Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Electrònica Industrial i Automàtica

Títol: Disseny i desenvolupament d'un robot submarí ROV GIRONA 25

Document: 1. Memòria

Alumne: Roger Feliu Serramitja

Tutor: Jordi Freixenet i Xavier Cufí

Departament: Arquitectura i Tecnologia de Computadors

Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): juny/2023

ÍNDEX

1. INTRODUCCIÓ

El present treball descriu els diferents sistemes que conformen un robot submarí, anomenat Girona 25, de baix cost, que pot ser operat remotament, i que en aquest TFG s'ha dissenyat, desenvolupat i testejat en diferents experiments.

En aquest primer apartat es descriu el plantejament del projecte, és a dir, els passos anteriors a l'inici d'aquest, com són els antecedents, l'objecte i les especificacions i abast del projecte.

1.1. Antecedents

El projecte que es presenta pretén redissenyar i desenvolupar un vehicle submarí operat remotament (ROV) de baix cost, a partir del robot submarí R2B2.

El robot submarí R2B2 és un projecte que neix de l'institut VICOROB fruit de la voluntat d'aproximar la cultura maker, el "do it yourself" (DIY) i "do it with others" (DIWO) per tal d'involucrar estudiants de secudària en enginyeria i ciència. Consisteix en dissenyar, construir i conduir un ROV (Remotely Operated Underwater Vehicle) amb materials de baix cost. Aquest robot és molt senzill, normalment aquets estudiants de secundària el construeixen pas a pas en un taller que es desenvolupa al CIRS durant 3 sessions, 20 hores.

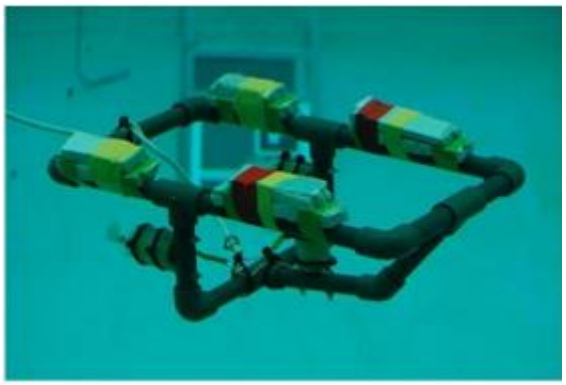

Figura 1. Robot original R2B2.

Aquest primer disseny és molt senzill, poc robust i poc sofisticat. Per aquest motiu, durant el curs 2018/2019, un estudiant de màster de la Universitat de Girona va desenvolupar un primer projecte titulat "Plataforma col·laborativa R2B2". La idea principal d'aquest primer projecte era redissenyar el robot R2B2 i construir un robot més

jugable. Aquest treball va assolir els objectius plantejats, però tenia molt de marge de millora. Principalment, tenia dos aspectes a millorar: els motors eren poc potents i com que estava cobert amb resina per aconseguir que fos estanc, cada vegada que es volia fer un canvi s'havia de refer per complet el robot.

Figura 2. Primer projecte R2B2.

Per aquest motiu, el següent curs, un nou estudiant d'enginyeria informàtica va començar un TFG per redissenyar aquest robot. Va solucionar el problema de l'encapsulat amb un cilindre acrílic que contenia tota l'electrònica del robot. Pel que fa al tema dels motors, va realitzar millores, però no van ser suficients per aconseguir els resultats esperats. Aquest projecte, a més, tenia altres aspectes a millorar com el cable USB que s'utilitzava per connectar l'ordinador de control amb l'antena que es trobava en una boia a la superfície.

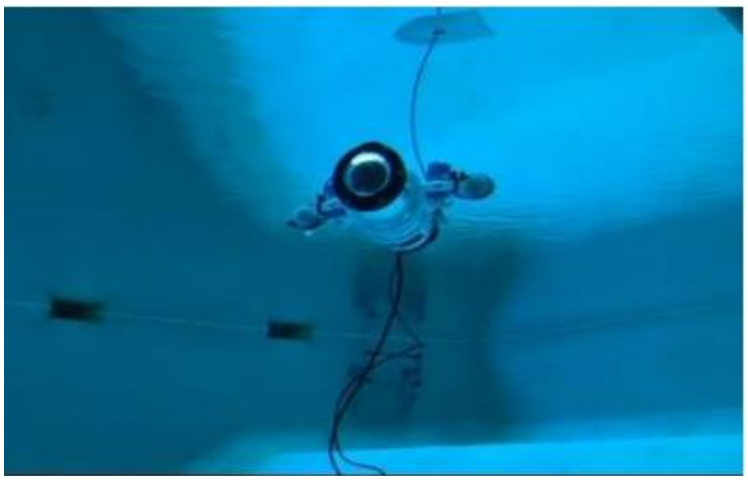

Figura 3. Segon projecte R2B2.

Per últim, el curs anterior, un altre estudiant d'enginyeria industrial va fer una altra iteració del projecte intentant tornar a corregir els problemes dels anteriors treballs. Es va plantejar dissenyar un robot que pogués baixar 25 metres i que fos de petites dimensions, de manera que una sola persona el pogués manejar. L'estudiant va utilitzar

uns nous motors més potents que aconseguien millorar substancialment la navegabilitat del R2B2. Aquests mateixos motors, però, li van causar molts de problemes d'estanquitat, ja que entrava aigua per a través dels cables d'aquests a dins del mòdul estanc. També va solucionar el problema del cable utilitzant un cable d'Ethernet creant una connexió més estable i amb una major distància.

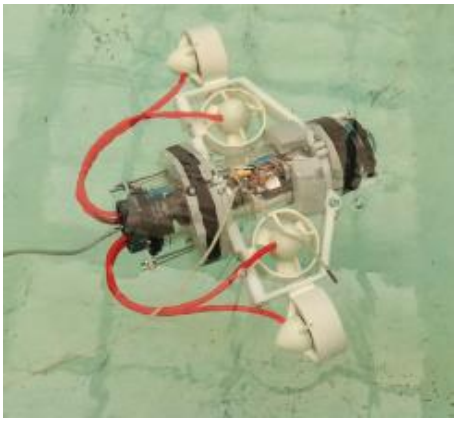

Figura 4. Tercer projecte R2B2.

