Motion control and Wi-Fi threat modelling analysis of small and affordable underwater ROV

Ethically hacking Chasing Dory and testing/optimizing a simple PID control on a underwater ROV model.

Raymond B. Cyiza

Thesis submitted for the degree of
Master in Applied Computer Information Technology(ACIT)
30 credits

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

Spring 2022

Motion control and Wi-Fi threat modelling analysis of small and affordable underwater ROV

Ethically hacking Chasing Dory and testing/optimizing a simple PID control on a underwater ROV model.

Raymond B. Cyiza

Motion control and Wi-Fi threat modelling analysis of small and affordable underwater ROV

http://www.oslomet.no/

Printed: Reprosentralen, Oslo Metropolitan University

Abstract

In this thesis we cover underwater ROVs open Wi-Fi access attacks, threats and vulnerabilities using ethical hacking techniques between passive and active information gathering. The information gathered is then compared to the STRIDE model, along with the OWASP 2021 Top 10 list, concluding with an assessment of risk analysis based on the CVSS(Common Vulnerability Scoring System) scores. Chasing Dory and a ROV model with similar topology are used in this project for threat modelling, hydrodynamic and hydrostatic analyses demonstrated with ANSYS, Simulink and Matlab software tools. The PID tuning model used for the ROV model is a simple PID controller based on a continuous time domain transfer function.

Contents

List of Figures

List of Tables

Preface

The motivation behind this thesis was to challenge our practices and theory learned in information security, robotics and control interdisciplinary fields. Where in the beginning phase of the project we defined the project goal of reverse engineering Chasing Dory's phone application and gather it's log files containing IMU, Sensor and Video transmission data, to further create a python script that controls the ROV. Due to the challenges involved in attaining resources, time constraint and difficulty on decrypting the collected log files, we decided to redefine our problem statement, presented in Part I, The Project. I want to especially thank my supervisor Alex Alcocer who guided me and advised on ideas to explore on Robotics and Control, including other professors from Information Technology and Data Security Departments, Aws Naser Jaber Alzarqawee and Lothar Fritsch. I also want to thank my friends, colleagues and family who along the way kept my morale up with great support.

Chapter 1

Introduction

Marine robots today are often used to conduct high risk underwater operations to gather specific data measurements, such as the temperature and salinity of the water, or the speed and direction of currents on unexplored ocean areas. Marine robots can conduct these experiments by either being remotely operated or autonomously controlled [28]. In other sea environments, such as the oil and gas industry, we also find use of underwater robots, but more often operating with the mission to decrease the amount of dangerous, dirty or dull operations from humans, and increase data quality collection [21]. As from the consumer end point, most affordable marine robots are more likely to be designed for exploration purposes, where users mainly use their underwater robots for collecting items at sea shore, documenting sea diving experiences or other sea related activities that satisfy sea enthusiasts. On the underwater drone forum one can usually find many of these shared sea experiences, ranging from underwater drone reviews provided by some of the globally known underwater drones companies such as Chasing Innovation, QYSEA and Blue Eye Robotics among the other less known, to guides on disassembling, modifying or repairing underwater ROVs [2].

Chasing Innovation is a technology company based in Shenzhen that develops and manufactures portable remotely operated and unmanned underwater vehicles, each with its own model design and specifications. All of Chasing's ROVs are able to dive deep underwater and perform simple to high demanding exploration missions(i.e Chasing M2) or data collection(i.e. Chasing F1). For our project we have been fortunate to be provided with Chasing Dory, the most affordable drone from their products, which comes with a 1 hour run time and can dive down to 15 meters underwater. While for instance in [6], we find the only case where Chasing Dory along with Chasing M2 are used to monitor the underwater sea environment to, gather, analyze and transmit data using a defined architecture model based-on edge computing. The edge IoUT architecture operates with Underwater Wireless Sensor Networks that use autonomous driven models to reduce energy consumption, monitoring and searching of data, while the Switch Device Network increases their ability to scale. i.e increase data

storage. Figure 1.1 shows the setup process of a search and monitoring case. However, more relevant research in the marine robotics field where use of Chasing Dory or other Chasing underwater drones seems to be limited.

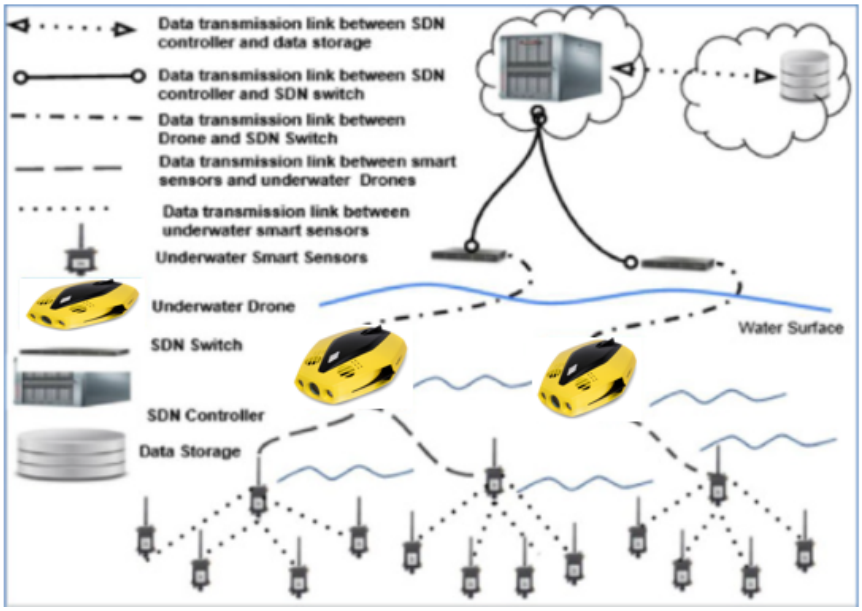


Figure 1.1: From [6], shows the data gathered from Underwater Smart Sensors to the ROVs, the ROVs are linked to the Software Defined Network(SDN) switches, where transmission of motion maneuvering and data collection are exchanged. The data gathered then is transmitted to the SDN controller with transmission link to the data storage.

As modern robotics keep growing across every field, especially in the marine environment, more online applications, services and software are included as drivers of the social and economic development. Most affordable drones whether they are designed for sea-, air- or ground-environments make use of open WiFi access to allow simple connections, and to establish easy communication when offering real-time data transmission, including remote or autonomous control. This leads to the fundamental importance of secure WiFi connections, which companies and end-users can exploit to avoid possible threats, vulnerabilities or attacks to prevent halt of production process, bad user experience or critical underwater operations. Safe communication and maneuvering of marine robotics entails reliable control and navigation, security protocols, and human interaction safety, which for instance opens possibilities to execute missions on large ships where finding corrosion, cracks and other serious problems can easily become difficult to perform underwater by humans. For instance in [13], a provided remotely controlled underwater drone via WIFI with the name SIR(ship inspector robot), is a lightweight robot with four magnetic wheels that can navigate on the bottom of a ship and execute the previously

named operations much more safe and effective.

In addition to operation and communication layers, reliable navigation and control must be approved and tested from the modeling and simulation phases. In modeling and simulation, software related experiments are conducted to understand the underwater robot systems behavior and give insights on possible model optimizations to produce a well functioning prototype. After the development process, ROVs are released for consumers and others to purchase and use. In most cases, we find ROVs used in the off-shore industry, i.e subsea pipeline structures constructions, ship recovery missions, naval mine hunting, fisheries science and other applications, where their durability, financial use, and safety, can with confidence replace human divers. For any tasks involving manipulation and maneuverability in offshore operations, ROVs usually come as the best cost-effective platform [21].

Part I

The project

In this project the initial problem statement was to demonstrate if it was possible to access Chasing Dory´s video transmission, position and navigation log files, to further use to create an alternative navigation and control system to run in python. In order to do this we followed an ethical reverse engineering process, where we would inspect Dory's Wi-Fi properties, attempt pen. tests(penetration tests) and writing of a python script that covers the communication protocols and control navigation functions. Due to the difficulty, amount of progress and time limit given, we redefined our problem statement to one that could fit within the time limit left and also allow opportunity of running tests and simulations. With that in mind, we chose to have our second part of this thesis contain a motion behavior analysis of an ROV using a simple PID control system. In Chapter 2 we present literature background on kinematics of ROV, use of PID controllers and fundamentals of OSI model, which are necessary for understanding use of ethical hacking techniques. In Chapter 3 we present the methodology of the mathematical modelling and threat analysis behind the ROV. Chapters 4 and 5 include the obtained results from code design, simulation design and PID tuning, which is followed by the discussion on the obtained results and conclusions section for further work.

Chapter 2

Literature Review

The literature section covers basic theory and background information about the different technologies used in order to understand the methodology of motion, control and ethical hacking analyses, in section 2.1, 2.2, 2,3 and 2.4, respectively. The presented theory and equations are derived from The Handbook of Marine Craft and Motion Control(2011) by Fossen [16].

2.1 ROV in general

Remotely underwater vehicles are usually known as ROVs and in some cases can be called unmanned vehicles controlled with the help of a tether. The tether, commonly referred to as an umbilical cord, maintains communication of information between a computer device and the ROV. The tether also provides electrical power, however other power solutions such as batteries are occasionally used in small ROVs instead. Initially, ROVs were developed mainly by the US military in the 1960s for recovery operations. While in the 1970s, development of ROVs started taking notice in the offshore oil and gas industry. Ever since, research, academics, and marine agriculture ROVs have increased in numbers, as well as functionality. Typical sensors and tools for industrial scale ROVs for instances we see consist of IMUs, positioning systems, depth sensors, magnetometers, cameras and in some cases robotic manipulators [30].

2.1.1 Kinematics

In the study of ROV dynamics, analysis of forces can be divided into two categories, the kinematics part where geometrical aspects of motions are treated, and the kinetics, where forces creating the motion are analyzed. The general marine craft of motions in 6-DOF according to Fossen [16], the handbook of marine craft hydrodynamics and motion control (2011):

$$\dot{\eta} = J_\theta(\eta)v \tag{2.1}$$

$$M\dot{v} + C(v)v + D(v)v + g(\eta) + g_0 = \tau + \tau_{wind} + \tau_{wave} \tag{2.2}$$

where in equation (2.1) has $\eta = [\eta_1 \eta_2]^T$ representing position ($\eta_1 = [x, y, z]^T$) and orientation ($\eta_2 = [\phi, \theta, \psi]^T$) in Euler angles in the North-East-Down(NED) frame. $\nu = [\nu_1 \nu_2]^T = [u, v, w, p, q, r]^T$ representing the linear and angular velocities in the BODY frame. And $J_\theta(\eta)$ is the Euler rotation transformation matrix from the Body frame to NED frame.

In equation (2.2) $M = M_{RB} + M_A \in \mathbb{R}^{6\times6}$ corresponds to the inertia mass matrix, of the rigid body and added mass respectively. The Coriolis and centripetal are represented by $C(v) = C_{RB}(v) + C_A(v) \in \mathbb{R}^{6\times6}$, and the linear and quadratic damping coefficients due to the surrounding fluid are denoted by $D(v) \in \mathbb{R}^{6\times6}$. $g(\eta) + g_0 \in \mathbb{R}^6$ represents the gravitational and buoyancy vector. The generalized vector due to hydrodynamics and hydrostatic forces are represented by τ, and the environmental forces are represented by τ_{wind} and τ_{wave}, which are typically negligible below 10m [16].

For sea crafts moving in 6DOF, 6 independent coordinates are mandatory to determine the orientation and position. The first 3 coordinates, including their time derivatives, specify the position and translational motions along the x, y and z axes, while the last 3 coordinates(ϕ, θ, ψ) define the orientation and rotational motions. These six different motion variables are often described as surge, sway, heave, roll, pitch and yaw.[16]. As seen in figure (2.1) and the SNAME(1950) notation in table 2.1.

Figure 2.1: 6DOF velocities u, v, w, p, q, and r in the body-fixed reference frame b=(x_b, y_b, z_b).

2.1.2 Reference frames

When conducting the motion of an underwater craft in 6DOF, it is important to describe an earth-centered frame and/or a geographic

reference frame. The Geocentric(earth central inertial) coordinate system {i}= (x_i,y_i,z_i) is an inertial coordinate system used for terrestrial navigation, a non accelerating reference coordinate system to which the Newtons laws of motion apply. The origin is at the center o_i of the earth and its axis is as shown in figure (2.2). The Fixed-Earth Geocentric(Earth-centered Earth-fixed), reference frame {e}= (x_e,y_e,z_e) with has o_e as the fixed origin at the center of earth, but the axis rotated with respect to the Earth center inertial frame in space. The angular speed of rotation is $w_e = 7.2921x10^{(-5)}rad/s$. For marine vessels moving at relatively low speeds, the earth's rotation is negligible. For drifting marine vessels however, the earth's rotation should not be ignored [16].

Table 2.1: The SNAME(1950) notation for marine crafts taken from [18].

DOF	Forces and moments	Linear and angular velocites	Position and Euler angles
1 motions in the x direction(surge)	X	u	x
2 motions in the y direction(sway)	Y	v	y
3 motions in the z direction(heave)	Z	w	z
4 rotations about the x axis(roll,heel)	K	p	ϕ
5 rotations about the y axis(pitch,trim)	M	q	θ
6 rotations about the z axis(yaw)	N	r	ψ

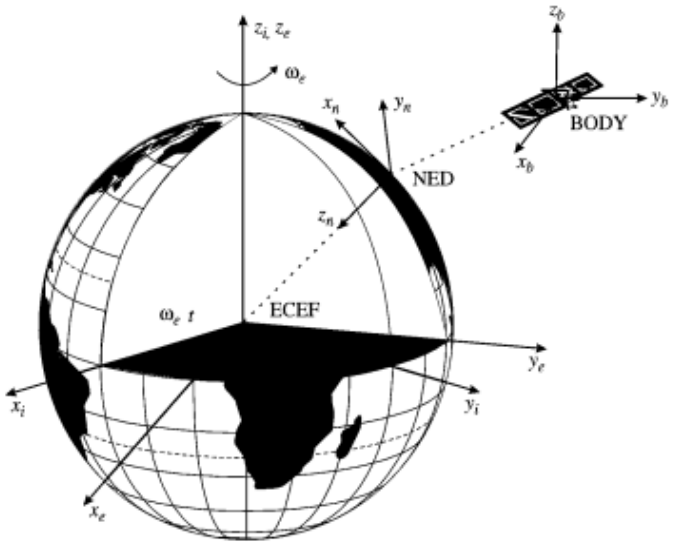


Figure 2.2: From [16], shows the Geocentric Earth fixed(ECEF) coordinate system x_e,y_e,z_e rotating at the angular rate w_e relative to the geocentric inertial (ECI) frame x_i,y_i,z_i fixed in space.

The Geographic reference frame addressed in this paper refer to the North-East-Down(NED) coordinate system {n} = (x_n, y_n, z_n) with its origin defined with respect to Earths reference ellipsoid WGS84 (Word Geodetic System, 1984). This coordinate system is usually one often implemented in hydrodynamics and hydrostatic problems, whether it is from sea related tasks, to air or ground. The NED reference frame is a tangential plane to the earth's surface moving with the sea vessel, with the x axis pointing towards the true North, the y axis points towards East, while z axis points downwards normal to the Earth's surface. The location of the coordinate system {n} relative to Earths coordinate system {e} is chosen by using two angles L and μ that denote the longitude and latitude values, respectively. For a sea vessel moving in a local area, with an approximate constant longitude and latitude, a tangential Earth-fixed plane on the surface can be utilized for navigation(Flat Earth navigation). For flat earth navigation one can assume that {n} is inertial, such that a craft or an object in motion remains in motion, while a craft at rest remains motionless unless acted upon by external force, where Newton's laws apply [16].

The body-fixed reference frame $b = (x_b, y_b, z_b)$ with origin o_b is defined as a coordinate frame in motion fixed to the sea vessel. The position and orientation of the vessel are formulated relative to the inertia reference frame, approximately extracted from the earths {e} and north-east-down {n} coordinate system for sea vessel, respectively, while the linear and angular vectors of the vessel are expressed in the body-fixed coordinate system {b}. The origin o_b is often determined to be concurrent with a point midships in the water line. This point is be denoted as CO as seen in figure (2.3). For a sea vessel, the body axes x_b, y_b and z_b are determined to match with the

principal axes of inertia, and are defined by [16]:

x_b – longitudinal axis(directed in positive rear direction and negative front direction)
y_b – transversal axis(from center of Dory to right direction)
z_b – normal axis(directed in positive bottom direction and negative top direction)

while the body-fixed reference points with respect to CO are defined as:

CG – Gravity center
CB – Buoyancy center
CF – Flotation center(located a distance LCF from CO in the x-direction)

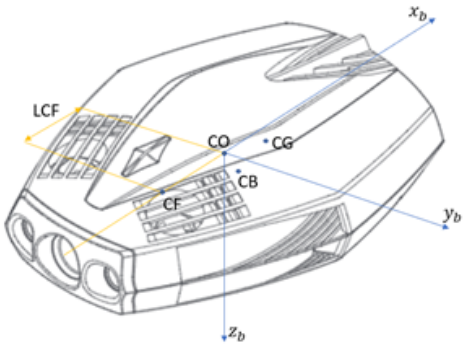


Figure 2.3: Body-fixed reference points.

2.1.3 Transformations between Body and NED

In a rotation matrix R we denote R_b^a as a matrix between two reference frames a and b, while also as an element in the special orthogonal group with order 3, SO(3):

$$SO(3) = \{\boldsymbol{R} \mid \boldsymbol{R} \in \mathbb{R}^{3\times 3}, \boldsymbol{R} => orthogonal, and \det \boldsymbol{R} = 1\} \tag{2.3}$$

which implies that R is orthogonal, i.e the dot product of the rotation matrix and its inverse rotation matrix correspond to an identity matrix, consequently making the inverse rotation matrix to be given by $R^{-1} = R^T$. In navigation, control and guidance, an often-used rotation matrix R_b^n between north-east-down frame {n} and body-fixed frame {b} when derived, make use skew symmetry of a matrix, cross-product operator, simple rotation and Euler's theorem on rotation properties [16]. For a skew-symmetry matrix $S \epsilon SS(n)$, which is the set of skew-symmetric matrices of order n, is defined to be skew-symmetrical if :

$$\boldsymbol{S} = -\boldsymbol{S}^T \tag{2.4}$$

Thus implying that the non-diagonal variables of S satisfy $s_{ij} = -s_{ji}\ for\ i \neq j$, while the diagonal variables equal to zero. While a vector cross-product operator (x) is defined by:

$$\boldsymbol{\lambda x a} := \boldsymbol{S}(\boldsymbol{\lambda})\boldsymbol{a} \tag{2.5}$$

where for example $S\ \epsilon\ SS(3)$ is defined as

$$\boldsymbol{S}(\lambda) = -\boldsymbol{S}^T(\lambda) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \tag{2.6}$$

A simple rotation is referred to as the motion of a rigid body relative to a reference frame A or the motion of the same rigid body relative to a reference frame B [16].

