed considering the height of the water in the pool and a depth value at which the rov will perform the various tasks. The structure is shown in the figure.

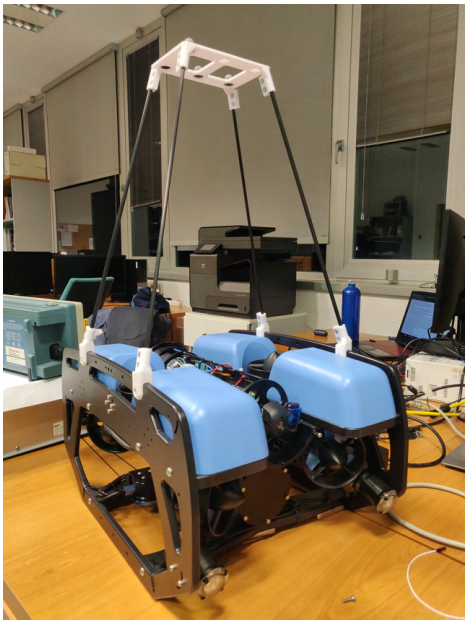

Figure 2.12: Capture Structure for BlueROV2

Chapter 3

ROS System

A ROS Network had to be used to interconnect all devices such as the Ground Control Station, the Rov, the haptic system and the motion capture system. Through the ROS environment, it was possible to communicate with any node in the network at any time. The requirements to set up such a network are:

- Complete two-way connectivity between all pairs of machines, on all ports.
- Each machine must publish itself with a name that all other machines can solve.

To establish full connectivity, it was necessary to create a local network, in which all devices are connected to an etherenet switch. Each device is assigned a static IP to remain stable over time. Therefore, the devices in the network are configured as follows:

- Ground Control Station : IP 192.168.2.1
- Qualisys Motion Capture : IP 192.168.2.3
- BlueROV2 Fhatom X : IP 192.168.2.2

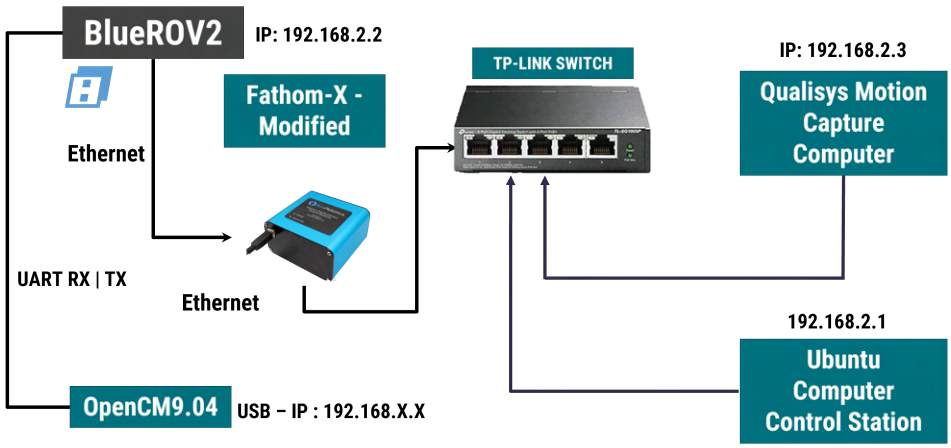

Figure 3.1: System Local Network

After configuring the Ros Network, we must define the Ros Master. For simplicity and efficiency, we use the Control Station as the master node, as it allows us to process data faster and have control over the entire network.

Let us now move on to the detailed explanation of the connection between the various devices. As a first step, it was essential to understand how to add the haptic system to the rov system.
The haptic system consisting of two Dynamixel motors and the torque force sensor requires two boards for acquiring and controlling the motors. As previously explained in the previous sections, where we went into the details of the components, we saw how the force-torque sensor data requires an adc converter and the arduino board to process the data. To obtain this data on the main board, OpenCM904, it was necessary to set up a connection via the uart protocol. The data sent is passed via strings containing the data of the six sensor channels. The data obtained in the following channels are related to the voltage variation, so they need to be converted to obtain the forces F_x, F_y, F_z and moments M_x, M_y, M_z that the antenna applies on the sensor.
Then after acquiring the sensor data from the arduino, we send this data via opencm on a ros topic, this data will then be processed to obtain the exact values of the applied forces and moments. The processing of the data is carried out on the ros side, using the python programming language and the numpy library, which provides us with optimised algorithms to perform matrix multiplication in a very short time. The force and moment values are obtained by means of the following calibration matrix (4.1), and the first few samples obtained in the channels are subtracted from the data in order to reset the offset error to zero and obtain more meaningful data.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} -0.1 & 0 & 0 & 3.1 & 0.2 & -3.4 \\ 0 & -3.7 & -0.2 & 1.8 & 0 & 0 \\ 5.667 & 0 & 6 & 0 & 6 & 0 \\ 0 & 0 & 0.073 & 0 & -0.0758 & 0 \\ -0.080 & 0 & 0.0436 & 0 & 0.0436 & 0 \\ 0 & -0.04367 & 0 & -0.0436 & 0.0022 & -0.043 \end{bmatrix} \begin{bmatrix} CH1 \\ CH2 \\ CH3 \\ CH4 \\ CH5 \\ CH6 \end{bmatrix} \tag{3.1}$$

For the implementation of the two motors on the main board, the RS485 protocol was used. RS485 is a standard that defines the electrical characteristics of the drivers and receivers for the communication protocol. It is transmitted over two signal lines, A and B, which must be balanced and differential.
Balanced signals are two lines sharing a pair in a twisted pair cable with the same impedance along each line, and matched impedance at the receiver and transmitter. Using a twisted pair cable provides noise immunity. Because RS-485 is a balanced transmission standard, there are two pins for each transmission and each reception a positive and a negative.

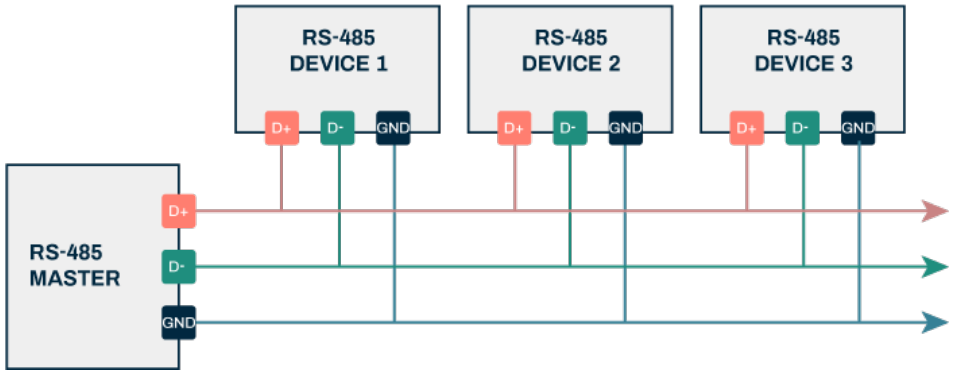

Figure 3.2: RS-485 Protocol

This protocol allowed us to control and acquire data in a secure and efficient manner. Therefore, after programming the main board to send data via serial communication to the Raspberry, it was necessary to find a way of accessing this device connected to the Rasperry via Ethernet, since, as seen in the explanation of BlueRov2, all the hardware of the rov connects to the control station via the udp protocol. The VirtualHere software was used to communicate with the haptic system.
This software allows us to assign an address to a device connected via a serial port to a device on the network; by means of the address, the program allows us to access the device connected in serial and thus use rosserial from the control station.

Let us now move on to the interconnection of all the devices. The main issue was to get position feedback from the capture motion system, which is not usable for linux systems, so it was necessary to set up a tall windows-based computer and interconnect all devices on a single network.
A switcher was then used to connect all the devices to a single local network, creating a ros network.

Figure 3.3: TP- LINK Switcher

To connect the Fhantom X Topside to the BlueROV2 was modified by removing the usb connectivity and exploiting ethernet connectivity, so that rov data can be accessed from multiple devices on the network.

Figure 3.4: Modified Fathom X Topside

The interconnection of the entire system is shown in the figure:

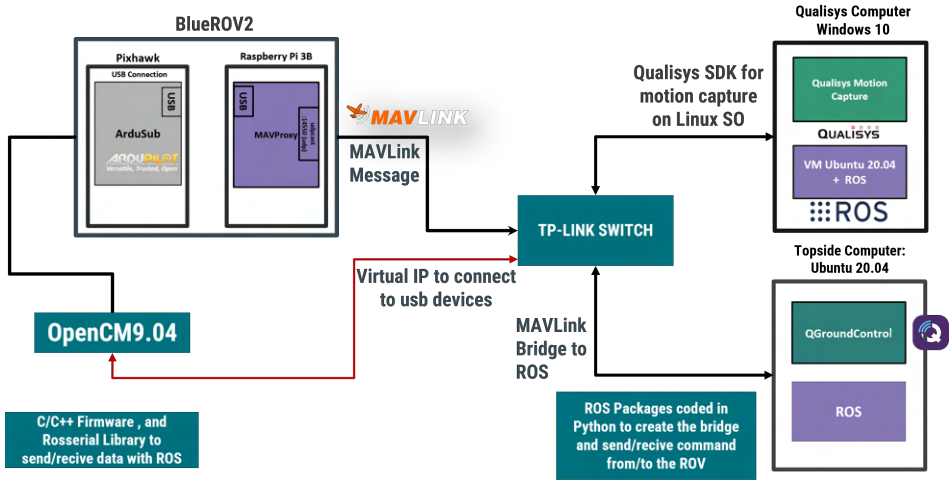

Figure 3.5: System Interconnection

3.1 Ros Implementation

3.1.1 Mavlink Bridge to ROS

The first approach to creating the entire ROS system was to implement the ROV control, i.e. to develop a bridge between the BlueROV2 messaging protocol, i.e. via Mavlink Message, and the ROS system at the ground station. To create this bridge, the Pymavlink library[13] provided by Mavlink, to communicate via scripts in the Python programming language.

```
    def decode_mode(self, base_mode, custom_mode):
        """ Decode mode from heartbeat
            http://mavlink.org/messages/common#heartbeat

        Args:
            base_mode (TYPE): System mode bitfield, see MAV_MODE_FLAG ENUM in mavlink/include/mavlink_types.h
            custom_mode (TYPE): A bitfield for use for autopilot-specific flags.

        Returns:
            [str, bool]: Type mode string, arm state
        """
        flight_mode = ""

        mode_list = [
            [mavutil.mavlink.MAV_MODE_FLAG_MANUAL_INPUT_ENABLED, 'MANUAL'],
            [mavutil.mavlink.MAV_MODE_FLAG_STABILIZE_ENABLED, 'STABILIZE'],
            [mavutil.mavlink.MAV_MODE_FLAG_GUIDED_ENABLED, 'GUIDED'],
            [mavutil.mavlink.MAV_MODE_FLAG_AUTO_ENABLED, 'AUTO'],
            [mavutil.mavlink.MAV_MODE_FLAG_TEST_ENABLED, 'TEST']
        ]

```

```
        if base_mode == 0:
            flight_mode = "PreFlight"
        elif base_mode & mavutil.mavlink.
    MAV_MODE_FLAG_CUSTOM_MODE_ENABLED:
            flight_mode = mavutil.mode_mapping_sub[custom_mode]
        else:
            for mode_value, mode_name in mode_list:
                if base_mode & mode_value:
                    flight_mode = mode_name

        arm = bool(base_mode & mavutil.mavlink.
    MAV_MODE_FLAG_SAFETY_ARMED)

        return flight_mode, arm
```

Listing 3.1: Example Function of Mavlink Bridge

3.1.2 ROV Control

The program created to execute this message conversion allows us to connect to the Fathom X system using the UDP protocol, and also allows us to collect data from the sensors on the ROV and to control the 6 thrusters, the servomotors for the cameras and the lights on the front of the ROV.

Once this connection is in place, we can move on to developing the ROV environment. In this case, a node called BlueRov_node was created to handle the transmission of data from the bridge to the ROS system. It then publishes the ROV data, such as the estimated position given by the Ardusub system, the battery status, the operating status of the ROV (driving mode: Manual, Guided and Depth Hold) and at the same time acts as a link to control the motors. For this last implementation, the node subscribes to the topic concerning the "cmd_accel" control. It also receives the Rov's position data from the motion capture system.

```
    def _create_rov_state(self, topic):
        """
        Create ROV state message from ROV data

        Raises:
            Exception: No data available
        """

        # Check if data is available
        if 'SERVO_OUTPUT_RAW' not in self.get_data():
            raise Exception('no SERVO_OUTPUT_RAW data')

        if 'HEARTBEAT' not in self.get_data():
            raise Exception('no HEARTBEAT data')

        servo_output_raw_msg = self.get_data()['SERVO_OUTPUT_RAW']
        servo_output_raw = [servo_output_raw_msg['servo{}_raw'.format(
    i + 1)] for i in range(8)]
        motor_throttle = [servo_output_raw[i] - 1500 for i in range(6)
    ]
        # 1100 -> -1 and 1900 -> 1
```

```
        for throttle in motor_throttle:
            if throttle < 0:
                throttle = throttle / 400
            else:
                throttle = throttle / 500

        light_on = (servo_output_raw[6] - 1100) / 8
        # need to check
        camera_angle = servo_output_raw[7] - 1500

        # Create angle from pwm
        camera_angle = -45 * camera_angle / 400

        base_mode = self.get_data()['HEARTBEAT']['base_mode']
        custom_mode = self.get_data()['HEARTBEAT']['custom_mode']

        mode, arm = self.decode_mode(base_mode, custom_mode)
        data = State()
        data.arm = arm
        data.rc1 = motor_throttle[0]
        data.rc2 = motor_throttle[1]
        data.rc3 = motor_throttle[2]
        data.rc4 = motor_throttle[3]
        data.rc5 = motor_throttle[4]
        data.rc6 = motor_throttle[5]
        data.light = light_on
        data.camera = camera_angle
        data.mode = mode

        self.pub_topics[topic][3].publish(data)
```

Listing 3.2: Example Function of BlueROV2_Node

The control of the motors is not direct, the pwm of the 6 motors is not changed directly. In the design of the program, only the control in the directions of Surge, Sway, Heave and Yaw has been included, so according to these directions we impose a value ranging from 1100 to 1900. For example, if we want to make the rov move linearly, we will set a value greater than 1500 in the first control variable. The value you set is then processed inside the Companion Computer located inside the rov and will only move the motors that allow the rov to move linearly, for the other directions the operation is the same. The range of values set, between 1100 and 1900, allows us to control the direction of each direction. We consider 1500 as the value in which the motors do not move. The node that acts as a bridge between ros and Mavlink has been implemented a function to convert from input value to PWM for the motors.
Regarding the control of rov, understood as speed control, a 'velocity_control' node was designed that acquires position and speed data from the bluerov node and resupplies the control input to be sent to the system.

