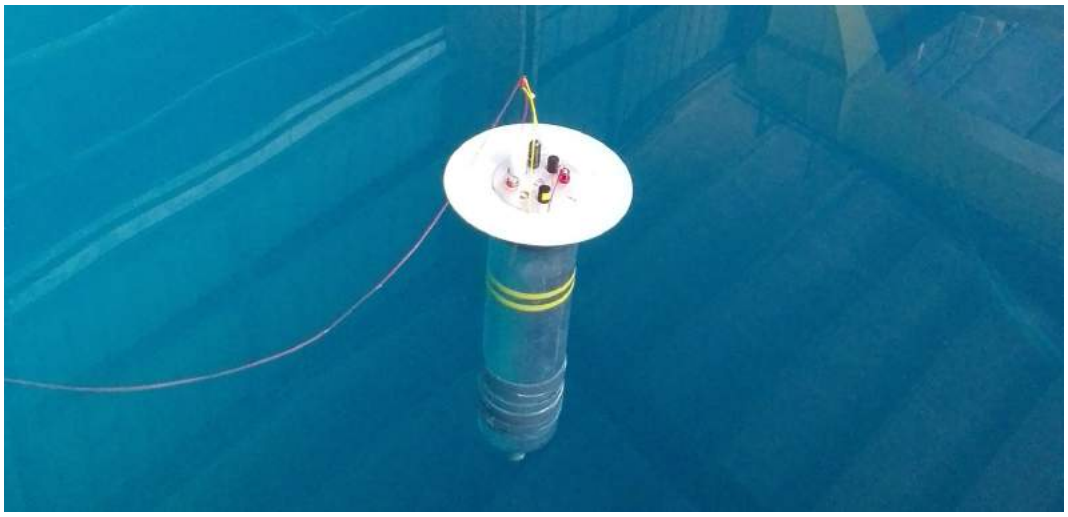
Internship report

Alexandre HOUDEVILLE
School year : CI 2019, Profile : SPID/ROB
E-mail address : alexandre.houdeville@ensta-bretagne.org

Depth regulation of autonomous underwater floats

August 22, 2019



SUPERVISOR : Aurélien PONTE
TUTORS: Luc JAULIN, Thomas LE MEZO

ADDRESS: Centre Ifremer Bretagne, ZI Pointe du Diable, CS 10070, 29280 PLOUZANÉ (France)

Acknowledgements

I would like to thank Aurélien PONTE profusely, who consented to take me on as a trainee student in the Ifremer company at Plouzané and whose reactivity made the organisation of this internship possible under the best possible conditions. I also thank him for all the benefits he provided for me during this internship.
Similarly, I would like to thank Professor Luc JAULIN, who had offered me his help to apply for this end-of-studies project offer and put me through to Aurélien PONTE. I would also highlight that the robotic courses he ensured during the year were very useful to perform some tasks in the project within the scope of the internship.
I also would like to thank the members of the Ifremer TOIS team as they helped me much during this internship and it was a pleasure to develop new skills with them : in particular Olivier PEDEN, the mechanics who assisted me during the tests performed in the Ifremer pool ; Michel HAMON, the electronics engineer who helped me to develop the software part of the Ifremer float ; Philippe LE BOT, the computer engineer who offered me assistance in network tasks ; Stéphane LEIZOUR, the technician who led the deployment phase of the float during the test campaign in La Seyne-sur-Mer ; and last but not least Catherine KERMABON, the project manager who helped me throughout this internship in the completion of some tasks.
Finally, I would like to thank very much Thomas LE MEZO from ENSTA Bretagne for the precious and regular help he brought to me during this internship and for having put his float prototype at my disposal.

Contents

Abstract / Résumé

Mots clés

Flotteur autonome sous-marin, asservissement en profondeur, régulation, algorithmes de positionnement, suivi d'isobare, suivi d'isotherme, commande par retour d'état, filtre de Kalman

Résumé

Ce rapport a pour but de décrire le travail effectué lors de mon projet de fin d'études dans le centre Ifremer de Plouzané, dans son Laboratoire d'Océanographie Physique et Spatiale (LOPS).
L'objectif du projet qui m'a été attribué pendant ce stage consistait en l'amélioration de la capacité des deux types de flotteurs conçus par l'Ifremer et l'ENSTA Bretagne, à contrôler leur positionnement en profondeur.

Keywords

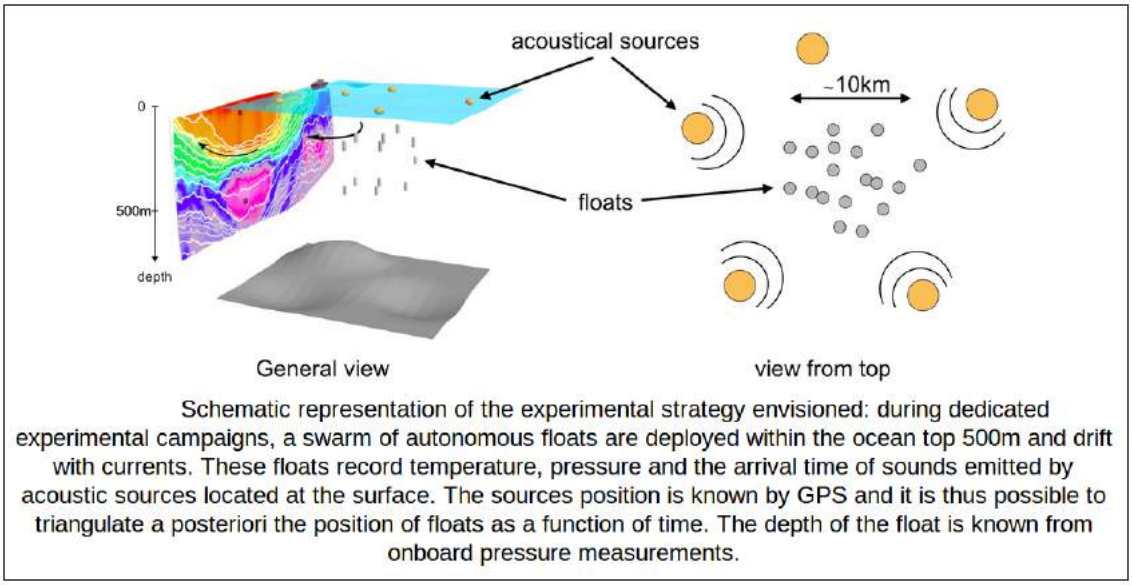
Autonomous underwater float, depth control loop, regulation, positioning algorithms, isobar follow-up, isotherm follow-up, state feedback regulation, Kalman filter

Abstract

This report aims at describing the work accomplished during my end-of-studies project in the Ifremer complex of Plouzané, in its Laboratory of Physical and Spatial Oceanography (LOPS). The goal of the project which was assigned to me consisted in improving the ability of two kinds of floats designed by Ifremer and ENSTA Bretagne, to control their depth positioning.

Introduction

This end-of-studies internship is a part of a project entitled COGNAC which stands for "mesure de la Circulation Océanique à fine échelle par GéolocalisatioN ACoustique de flotteurs dérivants autonomes" (measure of the Ocean Circulation at small scale by ACoustic GeolocalisatioN of autonomous drifting floats).
This project is led by Aurélien PONTE at Ifremer and aims at studying the ocean dynamics at small scales (<10km). In order to do this, he would like to deploy about thirty floats able to dive at different depths and follow isobars or isotherms.
Then, acoustic signals emitted by GPS tracked acoustic sources will be used so as to geolocate floats which are equipped with receivers by triangulation.
Finally, the data collected by the float sensors like pressure or temperature for now, will be used in order to describe the ocean dynamics at small scales (<10km).



Schematic representation of the experimental strategy envisioned: during dedicated experimental campaigns, a swarm of autonomous floats are deployed within the ocean top 500m and drift with currents. These floats record temperature, pressure and the arrival time of sounds emitted by acoustic sources located at the surface. The sources position is known by GPS and it is thus possible to triangulate a posteriori the position of floats as a function of time. The depth of the float is known from onboard pressure measurements.

Figure 1: Illustration of the COGNAC project [1]

My internship dealt with the issue of regulating the vertical position of the float according to the depth or temperature of the environment in which it is supposed to progress. The interest of making the float follow isobars and mostly isotherms would allow to keep the float in a same mass of water and therefore to study its movement by drifting with it. Indeed masses of water which are characterised by the same density often share the same temperature.

Within the scope of this project, I worked with the team of the Laboratory of Physical

and Spatial Oceanography (LOPS) at Ifremer but I also worked in collaboration with ENSTA Bretagne, particularly with Thomas LE MEZO, a PhD student developing a prototype similar to the one that Ifremer has designed.

In this report, I will first set the context of my internship by describing the company, the COGNAC project as a whole and I will also present the low-cost prototypes developed by ENSTA Bretagne and Ifremer. I will then describe the algorithmic and software parts of the Ifremer float, which were at the centre of my work during this internship. Afterwards, I will describe the different tests performed and analyse their results. Finally, I will complete by explaining the project management followed during this internship.

Chapter 1

Contextual setting

In this part, I will present the company in which I performed my end-of-studies project from the 1^{st} of April until the 30^{th} of August 2019 and I will describe the COGNAC project with more details.

1.1 Presentation of the hosting company: Ifremer

1.1.1 History

The company IFREMER, standing for "Institut Français de Recherche pour l'Exploitation de la MER" (French Institute of Research for the Exploitation of the Sea) was founded in 1984 from the merger of the CNEXO (Centre National pour l'EXploitation des Océans) and the ISTPM (Institut Scientifique et Technique des Pêches Maritimes).

1.1.2 Status and application fields

Ifremer has the EPIC [2] status in France : "Etablissement Public à caractère Industriel et Commercial" (public establishment of an industrial or commercial nature). This status allows it to lead activities related to sensitive sectors whose successful functioning is necessary and controlled by the state decision while this company does not belong to the state. Ifremer is attached to the ministry in charge of higher education, research and innovation, and ecological transition.

Ifremer takes part in several fields related to the monitoring and the exploitation of the ocean environment like physics (fluid mechanics), robotics, energy, biology, chemistry, statistics... The communal mission shared by all these fields consists in monitoring and understanding the marine environment and the ecosystems at different scales.

1.1.3 Means

To lead the different missions assigned to Ifremer, this organisation has at its disposal many means from cutting-edge technology. Indeed, Ifremer possesses three oceanographic ships along with many coastal boats. In addition, it has several submersibles like "Nautile", a manned submarine able to dive down to 6000 m or "Victor", a Remotely Operated Vehicle (ROV). Moreover, Ifremer owns different facilities allowing to simulate real and extreme conditions as well as possible so as to assess the behaviour of the prototypes studied, like laboratories, hyperbaric chambers or a large and 20-metre-deep pool able to reproduce swell and wind conditions.

Figure 1.1: Nautile

Figure 1.2: Hyperbaric chamber

Ifremer is spread on different geographical centres and its headquarters are in Brest since this year. These sites are mainly spread around France, but there are also other facilities all around the world.

Figure 1.3: Ifremer facilities in Metropolitan France [3]

1.1.4 Organisation

The whole company Ifremer is currently directed by François HOULLIER and comprises 1500 employees including 600 engineers/researchers with an annual budget of 200 million euros. It is divided into 4 scientific departments :

- Department of Biological Resources and Environment
- Department of Physical Resources and Deep-Sea Ecosystems
- Department of Oceanography and Ecosystem Dynamics
- Department of Marine and Digital Infrastructures

The Ifremer institute of Plouzané is currently directed by Antoine DOSDAT. This site employs around 1000 employees including 600 employees attached to this site.

Figure 1.4: Ifremer institute of Plouzané

1.1.5 My workplace: the Laboratory of Physical and Spatial Oceanography

During this internship, I worked in the Laboratory of Physical and Spatial Oceanography, belonging to the Department of Oceanography and Ecosystem Dynamics (ODE).

This laboratory whose organisation chart can be found in the appendix (see "Organisation chart of the LOPS"), aims at documenting the state of the oceans so has to have a better insight about processes which control the physical and biophysical evolution of the ocean.

Four research teams work on related subjects about the ocean. My supervisor Aurélien PONTE works in the Ocean Scale Interaction team (équipe Interaction d'Echelles Océaniques [IEO]) which aims at understanding how processes with different spatiotemporal scales interact in the ocean.

1.2 Presentation of the COGNAC project

1.2.1 State of the art

As a reminder, the COGNAC project aims at studying the dynamics of the oceans at small scale by deploying swarms of floats able to dive at different depths to follow isobars (constant depth) or isotherms and to measure data in order to characterise water masses.

Several types of autonomous drifting floats have already been developed for oceanographic purposes. For instance, this is the case for the floats used within the scope of the ARGO project launched since the early 2000s and to which Ifremer contributes. The ARGO project is an international project which consists in deploying profiling floats all around the marine world to collect data about temperature, salinity, currents, etc so as to study water masses' behaviour in the Earth's oceans at basin scale and climate change.

The deployment of the global array of 3,800 free-drifting profiling floats was completely achieved in 2007.

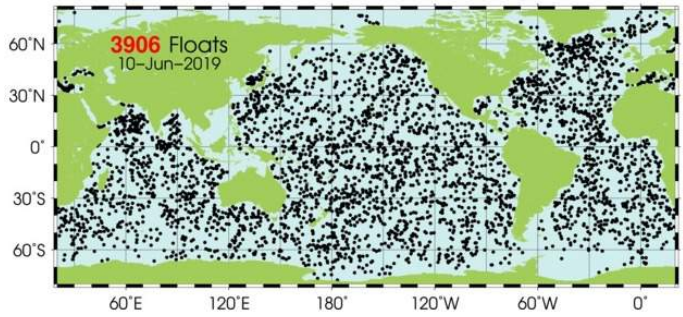


Figure 1.5: Positions of the ARGO floats all around the world on the 10^{th} of June 2019 [4]

Those floats [5] have several interesting features, indeed they are first easy to deploy thanks to their low weight (20–30 kg depending on the sensors embarked) and their restricted size (about one-metre long and 15-centimetre as diameter).

Figure 1.6: ARGO float [4]

Moreover, those floats have been made completely autonomous for ten-day missions corresponding to a sequence of different phases. First the probes are supposed to dive until 1000 metres and to drift at this depth for 9 days to analyse large-scale motions in the ocean. Then, the floats are supposed to descend to 2000 metres and to rise to the surface by measuring data like conductivity, temperature and pressure so as to establish salinity and density profiles of the water masses gone through. Seawater density is important in determining large-scale motions in the ocean. Once the float has resurfaced, satellites allow to transmit the data to shore and the position of the float is recorded thanks to its GPS coordinates.

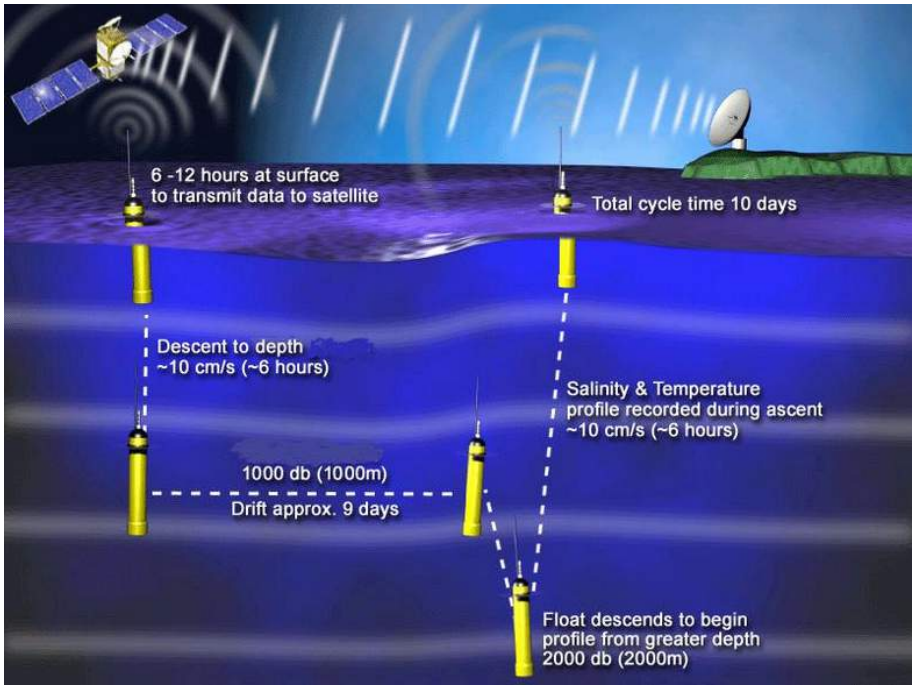


Figure 1.7: ARGO float : detailed diagram for a complete working cycle [4]

In short, the ARGO project has allowed to successfully develop a complete and efficient array of autonomous profiling floats all around the oceans, collecting data continuously to monitor the oceans in real time. However, deploying such a system is not cost free since each float costs on average 15,000 € for the manufacturing plus about 5000 € for the maintenance during all its lifespan estimated at five years. These costs combined with the need to deploy a large number of floats so as to have a fine-meshed network of the oceans are very limiting and do not suggest that such a system could be developed one day by a single company. Moreover, it would be complicated to implement new features on those floats like modifying their acoustic system or regulating them with different command laws, hence the launch of the COGNAC project.

In addition, as part of his doctorate, Thomas LE MEZO, has developed a low cost float prototype similar to the one that Ifremer has designed. Unlike Ifremer's float, Thomas LE MEZO's float aims at travelling vertically but also horizontally thanks to the ocean currents near the surface while the Ifremer float is only supposed to progress vertically and more deeply, which explains the different choices of design for both floats that will be described afterwards. Much collaboration and many exchanges have been set up between ENSTA Bretagne and Ifremer to lead the design of both prototypes : the ENSTA and Ifremer floats with the best opportunities. Thus during this internship, I was required to study the ENSTA prototype before working on the case of the prototype made by Ifremer.

1.2.2 Project issue and requirements

As mentioned before, the COGNAC project aims at studying the ocean dynamics by deploying about thirty floats able to submerge at different depths to follow isobars or isotherms. To design the COGNAC float, main requirements had been defined by Aurélien PONTE before

this internship. Indeed, it had been decided for the float to perform measurements down to 500 metres because this area is interesting as it demarcates the transition between the near-surface layer (i.e. where the ocean exchanges gas and heat with the atmosphere) and the deep ocean, which requires that the float has to resist a pressure of more than 50 bars.

In addition, the float will be deployed during experimental campaigns of one month on average. Thus the float will also have to be energetically-autonomous for such a duration. Regular data transmissions will also be required in the course of the missions to steadily analyse the results.

Finally, as explained before, the unit price of each float must be limited in order to deploy a large group of floats while respecting the typical budget allocated to oceanographic research projects (100 k€). With this aim in mind, A. PONTE has fixed a maximum unit cost of 3000 € for the production of each float.

When I started this internship, as the COGNAC project had already been launched for two years, many objectives had already been achieved. Indeed, in 2018 two engineering students (Emilie ARGOUARCH from ENSTA Bretagne and Paul TROADEC from Polytech Tours) respectively dealt with the electronic and mechanical part of the system. Consequently, when I started this internship, two employees of Ifremer working on the COGNAC project were finishing the mechanical assembly of the Ifremer prototype. Olivier PEDEN was finishing to assemble the different mechanical elements drawn by Paul TROADEC during his internship while Michel HAMON was completing the work of Emilie by installing on the float the different electronic elements chosen before like sensors, circuit boards and the piston.

Besides, as regards the software part, Thomas LE MEZO has developed the software part of its own float prototype on which I much relied so as to build the software part of the Ifremer prototype.

1.2.3 Formalisation through system engineering

For any kind of project, system engineering is very important since it allows to formalise the requirements given by the customer. In this way the customer can assess the understanding of its requirements by the engineer. System engineering is also a way to make engineers work according to the same basics. Finally, it allows to break down the main system by assigning specific functions to the different subsystems. Different methods can be used to apply system engineering in a project. I chose to use the APTE method [6] as I am in the habit of using it and as I think the tools offered by this method are very efficient. This method was created by the French company APTE.

To describe the specifications of the system, the APTE Method provides two interesting diagrams which allow people to have a quick and intuitive knowledge of the system. The horned beast diagram describes the goal of the system and its working environment.

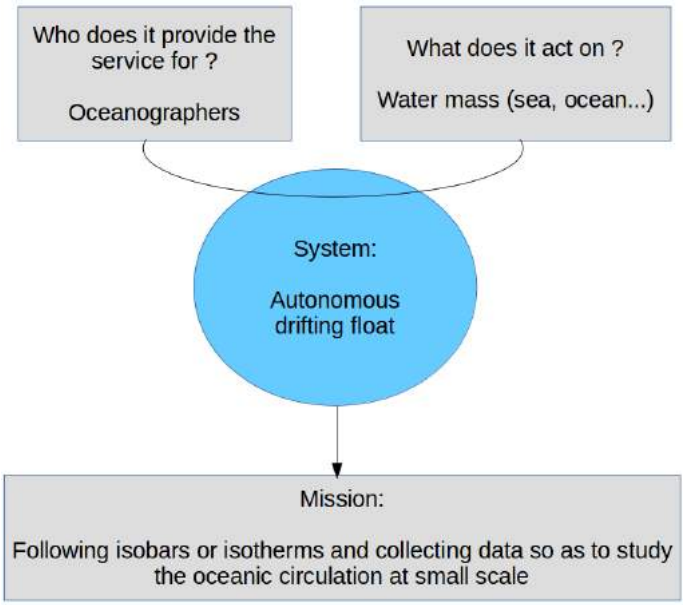
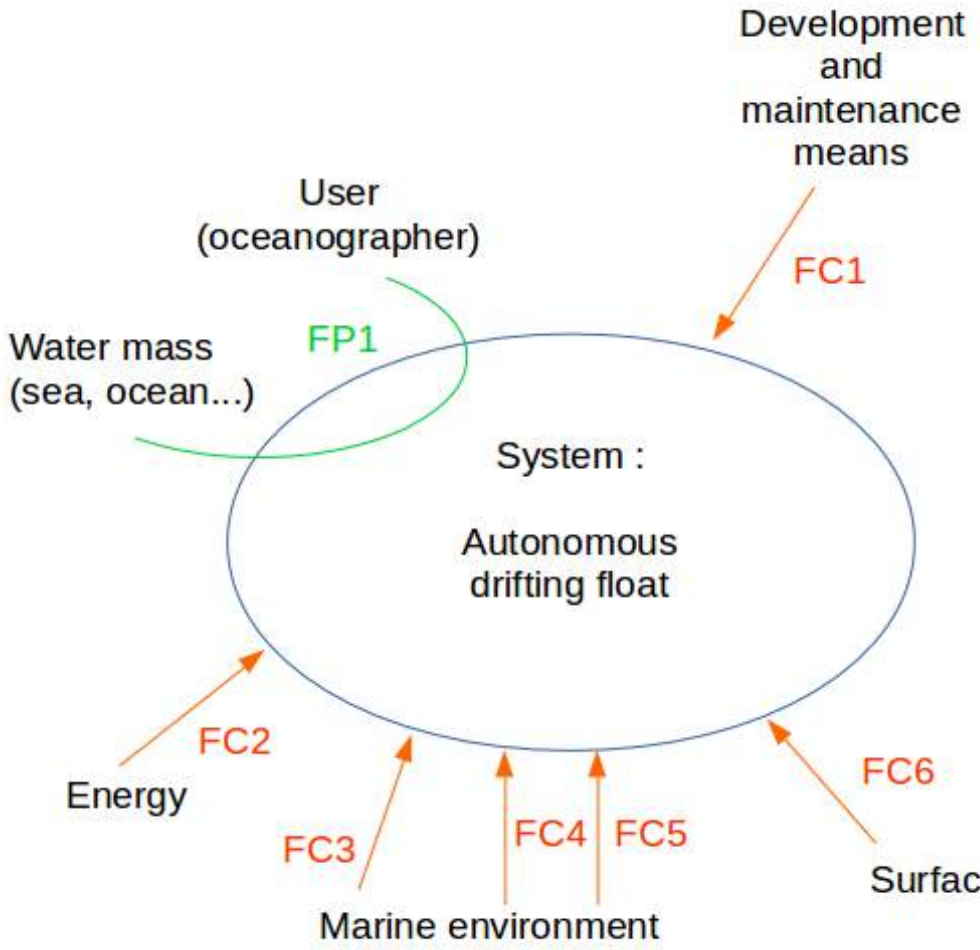


Figure 1.8: Horned beast diagram of the COGNAC product

The "octopus diagram" raises the different interactions between the system and its environment. It is a clever way to summarise the different functions of the system.



FP1 : collecting data from water masses so as to allow the user to study the oceanic circulation at small scale

FC1 : the float is supposed to be produced and maintained thanks to low cost means (production costs below 3000 €)

FC2 : the float is supposed to be energetically-autonomous for at least 30 consecutive days

FC3 : the float is supposed to behave autonomously in its environment

FC4 : the float is supposed to resist the marine environment

FC5 : the float is supposed to be easily deployed and recovered in the marine environment

FC6: the float is supposed to be located in relation to the surface with a 10-metre precision

Figure 1.9: Octopus diagram of the COGNAC product

Functional architecture

Functional architecture is the step which consists in dividing the system into different subsystems according to the functions it should carry out, however this step does not require knowing exactly the components which will be used to build the system yet. In order to describe the technical architecture of the COGNAC float, I leant on the following FAST diagrams which provide an overview of the different subsystems which comprise the main system:

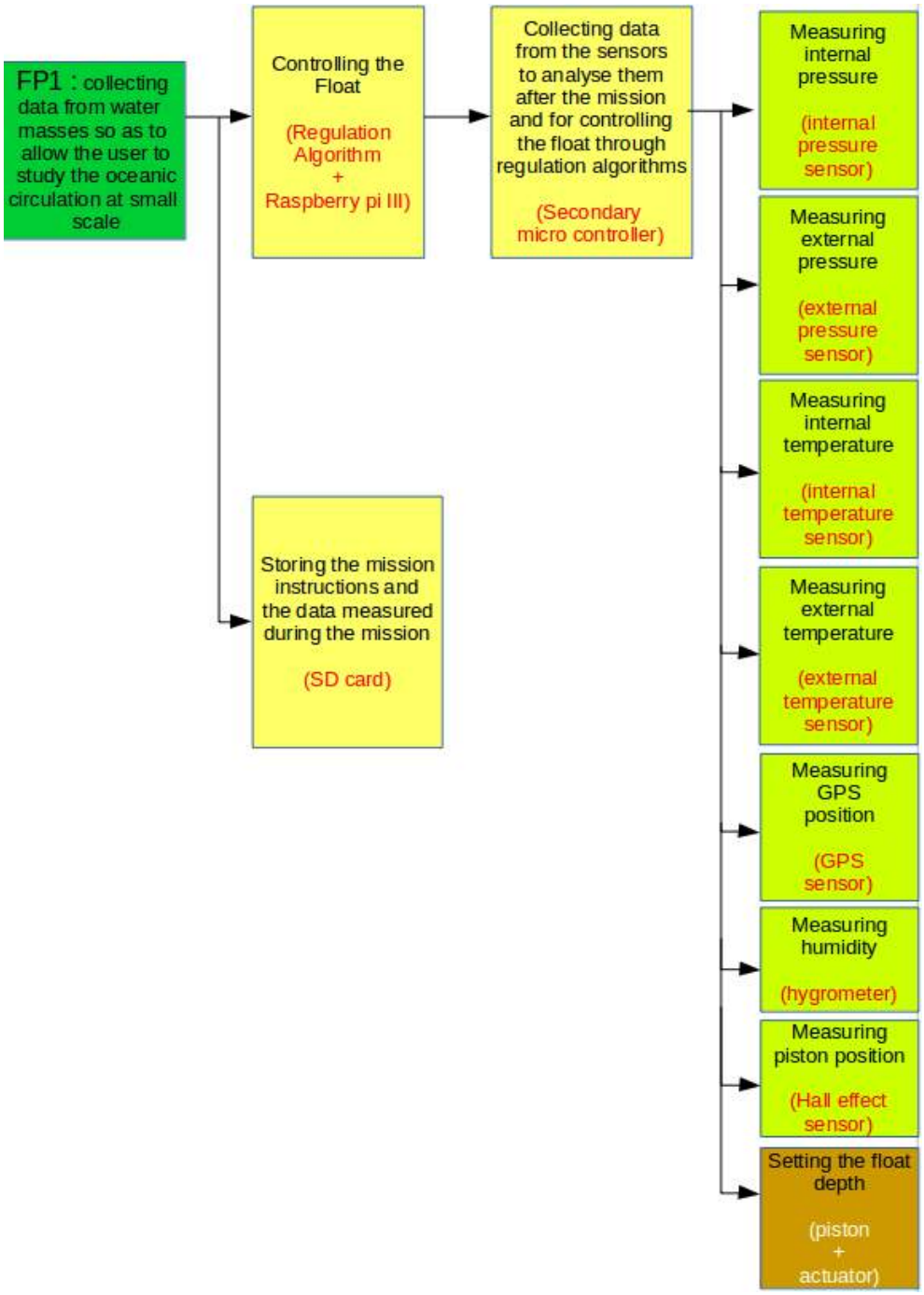


Figure 1.10: Fast diagram : FP1

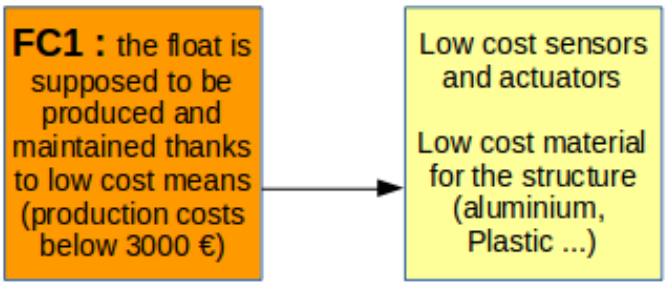


Figure 1.11: Fast diagram : FC1

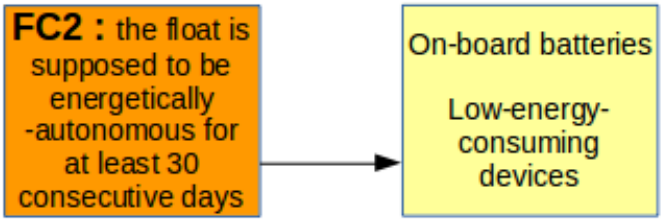


Figure 1.12: Fast diagram : FC2

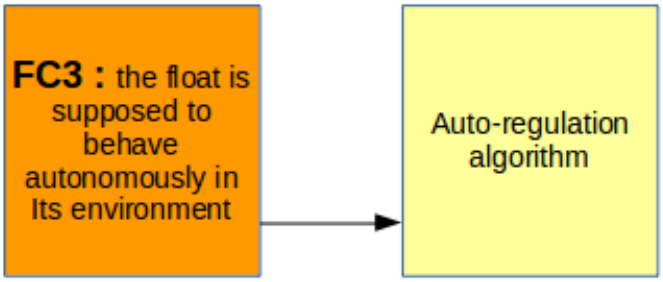


Figure 1.13: Fast diagram : FC3

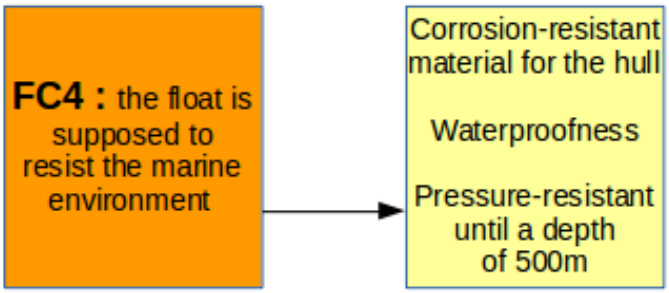


Figure 1.14: Fast diagram : FC4

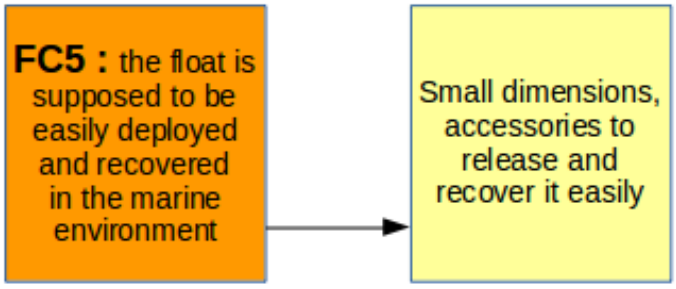


Figure 1.15: Fast diagram : FC5

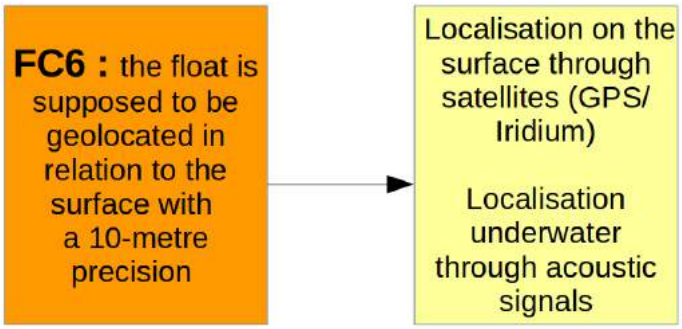


Figure 1.16: Fast diagram : FC6

1.3 The low-cost prototypes used within the scope of the COGNAC project

In this part, I will describe the prototypes considered during this internship. Then I will explain how the Ifremer float benefited from Thomas LE MEZO's float.

1.3.1 The ENSTA float

The ENSTA float has been built by Thomas LE MEZO and a team of researchers and engineers as part of his doctorate that consists in developing a submarine float aiming at reaching waypoints thanks to the ocean currents and propellers for horizontal movements and a movable piston for vertical displacements. This float is part of the COGNAC project since it has been studied by Ifremer as a model so as to design the electronic and software parts of the Ifremer float, nevertheless its mechanical part was designed independently of the ENSTA float.

The ENSTA float has been designed to be low-cost. Its plastic hull made of polycarbonate and its caps and piston made of polyoxymethylene allow to avoid corrosion due to the ocean environment. The different elements of the float have been designed to make it resist the pressure until a depth of 50 metres.

The float has available an auto-ballasting system working on a piston so as to modify the volume of the air inside, which directly modifies the Archimedean buoyant force and consequently controls the depth of the float.

Moreover, the float is more compressible than water which makes the float unstable. Indeed, according to the graph below, when the float is less compressible than water ($\chi < 0$), if the float is supposed to be neutral buoyant for a depth z, and if it moves of a displacement δz, thus the variation of volume due to this displacement will produce a force in the opposite direction of the movement so that the float will come back to the depth z.
Now, if we consider that the float is more compressible than water ($\chi > 0$), if the float moves of a displacement δz, the variation of volume due to this displacement will now produce a force in the same direction of the movement, consequently, the float will move away from its initial position.

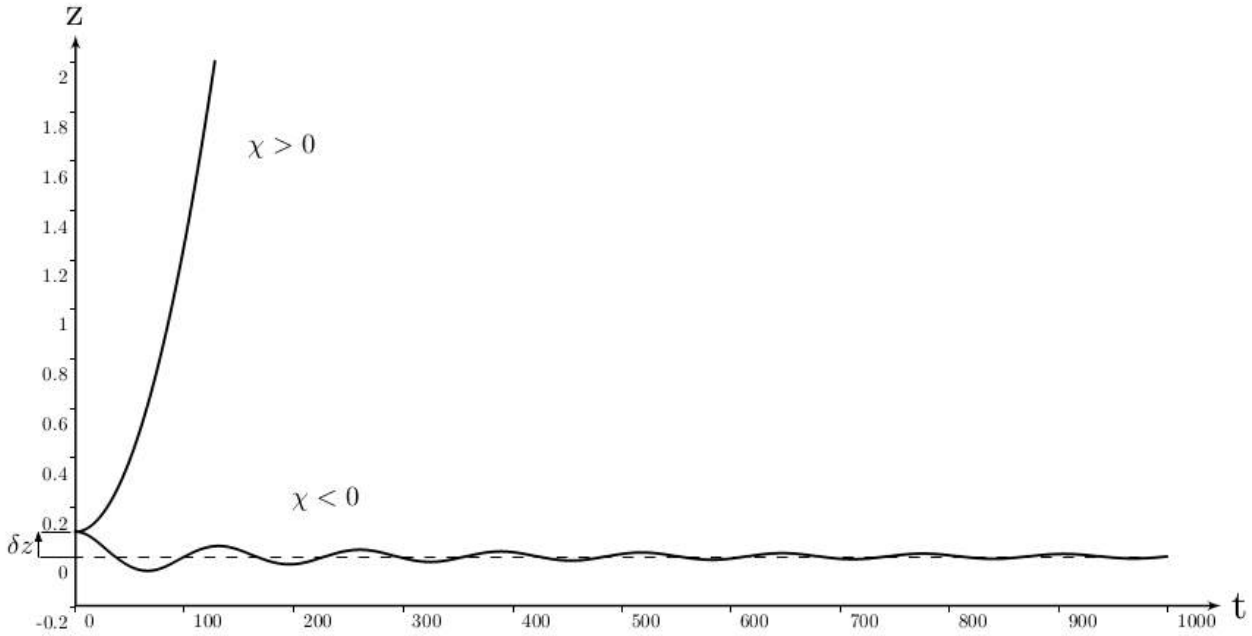


Figure 1.17: Evolution of float depth according to its compressibility in comparison to water [7]

Furthermore, the ENSTA float is also equipped with different sensors and actuators :

- an Iridium [8], GNSS [9] and WIFI antenna so as to communicate with the float by different means
- an internal sensor returning the temperature, pressure and humidity inside the float for safety issues
- an external pressure sensor used for the depth regulation
- an external temperature sensor planned to collect data from the ocean environment
- a GPS (Global Positioning System) to know the position of the float at the surface
- an IMU (Inertial Measurement Unit) [10] to know the heading of the float for horizontal regulation
- an optical rotary encoder [11] and reed switches [12] to know the state of the piston in real time
- a motor controlling the piston for depth regulation
- propellers for the horizontal regulation
- circuit boards including a Raspberry Pi 3 [13] to process data

Finally, the float is powered by four LiPo rechargeable batteries [14], providing an autonomy of one day for the float in such a configuration. Different pictures of the ENSTA float can be found in the appendix (see "Pictures of the ENSTA float").