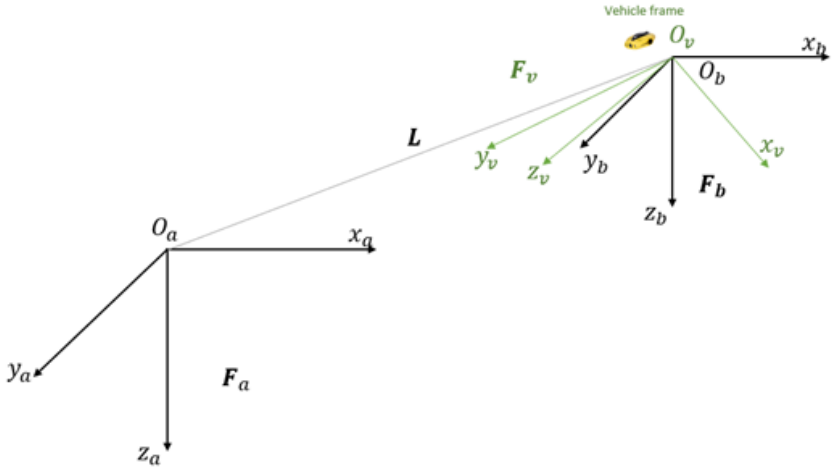


Figure 2.4: Simple rotation motion represented with reference frames A and B.

.

Based on the simple rotation definition, Euler also stated in (Euler, 1776) from [16], that the rotation theorem of two reference frames or rigid bodies, where every change in the relative vector orientation of reference frames A and B, or two rigid bodies, can be created applying use of a simple rotation of B in A. So while assuming we have a vector in the body-fixed frame and a vector in the north-east-down frame, with the unit vector $\lambda = [\lambda_1, \lambda_2, \lambda_3]^T, \lambda = 1$, parallel to the axis of rotation and β the angle the north-east-down frame is rotated, the rotation is thus expressed by:

$$\boldsymbol{R}_{\lambda,\beta} = \boldsymbol{I}_{3x3} + \sin(\beta)\boldsymbol{S}(\lambda) + [1 - \cos(\beta)]\boldsymbol{S}^2(\lambda) \tag{2.7}$$

where I_{3x3} is the identity matrix and $S\ (\lambda)$ is the skew symmetric matrix as defined in (2.4). When addressing Euler angle transformation equations,

Euler angles roll(ϕ), pitch(θ) and yaw(ψ) are added and are possible to use when decomposing the fixed BODY frames velocity vector $v^b_{\frac{b}{n}}$ in the north-east-down frame, by letting $R^n_b\left(\mathbf{\Theta}_{nb}\right) : S^3 \rightarrow SO\left(3\right)$ denote the Euler angle rotation matrix with the argument $\mathbf{\Theta}_{nb} = \left[\phi, \theta, \psi\right]^T$ [16].

2.1.4 Gravity and buoyancy of submerged vehicles

For a submerged craft, gravitational and buoyancy forces are determined by the volume space of the occupied fluid, the area of the water plane, the water planes associated moments and the center of buoyancy. In hydrostatic terminology, gravitational and buoyancy forces are referred to as the restoring forces. While considering a submarine as seen in figure(2.5), the gravitational pull force f^b_g will act through the center gravity(CG) defined by the vector $r^b_g = \left[x_g, y_g, z_g\right]^T$ with respect to center origin(CO). In similar ways, the buoyancy pull force f^b_b will act through the center buoyancy(CB) by the defined vector $r^b_b := \left[x_b, y_b, z_b\right]^T$ where both vectors refer to the reference point CO of the body-fixed frame [16].

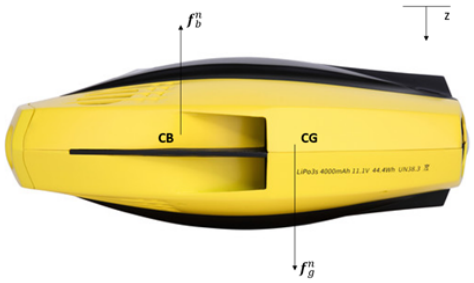


Figure 2.5: Restoring forces(gravity and buoyancy) that act on the CG(center gravity) and CB(center buoyancy) of Dory.

In a case where we let the mass of a submerged vehicle be m underwater, the volume of occupied fluid by the vehicle be ∇, the acceleration of gravity pointing downwards(positive) g and the water density ρ. The submerged vehicle weight of the body and buoyancy force according to the SNAME(1950) notation seen in table 2.1 can be written as:

$$\mathbf{W} = mg, \; B = \rho g \nabla \tag{2.8}$$

Since the forces acting in the vertical plane of {n} upon the submerged vehicle, we assume that:

$$f^n_g = \begin{bmatrix} 0 \\ 0 \\ W \end{bmatrix} \quad and \quad f^n_g = \begin{bmatrix} 0 \\ 0 \\ B \end{bmatrix} \tag{2.9}$$

While considering that the z axis is pointing positive downwards such that the gravity is positive and the buoyancy is negative. By implementing

the equation result from Euler angle transformation, the weight and buoyancy force can be denoted in the body-fixed frame {b} by:

$$\boldsymbol{f}_g^b = \boldsymbol{R}_b^n \left(\boldsymbol{\theta_{nb}}\right)^{-1} \boldsymbol{f}_g^n \tag{2.10}$$

$$\boldsymbol{f}_b^b = \boldsymbol{R}_b^n \left(\boldsymbol{\theta_{nb}}\right)^{-1} \boldsymbol{f}_b^n \tag{2.11}$$

where $R_b^n(\theta_{nb})$ is the Euler angle coordinate transformation matrix defined by the orientation unit vectors of the vehicles rigid body relationship to the body-fixed frame and the north-east-down frame. According to our specified coordinate system, the buoyancy, and gravitational forces and moments f_i^b and $m_i^b = r_i^b x f_i^b,\ i\,\epsilon\{g,\ b\}$ are expressed as:

$$\boldsymbol{g}(\boldsymbol{\eta}) = -\begin{bmatrix} \boldsymbol{f}_i^b \\ \boldsymbol{m}_i^b \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{f}_g^b + \boldsymbol{f}_b^b \\ \boldsymbol{r}_g^b x \boldsymbol{f}_g^b + \boldsymbol{r}_b^b x \boldsymbol{f}_b^b \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_b^n \left(\boldsymbol{\theta}_{nb}\right)^{-1} \left(\boldsymbol{f}_g^b + \boldsymbol{f}_b^b\right) \\ \boldsymbol{r}_g^b x \boldsymbol{R}_b^n \left(\boldsymbol{\theta}_{nb}\right)^{-1} \boldsymbol{f}_g^b + \boldsymbol{r}_b^b x \boldsymbol{R}_b^n \left(\boldsymbol{\theta}_{nb}\right)^{-1} \boldsymbol{f}_b^b \end{bmatrix} \tag{2.12}$$

When designing underwater vehicles it is convenient to have a slightly greater buoyancy than the weight of the submerged vessel(B >W :positive buoyancy) so that the vessel will naturally rise to the surface in case of an emergency situation, for instance power or engine failure. If the vessel happens to be designed so that B >> W, too much control energy is then required to keep the vehicle underwater [16].

2.1.5 Motion control

As mentioned previously in section 2.1.3, a motion control system is built upon three main non-dependent blocks, written as the guidance, navigation and control systems. These systems co-operate with each other through transmission of signals and data as seen in figure(2.6), where of a traditional sea craft is using autonomous navigation with a guidance system, operating based on alternative estimated positions and velocities. In more advanced guidance, navigation and control systems such as[23], its common to find tightly joined(coupled) or even control systems designed by one block. From an industrial perspective, tight and loose coupling is seen on as an indicator between high performance and modularity of component design. Although regardless of the differences, most advanced GNC blocks are often represented with three interconnected subsystems according to [16].

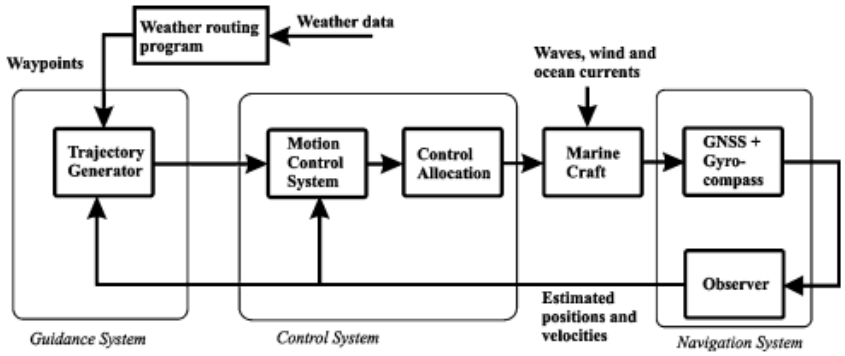


Figure 2.6: From [16], shows a simple guidance, navigation and control block simulation model.

Guidance, is a block in a system that continuously computes the desired position, velocity and acceleration of a marine vessel usually denoted to as the reference, and is used by the motion control system. The navigation data is often shared to a human operator operating with a navigation system. The fundamental values and devices of guidance system correspond of motion sensors, current speed and direction, and external data such as temperature, wind speed and direction, wave height and slope, operated through a computer. The computers task is to store and compute the given values, then send the result output to the motion control system. While in more advanced systems, optimization techniques and algorithms are implemented to plot the optimal path or trajectory for the sea vessel to trace and follow. Some known feature optimization techniques could be fuel consumption optimization, minimum time turning, weather passage planning, path planning(with collision avoidance), coordinated control for a fleet of robots and synchronization [16].

Navigation, is the block involved in directing a vessel by computing its position, course and distance values traveled. In some systems the required velocity and acceleration can as well be computed. The method implemented to perform the navigation task utilizes usually a global navigation satellite system(GNSS) co-operating together with mounted motion sensors, i.e accelerometers and gyroscopes [16].

The control block is used to determine the actual forces and moments needed by the vessel for it to maneuver as desired from the control objective. The block can also be referred to as motion control, where the desired control objective is usually obtained from the guidance system. For instance, known control objectives include minimization of energy, set-point regulation, path-tracking, trajectory-following and steering control. When building such a control model, often its design involves feedback and feedforward systems, where the outputs from the navigation system, position, velocity and acceleration are inserted as inputs in the feedback control system, while the feedforward control system is parametrized using signals

obtained in the guidance system and perhaps other external sensors [16].

2.1.6 PID-controllers

The control design model in its simplest form is utilized to determine stored parameter constant gains for a proportional, integral, derivative(PID) controller, which in the testing phase are implemented to create an optimal design of the motion control system. The PID regulator itsself requires two states, where one is defined for the integrator and one for the low-pass filter used to limit noise amplification. An exampled where common use of PID is observed is in Dynamic positioning(DP) systems. Dynamic positioning entails the capability of a vessel to orient its self on any point in a three dimensional defined coordinate system, using an integration of a variety of functions for motion and guidance systems through the mounted thrusters and propellers. Since the 1960s, Dynamic positioning systems have been available commercially for marine vessels, where the first DP systems were often designed using conventional PID regulators in conjunction with low-pass and/or notch filters to control the wave induced motion components. An example where use of a PID system for an underwater vehicle can usually consists of a control system that function based on a feedback control system providing the needed data to track the desired yaw angle $\phi(d)$ with the yaw moment(T_n) as the output [16].

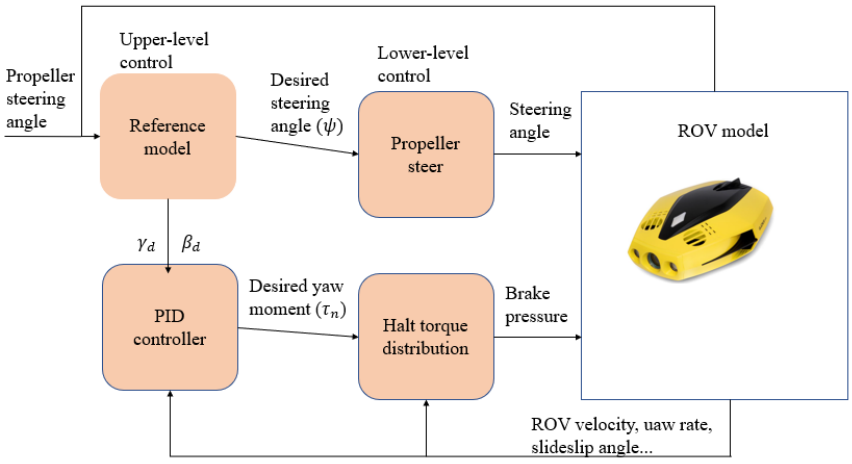


Figure 2.7: From [27], shows diagram of two-level yaw moment and propeller steering controller, where β_d is the minimum side-slip angle of the body, and γ_d is the desired yaw rate.

2.2 OSI Model

For computer systems and devices to communicate and share information interoperably, the Open System Interconnection(OSI) model is made of use.

The model was provided by the International Organization for Standardization(ISO) and defined in 1984, and became a way of subdividing a systems communication process into smaller layers. The layers consist of the application layer(7th), which provides users the ability to log on a system with a storage, where users can access files, retrieve and manage them, this includes forwarding of emails [26].

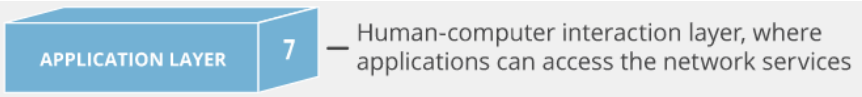


Figure 2.8: From [10], shows Layer 7.

The presentation layer(6th), is responsible for the translation of data received from the application layer(7), as the session layer(5th) provides users the ability to establish, maintain, synchronize and terminate sessions between end-user applications [26].

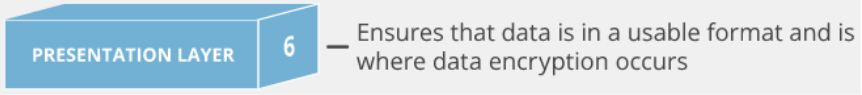


Figure 2.9: From [10], shows Layer 6.

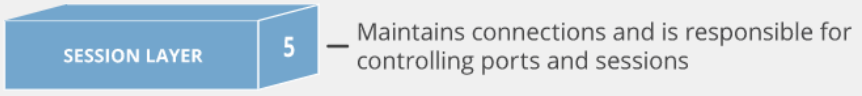


Figure 2.10: From [10], shows Layer 5.

The transport layer(4th) is responsible for transparent data transmission between end-users and reliable transmission to the upper layers(5, 6 and 7), by receiving data from the session layer and dividing it into smaller fractions to further send to the network layer[26].

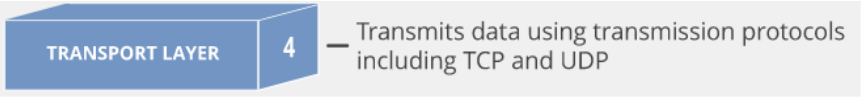


Figure 2.11: From [10], shows Layer 4.

The network layer(3rd) plays part in the transferring of data packets back and forth between different networks as part of the internet[26].

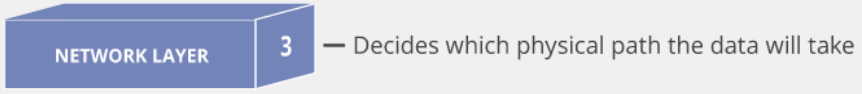


Figure 2.12: From [10], shows Layer 3.

While the last two layers, the data link layer(2nd) moves data into and out of the physical layer in a network, and the physical layer(1st) allows for connection between two or more physical objects[26].

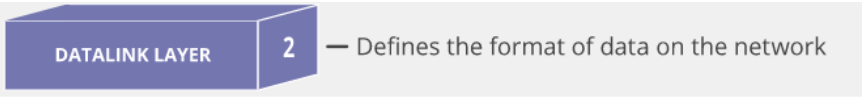


Figure 2.13: From [10], shows Layer 2.

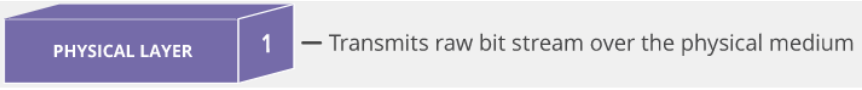


Figure 2.14: From [10], shows Layer 1.

2.2.1 TCP/IP and UDP

The internet protocol(IP) operates at the 3rd layer of the OSI model and is responsible for addressing and routing data packets to enable traveling between networks and arrive at the right destination. Once the IP is found the transmission control protocol(TCP) operating at the 4th layer of the OSI model, collects and reassembles data packets while ensuring reliable transmission of data. The packet format of the IP packet consists of an IP header and an IP data, where the TCP segment placed along the header contains information on the source port number, destination port number and checksum and the data segment. The transmission process of the TCP/IP packets begins by establishing a connection using a three-way handshake where the first device or a computer acting as a client sends a SYN(i.e. synchronize?) packet to start the connection, and the second device or computer acting as a server sends back a SYN/ACK. The client computer then replies with an ACK(i.e. acknowledge!) to finish the handshake and establish a connection [26].

The user datagram protocol(UDP) also operates at the transport layer(4) and is a much simpler alternative to TCP, however it provides unreliable connection between hosts as there are no guarantees for delivering the packets, but allows for speed and multi-casting. Similar to the TCP, its packet format that is sent over IP contains a header and data frames in the UDP segment with information on the source port number, destination port number, segment length and checksum [26].

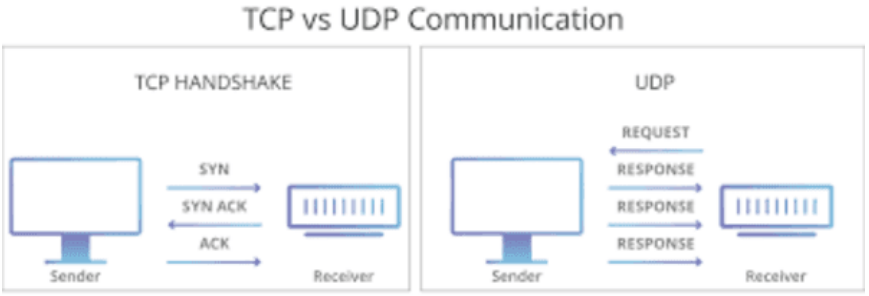


Figure 2.15: From [20], shows TCP/IP and UDP communication process.

2.2.2 WIFI(802.11)

At the data link layer(2) and physical layer(1) we have the IEEE 802.11 standard for wireless area networks(WLAN) which is a part of the family of standards IEEE 802, a collection set that covers physical and data link specifications for technologies such as Ethernet, which include local area networks(LAN) and metropolitan area networks(MAN). The IEEE 802.11 standard uses various bands of frequencies depending on the protocol type(n,g,b,a and ac), ranging from 2.4 GHz up to 5.9 GHz, where each frequency band is divided into a multitude of channels. Countries follow their own regulations when applying the allowable channels, allowed users and maximum power levels within the frequency ranges. In Japan the use of 14 channels is permitted in the 2.4GHz band, while countries like Spain supposedly started with only use of channels 10 and 11. Europe and Asia allow channels 1 through 13. North America and some central and south American countries allow channels 1 through 11. Channels 1, 6, and 11 were previously the non-overlapping channels, but with the new 802.11g standard channels 1, 5, 9 and 13 are now the non-overlapping ones [32].