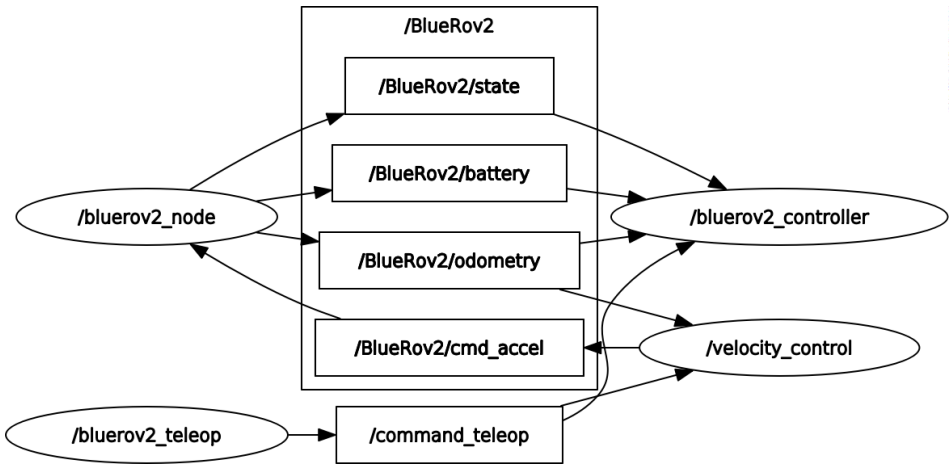

Figure 3.6: Rqt Graph BlueRov2 Ros Nodes

3.1.3 OpenCM904 Acquisition and Control

For the system consisting of the torque-force sensor and the two Dynamixel motors, a control via ROS was developed because the OpenCM board could use the ROS library, and it was also useful to simplify the connection with the ROS environment and to be able to control all the devices via a ground computer. As explained in the previous chapter on the board's firmware structure, it has a topic on which to publish a string containing data such as sensor data and the position of the two motors. And as far as the implementation is concerned, the board subscribes to the "motor_position" topic, on which the position references of the two motors are published by the ground computer.
On the ground-station side, the data received from the haptic system is processed and republished on other topics. As seen in the previous section, the sensor data must be processed and calibrated to obtain accurate values of the forces and torques, and the data obtained from the two motors must also be converted according to the encoder result.
So the ROS structure for controlling the haptic system is made up of the serial node, which handles the interface with the microcontroller, and then the opencm904_node, which interfaces with the serial node and allows us to control and acquire data from the board. Here is a piece of code for handling the data via Python and the structure of the connection between the nodes in ROS.

```
1     def data_sensor_conversion(self):
2
3         self.sensor_ft_data_raw = (self.sensor_ft_data_raw / 10000) -
      25
4         self.sensor_ft_data_conv = np.transpose(np.matmul(self.
      calib_mat, np.transpose(self.sensor_ft_data_raw)))
5
6         if self.offset_bool is True:
7             self.offset = self.sensor_ft_data_conv
8             self.sensor_ft_data_conv = np.subtract(self.
      sensor_ft_data_conv, self.offset)
9             print(self.offset)
10            self.offset_bool = False
11        else:
```

```
            self.sensor_ft_data_conv = np.subtract(self.
    sensor_ft_data_conv, self.offset)

def _data_sensor_callback(self, msg):
        if msg.data is not None:
            self.reset_status = 0
            self.data = msg.data  # String made of 36+/n char
            data_temp = self.data
            for i in range(0, 6):
                temp = data_temp[0:6]
                self.sensor_ft_data_raw[i] = float(temp)
                data_temp = data_temp[6:]

            self.motors_actual_position = [int(data_temp[0:4]) - 1000,
     int(data_temp[4:8]) - 1000]
            self.reset_status_board = int(data_temp[-1])
            self.data_sensor_conversion()
        elif msg.data is None:
            print("No MSG Available -- resetting the board")
            self.reset_status = 1
```

Listing 3.3: Example Function of OpenCM904_Node

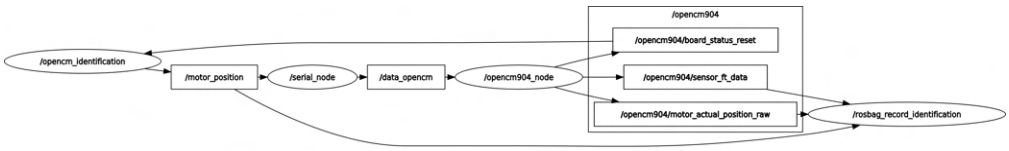

Figure 3.7: Rqt Graph OpenCm904 Ros Nodes

Chapter 4

Model

To analyse the entire mathematical model of the system, we start by analysing the various components and then combine them into a single model. We start by studying the flexible antenna and the system in a non-contact condition with a surface. Next we will consider contact, and move on to the study of the planar model of the rov, with and without the antenna. Finally, we are going to simplify the whole system to implement a control law.

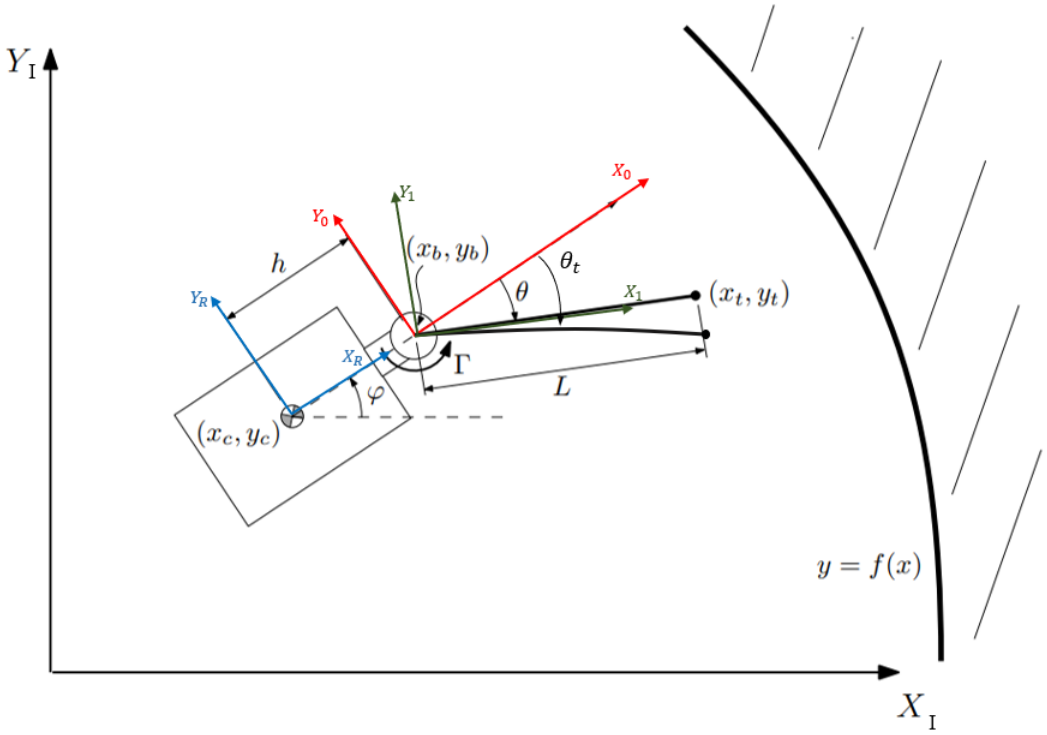

Figure 4.1: System without contact

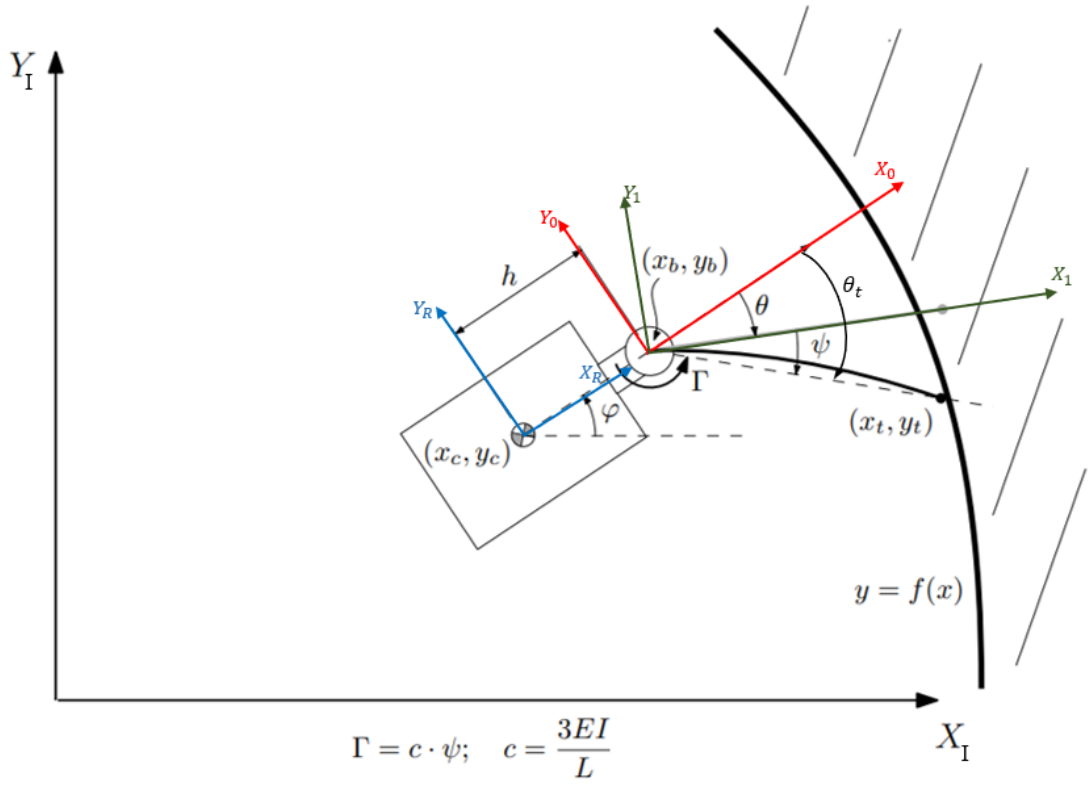

Figure 4.2: System with contact

4.1 Model Antenna Link

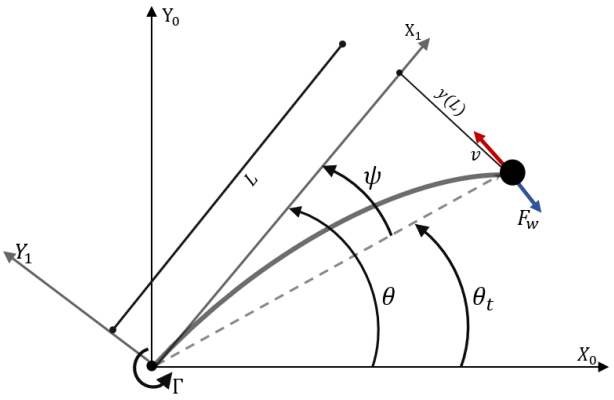

Figure 4.3: Link without contanct

Initially, we consider the rov and antenna in a non-contact condition with a surface. Here we consider the assumptions made by [25], with which we are able to simplify the dynamics of the flexible arm.
In the non-contact condition and especially in our case where we are considering a planar model in a water environment, then the only force we consider on the antenna

is the drag force given by the resistance of water on a moving surface. Instead, in case of contact the reaction force with the environment is considered.

The following assumptions are considered:

(1) All the mass concentrated at the tip position.

(2) The tip mass is considered to freely rotate, and therefore, no rotational inertia from it affects the link dynamics.

(3) There is only one collision point.

(4) The single-link flexible manipulator rotates in a horizontal plane (the z axis perpendicular to the plane of the figure).

(5) The deformations in the structure is very small, which allows to assume geometrical linearity, i.e., $sin(x) \approx x$, $tan(x) \approx x$.

(6) The structure oscillates with the fundamental mode of vibration without the higher modal densities being excited, since it is assumed that the mass of the load is much larger than the arm's weight.

(7) Neglecting the gravity force on the tip

(8) There are no axial deformations

(9) The stiffness parameter EI is constant throughout all of the beam

Two coordinate systems are established, both with origins coinciding with the rotational axis of the motor: the (X0,Y0) system is fixed in space, while the (X1,Y1) one moves with the motor shaft.
Thus, the y-coordinate of the point represents the deflection of the point on the beam from its initially straight configuration, corresponding to the x -axis.
The sum of forces represents the component of the resultant applied force that is normal simultaneously to the beam and to the joint rotation velocity vector: it represents the interaction of the beam with the environment. Assuming small deflections and mass less link, the beam dynamics can be described by Euler-Bernoulli equation for beam's deflection:

$$E \cdot I \cdot \frac{d^4 y}{dx^4} = 0 \tag{4.1}$$

Since the stiffness E · I is constant throughout the beam, the deflection is given by a third-order polynomial:

$$y(x) = u_0 + u_1 \cdot (x) + u_2 \cdot (x)^2 + u_{i,3} \cdot (x)^3 \tag{4.2}$$

where u_j are the polynomial coefficients.
The continuity condition between the motor and the beam imposes:

$$y_1\,(0) = \frac{dy_1}{dx}(0) = 0 \rightarrow u_0 = u_1 = 0 \tag{4.3}$$

4.2 Dynamic Equations

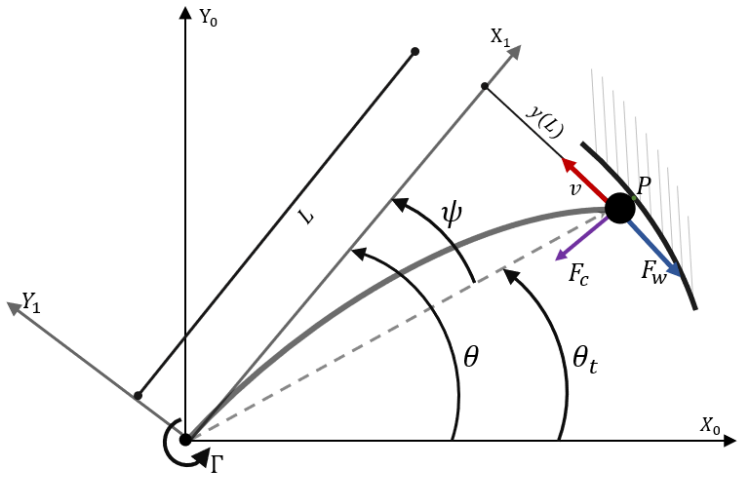

Figure 4.4

By studying forces and torques applied on the mass:

$$F(L) = -m \cdot (\frac{d^2y}{dx^2}) + F_w + F_{c,\parallel} = mL\ddot{\theta}_t - F_w - F_{c,\parallel} \tag{4.4}$$

$$F_w = \frac{1}{2}\rho C_{D,l} A_l v^2 = \frac{1}{2}\rho C_{D,l} A_l (L\dot{\theta}_t)^2 \tag{4.5}$$

where F_w is the drag force , $F_{c,\parallel}$ is the parallel component of contact force in P, ρ is the density of the fluid, A_l the reference area of the link, $C_{D,l}$ the drag coefficient referred to the link, v^2 the flow velocity respect to the tip of flexible link and m the tip mass. The tangential component of F_c is neglected, also for the drag force we consider $F^R_{w,\parallel} = F_w$ ($F^R_{w,\perp} = 0$). We could model the contact force as a spring-damper model, but we will violate the assumption number 8, where we do not consider all axial forces that cause deformation. Now, considering as boundary condition:

$$F(x) = -E \cdot I \cdot \frac{d^3y}{dx^3}(x) \tag{4.6}$$

$$T(x) = E \cdot I \cdot \frac{d^2y}{dx^2}(x) \tag{4.7}$$

we obtain the values of the remaining parameters by substituting the second and third derivative equation of polynomial (6.2) in (6.6)-(6.7), assuming $x = L$:

$$\frac{d^3y}{dx^3}(L) = 6u_3 \tag{4.8}$$

$$\frac{d^2y}{dx^2}(L) = 2u_2 + 6u_3 L \tag{4.9}$$

$$u_2 = \frac{F(L) \cdot L}{2 \cdot E \cdot I} \tag{4.10}$$

$$u_3 = -\frac{F(L)}{6 \cdot E \cdot I} \tag{4.11}$$

Once the polynomial coefficient values have been found, for small deflections we can assume:

$$y(L) \approx L \cdot (\theta_t - \theta) \tag{4.12}$$

in this way, by substituting (6.13) in the first equation (6.2), it is possible to obtain the tip dynamics:

$$L \cdot (\theta_t - \theta) = \frac{F(L) \cdot L}{2 \cdot E \cdot I} \cdot L^2 - \frac{F(L)}{6 \cdot E \cdot I} \cdot L^3 \tag{4.13}$$

$$L \cdot (\theta_t - \theta) = \frac{mL\ddot{\theta}_t - F_{c,\|} - \frac{1}{2}\rho C_{D,l} A_l (L\dot{\theta}_t)^2}{3EI} L^3 \tag{4.14}$$

$$\ddot{\theta}_t = \frac{1}{mL}\left(\frac{3EI}{L^2}(\theta - \theta_t) + F_{c,\|} + \frac{1}{2}\rho C_{D,l} A_l (L\dot{\theta}_t)^2\right) \tag{4.15}$$

$$\Gamma = \frac{3EI}{L}(\theta - \theta_t) \tag{4.16}$$

$$\psi = \theta - \theta_t \tag{4.17}$$

$$\ddot{\theta}_t = \frac{1}{mL^2}\left(\Gamma + F_{c,\|} L + \frac{1}{2}\rho C_{D,l} A_l (L\dot{\theta}_t)^2 L\right) \tag{4.18}$$

$$\ddot{\theta}_t = \frac{1}{mL^2}\left(\Gamma + F_c cos(\theta_c) L + \frac{1}{2}\rho C_{D,l} A_l (L\dot{\theta}_t)^2 L\right) \tag{4.19}$$

where Γ is bending moment at the base of the link and ψ the deflection angle.
The tip dynamic (Fig 6.3)without contact is :

$$F(L) = -m \cdot \left(\frac{d^2 y}{dx^2}\right) + F_w = mL\ddot{\theta}_t - F_w \tag{4.20}$$

$$\ddot{\theta}_t = \frac{1}{mL^2}\left(\Gamma + \frac{1}{2}\rho C_{D,l} A_l (L\dot{\theta}_t)^2 L\right) \tag{4.21}$$

In this case the contact force is not considered, so the deformation depends principally on the bending moment and drag force.