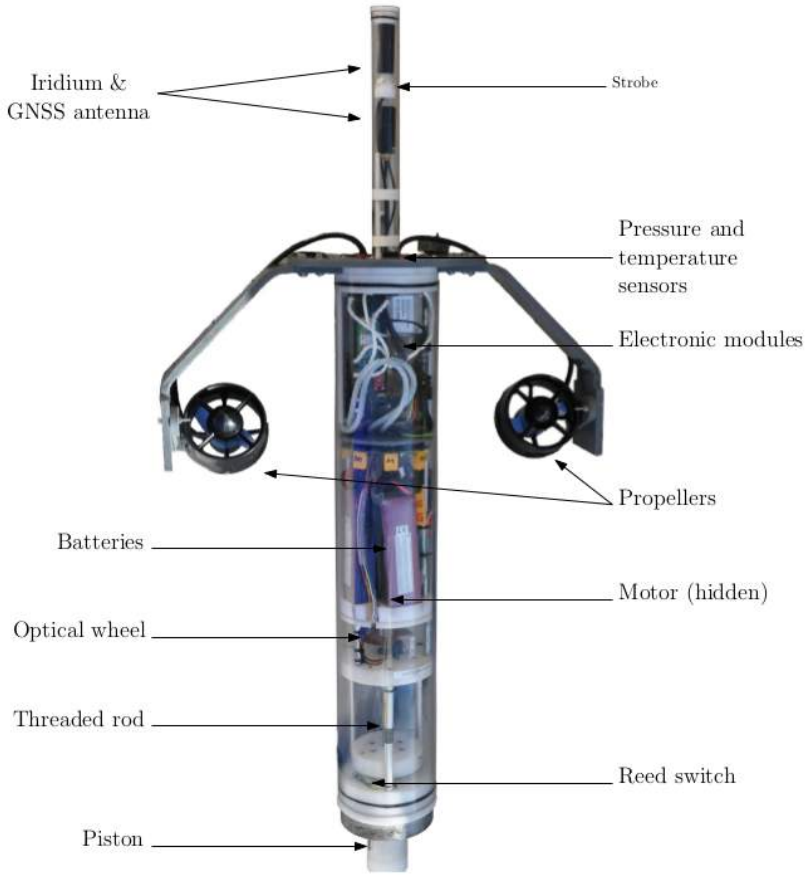


Figure 1.18: Physical architecture of the ENSTA float

1.3.2 The Ifremer float

As mentioned before, the Ifremer float will be the prototype on which the COGNAC project will be based. Although the float has been designed by following a physical architecture similar to the architecture of the ENSTA float, as the Ifremer float must fit different requirements, some key-points have been tackled differently during its design.

Indeed, first the Ifremer float is supposed to reach a maximum depth of 500 metres, that's why some sensors like the external pressure sensors and the hull made of aluminium have been chosen differently.

Choosing aluminium as the material for the caps the hull and the piston was also a way to make the float less compressible than water. The designers chose to build a float less compressible than water since they have considered that if the float is stable, it will consume less energy than an unstable float during the regulation phase as it will naturally keep its equilibrium position.

Moreover, the Ifremer float has only been dedicated to perform depth regulation. Consequently the float is not supposed to move horizontally by itself, thus it will not be equipped with an IMU or propellers.

As regards the energy part, the float is supposed to be energetically autonomous for about one month, consequently, 32 non-rechargeable alkaline batteries [15] will be used to power the float as this installation is more energetically dense and cheaper than using LiPo batteries. In addition, unlike the transportation of alkaline batteries, the transportation of LiPo batteries is more constraining for safety issues.

Finally, instead of using a simple piston for the auto-ballasting system, a double piston will be used. Indeed the piston will be divided in two cylinders characterised by different radius. The bigger part will be used near the surface so as to make the float quickly fall or resurface during the transition between air and water. The thinner part will be used during the regulation step when the float will be completely submerged to allow a better precision in the regulation. The float has been equipped with another reed switch so as to detect whether the larger part of the piston is retracted. Different pictures of the Ifremer float can be found in the appendix (see "Pictures of the Ifremer float").

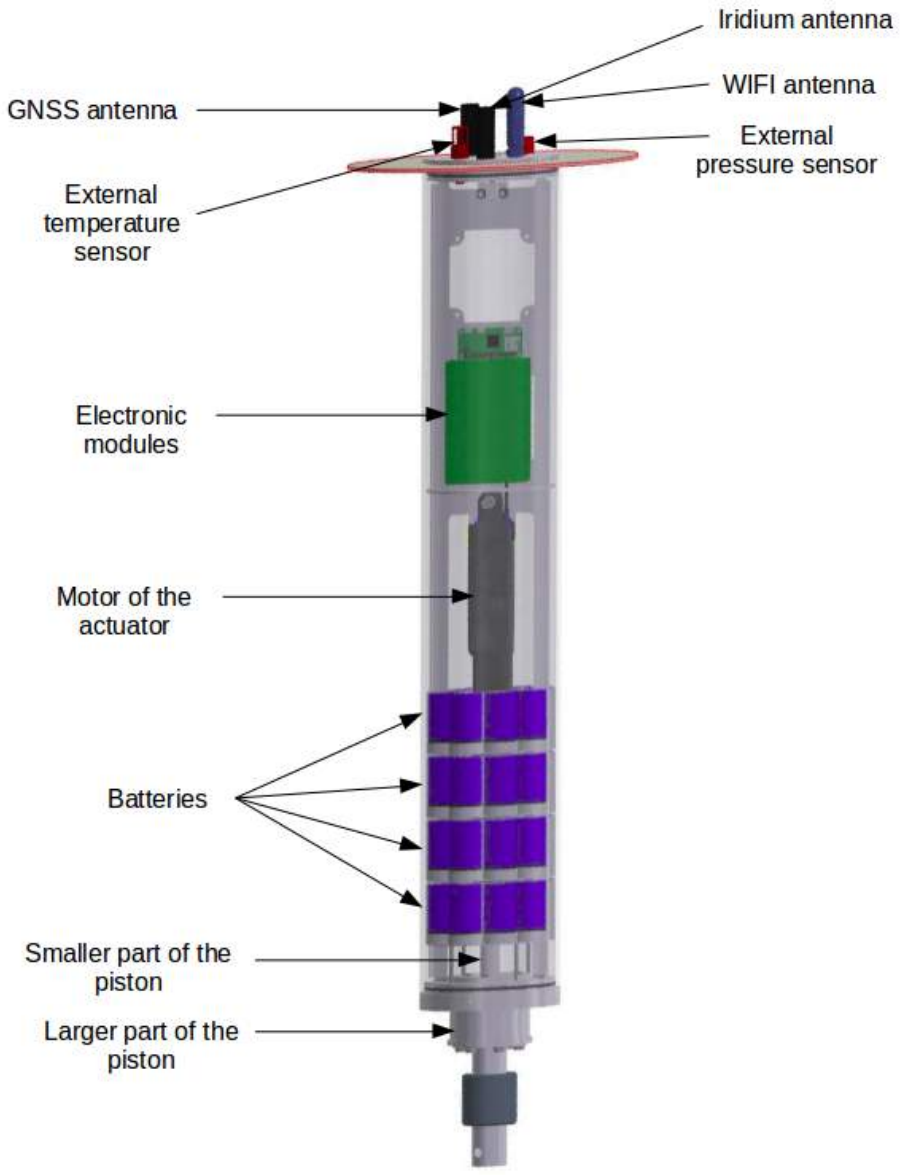


Figure 1.19: Physical architecture of the Ifremer float [7]

1.3.3 Recap of the differences between both floats

Overall, the most important difference to consider in this project between both floats is compressibility.

The ENSTA float is supposed to be more compressible than water whereas the Ifremer float is supposed to be less compressible than water. This different should have a great impact on the regulation phase. Indeed, thanks to this feature, unlike the ENSTA float, the Ifremer float is supposed to be stable, thus it should not tend to leave its equilibrium position by itself. This characteristic should greatly prevent the float from oscillating and consequently the float should consume less energy during the regulation process.

A second point to consider about the different compressibilities will be the limited depth. Indeed, in order to evolve vertically in the marine environment, the float is supposed to set its density by modifying its internal volume thanks to the piston, to make it lower or higher than the water density at its current depth. Nevertheless, the relative density between the float and water will also change according to the depth because of their compressibility. To understand this phenomenon, let's consider a simulation representing the possible densities of the ENSTA float and the water density as a function of the depth.

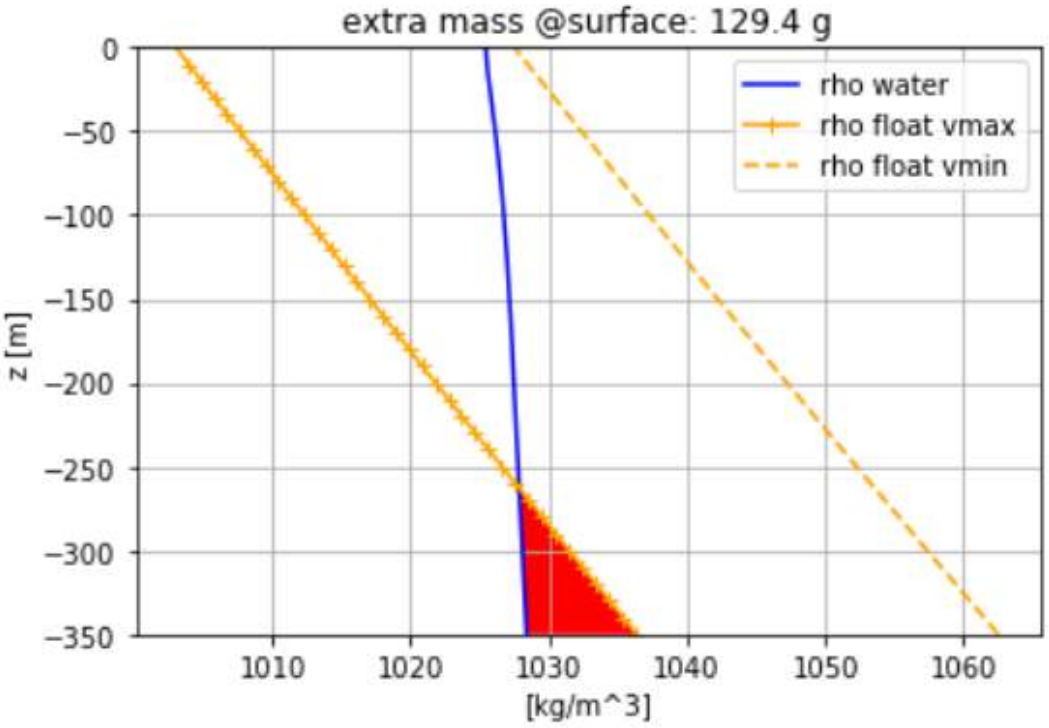


Figure 1.20: Simulation : compressibility of the well-ballasted ENSTA float as a function of the depth

As said previously, the float will be able to choose its depth in relation to water as long as it is able to change its density such as it is lower or higher than the water density at the corresponding depth, that is to say as long as the water density curve is located between the float maximum density and the float minimum density curves. In that case, as the ENSTA float

is more compressible than water, its density will increase far much faster than water density as the depth increases too, to the extent that it will not be able to set its density higher or lower than water density after reaching a certain depth which is around 260 m on this graph. In that case, the ENSTA float should not be able to resurface after reaching this depth. In the case of a less compressible float, this float would have faced the same issue yet with the opposite effect: after reaching a certain depth, such a float should not be able to sink more as the water will be denser than the float.

Now, if we consider the same simulation with the Ifremer float : as now the float is certainly less compressible than water, the water density curve evolves a little bit stronger than the float density curve but not so much since in that case, the float compressibility is almost similar to the water compressibility. Thus, according to this graph, the Ifremer float should be able to control its own depth beyond more than 5000 m.

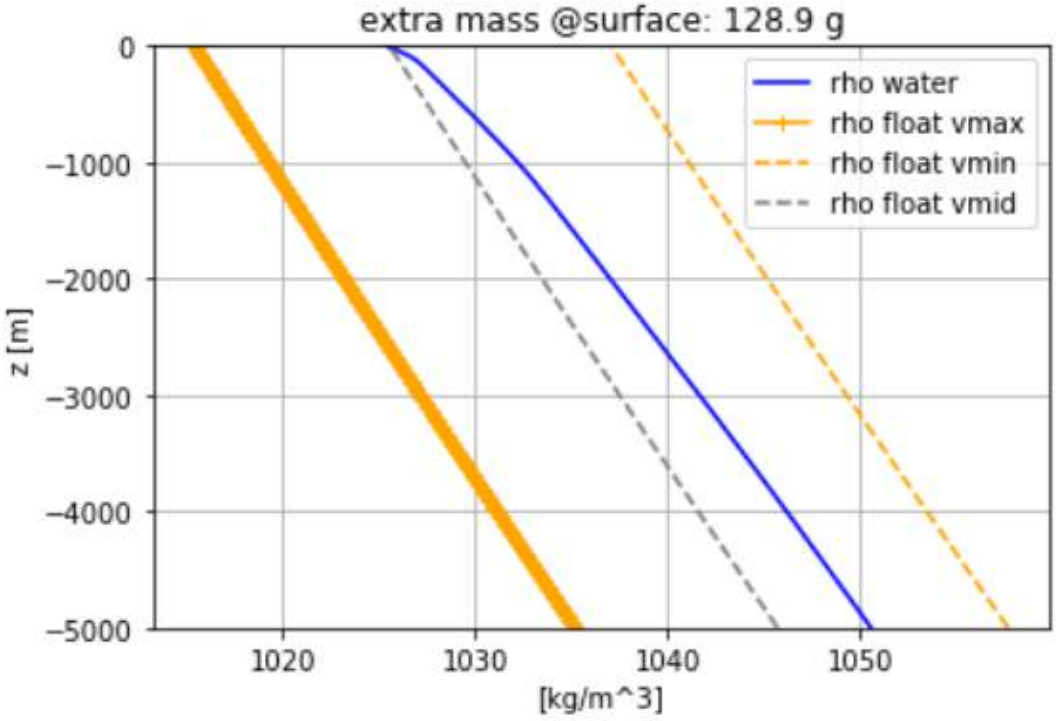


Figure 1.21: Simulation : compressibility of the well-ballasted Ifremer float as a function of the depth

Eventually, both floats are turned on or turned off once they are closed thanks to an internal switch that can be triggered by approaching a magnet. The differences between both floats are summarised in the recap chart below.

Kind of float Features	ENSTA float	Ifremer float
Mechanical features		
Compressibility	9.30e-5 $dbar^{-1}$ (more compressible than water 4.5e-06 $dbar^{-1}$)	3.78039e-06 $dbar^{-1}$ (less compressible than water 4.5e-06 $dbar^{-1}$)
Length of the tube	0.6 m	0.8278 m
Mass	9.045 kg	13.315 kg
Pressure advised inside the float to maintain the vacuum	About 650 mbar	About 650 mbar
Abilities		
Maximum depth allowed	50 m	500 m
Depth regulation	Yes	Yes
Horizontal regulation	Yes	No
Ability to follow isotherms	Not yet but possible	Not yet but possible
Material features		
Auto-ballasting system	Piston acting on the volume of air inside the float	Piston acting on the volume of air inside the float
Hull material	Polycarbonate	Aluminium
Caps material	Polyoxymethylene	Aluminium
Piston material	Polyoxymethylene	Aluminium
Piston structure	1 element	2 elements (larger and thinner part)
Electronic components		
Main circuit board	Raspberry Pi 3	Raspberry Pi 3
Iridium antenna	Yes	Yes
GNSS antenna	Yes	Yes
WIFI antenna	Yes	Yes
IMU sensor	Yes	No
GPS sensor	Yes	Yes
Internal sensor	BME 280 (pressure, temperature, humidity)	BME 280 (pressure, temperature, humidity)
External pressure sensor	89 bsd	Bar100 pressure sensor KellerLD
External temperature sensor	TSYS01	TSYS01
Piston state sensors	Optical rotary encoder + two reed switches	Hall effect sensor + two reed switches + a middle reed switch
Propellers	Yes	No
Energy		
Source of energy	4 LiPo rechargeable batteries (5 Ah 3S, hence 20 Ah total)	32 non-rechargeable alkaline batteries (1.5 V, 12-18Ah)
Hoped-for autonomy	1 day	1 month

Figure 1.22: Recap chart of the differences between ENSTA and Ifremer float

Chapter 2

Main contributions about algorithmic and software development

In this part, I will mainly present my work regarding regulation. First, I will present the different regulation algorithms which have been considered in the algorithmic development of the float and I will focus on the state feedback regulation which has been chosen. Then, I will describe in detail the software part of the float. This chapter has represented the largest part of my work during this internship.

2.1 Regulation part of the low-cost drifting float

The regulation part of the float forms the transition interface between the sensor part and the actuator part of the system. This interface is actually at the heart of the system and allows to control the state of the piston as a function of the data returned by the "observation" sensors of the system like the external pressure sensor if the regulation considered is a depth regulation. In short, regulation confers its autonomy on the float.

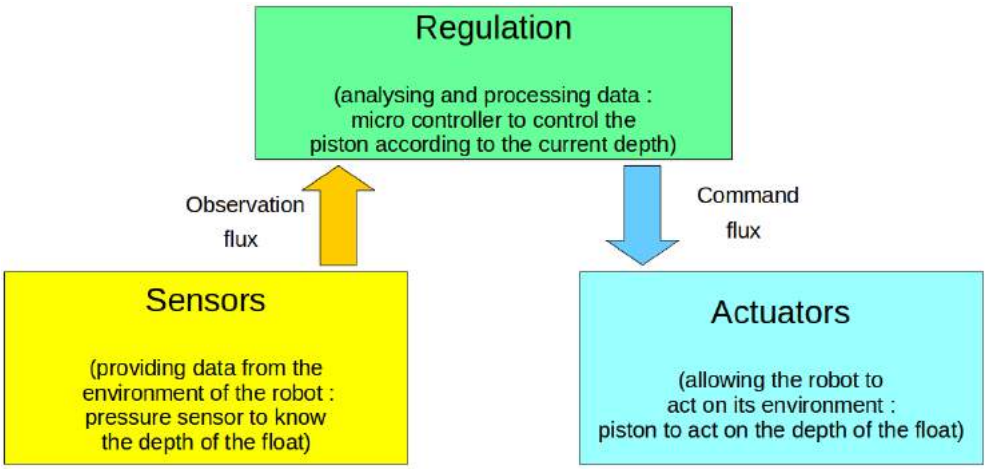


Figure 2.1: Regulation diagram

2.1.1 Different kinds of regulation conceivable to make the float autonomous

Several regulation patterns are conceivable, according to the targeted aims. The simplest models such as the sliding mode regulation or the proportional–integral–derivative regulation (PID controller) are regulation models whose implementation is very easy as they do not need a knowledge of the environment in which the system evolves, indeed their functioning is only based on a consideration of the error between the measure and the corresponding targeted value.

Overview of the sliding mode regulation

The sliding mode regulation [16] is one of the simplest regulation law. It consists in regulating the robot with a two-state command, depending on the sign of the error between the measure and the targeted value.

For instance, considering a depth regulation for a float where $e = z - z_{target}$ is the difference between the depth measured thanks to the external pressure sensor z (here it will be chosen to be always positive with 0 m at the surface) and the depth wanted z_{target}, u_{max} the command ordered to retract the piston entirely inside the float and u_{min} the command ordered to release the piston entirely out of the float, the regulation law will be :

$$\begin{cases} u = u_{max} \text{ if } z < z_{target} \text{ ie } e < 0 \text{ ie the float needs to sink} \\ u = u_{min} \text{ if } z > z_{target} \text{ ie } e > 0 \text{ ie the float needs to rise} \end{cases}$$

A simulation has been performed below with the same simulator about which I will give more details further, in another section. In this simulation, the float was supposed to regulate itself at 25 m through a sliding mode controller. As it can be seen, this controller generates many oscillations. These oscillations are due to the two opposite command thresholds used in the regulation. For instance, if the float is too deep, the piston will be completely released from the float so as to reach the target but as the command is strong, the float will easily exceed the targeted depth and will have to regulate again so as to come back to the targeted depth before exceeding it again and etc. Even if this regulation model seems to be very simple, it is clearly not adapted to this system since the oscillations are very wide as the effects from the piston are very slow before being considered. Moreover, by regulating as much, the float will consume much more energy while energy autonomy is a crucial criterion in this project.

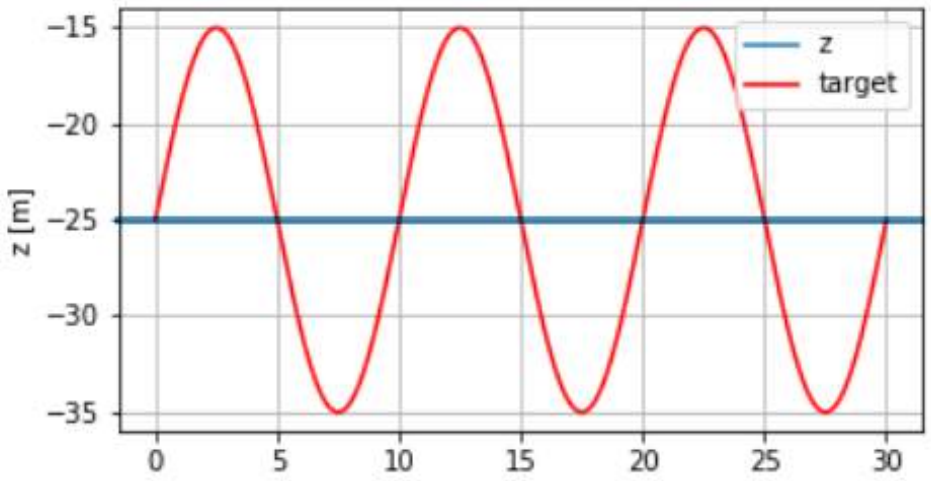


Figure 2.2: Sliding mode simulation : depth of the float as a function of time (in minutes)

Overview of the proportional–integral–derivative regulation

Like in the sliding mode regulation, the proportional–integral–derivative regulation consists in constructing a command from the error between a measure and a targeted value. The difference is that in the PID regulation, the command is calculated thanks to different forms of the error : the command is the sum of the error, its derivative and primitive, multiplied by different coefficients to set [17].

Theoretically, the equation of the command is the following with t the time, u the command, e the error for example e can be $e = z - z_{target}$ like in the previous regulation for a depth regulation, K_p, K_i and K_d the coefficients to choose :

$$\left\{ u(t) = K_p * e(t) + K_i * \int_0^t e(t)\,\mathrm{d}t + K_d * \frac{de(t)}{dt} \right.$$

The main issue with this kind of regulation is the choice of the coefficients [18] K_p, K_i and K_d. This coefficients have much influence on the resulting command, indeed overall the K_p coefficient allows to set the response time of the system that is to say the necessary time for the system to reach its target but a too strong coefficient can produce oscillations which are supposed to be avoided according to the previous subsection.
The K_i coefficient has an impact on the static error of the system, that is to say the difference between the actual and the wanted value after a long regulation time. Similarly to the previous coefficient, an inadequate choice can generate oscillations.
Finally, the K_d coefficient allows to act on the stability of the system that is to say, to control the oscillations. It has also an effect on the overshoot and on the settling time (necessary time for the system to have a response near the targeted value with a certain margin of error).

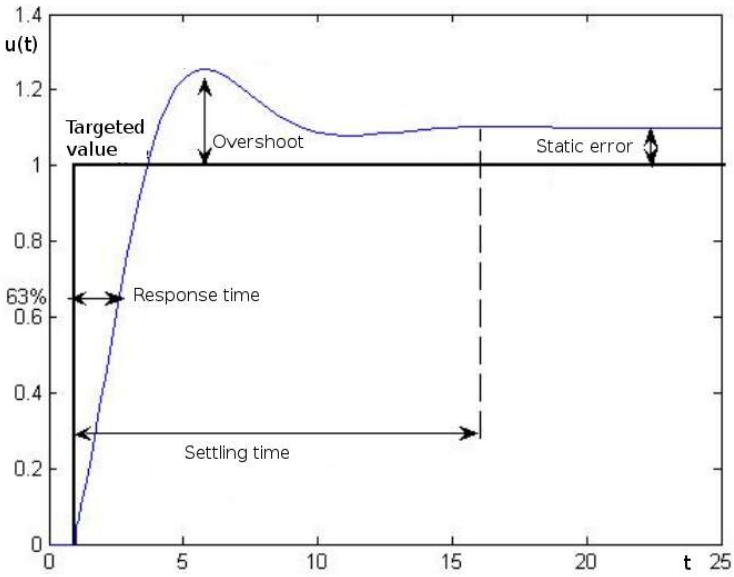


Figure 2.3: PID graph explanations: command as a function of time [19]

A simulation has been performed below, still with the same simulator as used previously. In this simulation, the float is still supposed to regulate itself at 25 m but now through a PID controller. As it can be seen, this controller still begets many oscillations however they are now less wide than for the sliding mode regulation. To perform this simulation, the coefficients K_p, K_i and K_d have been chosen one after another so as to obtain a fast response with the least oscillations.

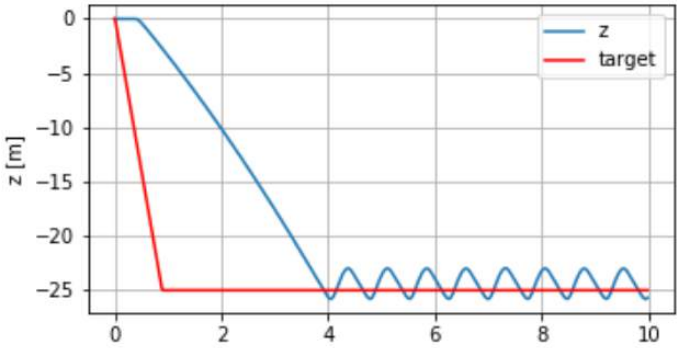


Figure 2.4: PID simulation: depth of the float as a function of time (in minutes)

Consequently, this regulation pattern is a little more suitable for the project than the sliding mode regulation, nevertheless it still raises a tricky issue since the choice of the coefficient can be very complex and unpredictable according to the environment in which the float is supposed to progress. Indeed, it can be easily accepted that appropriated coefficients can be chosen for a given situation but as the effect of this choice is very sensitive on the resulting behaviour of the float, a simple modification of the current environment about temperature, pressure, density or swell could easily jeopardise the efficiency of the PID regulation. Consequently, this regulation model will not be considered anymore to develop this project.

It has been demonstrated that simple regulation patterns were not adapted to such a system. Besides, any regulation pattern will not be strong enough to correctly regulate this system if it is unaware of the physics of the surrounding environment. Indeed, the regulation algorithms are not efficient enough as the system is supposed to progress in evolving environments like the marine environment.

Indeed, the marine environment is a non-homogeneous environment, pressure-stratified, temperature-stratified in particular and consequently density stratified. The behaviour of a system ruled by one of the regulation algorithms mentioned above will be different according to the depth or the area in which the float will stand.

Realising the impact of the environment on the regulation, the equations of the float dynamics must be taken into account like the state feedback regulation suggests it.

Overview of the state feedback regulation

The state feedback regulation is a little bit like the PID regulation but in much more general [20]. Indeed, instead of only taking into account an error and the different forms of this error, the state feedback regulation also considers other state variables directly related to the impact of this environment on the robot. Especially, it takes into account the volume and mechanical compressibility variations of the float, owing to its current depth. In addition, in this specific case, a Kalman filter [21] will be used in order to estimate the uncertainty of the state variables of the system related to the dynamical model and to the sensors like the external pressure sensor. The Kalman filter will be also employed here to estimate some unknown or wrongly-known variables such as the mechanical compressibility of the float.
The following subsection will describe the implementation of this more complex regulation pattern from the dynamical equations considered.

2.1.2 Implementation of the state feedback regulation

The implementation of the state feedback controller will be based on the studies of the corresponding dynamical equations. The demonstration made by leaning on several hypotheses can be found in the appendix ("Command calculation for the state feedback regulation").

Before calculating the command in this demonstration, the system had to be represented by a state vector. After discussing the subject with Thomas LE MEZO and Aurélien PONTE, it has been decided that the system state vector will be given by:

$$\underline{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -dz_f/dt \\ -z_f \\ \gamma_e \\ V_e \end{pmatrix}$$

where the state variables are therefore the depth temporal rate of evolution (x_0, in m/s), the depth (x_1, in m), the equivalent compressibility of the float (γ_e, in m^2) and the error volume (x_3, in m^3).

Considering this state vector, the demonstration finally leads to the following command equation :

$$u = \frac{1}{A}\Big[\lambda_1 \dot{y} + \lambda_2 y + \frac{\nu}{\delta}\frac{\dot{x}_0 D + 2e x_0^2/\delta^2}{D^2} - B\frac{d}{dt}\Big(|x_0|x_0\Big)\Big] + x_2 x_0. \tag{2.1}$$

where $A = g\rho/2m$, $B = c_1/4L$, $e = \overline{x_1} - x_1$, $D = 1 + e^2/\delta^2$, $y = x_0 - \nu \arctan \frac{\overline{x}_1 - x_1}{\delta}$, $\dot{y} = \dot{x}_0 + \nu\frac{x_0/\delta}{D}$ and where ν, τ and δ are regulation parameters.

Now, let's come back to the state vector which has been considered in the command law. Among the different state variables considered, the depth temporal rate of evolution (x_0, in m/s) and the depth (x_1, in m) will be obtained thanks to the external pressure sensor of the float and a time reference, nonetheless, the equivalent compressibility (γ_e, in m^2) and the error volume (x_3, in m^3) will still keep unknown. Indeed, the equivalent compressibility of the float is supposed to change according to the depth and cannot be measured. Similarly, the error volume V_e corresponds to the offset volume such that the float is at its equilibrium position at a depth of 0 m. This volume also depends on the depth of the float and on other variables like the compressibility. Consequently, the equivalent compressibility and the error volume will have to be estimated so as to be considered in the command law, a Kalman filter will be used with this aim in mind.

State estimation via a Kalman filter

A Kalman filter is an algorithm that can use measurements observed over time to estimate unknown variables or to estimate wrongly-known variables or affected by statistical noise, with more accuracy by estimating a joint probability distribution over the variables for each time frame [21].

A general formulation of the Kalman filter can be found in the appendix (see "General formulation of the Kalman filter"), this formulation contains two interesting matrices that will be presented in the following subsection through simulations.

2.1.3 The significance of the parameters in the state feedback regulation through simulations

By referring to the command equation, three regulation parameters had been defined before so as to compute the command. In this subsection, I will explain the relevance of these three parameters : an inverse time scale r, a velocity ν, and a length scale δ.

Thanks to a simplified simulation, r will be deduced so as to compute λ_1 and λ_2 which are relaxation parameters. Similarly, the characteristic length of the regulation δ and the characteristic velocity ν will be chosen according to the behaviour we want to give to the float.

The choice of the most suitable regulation parameters that will be presented in the simulation will rely on two functions defined thanks to a basic dynamical model. The computation of these functions can be found in the appendix (see "Computation of the functions to choose the most suitable regulation parameters"). In order to assess the appropriate time and distance parameters $1/r$ and δ, it will be useful to create a function t_ν that computes the necessary time for the float to reach a given speed ν and another function z_ν computing the necessary depth for the float to reach the same given speed ν.

These functions are :

$$t_\nu(\nu) = \sqrt{\frac{2m(1+a)\nu}{\rho_w u\ g}} \tag{2.2}$$

$$z_\nu(\nu) = z_f(t_\nu(\nu)) = \frac{\rho_w u\ g(\sqrt{\frac{2m(1+a)\nu}{\rho_w u\ g}})^3}{6m(1+a)} \tag{2.3}$$

where g is the acceleration of gravity, ν is the downward velocity, m the float mass, a its added mass, ρ_w the water density, u the piston volume rate of change (the piston is supposed to leave the cylinder of the float with a constant velocity u).

The following subsection will deal with the influence of each parameter on the regulation. To assess the behaviour of the float according to the parameters, I leant on a simulator made by Aurélien PONTE. I modified this simulator by implementing a state feedback controller with a Kalman filter and graphic tools so as to perform simulations and display the corresponding results while playing on the different regulation parameters. This simulator also permitted to compare the behaviours of the ENSTA and the Ifremer floats.

This simulator has been implemented with the programming language Python, such as each float is represented by an Object-class with all its characteristics according to the Object-oriented programming [22]. This simulator also takes into account some water properties like water density, compressibility variations due to depth, and drag. This simulator assumes that the piston of the float can move through an infinite number of positions in its displacement area and considers the float as a simple tube without any irregularity in its structure.
This simulator and the different tools used around this simulator can be found at these addresses :

https://github.com/houdeval/cognac_regulation
https://github.com/houdeval/cognac

Choice of ν

Let's choose first the velocity parameter ν that corresponds to the typical vertical velocity the float should move at to perform a correct regulation. This parameter will be used to limit the velocity of the float during the regulation time.

Reminding that (see "Command calculation for the state feedback regulation" in the appendix):

$$y = x_0 - \nu \arctan \frac{\overline{x}_1 - x_1}{\delta}, \tag{2.4}$$

The choice of y as law of the type $y \sim e^{-rt}$ is such that when $t \to \infty$:
$x_0 = \nu \arctan \frac{\overline{x}_1 - x_1}{\delta}$.

This means that the speed of the float : $x_0 \to 0$ when $|x_1 - \overline{x}_1| < \delta$. Moreover, the float vertical velocity should also never be greater that $\pm\nu \times \pi/2$: the asymptote of the curve x_0 as a function of $\overline{x}_1 - x_1$ as it can be seen on the graph below. This is the condition to choose the parameter ν.

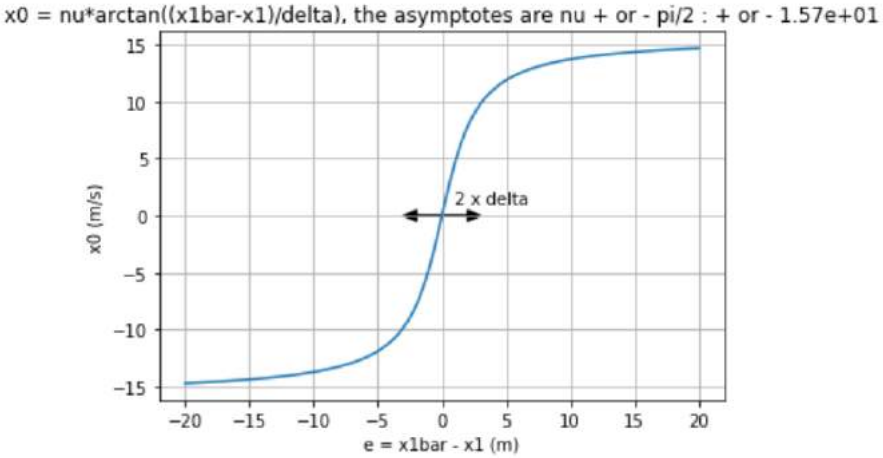


Figure 2.5: Graph of the function chosen to measure the depth error for the feedback regulation

In short: $\nu * \pi/2$ must correspond to the maximum velocity that can be reached by the float during the regulation time.

The choice of this velocity parameter is limited by two phenomena :

- On one hand because of drag, the float cannot move above a certain speed because of its geometry. This maximum velocity was roughly estimated at 40 cm/s for the Ifremer float and 44 cm/s for the ENSTA float. You can find the corresponding calculation at the end of the part "Command calculation for the state feedback regulation" in the appendix.
- On the other hand, a too high speed can beget oscillations when arriving around targeted depths.

Thus overall, $\nu * \pi/2$ must correspond to the lower speed among both of these.

The influence of ν is shown in the simulation graph below, representing the depth and velocity responses of the ENSTA float, regulating at 25 m after leaving an equilibrium position at 20 m, for different choices of $v_{max} = \nu * \pi/2$, assuming the state variables are perfectly

known, that is to say without using Kalman filter. For τ and δ fixed, it can be noted that the more important v_{max} is, the faster the float will reach the depth command. However, after a certain value of v_{max}, the float will face a depth overshoot. The value $v_{max} = 0.04$ m/s has been chosen as a reference value for the other simulations as it generates appropriate results and it is ten times smaller than the maximum velocity estimated above.

