Research Internship (PRE)

Field of Study: Robotics
Scholar Year: 2019-2020

Aquanaute

Discovery of Available Boat Elements to Simulate and Build a Catamaran Capable of Autonomous Driving

Confidentiality Notice
Non-confidential report and publishable on Internet

Author:
Ricardo RICO URIBE

Promotion:
2021

ENSTA Paris Tutor:
Thomas SIMON

Host Organism Tutor:
ENSTA Paris

Internship from 18/05/2020 to 31/07/2020

Name of the host organism: ENSTA Paris
Address: 828 Boulevard des Maréchaux
Palaiseau
France

Confidentiality Notice

This present document is not confidential. It can be communicated outside in paper format or distributed in electronic format.

Abstract

The first step to a simulation in Gazebo is understanding ROS, to make a correct maritime simulation it is necessary to include all expected forces and apply them to a boat, this was accomplished by using the VRX plugin. To allow for autopilot, a NAVIO2 running Ardupilot Rover software was used, and a plugin connecting the simulation with autopilot was implemented. Finally the boat was constructed with elements that the laboratory already had and we achieved a correct automatic driving. There were a lot of problems during development but they were quickly identified and solutions were proposed, but not implemented, because of a change in plans caused by the pandemic. Originally the project comprised on the construction of the boat, the closing of the laboratory shifted the objective to be only focused on the simulation, but the re-opening made the project to comprise of both aspects, simulation and construction.

Keywords: Maritime Simulation, Automatic Driving, Robotics, ROS, Gazebo, Ardupilot.

La première étape d'une simulation dans Gazebo est de comprendre ROS, pour faire une simulation maritime correcte il est nécessaire d'inclure toutes les forces attendues et de les appliquer à un bateau, ceci a été accompli en utilisant le plugin VRX. Pour permettre le pilotage automatique, un NAVIO2 utilisant le logiciel Ardupilot Rover a été utilisé, et un plugin reliant la simulation au pilote automatique a été mis en place. Finalement, le bateau a été construit avec des éléments que le laboratoire possédait déjà et nous avons obtenu une conduite automatique correcte. Il y a eu beaucoup de problèmes pendant le développement, mais ils ont été rapidement identifiés et des solutions ont été proposées, mais non mises en œuvre, en raison d'un changement de plans provoqué par la pandémie. À l'origine, le projet portait sur la construction du bateau, mais la fermeture du laboratoire a fait que l'objectif a été déplacé pour se concentrer uniquement sur la simulation, mais la réouverture a fait que le projet comprend maintenant les deux aspects, la simulation et la construction.

Mots-clés: Simulation Maritime, Conduction Automatique, Robotique, ROS, Gazebo, Ardupilot.

Contents

Confidentiality Notice **3**

Abstract **5**

Contents **7**

Introduction **9**

1 Objectives and Planing **11**
1.1 Objectives . . . 11
1.1.1 Primary . . . 11
1.1.2 Secondary . . . 11
1.2 Plan and Agenda . . . 11

2 Online ROS Class **13**

3 Simulation **15**
3.1 Importance . . . 15
3.2 Model . . . 15
3.2.1 Hulls . . . 15
3.2.2 Platform . . . 18
3.2.3 Motor Rudder Assembly . . . 19
3.2.4 URDF . . . 20
3.3 VRX . . . 22
3.4 Navio2 . . . 25
3.5 Ardupilot . . . 26
3.6 Ardupilot-Gazebo Plugin . . . 26
3.7 Final Result . . . 27

4 Analysis and Comparison with the existent Boat (Annorax) **29**
4.1 Annorax . . . 29
4.2 Differences . . . 30
4.3 Possible Technology Translation to Aquanaute . . . 30

5 Construction **31**
5.1 Components . . . 31
5.1.1 Remarks . . . 32
5.2 Connection Diagram . . . 35
5.3 Assembly . . . 37
5.3.1 Mechanical . . . 37

5.3.2 Electric . . . 38
5.3.3 Transportation . . . 39
5.3.4 Utilisation . . . 40
5.4 Final Result . . . 41
5.4.1 Test 1 . . . 42
5.4.2 Test 2 . . . 43
5.4.3 Changing Parameters and WARNING! . . . 43

Conclusion **45**

Bibliography **47**

Glossary **49**

List of Tables **51**

List of Figures **53**

Annexes **55**

Introduction

This project origins from ENSTA having an already working boat "Annorax", a boat capable of autonomous driving. The laboratory managed to get used parts from an international boat competition and proposed the project to translate the technology from the small boat into a bigger format.

This report documents the 10 week development period of Aquanaute. It started with the learning of ROS, Gazebo and Git, all of this knowledge was implemented in the second part of development, simulation. The simulation was done inside gazebo, and to achieve a proper simulated environment some plugins were used to make a maritime analysis and allow for a simulated automatic pilot. For this is it was necessary to research on the autopilot *Ardupilot* and the card that make it possible, the NAVIO2. 3D modeling was also used to translate the already existing components to the software. When the simulation was in a good enough shape and with only two weeks left, construction began, testing all components and running a dry test (no pun intended) were fundamental to close the learning loop. After that,came the actual building of the boat and the required tests to finally consider Aquanaute complete and the project finished.

Part 1

Objectives and Planing

1.1 Objectives

1.1.1 Primary

Simulation of the boat inside of ROS Gazebo with full maritime conditions like buoyancy, waves and wind. With the ability of being externally controlled by a simulated automatic pilot.

1.1.2 Secondary

Construction of the simulated boat using already existing components with an expected behaviour obtained from simulation.

1.2 Plan and Agenda

To ease the development of the project and to keep track of the advancements, online tools like Trello and Confluence were used. Trello was used to designate the goals an their status, between "TO DO". "DOING", "FINISHED" and others. Confluence provided a blog to write down ideas and notes from reunions.

Trello was especially useful because it there the road map, task and goals were clear and easily accessible, thanks to it i was able to continue with the development of the project without to much stopping for reviews and checks.

I had weekly meetings with my tutor, to check the progress and to define the tasks for the week, unexpected meetings were done to explain or to clarify something learned during the week.

The figure 1.1 shows roughly how the time was spent during the 10 week internship. Because of the pandemic, the objective of the internship was changed to only be focused on the simulation aspect. In July the laboratory was able to open and the objective grew to include the construction. To made the simulation I had to learn ROS to understand how it worked and to be able to see the bigger picture of creating a project that someone in the future can use and implement ROS more extensively.

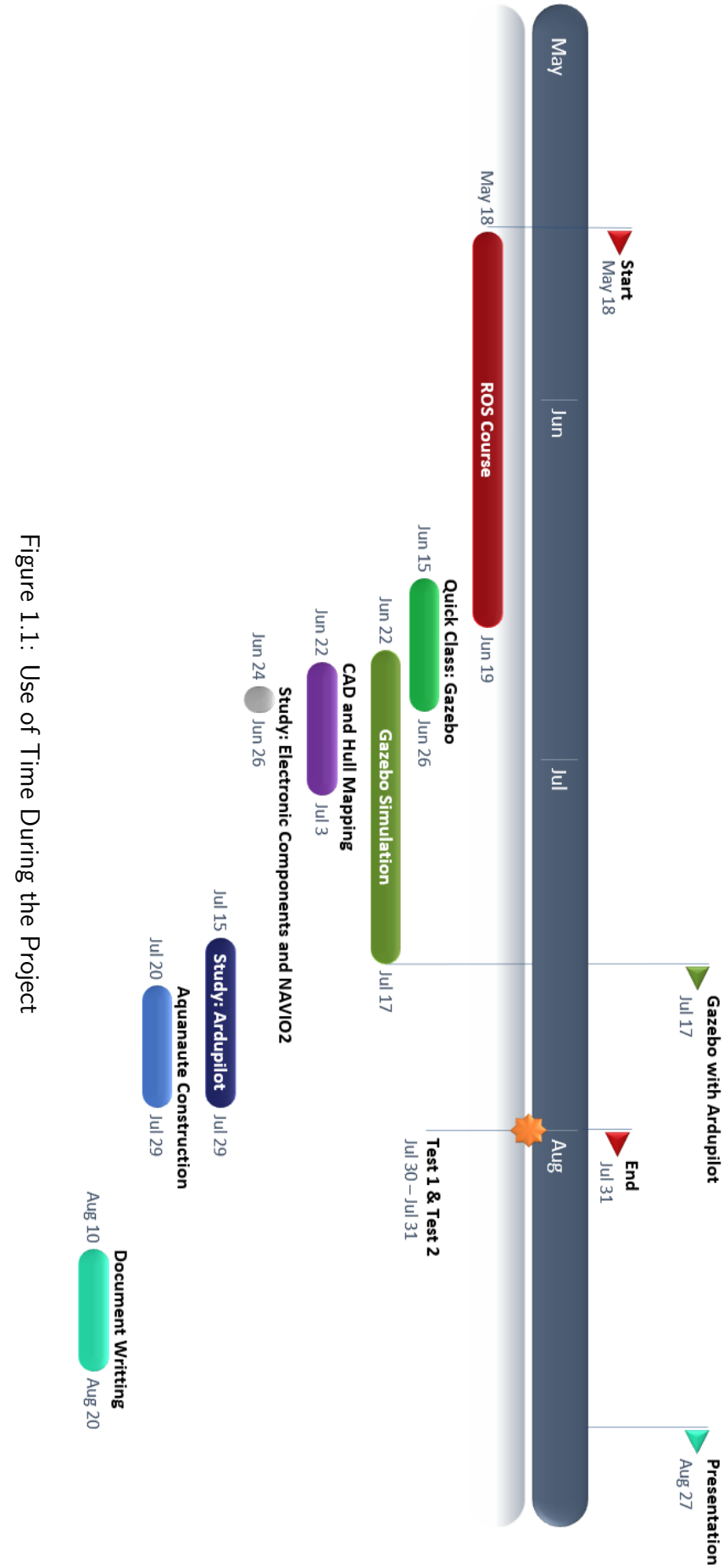

Figure 1.1: Use of Time During the Project

Part 2

Online ROS Class

To be able to simulate the boat learning ROS was the first step because Gazebo is the simulator chosen for this project and it works with ROS. ROS or Robot Operating System is an OS designed to work with and inside robots, it is community driven and its the leader in the industry [1].

I took courses provided by Robot Ignite Academy, and followed the path "ROS for beginners [2]" with a slight modification. In the end these were the courses I followed:

- Linux for Robotics
- ROS Basics C++ & Python
- URDF for Robot Modeling
- TF ROS 101
- Your First Robot
- DEBUG Cases
- Final Exam
- Git Tutorial

After a month learning ROS I am able to create, simulate and build a basic robot from scratch. ROS being based on Linux requires a somewhat advanced knowledge of Linux Commands to for example allow for permissions and run programs from the command line. A difficulty I found was that almost all the courses were taught in Python, a language that I don't know a lot and that I considered to be focused on machine learning or image recognition. ROS can be also programmed in C++ and this was an advantage for me because I had already work with it a lot during my 2nd year at ENSTA, C++ allows for a more real time execution, something critical in automatic driving.

The Git tutorial was fundamental, not solely for the project but code development in general, Git is a great way of saving all the development in a second location, and allows to share the code and documents. The laboratory has its own git and there resides all the information regarding this project.

Another crucial topic in the development of the project was the use of URDF or Universal Robot Descriptor File, the correct creation of this file is fundamental for the simulation, because this is the file that tells the system how the robot is constructed, its inertia and other parameters necessary for a correct simulation.

All other courses and the final exam were vital in cementing the basis of ROS, like nodes, topics, services and actions.

Because of this project being focus in simulation, the most important subject learned was the connection between ROS, Gazebo (ROS Simulator) and Ardupilot. Gazebo is an extensive simulation software capable of testing sensors, movements and speeds of a robot to later translate the parameters into real life.

Opinion

The classes were a great learning teacher, they managed to get the point across even when ROS is famous for its complexity. I have some remarks that can be improve upon, like the lack of examples for C++ programming (almost all the system is designed for Python). Also some courses were implemented incorrectly, without a proper evaluation system or a correction to learn from one's owns mistakes, there are forums available where I tried to find answers and where i posted my problems, but not much information was received. In the end I am happy with my results and the achievements I earned, and I'm grateful with the Laboratory for giving me the chance to learn ROS. It was great solving the problem proposed for the final exam, where you need to code an algorithm to extract a Husky Robot from a room in under a minute (figure 2.1) or the exam where you need to automatically fly an AR Drone while recording its position (figure 2.2).