4.2.1 Motor Model

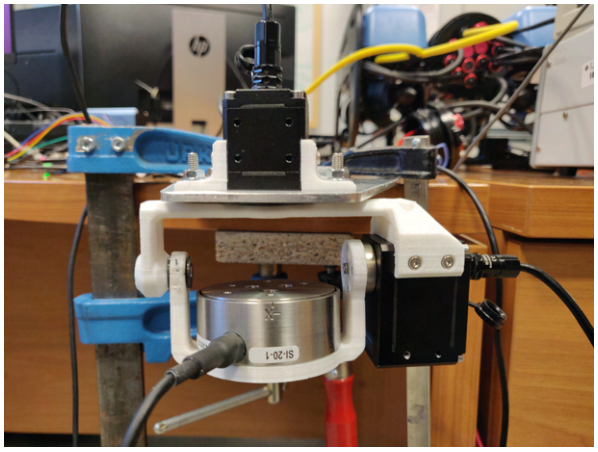

Figure 4.5: Motor + Sensor FT

As already described in the system architecture section, the motors used are two dynamixel wx. In this case, only the upper motor will have the function of rotating the antenna.
In order to model this system, it was essential to opt for a system identification. The motors already contain an internal control scheme, namely a position and speed control, but we are not provided with a transfer function to simulate such a system in Simulink / Matlab.
An APRBS (Amplitude-modulated Pseudo Random Binary Signal) input signal was created to proceed with the identification. This input signal will be used to identify the system and thus relate the input signal to the output signal.

- Input Signal : Position Reference for the motor
- Output Signal : Position of the motor, obtained by the absolute encoder

The main problem with this identification is the presence of saturation blocks within the system to be identified. A second problem is the identification of a closed-loop system such as the Dynamixel motor we are identifying. Within the closed loop, there are 2 control loops, 2 saturators and anti-wind-up gain.

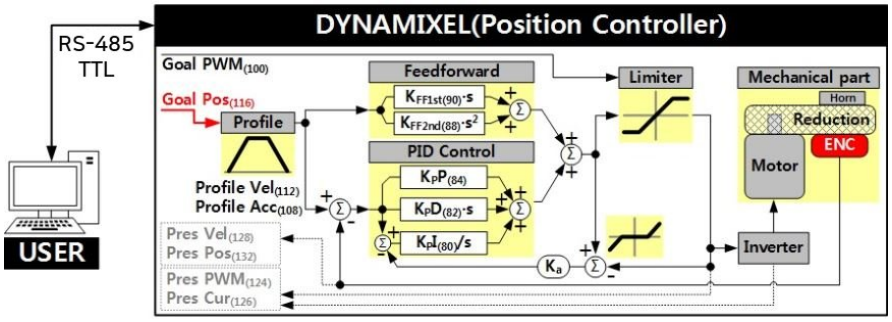

Figure 4.6: Position Controller Implemented into Dynamixel XW430-T200

We know about the position and speed controllers, but we do not know about the antiwindup gain.
As for the saturators, we are aware of the speed and current limits. So the first step was to design the input signal in such a way that saturation behaviour was not achieved. The variables for the identification are:

- max step angle : 180° - 185°
- min step angle : 180° - 180.5°
- max time to reach the reference : 0.25s
- min time to reach the reference : 0.15s
- frequency : 100 hz
- time of acquisition : 60s

The APRBS input signal obtained, considering these data, and the system's response to this input is shown in the figure.

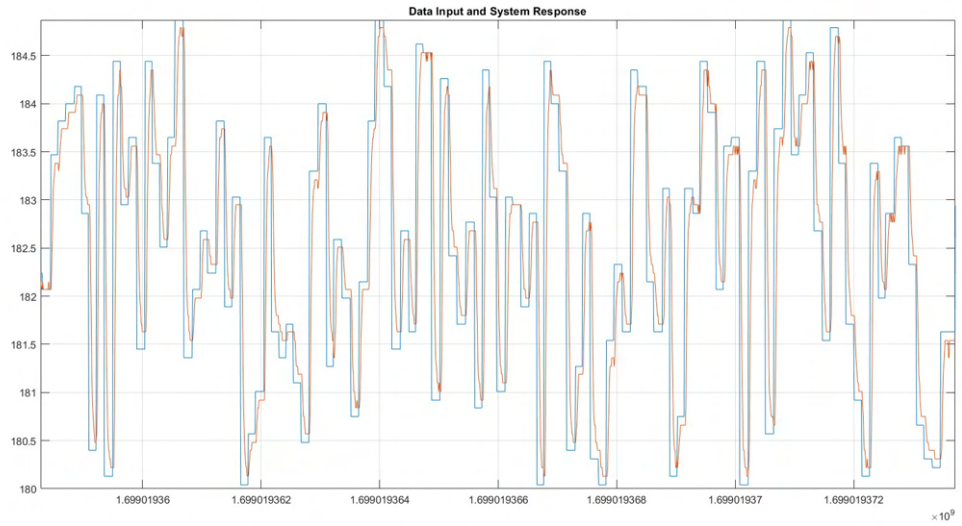

Figure 4.7: Data Input and System Response

After acquiring all the necessary data, they were divided into training data and validation data.
The subdivision allows us to compare the models obtained and thus understand whether the model succeeds in characterising the data or not, based on this we can opt whether to modify the model or change the model order.

One of the best models was the ARX (AutoRegressive eXogenous) Model with:

- $n_a = 2$;
- $n_b = 2$;
- $n_k = 1$;

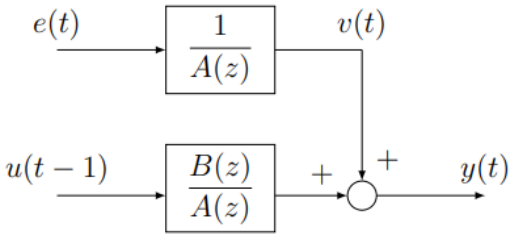

Figure 4.8: ARX Model

This type of models have two degree of freedom, they have an output that is the sum of three components:

- an auto regressive contribution on the output;
- a pure contribution on the variable error;

- a exogenous contribution in the moving average.

$$y(t) = \underbrace{a_1 \cdot y(t-1) + \dots a_{n_a} \cdot y(t-n_a)}_{AR} + e(t) + \underbrace{b_1 \cdot u(t-1) + \dots b_{n_b} \cdot u(t-n_b)}_{X}$$

For the propriety of temporal tralsation of Z-trasform, we have:

$$y(t) \circ \underbrace{\left[1 - a_1 z^{-1} - a_2 z^{-2} - \dots - a_{n_a} z^{-n_a}\right]}_{A(z)} = u(t-1) \circ \underbrace{\left[b_1 + b_2 z^{-1} + \dots + b_{n_b} z^{-n_b+1}\right]}_{B(z)} + e(t)$$

with:

$$A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - \dots - a_{n_a} z^{-n_a}$$
$$B(z) = b_1 + b_2 z^{-1} + b_2 z^{-2} + \dots + b_{n_b} z^{-n_b+1}$$

$$A(z) \circ y(t) = B(z) \circ u(t-1) + e(t)$$
$$\Downarrow$$
$$y(t) = \frac{B(z)}{A(z)} \circ u(t-1) + \frac{1}{A(z)} \circ e(t)$$

Where $G(z)$ and $W(z)$ must be asymptotically stables, therefore zeros of function $A(z)$ have to be in the circle of unit radius

The results obtained are as follows.

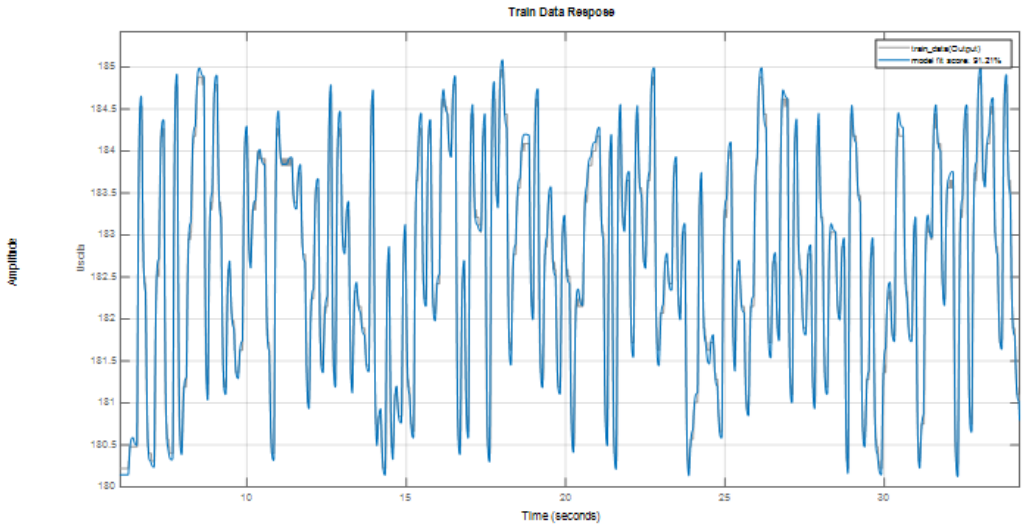

Figure 4.9: Training Data Response

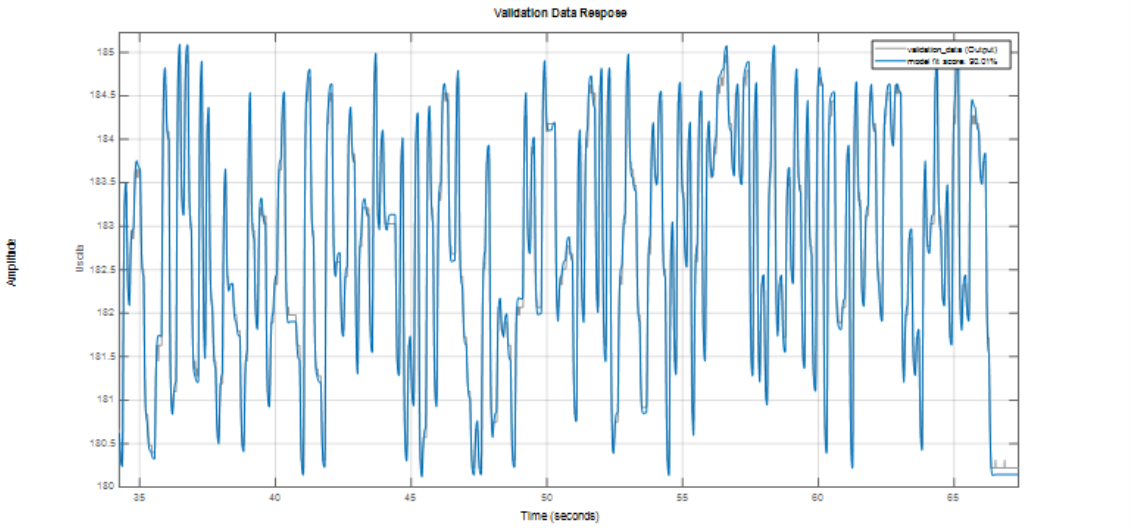

Figure 4.10: Validation Data Response

As we can see from the two figures, the results obtained from the model are 91.21% for the training data and 90.01% for the validation data. the values derived indicate the NRMSE (Normalized Root Mean Squared Error) , which is a fitness value indicator of how well the simulated or predicted model response matches the measurement data. This value is given by the function "$compare()$" in Matlab, and it calculate this value from this formula.

$$fit = 100(1 - \frac{||y - \widehat{y}||}{||y - mean(y)||}) \tag{4.22}$$

From the results obtained, it can be seen that the chosen model fits the system well. Despite the closed-loop system, we manage to obtain a transfer function that succeeds in describing the motor's behaviour. The transfer function obtained is as follows:

$$G = \frac{-8.13s + 1597}{s^2 + 68.11s + 1597} \tag{4.23}$$

From the motor documentation, we can obtain the position controller gains, in our case there are a proportional gain $Kp_p = 900/128$ and a integral gain $Kp_i = 10/65.536$. Knowing the controller gain, we can derive the transfer function of the speed control loop through the following steps.

$$G_v = \frac{G}{(1 - G) * (Kp_p + \frac{Kp_i}{s})} = \frac{-8.13s^4 + 1043s^3 + 9.579e04s^2 + 2.55e06s}{7.031s^5 + 1015s^4 + 4.776e04s^3 + 8.571e05s^2 + 1.858e04s} \tag{4.24}$$

Using Simulink, we verify the behaviour of the system obtained through simulation by also entering a saturator between $+2rad/s$ and $-2rad/s$.

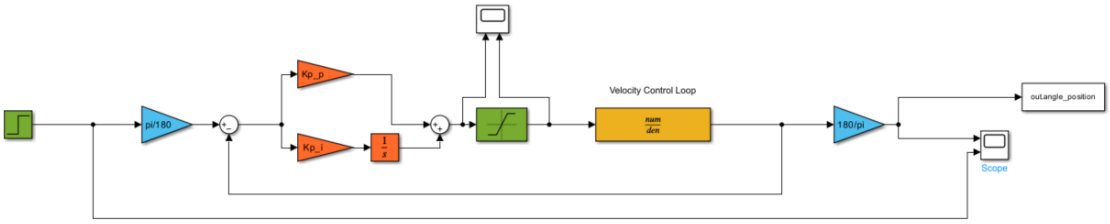

Figure 4.11: Simulink Scheme

Results

In order to compare the data actually obtained from the motor and the data obtained from the simulation, we compared the responses at variable steps. We can enter the data in the table referring to the settling time of the two systems.

Amplitude Step	Real System	Simulated System
1°	0.11 s	0.11 s
5°	0.145 s	0.145 s
10°	0.155 s	0.150 s
25°	0.205 s	0.195 s
45°	0.3 s	0.28 s
90°	0.47 s	0.435 s
180°	0.6045	0.58 s

Table 4.1: Settling Time Different Steps

In order to interpret the data obtained, we must consider that the data of the real system may be affected by errors and that the attainment of the reference value is not perfect as the motor control system keeps the value approximately 1%/2% below the real reference.

