UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

NTNU
Norwegian University of Science and Technology

Control system for the Remotely Operated Vehicle (ROV) Minerva II and subsea docking using Robot Operating System (ROS)

FINAL DEGREE PROJECT

Bachelor's Degree in Telecommunications Technologies and Services Engineering.

Author: María Carmen Muñoz Oliver

Supervisor: Vicente Herrero Bosch
Co-supervisor : Martin Ludvigsen

Universitat Politècnica de València
València, July 2020

Abstract

Underwater remotely operated vehicles (ROVs) are used for different tasks in the field of biology, archaeology, and engineering. ROV Minerva II, which is the focus of this thesis, performs research for marine science and engineering. The different modules for the regulation and management of the ROV constitute the control system. For now, Minerva II has worked with an in-house control system. The current system does not allow to improve communication between processes over multiple machines. This remarks on the need to work on an inter-communication framework. Therefore, to solve these limitations it is implemented a control system using the Robot Operating System (ROS). This software facilitates the development of projects by connecting existing solutions. In the present work, a control system for the handling of Minerva II has been developed based on the values received by the simulator. This system allows the engineer to control the ROV in its 4 degrees of freedom: surge, sway, heading, and heave. In these controllers, it is remarkable the importance of parameter tuning to improve the degree of accuracy. The project carried out in this thesis constitutes the first phase to integrate completely the control system using ROS. From here, new lines of development appear, such as adding functionalities to the controllers. For instance, navigation filters. Besides, it leaves the door open to the replacement of the communication core with the graphical user interface using ROS.

Keywords: control system, underwater remotely operated vehicle, robot operating system, marine navigation.

List of Figures

List of Tables

Contents

1 Introduction

This thesis is about the control system of a remotely operated vehicle (ROV) by using the Robot Operating System (ROS). It is on the line of projects developed in the Autonomy for Underwater Vehicles Group (NTNU) and contributes to the implementation of features for the marine exploration vehicle Minerva II. The goal of the work is to develop motion controllers for a control system following the principles of underwater vehicle control theory. Likewise, the most relevant work on ROS development is done based on an in-house control system. The results presented in this document come from tests performed on the computer (PC) where the simulator was running. Below is given an overview of the motivation to carry out this project. Later on, is detailed the current state of knowledge as well as the problem outline.

1.1 Motivation

ROVs are increasingly present in different sectors today. In the experimental field for academic purposes or the business field. They can be found in areas such as agriculture, automotive, industry, or maritime navigation, among others. Each sector uses ROVs with particular characteristics according to their purpose. Therefore, it is possible to find all kinds of shapes in these vehicles. However, its physical design is not the only difference to underline. Remotely operated vehicles are used for specific applications. Thus, they must have designed functionalities such as temperature, depth, or heading control. This control is often automatic instead of manual for greater precision. This difference increases when it is made a distinction between surface ships and underwater ships with the dynamic positioning (DP) development of the former being much more advanced [1].

For the case of underwater ROVs, a difficulty is found for depth control due to the complexity of the aquatic environment and possible disturbances. Other challenges are about unmodeled effects and hydrodynamic complexity. Many of these ROVs are used for inspection, maintenance, and repair tasks of subsea petroleum facilities [2]. Such complex applications require a complete and precise control system for the ROV. Thus, the control system for this underwater vehicle contains functions for I/O, signal processing, navigation estimation, controllers, thrust allocation, guidance signal, and mission management. An efficient implementation is required to ensure proper timing while keeping the code structured.

The main challenge is to enhance communication between processes over multiple machines so that researchers can assemble a complex system by connecting existing solutions for small problems [3]. For this purpose, ROS is the tool used. The mentioned challenge, the chance of contributing something additional to the field on advanced control, and underwater navigation are the key factors for working on this topic.

1.2 Background

In this section, an overview of ROV definition and the current in-house control system is given. ROS is defined as follows.

1.2.1 ROV definition

The acronym ROV stands for Remotely Operated Vehicle. In this case, an underwater remotely operated vehicle. It is a tethered ROV that makes possible to explore the ocean by using a joystick to control the vehicle. Even they come in different shapes they have common elements. These elements are thrusters, cameras, lights, a tether, a frame with buoyancy tools, and the pilot controls.

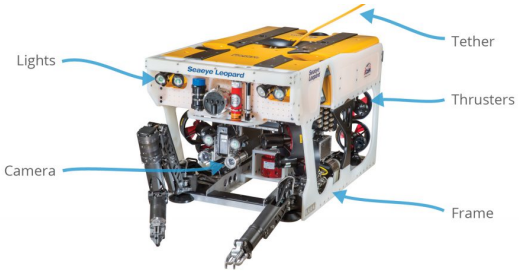

Figure 1: ROV parts (Credit: Saab Seaeye)

Minerva II is a ROV that was acquired by NTNU AUR-lab[1] to take part in researches for marine science and engineering. The vehicle can also carry sonars, cameras, and other instruments to document the subsea environment. For engineering research purposes the ROV is used as a test platform on research for underwater computer vision, navigation, guidance, control, and autonomy.

1.2.2 Current control system

ROV Minerva II works with an in-house control system developed on LabVIEW. LabVIEW, which stands for Laboratory Virtual Instrument Engineering Workbench is a software development environment for applications that require test, measurement, and control with rapid access to hardware and data insights [4]. The platform is single-sourced and not standardized.

The current system implements several modules for ROV such as the Autonomous System, Observer, Thrust Allocation, Controller, Hardware-In-Loop (HIL) Simulator, Guidance System, and Graphical User Interface. Communication between systems is essential for sending messages. For that purpose, User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are used. The control system itself is called Njord. Njord, in addition to HIL Simulator

[1] https://www.ntnu.edu/web/aur-lab

Verdandi and Graphical User Interface Frigg, defines the functional control system. This thesis focuses on the HIL Simulator module messages. The control system can be connected following the same communication to either the ROV or the HIL [5]. Communication between the control system and the simulator uses UDP messages. As mentioned, HIL Simulator can replace ROV for testing the control system. This way, the simulator sends back and forwards measurements like roll, pitch, yaw, north, east, down, etc.