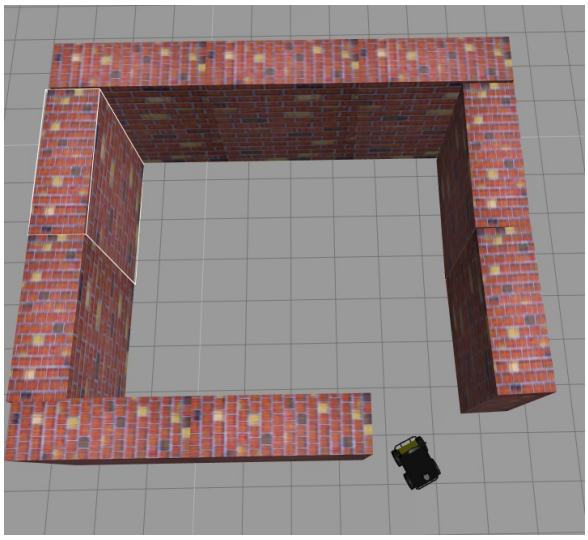

Figure 2.1: Final Exam with Robot Husky

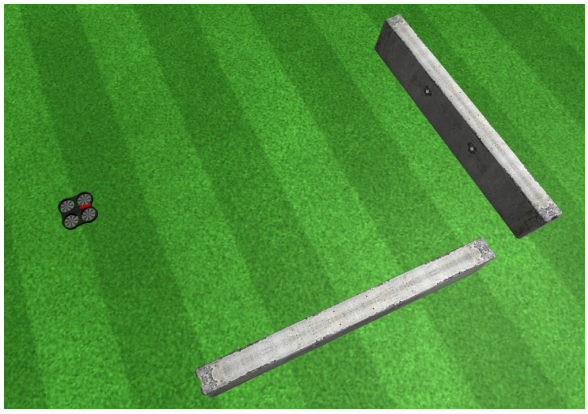

Figure 2.2: Final Exam with AR Drone

Part 3

Simulation

3.1 Importance

In a real world application, simulation is fundamental to know if a project will be a success or not from the early stages, simulation allow us to test different sensors, configurations and materials without spending money. When you are happy with a simulation you can pass to the construction phase. When done correctly the simulation can predict very accurately the result that will be obtained in the real experimentation.

For this project simulation was a key element because of the nature of the real objects used, they were unidentified and very difficult to change in case of a problem. that's why the simulation helped us understand the behaviour of the elements and the relationships between them.

difficulties

As you will learn later, the simulation is only important if done correctly, simulation will give you the results depending on the inputs you give to it. For Aquanaute there were a lot of problems on simulation that caused more work for the future.

3.2 Model

To be able to simulate the boat in almost real conditions, it was necessary a model that was a perfect representation of the elements that were available. The components were, the two hulls of the catamaran, the platform connecting them, the direction motor, and the motor rudder assembly.

3.2.1 Hulls

The hulls were by far the most difficult part to model, because of their craft nature, there was no straight forward way to obtain the curves that allowed to create the CAD. The preferred method for obtaining the model was to use a camera to make a 3D digital scan. Photogrametry, Depth Scan and Laser Cloud were tested, but, because of the reflectance of the hull as seen in the figure 3.1 and lack of clear points of reference, the models obtained were filled with errors and to difficult to work with, as an example the figure 3.3 shows the mesh obtained with an stereo camera. At the end, the solution chosen was to model the hulls from zero, like the crafts they are. With the use of the motion tracking system ENSTA

has, the coordinate points that are part of the curves of the hull were extracted by moving an infrared marker and capturing its position as shown in the figure 3.2. After obtaining the relative distances between the points on the curves (the tables can be found here [3]) they were introduced inside the CAD software (Fusion360 in this case) to form the characteristic splines that were later combined with a "loft" function. Some eye balling was done to the curves to obtain a form more similar to the real object, the final model can be seen in the figure 3.4. It is necessary to point out that this modeling wasn't done in a perfect way it was more craft than a 3D representation, it would be more accurate to obtain a mesh from a digital scan.

Figure 3.1: Test of Photogrametry

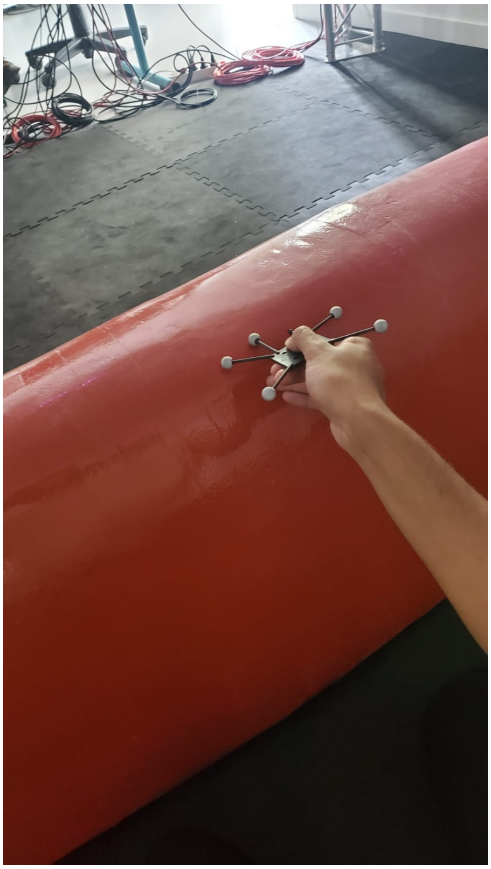

Figure 3.2: Gathering of the relative position of points along the hull's curve.

Figure 3.3: Mesh obtained with a Stereo Camera (location of the hull in red)

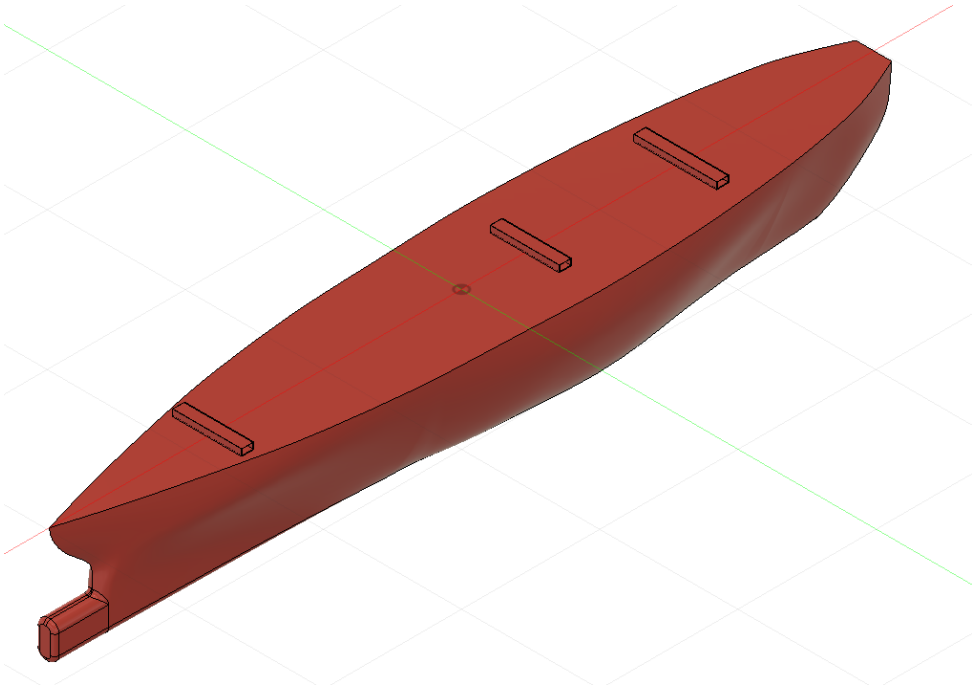

Figure 3.4: Final model for the Hull

The end result is sufficiently good for what we were looking at the beginning that was a correct representation of the real object inside the simulation. But as you will read in the VRX section (3.3) this was not necessary because of the way the VRX simulation treats the hull in its calculations.

3.2.2 Platform

The platform is a simple square with a side length of 1.80 m. It has a small structure added to support the motor rudder assembly, and has the necessary holes to pass the bolts that join the platform with the hulls.

An unexpected design choice made by the original team that worked with the material available made for an uncommon orientation of the bracket placed to hold the motor as shown in the figure 3.5,this raises the motor above the flotation line making it necessary to add more weight to the boat (the boat's flotation line is approximately 10cm above to bottom of the hulls, it is necessary to between 80kg and 100kg to lower the motor deep enough to achieve efficiency, all of this was discovered during test 1 and is explained in the section 5.4.1).

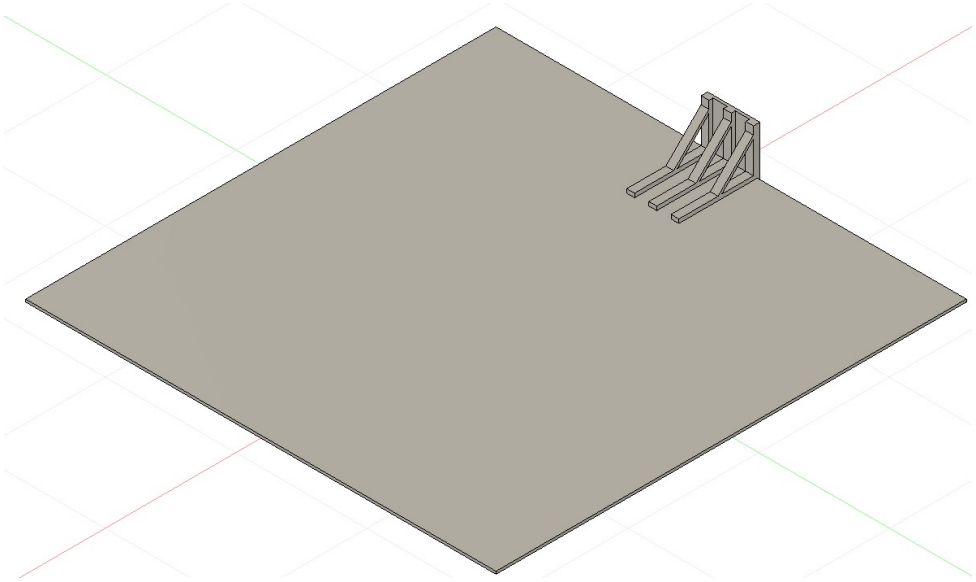

Figure 3.5: Platform model with its correct orientation

A possible improvement would be the rotation of the structure that holds the motor rudder assembly to lower the motor into the water.

3.2.3 Motor Rudder Assembly

This part is not an exact copy of the real component because of the difficulty of the curves that is the rudder and the lack of documentation regarding its measurements. The final model is close enough in shape and weight distribution to be able work in a simulated space and can be seen in the figure 3.6.