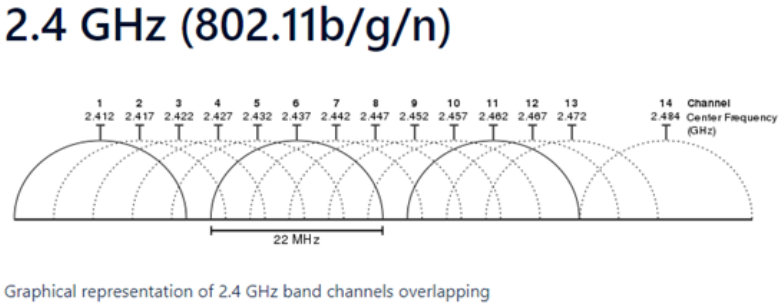


Figure 2.16: From [17], shows graphical representation of 2.4 GHz band channels overlapping.

2.2.3 ARP

Also at the link layer(2) the address resolution protocol(ARP) is a procedure that occurs when connecting an ever-changing Internet Protocol (IP)

address to a fixed physical machine address, also known as a media access control(MAC) address, in a local area network (LAN). As IP and MAC addresses often differ in length, a translation using the ARP process is needed so that systems can recognize one another. Today most IP addresses used are from the IPv4 which is 32 bits long, while MAC addresses are 48 bits long. At the link layer, ARP helps translate the 32 bit-address to 48 bits including the other way [32].

2.3 WEP,WPA,WPA2/WPA3

To protect wireless connections, WIFI security measures are taken at the data link layer (2) where we have use of different encryption standards, WEP, WPA,WPA2 and WPA3. Wi-Fi protected Access(WPA) was an improvement developed by the Wi-Fi alliance to allow better data encryption and user authentication opposed to the Wired Equivalent Privacy(WEP). WEP was the original WiFi security standard and was an attempt at wireless protection. The goal was to implement security to wireless networks by encrypting data, making the data unrecognizable from unauthorized systems point of view, however, allowing authorized systems to be able to recognize and decrypt the data. The WPA that came after it was introduced in 2003. It shared a lot of similarities with WEP but came with improved solutions on use of security keys and the way users are authorized. As WEP allows each authorized system to have the same key, WPA provides the temporal key integrity protocol(TKIP), which dynamically updates the key authorized systems use. Which in turn prevents intruders from creating their own encryption key to match the one used by the secured network. WPA2 which was introduced in 2004 was an upgrade of WPA. WPA is based on robust security network(RSN) mechanism which operates on pre-shared key (WPA2-PSK), basically the wifi passphrase for access and usually used in home environments, or enterprise mode(WPA-EAP) which is more used in organizational or business environments. WPA3 is the third version of the Wi-Fi protected access protocol, introduced in 2018 and came with new features for both personal and enterprise use. Mainly individual data encryption, simultaneous authentication of protocols, and stronger brute force attack protection [32].

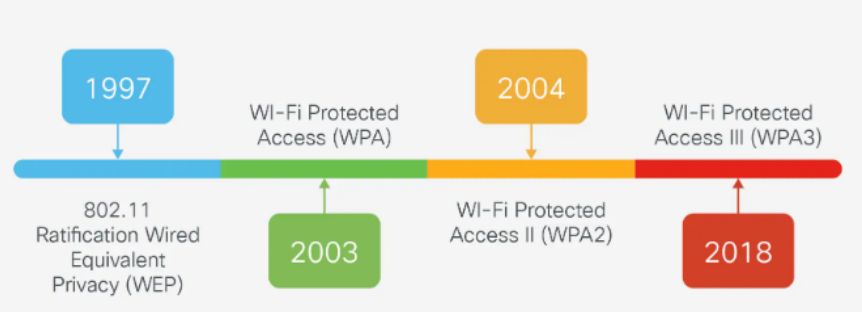


Figure 2.17: From [9], shows timeline of deployment between WEP, WPA, WPA2/WPA3.

2.4 Kali Linux

Kali Linux which is formerly known as BackTrack Linux and is an open source, Debian-based Linux operating system(OS) designed to undertake advanced penetration testing and security auditing. Kali Linux contains several tools available for many kinds of information security tasks, such as wireshark, nmap and aircrack-ng which are used in the threat modelling section (3.3.3) in this paper. The multi platform solution is accessible and free to information security professionals and hobbyists [32].

Chapter 3

Methodology

3.1 Chasing Dory

For this project the provided Chasing Dory ROV is used for our experiments. It is a small compact drone with five thrusters and can explore, take photographs, record videos, including video transmission online. The drone can be maneuvered up and down vertically and can rotate +/-45 degrees. Dory can dive dive up to 50 meters and has a high definition camera(1) for shooting photos and videos. The drone is simple to operate and it comes at a travel size. It has led lights(2) facing towards in the v axis. In the front top of the ROV is the flash light(3), the two vertical propellers(4) and six vent holes(5) above them. The back top of the ROV has mounted two vertical propeller with one horizontal propeller in between them. The tether cable interface(6) allows for connection between the drone and the phone device. The phone used is a HUWAEi PRO 30 and accesses Dory via Wi-Fi using the Chasing GO2 app found on Google store. The underwater robot also comes with a Wi-Fi buoy that has TransFlash(TF) for memory and GPS for ROV tracking. The drones has 8 states of navigation consisting of acsending, descending, left turn, right turn, forward, backward, lowering the head and lifting the head.

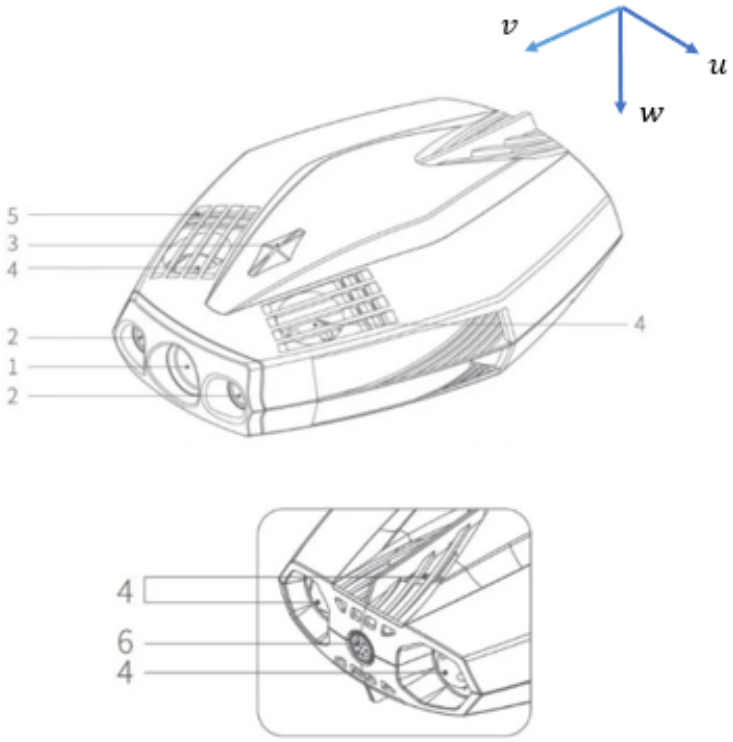


Figure 3.1: Chasing Dory drone components, 1. Camera, 2. Led Lights, 3. Flash light, 4. Thrusters, 5. Drain hole/vent hole, 6. Tether/charging socket.

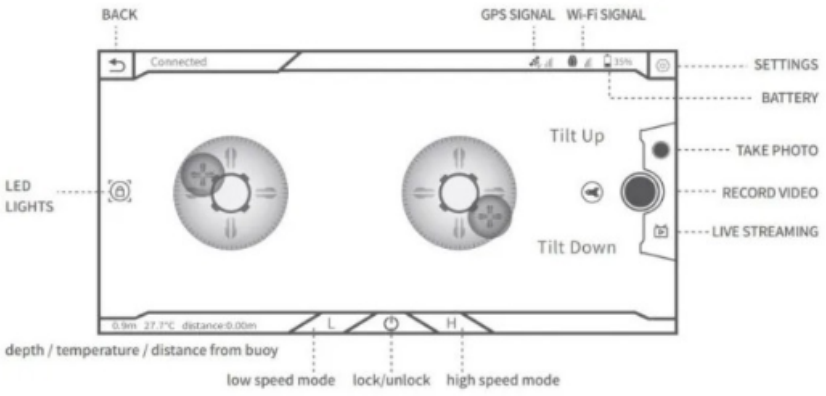


Figure 3.2: Chasing GO2 application interface guide from Chasing Dory's introduction manual.

3.2 ROV Mathematic Model

3.2.1 Dynamic Model

When implementing the dynamic modeling of equations of the ROV, we present the basic design of an assumed ROV similar to Chasing Dory. Dory has an open frame structure where with model dimensions of 0.247 m long, 0.188 m wide and 0.092 m high are obtained from the Dory's manual. It has a dry weight of 1.1 kg and operates at a maximum depth of 15 m. The tasks Dory is mainly designed for are of diving, fishing, taking underwater photography or yatching. Chasing Dory has five thrusters input for six degree of freedom(DOF)(surge, sway, heave, roll, pitch and yaw velocities). The ROV is mounted with four vertical thrusters for heave directions, and one horizontal thruster for surge directions. Roll, pitch and yaw motions are passive and are obtained through maneuvering a selection of thrusters at a time. The ROV also has a package of sensors for IMU(position, velocities and compass), depth and temperature measurements. Prior to modeling the ROV, the following assumptions are made when deriving the general dynamic equations:

a) The ROV is a rigid body and is fully submerged once in water;
b) Water is assumed to be ideal fluid that is in-compressible, friction-less and irrotational;
c) The ROV is slow moving for operation such as underwater exploration;
d) The earth-fixed frame of reference is inertial;
e) Disturbance due to wave excitation is neglected as it is fully submerged;
f) Tether dynamics attached to the ROV are not modeled.

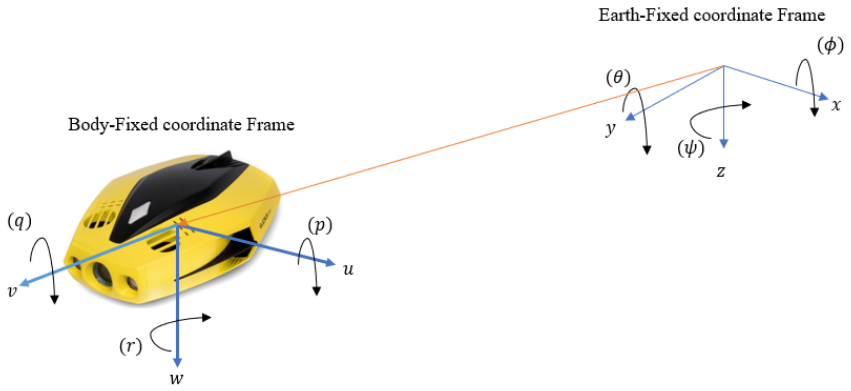


Figure 3.3: Body-fixed coordinate of the ROV with respect to the earth-fixed coordinate frame.

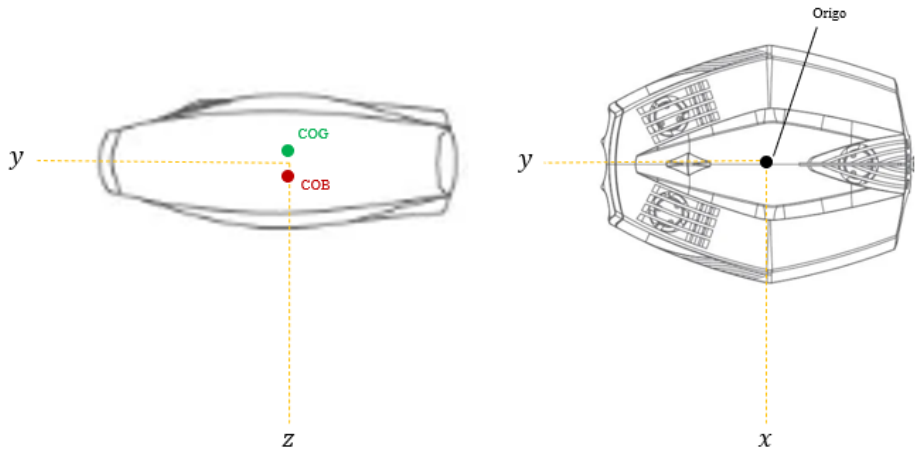


Figure 3.4: ROV placement with respect to earth-fixed coordinate, where COB(-0.12m) and COG(0.15m) placements on z-axis and the ROV remains in center for y- and x-axis.

The ROV dimensions due to restricted possibility of collecting Chasing Dory's hydrodynamic and hydrostatic properties through free decay tests, where chosen according to Minerva's tested and sampled dimensions from [8]. We upscale Dory's volume to be $0.49m^3$, however maintain its weight of $1.1kg$ in the simulations.

Table 3.1: ROV modified simulation values.

ROV defined conditions and dimensions.	
Field	Value
Gravity	9.8070 m/s
Density	1025 kgm^3
Buoyancy(COB)	[0,0,-0.12] m
Gravity(COG)	[0,0,0.15] m
Propeller Diameter	[0.19,0.22,0.22,0.22,0.22] m
Volume	0.49 m^3
Mass	1.1 kg

The ROV model is modeled by a six-degree-of-freedom nonlinear system of first order differential motion equations, which can mathematically represent the linear and angular velocities of the ROV under given adjusted initial conditions. The state of the vehicle is described using two frames seen in figure (3.3), one is the inertial frame(terrestial frame), the other is the local-body frame whose origin coincides with the center of gravity of the vehicle, and the main axis of the steering, surge, sway and heave [8]. For marine vessel it is common that the six degrees of freedom are defined by

the following vectors:

- $\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\eta}_1 & \boldsymbol{\eta}_2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x & y & z & | & \phi & \theta & \psi \end{bmatrix}^{\mathrm{T}}$: position and orientation (Euler angles) in inertia frame;

- $\mathbf{v} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^{\mathrm{T}}$: linear and angular velocities in body-fixed frame;

- $\tau = \begin{bmatrix} \tau_1 & \tau_2 \end{bmatrix}^{T} = \begin{bmatrix} \tau_x & \tau_y & \tau_z & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}$: forces and moments acting on the vehicle in body-fixed frame.

The dynamic equations of the rigid-body ROV are represented in the fixed frame of the body, because the driving forces and the measuring devices are intuitively linked to this frame of the body [8]. Using Newton's method, the motion of a rigid boy at the origin with respect to the body frame as seen in figure (3.3), is given by:

$$\mathbf{M}_{\mathrm{RB}}\dot{\mathbf{v}} + \mathbf{C}_{\mathrm{RB}}(\mathbf{v}) = \tau \tag{3.1}$$

where $\mathbf{M}_{\mathrm{RB}} \in \Re^{6\times 6}$ represents the mass-inertia matrix and $\mathbf{C}_{\mathrm{RB}}(\mathbf{v}) \in \Re^{6\times 6}$ expresses the Coriolis and centripetal matrix. The mass inertia matrix given in (1) can be written as:

$$\mathbf{M}_{\mathrm{RB}} = \begin{bmatrix} m & 0 & 0 & 0 & mz_G & -my_G \\ 0 & m & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m & my_G & -mx_G & 0 \\ 0 & -mz_G & my_G & I_x & -I_{xy} & -I_{xz} \\ mz_G & 0 & -mx_G & -I_{yx} & I_y & -I_{yz} \\ -my_G & mx_G & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix} \tag{3.2}$$

The Coriolis and centripetal terms, describing the angular motion of the ROV can be expressed as:

$$\mathbf{C_{28}}(\mathbf{v}) = \begin{bmatrix} \mathbf{0}_{3\times 3} & \mathbf{C}_{12}(\mathbf{v}) \\ -\mathbf{C}_{12}^{\mathrm{T}}(\mathbf{v}) & \mathbf{C}_{22}(\mathbf{v}) \end{bmatrix} \tag{3.3}$$

with

$$\mathbf{C}_{12}(\mathbf{v}) = \begin{bmatrix} m\left(y_G q + z_G r\right) & -m\left(x_G q - w\right) & -m\left(x_G r + v\right) \\ -m\left(y_G p + w\right) & m\left(z_G r + x_G p\right) & -m\left(y_G r - u\right) \\ -m\left(z_G p - v\right) & -m\left(z_G q + u\right) & m\left(x_G p + y_G q\right) \end{bmatrix} \tag{3.4}$$

$$\mathbf{C}_{22}(\mathbf{v}) = \begin{bmatrix} 0 & -I_{yz}q - I_{xz}p + I_z r & I_{yz}r + I_{xy}p - I_y q \\ I_{yz}q + I_{xz}p - I_z r & 0 & -I_{xz}r - I_{xy}q + I_x p \\ -I_{yz}r - I_{xy}p + I_y q & I_{xz}r + I_{xy}q - I_x p & 0 \end{bmatrix} \tag{3.5}$$