1.2.3 Robot Operating System (ROS)

ROS is an open-source, meta-operating system for robots that work on top of Linux Ubuntu. ROS facilitates robot programming by imitating the operating systems for computers. This operating system provides services such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management as well as tools and libraries for obtaining, building, writing, and running code across multiple computers [6].

Its structure of little subsets of tasks, known as nodes, allows solving programs using small tasks in larger projects. The reason is that nodes communicate with each other. By using ROS, one can control a robot from anywhere. Furthermore, if nodes connect by a network (e.g. VPN), they can run on different hardware.

1.3 Problem Outline

Relevant aspects must be taken into consideration during the work.

- Surge and sway motions are relative to the body-fixed frame and the values measured by the sensors are in terms of the Earth-fixed frame. Due to this, it is necessary to perform frame rotation.
- When ROV can't be used to test is still necessary to run the simulator in LabVIEW. Thus, the ROS control system is not independent for now.
- The gain parameters of the controllers determine the force applied on the thrusters. If these values are not adjusted, the ROV would not move to the desired values. This condition implies that tuning is challenging.
- Another aspect that potentially influences the action of controllers is the wind-up effect. This is problematic since the controller might saturate due to the error that accumulates the integrative term.

1.4 List of Publications

Gao, F., Shen, S., Moltu, S., Vollan, E. R., Muñoz Oliver, M. C., and Ludvigsen, M. (2020). Increased autonomy and situation awareness for ROV operations. (in prep)

2 Method

In this section, it is explained the development of the blocks that constitute the project. In the course of the thesis, control theory for underwater vehicles played an important role. As a marine craft can experience motion in six degrees of freedom (DOFs) [7], motion controllers were the main point in software development in this project.

Since Minerva II is stable in roll and pitch, there was no need to control in those degrees of freedom. Therefore, the model downgraded to 4 DOFs. Thus, a model on the horizontal plane whose displacements referred to longitudinal motion (surge), sideways motion (sway), and the rotation about the vertical axis (yaw or heading) in addition to a vertical motion (heave). The algorithm for the PID controller and a coordinate converter was also necessary for the operation of the controllers.

Following, tools and software used in the thesis are presented. In addition, the foundations are established in terms of control theory as well as communication between main modules in ROS. Finally, there is a detailed section on the operation and structure of the controllers. Later on, the explanation of the code written for that purpose is given.

2.1 Tools and Software

- ROS Melodic running on Linux Ubuntu 18.04 LTS
- PyCharm Integrated Development Environment (IDE) running on Linux Ubuntu 18.04 LTS
- MATLAB (MathWorks)
- NI LabVIEW 2018 (National Instruments)
- Personal computer (PC) running LabVIEW and ROS
- Xming server

Python version 2.7.17 was used along with the Pycharm IDE.

2.2 Reference frames

The main contribution of this thesis is the transfer of the current control system of the ROV Minerva II to the ROS framework. Although it is part of an existing system, it is convenient to define important aspects to take into account. On the one hand, knowledge about reference frames is essential to understand the values collected by the ROV sensors and, where appropriate, transform them into another reference frame. When dealing with the motion in a marine craft two reference frames need to be distinguished: Earth-Centered and Geographic. In turn, Geographic frame is defined by two systems called NED and BODY:

- NED: Nort-East-Down $\{n\}$ = (xn, yn, zn). This system is defined relative to the Earth's reference ellipsoid [8].

- BODY: $\{b\}$ = (xb, yb, zb). It is the body-fixed reference frame which is fixed to the craft [9].

The Earth-Centered reference frame is also composed by two systems:

- ECI: The Earth-centered inertial frame $\{i\}$ = (xi, yi, zi) is for terrestrial navigation and is fixed in space [10].

- ECEF: The Earth-centered Earth-fixed frame $\{e\}$ = (xe, ye, ze) has its origin fixed to the center of the Earth but the axes rotate relative to the inertial frame ECI [11].

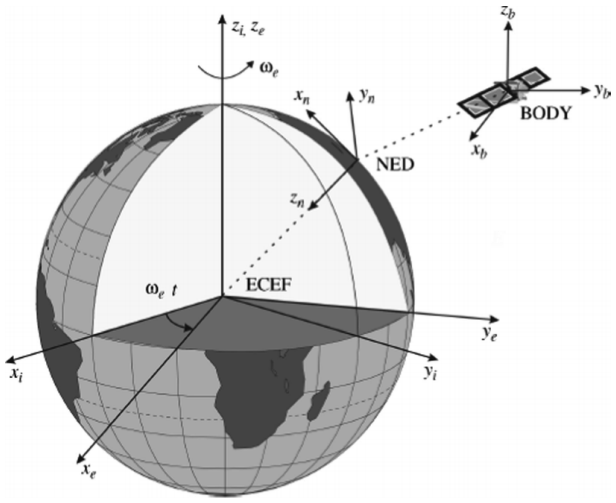

Figure 2: Reference frames [12].

2.3 ROS Core

For the control system to send orders to the simulator or ROV, a module was required acting as a gateway to send and receive messages between them. As the purpose was to work with ROS, a module called minerva2_iface (see Appendix) had already been implemented in that framework.

The interface consists of 3 modules to allow communication between the simulator and the control system. One of them establishes a connection with port 4101 to read simulated attitude values of Xsens Mti and stores them in the Mtipro message. Another one reads the position and speed data from 4302 Navipac port which is then stored in Navipac message. All these values were used within the motion controllers. Finally, a module reads and stores the values the control system wants to send to the simulator.

2.4 Software development

The files in which both motion and PID controllers have been developed are part of the minerva2_control project (see Appendix).

2.4.1 Motion Controllers

A controller is defined as a module that receives an input signal from a measured process variable and compares it to a set point. After this, computes the necessary output value for corrective action in the control loop. Thus, controllers are the key to this thesis.