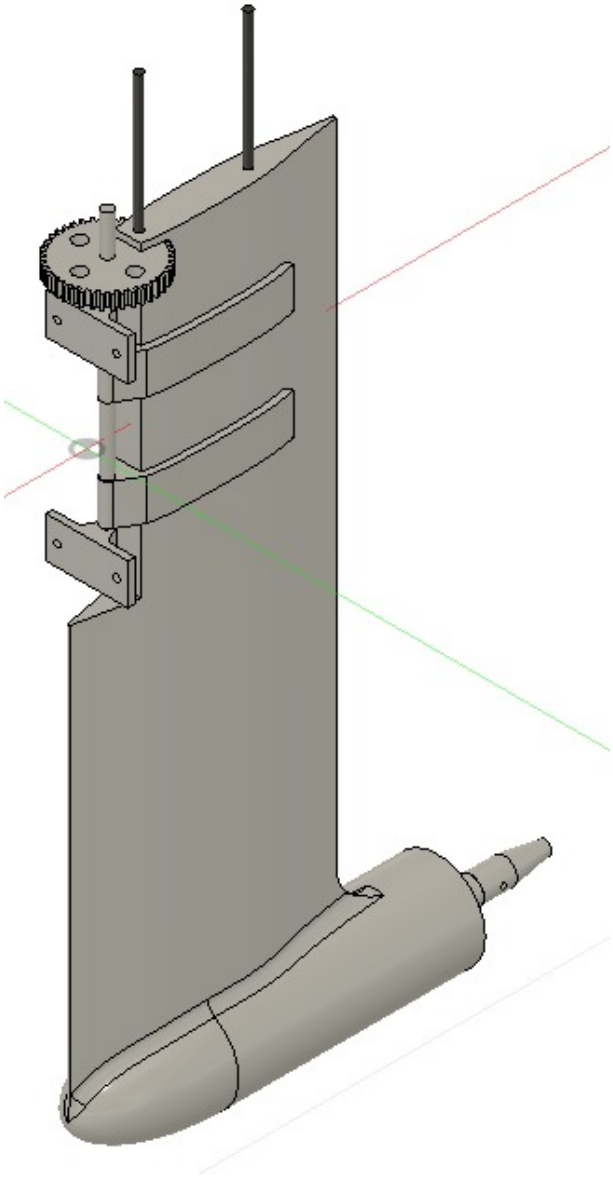

Figure 3.6: Motor Rudder Assembly Model

A possible improvement would be to contact the manufacturer an request their 3D model for the motor and the rudder. This carries some difficulties that will be explained in further detail in the section 5.1.1.

Final CAD Model

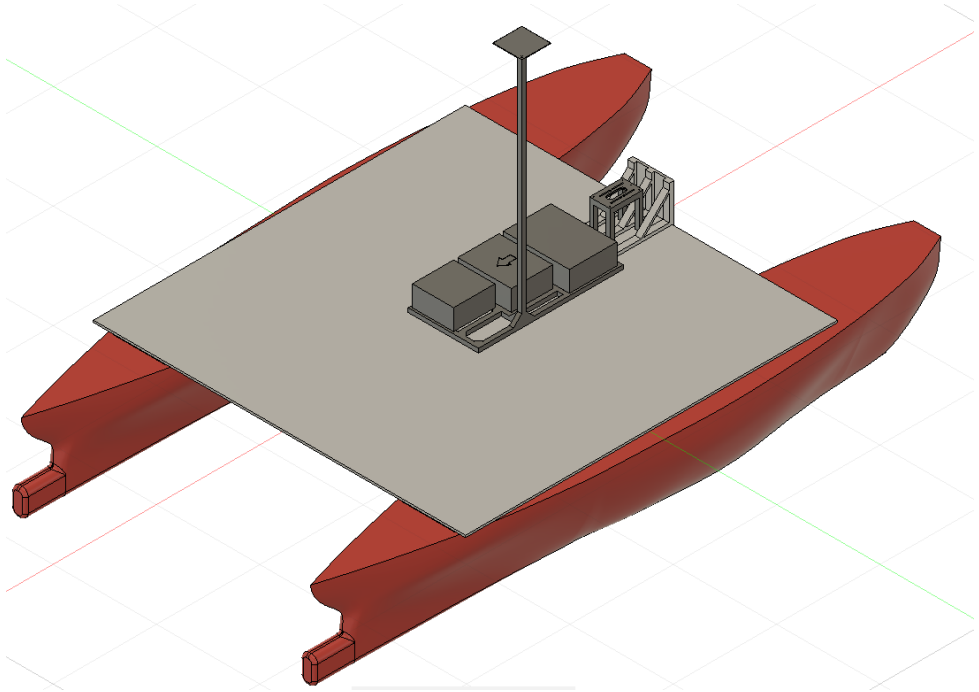

Figure 3.7: 3D model of Aquanaute Assembled

3.2.4 URDF

This file is extremely simple and extremely important. As mentioned above on part 2, the URDF describes the robot to the software so that it understands it like we want to. The specific file of this project was created using as a base the URDF XACRO file provided by the Robot Ignite Academy in the course URDF [4].

XACRO

XACRO is a macro tool that can be used with URDF files. URDF files are coded in XML format and don't allow the use of variables or calculations, XACRO eases this by opening the options to use more common C programming syntax, like global variables, condition blocks and more.

The project's file has the visual and collision definitions as represented by the CAD files, and the inertial definition obtained from the CAD software. The build model as described by this file [5] can be seen in the figure 3.8.

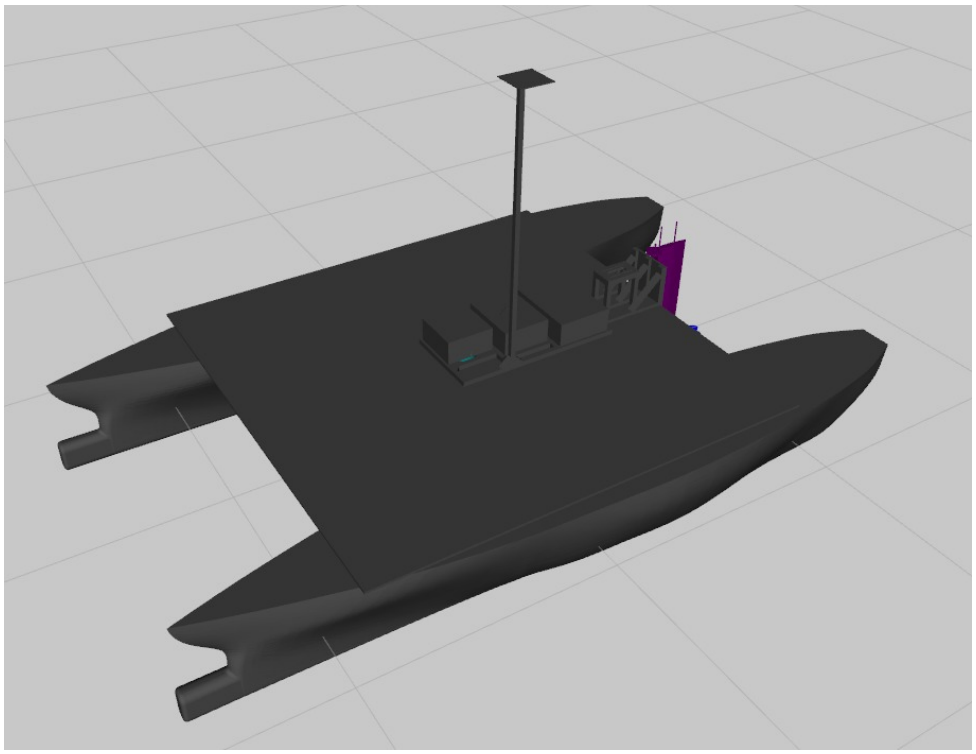

Figure 3.8: Model as Described by the URDF (In RVIZ)

Internally the collisions are not simple bounding boxes because the laboratory has powerful enough computers to run a proper wave force simulation.

3.3 VRX

VRX or Virtual RobotX is the virtual or simulated companion of the RobotX Challenge [6], a competition aimed at university students to create autonomous algorithms for constructed boats. Students can modify the amount of motors and their distribution, add sensors and weight, but the chassis and the motor reference has to remain the same. The boat or WAM-V can be seen in the figure 3.9.

Figure 3.9: WAM-V Boat of the RobotX Challenge [7]

Our interest in using VRX as a simulation platform was that nowadays there aren't open source, generalist maritime simulators. All of the available ones are proprietary or specific to a domain, We were searching a simulation environment capable of simulating the forces exerted on the vehicle caused by waves, wind, motors, and buoyancy. VRX trying to solve this issue created plugins to work inside gazebo to simulate all previous mentioned forces. They managed to get the result shown in the figure 3.10

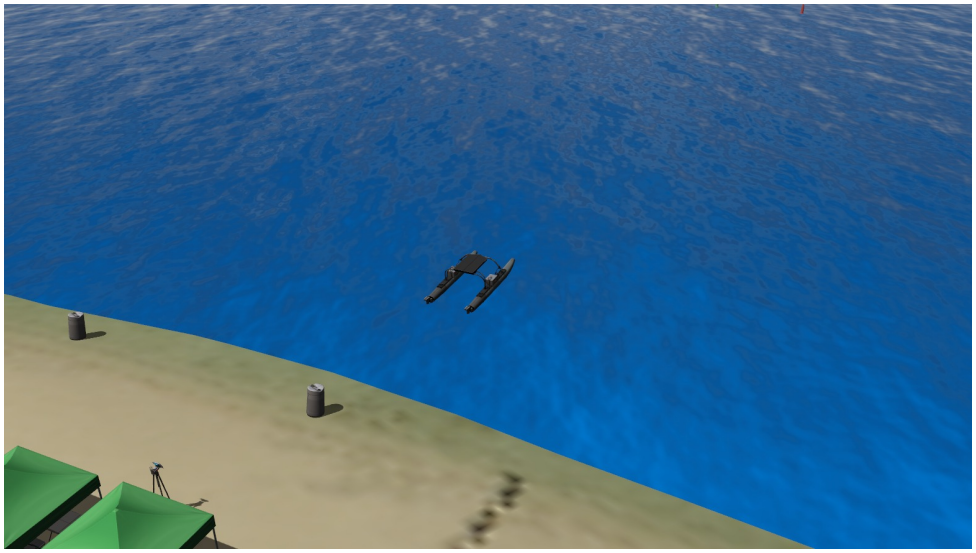

Figure 3.10: VRX Simulation Running in Gazebo

All the information on how they managed to simulate waves can be found in the article *Toward Maritime Robotic Simulation in Gazebo* [8], there you can find all information regarding how to create a wave field, randomize it and make it appear as natural as possible, also it talks about wind generation and the effects this to forces have on a simulated boat. The Article can be used as a guide to create our own gazebo plugin. We were looking for something ready to use and that is why we used the VRX one and found its limitations.

- The VRX simulation locks all other forces from acting on the simulated body, the boat will only experience the forces generated by the waves, the wind and the motors (all three plugins must be the ones provided by VRX), on Aquanaute only the plugins of waves and wind were used, the effect of the forces according to [8] were applied as shown on the figure 3.11.

- The hulls are simulated as 2 cylindrical tubes to simplify the wave force vector generated. The parameters to change is their length, radius and distance between them (this is why it wasn't necessary the real collision of the hulls).

- The boat weight and its distribution is already set in the calculations and cannot be changed.

- The effect of the waves an the wind is transmitted to the boat via coefficients that must be obtained experimentally before hand (This procedure is not explained).

- The gazebo world cannot be changed. When running a ROS environment you have to compile by running **catkinmake**, the problem is that the worlds provided are hard coded into the compilation and are the only ones taken into consideration to run the plugins, you cannot add your own world to the simulation.

- There isn't the possibility of using a custom motor distribution, the rules of the competition only allow for 3 modes, none of which are the single vectored thrust we have available for our boat.