1.2. Objecte

L'objectiu d'aquest projecte és redissenyar i construir un nou robot submarí utilitzant com a base els seus predecessors, corregint els errors comesos i millorant-ne les prestacions. Sempre, però, mantenint l'objectiu original de ser un robot de baix cost i que pugui ser utilitzat amb fins educatius.

1.3. Especificacions i abast

El ROV es construirà des de zero, prenent idees dels seus antecessors i innovant per solucionar els principals problemes relacionats apuntats en els anteriors projectes, molts relacionats amb el medi on ha de funcionar, l'aigua. Els problemes d'estabilitat i flotabilitat sota l'aigua es resoldran amb una nova estructura, que es dissenyarà a semblança dels robots Girona 500 i Girona 1000, d'aquí el seu nom: Girona 25.

Figura 5. AUV Girona 500.

L'estanquitat també ha estat un problema molt rellevant en els anteriors projectes, que es resoldrà millorant les juntes i utilitzant nous motors més resistents a l'aigua. Aquests motors es distribuiran en una nova configuració que aporta molta més precisió als moviments del ROV i millorarà la navegabilitat, aconseguint que sigui molt més manejable i ràpid. També es vol millorar el comportament autònom de seguiment d'objectes, ja que en els projectes anteriors el robot li costava seguir l'objecte de color de manera autònoma i no mantenia una distància amb l'objecte.

Un altre aspecte rellevant que s'hi inclourà és la instrumentació, per la qual cosa s'hi afegiran sistemes d'il·luminació i sensors de consum, orientació, profunditat, pressió i temperatura.

Finalment, a més del disseny i desenvolupament del nou robot, es plantegen un seguit d'experiments que es volen dur a terme: el funcionament amb l'ordinador, amb un comandament, amb el telèfon mòbil, amb les ulleres de realitat virtual, el mode per mantenir la profuncidtat, el comportament autònom i provar com funciona al mar, reportant els corresponents resultats i explorar possibles treballs futurs.

En el treball, en els diferents capítols, es descriurà com s'ha dissenyat i construït l'estructura, quin maquinari electrònic s'ha escollit i perquè, com s'ha programat el robot, quines comunicacions utilitza i els experiments que s'han realitzat.

2. ESTRUCTURA

L'estructura del Girona 25 està formada per dues parts: l'estructura principal que aguanta els encapsulats i els motors i el mòdul de l'electrònica que es troba dins l'encapsulat inferior que conté tota l'electrònica. Amb aquest disseny s'aconsegueixen unes dimensions generals de 40 cm d'alçada, 50 cm d'amplada i 40 cm de llargada.

Un cable CAT5E connecta el robot amb la seva boia que sura sobre l'aigua. Aquesta boia, a través d'una xarxa WiFi, es comunica amb els diferents dispositius de forma inalàmbrica. És important també el disseny d'aquesta boia, ja que si s'enfonsa o hi entra aigua, el robot no funcionarà correctament.

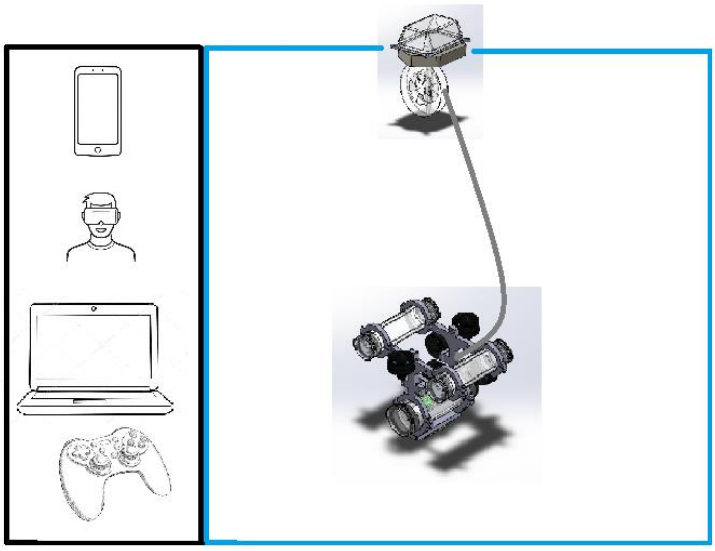

Figura 6. Esquema de com funciona el Girona 25.

2.1. Estructura principal

L'objectiu de l'estructura principal és subjectar els tres recipients estancs on es troba l'electrònica i poder-los propulsar per mitjà dels motors que també es troben enganxats a aquesta estructura.

Aquesta estructura s'ha dissenyat de manera que pugui ser fresada, tot i així, cal aconseguir que tingui una bona hidrodinàmica, d'aquesta manera, s'evita perdre energia innecessàriament. Les parts de l'estructura que han de vèncer la resistència de l'aigua quan hi ha moviment, s'intenta que la superificie sigui el més reduïda possible, però no se'ls hi pot donar forma de punxeguda, més hidrodinàmica, degut a la màquinaria i material utilitzats per construir les peces. El material que s'escull, té molt bona relació

pes-volum-resistència, que també ajuda amb la flotabilitat. Aquest material és Delrin, un termoplàstic, semi cristal·lí de gran duresa i resistència, amb el qual s'obtenen excel·lents resultats quan es mecanitza. Aquest material té una densitat superior a l'aigua, 1410 kg/m^3.

Per a subjectar els diferents encapsulats, s'han dissenyat uns anells formats per dues peces que s'ajunten per mitjà de cargols de mètric sis per a ajuntar-les. Per a garantir que no es desplacin, aquests anells són de diàmetre igual al de l'encapsulat i es deixa un espai entre les dues peces més gran del necessari, ja que d'aquesta manera els encapsulats queden ben fixats i els cargols treballen a tracció, que és quan realment exerceixen bé la seva funció. El material és un plàstic, per tant s'adapta bé a aquests esforços als quals es sotmet. Si els encapsulats fossin d'alumini anoditzat, s'hauria de posar una goma entre la peça i el tub per evitar ratllar el material i que es pogués corrompre.

Els motors es subjecten a l'estructura per mitjà de quatre forats passants per on travessen cargols de mètric tres que es collen als corresponents forats roscats que ja incorporen aquests mateixos dispositius, sempre amb les corresponents volanderes. Aquests propulsors es disposen a l'estructura amb la mateixa configuració que el Girona 500 i el Girona 1000; dos motors a darrere, un al mig i dos a dalt.