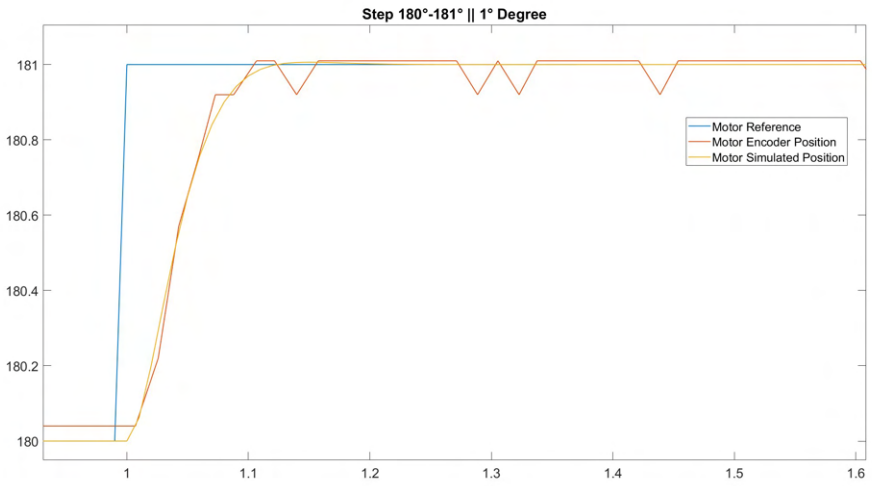

Figure 4.12: Step 1°

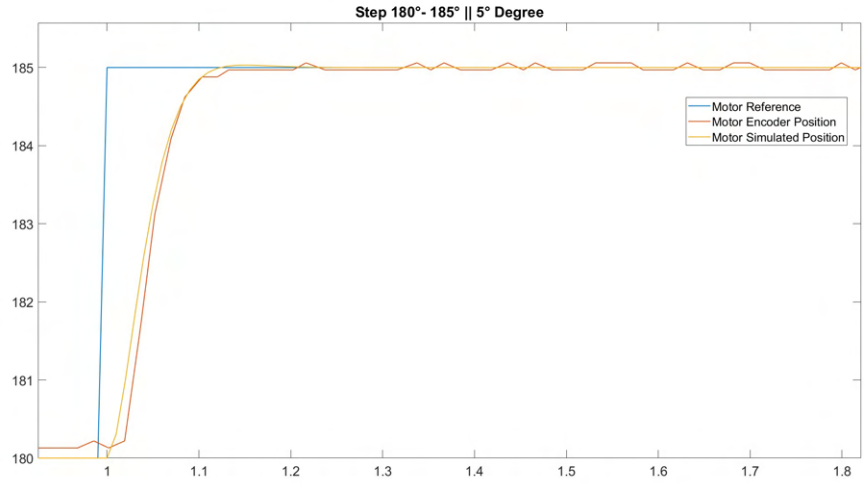

Figure 4.13: Step 5°

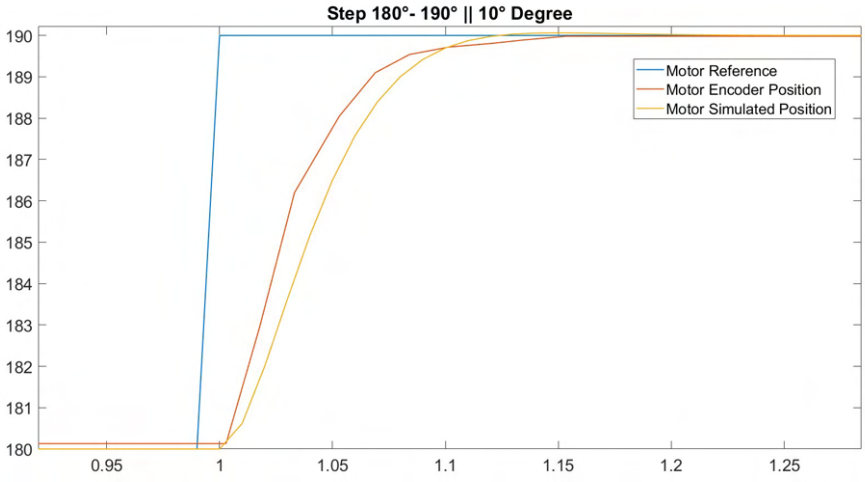

Figure 4.14: Step 10°

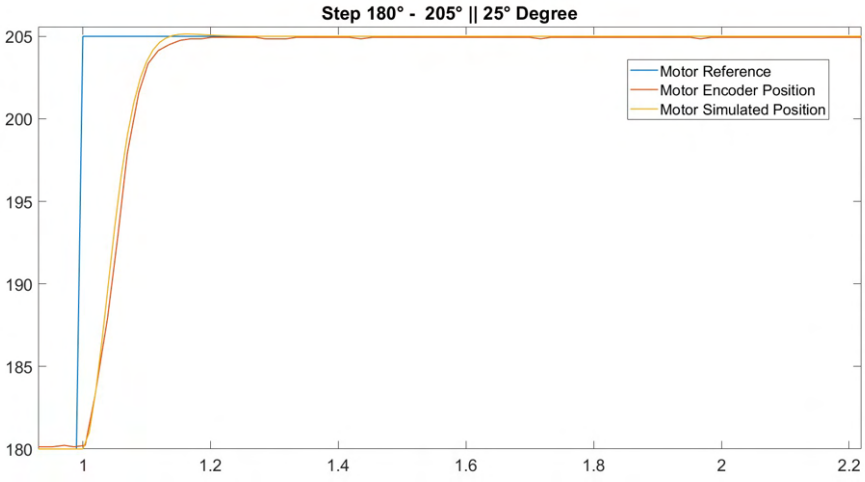

Figure 4.15: Step 25°

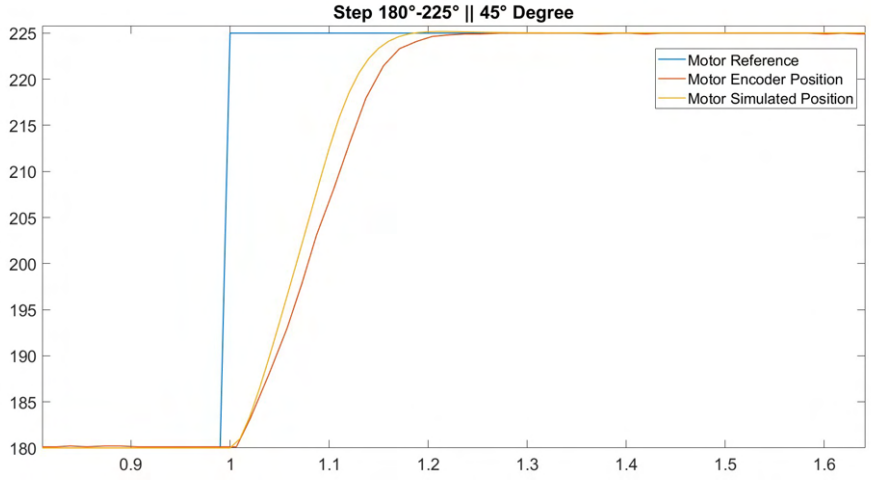

Figure 4.16: Step 45°

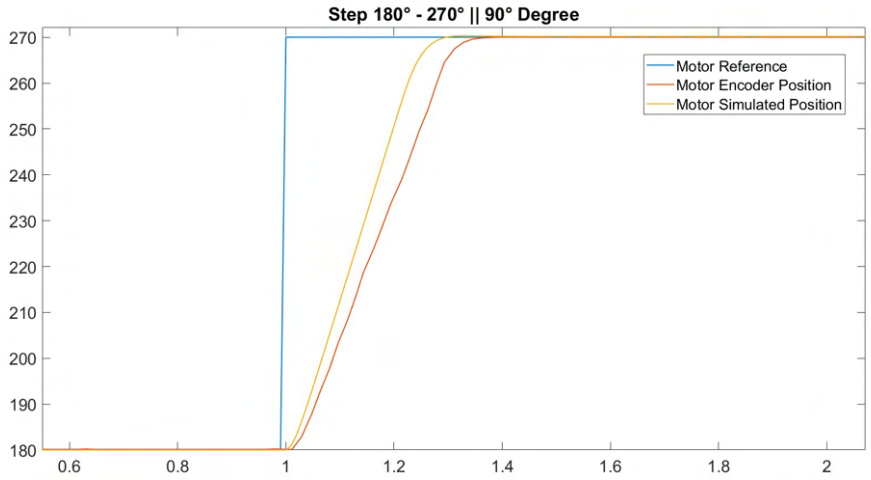

Figure 4.17: Step 90°

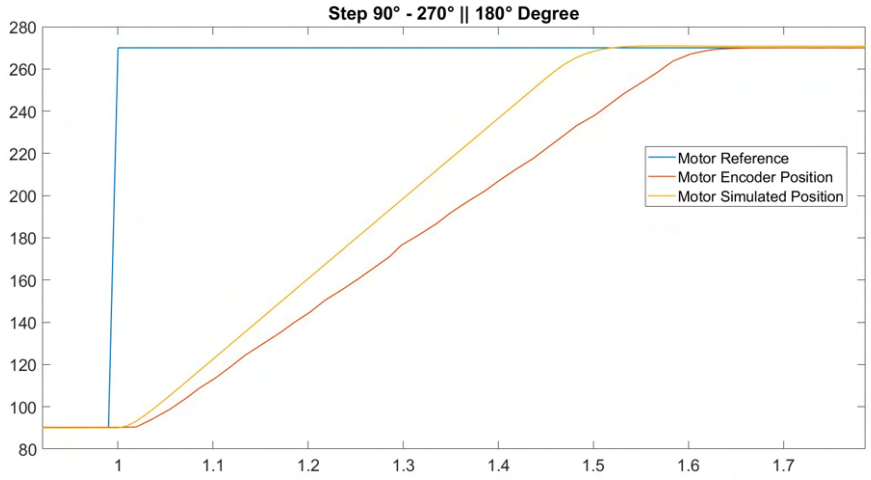

Figure 4.18: Step 180°

As we can see from the images above, the transfer function found succeeds in describing the behaviour of the motor in various steps.
After demonstrating that the obtained transfer function reproduces the same responses as the real system, we go on to simplify the motor model and then obtain the state-space model, which we will later use to describe the behaviour of the motors in the simplified model.

$$G_p(s) = \frac{1597}{s^2 + 68.11s + 1597} \tag{4.25}$$

let's remove the contribution of $-8.13s$, as compared to 1597 it does not affect the response of the system, so we can neglect it.Obviously, the closed-loop speed system will also change.

$$G_v(s) = \frac{G_p(s)}{(1 - G_p(s)) * (Kp_p + \frac{Kp_i}{s})} = \frac{1597s^3 + 1.088e05s^2 + 2.55e06s}{7.031s^5 + 957.9s^4 + 4.387e04s^3 + 7.658e05s^2 + 1.66e04s} \tag{4.26}$$

We now derive the state-space model, considering that the position control loop is the relationship between the actual motor angle and the position reference.

$$G_p(s) = \frac{\Theta(s)}{\Theta^*(s)} = \frac{1597}{s^2 + 68.11s + 1597} \tag{4.27}$$

$$\Theta(s)(s^2 + 68.11s + 1597) = \Theta^*(s)(1597) \tag{4.28}$$

$$\ddot{\theta} + 68.11 * \dot{\theta} + 1597 * \theta = 1597 * \theta^* \tag{4.29}$$

$$x_1 = \theta, x_2 = \dot{x_1} = \dot{\theta}, u_1 = \theta^* \tag{4.30}$$

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1597 & -68.11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_1 \tag{4.31}$$

4.3 BlueROV 2 Model

Let us study the model of the ROV (Remote Operating Vehicle) manufactured by Bluerobotics Inc. and then go on to simplify it with a planar model that will serve as a reference for the system to be simulated.
As described by Fossen [27], two reference frames are needed to build the mathematical model that represents the underwater vehicle: one referenced to the Earth (referred to as the Earth-fixed frame) and the other referenced to the vehicle (referred to as the body-fixed frame). Earth-fixed frame) and the other referenced to the vehicle (referred to as the body-fixed frame). The orthonormal axes are denoted by x_n, y_n, z_n for the earth fixed frame and by x_b,y_b,z_b for the body fixed frame. These reference frames are shown in figure.

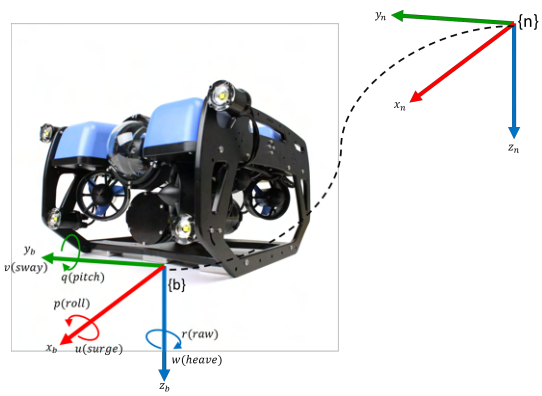

Figure 4.19: Reference frames needed to describe the BlueROV2

The Society of Naval Architects and Marine Engineers (SNAME) has a convention for expressing the position and orientation of vehicles and the forces and moments applied to them. The SNAME nomenclature for position, velocity and forces in underwater vehicles [27] is given in table.

Movement	**Name**	**Position**	**Velocity**	**Force/Moment**
X traslation	Surge	x	u	X
Y traslation	Sway	y	v	Y
Z traslation	Heave	z	w	Z
X rotation	Roll	φ	p	K
Y rotation	Pitch	θ	q	M
Z roation	Yaw	ψ	r	N

Table 4.2: Notation for underwater vehicle's movements, positions, velocities, and forces .

The position of the vehicle with respect to the Earth-fixed frame η and its velocity with respect the body-fixed frame ν can be represented as

$$\eta = (x, y, z, \varphi, \theta, \psi)^T \tag{4.32}$$

$$\nu = (u, v, w, p, q, r)^T \tag{4.33}$$

The forces and moments of the vehicle with respect to the body-fixed frame are described as

$$\tau = (X, Y, Z, K, M, N)^T \tag{4.34}$$

The hydrodynamic model of a UUV can be described by Newton–Euler equations [27] as

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau + \omega \tag{4.35}$$

$$\tau = B_t u_t \tag{4.36}$$

where

- $M \in R^{6x6}$ is the inertial and added mass matrix,
- $C \in R^{6x6}$ is the rigid body and added mass centripetal and Coriolis matrix,

- $D \in R^{6x6}$ is the hydrodynamic damping matrix,

- $g \in R^{6x1}$ is the restitution forces vector,

- $B_t \in R^{6x6}$ is the thruster allocation matrix,

- $u_t \in R^{6x1}$ is a vector containing the force generated by the thrusters, and

- $\omega \in R^{6x1}$ represents environmental disturbances.

Matrices and vectores involved in the vehicle's hydrodynamics contain numerous unknown parameters, making it infeasible to estimate.
However, there are some assumptions and considerations that can be made due to the geometry and operation of BlueROV2, resulting in a significant reduction of unknown parameters. These conditions are listed below:

- Since BlueROV2 operates at relatively low speeds (i.e. less than 2 m/s), lift forces can be neglected.

- BlueROV2 is assumed to have port to starboard and forward to aft symmetry, and the centre of gravity (CG) is in the symmetry planes.

- BlueROV2 is assumed to operate below the wave-affected zone. As a result, the effects of waves ω on the vehicle are negligible.

- The thruster allocation of BlueROV2 does not allow active control of the pitch orientation q. However, the motion about this axis is considered to be self-regulated due to the vehicle bouyant restoring moment, resulting in the following reduced system.

$$\nu = (u, v, w, p, 0, r)^T \tag{4.37}$$

Considering the reduced system detailed before, the physical and hydrodynamic parameters of BlueROV2 are summarized in the table below as identified in [[4],[23],[28]].

Parameters	Symbol	Value	Unit
Mass	m	10	kg
Bouyancy	B	100.06	N
Weight	W	98.1	N
Center of gravity	$r_G = (x_g, y_g, z_g)$	$r_G = (0, 0, 0)$	m
Center of bouyancy	$r_B = (x_b, y_b, z_b)$	$r_B = (0, 0, 0.02)$	m
Inertia moment	$I = diag(I_x, I_y, I_z)$	$I = diag(0.16, 0.16, 0.16)$	kgm^2
Added mass parameters	$X_{\dot{u}}$	-5.5	kg
	$Y_{\dot{\nu}}$	-12.7	kg
	$Z_{\dot{\omega}}$	-14.57	kg
	$P_{\dot{\rho}}$	-0.12	$kg\frac{m^2}{rad}$
	$M_{\dot{q}}$	-0.12	$kg\frac{m^2}{rad}$
	$N_{\dot{r}}$	-0.12	$kg\frac{m^2}{rad}$
Linear Damping parameters	X_u	-4.03	$N\frac{s}{m}$
	Y_ν	-6.22	$N\frac{s}{m}$
	Z_ω	-5.18	$N\frac{s}{m}$
	P_ρ	-0.07	$N\frac{s}{rad}$
	M_q	-0.07	$N\frac{s}{rad}$
	N_r	-0.07	$N\frac{s}{rad}$
Quadratic Damping parameters	$X_{u\|u\|}$	-18.18	$N\frac{s^2}{m^2}$
	$Y_{\nu\|\nu\|}$	-21.66	$N\frac{s^2}{m^2}$
	$Z_{\omega\|\omega\|}$	-36.99	$N\frac{s^2}{m^2}$
	$P_{p\|p\|}$	-1.55	$N\frac{s^2}{rad^2}$
	$M_{q\|q\|}$	-1.55	$N\frac{s^2}{rad^2}$
	$N_{r\|r\|}$	-1.55	$N\frac{s^2}{rad^2}$

Table 4.3: BlueROV2 parameters

The matrices that comprise the hydrodynamical model in Equation (6.35) are considered as defined in [27]. Considering the thruster distribution in the BlueROV2, the thruster allocation matrix is assembled as

$$\big[B_t\big] = \begin{bmatrix} 0.7071 & 0.7071 & 0.7071 & 0.7071 & 0 & 0 \\ 0.7071 & -0.7071 & -0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0.115 & -0.115 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.1773 & -0.1773 & -0.1773 & -0.1773 & 0 & 0 \end{bmatrix} \tag{4.38}$$

4.4 BlueROV 2 Planar Model

As stated in the paper [30], we focus on 2D planar dynamics, i.e., only x-y plane motions are considered and heave, roll and pitch motions are not considered. In addition, the following assumptions are made:
- It is assumed that BlueROV2 is hydrodynamically symmetrical, i.e., there are no hydrodynamic coupling terms.

- It is assumed that BlueROV2 operates away from the area affected by waves, i.e., wave effects are negligible and only currents are considered.

