

Design and implementation of a remotely operated vehicle for marine exploration and archaeological research

Treball Final de Grau



Facultat de Nàutica de Barcelona
Universitat Politècnica de Catalunya

Treball realitzat per:
Xavier Pagà Pagà
Christopher Rodríguez Conde

Dirigit per:
Jordi Fonollosa Magrinyà
Sergio I. Velásquez Correa

Grau en Enginyeria en Sistemes i Tecnologia Naval

Barcelona, 5 d'Octubre de 2019

Departaments de CEN i ESAII



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat de Nàutica de Barcelona



Resum

Aquest projecte aporta el disseny i la construcció d'un vehicle de tipus submarí per tal de rastrejar o visualitzar el fons marí amb fins arqueològics, d'inspecció i de presa de dades en zones marines sensibles. El ROV (Vehicle Operat Remotament) està pensat per ser dotat amb un sistema electrònic a bord per tal d'aconseguir tals funcions. El projecte ha estat desenvolupat amb criteris de disseny naval per tal d'assolir l'acompliment dels propòsits a sota l'aigua, garantint la integritat del vehicle i dels sensors instal·lats. Tot i haver dissenyat un bon sistema d'estanqueïtat, durant la fase d'assemblatge, no s'ha pogut assolir del tot. Això s'explica perquè durant la fase de proves i un cop submergit, el tub filtrava aigua. Això es podria solucionar en etapes posteriors afegint silicona resistent a l'aigua. El document conté els plànols generats de forma clara i concisa. A més a més, s'ha creat una web amb una base de dades, constituïda per diferents ordinadors, que connectats entre sí poden comunicar-se, i a través de la presa de dades amb els diferents dispositius electrònics generen aquesta base de dades. També s'ha aconseguit fer funcionar els diferents propulsors amb un comandament de PlayStation. Aquest control, mitjançant la interfície gràfica, també creada explícitament per aquest projecte, permet visualitzar en pantalla les diferents tasques que realitza el dispositiu. Ja sigui a través de la càmera que envia senyals a temps real o mitjançant la petita icona, una miniatura del model 3D del mateix ROV. Per motius econòmics, el prototip ha estat fabricat amb materials de baix cost com és el PVC, permetent així assolir la totalitat en quant a construcció es refereix, però de totes formes no s'ha desaprofitat cap recurs existent per tal d'evitar el deteriorament de l'entorn.

Aquest projecte persegueix el format d'accés obert, significat això que tots els dibuixos i la informació necessària i desenvolupada en aquest projecte són accessibles per tal d'aconseguir el producte que es presenta a continuació. Els arxius necessaris i tot el codi desenvolupat es troben tant en aquest mateix document com al següent enllaç: <https://github.com/crodriguezconde/finaldegreeproject>

Abstract

This project contributes to the design and construction of a submarine type vehicle in order to track or visualize the seabed for archaeological, inspection and data purposes in sensitive marine areas. The ROV (Remote-Operated Vehicle) is designed to be equipped with an on-board electronic system in order to achieve such functions. The project has been developed with naval design criteria to achieve the fulfillment of the purposes underwater, guaranteeing the integrity of the vehicle and the installed sensors. Despite having designed a good watertight system, during the assembling phase, it has not been fully achieved. This is explained because during the testing phase and when submerged, the tube leaked water. This could be solved in later stages by adding water-resistant silicone. The document contains the plans generated concisely. In addition, a database-based web has been created. This database is made up of the different computers, which can be connected to each other, and through the collection of data with the different electronic devices they generate this database. It has also been possible to operate the different propulsors with a PlayStation command. This control, through the graphical interface, also explicitly created for this project, allows to visualize on screen the different tasks carried out by the device. Either through the camera that sends signals in real time or through the small icon, a miniature of the 3D model of the same ROV. For economic reasons, the prototype has been manufactured with materials of low cost such as PVC, thus allowing the totality in terms of construction refers, but in any case, no existing resource has been used to avoid deterioration of the surroundings.

This project pursues the open access format, meaning that all the drawings and information necessary and developed in this project are accessible in order to achieve the product that is presented below. The necessary files and all the code developed are found in this document as well as in the following link: <https://github.com/crodriguezconde/finaldegreeproject>

Contents

RESUM	III
ABSTRACT	IV
CONTENTS	V
LIST OF FIGURES	IX
LIST OF TABLES	XIII
CHAPTER 1. INTRODUCTION	1
1.1 HISTORICAL CONTEXT	1
1.2 TECHNICAL CONTEXT: TYPOLOGIES	2
CHAPTER 2. OBJECTIVES	5
2.1 MAIN OBJECTIVE	5
2.2 SECONDARY OBJECTIVES	5
1. DESIGN OF THE ROV ITSELF AND ELECTRONIC DEVICES	5
2. DEVICES IMPLEMENTATION ON THE ROV	5
3. MECHANICAL WORKSHOP TO CREATE DIFFERENT NEEDED PARTS AND PIECES	5
4. ASSEMBLY OF THE DIFFERENT DEVICES AND OBTAINED PARTS	6
5. TESTS AND ITERATIONS WITH ELECTRONICS AND ROV PARTS	6
6. PROJECT MANAGEMENT OF THE PROJECT	6
CHAPTER 3. BUDGET	7
CHAPTER 4. PROJECT MANAGEMENT	11
GANTT	11
GITHUB	13
TRELLO	14
CHAPTER 5. DESIGN	15
5.1 CONCEPTUAL DESIGN.	15
5.2 DETAILED FORMS AND DESIGN	17
5.3 FINAL COMPOSITION	18
CHAPTER 6. DEVICES AND MATERIALS	21

A. ELECTRONIC COMPONENTS AND SYSTEMS	21
SENSORS	21
BATTERY PACK	22
PROPULSION AND MANEUVERABILITY	22
B. ASSEMBLY PARTS AND PIECES	24
TUBE	25
STRUCTURAL BODY PARTS	27
OTHERS	28
CHAPTER 7. FUNCTION PROGRAMMING: DESIGN AND IMPLEMENTATION	31
7.1. FUNCTION PROGRAMMING: PROGRAMMING LANGUAGE ELECTION OF THE ROV'S INSIDE SOFTWARE.	31
ARDUINO	31
RASPBERRY PI	32
PYTHON	33
C++	33
7.2. ROV'S INSIDE SCRIPTS	34
PRE-PROCESSING USER INPUTS	36
7.3. BUILDING THE GUI AND COMMUNICATING THE ROV WITH THE DATABASE AND OUTSIDE SYSTEMS	53
LOGIN SYSTEM	54
DATA MEASUREMENTS	55
CAMERA VIEW	56
MY GALLERY	60
CHAPTER 8. MANUFACTURING AND ASSEMBLY	61
8.1 MAIN BODY MADE WITH PVC.	61
8.2 TUBE CAPS	64
BACK PLUG.	66
FRONT CAP	68
RESULT	70
JOINTS AND SCREWS	71
CHAPTER 9. DIFFICULTIES AND DEVIATIONS	73
ESC	73
CHANGING THE BACKEND PROGRAMMING LANGUAGE	75
PRINTING WITH 3D TECHNOLOGY	75
CHAPTER 10. FUTURE WORK	81
CHAPTER 11. CONCLUSIONS AND RECOMMENDATIONS	83

BIBLIOGRAPHY	85
---------------------	-----------

ANNEXES	89
----------------	-----------

A1. TUTORIALS	89
A1.1 BATTERY TUTORIAL	89
A1.2 ESC WELDING PROCEDURE TUTORIAL	89

APPENDICES	91
-------------------	-----------

AP 1. BATTERIES ASSEMBLY	91
AP 2. DESIGN PLANS	93
AP2.1 BASE	93
AP2.2 RIGHT SIDE	93
AP2.3 LEFT SIDE	93
AP2.4 TUBE	93
AP2.5 FRONT PLUG	93
AP2.6 BACK PLUG	93
AP2.7 DOME RING	93
AP2.8 MAIN BASE	93
APPENDIX 2.8 FAREN BE	94

List of Figures

All the figures contained in this document have been created by ourselves unless indicated otherwise by the foot of the figure.

FIGURE 1. HERCULES DURING A LAUNCH IN 2005. SOURCE: WIKIPEDIA.	1
FIGURE 2. SEA EYE FALCON ROV IN AUGUST 2012. SOURCE: FLICKR.	2
FIGURE 3. THETIS. CASC'S OPERATIONS SHIP. SOURCE: MUSEU D'ARQUEOLOGIA DE CATALUNYA.	3
FIGURE 4. SCUBA DIVERS AT A SUBMARINE SITE ON THE COAST OF CATALONIA. SOURCE: MUSEU D'ARQUEOLOGIA DE CATALUNYA.	4
FIGURE 5. GANTT CHART.	11
FIGURE 6. VISUAL REPRESENTATION OF SOURCE TREE AND GITHUB SYSTEM CONTROL VERSIONS. SOURCE: SMARTNINJA.SI.	13
FIGURE 7. MVP GRAPHIC DESCRIPTION.	14
FIGURE 8. TRELLO ON-GOING PLANIFICATION.	14
FIGURE 9. GRAPHIC DESCRIPTION OF THE DESIGN PROCESS. SOURCE: JET PROPULSION LABORATORY, NASA.	15
FIGURE 10. FIRST ROV VERSION.	16
FIGURE 11. ARIANA-I ROV (MARZBANRAD, A., SHARAFI, J., EGHTEHAD, M., & KAMALI, R. (2011)).	17
FIGURE 12. ON THE LEFT, THE SECOND VERSION OF THE ROV. ON THE RIGHT, THE THIRD VERSION.	17
FIGURE 13. ON THE LEFT, PROPELLER SUPPORT BEING PRINTED ON A SIGMA R19. ON THE RIGHT, THE PRINTED PROPELLER.	18
FIGURE 14. PHYSICAL PROPERTIES AND ISOMETRIC VIEW OF THE FINAL ROV VERSION.	19
FIGURE 15. LAYOUT OF ELECTRONIC COMPONENTS, WIRES AND DEVICES.	21
FIGURE 16. ROV SIDE VIEW FROM <i>SOLID WORKS</i>	22
FIGURE 17. BRUSHLESS DIRECT CURRENT MOTOR EMAX CF2822.	23
FIGURE 18. ELECTRONIC SPEED CONTROLLER, EMAX BLHELI 30A.	23
FIGURE 19. PROPELLER PARTS. M4 SCREWS (1), PROPELLER SUPPORT (2), BLDC MOTOR (3), PROPELLER (4) AND NOZZLE (5).	24
FIGURE 20. TUBE.	25
FIGURE 21. DOME.	25
FIGURE 22. UP, THE BACK CAP. DOWN, THE FRONT "PLUG".	26
FIGURE 23. NYLON RING.	26
FIGURE 24. RING AND FRONT CAP ASSEMBLY.	27
FIGURE 25. MAIN ROV.	27
FIGURE 26. ELECTRONIC BASE SUPPORTS.	28
FIGURE 27. 3D VIEW OF THE TUBE ASSEMBLY.	28
FIGURE 28. PAIR OF CLAMPS.	29
FIGURE 29. THE CYLINDER WITH A THREADED ROD AND NUTS AT THE ENDS.	29
FIGURE 30. FULL ROV MODEL.	30
FIGURE 31. RASPBIAN OPERATING SYSTEM RUNNING ON THE RASPBERRY PI.	32
FIGURE 32. C++ STRONG TYPED EXAMPLE.	34
FIGURE 33. WIRING.	34
FIGURE 34. OVERALL GRAPH VISUALIZATION OF ALL THE INFRASTRUCTURE.	35
FIGURE 35. MAPPING OF ALL THE AVAILABLE INPUTS OF THE CONTROLLER.	36
FIGURE 36. WINDOWS OS PATH SYSTEM.	37
FIGURE 37. JSTEST PROGRAM RUNNING.	38
FIGURE 38. DIAGRAM SHOWING THE DIFFERENT VALUES AND THE CORRESPONDING ROTATION OF THE MOTOR.	40
FIGURE 39. RAW VALUES OF THE CONTROLLER'S JOYSTICKS. SOURCE: ROBOTS AND PHYSICAL COMPUTING.	40

FIGURE 40. TORQUE VISUALIZATION IN THE FORWARD MOVEMENT OF THE ROV. SOURCE: ROBERT D. CHRIST, ROBERT L. WERNLI SR., IN THE ROV MANUAL (SECOND EDITION), 2014	41
FIGURE 41. RESULT OUTPUT STRING SEND OVER THE SERIAL PORT.	43
FIGURE 42. MOTORS' CONNECTION TO THE ARDUINO.	44
FIGURE 43. PROGRAM EXECUTION USING AN EXAMPLE OUTPUT STRING FROM THE RASPBERRY PI.	46
FIGURE 44. DYNAMIC TYPING VS CASTING TYPES.	46
FIGURE 45. CONNECTION DIAGRAM OF THE TEMPERATURE'S SENSOR DS18B20. SOURCE: CIRCUITBASICS.COM.....	48
FIGURE 46. PULL-UP AND PULL-DOWN RESISTANCE	49
FIGURE 47. CONNECTION DIAGRAM BETWEEN THE SOIL SENSOR AND THE MCP3008 WITH THE RASPBERRY PI. SOURCE: GITHUB.....	50
FIGURE 48. CONNECTION BETWEEN THE MPU-9250 AND THE RASPBERRY PI. SOURCE: RASPBERRYPI.COM	51
FIGURE 50. RAW QUERY SENTENCES OVER A DATABASE.....	53
FIGURE 51. ROV'S WEBSITE SIGN IN FORM.	54
FIGURE 52. ROV'S WEBSITE SIGN UP FORM.	54
FIGURE 53. DATABASE PROOF OF HASHED PASSWORDS.....	55
FIGURE 54. MAIN INTRODUCTION TO DATA MEASUREMENTS.	55
FIGURE 55. MEASUREMENTS TABLE WITH ALL THE SENSOR DATA.	55
FIGURE 56. DIAGRAMS AND BRIEF EXPLANATION OF THE DIFFERENT SENSORS COMPOSING THE ROV.	56
FIGURE 57. CAMERA VIEW SECTION.	56
FIGURE 58. IMAGE'S PREVIEW.....	57
FIGURE 59. GMAIL'S EXPORT FEATURE.	58
FIGURE 60. MESSAGE IF THE DATABASE RECOGNIZES THERE ARE PICTURES AVAILABLE TO SEND A REPORT.....	58
FIGURE 61. MESSAGE IF THE DATABASE DOES NOT FOUND IMAGES ON THE DATE SPECIFIED.....	58
FIGURE 62. SENDING REPORT'S MESSAGE SCREEN.	59
FIGURE 63. FINAL EMAIL SENDS WITH THE IMAGES IN A DRIVE FOLDER.	59
FIGURE 64. COLLECTION OF PICTURES MADE BY ONE USER.	60
FIGURE 65. ALERT MESSAGE IN CASE OF NOT TAKING ANY PICTURE USING THE ROV.	60
FIGURE 66. ROV BEING ASSEMBLED.	61
FIGURE 67. 5MM THICK PVC PLATE. ON THE LEFT WITH LEFTOVER SIDES CUT. ON THE RIGHT WITH PROPELLER HOLES.	62
FIGURE 68. NOZZLE VIEW OF THE EQUILATERAL HOLE CONFIGURATION.	62
FIGURE 69. THREADING THE PROFILE OF THE 10MM THICKNESS PLATE.	63
FIGURE 70. ASSEMBLING THE BASE WITH SIDES.	63
FIGURE 71. INSIDE LATERAL CUT WITH SHAPE.	64
FIGURE 72. SWAY MOVEMENT ALONG Y AXIS. SOURCE: WIKIPEDIA.....	64
FIGURE 73. MAIN TUBE WITH SOME ELECTRONIC DEVICES.	65
FIGURE 74. THE ORIGINAL SOLID CYLINDER MADE OF NYLON. Ø120MM.	65
FIGURE 75. OPERATION OF EMPTYING OF THE BACK PLUG.....	66
FIGURE 76. FITTING THE TUBE WITH THE CAP ON THE LATHE.....	66
FIGURE 77. BACK PLUG ON THE LATHE.	67
FIGURE 78. BACK PLUG, BEFORE AND AFTER DRILLING HOLES.	67
FIGURE 79. FRONT PLUG BEING CUT.....	68
FIGURE 80. EXPLODED VIEW OF THE FRONT PLUG COMPOSITION.....	68
FIGURE 81. FROM LEFT TO RIGHT: RING, DOME AND FRONT PLUG.	69
FIGURE 82. FRONT PLUG ASSEMBLED.	69
FIGURE 83. RESULT OF THE TUBE COMPOSITION.	70
FIGURE 84. GLUE USED.....	71
FIGURE 85. ON THE LEFT, ARDUINO UNO. ON THE RIGHT, INSIDE AN ESC.	74
FIGURE 86. BLHELI SUITE V16 USER INTERFACE.	74

FIGURE 87. PROPELLER SUPPORT AFTER BEING PRINTED.	75
FIGURE 88. DEFECTIVE PROPELLER SUPPORT.	76
FIGURE 89. ANYCUBIC I3 PRINTER AFTER FINISH THE PRINT.	76
FIGURE 90. RASPBERRY CASE.	76
FIGURE 91. TEST PLUG TO CHECK TIGHTNESS.	77
FIGURE 92. TIGHTNESS TEST WITH PAPER INSIDE TO VERIFY IF WATER GOES IN.....	77
FIGURE 93. DISPOSAL OF THE REAL PLUG SIMULATING CABLES.	78
FIGURE 94. SETUP OF THE POWER SOLUTION FOR THE RASPBERRY.....	78
FIGURE 95. STEP DOWN APPEARANCE ONCE MOUNTED ON THE RASPBERRY.	78
FIGURE 96. BLDC WIRES. LEFT: FIRST VERSION. RIGHT: TRIPHASIC RECYCLED CABLE.....	79
FIGURE 97. GROUND CONTROL STATION. SOURCE: WIKIPEDIA	81

List of Tables

TABLE 1. MATERIAL AND ASSEMBLY ITEMS 7

TABLE 2. ELECTRONIC DEVICES AND 3D PRINTED PARTS 8

TABLE 3. HOURS ACCORDING TO CATEGORY. 9

TABLE 4. COST ESTIMATE AN/HOUR..... 9

TABLE 5. PROJECT PLANIFICATION. 12

TABLE 6. LIST OF EXISTING ROVS AND UNDERWATER DRONES. 16

TABLE 7. ROV MAIN CHARACTERISTICS. 19

TABLE 8. ELECTRIC BALANCE..... 20

TABLE 9. OUTPUT FOR THE DIFFERENT MOVEMENT’S EQUATIONS. 42

TABLE 10. ASSEMBLY ELEMENTS. 71

Chapter 1. Introduction

1.1 Historical context

A ROV, acronym for Remotely Operated Vehicle is as itself describes, a vehicle that is managed by an operator that is not on the vehicle. This kind of system has as many different applications as described below. It is important to remark that this project uses a *Tether*, an umbilical cable that usually provides energy and data transfer between the ROV and its operator who is over the surface.

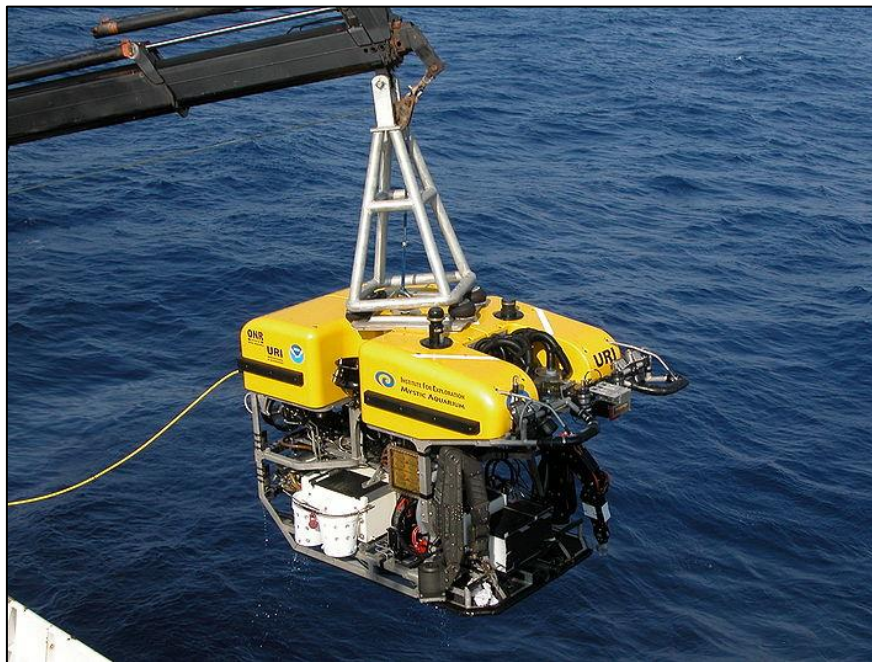


Figure 1. Hercules during a launch in 2005. Source: Wikipedia.

In the early 1960's, ROVs were used on military applications by several armies carrying out tasks such as mine hunting and mine breaking. Years later, by the 1980's the ROV industry experienced a small recession on the industry field, but a few years later it became a field of investments and research until these days, where this type of drones are very common and develop a huge range of functions and tasks.

Many robots can nowadays immerse itself and perform different tasks with relatively autonomy. Operations like sea bottom mapping either through sonar systems for reduced visibility zones or high-quality video capture on the other hand, data collection about water levels on river basins, the detection of invasive species and its extinguishing or just the opposite, the underwater biosphere observation and documentation. In these days there is more concern about climate change and pollution and many companies have developed kind of debris recollection systems in which ROVs are involved too. Also, there

can be found studies about high gathering of data like the study of the hydrothermal dynamics of Yellowstone Lake (Wyoming, USA). [See Bibliography 18]

1.2 Technical context: typologies

ROVs can be classified in two main groups: military-class and work-class. It is not the subject of this paperwork to study military-class ROV so it will be focused on the work-class ones. At the same time, within the work-class group it is possible to subdivide them into many other subgroups such as its size, weight, abilities or power.



Figure 2. Sea Eye Falcon ROV in August 2012. Source: Flickr.

Into the size group it would be the micro and the mini. Micros do not exceed 3 kg of weight while minis can weigh up to 15 kg but always being possible its manipulation and being carried by a unique person or boat.

In terms of abilities there are two groups: general purpose and inspection class. Both of them can have arms or fingers that allow them to manipulate objects or gather data. Inspection ones can often be differentiated from generals for having live-video cameras, sonar, photography gadgets or also other kind of parts for data gathering. The ROV presented on this paper will be fitted into this category, inspection class.

Finally, the work class are possible to distinguish for its range of power. It is possible to encounter light work class or heavy work class ROVs. In general terms, light power ROVs have less than 50 hp propulsion against a 200 hp of propulsion power for the heavy work class.

1.3 Origin

The idea for this project was born after one of us read a book of local history called “*La Torre de l’Àngel Custodi*” (Josep Bertomeu Mercé, 2010). The book narrates the history of a watch tower placed near the sea. Concretely on the “*Badia del fangar*” which translation could be “mud bay”, which is located in the Ebro Delta (Catalonia, Spain). On this location a reduced population of different people created a community to live approximately in the XVI century. Nowadays it is possible to visit the remains of the tower that rests six meters deep. Some fishermen know their location because their fishing nets are trapped. From the surface it is possible to see the rests of the old tower on clear water days. Then, the only way to achieve it is by scuba diving near the zone and trying not to get blinded by the sea current.



1.4 Research work

As we decided to provide a ROV for subaquatic inspection for archaeological purposes, it has been contacted the **Subaquatic Archaeological Centre of Catalonia** (Centre d’Arqueologia Subaquàtica de Catalunya, CASC -). The reason for that is to provide us with the maximum information about how do they work under the sea and which are the systems or devices that they use during immersions. Basically, we asked for 4 specific questions that they answered.



Figure 3. Thetis. CASC's operations ship. Source: Museu d’Arqueologia de Catalunya.

The first question was about the systems they must do an archaeological research. The second one, if systems are used, the frequency and the profitability of using them. Regarding the third question, we referred to the cleanliness of water at some depths. The last question leans us to the visibility once the scuba diver is immersed, in order to approximately quantify the lumens below.

The Head of the Subaquatic Archeology Center of Catalonia that hangs from the Department of Culture - Generalitat de Catalunya rapidly informed us about these procedures. On a first comment and answering the first question, she exposed that they start working with this kind of ROV when the site is located at fifty meters deep. Until these fifty meters they are capable of immerse themselves with their means and equipment. But this does not mean they do not use it. They work on AUV more than ROV on 25-meter depth sites. These AUV allow them to contrast information that archeologists had registered by hand. Also, answering the second one, they do not have these equipment, AUVs, ROVs or other means, but they do collaborations with institutions or universities when needed.

Regarding visibility, the third question was answered exhaustively. She answered with two examples. The first one on shallow waters on the “Costa Brava” near Girona. There is a place with usually good visibility. On the other hand, we have the Ebro Delta’s basin where visibility is always scared (it is there where we wanted to immerse the ROV in order to regard an old site at six-meter depth). Lastly but not less important, she told us about illumination. They obviously use artificial lights and not only for seeing but also for coloring. She affirmed it is normal to lose colors while you are going deeper. This brings you to see it all with blue tonalities that will badly affect the pictures.

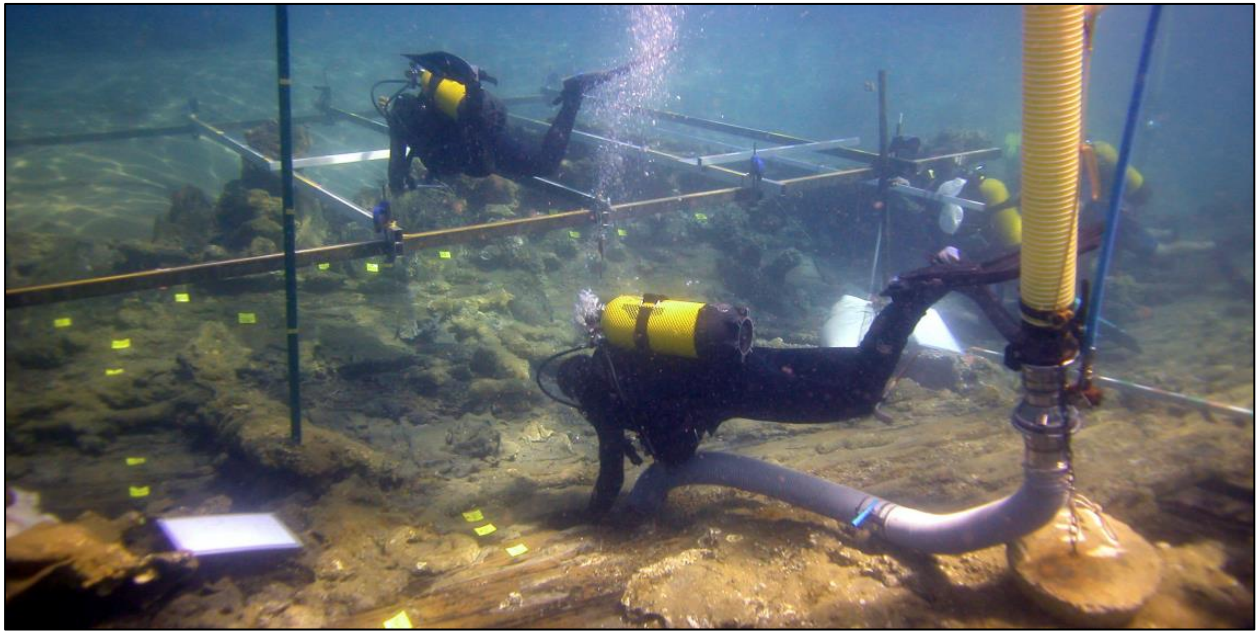


Figure 4. Scuba divers at a submarine site on the coast of Catalonia. Source: Museu d’Arqueologia de Catalunya.

So, with all this information provided by the CASC, we had pursued to develop our systems with a special focus on those comments we had as inputs.

Chapter 2. Objectives

2.1 Main objective

As it has been described before, the first commitment of this work is to being able to provide a useful and fully operational ROV, with cheapest cost (even being a prototype) and that any consumer that could need it for any service can construct it by himself without having to resort to anybody. This ROV is thought to people who perform their work in the field of subaquatic archeology, diving, inspection of submarine systems such as piping lines, sea structures or any offshore industry. Also, fish factories are nowadays using these systems to monitor their farms so it will be able to perform these tasks too.

2.2 Secondary objectives

Once primary goals presented it is also important to describe secondary objectives. These objectives are not what this work pursues although they are fundamental. These second goals are those without whom the main objective would be almost impossible to achieve. They can be named “collateral knowledges” since they are the knowledge acquired throughout the degree and the construction process later.

1. Design of the ROV itself and electronic devices

Almost from the very beginning of the project, computer aided design has been a key piece. This due to the capability of imagine and visualize how the future devices would take part, both electronics and structural body materials. Specifically, we have used SolidWorks 2018 as the CAD software. It is important to note that at the beginning of this project we have no idea of this software. We both took a sixty-hour course in order to learn it. On **Chapter 4: Design**, they are described all the design and choices taken from the conceptualization of the ROV to the execution of the final shapes. Also, on **Chapter 6: Function programming: design and implementation** it appears the design related to the electronic components and devices.

2. Devices implementation on the ROV

In this project it has been used *Arduino* and *Raspberry Pi 3+* as the two main processors. Arduino has been chosen due to its simplifications and user-friendly interfaces and ease connection for prototyping circuitry. It controls all the sensorial and instrumentations in terms of monitorizing. On the other hand, Raspberry was on a first instance chosen to control the camera system. As we progressed, we realized Raspberry would be useful to provide us the means of control. Then, it is also the translator between the control command and the Raspberry. Again, you can see the process on **Chapter 6: Function programming: Design and implementation**.

3. Mechanical workshop to create different needed parts and pieces

Taking advantage of the available means and the knowledge we have, we decided to produce all the parts in the faculty workshop. It has been used conventional machines such as the one, band saw or keyhole

saw and also using most advanced technologies such as Sigma 3D printer from BCN3D company. These machines and the site provided us the ability to do some cutting work, bored, turning, gluing, electronic tests and even 3D printing for some indispensable pieces such as propellers.

Both *Mechanical workshop* and the next one *Assembly*, can be consulted on **Chapter 7: Manufacturing and assembly**.

4. Assembly of the different devices and obtained parts

Differentiating this from “Mechanical workshop” is due to the main purpose of the process. On the assembly stage, both devices and mechanized pieces and materials come together in order to get all desired and predesigned shapes.

5. Tests and iterations with electronics and ROV parts

As in every design and prototyping process, tests, trials and experimentation need to be done. Furthermore, being an immersible device, forces you do to it tight. Tightness, as it will be presented moreover in later chapters is the most committed and critical part. This has been reflected on **Chapter 8: Tests and trials**

6. Project management of the project

The plan for the whole project needed to be premeditated and organized in order to achieve the final objective on time. Although it is something estimated, this can give you an idea about how much time and resources it will take. This is one of the most enrichment part of the whole work. As young students and soon graduated ones, we have very little experience on timings or milestones in order to optimize time and resources. On **Chapter 4: Project management** it can be seen the timings and calendar.

Chapter 3. Budget

In this section there are presented all different components acquired, both assembly and electronic parts. Despite they are individually presented in a more accurate way on the following chapter, here you can find every piece or component used during the construction of the ROV. Specifically, there have been created three tables. One containing the budget for the materials and those elements needed in order to assemble the vehicle (screws, glue...). Another with all the electronic elements, whether sensors, processors or lights among others. In the last one it appears an estimate of the cost man/hour.

NYLON, PMMA & PVC					
NAME	Lenght (m)	Width (m)	PRICE UD.	QUANT	PRICE
PMMA Semisphere			10,1	1	10,1
PVC Tube Ø110mm	0,35		13,56	1	13,56
PVC Barra gris Ø20mm	1			1	5,3
NYLON 6 (PA6) STABILIZED EXTRUDED Ø120 mm	0,12			1	10,29
					3,8
					2,97
					17,06
NYLON 6 (PA6) STABILIZED EXTRUDED Ø130 mm	0,08			1	7,72
					3,8
					2,42
					13,94
PVC 4mm DIN A5			2,25	1	2,25
PVC 10mm	0,5	0,5	33,75	1	33,75
PVC 4mm	0,5	0,5	13,8	1	13,8
PVC 3mm DIN A3			5,3	1	5,3
SUB-TOTAL PLASTIC					115,06 €
OTHERS					
NAME	Lenght (m)	Width (m)	PRICE UD.	QUANT	PRICE
Sandpaper 320	-	-	0,9	1	0,9
DIN 7985 A2 3x20	100 uds			1	3,8
Inox nut DIN 934 M-3	100 uds			1	3,4
M-4 x16					Free
M-5 x75					Free
M-6 x13					Free
M-8 Self locking thread	10 uds		3,8	1	3,8
Thermoretractile	1			1	1,1
M-8 Thread rod	90mm			2	Free
Clamp Ø120mm				1	3,6
Sikaflex 11Fc				1	7,9
Ceys plastic Welding			3,65	1	3,65
Gasket Ø3,4 mm			1,6	1	1,6
Nitrile Gaskets Ø110mm	x5 uds		4,27	1	4,27
Stuffing Box Electraline PG9			0,5	1	0,5
Stuffing Box Electraline PG7	20 uds		0,104	1	2,08
SUB-TOTAL OTHERS					32,80 €
TOTAL AMOUNT					147,86 €

Table 1. Material and assembly items.

SENSORS					
NAME	REF	PRICE	QUANTITY	SHIPPING COST	TOTAL COST
Temperature	Diotronic	16,18 €	1		16,18 €
	DS18B20	1,32 €	1	0,00 €	1,32 €
Pressure	MS5540	10,47 €	1	0,00 €	10,47 €
Water level/Humidity	W6K6	1,00 €	2	0,00 €	2,00 €
Water level/Humidity	Diotronic	5,32 €	1	0,00 €	5,32 €
Gyroscope/Acelerometer	MPU6050	1,04 €	1	0,00 €	1,04 €
Gyroscope/Acelerometer/Magnetometer	MPU9250	2,73 €	1	0,00 €	2,73 €
TOTAL SENSORS					39,06 €
LIGHTS					
NAME	REF	PRICE	QUANTITY	SHIPPING COST	TOTAL COST
LED spotlight 12V		3,60 €	1	3,00 €	6,60 €
LED		0,50 €	1	0,55 €	1,05 €
			0	0,00 €	0,00 €
TOTAL LIGHTS					7,65 €
MICROCONTROLLERS & CAMERA					
TYPE	REF	PRICE	QUANTITY	SHIPPING COST	TOTAL COST
Raspberry Pi 3 Model B	Model B	33,78 €	1	0,00 €	33,78 €
Arduino UNO	FNB	FREE	0		0,00 €
Raspberry camera	Diotronic	30,08 €	1		30,08 €
TOTAL MICROCONTROLLERS & CAMERA					63,86 €
BATTERIES & BMS					
NAME	REF	PRICE	QUANTITY	SHIPPING COST	TOTAL COST
18650 Li Ion	12x2500mAh	24,25 €	1	0,00 €	24,25 €
BMS	3S 10A	2,68 €	1	0,00 €	2,68 €
BMS	3S 20A	1,47 €	1	0,00 €	1,47 €
Nickel Plated Seel	15x5mm	3,13 €	1	0,00 €	3,13 €
High Temp. Heat Resistant	10mm BGA	1,18 €	1	0,00 €	1,18 €
Battery Spacers	18650 Batteries	3,73 €	1	0,00 €	3,73 €
Charger Balancer LiPo 3S	RC	22,00 €	1		22,00 €
LiPo battery	RC	22,51 €	1		22,51 €
Adapter XT60 to T-DEAN	RC	5,90 €	1		5,90 €
PDB XT60	RC	8,91 €	1		8,91 €
Adapter XT60 to T-DEAN	RC	5,90 €	1		5,90 €
PDB XT60	XSource	7,59 €	1	0,00 €	7,59 €
DC-DC Step Down Conversor	SODIAL(R)	4,60 €	1	0,00 €	4,60 €
TOTAL BATTERIES					113,85 €
MOTORS & PROPELLERS					
NAME	REF	PRICE	QUANTITY	SHIPPING COST	TOTAL COST
Electronic Speed Controllers	ESC BHELI 30A	6,32 €	6	1,76 €	39,68 €
BLDC 1200kV	CF2822	7,09 €	8	2,46 €	59,15 €
3D printed propeller (3 pieces)	ETSEIB	36,64 €			36,64 €
TOTAL MOTORS & PROPELLERS					135,47 €
CABLES & OTHERS					
NAME	REF	PRICE	QUANTITY	SHIPPING COST	TOTAL COST
Potentiometer 10k	RC TEC	1,95 €	1	0,00 €	1,95 €
40 M-H 20cm		1,00 €	1	0,00 €	1,00 €
65 M-M		1,31 €	1	0,00 €	1,31 €
JST-XH x10		2,37 €	1	0,00 €	2,37 €
16 AWG 2m		1,93 €	1	0,00 €	1,93 €
XT60 x5 pairs		1,64 €	1	0,00 €	1,64 €
4mm bananas	RC TEC	0,90 €	1	0,00 €	16,12 €
TOTAL CABLE MANAGEMENT					26,32 €
TOTAL					386,21 €

Table 2. Electronic devices and 3D printed parts.