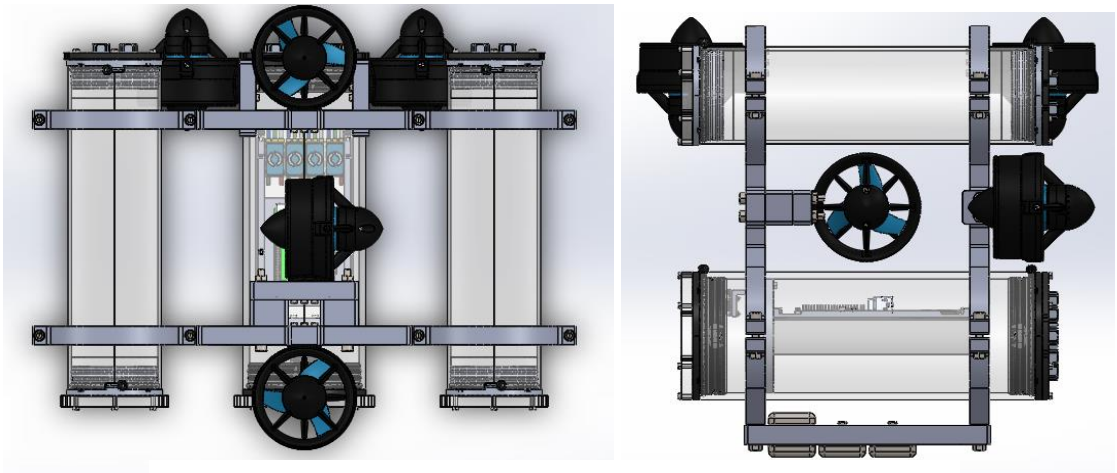

Figura 7. Alçat i perfil del disseny per ordinador del Girona 25.

Aquesta nova configuració dels motors aporta una bona mobilitat al ROV, els dos motors de darrere són els que permeten anar endavant i endarrere, igual que girar cap a un costat o l'altre, canviant el sentit de gir d'aquests. El motor del mig permet el desplaçament lateral del robot i els dos motors de dalt permeten fer-lo pujar i baixar.

És necessari que quan hi hagi dos motors de costat, la configuració de les hèlices d'aquests dos siguin oposades, és a dir, que unes siguin sentit horari i les altres sentit antihorari. Això es fa per compensar el moment que produeix el motor en girar, i evitar que el robot giri sobre ell mateix.

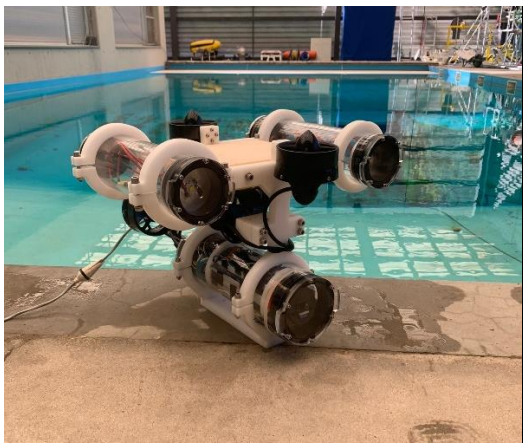

Figura 8. Resultat final del Girona 25.

2.2. Mòdul de l'electrònica

L'objectiu principal del mòdul de l'electrònica és subjectar tots els components electrònics que es troben a l'encapsulat inferior, per poder treure i posar tots aquests dispositius de manera fàcil i en conjunt de l'interior del recipient.

El disseny del mòdul s'ha fet de manera que l'ordinador de control, la Raspberry pi 4 model B, quedi al centre per poder comunicar-se amb la resta de dispositius. Un dels dispositius és la càmera que es situa a la part davantera, per poder observar el que el ROV té a davant a través de la tapa transparent de l'encapsulat. Els diferents sensors també es disposaran al voltant de l'ordinador de control, ja que han d'estar al màxim lluny possible dels controladors dels motors que poden induir corrents als dispositius més pròxims, si els cables estan molt a prop uns dels altres.

Al mig del mòdul, s'hi troba la bateria. L'estructura s'ha dissenyat de manera que si per algun motiu no es pot carregar la bateria a través del connector destinat a aquesta funció, només s'hagi d'obrir l'encapsulat per davant, evitant problemes d'estanquitat que es poden produir si es mouen constantment els cables de la part de darrere.

Un altre aspecte a tenir en compte del disseny d'aquest mòdul és que quedarà tancat a dins de l'encapsulat. S'ha dissenyat un sistema de manera que no s'hagi d'estar

contínuament traient i posant el mòdul per carregar la bateria o encendre'l i apagar-lo, que s'explicarà en el corresponent apartat, dins el capítol de maquinari electrònic. D'aquesta manera es reduiran les possibilitats que entri aigua, ja sigui pels penetrators, pel moviment dels cables, o per les tòriques, de treure i posar les tapes.

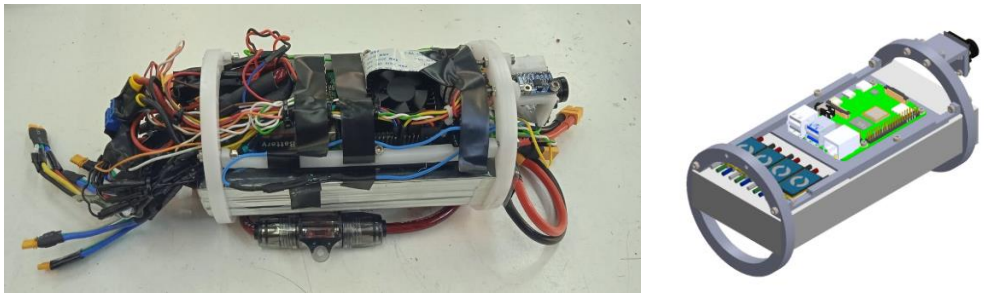

Figura 9. Conjunt del mòdul d'electrònica del Girona 25, disseny i realitat.

2.3. Boia

La boia és l'element que permet al robot comunicar-se amb els dispositius des dels quals es comanda el robot. Com que a l'interior d'aquesta s'hi troba un router que crea una xarxa WiFi, és molt important que aquest dispositiu no s'enfonsi per no atenuar el senyal.