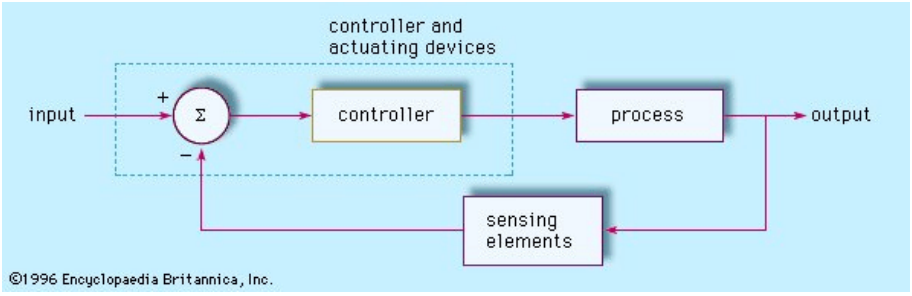

Figure 3: Basic control system structure [13].

The following sections detail the operation of each controller to implement 4 DOFs on the ROV Minerva II.

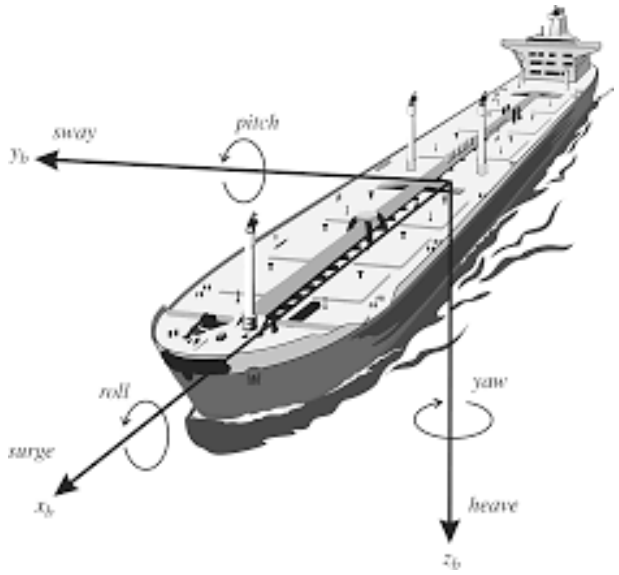

Figure 4: Degrees of freedom in marine craft [14].

Heave

Marine craft motion in the z-direction is called heave. This controller was programmed in the control_heave.py file. In the first lines of code, it was imported the Python client library for ROS, rospy. It was also imported the PID controller, and, from the project minerva2_iface, the message containing the simulation values (Navipac) as well as the corrective action message (Forcevector). Right after, Depth class was created to declare class variables and be able to access them from any method of the same. Within __init__ method, all variables used in the classes were initialized to 0 in addition to creating the PID object and the Publisher for topic ROV/In/Desiredforce in which the corrective forces were stored. For tuning purposes, Kcr and Pcr (see section 3.1) variables were declared as well. It was also created a setpoint variable to store the desired depth value.

In the callback method, it was developed the main algorithm for calculating the corrective action. This method received the data that the listener method read. Position values 'down' and speed 'w' were stored. After this, it was computed the difference subtracting the desired heave to the one measured by the sensor. Lately, difference and speed values were sent to the PID method (see section 2.4.2) such that its output was stored in the force variable from class Depth. Later on, Forcevector was created and all its variables (fx, fy, fpitch, froll, and fyaw) were filled with zeros except fz. PID's output force was assigned to fz. Then, the message was published.

In the listener method, a node with the same name was started to subsequently subscribe to topic ROV/Out/Sensor/Navipac, read the Navipac message, and enter the callback loop.

Finally, the code was concluded creating object DP for Depth class initialization and thus, to deal with possible errors using ROSInterruptException.

Heading

Heading has to do with rotation about the z-axis so that involves the yaw angle. The procedure and structure of the code were the same as in the heave controller. In this case, the class created was named Yaw. For this motion, Mtipro message was imported instead of the Navipac since the simulation values of the yaw angle were needed.

Both the listener and __init__ method were as in control_heave but subscribing to topic ROV/Out/Sensor/Mtipro. Thus, error handling was done on Yaw class 'listener'. The callback loop was structured in the same way as in the heave controller and the other controllers, but it incorporates some lines of code necessary to handle heading more efficiently. Using a while loop the error value was corrected if it is greater than 180 degrees or less than -180. In the first case, 360 is subtracted (corresponding to a complete turn) from the current error. In the second case, 360 is added. Then, returns to the head of the loop to check if it meets the condition. In this way, it was determined if the rotation occurs clockwise (positive difference) or counterclockwise (negative difference). Thus, if the set point is reached by applying a lower force counterclockwise than clockwise or vice versa it will do so counterclockwise.

Surge and Sway

Motion in the x-direction is called surge and in y-direction sway. Both were coded with the same features and key aspects. Here in this section, those aspects are going to be explained. The surge controller code is located in the file control_surge.py and the sway controller in control_sway.py (see Appendix). As in the rest of the controllers, they have a (Surge) class made up of the __init__ method to initialize variables, the listener method to initialize the node and subscribe to topics, and the callback loop for the main algorithm. In this case, two callback loops were defined in the class. One for each topic to which the node was subscribed.

For calculating the corrective action, an important aspect related to surge and sway had to be considered. Both movement actions are relative to the body-fixed frame and not the Earth-fixed frame. Since the values received from the simulator (North-East) are concerning the Earth-fixed frame, it was necessary to do a rotation with heading (yaw) to obtain the body-fixed coordinates. This transformation consisted of multiplying the North-East vector by the rotation matrix. This way, the coordinates were defined according to the following equa-

tions:

$$rx = North \cdot cos(\theta) + East \cdot sin(\theta) \tag{1}$$

$$ry = -North \cdot sin(\theta) + East \cdot cos(\theta) \tag{2}$$

Where θ is the heading angle positive clockwise.

The first substantial difference for the heave and heading controllers is that, in surge and sway, the simulation values of Navipac and Mtipro were needed. Thus, two callback loops were defined to read the message of each topic and store the value of Yaw, North, and East. In the loop in which North and East are received, the call to the PID controller and the publication of the topic is introduced. Before calculating the error, the origin point of North and East was changed by subtracting North_0 and East_0. North_ and East_ received the corresponding value of North and East in the first iteration of the loop.