In order to calculate the following man/hour estimation there was been controlled all the stages we pass through. Basically, “Main tasks” inside the Figure 5. Gantt chart on next chapter. Design (3), programming (5), devices and material selection (6), workshop operations and assembly (7), tests and analysis results (8, 9 & 10). Although we have done everything for ourselves, it has been done to give a real idea of what it would have been like in a real manufacturing situation. All these prices are only indicative in order to give an idea.

Taking as a starting point the days of each task is as follows summarized in Table 3. Number of hours are obtained by dividing by 5 (the average of worked hours per day).

	Days	Hours	Category
Design	42	210	A
Programming	16	80	A
Devices and material selection	12	60	A
Assembly	30	150	C
Tests	17	85	B
Conclusion results	8	40	A

Table 3. Hours according to category.

Man/hour estimation				
Subject	Category	Hours	price/hour (€)	Price
Engineer	A	390	30,00 €	11,700,00 €
Technician	B	85	22,00 €	1,870,00 €
Artisan	C	150	15,00 €	2,250,00 €
			TOTAL	15,820,00 €

Table 4. Cost estimate an/hour.

It is important to keep in mind that if the categories of artisan and technician could mean the same, we think it was better to divide it due to the type of work done by each one. The craftsman has a more decisive stance than the technician. He can take a decisive role in terms of changes or improvements during the mechanization and assembly process. By technician we refer to those tasks of data collection, analysis and possible iterations. So, the man/hour results for each one of the categories are as follows: 390 man/hour for the engineer category, followed by the artisan work with 150 man/hour and lastly 85 man/hour for what concerns to technical and inspection work.

Chapter 4. Project management

Gantt

On a first planification of this project, and in order of being capable of see the work volume, a Gantt planification was done. This first approach gave us the main idea of how address each task and give to it an estimated time to develop it. As it is said, a main idea, and a first planification is not always the best nor the optimum. Due to this, throughout the design and, above all, the construction process, we have adapted in order of satisfying the requirements found. The Gantt can be checked out on Figure 5.

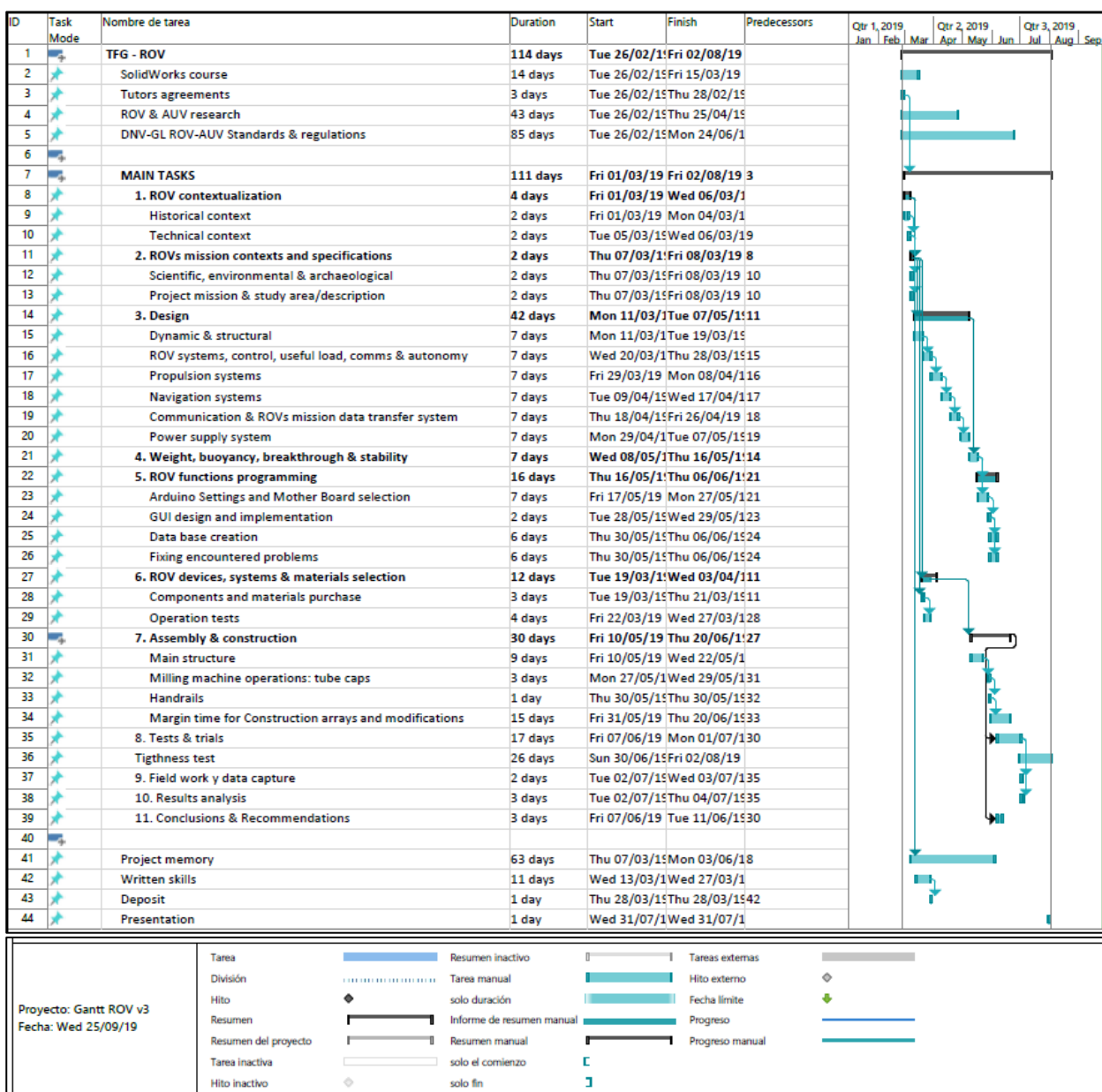


Figure 5. Gantt chart.

What it is exposed below is the most accurate timing planification, with modifications arranged while the project was going deeper. Basically, there are the main tasks we had covered during the whole process, and in some cases, the inner tasks or specific operations that we thought that require an own section.

1. Pre-process	1.1 Tutor agreements	
	1.2 Written Skills course	
	1.3 Solidworks course	
2. Design	2.1 Physical	2.1.1 Main structure
		2.1.2 Watertight enclosure
		2.1.3 Propulsion systems
		2.1.4 Electronic base
		2.1.5 Reinforcements
	2.2 Electronic	2.2.1 Main functions with Arduino MVP 1
		2.2.2 Raspberry implementation
		2.2.3 Propulsion systems
		2.2.4 Electronic base
		2.2.5 Reinforcements
	2.3 Web server	2.3.1 Minimum Valuable Product: PHP, CSS & HTML
		2.3.2 Graphs
		2.3.3 Camera view
	2.3.4 GUI. User Interface – User Experience	
3. Assembly	3.1 Mechanical procedures	3.1.1 Manufacturing parts
		3.1.2 Verifying assemblies
		3.1.3 Mount
	3.2 Electronics	3.2.1 Adjust support joints
		3.2.2 Verify watertight disposal
		3.2.3 Mount
4. Report	4.1 Writing	
	4.2 Disposal	
5. Presentation	5.1 PPT Creation	
	5.2 Tribunal defense	

Table 5. Project planification.

GitHub

Finally, to close this section, when arrived at the design of the web server (GUI) and the different databases that conforms the ROV GitHub was also used. GitHub is worth pointing out in this chapter because of is a control version system that has allowed us to write code at the same time. Is important to notice that in programming there is no code in real live between two person that is good enough to be used in this project. In order to maintain the code written in later chapters, and to maintain a clear order and an organized file structure we used GitHub in conjugation with SourceTree.

SourceTree is a visual representation of the mentioned version system control that we used throughout the project. SourceTree allowed us to have a master branch in which always has been the last updated and maintained code that was available, in this case the code forming the GUI. As the time has passed new features has been planned and added carefully and slowly.

With SourceTree, we can create a separate clone version of this master branch and recalled it as it wants to make changes in the code or implement some new feature or even fix bugs that keep appearing in the code. With this branches of the initial master file, multiple people can work at the same time in a huge project like this one and then merge the branches to the principal one in order to make the changes visible. Not only helps being able to write code without the fear of being removed or superscript by another person but also open the possibility to have an organized file structure. Furthermore, SourceTree brings the possibility to see the code that has been changed in the updates to the master branch and has the possibility to redo the steps if a new feature is not finished yet and the branch is being merge already.

Finally, to close the chapter, in Figure 6 it can be seen a visual representation of what is mentioned before with SourceTree and the branch system.

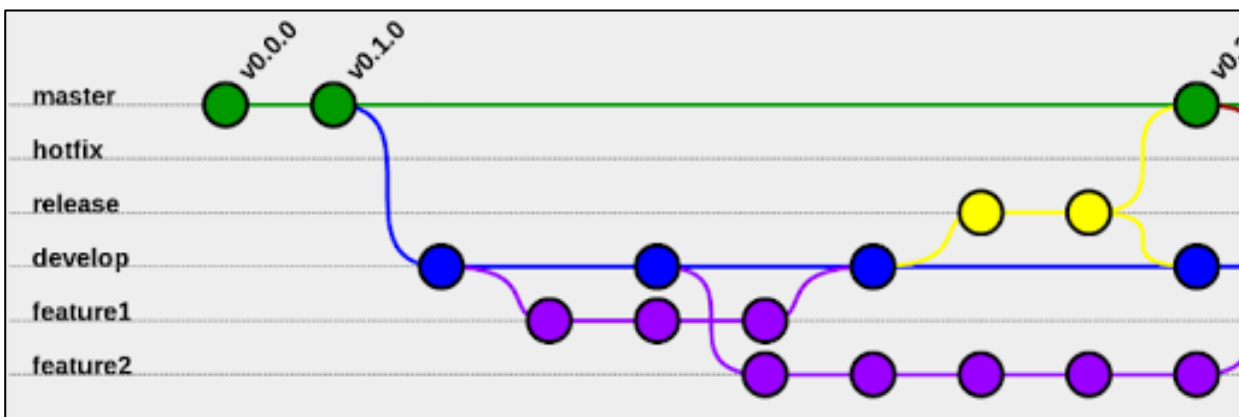


Figure 6. Visual representation of SourceTree and GitHub System Control versions. Source: Smartninja.si

Trello

Also, and in order to be able to join all the different processes during the project, a time planning chart with Trello was created.

Trello is a useful support application that allows you to organize different resources. With his help we had been able to work in a more efficient way. This has been done using the Minimum Valuable Product technique, known as MVP. As it can be seen on Figure 7 it consists in going to the point A to B with a minimum useful product. If we start from scratch, we can start going by skateboard. As we move forward, we can improve this product to get an airplane, if desired.

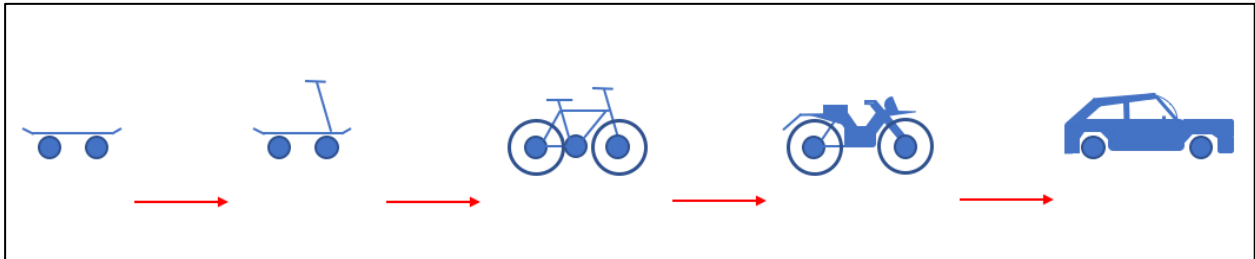


Figure 7. MVP graphic description.

So, using Trello, we have divided the project into different stages and brought them to its minimum definition of valuable product. We start by defining our ROV and identifying which parts are indispensable for it to work. In that way we ended up with three different iterations stages. The first one with the main body and a single-propeller configuration on each axis. On the last iteration, or the third evolved product has, as an example, two propellers on each axis, a light and the camera on its front.

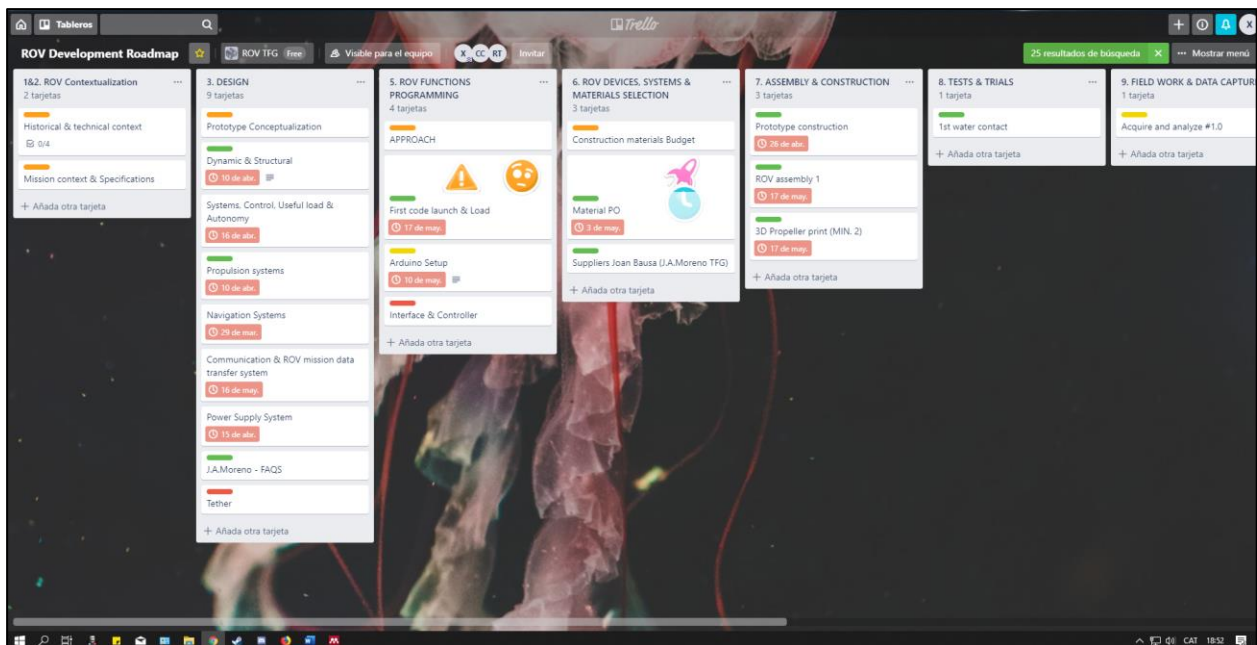


Figure 8. Trello on-going planification

Chapter 5. Design

For what concerns the design stage it is in this chapter where it is approached. This is presented in different sections where it is possible to differentiate between conceptual design. The first approach to the shapes and idea. Detail forms and design, where it is developed in a more specific and accurate way. Finally, the final composition in order to join all the different components and parts together.

The meaning of the word design can have multiple meanings and significates. Design is the word used for describing technically or graphically some product or service. It defines how a product will be assembled or conceived. So, the content provided on of this chapter will try to explain how the main product of this paper has been conceived and consequently arranged. Not only for a mechanical design but also for the different electronic setup choices. The chapter then will present the main ideas that inspired us to reach the goal.

According to an article published on the NASA’s Jet Propulsion Laboratory web page, design stages can be differentiated between them. At the beginning of the process there are the stages that have been addressed before, such as identify the product or the solution for a shortage in order to start designing it. It is at this point, before starting to build the prototype.

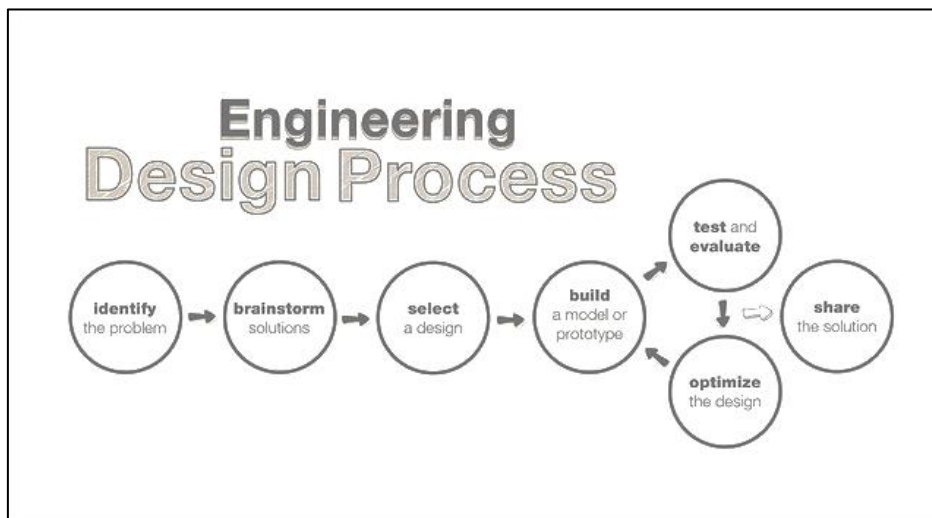


Figure 9. Graphic description of the design process. Source: Jet Propulsion laboratory, NASA.

5.1 Conceptual design.

The concept design refers to the phase where a first solution is presented. An empirical process takes part at this point. A first model is done, a simple drawing. Acquired from external inspiration about something or similar products that already exists. For this process a kind of database has been arranged to compare and find what would be our model or “muse”. The list below shows all the different submarine drones we checked out to achieve it.

number	Name	Web consulted
1	PowerRay	https://store.us.powervision.me/products/powerray
2	Gladius	https://www.indiegogo.com/projects/gladius-submersible-underwater-drone#/
3	Blueye	https://www.blueyerobotics.com/
4	FiFish P3	https://www.qysea.com/product.html
5	iBubble	https://ibubble.camera/
6	OpenROV	https://www.openrov.com/
7	DIY Drone	https://www.youtube.com/watch?v=kSMpLJIrY40
8	Making an underwater drone	https://www.youtube.com/watch?v=6pBH-3lXXO0
9	Build ROV	https://www.youtube.com/watch?v=yJcbAw-JYN0
10	BlueROV2	https://www.bluerobotics.com/store/rov/bluerov2/bluerov2/
11	Rover PX4	https://docs.px4.io/en/frames_rover/
12	Oceanbotics	https://oceanbotics.com/srv-8/

Table 6. List of existing ROVs and underwater drones.

As it is a conceptual stage, for this first design nonelectronic parts, specific materials or mechanization types had been considered. This lets you make a draft about which line are you going to follow. On the picture below, it can be appreciated the first model assembled in SolidWorks as it was.

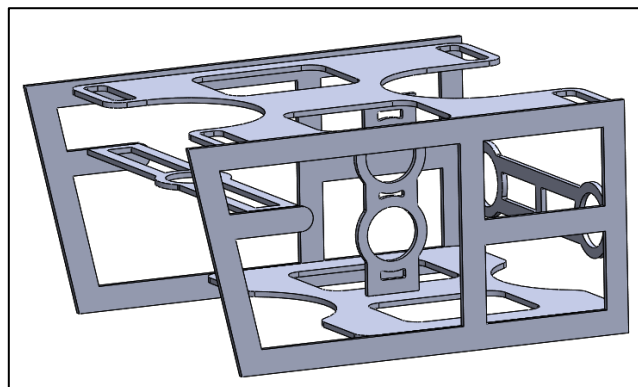


Figure 10. First ROV version.

This ROV was hugely inspired by the Ariana-I ROV (Marzbanrad, A., Sharafi, J., Eghtesad, M., & Kamali, R., 2001). So, it was very similar to it. In following stages and designs it will have more personality and different and new systems and features. Basically, it is the main idea about what it should be. The ROV would be able to fit six propellers. One pair of propellers on each different axis. It is more understandable if we regard to Figure 11 where it can be appreciated all different propellers on its place.

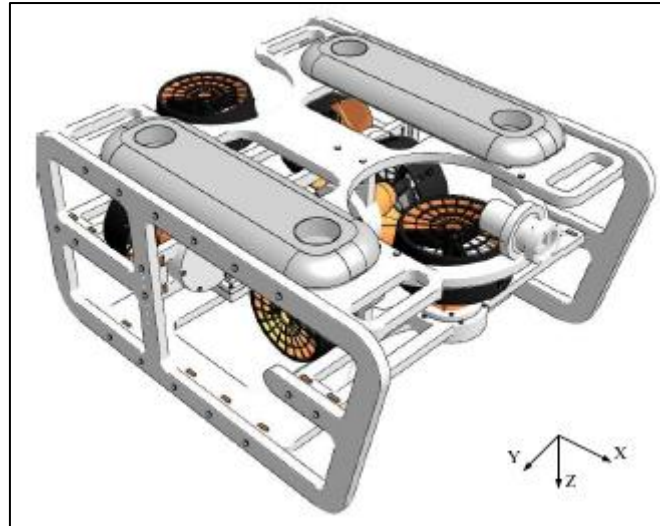


Figure 11. Ariana-I ROV (Marzbanrad, A., Sharafi, J., Eghtesad, M., & Kamali, R. (2011).

5.2 Detailed forms and design

Once the main idea of the concept is clear it is now the turn for the most accurate devices or parts. This process enables the different parts of the ROV to assembly between each other. This stage also demands the electronic devices. It means that different devices or utilities that will be mounted on it has to be most clear in order to being able of fitting it on board. Also, materials and fabrication processes must be taken in count.

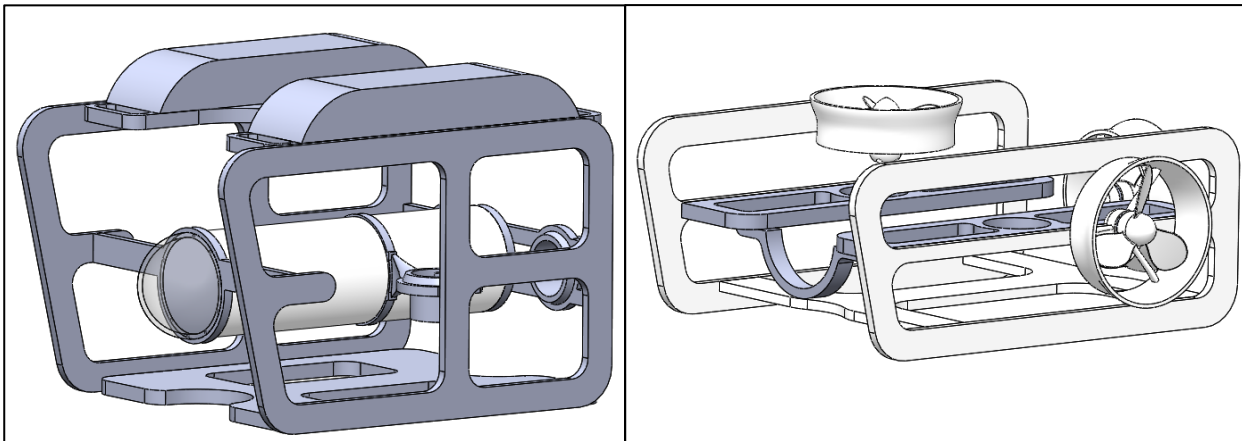


Figure 12. On the left, the second version of the ROV. On the right, the third version.

This section is the second with the most work charge on it. Due to its material and devices selection, and the need to find a design to assembly it all together. Despite it will not be until the third and final stage of the design process, where it all will be assembled, this stage is also important in terms of assembly. This means that every screw that needs to be fitted has to be defined and, in the position, where it will be. This does not mean that in later stages it cannot be changed or modified. On **Chapter 8. Manufacturing and assembly** there are presented different issues encountered and how they are solved.

Then, apart from having rounded out the forms, it has also been chosen a different body which will contain the electronic components. It is as it figures on Figure 12. Once this cylinder is disposed it is the time for choosing a more accurate place for the different propellers. In addition, as it is present in Ariana, a temporary space for a future buoyancy system has also been enabled on the top of the ROV. But this is rapidly removed on the third version (right image of Figure 12). We thought it will be better for the design to adding it at the end, if necessary. If necessary, a suitable ballast system would be implemented, such as lead plates or foam parts.

Also, one of the biggest issues on this stage was the propeller selection. We had a nice ROV that seemed to work perfectly in terms of design harmony. But then, propellers were not available. On a first moment 6 propellers were needed. This is due to the maneuverability design. So, it cannot be expended 80€ per each, it will be enormous expensive. One of the propellers we liked is on eBay for 34\$, it is also too expensive. Then we decided to design and manufacture it by ourselves, as it has been done. Despite having an existing one on eBay, on the ETSEIB (UPC) they have a 3D printing service. It costed us about 36€ to print a first propeller set, more expensive in comparison with eBay. But this is justified now we want to print more sets for free at the FNB workshop.

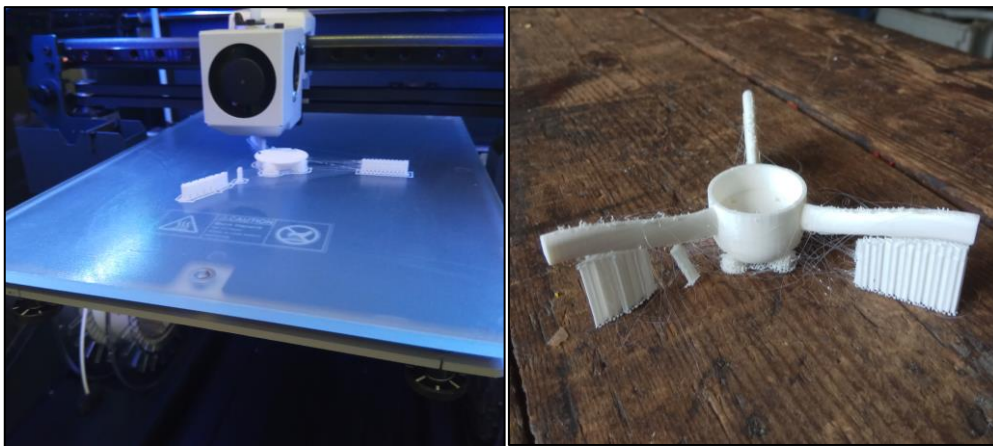


Figure 13. On the left, propeller support being printed on a Sigma R19. On the right, the printed propeller.

As the college have access to 3D printing technology, a Sigma from BCN3D Technologies, now we had a 3D printed propeller set for free. This without including the BLDC motor. In further sections it would be presented other components printed with 3D printer.

5.3 Final composition

On this final process, all different devices and element that needed to be mounted are fitted on the ROV. Most of them inside the main tube. Others like lights and propellers outside. In this stage it can be appreciate a good idea of how about it will be when it is finally built.

This is a stage where making huge changes can be dangerous or compromising, but even if a modification is needed it is relatively easy to deal with it. Especially with the help of the 3D CAD software and a minimum of workshop skills.

As it can be appreciated on Figure 14, the ROV looks gorgeous and theoretically it all fitted. Not even one screw has been left behind, so the software indicates us the whole weight of the product (an estimation) and also several physical properties such as the gravity center or the volume it occupies.

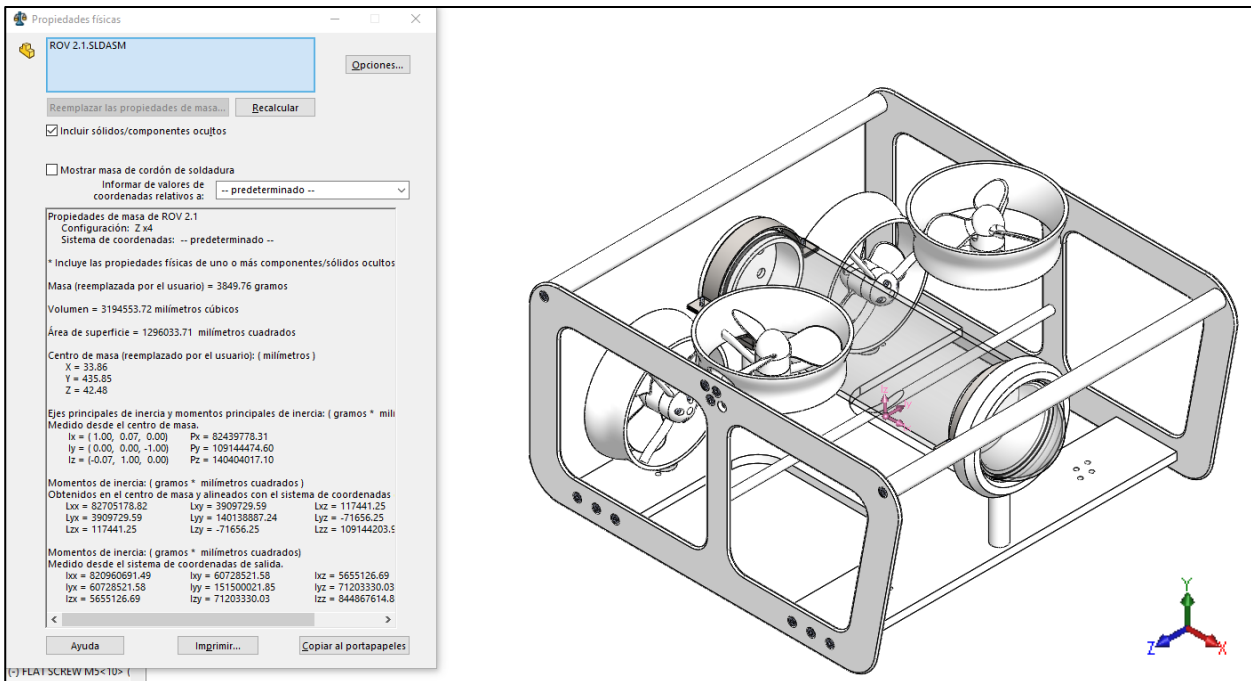


Figure 14. Physical properties and isometric view of the final ROV version.

If you look at the final design it can be observed that there are only four propellers, on two different axes. Therefore, to get around the center, one of the propellers on X-axes needs to act as if it were a row. It allows you to avoid using two more propellers. Despite of it, two more propellers can be added in front of the ROV, along the Z-axis, this is due to an equilateral triangle configuration. Also, the idea in order to achieve good and stable buoyancy is to put the heavier elements on the bottom of the structure. So, it would be on the bottom base where lead pieces would fit in order to achieve the desired ballast. Another important part is the transparent tube, which is full of air, which would act as a float, encouraging the ROV to ascend if not well ballasted. Then, the distance between its center of buoyancy (close to the tube) and its center of gravity is what will maintain the ROV stable. The more the distance, the more the stability. It is known as metacentric height.

Below it is showed a table containing the main parameters of the desired prototype.

Weight	4 kg	Depth	6-10 m
Displacement	3,194 L	Propellers	4
X-dimension	440 mm	Tether - USB	10 m
Y-dimension	364 mm	Tether - Ethernet	
Z-dimension	200 mm	Tether - Strong wire	

Table 7. ROV main characteristics.

For a subsequent electrical supply, an estimate of consumers has been made on board. To those that have been calculated by means of a multimeter they are emphasized with an asterisk...

Item	Amperage (mA)
BLDC*	1680
ESC	800
Raspberry Pi	3000
Arduino	1000
Sensors	88
LED*	418,2
	6986,2

Table 8. Electric balance.

This balance is an estimate of how all the components will affect the system. The values of BLDC and ESC are the corresponding ones for two units of this one. This means that in a very consumer scenario two propellants will be working at one hundred percent. In addition, about the sensors, it is an amount of energy that can never be achieved. In any case, it is a very pessimistic consumption situation. In this way we can have a battery pack with a safety factor for backup power.

Once finished the final three-dimensional model it is now the turn for the manual work with the mechanical and electronic devices.

Chapter 6. Devices and materials

In this section it will be reflected all the different components, materials and devices which are used in order to assembly the full structure, such as shells or screws and the micro systems on board forming the different navigation and control systems such as accelerometers or thrusters. In this case and in order to being attractive to read an organized list of materials with its cost has been presented on **Chapter 3. Budget**. So, it is both a descriptive section for materials and components and a chapter containing the main idea about every component detail.

Note that what is written in here is about our experience and troubles encountered with our kind of devices and components. This means that if in some section there is a troubleshooting for a specific kind of ESC, may be possible that for another trademark or earliest version of the same product it could not have the same problems.



Figure 15. Layout of electronic components, wires and devices.

A. Electronic components and systems

Starting from the inside of the ROV, from the smaller components, it is the first presentation of the main devices desired to be fitted on board. Some of these components such as the Raspberry and the Arduino, are exposed in a deeper way on **Chapter 7**.

Arduino Uno.

Raspberry Pi 3+.

Logitech camera.

Sensors

Temperature. Digital waterproof DS18B20 temperature sensor for Arduino.

Humidity. Detection module for soil moisture of 3,3V-5V.

MPU-6050. Gyroscope + Accelerometer: 3-axis module gyroscope + 3-axis accelerometer GY-521.

MPU-9250. GY-9250 9-axis sensor. Triaxial gyroscope, accelerometer and magnetometer.

Battery Pack

In order to provide powering to all the different systems and devices presented and to be presented a battery pack has been taken in count. Nowadays it is easier to encounter cheap and effective 18650 Li-Ion battery packs all around the globe, which is the one is used in this project. These 18650 cells have a capacity of approximately 2500mAh and they can be arranged with a 3S setup, among others, which means three packs in series of two batteries arranged in parallel. By doing this it has been achieved a battery of 12,4V and 5000mAh of capacity. This battery pack can be designed and built by yourself if you prefer to achieve a more affordable price or to obtain capacity as well as voltage, as it has been done in here. In order to achieve this, all the different components and wires are presented on Appendix AP 1.

With this battery pack and the pre-calculated consumption of about 6986 mA, it turns out that ROV can operate for a period of 42 minutes. As there is a second battery pack, the ROV could be operational for 84 minutes, one hour and 24 minutes...