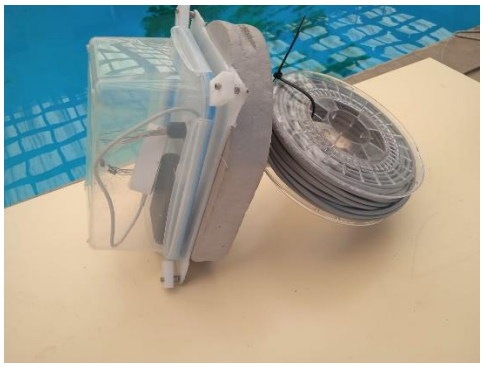

Figura 10. Conjunt de la boia.

Per tal d'aconseguir no atenuar la senyal WiFi, s'ha dissenyat la boia de manera que no s'enfonsi aquest element, posant-lo a la part més alta dins d'un recipient tancat, un taper, i fent que no es tombi. Aquest taper que conté el router i una bateria per alimentar-lo, té un forat pel qual entra el cable d'Ethernet provinent del robot, a través d'un penetrator. L'encapsulat es subjecta a la resta de l'estructura amb unes peces dissenyades per tal de que quedi fixa, fent pressió a les cantonades, i es pugui obrir i tancar sense problema. Perquè no s'enfonsi i quedi equilibrat, s'utilitza un mòdul de flotació, just a sota del taper, resseguint el seu contorn. Per tal d'aconseguir que no es tombi, l'estructura incorpora un perfil d'acer inoxidable que fa que el centre de gravetat de la boia estigui per sota el centre de flotació, mantenint així l'estabilitat. La bobina que recull el cable d'Ethernet

que no és necessari en funció de la profunditat a la que es troba el Girona 25, es troba submergida enganxada en el perfil de manera que també ajuda a mantenir el centre de gravetat baix.

Figura 11. La boia funcionant dintre l'aigua.

2.4. Anàlisi de l'estructura

Abans de donar per acabat el disseny de l'estructura cal tenir diferents aspectes en compte com poden ser els graus de llibertat, la flotabilitat i l'estanquitat. Amb els acabats també cal ser curós, s'utilitzen brides per a lligar els cables a l'estructura per evitar que afectin la hidrodinàmica o enganxar-se a les hèlices d'algun motor. A més, la punta dels cargols es banya amb Loctite 242 per evitar el fet que entri aigua als espais o el propi funcionament pugui fer que s'afluixin.

2.4.1. Anàlisi graus de llibertat

Degut a la distribució dels motors, el Girona 25 té quatre graus de llibertat: el moviment longitudinal del robot, "Surge", el moviment cap a l'esquerra i cap a la dreta, "Sway", el moviment del robot respecte al seu eix vertical, "Heave", i la rotació sobre l'eix vertical, "Yaw". Aquests quatre graus de llibertat són més que suficients pels objectius que ha de complir aquest robot.

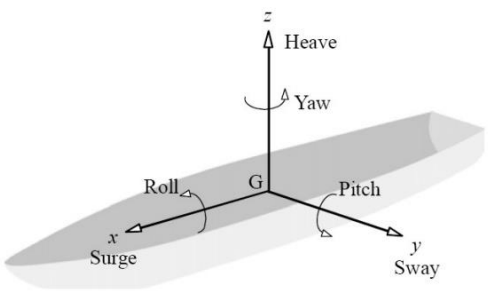

Figura 12. Dibuix per veure els graus de llibertat del moviment d'un cos.

2.4.2. Anàlisi flotabilitat

En aquests tipus de robot, es treballa amb flotabilitat neutra, és a dir, que el pes del robot quedi compensat per la força de baix cap a dalt igual al pes del volum del fluid que desallotja, seguint el principi d'Arquímedes, i estabilitat, que el centre de masses quedi per sota el centre de flotabilitat. Per a dissenyar el robot seguint aquests principis, s'ha utilitzat el Solidworks, dissenyant l'estructura assignant tots els materials a les peces que s'utilitzen i després assignant com a material l'aigua, per saber quin volum del fluid estan desplaçant. D'aquesta manera s'obtenen els centres de massa i flotació.

Un cop s'ha acabat el disseny, cal comprovar que el centre de masses i el centre de flotabilitat estiguin alineats i al centre de l'estructura, sempre amb el centre de masses per sota de l'altre. Això es fa perquè quan el cos es posa a l'aigua aquests dos centres queden alineats, així el robot quedarà estable, i amb el centre de masses per sota per aconseguir que qualsevol moviment en el "Roll" o el "Pitch" quedi anul·lat per l'efecte de la gravetat i la força de flotació que creen un parell que torna el robot una altra vegada a la posició vertical.

El programa indica que el centre de gravetat del cos es troba a X=0 mm, Y=81mm i Z=96mm, respecte l'eix de coordenades de la figura 11, mentre que el centre de flotabilitat es troba a X=0, Y=119mm i Z=95 mm, respecte l'eix de coordenades de la figura 12.

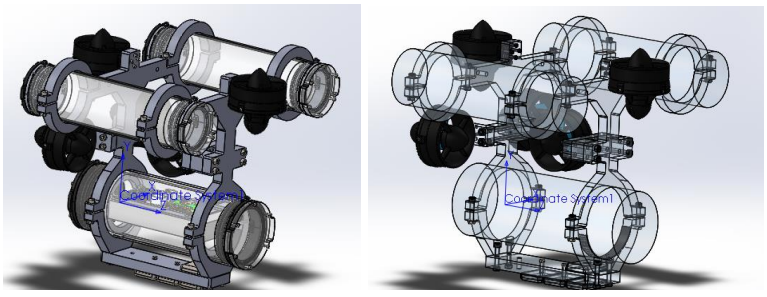

Figura 13. Centre de flotació, centre masses i centre de coordenades del Girona 25.

Es pot comprovar que els dos centres estan pràcticament alineats entre ells i amb el centre del robot, el centre de masses es troba uns quatre centímetres per sota el de flotabilitat. Cal tenir en compte que això és una simulació i que a l'hora de la posada en marxa caldrà fer ajustaments de la distribució dels pesos perquè quedin al màxim d'alineats.

L'altre aspecte que cal comprovar és que la flotabilitat sigui nul·la, és a dir, que el pes que el cos desplaça d'aigua, sigui igual que el pes del cos. D'aquesta manera s'aconsegueix que el robot ni pugi ni baixi quan no s'acciona cap motor, encara que sempre es deixa una flotabilitat una mica positiva per si en algun moment hi hagués algun error o el robot es quedés sense bateria, acabes tornant a la superfície. Encara que si la boia és a prop, es podria arribar a estirar pel cable.