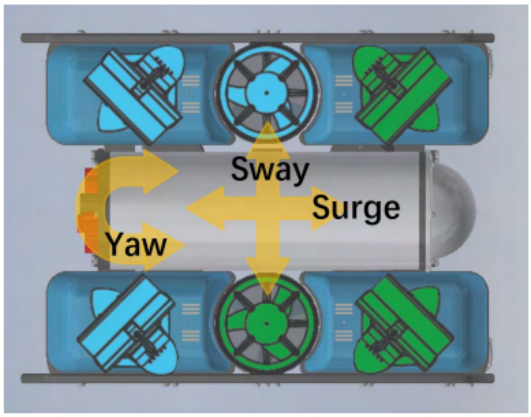

Figure 4.20

Parameters	Symbol	Value	Unit
Length	L	0.457	m
Width	W	0.338	m
Height	H	0.254	m
Mass	m	10	kg
Yaw moment	I_{zz}	0.16	$\text{kg} \cdot \text{m}^2$
Surge added mass	I_{Ax}	-5.5	kg
Sway added mass	I_{Ay}	-12.7	kg
Yaw added mass	I_{An}	- 0.12	$\text{kg} \cdot \text{m}^2/\text{rad}$
Quadratic damping coefficient	C_D	0.5	-
Surge cross section area	A_x	0.048	m^2
Sway cross section area	A_y	0.10	m^2
Yaw cross section area	A_n	0.07	m^5

The equation of motion for an ROV can be described by the Newton-Euler equation presented by [27]:

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \tag{4.39}$$

- M mass matrix
- $C(v)$ Coriolis matrix
- $D(v)$ damping matrix
- $g(\eta)$ gravitational forces and moments
- v is the velocity
- τ is the external driving forces

For an x-y planar dynamics in a global reference system at a fixed latitude, $C(v)$ and $g(\eta)$ are zero, and equation (21) is simplified and expanded to:

$$\begin{bmatrix} m+I_{Ax} & 0 & 0 \\ 0 & m+I_{Ay} & 0 \\ 0 & 0 & I_{zZ}+I_{An} \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} + \frac{1}{2}\rho C_D A \begin{bmatrix} |u| \\ |v| \\ |r| \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} \quad (4.40)$$

where $[u, v, w]^T$ are the surge sway and yaw velocities. Respectively $[\dot{u}, \dot{v}, \dot{r}]$ are the accelerations in surge, sway, yaw and $[X, Y, N]^T$ are the forces and moments in surge, sway, yaw . $[I_{Ax}I_{Ay}I_{An}]$ are components of added mass, ρ is the density of water, C_D is the drag coefficient and A is the cross-sectional area for drag.
The added mass on a rigid body is a virtual mass caused by the surrounding fluid. In this study, we assume that the added mass $[I_{Ax}I_{Ay}I_{An}]$ is equal to the mass and moment of inertia of BlueROV2.
Under the assumption of hydrodynamic symmetry, all coupled terms are not considered. Consequently, the drag force can be considered proportional to the square of the relative velocity between the currents and act in the opposite direction to the motion of the ROV. Based on the above, the drag forces of forward, sway and yaw can be expressed as:

$$D_x = -\frac{1}{2}\rho C_D A_x |u| u$$
$$D_y = -\frac{1}{2}\rho C_D A_y |v| v$$
$$D_\psi = -\frac{1}{2}\rho C_D A_n |r| r$$

In this study, the drag coefficients for 3 DOF were all assumed to be 0.5. The cross-sectional areas for each direction are listed in the Table.

In the case where the flexible arm is positioning on the ROV, equation (28) becomes:

$$\begin{bmatrix} m+m_l+I_{Ax}+I_{Axl} & 0 & 0 \\ 0 & m+m_l+I_{Ay}+I_{Ayl} & 0 \\ 0 & 0 & I_{zZ}+I_{An}+I_{Anl} \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} + \frac{1}{2}\rho C_D A \begin{bmatrix} |u| \\ |v| \\ |r| \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \begin{bmatrix} X+X_d \\ Y+Y_d \\ N+N_d \end{bmatrix} \quad (4.41)$$

where m_l and m are the mass of link and motor , $[I_{Axl}, I_{Ayl}, I_{Anl}]^T$ and $[I_{A_x}, I_{A_y}, I_{A_z}]^T$ are the components of added mass of the motor and link, and $[X_d, Y_d, N_d]^T$ are the forces and moments generated by the flexible arm during ROV motion in contact and no-contact cases. From free body static equilibrium of the flexible link, by consider it as a rigid body with a length of L:

$$X_d = F_{x,0} \quad (4.42)$$
$$Y_d = F_{y,0} \quad (4.43)$$
$$N_d = F_{x,0} * h + F_{y,0} * h \quad (4.44)$$
$$(4.45)$$

where h is the distance from center of ROV to motor joint, $F_{x,0}$ and $F_{y,0}$ are the component of contact force. In this case we consider the axial component of contact force although we are not meeting assumption 8. In this case, however from the perspective of Rov's model it can be seen as a measurable disturbance. With regard to moment equilibrium the only force acting is along the y-axis.

4.4.1 Interaction forces between ROV and Antenna Link in case of contact

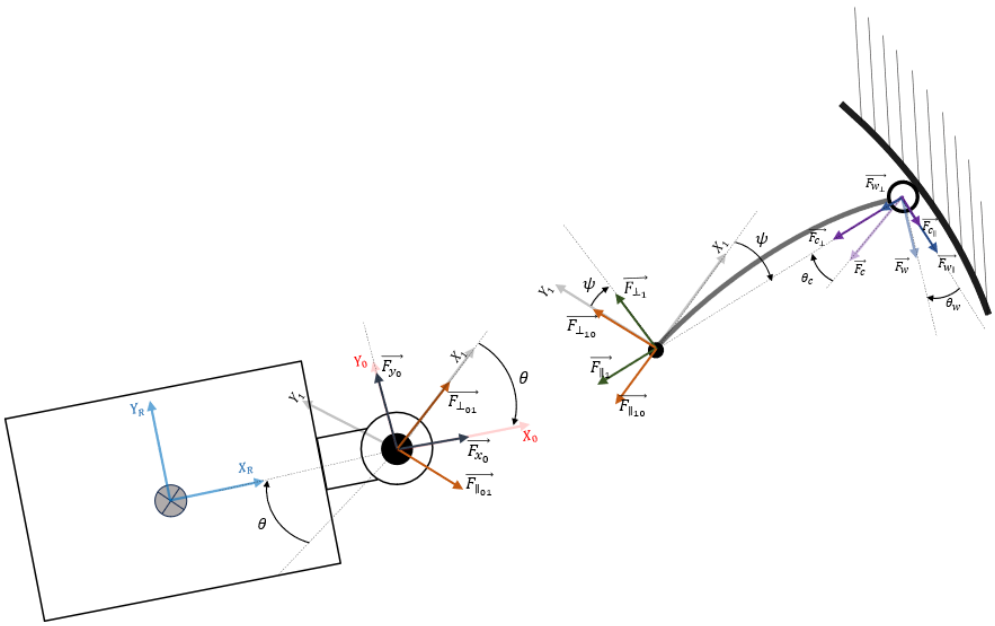

Figure 4.21: Interaction Forces between both systems

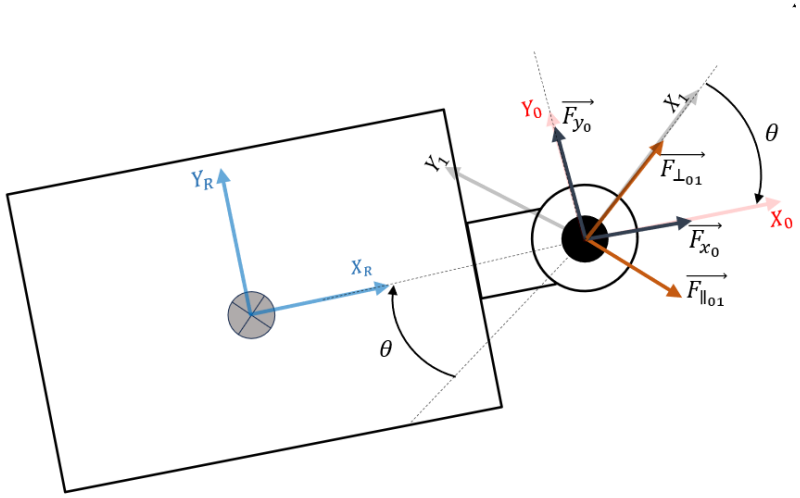

Figure 4.22: Interaction Forces on ROV systems

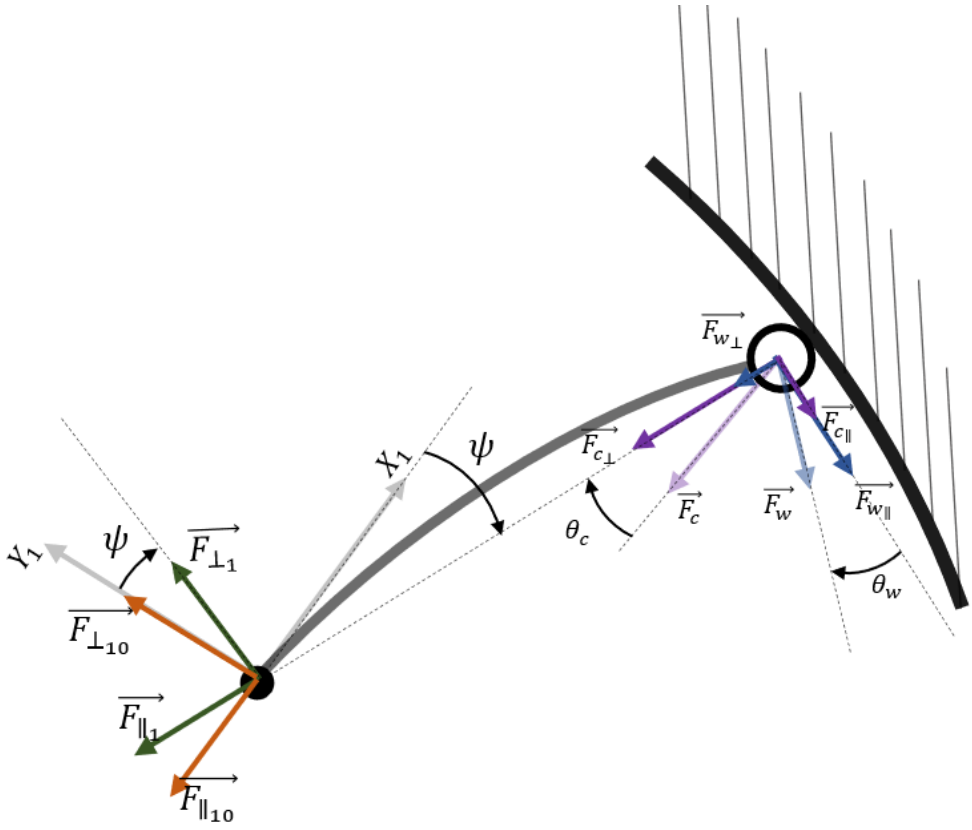

Figure 4.23: Interaction Forces on Antenna systems

Starting from the forces acting on the antenna tip, namely the contact force and the drag force, we can decompose them on the axis joining the position of the joint and the

antenna tip. We obtain two components parallel and perpendicular to the axis:

$$\overrightarrow{F_w} = \overrightarrow{F_{w,\parallel}} + \overrightarrow{F_{w,\perp}} \tag{4.46}$$

$$F_w = F_w cos(\theta_w) + F_w sen(\theta_w) \tag{4.47}$$

$$\overrightarrow{F_c} = \overrightarrow{F_{c,\parallel}} + \overrightarrow{F_{c,\perp}} \tag{4.48}$$

$$F_c = F_c cos(\theta_c) + F_c sen(\theta_c) \tag{4.49}$$

Considering the reacting forces on the joint from antenna point of view, we obtain:

$$\overrightarrow{F_{\parallel,1}} = \overrightarrow{F_{c,\parallel}} + \overrightarrow{F_{w,\parallel}} \tag{4.50}$$

$$\overrightarrow{F_{\perp,1}} = \overrightarrow{F_{c,\perp}} + \overrightarrow{F_{w,\perp}} \tag{4.51}$$

$$F_{\parallel,1} = F_w cos(\theta_w) + F_c cos(\theta_c) \tag{4.52}$$

$$F_{\perp,1} = F_w sen(\theta_w) + F_c sen(\theta_c) \tag{4.53}$$

Now, by projecting the forces onto reference system X_1, Y_1 we get the forces $\overrightarrow{F_{\parallel,10}}$ and $\overrightarrow{F_{\perp,10}}$, which can be considered as reaction forces of joint respect to frame X_1, Y_1.

$$F_{\parallel,10} = F_{\perp,1} sen(\psi) + F_{\parallel,1} cos(\psi) \tag{4.54}$$

$$F_{\perp,10} = F_{\perp,1} cos(\psi) + F_{\parallel,1} sen(\psi) \tag{4.55}$$

From the perspective of rov, the reaction forces are $\overrightarrow{F_{\parallel,01}}$ and $\overrightarrow{F_{\perp,01}}$.

$$\overrightarrow{F_{\parallel,01}} = \overrightarrow{F_{\parallel,10}} \tag{4.56}$$

$$\overrightarrow{F_{\perp,01}} = \overrightarrow{F_{\perp,10}} \tag{4.57}$$

By projecting the forces onto reference system X_0, Y_0 we get the forces $\overrightarrow{F_{x,0}}$ and $\overrightarrow{F_{y,0}}$, which can be considered as reaction forces of joint respect to frame X_0, Y_0.

$$F_{x,0} = F_{\perp,01} sen(\theta) + F_{\parallel,01} cos(\theta) \tag{4.58}$$

$$F_{y,0} = F_{\perp,01} cos(\theta) + F_{\parallel,01} sen(\theta) \tag{4.59}$$

Substituting 7-10 into 13-14:

$$F_{x,0} = (F_{\perp,1} cos(\psi) + F_{\parallel,1} sen(\psi)) sen(\theta) + (F_{\perp,1} sen(\psi) + F_{\parallel,1} cos(\psi)) cos(\theta) \tag{4.60}$$

$$F_{y,0} = (F_{\perp,1} cos(\psi) + F_{\parallel,1} sen(\psi)) cos(\theta) + (F_{\perp,1} sen(\psi) + F_{\parallel,1} cos(\psi)) sen(\theta) \tag{4.61}$$

$$\begin{aligned} F_{x,0} = {} & ((F_w sen(\theta_w) + F_c sen(\theta_c)) cos(\psi) + (F_w cos(\theta_w) + F_c cos(\theta_c)) sen(\psi)) sen(\theta) \\ & + ((F_w sen(\theta_w) + F_c sen(\theta_c)) sen(\psi) + ((F_w cos(\theta_w) + F_c cos(\theta_c)) cos(\psi)) cos(\theta) \end{aligned} \tag{4.62}$$

$$\begin{aligned} F_{y,0} = {} & ((F_w sen(\theta_w) + F_c sen(\theta_c)) cos(\psi) + (F_w cos(\theta_w) + F_c cos(\theta_c)) sen(\psi)) cos(\theta) \\ & + ((F_w sen(\theta_w) + F_c sen(\theta_c)) sen(\psi) + (F_w cos(\theta_w) + F_c cos(\theta_c)) cos(\psi)) sen(\theta) \end{aligned} \tag{4.63}$$

Once we have obtained the two forces on the x0 and y0 axes, we have also obtained the forces acting on the rov, which we are going to insert as disturbance within the mathematical formulation of the ROV.

$$X_d = F_{x,0} \tag{4.64}$$
$$Y_d = F_{y,0} \tag{4.65}$$
$$N_d = F_{x,0} * h + F_{y,0} * h \tag{4.66}$$
$$\tag{4.67}$$

where h is the distance from center of ROV to motor joint. Considering that the torque provided by the motor does not affect directly the disturbance N_d. We should consider θ_w equal to 0°, and made an assumption that the drag forces act as a tangential force respect to the axis from tip antenna end position of motor joint. In this case we obtain $\overrightarrow{F_{w,\parallel}}$ equal to $\overrightarrow{F_w}$ and $\overrightarrow{F_{w,\perp}}$ equal to zero.

$$\begin{aligned} F_{x,0} = &((F_c cos(\theta_c))cos(\psi) + (F_w + F_c cos(\theta_c))sen(\psi))sen(\theta) \\ &+((F_c cos(\theta_c))sen(\psi) + ((F_w + F_c cos(\theta_c))cos(\psi))cos(\theta) \end{aligned} \tag{4.68}$$

$$\begin{aligned} F_{y,0} = &((F_c sen(\theta_c))cos(\psi) + (F_w + F_c cos(\theta_c))sen(\psi))cos(\theta) \\ &+((F_c sen(\theta_c))sen(\psi) + (F_w + F_c cos(\theta_c))cos(\psi))sen(\theta) \end{aligned} \tag{4.69}$$