Propulsion and Maneuverability

In order to provide thrust and self-supporting propulsion system, specific brushless direct current motor (BLDC) had been chosen. This kind of motor, combined with a 2-axis (X and Y) and a four-propeller configuration, two on each axis, allows to navigate and control the device as desired. It means that by powering one of the two longitudinal-axis propellers (the ones on X-axis) the ROV has the capability to turn around its center of buoyancy. Also, it can move up and down depending on the direction both motors oriented on Y-axis rotate.

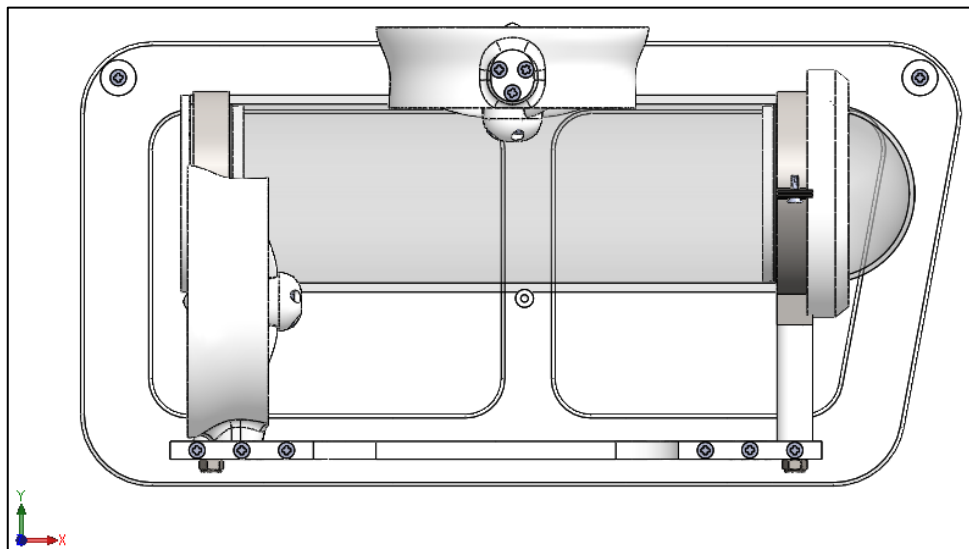


Figure 16. ROV side view from *Solid works*.

The elements used are listed below, with some description containing whether problems encountered or tips for an easy manipulation.

Brushless Direct Current motor: EMAX 1200 kV

This, together with the batteries, is the heart of the ROV. Is the element which give the propellers the power to generate thrust and being then capable of generating movement backwards or frontwards and upwards or downwards. It has been chosen this motor, due to its reduced price and capacity of generate a good number of revolutions that will be transformed into thrust. Concretely it is a 1200kV motor. It means it has 1200 revolutions per minute (rpm) for each Volt inputted. It has a range of operation between 7V and 12V, being 11,1V the optimal one. Basically, if we are injecting 8V to the BLDC, it will generate 9300 rpm.



Figure 17. Brushless direct current motor EMAX CF2822.

This brushless motor is a good option, but there are a lot of different types of BLDC motor, varying on shapes and even more on work ranges for kV, or sizes. These BLDC, despite being interesting with its capacities, does not work without an Electronic Speed Controller (ESC), which is presented below.

Electronic Speed Controller: EMAX BLHeli 30A

The Electronic Speed Controller is the one in charge of telling the BLDC how to move, the direction of rotation, velocity and when to stop. These features are controlled with itself intern circuitry. Far above, is composed with some MOSFETs and an integrated circuit (IC). Like an Arduino IC. They are the responsible of sending pulse width modulation signals (PWM), in order to achieve these mentioned characteristics. We are not going to explain how PWM works. If you are interested there is a good source: <https://www.techopedia.com/definition/9034/pulse-width-modulation-pwm>



Figure 18. Electronic Speed controller, EMAX BLHeli 30A.

Also, this component, the ESC, can be modified by using a software from the same company, called *BL Heli suite 32*, or the plug-in for Google chrome, called *BLHeli Suite*. If you are able to connect by using

Arduino devices or other devices that are listed on the software itself, is very easy to use and to told the ESC how you want it to work. In our case, it was not like that. On **Chapter 9. Problems and deviations**, it will be presented the different issues encountered and the solution.

Thrusters

As it has been advanced on **section 5.1 Conceptual design**, in a very first moment it existed a huge complexity in order to get housing parts for the propellers. By “Housing” parts it is understood a piece, being made with plastic, metal or some other resistant material that allows to mount in the BLDC motors and the propellers to generate propulsion in our device. This difficulty added to the limitations for the ROV BLDC selection needed to be passed through and to start the ROV construction itself. After some time spent researching for these parts it was taken the decision of designing and fabricate the BLDC motor propellers and their accommodation. Down below are described the different three parts designed in order to accommodate the EMAX CF2822 BLDC motor. All these three parts printed with *BCN3D Sigma* printer and an *Anycubic i3 Mega*, using PLA filament. On the picture below it can be clearly appreciated every single part with an exploded view of this set.

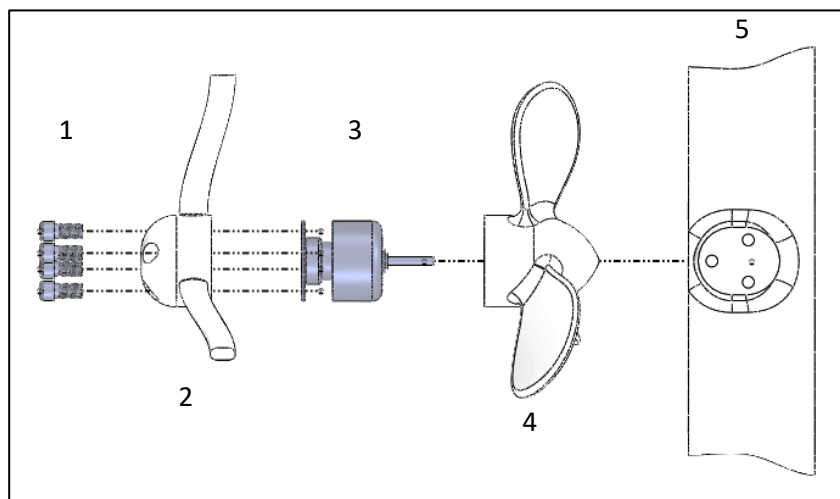


Figure 19. Propeller parts. M4 screws (1), propeller support (2), BLDC motor (3), propeller (4) and nozzle (5).

One little problem encountered in posterior phases was that the propeller support part does not have a hole. This hole is the one that allows the three wires of each phase of the BLDC motor to get in. This resulted in having to mechanize this hole after. This is a good enough solution, but in subsequent iterations it would be changed in order to not having to mechanize it. Instead it will be designed properly in order to get the printed part with the hole.

B. Assembly parts and pieces

In this subsection they are presented the different parts that have been designed to assembly the integrity of the vehicle. Note the colors and tonalities of the different drawings are not relevant. It is due to its “material properties” within *Solidworks* that they are colored by a color or another. As it is presented on **Chapter 8. Manufacturing and assembly**, parts from PVC are dark grey and for 3D printed parts are, in its majority white and some of them black.

Tube

Transparent PVC tube. The reason for choosing a translucent material is the idea of being capable of seeing all different components mounted and to check easily its well assembly and not throwing the device with some disconnected cables.

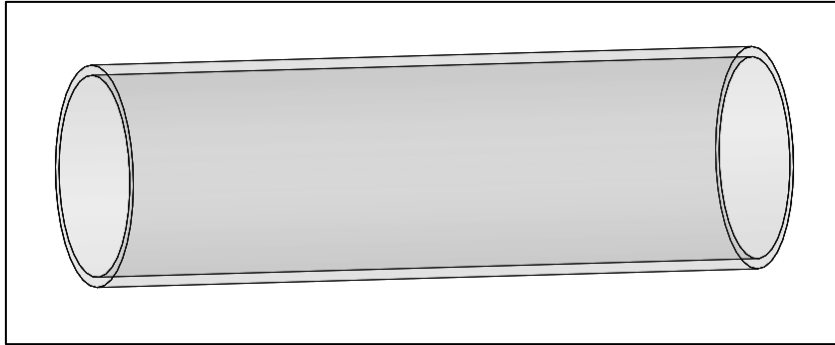


Figure 20. Tube.

Poly(methyl methacrylate) (PMMA) Semi sphere. The reason for choosing a translucent material is the idea of being capable of seeing all different components mounted and to check easily its well assembly and not throwing the device with some disconnected cables.

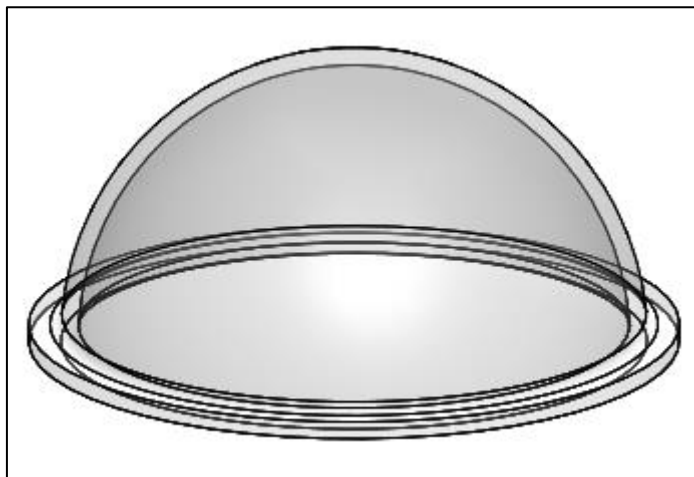


Figure 21. Dome.

Nylon plugs. Its main function is to provide the tube with totally tightness, using nitrile gaskets. Unlike the tube and the dome, which are bought, assembled as they came, these caps are designed and mechanized by us. This means it was originally a Nylon cylindrical piece and then it turned into two new shaped pieces. The original cylinder was about 120mm diameter and 120mm length. With this cylinder it was possible to obtain both plugs. Despite they are called plugs, the front one is technically not. This is because this is open so that you can put the dome. The transformation process is possible to see on **Chapter 8. Manufacturing and assembly.**

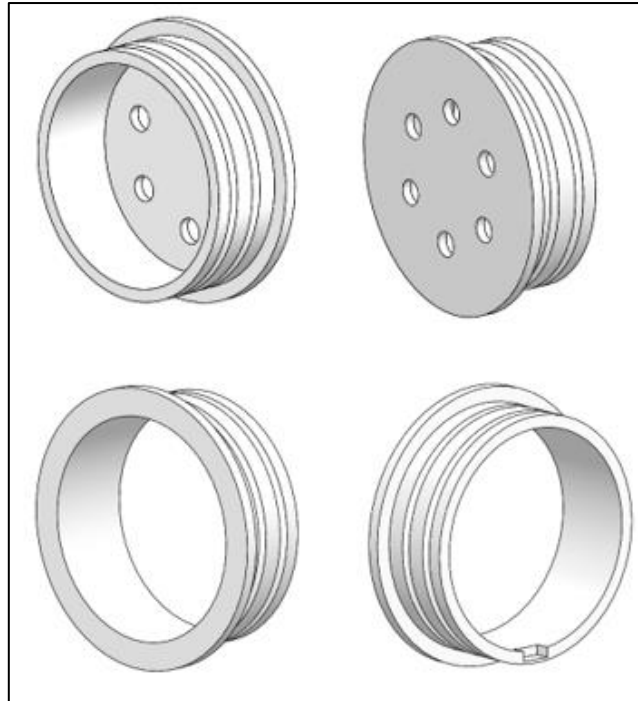


Figure 22. Up, the back cap. Down, the front "plug".

Nylon ring. During the development of the caps the need arose to design a system that held the semi sphere against the tube. So, it was decided to turn a nylon ring. But this should be of greater diameter, logically. Then another cylinder of nylon was bought. This one about 130mm diameter and 60mm length.

The idea is that the ring presses the transparent dome against the front cap once sealed with silicone, so that no water enters.

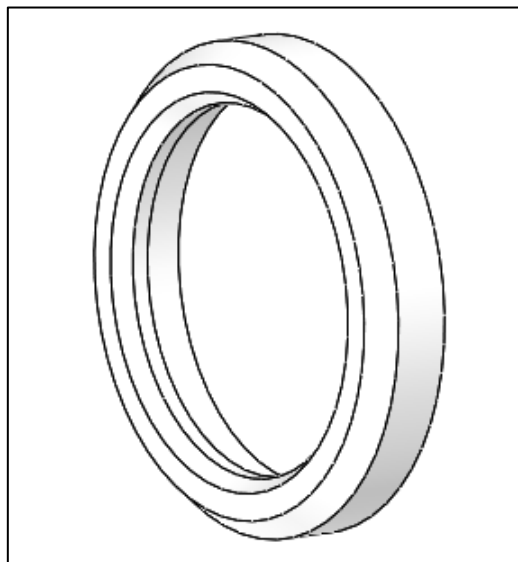


Figure 23. Nylon ring.

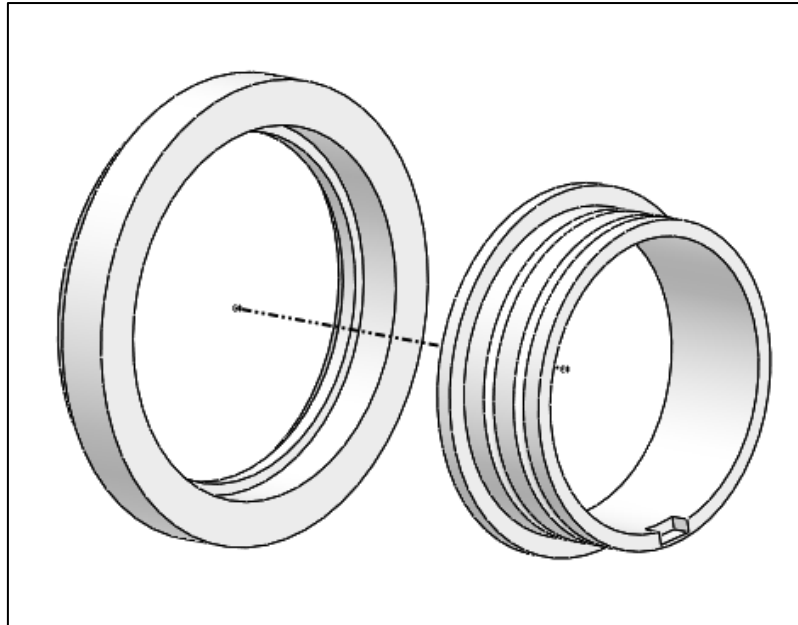


Figure 24. Ring and front cap assembly.

Structural body parts

Basically, all the structure of the ROV is assembled using Polyvinyl Chloride, known as PVC. Using thicknesses of 4mm for the laterals and 10mm for the bottom base. Also, two Handrails working as reinforcements providing strength to the full system are placed. These handrails are 20mm circular section bars, of about 440mm length, measurements from side to side of the ROV.

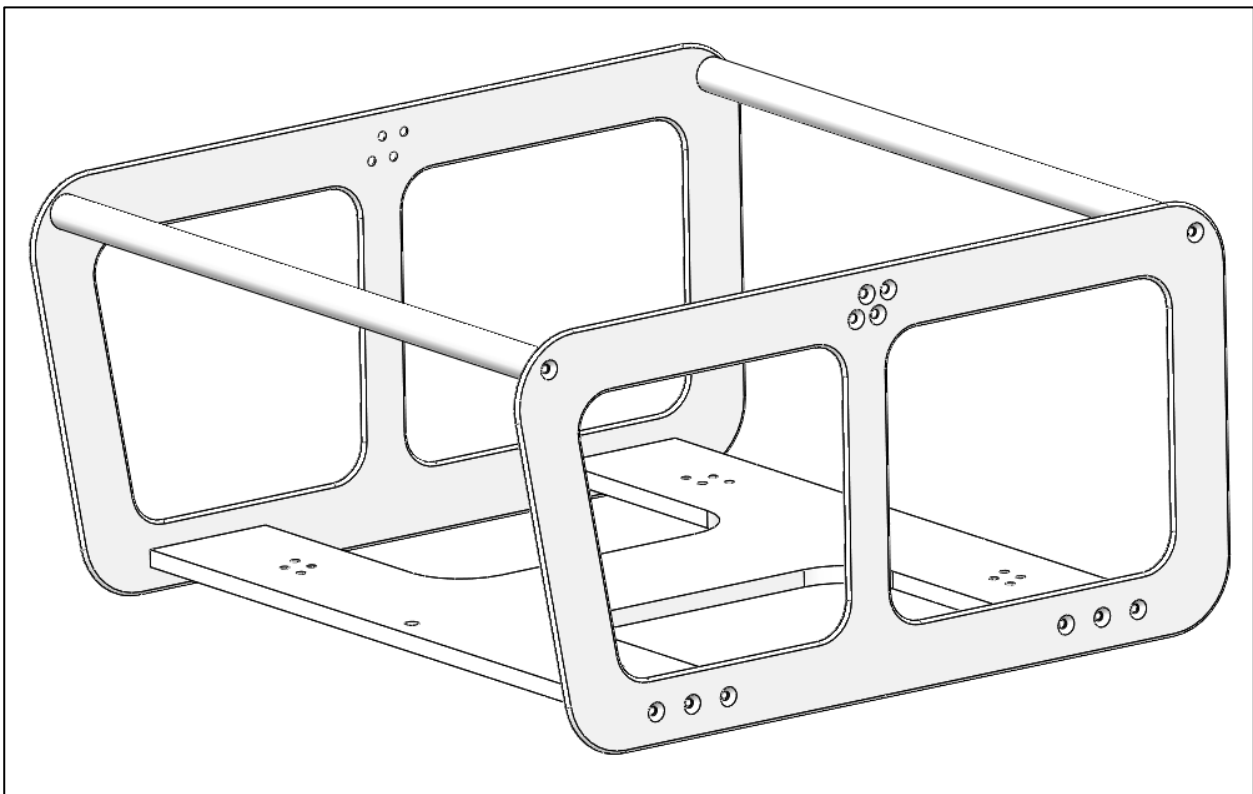


Figure 25. Main ROV.

Others

3D printed base support. In order to avoid the main base to turn inside the tube there has been designed these “half-moon”-shaped parts. It is achieved by a mechanical shaft key. A more understandable image of these pieces is on Figure 27. Also, on Figure 24 it can be appreciated the shaft key where it fits the printed support.

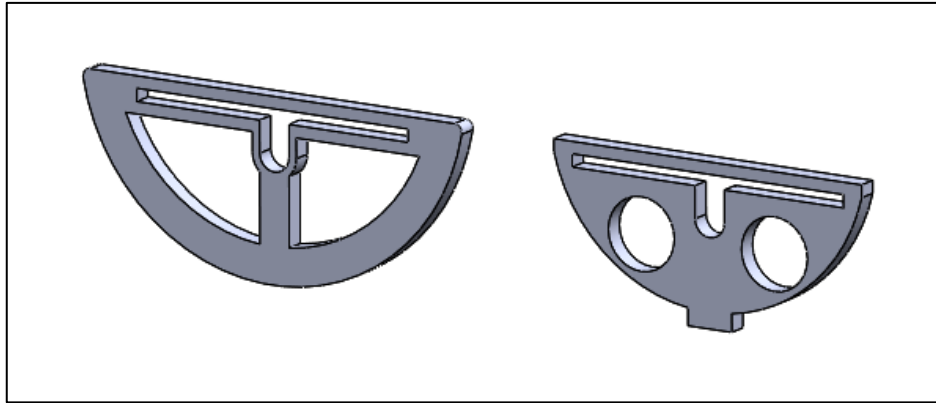


Figure 26. Electronic base supports.

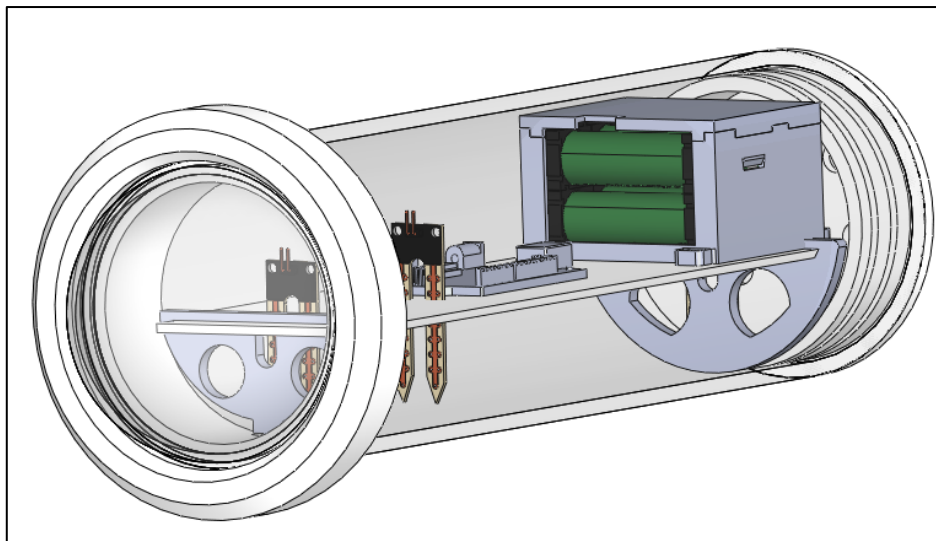


Figure 27. 3D view of the tube assembly.

Clamp. It is formed by a semi-circular metal clamp covered with plastic paint. These have been bought at the hardware store. Its function is to attach the tube to the main base. It is achieved by using two short PVC cylinders through which is attached to the base.

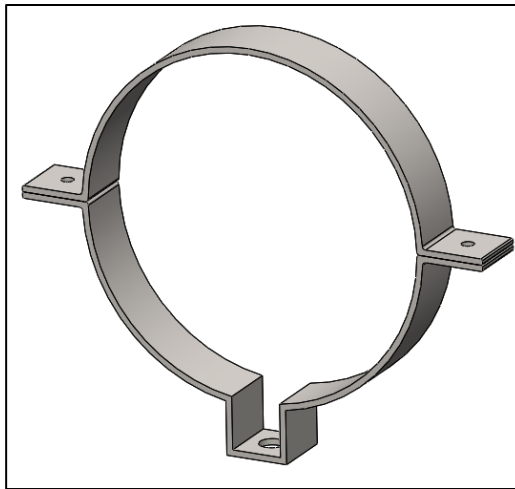


Figure 28. Pair of clamps.

PVC 20mm cylinder. This is the short piece made of a 20mm diameter PVC cylinder. A hole has been made to the cylinder to pass the M8 threaded rod. Its length is about 90 millimeters. Once the cylinder is mounted and two nuts tighten it at the ends is when the tube is perfectly rigid to the body. This, for example, is one of those designs that have been adapted once construction has begun.

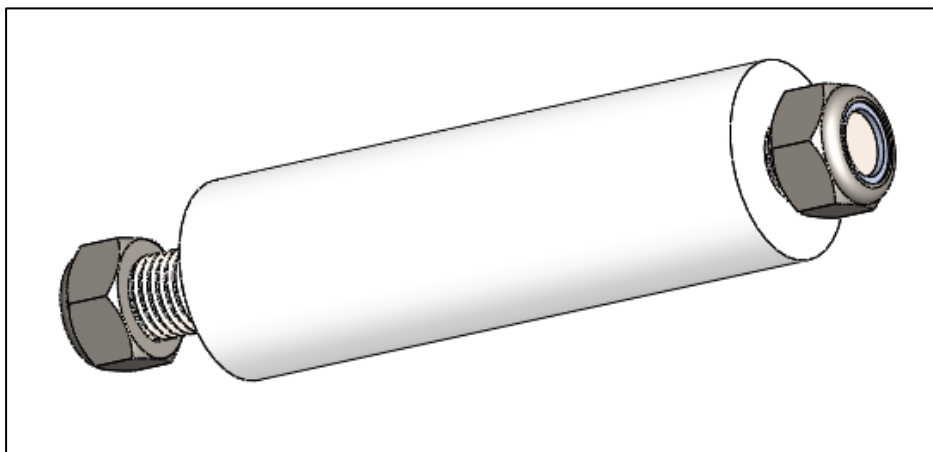


Figure 29. The cylinder with a threaded rod and nuts at the ends.

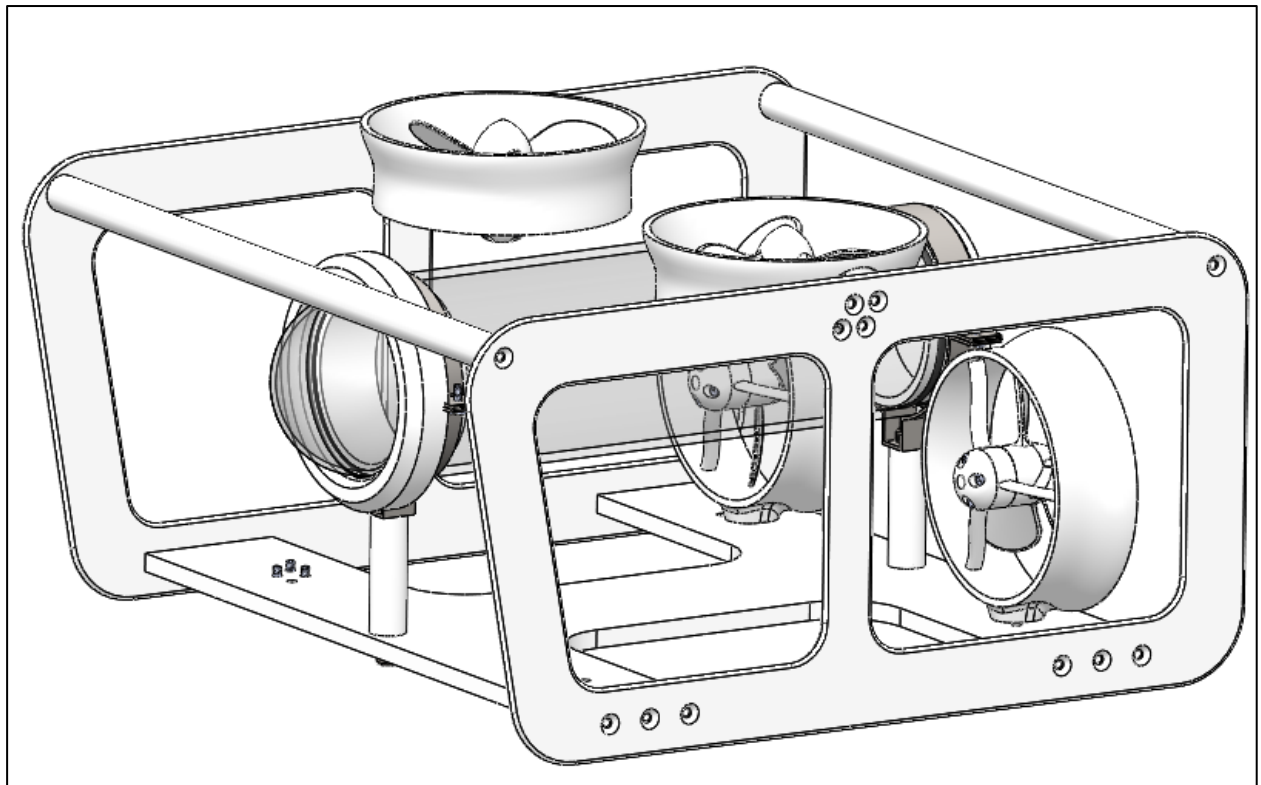


Figure 30. Full ROV model.

Chapter 7. Function programming: Design and implementation

Chapter 7 will tackle one of the main objectives to acquire in the ROV's design: the capability to move freely in the 3-dimensional plain with some freedom. Moreover, the ROV's capability to measure different data from the environment that surrounds it and send it successfully to a database to be stored and processed correctly.

In conclusion, Chapter 7 will focus on the ROV's software to archive this main two objectives. Furthermore, to see clearly the different measurements, a GUI will be built also to be able to interact with the data gathered by the ROV.

The chapter will be divided in three parts: in the first one, the different programming languages as well as the different components will be introduced and will be explained giving reasons that defends the election of them to build the ROV's internal software. Then, in the second part it will be tackle the different scripts and the processed of building the distinct systems that together make the ROV's software.

Finally, in the third part it will be explained the GUI interface as well as connecting the ROV's software with the GUI which will be encountered outside the water and the different languages used to build the outside ROV's software.

7.1. Function programming: programming language election of the ROV's inside software.

As it was mentioned before, the system has different sensors and microprocessors that builds the ROV. Those microprocessors and microcomputers often use programming languages to make different scripts, so in order to understand better the programming language election, these components will be briefly remembered and will introduced some software's specification.

Arduino

In first place, Arduino as it is mentioned before, is the microprocessor of the ROV. Nowadays, Arduino is known for be one of the most powerful and sustained open source microprocessor of the market. Therefore, documentation and information about the different sensors can be acquired easily providing a strong base around the ROV's construction and specifications requirements.

The Arduino will oversee processing signals and make different decisions in order to turn off or on the motors depending on the input that the user give us. Arduino is built and coded around a slightly changed version of C++ center on i/o devices and address ports.

Raspberry Pi

Raspberry is a microcomputer able to execute complex orders and it is used in a variety of applications, since more simple ones up to some complexes like fingerprint recognition. Is important to consider the difference between a microprocessor and a microcomputer. While the Arduino is good at making operations repeatedly at a fast rate, a microcomputer like the Raspberry can execute more complex programs, and it is built with the capability of having an operating system. Despite being so powerful, the Raspberry can also substitute the Arduino as it is also capable of doing repeatedly tasks rapidly. In this project, is the most important part of the ROV's software. It is vital because of the need to have a "brain" that it is capable not only of transforming different inputs but being able to communicate with the other systems that are built around the ROV like the Arduino, or the database that will talk about later.

In other words, the Raspberry is the connection link between the different systems, therefore, being one of the most important parts of the software and one that it will hold the majority of the scripts responsible for the distinct actions that the ROV will be able to do. The Raspberry will oversee understanding and making connections between the different ROV's systems as well as oversee recollecting measurements from the environment surrounding it.

Finally, the Raspberry can process a lot of programming languages that can do the job as demanded, but it really shines when using Python. It is used that much, that in most distributions of the operating system that the Raspberry uses, Python is directly pre-installed.

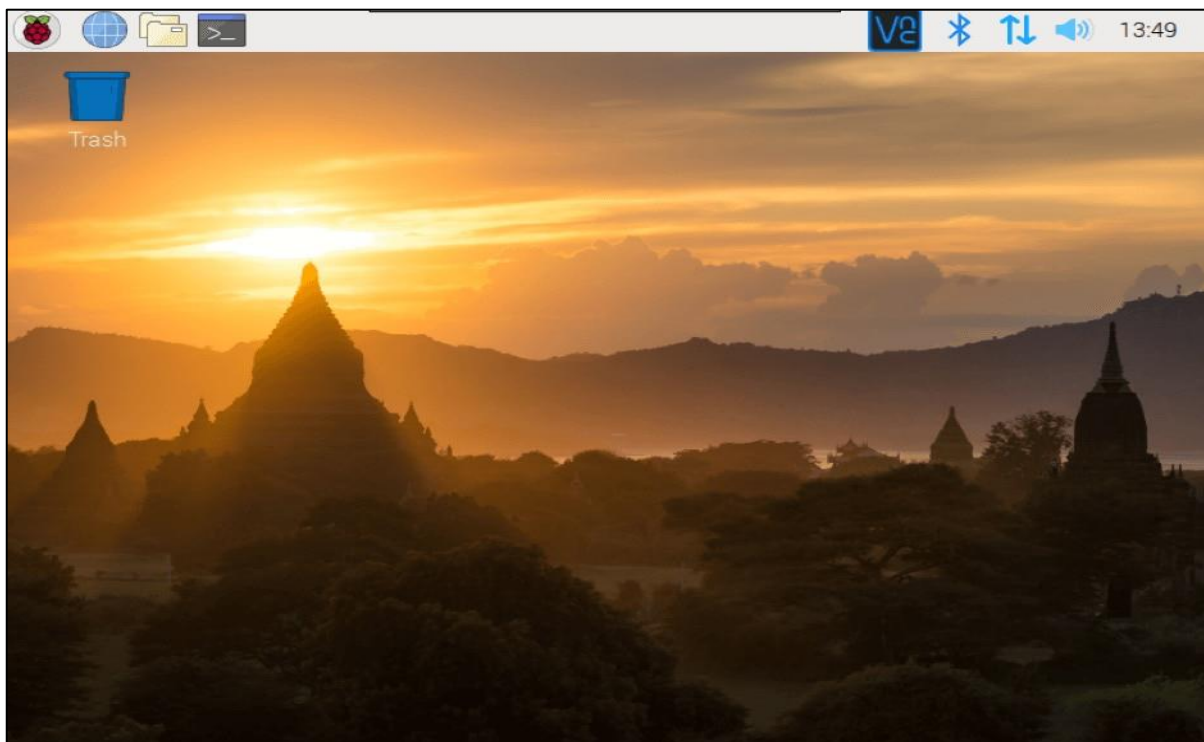


Figure 31. Raspbian operating system running on the Raspberry Pi.

The different scripts will be built around these two pillars, that represents the most connections that the ROV has. Once commented it can be introduced the different programming languages that it will be using:

Python

Python as it was mentioned before, make up the script's majority inside the ROV. Python is an open source programming language used now by a lot of different people in different applications. It has been decided to choose to use Python because of different reason. The first one and most important is that as mentioned before Python is the core component of the Raspberry ecosystem forcing us in some way to use it before anyone else. Furthermore, as it so used nowadays, the syntaxis to write in Python has become one of the easiest to take on at a beginner level, being so much easier compare to C or C++ for example.

Python is used by most in sciences fields like deep learning, neural networks or machine learning. Therefore, Python has libraries or code that is prewritten that helps us doing complex operations like matrix multiplication, or in this case, processing inputs from a user. This libraries or pre-written code also help us reduce code and maintain the project cleaner and more expandable in the future. One of the most important things at the time of making the ROV's software was expandability and maintainability. If a project it is not easily maintainable, it cannot be revised and corrected. If it is not expandable, the project stops are a certain point since the complexity jump so much that it cannot be improved any more.

Those consideration make Python the primarily language used on this project. Finally, before continuing explaining the other languages, it must consider that the project has been built using two different versions of Python. It is important to considered because of some scripts are written using Python 2.7 and can be problematic if it is processed by a further Python's version. Consequently, in the coming scripts the version will be clearly specified to prevent bugs from happening then reducing the total errors.

C++

C++ is the base of modern programming and it has always been an excellent programming language to program microprocessors like the Arduino. As happens with Python, C++ is used primarily because of compatibilities with Arduino, due to Arduino's code editor only understanding and able to save scripts in Arduino's memories using C++. Furthermore, C++ is more rapid that Python at compilation times. That means that C++ runs much faster that Python. Is an interesting fact that help understand why a microprocessor is coded using C++, in the end, the microprocessor has to be able to compile and execute millions of transactions every second without creating delay, which lead to a correct functioning of the different motors that the ROV has. Despite being more complicated to learn than Python, it also provides some libraries that helps reducing code and making the process a lot simpler. That does not mean that is less complicated to code in C++ due to be an older language than Python. Another point in favor of C++ is that helps in the mission of archiving code more expandable and maintainable because of a syntaxis characteristic. C++ is strongly typed which means that all the variables declared in the ROV's scripts must have a valid type and cannot be changed during execution.

Consequently, errors may be less pronouncing that in Python, and the appearing of bugs also is drastically decreased due to the fact of being stricter. Figure 32 can show better what means to be a strongly typed language. Strong typed languages evaluate the variable before running while other languages can change the variable types on the process, Python is one of them, they are known as dynamic typed languages.