The external force and moment vectors τ include the hydrodynamic forces and moments caused by the damping and inertia of the surrounding fluid, called added mass, and the restoring forces and moments. These

forces and moments tend to stop the motion of the ROV, and the restoring forces that depend on vehicle speed and acceleration are expressed in the body [8]. The mathematical model of the underwater vehicle can be represented by nonlinear motion equations in matrix form, with respect to a local fixed frame of reference:

$$\mathbf{M}\dot{v} + \mathbf{C}(\mathbf{v})\mathbf{v} + \mathbf{D}(\mathbf{v})\mathbf{v} + \mathbf{g}(\boldsymbol{\eta}) = \tau \tag{3.6}$$

$$\dot{\eta} = \mathbf{J}\left(\boldsymbol{\eta}_2\right)\mathbf{v} \tag{3.7}$$

where $\mathbf{v} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^{\mathrm{T}}$ is the body-fixed velocity vector, rigid body inertia matrix and additional mass. $\mathbf{g}(\boldsymbol{\eta}) \in \Re^6$ are the gravity and buoyancy vectors. $\mathbf{C}(\mathbf{v}) = \mathbf{C}_{\mathbb{RB}}(\mathbf{v}) + \mathbf{C}_{\mathrm{A}}(\mathbf{v}) \in \Re^{6x6}$ is the Coriolis and centripetal matrix of the rigid body and added mass separately. $\mathbf{D}(\mathbf{v}) = \mathbf{D}_{\mathbf{L}} + \mathbf{D}_{\mathrm{Q}}(\mathbf{v}) \in \Re^{6\times6}$ corresponds to linear and quadratic damping matrices. The input force and moment vectors $\tau = \mathbf{Tu} \in \Re^6$ combines the thrusts output vector $\mathbf{u} = \mathbf{F}_{\mathrm{T}}\overline{\mathbf{u}} \in \Re^5$ with the thruster configuration matrix $T \in R^{6\times5}$, where $\mathbf{F}_{\mathrm{T}} \in \Re^{5\times5}$ defines the dynamics of each thruster that converts the input voltage command $\overline{\mathbf{u}} \in \Re^5$ into thrust that /pushes/propels the vehicle. $\mathbf{J}\left(\boldsymbol{\eta}_2\right)$ is the Euler transformation matrix [8], which brings the inertia frame aligned with the body frame:

$$\mathbf{J}\left(\boldsymbol{\eta}_2\right) = \begin{bmatrix} \mathbf{J}_1\left(\boldsymbol{\eta}_2\right) & 0 \\ 0 & \mathbf{J}_2\left(\boldsymbol{\eta}_2\right) \end{bmatrix} \tag{3.8}$$

and

$$\mathbf{J}_1\left(\boldsymbol{\eta}_2\right) = \begin{bmatrix} \mathrm{c}(\psi)\mathrm{c}(\theta) & -\mathrm{s}(\psi)\mathrm{c}(\phi) + \mathrm{c}(\psi)\mathrm{s}(\theta)\mathrm{s}(\phi) & \mathrm{s}(\psi)\mathrm{s}(\phi) + \mathrm{c}(\psi)\mathrm{c}(\phi)\mathrm{s}(\theta) \\ \mathrm{s}(\psi)\mathrm{c}(\theta) & \mathrm{c}(\psi)\mathrm{c}(\phi) + \mathrm{s}(\phi)\mathrm{s}(\theta)\mathrm{s}(\psi) & -\mathrm{c}(\psi)\mathrm{s}(\phi) + \mathrm{s}(\theta)\mathrm{s}(\psi)\mathrm{c}(\theta) \\ -\mathrm{s}(\theta) & \mathrm{c}(\theta)\mathrm{s}(\phi) & \mathrm{c}(\theta)\mathrm{c}(\phi) \end{bmatrix} \tag{3.9}$$

$$\mathbf{J}_2\left(\boldsymbol{\eta}_2\right) = \begin{bmatrix} 1 & \mathrm{s}(\phi)\mathrm{t}(\theta) & \mathrm{c}(\phi)\mathrm{t}(\theta) \\ 0 & \mathrm{s}(\phi) & -\mathrm{s}(\phi) \\ 0 & \frac{\mathrm{s}(\phi)}{\mathrm{c}(\theta)} & \frac{\mathrm{c}(\phi)}{\mathrm{c}(\theta)} \end{bmatrix} \tag{3.10}$$

The term $\mathbf{g}(\boldsymbol{\eta})$ is used to describe gravity and buoyancy vectors which are imposed on the ROV in water. Gravity and buoyancy are functions of direction and independent of the vehicles movement. When fully submerged, the ROV buoyancy is equal to the weight of the water displaced, i.e $B = \rho g \nabla$, where ρ is the density of the fluid and ∇ is the volume displaced by the underwater ROV [8]. In the fixed coordinate system of the body, the restoring force vector becomes :

$$\mathbf{g}(\boldsymbol{\eta}) = \begin{bmatrix} (W - B)\sin\theta \\ -(W - B)\cos\theta\sin\phi \\ -(W - B)\cos\theta\cos\phi \\ -(y_G W - y_B B)\cos\theta\cos\phi + (z_G W - z_B B)\cos\theta\sin\phi \\ (z_G W - z_B B)\sin\theta + (x_G W - x_B B)\cos\theta\cos\phi \\ -(x_G W - x_B B)\cos\theta\sin\phi - (y_G W - y_B B)\sin\theta \end{bmatrix} \tag{3.11}$$

Simplified buoyancy and hydrodynamic gravity damping calculation matrix, $\mathbf{g}(\eta)$ is obtained using the following design rules. ROV achieves neutral buoyancy by adding additional float or trim. The lifting mass, due to the gravity of the weight of the ROV, is assumed equal to buoyancy, i.e $W = B$, while additional placed mass on top of ROV leads to the XY coordinates of the center of buoyancy to coincide with the XY coordinates of the center of gravity [8], that is $x_G = x_B = 0, y_G = y_B = 0$, rewriting equation (3.11) to :

$$\mathbf{g}(\boldsymbol{\eta}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (z_G - z_B)\, W \cos\theta \sin\phi \\ (z_G - z_B)\, W \sin\theta \\ 0 \end{bmatrix} \tag{3.12}$$

3.2.2 Hydrodynamic Coefficients

In the vehicle equations of motion (3.6) and (3.7), the external force and torque(moment), such as hydrodynamic resistance(drag), actuator thrust and dynamically increasing mass(added mass) forces are described accordingly to match the vehicle's hydrodynamic coefficients. These coefficients are expressed as the shape of the hydrodynamic derivative fits SNAME notation(1950). For example, the axial quadratic resistance(drag) can be modeled as [8]:

$$X = -\left(\frac{1}{2}\rho C_d A\right) u|u| = X_{u|u|}\, u|u| \tag{3.13}$$

this means that the derivative of drag in the thrust direction with respect to $u|u|$ is:

$$X_{u|u|} = \frac{\partial X}{\partial(u|u|)} = -\frac{1}{2}\rho C_d A \tag{3.14}$$

Since the ROV model used is symmetric with respect to the XZ plane and is nearly symmetric with respect to the YZ plane, we assume that the thrust, roll, pitch and yaw motions are decoupled [8]. Despite the asymmetry about the XY plane, thrust and heave motions are considered to be decoupled because the vehicle essentially operates at relatively low speeds where coupling effects are negligible. With this assumption, the resistance(drag) matrix in (3.6) becomes :

$$\mathbf{D} = \mathbf{D_L} = -\operatorname{diag}\left\{X_u, Y_v, Z_w, K_p, M_q, N_{\mathrm{r}}\right\} \tag{3.15}$$

The relationship between fluid dynamics and torque, and acceleration can be expressed in terms of added mass. For example, if there is an acceleration $\dot{u}$ in the direction of X, the hydrodynamic force X produced by this motion can be given as follows: $X = X_{\dot{u}}\dot{u}$, where $X_{\dot{u}} = \partial X/\partial\dot{u}$ is the hydrodynamic derivative. Off-diagonal elements in $\mathbf{M}_{\mathrm{A}}$ have less impact on

underwater vehicles than diagonal elements. For most low-speed underwater vehicles, these off-diagonal terms are usually ignored [8]. Therefore, $\mathbf{M_A}$ simplifies to the diagonal form as follows:

$$\mathbf{M_A} = -\operatorname{diag}\left\{X_u, Y_v, Z_w, K_p, M_q, N_p\right\} \tag{3.16}$$

The minus sign denoted in $\mathbf{M}_\mathrm{A}$ is due to the pressure on the ROV, that tends to impede vehicle motion. The real mass (or rigid body mass) and the virtual additional mass lie at the origin on both sides of the equation; one is the rigid body property, and the other is related to the force(pressure) the vehicle experiences when the real mass "subtracts" the virtual mass. In most degrees of freedom, the net effect has a larger apparent mass, so the virtual mass is " additional mass". Since the off diagonal elements of $\mathbf{M}_A$ are ignored, the corresponding centripetal additive mass Coriolis matrix $\mathbf{C}_A(\mathbf{v})$ becomes[8]:

$$\mathbf{C}_A(\mathbf{v}) = \begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{i}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{i}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{p}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{i}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{bmatrix} \tag{3.17}$$

3.3 Ethical Hacking

The method used for inspection and gathering of data is ethical hacking, which from a general perspective is seen upon as a malicious activity. Many government agencies, private companies and critical network infrastructures have been victims of hacking attacks, and throughout the years have raised the demand of Cyber security experts to assess the damage caused by these attacks to help strengthen the security, necessary to reduce vulnerabilities, as well as threat detection and defend against potential attackers, usually referred to as Black Hats. Ethical hacking on the other hand, or pen. testing(penetration testing), is the activity executed by Cyber security specialists, usually known as White Hats. Ethical hacking involves information gathering, threat modelling, categorization of attacks and risk analysis on computer devices, network architectures, or other components that utilizes concepts of the OSI model. These following sections (3.3.1), (3.3.2), (3.3.3) and (3.3.4) present knowledge in these areas, including use of it's techniques. The tools used for ethical hacking are

a phone(HUWAEI P30) and a Windows PC Laptop dual booted with Kali-Linux OS, where Wireshark, NMAP and Aircrack-ng softwares are used. Seen in figure (3.4) is the flowchart diagram of the activities and information flow process when conducting ethical hacking.

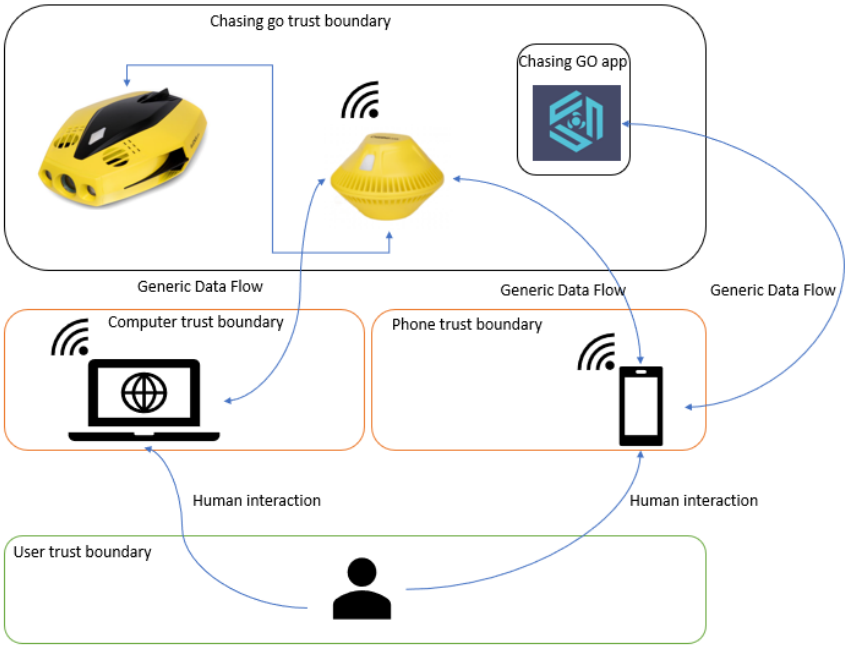


Figure 3.5: Architecture diagram of ethical hacking process.

3.3.1 Overview of 802 protocol and encryption

In this section we assess some fundamental steps of scanning networks before the threat modelling section (3.3.2) of Chasing Dory's wireless network. We begin by doing a light internet(managed network) detection on our computer device using Kali-Linux terminal. In a managed network are Access Points(AP) that could represent a router, computer or a device that can be connected to wireless or via Ethernet cable access. The device can be identified by an ESSID(Extended Service Set Identification) variable, which means the identifying name on the network(wireless og through Ethernet cable). To access an Access Point, a command provided by $\boldsymbol{iwlist} < \boldsymbol{interface} >$ can be ran into linux terminal, where *iwlist* specifies Wi-Fi property, and *scanning* functions shows cell details of the interface. To do a general interface status check, the command $\boldsymbol{iwconfig}$ can instead be run as seen in figure (3.6).

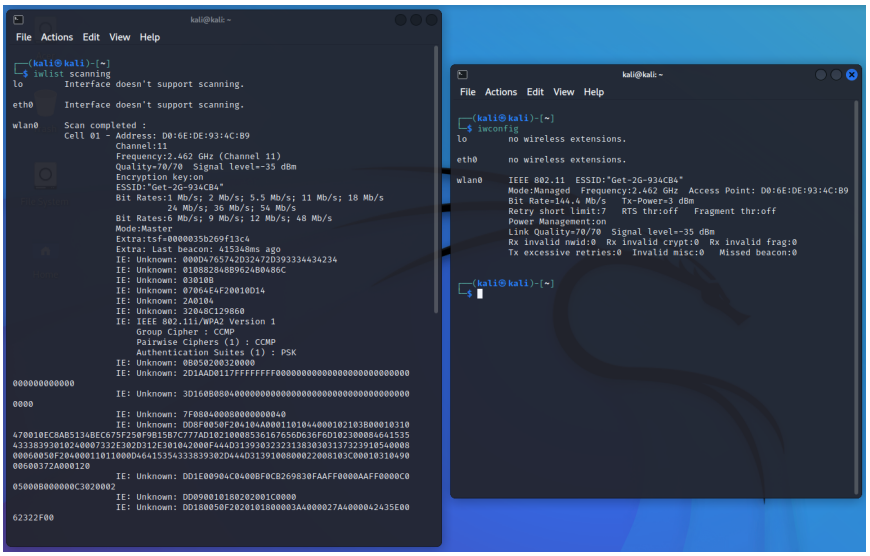


Figure 3.6: Network scanning commands *iwlist* on the left, and *iwconfig* on the right, give information about the providing wlan0 network we are connected to, with the name Get-2G-93CB4(ESSID), the Access Point(D0:6E:DE:93:4C:B9), IEEE version 802.11(802.11i/WPA2) and using channel 11.

To access the packet frames being sent between networks, we can also make use of aircrack-ng's scanning tool "airmon-ng"(further illustrated in threat modelling section (3.3.2)). By running the command we obtain information on networks operating in range that are identified by unique MAC addresses(i.e. the Access Points). A MAC address, contains 48 bit of 6 pair hexadecimal numbers(00:23:1F:2D:34:E1). As observed from the previous figure (3.6), we found our Internet's AP using linux terminal commands, including some information on the encryption type used(WPA Version 1) and other. The encryption in most cases works based of the Open System Authentication(OSA) process to increase security. The process consists of a computer device attempting to connect by requesting authentication from nearby AP, which it waits for to respond with an "OK" message, when received the AP follows that with an association message containing the chosen encryption type(WPA,WPA2/WPA3).

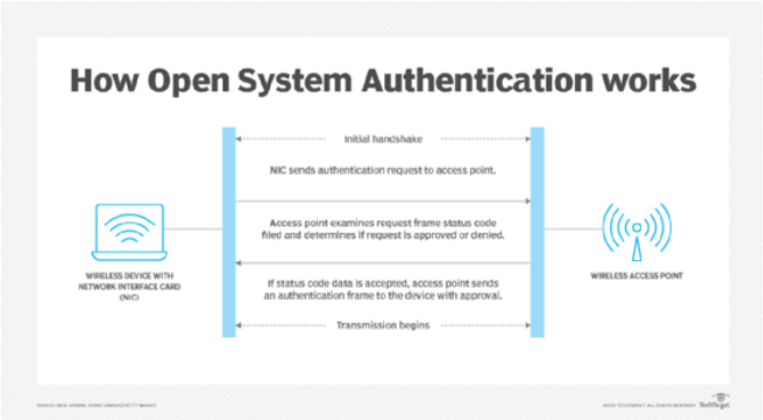


Figure 3.7: From [25], show steps involved in the OSA process.

In our case where WPA/WPA2 is used, a following association message by the name of EAPOL(Extensible Authentication Protocol), a more secure alternative to the OSA protocol from the 802 standard, goes through a more complex handshake process unlike the OSA process, between the connecting device and the AP, where the connecting device must share right EAPOL responses to the EAPOL requests sent by the AP to establish connection. In addition, there is RADIUS device responsible for the authentication of users as seen in figure (3.8) [11].

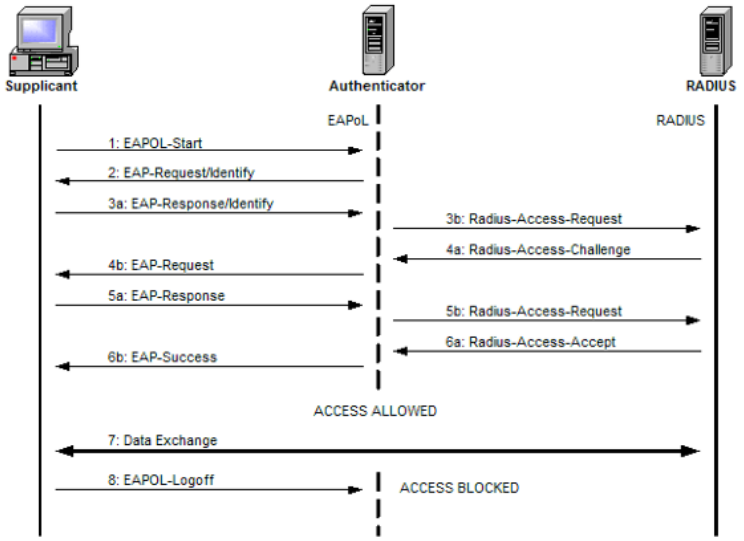


Figure 3.8: From [11], shows increase of complexity of authentication steps that allows for more secure connections.

In other cases where WEP encryption is used, the Shared Key Authentication message is used, this encryption however has been proved to be flawed and is no longer recommended due to the small amount of time it takes to decrypt as seen in figure(3.9) [29].

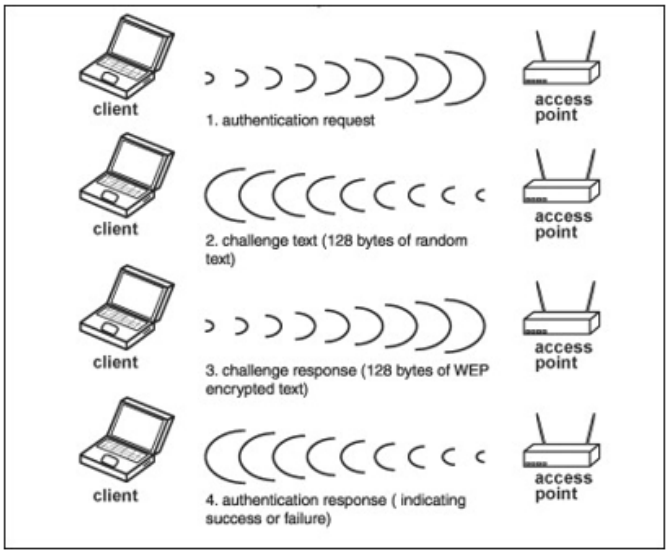


Figure 3.9: From [5], shows simplicity of the SKA encryption.

Understanding use of Aircrack-ng

After a very brief overview of the 802.11 standard fundamentals we can proceed to establish connection with the AP. In our case, the Wi-Fi buoy is the AP we connect to through our computer operating with Kali Linux. We insert the first commands using the aircrack-ng tool that list the amount of data frames being sent and received, type of encryption, the channel operating and so on. The steps implemented in the scanning process contains of first switching our wireless card from managed mode to monitor mode. In monitor mode we enable listening to all wireless packets in range(for Network Band: 2.4 GHz as seen in table 3.2). In managed mode we are only able to listen to packets from our Wi-Fi provider and other physical connected Ethernet interfaces. To view details on our Wi-Fi mode when in managed mode we enter the command $\boldsymbol{airmon-ng}$ as seen in figure (3.10).

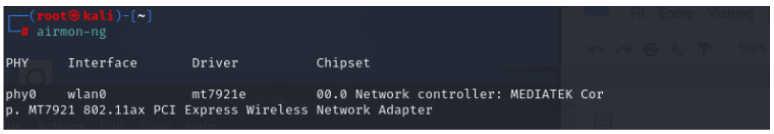


Figure 3.10: airmon-ng command while still in managed mode.