4.5 Complete Model

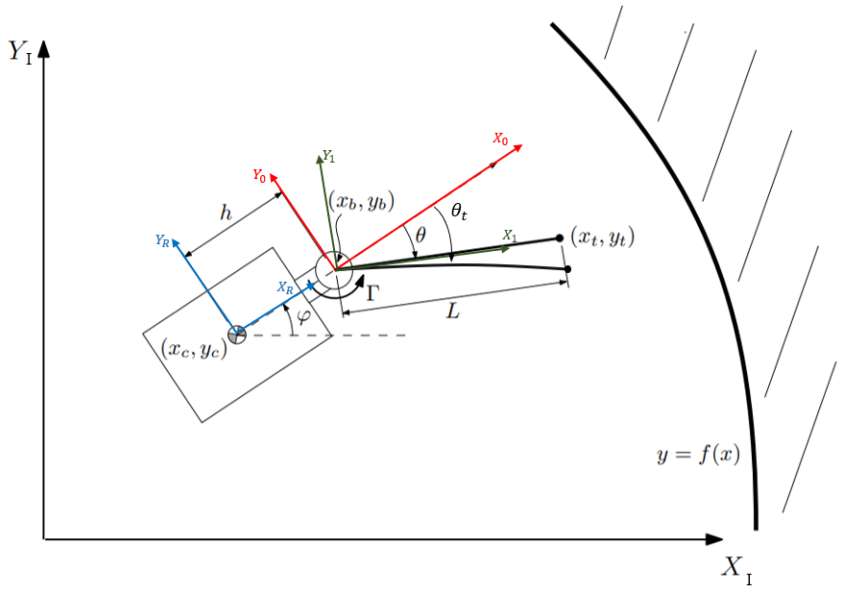

Figure 4.24: ROV and Flexible Antenna

We construct the kinematic model with respect to the point (x_c, y_c) and the antenna tip (x_t, y_t). Thus, the variables we consider are x_c the position on the X_I axis of the ROV, y_c the position on the Y_I axis, x_t the position of the antenna tip on the X_I axis, y_c the position of the antenna tip on the Y_I axis, θ_t is the angular position of the tip

mass and φ is the angular position of the ROV. L and h are the length of the antenna and the distance between (x_c, y_c) and (x_b, y_b) that is the position of actuated joint of the flexible antenna.
It is possible to obtain the point (x_t, y_t) with respect to the position of the ROV by expressing it in this form:

$$\begin{aligned} x_{tR} &= x_c + hcos(\varphi) + Lcos(\varphi - \theta_t) \\ y_{tR} &= y_c + hsin(\varphi) + Lsin(\varphi - \theta_t) \end{aligned} \tag{4.70}$$

(x_{tR}, y_{tR}) is the point respect to ROV reference frame, to obtain it in the global reference frame:

$$R(\varphi)^{-1} = \begin{bmatrix} cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) \end{bmatrix} \tag{4.71}$$

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} cos(\varphi)x_{tR} - sin(\varphi)y_{tR} \\ sin(\varphi)x_{tR} + cos(\varphi)y_{tR} \end{bmatrix} \tag{4.72}$$

To linearly relate angle velocity to point velocity the Jabobian matrix is derived as:

$$J = \begin{bmatrix} \frac{\partial x_t}{\partial \theta_t} & \frac{\partial x_t}{\partial \varphi} \\ \frac{\partial y_t}{\partial \theta_t} & \frac{\partial y_t}{\partial \varphi} \end{bmatrix} \tag{4.73}$$

$$J = \begin{bmatrix} J_{1,1} & J_{1,2} \\ J_{2,1} & J_{2,2} \end{bmatrix} \tag{4.74}$$

$$J_{1,1} = -L\, cos(\theta_t - \varphi)\ \sin(\varphi) - L\ \sin(\theta_t - \varphi)cos(\varphi) \tag{4.75}$$

$$J_{2,1} = L\ \cos\left(\theta_t - \varphi\right)\ \cos\left(\varphi\right) - L\ \sin\left(\theta_t - \varphi\right)\ \sin\left(\varphi\right) \tag{4.76}$$

$$\begin{aligned} J_{1,2} = -\sin\left(\varphi\right)\ \left(h\ \cos\left(\varphi\right) - L\ \cos\left(\theta_t - \varphi\right)\right) - \cos\left(\varphi\right)\ \left(h\ \sin\left(\varphi\right) - L\ \sin\left(\theta_t - \varphi\right)\right) - \sin\left(\varphi\right) \\ , \left(x_c + h\ \cos\left(\varphi\right) + L\ \cos\left(\left(\theta_t - \varphi\right)\right) - \cos(\varphi)\, y_c + h\ \sin(\varphi) + L\ \sin(\theta_t - \varphi)\right) \end{aligned} \tag{4.77}$$

$$\begin{aligned} J_{2,2} = \cos\left(\varphi\right)\ \left(x_c + h\ \cos\left(\varphi\right) + L\ \cos\left(\theta_t - \varphi\right)\right) - \sin\left(\varphi\right)\ \left(h\ \sin\left(\varphi\right) - L\ \sin\left(\theta_t - \varphi\right)\right) - \sin\left(\varphi\right) \\ \left(y_c + h\ \sin\left(\varphi\right) + L\ \sin\left(\theta_t - \varphi\right)\right) + \cos\left(\varphi\right)\ \left(h\ \cos\left(\varphi\right) - L\ \cos\left(\theta_t - \varphi\right)\right) \end{aligned} \tag{4.78}$$

The kinematic model of the flexible link consider motion of ROV is:

$$\begin{bmatrix} \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} J_L \end{bmatrix} \begin{bmatrix} \dot{\theta}_t \\ \dot{\varphi} \end{bmatrix} \tag{4.79}$$

The kinematic model of ROV is:

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\varphi}_c \end{bmatrix} = \begin{bmatrix} cos(\varphi) & -sin(\varphi) & 0 \\ sin(\varphi) & cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{4.80}$$

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\varphi}_c \end{bmatrix} = [J_R] \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{4.81}$$

where $[u, v, r]^T$ are the surge sway and yaw velocities. The entire model is:

$$\begin{bmatrix} \dot{x}_L \\ \dot{y}_L \\ \dot{x}_R \\ \dot{y}_R \\ \dot{\varphi}_R \end{bmatrix} = \begin{bmatrix} J_L & 0 \\ 0 & J_R \end{bmatrix} \begin{bmatrix} \dot{\theta}_t \\ \dot{\varphi} \\ u \\ v \\ r \end{bmatrix} \tag{4.82}$$

To obtain the velocity of the tip of antenna we need the angular velocity given by (6.19), in this formulation it is considered the case of contact with the environment. Furthermore, to consider the forces acting on ROV given by the flexible arm at the moment of contact and non-contact, we add $[X_d, Y_d, N_d]^T$ to the forces and moments $[X, Y, N]^T$ in surge, sway, yaw, in equation (6.40). These values will be given by an F-T sensor placed on the flexible arm actuator.

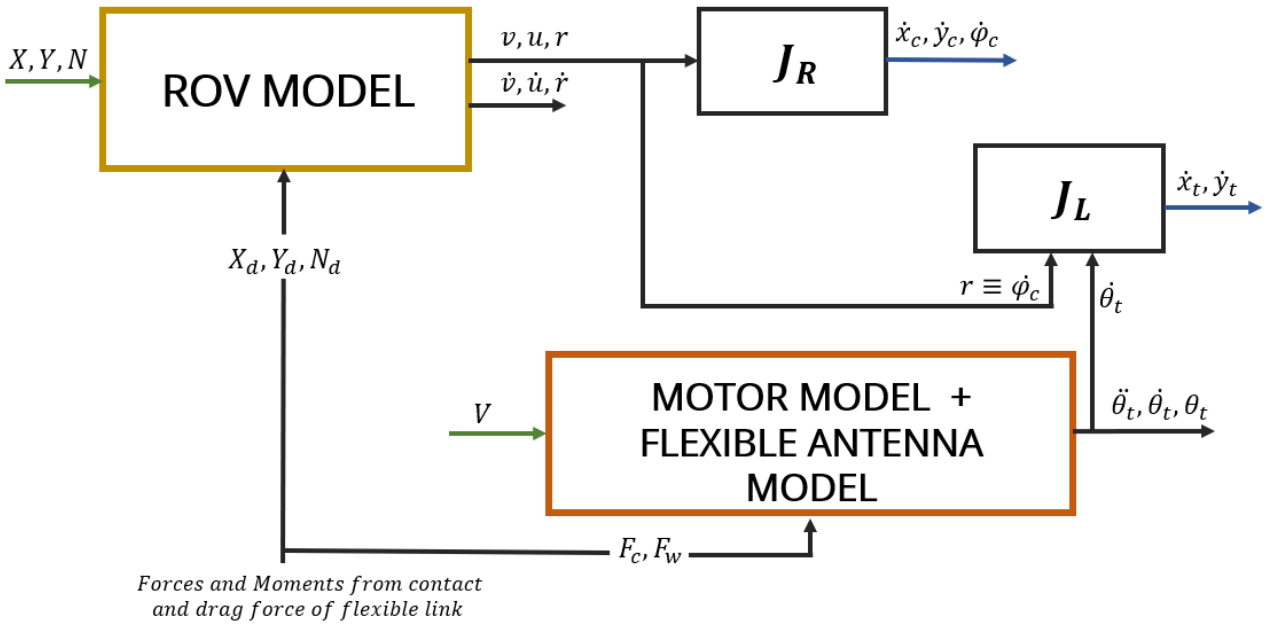

Figure 4.25: Scheme of plant

4.6 Simplified Model

Now let us consider a simplified model, this model only represents the condition of contact with the wall, in fact as we can see from the equations below, a constraint is imposed on the position of the antenna tip with respect to the wall. This represents that during the simulation we always have contact, and the rov will vary its position and orientation to ensure the desired deflection and distance.

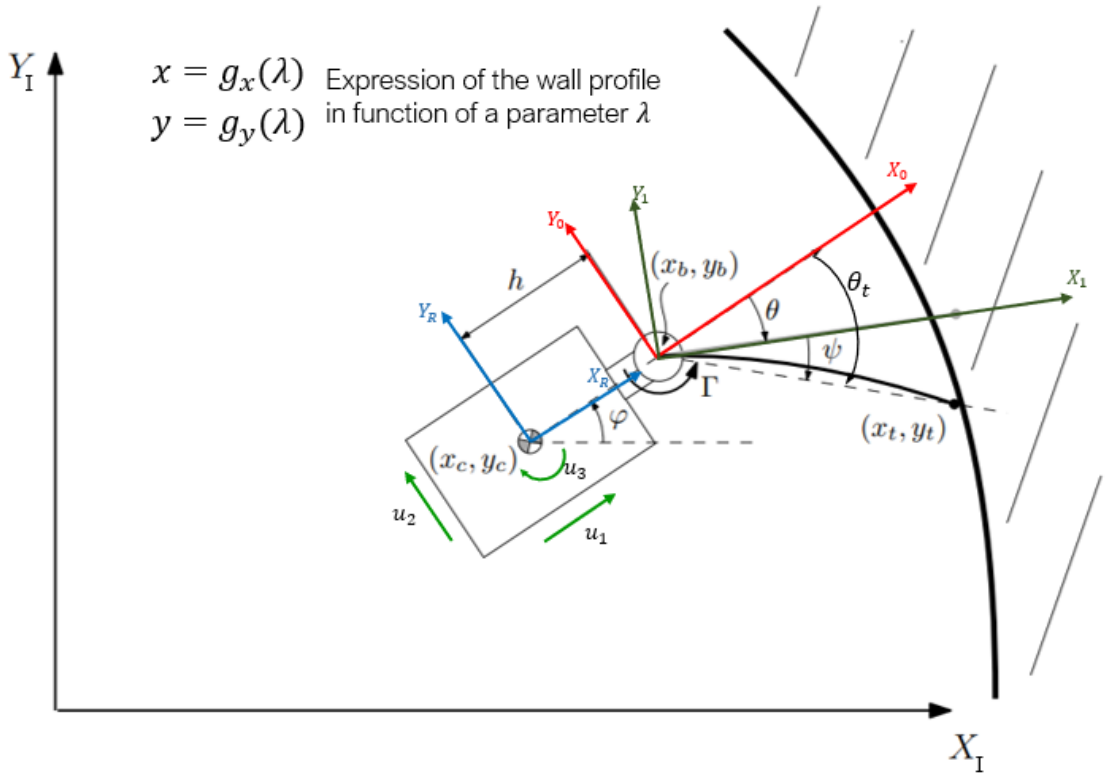

Figure 4.26: Rov + Antenna in Case of Contact

$$\begin{cases} x_c + h * cos(\varphi) + h * cos(\varphi + \theta + \psi) = g_x(\lambda) & (6.83.1) \\ y_c + h * sen(\varphi) + h * sen(\varphi + \theta + \psi) = g_y(\lambda) & (6.83.2) \\ \dot{\varphi} = u_3 & (6.83.3) \\ \dot{x_c} = cos(\varphi)u_1 - sen(\varphi)u_2 & (6.83.4) \\ \dot{y_c} = sen(\varphi)u_1 + cos(\varphi)u_2 & (6.83.5) \\ \dot{\theta} = \theta & (6.83.6) \\ \ddot{\theta} = -1597\theta - 68.11\dot{\theta} + 1597 * u_4 & (6.83.7) \\ \Gamma = c\psi & (6.83.8) \\ d = h * sen(\theta + \psi) & (6.83.9) \end{cases} \quad (4.83)$$

We have:

- 6 internal variables : $x_c, y_c, \varphi, \theta, \psi, \lambda$
- 4 inputs : u_1, u_2, u_3, u_4
- 2 algebraic equations: (6.83.1), (6.83.2)
- 5 state equations: (6.83.3), (6.83.4), (6.83.5), (6.83.6),(6.83.7)
- 2 output: d(6.83.9), ψ (6.83.1-6.83.2)
- 1 equation to estimate ψ from a measurement : (6.83.8)

where x_c and y_c is the position of the rov with respect to the reference system (X_I, Y_I), φ is the orientation of the rov, θ the angle position of the motor, $\dot{\theta}$ the angular velocity of the motor, ψ the deflection, λ parameter to describe the profile of the wall. The inputs shown represent u_1 the linear velocity along the x-axis. u_2 the linear velocity along the y-axis, u_3 the angular velocity with respect to the z-axis, u_4 the motor position reference. Equations (6.83.3)-(6.83.7) show that y can control $x_c, y_c, \varphi, \theta$ Equations (6.83.1)-(6.83.2) show that, consequently, ψ and λ are defined by the previous ones. Let us define: $U = [u_1, u_2, u_3, u_4]^T$; $X = [x_c, y_c, \varphi, \theta, \dot{\theta}]^T$; $y = [d, \psi]$

$$\dot{x} = Ax + Bu \tag{4.84}$$

$$y = l * sen(x4 + \psi) \tag{4.85}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1597 & -68.11 \end{bmatrix} \tag{4.86}$$

$$B(\varphi) = \begin{bmatrix} cos(\varphi) & -sen(\varphi) & 0 & 0 \\ sen(\varphi) & cos(\varphi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1597 \end{bmatrix} \tag{4.87}$$

$$\begin{bmatrix} \psi \\ d \end{bmatrix} = C(x) \tag{4.88}$$

Let us assume that the rover moves at a constant linear velocity V . then u_1 and u_2 will be used only to define the orientation of the movement of the rover ξ:

$$\begin{aligned} \dot{x_c} &= V cos(\xi) \\ \dot{y_c} &= V sen(\xi) \end{aligned} \tag{4.89}$$

Then

$$\begin{bmatrix} \dot{x_c} \\ \dot{y_c} \end{bmatrix} = V \begin{bmatrix} cos(\xi) \\ sen(\xi) \end{bmatrix} = \begin{bmatrix} cos(\varphi) & -sen(\varphi) \\ sen(\varphi) & cos(\varphi) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.90}$$

Moreover, we will pursue that φ tends to ξ, i.e, moreover orientation is aligned with the velocity direction. Then we propose the control law $u_3 = (\xi - \varphi)k_3$ begin k_3 the gain of this inner loop. Taking into account the state equation corresponding to φ, we have that:

$$\dot{\varphi} = k_3(\xi - \varphi) \tag{4.91}$$

$$k_3\xi = \dot{\varphi} - k_3\varphi \tag{4.92}$$

whose transfer function

$$\frac{\varphi(s)}{\xi(s)} = \frac{1}{1 + \frac{s}{k_3}} \tag{4.93}$$

increasing k_3 we get a faster system. Then the state space model is modified as:

$$\dot{x} = Ax + \begin{bmatrix} Vcos(\xi) \\ Vsen(\xi) \\ k_3(\xi - \varphi) \\ 0 \\ 1597 * u_4 \end{bmatrix} \tag{4.94}$$

$$y = \begin{bmatrix} d \\ \psi \end{bmatrix} \tag{4.95}$$

then, now, the system has only two input ξ, u_4 . And a control loop to be closed that makes the rover follow a reference : d^* (desired distance) , ψ (related to the desired contact force F^* which is given by $\Gamma = c * \psi^* = h * F^*$ $F^* = \frac{c}{h}\psi^*$).