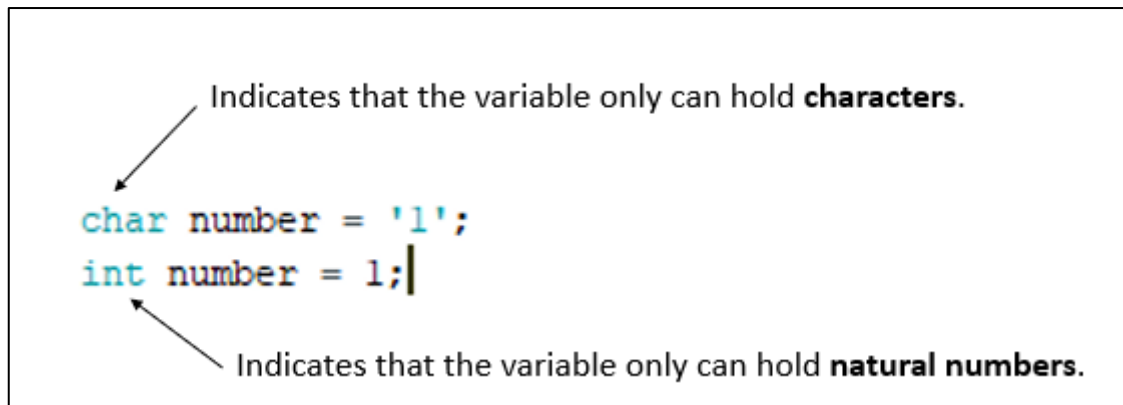


Figure 32. C++ strong typed example.

In conclusion, C++ normally presents a more robust structure and rapidness than most of the other programming languages, making C++ one of the languages used to run the ROV's inside software.

7.2. ROV's inside scripts

In this section, it will be explained the different scripts that are located only inside the ROV. This differentiation is made to make an easier distinction between the GUI, database and the ROV's movements and recollecting measurements. The main objective of this section is to discuss and explain the different scripts that build the ecosystem of the ROV's movement and the measurement recollecting and processed.

The structure of the different components that will be considered in order to build the different scripts can be seen above in other Figures, but the Figure 18 shown below zooms in the necessary part that will be discussed in this section of the chapter.



Figure 33. Wiring.

As it can be seen in Figure 33, it has been decided to use the most comfortable way for the user to produce different inputs. Consequently, a PlayStation 3 controller has been used in order to produce the distinct inputs that will move the motors. This controller is plugged in to the Raspberry using the USB port, that can be located inside the ROV. The Raspberry will oversee process the inputs made by the controller and transforming them into valid outputs to the Arduino. On the other hand, the Raspberry will be connected

as seen in Figure 34 using a micro-USB to USB-A adapter to the Arduino. All this connection will be very important later when discussing and building the different scripts since are the only bridge that communicates the different systems.

Moreover, all sensors are connected using the Raspberry GPIO's pins. These sensors will oversee recollecting the different measurements that will be processed later by the GUI and stored in the database.

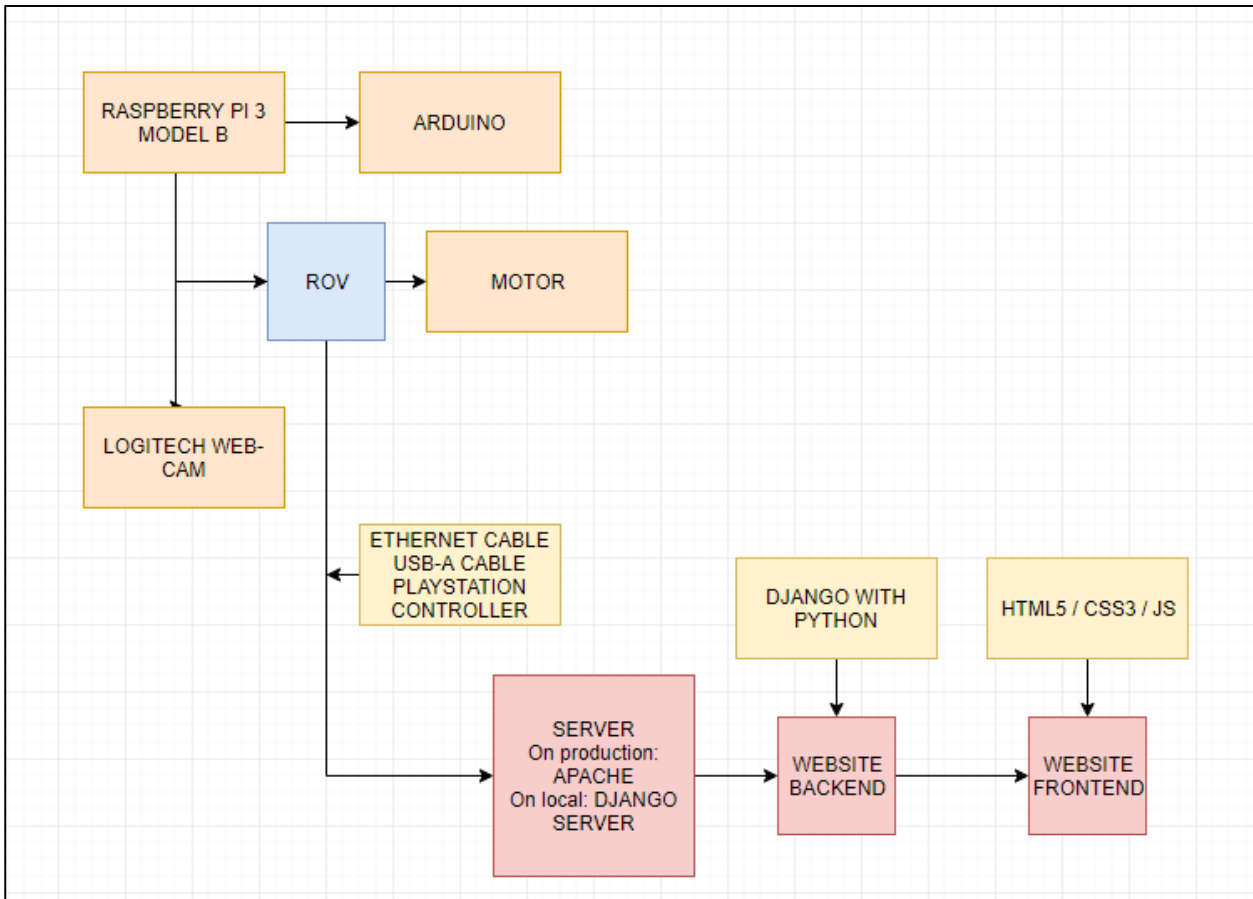


Figure 34. Overall graph visualization of all the infrastructure

These measurements will be sent through an ethernet cable to maximize the efficiency and data transport, in other words, reducing the delay between taking the measurement and storing it into the database and finally visualizing the data in the GUI. As mentioned before the Raspberry is the connection link between all systems from the inside and outside.

Pre-processing user inputs

In the first part, the main objective of the first script is to be able to communicate the PlayStation controller to the Raspberry and be able to receive the different inputs and process them correctly. This script is written in the Raspberry with Python. Before continuing and to make clear which inputs the Raspberry will hear to, Figure 35 shown below the input mapping that was made in order to not only maximize the comfortability of the user using the controller but also thinking about expandability and maintainability. Consequently, the following input's mapping has been made:



Figure 35. Mapping of all the available inputs of the controller.

One of the main objectives of making this separation of concepts between the different scripts is to minimize the code duplication and to minimize the appearance of bugs and errors that can cause conflict between the devices and even making the scripts stop running therefore leaving the motors without any motor movements.

1. Introduction

To begin with, the first lines of the scripts to process the input by the user's controller is the import's statement as seen below:

```
import serial
import pygame
import time
```

In Python import's statements are used to use pre-established code that someone written in order to use their functions and methods in the script. In this case it imports serial, which is a Python library for communicating through the serial ports which are the USB's on the Raspberry Pi. This library is clue for being able to send the processed inputs to the Arduino correctly. Furthermore, pygame is another Python library that facilitates the use of controllers with the Raspberry PI. This library is primarily used to code games and GUI but in this case, pygame helps by bringing some functions and methods that make processing the controller inputs easier.

Finally, to close the import section, the time's library enables Python to pause the program execution when need it and some other functions and classes to control time in the execution.

2. Main part

```
pygame.init()
j = pygame.joystick.Joystick(0)
j.init()
print('Detected controller : {}'.format(j.get_name()))
```

In the introduction part the controller must be declared in the script to be able to send inputs and be retrieved correctly. It declares a variable called `j` which is a `Joystick` class representing the controller that is plugged in. Then it initializes the controller to be able to use along in the script.

Finally, it prints to the terminal or console the detected controller if any is plugged in the Raspberry Pi. The print statement does nothing regards the actual execution of the program but is good to have it for testing purpose or in case the controller does not work, it can be because the script does not recognize well the controller in first place.

3. Testing and communication

```
serial_is_on = False
serial_port = '/dev/ttyACM0'
baud_rate = 2000000

if serial_is_on == True:
    ser = serial.Serial(serial_port, baud_rate, timeout = 1)
```

To continue, it declares a variable called `serial_is_on` which holds a Boolean value depending or where the Arduino is connected or not.

This is used for testing purposes because of when the Arduino is not connected and want to test if the scripts process correctly the inputs, it is not necessary to have the Arduino connected possibly receiving wrong values since it is entirely for testing purposes.

`Serial_port` is referring to the physical connection between the Arduino and the Raspberry and the name that the operating system gives to that connection. In Linux the path system is a bit different but in the Figure 36 it can be seen an alternative for Windows operating system

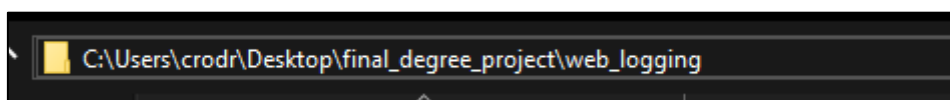


Figure 36. Windows OS path system.

Finally, for the declaration part the baud rate refers to the amount of changes per second that the USB will support, in other words, the amount of data that the Raspberry can send, and the Arduino will receive. This variable is the most important or one of the most important because it gives the rapidness or slowness of the process of sending and receiving the data between the Raspberry Pi and the Arduino. If the baud rate happens to be set very low, a delay will be created between the user making inputs on the controller and the actual motors moving because of that inputs, which can produce frustration since the ROV is acting with delay according to a set of instructions.

The following lines refers to the testing part. If the serial is on a serial object is created holding all the information commented above. Otherwise the object is not created and therefore the connection is not established.

4. Checking controller's axis

```
axes_to_check = [1,4,2,5]
breakout_button = 3
directions = ['L', 'R', 'U', 'D']
```

To understand better the part where the scripts obtained data based on the inputs that the user make, see in which it can be seen an auxiliary script that maps every button and joystick position of the given controller and let's see the "python value" that represents each and every input makeable in the controller.

```
Axes: 0: 0 1: -8108 2: 0 3: 0 4: 0 5: 0 6:-32767 Buttons: 0:off 1:
Axes: 0: 0 1: 0 2: 0 3: 0 4: 0 5: 0 6:-32767 Buttons: 0:off 1:
Axes: 0: 0 1: 8445 2: 0 3: 0 4: 0 5: 0 6:-32767 Buttons: 0:off 1:
Axes: 0: 0 1: 24998 2: 0 3: 0 4: 0 5: 0 6:-32767 Buttons: 0:off 1:
Axes: 0: 0 1: 32767 2: 0 3: 0 4: 0 5: 0 6:-32767 Buttons: 0:off 1:
Axes: 0: 0 1: 675 2: 0 3: 0 4: 0 5: 0 6:-32767 Buttons: 0:off 1:
Axes: 0: 0 1: 0 2: 0 3: 0 4: 0 5: 0 6:-32767 Buttons: 0:off 1:
```

Figure 37. Jstest program running

As commented before, the only inputs that the script will be able to recognize, and process are those related two the two joysticks and the buttons labelled as R2 and L2 since they are selected to be the 4 principal movements. Is important to keep the focus on the main objective which is the ROV's movement. This is one of the reasons why only 4 inputs are checked, because are the principal ones as commented before and seen in Figure 37 when we are talking about control distribution in the controller and the best buttons for maximum efficiency.

As it can be seen in Figure 37 joysticks are labelled as 1 and 4, while the R2 and L2 buttons are labelled as 2 and 5. This axis are being save as a list in Python, or as a 1-dimension vector equivalent. It declares also a breakout button, which will stop execution completely, in case of testing, or when the ROV is fully armed, to turn off the motors and save battery. This button is labelled as the square in the controller, and it is given a 3 in the auxiliary script as seen in Figure 37.

5. Pre-processing controller's inputs

```

while True:
    pygame.event.pump()
    recent_values = []
    serial_values = []
    for current_axis in axes_to_check:
        latest_value = j.get_axis(current_axis)

        if current_axis == 2:
            serial_value = int(960 + (((1+latest_value)/(2+2)*1000)))
            serial_value = str(serial_value)
            recent_values.append(serial_value)
        elif current_axis == 5:
            serial_value = int(960 - (((1+latest_value)/(2+2)*1000)))
            serial_value = str(serial_value)
            recent_values.append(serial_value)
        elif current_axis == 1:
            serial_value = int(1460 - (((1+latest_value)/2)*1000))
            serial_value = str(serial_value)
            recent_values.append(serial_value)
        else:
            serial_value = int(1460 - (((1-latest_value)/2)*1000))
            serial_value = str(serial_value)
            recent_values.append(serial_value)

```

In this part finally the program can process the inputs make by the controller. The objective of the script is to run continuously without stopping unless the breakout button is held as it will be seen later. To make the script run continuously it uses the while statement, in which the code inside the with statement will run only if the condition met is true.

In this case while True always will return True, so the code inside will run always. This approach is the cleanest but also the most dangerous since if the breakout button does not exist the program will run infinitely costing resources to the Raspberry and ultimately causing lag and ever more delay.

The while code must go through each value in the vector and checked the value each time repeatedly. To do that the for statement is used because of it is need it to iterate or loop though every single axis to see if an input has been made.

The new axis value is saved in another 1-dimensional vector called recent values and it is processed. The main objective is to transform the raw input of the controller into some value that is recognizable by the Arduino and can be sent to activate or turn off the motors. To better understand how the motors work, in Figure 38 it can be seen the activation range of the motors in microseconds (μ s):

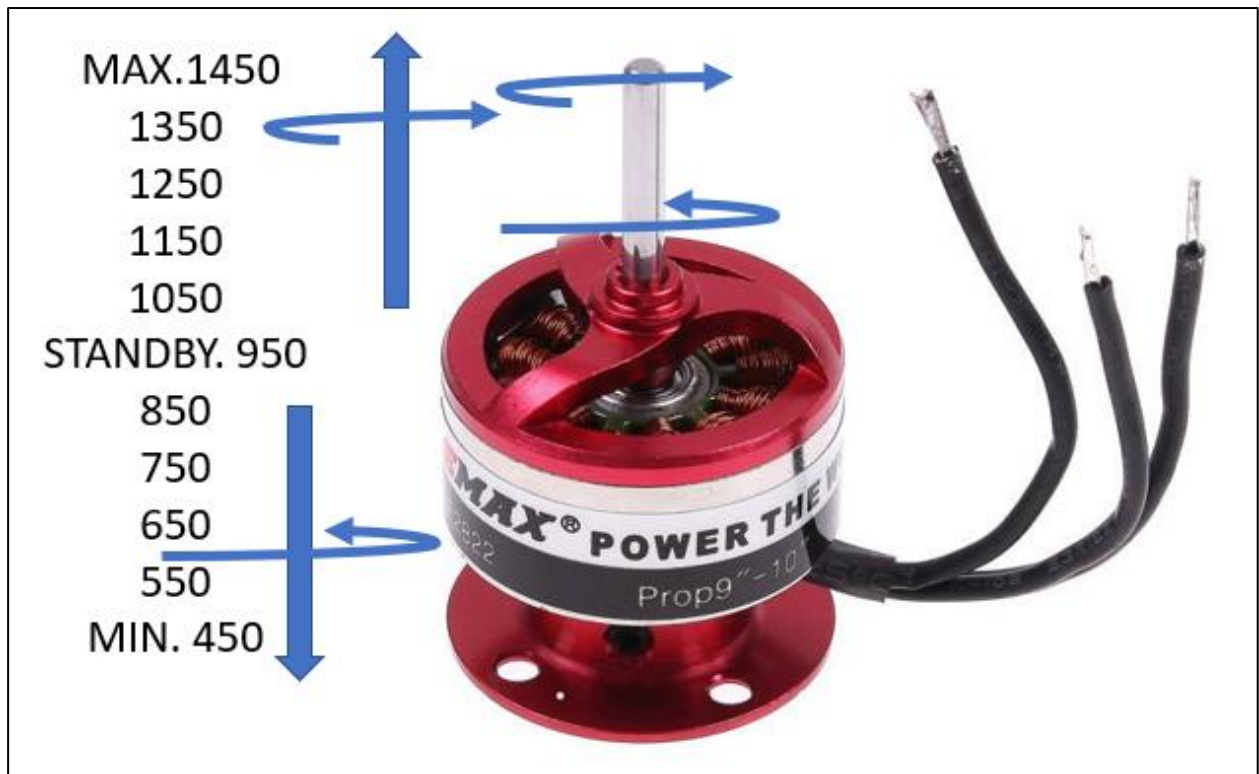


Figure 38. Diagram showing the different values and the corresponding rotation of the motor.

So, the main objective as seen above in the Figure 38 is to send to the Arduino the axis values compressed between 450 and 1450 to see if the motor must move or stay stopped. The raw controller's input is nothing like that and because of that there is a need to transform, to process those raw values into something that the Arduino can processed.

To be more accurate, the controller joysticks gives a value between -1 and 1 depending on the position that are hold on, and the axis of the movement (horizontal or vertical). The same happens to R2 and L2 buttons, they are what is called triggers buttons and have a larger travel than other buttons in the controller. Those ranges from -1 when it is not pressed all the way to 1 when is fully pressed. To better grasp and visualized those values, Figure 39 is shown below:

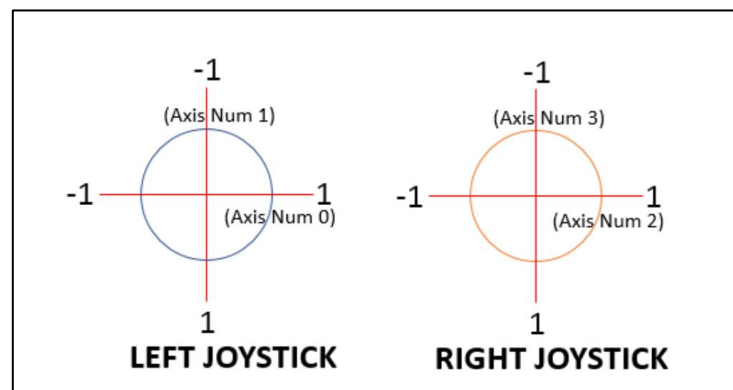


Figure 39. Raw values of the controller's joysticks. Source: robotsandphysicalcomputing.com

Now that the raw values of the joysticks are known the program needs to transform those values into ranged values from 450 to 1450 and depending on the button pressed limit how far this range is going to be. The joysticks must be mirrored one from the other because of the torque produce by the propeller. If both motors turn in the same direction, they will produce a torque resulting in a deviation of the ROV direction which is not desired. Therefore, if one propeller rotates left-handed the other propeller must rotate right-handed to cancel each other, cancelling the torque, as shown in Figure 40.

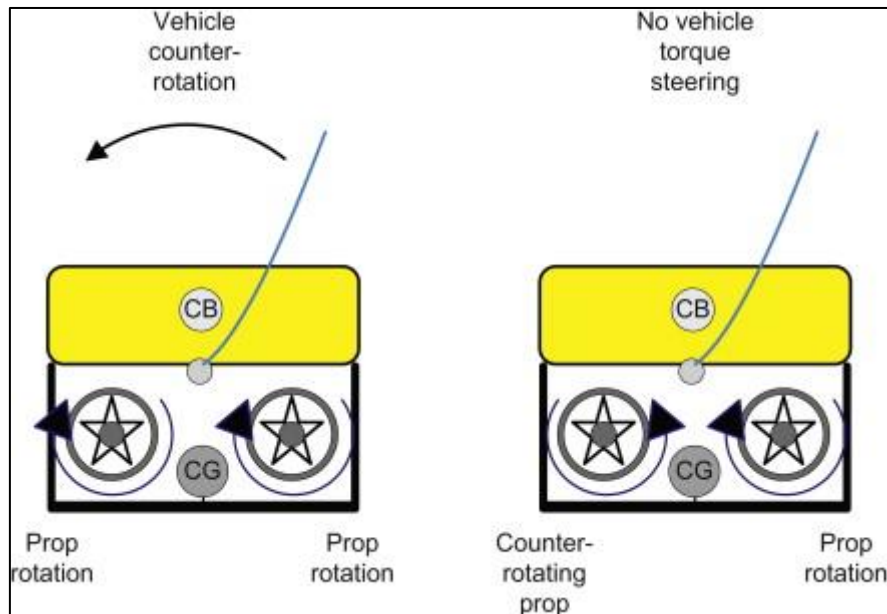


Figure 40. Torque visualization in the forward movement of the ROV. Source: Robert D. Christ, Robert L. Wernli Sr., in The ROV Manual (Second Edition), 2014

To R2 and L2 propellers happens the same, but as mentioned before, one button will be map to two propellers, so when L2 is pressed for example, both propellers go down and when R2 is pressed both go up again. Consequently, both propellers also must be mirrored to cancel the torque mentioned before.

In conclusion, 4 equations are needed to fully process the raw inputs and outputs the propeller's value that are needed in each situation. 2 equations will be mirrored because of the joysticks, and the other two will be limited since to go up or go down they do not need the full range of values. The equations can be seen below:

$$movement = 960 + \left(\left(\frac{1 + latest\ value}{4} \right) * 1000 \right)$$

$$movement = 960 - \left(\left(\frac{1 + latest\ value}{4} \right) * 1000 \right)$$

$$movement = 1460 - \left(\left(\frac{1 + latest\ value}{4} \right) * 1000 \right)$$

$$movement = 1460 - \left(\left(\frac{1 - latest\ value}{4} \right) * 1000 \right)$$

Also attached in Table 9 the different outputs expected for the 4 different equations:

Inputs	Movement	Movement 2	Movement 3	Movement 4
1	1460	460	460	1460
0,9	1435	485	510	1410
0,8	1410	510	560	1360
0,7	1385	535	610	1310
0,6	1360	560	660	1260
0,5	1335	585	710	1210
0,4	1310	610	760	1160
0,3	1285	635	810	1110
0,2	1260	660	860	1060
0,1	1235	685	910	1010
0	1210	710	960	960
-0,1	1185	735	1010	910
-0,2	1160	760	1060	860
-0,3	1135	785	1110	810
-0,4	1110	810	1160	760
-0,5	1085	835	1210	710
-0,6	1060	860	1260	660
-0,7	1035	885	1310	610
-0,8	1010	910	1360	560
-0,9	985	935	1410	510
-1	960	960	1460	460

Table 9. Output for the different movement's equations.

```

if current_axis == 2:
    serial_value = int(960 + (((1+latest_value)/(2+2)*1000)))
    serial_value = str(serial_value)
    recent_values.append(serial_value)
elif current_axis == 5:
    serial_value = int(960 - (((1+latest_value)/(2+2)*1000)))
    serial_value = str(serial_value)
    recent_values.append(serial_value)
elif current_axis == 1:
    serial_value = int(1460 - (((1+latest_value)/2)*1000))
    serial_value = str(serial_value)
    recent_values.append(serial_value)
else:
    serial_value = int(1460 - (((1-latest_value)/2)*1000))
    serial_value = str(serial_value)
    recent_values.append(serial_value)

```

This part divides the inputs according to the axis that is being read and computes the serial value for the raw input on that moment.

a. Communicating with Arduino: Building outputs

```

i = 0
for direction in directions:
    serial_values.append(str(recent_values[i]) + direction)
    i += 1
serial_output = "".join(serial_values)
print(serial_output)

```

These values are not valid and its need a better and concise way to send all the values in one stream concatenating all the values into a single string (group of characters) according to the following structure present as seen below in Figure 41:

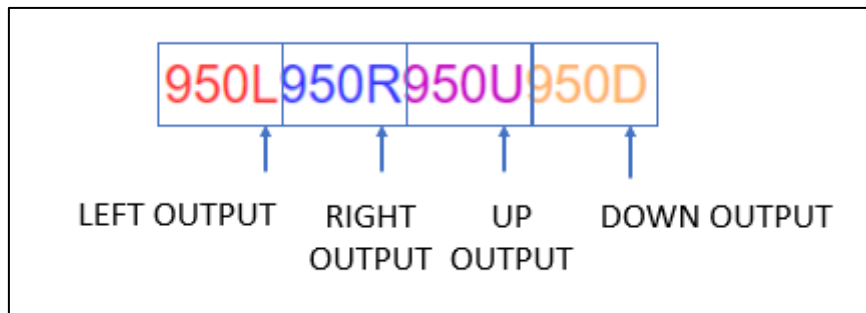


Figure 41. Result output string send over the serial port.

```

if serial_is_on == True:
    ser.write(serial_output)

if j.get_button(breakout_button) == 1:
    break

```

Finally, the scripts check if the serial is on variable is set to true, and in that case, it writes to the serial the value seen in Figure 41, in other words, the data is sent through the serial to be received later by the Arduino. In case of the users holds the breakout button, the execution stops completely.

6. Arduino's script for processing the Raspberry's output and moving the motors.

To successfully move the motors, it was necessary to pre-process the raw inputs from the controller in order to have readable value that can be interpreted by the motors aboard the ROV. However, in order to write to this motor and make them move, the Arduino must receive the serial outputs produce by the Raspberry and then choose which motor must move according to the value it is receiving from the Raspberry.

1. Introduction

```
#include <Servo.h>
```

As in Python there is the import statement which clarifies that there will be using libraries written by other persons, in C++ the include statements makes the same effect as seen in Python but written in C++ syntax. In this case and for the script in charge of controlling correctly the motors, only a single library is need it (Servo.h). Servo.h is the most known library for working with servos and motors in Arduino. It provides Servo object with different functions and methods to write and make the motors run as the input continue to go in the script.

```
Servo esc_left;  
Servo esc_right;  
Servo esc_up_down;
```

After includes servo's library, it defines 3 different motors with the name variable `esc_left`, `esc_right` and `esc_up_down` where `esc` stands for the electronic speed controller used to connect the motors to. As it can be seen already, in C++ because of the strong type language is need it to specify the type each variable represents in the whole context of the script. In this case ther is three objects with the Servo's types and therefore it cannot be changed during the script execution.

Consequently, it declares the different variables that will hold the thruster/propeller value to be written in the motors.

```
int thruster_left;  
int thruster_right;  
int thruster_up;  
int thruster_down;
```

As it can be seen there are 4 variables but only three servo's objects. The reason behind that is that two motors will hold the `esc_up_down` variable because as mentioned before, the L2 and R2 must move two motors with one input. Consequently, it is necessary to have two variables that holds up and down values but only one servo to represent the up or down value. Wiring connection visualization can be seen in Figure 42 to visualized better the variables purpose in the program.

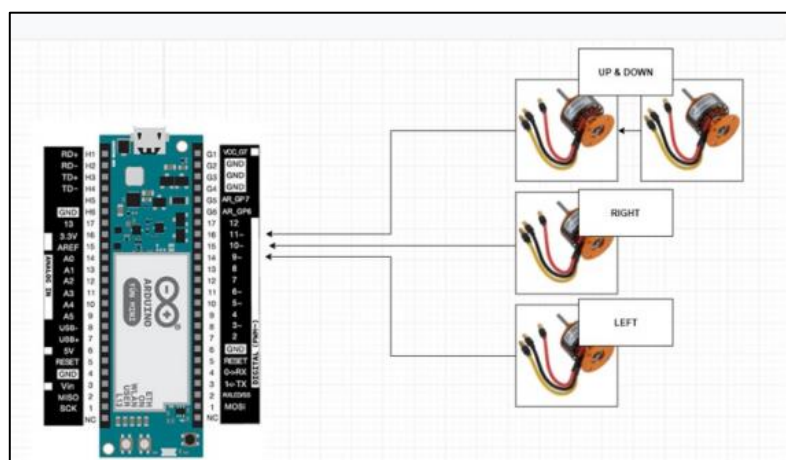


Figure 42. Motors' connection to the Arduino.

a. Void setup

One of the most important parts of the script execution in Arduino. The void keyword express that the function called setup does not expect to return any information during the execution.

```
void setup() {  
    Serial.begin(2000000);  
    esc_left.attach(9);  
    esc_right.attach(10);  
    esc_up_down.attach(11);  
}
```

In Arduino, the setup's function is used to prepare the different pins of the microprocessor as well as setting up the baud rate which in this case is very important to be equal to the one that is written in Python's script. The reason behind is that the baud rate as mentioned before holds the amount of changes that the Arduino can receive or send per second, by his ports or the pins. In this case, the baud rate is set to the same amount as the Python script to minimize errors, and to ensure that the Python script is capable of sending the most data and the Arduino can receive as much data as need it to try to minimize the delay between input's user on the controller and the motors moving.

Furthermore, the different servo objects are attached to the different pins of the Arduino, that will oversee sending signals to the different motors. In this case the servos are setup in pins 9, 10 and 11. The wiring connection can be seen in the Figure 42 shown above earlier.

b. Void loop

As the setup was the part of the execution that was in charge of setting up the different parts need it though the execution, the loop's function is the heart of the script, in which all the code that is want to be repeated is placed here. As the name implies the loop's function runs continuously, in this case in order to be able to send signals to the different motors as rapid as possible minimizing the delay as much as possible.

```
if (Serial.available()) {  
    bytesAvailable = Serial.read();  
    charAvailable+=char(bytesAvailable);
```

Firstly, to see if any data is buffered in the serial waiting to be process by the Arduino, Serial.available() is used which returns true or false in case of being no data buffered in the Arduino ports. Serial available is used to make the program more efficient and make the logic to signal motors only when there is data waiting in the serial port. With that the scripts do not run saving up resources and possibly future errors if there is no signal data to be passed to the motors.

When data is buffered in the Arduino's port waiting to be processed the logic afterwards can be executed.

```
bytesAvailable = Serial.read();  
charAvailable +=char(bytesAvailable);
```

One of the main characteristics of the Arduino's Serial is that with the function Serial.read() only returns the first byte incoming data available, which is stored in an int variable called bytesAvailable referring to

the bytes that are available to be processed by the script. This means that the `Serial.read()` function only reads one character of the stream mentioned and visualized in the Figure 43 visualizing string. Consequently, to be able to make one single motor move the loop function must run more than one time, but due to efficient code written, this process is made very rapidly. To better visualize what is to only read the first byte of data, Figure 43 can be seen below.

```
Example string: 950L1050R960U1000D
9 ← Reads one byte at a time
95
950
950L
    It detects the L for left and resets the string
1
10
105
1050
1050R
    It detects the R for left and resets the string
.
.
.
```

Figure 43. Program execution using an example output string from the Raspberry Pi.

In other words, the stream that the Raspberry Pi sends must be reconstructed byte per byte, and that is stored in `charAvailable` firstly casting the `bytesAvailable` type to `char` (character). Not confuse casting a variable with dynamic typing, there are two different terms and the possibility to cast variables does not mean that the language is less strongly typed. On Figure 44, you can see the difference between casting and dynamic typing variables.

Dynamic typing	Casting types
<pre>dog_age = '34' dog_age = True</pre>	<pre>char dog_age = "25"; int number_dog_age = (int) dog_age;</pre>
We can change the type of the variable without specifying the result type	We can change the type of the variable specifying the output type

Figure 44. Dynamic typing vs casting types.

While the stream is reconstructing itself, the script check if in the last byte there is a char belonging to L for left, R for right, U for up and D for down. These are the separators for the values send by the Raspberry PI, like in a csv (comma separated value) the separator between values ordinarily is the comma. In this case each separator represents the processed value for one of the motor's direction. To better explain the logic behind the script processing values, we will look at left directions and down directions, since R and U have a similar logic and it will be inefficient to comment them.

c. Left, right and up movement

```
if (charAvailable == 'L') {  
    thruster_left = charAvailable.toInt();  
    charAvailable = "";  
}
```

If the incoming byte read is a L character, the value of the left thruster will be the actual variable charAvailable cast to an integer value (natural numbers). Consequently, the charAvailable variable is erased to reset the stream and wait for the next direction that will be the right direction.

d. Down movement

Since the D character is the last character to be retrieve, all the logic that goes into moving the actual motors is placed here. Consequently, the logic's part behind the D character is a bit larger. The first part of the logic is the same as the rest of thrusters' directions.

In the second part, there is commented code because of it's for testing purposes. Alike the Python script this code soul for purposes must be commented if not use, because of uses Serial.prints which are functions that prints to the serial console the variable's value at a single moment in time. These serials.prints buffered and pause execution for each function call causing an enormous delay between user's inputs and motors action.

Finally, for the right and left thrusters, the writeMicroSeconds function is called allowing the values of each variable to be passed into each of the motors in charge of right and left thrusters. The writeMicroSeconds writes the position given to the motors shaft, being in this case express in microseconds (μ s).

```
esc_left.writeMicroseconds(thruster_left);  
esc_right.writeMicroseconds(thruster_right);  
  
if((thruster_up != 950) && (thruster_down != 950)) {  
    //Serial.println("DONT MOVE");  
} else {  
    if((thruster_up == 950) && (thruster_down!=950)) {  
        //Serial.println("DOWN");  
        esc_up_down.writeMicroseconds(thruster_down);  
    } else {  
        //Serial.println("UP");  
        esc_up_down.writeMicroseconds(thruster_up);  
    }  
}
```

In this last part the scripts compare the two values for thruster's up direction and down's direction. If the two triggers, L2 and R2 are pressed the script will result in a blank line of code representing that no motor moves. If the trigger for down direction is pressed while the up is not, the motor will be going down and vice versa.

2. Data measurements' script

As seen before in Figure 45 in which can be seen the wiring between all the components, all the sensors and measurements tool were connected to the Raspberry GPIO's, so because of that this last script for the inside ROV's software has been written back in Python. There are different sensors that must consider in order to make the script process correctly the distinct measurements:

- Temperature, which can be express in Fahrenheit or Celsius. For this project the temperature will be taken as Celsius.
- Pressure, which can be express in Pascal. For this project and considering the low depth that the ROV is able to achieve, pressure will be measured as Pascal.
- Humidity, which is express in this case in percentage, referring to the humidity levels inside the electronics' tube.
- Acceleration, gyroscope and magnetometer measures can give us some information about the position and orientation when the ROV's moves.

After considering all the sensors that are available, it has been decided to make a general script containing all the logic for the data measurement's collection and pre-processing. Furthermore, considering that this script contains logic part referring to the communication with the database, it has been decided to split it up in this section and focus on only on the collection's part of each sensor in order to follow a clearly order. Later in the next section when we talk about the different systems outside the ROV, we will recover part of this script since contains communication between the interior software of the ROV and the outside, more concise, the database connection between the ROV and the GUI.

a. Temperature

Recollecting temperature measurements is simpler that recollecting other measurements data like the acceleration, magnetometer or gyroscope. Despite of being simpler to code in thanks to the libraries that Python and the community provides, wiring all the connections is more important than coding the actual recollection of data. In Figure 45 it can be seen the wiring connections to make the sensor work with the Raspberry. As mentioned in other chapters, the sensor which the ROV will be working on is the DS18B20 and it is waterproof.