We switch from managed mode to monitor mode by running $\boldsymbol{airmon-ng\ startwlan0}$ seen in figure (3.11), where "mon" is appended to our "wlan0" network, disabling our wireless network, while enabling "listen/monitor" wireless card instead.

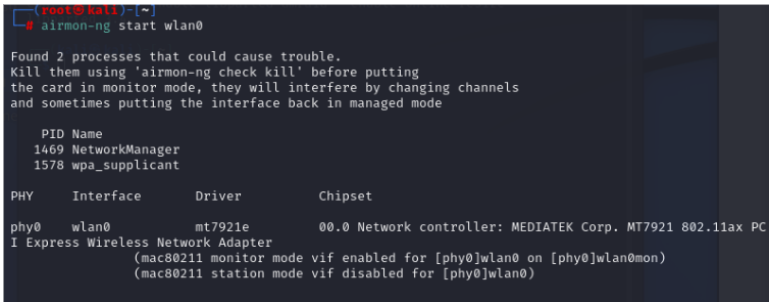


Figure 3.11: airmon-ng start wlan0 command to activate monitor mode.

We view our Wi-Fi card by running ***airmon − ng*** seen in figure (3.12), which shows the card switched to "wlan0mon" monitor.

Figure 3.12: airmon-ng command after switch to monitor mode.

To list out all networks in range from our Laptop, we run ***airodump − ngwlan0mon*** as seen in figure(3.13).

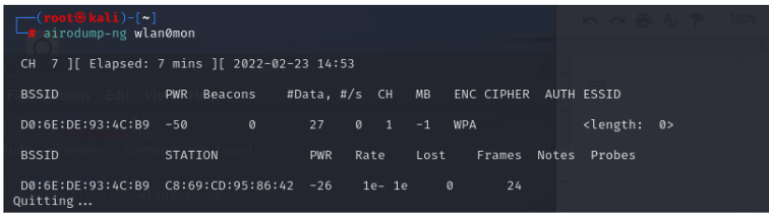


Figure 3.13: airodump-ng wlan0mon to list AP in range we are able to listen to.

What airodump-ng does is hop from channel to channel while listing all APs it can receive frames from. As mentioned in literature review section on the WIFI(802.11) protocol, channel 1 to 13 is the standard used in Europe. The current channel we obtained from our search we observe in the top left corner(channel 7) in figure (3.13) [4].

The upper data column listed by the airodump-ng command shows the (BSSID), i.e the MAC address of the AP. (PWR), representing the signal strength. (Beacons), which is the number of beacon frames received where the more beacons received is an indicator to better quality. The (Data) row gives information on the number of frames received. (CH), stands for channel the AP is operating on. (MB) stands for the maximum speed support by the AP, which gives an indication on the the version of 802.11 used based on the differing band-with lengths. (ENC) is the encryption type, with (CIPHER) giving additional information of the Encryption version if

any. (AUTH), shows the authentication used, while the (ESSID) is the network name (can sometimes be hidden under value "<length: 0>") [4].

The lower data column also contains a range of similar tabs to the upper column.The new tabs correspond to (STATION), which is the MAC address of the client connected to the AP. (Rate) being the number of data packets sent by the client. (Lost) is the amount of packets list over the last 10 seconds. The (notes) gives additional information about the client such as EAPOL or PMKID. While (probes) is the network names(ESSIDs) the client has probed, meaning the networks the client is trying to connect to if it still hasn't established connection. [4]

3.3.2 Threat Modelling

The scope of the threat modelling process is done through sniffing, capturing and analysis of packets between the communication medium of the WIFI buoy tether connected to Chasing Dory. The implementation steps consists of the identifying of assets through passive, semi-passive and active information gathering, where we collect as much information about the ROV as we can.

Passive information gathering

In the passive information gathering step we search for data on Chasing Dory obtained lightly from its introduction manual, phone application and the physical drone. The manual lists the drones specifications, Buoy specification, installation guide and connection credentials, i.e. password to the wireless network. Table (3.2) showcases details found on Chasing Dory's Wi-Fi when connected to the drone via phone app(Chasing GO2). In the app we have options to take pictures, recording videos and streaming online. Inside the setting options reached by tapping >myself>settings, we find access to Chasing Dory website, feedback link and supplementary sensor data(Depth, IMU, GPS data) seen in figure(3.14) and (3.15).

Table 3.2: Chasing Dory Internet access property details .

Chasing Dory Wi-Fi properties	
SSID	Dory44D878BA193B
Protocol	Wi-Fi (802.11n)
Security Type	WPA2-Personal
Network Band	2.4 GHZ
Network Channel	6
Link speed	11/144 (Mbps)
IPv4 address	192.168.1.21
Manufacturer	Intel Corporation
Description	Intel(R) Wi-Fi 6 AX200 160MHz
Driver version	22.10.0.7
Physical address (MAC)	B8-9A-2A-55-81-92
ssh-version	Dropbear sshd 0.53(protocol 2.0)
Running	Linux 4.X
OS CPE	linux_kernel:4.4.2
OS details	DD-WRT v3.0 (Linux 4.4.2)

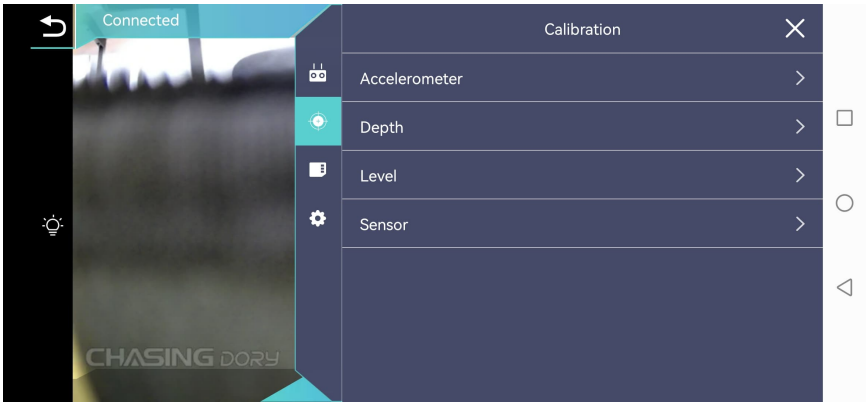


Figure 3.14: Calibration settings inside Chasing GO2 app while connected to wifi buoy.

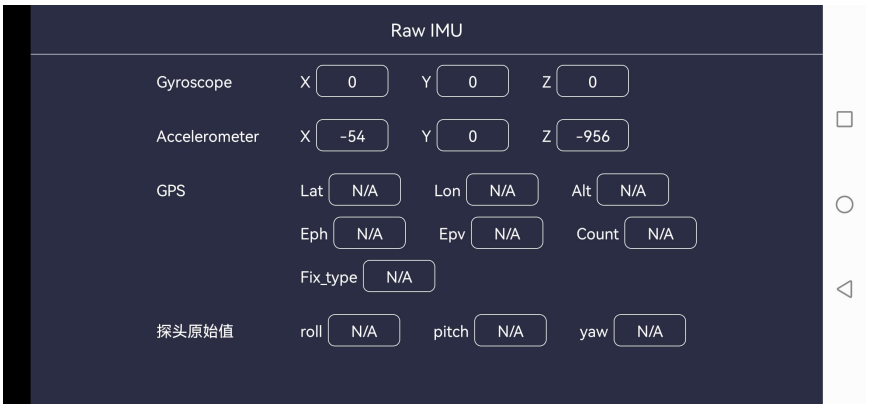


Figure 3.15: RAW IMU data accessed from Sensor tab from Calibration settings.

Semi-passive information gathering

In the semi-passive information gathering approach, we use our available tools aircrack-ng, nmap and wireshark to collect more information on the wireless protocol specification. We later find it to be defined by IEEE 802.11n protocol. The SSID(Service Set Identifier), the name of the wireless network is Dory44D878BA193B. From the table (3.2) done from the passive gathering phase, we obtained some IP address corresponding to the WIFI buoy was 192.168.1.1 and 192.168.1.88, and the IP used from our Laptop device was 192.168.1.20. The more devices we connect to the buoy, the more IP addresses would get allocated, i.e. 192.168.1.21,x.x.x.22, x.x.x.23 and so on. Now that we have obtained the IP address, we can search for ports/type of protocol used to transfer packets, issuing the command $\boldsymbol{nmap}$ **192.168.1.1/24** $-\boldsymbol{sU}$ for UDP port scans, and $\boldsymbol{nmap}$ **192.168.1.1/24** $-\boldsymbol{sT}$ for TCP port scans. With this information an exploiter could create simple scripts with socket programming that allow to connect to Dory and establish communication. Seen in figure (3.16) and (3.17) are the UDP and TCP scans. Our laptop has the MAC-address of (B8:9A:2A:55:81:92) with IP address(192.168.1.21), and the phone device has MAC-address of (FC:AB:90:94:51:FB) with IP address(192.168.1.20), where we connected the phone first before the laptop. Dory Wi-Fi Buoy has MAC-address of (44:D8:78:BA:19:3B) with IP addresses 192.168.1.1 and 192.168.1.88.

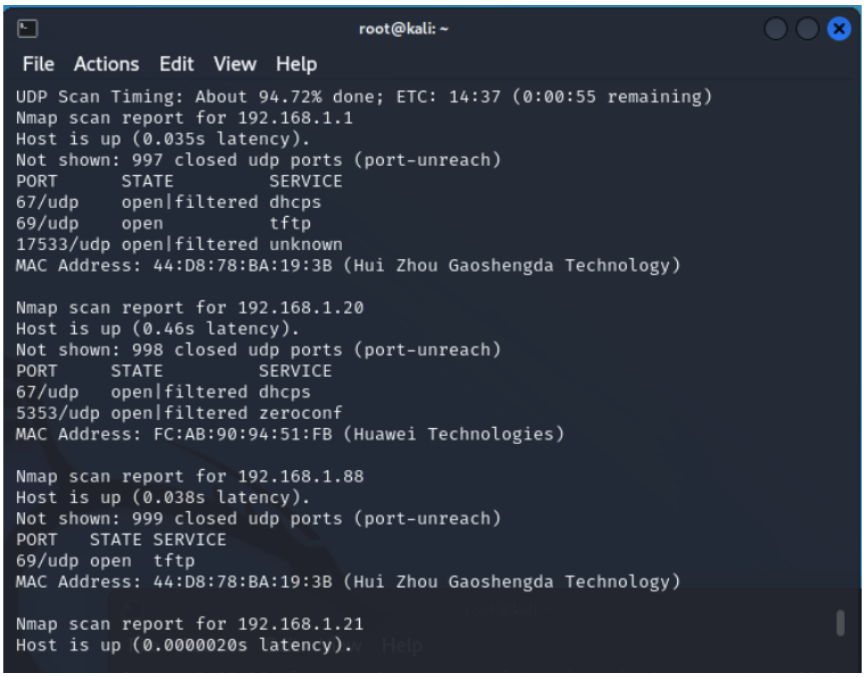


Figure 3.16: UDP port IP scan with nmap.

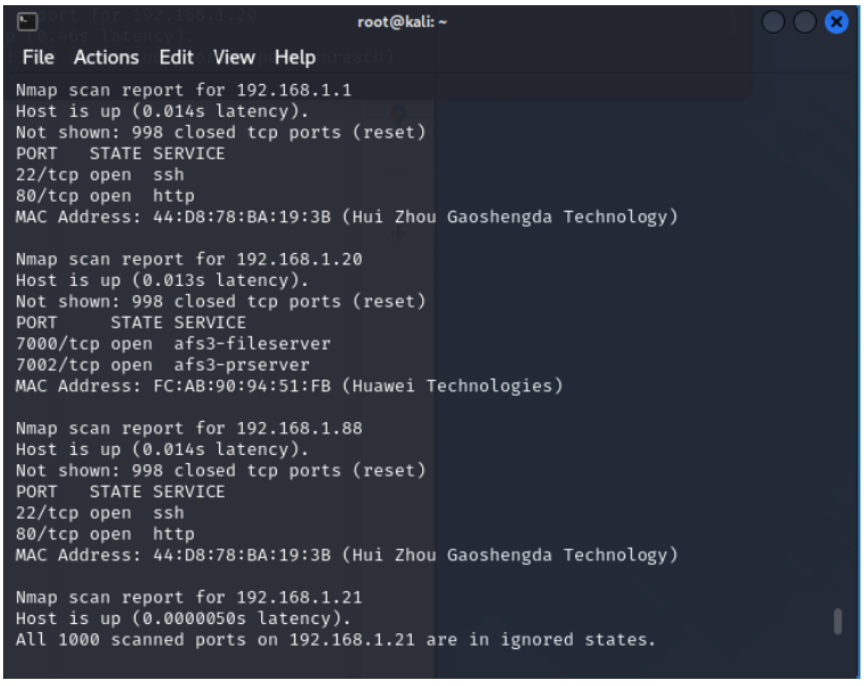


Figure 3.17: TCP port IP scan with nmap.

Active information gathering

In the active information gathering, we proceed with identifying threats as we use the OWASP top 10 list. The list contains previous known attacks

and threats that we also compare against information previously gathered from passive and semi-passive approaches.

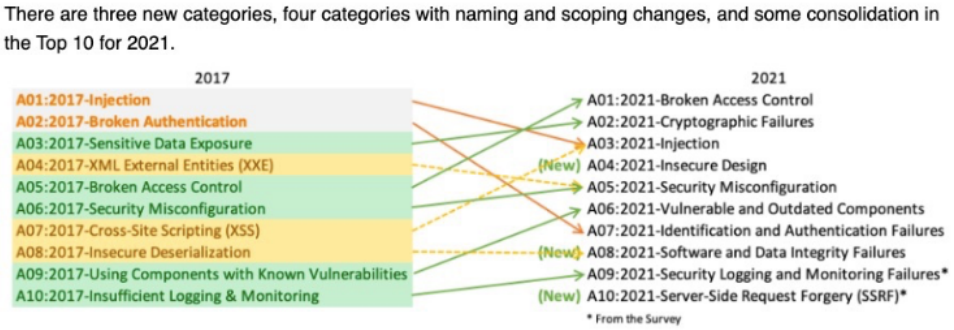


Figure 3.18: From [31], shows the top 10 list of Wi-Fi security risks updated from 2017 to 2021. Where the relevant categories for project are broken access control, insecure design, insecure data transfer storage, identification and authentication failures

.

In section 5, Discussion, we discuss the main comparisons made between the OWASP Top 10 list, the STRIDE model used to categorize threats, section (3.3.3), and risk analysis based on CVSS scores, section (3.3.4).

3.3.3 Categorize threats

The Categorization threats sections consists of identifying threats with the support of STRIDE and the previous mentioned vulnerabilities seen in the table below.

Threat Type	Threat
Spoofing	• Connect to the drone using a default password • Brute force technique is not safe for weak passwords • ARP spoofing(man-in-the-middle) • GPS damage due to GPS spoofing
Tampering	• Injection of instructions to drones from another access point other than the owner • Exploitation of backdoors to access file system
Information disclosure	• Gaining access to communication between drone and the controller and attain instructions and position information • Gaining access to camera and video streams • Gaining access to control logic that works with information about device through reverse engineering of the application technology
Denial of service	• Sending attacks to prevent the controller from communicating with the drone • GPS damage due to GPS jamming
Elevation of privilege	• Access the operating system and enabling root privileges of the drone

Figure 3.19: Table of stride where repudiation threat type is not included due to lack knowledge of whether Dory has any form of malicious tracking detection embedded via file systems.

3.3.4 Risk analysis

From the list of risk factors listed in the STRIDE table in figure(3.19), 5 threats selected for analysis can be attributed with respect to the scope this thesis, which we based on the time constraint and their chances of overcoming potential consequences. The Common Vulnerability Scoring System(CVSS) is used to illustrate the vulnerability and impact of the selected threats. The CVSS qualitative ratings are based on the CVSS scores. 0.1-3.9 are treated as low, 4.0-6.9 are treated as medium, 7.0-8.9 are treated as high, while 9.0-10.0 are critical [4].

#	Threat	Overall	Impact	Exploitability
1	**Password crack** Bypass the password of the drone's network	8.0	4.7	2.5
2	**Denial of Service** Jam the signal between the controller and the user	8.3	3.5	2.5
3	**ARP spoofing** Man-in-the-middle attack between the drone and the controller	7.4	5	2.5
4	**Injecting instructions** Inject instructions to control the drone in flight	7.0	5.5	1.6
5	**Intercept video** Access and decode video data stream between drone and controller	6.1	4.7	1.6

Figure 3.20: CVSS(Common Vulnerability Scoring System table).

The matrix can show the potential and severity of use based on threats received from the CVSS scores.

Probability	**Harm severity**			
	Negligible	**Marginal**	**Critical**	**Catastrophic**
Certain				
Likely		Threat 2	Threat 3	
Possible			Threat 1 & 4	
Unlikely			Threat 5	
Rare				
Eliminated	**Eliminated**			

Severity colour mapping

Low	Medium	High	Very high

Figure 3.21: Risk matrix table.

Part II

Conclusion

Chapter 4

Results

4.1 (CFD) ANSYS-Fluent and ANSYS-AQWA simulations

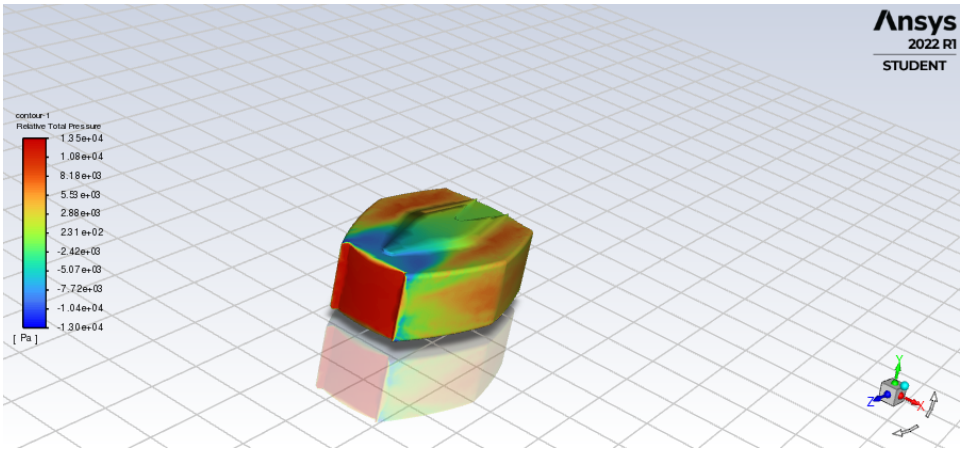


Figure 4.1: Total pressure simulation of ROV in ANSYS Fluent, where the underwater drone is captured in motion towards the z-axis around 13.5 kPa covered in red, and -13.0 kPa in low covered in blue.

With a semi-predictive approach using a closed-tank environment in ANSYS-Fluent as seen in figure (4.1), (4.2),(4.3) and (4.4), we are able to determine the visual ROV's motion active pressures and velocities.