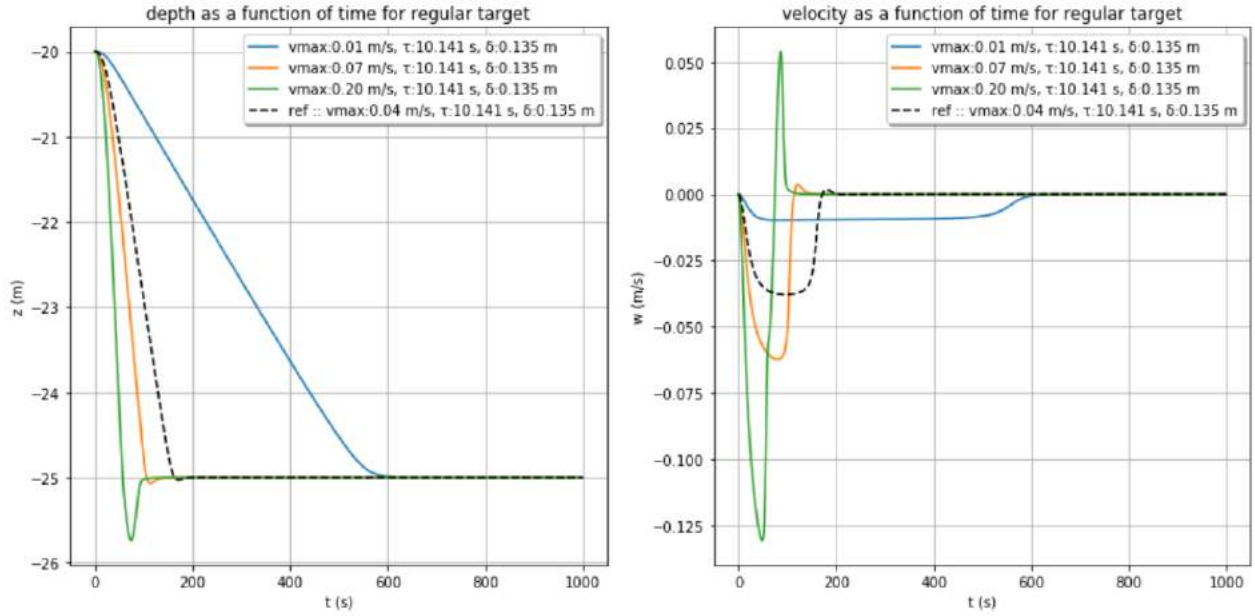


Figure 2.6: Simulations for the ENSTA float with v_{max} variable

Estimation of δ

δ is a length scale that defines the zone of influence of the regulation around the target depth.

That mathematically corresponds to the previous condition :

$x_0 \to 0$ when $|x_1 - \overline{x}_1| < \delta$

In short, δ corresponds to the depth traveled by the float before reaching the maximum speed $\nu * \pi/2$ from an equilibrium. Therefore, to estimate the suitable δ, we can use the previous formula which has been defined previously :

$$\delta = z_\nu(\nu\pi/2) = z_f(t_\nu(\nu\pi/2)) = \frac{\rho_w u\ g(\sqrt{\frac{2m(1+a)*\nu\pi/2}{\rho_w u\ g}})^3}{6m(1+a)} \tag{2.5}$$

The influence of δ is highlighted on the simulation graph below, representing the depth and velocity responses of the ENSTA float, regulating at 25 m after leaving an equilibrium position at 20 m, for different choices of δ, assuming the state variables are perfectly known, that is to say without using Kalman filter. For τ and v_{max} fixed, it can be noted that the lower δ is, the faster the float will reach the depth command. However, when δ is too low (0.001

m on the graph), the float will oscillate to succeed in its regulation. The reference value of $\delta = z_\nu(\nu\pi/2) = z_\nu(0.04) = 0.14$ m has been computed thanks to the function established above, in accordance with the explanations given above. This value has been kept as a reference value for the other simulations as it generates appropriate results.

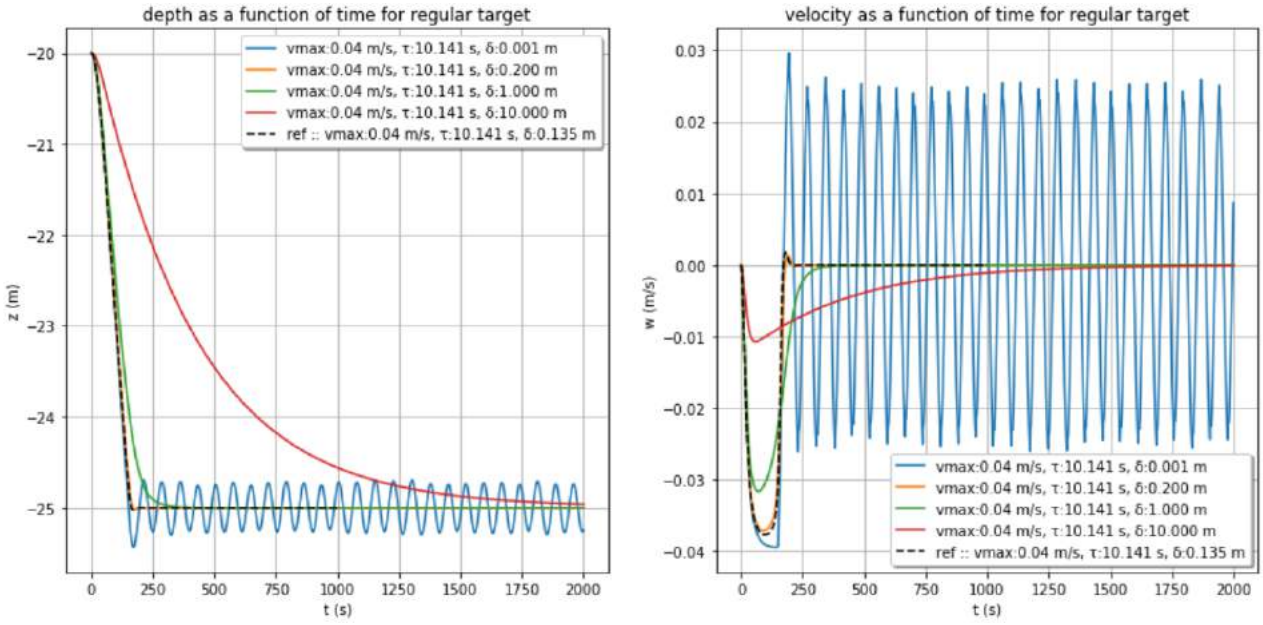


Figure 2.7: Simulations for the ENSTA float with δ variable

Estimation of r

The control time scale $1/r$ which will be called τ from now on, corresponds to the response time we want for the float ruled by the state feedback regulation.

To be suitable, τ needs to be equal or larger than the time taken for the float to reach the maximum speed $\nu * \pi/2$ from an equilibrium.

Thus, a suitable ratio $1/r$ is given by :

$$t_\nu(\nu\pi/2) = \sqrt{\frac{2m(1+a) * \nu\pi/2}{\rho_w u\ g}} \tag{2.6}$$

$$\tag{2.7}$$

The impact of $\tau = 1/r$ is displayed by the simulation graph below, representing the depth and velocity responses of the ENSTA float, regulating at 25 m after leaving an equilibrium position at 20 m, for different choices of τ, assuming the state variables are perfectly known, that is to say without using Kalman filter. For δ and v_{max} fixed, it can be noted that the lower τ is, the faster the float will reach the depth command. However, when τ is too small, the float will have to oscillate to succeed in its regulation. The reference value of $\tau = t_\nu(\nu\pi/2) = t_\nu(0.04) = 10.14$ s has been computed thanks to the function established above,

in accordance with the explanations given above. This value has been kept as a reference value for the other simulations as it generates appropriate results.

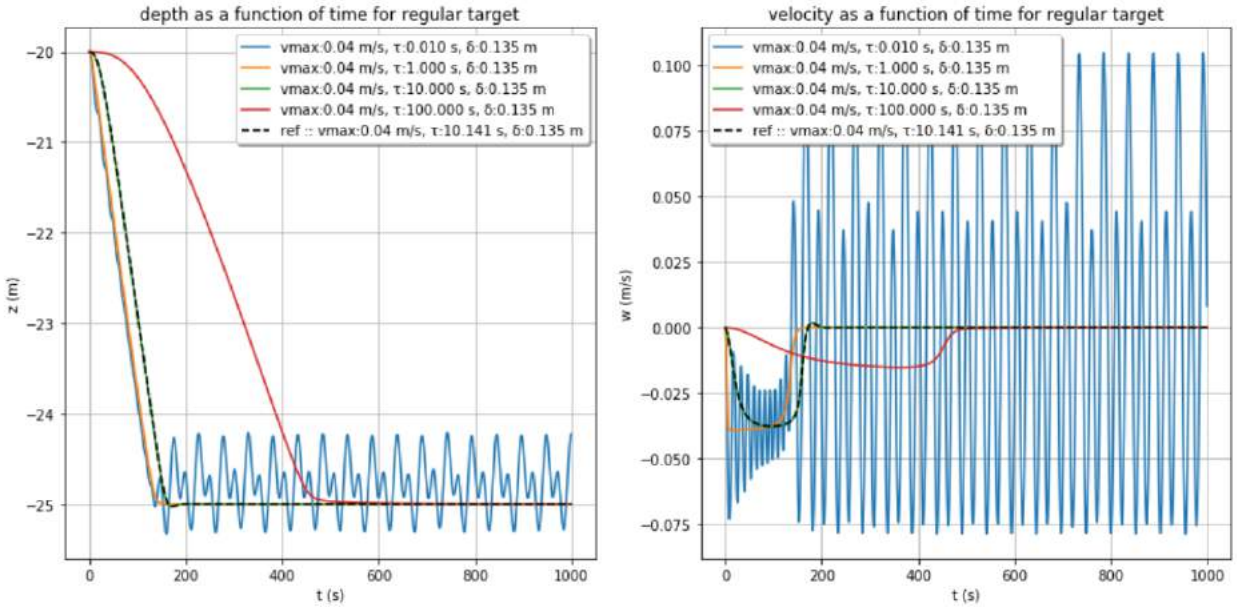


Figure 2.8: Simulations for the ENSTA float with τ variable

Guiding values of the parameters for the state feedback regulation

The simulations above have allowed to point out the influence of the three regulation parameters on the resulting behaviour of the float during the regulation. The simulations have also permitted to approve the different expressions established so as to try to calculate ideal values of δ and τ as a function of the chosen value ν. Consequently, combining all these guiding values together should lead to satisfactory simulations for both floats as it can be seen on the reference curves below that correspond to simulations performed with the guiding values of the parameters for each float considered, regulating at 25 m after leaving an equilibrium position at 20 m, assuming the state variables are perfectly known, that is to say without using Kalman filter. Besides, it can be noted that the reference values calculated thanks to the theory which has been considered before, are similar between both floats, which means both floats share almost the same reactivity during the regulation phase, which was not obvious. Indeed, the command variable is the piston volume rate of change which corresponds to the variation of volume produced by the piston per time unit (m^3/s). The ability to follow this command depends on the speed of the motor and on the section of the piston. Concerning the Ifremer float, the section is two and a half times smaller than the section of the piston of the ENSTA float, however this is compensated by a stronger motor speed, hence a same ability for both floats to follow such a command.

The two following curves have been plotted to show that the regulation parameters can be still adjusted so as to reduce again the response time of the float. It can be noted that playing on the δ parameter is not very efficient to reduce that time and even produces oscillations while reducing the τ parameter still greatly reduces the response time of the float.

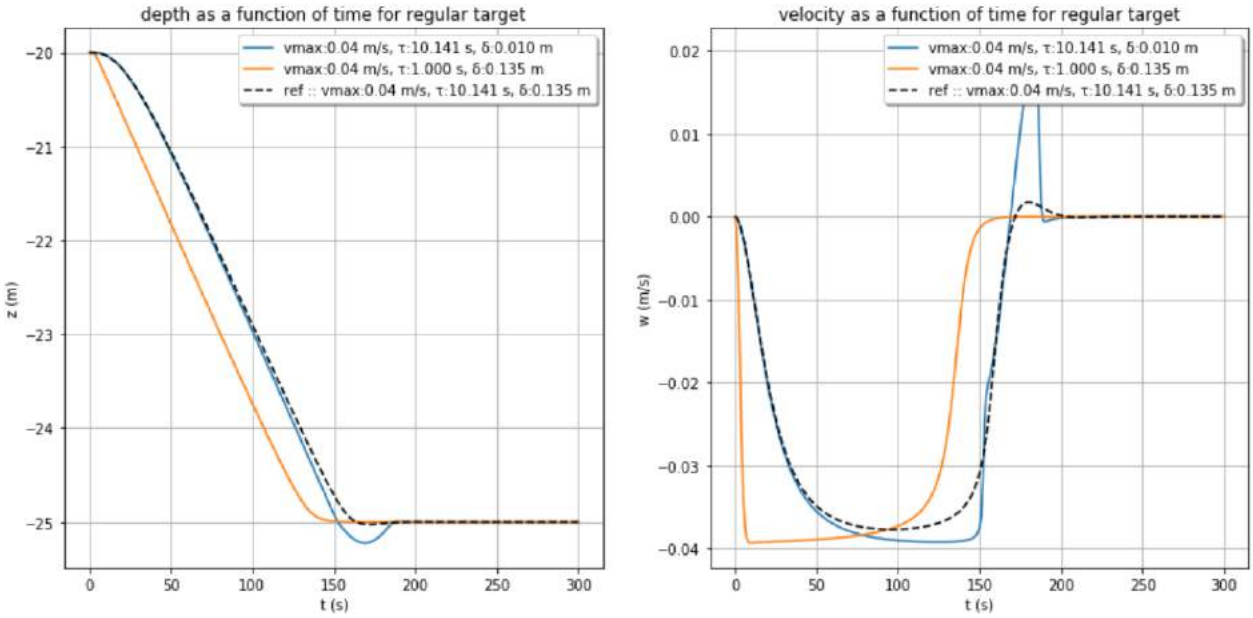


Figure 2.9: Simulations for the ENSTA float with guiding regulation parameters

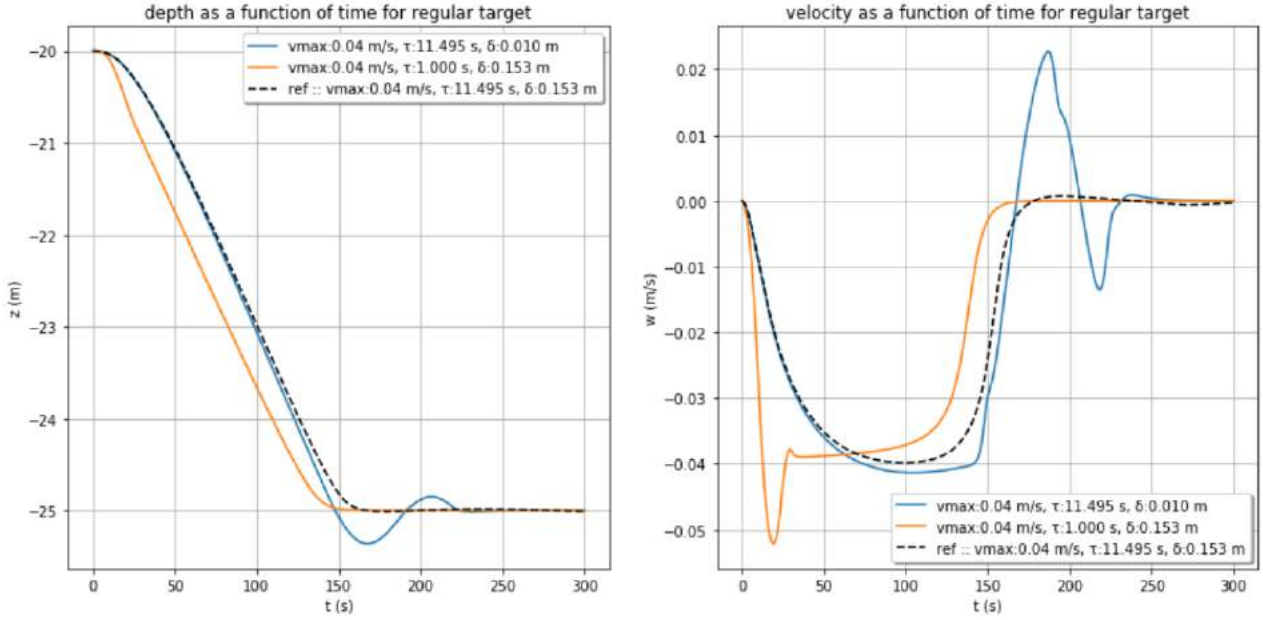


Figure 2.10: Simulations for the Ifremer float with guiding regulation parameters

Parameters of the Kalman filter

The Kalman filter parameters have been adjusted to compute Γ_β and Γ_{α_k}, which are respectively the observation noise matrix and the model error matrix of the float (see "General formulation of the Kalman filter" in the appendix). The influence of these parameters has been tested with the simulator. In the following simulations, the float will try to regulate itself at 25 m after leaving an equilibrium position at 20 m through the state feedback regulation with the guiding values respectively computed above for each float. The time step parameter dt which is the

period between each application of the Kalman filter will be set to 1 s. The estimated variables will be compared to the real corresponding values provided by the simulator.

$$\mathbf{\Gamma}_\beta = \begin{bmatrix} e_z^2 \end{bmatrix} \tag{2.8}$$

e_z represents the uncertainty on the depth related to the external pressure sensor. It will be chosen in accordance with the accuracy on the depth returned by the external pressure sensor, thus changing this value will also modify the strength of the noise produced by the sensor in the simulator.

$$\mathbf{\Gamma}_{\alpha_k} = dt^2 \times \begin{bmatrix} e_{acceleration}^2 & 0 & 0 & 0 \\ 0 & e_{velocity}^2 & 0 & 0 \\ 0 & 0 & e_{\gamma_e}^2 & 0 \\ 0 & 0 & 0 & e_{V_e}^2 \end{bmatrix}, \tag{2.9}$$

$e_{velocity}$ is the velocity error term. It will be computed according to the corresponding accuracy obtained after deriving the depth given by the external pressure sensor : $e_{velocity} = \frac{e_z}{dt}$

e_{V_e} is the volume error term. A possible definition for this term would be an order of magnitude of the smallest possible variation of the volume produced by the piston inside the float. For both floats, this parameter has been set to $10^{-7}m^3$.

$e_{acceleration}$ is the acceleration error term. According to the state equation of the float (4.16) that can be find in the appendix (see "Command calculation for the state feedback regulation"), a suitable definition for this term would be :
$e_{acceleration} = Ae_{V_e}$ where $A = g\rho/2m$.

e_{γ_e} represents the uncertainty on the equivalent compressibility of the float :
$\gamma_e = V_0\partial_z\rho_w/\rho + \gamma_V$. The term γ_e is described at the end of the equation (4.15) in the appendix (see "Command calculation for the state feedback regulation"). Basically, it corresponds to the sum of a term related to water density changes as a function of the depth and a prevailing term corresponding to the product of the compressibility and the whole volume of the float. In the following simulations, the estimated equivalent compressibility will be only compared with this prevailing term as the simulator does not provide the real value of the first term, which can explain the gap that will be noted between both curves. e_{γ_e} will be set to $10^{-8}m^2$ for both floats, which corresponds to the amplitude of γ_e predicted in the simulations divided by 10.

After having computed the matrices with the parameters described above, the simulator produces the following graphs assuming that the external pressure sensor accuracy is 1 cm :

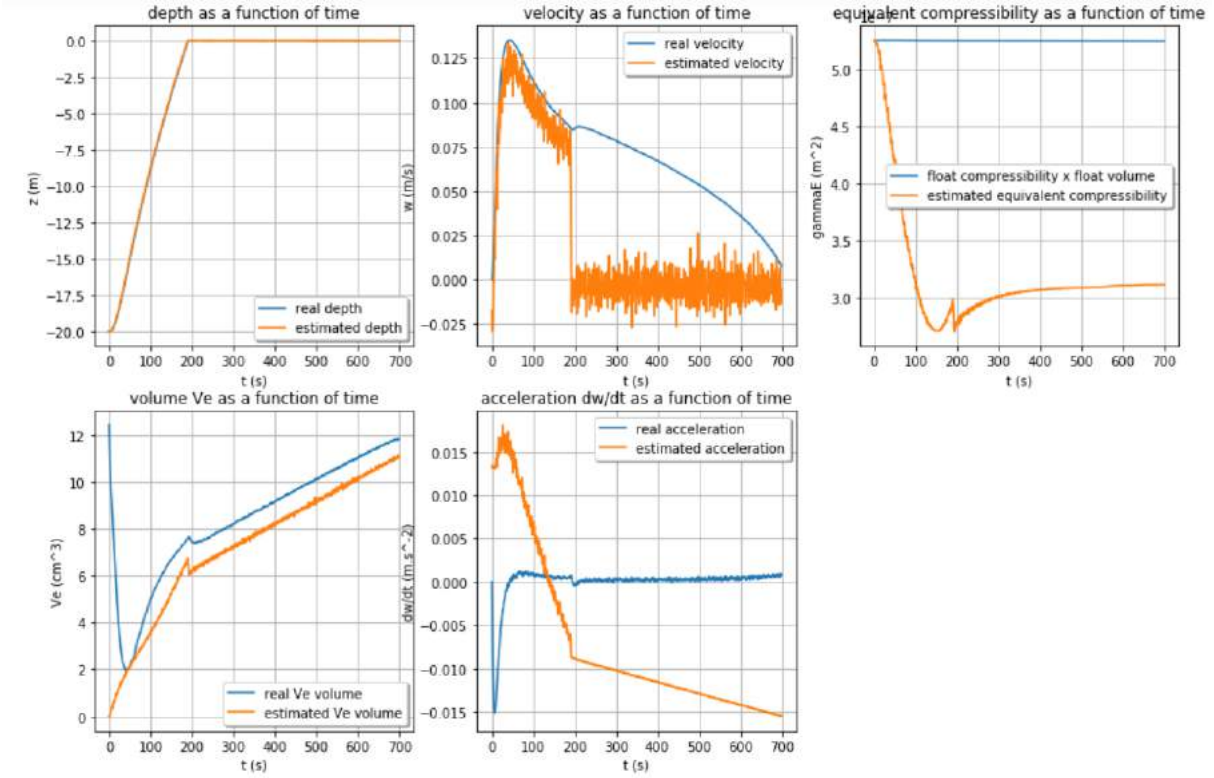


Figure 2.11: Simulation for the ENSTA float with Kalman filter : $e_z = 10^{-2}m$; $e_{V_e} = 10^{-7}m^3$; $e_{\gamma_e} = 10^{-8}m^2$

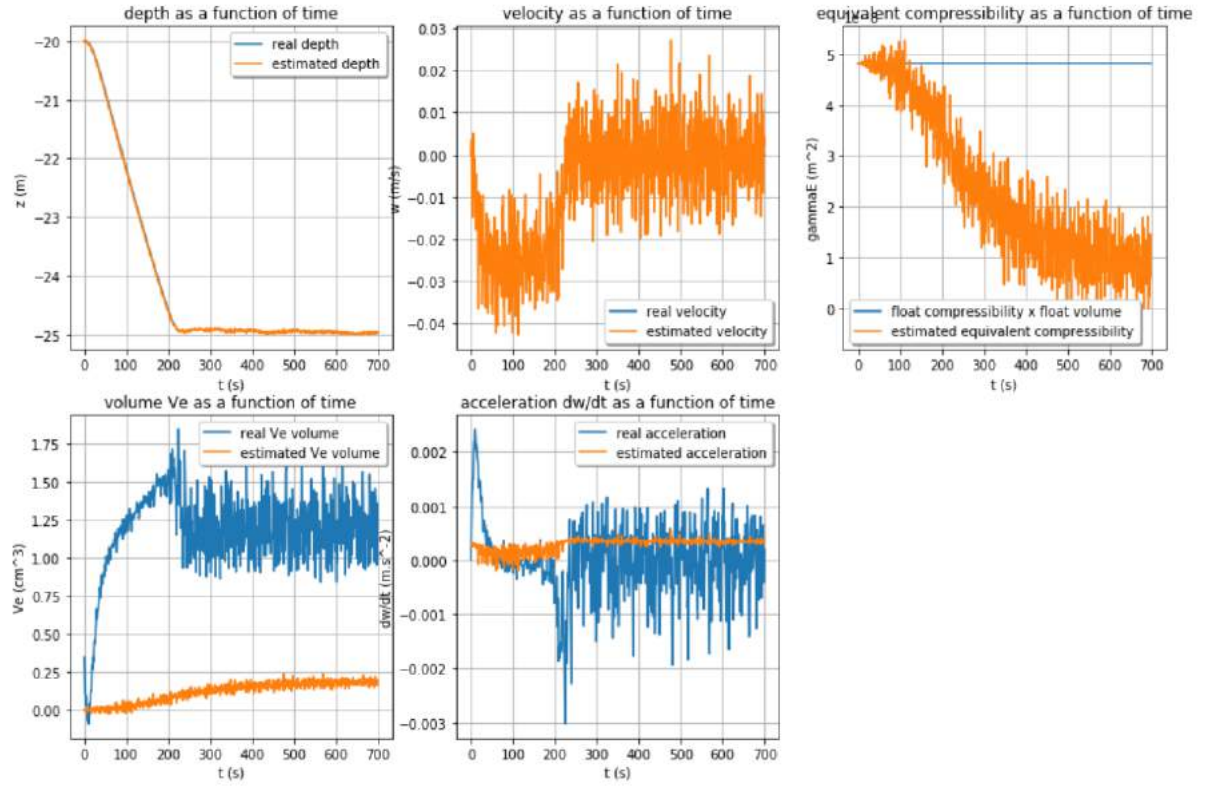


Figure 2.12: Simulation for the Ifremer float with Kalman filter : $e_z = 10^{-2}m$; $e_{V_e} = 10^{-7}m^3$; $e_{\gamma_e} = 10^{-8}m^2$

These graphs show that with such parameters, the Ifremer float manages to regulate itself with an accurate estimation of its depth and its velocity. Nevertheless, its acceleration and the error volume V_e are not correctly estimated. As regards the equivalent compressibility, as mentioned before, the real value compared is not exactly the whole equivalent compressibility, however it seems to be consistent with the estimated values provided by the Kalman filter. Concerning the ENSTA float, even if the volume error V_e is rather well-estimated, the other variables and especially the velocity are not correctly estimated, hence the failure in the regulation of the ENSTA float.

Now let's consider that the external pressure sensor is more accurate with an accuracy of 1 mm. The simulator predicts the following results :

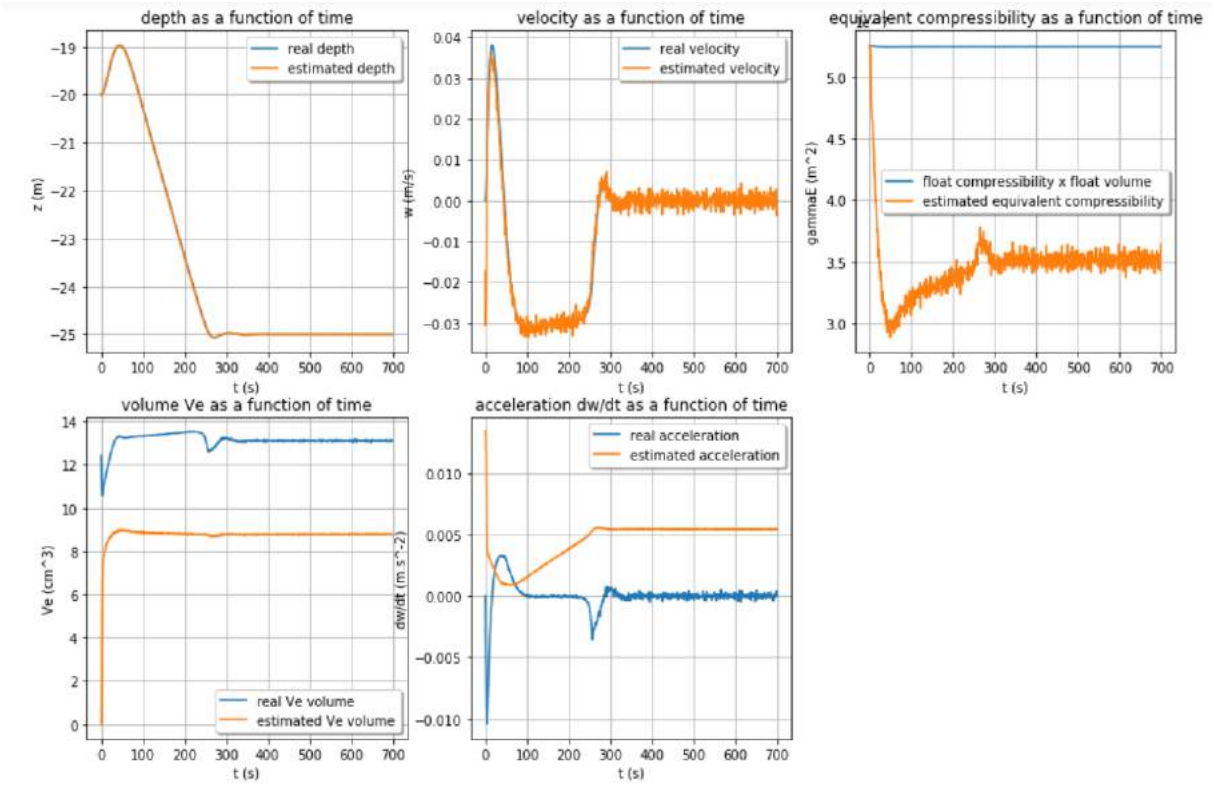


Figure 2.13: Simulation for the ENSTA float with Kalman filter : $e_z = 10^{-3}m$; $e_{V_e} = 10^{-7}m^3$; $e_{\gamma_e} = 10^{-8}m^2$

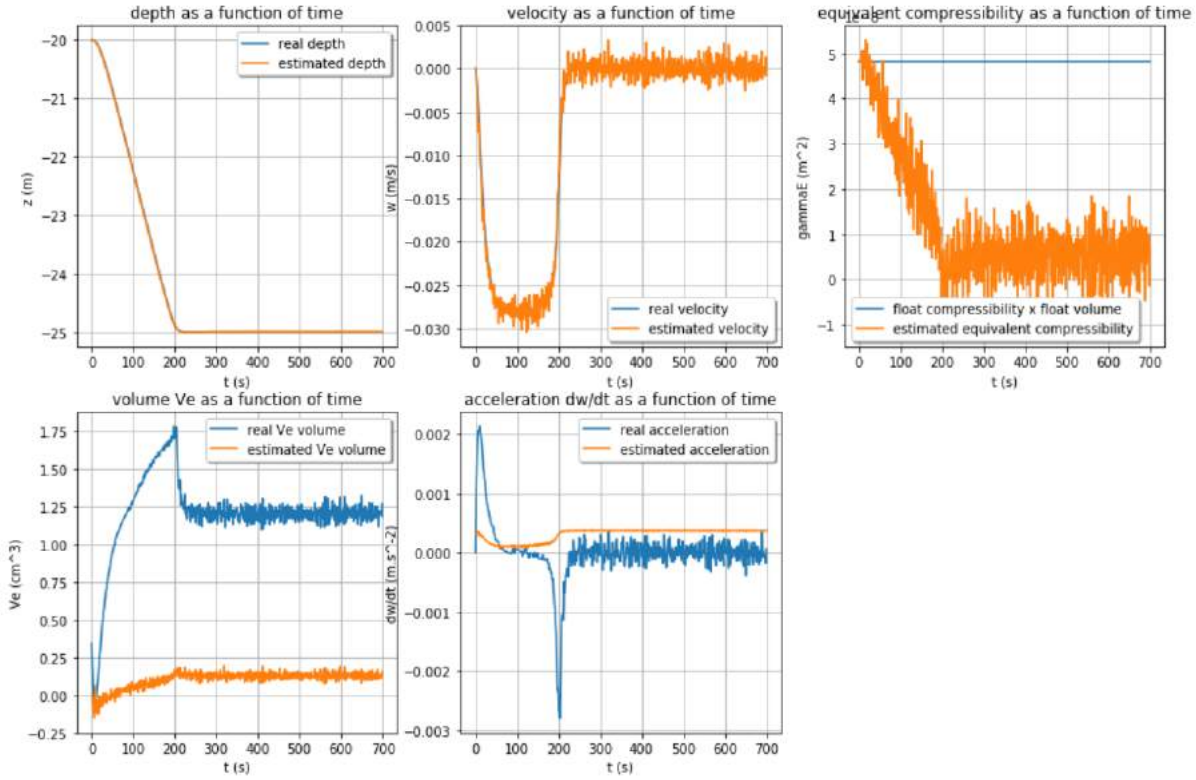


Figure 2.14: Simulation for the Ifremer float with Kalman filter : $e_z = 10^{-3}m$; $e_{V_e} = 10^{-7}m^3$; $e_{\gamma_e} = 10^{-8}m^2$

As regards the Ifremer float, the estimation of the state variables is now a little more accurate, especially on the velocity, hence a more efficient regulation with less oscillations.
As regards the ENSTA float, despite the worse estimation on the volume error V_e, the velocity is now well-estimated, this is why the float succeeds in is regulation.

A last simulation will now consist in considering that there is more uncertainty on the volume error, let's set $e_{V_e} = 10^{-6}m^3$:

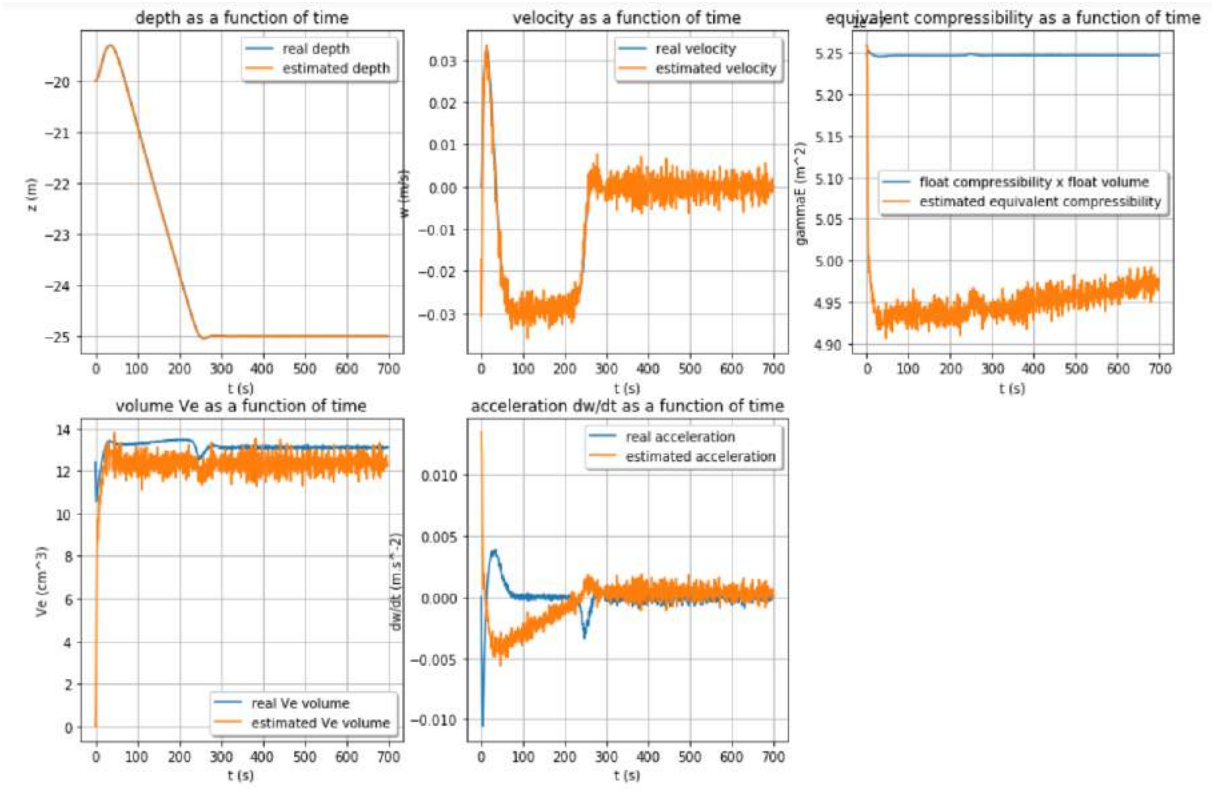


Figure 2.15: Simulation for the ENSTA float with Kalman filter : $e_z = 10^{-3}m$; $e_{V_e} = 10^{-6}m^3$; $e_{\gamma_e} = 10^{-8}m^2$

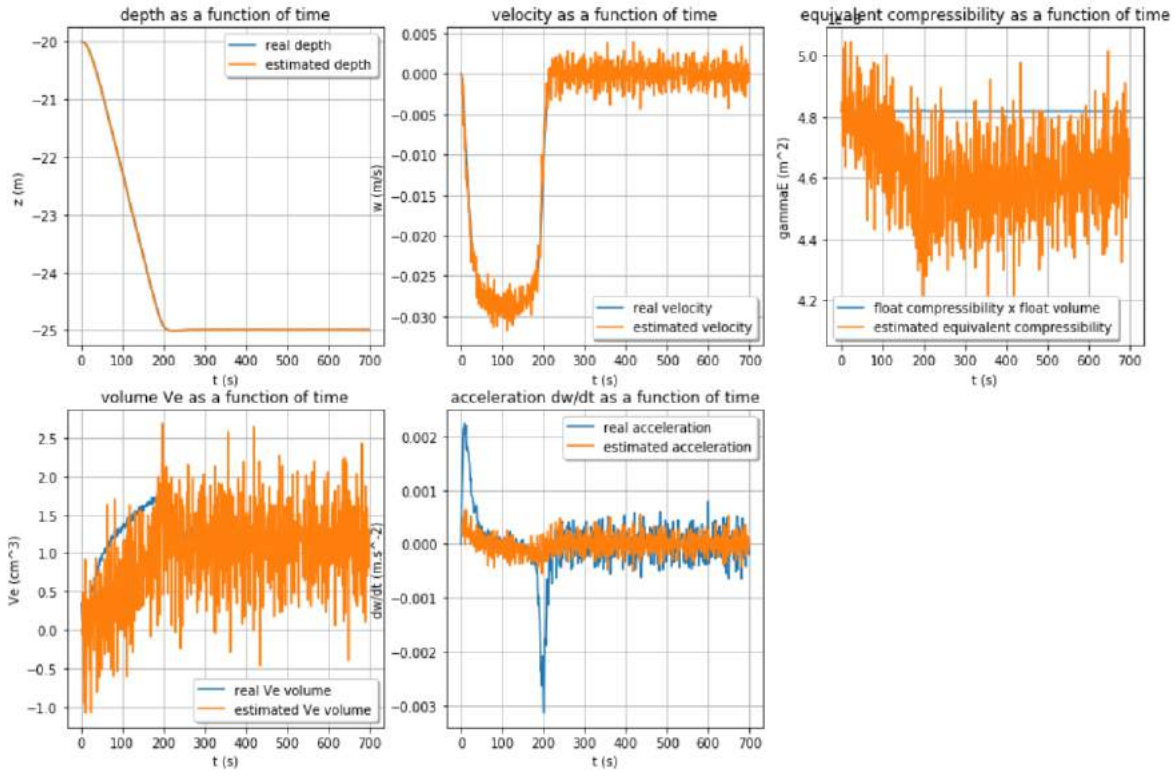


Figure 2.16: Simulation for the Ifremer float with Kalman filter : $e_z = 10^{-3}m$; $e_{V_e} = 10^{-6}m^3$; $e_{\gamma_e} = 10^{-8}m^2$

According to the curves below, for both floats, the estimated values contain more noise except for the depth. However, now the estimated values better match. Moreover, it can be noted on the graphs above that the variations of the error volume V_e are rather closer to $10^{-6}m^3$ than $10^{-7}m^3$. This observation led me to call into question the choice of the definition adopted for the calculation of e_{V_e}. Indeed, after having considered approximations thanks to the equation $\delta V = \gamma_V z$ (4.13) that can be found in the appendix (see "Command calculation for the state feedback regulation"), the value $e_{V_e} = 10^{-6}m^3$ seems to be indeed more suitable. Consequently, the results of these simulations have suggested choosing the following parameters: $e_{acceleration} = 10^{-4}m/s^2$; $e_{velocity} = 10^{-3}m/s$; $e_{V_e} = 10^{-6}m^3$; $e_{\gamma_e} = 10^{-8}m^2$; $e_z = 10^{-3}m$

For the practical trials performed with the ENSTA and the Ifremer floats, the Kalman filter was configured with the following default values chosen by Thomas LE MEZO : $e_{acceleration} = 10^{-4}m/s^2$; $e_{velocity} = 10^{-5}m/s$; $e_{V_e} = 10^{-8}m^3$; $e_{\gamma_e} = 10^{-8}m^2$; $e_z = 10^{-4}m$

2.2 Software development of the low-cost drifting float

In this section, I will describe the ENSTA float through its software architecture and explain how I adapted it to the Ifremer float.