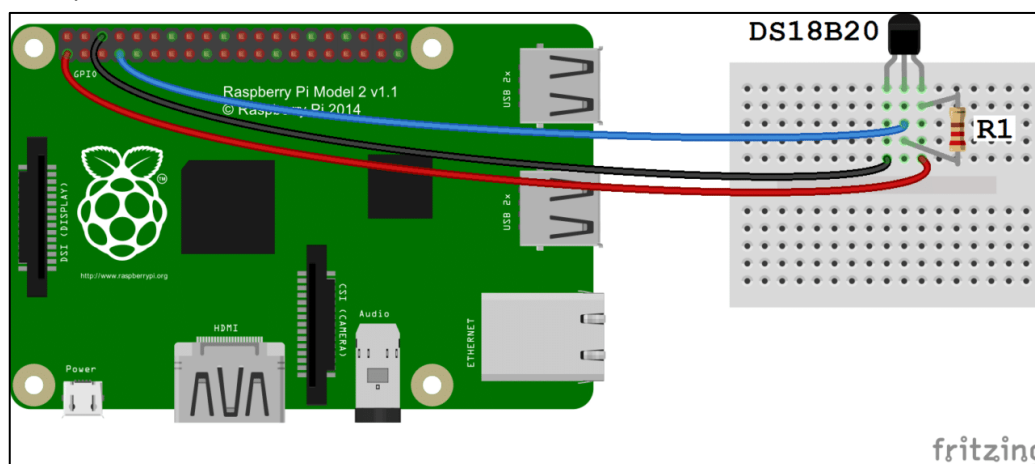


Figure 45. Connection diagram of the temperature's sensor DS18B20. Source: Circuitbasics.com

In this case the resistance is rated at 4.7 kΩ. The reason behind the resistance placement is to make it a pull-up resistance. A pull-up resistance is simply a resistance placed in a different configuration, which leads to establishing a logic state in one of the pins or a logic input when the circuit is in standby. In this case, the pull-up resistance establishes a HIGH state when the pin is in standby. Therefore, reducing false states produced by the electronics' noises.

In the case of pull-up resistances, when the circuit is not in standby, all the current is derived to mass and the fall of voltage is 0V (LOW). In the other case, when the circuit is in standby the voltage difference is 5V (HIGH). In conclusion, this resistance only reduces the effects of external noises produced by the circuit, then affecting the precision of our measurements. In Figure 47 it can be seen the diagram for this pull-up and down resistances, which will lead to a better understanding on the effects of placing one of them in our circuit.

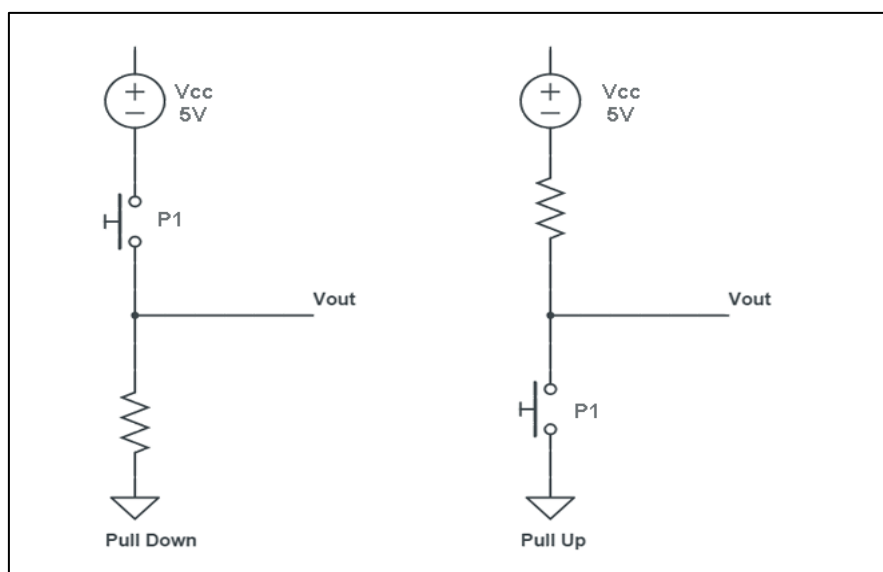


Figure 46. Pull-up and pull-down resistance

Script for temperature's measurements

As mentioned before, to make every sensor clear, the general script will be split into the different sensors in order to make an easier understanding of what are doing each one of the sensors present at the ROV.

```
from w1thermsensor import W1ThermSensor
temperature_sensor = W1ThermSensor()
temperature = temperature_sensor.get_temperature()
```

As seen before, the import statement let us take part of code that someone else wrote to use it in the code in order to reduce the code's duplication. In this case, from the w1thermsensor the object W1ThermSensor is imported. To sense the temperature in the ROV an object must be created. This object is called temperature sensor. Now that the object lives on the variable called temperature_sensor, the variable gain access to the objects' methods and properties. In this case, the only need is to get the temperature in Celsius. In order to measure the temperature in Celsius, the get_temperature method is called, which returns the actual temperature measure by the sensor in Celsius.

Is not the case, but with this library can also measure the temperature in Fahrenheit by passing as an argument of the method the property called `W1ThermSensor.DEGREES_F` which will result in:

```
temperature_in_fahrenheit = sensor.get_temperature(W1ThermSensor.DEGREES_F)
```

b. Humidity

The sensor in charge of measuring humidity is a soil sensor with two brackets that let us measure the humidity. The problem we were facing during the testing of the humidity sensor connected to the Raspberry is that the Raspberry does not accept analogic inputs at all. That means that the humidity sensor that we have now only produces an analogic output:

- 0 when it does not detect water.
- 1 when it detects water.

This can be also interesting to measure the humidity by binary levels, HIGH or LOW, but the problem with that is precision. If the humidity sensor was mounted to measure humidity in an analogic way, the results will first very inaccurate and second will be difficult to analyse or conceptualize, because of some water can entered the tube but not affect the system in a wrong way. With the analogic measure, the ROV will always make false assumptions of danger inside the tube, when possibly, only a drop of water falls into the sensor. Consequently, it has been decided to turn the analogic outputs into digital ones in order to be able to communicate with the Raspberry correctly. To archive this transformation, it has been chosen to use an MCP3008. The MCP3008 is simply a low cost 8-channel 10-bit analog to digital converter. The precision is similar to that found on the Arduino's family. The MCP3008 is precise enough to be able to convert the signals and gives accurate results. In Figure 47 shown below can see the wiring for the humidity sensor on the Raspberry Pi.

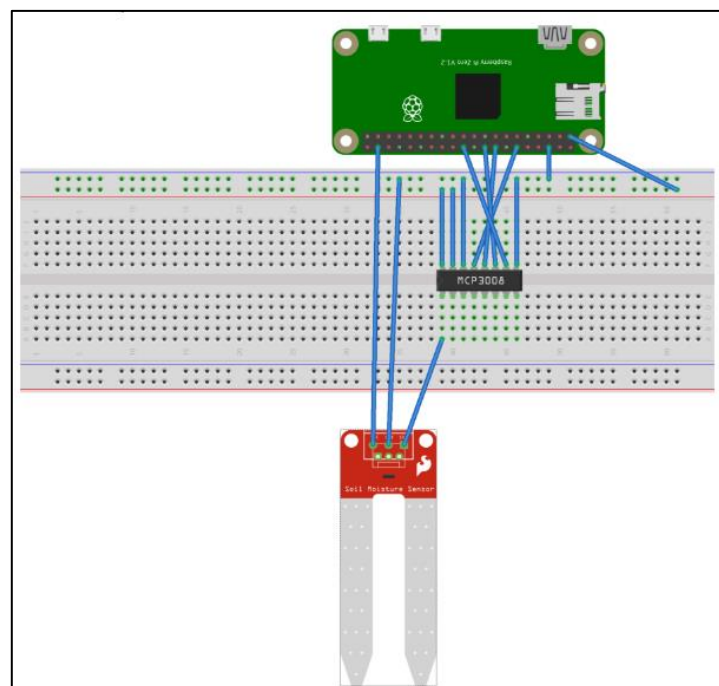


Figure 47. Connection diagram between the soil sensor and the MCP3008 with the Raspberry Pi. Source: github.

Script for humidity's measurements

try:

Main program loop.

while True:

GPIO.output(SWITCH, GPIO.HIGH)

time.sleep(0.1)

value = float(mcp.read_adc(0))

print("The soil moisture reading is currently at {:.2f}%").format(value / 1023 * 100)

GPIO.output(SWITCH, GPIO.LOW)

time.sleep(10)

except KeyboardInterrupt:

GPIO.output(SWITCH, GPIO.LOW)

GPIO.cleanup()

c. MPU-9250: Accelerometer, gyroscope and magnetometer

The MPU-9250 is a 9-axis accelerometer, gyroscope and magnetometer. This sensor is used in the ROV to see it is positioned every time. With the accelerometer, the script can determine the velocity at which is moving the ROV, therefore knowing the speeds of the motors. On the other hand, the gyroscope gives information about rotation, so underwater in which only can see the camera and not the full ROV, it can be conceptualise the actual position and rotation that the ROV has underwater.

In Figure 48 can be seen the wiring for the MPU-9250 in conjugation with the Raspberry Pi.

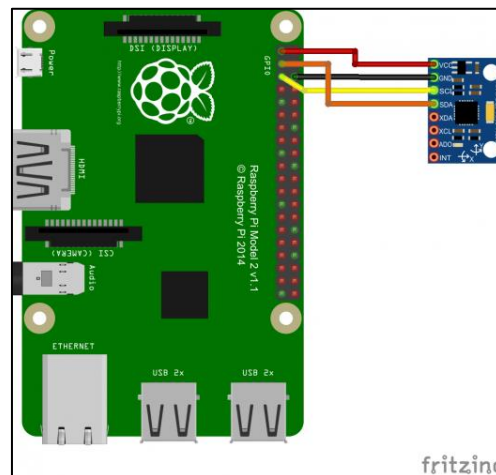


Figure 48. Connection between the MPU-9250 and the Raspberry Pi. Source: RaspberryPi.com

Script for accelerometer, gyroscope and magnetometer's measurement

As it can be seen in GitHub

In the first part of the script, the math module is imported using the import statement. This library gives all mathematical functions needed later to compute the rotations and accelerations. The first three functions are pre-set to be able to read the values from the MPU-9250.

The next two functions are only to be able to calculate rotations using 3 axes. The formula to compute the different rotation in x and y axis is shown below:

$$\text{Rotation in } X = \arctan\left(\frac{X}{\sqrt{X^2 + Z^2}}\right)$$

$$\text{Rotation in } Y = \arctan\left(\frac{Y}{\sqrt{X^2 + Z^2}}\right)$$

Finally, to see the different magnitudes for the accelerometer and gyroscope, only must convert the results to the international system and then calculate the rotation with these values:

```
gyroscope_x = read_word_2c(0x43) / 131
```

```
gyroscope_y = read_word_2c(0x45) / 131
```

```
gyroscope_z = read_word_2c(0x47) / 131
```

```
accelerometer_x = read_word_2c(0x3b) / 16384.0
```

```
accelerometer_y = read_word_2c(0x3d) / 16384.0
```

```
accelerometer_z = read_word_2c(0x3f) / 16384.0
```

```
rotation_x = get_x_rotation(accelerometer_x, accelerometer_y, accelerometer_z)
```

```
rotation_y = get_y_rotation(accelerometer_x, accelerometer_y, accelerometer_z)
```

7.3. Building the GUI and communicating the ROV with the database and outside systems

The outside systems that enables the user to use properly the ROV's sensors as well as their camera are also important and worth to be mentioned. The outside system is composed by the Raspberry, a basic UI website and a database. A diagram for better visualization can be seen in Figure 34.

The insides of how the websites works and the different options and features present are present in the GitHub project for the ROV. In this section it will talk about a general overview of what is archived, and the different features implemented to make the ROV's users more comfortable and useful while using it.

As mentioned before all the measurements recollected by the different sensors are processed throughout the Raspberry Pi and send it through an Ethernet cable to the database to be stored. This database is one of the most important aspects in this outside system since it is received different inputs from the ROV being the different measurements recollected and stored it safely to be recovered later.

The database builds on top of MySQL, a query language with can access the database to retrieve some information. This retrieval of information can be done directly over the database, but it is also dangerous to give permissions to anyone to access the database, since they can delete some important information or measurements recollected during a ROV's investigation. Due to security measures and accessibility to everyone not knowing how to execute this query sentences or simply because of the software complexity, it was decided to build a website around the database to give users the possibility to access the database in a control and secure way ensuring the protection of both the ROV data and the users themselves.

On the other hand, we think that the pillars that defends the reason behind the website building, is a part of the database's security, is the accessibility and the easiness of being capable of interact with buttons in a website. Moreover, the expandability and the maintainability of the project were also reasons to build a simply but powerful interface to interact with the database and make future changes regarding to the information given by the ROV's sensors.

Before explaining the main features of the visual interface that is being created, the Figure 49 shows the raw aspect of trying to retrieve some measurements from the database using MySQL.

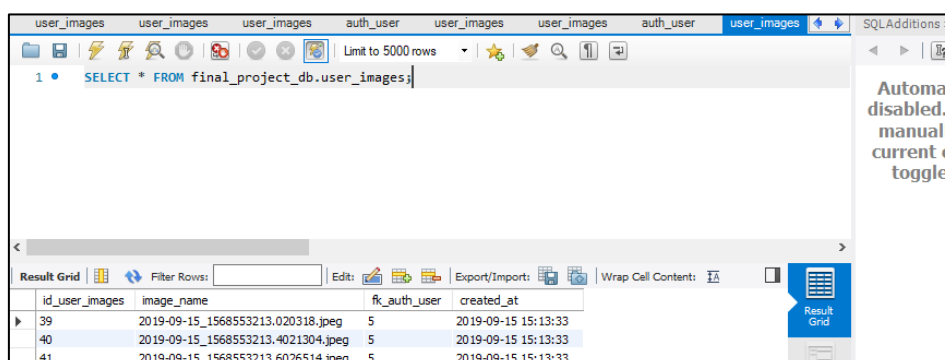
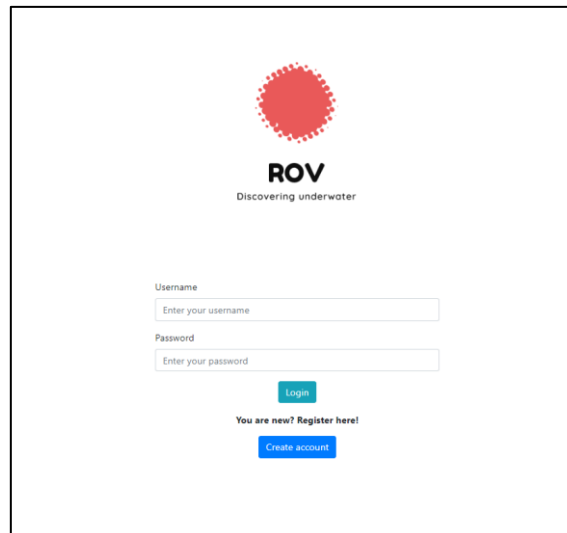


Figure 49. Raw query sentences over a database.

The website as well as the visual interface is local hosted, in other words, it runs on a local server in a pc. It is not available on the internet, so the URL to access it works only in the pc with the server working and turned on. To explain more about the different features that the web provides, we divided it into different sections.

Login system

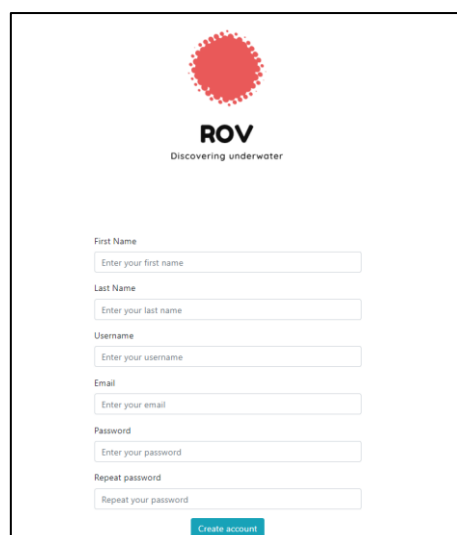
As it is mentioned before, one of the most important reasons to build a website to visualize the data was protection and security. Due to that reason, the UI have a complete login system which recognize the user and let's sign in if the username and the password are correct. Later when we talk about the ROV's camera, the login system will be more understandable. If a new user has to gained access to the URL can be sign up using the 'Create Account' button shown in Figure 50.



The image shows a login form for the ROV website. At the top center is a red circular logo with a dotted pattern, followed by the text "ROV" and "Discovering underwater". Below this, there are two input fields: "Username" with the placeholder "Enter your username" and "Password" with the placeholder "Enter your password". A green "Login" button is positioned below the password field. Underneath the login button, there is a link that says "You are new? Register here!" and a blue "Create account" button.

Figure 50. ROV's website sign in form.

When the button is clicked you are redirected to another section of the page where the user has to fill up some information as well as established a new password for the incoming new user in the website. This login systems also counts with password matching in order to prevent the user to enter two different passwords. It also checks if the username is already taken and finally if all the process is handled correctly, a new username is registered in the database and his password is hashed for security reasons, since if a person gain access to the database cannot see the actual password that a user has. In the Figure 50 and Figure 51 can be seen both the sign-up form and the password of a new user hashed and protected against vulnerabilities.



The image shows a sign-up form for the ROV website. At the top center is the same red circular logo as in Figure 50, followed by "ROV" and "Discovering underwater". Below this, there are several input fields: "First Name" (placeholder: "Enter your first name"), "Last Name" (placeholder: "Enter your last name"), "Username" (placeholder: "Enter your username"), "Email" (placeholder: "Enter your email"), "Password" (placeholder: "Enter your password"), and "Repeat password" (placeholder: "Repeat your password"). A blue "Create account" button is located at the bottom center of the form.

Figure 51. ROV's website sign up form.

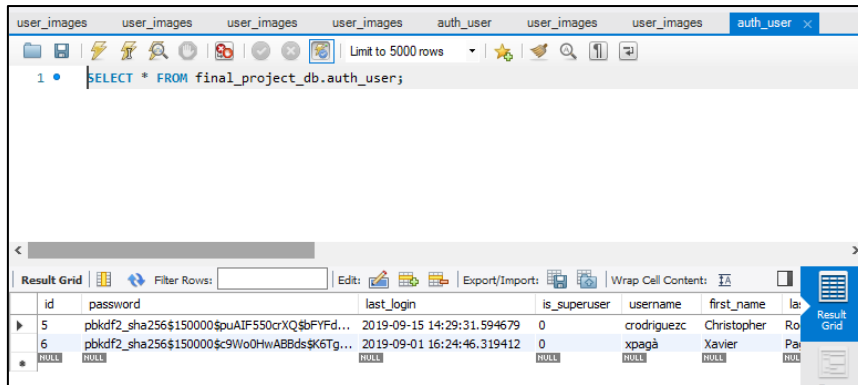


Figure 52. Database proof of hashed passwords.

Data measurements

When the login is made successfully the user is presented in the Data Measurements section, in which it can see the different sensors placed in a table as well as some explanation about the different sensors and wiring with the Raspberry to help the user visualize the different systems that are built in the ROV.

Furthermore, the table is updated every few seconds with the records that are being send it though the Raspberry scripts and are being stored in the database. The table retrieves the information about the sensors directly from the database which is being at the same time feed it with the data that the ROV is sending every few seconds, so the different measurements are being presented in real time. In the next figures it can be seen the different parts of the Data Measurement section:

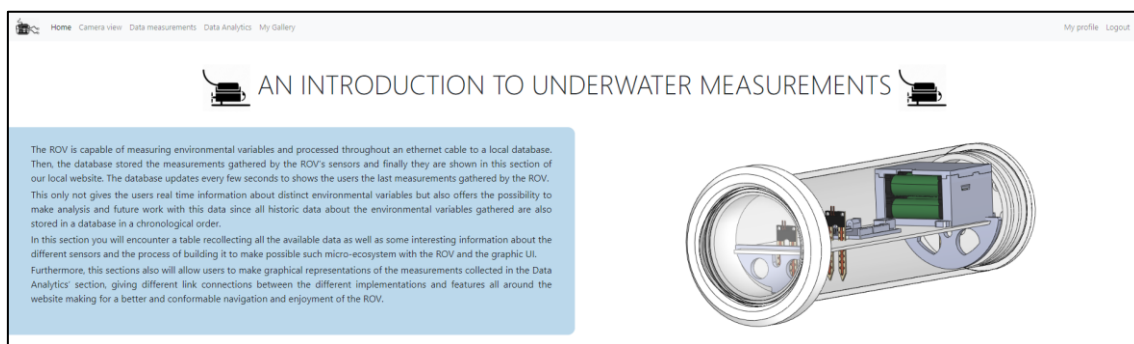


Figure 53. Main introduction to Data Measurements.

Date	Temperature (°C)	Pressure (Pa)	Humidity (%)	x_acceleration	y_acceleration	z_acceleration	x_gyroscope	y_gyroscope	z_gyroscope	x_magnetometer	y_magnetometer	z_magnetometer
2019-06-15T17:07:02Z	24	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:02Z	25	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:03Z	22	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:03Z	20	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:04Z	26	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:04Z	20	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:04Z	21	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:05Z	20	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:05Z	25	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626
2019-06-15T17:07:06Z	22	0	0	-22616	-22616	22671	13345	5322	17408	-14723	22569	10626

Figure 54. Measurements table with all the sensor data.

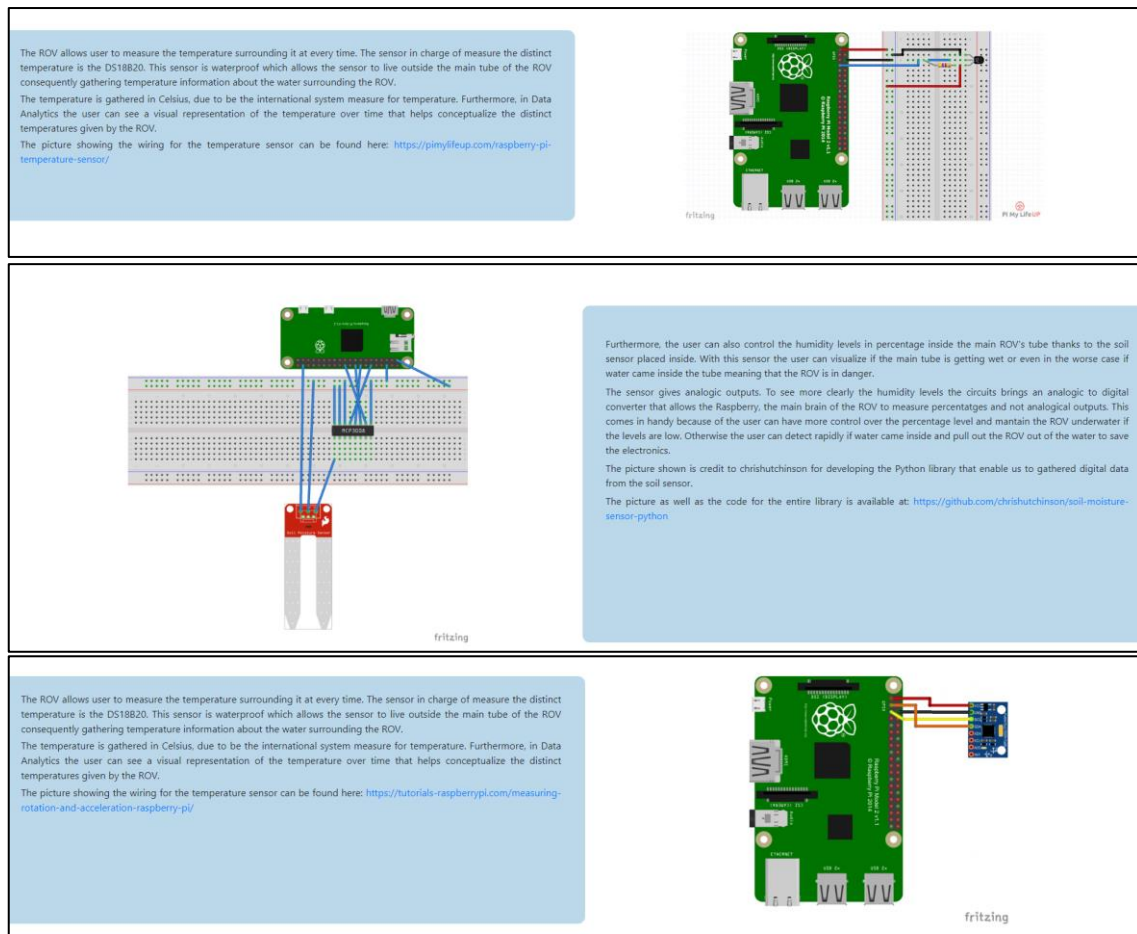


Figure 55. Diagrams and brief explanation of the different sensors composing the ROV.

Camera view

One of the other more important parts is to see exactly what the ROV actually sees. Without a camera it would be impossible to know the actual position or rotation that the ROV has. The camera view section connects directly with a USB camera that is recording at a real time. In the website this recording is visible, and it is at real time allowing the user to see exactly what the ROV sees at every single moment.

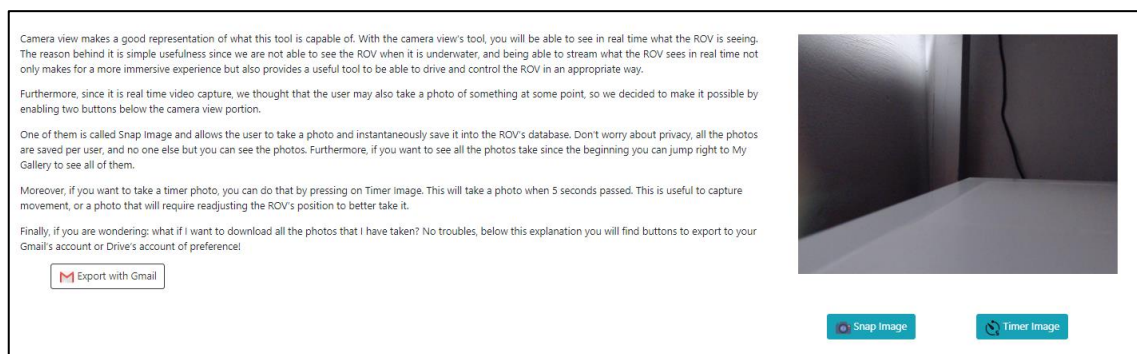


Figure 56. Camera view section.

Moreover, if the user wants to capture a picture, or sees something that is interesting to keep on, it can take a picture directly from the website picking the Snap Image button. This allows the user to interact with the ROV's system without the need to be underwater or must programme something more complex. The different photos are stored in the database with a randomly generated name in conjugation with the user that took the photo. Furthermore, a little preview can be seen below the buttons and can also be checked in My Gallery section which will talk about later.

The login system is useful in this kind of scenario, since a user can made photos that do not want to share or simply because they are took behind his user it will not to be share. The login system enables the possibility to save the picture in the database knowing which picture is from which user, consequently, only showing the picture that are from a specific user and not all of them.

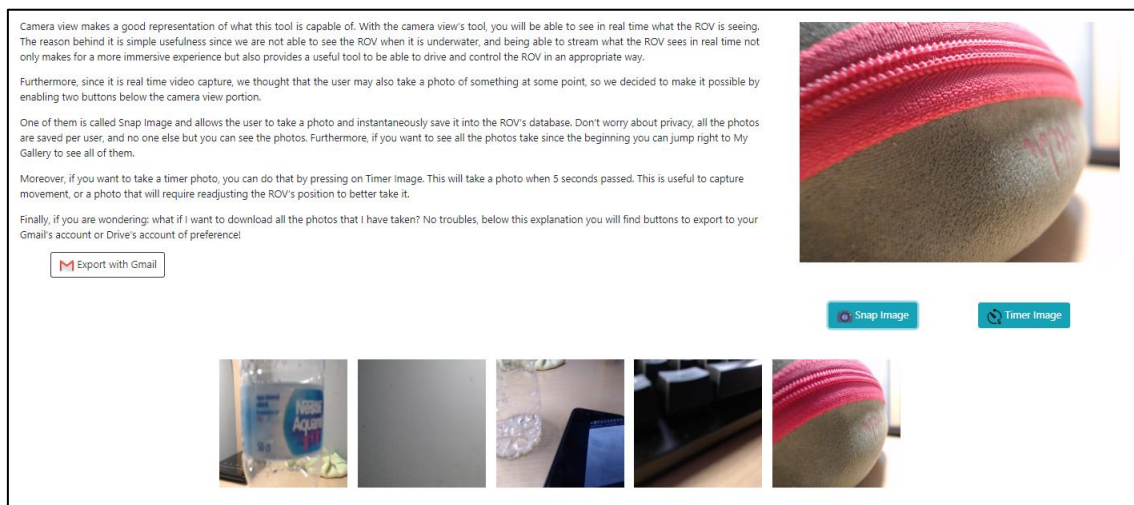


Figure 57. Image's preview.

Lastly, if the user, after an investigation has been completed, want to retrieve those images to make a study or simply to keep it on another platform, can do it use the Gmail's export button. This button allows the user to send reports with the different images took on a specific day that the user can choose.

The website automatically detects if any photo were taken on that day by this user and if are pictures available, the user can specified an email address in which our automatic system generates a shared drive folder and sends an email with the folder's link to the email address specified automatically with all the pictures that are chosen. In the next Figures you can see the different sections that are mentioned in the Camera view section.

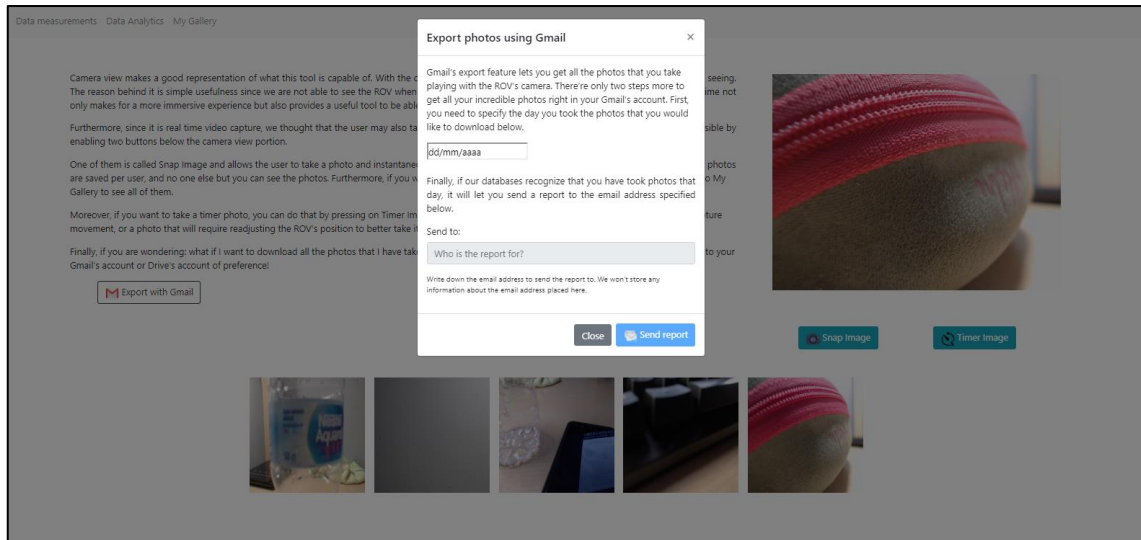


Figure 58. Gmail's export feature.

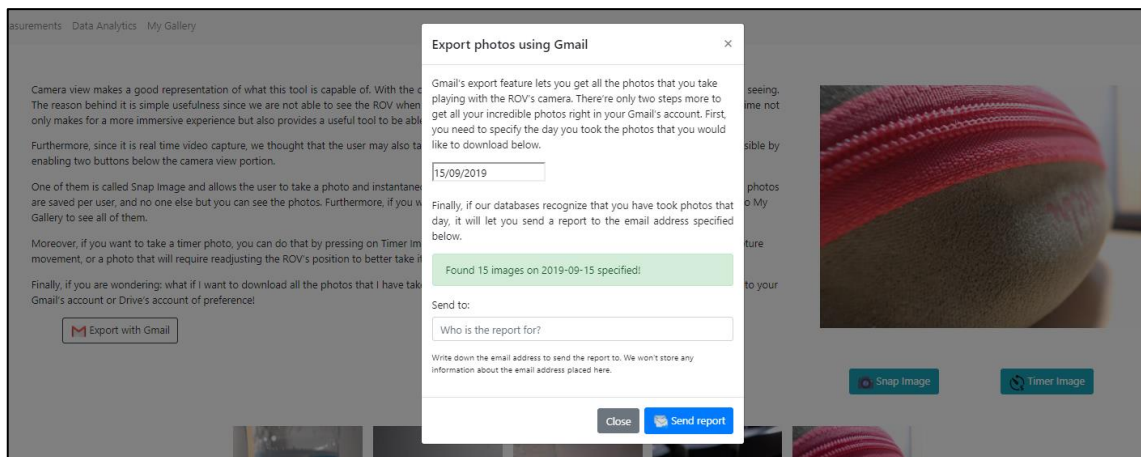


Figure 59. Message if the database recognizes there are pictures available to send a report.

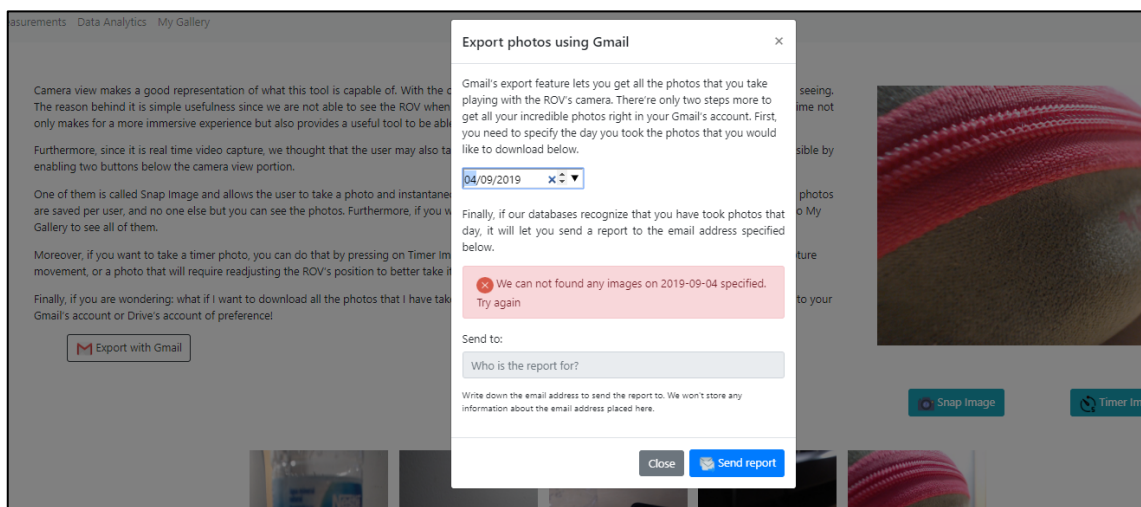


Figure 60. Message if the database does not found images on the date specified.