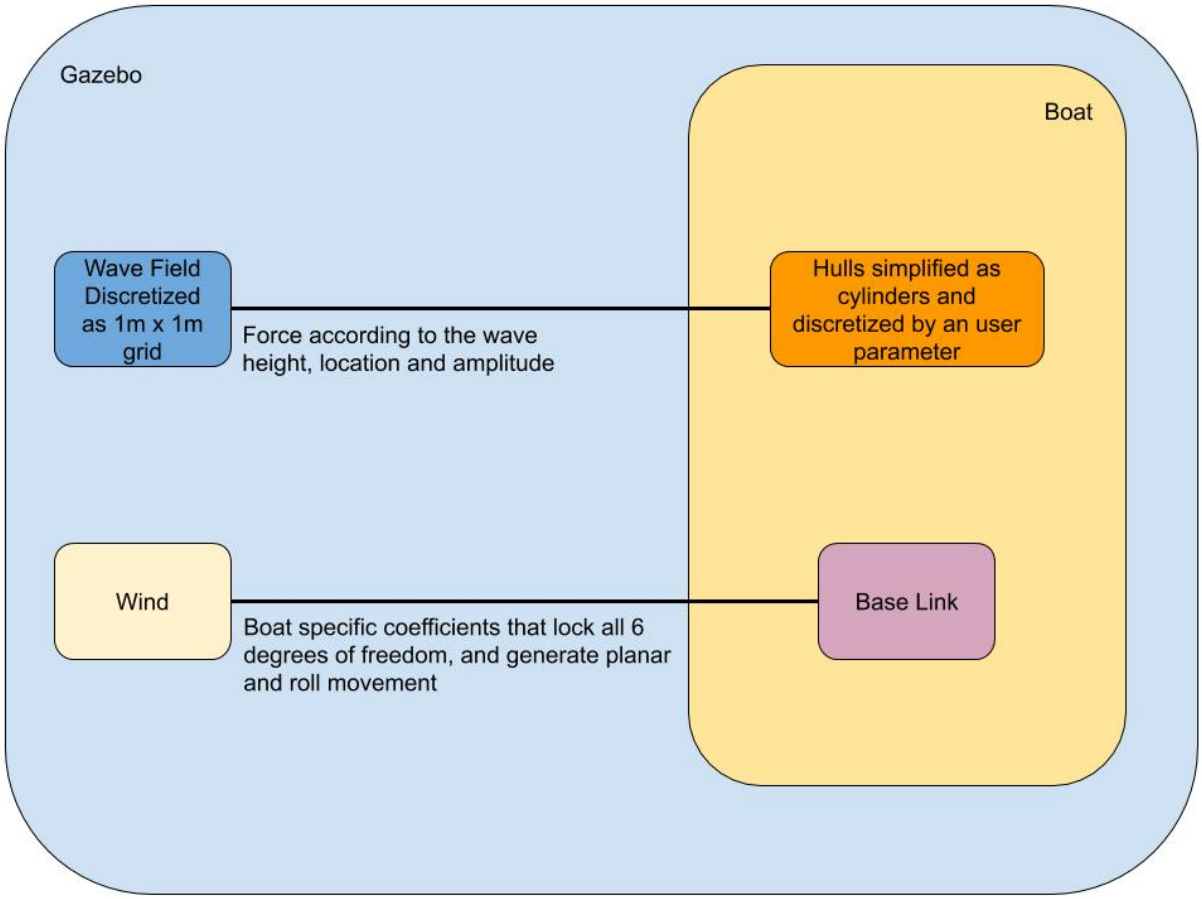

Figure 3.11: VRX forces applied to Aquanaute

VRX works great as a WAM-V simulator in maritime conditions, but because of its close relationship with the RobotX challenge, it was designed exclusively their own boat with their own restrictions. I recommend to use the article [8] and the VRX plugin as the basis to create our own plugin for future simulations.

3.4 Navio2

Navio2 is a hat (additional board that add an specific function) for the Raspberry microcomputer it attaches as shown in the figure 3.12, it is designed to add autopilot capabilities to a robot or system that uses the raspberry as it's brain. Navio2 has two redundant compasses, a Glonass GPS chip, barometer (for altitude) and the slots to connect external systems like telemetry. It communicates with its actuators via RC signal, and receives commands via ppm. Navio2 is an almost in real time platform to create RC controlled vehicles, it can use the computational power of the raspberry underneath to augment the capabilities of the final product. More information can be found on NAVIO2 Documentation Web-Page [9].

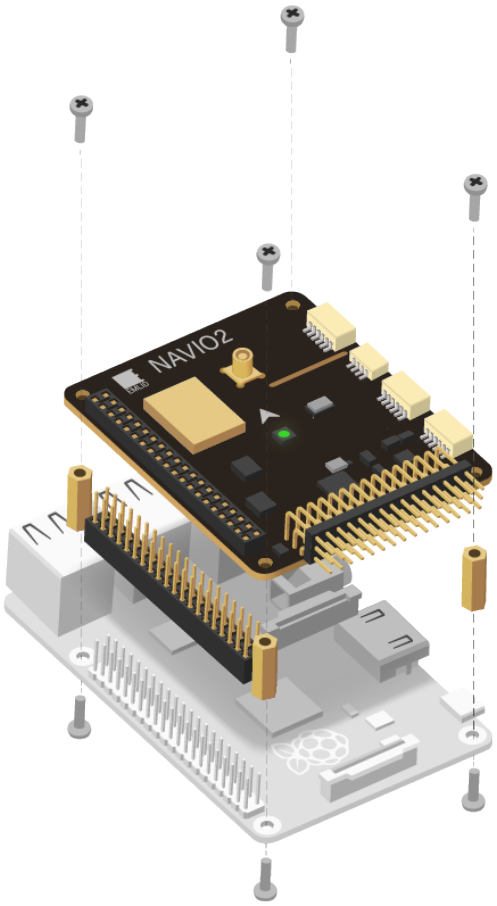

Figure 3.12: How to Mount Navio to Raspberry [9]

3.5 Ardupilot

Ardupilot is a software that gives to a compatible chip like the NAVIO2 the ability of autopilot. Ardupilot is a Linux based software designed to read the sensors, write to the motors, and accomplish automated tasks. It primary focus are drones, but there are development teams for planes, rovers, and submarines. Boats are a subcategory of rovers because of their similarities in throttle and steering. Ardupilot provides a simulation software called SITL (simulation in the loop), figure 3.13

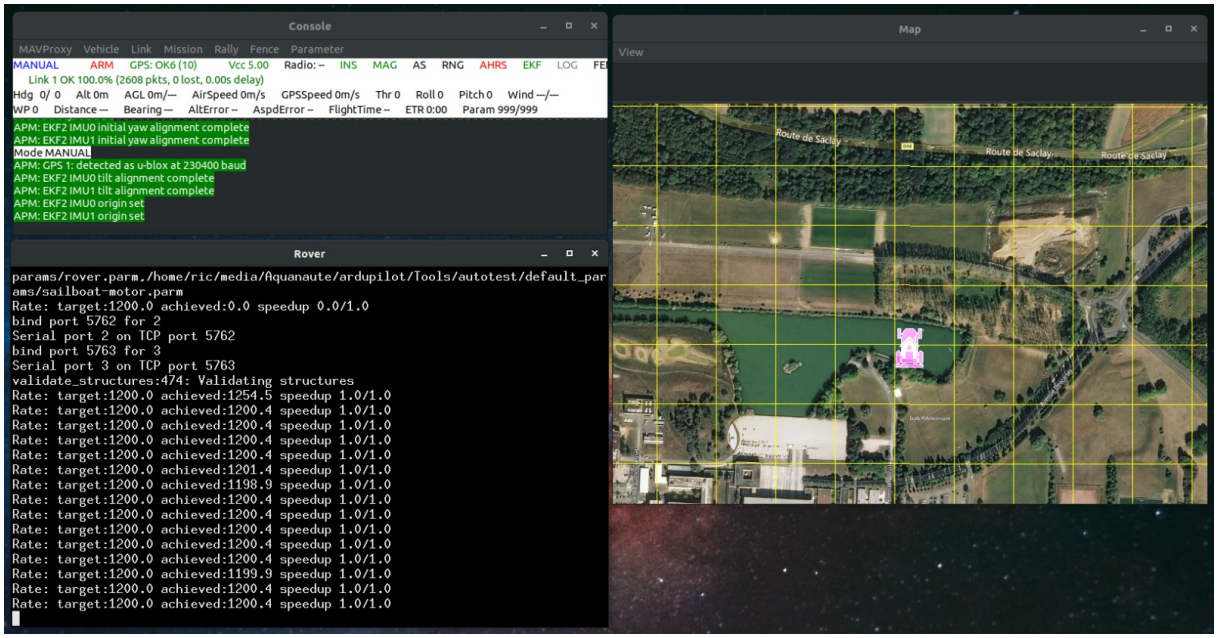

Figure 3.13: Normal Configuration of SITL

SITL allows you to run the desired parameters of your vehicle, experiment and tweak them to improve the real vehicle. As stated before, simulation is an essential step before moving forward to a real object construction.

3.6 Ardupilot-Gazebo Plugin

The Plugin was originally created by khancyr (https://github.com/khancyr/ardupilot_gazebo), an Ardupilot Copter developer, it allows Ardupilot SITL to control the motion of a Gazebo element and make think Ardupilot that it is controlling a vehicle in the real world. The goal of this plugin is to transmit motion to the Gazebo simulation and allow another plugin to read this motion and translate it to a force applicated to the object.

Ardupilot only recommends the original one or the forked version by SwiftGust (https://github.com/SwiftGust/ardupilot_gazebo), but neither of them has been updated in the last year to work properly with a Rover frame, the khancyr one has only an example for a quadricopter, and the forked version has examples but cannot be run at all because it was created for an older version of Gazebo. When writing this report I found out that there are multiple forks done to the original (https://github.com/khancyr/ardupilot_gazebo/network/members) and from the SwiftGust one, maybe one of them works for us.

3.7 Final Result

The final simulation can be run by cloning the repository [10] named Aquanaute available in the laboratory's gitlab and following its instructions. The figure 3.14 explains how the different softwares interact with each other.

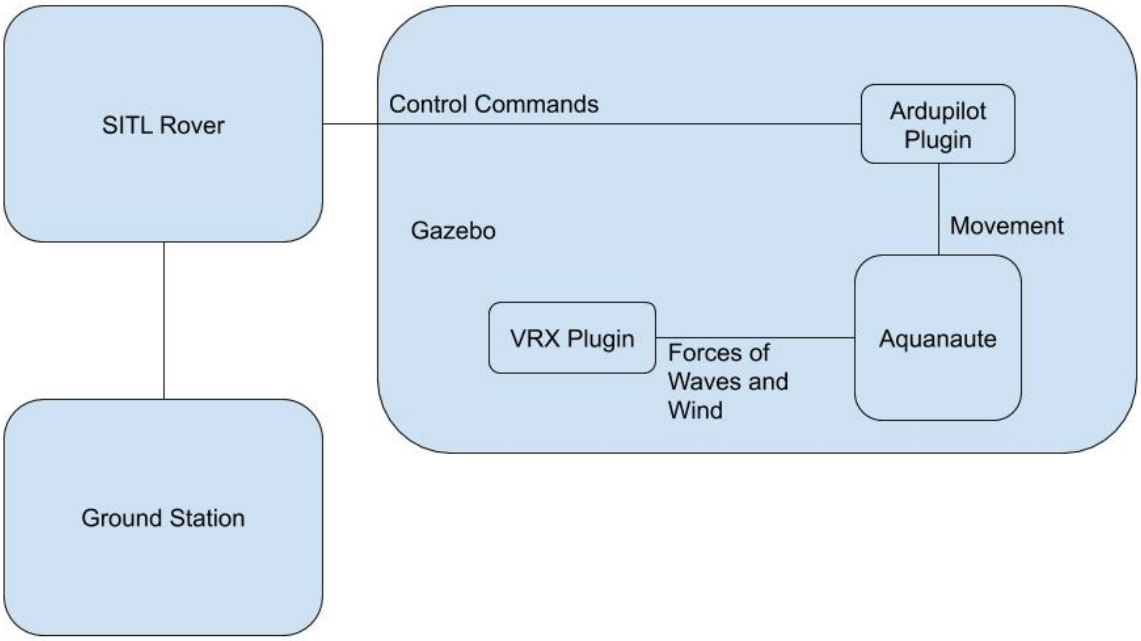

Figure 3.14: Simulation Diagram

The simulation can be seen on the figure 3.15 and it is comprised of the VRX plugins for wind and waves, and the Ardupilot-Gazebo Plugin. To run the simulation with SITL you need to have installed Ardupilot on your machine and to run inside the Ardupilot folder the simulation vehicle (keep in mind that only frames with "gazebo" on their name will be recognized by the plugin) Besides the already mentioned problems with the VRX simulation, I had some issues with the use of the plugin that I was unable to solve. The URLs to the forum questions are:

- https://discuss.ardupilot.org/t/rover-boat-in-sitl/59157
- https://discuss.ardupilot.org/t/quadplane-sitl-with-gazebo/34145/21
- https://github.com/khancyr/ardupilot_gazebo/issues/36
- https://github.com/khancyr/ardupilot_gazebo/issues/17

The general issue addressed is that the motion seen in Gazebo doesn't correspond to and expected behaviour from the autopilot. The motion is erratic and uncontrollable. Because of the nature of the plugin other people simulating a vehicle other than a quadricopter have had this problem but have been unable to find a solution.