The next step was to transform the simulated yaw value into radians since the 'math' function used in the rotation vectors works in radians and not in degrees. Subsequently, the calculation of the new body-fixed coordinates was performed using equations (1) and (2) and the error or difference was obtained for each controller (using rx for surge and ry for sway). The difference value along with the speed data ('u' for surge, 'v' for sway) was sent to the PID method to calculate the corrective force.

2.4.2 PID Controller

A PID can be defined as a closed-loop controller architecture used as an instrument to regulate process variables. PID stands for Proportional, Derivative, and Integral controller. In this project, it is used to calculate the necessary corrective force to be sent to the ROV thruster or simulator. Its structure is as follows.

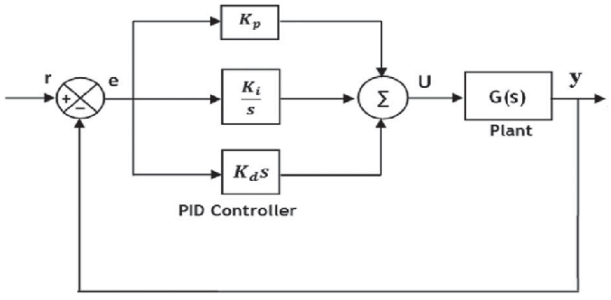

Figure 5: Basic PID block diagram within a closed loop system [15].

As seen in the block diagram above, the PID controller is made up of three corrective terms whose sum gives a result of the corrective output:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt} \tag{3}$$

The proportional term depends on the difference value and produces a value that is proportional to the difference value. This term can be modified by adjusting the proportional gain called Kp. The integral component has to do with the error term over time. It can be also adjusted by the gain Ki. The response of the derivative term is proportional to the rate of change of the simulated value. Its gain Kd can adjust its output.

In the code for this project, the gains were defined when creating the PID class object in the motion controllers. In the call to the newpid method (see Appendix) of the PID, the error or difference was sent and, when applicable, the speed data (surge, sway, and heave). Within this method, the corrective output was calculated according to equation (3). In the case of heading, a low-pass filter was used to filter out noise in the measurement and determine the displacement in space.

On the other hand, the wind-up problem presented by controllers with integral action was taken into account. Thus, it leads to overshooting due to the error that accumulates the integrative term when the setpoint suddenly increases or decreases in excess. This leads to controller saturation.

To avoid the effect, an if-else loop was added. In this loop, the value of the integral term was limited both when its value was positive and negative. Finally, all the terms are added and the result is returned to the controller.

2.5 Project running

The interaction between the simulator and the control system implemented with ROS was possible by following the steps given. First of all, on the PC, it is necessary to open the Verdandi simulator in LabVIEW and run it. Then, in Ubuntu, the launch file of the minerva2_iface project is executed using the 'roslaunch' command so that the interface begins to manage communication. After this, the controller was run in another window using 'rosrun'. The results were displayed on the 'Plot' tab of the simulator. One can check also the values typing 'rostopic echo' of the topic in which we want to see the values.

3 Results

3.1 Ziegler-Nichols tuning

In order to make the operation as accurate as possible, Ziegler-Nichols closed-loop tuning method was chosen. Thus, following the table below: Where Kcr

Control Type	Kp	Ki	Kd
P	0.5Ku	-	-
PI	0.45Ku	1.2Kp/Pu	-
PID	0.6Ku	2Kp/Pu	KpPu/8

Table 1: Ziegler-Nichols closed-loop method

is the critical gain and Pcr the critical period. The most relevant results of the action of the controllers in the 4 degrees of freedom are shown below. It has been simulated for an ideal case so no disturbances were introduced. Different gains were applied to visualize how simulations varied in their behavior. The post-processing of the data and its subsequent representation has been done using MATLAB.

3.1.1 Heave

For all simulations here, the same set point of 30 m was established.