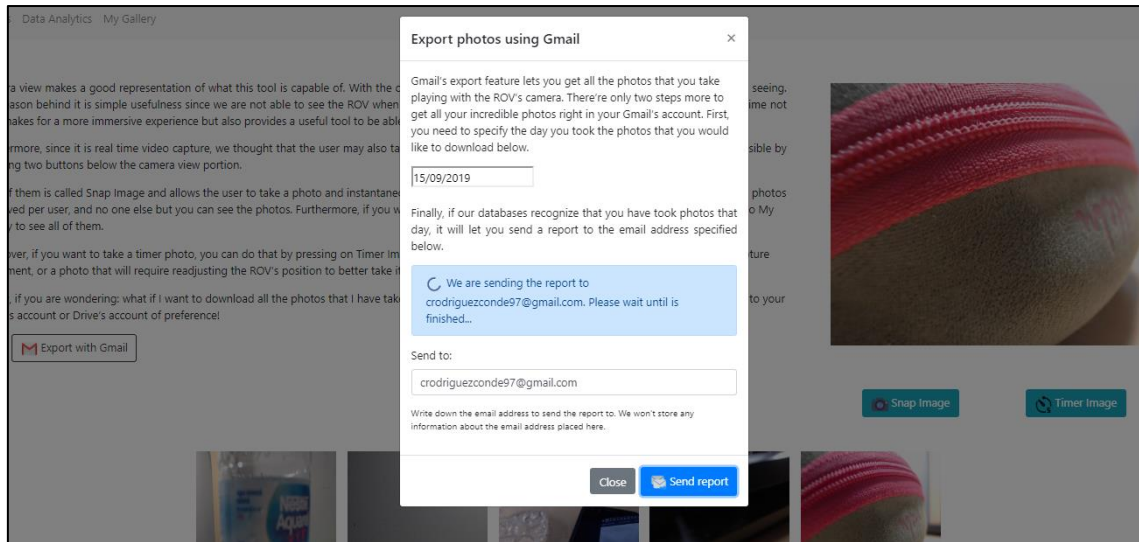


Figure 61. Sending report's message screen.

Finally, in Figure 62 it can be seen the final email that is being send with the folder attached to the user for downloading the different images.

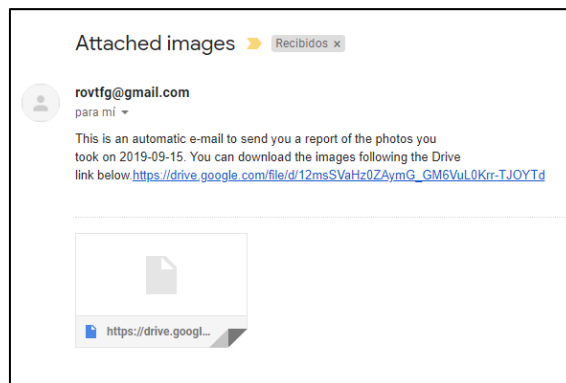


Figure 62. Final email sends with the images in a drive folder.

My Gallery

Finally, the gallery section allows the user to see all the picture taken while using the ROV. The gallery is linked per user, in other words, the picture is linked to each individual and cannot be transferred between users. This also gives the website more security and protection, since different people can use the ROV in different investigation thus causing problems if the images were not linked since each user will have to find his pictures inside the others.

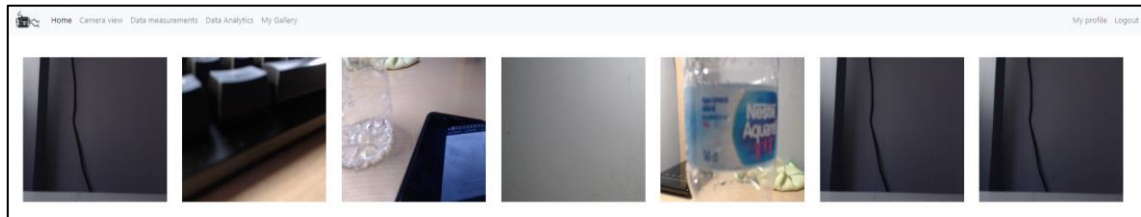


Figure 63. Collection of pictures made by one user.

In case of not having already take a photo the system will advert you to use the Camera view and make some pictures of your investigation.

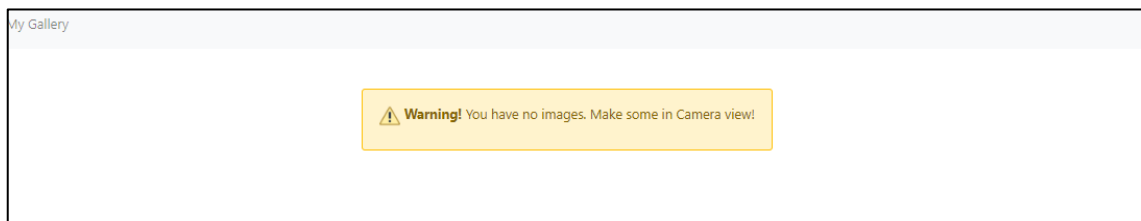


Figure 64. Alert message in case of not taking any picture using the ROV.

Chapter 8. Manufacturing and assembly

Once all components and structural parts have been described it is now the turn for the construction and assembly process. This chapter is the one in which has been spent the more time. It is due to experimentation process and learning curve experimented by doing this project. In this chapter, it passes from all the theory from the degree acknowledgements, through the computer aided design to the final execution process: the construction.

Generally, it will be described every process and obtaining criteria for the different parts and components. This does not mean that every single detail is going to be explained step by step. It is focused on those steps we had experimented the most difficult or those that are interesting to put on comments in a few lines. Below the section title is listed the different appendices containing these parts and the name of parts used in that step, whether being a *Solidworks file part* (.sldprt) or a *Solidworks assembly file* (.sldasm).

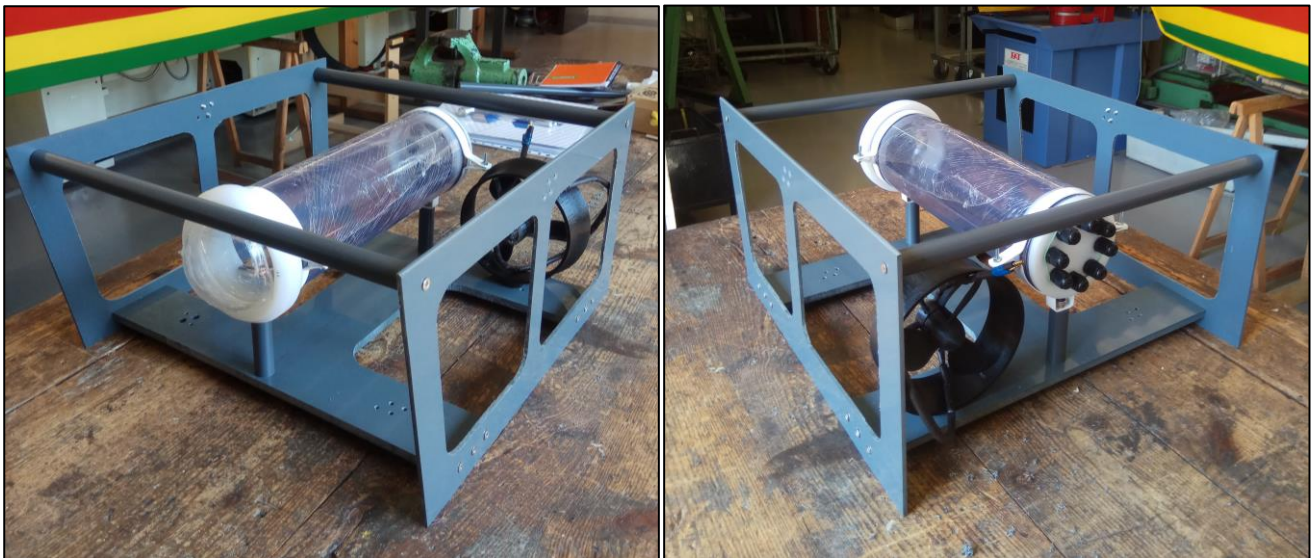


Figure 65. ROV being assembled.

8.1 Main body made with PVC.

[Appendixes: AP2.1, AP2.2 & AP2.3 – File names: *Main base.sldprt*; *Laterals.sldprt*]

It all started with two PVC plates. One of 4mm thick and the other one of 10mm thick. Originally these plates were 500mm x 500mm. Utopically, this should have been mechanized by CNC milling machines in order to achieve the desired shapes, but resources at hand could not afford this process. These plates were marked properly in order to practice some holes through it and cutting the leftover sides.

On the picture below, it can be appreciated that it is not a square shape anymore. The plates were cut with the help of a woodworking machine, the jigsaw, also valid for cutting PVC or other soft materials.

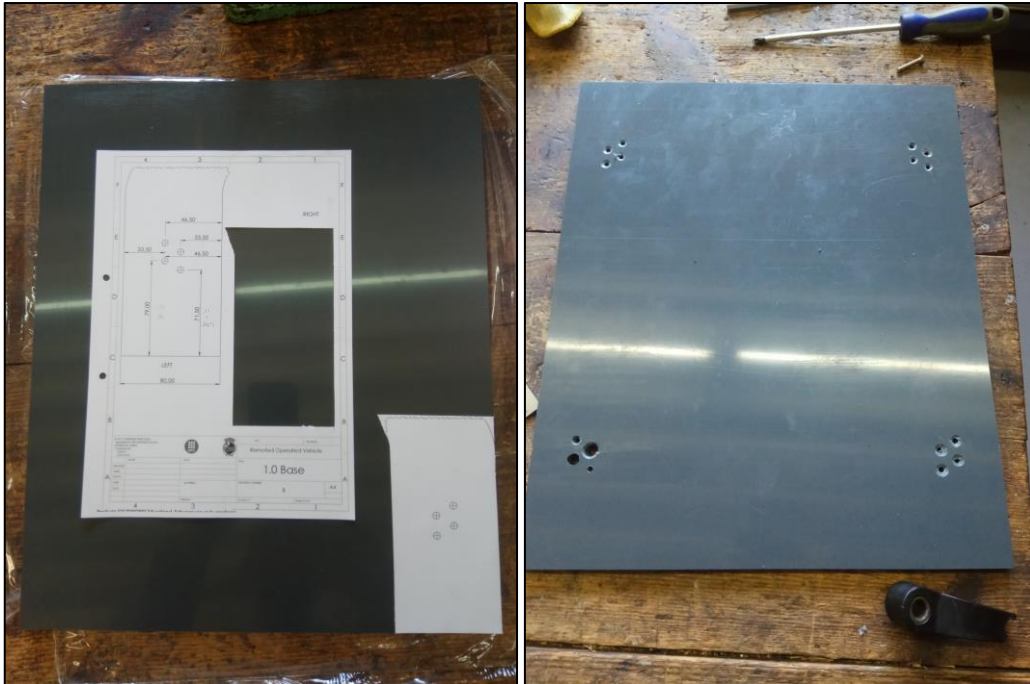


Figure 66. 5mm thick PVC plate. On the left with leftover sides cut. On the right with propeller holes.

Figure 63 shows the marking process with the help of a printed drawing. In addition to the main drawing with different sizes and annotations, a cut of the base drawing is printed at 1: 1 scale. This is done in order to mark easily once in the workshop. This way, the person in charge can easily mark the shapes and holes. The holes on Figure 67 are used to attach the propeller nozzle to the ROVs body as we will see on next steps.

If we compare the drawings of the propeller nozzle and the holes practiced on the main base made of PVC it will be more understandable. This equilateral configuration allows the user to change the orientation of the propeller manually. This is settled for future iterations once the ROV is on the water in order to optimize the propulsion of itself.

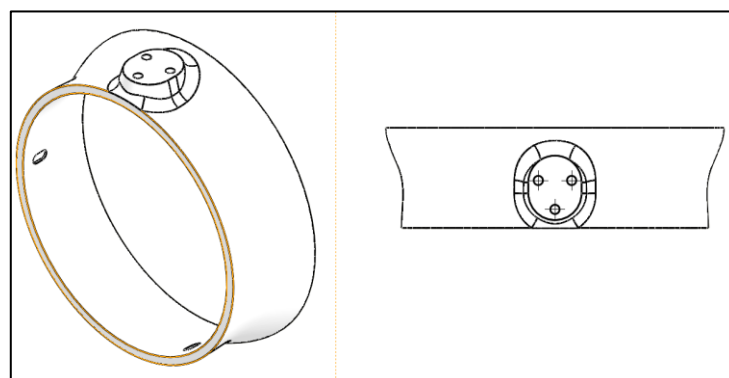


Figure 67. Nozzle view of the equilateral hole configuration.

After having the main shapes of the two lateral plates and the main base, six holes to each “leg” of the base were made. These holes are the ones who let the lateral plates being attached to the base. It can be seen below.



Figure 68. Threading the profile of the 10mm thickness plate.

These threaded holes allow us now to assemble the main base to it. This can bring us a first idea of the dimensions of the ROV as it will be. Despite it all was good, a little issue with facing the holes appeared. Due to some drawing problem into the CAD software made us commit a mistake. The holes that were going to attach both parts were not aligned. This is a very small problem that can be solved using some workshop skills. It was rectified and the structure was able to be mounted as it shows the following image. Both parts are joined by M5 screws cut at 15mm long. 12 screws in total are needed to assembly the three parts together.



Figure 69. Assembling the base with sides.

After checking all lateral holes were aligned with the ones on Figure 69, we proceed to cut the interior pieces of the lateral ones. It will be more understood with an image.



Figure 70. Inside lateral cut with shape.

The laterals now have the predesigned shape on the computer. It is motivated due to the loose of weight. Despite the weight is not high, the other reason for doing this is the resistance it offers to moving to left and right on a possible sway movement. Shown in Figure 70. The existent holes on the upper left and right parts of the laterals are made for future colocation of transversal bars. These bars will provide the structure a more solid rigidity.

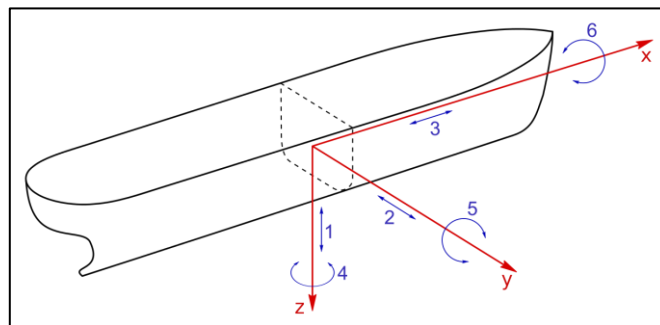


Figure 71. Sway movement along Y axis. Source: Wikipedia.

Once both lateral and base parts are ready to join, it is time now for prepare all the needs screws.

8.2 Tube caps

[Appendix AP2.4 & AP2.8 - File names: *Tube.sldprt*; *Electronic layout.sldasm*]

As it has been presented on **Chapter 6. Devices and materials**, the ROV is supposed to be controlled by some electronic devices such as an Arduino and a Raspberry Pi. These devices need an enclosure in order to be fitted on board. Due to that this enclosure needs to be sealed in order to achieve a good watertight. All devices, other sensors or the battery are going to be placed on a PVC plate board. This board will be fitted inside a transparent PVC tube. The reason that brings it to be transparent is to be able to regard the inside of the enclosure. This will provide an easy way of detecting issues or incompatibilities with the interior devices and wires. It also provides the ROV of such a nice appearance.

In this section it will be described the procedure used in order to mechanize the caps.

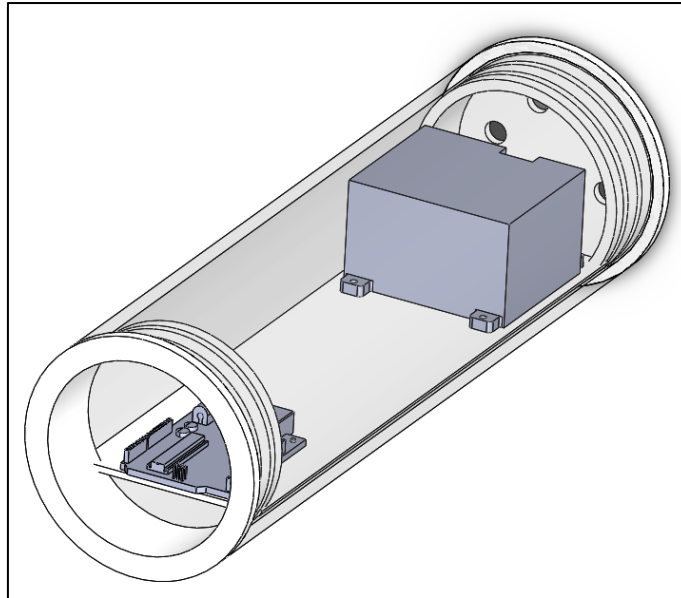


Figure 72. Main tube with some electronic devices.

The tube is basically a cylinder of 350mm length and an exterior diameter of 110mm. The inner diameter it is about 105mm. This value is the first one that provides the main dimension of the caps. Then the cap needs to be 105mm as much.

Let's focus on the caps. Both are made with extruded white nylon. On the one hand, the front cap has the particularity of being able to contain a transparent dome. This dome will be the eye of the ROV. The dome, with the camera will provide the operator the capability of regarding the boundaries of the ROV. Specifically, what lies ahead. Both plugs are made from a unique and solid cylinder of nylon. This cylinder has been machined by us with the help of a lathe. It has cost seven hours to machining each plug, but considering the little experience working on a lathe, is a good balance between the time invested and the quality achieved.

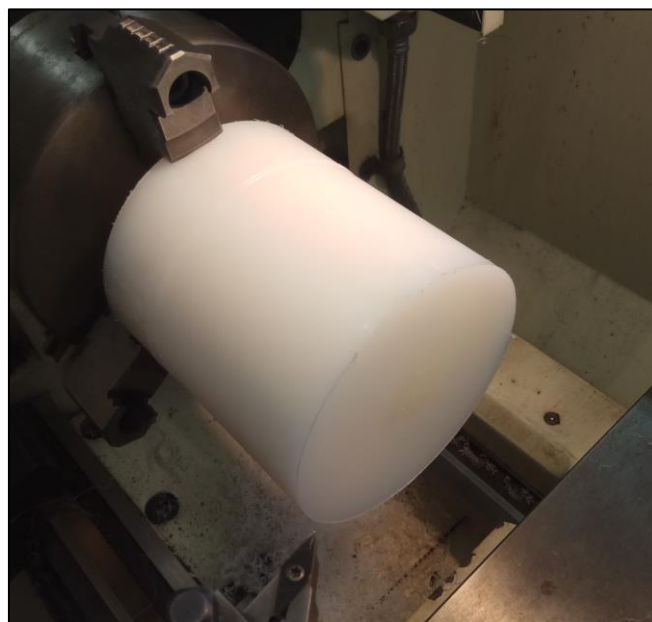


Figure 73. The original solid cylinder made of nylon. $\varnothing 120\text{mm}$.

Back plug.

[Appendix AP2.6 - File name: *Tube cap 8 hole.sldprt*]

First, and with the drawings in hand, it needs to be the desired diameter. Originally it was about 120mm. The caps are designed to be 110mm, just as the tube. So, the first operation is to cylinder until reaching this diameter along 40mm. It is said 40mm due to the final size in the drawing is 35mm. So, it is a security margin that allows you to face it after until the final measure of 35mm. This first cap is the one that will contain several holes to pass cables from the outside to the inside. That is the first due to its difficulty. It also needs to be emptied until reaching a 5mm wall (See Figure 74). In case that wall was accidentally crossed, it could serve as front plug. The picture below shows an idea of what is being done.



Figure 74. Operation of emptying of the back plug.

For what matters the adjustment, it is a matter of testing if the tube comes in tight. On Figure 75 it appears the tube fitting on the plug. Once the desired diameter is achieved and it fits, it is now the turn for the toric joints grooves.

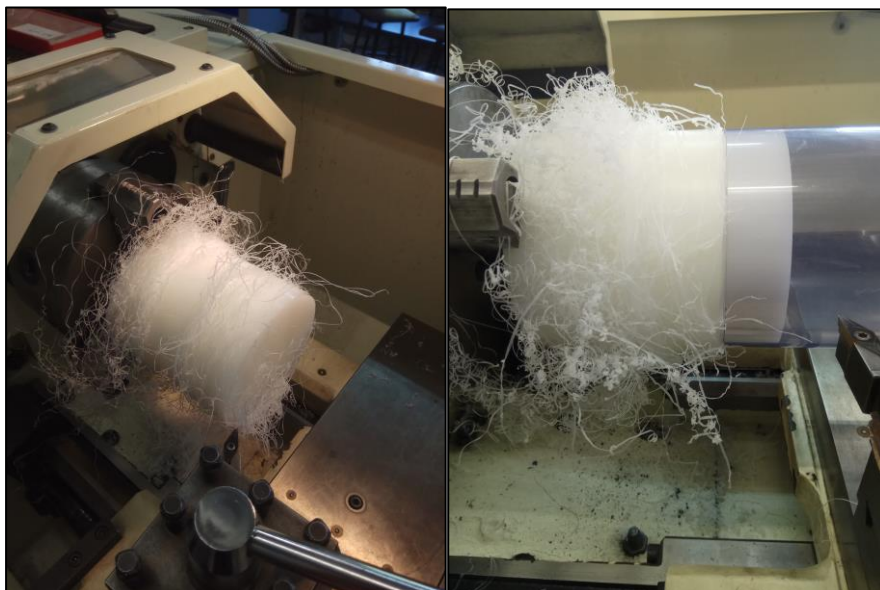


Figure 75. Fitting the tube with the cap on the lathe.

With no experience on adjusting nor fitting gaskets it has been proceeded as follows. In order to do the groove, it is only necessary knowing the width of the ring. In this case they are 4mm width and 110mm diameter. To house the rings, it is going to be machined a 4mm groove. The depth of this operation has been achieved by testing the tube with the cap until it fitted perfectly. Then, after testing and testing with the nitrile gaskets against the tube fits the millimeter with the cap.

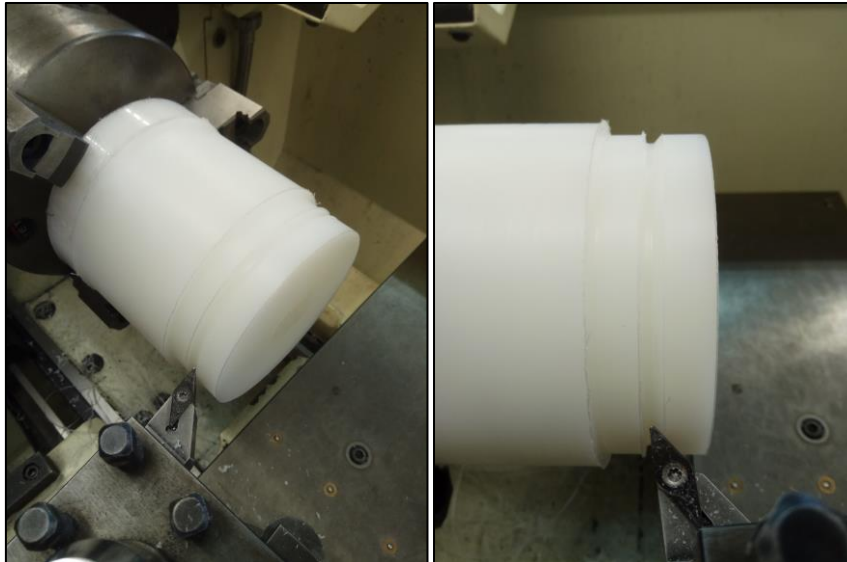


Figure 76. Back plug on the lathe.

Once arrived at this point, it is important to note that two back plugs have been done. The reason why this was done is the following. Before drilling the cap, the tightness should have been tested. Once confirmed, the eight holes could be drilled. But in reliance the holes were made without having previously tested the tube with the plugs under water. This forced us to mechanize the third plug, but this time without diminishing the interior. On the picture below, it can be appreciated the result of the final cap before drilling the holes. The redundant plug can be seen in Figure 82.



Figure 77. Back plug, before and after drilling holes.

Front cap

[Appendix: AP2.5 – File names: *Tube cap.sldprt*; *Dome ring.sldprt*]

All the process is equal to the other one. The main difference is that this cap will be emptied through the end. The picture below shows the moment when the saw is cutting the mechanized part from the other one. Also, it can be appreciated a thin layer of nylon that will be putted after on the lathe for turning around and polish it.

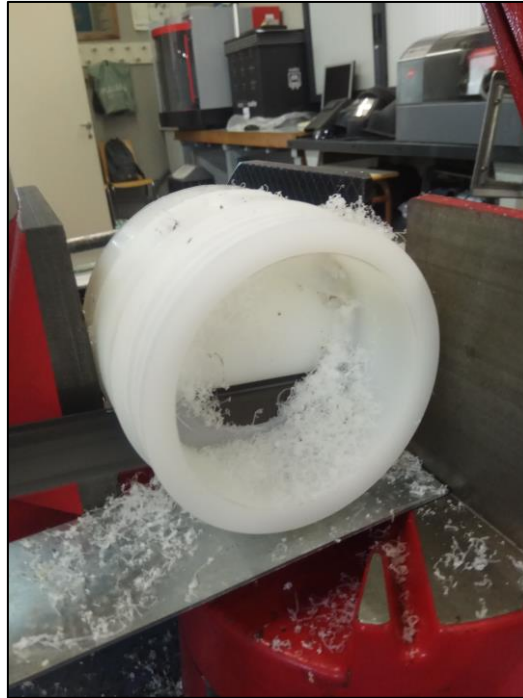


Figure 78. Front plug being cut.

This ring appearance brings the ROV of being able to put the camera and see through its eye. This part is the one that seals the transparent dome through the nylon ring to the transparent PVC tube.

In Figure 79 it is shown how it is achieved. A three-part configuration to guarantee the tightness of this part. Then on Figure 80 it is showed the real pieces obtained. Finally, on Figure 81 it appears the front plug assembled. In order to join the three parts, it has been used the Sikaflex silicone.

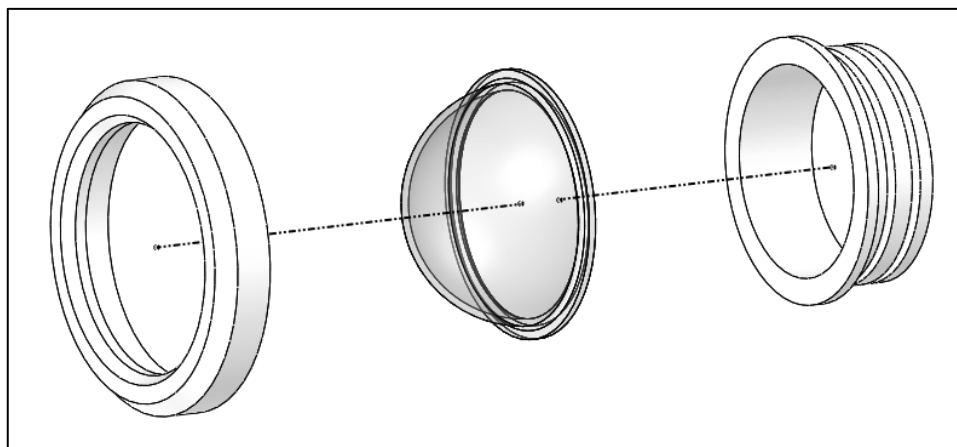


Figure 79. Exploded view of the front plug composition.



Figure 80. From left to right: ring, dome and front plug.



Figure 81. Front plug assembled.

Result

Below, on Figure 82, it appears the ended tube with its caps. Also, it can be appreciated a few black lines. This are the sealing toric nitrile joints. Two on each cap. They guarantee the tightness of this assembly. It is also remarkable on this picture that the back plug is not the final one. It is because this is the first tightness test. For this test we wanted a full closed tube. In order to achieve this, a third plug was done. On its final version, the back plug is as it appears on the right of Figure 77.



Figure 82. Result of the tube composition.

Joins and screws

Glue for plastics are used for joining the 3D printed parts. Nozzle of the propellers to the support of the BLDC motors.



Figure 83. Glue used.

Screws and nuts are also needed in order to assembly easily all the components and the different devices. They all appear on the table below:

Type	Metric	Length (mm)	Quantity	Part or function
Screw	M4	10	16	BLDC
	M5	25	28	Assembly
Threaded rod	M8	100	2	Tube stand
Nut	M3	30	4	Clamp
	M8	-	2	Tube stand

Table 10. Assembly elements.

Chapter 9. Difficulties and deviations

The purpose of this chapter is to provide a useful and informative section of the problems encountered. Problems that may appear to anyone who wants to reproduce this prototype. Therefore, it is a selection of different incidents that have appeared in different stages throughout the project. These, obviously, have become difficulties added to the project to a lesser or greater extent.

ESC

Issues and incidents

While configuring the ESC, they presented many problems and troubles that were coming up from everywhere. These problems were attached to its internal configuration and a lack of communication with the computer.

On a first assembly of the ESC and BLDC it all worked together, and we got a propeller turning at very huge number of revolutions, and controlling its speed using Arduino and a 10k potentiometer. But it was the first and the “last” day it worked until a solution was found. Then we were having 6 BLDC, and 6 ESC that does not know how to deal with.

Solving troubles

On a first instance, on every forum or video that we used to do it, it seemed very easy and all they worked. But the reality was that our ESC with the BL Heli suite (both BL HeliSuite16 and BL HeliSuite32, the two main versions of this software) could not communicate to each other. So, we have two problems here; on one hand we had no idea in how to deal with it and by the other hand we needed to pass through it having all the speed controllers working. In order to do that, we realized that there was three main ways of solving it. The easy one, the one which appeared in almost every video or link we encountered, was using an Arduino device. It was supposed to be able to get inside the ESC by means of a few and simple connections. The second one was getting a specific USB device which allows your ESC to communicate with your computer. This, after some time waiting for it to arrive, did not work either. Lastly, we had the hardest way to connect the ESC with the computer. This way was proven while we waited for the USB device to arrive. This option, despite not being the easiest nor the most accessible way, ended up being the most direct and useful way. This third option, the one used in here consists as follows.

In order to get to communicate with the computer it was needed to weld three wires to three specific contacts inside the ESC (see Figure 84, on the left). The other extreme of each wire has to be joined on the coloured pins on the right of Figure 84.

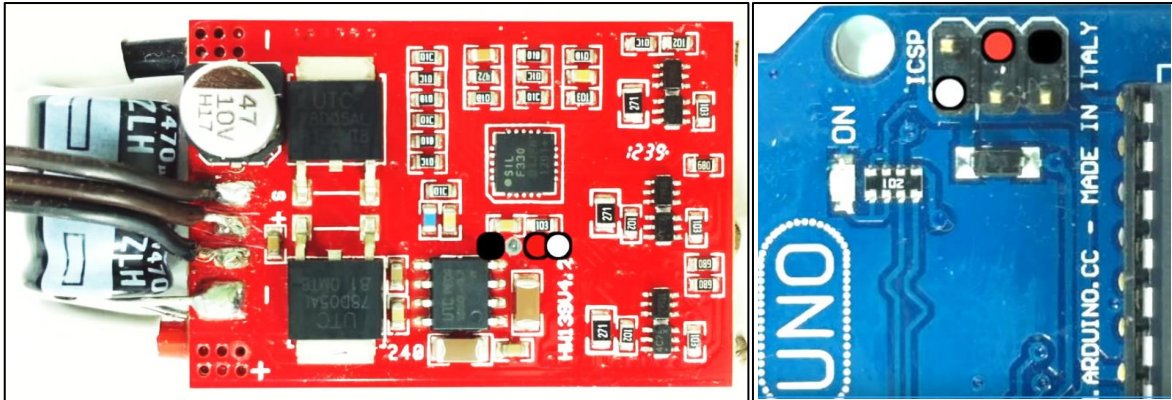


Figure 84. On the left, Arduino UNO. On the right, inside an ESC.

Once it was done, it was tested and it all worked. For the first time we were able to get to the bowels of the Electronic Speed Controller. Then we were able to change a huge number of parameters affecting the behaviour of the brushless DC motor.

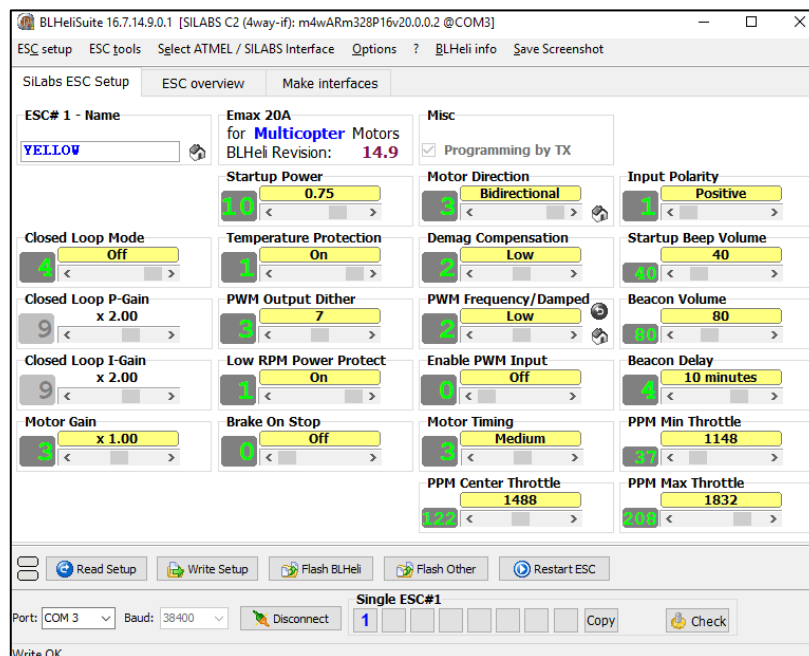


Figure 85. BLHeli Suite v16 user interface.

Once the USB device arrived it was not as it was expected. By connecting it and following the given instructions it did not work either. So, at this point, we needed to access inside every ESC to weld the same three wires on each one. This part would not have been possible without the help of a professional and a friend of ours. A handyman specialized in electronics who gave us all his support during the whole process of the ROV assembly. Specially with ESCs, batteries and Arduino encountered problems. Once we had all the wires welded, we could change and modify the different parameters on each ESC. On a first time we put the same values and configuration in all ESC that will be mounted on next steps.

Changing the backend programming language

During the first iteration of the different parts of the programming period, the website or graphical interface (GUI) was coded using PHP and Symfony as a backend programming language. This backend as mentioned before is in charge of making the different communication with the database and sending the information successfully to the main page of the website. However, during this building part of the website, the backend infrastructure was becoming very inefficient treating the different data that was coming from the ROV and ultimately, we decided to stop and redo it all again. This was a big problem because of PHP was not capable of handling very well the data that was coming from the database, and it was becoming unscalable and un-expandable due to the poor connection between Python on the Raspberry Pi and the web server using PHP. Ultimately it was decided to change the backend language for Python, because of the flexibility and scalability it provides due to the easier syntax and the better modules or plugins to work with database and data in general. This resulted in a huge waste of time because the entire backend has to be redone it from scratch and part of the front website too. This was a difficult situation due to the constraints in time and the little knowledge about Python and Django at the time, but we were able to pull through and redo the backend from scratch. PHP not only was not efficient at treating data but also was very slow to make the transactions making the overall feeling and paste of the web not fun and enjoyable to use though long periods of time. Finally, however, we were able to migrate from one programming language to another giving us some scalability, more rapidness in transactions and an overall feeling of smoothness, making it easier to make big changes in the future and expand the project even further.

Printing with 3D technology

Another of the mishaps we have faced was the impression of some parts of the project. As an example, we have the support for the propeller. With this part, which can be seen in Figure 85, there had been no problem until then. What seems, if we look at Figure 87, is that at a certain layer height, the machine has moved sideways. This indicates that the resulting piece seems to be displaced from its origin. Obviously, it is not a valid piece. In addition, it is a structural part for the whole helix and must be resistant.



Figure 86. Propeller support after being printed.

In 3D printing jargon, it is understood "layer height" the units the Z-axis has moved upwards or downwards. This layer height is defined before sending the file to the machine, and depending on the machine parameters it can be one or another. For deeper information about 3D printing knowledges there is a book on the Bibliography [65].



Figure 87. Defective propeller support.

After thinking about what could have happened, we believed that it could only be a machine error. It is true that it was printed with the other printer extruder, but it was supposed to have worked well. Assuming we had the files and the techniques well, we decided to stop printing with that machine. Then we went to another machine. The result of the printed support on this other printer can be seen below.