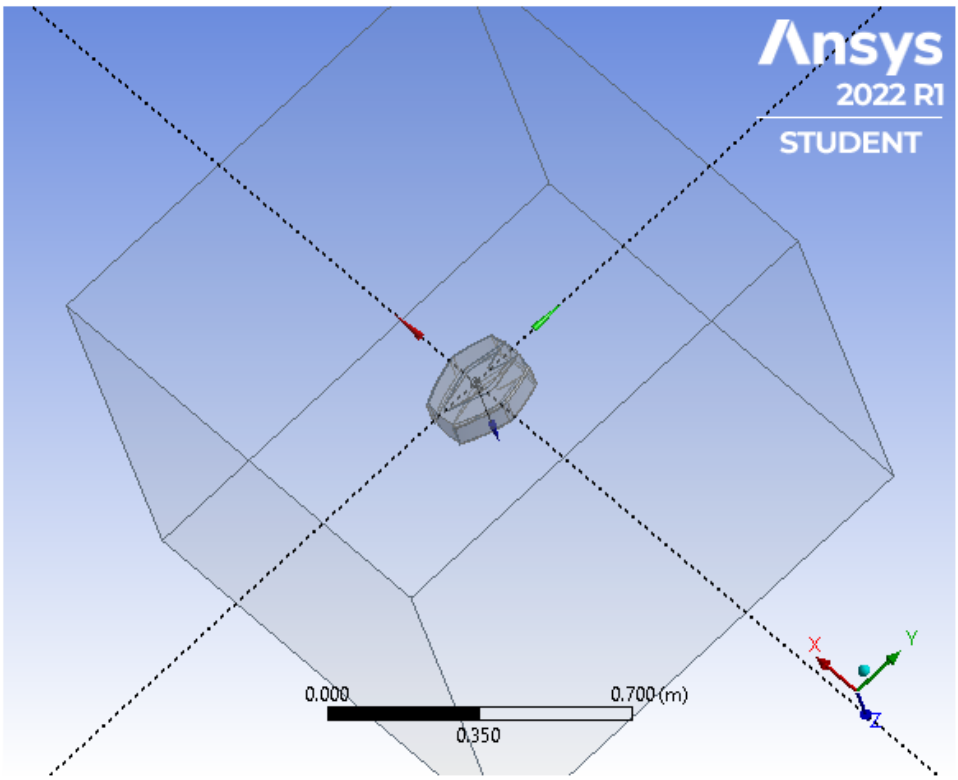


Figure 4.2: Closed tank simulation of total pressure based on underwater wave velocity in ANSYS Fluent.

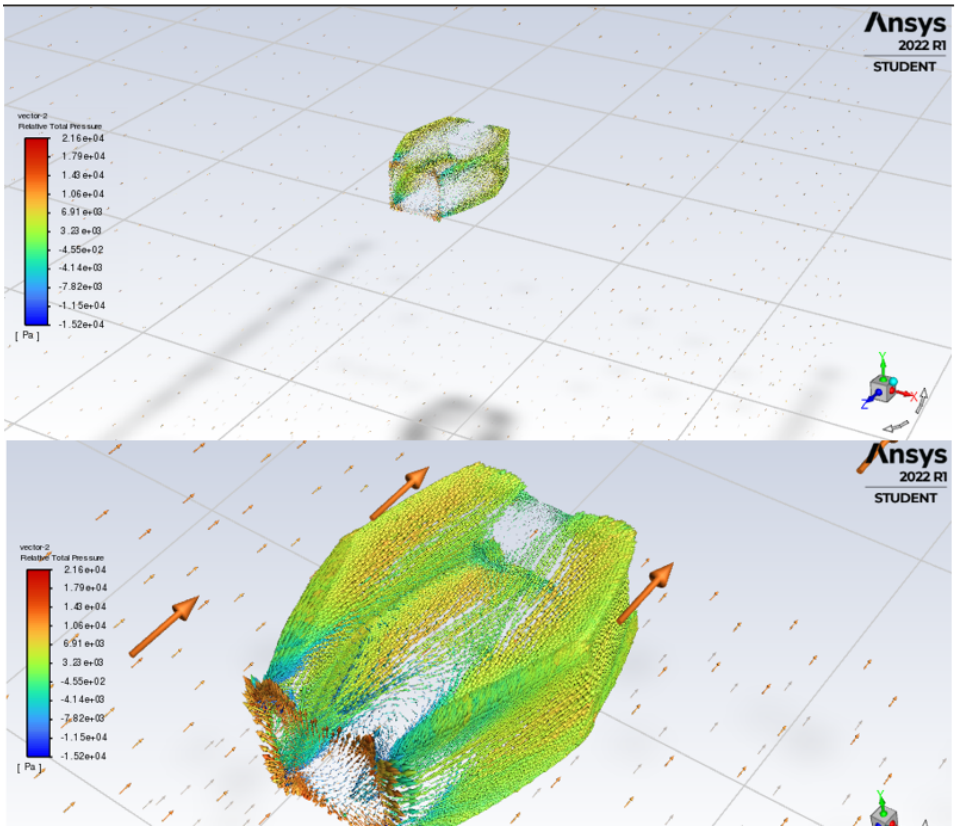


Figure 4.3: Open tank simulation of total pressure based on underwater wave velocity in ANSYS Fluent.

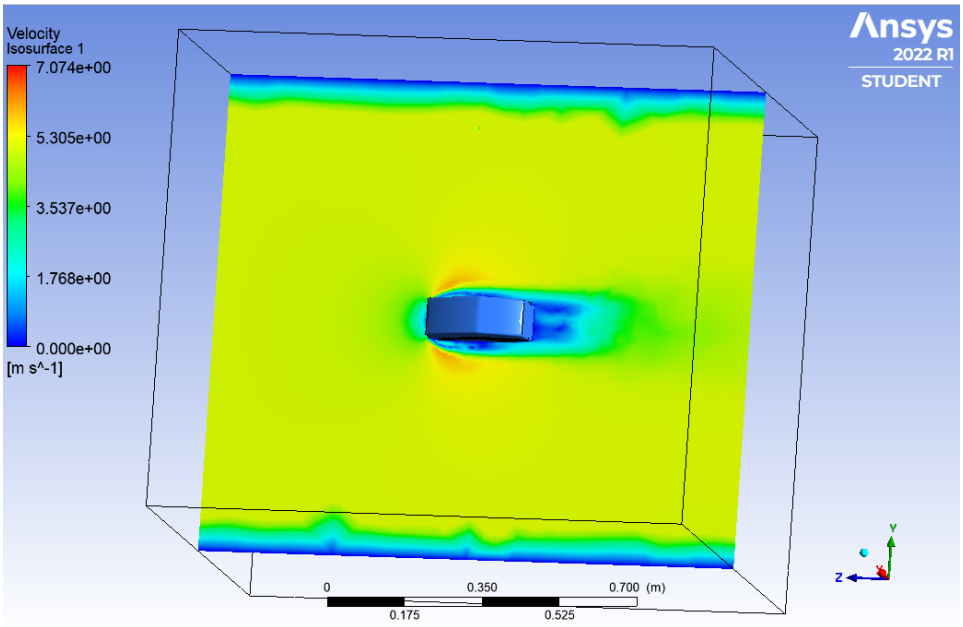


Figure 4.4: Velocity graph of Dory model in closed tank simulation.

4.2 ROV thruster and dynamic in simulation

The models used to simulate the dynamics of the ROV are presented, which include inertia, added mass models, Euler transform, Coriolis and centripetal forces, gravity and buoyancy, and damping and thruster models. The latest version of MATLAB and Simulink is used to configure and implement the ROV dynamics model. The Simulink model is an orginal from Enrico Ande who worked on the UUV dynamics modelling in Matlab and Simulink in [1]. In the repository we find models on an uuv dynamics and a rov thruster dynamics of the Minerav ROV from NTNU, rewritten for this project as the .c files(*rov_dynamics.c* and *rov_thruster.c*). In the *rov_dynamics* file we find computation components on inertia, added mass, euler transform, Coriolis and centripetal forces, gravity and buoyancy forces and damping models. While in the *rov_thruster* file we find computation components on the thruster model that generate speed and navigation values from thrust revolution of the ROV. In our attempt of modeling the behaviors of the ROV, we merge computation components from the *rov_dynamics* and *rov_thruster* inside Simulink in block diagrams. The MEX-add on from Matlab is used to compile the .c file into a mex type file which can be ran inside Simulink. Simulink is another Matlab add-on that provides an interactive graphical environment of the modeling, simulation and analysis of dynamic systems. It can quickly create virtual prototypes to explore the graphical user interface(GUI) used to create models. It comes with a comprehensive library of predefined blocks for building a graphical model of a ROV system using drag-and-drop mouse operations.

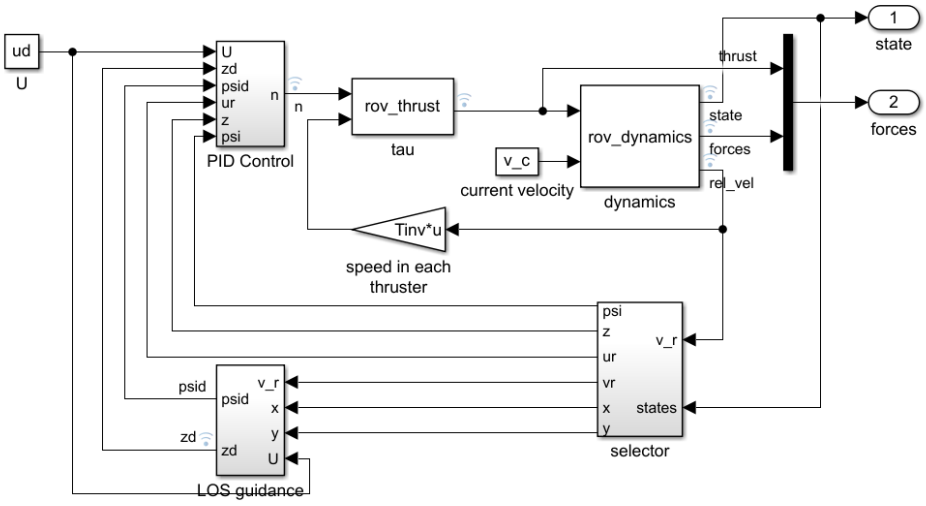


Figure 4.5: ROV dynamics model in Simulink.

An overview of the ROV dynamic model can be seen in figure (4.5). The desired goal of the ROV to move straight across the XY plane, i.e $y_d = 0m$ to $x_d = 10m$. By adjusting the variable controller gain inputs of the PID controller which takes in the error computed from the desired

input(heading(*psid*) and depth(*zd*)) and the actual input(heading(*psi*) and depth(*z*)). The PID controller then optimizes the input signal to six degrees of freedom in the five available thrusters determined by the thrust configuration matrix(defined in rov_thrust block) in figure (4.5). The total body dynamics forces of the ROV are characterized by a rigid body and additional mass matrices, Coriolis and centripetal forces, damping forces and gravity and buoyancy forces, equation (3.6). The sum of these forces multiplied by the inverse of the rigid body and adding the mass matrix gives the acceleration of the vehicle, which integrated gives the velocity vectors. The integral function is designed to calculate the speed and also return it to the next iteration until the simulation time is reached. Besides attaining the fixed speed of the object, the ROV position with respect to the earth-fixed coordinate system can be obtained from the Euler transform, equation (3.8).The residual Euler angles are used to compute the gravity and buoyancy required for the dynamics of the ROV.

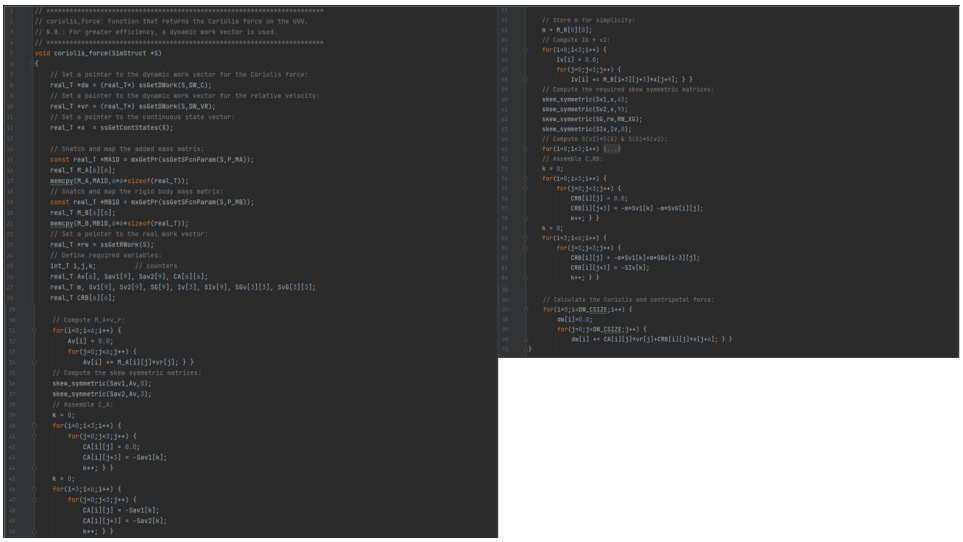
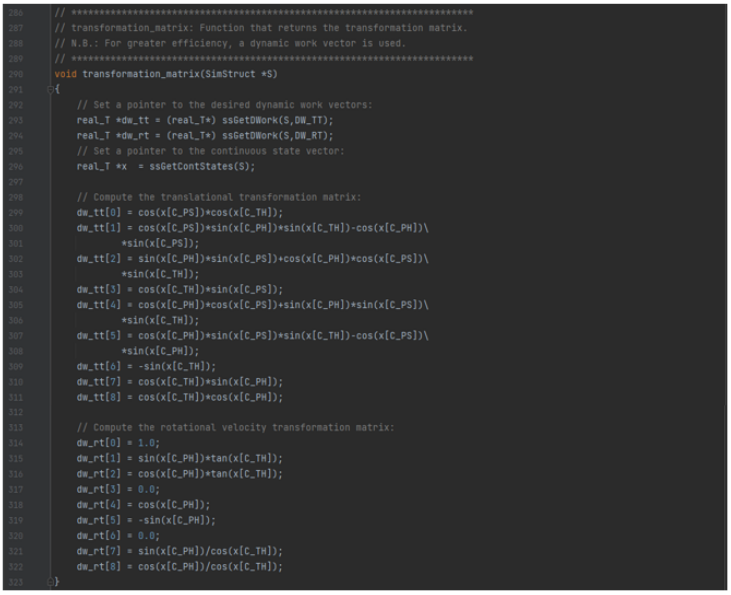


Figure 4.6: Coriolis computation in C of the ROV.

Coriolis and centripetal matrices consisting of a rigid body and additional mass components are computed inside .c as shown in figure (4.6). As Simulink handles column/row vectors operations, the Coriolis and centripetal defined in .c file is written in order to consist of matrix additions - C_{RB}(in line 72 and below) and C_A(in line 38 to 58), which assemble functions of defined added mass, rigid body mass and skew symmetric matrices. In figure (4.7) is the Euler transform J consisting of 2 sub matrices, one corresponding to the translational transformation matrix(line 298 to line 311), and the other corresponding to the rotational velocity transformation matrix. In figure (4.8) is the gravity and buoyancy vectors modeled, they are simplified to neutral buoyancy conditions by reducing W = B. As observed, the zeros shown in equation(3.12) are also reflected in the code.

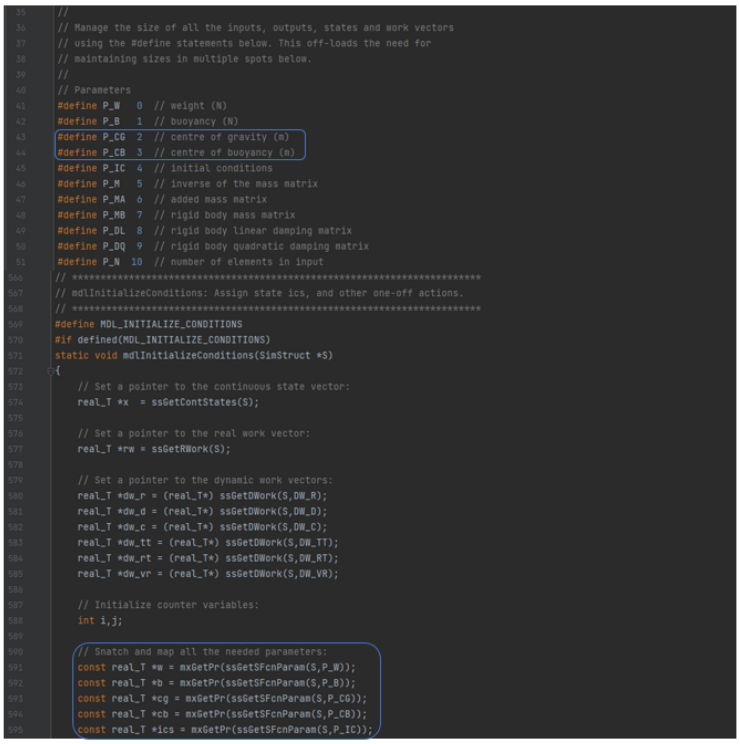


```
// *********************************************************************
// transformation_matrix: Function that returns the transformation matrix.
// N.B.: For greater efficiency, a dynamic work vector is used.
// *********************************************************************
void transformation_matrix(SimStruct *S)
{
    // Set a pointer to the desired dynamic work vectors:
    real_T *dw_tt = (real_T*) ssGetDWork(S,DW_TT);
    real_T *dw_rt = (real_T*) ssGetDWork(S,DW_RT);
    // Set a pointer to the continuous state vector:
    real_T *x  = ssGetContStates(S);

    // Compute the translational transformation matrix:
    dw_tt[0] = cos(x[C_PS])*cos(x[C_TH]);
    dw_tt[1] = cos(x[C_PS])*sin(x[C_PH])*sin(x[C_TH])-cos(x[C_PH])\
           *sin(x[C_PS]);
    dw_tt[2] = sin(x[C_PH])*sin(x[C_PS])+cos(x[C_PH])*cos(x[C_PS])\
           *sin(x[C_TH]);
    dw_tt[3] = cos(x[C_TH])*sin(x[C_PS]);
    dw_tt[4] = cos(x[C_PH])*cos(x[C_PS])+sin(x[C_PH])*sin(x[C_PS])\
           *sin(x[C_TH]);
    dw_tt[5] = cos(x[C_PH])*sin(x[C_PS])*sin(x[C_TH])-cos(x[C_PS])\
           *sin(x[C_PH]);
    dw_tt[6] = -sin(x[C_TH]);
    dw_tt[7] = cos(x[C_TH])*sin(x[C_PH]);
    dw_tt[8] = cos(x[C_TH])*cos(x[C_PH]);

    // Compute the rotational velocity transformation matrix:
    dw_rt[0] = 1.0;
    dw_rt[1] = sin(x[C_PH])*tan(x[C_TH]);
    dw_rt[2] = cos(x[C_PH])*tan(x[C_TH]);
    dw_rt[3] = 0.0;
    dw_rt[4] = cos(x[C_PH]);
    dw_rt[5] = -sin(x[C_PH]);
    dw_rt[6] = 0.0;
    dw_rt[7] = sin(x[C_PH])/cos(x[C_TH]);
    dw_rt[8] = cos(x[C_PH])/cos(x[C_TH]);
}
```

Figure 4.7: J the transformation matrix of ROV.



```
//
// Manage the size of all the inputs, outputs, states and work vectors
// using the #define statements below. This off-loads the need for
// maintaining sizes in multiple spots below.
//
// Parameters
#define P_W   0  // weight (N)
#define P_B   1  // buoyancy (N)
#define P_CG  2  // centre of gravity (m)
#define P_CB  3  // centre of buoyancy (m)
#define P_IC  4  // initial conditions
#define P_M   5  // inverse of the mass matrix
#define P_MA  6  // added mass matrix
#define P_MB  7  // rigid body mass matrix
#define P_DL  8  // rigid body linear damping matrix
#define P_DQ  9  // rigid body quadratic damping matrix
#define P_N  10  // number of elements in input
// *********************************************************************
// mdlInitializeConditions: Assign state ics, and other one-off actions.
// *********************************************************************
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{
    // Set a pointer to the continuous state vector:
    real_T *x  = ssGetContStates(S);

    // Set a pointer to the real work vector:
    real_T *rw = ssGetRWork(S);

    // Set a pointer to the dynamic work vectors:
    real_T *dw_r = (real_T*) ssGetDWork(S,DW_R);
    real_T *dw_d = (real_T*) ssGetDWork(S,DW_D);
    real_T *dw_c = (real_T*) ssGetDWork(S,DW_C);
    real_T *dw_tt = (real_T*) ssGetDWork(S,DW_TT);
    real_T *dw_rt = (real_T*) ssGetDWork(S,DW_RT);
    real_T *dw_vr = (real_T*) ssGetDWork(S,DW_VR);

    // Initialize counter variables:
    int i,j;

    // Snatch and map all the needed parameters:
    const real_T *w = mxGetPr(ssGetSFcnParam(S,P_W));
    const real_T *b = mxGetPr(ssGetSFcnParam(S,P_B));
    const real_T *cg = mxGetPr(ssGetSFcnParam(S,P_CG));
    const real_T *cb = mxGetPr(ssGetSFcnParam(S,P_CB));
    const real_T *ics = mxGetPr(ssGetSFcnParam(S,P_IC));
```

Figure 4.8: Buoyancy and Gravity mapping of ROV.