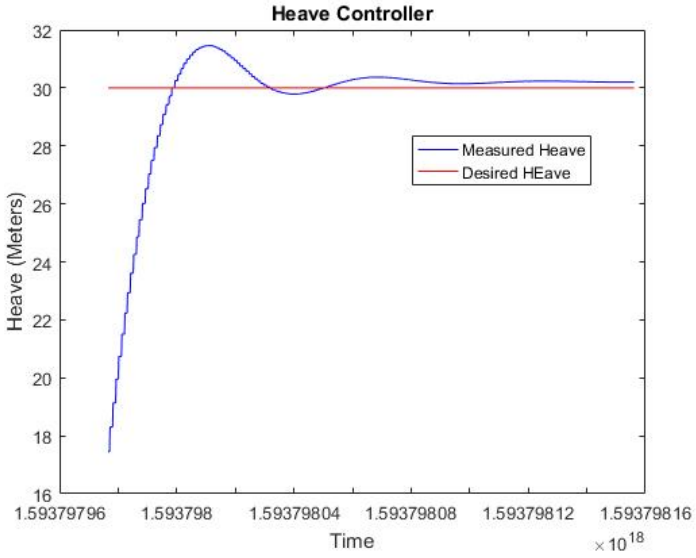

Figure 6: Heave over time. Kp = 120 Ki = 0.5 Kd = 0.125

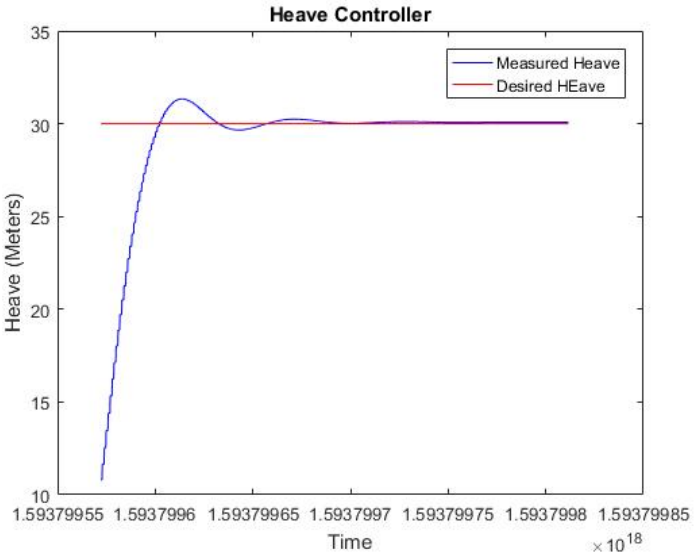

Figure 7: Heave over time. Kp = 120 Ki = 0.015 Kd = 0.00375

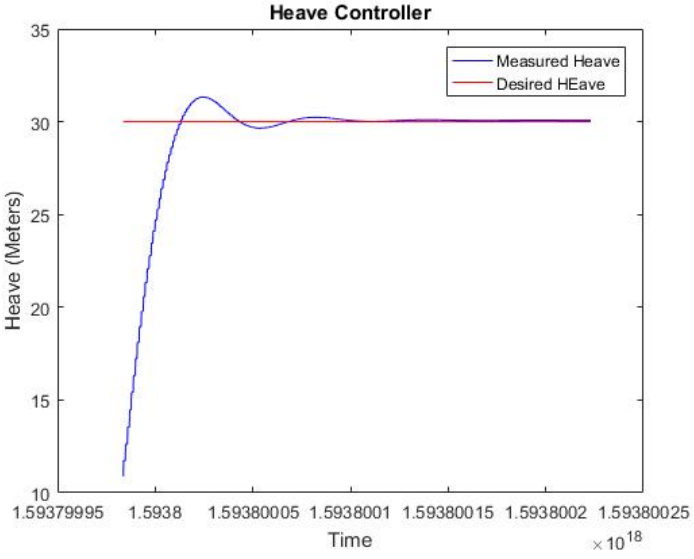

Figure 8: Heave over time. Kp = 120 Ki = 0.0125 Kd = 0.003125

3.1.2 Heading

For heading, 180º was established as set point for all different gains in this document.

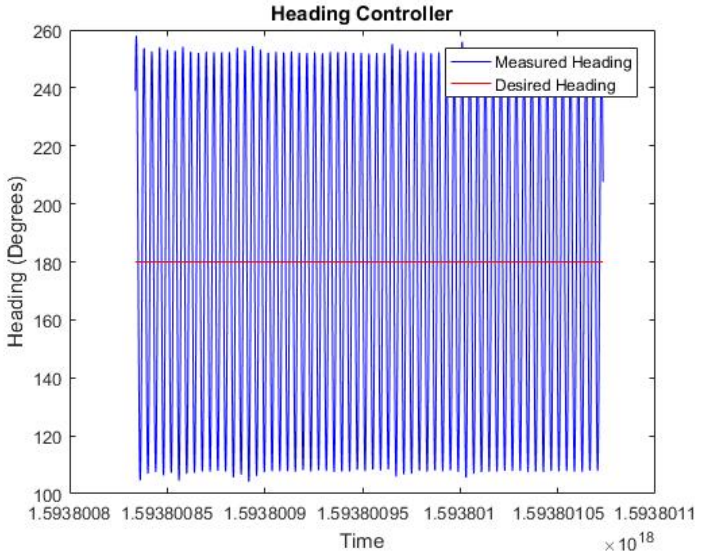

Figure 9: Heading over time. Kp = 60 Ki = 0.015 Kd = 0.00375

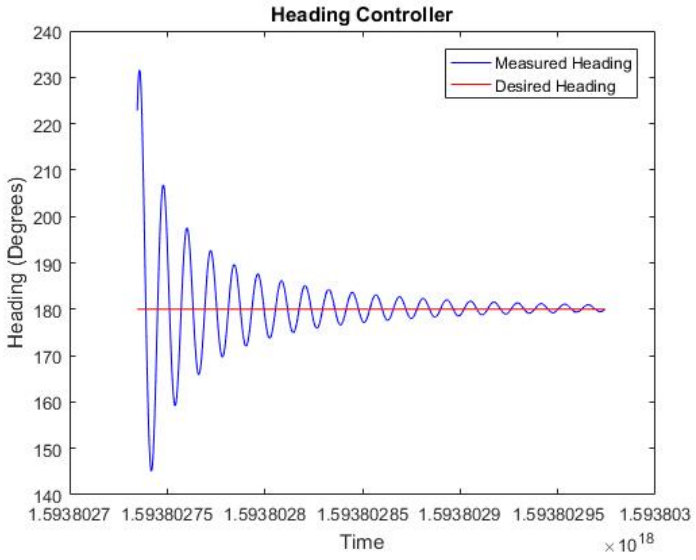

Figure 10: Heading over time. Kp = 6 Ki = 0.015 Kd = 0.00375

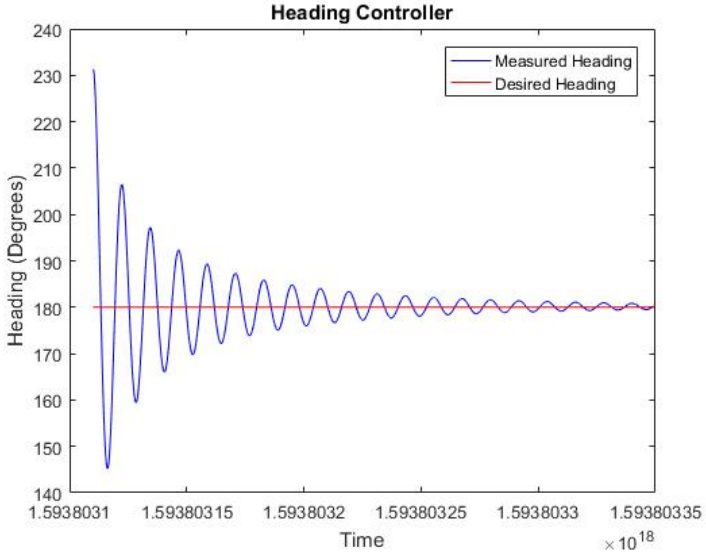

Figure 11: Heading over time. Kp = 6 Ki = 0.01 Kd = 0.025

3.1.3 Surge

For checking this controller, a set point of 4 m from the origin was assigned. Thus, a desired surge of 7036897.5 m.