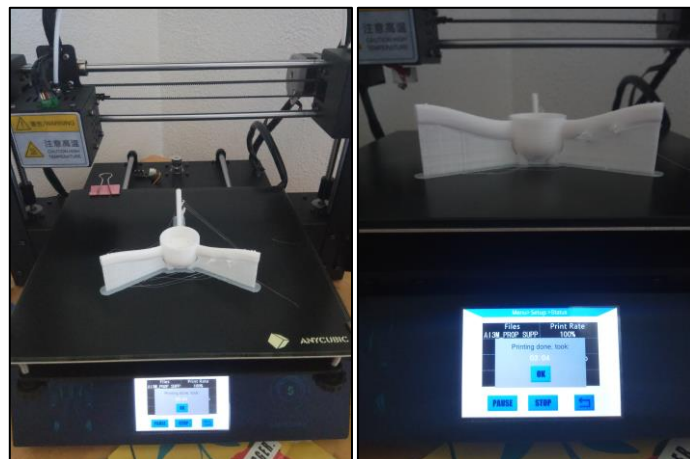


Figure 88. Anycubic i3 printer after finish the print.

Raspberry Pi case. This time in order to not damage the raspberry it was decided to print a suitable case. This case, at its first test resulted as it can be seen on Figure 89.



Figure 89. Raspberry case.

As mentioned in previous chapters, a solid lid has been obtained. This lid must test the tightness of the tube. Fortunately, and for being the first water test, the plug turned out to be perfect and prevented the leak of water.



Figure 90. Test plug to check tightness.

In order to determine the required ballast in an empirical way, this tube was mounted to the ROV to see the necessary weight on board to keep the submersible sunken. The plug was still watertight. On the other hand, the fact of just having put a weight estimate inside, only allowed estimating the value of this ballast that is about 3.5 kg. This is a task that is pending for future iterations since the necessary elements were not available for doing so. These elements would be leads of different cards. The pieces we had were one kilogram each.



Figure 91. Tightness test with paper inside to verify if water goes in.

Finally, the last water test was with the actual end cap. This, though, was not able to stop water from getting inside. This is not due to the manufactory of it but for the cable leakers that were not tight to each wire. On Figure 92 is possible to see the configuration adopted in order to simulate the different wires (corresponding to temperature sensors, lights, propellers or data transfer) through the cap. And as it was thought they were not watertight.

On a situation in which having enough time it would be possible to try sealing it with the help of some resin or silicone to achieve a complete tightness.



Figure 92. Disposal of the real plug simulating cables.

Also there has been issues with the Raspberry Pi module. It is about a power lack. On the Raspberry Linux's interface, it appeared a lightning bolt on the upper right side. After finding out what it was, we thought about possible solutions. The solution adopted below is based on the implementation of a small transformer. This one has the capacity to supply the amperage and the necessary tension. It is a good solution since it occupies little space and is economic. With this device, Raspberry and other devices were able to run normally (keyboard, PS3 and Arduino controller).

The picture below shows the amperage value while climbing from 2 to 3 amperes. On Figure 94 appears the result once the step-down transformer is mounted at the back of the Raspberry Pi

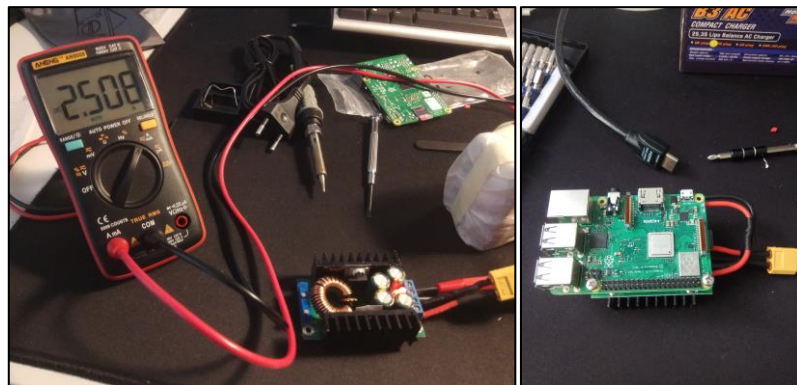


Figure 93. Setup of the power solution for the Raspberry.

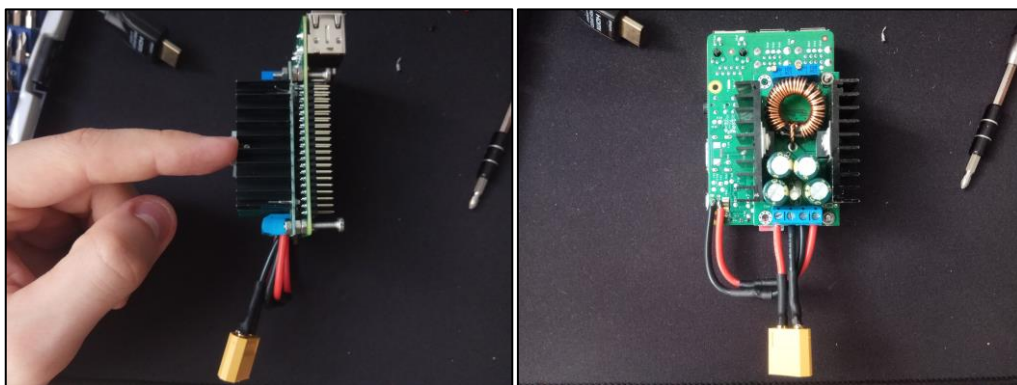


Figure 94. Step down appearance once mounted on the Raspberry.

Another last-minute modification was to change the cables that configured untainted motors for a triaxial cable, recycled from power sources. We noticed that this cable had three drivers housed inside a hose. This would allow us to move more easily, and supposedly, by better guaranteeing the tightness of the tube. Down below it is possible to see the two configurations. On the one hand and at the left it is the appearance as it was at its first version. There were thinner cables, also recycled that were braided manually. On the other side, at the right picture it appears the final configuration. It consists of a triaxial cable with the bananas at the ends of the cable.



Figure 95. BLDC wires. Left: first version. Right: triphasic recycled cable.

Chapter 10. Future work

This chapter presents a certain variety of ideas that can be pursued or implemented in future work. Its main objective is to encourage the reader to start working with any of the ideas presented or those that may occur. It is important to remember, once again, that from the beginning, this work is based on the philosophy of open access.

Adding a ballast system to achieve the desired floatability could be the first. This new feature will also provide the ROV with a variable range of useful load. Being this way, it can carry a variety of different gadgets or tools.

Usually, when seeing a portable system for field operations it is also fitted inside a suitcase. Taking as an example the one on Figure 96, a suitcase could also be developed for the ROV transport. Either as an additional delivery, or as a project.



Figure 96. Ground control station. Source: Wikipedia

There are also several kinds of work to be done in the electronics part. First, the GUI can be improved furthermore by using more environmental data. This translates to using more sensors to be able to gather more information about the environment. Furthermore, the data can be cleaned more using data-science techniques. The data is already cleaned using automatic procedures to detect out of context data or unfilled data. However, this technique becomes more inefficient when the data amount becomes larger. With data science cleaning techniques, we can improve the rate of filtering much further, consequently improving the efficiency and the possibility to stored large amount of data in a more organized way.

Concepts of machine learning and basic AI can also be implemented in the future, for example, to make the camera able to detect certain objects. Consequently, this will enable the ROV to make predictions and to be able to search underwater for a certain species or even a simple object.

Neural networks can also be useful in order to make the ROV 100% autonomous. In other words, the ROV can be programed to follow a simple pattern and through experience and repetition be able to make guesses or even follow a route predicted by himself, depending on historical data that we already have because of the different sensors present in the ROV. This will be translated into better decisions on route planning because of the ROV will be basing his hypothesis in real data, historical data that can be

expressed and formulated to make accurate predictions in temperature, water flow or even presence of certain species, following one or another route.

Lastly, achieve an Autonomous Underwater Vehicle. Investing a necessary amount of money, trying to provide this ROV within an AUV. In order to being able to explore deeper among others. One of the most ambitious things is to be able to implement the lack of cables connecting the ROV with the outside. This not only reduces costs but also reduces the failure rate because it has no cables that can be broken or lost apart during expedition. The lack of cables will imply that the ROV will be using waves to communicate the different measures gathered by the sensors to the outside. Furthermore, the controller for the different motors would have to be wireless. This last point can be achieved already in this version of the ROV, but we are then unable to follow along into more depth waters. The version of the wireless controller works on very shallow water being unable to operate under certain depths making it not useful in the current version.

After having drawn some ideas for you it is now time for you to develop and investigate new features to implement on the vehicle.

Chapter 11. Conclusions and recommendations

As in all the projects or tasks carried out, it allows us to draw some considerations about what has been happening. These considerations always contribute positively. If you look at all the work in perspective, you can know what and why of the things that have been done. In addition, the "how" has been done has its consequences. That's why having made this project you realize your capabilities and limitations. In this chapter, some reflections will also be presented to capture these extracted conclusions.

At first you are in front of a page with nothing but a simple and basic draft. It's a very special moment. You think everything is possible and achievable. This is partly true, but on the other hand, you still do not realize the embellishment of the project. Then, you're moving forward and doing schematics on how you want it to be. Theoretically, it is still a viable and achievable project. And it is. But when you begin to search for specific materials or devices you need to build it, you begin to realize the actual size of the project.

The learning curve you experience is almost the greatest in your experience and during university education. It is because of these difficulties you find when you start thinking about other methods or solutions. All this considering all the recognitions acquired throughout the different courses. Even courses done outside the university. Therefore, you make mistakes, but it would not be possible without these errors achieving a solid final product. And that does not mean that in this work, the most powerful, useful and well-designed ROV has been presented. This just means the opposite. With the experience gained, we are one hundred percent safe, it will be a second version of ROV work a hundred times better than this. Thanks to all this trial and error.

Leaving aside the personal experiences we had, let's show from little to huge things that would be improved easily or mandatory. As we wrote, the try for a first version is only a first approach of our design towards the universe of ROVs.

The most relevant criteria are, of course, resources and time factors. The first one for the expenses that avoid this type of projects. It is not a theoretical project that remains printed on paper but must be built and constructed. Over time, it is understood as an "endless" story. It has been developed during the months compressed between March and September, seven months. It means that in this time frame a ROV had to be built from scratch to be a useful device.

A good recommendation is to start your project with what you must start with. It is difficult to anticipate situations when it is not experienced. The problems will always come out somehow sooner or later. Fortunately, the solutions will also leave the head. In this project, it has been about a month and a half drawing and writing what the product will be. This is not a good practice. In the following steps it has been shown that the first five versions of the product will not be the last. Therefore, as the prototype is being built, the solutions will emerge. Taking as an example parts such as helix, when a mishap has been found, it has been addressed directly to find a solution.

Another advice we can provide is not thinking that you should have bought everything. It will be easy to understand with an example. We went one day, before starting to build the ROV and bought some electronics. Things like a Raspberry Pi, its camera, some sensors of humidity and temperature, etc. Later, after two months, we finished assembling the ROV and we had not even tried all the devices that we bought on that day a couple of months before. So, do not think you need all the stuff to continue. Almost nothing is indispensable. Go step by step, number one before going to number two. Linked to the above issues, you may only have this knowledge once you live and experience yourself.

One of the biggest complications has been to be able to program a simple but efficient way in order to see the data in real time. Learn different programming languages at a short period of time has also been very difficult and the learning curve has been very steep. Anyone who wants to continue this project has to have very clear concepts of web design and implementation and also database control by using one query language. Using so many different programming languages and changing our GUI infrastructure has been tough but achievable. Firstly, we begun building our GUI using a programming language called PHP, very popular for building websites but also very inefficient regarding the data management and the different connections with Google's API. On the other hand, Python has brought us flexibility and scalability to re-do the entire backend of our website and make it more friendly to new users or other people that might work on the project in the future. This change in the backend caused us to take all our progress so far and the need to start programming the web again through Python and Django. This change was an incredible loss of time, but also, we learned that we can not only rely on the experience we had but also consider other options that could scare us at the beginning. All of this ended with good results in terms of improvements in efficiency, scalability and profitability of expansion. Some of the decisions made during the whole programming period were bad, but we learned to be more careful when designing such infrastructure and learned how to choose wisely from the multiple choices that the different programming languages gave us.

Finally, after months of really hard work we were able to archive most of the objectives defined at the beginning, being self-taught for the most part of the project, achieving not only the ROV's movement, but also the construction and assembly of the ROV's main body. Also, the different sensors and connections that made possible to the ROV be able to send successfully this data through a database and finally to be seen in a graphical interface (website).

In the end it has been a hugely satisfaction to us being able to achieve most of the objectives raised. It has been almost built in its totality what demonstrates it is not finished at all. But what is most important, every step has a good foundation. A starting point for any who wants to take it from this point and continue developing it. Not only for completing this paper but going beyond, challenging itself. Furthermore, it can be changed or adapted in every single step in order to achieve a different aesthetical aspect or some different features.

Bibliography

- [1] 3.7.5rc1 Documentation. (n.d.). Retrieved October 3, 2019, from <https://docs.python.org/3/>
- [2] 36 Types of Screws and Screw Heads (Ultimate Chart & Guide). (n.d.). Retrieved May 17, 2019, from <https://www.homestratosphere.com/types-of-screws/>
- [3] Akmal, M., Yusoff, M., & Arshad, M. R. (2012). Active Fault Tolerant Control of a Remotely Operated Vehicle Propulsion System. *Procedia Engineering*, 41, 622–628. <https://doi.org/10.1016/j.proeng.2012.07.221>
- [4] Antonelli, G., Fossen, T. I., & Yoerger, D. R. (2008). Underwater Robotics. In *Springer Handbook of Robotics* (pp. 987–1008). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-30301-5_44
- [5] API Reference | Google Drive API | Google Developers. (n.d.). Retrieved October 3, 2019, from <https://developers.google.com/drive/api/v3/reference/>
- [6] Archivo:ROV Hercules 2005.JPG - Wikipedia, la enciclopedia libre. (n.d.). Retrieved September 13, 2019, from https://es.m.wikipedia.org/wiki/Archivo:ROV_Hercules_2005.JPG
- [7] BIBLIOGRAFIA - Documentos de Google. (n.d.). Retrieved October 3, 2019, from <https://docs.google.com/document/d/1jKYt1hFKinKIEWOtXtnzd8lt1kLqKGHARProhqffUvs/edit>
- [8] Bitify: 3D OpenGL visualisation of the data from an MPU-6050 connected to a Raspberry Pi. (n.d.). Retrieved October 3, 2019, from <http://blog.bitify.co.uk/2013/11/3d-opengl-visualisation-of-data-from.html?m=1>
- [9] Build a Raspberry Pi Moisture Sensor to Monitor Your Plants. (n.d.). Retrieved October 3, 2019, from <https://computers.tutsplus.com/tutorials/build-a-raspberry-pi-moisture-sensor-to-monitor-your-plants--mac-52875>
- [10] Cardozo de Souza Ribeiro, P. O., Machado dos Santos, M., Lilles Jorge Drews, P., & Silva da Costa Botelho, S. (2017). Forward Looking Sonar Scene Matching Using Deep Learning. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 574–579). IEEE. <https://doi.org/10.1109/ICMLA.2017.00-99>
- [11] Christ, R. D., & Wernli, R. L. (2007). *The ROV manual: a user guide to observation-class remotely operated vehicles*. Butterworth-Heinemann.
- [12] Connect to a MySQL database remotely. (n.d.). Retrieved October 3, 2019, from <https://support.rackspace.com/how-to/mysql-connect-to-your-database-remotely/>
- [13] django-admin and manage.py | Django documentation | Django. (n.d.). Retrieved October 3, 2019, from <https://docs.djangoproject.com/en/2.2/ref/django-admin/#django-admin-runserver>
- [14] DS18B20 Temperature Sensor With Python (Raspberry Pi). (n.d.). Retrieved October 3, 2019, from <https://bigl.es/ds18b20-temperature-sensor-with-python-raspberry-pi/>
- [15] Dürkefalden, A., Hoernle, K., Hauff, F., Wartho, J.-A., van den Bogaard, P., & Werner, R. (2019). Age and geochemistry of the Beata Ridge: Primary formation during the main phase (~89 Ma) of the Caribbean Large Igneous Province. *Lithos*, 328–329, 69–87. <https://doi.org/10.1016/j.lithos.2018.12.021>
- [16] Embedding usb webcam livestream into html - Raspberry Pi Forums. (n.d.). Retrieved October 3, 2019, from <https://www.raspberrypi.org/forums/viewtopic.php?t=210338>

- [17] Erena, M., Atenza, J., García-Galiano, S., Domínguez, J., & Bernabé, J. (2019). Use of Drones for the Topo-Bathymetric Monitoring of the Reservoirs of the Segura River Basin. *Water*, 11(3), 445. <https://doi.org/10.3390/w11030445>
- [18] Fowler, A. P. G., Tan, C., Cino, C., Scheuermann, P., Volk, M. W. R., Shanks, W. C. P., & Seyfried, W. E. (2019). Vapor-driven sublacustrine vents in Yellowstone Lake, Wyoming, USA. *Geology*, 47(3), 223–226. <https://doi.org/10.1130/G45577.1>
- [19] Ghilezan, A., & Hnatiuc, M. (2017). The ROV communication and control. In *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)* (pp. 336–339). IEEE. <https://doi.org/10.1109/SIITME.2017.8259920>
- [20] GitHub - ageron/hands-on-ml: A series of Jupyter notebooks that walk you through the fundamentals of Machine Learning and Deep Learning in python using Scikit-Learn and TensorFlow. (n.d.). Retrieved October 3, 2019, from <https://github.com/ageron/hands-on-ml>
- [21] GitHub - wrobell/librpims5x: MS5x family pressure sensors library. (n.d.). Retrieved October 3, 2019, from <https://github.com/wrobell/librpims5x>
- [22] Gland-PG7-IP68 | 3D CAD Model Library | GrabCAD. (n.d.). Retrieved July 16, 2019, from <https://grabcad.com/library/gland-pg7-ip68-1>
- [23] google-drive-proxy/MimeType.cs at master · google/google-drive-proxy · GitHub. (n.d.). Retrieved October 3, 2019, from <https://github.com/google/google-drive-proxy/blob/master/DriveProxy/API/MimeType.cs>
- [24] gpio - Live video streaming embedded in a web page - Raspberry Pi Stack Exchange. (n.d.). Retrieved October 3, 2019, from <https://raspberrypi.stackexchange.com/questions/14740/live-video-streaming-embedded-in-a-web-page>
- [25] Harris, H. E., Patterson, W. F., Ahrens, R. N. M., & Allen, M. S. (2019). Detection and removal efficiency of invasive lionfish in the northern Gulf of Mexico. *Fisheries Research*, 213, 22–32. <https://doi.org/10.1016/j.fishres.2019.01.002>
- [26] How do I use google.auth instead of oauth2client in Python to get access to my Google Calendar - Stack Overflow. (n.d.). Retrieved October 3, 2019, from <https://stackoverflow.com/questions/52085054/how-do-i-use-google-auth-instead-of-oauth2client-in-python-to-get-access-to-my-g/52101480>
- [27] how stream video into html5 - Raspberry Pi Forums. (n.d.). Retrieved October 3, 2019, from <https://www.raspberrypi.org/forums/viewtopic.php?t=45681>
- [28] How to - Flashing BLHeli firmware using Arduino Uno - eluminerRC - YouTube. (n.d.). Retrieved October 3, 2019, from <https://www.youtube.com/watch?v=P2agnNY5D-8>
- [29] How to allow remote connection to mysql - Stack Overflow. (n.d.). Retrieved October 3, 2019, from <https://stackoverflow.com/questions/14779104/how-to-allow-remote-connection-to-mysql>
- [30] How to Make a Lithium Battery Charger - YouTube. (n.d.). Retrieved May 14, 2019, from <https://www.youtube.com/watch?v=DAQoyPZU4Z8>
- [31] How to use git-flow in SourceTree | SmartNinja šola programiranja. (n.d.). Retrieved September 30, 2019, from <https://www.smartninja.si/blog/how-to-use-git-flow-in-sourcetree-1474046746446>
- [32] How to Work With AJAX Request With Django. (n.d.). Retrieved October 3, 2019, from <https://simpleisbetterthancomplex.com/tutorial/2016/08/29/how-to-work-with-ajax-request-with-django.html>
- [33] ISO 13628-8:2002(en), Petroleum and natural gas industries — Design and operation of subsea production systems — Part 8: Remotely Operated Vehicle (ROV) interfaces on subsea production systems. (n.d.). Retrieved April 9, 2019, from <https://www.iso.org/obp/ui/#iso:std:iso:13628:-8:ed-1:v1:en>
- [34] Karen Sanamyan, N P Sanamyan, Victor V. Ivin, S. V. G. (2018). A record of deep-water benthic siphonophore (Siphonophorae: Physonectae: Rhodaliidae) in vicinity of submarine Piyp Volcano (North-Western Pacific). <https://doi.org/10.15298/invertzool.15.4.01>
- [35] Kaushal, H., & Kaddoum, G. (2016). Underwater Optical Wireless Communication. *IEEE Access*, 4, 1518–1547. <https://doi.org/10.1109/ACCESS.2016.2552538>

- [56] soil-moisture-sensor-python/app.py at master · chrishutchinson/soil-moisture-sensor-python · GitHub. (n.d.). Retrieved October 3, 2019, from <https://github.com/chrishutchinson/soil-moisture-sensor-python/blob/master/app.py>
- [57] Speaking in Phases Activity | NASA/JPL Edu. (n.d.). Retrieved August 13, 2019, from <https://www.jpl.nasa.gov/edu/teach/activity/speaking-in-phases/>
- [58] Stein M. Nornes. (2018). (PDF) *Guidance and Control of Marine Robotics for Ocean Mapping and Monitoring*. Norwegian University of Science and Technology. Retrieved from https://www.researchgate.net/publication/327201129_Guidance_and_Control_of_Marine_Robotics_for_Ocean_Mapping_and_Monitoring
- [59] Streaming Raspberry Pi Camera H264 into HTML over RTMP - Raspberry Pi Forums. (n.d.). Retrieved October 3, 2019, from <https://www.raspberrypi.org/forums/viewtopic.php?f=43&t=45368>
- [60] Subsea Lights - Wide Variety of Applications - DeepSea Power & Light. (n.d.). Retrieved March 23, 2019, from <http://www.deepsea.com/products/lights/>
- [61] Time zones | Django documentation | Django. (n.d.). Retrieved October 3, 2019, from <https://docs.djangoproject.com/en/2.2/topics/i18n/timezones/>
- [62] Trello. (n.d.). Retrieved September 30, 2019, from <https://trello.com/>
- [63] Un museu sota l'aigua - Museu d'Arqueologia de Catalunya - Seu Macportal. (n.d.). Retrieved September 13, 2019, from <http://www.mac.cat/XARXES-I-RUTES/Un-museu-sota-l-aigua>
- [64] w1thermsensor/core.py at master · timofurrer/w1thermsensor · GitHub. (n.d.). Retrieved October 3, 2019, from <https://github.com/timofurrer/w1thermsensor/blob/master/w1thermsensor/core.py>
- [65] Wallack Kloski, L., & Kloski, N. (2019). *Make: Getting Started with 3D Printing*. *Journal of Chemical Information and Modeling* (Vol. 53). <https://doi.org/10.1017/CBO9781107415324.004>
- [66] What is Pulse Width Modulation (PWM)? - Definition from Techopedia. (n.d.). Retrieved July 9, 2019, from <https://www.techopedia.com/definition/9034/pulse-width-modulation-pwm>

Annexes

A1. Tutorials

A1.1 Battery tutorial

Source [See Bibliography 37]: <https://www.youtube.com/watch?v=hwhqn4BmC2I>



Figure A 1. *GreatScott!* battery pack tutorial

A1.2 ESC welding procedure tutorial

Source [See Bibliography 28]: <https://www.youtube.com/watch?v=P2agnNY5D-8>

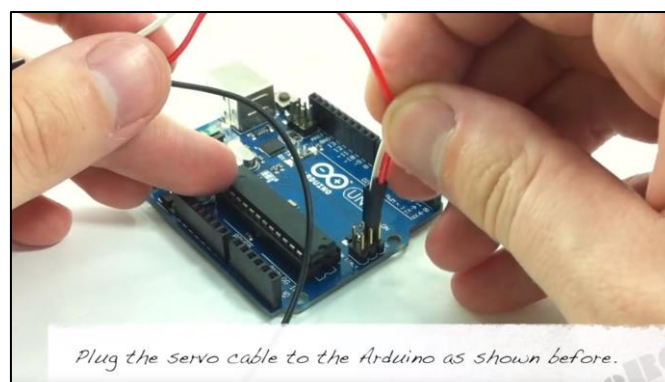


Figure A 2. ESC welding points from *eluminerRC* YouTube's channel.

Appendices

AP 1. Batteries assembly

Although it is not the main objective of this paper, as it has been mentioned before on **Chapter 2**, and due to it is part of a secondary objective, which is the fact of being able to arrange a self-built battery pack. This section contains the information of how it has been assembled this battery pack. But before proceeding let's introduce the motivator to perform this practice, he is called *GreatScott!* on his YouTube channel. On the video attached on Annex A1.1 he teaches how to build one. Down below it is shown all the devices and needed parts in order to assemble it.

Element	Specifications
6x Li-ion 18650 Batteries	25R 3.7V High Drain INR
Battery management System (BMS)	3S 20A
6 PCS Battery Spacer 18650	
1 pair XT60 Male+ Female Bullet	
OTHERS (no Quantifiable)	
2m 16AWG Standard Cable	1m Black and 1m Red
15x5mm Nickel plated steel sheet strap tape	
10mm 100ft BGA High Temperature Heat Resistant	
1 x JST-XH 3S Lipo Connector	

Table AP 1. Battery pack components.

First of all, ensure having a spot welding in order to weld the nickel ribbon to the batteries. It is mandatory to say, that soldering batteries with a tin soldering iron it is not the best nor recommended practice. This can be dangerous if the battery cells got over warmed and it could explode. On the same video attached on Annex A1.1 he teaches how to build one.



Figure AP 1. 18650 Samsung batteries

Below there are presented two figures with the final arrangement of these two packs.

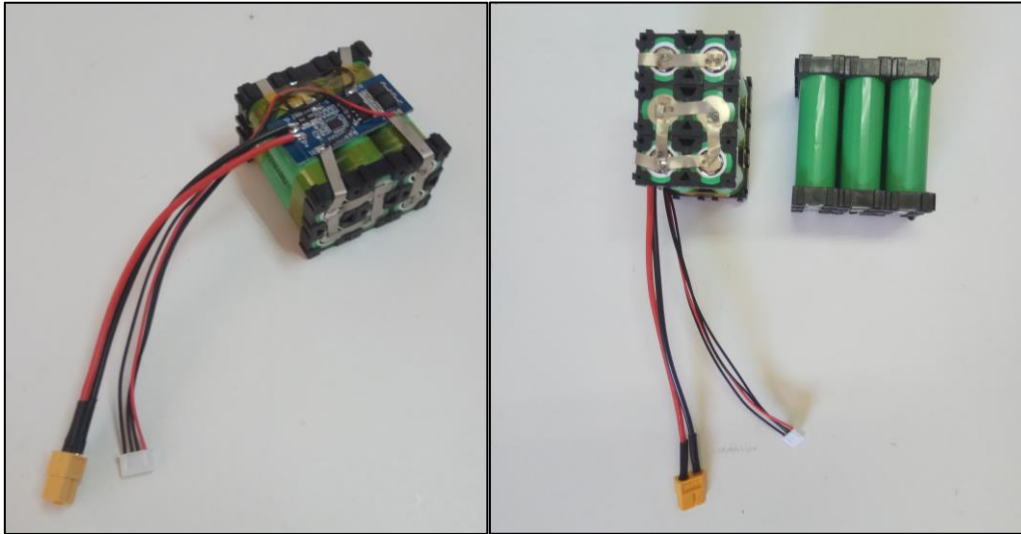


Figure AP 2. 12,4V and 2500mAh home-made battery packs.

Basically, we passed through four main stages.

Preparation It refers to acquiring all the necessary parts and components and ensuring that you have the knowledge and the tools to do it.

Ongoing verification. After welding each nickel cable or tape to the battery, the voltages have been measured to guarantee the desired voltage value.

Final verification. Once the system is mounted and finished, it is time to ensure that the system outputs have the desired voltage of 12.4 V (in this case).

Ensure and isolate. To prevent unwanted contacts with the hands or other components or parts, insulating tape has been placed.

AP 2. Design plans

AP2.1 Base

AP2.2 Right side

AP2.3 Left side

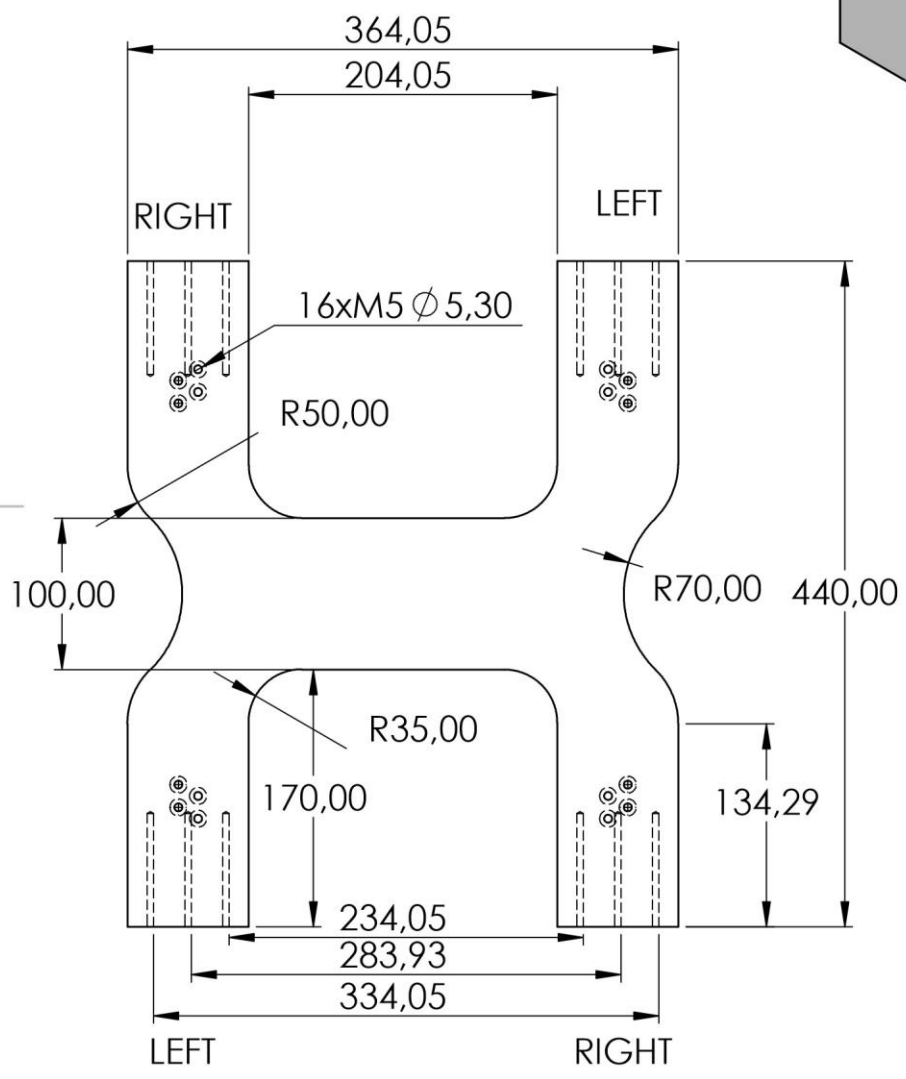
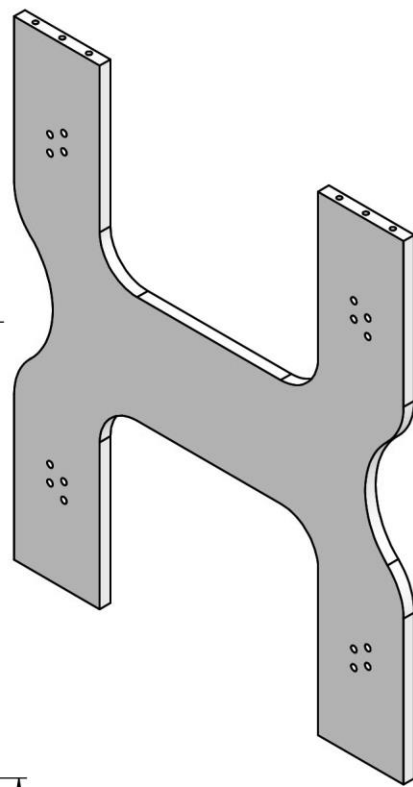
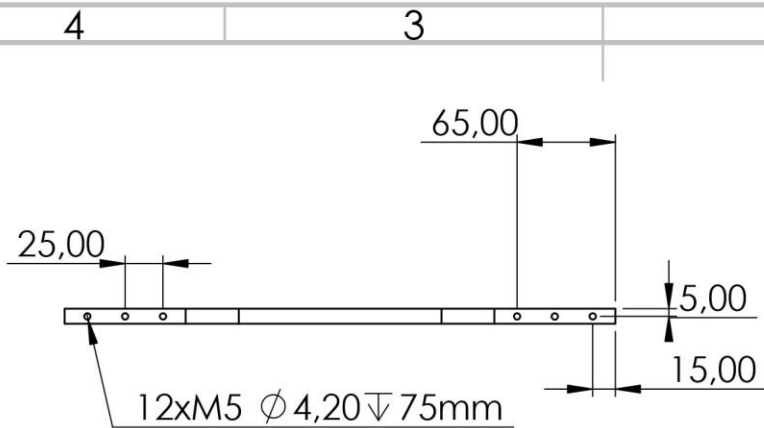
AP2.4 Tube

AP2.5 Front plug

AP2.6 Back plug

AP2.7 Dome ring

AP2.8 Main base



IF NOT OTHERWISE INDICATED:
 DIMENSIONS ARE EXPRESSED IN MM
 SUPERFICIAL FINISH:
 TOLERANCES:
 LINEAR:
 ANGULAR:



TFG REVISION

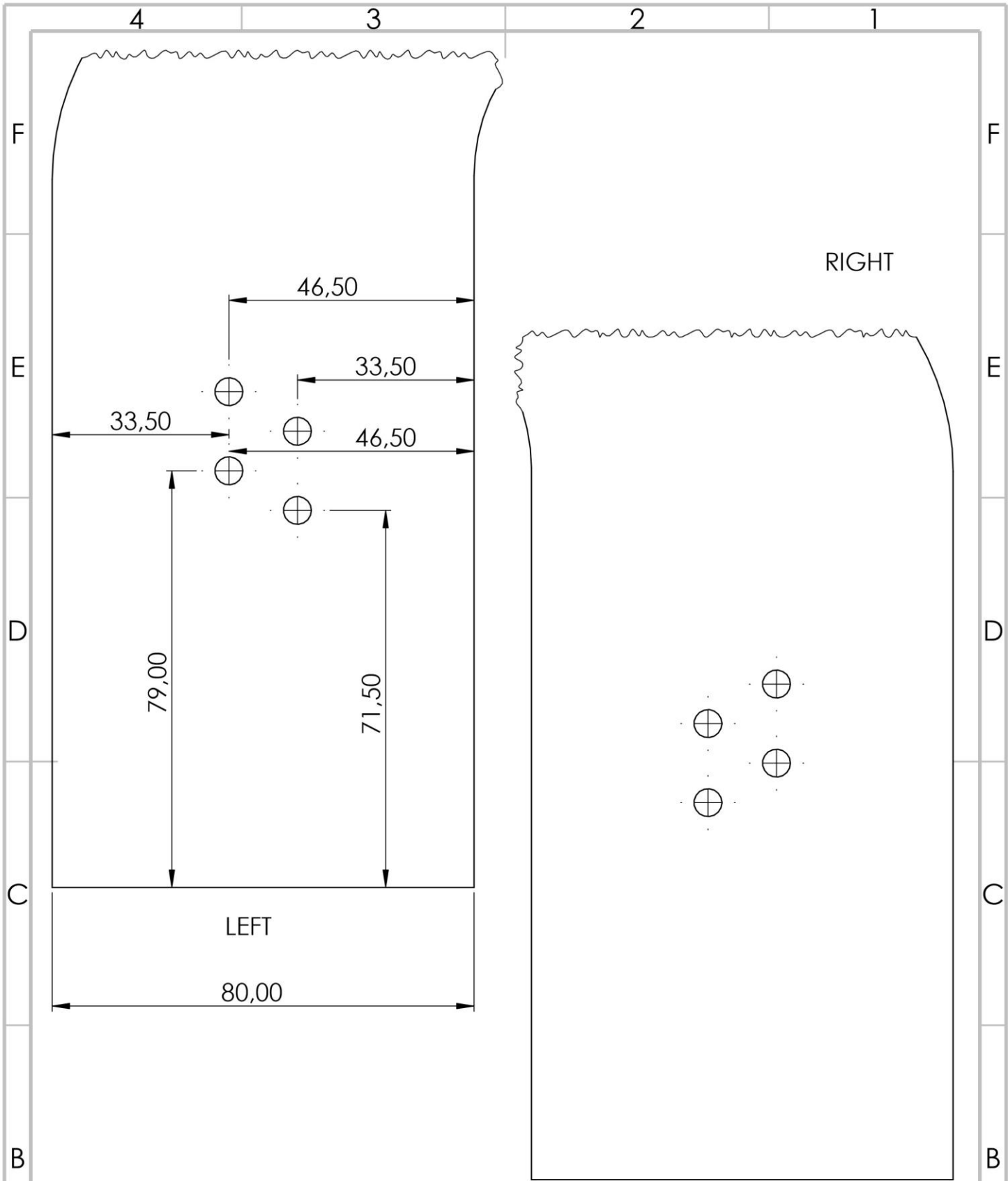
Remoted Operated Vehicle

TITLE:
1.0 Base

NAME	DATE
DRAWER	
VERIF.	
APROV.	
FABR.	
QUAL.	