Seguint el procediment descrit al primer apartat, s'obté una massa de 9,89 kg del robot i desplaça 9,95 dm^3 de fluid, que es considera aigua, per tant són 9,95 kg. Per tant, hi haurà una força resultant de 0,06 kg*9,81 m/s = 0,6 N que farà pujar el ROV cap a la superfície.

Si es vol fer servir el robot al mar, la densitat de l'aigua seria diferent, per tant, caldrà afegir un pes. El pes teòric que cal afegir al robot, tenint en compte que la densitat de l'aigua del mar és de 1,025 kg/dm^3, que el volum del model del robot és de 9,95 dm^3, el pes del robot fora l'aigua és de 9,89 kg i es vol mantenir una força que faci emergir el robot de 0,6 N, ha de ser de 250 grams.

Tot i això, aquets models són una simulació, per tant en el moment de la posada en marxa caldrà fer ajustaments. Per aquest motiu a l'estructura es deixen forats o espais per posar mòduls de flotació o pesos, per acabar d'ajustar la flotabilitat.

2.4.3. Estanquitat

Els encapsulats utilitzats són estancs per aconseguir que l'aigua no entri en contacte amb l'electrònica. Per aconseguir aquesta estanquitat s'utilitzen tubs acrílics que queden tancats amb dues juntes tòriques, per cada costat, correctament lubricats amb grassa de silicona i prèviament netejats amb alcohol isopropílic. Una tercera tòrica evita que passi aigua entre la tapa i la brida amb juntes tòriques (flange o-ring).

Figura 14. Brides amb juntes tòriques dels encapsulats de diferentes mides.

Cal tenir en compte, però, que alguns cables, com els dels motors, han de sortir fora dels encapsulats. Per aquest motiu s'utilitzen aquests elements anomenats "penetrators", que també utilitzen juntes tòriques per a impedir que l'aigua entri dins el recipient pel forat i també una resina epoxi perquè no entri aigua per dins el cable o el forat per on passa aquest. Cal escollir una epoxi que sigui bastant tou per tal que s'ajusti als moviments del cable.

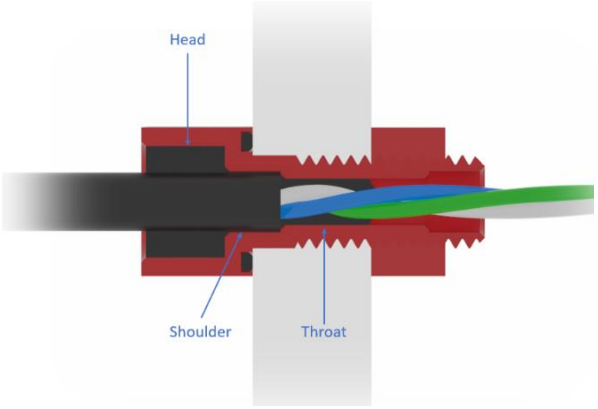

Figura 15. Assemblatge d'un penetrator amb el corresponent cable.

Pel que fa al subconn, funciona d'una manera diferent, molt més efectiva. El cable que entra dins l'encapsulat i el que està en contacte amb l'aigua no són el mateix, s'ajunten dins el connector per mitjà d'un element de contacte que aconsegueix l'estanquitat utilitzant neoprè. Aquests connectors també utilitzen epoxi per aïllar els cables. D'aquesta manera amb aquests connectors es pot canviar el cable de sortida sense haver de desacollar els connectors ni refer-los. A més, el sistema d'aïllament amb el neoprè és molt millor i fiable, igual que el fet que el cable de fora i de dins siguin diferents.

Figura 16. Connector subconn de 6 pins.

Un altre aspecte important a tenir en compte és la pressió a dins els encapsulats, ja que quan es tanquen, les tapes no deixen sortir l'aire a l'exterior d'aquest, de la mateixa manera que no permeten que l'aigua entri a dins. Això pot produir que la pressió a

l'interior augmenti i puguin obrir-se les tapes mentre el robot està dins de l'aigua. Per aquest motiu, es posen uns taps en forma de cargol de M5, que porten una junta tòrica per evitar que entri aigua a través d'ells. Aquests cargols permeten que l'aire pugui sortir dels encapsulats quan es posen les tapes, alliberant la pressió, i que quedin tancats per tal que no hi entri aigua.

Figura 17. Cargol M5 amb tòrica.

És molt important que no entri aigua a l'electrònica, ja que podria fer malbé els components, produir curtcircuits o provocar que es cremi algun component. Cal seguir les indicacions del fabricant i vigilar la profunditat a la qual es baixa, ja que com més avall, més pressió i més possibilitats que es produeixi una fuga.

Cal tenir en compte, també, que tots els materials que estan en contacte amb l'aigua no s'oxidin, per això la majoria són d'acer inoxidable o d'alumini anoditzat, aquests últims s'ha de vigilar de no ratllar-los.

Pel que fa a la boia, s'ha comprovat que l'encapsulat en forma de taper que conté la bateria i el router, no hi entra aigua fins a 5 metres sota l'aigua. Aconseguint que un element de baix cost sigui funcional dins aquest robot que intenta mantenir el seu pressupost al mínim.

3. MAQUINARI ELECTRÒNIC

Degut als objectius proposats per a aquest ROV, es necessiten incorporar diferents sistemes. El principal i més important, és moure'l, amb els motors, i poder veure per on es mou, amb la càmera. Com que el robot pot baixar fins a profunditats on la il·luminació és escassa, s'incorporen llums LED per a poder veure-hi, igual que sensors d'orientació per saber el posicionament d'aquest. S'incorporen per seguretat sensors de consum, per no quedar-se sense bateria, i profunditat, pressió i temperatura, per poder avaluar les condicions ambientals. Pel que fa a l'estanquitat, hi ha sensors d'aigua l'interior del robot, sistemes per poder engegar, apagar i carregar-lo sense necessitat d'obrir-lo. Tots aquests elements necessitaran ser alimentats i controlats per altres dispositius que s'aniran comentant al llarg d'aquest capítol.

3.1. Raspberry Pi 4

Com en els dos treballs anteriors, s'ha optat per utilitzar la Raspberry Pi 4 model B 4 GB per a fer el control i comandament de tots els elements del ROV. S'utilitza una targeta de memòria SD de 16GB que conté el sistema operatiu i la memòria de la computadora i, per tal de reduir la temperatura, s'incorpora un dissipador de calor amb ventilador. La mateixa Raspberry incorpora un programa que monitoritza la temperatura i dona una senyal al ventilador per a què funcioni quan és necessari amb una senyal PWM.