2.2.1 Background check on the ENSTA float

The main part of the software architecture which has been used on the Ifremer float during this project has been originally developed by Thomas LE MEZO for the ENSTA prototype. I will present the different features of this architecture.

Software tools used in this project

Thomas LE MEZO has developed all the software architecture of the ENSTA float around the ROS middleware [23] (Kinetic version compatible with the Ubuntu MATE distribution for Linux) standing for Robot Operating System (be careful : unlike the name suggests it, ROS is not a real operating system). A middleware [24] is a software tool that creates a data network between two or more software applications so as to make them exchange data. The use of such a tool is very interesting since it allows to connect the different subsystems together while keeping each individual structure independent. That is to say, several subsystems can be developed individually and then quickly assembled together thanks to the setting up of basic links or “nodes” which are simple communication channels through which the software applications can exchange data by messages called "topic" or "services" depending on the chosen way of communication. Consequently, this software program is interesting to perform teamwork and to test each function of the system separately from the rest of the system, thus without necessary having implemented all the functions yet. Overall, this tool provides many useful services to the user to quickly and easily develop the software of a complex project, and also facilitates the simulation protocols.
For more details about the middleware ROS, you can have a look at the different tools on which

I leant during the training course I offered to my colleagues at the following address :

https://github.com/houdeval/cognac_regulation/tree/master/ROS_presentation

Thomas developed the lower-level layers through the programming languages C++ and Python. More details about this layers will be given next.

Software architecture of the system

As the ROS middleware favours the separability of the system, Thomas LE MEZO has divided the system in several subsystems, each of them being in charge of a certain task or group of tasks, playing a role in the whole system. A simplified organisation chart of the system can be found below.

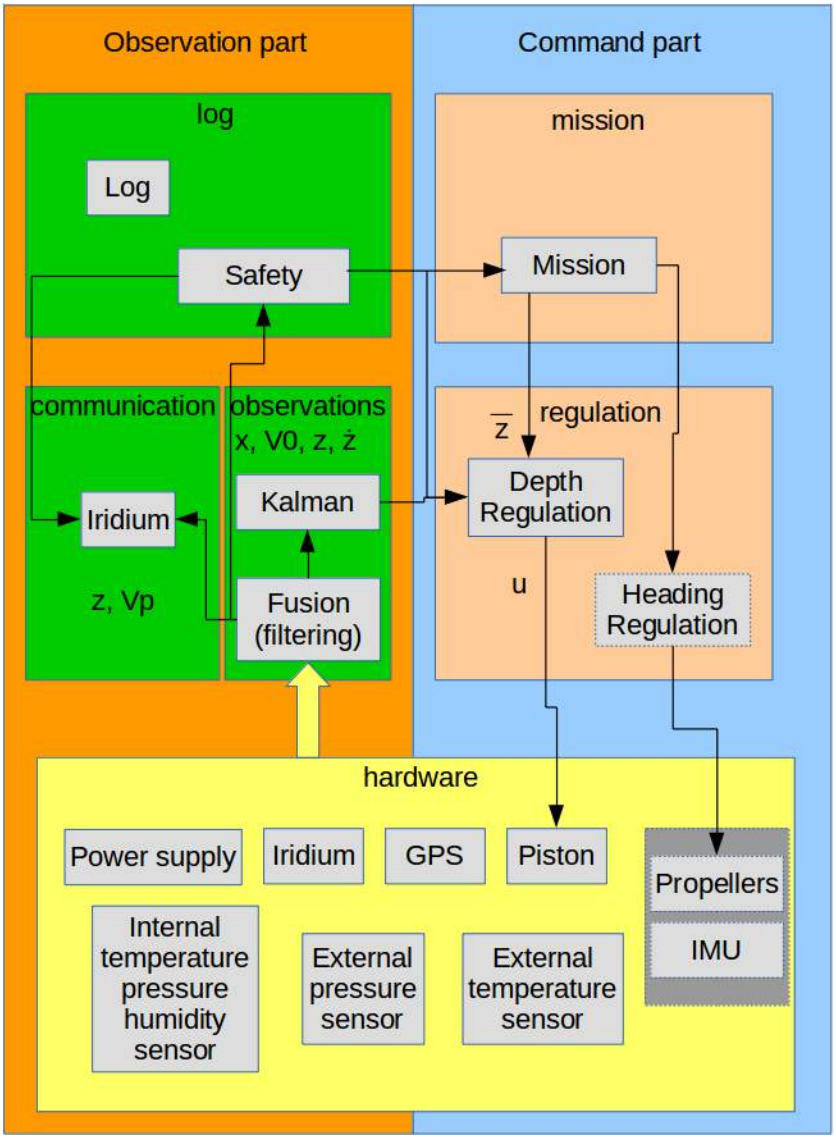


Figure 2.17: Software architecture of the ENSTA float

According to this flow diagram, the whole system can be divided in three main parts.

The hardware part that contains all the physical elements of the system, dedicated to make the system directly interact with its environment. This part contains two kinds of elements : on one hand, the sensors which are the "eyes" of the float and regularly provide data for the float to make it aware of its environment like the external pressure sensor. On the other hand, the actuators allow the float to act directly on its environment like with the piston which allows to change the density of the float. The piston also provides data for the float by indicating its own state thanks to a rotary encoder [11] in the case of the ENSTA float, thus this element can also be considered as a sensor. In addition, the ENSTA float is equipped with an inertial measurement unit (IMU) and propellers as it has been also designed to regulate itself horizontally (heading regulation).
The observation part permits to analyse the data provided by the sensors, this part is in charge of filtering the data through the block "observations". Basically, the data are first filtered once to select only valid data for the rest of the processing chain. Then the data are transmitted to the block "log" that stores them to be analysed by the user after the mission. The block "log" also allows to control the state of the system by the block "safety". Indeed, this part controls for example the internal pressure or the state of the battery so as to detect emergency cases and to make the system resurface in such a case. Finally, the data from the sensors and the safety alerts are transmitted to the block "communication" where they are supposed to be sent to the user through an iridium communication.
Then the data are filtered a second time by a Kalman filter to estimate unknown necessary variables for the regulation before being sent to the command part through the block "regulation". The block "regulation" is directly under the orders of the block "mission". This block "mission" contains a finite-state machine (FSM) [25]. The FSM simply allows the float to be in a certain configuration depending on the results of different variables. To sum up, after launching the mission for a depth regulation, the float will enter the "sinking" state, that is to say, before reaching a given depth which can be chosen among the parameters, the float will spend time to set some fundamental values, for instance the float will compute an offset for the external pressure sensor so as to correctly estimate its depth underwater. The float will also try to estimate its equilibrium position to know its buoyancy reserve. Then the float will sink until reaching the necessary depth to enter the "regulation" state. This state corresponds to the regulation phase of the float. During this phase, the float will try to regulate around the different waypoints which have been defined in a mission file. If the float finishes the regulation phase correctly, it enters the state "mission finished" and waits until the user closes the program. If the float faces a safety issue during the mission, it will enter the "emergency" state that directly orders the float to resurface. During the tests performed with float, it has been noted that sometimes, the float enters the "emergency" state while there is no real reason for that. The different reasons and corresponding parameters on which it is possible to play to avoid this kind of behaviour have been listed in the "Recap chart of the reasons for the float to go to safety mode" that can be found in the appendix.
Finally, during the "regulation" state, the block "regulation" computes from the Kalman filter data, the command to send to the piston so as to reached the depth specified in the mission file for the corresponding waypoint. Specifically to the ENSTA float, there are two blocks in the block "regulation" : the block "Depth Regulation" is the block which has been considered in this project for vertical regulation whereas the block "Heading Regulation" corresponds to the horizontal regulation.

In accordance with the ROS middleware, the contents of these blocks have to respect a certain organisation, overall each block comprises a compilation file for C++ compilation,

C++ and python files, msg files to define the format of the communication messages exchanged between the different blocks and launch files used to launch the functions of each block independently of each other. These launch files also permit to modify some parameters without compiling again the whole software.

The contents of this blocks can be found on the github link of Thomas's float at the following address :

https://github.com/ThomasLeMezo/workspaceFlotteur

All the elements I have just presented concern the high-level layer of the float. These elements are handled by the Raspberry PI 3. In this report, I will not give more details about the electronic part of the float as it had been already studied last year by the student trainee Emilie ARGOUACH but basically, the Raspberry PI is under the orders of a more general PIC micro controller [26]. This micro controller establishes the connection between all the hardware of the float and the Raspberry PI. It also manages some basic functions like turning on the Raspberry PI when the whole system is turned on or getting all the piston out of the float when the Raspberry does not communicate anymore with it for safety issues.

2.2.2 Software adaptation on the Ifremer float

In this subsection, I will describe all the software adaptations I made for the Ifremer float by being inspired by the ENSTA float.

Firstly, I decided to put aside the software elements specific to the heading regulation as it will not be taken into account in the COGNAC project. These modifications have led to the following software architecture for the Ifremer float :

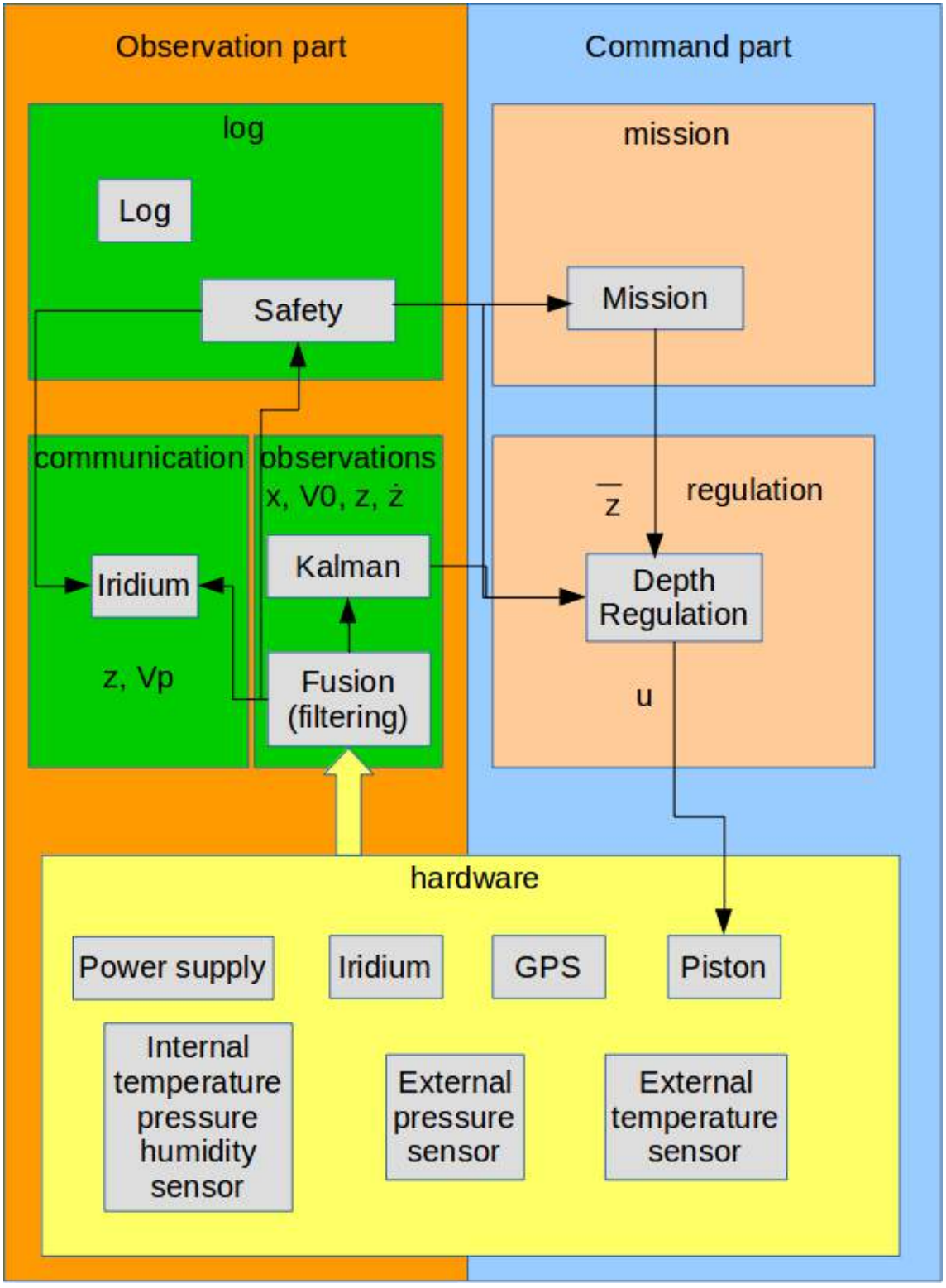


Figure 2.18: Software architecture of the Ifremer float

Secondly, even if the external temperature sensor used in the Ifremer float was the same as the one used in the ENSTA float, its address on the I2C bus [27] had been set differently by the electronics engineer because this address was dedicated to another device. Indeed, all the hardware devices of the float have to communicate through a main channel called a "bus" [28]. So as to identify the data of each device, a unique address is allocated to each part of

each device able to transmit data through this bus, thus I was required to change an address in the driver [29] or software application that manages the external temperature sensor. I also carried out the same task so as to read the energy data provided by the batteries.

In addition, as the Ifremer is supposed to progress deeper than the ENSTA float, the float has been equipped with another external pressure sensor able to provide data until 100 bar (around 1000 m). I was in charge of the implementation of the driver to manage such a sensor through the ROS middleware.

Moreover, a large part of this software development task consisted in implementing a new driver so as to manage the piston. Indeed, unlike the ENSTA float, different addresses have been set for the different functions handled by the piston. Besides, different functions have been managed differently by the Ifremer float, for example the piston of the ENSTA float is equipped with two reed switches [12] whose role consists in detecting if the piston is completely inside or outside the float. As far as the Ifremer float is concerned, these switches still exist but the data related to their activation can only be managed inside the system of the motor to prevent it from forcing if the piston is at the end of its maximum position. Consequently, I had to simulate such information in the software by taking into account the current position of the piston returned by the Hall effect sensor [30] of the piston instead of relying on the value supposed to be returned by the switches in the case of the ENSTA float.
The Ifremer float has been also equipped with a third reed switch that allows to detect whether the larger part of the piston is completely out, therefore I also had to take into account this specific feature in the piston driver since this time, the values returned by the switch could be directly read by the software part of the float.

Eventually, I dealt with the modification in the software part of all the parameters related to the mechanical features of the float.

Chapter 3

Deployments

In this part, I will describe the different tests performed to assess the behaviour of the floats and analyse the corresponding results.

3.1 Tests in the Ifremer pool for the ENSTA float

The Ifremer site in which I worked, has available a 50-metre-long 12.5-metre-large and 20-metre-deep pool to perform some tests for any application. The pool is even provided with a system able to generate swell.

Figure 3.1: Test pool at Centre Ifremer Bretagne

The most interesting feature of this pool is its 20-metre depth, indeed this depth is sufficient to try the float in almost real conditions. Another point is about the kind of water in the pool: the pool is filled with salty water, which allows to work with conditions even more suitable for the final goal of the float which is progressing in a marine environment.

Testing the float in this pool had required several requests so as to book the access to the pool as the pool is often solicited and for safety reasons. For each trial, the float was attached to a ballasted line, deployed until the floor of the pool to recover the float in case the float cannot resurface by itself.

Figure 3.2: Test of the ENSTA float in the pool

Most of the tests performed in the pool consisted in ordering the float to reach and to stabilise at given depths.

These tests allowed me to learn how to handle the float and how to read the log files at the beginning, which will have been useful thereafter as the testing sessions have been similar for the Ifremer float. After several attempts, I faced many problems by making more complex trials. For example, I noted many undesired results over several diving sessions, due to parameters and safety issues in the software, this will be detailed in the result part. I also noticed some deployment issues : at the beginning, it was difficult to drop and recover the float from the water as the height between the ground and the surface of the pool is rather important. Temporarily, a rope was fixed to the float to hold it thanks to a rod.

ENSTA float : results of the tests performed in the pool

As regards the tests performed in the pool, most of the tests consisted in setting instruction thresholds as it can be seen on the graph below where the green line is the depth command and the red curve is the depth effective response. The blue curve in the other graph shows the position command of the piston in the float during the regulation and the red curve refers to the effective position of the piston. Other data can be accessed thanks to other graphs like

the internal and external pressure and temperature. All these data have been very useful to diagnose the different problems encountered by the float.

Key-points to know :

On one hand the regulation speed is a very important parameter as regards regulation, it defines the maximal speed allowed to reach a depth threshold. When the regulation speed is set as low, the float will need much time to reach the ordered depth, however it will stabilise easily when the float will be near the depth threshold as the regulation movements will occur with a small amplitude.
On the contrary, when the regulation speed is set as high, the float will quickly reach the depth threshold but once it will be near the ordered depth, the float will have much inertia and regulation movements will occur with much more amplitude, which will produce oscillations in the depth response in accordance with the results obtained while studying the ν parameter in the simulation part (section 2.1).
Moreover, physics limits the velocity, depending on the marine environment or on the float features, for example, for the ENSTA float, the velocity is limited to 0.15 m/s on the whole, which is almost three times less than the maximum value roughly estimated in the simulation part (section 2.1). The previous statements are illustrated by the graph below.

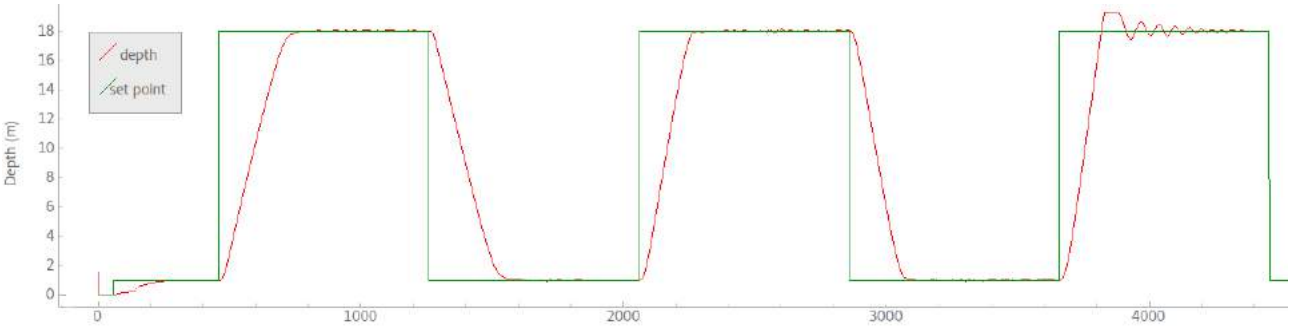


Figure 3.3: Depth response as a function of time for different regulation speeds : 0.04 m/s, 0.10 m/s, 0.15 m/s (from left to right) with $\delta = 1$ m and $\tau = 1$ s

On the other hand, another point to consider is the time set for the float to reach the command. Indeed, as implemented in the software part, this time corresponds to the necessary time to reach the corresponding depth order plus the remaining time dedicated to the stabilisation of the float. That is to say, while planing a mission, the traveling time has to be calculated by taking into account the regulation speed before roughly assessing the stabilisation time.

Description of the mission corresponding to the graph below :

- 0m to 1m at a regulation speed of 0.04 m/s for 400s (in any mission, this first step is necessary since first the float tries to estimate parameters near the surface like its buoyancy reserve thus it will need much more traveling time until the parameters are

estimated, moreover the float needs more time before completely submerging than for any other step).

- 1m to 5m at a regulation speed of 0.04 m/s for 400 s (first real depth threshold)
- 5m to 10m at a regulation speed of 0.04 m/s for 400 s (the float will dive from 5m to 10m for roughly $5/0.04 = 125$ s and consequently it will stabilise for roughly 275 s)
- 10m to 15m at a regulation speed of 0.04 m/s for 400 s
- 15m to 18m at a regulation speed of 0.04 m/s for 400 s
- 18m to 15m at a regulation speed of 0.04 m/s for 400 s
- 15m to 10m at a regulation speed of 0.04 m/s for 400 s
- 10m to 5m at a regulation speed of 0.04 m/s for 400 s
- 5m to 1m at a regulation speed of 0.04 m/s for 400 s
- 1m to 0m at a regulation speed of 0.04 m/s for 400 s (in any mission, this last step is necessary to prepare the transition before the float reaches the surface similarly to the first step).

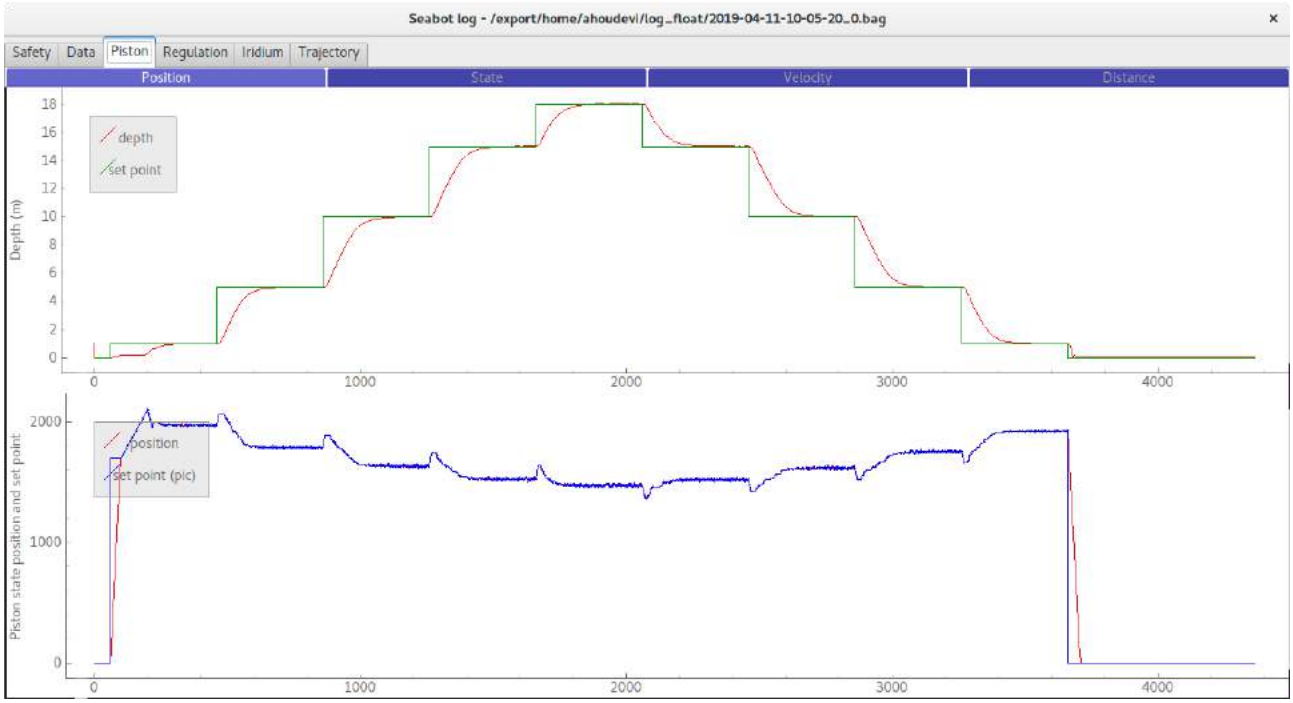


Figure 3.4: Results of a successful test in the pool : regulation speed set at 0.04 m/s, depth and position of the piston as functions of time with $\delta = 1$ m and $\tau = 1$ s

By analysing the graph given above, it can be concluded that the mission was successfully led. Indeed, the effective position curve of the float almost always follows its depth command curve. The curves are not completely similar around the corners of the depth com-

mand curve because the float needs to get a certain inertia before each instruction change. Raising the maximal regulation speed would be a relevant solution to make the curves more similar but only up to a certain point, not to produce oscillations.

3.2 Test campaign in the Mediterranean sea with the ENSTA float

Within the scope of this internship, I was granted the opportunity to embark on a scientific ship so as to perform some tests in real conditions off the coast of La Seyne-sur-Mer near Toulon (France), where another site of Ifremer is located. The interest of traveling to the Mediterranean sea instead of staying near Brest to carry out experiments is the profile of the seafloor. Indeed in the Mediterranean sea, the seafloor becomes deep enough after traveling only some tens miles while reaching similar depths starting from Brest would require to sail during several hundred miles. When we went to La Seyne-sur-Mer, we had to try the ENSTA float which is supposed not to dive after 50 metres thus experimenting at Brest would have been sufficient to perform the tests yet this travel was also planned within the framework of several projects tackled by the laboratory LOPS like testing the acoustic source and the corresponding receiver of the COGNAC project or testing other devices requesting depths of several thousand metres. Belonging to this campaign was the second reason why we went until the Mediterranean sea.

This expedition was mainly dedicated to study the behaviour of the float in real conditions but also to evidence difficulties and unexpected events that could occur during an experimental campaign from an oceanographic research vessel. More details about the results will be presented in the next subsection.

"L'Europe" was the ship used during this expedition. It is a ship equipped with several cabins and a laboratory able to receive a scientific team of eight members. Its structure based on a catamaran structure makes it particularly unstable.

Figure 3.5: Europe ship used for the expedition

The ship provides a large deck and a crane astern to launch the different devices overboard. A zodiac is also available on board.

Figure 3.6: Rear deck of the ship

ENSTA float : results of the tests performed in La Seyne-sur-Mer

Concerning the tests performed during the expedition in the Mediterranean sea, we faced many difficulties by trying the float in real conditions.

First of all, ballasting the ENSTA float was unsuccessful at the beginning, indeed according to the graph below produced by the float in such conditions, it can be noted that even when the piston was completely pulled in by the float, the float encountered difficulties to dive. At first glance, I thought this behaviour was due to swell.

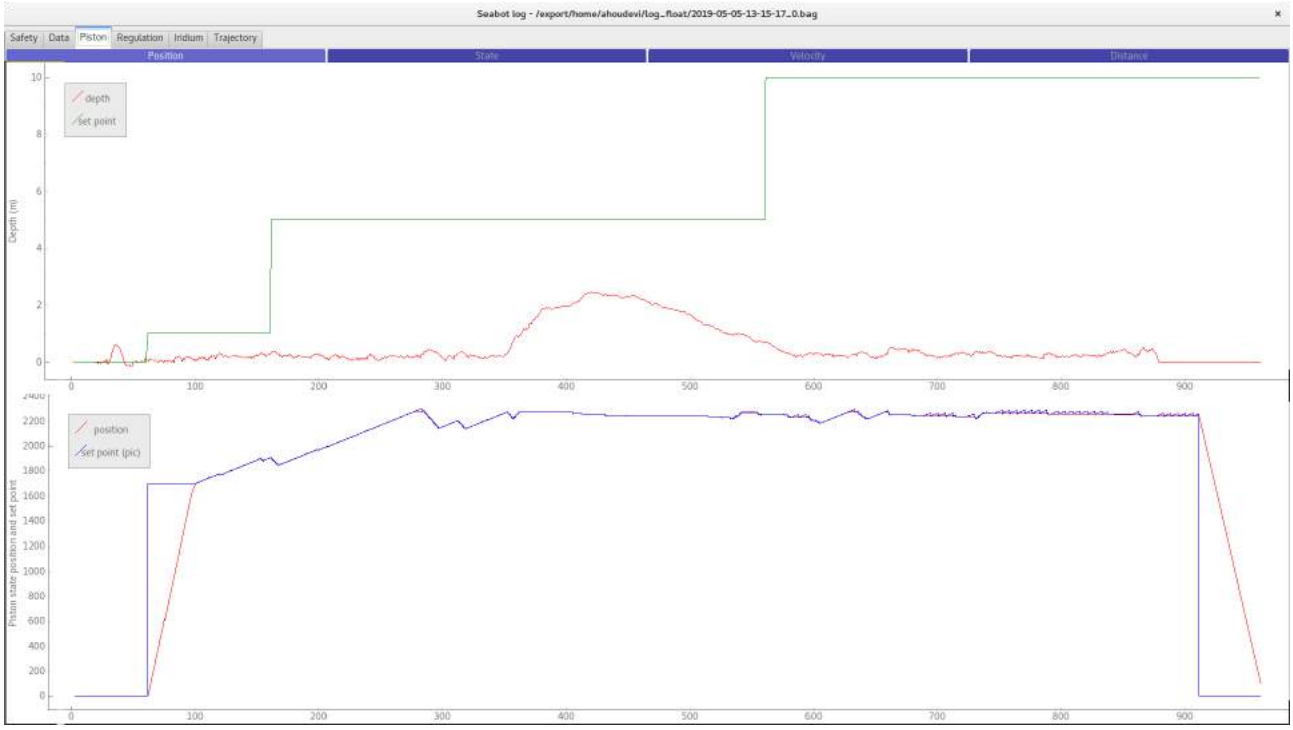


Figure 3.7: Results of a mission at sea with the ENSTA float not ballasted enough with a regulation speed of 0.04 m/s, $\delta = 1$ m and $\tau = 1$ s : depth and position of the piston as functions of time

Another interesting point to test was the remote communication to the float and the deployment process. To deploy the float, a mobile line attached to a buoy was used so as to

test the float without impacting its behaviour by the drift of the ship.

Figure 3.8: Deployment of the line detached from the ship

As regards the remote access to the float, a bullet antenna was used to connect to the raspberry from the ship with a computer. The particularity of this antenna is its very long range of 30km. Moreover, this antenna can be used as a router which was exactly the same remote process used to connect to the float up to then. During the tests, the connection was only limited by the range of the Raspberry antenna which is around 50m but for the long time, it could be conceivable to use a bullet antenna as well in the float to have a range of more than 30km between the Raspberry and the computer. Another interesting point to note was when the float was rather far from the computer and on the surface : the waves disturbed the connection between both devices. I wrote a user manual that can be found in the appendix, in order to configure the bullet antenna (see "Bullet antenna configuration guide").

Figure 3.9: Deployment of the float through a zodiac

Figure 3.10: Bullet antenna attached to the ship

Finally, to avoid deploying the zodiac and the mobile line, the float was tested directly along a line going through the rear crossbar of the ship. The corresponding results have been unsatisfying. Indeed, as it can be seen on the graph below, the float did not succeed in stabilising at the depth ordered and continued to dive even when the piston was completely out. This tendency can be explained by the relative movement between the ship and water due to windage. Indeed, while the line was supposed to be straight thanks to a weight attached to the end, the drift of the ship tilted the line a little horizontally and the friction induced by the line on the float in this situation or the horizontal flow associated with the drift prevented the float from rising. In the last part of the graph below, it can be noted that the float is rising to the surface because the line had been recovered manually.

Figure 3.11: Float performing a mission along the line directly attached to the ship

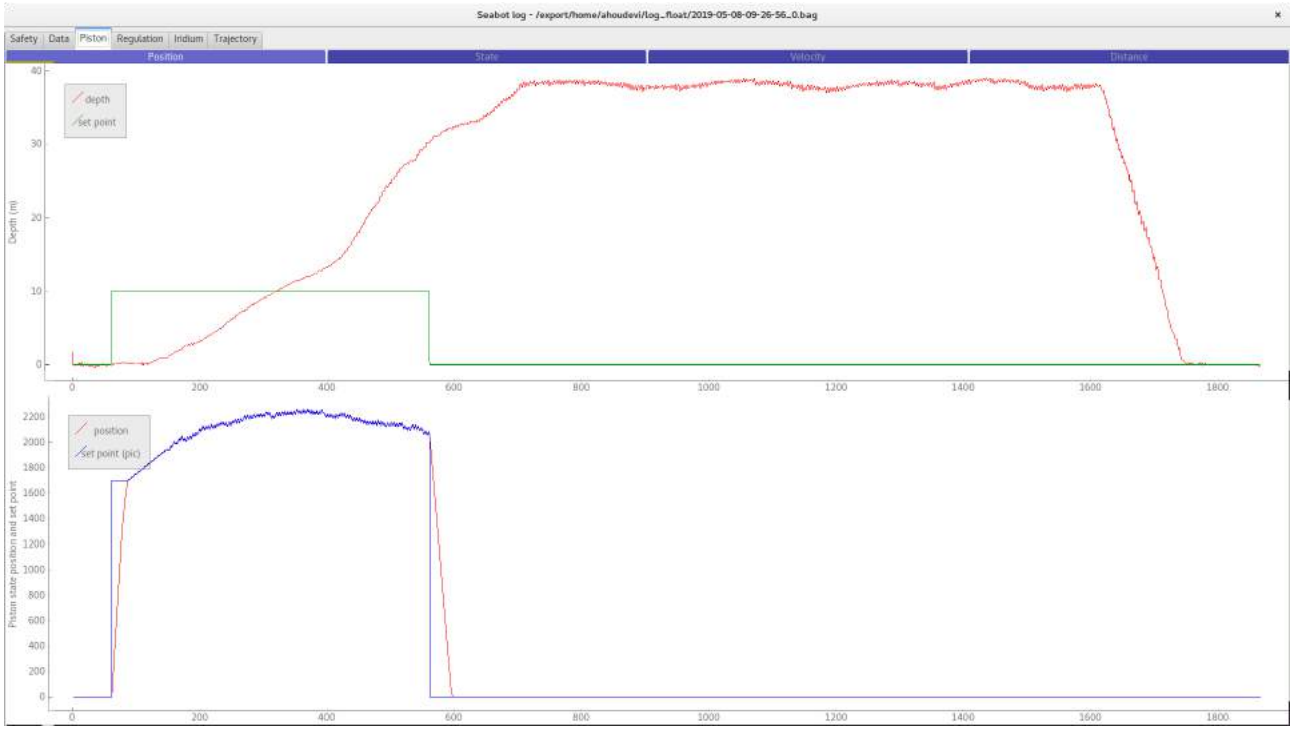


Figure 3.12: Results of a mission at sea with the ENSTA float along the line directly attached to the ship with a regulation speed of 0.04 m/s, $\delta = 1$ m and $\tau = 1$ s : depth and position of the piston as functions of time

3.3 Tests in the Ifremer pool for the Ifremer float

Once the software part of the float was supposed to be completely implemented, the Ifremer float was planned to be tested in the Ifremer pool. Regarding its pressure resistance and waterproofness, the float had already been successfully tested in a hyperbaric chamber with a pressure of 50 bar, corresponding to a depth of 500 m as settled on in the technical specifications.

Before testing the float in the pool, the float had to be ballasted correctly. This step is very important as it allows to set the length of the piston to keep inside the float so as to make it begin sinking easily from its equilibrium position but still keeping a sufficient buoyancy at the surface when the piston is completely out. More details about the ballasting procedure can be found in the appendix (see "Ballasting procedure of the Ifremer float").

After having correctly ballasted the float, it was tested in the Ifremer pool. Two kinds of tests were planned : tests for which the piston was manually controlled and tests with the state feedback regulation where the float was supposed to reach some depths defined before the mission like in the tests performed by the ENSTA float.