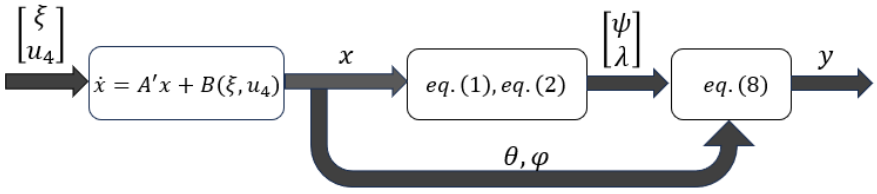

Figure 4.27: Scheme of the System

The measured variables are:

- ψ : $\Gamma = c\psi$
- θ : $d = l * sen(\theta + \frac{1}{c}\psi)$
- φ : measured with optotrack

We have finally obtained a simplified system of the whole system that we will need to build a simulation and in the future test possible controllers.

Chapter 5

Test And Result

The tests carried out on the system were primarily aimed at understanding and analysing the behaviour of the antenna in a different environment, such as water. From this test, we will finally be able to establish a relationship between the position of the motor and the torque generated by the antenna on the torque sensor.

5.1 Setting Test

The test was carried out using an antenna mounted on the force-torque sensor, with the following antenna characteristics:

- Material: carbon fibre
- Length : $0.491m$
- Diametro : $0.00197m$
- Mass : $0.0024kg$
- Additional Mass Sphere : $0.00117kg$
- Modulo di Young : $1.15x10^{1}1N/m^{2}$
- Inertia : $7.85x10^{-}13Kg/m^{2}$

As we can see from the list, there is also an additional mass. This ball has been placed at the tip of the rod in order to reduce the number of points of contact with a surface in the future and thus obtain a simpler model of the system that can describe the behaviour with a contact surface

5.2 Test and Results System in Air

Before analysing the system in an underwater environment, it was necessary to observe the behaviour of the antenna in air. Such a test would allow us to note the behaviour and the relationship between the torque and the position of the motor in an environment different from the one we will be analysing later.

Figure 5.1: Antenna System on Earth

The test was carried out in a different system to the one developed in the previous chapters. The main advantage of this system is that it allows us to have a higher data frequency, around 200 Hz, and also a higher quality signal. The signal used to test the system is a chirp signal, i.e. a sinusoidal signal with a variable frequency. By applying this signal, we obtain the frequency response of the system by relating the force-torque sensor data to the motor position given by the motor. The frequency at which the chirp is realised is 60 Hz, because the tests carried out previously did not mention that this frequency value allows us to visualise the second resonance peak of the system.

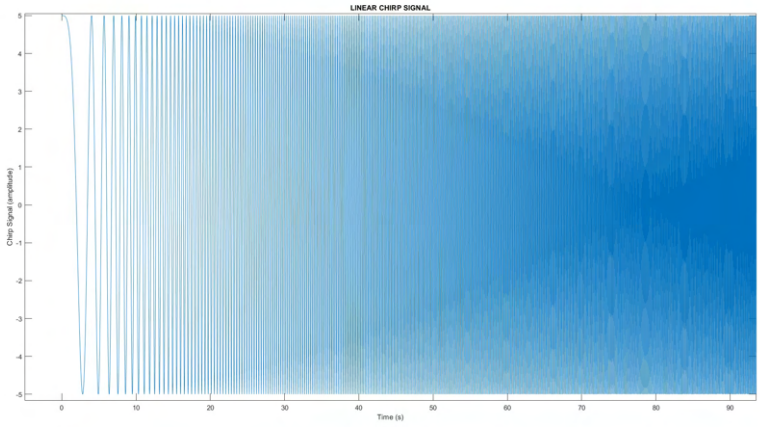

Figure 5.2: Linear Chirp Signal

The data obtained are as follows.

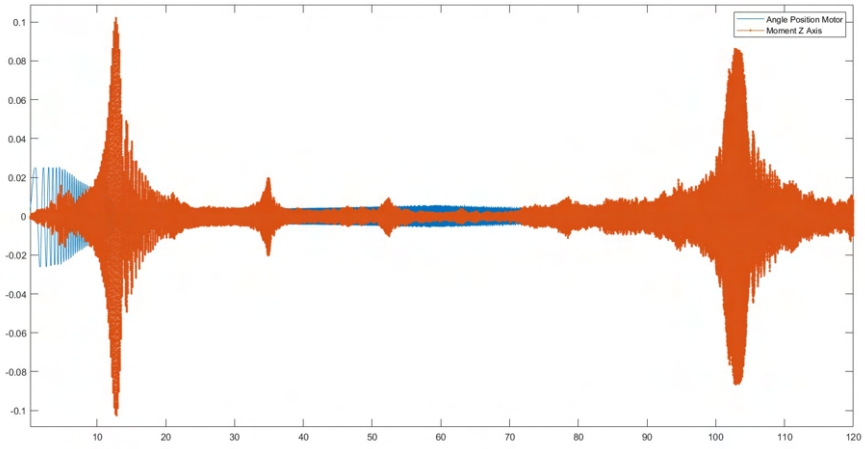

Figure 5.3: Data Response of Moment on Z Axis and Motor Position

From the following data, the Bode plot is obtained by FFT (Fast Furier Transform)[28] and by Welch (Spectral Density Estimation)[22].

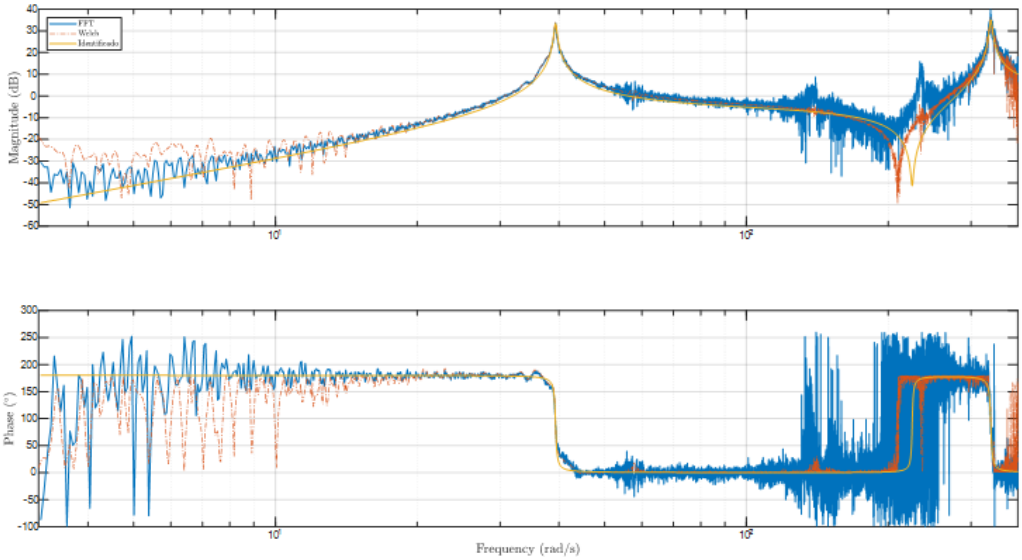

Figure 5.4: Bode Diagram Test Antenna in Air

As can be seen, we have two resonance peaks, at $39.3069 rad/s$ and $329.9 rad/s$ respectively. The two methods of representation give us the same results, from which we will later estimate a transfer function representing the behaviour of the system studied. This transfer function, estimated by parameter optimisation, is as follows

$$\frac{\Gamma(s)}{\theta(s)} = \frac{1.14s^4 + 2.152s^3 + 5.768e04s^2}{s^4 + 4.058s^3 + 1.107e05s^2 + 5.311e04s + 1.689e08} \tag{5.1}$$

5.3 Test and Results System in Water

The system developed in the following thesis and in the previous paragraphs was used for the water test. The aim of these tests was twofold: to understand the behaviour of

the antenna in a different environment and to test the data acquisition and processing system for future implementation on the ROV.

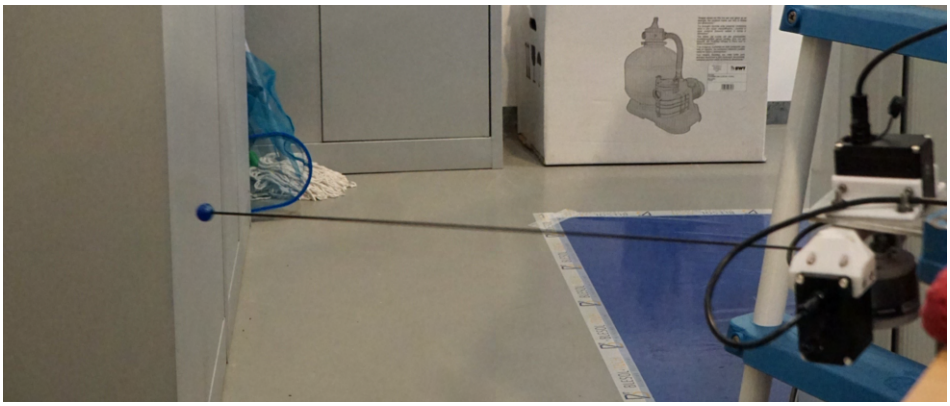

Figure 5.5: Antenna System

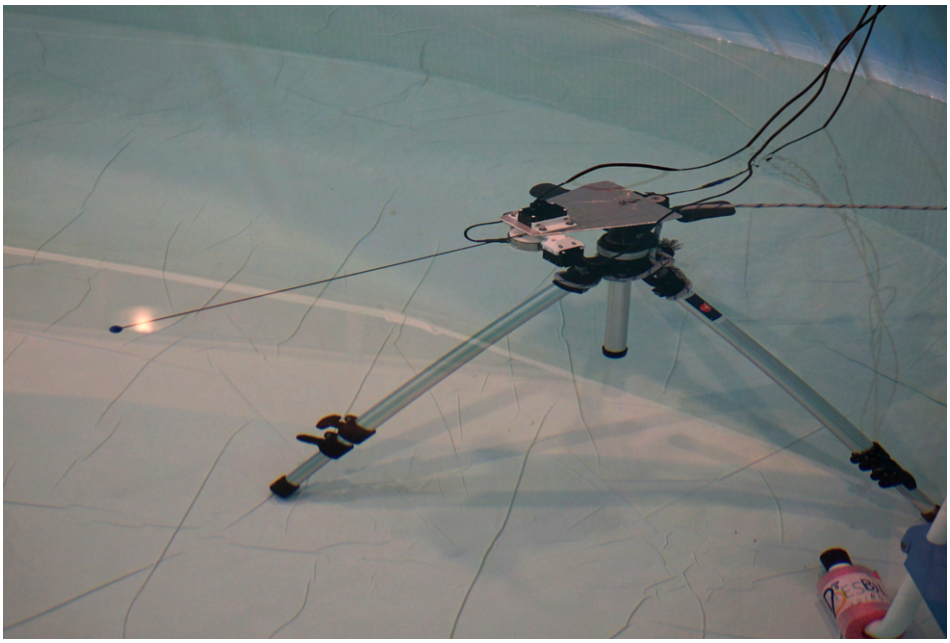

Figure 5.6: Antenna System in Water

To observe the behaviour of the system, a variable frequency signal with constant amplitude was used, as in the previous test. However, taking into account the limitations of the acquisition system, which can only reach a sampling frequency of 77Hz, a value of 30Hz was chosen for the frequency of the chip, at which we will only be able to see the first resonance peak of the system. However, this result is optimal for modelling the system as it is not necessary to reach a very high frequency. As in the following test, the input system was created, i.e. the chirp at a maximum frequency of 30 Hz, the signal is shown in the figure.

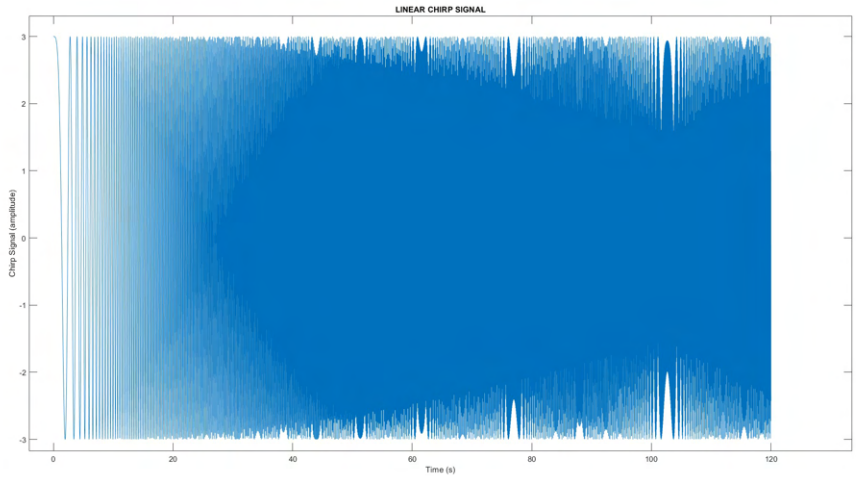

Figure 5.7: Linear Chirp Signal

Let us evaluate the results obtained from the system, i.e. the encoder and force-torque sensor data

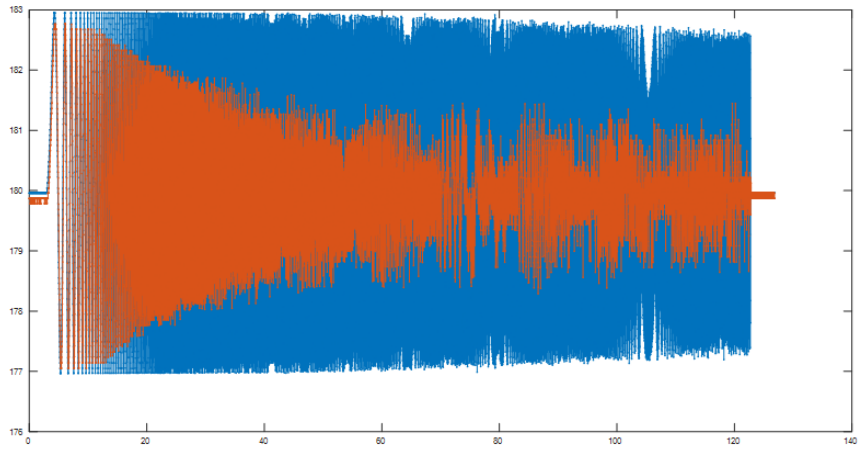

Figure 5.8: Motor Response to Chirp Signal

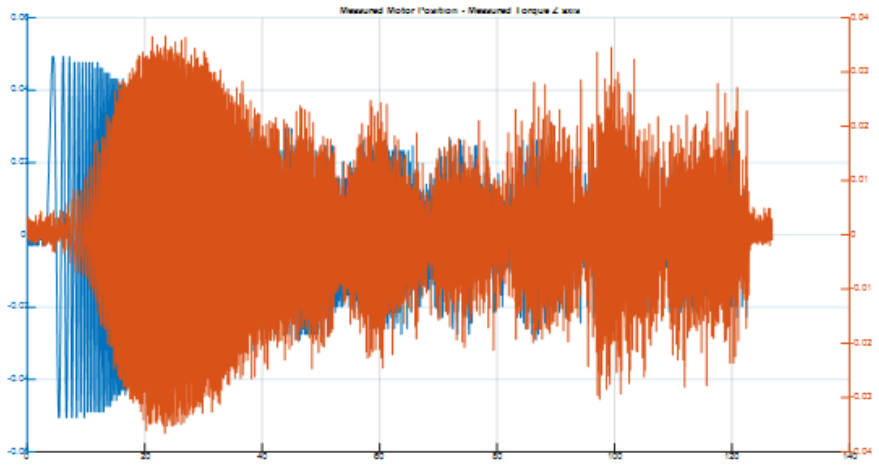

Figure 5.9: Data Response of Moment on Z Axis and Motor Position

The data obtained from the force-torque sensor show that movement in water tends to greatly reduce the force exerted by the antenna on the sensor, as the water reduces the vibrations during movement and therefore the rod is only affected by a deflection. These results confirm the assumptions made previously and allow us to understand the behaviour of the rod through the analysis of the bode diagram obtained, as in the previous test, by means of fft and welch[26][22].