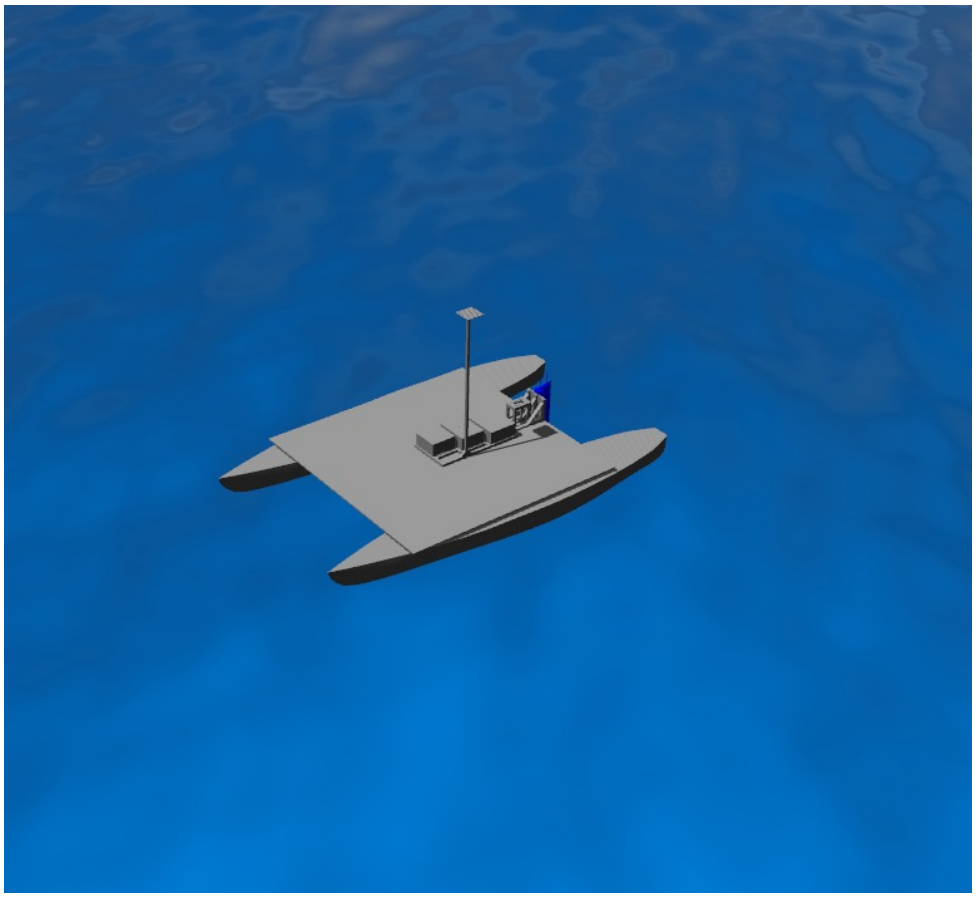

Figure 3.15: Maritime Simulation of Aquanaute

The figure 3.15 shows Aquanaute in wave world being affected by wind using the WAM-V coefficients, and by waves that treat the hulls as cylinders. Its relative motion can be tracked because its change in position is being registered by an IMU, transmitted to the plugin and from there passed to the mission planner. You can see the motors moving respecting their limits, but without control.

Proposed Changes

With all the difficulties brought by the VRX simulation I think it would be in the best interest of the future development of the project to create our own simulation environment. There are the resource like the article [8] or using the hydrodynamics and wind plugin native to Gazebo.

In regards of the SITL and the Ardupilot-Gazebo plugin, maybe the best is to wait for the Ardupilot team to find a solution to the problem. In the case that a more direct solution is required the laboratory could create its own plugin to connect SITL to gazebo.

A more drastic change that could help with the SITL connection is to use a native Gazebo frame proposed by Ardupilot. As it stands, Aquanaute is a rover with one motor for thrust and another one for direction, but currently, the development for the rover in SITL that works with gazebo is vectored thrust controlled. It's worth a shot if this implementation solves the issue with the plugin.

Part 4

Analysis and Comparison with the existent Boat (Annorax)

4.1 Annorax

The existing boat is already capable of autopilot with the use of a NAVIO2 with a Raspberry4, it has a suite of instruments and communications like WiFi, telemetry, radio, camera a sonar and it can carry a ROV. In the figure 4.1 you can see its connection diagram.

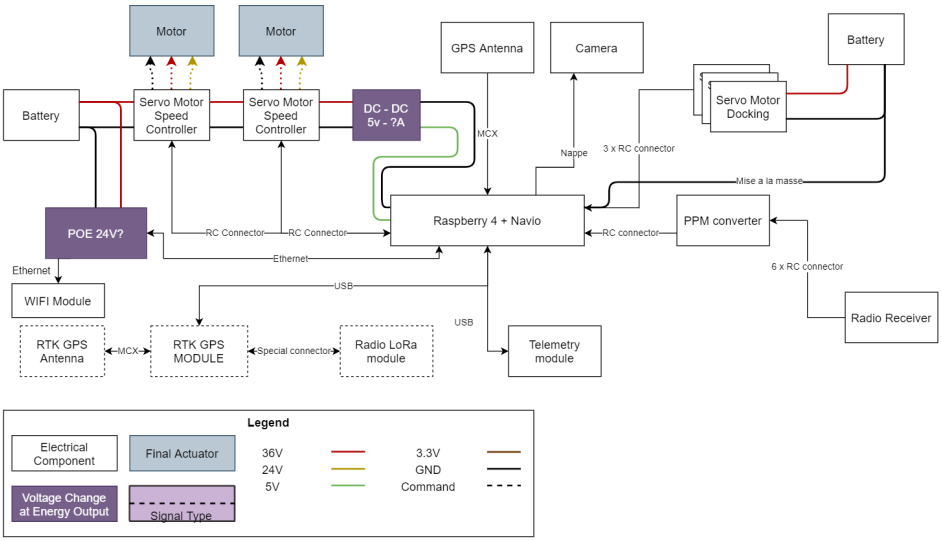

Figure 4.1: Annorax's Connection Diagram (HD image [11])

With the diagram 4.1 we will be able to find what elements are similar to what is already available for Aquanaute and what changes need to be made. The goal of the project is to implement Annorax technology in a bigger and more capable platform.

4.2 Differences

Originally Aquanaute was being used with a ST32 board, the lack of information and the impossibility to implement ROS with it, made it mandatory to make the change to use in Aquanaute as used in Annorax a Navio2. Another important difference between the boats is that Annorax is controlled by changing the thrust between its motors, the elements available for Aquanaute allows us to use on motor for thrust and another for direction.

4.3 Possible Technology Translation to Aquanaute

Aquanaute will receive a NAVIO2 as brain for its control and this will allow the use of all the peripherals used within Annorax. The more cumbersome change generated by the use of the NAVIO2 as stated before is that because of the nature of the RC output to control the motors, the motors themselves should be controllable via RC. This is not the case for the direction motor, this motor is a direct drive motor with an encoder. To control it with an RC signal we have to add an adapter that reads the position of the motor and does a position control by sending speed commands to the motor and reads RC signals from the NAVIO. More information is available in section 5.1.1.

Part 5

Construction

5.1 Components

In the next table you will find the information regarding the components already used in the original boat that were re-used in Aquanaute.

Boat Component	**Product Name Reference**	**Operational Voltage**	**Operational Current**
Water Pump		9V - 24V	0.55A - 1.12A
throttle motor Controller	Watt&Sea POD 1400W Speed Controller	40V (60V max)	40A max
throttle motor	Watt&Sea POD 1400W	powered by controller	
Direction Motor	Faulhaber MD81/1053	24V	80mA max
Battery	LiPo S10 36V	42V - 36V	10.4Ah (with 60A fuse)
DC - DC Converter	Voltage Step-Down	[8V-60V] in [1V-36V] out	10A (15A max)
Direction Motor Controller	Sabertooth 2x25 V2.0	6V - 30V	25A (per side) 50A max
DC - DC Converter	Turnigy Multistar SBEC	[6V-50V]in [5V or 12V]out	5A(5V) or 10A(12V)
Direction Motor Translator	DFRobot Multiplexer V2.0	5V	
Radio Receiver	HPD-07RH QPCM	3.3V or 5V	

Table 5.1: Available Components used in Aquanaute [12]

5.1.1 Remarks

In the Connection Diagram 5.2 you will find how all components are connected between them and the necessary signals for their correct use.

Watt & Sea Motor and Controller

The motor, rudder and speed controller for the motor where obtained from the HydroContest, a competition aimed at university students, to build boats in three categories, weight transport, distance in a single charge and top speed [13].

There isn't a lot of information regarding the motor or its controller, but thanks to the help of the Colombian team **Hydrómetra** [14], we have the characteristics of the motor (figure 5.1), and some information regarding the motor's origin.

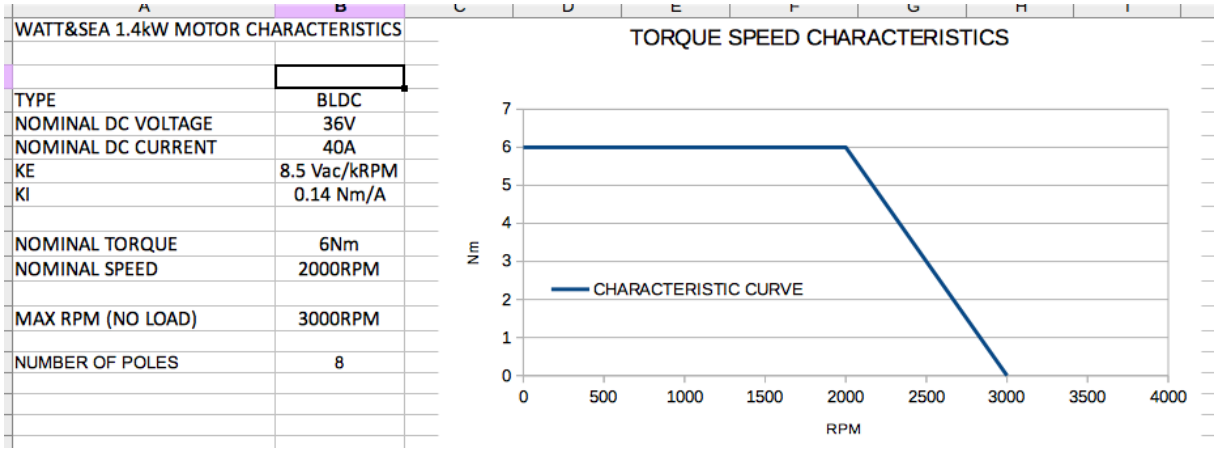

Figure 5.1: Watt & Sea Motor Characteristics [14]

The difficulty to obtain additional information on the motor comes from the fact, that the motor was ordered by the HydroContest Organizers to be manufactured by Watt & Sea, a company that makes hydro-generators, not motors [15]. Watt & Sea themselves outsourced the creation of the motor speed controller to an unknown Company. The motor was made for the HydroContest 2018 Competition.

The controller needs to be water cooled to work properly and to avoid any damage to the components, the switch shown in the figure 5.13 gives energy to the pump, which has to run with water and without air bubbles to avoid damage.

Direction Motor

The available motor to control the direction is a Faulhaber Industrial Actuator, a continuous drive motor, with an absolute encoder [16]. This motor needs a controller, the one available is a Sabertooth Motor Driver [17], it can control the motor speed and direction by receiving a simplified serial signal (Only TX from TTL) if its configured as shown in the figure 5.2

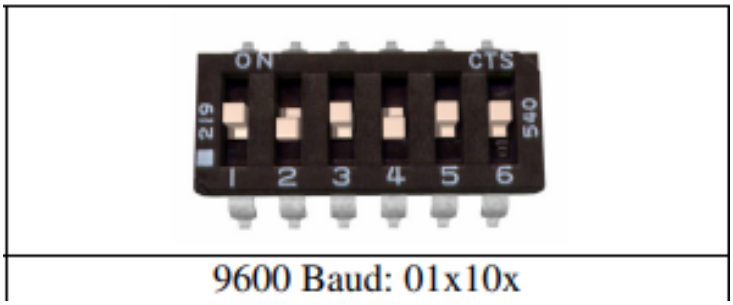

Figure 5.2: Sabertooth Configuration

To control a continuous motor in position we need to read its position, the encoder of the motor gives the motor position using the communication protocol RS-485 [18], with the available components we cannot read this protocol, that is why, we use a multiplexer [19] to transform the RS-485 protocol into TTL.