As for hydrodynamic damping models seen in figure (4.9), are defined to return the linear and quadratic damping matrices of the ROV. The rep-

resentation of linear damping matrix is seen in equation (3.15), with the quadratic term also modeled accordingly by six dimensional column vectors. The thrust distribution optimization problem aims to generate the desired generalized efforts while minimizing the use of control effort. There are few solutions for the functions of distributing or configuring the thrust, however the value of the actuator command is obtained by solving the optimization problem at each sampling period of the control loop.

```
// ***************************************************************************
// damping_force: Function that returns the damping force on the UUV.
// N.B.: For greater efficiency, a dynamic work vector is used.
// ***************************************************************************
void damping_force(SimStruct *S)
{
    // Set a pointer to the dynamic work vector for the damping force:
    real_T *dw = (real_T*) ssGetDWork(S,DW_D);
    // Set a pointer to the dynamic work vector for the relative velocity:
    real_T *vr = (real_T*) ssGetDWork(S,DW_VR);
    // Snatch and map the linear damping matrix:
    const real_T *DL1D = mxGetPr(ssGetSFcnParam(S,P_DL));
    real_T D_L[6][6];
    memcpy(D_L,DL1D,6*6*sizeof(real_T));
    // Snatch and map the quadratic damping matrix:
    const real_T *DQ1D = mxGetPr(ssGetSFcnParam(S,P_DQ));
    real_T D_Q[6][6];
    memcpy(D_Q,DQ1D,6*6*sizeof(real_T));
    // Counter:
    int_T i, j;

    // Compute the damping force:
    for(i=0;i<DW_DSIZE;i++) {
        dw[i] = 0.0;
        for (j=0;j<DW_DSIZE;j++) {
            dw[i] += D_L[i][j]*vr[j] + D_Q[i][j]*fabs(vr[j])*vr[j]; }  }
}
```

Figure 4.9: Linear and quadratic matrices mapping of ROV.

4.3 PID controller design

The PID controller designed in Simulink is analyzed, tested and optimized. The setup environment of the control system does not take into account uncertainties and external disturbances, such as underwater current. The ROV is instructed to move straight in the XY plane with the help of a line of sight guidance navigation system and PID controllers, the model architecture is shown in figure (4.5). The line of sight guidance system works based on a steering mechanism with inputs acquired from selector block that receives values computed from the rov_thruster and rov_dynamics files, by returning velocity thrusts states and forces. The PID controller block takes inputs from LOS guidance and selector blocks, and computes an output $4n$ based on depth, speed and heading controller systems. Figure (4.10) shows the how the PID controller system looks like. In this paper we only focus on the observing and optimizing controller gains for the depth and heading sensors.

The modified PID regulator gain matrices are $K_p, K_d, andK_i$, according to the thrust force vector returned from the Simulink thrust block figure(4.5) that takes input of propeller revolutions forces. With computation done in rov_thruster.c and rov_dynamics.c files,we use Minerva ROV from NTNU thrust dimensions, due to restricted time of attaining Chasing Dory's thrust dimensions doing free decay experiments. The mathematical

thrust distribution used is then written as:

$$\mathbf{f}_{\text{th}} = \mathbf{T}^{-1}\mathbf{p}_{\text{c}}, \tag{4.1}$$

where $\mathbf{p_c}$ is the (6×1) vector of control parameters and is given by

$$\mathbf{p}_{\text{c}} = \begin{bmatrix} p_{\text{c, speed}} & 0 & p_{\text{c, depth}} & 0 & 0 & p_{\text{c, heading}} \end{bmatrix}^T. \tag{4.2}$$

The depth and heading control parameters can be obtained by:

$$\begin{aligned} p_{\text{c, speed}} &= -k_{\text{p, s}}\ (u(t) - u_{\text{d}}(t)) - k_{\text{i, s}} \int_0^t (u(\tau) - u_{\text{d}}(\tau))\,\text{d}\tau - k_{\text{d, s}}\ (\dot{u}(t) - \dot{u}_{\text{d}}(t)) \\ p_{\text{c, depth}} &= -k_{\text{p, d}}\ (z(t) - z_{\text{d}}(t)) - k_{\text{i, d}} \int_0^t (z(\tau) - z_{\text{d}}(\tau))\,\text{d}\tau - k_{\text{d, d}}\ (\dot{z}(t) - \dot{z}_{\text{d}}(t)) \\ p_{\text{c,heading}} &= -k_{\text{p,h}}\ (\psi(t) - \psi_{\text{d}}(t)) - k_{\text{i, h}} \int_0^t (\psi(\tau) - \psi_{\text{d}}(\tau))\,\text{d}\tau - k_{\text{d,}} \\ &\qquad \text{h}\ (\dot{\psi}(t) - \dot{\psi}_{\text{d}}(t)) \end{aligned} \tag{4.3}$$

where $k_{\text{p, speed}}$, $k_{\text{i,speed}}$, $k_{\text{d,speed}}$, $k_{\text{p, depth}}$, $k_{\text{i,depth}}$, $k_{\text{d, depth}}$, $k_{\text{p,heading,}}$ $k_{\text{i, heading}}$ and $k_{\text{d, heading}}$ are the proportional, integral and derivative gains for the depth and heading, respectively. Also, u_{d}, z_{d} and ψ_{d} are the desired speed, depth and heading.

In Simulink, the PID controller can be modeled as shown in figure (4.10). The block diagram uses the controller shown in equation (4.1). By right-clicking the block diagram and going to "selecting > Look under Mask" we see the actual representation of the controller as seen in figure (4.12). Where P, I and D refer to the respective positive controller gains in the PID control law.

The PID controller receives the error signal and provides control action to the ROV. The initial simulation speed, depth, steering and heading gains are:

Table 4.1: PID controller initial gains tested.

PID controller	Proportional gain	Derivative gain	Integral gain
Depth controller	$K_{p,d} = 120$	$K_{d,d} = 20$	$K_{i,d} = 70$;
Heading controller	$K_{p,psi} = 100$	$K_{d,psi} = 30$	$K_{i,psi} = 30$;
Speed controller	$K_{p,u} = 100$	$K_{d,u} = 10$	$K_{i,u} = 20$;
Steering controller	$K_{p,s} = 120$	$K_{d,s} = 20$	$K_{i,s} = 30$;

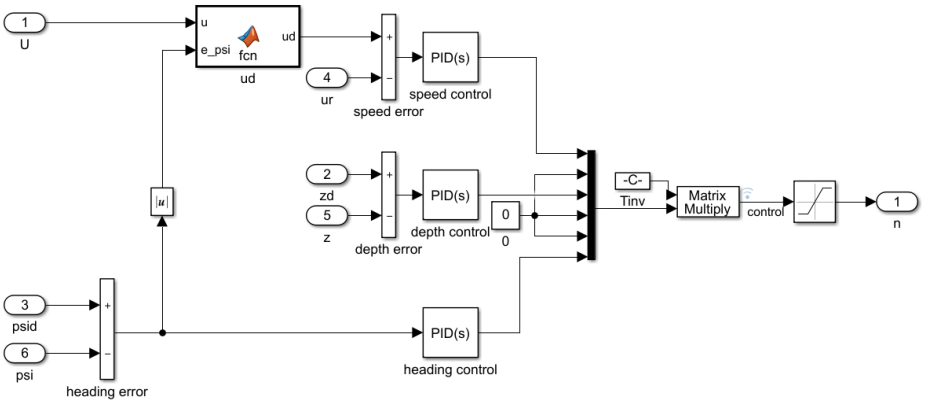


Figure 4.10: PID controller for (6) DOF model.

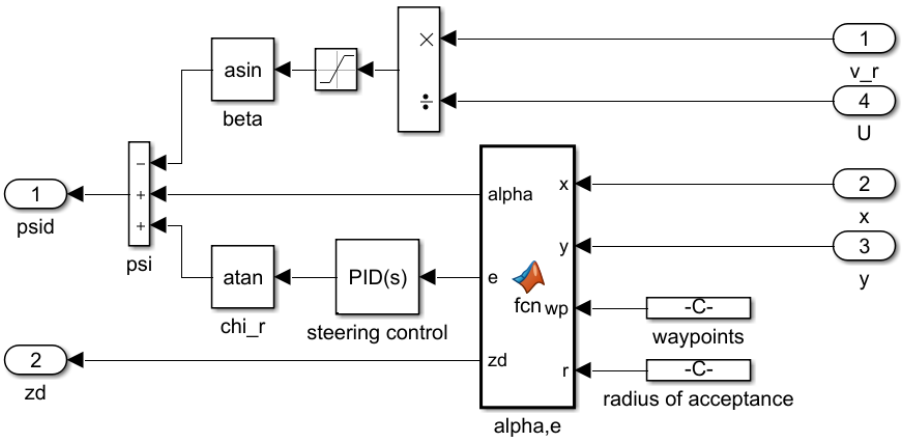


Figure 4.11: LOS guidance system model with steering PID control for the waypoints and their shape, which the ROVs navigation system also uses to determine its path when orientating and moving.

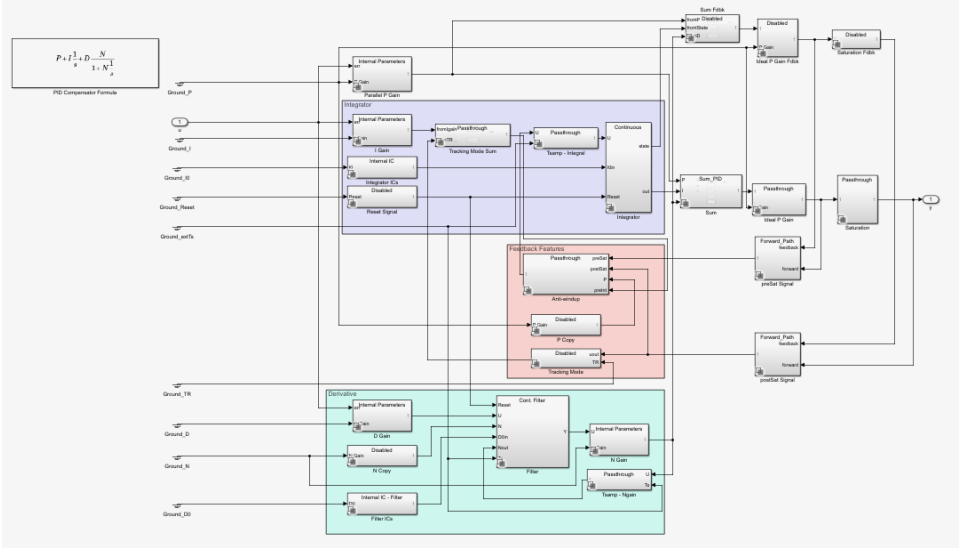


Figure 4.12: Details of PID controller under mask.

The position and velocity time responses plots are seen in observed in figure (4.13) and (4.14), which show how the ROV starts moving from the defined entrance and attempts to navigate straight through 6 small circles, in a period of 60 seconds based on the output signal from the depth and heading PID controllers.

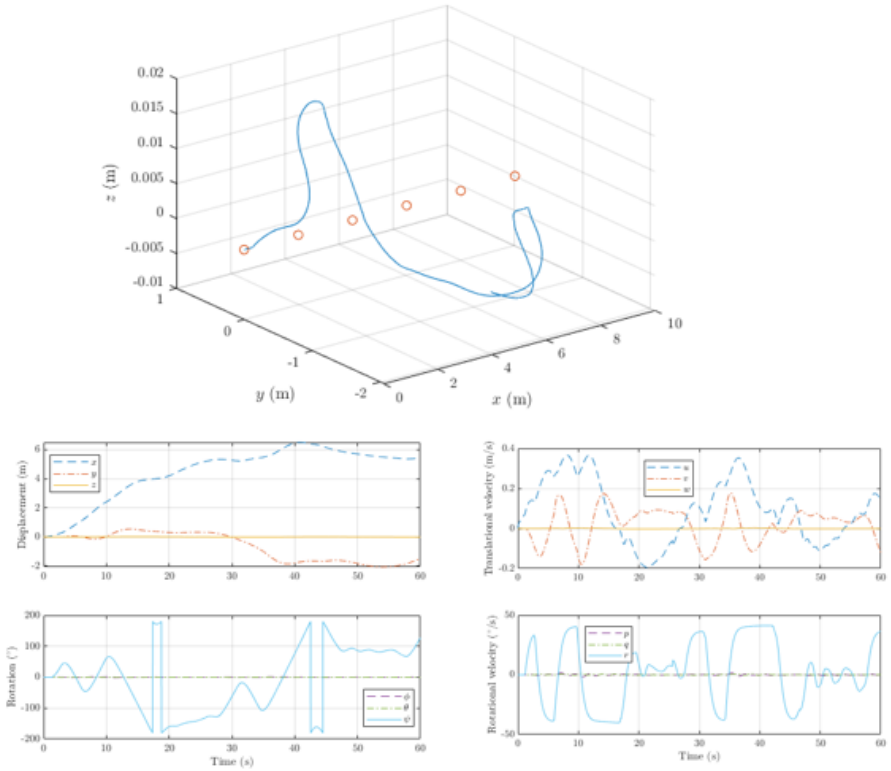


Figure 4.13: Top middle is the line of sight path the ROV follows based on the proportional, derivative and integral gain used as listed in table (4.1), the red circles across the XY plane represent the desired path to follow. Middle (left) plot is the Displacement, bottom (left) plot the Rotation, middle (right) plot the Translational velocity and bottom (right) plot the Rotational velocity.

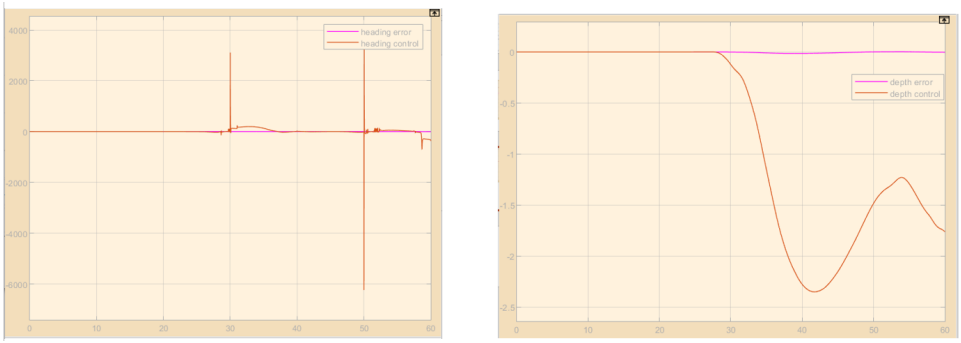


Figure 4.14: Shows desired input and controlled output signals using the same controller gains as listen in table (4.1), where on the (left) we observe the heading error input seems to be constant, while in fact it remains with lower amplitude and frequency rate than the heading control. Figure (4.15) show a close up of the (left) plot. The (right) side plot shows the similar pattern of the depth error remaining with amplitude and frequency rate opposed to the depth controller.

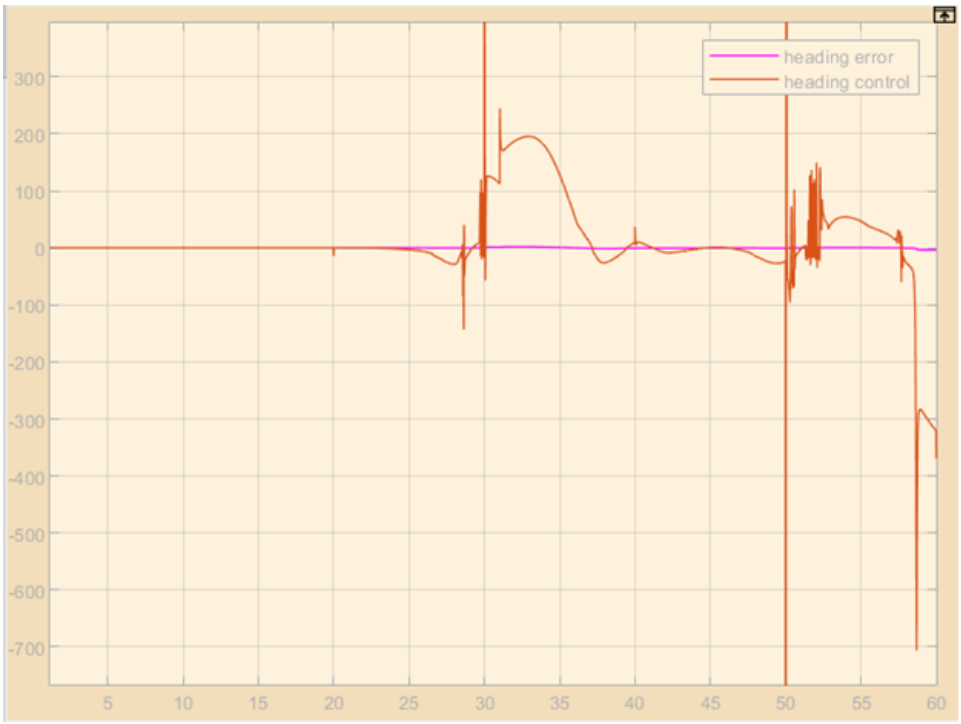


Figure 4.15: The heading control signal is given a closer look where we observe signs of high oscillations compared to the heading error signal. It is also seen that the desired input and contrlled output signals do not align well after going through the controller with gains used from table (4.1).