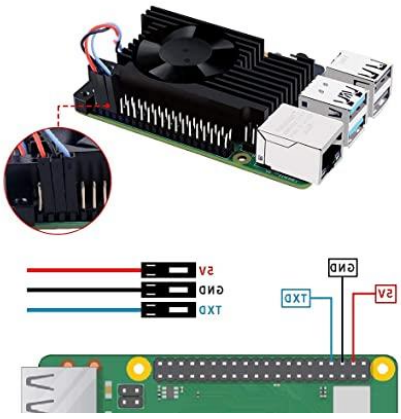

Figura 18. Raspberry Pi 4 Model B 4GB amb targeta SD de 16GB i dissipador de calor.

La Raspberry Pi és un ordinador de placa única que va ser dissenyat amb l'objectiu d'ensenyar a programar a gent jove. Tot i això, pel seu baix cost, l'ús d'aquests ordinadors s'ha estès a molts altres camps com la robòtica o la indústria. El model utilitzat, compta amb un processador de quatre nuclis, un port Ethernet, 2,4 GHz i 5 GHz

IEEE 802.11ac i un port USB-C per l'alimentació. A més, aquest ordinador fa servir el sistema operatiu Raspberry Pi OS el qual està basat en GNU/Linux Debian i pot ser programada amb el llenguatge Python. Pel seu senzill funcionament, el llenguatge de programació, les reduïdes dimensions i la resta de prestacions, s'escull aquest ordinador de control per al ROV.

3.2. Càmera Raspberry

La càmera utilitzada és una càmera per Raspberry de 5MP, la qual ve equipada amb un connector per Raspberry, a més d'una lent d'ull de peix la qual permet obtenir un angle de visió de fins a 200º. S'utilitza aquesta càmera, ja que aporta una resolució suficient pels objectius del robot i un rang de visió molt extens sense la necessitat d'utilitzar cap motor per tenir un rang de visió més ampli.

Figura 19. Càmera per Raspberry de 5MP amb angle de 200º de visió.

3.3. Motors

Els propulsors utilitzats per aquest ROV són els de Bluerobotics, els T200. Són motors lubricats amb aigua, amb molta potència, eficients i compactes, a un preu assequible.

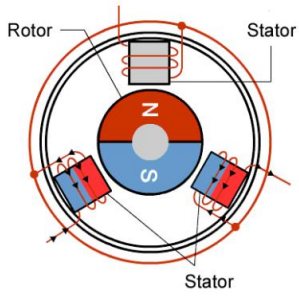

Figura 20. Esquema intern d'un motor trifàsic sense escombretes.

Aquest motor és trifàsic i es pot fer funcionar amb tensions entre 10 i 20V. Com la resta de motors, consten de dues parts principals: l'estator i el rotor. L'estator correspon a la part fixa del motor; que en aquest cas consta de tres bobines situades a 120º entre elles, on en cada una d'elles es pot induir un camp magnètic. El rotor correspon a la part mòbil

del motor; consistint en un imant permanent, el qual genera un camp magnètic. La rotació s'aconsegueix induint un camp magnètic convenient a les bobines de l'estator, fent que aquestes atreguin o repelin l'imant del rotor.

Aquest propulsor consta d'un motor sense escombretes completament inundat amb el debanat i l'estator del motor encapsulats, amb imants i rotor revestits. El cos del propulsor i les hèlices estan fetes de plàstic policarbonat i els components que estan en contacte amb l'aigua d'acer inoxidable, perquè no s'oxidin.

Figura 21. Foto del propulsor T200.

Per a fer-lo funcionar, activant les bobines de l'estator per crear els camps magnètics, és necessari un controlador de velocitat electrònic (ESC), que s'activa amb senyal PWM. El fabricant proporciona el següent gràfic, on relaciona els kg de força que exerceix el motor en funció de la senyal de control. Es pot observar la diferència de consum en funció de la tensió que s'aplica. En aquest projecte es farà servir un màxim de 1 kg de força, a una tensió d'uns 12V, per tant, el rang de valors del senyal de control serà de 1300 a 1680 microsegons. El valor de 1kg s'ha obtingut tenint en compte la relació potència i pes del Girona 500 i el Girona 1000, ja que el Girona pesa uns 100 kg, depenent de la configuració, i cadascun dels seus motors té 10 kg de força, per tant pel Girona 25, que pesa uns 10 kg, la força que ha de fer cada motor és de 1 kg.

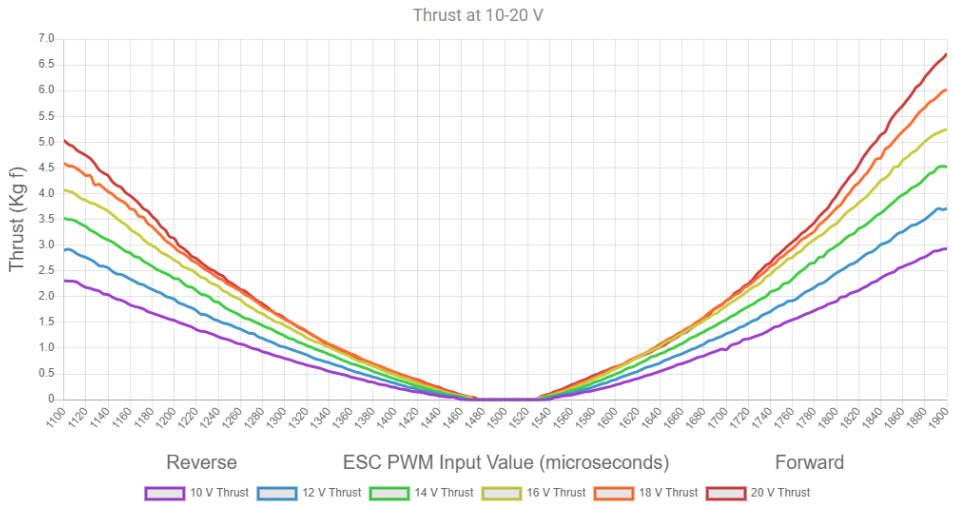

Figura 22. Gràfic de la força en kg que exerceixen els motors en funció de la senyal de control.