Now we can control the motor direction and speed with a TTL signal and we can read its position in TTL format. The problem is that as far as we know, the NAVIO2 blocks all ports on the Raspberry and only gives output in RC format. This is why we used an Arduino as an adapter, the Arduino reads the RC output from the NAVIO, interprets it as a position, an does a proportional position control by closing the loop reading from the encoder.

An improvement would be to characterize the motor, and do a proper controller, adding an integration to reduce the error. But a Huge improvement, that would simplify the project would be to change the motor, for a RC controlled Servo with similar torque capabilities.

RC Controller and Receiver

The receiver is a discontinued product [20] that is exclusively paired with the provided controller. neither of them can work with a different product, this add the problem where if at any given moment one of them needs to be changed because of damage or an upgrade, the replacement would be difficult to find. For the moment the pair work great, but there isn't information on how to configure the controller to use it at its fullest.

Battery

There is no information online about the battery, all we have is the sticker attached to it and can be seen in the figure 5.3.

Figure 5.3: Battery Information

Figure 5.4: Fuse Available

The battery is Lithium Ion with 10 Cells (an uncommon configuration for an RC type build). It also has a 60A fuse, seen in figure 5.4, the fuse is only common in trucks in the USA (https://www.ebay.com/itm/J-CASE-LOW-PROFILE-FUSE-60A-AMP-32V-FUSE-CARTRIDGE-CAR-AUTOMOBILE-FUSE-Yellow-/113757476607).

According to the information gathered, the battery and fuse were provided by the Hydro-Contest Organizers.

I recommend the addition of a battery monitor to know the voltage level from the mission planner and avoid any damage.

5.2 Connection Diagram

The diagram 5.5 shows the current configuration of Aquanaute and the general connections inside the electronic boxes and with the outside world, it was impossible to fit all the components inside one box that is why they were separated in three boxes as shown in the figure 5.6. The figure 5.7 shows a close up of the middle box, the most important one, because it carries the NAVIO. The NAVIO was itself connected similarly to what EMLID recommends for a rover (figure 5.8). The pins used for the connections can be found on the figure 5.5. The information architecture and connections with the outside world can be seen on the diagram 5.9).

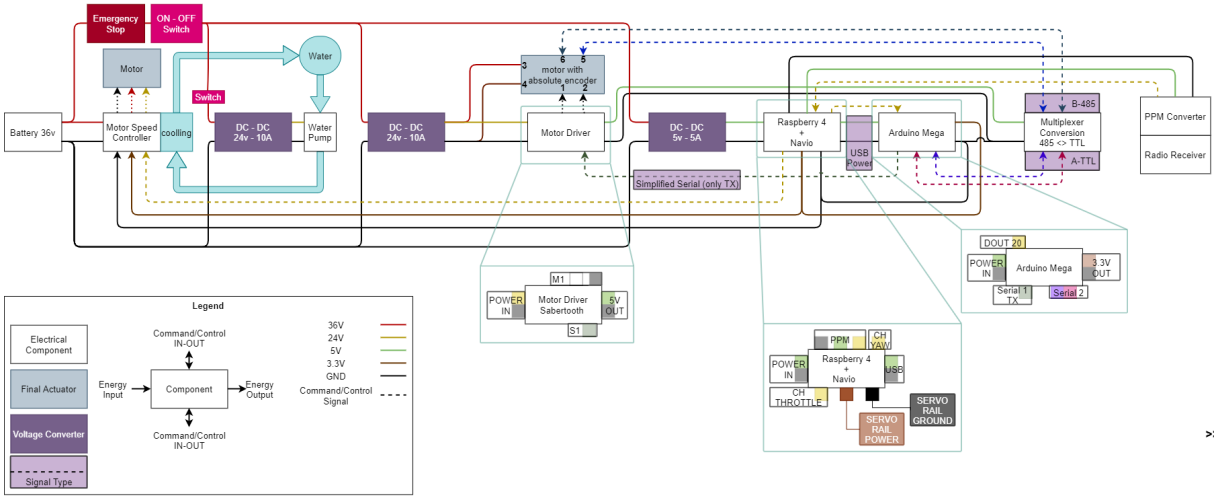

Figure 5.5: Aquanaute's Connection Diagram (HD Image [21])

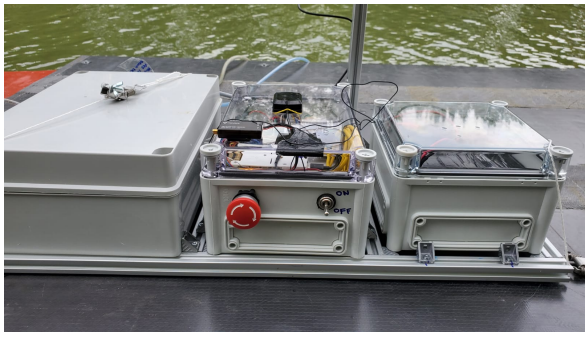

Figure 5.6: Electronic Boxes (From left to right - Pump with Speed Controller Box, Power and Control Box, Battery Box)

Figure 5.7: Close up of Power and Control Box

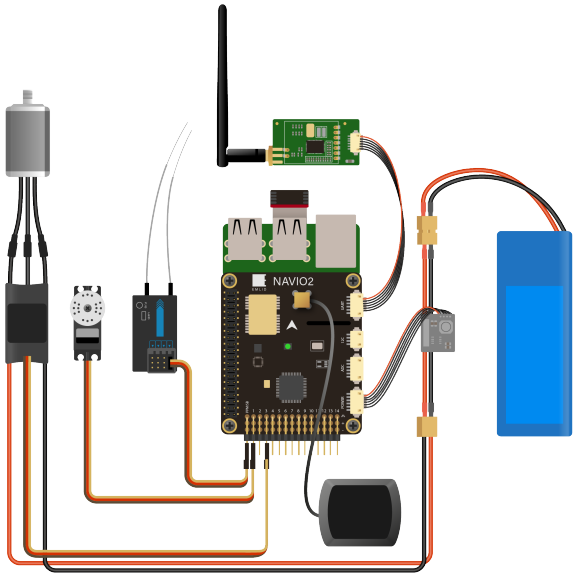

Figure 5.8: NAVIO2: Rover Typical Setup [9]

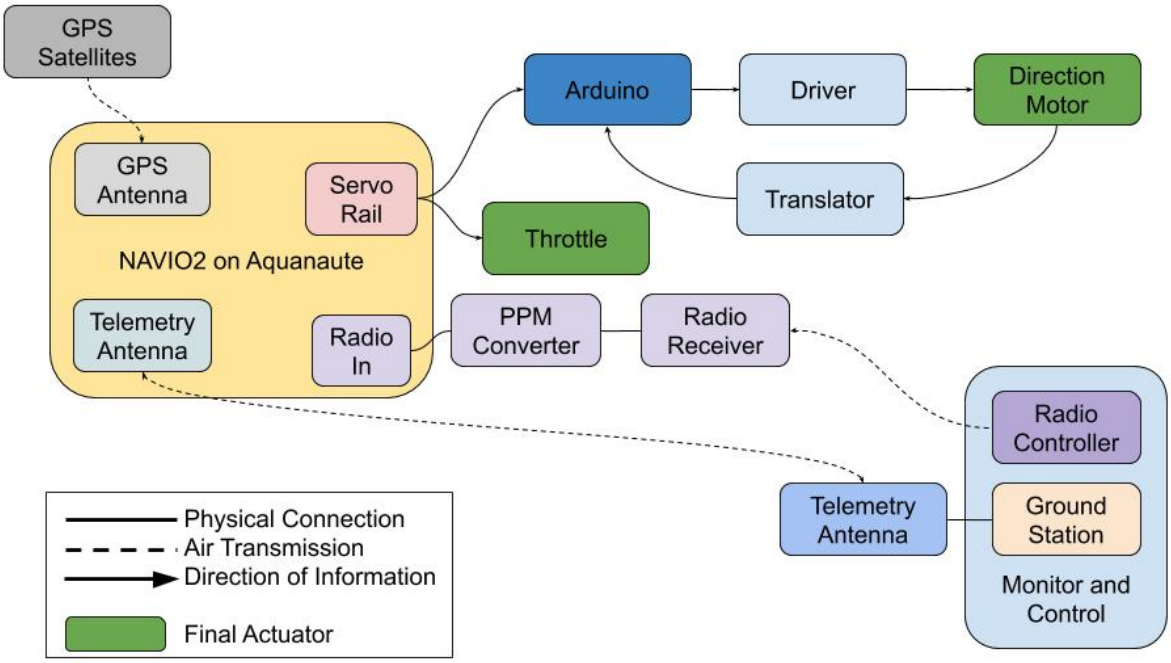

Figure 5.9: Information Architecture

Ricardo RICO URIBE / ENSTA Paris

5.3 Assembly

5.3.1 Mechanical

Each Hull has 6 holes to pass bolts and connect them to the platform, only the holes marked in blue on the figure 5.10 are easily accessible and where used to make the connection, the right hull is a mirrored version of the left.

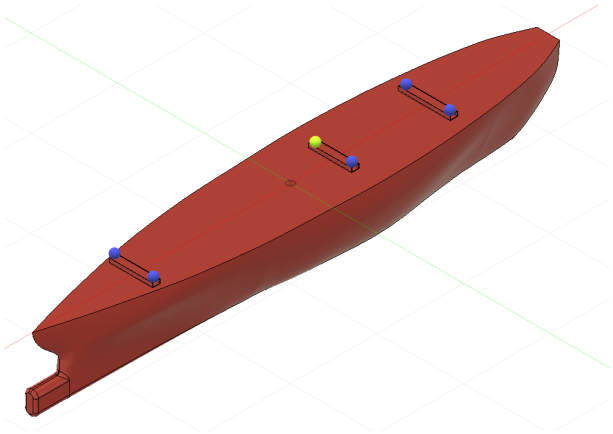

Figure 5.10: 3D Model of the left indicating the holes for the bolts to connect it to the platform

When the structure is assembled, you can rest the electronics box on top, leaving the battery box on the front, and closing the latches seen in the figure 5.11 to fix it to the platform.

Figure 5.11: Mechanical Latch used quickly lock the electronics to the platform

When you are finished with the mechanical connections the boat should look like this 5.12, keep in mind that the stickers glued to the hulls must be facing outward, and the bulb on the hulls goes to the front.

Figure 5.12: Aquanaute as seen form the right, with the front on the right of the image

5.3.2 Electric

When you have the electrical box mounted on the platform, connecting the throttle motor is very easy. The motor has three color coded cables that connect to the Pump and Controller Box and are color coded, a picture for this can be found in the figure 5.13. Also this is the switch that needs to be activated to run the pump.

Figure 5.13: Pump and Controller Box (from left to right, throttle motor connections(brown, blue and black), water exhaust(smaller diameter), water intake, pump ON-OFF switch)

The cable for the direction motor is attached to the central box (Power and Control) and can only connect in one way to the motor because of the nature of the physical connector, the direction motor connected on the figure 5.14.