Like for the ENSTA float, the Ifremer float was attached to a line in order to get it back in case it does not resurface and to deploy it easily in the pool.

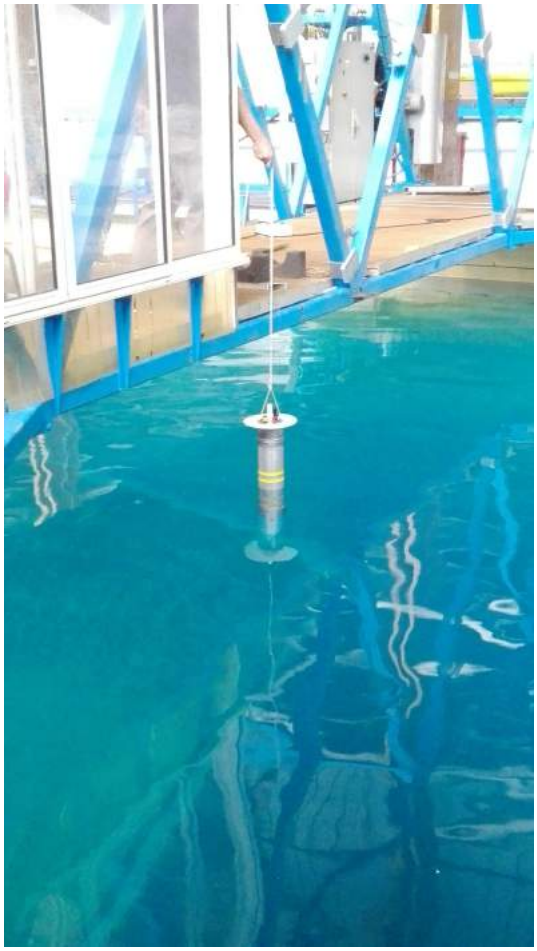


Figure 3.13: Deployment of the Ifremer float in the pool with a line

Figure 3.14: Test of the Ifremer float in the pool

Ifremer float : results of the tests performed in the pool

These tests in the pool were very enriching as they allow to correct many issues. First, we noticed that the float had sometimes difficulties to sink while the ballasting had not changed between the tests for which we noted such differences in the behaviour of the float. This observation was due to the fact that some air was stuck under the flange of the float and in the cover part dedicated to protect the piston. To solve this problem, some holes were drilled in those different elements to allow the air to escape.

Figure 3.15: Flange with holes to let the air escape

Moreover, we also took into account that turning off the float with the magnet after having properly turned off the Raspberry Pi was not efficient enough to assure that the float is actually off. Indeed, as the float is not transparent, there is no LED to witness the state of the float and testing the ssh connection with the Raspberry Pi does not work anyway until the main micro controller is not rebooted. In the long term, a sound signal should be produced to assess whether the float has been turned off correctly.

Test without regulation law

The first test performed in the pool consisted in sending several piston position commands to the float, at regular time intervals.

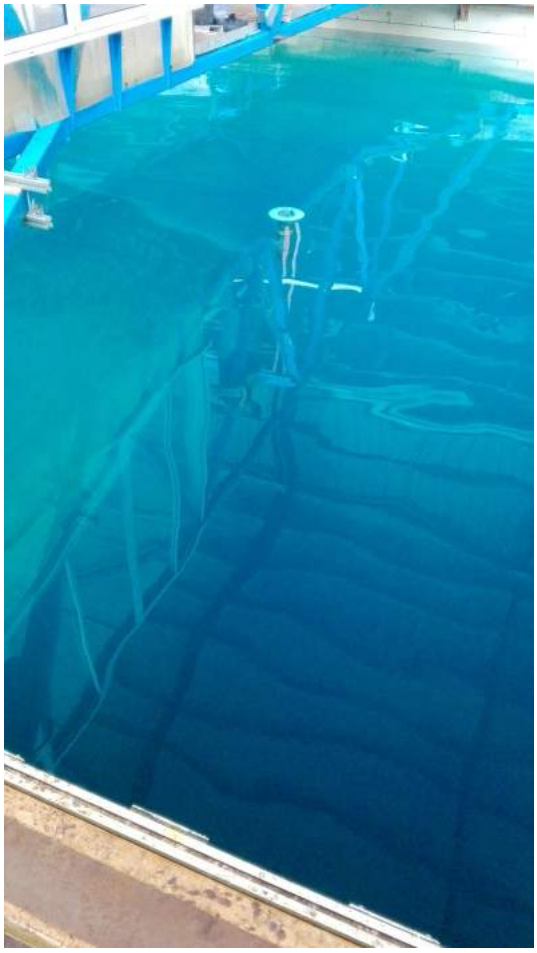


Figure 3.16: The Ifremer float underwater in the pool during the test without regulation

The mission corresponding to the graph below is the following one :

- t = 0 s : keeping all the length of the piston inside the float
- t = 200 s : releasing only all the length of the thinner part of the piston outside the float
- t = 400 s : releasing all the length of the piston outside the float
- t = 600 s : keeping all the length of the piston inside the float
- t = 800 s : releasing only all the length of the thinner part of the piston outside the float
- t = 1000 s : releasing all the length of the piston outside the float

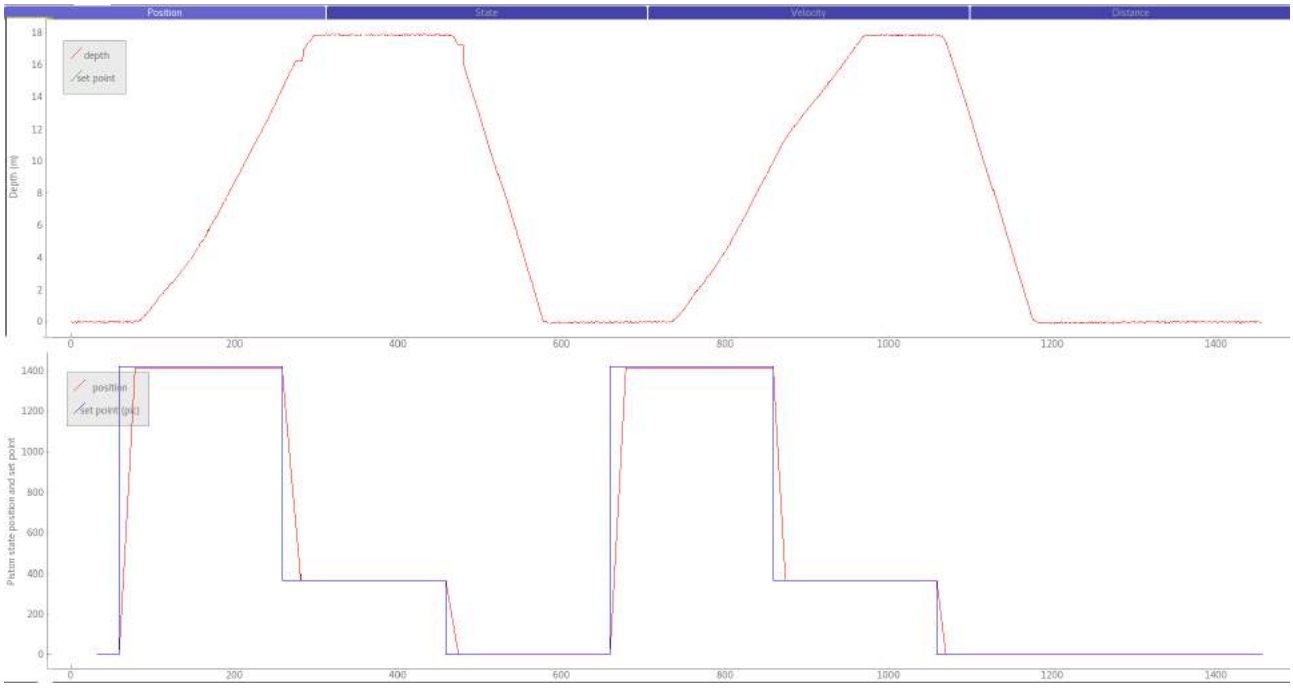


Figure 3.17: Testing Ifremer float without regulation : effective depth and piston position as functions of time

According to the graph, the mission was successfully led since on the graph at the bottom, the red curve corresponding to the effective position of the piston is almost the same as the blue curve describing the position command of the piston. Moreover, the red curve on the graph at the top, representing the effective depth of the float proves the correct behaviour of the float. Indeed, the float is supposed to sink only after having kept all the length of the larger part of the piston plus a small length of the thinner part of the piston inside itself which is consistent with the graph.

Test with state feedback regulation

The second testing phase consisted in regulating the float at some specific depths by sending instruction thresholds to the float before the mission. The mission corresponding to the graph below was the following one :

- 0m to 1m at a regulation speed of 0.04 m/s for 300s
- 1m to 10m at a regulation speed of 0.04 m/s for 600s
- 10m to 1m at a regulation speed of 0.04 m/s for 600s
- 1m to 0m at a regulation speed of 0.04 m/s for 300s

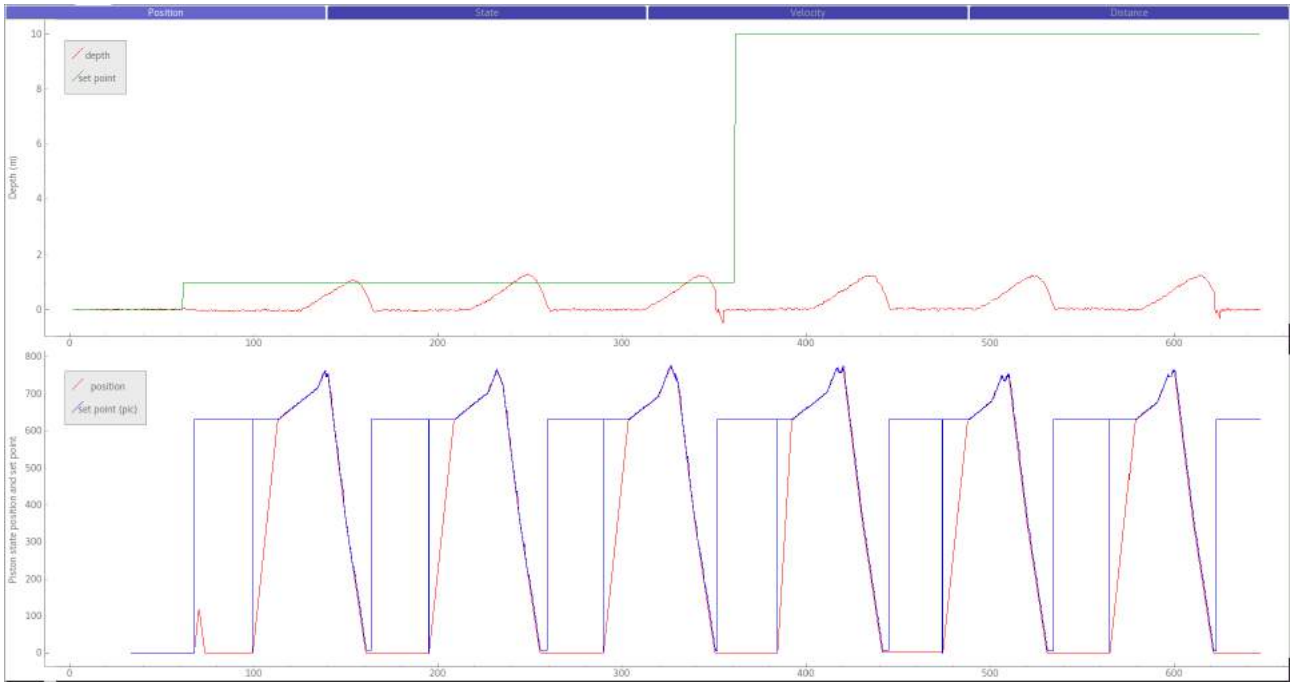


Figure 3.18: Testing Ifremer float with regulation with a regulation speed of 0.04 m/s, $\delta = 1$ m and $\tau = 1$ s : effective depth and piston position as functions of time

Like in the other graph, the red curve at the top represents the depth effective response of the float as a function of time. The green curve is the corresponding depth command. At the bottom, the red curve corresponds to the effective position of the piston as a function of time and the blue one is the corresponding position command of the piston.

According to the graph, the mission failed because the float did not succeed in following the depth command at all. This result is due to the main micro controller which ordered to reset the piston during the regulation process. I did not manage to solve this problem before the end of my internship because the electronics engineer who dealt with the main micro controller was not available to discuss the issue encountered and I did not know anything about the programming language he used.

The tests performed in the pool with the Ifremer float have requested much more time than expected since at the end of one test, the float strangely behaved : a leak had let water go inside the float. After this incident, we had to repair the main electronic circuit board for the most part because it had been damaged by seawater.

As the leak was not easy to find, a test with helium was led. This test is very efficient to detect leaks, indeed unlike detecting a leak in a bike inner tube, this test does not require to put the float in water, which could have damaged the mechanical and electronic parts inside the float. More details about this test can be found in the appendix (see "Leak detection thanks to a test with helium")

The test concluded that the leak came from the external temperature sensor, thus the sensor was changed to solve the problem.

Chapter 4

Project management

In this chapter, I will explain how my working time was organised during this internship. In addition, I will highlight the importance of teamwork and detail the relevance of using some software development tools for this project.

4.1 Schedule and organisation

For this internship, I was required to work 35 hours a week. My supervisor A. PONTE allowed me to manage my own timetable as long as the previous condition was respected. Thus I decided to be based on the following schedule : 9h00-16h30 from Monday to Friday. During this internship I was paid 600 € a month in accordance with the French law for the internships longer than two months.

As regards the different tasks which were assigned to me, the project was mainly divided in two parts : a theoretical part about regulation algorithms and simulation, and a practical part about software development and testing.

The different tasks carried out during this internship are represented and detailed in the Gantt chart that can be found in the appendix (see "Gantt chart of my internship"). This Gantt chart was drawn at the end of the internship since the different tasks have been assigned to me in the course of my progression and of the progression of my colleagues, thus I could not plan ahead the different tasks with exactness at the beginning of my internship.

4.2 Teamwork and software development tools

During this internship, I was required to work with other people as a team. Indeed, the COGNAC project involves five people with whom I collaborated.

Philippe LE BOT, a computer engineer helped me to solve some issues regarding the network procedure on the Ifremer float. I taught him how to set the bullet antenna.

Michel HAMON, the electronics engineer of the project dealt with the electronic part of the Ifremer float. We worked together in order to link the hardware part to the software part of the Ifremer float as he made the addressing of the different electronic components and as I was comfortable with the software part.

Olivier PEDEN, the mechanic of the project went to the Ifremer pool with me each time I needed to perform a test and helped me to keep the float safely with a line. He prepared the float by emptying the air out of the float and by closing it in a waterproof way. We also carried out the ballasting procedure together.

Catherine KERMABON, the Project Manager often came to me in order to have feedback on the project concerning my part. She also helped me in solving some issues encountered in my work.

Aurélien PONTE, a research scientist and also my work placement mentor assigned to me the tasks I had to complete, throughout the internship. We had many feedback moments so as to assess my work and guide me in my work. To share my work easily, I was required to use Git and Github. Here can be found the different tools used to perform the simulations as regards regulation issues :

https://github.com/houdeval/cognac.git

Here can be found the theoretical demonstrations so as to choose the regulation parameters for the state feedback regulation :

https://github.com/houdeval/cognac_regulation

In this last Git repository, there are also some useful user instructions I made so as to handle different procedures used in this project such as cloning a SD card for a Raspberry Pi. All these user instructions can be found in the appendix of this report.

In addition, during a large part of my internship, meetings were organised every two weeks with all the team so as to report the work of each of us about the COGNAC project.

Finally, as the middleware ROS was used in the software part of the Ifremer float like in the ENSTA float and as nobody in the team was comfortable with this tool, I was required to give a short one-day training course to the team by teaching them the rudiments and the utility of ROS. I chose to make the course interactive through live examples, since having taken such a training course before by myself, I know that learning the use of such a tool requires much practice so as to understand and remember the use of this software. The different tools on which I relied during my presentation can be found at this address :

https://github.com/houdeval/cognac_regulation/tree/master/ROS_presentation

Conclusion

The COGNAC project was well suited to my robotics engineer profile during this internship. Indeed, contributing to the algorithmic and software development of an Autonomous Underwater Vehicle (AUV), whose mechanical part had already been designed, allowed me to make good use of my knowledge about robotics, in order to adapt the software architecture of the ENSTA float to the Ifremer float.
Besides, the simulation work assigned to me by Aurélien PONTE allowed me to have the benefit of hindsight on the whole project, considering in this way the whole theoretical part at the root of the development of the float. This simulation part also obligated me to understand and to establish connections between all the tasks which have been assigned to me.
As regards the project, although I ran out of time at the end of my internship to perform more regulation trials at the pool with the Ifremer float, I eventually managed to implement all the software structure of the Ifremer float by drawing my inspiration from the structure of the ENSTA float. I have also made much progress with the theoretical part on the state feedback regulation so that better choose the regulation parameters of the float in order to give it the wanted behaviour. Finally, I have developed and presented to my colleagues, many short user manuals to handle the Ifremer float, which I hope, will be able to help them to keep up with the development of this project after the end of my internship. For a matter of time, I was not able to study the temperature regulation of the float and the energy consumption issues at all, whereas they had been suggested as potential subjects to tackle at the beginning of my internship. To carry on with this project, it would be relevant to continue the regulation trials at the pool with the Ifremer float, to implement temperature regulation algorithms based on the same protocol that has been used for the depth regulation. Moreover, undertaking a comparative study on the energy consumption between the ENSTA and the Ifremer floats would be a step forward in the wake of this project.

Even if some of the tasks of the project have seemed less attractive to me, especially as regards the modification of the simulator to adapt it to the state feedback regulation, I honestly enjoyed working on the project. Indeed, thanks to its versatile nature, this project allowed me to make use and mostly to learn a great deal of new skills, for instance in networking while implementing the communication protocol with the float or while integrating the different sensors to the software part of the float.
This project also allowed me to strengthen some basics which were still quite weak at the beginning of my internship, especially about the use of the ROS middleware or the programming in C++. Asking me to prepare a one-day training course about ROS obligated me to well-understand the functioning of this middleware and not to only apply commands copied from a tutorial without understanding as I used to do up to now because I had not understood everything last year.

Furthermore, it has been the first ever time I found myself working with a real professional team on such a major project. This has made me realise the importance of the communication and the understanding of each one within the scope of teamwork. I have been also very satisfied with all the means which were provided for me to successfully complete my different tasks during this internship, especially during the trials at the Ifremer pool.
Eventually, even if it gave rise to much apprehension and anxiety to me, I was glad to be able to embark during the test expedition in La Seyne-sur-Mer in May, this has made me realise that the work of a robotics engineer does not only consist in staying in an office while occasionally performing trials in the field but it can also require to dedicate oneself much more to one's work by forgetting a little about one's personal life so as to fulfil the requirements of a project.

In conclusion, I have really enjoyed this internship at Ifremer and I have found it very enriching. In addition, for having directly held the position of a robotics engineer, I have liked working with the team of the project. Eventually, thanks to this internship, I have the impression of having grown in maturity as regards the professional world which will open to me in the very near future.

List of Figures

Bibliography

[1] Aurélien PONTE. Scientific proposal, anr astrid 2019. page 13, 2019.

[2] Wikipedia. Établissement public à caractère industriel et commercial en France. https://fr.wikipedia.org/wiki/%C3%89tablissement_public_%C3%A0_caract%C3%A8re_industriel_et_commercial_en_France. Accessed: 2019-20-08.

[3] Ifremer web site. Carte des implantations. https://wwz.ifremer.fr/L-institut/Organisation/Carte-des-implantations. Accessed: 2019-05-17.

[4] Mathieu BELBEOCH. *Argo part of the itegrated gloabal observation strategy*. Accessed: 2019-06-03.

[5] Wikipedia. Argo (oceanography). https://en.wikipedia.org/wiki/Argo_(oceanography). Accessed: 2019-06-03.

[6] WikiMéca. Méthode APTE. https://www.wikimeca.org/index.php/M%C3%A9thode_APTE. Accessed: 2019-06-02.

[7] Thierry ROPERT Luc JAULIN Aurélien PONTE Benoît ZERR Thomas LE MEZO, Gilles LE MAILLOT. Design and control of a low-cost autonomous profiling float. pages 3–4, 2019.

[8] Wikipedia. Iridium satellite constellation. https://en.wikipedia.org/wiki/Iridium_satellite_constellation. Accessed: 2019-13-08.

[9] Wikipedia. Satellite navigation. https://en.wikipedia.org/wiki/Satellite_navigation. Accessed: 2019-13-08.

[10] Wikipedia. Inertial measurement unit. https://en.wikipedia.org/wiki/Inertial_measurement_unit. Accessed: 2019-13-08.

[11] Wikipedia. Rotary encoder. https://en.wikipedia.org/wiki/Rotary_encoder. Accessed: 2019-09-08.

[12] Wikipedia. Reed switch. https://en.wikipedia.org/wiki/Reed_switch. Accessed: 2019-13-08.

[13] Wikipedia. Raspberry Pi. https://en.wikipedia.org/wiki/Raspberry_Pi. Accessed: 2019-13-08.

[14] Wikipedia. Lithium polymer battery. https://en.wikipedia.org/wiki/Lithium_polymer_battery. Accessed: 2019-13-08.

[15] Wikipedia. Alkaline battery. https://en.wikipedia.org/wiki/Alkaline_battery. Accessed: 2019-13-08.

[16] T. AMIEUR. La commande par mode glissant. http://thesis.univ-biskra.dz/1152/6/Chapitre%2003.pdf. Published in 2009.

[17] Wikipedia. PID controller. https://en.wikipedia.org/wiki/PID_controller. Accessed: 2019-05-06.

[18] Ferdinand PIETTE. Implémenter un PID sans faire de calculs ! http://www.ferdinandpiette.com/blog/2011/08/implementer-un-pid-sans-faire-de-calculs/. Accessed: 2019-05-06.

[19] Wikipedia. Régulateur PID. https://fr.wikipedia.org/wiki/R%C3%A9gulateur_PID. Accessed: 2019-05-06.

[20] Jim MOORE. What is the difference between state feedback and PD controller? https://www.quora.com/What-is-the-difference-between-state-feedback-and-PD-controller. Accessed: 2019-06-17.

[21] Wikipedia. Kalman filter. https://en.wikipedia.org/wiki/Kalman_filter. Accessed: 2019-06-17.

[22] Wikipedia. Object-oriented programming. https://en.wikipedia.org/wiki/Object-oriented_programming. Accessed: 2019-08-07.

[23] ROS.org. About ROS. https://www.ros.org/about-ros/. Accessed: 2019-09-08.

[24] Wikipedia. Middleware. https://en.wikipedia.org/wiki/Middleware. Accessed: 2019-09-08.

[25] Wikipedia. Finite-state machine. https://en.wikipedia.org/wiki/Finite-state_machine. Accessed: 2019-09-08.

[26] Wikipedia. PIC microcontrollers. https://en.wikipedia.org/wiki/PIC_microcontrollers. Accessed: 2019-09-08.

[27] Wikipedia. I2C. https://en.wikipedia.org/wiki/I%C2%B2C. Accessed: 2019-12-08.

[28] Wikipedia. Bus (computing). https://en.wikipedia.org/wiki/Bus_(computing). Accessed: 2019-12-08.

[29] Wikipedia. Device driver. https://en.wikipedia.org/wiki/Device_driver. Accessed: 2019-12-08.

[30] Wikipedia. Hall effect sensor. https://en.wikipedia.org/wiki/Hall_effect_sensor. Accessed: 2019-12-08.

[31] Wikipedia. Mass spectrometry. https://en.wikipedia.org/wiki/Mass_spectrometry. Accessed: 2019-13-08.

Appendix

Organisation chart of the LOPS

Pictures of the ENSTA float

Figure 4.1: ENSTA float : global view

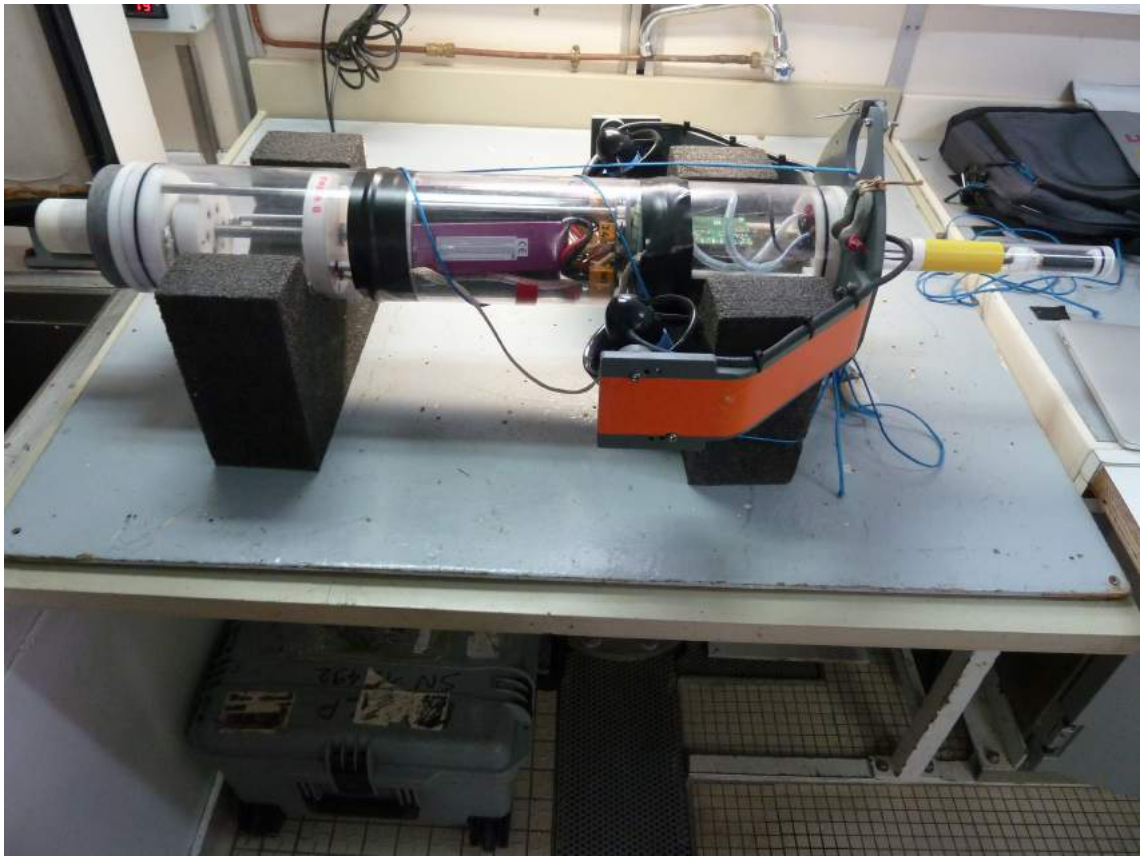


Figure 4.2: ENSTA float : lateral view

Figure 4.3: ENSTA float : antenna

Figure 4.4: ENSTA float : battery container

Figure 4.5: ENSTA float : cylinder

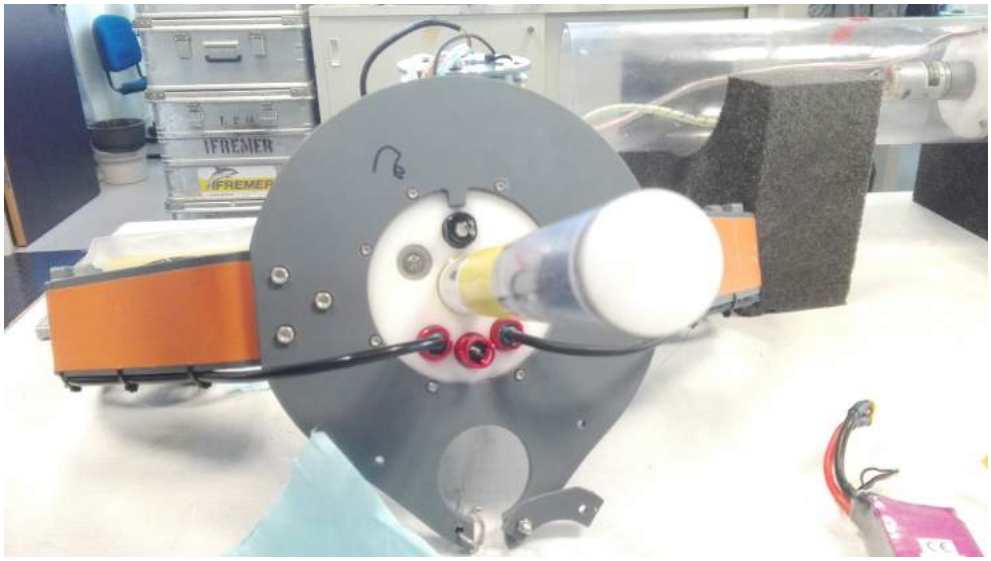


Figure 4.6: ENSTA float : head 1

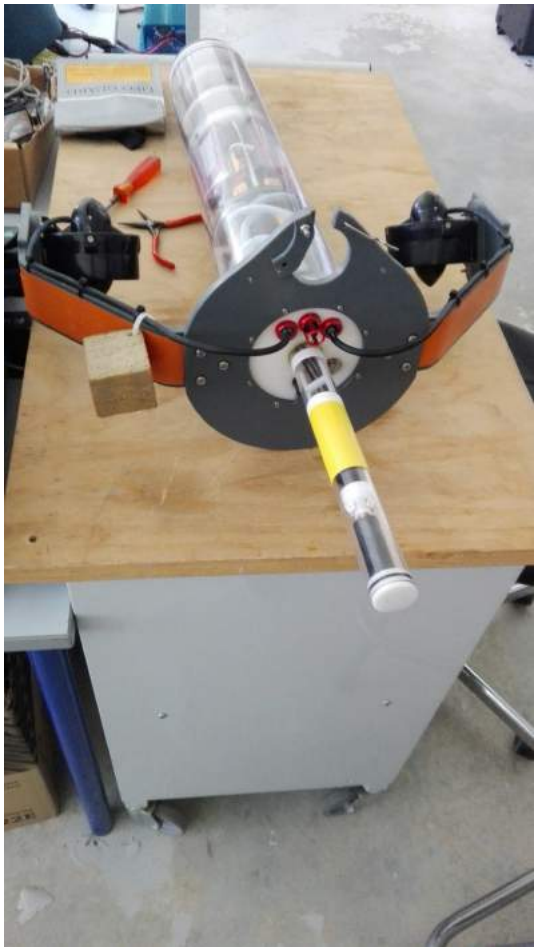


Figure 4.7: ENSTA float : head 2

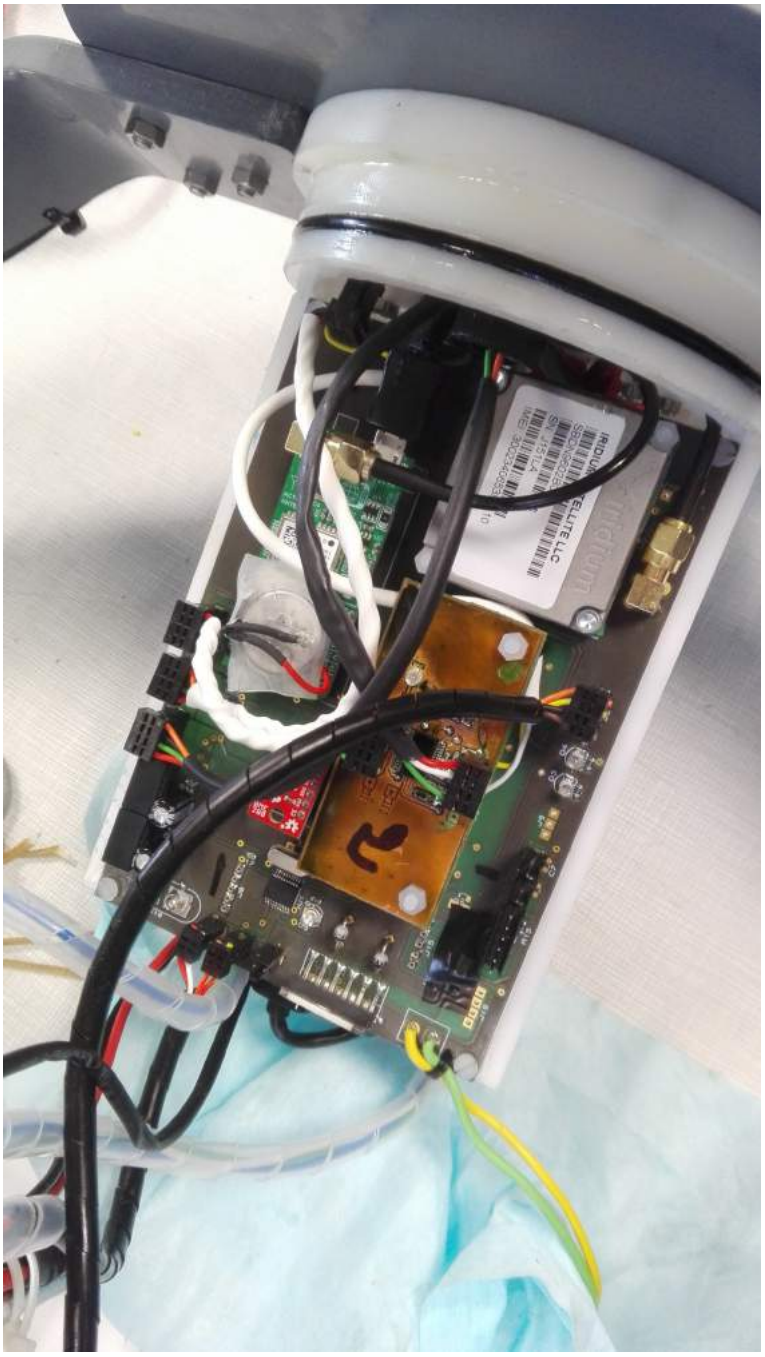


Figure 4.8: ENSTA float : head, face 1

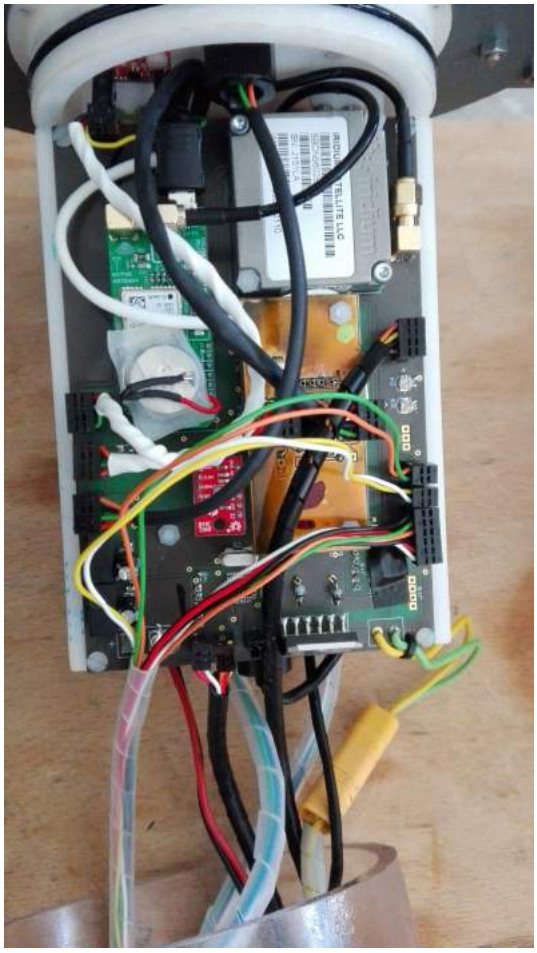


Figure 4.9: ENSTA float : head, face 2

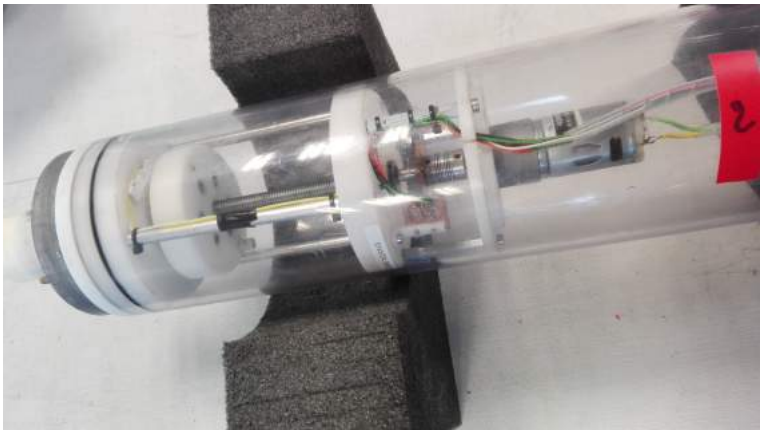


Figure 4.10: ENSTA float : piston

Pictures of the Ifremer float

Figure 4.11: Ifremer float : global view (closed)