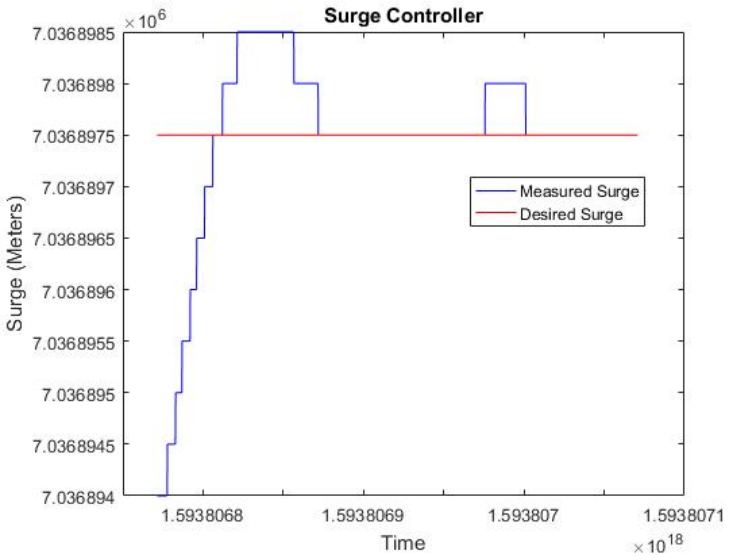

Figure 12: Surge over time. Kp = 6 Ki = 0.015 Kd = 0.00375

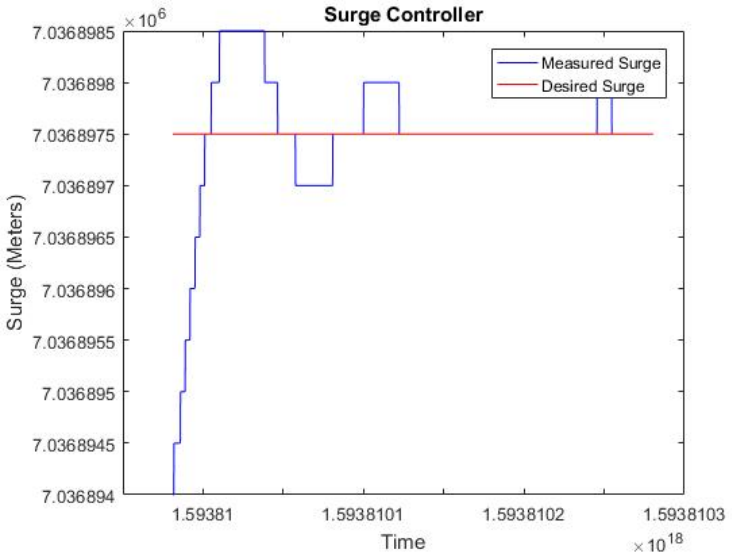

Figure 13: Surge over time. Kp = 12 Ki = 0.015 Kd = 0.00375

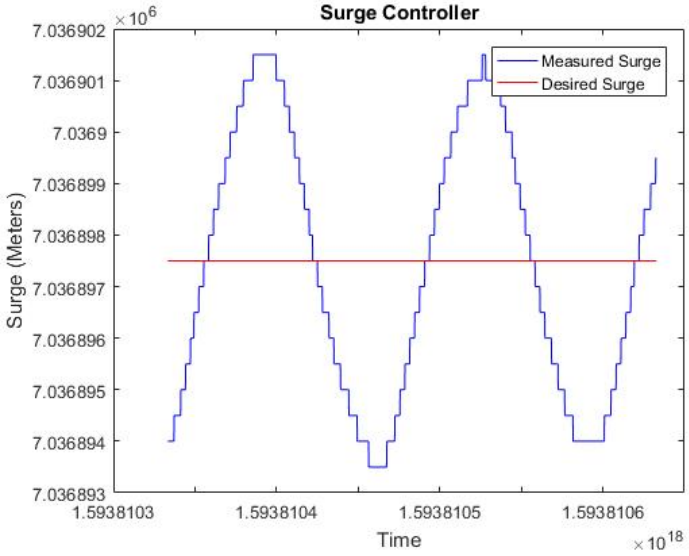

Figure 14: Surge over time. Kp = 6 Ki = 0.5 Kd = 0.125

3.1.4 Sway

The set point for the simulations in this document was 570131.75 m.