Aquest altre, relaciona la intensitat, en ampers, que consumeix en funció, també, del senyal de control, en microsegons. En aquest cas també es pot observar la diferència de consum en funció de la tensió que s'aplica. Tenint en compte els resultats de la gràfica anterior, s'obté que el consum de cadascun dels motors serà entre 0 i 3 A.

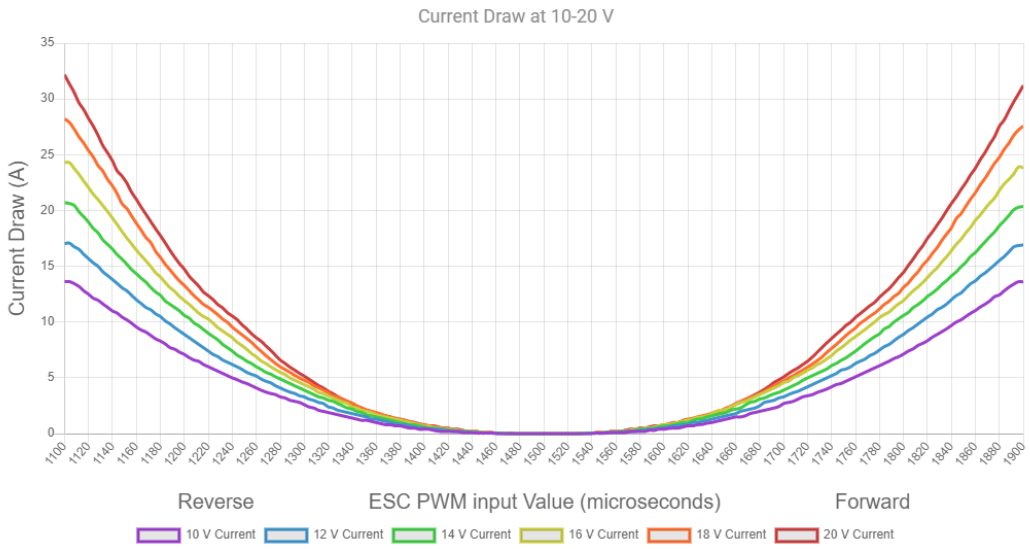

Figura 23. Gràfic del corrent que consumeixen els motors en funció de la senyal de control.

S'escullen aquests motors, per la seves prestacions i, sobretot, per la fiabilitat, ja que s'ha comprovat que no entra aigua a l'interior de l'encapsulat a través d'ells.

3.4. Controlador electrònic de velocitat

Un controlador electrònic de velocitat o ESC, és un dispositiu amb la capacitat de controlar la velocitat i sentit de gir de motors trifàsics mitjançant l'intercanvi de la polaritat de les bobines dels motors. Aquest element és bàsicament un circuit electrònic format per transistors i que pot ser controlat amb l'ús de senyals de modulació d'amplada de polsos (PWM). En aquest projecte s'utilitzen els controladors proporcionats per la mateixa empresa que els propulsors, Bluerobotics, s'anomenen Basic ESC.

Figura 24. Controlador Basic ESC.

El color dels tres cables que s'han de connectar al motor és només per guiar-se, ja que intercanviar dos d'aquest únicament invertiria el sentit de gir del motor, que al ser bidireccionals, no representa cap problema.

Són aquests controladors els encarregats de processar el senyal que reben de la Raspberry i enviar-la als motors, amb la potència corresponent, ja que el senyal de la Raspberry és de 3,3V. S'utilitzen aquests controladors, ja que són de la mateixa empresa que fabrica els motors i permeten ser controlats per una Raspberry.

3.5. BAR30

Un altre component que també és de Bluerobotics és el sensor de temperatura, pressió i profunditat anomenat BAR30. El seu nom és degut al fet que el seu rang de mesura és de 0 a 30 bar, és a dir, pot mesurar fins a una profunditat de 300 metres.

Figura 25. Sensor BAR30.

Aquest sensor es basa en l'integrat MS5837-30BA, proporcionant un encapsulat d'alumini anoditzat molt pràctic per a poder-lo incloure en els encapsulats d'aquesta mateixa empresa. El sensor es comunica amb el protocol I^2C de 3,3V i es pot alimentar en un rang de 3,3 a 5,5V. El seu consum màxim és de 1,25 mA. En la següent figura es pot veure l'esquema electrònic del sensor.

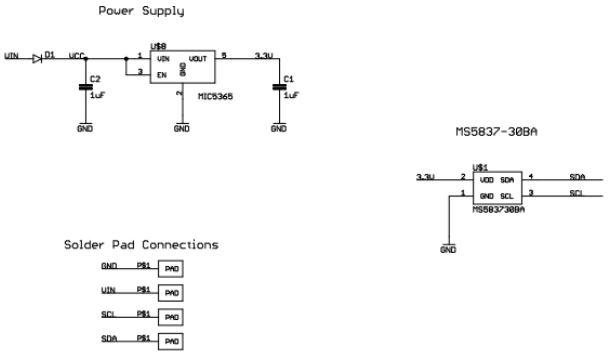

Figura 26. Esquema electrònic del sensor BAR30.

S'escull aquest sensor, ja que és del mateix fabricant que els encapsulats i la seva forma de penetrator fa que sigui fàcil integrar-lo al robot. A més, el fabricant també proporciona una llibreria per a poder llegir els valors de temperatura i pressió que proporciona i obtenir la profunditat a la qual es troba. També es recomana assecar el sensor una vegada al dia, ja que les lectures de la pressió i la temperatura poden desviar-se.

3.6. Sensor d'aigua

El sensor d'aigua escollit per a detectar aigua dins de l'encapsulat inferior del ROV és el sensor d'aigua d'Arduino. Aquest sensor consta de pistes al descobert que alternen el negatiu i el positiu, de manera que quan una gota d'aigua fa contacte entre dues pistes permet que el corrent passi a través seu i un circuit amb un transistor retorna un valor analògic corresponent a la quantitat d'aigua que està en contacte amb les pistes al descobert.

El problema és que aquest dispositiu està dissenyat per a Arduino, que té entrades analògiques, en canvi, la Raspberry no en té. És per aquest motiu que s'ha hagut de modificar el circuit de l'integrat perquè funcioni de la manera desitjada.