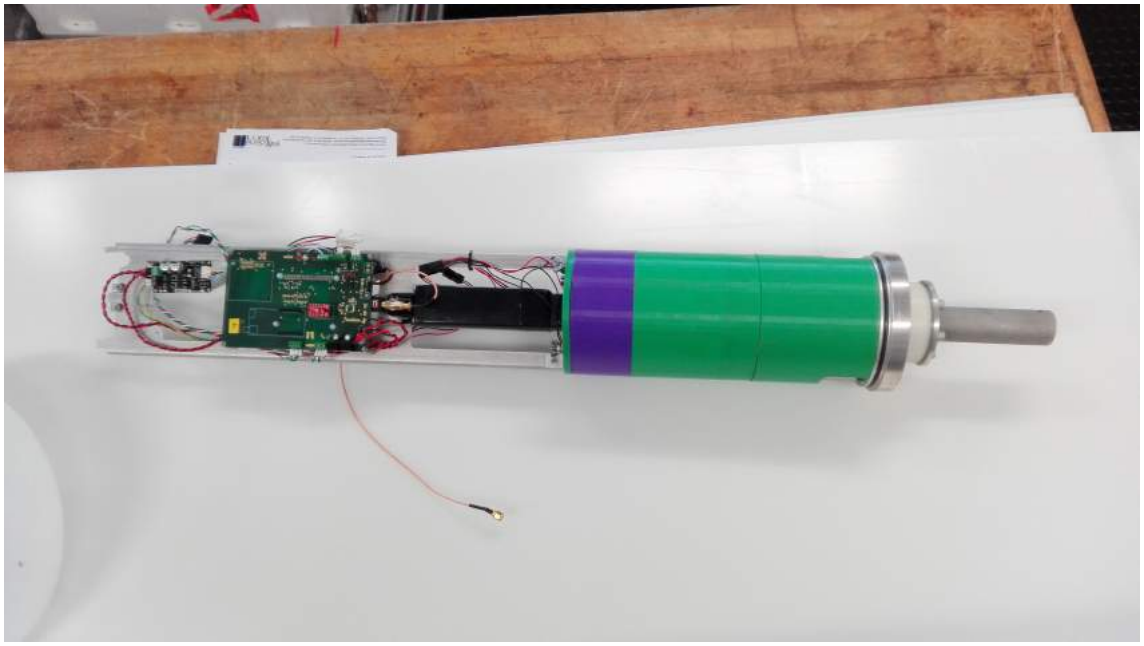


Figure 4.12: Ifremer float : global view face 1 (opened)

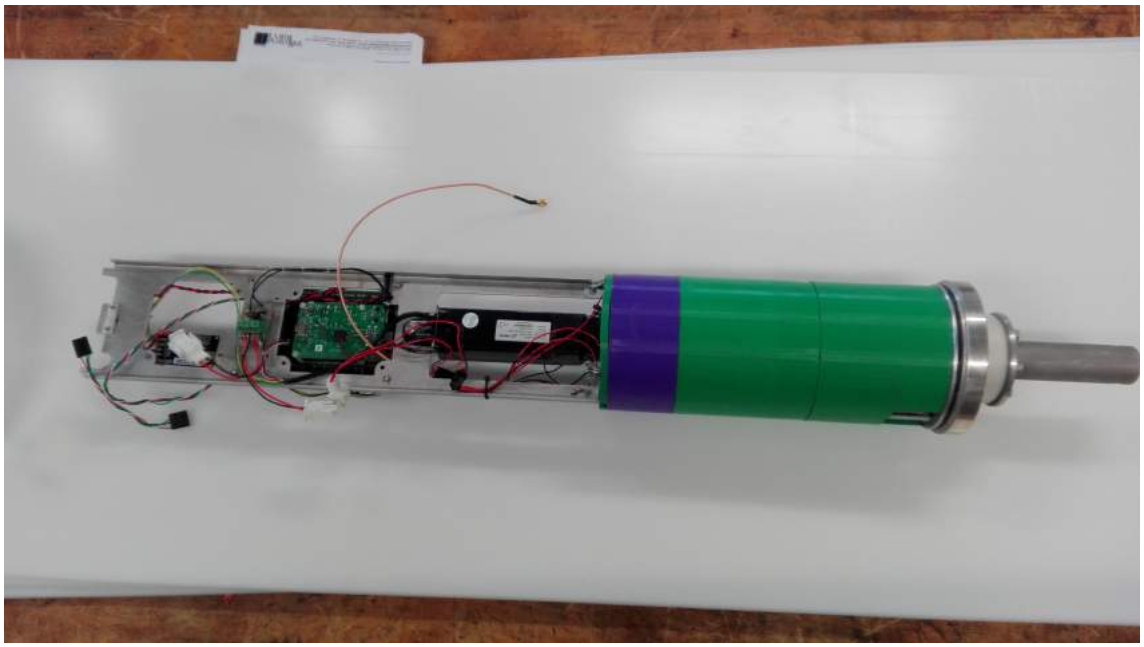


Figure 4.13: Ifremer float : global view face 2 (opened)

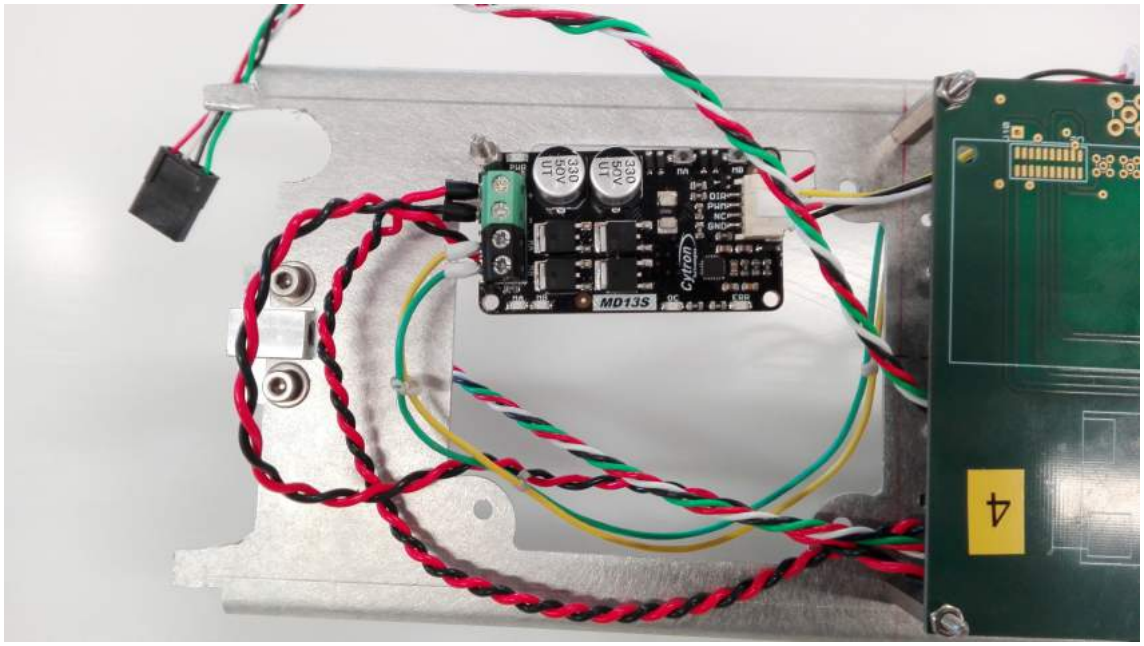


Figure 4.14: Ifremer float : power circuit board

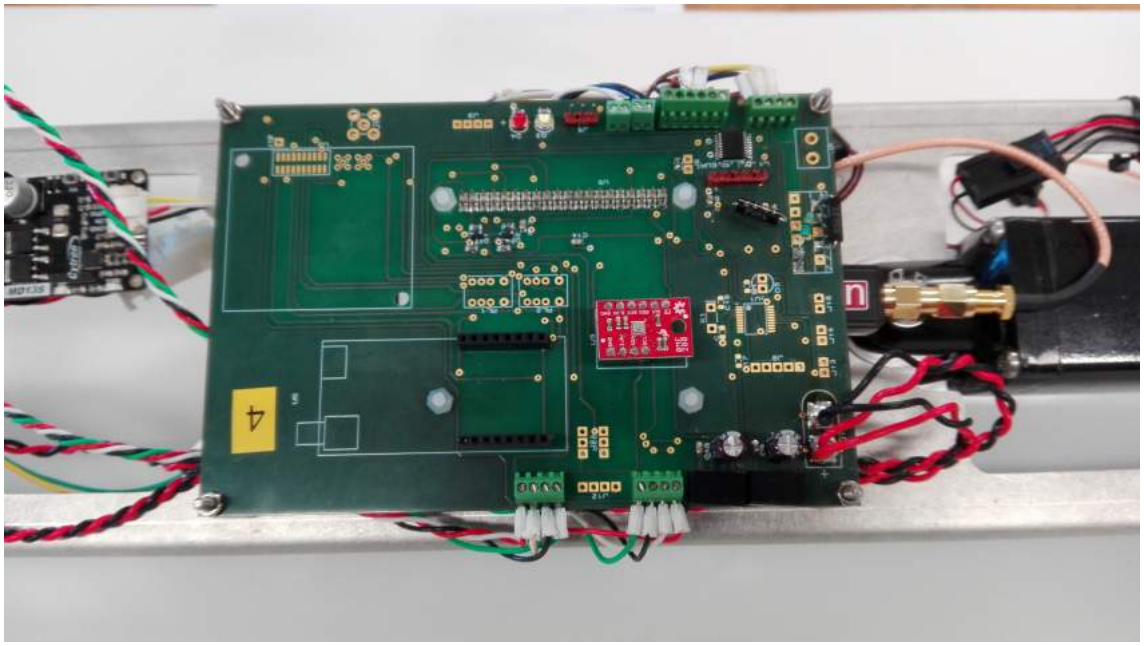


Figure 4.15: Ifremer float : main circuit board

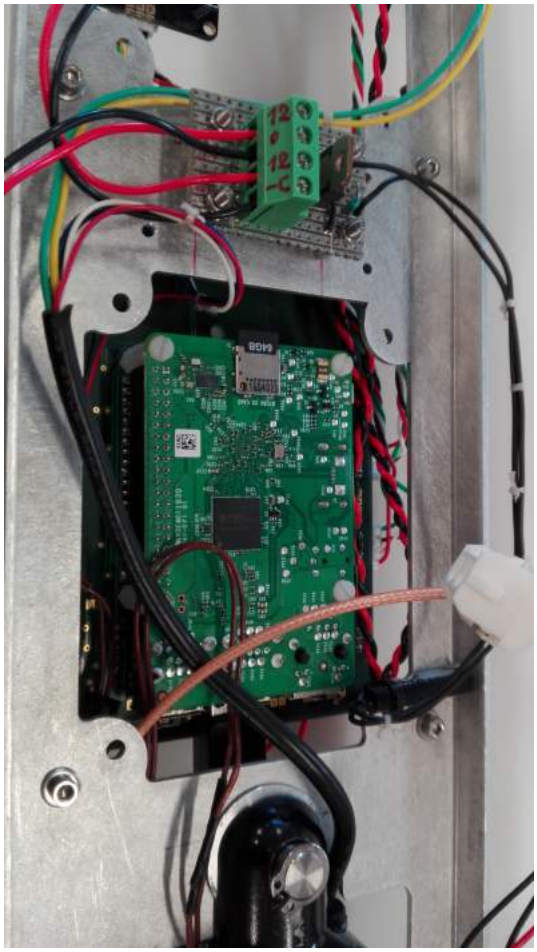


Figure 4.16: Ifremer float : Raspberry PI 3

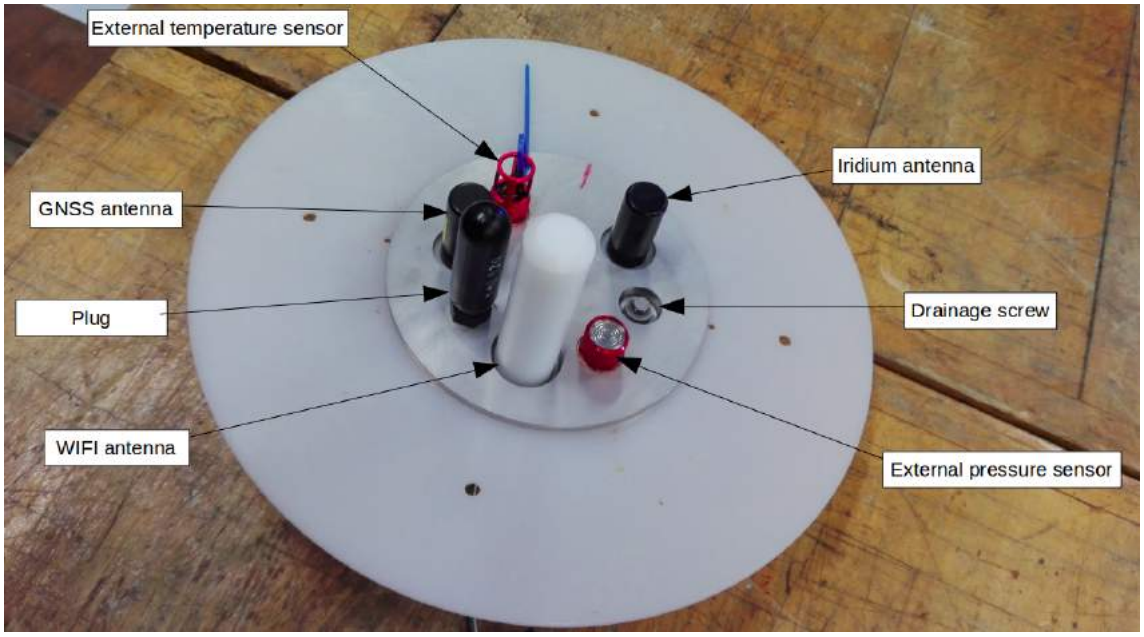


Figure 4.17: Ifremer float : head 1

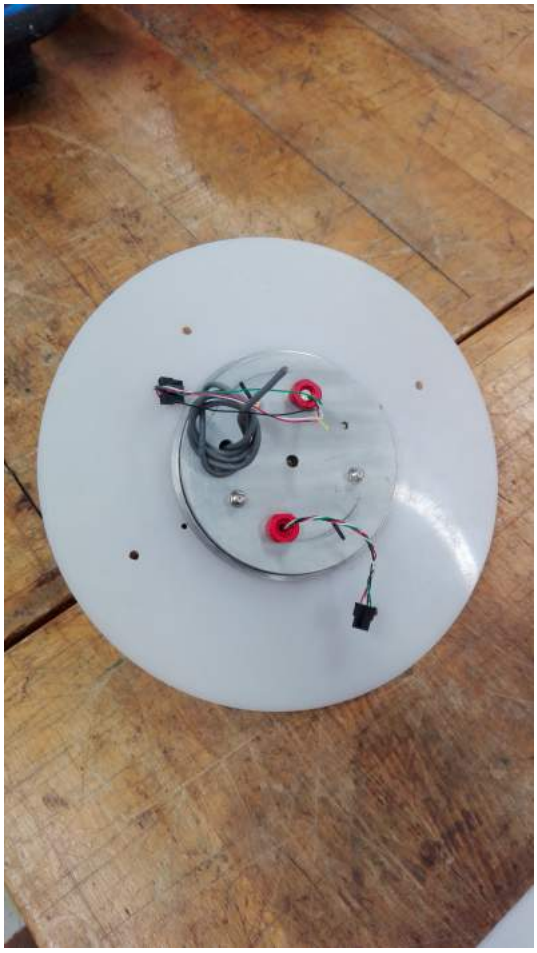


Figure 4.18: Ifremer float : head 2

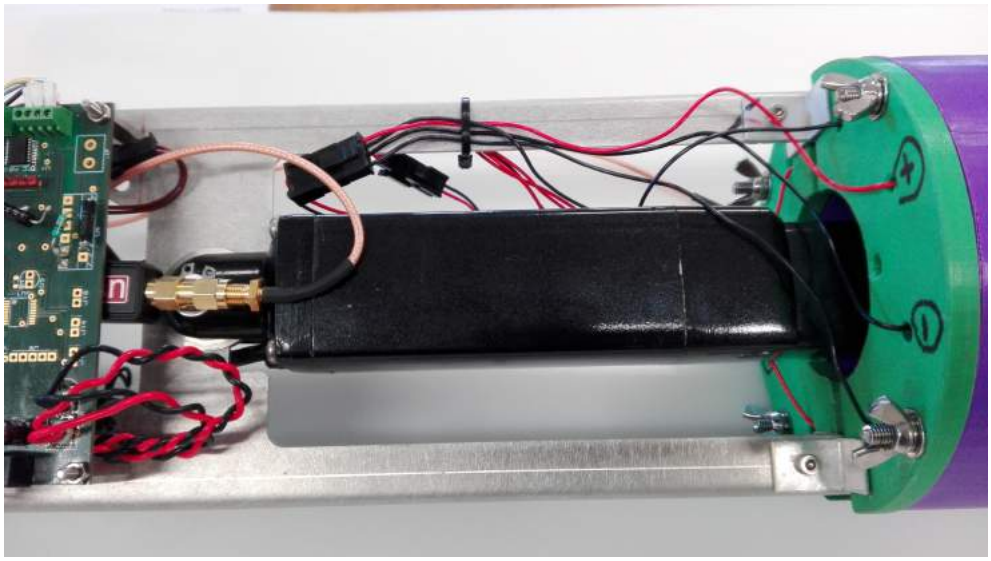


Figure 4.19: Ifremer float : motor

Figure 4.20: Ifremer float : piston

Recap chart of the reasons for the float to go to safety mode

The parameters at stake can be changed in the file $safety.launch$ in the package $seabot\_safety/launch$ without compilation.

Safety reasons:	**published frequency**	**depth limit**	**battery limit**	**depressurization**	**seafloor**
Explanation:	no data received from batteries, internal sensors, external sensors, piston sensor	the float is too deep	low battery	-water leak -too high internal pressure -too high variation of volume	-seafloor reached
Parameters at stake:	-time_delay_batteries_msg -time_delay_internal_sensor_msg -time_delay_external_sensor_msg -time_delay_depth_msg -time_delay_piston_state_msg	-safety_pressure_limit -pressure_limit -time_before_pressure_emergency	-safety_battery -battery_limit	-humidity_limit -pressure_internal_max -delta_ref_allowed -delta_volume_allowed -volume_ref -transition_tick_law -safety_depressure	-max_speed_reset_zero -time_before_seafloor_emergency
Solutions:	-check connections -addressing issues -restart mission	restart mission with lower depth	Change batteries	-check internal pressure -check humidity -allow higher volume variation: change delta_volume_allowed	-restart mission with lower depth -piston completely held in but speed is low: change max_speed_reset_zero

Bullet antenna configuration guide

Installing bullet antenna to allow a long-distance communication between a computer and the Raspberry circuit board of the float

Here, the bullet will be used as a router, that is to say many devices will be able to connect to the antenna so as to be part of a same network. The aim is to access the Raspberry circuit board easily, up to 52.3 km far from the antenna if the Raspberry is equipped with such an emitter.

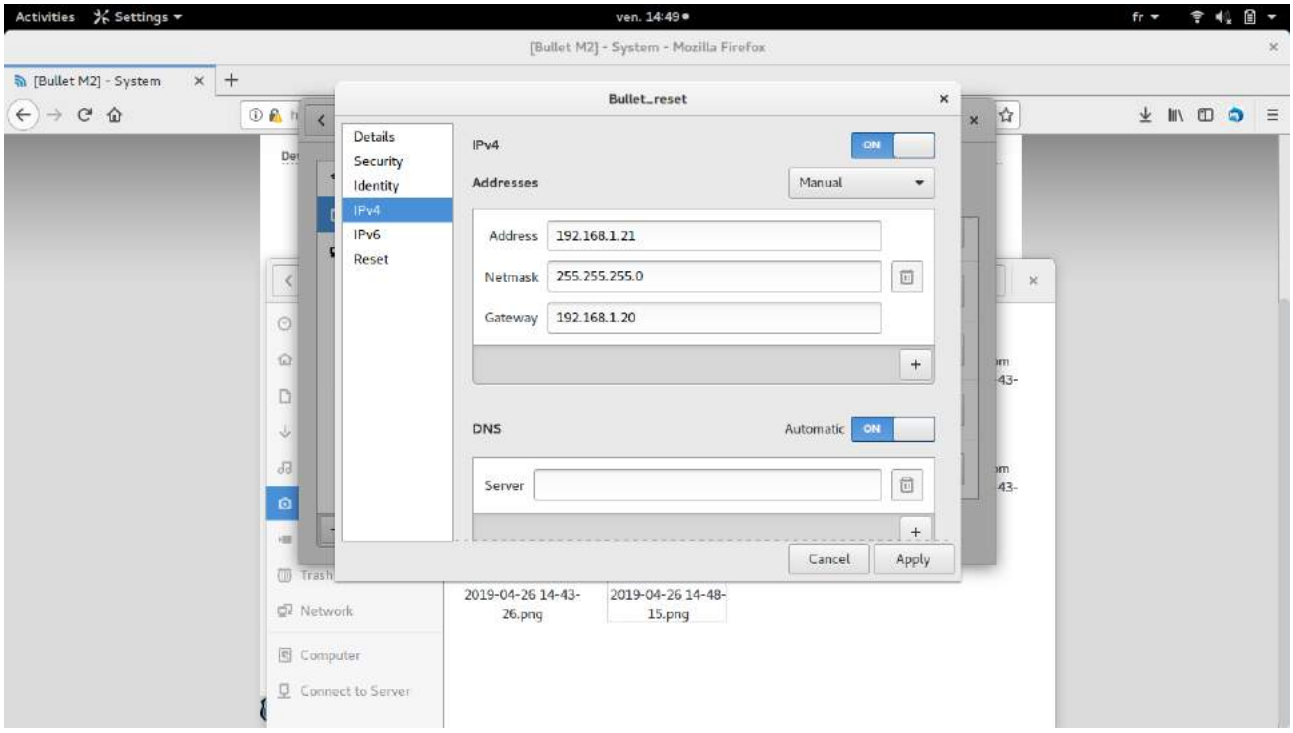
Settings :

To set the antenna considering that the IP or the password to access the antenna configuration page are unknown :

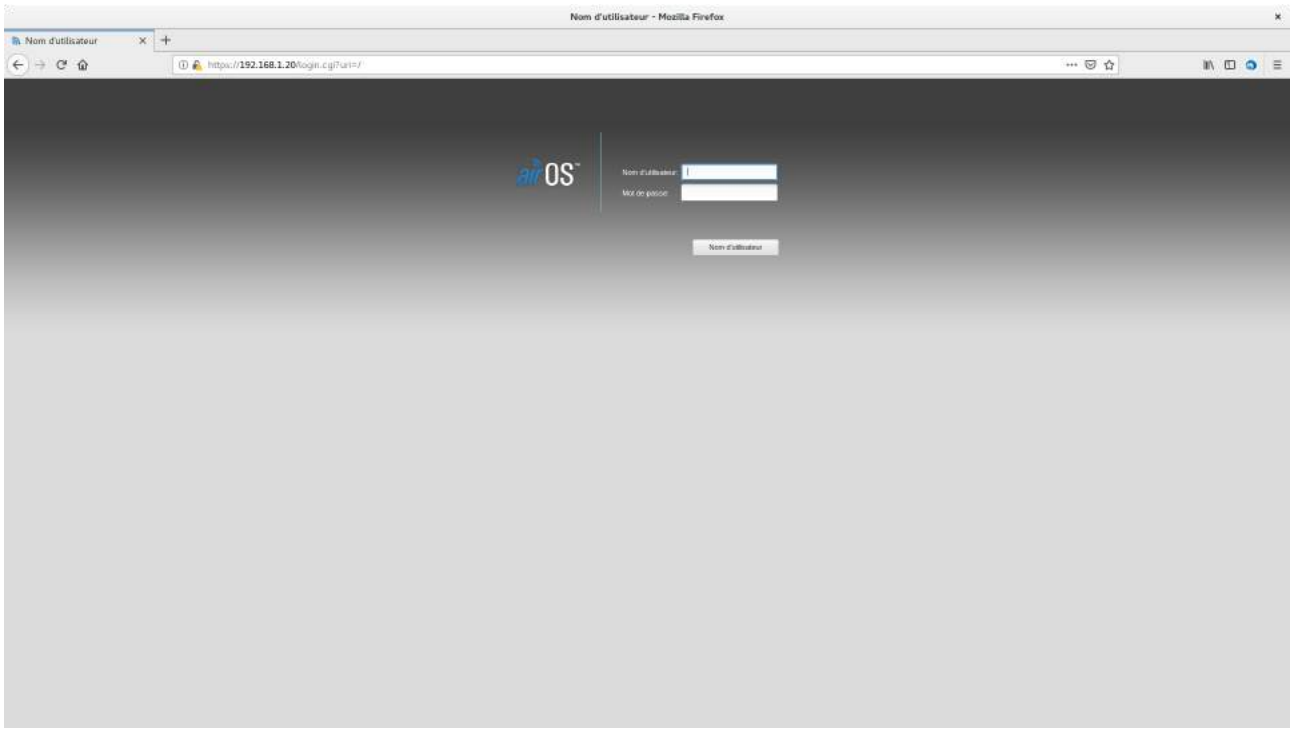
- First, connect both network cables to the adaptator : PWR+DATA OUT is dedicated to the antenna and DATA IN is dedicated to the computer.
- Afterwards, keep pushing the Reset button under the antenna support (about 30 sec) until the light are blinking.
- Then, set your wired connection so as to connect to the antenna whose IP address should be 192.168.1.20 .

  With this aim in mind, you has to set a fixed IP address for your computer, compatible with this network such as 192.168.1.21 .

  Just fill the fields as in the screenshot below, in the IPv4 tab to choose the configuration previously described.



- Then open your internet browser and write the IP address of the antenna in the search bar as below. A user name and a password will be required to continue, by default, both should be "ubnt".



- Next, fill the different fields to configure the antenna as in the following screenshots.

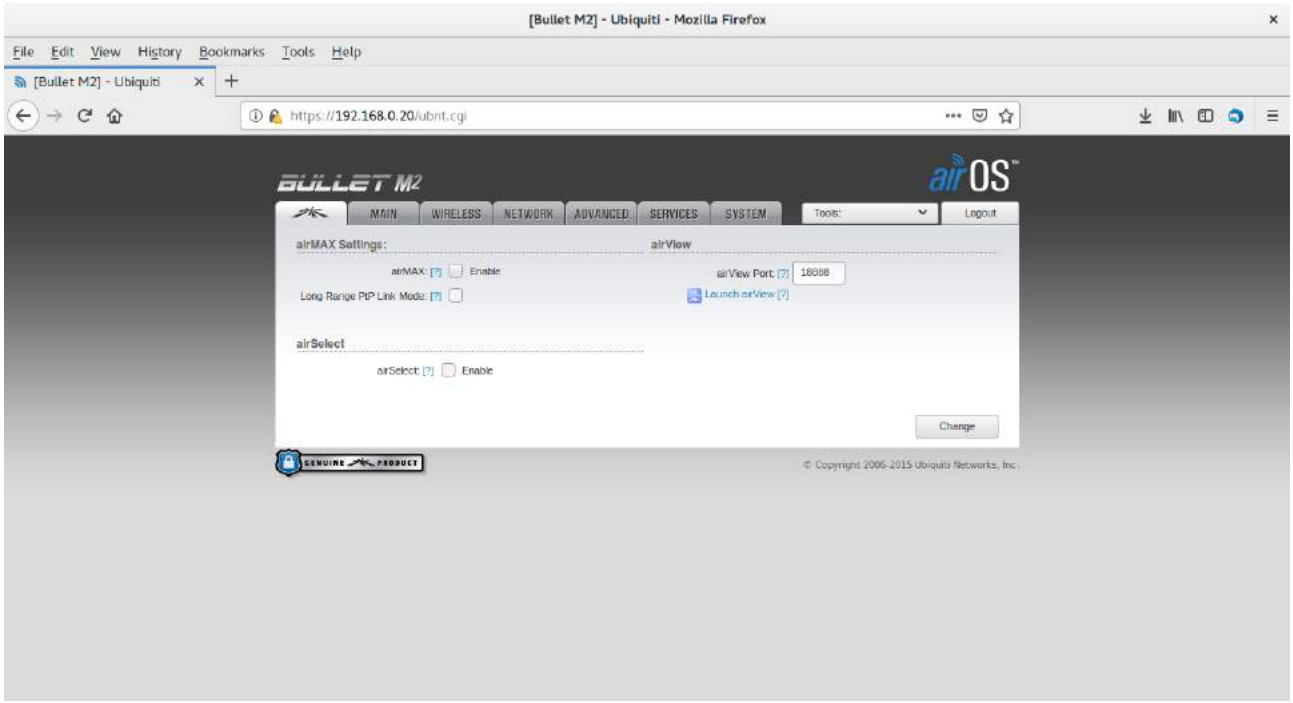
  Note that you can change the distance range of the antenna as you prefer in the section "ADVANCED". This distance also depends on the country code you chose in the

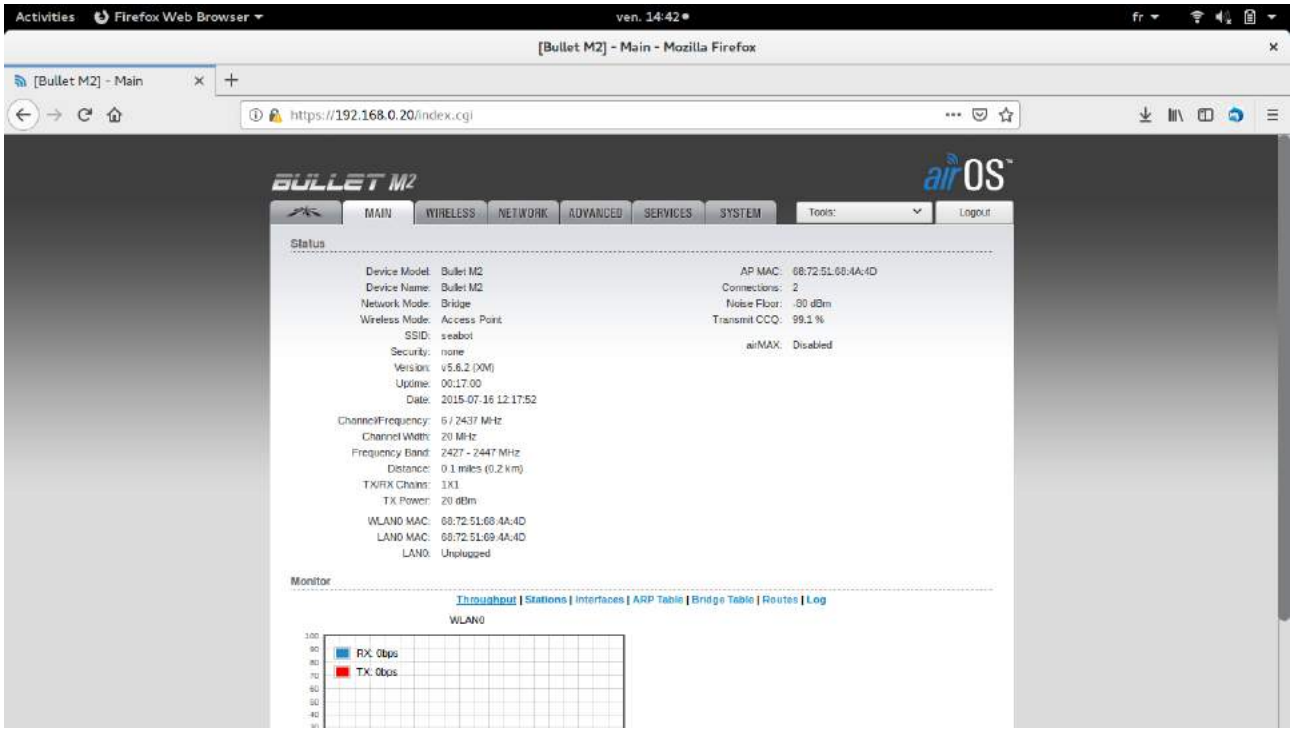
"WIRELESS" section.

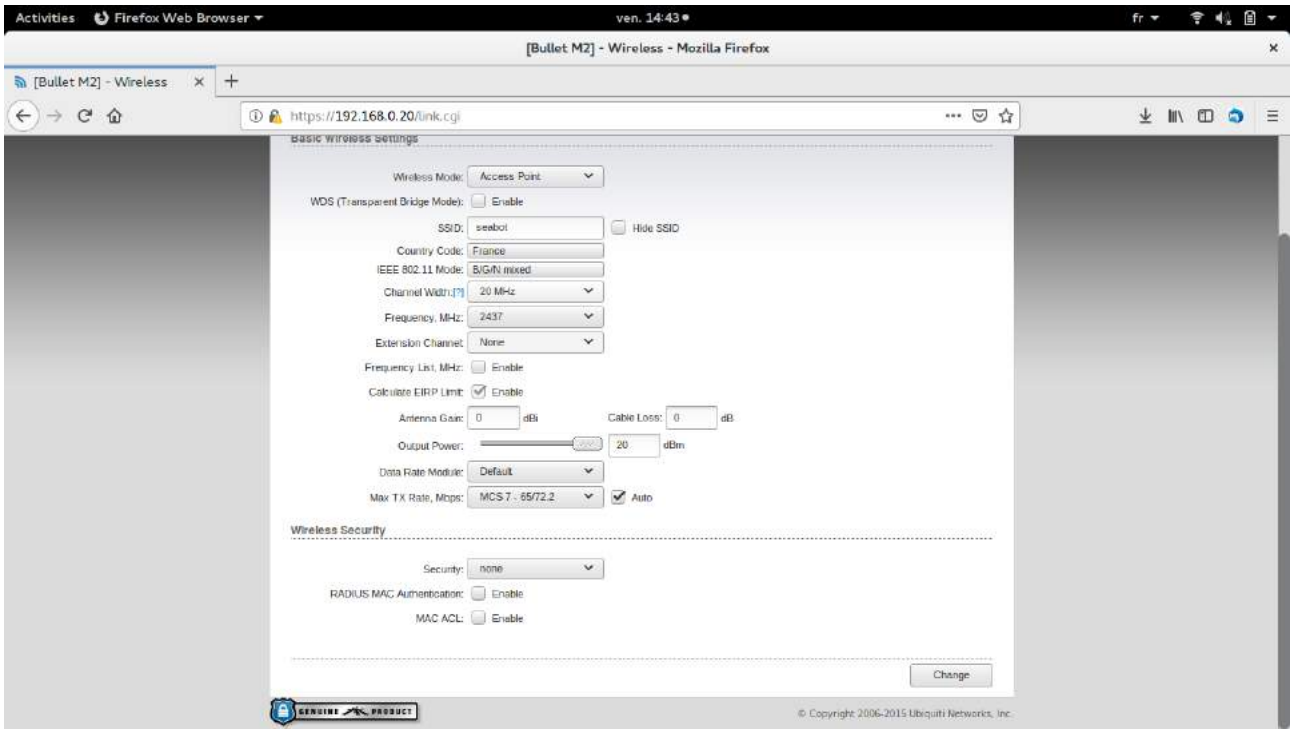
Note that the settings for the choice of IP address family is in the network section for the suggested following settings, the IP address class is 192.168.0.X where the IP address of the antenna is 192.168.0.20 .

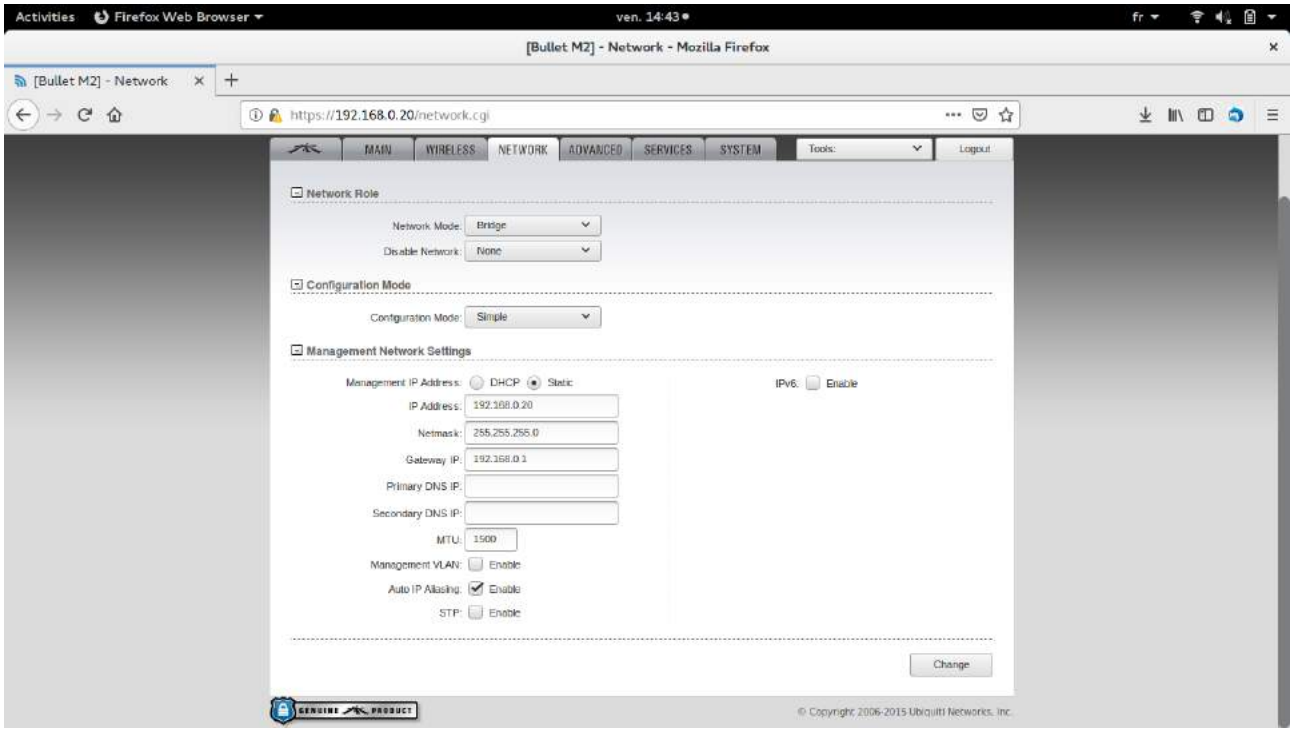
Do not forget to save changes at the bottom of the page, but you must apply the changes only at the end of the configuration process.