MATERIAL:
 WEIGHT:

DRAWING NUMBER
 A A4
 SCALE:1:5
 Sheet 1 of 2



IF NOT OTHERWISE INDICATED:
 DIMENSIONS ARE EXPRESSED IN MM
 SUPERFICIAL FINISH:
 TOLERANCES:
 LINEAR:
 ANGULAR:



TFG REVISION

Remoted Operated Vehicle

NAME	DATE
DRAWER	
VERIF.	
APROV.	
FABR.	MATERIAL:
QUAL.	
	WEIGHT:

TITLE:
1.0 Base
 DRAWING NUMBER
 B A4
 SCALE:1:1 Sheet 2 of 2

4 3 2 1

F

F

E

E

D

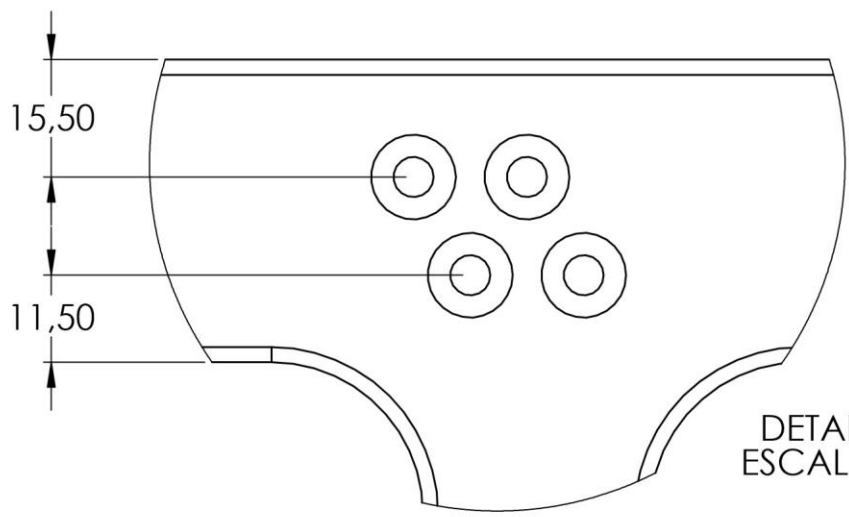
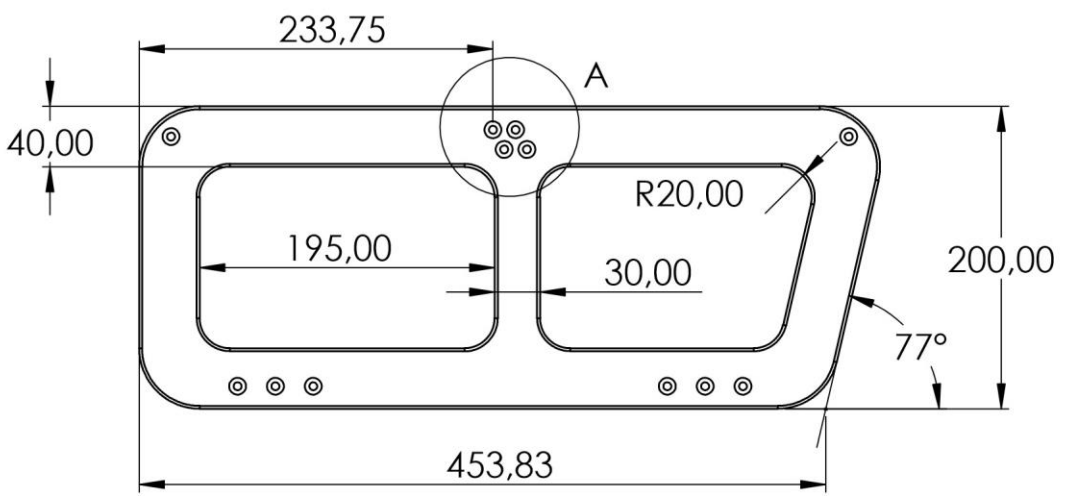
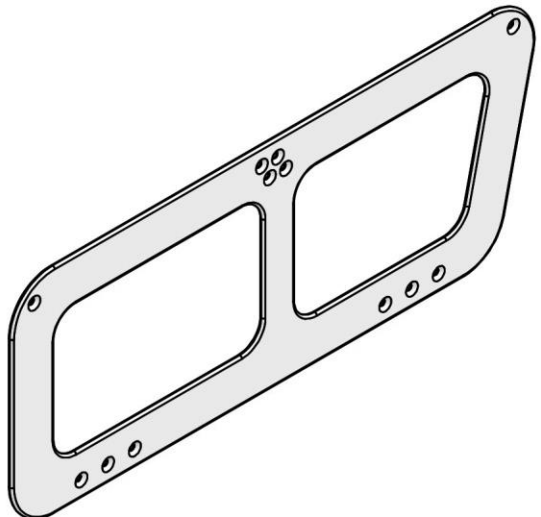
D

C

C

B

B



DETALLE A
ESCALA 1 : 1

IF NOT OTHERWISE INDICATED:
DIMENSIONS ARE EXPRESSED IN MM
SUPERFICIAL FINISH:
TOLERANCES:
LINEAR:
ANGULAR:



TFG

REVISION

Remoted Operated Vehicle

TITLE:

2.1 RIGHT SIDE

NAME	DATE
DRAWER	
VERIF.	
APROV.	
FABR.	
QUAL.	

MATERIAL:

PVC

DRAWING NUMBER

A

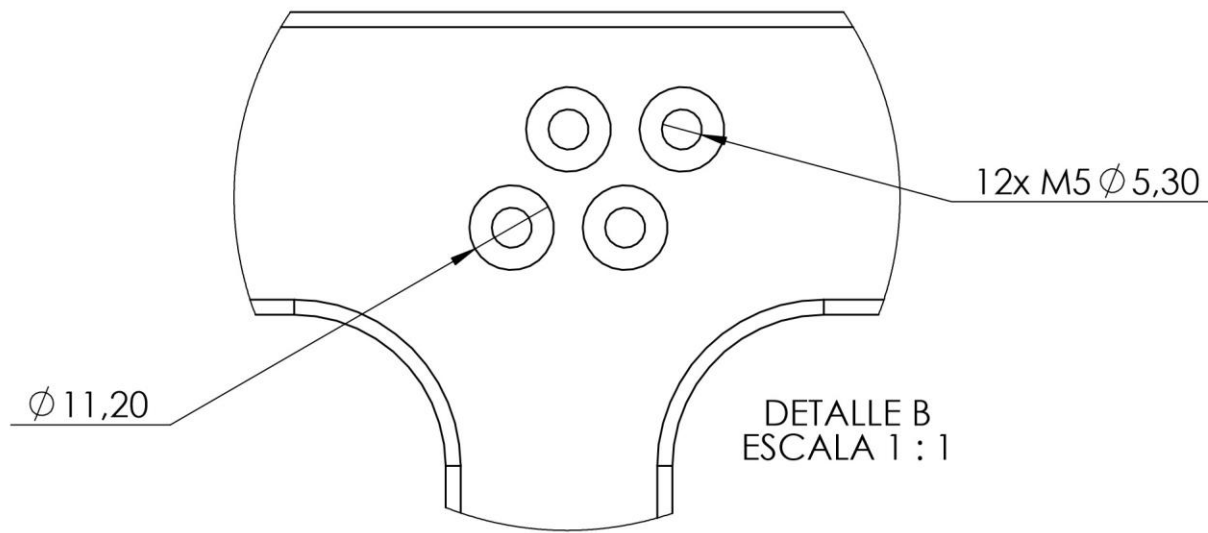
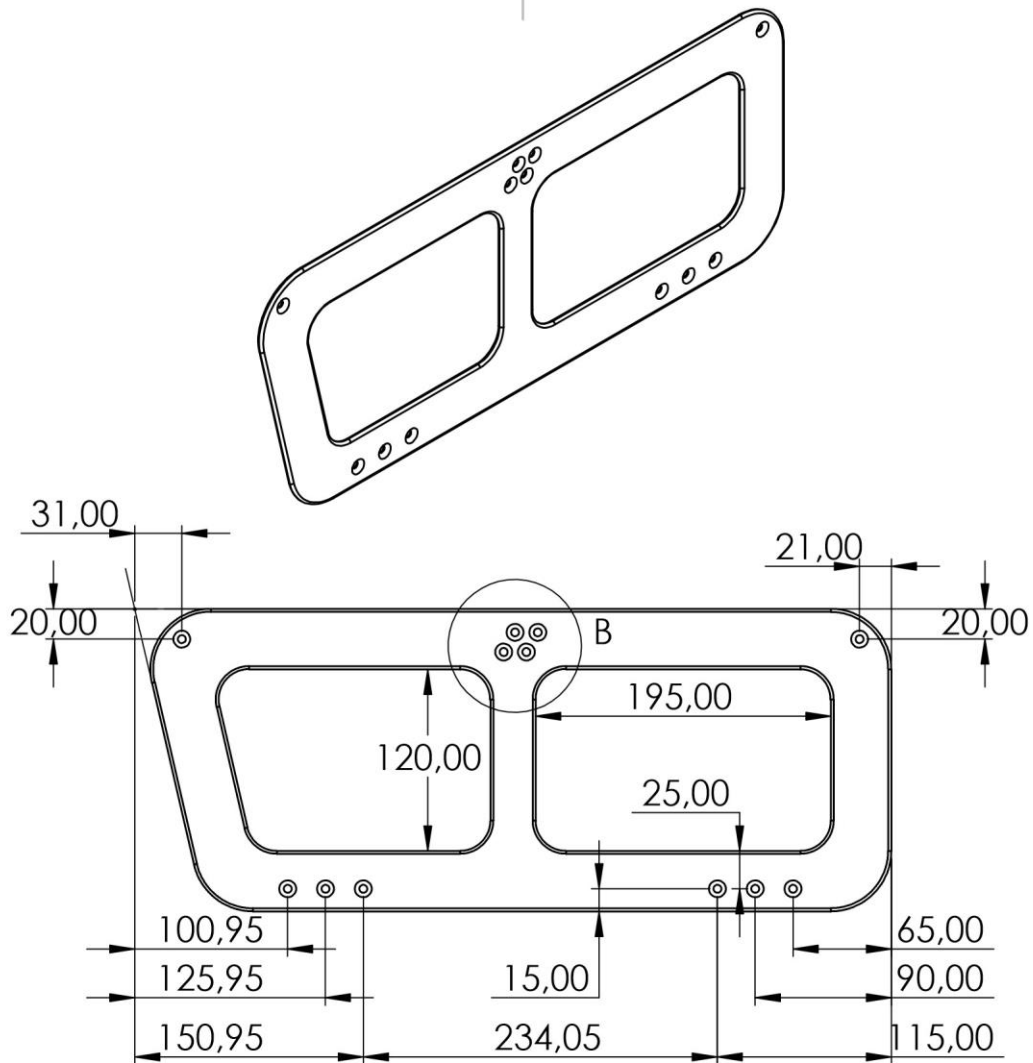
A4

WEIGHT:

SCALE:1:5

Sheet 1 of 1

4 3 2 1



IF NOT OTHERWISE INDICATED:
 DIMENSIONS ARE EXPRESSED IN MM
 SUPERFICIAL FINISH:
 TOLERANCES:
 LINEAR:
 ANGULAR:



TFG REVISION

Remoted Operated Vehicle

TITLE:
2.2 LEFT SIDE

NAME	DATE
DRAWER	
VERIF.	
APROV.	
FABR.	
QUAL.	

MATERIAL:
PVC

DRAWING NUMBER
B A4

WEIGHT: SCALE:1:5 Sheet 2 of 2

4 3 2 1

F

F

E

E

D

D

C

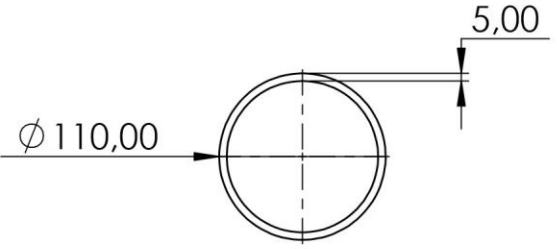
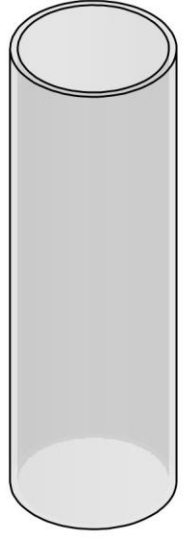
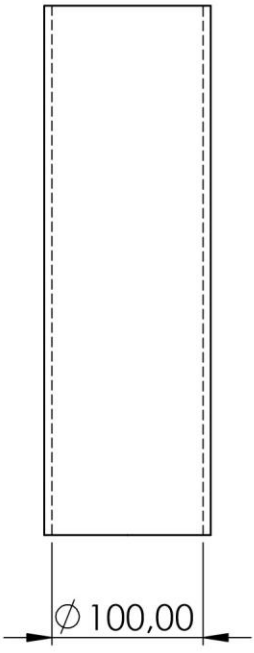
C

B

B

A

A



IF NOT OTHERWISE INDICATED:
 DIMENSIONS ARE EXPRESSED IN MM
 SUPERFICIAL FINISH:
 TOLERANCES:
 LINEAR:
 ANGULAR:



TFG

REVISION

Remoted Operated Vehicle

TITLE:

3. TUBE

	NAME	DATE
DRAWER		
VERIF.		
APROV.		
FABR.		
QUAL.		

MATERIAL:
 PMMA/PVC

DRAWING NUMBER

A4

WEIGHT:

SCALE:1:5

Sheet 1 of 1

4 3 2 1

4 3 2 1

F

F

E

E

D

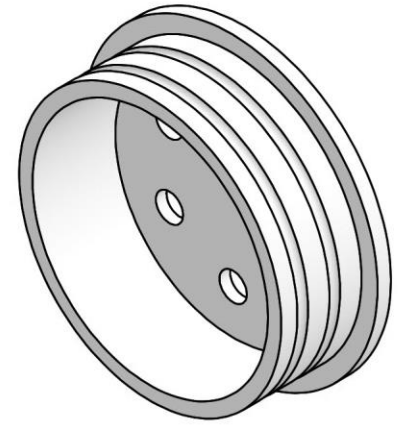
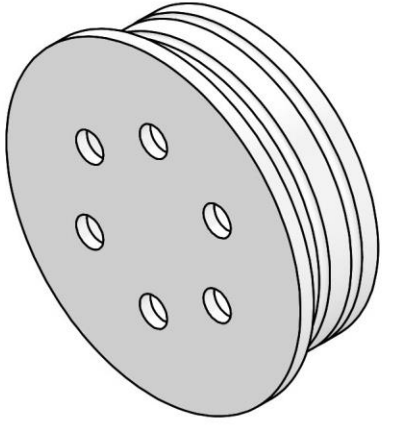
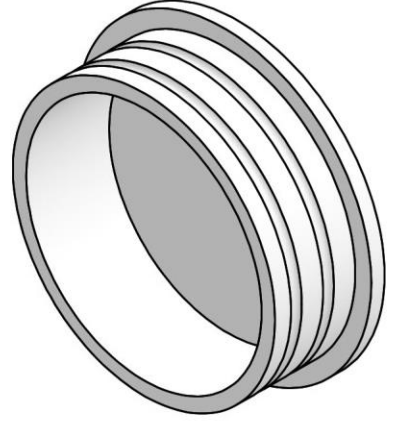
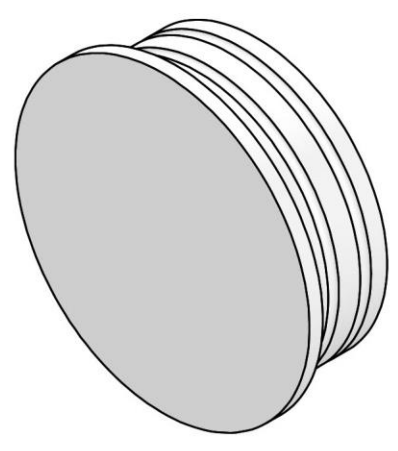
D

C

C

B

B



IF NOT OTHERWISE INDICATED:
 DIMENSIONS ARE EXPRESSED IN MM
 SUPERFICIAL FINISH:
 TOLERANCES:
 LINEAR:
 ANGULAR:



TFG REVISION

Remoted Operated Vehicle

TITLE:
4. TUBE CAPS

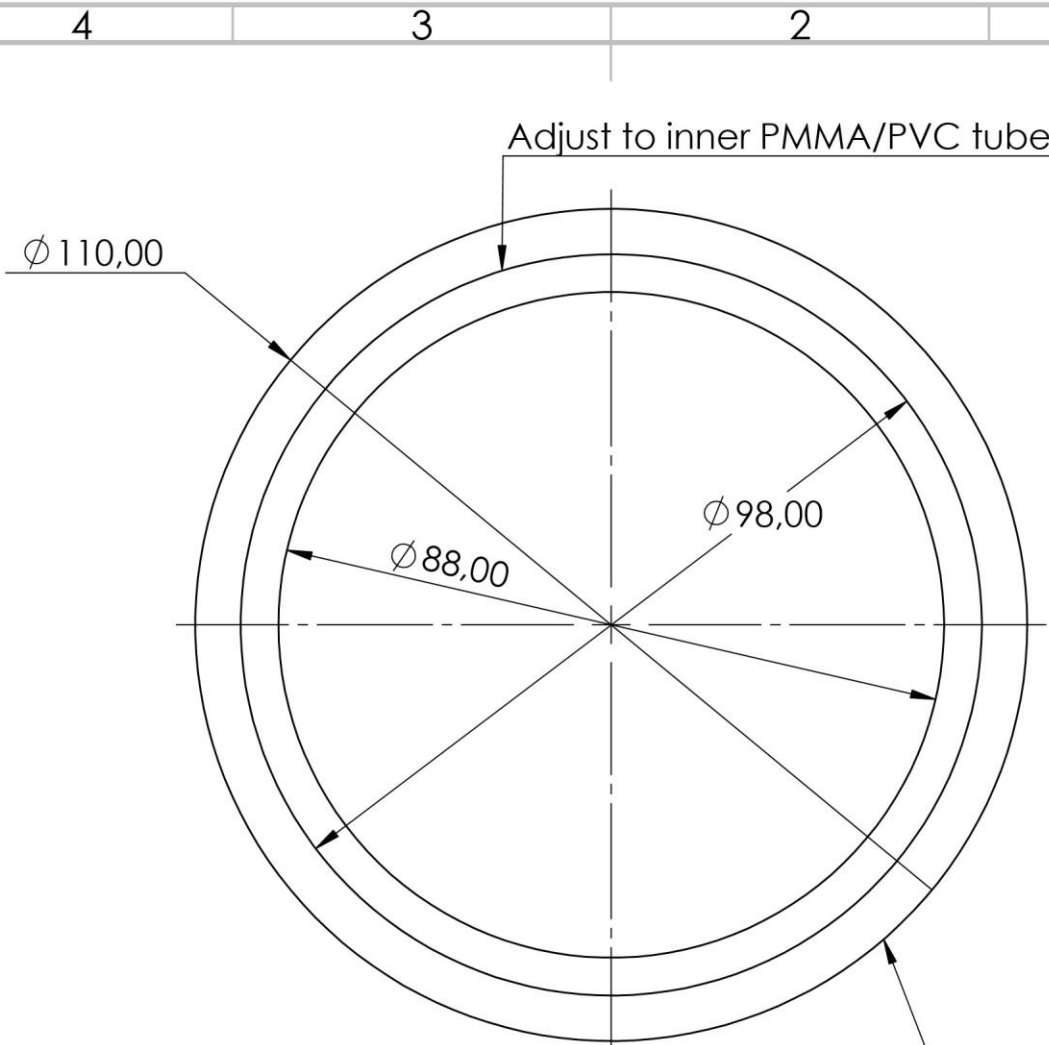
NAME	DATE
DRAWER	
VERIF.	
APROV.	
FABR.	MATERIAL: PVC/NYLON
QUAL.	
	WEIGHT:

DRAWING NUMBER
 A4
 SCALE:1:2
 Sheet 1 of 3

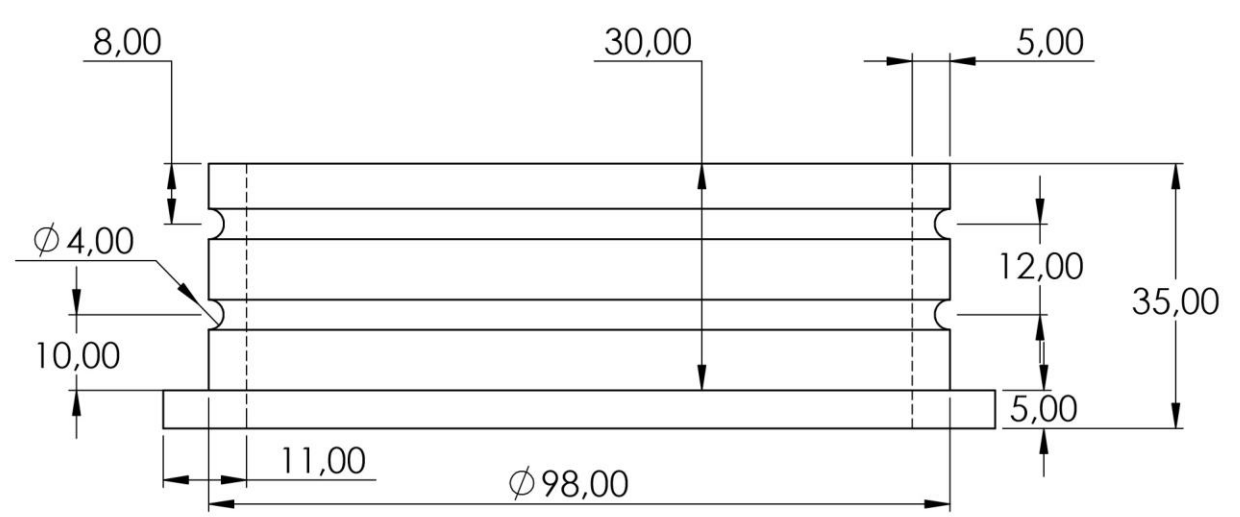
4 3 2 1

A

A



Adjust to outer PMMA/PVC tube diameter if possible



IF NOT OTHERWISE INDICATED:
 DIMENSIONS ARE EXPRESSED IN MM
 SUPERFICIAL FINISH:
 TOLERANCES:
 LINEAR:
 ANGULAR:

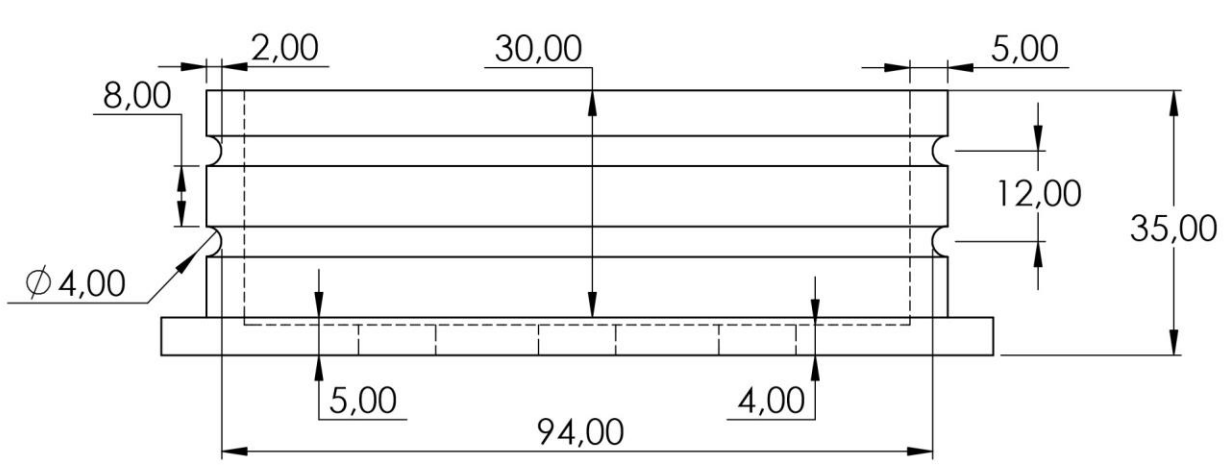
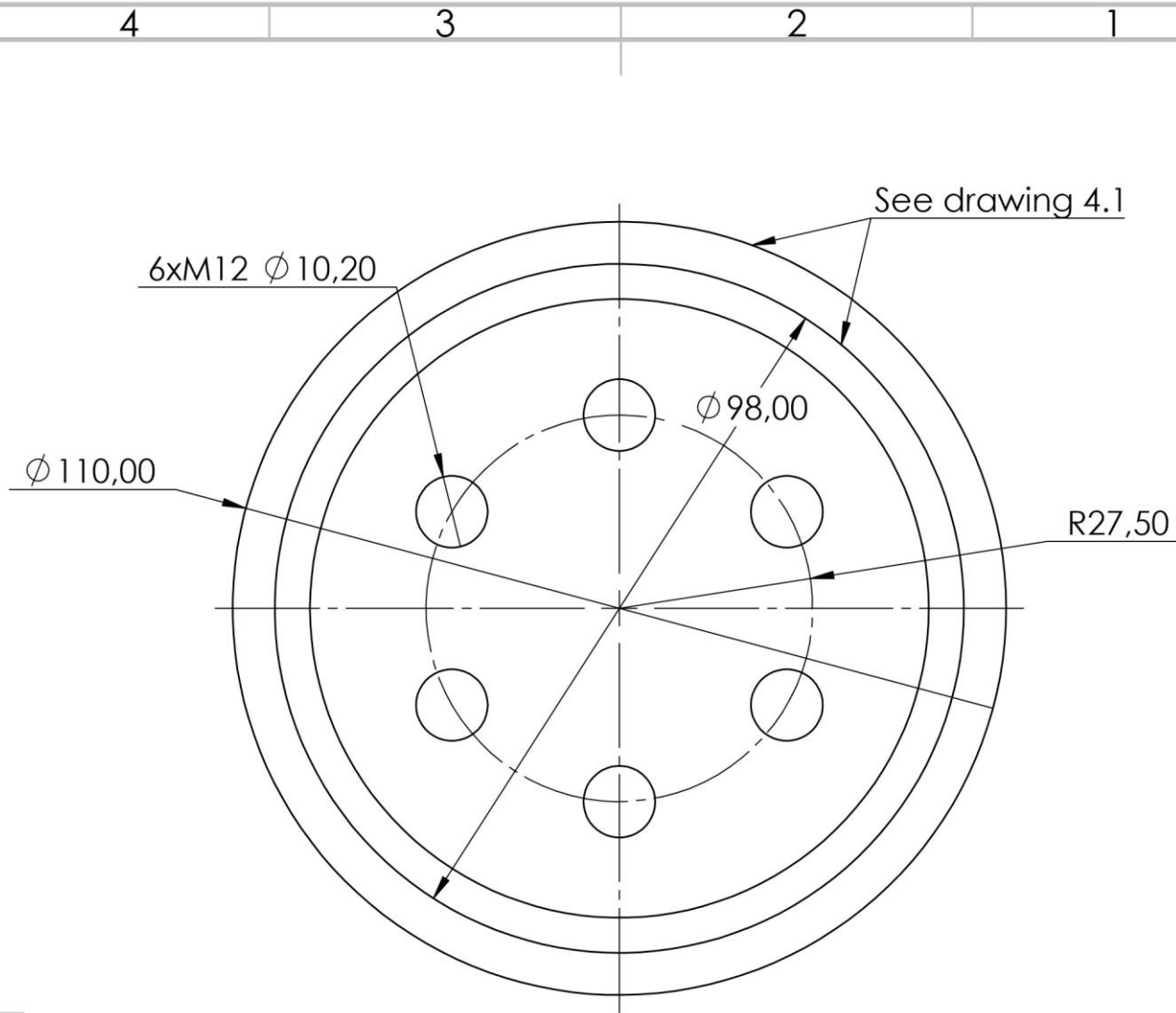


TFG REVISION

Remoted Operated Vehicle

NAME	DATE
DRAWER	09/05/2019
VERIF.	
APROV.	
FABR.	MATERIAL:
QUAL.	PVC/NYLON
	WEIGHT:

TITLE:	DRAWING NUMBER
4.1 TUBE CAP - FRONT PLUG	Front plug
	A4
SCALE:1:1	Sheet 2 of 3



IF NOT OTHERWISE INDICATED:
 DIMENSIONS ARE EXPRESSED IN MM
 SUPERFICIAL FINISH:
 TOLERANCES:
 LINEAR:
 ANGULAR:



TFG REVISION

Remoted Operated Vehicle

NAME	DATE
DRAWER	09/05/2019
VERIF.	
APROV.	
FABR.	MATERIAL:
QUAL.	PVC/NYLON
	WEIGHT:

TITLE:	
4.2 TUBE CAP - BACK PLUG	
DRAWING NUMBER	
Back plug	A4
SCALE:1:1	Sheet 3 of 3

4 3 2 1

F

F

E

E

D

D

C

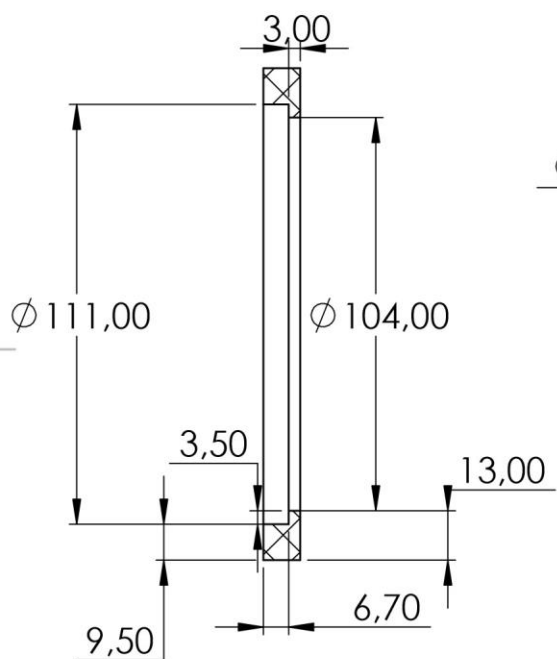
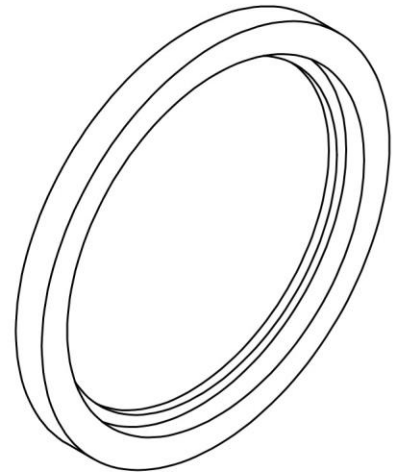
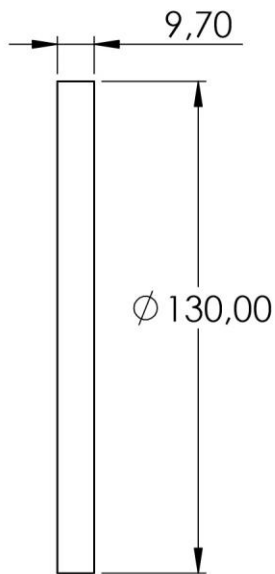
C

B

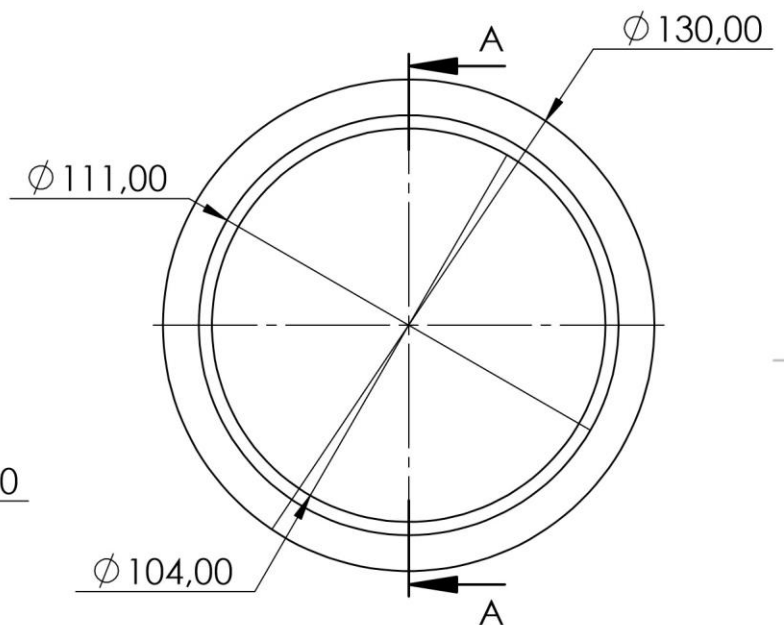
B

A

A



SECCIÓN A-A



IF NOT OTHERWISE INDICATED:
 DIMENSIONS ARE EXPRESSED IN MM
 SUPERFICIAL FINISH:
 TOLERANCES:
 LINEAR:
 ANGULAR:



TFG

REVISION

Remoted Operated Vehicle

TITLE:

DOME RING

NAME	DATE
DRAWER	26/06/2019
VERIF.	
APROV.	
FABR.	
QUAL.	

MATERIAL:
PVC/NYLON

DRAWING NUMBER

A4

WEIGHT:

SCALE:1:2

Sheet 1 of 1

4 3 2 1