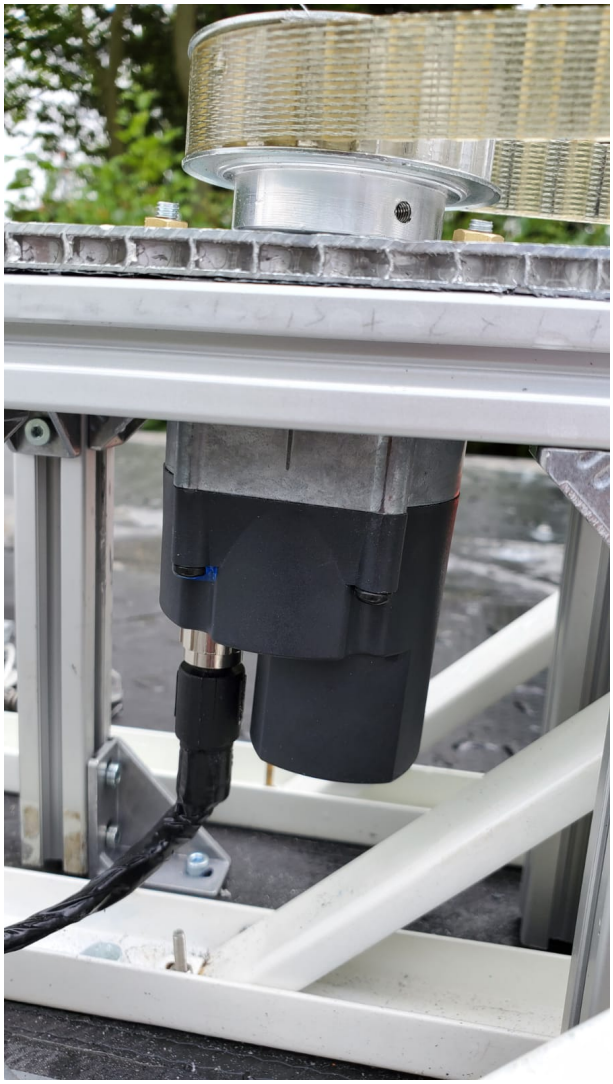

Figure 5.14: Cable Connected to the Direction Motor

If the direction motor has been turned outside of its range while disconnected from the rudder (the belt was not attached), you should power on the system without the belt to allow for the motor to center itself (part of the Arduino start-up sequence) To avoid any damage to the motor.

5.3.3 Transportation

The easiest and most efficient way to transport the boat ir to put it in the truck already assembled, with the hulls connected to the platform, and the electronics separately to avoid any damage. The boat **only** fits inside the largest vehicle ENSTA has.

5.3.4 Utilisation

For first time utilisation, you should power up the system without the belt connecting the rudder to the direction motor, to let the motor center itself as mentioned in the section 5.3.2, when the motor is centered you can connect the belt from the rudder making sure that the rudder itself its also centered. The belt can be tensioned by turning the blot seen in the figure 5.15, and in the end the belt should look like this 5.16.

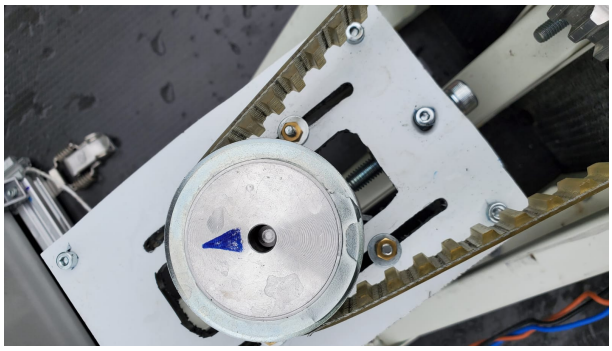

Figure 5.15: Tensioning Bolt on the right pushing on the motor

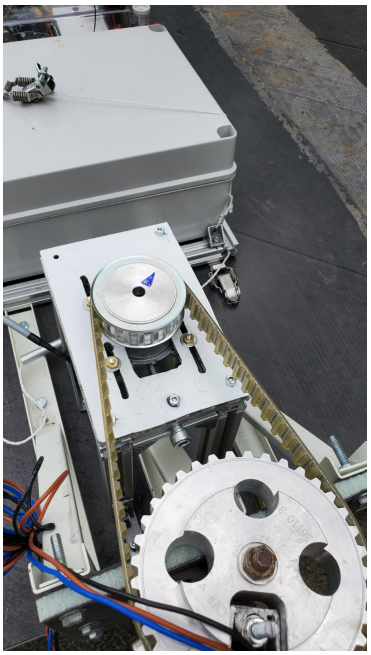

Figure 5.16: Belt Tensioned

Before running the throttle motor you have to ensure that both plastic hoses are underneath the water or at least the suction one (bigger diameter) to avoid damage to the pump.

To be able to use the throttle motor the system has to be **armed**, for this you can connect an Ethernet cable to the Raspberry or connect to the Navio via telemetry and a mission planner.

For now the system works great in manual, but the parameters for automatic use are not tuned nor tested. Use auto mode at your own caution.

5.4 Final Result

I got a functional boat, with both autopilot and manual mode capabilities. It can carry more than 80kg or a person (figure 5.17), it can be expanded to use more sensors or the actuators necessary to carry the ROV.

Figure 5.17: Aquanaute

We managed to conduct 2 test before losing the propeller to an avoidable error at the beginning of test 3.

5.4.1 Test 1

This test was at the beginning a discovery to see if the boat will float, we found that it floats too well, its flotation line is very low or the motor sits to high causing the motor to be outside the water as seen in the figure 5.18, We figure that for the time being a person has to be on top of the boat, approximately 80kg shifted toward the back are necessary to lower the motor. After this problem was resolved, the test focused on the manual parameters and communication with the base. For this test a mission planner was only necessary to arm the vehicle as the auto mode was not tested.

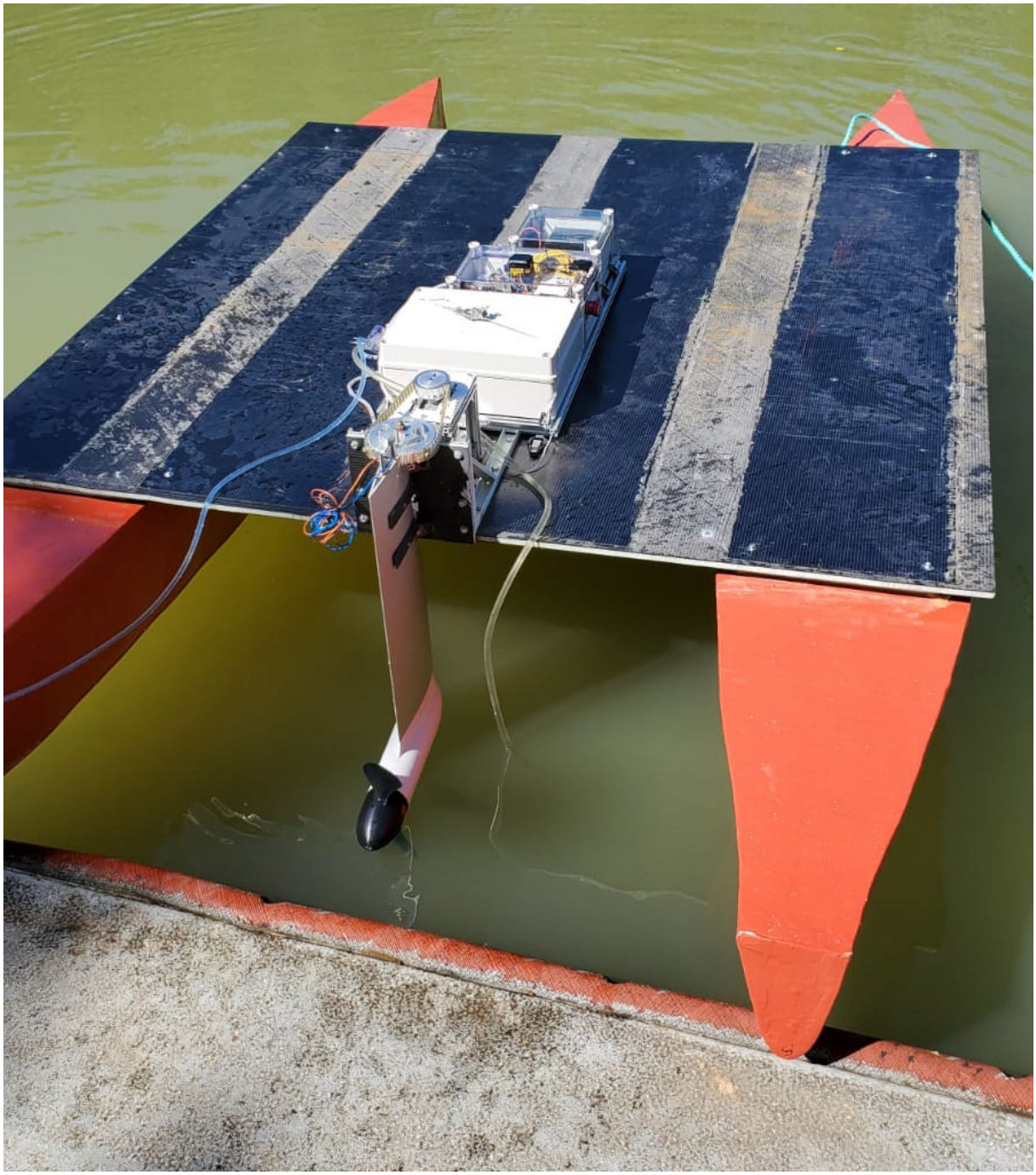

Figure 5.18: Aquanaute on the water

5.4.2 Test 2

This test was focused on trying the auto mode with out tuned parameters, we were using the default options and the test proved to be a success, the GPS was accurate, and the compass was giving a correct heading which allowed Aquanaute to move towards its target. The movement was far from perfect, it was oscillating a lot and circling a way-point when it arrived to it as seen in the figure 5.19.

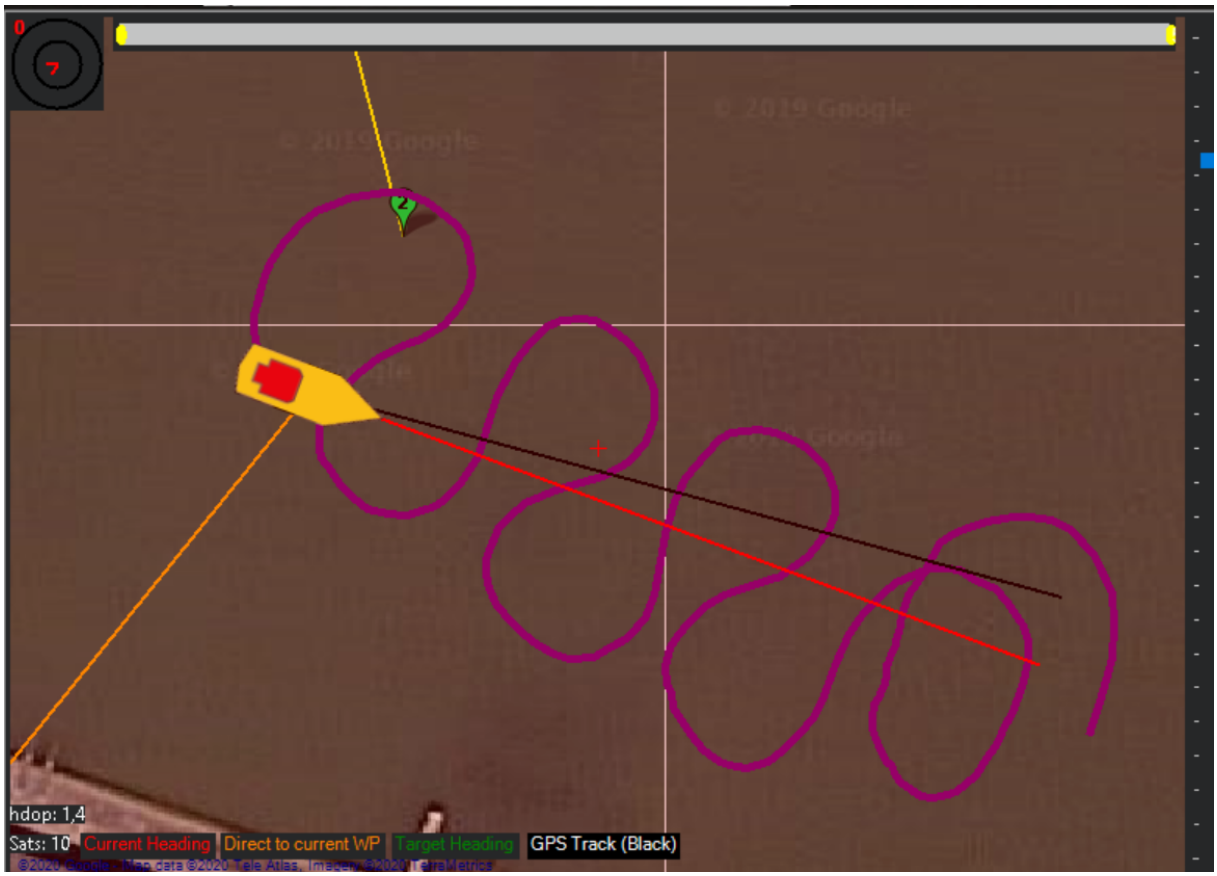

Figure 5.19: Registered Trajectory from test 2

After this test we moved to simulate the parameters and found that in simulation the boat was also oscillating like in real life. We then started to tweak the parameters in simulation and after that we loaded the new parameters to the NAVIO.