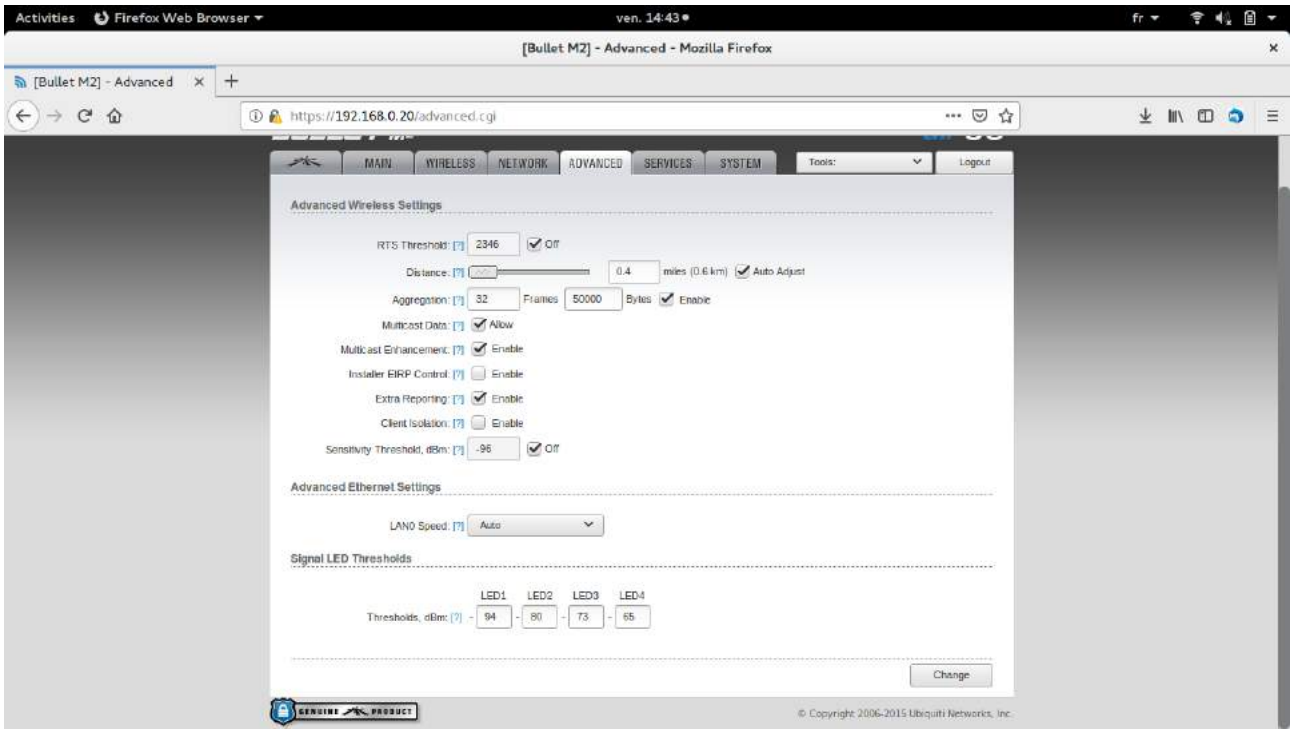
If you want to complete this step faster, you can directly load the file "XM-687251684A4D.cfg" at https://github.com/houdeval/cognac_regulation/blob/master/bullet_antenna_setting/XM-687251684A4D.cfg thanks to the "Tools" tab on the configuration page.

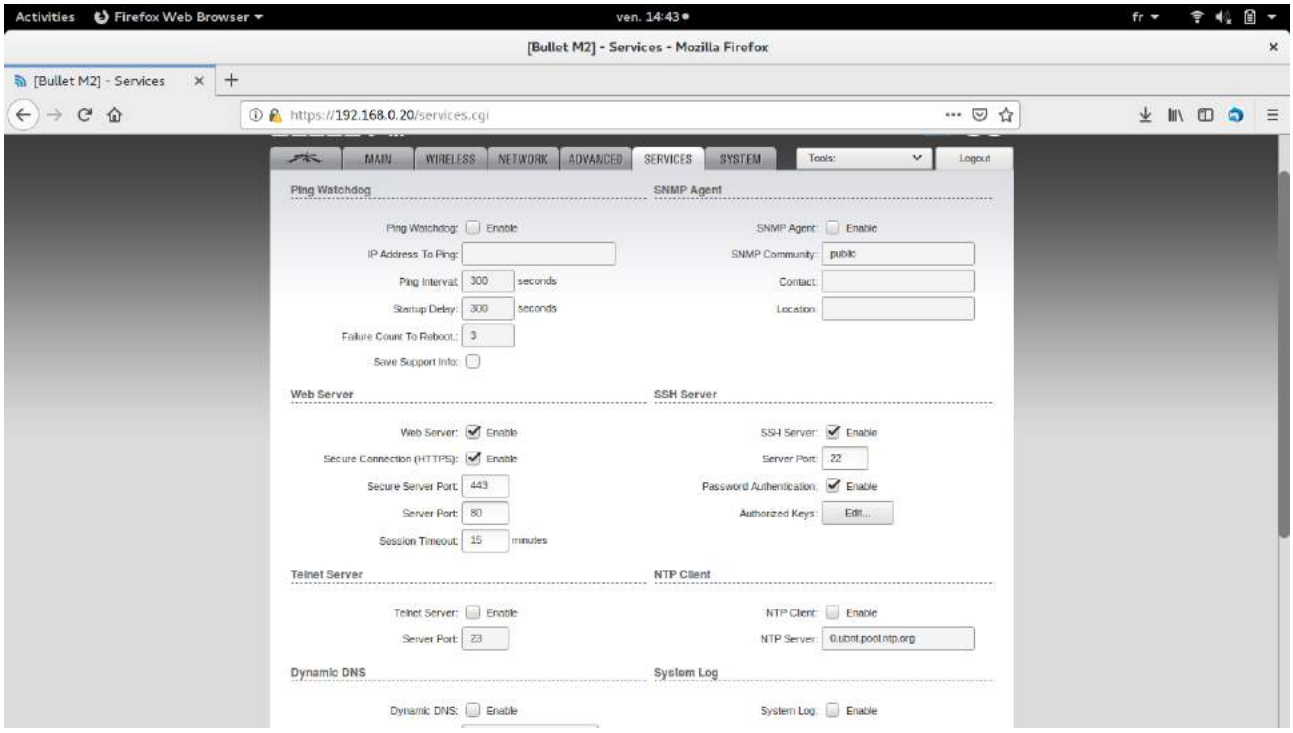


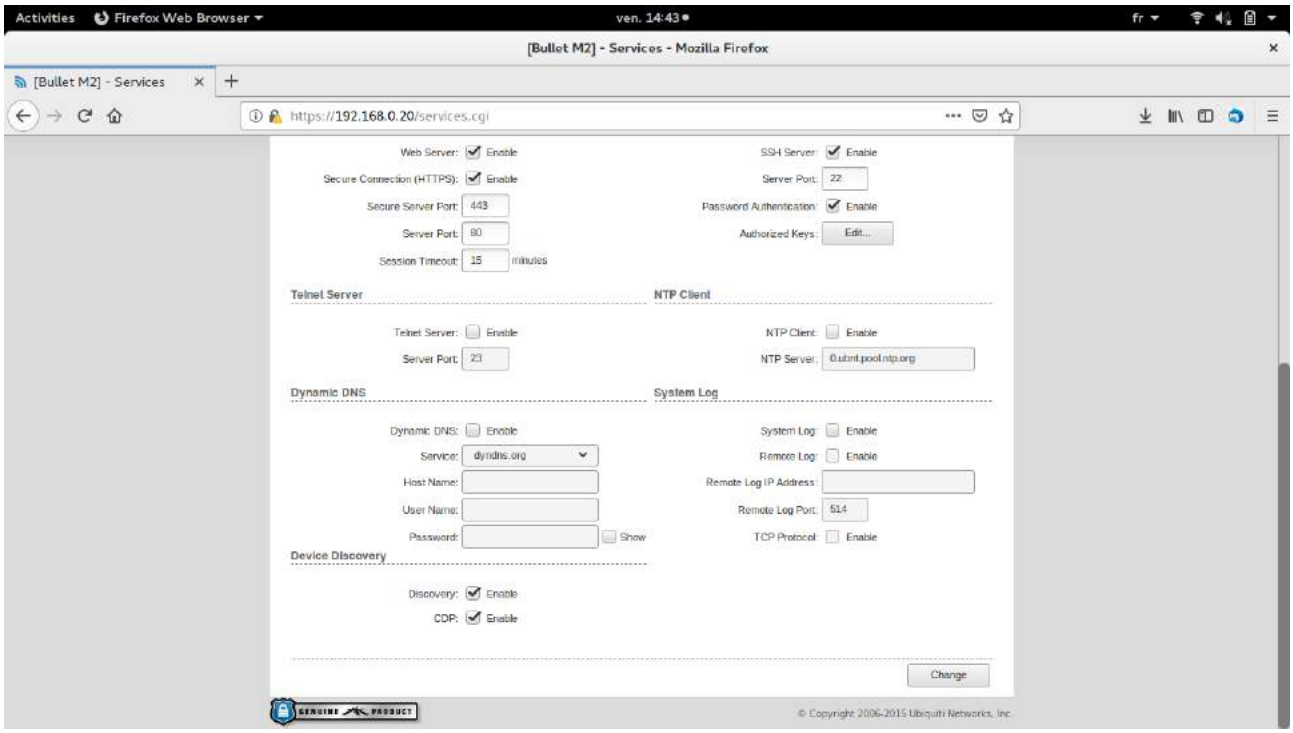

Activities
Firefox Web Browser
ven. 14:42
fr
[Bullet M2] - Main - Mozilla Firefox
[Bullet M2] - Main
https://192.168.0.20/index.cgi
BULLET M2
airOS
MAIN
WIRELESS
NETWORK
ADVANCED
SERVICES
SYSTEM
Tools:
Logout
Status
Device Model: Bullet M2
Device Name: Bullet M2
Network Mode: Bridge
Wireless Mode: Access Point
SSID: seabot
Security: none
Version: v5.6.2 (XM)
Uptime: 00:17:00
Date: 2015-07-16 12:17:52
Channel/Frequency: 5 / 2437 MHz
Channel Width: 20 MHz
Frequency Band: 2427 - 2447 MHz
Distance: 0.1 miles (0.2 km)
TX/RX Chains: 1X1
TX Power: 20 dBm
WLAN0 MAC: 68:72:51:68:4A:4D
LAN0 MAC: 68:72:51:69:4A:4D
LAN0: Unplugged
AP MAC: 68:72:51:68:4A:4D
Connections: 2
Noise Floor: -90 dBm
Transmit CCQ: 99.1 %
airMAX: Disabled
Monitor
Throughput | Stations | Interfaces | ARP Table | Bridge Table | Routes | Log
WLAN0
RX: 0bps
TX: 0bps

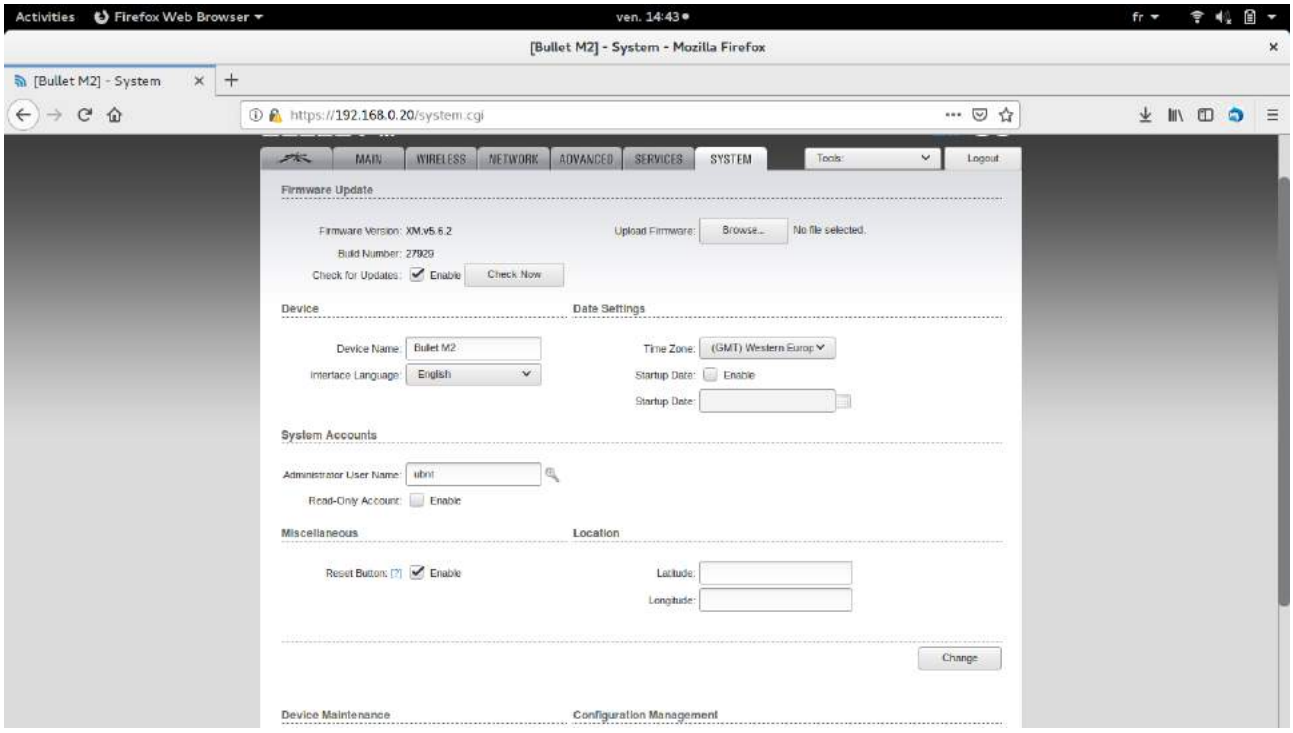

Activities
Firefox Web Browser
ven. 14:43
fr
[Bullet M2] - Wireless - Mozilla Firefox
[Bullet M2] - Wireless
https://192.168.0.20/link.cgi
Basic Wireless Settings
Wireless Mode: Access Point
WDS (Transparent Bridge Mode): Enable
SSID: seabot
Hide SSID
Country Code: France
IEEE 802.11 Mode: B/G/N mixed
Channel Width: 20 MHz
Frequency, MHz: 2437
Extension Channel: None
Frequency List, MHz: Enable
Calculate EIRP Limit: Enable
Antenna Gain: 0 dBi
Cable Loss: 0 dB
Output Power: 20 dBm
Data Rate Module: Default
Max TX Rate, Mbps: MCS 7 - 65/72.2
Auto
Wireless Security
Security: none
RADIUS MAC Authentication: Enable
MAC ACL: Enable
Change
GENUINE PRODUCT
© Copyright 2006-2015 Ubiquiti Networks, Inc.

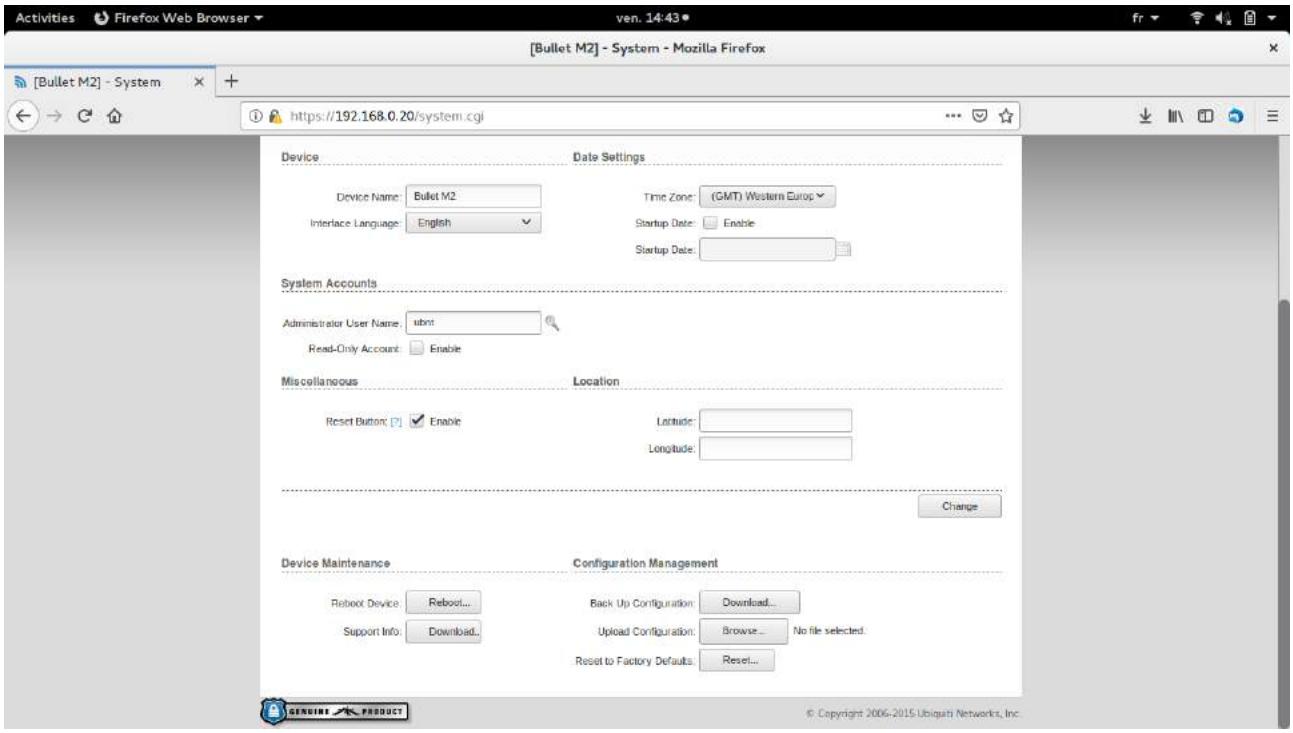

Activities
Firefox Web Browser
ven. 14:43
fr
[Bullet M2] - Network - Mozilla Firefox
[Bullet M2] - Network
https://192.168.0.20/network.cgi
MAIN
WIRELESS
NETWORK
ADVANCED
SERVICES
SYSTEM
Tools:
Logout
Network Role
Network Mode: Bridge
Disable Network: None
Configuration Mode
Configuration Mode: Simple
Management Network Settings
Management IP Address: DHCP Static
IP Address: 192.168.0.20
Netmask: 255.255.255.0
Gateway IP: 192.168.0.1
Primary DNS IP:
Secondary DNS IP:
MTU: 1500
Management VLAN: Enable
Auto IP Aliasing: Enable
STP: Enable
IPv6: Enable
Change
GENUINE PRODUCT
© Copyright 2006-2015 Ubiquiti Networks, Inc.


Activities
Firefox Web Browser
ven. 14:43
fr
[Bullet M2] - Advanced - Mozilla Firefox
[Bullet M2] - Advanced
https://192.168.0.20/advanced.cgi
MAIN
WIRELESS
NETWORK
ADVANCED
SERVICES
SYSTEM
Tools:
Logout
Advanced Wireless Settings
RTS Threshold: [?] 2346 Off
Distance: [?] 0.4 miles (0.6 km) Auto Adjust
Aggregation: [?] 32 Frames 50000 Bytes Enable
Multicast Data: [?] Allow
Multicast Enhancement: [?] Enable
Installer EIRP Control: [?] Enable
Extra Reporting: [?] Enable
Client Isolation: [?] Enable
Sensitivity Threshold, dBm: [?] -96 Off
Advanced Ethernet Settings
LAN0 Speed: [?] Auto
Signal LED Thresholds
LED1 LED2 LED3 LED4
Thresholds, dBm: [?] - 94 - 80 - 73 - 65
Change
GENUINE PRODUCT
© Copyright 2006-2015 Ubiquiti Networks, Inc.

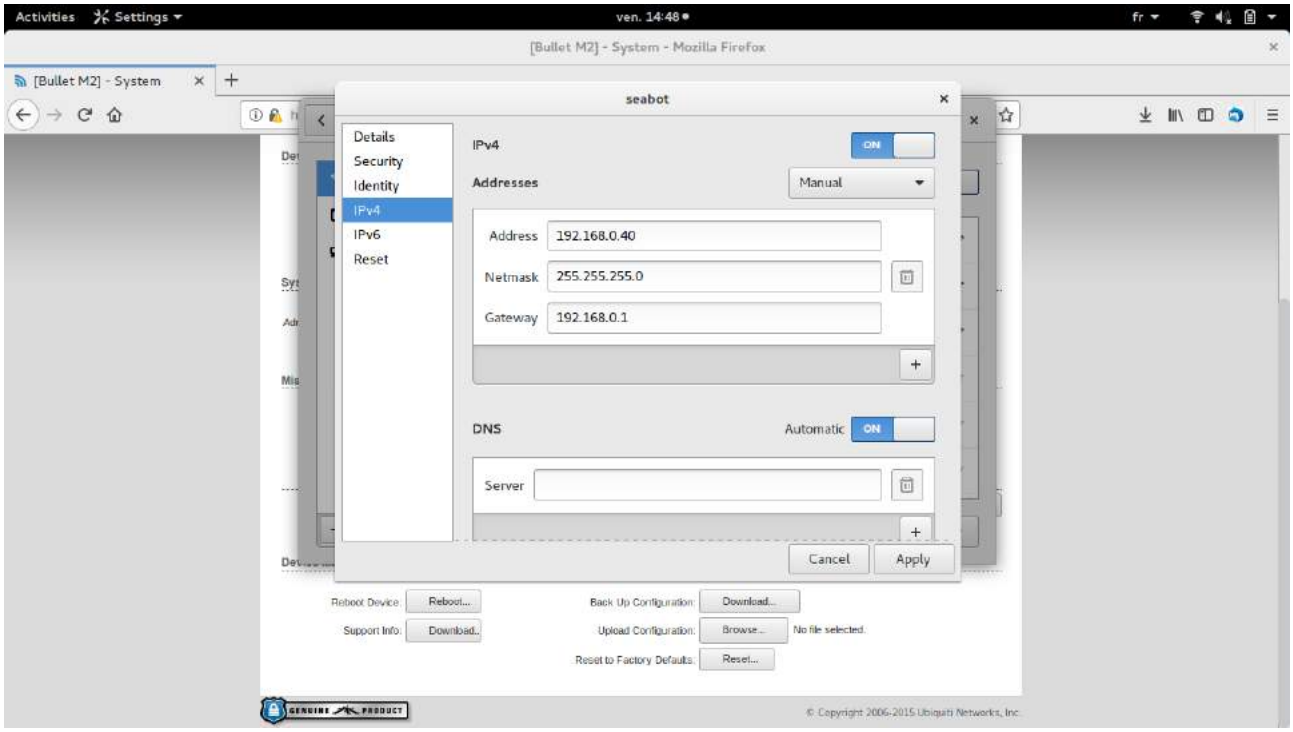

Activities
Firefox Web Browser
ven. 14:43
fr
[Bullet M2] - Services - Mozilla Firefox
[Bullet M2] - Services
https://192.168.0.20/services.cgi
MAIN
WIRELESS
NETWORK
ADVANCED
SERVICES
SYSTEM
Tools:
Logout
Ping Watchdog
Ping Watchdog: Enable
IP Address To Ping:
Ping Interval: 300 seconds
Startup Delay: 300 seconds
Failure Count To Reboot: 3
Save Support Info:
SNMP Agent
SNMP Agent: Enable
SNMP Community: public
Contact:
Location:
Web Server
Web Server: Enable
Secure Connection (HTTPS): Enable
Secure Server Port: 443
Server Port: 80
Session Timeout: 15 minutes
SSH Server
SSH Server: Enable
Server Port: 22
Password Authentication: Enable
Authorized Keys: Edit...
Telnet Server
Telnet Server: Enable
Server Port: 23
NTP Client
NTP Client: Enable
NTP Server: 0.ubnt.pool.ntp.org
Dynamic DNS
Dynamic DNS: Enable
System Log
System Log: Enable


Activities
Firefox Web Browser
ven. 14:43
fr
[Bullet M2] - Services - Mozilla Firefox
[Bullet M2] - Services
https://192.168.0.20/services.cgi
Web Server: Enable
Secure Connection (HTTPS): Enable
Secure Server Port: 443
Server Port: 80
Session Timeout: 15 minutes
SSH Server: Enable
Server Port: 22
Password Authentication: Enable
Authorized Keys: Edit...
Telnet Server
Telnet Server: Enable
Server Port: 23
NTP Client
NTP Client: Enable
NTP Server: 0.ubnt.pool.ntp.org
Dynamic DNS
Dynamic DNS: Enable
Service: dyndns.org
Host Name:
User Name:
Password: Show
System Log
System Log: Enable
Remote Log: Enable
Remote Log IP Address:
Remote Log Port: 514
TCP Protocol: Enable
Device Discovery
Discovery: Enable
CDP: Enable
Change
GENUINE PRODUCT
© Copyright 2006-2015 Ubiquiti Networks, Inc.





- Then, in such a configuration, after applying the modifications, you should be able to detect the WIFI network related to the antenna whose name was defined here as "seabot". You should not need to keep connecting by cables between the antenna and the computer. Set you computer network parameters as below to be able to connect to this new network. Note that the IP address 192.168.0.40 could be any address of the form 192.168.0.X except 192.168.0.0, 192.168.0.255, 192.168.0.20, 192.168.0.1 .

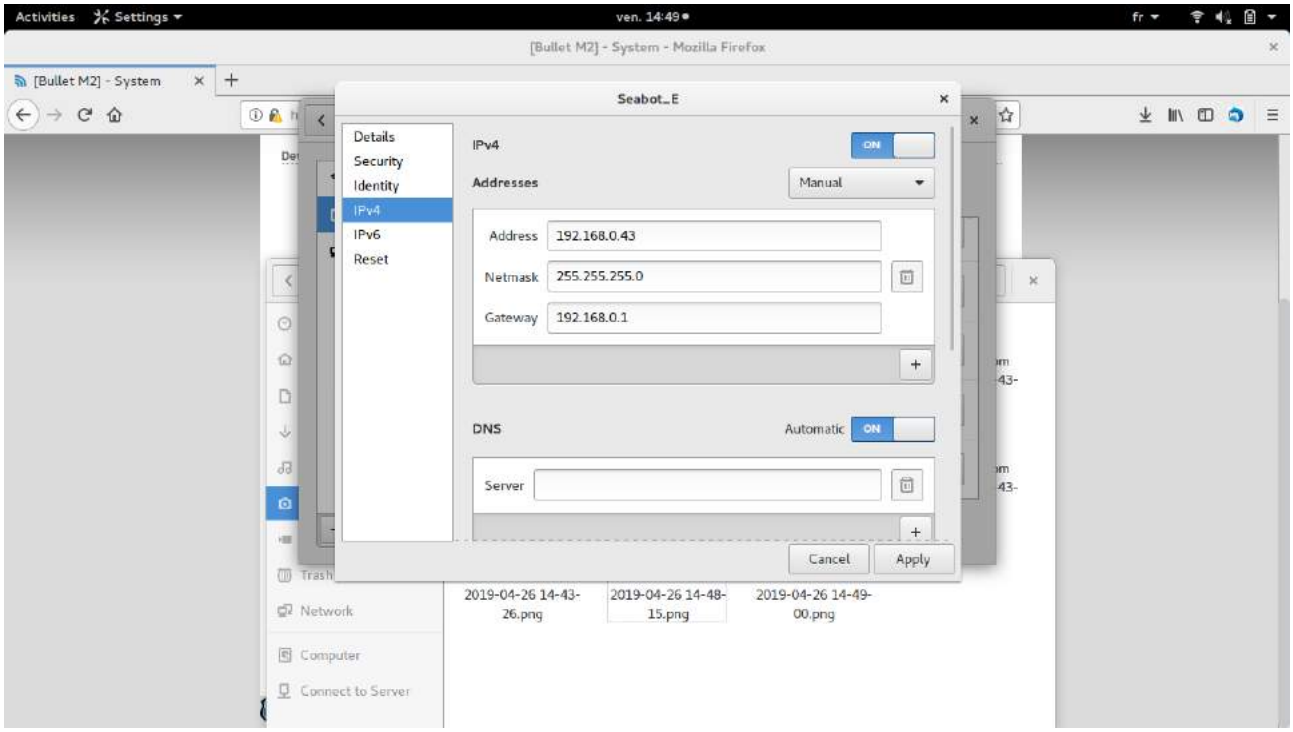


- Now, you should be able to connect to the antenna as a router. To check the connection, just try to write the address of the antenna (192.168.0.20) in your browser to access the configuration page. To connect the raspberry, set the same network configuration with a different IP address from the computer.

If you still face problems :

– Try to plug and unplug the different cables.

– Try to connect to the antenna with your computer by cables with this kind of configuration:



– Check whether you chose the right parameters.

– Check if the IP address class used is compatible with your Raspberry circuit board.

– Restart your computer.

– Reset the antenna and do this tutorial again from the beginning.

Cloning process to copy a raspberry image from a SD card to another one

Clone raspberry image from a SD card to another one

Installation process coming from the "thepihut" website (https://thepihut.com/blogs/raspberry-pi-t
17789160-backing-up-and-restoring-your-raspberry-pis-sd-card)

Using linux :

Before inserting the SD card into the reader on your Linux PC, run the following command to find out which devices are currently available:

```
df -h
```

Which will return something like this:

```
Filesystem 1K-blocks Used Available Use% Mounted on
rootfs 29834204 15679020 12892692 55% /
/dev/root 29834204 15679020 12892692 55% /
devtmpfs 437856 0 437856 0% /dev
tmpfs 88432 284 88148 1% /run
tmpfs 5120 0 5120 0% /run/lock
tmpfs 176860 0 176860 0% /run/shm
/dev/mmcblk0p1 57288 14752 42536 26% /boot
```

Insert the SD card into a card reader and use the same df -h command to find out what is now available:

```
Filesystem 1K-blocks Used Available Use% Mounted on
rootfs 29834204 15679020 12892692 55% /
/dev/root 29834204 15679020 12892692 55% /
devtmpfs 437856 0 437856 0% /dev
tmpfs 88432 284 88148 1% /run
tmpfs 5120 0 5120 0% /run/lock
tmpfs 176860 0 176860 0% /run/shm
/dev/mmcblk0p1 57288 14752 42536 26% /boot
/dev/sda5 57288 9920 47368 18% /media/boot
/dev/sda6 6420000 2549088 3526652 42% /media/41cd5baa-7a62-4706-b8e8-02c43ccee8d9
```

The new device that wasn't there last time is your SD card.

The left column gives the device name of your SD card, and will look like '/dev/mmcblk0p1' or '/dev/sdb1'. The last part ('p1' or '1') is the partition number, but you want to use the whole SD card, so you need to remove that part from the name leaving '/dev/mmcblk0' or

'/dev/sdb' as the disk you want to read from.

Open a terminal window and use the following to backup your SD card:

```
sudo dd if=/dev/sdb of=~/SDCardBackup.img
```

As on the Mac, the dd command does not show any feedback so you just need to wait until the command prompt re-appears.

To restore the image, do exactly the same again to discover which device is your SD card. As with the Mac, you need to unmount it first, but this time you need to use the partition number as well (the 'p1' or '1' after the device name). If there is more than one partition on the device, you will need to repeat the umount command for all partition numbers. For example, if the df -h shows that there are two partitions on the SD card, you will need to unmount both of them:

```
sudo umount /dev/sdb1
sudo umount /dev/sdb2
```

Now you are able to write the original image to the SD drive:

```
sudo dd bs=4M if=~/SDCardBackup.img of=/dev/sdb
```

The bs=4M option sets the 'block size' on the SD card to 4Meg. If you get any warnings, then change this to 1M instead, but that will take a little longer to write.

Again, wait while it completes. Before ejecting the SD card, make sure that your Linux PC has completed writing to it using the command:

```
sudo sync
```

User manual to set an AD-HOC connection between a raspberry pi 3 and another connected device

Creating an Ad-Hoc network in the Raspberry Pi to connect a PC to the Raspberry Pi directly

1. Save a copy of the file /etc/network/interfaces not to lose the current settings for other applications :

```
cp /etc/network/interfaces /etc/network/copy-interfaces
```

2. Then replace the file with the following content

```
nano /etc/network/interfaces

auto wlan0
iface wlan0 inet static
     address 192.168.0.13
     netmask 255.255.255.0
     wireless-channel 1
     wireless-essid seaboat
     wireless-mode ad-hoc
```

3. Reboot the Raspberry Pi.

4. The network will appear in the computer in the list of wireless networks. Connect to it.

5. Add a static IP address to the Mac/PC with 192.168.0.X/255.255.255.0 where X can belong to the interval [1, 254] except 13 (address dedicated to the Raspberry Pi). This step is necessary because there is no DHCP server in the Raspberry Pi.