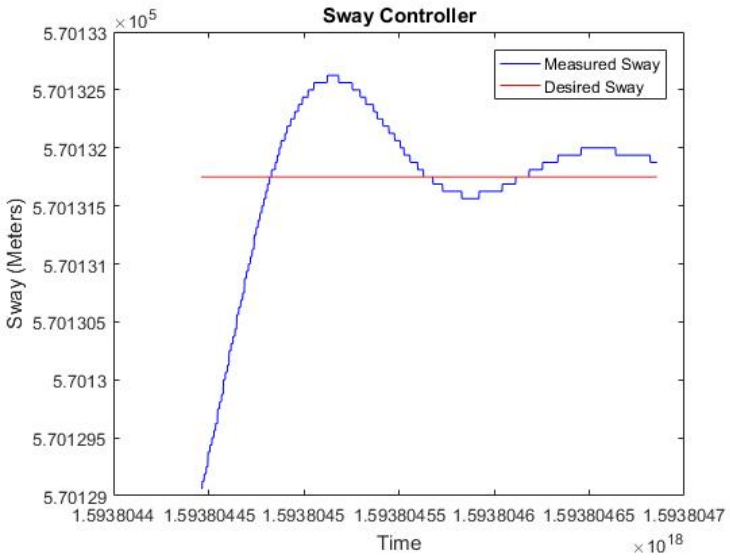

Figure 15: Sway over time. Kp = 6 Ki = 0.015 Kd = 0.00375

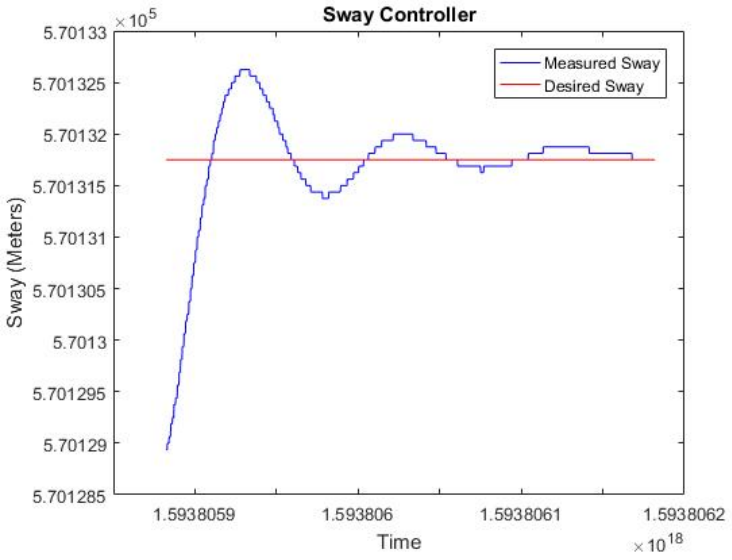

Figure 16: Sway over time. Kp = 12 Ki 0.01 = Kd = 0.0025

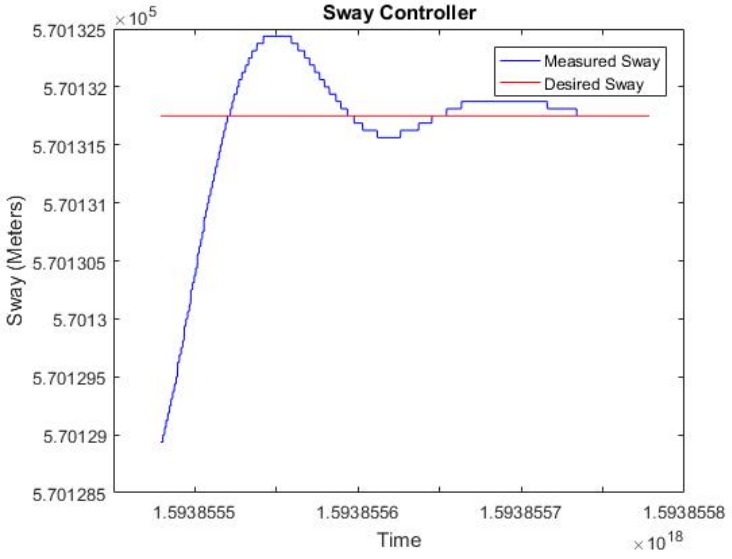

Figure 17: Sway over time. Kp = 6 Ki = 0.005 Kd = 0.00125

3.2 Other remarkable results

For surge and sway movement, when a navigation filter is not implemented the setpoint regulation is not suitable for use over long distances (> 5m).

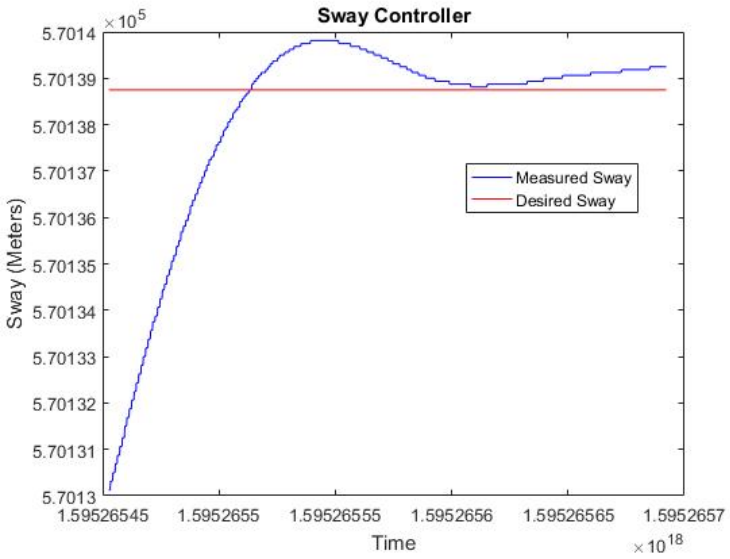

Figure 18: Sway over time. Kp = 6 Ki = 0.005 Kd = 0.00125. Setpoint is 10 m

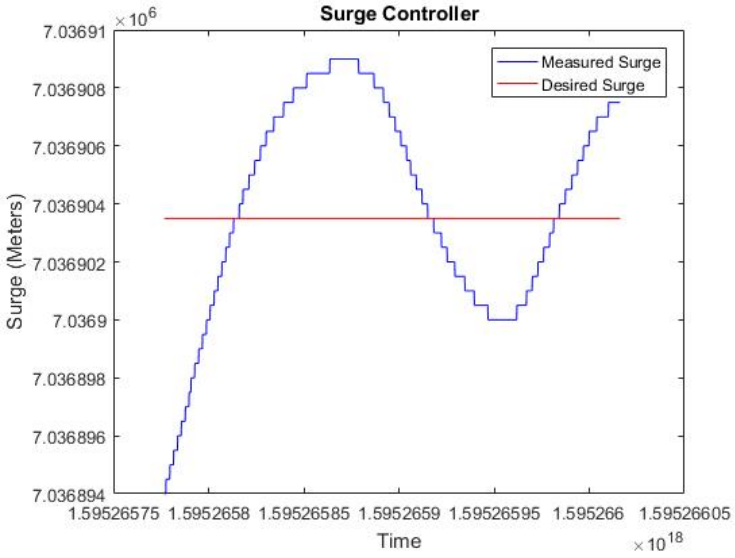

Figure 19: Surge over time. Kp = 6 Ki = 0.5 Kd = 0.125. Setpoint is 10 m

4 Discussion

4.1 Heave

The general behavior observed in figures 6,7 and 8 is stable and hardly fluctuates for all the values tested. In the case of a worse tuning adjustment, the deviation from the setpoint is not very large but reaches a maximum of 30 cm. As integrative and derivative gains decrease, the difference between the setpoint and the measured heave decreases. In this way, the values in Figure 8 were chosen as optimal for the controller. Therefore, it can be highlighted that the controller performs its function correctly. Furthermore, the high value of the chosen proportional gain resulted in a quick response to the system.

As shown in the figures, the controller performed its function with accuracy according to the values of the gains. This degree of precision was generally quite high. Thus, the deviation to the setpoint is in the order of centimeters (20 in the worst case). In part, this may have been due to the correct adjustment in gains. However, with the best adjustment, a deviation of 4 cm to the setpoint is obtained. Therefore, to achieve a better adjustment, the value of gains should be slightly modified in its thousandths and hundredths. On the other hand, when the environment is not simulated and the control system runs on the ROV, the disturbances will be uncontrollable. Furthermore, the ROV may have mechanical limitations so this must be taken into account when adjusting terms.