5.4.3 Changing Parameters and WARNING!

You can find the comparison between the original parameters that worked fine and the incorrect parameters that caused the lost of the propeller here (https://www.diffchecker.com/wc7uyiLf7). Our focus was to change the parameters of turning rate and turning speed to improve the oscillation, we found out that the default parameters are for a small boat that can turn very quickly, when we changed these parameters the simulation showed an improvement. We failed to notice that when loading the simulation parameters into the NAVIO we were also loading the simulated NAVIO. This erased all calibration from the GPS, compass and other sensors in the NAVIO, and also gave the throttle motor the possibility to go in reverse. When the reverse was wrongfully activated the propeller unscrew itself from the shaft and sunk to the lake bed.

For the replacement propeller the blades must be installed in the correct orientation, both producing forward force (figure 5.20) but more importantly, it is essential to find a way to secure the propeller to the shaft, it is not clear how this was done originally.

Figure 5.20: Propeller attached to the motor

Checklist for parameter file

More than checking a lengthy file that is hard to read, the best approach is to run the NAVIO separately, disconnecting it from all other hardware, you should check on the mission planner the following items.

- Calibration of the sensors
- Calibration of the RC controller
- The frame selected is a boat and not a rover (vehicle is different from frame, the vehicle must be a rover, but the frame should be a boat)
- The throttle and yaw channel are correctly set (part of the RC Calibration)
- The servo channels used for throttle and direction should have their respective functions (throttle and ground steering) and the servo pins should be mapped to the correct RC channel.

A problem caused by the simulation was the change of the Ardupilot Rover Version, this can only be checked on the file itself and must be the same as the version installed on the NAVIO.

Conclusion

Simulation is just as important as a final real object. Simulation gives you an unlimited budget, you can test all kinds of motors, sensors, chassis and configurations. But also allows you to see what problems you will find down the road when developing the product. Because Aquanaute had already its parts, the simulation allows for future development, you can test in simulation autonomous algorithms, or different sensor configurations that if work in simulation, will most likely work in real life.

You cannot do simulation and then real life or vice-versa, it must be a back and forward, because they complete each other, simulation allows for tests to dangerous for real life, but the results obtained in real life can improve the simulation, this generates a loop that will get eventually to a great final product.

Problems arise all the time, but if you understand them well enough you can plan an effective solution. Because of time constrains I was only able to identify the problems regarding Aquanaute, them being in the simulation or in the real life components. But thanks to this identification someone else can tackle these problems directly and solve them. The simulation alone should be a complete internship, as stated, we need a new simulation environment with new plugins to be able to use Aquanaute as a simulated platform. In real life, the boat can be used with caution, but a previous simulation will save the boat before any problem presents itself.

Problem Solving is the most important skill when doing a project, the project's idea wasn't to buy or to chose components, the purpose was to use what was already available and make it work, even without its proper documentation. A well documented project would had solved me a lot of headaches when building Aquanaute, understanding the original team decision and reasoning could had speed up the project. I hope that by using this document and the provided git [10] you can solve any issue, if not feel free to contact me and if that doesn't work, I am sure that an ENSTA student or engineer can solve the problem.

Bibliography

[1] Open Robotics. Is ros for me? https://www.ros.org/is-ros-for-me/, 2020. Accessed: 2020-08-15.

[2] Robot Ignite Academy. Path: "ros for beginners". https://www.robotigniteacademy.com/en/path/ros-for-beginners/, 2020. Accessed: 2020-08-15.

[3] Ricardo RICO URIBE. Mesurements to obtain the cad of the hulls. https://u2is.atlassian.net/wiki/spaces/SR/pages/104071194/Mesures+pour+faire+le+CAO+des+coques, 2020. Accessed: 2020-08-15.

[4] Robot Ignite Academy. Course: "urdf for robot modeling". https://www.robotigniteacademy.com/en/course/robot-creation-with-urdf-ros/details/, 2020. Accessed: 2020-08-15.

[5] Ricardo RICO URIBE. Aquanate urdf. http://git-u2is.ensta.fr/u2is/aquanaute/blob/develop/aquanaute_description/urdf/aquanaute_description.urdf.xacro, 2020. Accessed: 2020-08-15.

[6] RoboNation. Virtual robotx competition. https://robotx.org/programs/2019-virtual-robotx-competition/, 2020. Accessed: 2020-08-15.

[7] Marine Advanced Research. Wam-v usv. https://www.marinetechnologynews.com/news/drone-boats-global-robotx-563204, 2020. Accessed: 2020-08-15.

[8] Brian Bingham, Carlos Aguero, Michael McCarrin, Joseph Klamo, Joshua Malia, Kevin Allen, Tyler Lum, Marshall Rawson, and Rumman Waqar. Toward maritime robotic simulation in gazebo. In *Proceedings of MTS/IEEE OCEANS Conference*, Seattle, WA, October 2019.

[9] EMLID. Navio2 documentation. https://docs.emlid.com/navio2/, 2020. Accessed: 2020-08-15.

[10] Aquanaute repository. http://git-u2is.ensta.fr/u2is/aquanaute, 2020. Accessed: 2020-08-15.

[11] Thomas Simon. Aronnax's connection diagram. https://u2is.atlassian.net/wiki/plugins/servlet/ac/com.mxgraph.confluence.plugins.diagramly/customContentViewer?content.plugin=ac%3Acom.mxgraph.confluence.plugins.diagramly%3Adrawio-diagram&space.key=SR&content.id=96142292&content.version=5&space.id=62554114&content.type=custom, 2020. Accessed: 2020-08-15.

[12] Ricardo RICO URIBE. Available components. https://u2is.atlassian.net/wiki/spaces/SR/pages/92143624/Caracteristiques+des+Composants, 2020. Accessed: 2020-08-15.

[13] Hydrocontest documentation. https://www.hydrocontest-x.ch/?page_id=324&lang=en, 2020. Accessed: 2020-08-15.

[14] Hydrometra contact. https://www.facebook.com/hydrometra/, 2020. Accessed: 2020-08-15.

[15] Watt & sea web-page. https://www.wattandsea.com/fr/, 2020. Accessed: 2020-08-15.

[16] Faulhaber. Industrial actuator with multiturn capabilities. https://www.faulhaber.com/fileadmin/Import/Media/EN_6091_DFF.pdf, 2020. Accessed: 2020-08-15.

[17] Dimension Engineering. Sabertooth dual 25a motor driver. https://www.dimensionengineering.com/products/sabertooth2x25, 2020. Accessed: 2020-08-15.

[18] Faulhaber. Industrial actuator, technical information. https://www.faulhaber.com/fileadmin/Import/Media/EN_TECHNICAL_INFORMATION_INDUSTRIAL_ACTUATOR.pdf, 2020. Accessed: 2020-08-15.

[19] DF Robot. TI0070 multi usb rs232 rs485 ttl converter. https://wiki.dfrobot.com/Multi_USB_RS232_RS485_TTL_Converter__SKU_TEL0070_, 2020. Accessed: 2020-08-15.

[20] HiTec. Hpd-07rh qpcm - 7 ch. fm dual conversion qpcm receiver. https://hitecrcd.com/products/aircraft-radios-receivers-and-accessories/discontinued-aircraft-products/hpd-07rh-qpcm-7-ch.-fm-dual-conversion-qpcm-receiver/product, 2020. Accessed: 2020-08-15.

[21] Ricardo RICO URIBE. Aquanaute's connection diagram. https://u2is.atlassian.net/wiki/plugins/servlet/ac/com.mxgraph.confluence.plugins.diagramly/customContentViewer?content.plugin=ac%3Acom.mxgraph.confluence.plugins.diagramly%3Adrawio-diagram&space.key=SR&content.id=93684186&content.version=11&space.id=62554114&content.type=custom, 2020. Accessed: 2020-08-15.

Glossary

A Amperes

CAD Computer Assisted Design

ENSTA École Nationale Supérieure de Techniques Avancées

GPS Global Positioning System

IMU Inertial measurement unit

OS Operating System

ppm Pulse Position Modulation

RC Radio Control Signal

ROS Robot Operating System

ROV Remotely operated underwater vehicle

RVIZ Robot Visualization Interface

SITL Simulation In The Loop

TF Kinematic Transformation Between Coordinate Frames

TTL Transistor-Transistor Logic

TX Transmission

URDF Universal Robot Descriptor File

USA United States of America

VRX Virtual RobotX

XACRO Macro for XML

List of Tables

5.1 Available Components used in Aquanaute [12] 31

List of Figures

1.1 Use of Time During the Project . . . 12

2.1 Final Exam with Robot Husky . . . 14
2.2 Final Exam with AR Drone . . . 14

3.1 Test of Photogrametry . . . 16
3.2 Gathering of the relative position of points along the hull's curve. . . . 16
3.3 Mesh obtained with a Stereo Camera (location of the hull in red) . . . 17
3.4 Final model for the Hull . . . 17
3.5 Platform model with its correct orientation . . . 18
3.6 Motor Rudder Assembly Model . . . 19
3.7 3D model of Aquanaute Assembled . . . 20
3.8 Model as Described by the URDF (In RVIZ) . . . 21
3.9 WAM-V Boat of the RobotX Challenge [7] . . . 22
3.10 VRX Simulation Running in Gazebo . . . 22
3.11 VRX forces applied to Aquanaute . . . 24
3.12 How to Mount Navio to Raspberry [9] . . . 25
3.13 Normal Configuration of SITL . . . 26
3.14 Simulation Diagram . . . 27
3.15 Maritime Simulation of Aquanaute . . . 28

4.1 Annorax's Connection Diagram (HD image [11]) . . . 29

5.1 Watt & Sea Motor Characteristics [14] . . . 32
5.2 Sabertooth Configuration . . . 33
5.3 Battery Information . . . 34
5.4 Fuse Available . . . 34
5.5 Aquanaute's Connection Diagram (HD Image [21]) . . . 35
5.6 Electronic Boxes (From left to right - Pump with Speed Controller Box, Power and Control Box, Battery Box) . . . 35
5.7 Close up of Power and Control Box . . . 35
5.8 NAVIO2: Rover Typical Setup [9] . . . 36
5.9 Information Architecture . . . 36
5.10 3D Model of the left indicating the holes for the bolts to connect it to the platform . . . 37
5.11 Mechanical Latch used quickly lock the electronics to the platform . . . 37
5.12 Aquanaute as seen form the right, with the front on the right of the image . . . 38

5.13 Pump and Controller Box (from left to right, throttle motor connections(brown, blue and black), water exhaust(smaller diameter), water intake, pump ON-OFF switch) 38
5.14 Cable Connected to the Direction Motor 39
5.15 Tensioning Bolt on the right pushing on the motor 40
5.16 Belt Tensioned 40
5.17 Aquanaute 41
5.18 Aquanaute on the water 42
5.19 Registered Trajectory from test 2 43
5.20 Propeller attached to the motor 44

Annexes

With this report you will find:

- The logs for the tests
- Full resolution images of the diagrams
- The Code for the Arduino adapter
- The code to run the simulations (you need to have installed ROS-gazebo, Ardupilot and a mission planner)
- The meshes obtained from the Stereo Camera when scanning the Hull
- The CAD models in Fusion360, Inventor and STP format
- The article *Toward Maritime Robotic Simulation in Gazebo*[8] used to understand the maritime simulation
- The Ardupilot parameters used when simulating, the parameters used in test 1 and 2, and the incorrect parameters

If you need more information you can refer yourself to the laboratory's Wiki