6. Then you can ssh to the Raspberry Pi to pi@192.168.0.13 from your computer.

```
ssh pi@192.168.0.13
```

Source : https://www.raspberrypi.org/forums/viewtopic.php?f=29&t=39927

Bash commands to install the latest version of ROS kinetic for Ubuntu

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
    /etc/apt/sources.list.d/ros-latest.list'

sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
    C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

sudo apt-get update

sudo apt-get install ros-kinetic-desktop-full

apt-cache search ros-kinetic

sudo rosdep init
rosdep update

echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc

sudo apt install python-rosinstall python-rosinstall-generator python-wstool
    build-essential
```

Try the command "roscore" to check whether the installation is successful.

Source : http://wiki.ros.org/kinetic/Installation/Ubuntu

Recap form of the main ROS commands

ROS Indigo Cheatsheet

Filesystem Management Tools

rospack	A tool for inspecting packages.
rospack profile	Fixes path and pluginlib problems.
roscd	Change directory to a package.
rospd/rosd	Pushd equivalent for ROS.
rosls	Lists package or stack information.
rosed	Open requested ROS file in a text editor.
roscp	Copy a file from one place to another.
rosdep	Installs package system dependencies.
roswtf	Displays a errors and warnings about a running ROS system or launch file.
catkin_create_pkg	Creates a new ROS stack.
wstool	Manage many repos in workspace.
catkin_make	Builds a ROS catkin workspace.
rqt_dep	Displays package structure and dependencies.

Usage:

```
$ rospack find [package]
$ roscd [package[/subdir]]
$ rospd [package[/subdir] | +N | -N]
$ rosd
$ rosls [package[/subdir]]
$ rosed [package] [file]
$ roscp [package] [file] [destination]
$ rosdep install [package]
$ roswtf or roswtf [file]
$ catkin_create_pkg [package_name] [depend1]..[dependN]
$ wstool [init | set | update]
$ catkin_make
$ rqt_dep [options]
```

Start-up and Process Launch Tools

roscore

The basis nodes and programs for ROS-based systems. A roscore must be running for ROS nodes to communicate.

Usage:

```
$ roscore
```

rosrun

Runs a ROS package's executable with minimal typing.

Usage:

```
$ rosrun package_name executable_name
```

Example (runs turtlesim):

```
$ rosrun turtlesim turtlesim_node
```

roslaunch

Starts a roscore (if needed), local nodes, remote nodes via SSH, and sets parameter server parameters.

Examples:

Launch a file in a package:

```
$ roslaunch package_name file_name.launch
```

Launch on a different port:

```
$ roslaunch -p 1234 package_name file_name.launch
```

Launch on the local nodes:

```
$ roslaunch --local package_name file_name.launch
```

Logging Tools

rosbag

A set of tools for recording and playing back of ROS topics.

Commands:

rosbag record	Record a bag file with specified topics.
rosbag play	Play content of one or more bag files.
rosbag compress	Compress one or more bag files.
rosbag decompress	Decompress one or more bag files.
rosbag filter	Filter the contents of the bag.

Examples:

Record select topics:

```
$ rosbag record topic1 topic2
```

Replay all messages without waiting:

```
$ rosbag play -a demo_log.bag
```

Replay several bag files at once:

```
$ rosbag play demo1.bag demo2.bag
```

Introspection and Command Tools

rosmsg/rossrv

Displays Message/Service (msg/srv) data structure definitions.

Commands:

rosmsg show	Display the fields in the msg/srv.
rosmsg list	Display names of all msg/srv.
rosmsg md5	Display the msg/srv md5 sum.
rosmsg package	List all the msg/srv in a package.
rosmsg packages	List all packages containing the msg/srv.

Examples:

Display the Pose msg:

```
$ rosmsg show Pose
```

List the messages in the nav_msgs package:

```
$ rosmsg package nav_msgs
```

List the packages using sensor_msgs/CameraInfo:

```
$ rosmsg packages sensor_msgs/CameraInfo
```

rosnode

Displays debugging information about ROS nodes, including publications, subscriptions and connections.

Commands:

rosnode ping	Test connectivity to node.
rosnode list	List active nodes.
rosnode info	Print information about a node.
rosnode machine	List nodes running on a machine.
rosnode kill	Kill a running node.

Examples:

Kill all nodes:

```
$ rosnode kill -a
```

List nodes on a machine:

```
$ rosnode machine aqy.local
```

Ping all nodes:

```
$ rosnode ping --all
```

rostopic

A tool for displaying information about ROS topics, including publishers, subscribers, publishing rate, and messages.

Commands:

rostopic bw	Display bandwidth used by topic.
rostopic echo	Print messages to screen.
rostopic find	Find topics by type.
rostopic hz	Display publishing rate of topic.
rostopic info	Print information about an active topic.
rostopic list	List all published topics.
rostopic pub	Publish data to topic.
rostopic type	Print topic type.

Examples:

Publish hello at 10 Hz:

```
$ rostopic pub -r 10 /topic_name std_msgs/String hello
```

Clear the screen after each message is published:

```
$ rostopic echo -c /topic_name
```

Display messages that match a given Python expression:

```
$ rostopic echo --filter "m.data=='foo'" /topic_name
```

Pipe the output of rostopic to rosmsg to view the msg type:

```
$ rostopic type /topic_name | rosmsg show
```

rosparam

A tool for getting and setting ROS parameters on the parameter server using YAML-encoded files.

Commands:

rosparam set	Set a parameter.
rosparam get	Get a parameter.
rosparam load	Load parameters from a file.
rosparam dump	Dump parameters to a file.
rosparam delete	Delete a parameter.
rosparam list	List parameter names.

Examples:

List all the parameters in a namespace:

```
$ rosparam list /namespace
```

Setting a list with one as a string, integer, and float:

```
$ rosparam set /foo "['1', 1, 1.0]"
```

Dump only the parameters in a specific namespace to file:

```
$ rosparam dump dump.yaml /namespace
```

rosservice

A tool for listing and querying ROS services.

Commands:

rosservice list	Print information about active services.
rosservice node	Print name of node providing a service.
rosservice call	Call the service with the given args.
rosservice args	List the arguments of a service.
rosservice type	Print the service type.
rosservice uri	Print the service ROSRPC uri.
rosservice find	Find services by service type.

Examples:

Call a service from the command-line:

```
$ rosservice call /add_two_ints 1 2
```

Pipe the output of rosservice to rossrv to view the srv type:

```
$ rosservice type add_two_ints | rossrv show
```

Display all services of a particular type:

```
$ rosservice find rospy_tutorials/AddTwoInts
```

ROS Indigo Cheatsheet

Logging Tools

rqt_console

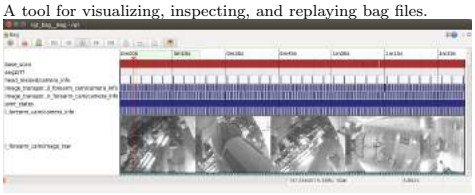
A tool to display and filtering messages published on rosout.

Usage:

```
$ rqt_console
```

rqt_bag

A tool for visualizing, inspecting, and replaying bag files.



Usage, viewing:

```
$ rqt_bag bag_file.bag
```

Usage, bagging:

```
$ rqt_bag *press the big red record button.*
```

rqt_logger_level

Change the logger level of ROS nodes. This will increase or decrease the information they log to the screen and rqt_console.

Usage:

```
viewing $ rqt_logger_level
```

Introspection & Command Tools

rqt_topic

A tool for viewing published topics in real time.

Usage:

```
$ rqt
Plugin Menu->Topic->Topic Monitor
```

rqt_msg, rqt_srv, and rqt_action

A tool for viewing available msgs, srvs, and actions.

Usage:

```
$ rqt
Plugin Menu->Topic->Message Type Browser
Plugin Menu->Service->Service Type Browser
Plugin Menu->Action->Action Type Browser
```

rqt_publisher, and rqt_service_caller

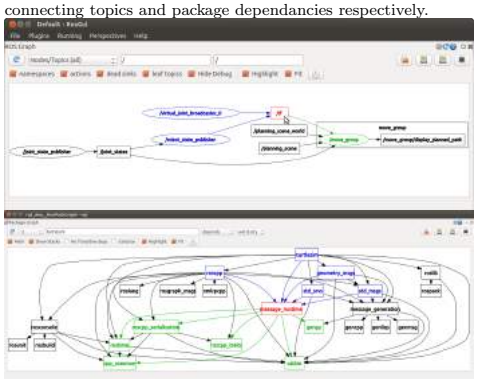
Tools for publishing messages and calling services.

Usage:

```
$ rqt
Plugin Menu->Topic->Message Publisher
Plugin Menu->Service->Service Caller
```

rqt_graph, and rqt_dep

Tools for displaying graphs of running ROS nodes with connecting topics and package dependancies respectively.



Usage:

```
$ rqt_graph
$ rqt_dep
```

rqt_top

A tool for ROS specific process monitoring.

Usage:

```
$ rqt
Plugin Menu->Introspection->Process Monitor
```

rqt_reconfigure

A tool for dynamically reconfiguring ROS parameters.

Usage:

```
$ rqt
Plugin Menu->Configuration->Dynamic Reconfigure
```

Development Environments

rqt_shell, and rqt_py_console

Two tools for accessing an xterm shell and python console respectively.

Usage:

```
$ rqt
Plugin Menu->Miscellaneous Tools->Shell
Plugin Menu->Miscellaneous Tools->Python Console
```

Data Visualization Tools

tf_echo

A tool that prints the information about a particular transformation between a source_frame and a target_frame.

Usage:

```
$ rosrun tf tf_echo <source_frame> <target_frame>
```

Examples:

To echo the transform between /map and /odom:

```
$ rosrun tf tf_echo /map /odom
```

view_frames

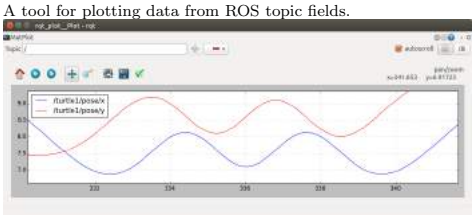
A tool for visualizing the full tree of coordinate transforms.

Usage:

```
$ rosrun tf2_tools view_frames.py
$ evince frames.pdf
```

rqt_plot

A tool for plotting data from ROS topic fields.



Examples:

To graph the data in different plots:

```
$ rqt_plot /topic1/field1 /topic2/field2
```

To graph the data all on the same plot:

```
$ rqt_plot /topic1/field1,/topic2/field2
```

To graph multiple fields of a message:

```
$ rqt_plot /topic1/field1:field2:field3
```

rqt_image_view

A tool to display image topics.

Usage:

```
$ rqt_image_view
```

ROS Indigo Catkin Workspaces

Create a catkin workspace

Setup and use a new catkin workspace from scratch.

Example:

```
$ source /opt/ros/hydro/setup.bash
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
```

Checkout an existing ROS package

Get a local copy of the code for an existing package and keep it up to date using wstool.

Examples:

```
$ cd ~/catkin_ws/src
$ wstool init
$ wstool set tutorials --git git://github.com/ros/ros_tutorials.git
$ wstool update
```

Create a new catkin ROS package

Create a new ROS catkin package in an existing workspace with catkin create package. After using this you will need to edit the CMakeLists.txt to detail how you want your package built and add information to your package.xml.

Usage:

```
$ catkin_create_pkg <package_name> [depend1] [depend2]
```

Example:

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg tutorials std_msgs rospy roscpp
```

Build all packages in a workspace

Use catkin_make to build all the packages in the workspace and then source the setup.bash to add the workspace to the ROS_PACKAGE_PATH.

Examples:

```
$ cd ~/catkin_ws
$ ~/catkin_make
$ source devel/setup.bash
```

Command calculation for the state feedback regulation

Float dynamics

Full dynamics

In this section, equations governing float vertical motions are adapted from P. Lherminier thesis (Annex 3, https://tel.archives-ouvertes.fr/tel-00881646) which is strongly inspired from Voorhis (1971). They may account for water motions. *We assume the float follows water horizontal motions perfectly. We ignore float rotation* which is less relevant when the float is not equipped with propellers and are solely concerned by float vertical motions. Key position variables are thus the following ones:

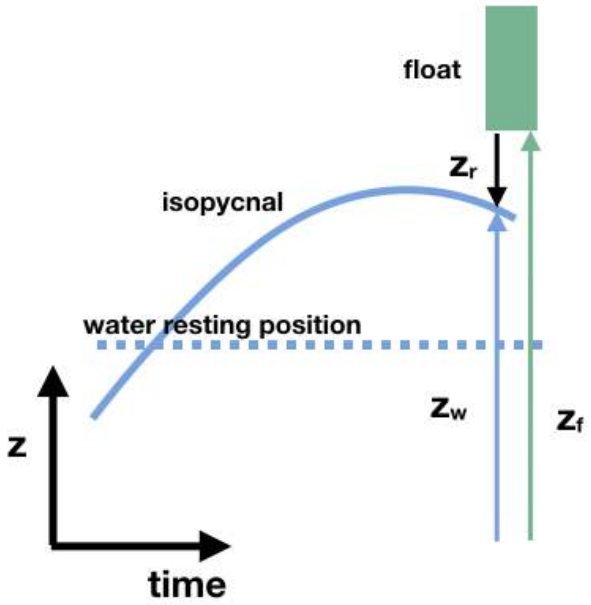
- z_f : float vertical position
- $z_w(z,t) = z + \eta_w(z,t)$: water vertical position around a resting level z
- $z_r = z_w - z_f$: water vertical position with respect to the float position

Corresponding velocities are: $w_f = dz_f/dt$, $w = dz_w/dt$, $w_r = w - w_f$

The dynamical equation ruling the float vertical displacement is:

$$m\frac{d^2 z_f}{dt^2} = m\frac{dw_f}{dt} = \mathcal{F}_w - mg, \tag{4.1}$$

where $\mathcal{F}_w$ represents the vertical component of induced by pressure and viscous forces on the float.



One contribution, labelled $\mathcal{F}_0$, of forces exerted on the float by water is assumed to

be given by pressure forces due to pressure fields in the absence of the float:

$$\mathcal{F}_0 = -V\partial_z p = -\frac{m}{\rho_f}\partial_z p \tag{4.2}$$

$$= \frac{mg\rho_w}{\rho_f} + m\frac{\rho_w}{\rho_f}\Big[\frac{Dw}{Dt} + w_r\partial_z w\Big], \tag{4.3}$$

where Dw/Dt is the vertical acceleration of water. Densities are given by the float and water equations of state:

$$\rho_w = \rho_w(p, T, S) \tag{4.4}$$

$$\rho_f = \rho_f(p, T) = \frac{m}{V(T, p) + v} \tag{4.5}$$

where V represents the bulk of the float volume that may be affected by pressure (p) and temperature (T) and v is the volume taken in and out by the pump/piston. *We assume next that the float is in thermal equilibrium with surrounding water*. Relaxing this assumption would require carrying an equation for the float thermal evolution and including sources and sinks.

The other contribution, labelled $\mathcal{F}'$ is due to the flow induced by float motions. It may be parametrized as:

$$\mathcal{F}' = \rho_w V a\frac{d^2 z_r}{dt^2} + \rho_w V c_d\frac{dz_r}{dt} \tag{4.6}$$

$$\sim m\Big[a\frac{d^2 z_r}{dt^2} + c_d\frac{dz_r}{dt}\Big], \tag{4.7}$$

with:

$$c_d = \Big(c_0 N + \frac{c_1}{2L}\Big|\frac{dz_r}{dt}\Big|\Big), \tag{4.8}$$

where a, $c_{0,1}$ are respectively added mass and drag parameters and of order 1 parameters, where N is the buoyancy frequency, and, where L is the float length. *Parameters a, $c_{0,1}$ need to be estimated experimentally* and are expected to vary depending float motions (e.g. history, nature, amplitude) and environmental conditions (stratification). The first time on the right hand-side represents an added mass term and others are damping like terms.

Combining (4.1), (4.3) and (4.7) leads to a forced equation for float vertical motions:

$$(1 + a)\frac{d^2 z_f}{dt^2} + c_d\frac{dz_f}{dt} = g\frac{\rho_w - \rho_f}{\rho_f} + c_d w + a\frac{dw}{dt} + \frac{\rho_w}{\rho_f}\Big[\frac{Dw}{Dt} + w_r\partial_z w\Big]. \tag{4.9}$$

Or in a more compact form:

$$\underbrace{(1 + a)\frac{d^2 z_f}{dt^2}}_{1} + \underbrace{c_d\frac{dz_f}{dt}}_{2} = \underbrace{\mathcal{F}_b}_{3} + \underbrace{\mathcal{F}_w}_{4} \tag{4.10}$$

Degree of knowledge of each variables involved in (4.10):

- ρ_w: in situ water density and uncertainty may be approximately known from climatology; its knowledge may be fine tuned with hydrographic data collection prior to a float deployment. Onboard measurements of temperature and pressure may improve its estimate (via T-S relationship).

- w, dw/dt - water motions. Unkown but of order 10^{-3}m/s and 10^{-8}m^2/s typically (10m isopycnal displacement over 5h)
- $\rho_f(T, P, v)$: float density. Could be measured prior to deployments. Uncertainties are associated with piston position knowledge (quantifiable) and float pressure compressibility and thermal expansion (unknown, measurable a priori?).
- a, $c_{0,1}$.

Regimes

Several regimes are possible :

- steady high speed: terms 3 (buoyancy) and 2 (drag) balance in (4.10). Water motions do not matter. Requires: $w_f/T \ll g(\rho_w - \rho_f)/\rho_f$ (T time scale of float motions), $w_f \gg w$
- unsteady buoyancy inequilibrium: terms 1 and 3, maybe 2, balance in (4.10). Water motions do not matter. Requires: $w_f/T \sim g(\rho_w - \rho_f)/\rho_f$, $w_f \gg w$
- buoyancy equilibrium: terms 1, 2 and 4 may balance in (4.10). Water motions may accelerate the float. Requires: $g(\rho_w - \rho_f)/\rho_f \ll c_d w_f, w_f/T$ and $w_f \gg w$; or $w_f \sim w$; or $g(\rho_w - \rho_f)/\rho_f \ll c_d w$

Control

We ignore water vertical velocity ($w = 0$), assume an added mass coefficient of 1 ($a = 1$) and a purely quadratic drag ($c_0 = 0$):

$$2m\frac{d^2z_f}{dt^2} + m\frac{c_1}{2L}\left|\frac{dz_f}{dt}\right|\frac{dz_f}{dt} = g\big[\rho_w(V + v) - m\big] \tag{4.11}$$

We introduce departure from reference values for water density and float volume: $V = V_0 + \delta V$, $\rho_w = \rho + \delta\rho_w$. The reference for water is with respect to $\rho = m/V_0$. Neglecting second order corrections leads to:

$$2m\frac{d^2z_f}{dt^2} + m\frac{c_1}{2L}\left|\frac{dz_f}{dt}\right|\frac{dz_f}{dt} = g\rho\big[V_0\delta\rho_w/\rho + \delta V + v\big] \tag{4.12}$$

Float's variations of volume due to pressure fluctuations are assumed to be approximately known:

$$\delta V = \gamma_V z. \tag{4.13}$$

Note that thermal expansion is neglected. Water density is linearized around the target level:

$$\delta\rho_w = \delta\rho_w(\overline{z}) + (z - \overline{z})\partial_z\rho_w \tag{4.14}$$

This leads to the following equation for dynamics:

$$2m\frac{d^2z_f}{dt^2} + m\frac{c_1}{2L}\left|\frac{dz_f}{dt}\right|\frac{dz_f}{dt} = g\rho(V_e + \gamma_e z + v), \tag{4.15}$$

where $V_e = V_0[\delta\rho_w(\overline{z}) - \overline{z}\partial_z\rho_w]/\rho$ and $\gamma_e = V_0\partial_z\rho_w/\rho + \gamma_V$.

Reformulation

The system state vector is given by:

$$\underline{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -dz_f/dt \\ -z_f \\ \gamma_e \\ V_e \end{pmatrix}$$

where the state variables are therefore the depth temporal rate of evolution (x_0, in m/s), the depth (x_1, in m), the equivalent compressibility of the float (γ_e, in m^2) and the error volume (x_3, in m^3).

The float evolution equation then becomes:

$$d\underline{x}/dt = \begin{pmatrix} dx_0/dt \\ dx_1/dt \\ dx_2/dt \\ dx_3/dt \end{pmatrix} = \begin{pmatrix} -d^2z_f/dt^2 \\ -dz_f/dt \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -A(x_3 - x_2x_1 + v) - B|x_0|x_0 \\ x_0 \\ 0 \\ 0 \end{pmatrix}, \tag{4.16}$$

where several parameters have been introduced: $A = g\rho/2m$ and $B = c_1/4L$.

Then the following function has been chosen to measure the depth error:

$$y = x_0 - \nu \arctan\frac{\overline{x}_1 - x_1}{\delta}, \tag{4.17}$$

where $\overline{x}_1$ is the targeted depth, ν is the typical vertical velocity the float should move at, and δ is a length scale that defines the zone of influence around the targeted depth.

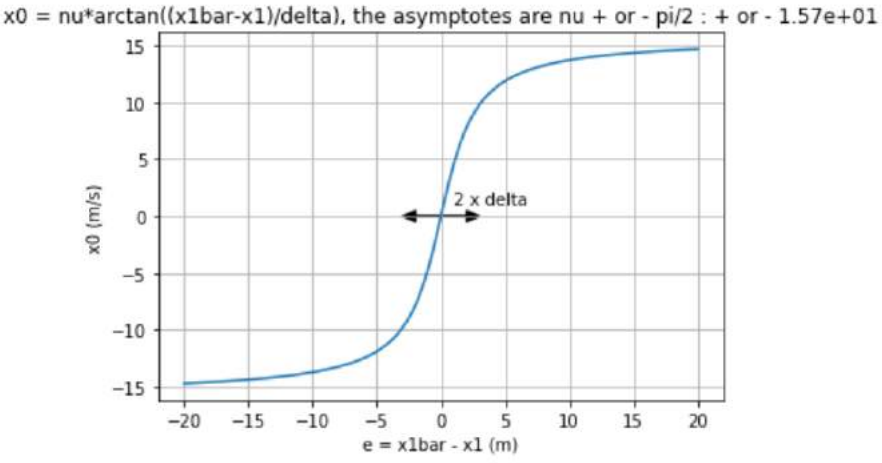


Figure 4.21: Graph of the function chosen to measure the depth error for the feedback regulation

We are looking for λ_1 $(1/s)$, λ_2 $(1/s^2)$ and the command u such that y is the solution of:

$$\ddot{y} + \lambda_1 \dot{y} + \lambda_2 y = 0. \tag{4.18}$$

For an exponential law of the type $y \sim e^{-rt}$ to be solution of the latter equations, we may choose: $\lambda_1 = 2r$, $\lambda_2 = r^2$. For $r = 0.1$ s^{-1}: $\lambda_1 = 0.2$ s^{-1}, $\lambda_2 = 0.01$ s^{-2}.

This choice of y is such that when $t \to \infty$: $x_0 = \nu \arctan \frac{\overline{x}_1 - x_1}{\delta}$. This means that $x_0 \to 0$ when $|x_1 - \overline{x}_1| < \delta$. The float vertical velocity should also never be greater that $\pm\nu \times \pi/2$.

Differentiating (4.17) with respect to time leads:

$$\dot{y} = \dot{x}_0 + \nu \frac{\dot{x}_1/\delta}{1 + e^2/\delta^2}, \tag{4.19}$$

$$= \dot{x}_0 + \nu \frac{x_0/\delta}{D}, \tag{4.20}$$

where $e = \overline{x_1} - x_1$ and $D = 1 + e^2/\delta^2$. Note that time derivatives of $\overline{x}_1$ are assumed to be small.

A second derivative leads to:

$$\ddot{y} = \ddot{x}_0 + \frac{\nu}{\delta} \frac{\dot{x}_0 D - x_0 \dot{D}}{D^2}, \tag{4.21}$$

$$= \ddot{x}_0 + \frac{\nu}{\delta} \frac{\dot{x}_0 D + 2e x_0^2/\delta^2}{D^2}, \tag{4.22}$$

where we have used the fact that $\dot{D} = 2e\dot{e}/\delta^2 = -2ex_1/\delta^2$

From (4.16):

$$\dot{x}_0 = -A(x_3 - x_2 x_1) - B|x_0|x_0, \tag{4.23}$$

$$\ddot{x}_0 = -A(u - x_2 x_0) - B\frac{d}{dt}\Big(|x_0|x_0\Big). \tag{4.24}$$

With (4.22), this leads to:

$$\ddot{y} = -A(u - x_2 x_0) - B\frac{d}{dt}\Big(|x_0|x_0\Big) + \frac{\nu}{\delta} \frac{\dot{x}_0 D + 2e x_0^2/\delta^2}{D^2}. \tag{4.25}$$

The following expression is derived for u:

$$u = -\frac{1}{A}\Big[\ddot{y} - (A x_2 x_0 - B\frac{d}{dt}\Big(|x_0|x_0\Big) + \frac{\nu}{\delta} \frac{\dot{x}_0 D + 2e x_0^2/\delta^2}{D^2})\Big] \tag{4.26}$$

Plugging in (4.18) hence the command law:

$$u = -\frac{1}{A}\Big[- \lambda_1 \dot{y} - \lambda_2 y - (A x_2 x_0 - B\frac{d}{dt}\Big(|x_0|x_0\Big) + \frac{\nu}{\delta} \frac{\dot{x}_0 D + 2e x_0^2/\delta^2}{D^2})\Big], \tag{4.27}$$

$$= \frac{1}{A}\Big[\lambda_1 \dot{y} + \lambda_2 y + \frac{\nu}{\delta} \frac{\dot{x}_0 D + 2e x_0^2/\delta^2}{D^2} - B\frac{d}{dt}\Big(|x_0|x_0\Big)\Big] + x_2 x_0. \tag{4.28}$$

Bounds on velocity and accelerations

- Vertical position: We want z_f to be within $[-500\text{m}, 0]$
- Speed: we need first an estimate of the maximum buoyancy force that can exerted on the float. The adjustable volume is such that the density of the float can be that of the water at the surface and at -500m. This leads to the following bound on the float velocity assuming a balance between term 2 and 3 in (4.10):

$$\begin{aligned}\max|w_f| &= \sqrt{2Lg/c_1 \times \max\left|1 - \frac{\rho_w}{\rho_f}\right|}, \\ &= \sqrt{2Lg/c_1 \times \left(1 - \frac{\rho_w(0)}{\rho_w(-500m)}\right)}.\end{aligned}$$

  Acceleration: the maximum acceleration is given by the maximum buoyancy force that may be applied on the float:

$$\max\left|\frac{dw_f}{dt}\right| = g/(1+a) \times \max\left|1 - \frac{\rho_w}{\rho_f}\right|$$

General formulation of the Kalman filter

General formulation of the Kalman filter:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \hat{\mathbf{u}}_k + \boldsymbol{\alpha}_k, \quad \text{predicted estimation } [N_x] \tag{4.29}$$

$$\hat{\mathbf{\Gamma}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{\Gamma}}_{k|k} \mathbf{A}_k^T + \mathbf{\Gamma}_{\alpha_k}, \quad \text{predicted covariance } [N_x \times N_x] \tag{4.30}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k, \quad \text{corrected estimation } [N_x] \tag{4.31}$$

$$\hat{\mathbf{\Gamma}}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \hat{\mathbf{\Gamma}}_{k|k-1}, \quad \text{corrected covariance } [N_x \times N_x] \tag{4.32}$$

$$\tilde{\mathbf{y}}_k = \mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1}, \quad \text{innovation } [N_h] \tag{4.33}$$

$$\mathbf{S}_k = \mathbf{C}_k \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^T + \mathbf{\Gamma}_{\beta_k}, \quad \text{innovation covariance } [N_h \times N_h] \tag{4.34}$$

$$\mathbf{K}_k = \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^T \mathbf{S}_k^{-1}, \quad \text{Kalman gain } [N_x \times N_h] \tag{4.35}$$

Observations of the system are the float depth (x_1): $\mathbf{y} = (x_1)$.

The control vector is: $\hat{\mathbf{u}}_k = (-Av, 0, 0, 0)$.

The model is given by:

$$\mathbf{A}_k = \mathbf{I} + dt \times \begin{bmatrix} -B|x_0| & Ax_2 & Ax_1 & -A \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.36}$$

Observations operator is given by:

$$\mathbf{C}_k = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.37}$$

Model error is assumed to be given by:

$$\mathbf{\Gamma}_{\alpha_k} = dt^2 \times \begin{bmatrix} e^2_{acceleration} & 0 & 0 & 0 \\ 0 & e^2_{velocity} & 0 & 0 \\ 0 & 0 & e^2_{\gamma_e} & 0 \\ 0 & 0 & 0 & e^2_{V_e} \end{bmatrix}, \tag{4.38}$$

where e_{γ_e} and e_{V_e} have respectively units of compressibility over time and volume over time.

If temporal variations of γ_e and V_e are assumed to be related to float vertical motions (they could also be related to water motions), we may estimate the amplitude of these temporal variations with:

$$\frac{d\gamma_e}{dt} = -\frac{dz_f}{dt} \times \frac{d\gamma_e}{dz}, \tag{4.39}$$

$$\frac{dV_e}{dt} = -\frac{dz_f}{dt} \times \frac{dV_e}{dz}. \tag{4.40}$$

This leads to the following choices for e_{γ_e} and e_{V_e}:

$$e_{\gamma_e} = W_f \left| \frac{d\gamma_e}{dz} \right| \tag{4.41}$$

$$e_{V_e} = W_f \left| \frac{dV_e}{dz} \right|, \tag{4.42}$$

where W_f is the typical vertical velocity of the float and may be related to the ν parameter of the state feedback (e.g. $W_f = \nu\pi/2$). γ_e and V_e and their derivatives may be estimated from water profile properties and/or a higher order knowledge of the float compressibility.

Observation noise matrix is assumed to be:

$$\mathbf{\Gamma}_\beta = \left[e_z^2\right] \tag{4.43}$$

Computation of the functions to choose the most suitable regulation parameters

Let's consider a basic dynamical model. We assume that the float is only subjected to its weight and buoyancy forces according to Archimedes' principle :

$$m(1+a)\frac{dv_f}{dt} = \frac{\rho_w - \rho_f}{\rho_f}\ g \tag{4.44}$$

It can be developed as :

$$m(1+a)\frac{dv_f}{dt} = (\rho_w(V+v) - m)\ g \tag{4.45}$$

where g is the acceleration of gravity, v_f is the downward velocity, ρ_f is the float density (assumed to be constant here as low variations of pressure and temperature are supposed), m its mass, a its added mass, ρ_w the water density, V the bulk of the float volume that may be affected by pressure and temperature and v is the volume taken in and out by the piston, v is supposed to depend on the time and to be equal to ut where u is the piston volume rate of change (the piston is supposed to leave the cylinder of the float with a constant velocity u).

At the equilibrium (t=0), the float is supposed to be motionless, that is to say the weight of the float and its buoyancy force associated to the volume V must balance each other out, hence :

$$m(1+a)\frac{dv_f}{dt} = \rho_w v\ g = \rho_w ut\ g \tag{4.46}$$

$$\tag{4.47}$$

Integrating twice these equations allows to express the velocity (v_f) and the depth (z_f) of the float as functions of the time with the following initial conditions :

$$\begin{cases} t = 0 : v_f(0) = 0 \\ t = 0 : z_f(0) = 0 \end{cases}$$

Hence :

$$v_f(t) = \frac{\rho_w u\ gt^2}{2m(1+a)} \tag{4.48}$$

$$z_f(t) = \frac{\rho_w u\ gt^3}{6m(1+a)} \tag{4.49}$$

In order to assess the appropriate time and distance parameters $1/r$ and δ, it will be useful to create a function t_ν that computes the necessary time for the float to reach a given speed ν and another function z_ν computing the necessary depth for the float to reach the same given speed ν.

These functions are :

$$t_\nu(\nu) = \sqrt{\frac{2m(1+a)\nu}{\rho_w u\ g}} \tag{4.50}$$

$$z_\nu(\nu) = z_f(t_\nu(\nu)) = \frac{\rho_w u\ g(\sqrt{\frac{2m(1+a)\nu}{\rho_w u\ g}})^3}{6m(1+a)} \tag{4.51}$$

Ballasting procedure of the Ifremer float

A criterion had to be chosen to avoid buoyancy issues in the float characteristics, indeed on one hand, if the float is too heavy, it will be difficult for it to be stable at the surface and it will require much time to resurface after having sunk. In addition, to reach its equilibrium position at the surface, it will have to put a large part of the length of the piston out, thus if the float is used in a water with a different density than the water in which it has been ballasted, it may be even denser in comparison with water and it may never reach its equilibrium position at the surface.
On the other hand, if the float is too light, it will have to keep a larger part of the piston inside itself so as to leave its equilibrium position at the surface, this means the float will have less available length to keep the rest of the piston inside itself in comparison with the length already inside at the surface. As a consequence, a small available variation of volume inside the float directly restricts the maximum available depth reachable by the float. Thus, so as to correctly set the buoyancy of the float, it was decided to set the equilibrium position such as the larger part of the piston plus 25 % of its thinner part are completely inside the float. This choice will allow the float to have enough buoyancy reserve to set its buoyancy at its equilibrium position by itself without being in the borderline cases described before.

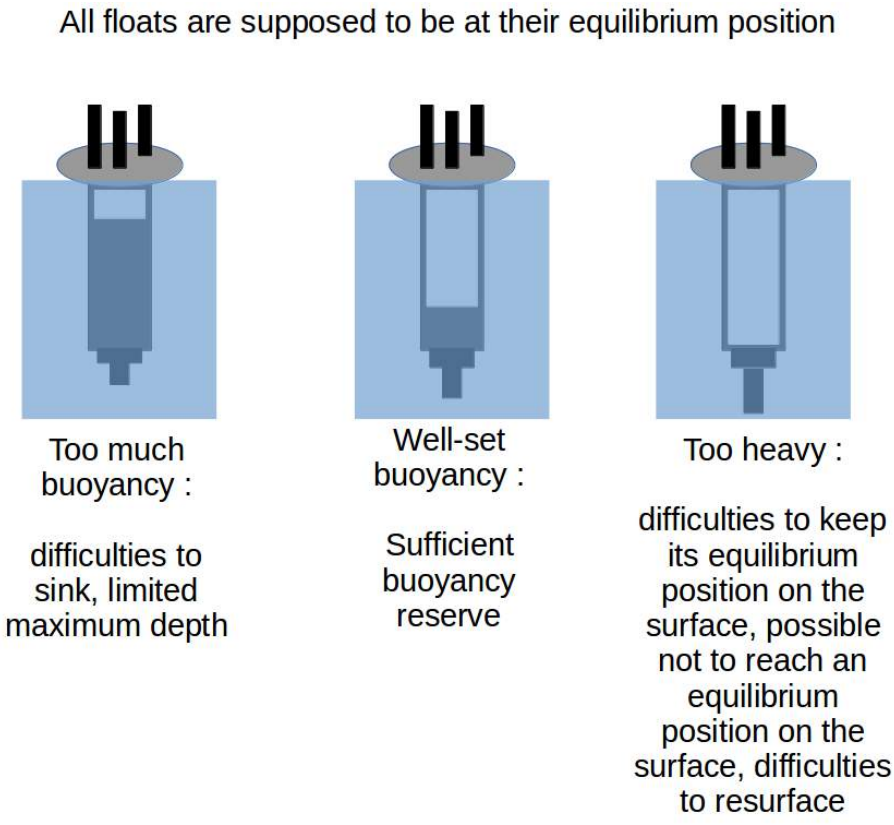


Figure 4.22: Recap of the different cases of ballasting

In practice, to ballast the float, the float was dropped in a small tank filled with seawater after having set the piston position as defined before : the whole larger part of the piston and 25 % of the thinner part completely inside the float. Then, small ballasts were added to the float until the float reaches its equilibrium position, that is to say the transition

position between the sinking and the resurfacing phase. Once the necessary weight to reach this position was estimated, the float was wrapped with a corresponding mass of lead sheets.

Figure 4.23: Ballasting installation

When the piston is completely out in such a configuration, the different antennas are completely out of the water to be as efficient as possible as it can be seen on the picture below.

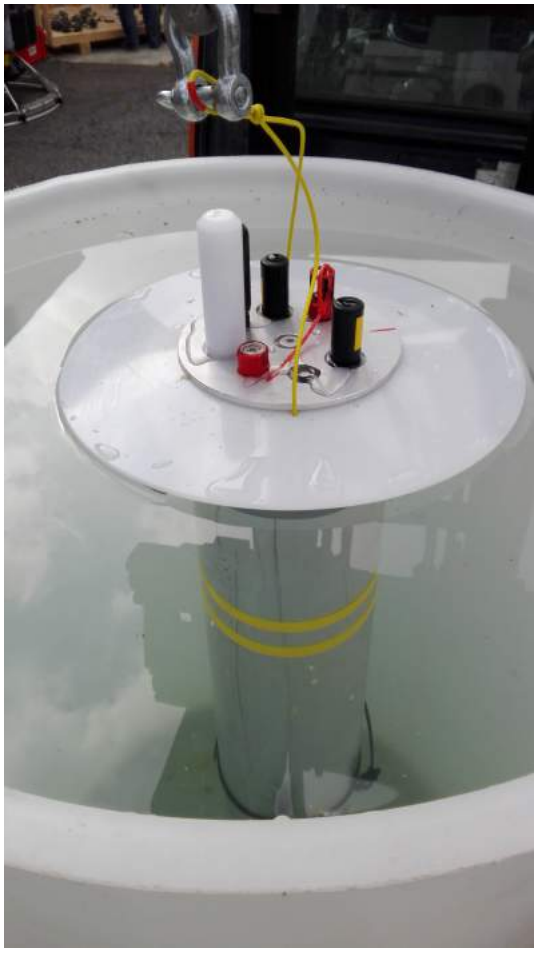


Figure 4.24: Ballasted float

Something important to mention is that the float was originally supposed to comprise 32 batteries, however with such a number of batteries, the float is too heavy and does not manage to float, thus 16 batteries have been removed and the tests with the Ifremer float were performed with only 16 batteries.

Leak detection thanks to a test with helium

This test consists in removing the air from the float and approaching a probe near the area where the leak is supposed to be. Then, a command allows to release helium from the probe while a mass spectrometer [31] counts the particles of helium in the area. If there is no leak, the number of particles will greatly increase in the area whereas if the probe is near the leak, the particles will be absorbed by the float so as to fill the vacuum produced, thus the number of particles will decrease steadily in this area as it can be seen on the pictures below.

Figure 4.25: Installation of the mass spectrometer to detect the leak in the float

Figure 4.26: Before the detection of the leak

Figure 4.27: After the detection of the leak

The test concluded that the leak came from the external temperature sensor, thus the sensor was changed to solve the problem.

Gantt chart of my internship

Nom de la tâche
avr.
mai
juin
juil.
août
avr. 1
avr. 8
avr. 15
avr. 22
avr. 29
mai 6
mai 13
mai 20
mai 27
juin 3
juin 10
juin 17
juin 24
juil. 1
juil. 8
juil. 15
juil. 22
juil. 29
août 5
août 12
août 19
août 26
1 Theoretical development
2 Establishing simple modeling of the float through mechanical equations
3 Implementation of a feedback regulator on the simulator and analysis of the regulation parameters
4 Implementation of a feedback regulator with Kalman filter on the simulator and analysis of the Kalman parameters
5 Software development
6 Deep study of the ENSTA float software architecture
7 Implementing the driver of the external temperature sensor
8 Implementing the driver of the external pressure sensor
9 Implementing the driver of the piston
10 Implementing the driver to get data from the batteries
11 Keeping progressing with the software development on the Ifremer float
12 Practical tests
13 ENSTA float tests in the pool of Ifremer
14 Test expedition from the Ifremer complex of La Seyne-sur-Mer
15 Test ENSTA float at Le Trez Hir
16 Ballasting tests for the Ifremer float in a small container filled with water
17 First test session in the pool for the Ifremer float
18 Ifremer float tests in the pool of Ifremer 2
19 Ifremer float tests in the pool of Ifremer 3
20 Miscellaneous
21 Learning session to know how to use the ENSTA float
22 Configuration of the bullet antenna
23 Analysis of the safety mode on the ENSTA float
24 Meeting with Thomas to talk about the results obtained from the test expedition
25 Meeting with Thomas to present my results about the choice of regulation parameters for feedback regulation
26 Fixing the Ifremer float after noting a leak
27 Project management
28 Meeting with the team of the COGNAC project 1
29 Meeting with the team of the COGNAC project 2
30 Meeting with the team of the COGNAC project 3
31 Teaching the use of ROS to the team of the COGNAC project
32 Meeting with the team of the COGNAC project 4
33 Meeting with the team of the COGNAC project 5
34 Writing documents about the Ifremer float to keep a record of my internship for the team of the project
Establishing simple modeling of the float through mechanical equations
Implementation
Implementa
Deep study of the ENSTA float software architecture
Implementing the driver of the external temperature sensor
Implementing the driver of the external pressure sensor
Implementing the driver of the piston
Implementing the driver to get data from the batteries
Keeping pr
ENSTA float tests in the pool of Ifremer
Test expedition from the Ifremer complex of La Seyne-sur-Mer
Test ENSTA float at Le Trez Hir
Ballasting tests for the Ifremer float in a small container filled with wa
First test session in the pool for the Ifremer float
Ifremer float tests in the pool of Ifremer 2
Ifremer float tests in the pool of Ifremer 3
Learning session to know how to use the ENSTA float
Configuration of the bullet antenna
Analysis of the safety mode on the ENSTA float
Meeting with Thomas to talk about the results obtained from the test expedition
Meeting with Thomas to present my results about the choice of regulation parameters for feedback regulation
Fixing the Ifremer float after noting a leak
Meeting with the team of the COGNAC project 1
Meeting with the team of the COGNAC project 2
Meeting with the team of the COGNAC project 3
Teaching the use of ROS to the team of the COGNAC project
Meeting with the team of the COGNAC project 4
Meeting with the team of the COGNAC project 5