Table 4.2: PID controller final gains tested.

PID controller	Proportional gain	Derivative gain	Integral gain
Depth controller	$K_{p,d} = 100$	$K_{d,d} = 10$	$K_{i,d} = 10$;
Heading controller	$K_{p,psi} = 80$	$K_{d,psi} = 19$	$K_{i,psi} = 10$;
Speed controller	$K_{p,u} = 100$	$K_{d,u} = 10$	$K_{i,u} = 50$;
Steering controller	$K_{p,s} = 10$	$K_{d,s} = 80$	$K_{i,s} = 50$;

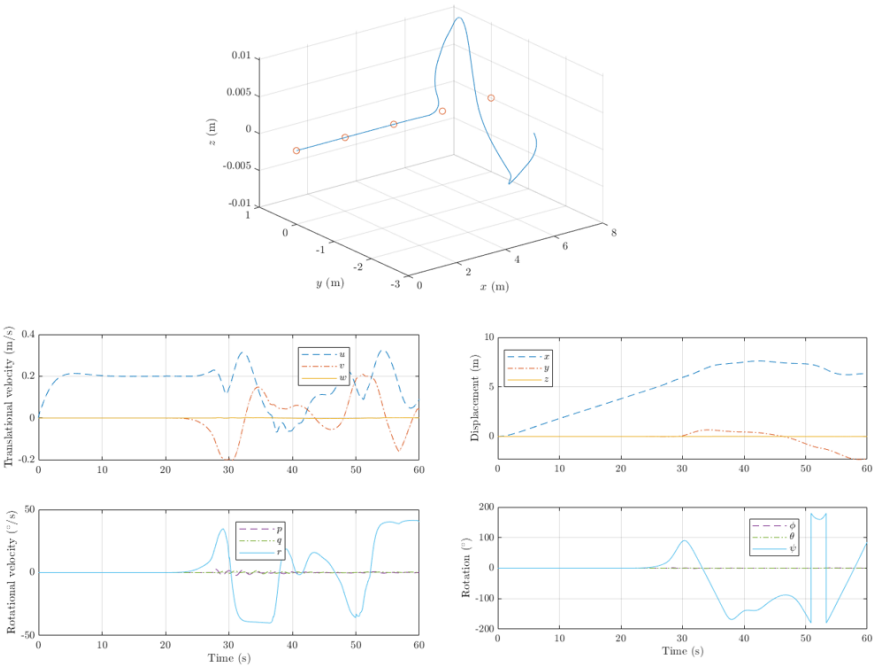


Figure 4.16: By using controller gains listed in table (4.2), the ROV traveled path can be observed as it passes through 3 out of the 6 circles, while almost reaching the 4th circle, it loses track and fails to follow the defined path. The displacement, rotation, rotational velocity and translational velocity can be observed in middle (right), bottom(right), bottom (left) and middle (left).

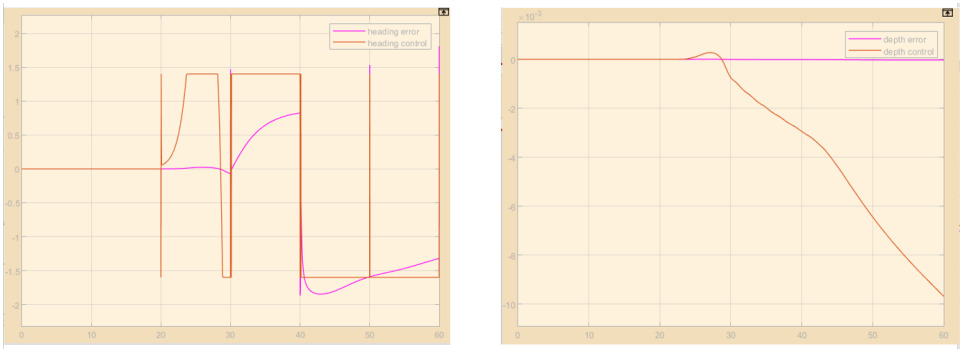


Figure 4.17: Similar to figure (4.16) the PID control uses gains from table (4.2) including defined output saturation limits to reduce the amount of over- and undershoot received. The (left) plot shows better step response from output heading control with less amount of overshoot and better accuracy of following the heading error input signal. The (right) plot shows somewhat similar results to figure (4.14), with not much overshoot decreased between the input depth error and output depth control.

Chapter 5

Discussion

A proportional-integral-derivative(PID) controller is the most widely used controller structure in industrial applications. It's simplicity of construction and sufficient ability to solve many practical control problems has contributed greatly to a widespread acceptance. Over the past decades, many PID design techniques have been proposed as seen in [34] and [7] for industrial use and for ROVs, respectively. Most of these design techniques are based on simple characterization of process dynamics, such as first order models with time delays. Nevertheless, there a few generally accepted design approaches for higher order installations such as ROVs. To obtain robust performance, one must overcome one of its fundamental issues in control, which is to synthesize a controller whose closed-loop system is internally stable and meets the required specifications despite factory uncertainty.

ROV differential equations can be solved by ODEs such as ODE23, ODE45, ODE23s, ODE113 or ODE15s. However, some of these solvers can cause problems as many of them rely on computing gradients, which are the changes in the values of functions when perturbed, i.e external or internal noises afflicting the controlling systems output. When the value of the function is determined by the numerical solution of the ODE, the value of the function may not smoothly depend on the disturbance. This means that if you move the disturbance a little up, the function will move in one direction, if you move it up more, the function may move in the opposite direction. This outcome is caused by the variable step size of the ODE solver. However, errors can be eliminated using a fixed-step solver which might affect the accuracy of the result, but still give reliable control system. For this paper, the solver options used are: fixed-step size, ODE4(Runge-kutta solver with relative tolerance 0.01).

In figure (4.13) we obtain our first results with the finalized PID gains from table(4.1), through the depth and heading PID controllers linked to the mux that outputs the inverse thrust allocation matrix (*Tinv*). The ROV seems to not be able to follow the defined path(6 red circles) due to the step response observed from the PID controllers output, which does not to align well with the desired input. We observe large overshoot between the

input values(desired and error) from both the depth and heading controllers as seen in figure (4.14). We attempt to improve our PID by manually changing the controller gains from trial and error, resulting in final chosen gains in table (4.2), in addition we set output saturation limits on both the depth and heading PID controllers in simulink of, depth upper:0.0023m and lower:-0.009 and heading upper: 1.6m and lower: -1.5m, with the goal of reducing the amount of rise time and overshoot between the input error and output step response. This shows to give somewhat better outcomes as seen in figure (4.16) and in figure (4.17) for one PID controller, we observe the improved signal step responses of the heading PID control, while the depth control still shows to remain in similar state as before in figure (4.14). In figure (4.17), we obtain step response results from the PID controllers linked to the 6 input Simulink mux block that outputs the inverse thrust allocation matrix (*Tinv*). We also observe that the ROV is able to almost follow all of the 6 placed circled on the XY plane, in figure (4.16), however still goes off course right before the reaching the 4th circle.

Remotely Operated vehicles that make use of the Wi-Fi protocols for communication should have a secure way of transferring and storage of data. It is the administrators and consumers tasks to ensure that their data is encrypted when at rest, in transit, or in process. In a situation when the encryption is faulty implemented or not strictly enforced, it opens for possibility for data vulnerabilities and creates further security issues [35]. In relation to broken access control, identification and authentication failures, underwater drones could for instance have no password used for their Wi-Fi network, which allows for vulnerability against attacks if no password is configured by administrators or consumers, although Dory avoids this problem by already having a default set password.

There are very few cases of previous known attacks in relation to maritime IoT robotics, as was shown from figure (3.19) of the STRIDE model, where some may involve GPS data spoofing/jamming, ROV data transfer spoofing/jamming, DoS(Denial of Service), man-in-the-middle attacks, data stream interception, file system back-doors, full drone control, wifi-password cracking and reverse engineering application controllers, also mentioned in [12]. GPS jamming/spoofing attacks are possible when GPS signals are unencrypted and unauthorized for the user to configure, which creates opportunity for drones to be vulnerable to spoofing attacks, where an exploiter can generate fake GPS signals and transmit them to the drone with the intention of altering the original coordinated position steered by the user. In addition to spoofing, GPS signals can also be jammed, this involves the external navigation information received by the drone's GPS receiver(user) tampered with, causing disorientation and possible accident of the drone.

In relation to Denial of Service(DoS) attacks, attackers make use of them to prevent legitimate users from accessing the ROV's service. In this case by denying the users ability to connect to the drone due to multiple connec-

tion requests sent by the attacker, leading to the drone blocking all requests sents, even those from the user. Instead of open Wi-Fi access configurated ROVs chosen by ISP providers and drone company for it's simplicity and speed of access, alternative stronger key encryption or enforced private Wi-Fi access use on ROVs, can increase the security of the drone, however might be more complicated to use.

Underwater ROVs can also fall victim to man-in-the middle attacks, where the user controlling the drone thinks they are talking to the drone, and the drone thinks it is talking to the user, while in fact there is a link established in between for the hacker who might be listening and sending commands to the drone. Similar to the man-in-the-middle attack is the interception of data streams. This is where a drone equipped with camera usually sends video streams back to the controller in real time, which is attainable from a third party stance so long they know what port to connect to, to access the stream information. These types of attacks are commonly carried out over unsecured communication links using cheap softwares such as SkyGrabber, a product used to intercept satellite signals [19]. A downside to this attack is that data streams from interceptors may seem innocuous, and are difficult to detect or even defend against, However to prevent this, the video stream data can be encrypted such as in the Dory case, where an unknown encryption protocol used on its UDP packets is difficult to find as well as decrypt.

In the case of wireless network password cracking, it is commonly known that Wi-Fi has some known flaws, where one of the major known is the WPA-SPK encryption. In a given scenario, an attacker could disable a device on the network and capture the WPA handshake transmission, that contains the hashed network password. The attacker can then follow up with a dictionary attack or brute force attempt set to identify the password based on his own generated passwords, when a match is found, the password is attained. As for Dory that uses WPA2 encryption, a more secure protocol opposed to the WPA, had EAPOL handshake missing from the captured packets by aircrack-ng. This prevents further brute forcing of its password. In regards to reverse engineering and the controller application, the attacker can inspect the application code, often residing in the software development kit(SDK) or the android application package kit(APK), that establishes communication links and controlling features between the drone and the user through a phone application. In previous attacks [15], it is possible to recover structures, java clauses, and libraries to encrypt and decrypt drone log data. Although not presented in this paper, Dorys java clauses and libraries had unconventional code structures that closed possibilities of modifying unless contacting the vendors at Chasing Innovation.

Chapter 6

Conclusion

Similar to many underwater ROVs that uses Open Wi-Fi for transferring of communication signals for controls and video transmission purposes, Chasing Dory showed to be a moderately secure device regarding the OWASP top 10 list, the STRIDE model, including the Risk analysis based on the CVSS scores. As this thesis only presents work on ethical hacking between passive and active information gathering, a further step could be taken as was initially defined of obtaining Chasing Dory's log files, to then create navigation control and video feedback script, alternative to the intended Chasing GO2 Application. Seen from an ethical hacking standpoint, this step would be called the penetration testing phase, where attempts on accessing the right IP ports using socket programming [22] to achieve the log files containing IMU, GPS and Video streams, and if successfully collected and decrypted could allow more possible actions of adding AI models, such as object detection [33] or even reinforcement learning for ROV motion tracking [24] underwater.

Regarding the PID tuning of underwater ROVs motion behaviors, we made use of NTNU Minerva's hydrodynamic and hydrostatic values collected through trials of experiments in free decay in [14]. In this paper we present work where we observed and adjusted the controller gains manually to improve the ROVs motions with the goal to follow the desired path and receive a good step response based on the speed, steering, depth and heading sensors. However, future work could involve free decay experiments on Chasing Dory where accurate measures of depth sensor for instance could be modifired and improved using the simple PID model presented, in addition the further optimization of the depth controller could be achieved through use of system model identification toolbox from Simulink and Matlab [3].

Chapter 7

Reference list

Bibliography

[1] Enrico Ande. 'UUV Matlab/Simulink Model of UUV dynamics'. In: *Github Acumen.io* (2017).

[2] Eternal Angler. 'Underwater Drones Companies'. In: *underwaterdroneforum.com* (2020).

[3] Mohd Aras et al. 'Development and modeling of unmanned underwater remotely operated vehicle using system identification for depth control'. In: *Journal of Theoretical and Applied Information Technology* 56.1 (2013), pp. 136–145.

[4] Elyas Baray and Nitish Kumar Ojha. 'WLAN security protocols and WPA3 security approach measurement through aircrack-ng technique'. In: *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE. 2021, pp. 23–30.

[5] Lee Barken. *Wireless Hacking: Projects for Wi-Fi Enthusiasts: Cut the cord and discover the world of wireless hacks!* Elsevier, 2004.

[6] Kamalika Bhattacharjya and Debashis De. 'IoUT: Modelling and simulation of Edge-Drone-based software-defined smart internet of underwater things'. In: *Simulation Modelling Practice and Theory* 109 (2021), p. 102304.

[7] Eko Henfri Binugroho et al. 'EROV: depth and balance control for rov motion using fuzzy pid method'. In: *2019 International Electronics Symposium (IES)*. IEEE. 2019, pp. 637–643.

[8] Cheng Siong Chin. 'Systematic modeling and model-based simulation of a remotely operated vehicle using matlab and simulink'. In: *International Journal of Modeling, Simulation, and Scientific Computing* 2.04 (2011), pp. 481–511.

[9] cisco. 'Introduction to WPA3'. In: *WPA3 Deployment Guide* (2021).

[10] Cloudfare. 'What is the OSI model?' In: *The OSI Model breaks down network communication into seven layers. These layers are useful for identifying network issues.* (2021).

[11] Secure Communication. 'EAPol protocol - Extensible Authentication Protocol over LAN'. In: *Vocal complete Design Solutions VoIP Voice Video Fax Data* (2020).

[12] Bakkiam David Deebak and Fadi Al-Turjman. 'Aerial and underwater drone communication: potentials and vulnerabilities'. In: *Drones in Smart-Cities*. Elsevier, 2020, pp. 1–26.

[13] Marine Digital. 'How robotics is chaning the maritime industry and what kind of robots are being used in shipping?' In: *Modern robotics is spreading across almost all sectors, including the maritime industry. A huge number of online services, devices, software and, of course, robots have become the drivers of economic and social development.* (2020).

[14] YH Eng et al. 'Estimation of the Hydrodynamics Coefficients of an ROV using Free Decay Pendulum Motion.' In: *Eng. Lett.* 16.3 (2008), pp. 326–331.

[15] Pau Oliva Fora. 'Beginners guide to reverse engineering android apps'. In: *RSA Conference*. 2014, pp. 21–22.

[16] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[17] Cable Free. 'WLAN Frequency Bands Channels'. In: *WLAN Frequency Bands and Channels showing allowed wireless local are network channels using IEEE 802.11 protocols as used in Wi-Fi networks.* (2020).

[18] Mohanad M Hammad, Ahmed K Elshenawy and MI El Singaby. 'Trajectory following and stabilization control of fully actuated AUV using inverse kinematics and self-tuning fuzzy PID'. In: *PloS one* 12.7 (2017), e0179611.

[19] JP Hood. 'Understanding Hostile Use and Cyber-Vulnerabilities of UAS: Components, Autonomy v Automation, Sensors, SAA, SCADA and Cyber Attack Taxonomy'. In: ().

[20] Nisha Jiju. 'TCP/IP vs UDP: What's the Difference?' In: *Colocation America* (2018).

[21] SO Johnsen et al. 'Safety and security of drones in the oil and gas industry'. In: *Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference, ESREL2020-PSAM15 Organizers, Singapore*. 2020.

[22] Limi Kalita. 'Socket programming'. In: *International Journal of Computer Science and Information Technologies* 5.3 (2014), pp. 4802–4807.

[23] Farid Kendoul. 'Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems'. In: *Journal of Field Robotics* 29.2 (2012), pp. 315–378.

[24] Abir Khan. 'Deep reinforcement learning based tracking behavior for Underwater vehicles'. MA thesis. NTNU, 2018.

[25] Paul Kirvan. 'Open System Authentication(OSA)'. In: *TechTarget Search Security* (2022).

[26] Sumit Kumar, Sumit Dalal and Vivek Dixit. 'The OSI model: Overview on the seven layers of computer networks'. In: *International Journal of Computer Science and Information Technology Research* 2.3 (2014), pp. 461–466.

[27] Bin Li, Subhash Rakheja and Ying Feng. 'Enhancement of vehicle stability through integration of direct yaw moment and active rear steering'. In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 230.6 (2016), pp. 830–840.

[28] Evan Lubofsky. 'Underwater robots swarm the ocean'. In: *Woods Hole Oceanographic Institution* (2021).

[29] Gregory Manley. 'Why you shouldn't use WEP encryption'. In: *EngEd community Security Section* (2020).

[30] The Mariners' Museuem and Park. 'ROV'. In: *The agers of exploration* (2022).

[31] OWASP. 'OWASP Top Ten'. In: *Standard awereness document for developes and web application security.* (2021).

[32] Gustav Rubbestad and William Söderqvist. *Hacking a Wi-Fi based drone*. 2021.

[33] Behzad Sadrfaridpour et al. 'Detecting and Counting Oysters'. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2156–2162.

[34] Blas M Vinagre et al. 'Fractional PID controllers for industry application. A brief introduction'. In: *Journal of Vibration and Control* 13.9-10 (2007), pp. 1419–1429.

[35] Kazi Zunnurhain. 'Vulnerabilities with internet of things'. In: *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer ... 2016, p. 83.

Chapter 8

Appendix

Appendix A: The *rovSimRun.m* matlab file used to run the simulation for the ROV with line of sight guidance PID control.....

Appendix B: The *rovSim_los.slx* simulink model of the ROV with line of sight guidance and PID control.....

Appendix C: The *rovSimSetup.m* matlab model that set-up initial simulation time and rov object conditions.....

Appendix D: The *rov_dynamics.c* C file of the 6DOF dynamics of the ROV.....

Appendix E: The *rov_thruster.c* C file of the thrust dynamics of the ROV.....

Appendix F: The *rov_dynamics.mexw*64 mex file that compiles C code to run into Simulink and Matlab.....

Appendix G: The *rov_thruster.mexw*64 mex file that compiles C code to run into Simulink and Matlab.....