4.2 Heading

Heading value. When the proportional gain is high, the oscillation period decreases considerably giving rise to a constant oscillating steady state. However, with low proportional gain values on the order of 1 to 10 the controller generates a damped wave for the same integrative and derivative values. Therefore, the great influence of Kp on the rotation of the boat about the z-axis is highlighted. The movement, as seen in figures, is stabilizing more slowly than in heave motion. As moving within the time axis, the oscillation decreases in amplitude around the set point.

4.3 Surge

The surge controller behaves as shown in Figures 12, 13, and 14. First, it is denoted that the values over time do not have the characteristic sinusoidal shape. In this case, any value used for all the gains leads the ROV/Simulator to remain in the same position for a longer period. For this reason, the figures for the simulated values have a striking appearance. This is due to measurements made using a positioning system called ultra-short-baseline (USBL). USBL has a low update rate. This system allows a ROV to localize itself relative to a surface ship [16].

On the other hand, the oscillation is greatly reduced when derivative and integrative gains are decreased (figures 12 and 13). Thus, proportional gain does not have as much influence for this controller.

As shown in the results, the controller corrects the surging value to reach the set point. However, when the signal does not oscillate, spikes suddenly appear. These disturbances occur when the measured value has already reached the set point. These spikes may be due to the high noise susceptibility of the USBL system.

Finally, it is necessary to highlight the limitation of this controller in long distances (> 5m) since it does not incorporate a navigation filter. The navigation filter would be used to map the path to the desired position. In this way, the ROV would move along the line at a certain speed. Thus, the desired position would be always close to the robot, but moving towards the target.

4.4 Sway

The trend observed for all simulations performed was a damped wave of a large period. It is shown in the figures that the control system can follow the desired position even disturbances appear as well. By increasing the proportional gain the rise time was decreased and the integrative was adjusted to stop the oscillations. Since overshoot appeared the derivative gain was decreased to reduce it.

As with the surging movement, sway measurements come from the USBL system. In this way, the characteristic shape of the figures is also due to the low update rate. It is also limited to short distances. Therefore, it is necessary to implement a navigation filter as it happens in the surge controller.

In essence, independent results of each controller show that the control system behaves like a whole. It is significant how tuning influences in some controllers more than others. For example, adjusting the parameters in the depth controller does not drastically influence the output values. The results shown in the graphs correspond to a system tested in an ideal environment without disturbances. This means that, for the ROV Minerva II, the corrective forces will be different for the same setpoint and tuning since external forces appear in the physical environment. Therefore, for testing in Minerva II, tuning should be needed.

5 Conclusion

The control system has been developed for ROV Minerva II. In this way, the joystick has been replaced by controllers to interact with the simulator and manage the marine craft in 4 DoFs. This system is the beginning of the migration to ROS of the current in-house control system.

When the thesis was proposed, it was intended to also cover communications with the graphical user interface. However, it was finally decided to focus the topic on the control system interacting only with the simulator. The reason for this change was the motivation to go deep into the development of the controllers using the values received by the simulator. In this way, lines of development are opened for future work around communications with the interface through ROS.

During the development of the project, we have worked with the LabVIEW and Pycharm IDE platform, being necessary for the integration of the system with ROS. Although the work has focused on software development, there are several areas involved in the topic of the thesis. First, the study of control theory was deepened. Here it was especially important to know the operation of the PID controller in detail. On the other hand, since the control system was implemented for an underwater ROV, the reference frames in the navigation field had to be studied.

Finally, it is remarked that the control system that has been developed is functional and allows the ROV to be controlled. However, it is the starting point for full implementation with ROS. It is possible to add features such as the navigation controller. This controller would be used to move Minerva II to a specific position (surge and sway) when a speed value is given. One could also add the navigation filter on sway and surge controller and so it can manage for long distances. For future projects in the field, this thesis serves as the basis from which to expand the ROS control system. The next step would be develop communication with the graphical interface and its control, as mentioned above.

Appendix

The code used in this thesis is given below.

- minerva2_iface project code: https://github.com/toremobjo/minerva2_iface.git
  Author: Tore Mo-Bjørkelund. Contact: tore.mo-bjorkelund@ntnu.no

- minerva2_control project code: https://github.com/mamuool/minerva2_control.git
  Developed by the author of this thesis. Contact: munozmaika@gmail.com

References

[1] Dukan, F., *ROV Motion Control Systems*, Doktoravhandlinger ved NTNU (trykt utg.). Norwegian University of Science and Technology, Department of Marine Technology, 2014.

[2] Henriksen, E. H., Schjølberg, I., Gjersvik, T. B., *Adaptable Joystick Control System for Underwater Remotely Operated Vehicles*, IFAC-PapersOnLine Volumes, 49(23):167-172, 2016.

[3] Tavares, P., Sousa, A. (2015). Flexible pick and place architecture using ROS framework. 2015 10th Iberian Conference on Information Systems and Technologies, CISTI 2015. 10.1109/CISTI.2015.7170602.

[4] Retrieved from: https://www.ni.com/en-no/shop/labview.html

[5] Holven, E. B., *Control System for ROV Minerva 2*, Master thesis, partment of Marine Technology, 2018.

[6] Retrieved from: http://wiki.ros.org/ROS/Introduction

[7] [8] [9] [10] [11] [12] [14] Fossen, T. I. (2011). *Handbook of marine craft hydrodynamics and motion control (First edition).* Trondheim, Norway: John Wiley & Sons Ltd.

[13] *Encyclopædia Britannica*. https://www.britannica.com/technology/feedback-control-electronics#/media/1/1133863/691, Accessed: 26.07.2020

[15]Saravanan, V., Kumar, R. B.. (2019). Mathematical modelling and controller design using electromagnetic techniques for sugar industry process. Automatika. 1-8. 10.1080/00051144.2019.1653667.

[16] Paull, L., Saeedi, S., Seto, M., Li, H. (2014). *AUV Navigation and Localization - A Review.*