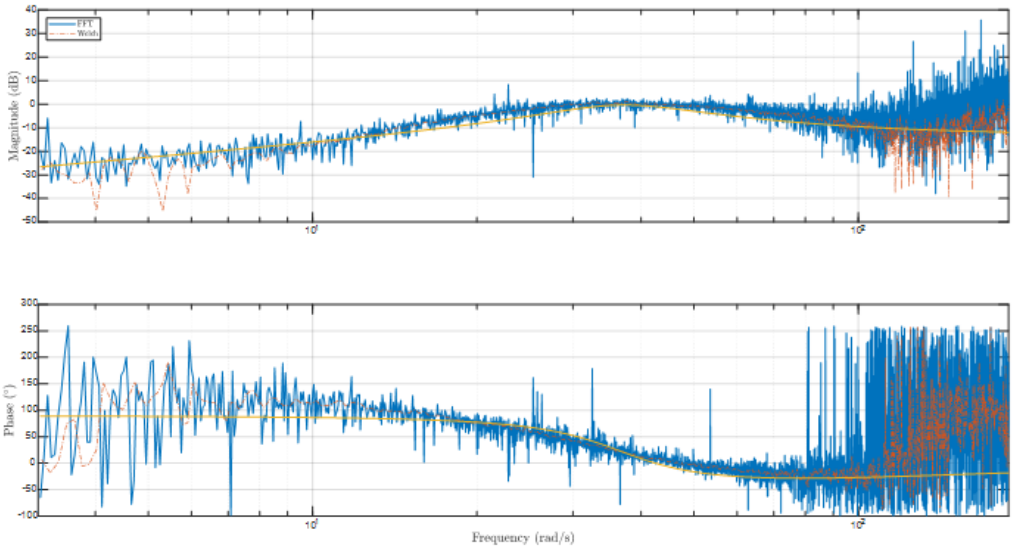

Figure 5.10: Bode Diagram Test Antenna in Water

From the Bode diagram we can see the attenuation given by the resistance in water, indeed we can see that there is still a slight resonance peak at $39.3 rad/s$ as in the previous test, but the value of the modulus is smaller. In this case, the estimate of a transfer function is as follows

$$\frac{\Gamma(s)}{\theta(s)} = \frac{0.002074s^2 + 0.2369s}{s^2 + 23.77s + 1498} \tag{5.2}$$

This linear model has shown to provide a good approximation to the antenna dynamics in the case of low velocities, as is it will be our operating mode, in which the antenna will be slowly tapping the wall of the working environment. Therefore, this model does not describe the behaviour of the antenna under multiple operating conditions, since we know that damping in water is non-linear and depends on the square of the velocity.

5.4 Matlab Simulation of the Simplified Model

As we saw in the chapter on modelling the system, we arrived at modelling a simplified system of ROV and antenna in a contact state. In order to validate the mathematical modelling, i.e. to verify that the equations developed can actually represent a working state of the system, a Simulink/Matlab simulation was created to verify this. Below is the Matlab code used to set the various parameters and the complete model in Simulink.

```
% SIMULATION BLUERov2 + Flexible Antenna Simplified Model %
clear
clc
close all
% Paramameters
% Antenna
L = 0.491; % m  Length of flexible link
E = 1.15e11; % Young module of antenna
I = 7.85e-13; % Inertia of antenna
m_l = 0.0024; % Mass of link
m_s = 0.00117; % Mass of Sphere
d = 0.00197; % Diameter of antenna
c = (3*E*I)/L;

% Motor
s = tf('s');
Kp_p = 900/128;
G = (-8.13*s + 1597)/(s^2 + 68.11*s + 1597);% Position Control TF
G_1 = (G)/((1 - G)*Kp_p); % Velocity Control TF
[num den] = tfdata(G_1, 'v');

% Blue Rov2
h = 0.3; % Distance fron COM to position of motor link
% Initial position
x_c = 0;
y_c = 0;
varphi = pi/2; % rad [0,2*pi]
theta = pi/2;  % rad [pi,(3/2)*pi]
t_s = 0.001;
t_f = 1;
out = sim('Simulink_simplified_model',t_f);
```

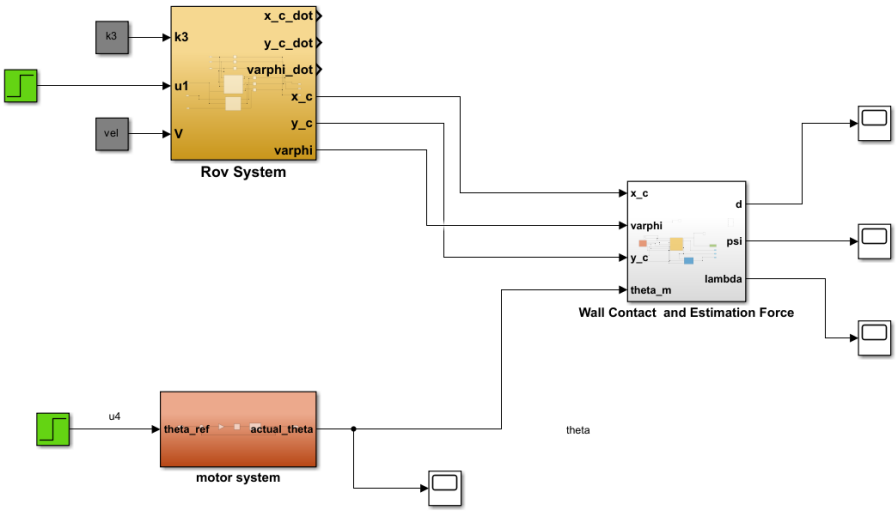

Figure 5.11: Complete Model Simulation

From the equations written in the simplified model, we assume that some variables, such as the torque exerted by the rod on the motor system and the bending of the rod, are estimated by the data obtained from the sensors. In the simulation it is not possible to obtain this data, so we go to obtain the data of the rod bending from the constraints imposed on the contact with the wall, going to solve the equations we can estimate the value of the torque obtained at the moment of contact. In this way we construct an external block that simulates the contact with the wall.

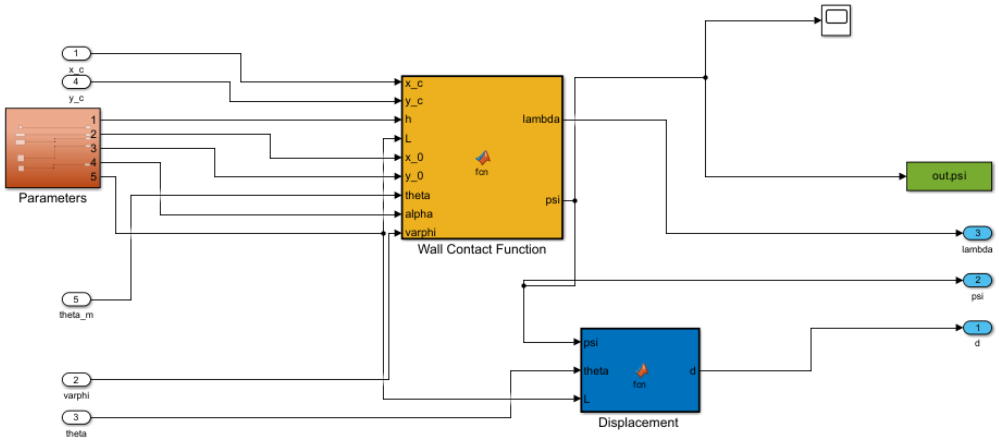

Figure 5.12: Contact with Wall Simulation

In the figure above we can see how, from the parameters and variables obtained, we obtain the values of λ and ψ, from which we obtain the displacement of the antenna tip with respect to the motor reference system. The developed function is as follows

```
function [lambda,psi] = solve_psi(x_c,y_c,h,L,
                                  x_0,y_0,theta,alpha,varphi)

lambda = sym("lambda");
```

```
psi = sym("psi");
out = zeros(1,2);
eq1 = x_c + h*cos(varphi) + L*cos(varphi+theta ...
            ...+psi)==lambda*cos(alpha)+x_0;
eq2 = y_c + h*sin(varphi) + L*sin(varphi+theta ...
            ...+psi)==lambda*sin(alpha)+y_0;
Y = vpasolve([eq1,eq2],[lambda,psi]);
lambda = double(Y.lambda);
psi = double(Y.psi);
```

Use the equations :

$$x_c + h * cos(\varphi) + h * cos(\varphi + \theta + \psi) = g_x(\lambda) \tag{5.3}$$

$$y_c + h * sen(\varphi) + h * sen(\varphi + \theta + \psi) = g_y(\lambda) \tag{5.4}$$

In this case, it is not possible to use the values measured by the sensors in the simulation, so knowing all the other parameters and variables such as x_c, y_c, φ, h and θ, we are able to obtain the values of λ and ψ through the result of a system of equations with two unknowns during the simulation. However, this simulation is very simplified and the initial conditions were set by considering the distance of the ROV from the wall and then checking that the ROV position actually maintains contact with the wall. Therefore, by considering a condition where the ROV moves along the y-axis, oriented at 90 degrees, and the wall is parallel to the movement of the ROV, we will check whether with the equations set we obtain decent results for the system considered. The results are as follows

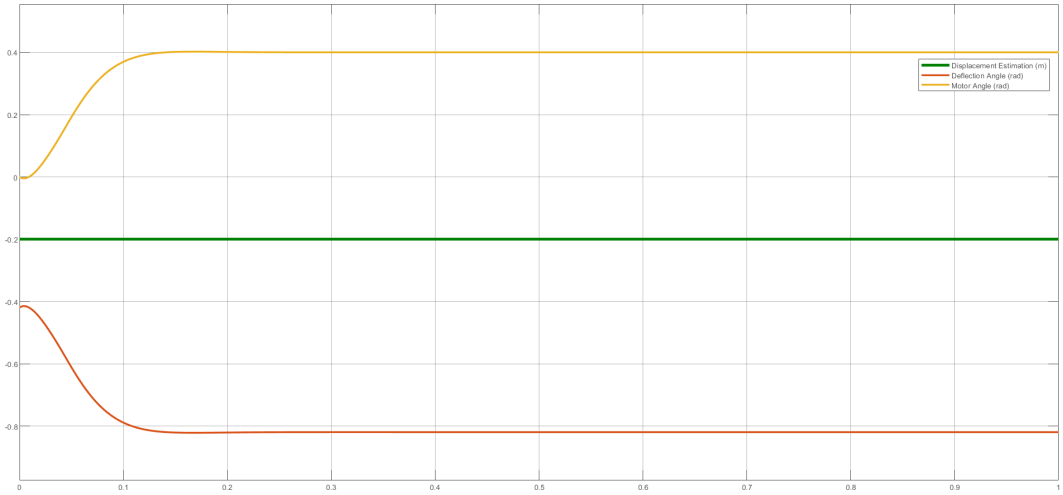

Figure 5.13: Result Motor Angle - Antenna Displacement - Antenna Deflection

As we can see, the displacement of the tip (green curve) of the antenna is kept constant, which means that contact with the wall is maintained. The other variables, such as motor angle and rod deflection, are kept constant after a short transient given by an external input, without varying, which means that given the ROV's movement, the equations are able to decipher a correct functioning of the model, even if not really validated.

Chapter 6

Conclusion and Future Work

In conclusion, a prototype haptic system for an underwater robot was studied, developed and implemented which, as already seen, consists of a flexible robotic antenna built at the host university. In the project, tests were carried out to observe the properties of an antenna in a different environment such as the freshwater, and thus simulate the behaviour of the antennae of mammals and crustaceans when navigating in low-light environments and thus without the aid of sight. The purpose of the system created is to inspect the seabed through the use of an antenna. For this purpose, several issues were addressed, which included modelling, controlling the dynamics of the ROV and the development of the entire motion capture system to ensure the robot's position feedback, as well as the design of the flexible antenna acquisition and control system and its implementation on the ROV. Finally, it was essential to build a model of the entire system and understand how to handle the interaction of internal and external forces with the environment.

Future developments are manifold, as the project is still in the completion phase. Starting with the system's hardware components, the on-board electronics could be improved in the future, improving the system's criticalities and optimising the on-board firmware to achieve a much more robust system. With regard to the software and the management of all system devices, the position acquisition system and the entire network between devices should be tested and updated. From the control point of view, it is necessary to revise the model of the ROV by obtaining new system parameters because tests carried out on Solidworks showed that the presence of a component in the lower part of the ROV could affect the movement in the water.

Bibliography

[1] Datasheet adc converter. https://www.mouser.es/ProductDetail/Analog-Devices/DC682A?qs=ytflclh7QUU1sdPvjEFblQ%3D%3D.

[2] Documentation arduino. https://www.javatpoint.com/arduino-ide#:~:text=The%20Arduino%20IDE%20is%20an,languages%20C%20and%20C%2B%2B.

[3] Documentation ardusub. https://www.ardusub.com/introduction/required-software/ardusub-autopilot-firmware.html.

[4] Documentation bluerov2 : https://bluerobotics.com/store/rov/bluerov2/.

[5] Documentation c++. https://en.wikipedia.org/wiki/C%2B%2B.

[6] Documentation fathomx. https://eu.robotshop.com/products/fathom-x-tether-interface-board-set/.

[7] Documentation mavlink. https://mavlink.io/en/.

[8] Documentation mavproxy. https://ardupilot.org/mavproxy/.

[9] Documentation motor dynamixel. https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/.

[10] Documentation opencm expansion board. https://www.mybotshop.de/ROBOTIS-OpenCM-485-Expansion-Board_1.

[11] Documentation opencm9.04. https://emanual.robotis.com/docs/en/parts/controller/opencm904/.

[12] Documentation pixhawk. https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk.html/.

[13] Documentation pymavlink. https://mavlink.io/en/pymavlink.

[14] Documentation python. https://www.python.org/doc/essays/blurb/.

[15] Documentation qualisys. https://odr.chalmers.se/server/api/core/bitstreams/8e81b559-ccb7-40c6-bd7f-e29be2759f48/content.

[16] Documentation qualisys sdk: https://github.com/qualisys/qualisys$_cpp_sdk$.

[17] Documentation ros. https://www.ros.org/.

[18] Documentation rpi3+: https://www.pololu.com/product/2759.

[19] Documentation sensor force-tourque. https://schunk.com/it/it/tecnologia-di-automazione/sensore-di-forza/coppia/ft/ftd-mini-40-si-20-1/p/EPIM_ID-30832.

[20] Documentation spi protocolo : https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all.

[21] Documentation virtualhere. https://www.virtualhere.com/client_api.

[22] Welch method : https://ccrma.stanford.edu/ jos/sasp/welch_s_method.html.

[23] Wu, c. 6-dof modelling and control of a remotely operated vehicle. master's thesis, flinders university, adelaide, australia, 2018.

[24] Yang Cong, Changjun Gu, Tao Zhang, and Yajun Gao. Underwater robot sensing technology: A survey. *Fundamental Research*, 1(3):337–345, 2021.

[25] A. Fayazi, N. Pariz, Ali Karimpour, Vicente Feliu, and Hassan Hosseinnia. Adaptive sliding mode impedance control of single-link flexible manipulators interacting with the environment at an unknown intermediate point. *Robotica*, 38:1–23, 11 2019.

[26] A.C. Fischer-Cripps. 3.5 - digital signal processing. In A.C. Fischer-Cripps, editor, *Newnes Interfacing Companion*, pages 269–283. Newnes, Oxford, 2002.

[27] Thor Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. 04 2021.

[28] Sven Lack, Erik Rentzow, and Torsten Jeinsch. Experimental parameter identification for an open-frame rov: Comparison of towing tank tests and open water self-propelled tests. *IFAC-PapersOnLine*, 52(21):271–276, 2019. 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2019.

[29] Malte von Benzon, Fredrik Fogh Sørensen, Esben Uth, Jerome Jouffroy, Jesper Liniger, and Simon Pedersen. An open-source benchmark simulator: Control of a bluerov2 underwater robot. *Journal of Marine Science and Engineering*, 10(12), 2022.

[30] X Yang and Y Xing. Tuning for robust and optimal dynamic positioning control in bluerov2. *IOP Conference Series: Materials Science and Engineering*, 1201:012015, 11 2021.