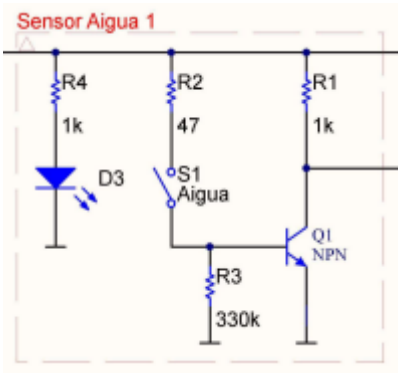

Figura 27. Circuit dissenyat per al sensor d'aigua.

El circuit retorna un senyal de sortida de nivell alt, 3,3V, quan no es detecta aigua, funcionant com un polsador d'emergència, de manera que si deixés de funcionar o es desconnectés l'ordinador de control se n'adonaria, ja que el fet que entri aigua dins el robot és molt important detectar-lo.

Aquest circuit funciona a una tensió de 3,3V, quan s'alimenta s'encén un led per a indicar que hi ha voltatge. El circuit es basa en un transistor JY3 treballant a tall i saturació. S'escull una resistència de base bastant baixa, perquè el corrent quan hi hagi contacte

amb l'aigua sigui bastant elevat i una resistència de col·lector més elevada perquè el corrent de col·lector-emissor sigui menor i el consum també sigui menor. Es posa una resistència molt gran, per evitar estirar corrent de base, de 330 kΩ en paral·lel amb la base i l'emissor per evitar que la base quedi a l'aire mentre no es detecta aigua, ja que la base podria prendre un nivell de voltatge no desitjat i fer que el circuit no funcioni correctament.

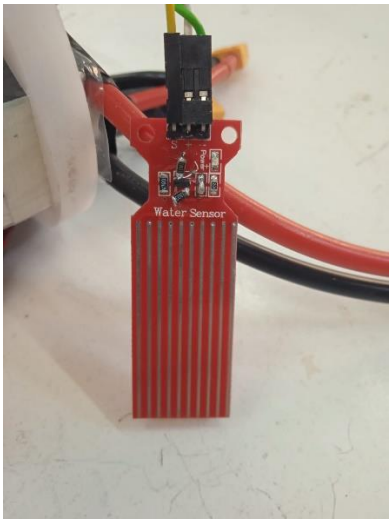

Figura 28. Resultat final del sensor d'aigua.

3.7. Llums LED

A l'hora de seleccionar els llums LED per a un vehicle submarí cal tenir en compte diferents aspectes. En primer lloc, cal vigilar que la tensió del mòdul LED seleccionat sigui corresponent a la tensió que es subministra des de la bateria, en aquest cas 12V. A continuació, cal seleccionar uns llums que funcionin correctament a sota l'aigua. Per aquest motiu, s'ha fet recerca de quins llums utilitzen diferents llanternes submarines. El resultat d'aquesta cerca ha estat que els LED XHP70.2 són els més adients per aquesta aplicació.

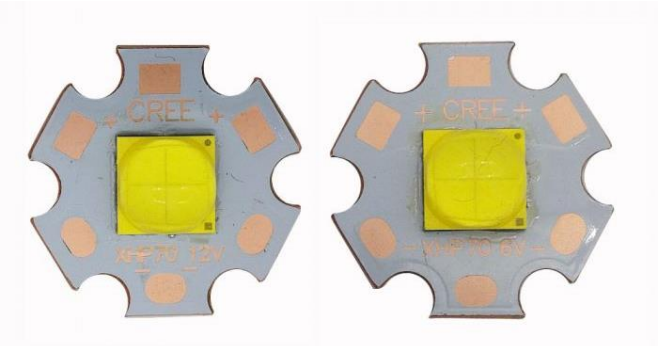

Figura 29. Mòdul LED XHP70.2.

El color de la llum és l'aspecte més diferencial dels LED que s'utilitzen per a funcionar sota l'aigua. Cal que la llum sigui de 6500K, anomenada blanc fred, que és la utilitzada per tots els ROV i AUV, ja que és la que aporta més visibilitat sota l'aigua. La potència també és un aspecte important, aquests mòduls tenen una potència de 12W més que suficient per a la profunditat a la qual es pretén arribar. Aquest mòdul té una relació de 100 lúmens/W, per tant, s'obtenen 1200 lúmens.

Cal tenir en compte que aquests LED produeixen bastant calor i es troben dins un mòdul tancat de material acrílic on la circulació d'aire és casi nul·la, que tan sols es refrigera amb l'aigua exterior. Per aquest motiu, és molt important situar el mòdul LED enganxat a la peça que aguanta les juntes tòriques, ja que aquesta és d'alumini, un bon conductor tèrmic, que està en contacte amb l'aigua exterior. S'utilitza aquesta peça per a dissipar la calor que produeixen els LED, juntament amb una altra peça d'alumini que fa de suport del mòdul millorant el contacte entre ambdós amb una mica de pasta tèrmica. El cable que s'escull per a aquest mòdul és una mànega de 2x1,5 mm^2, per reduir la caiguda de tensió.

3.8. Controlador PWM per LED

Per a controlar la potència que es subministra als dos mòduls LED utilitzats, un a cada encapsulat superior, s'utilitza un mòdul de control PWM. El mòdul de control en qüestió també permet una tensió de 12V i utilitza la tecnologia MOSFET per a realitzar el control.

Aquest dispositiu és capaç de subministrar fins a 15 A, encara que en aquest cas només es necessitarà menys d'un amper per cadascun dels dos mòduls LED. Aquesta intensitat es regula amb el senyal PWM que la Raspberry passa a aquest dispositiu, de 3,3V. Per aquests motius, s'escull aquest controlador pel projecte.

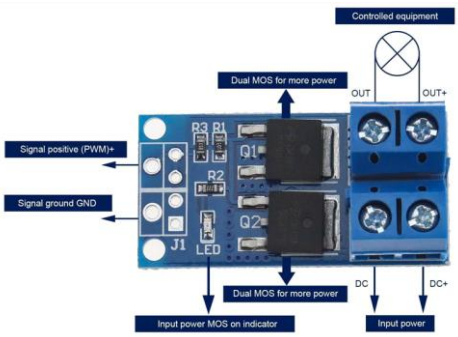

Figura 30. Mòdul de control PWM que s'utilitza per alimentar els LED.

3.9. Giroscopi i acceleròmetre MPU6050

Aquest sensor anomenat MPU6050 consisteix en un acceleròmetre i giroscopi de tres eixos, que es posaran en la mateixa direcció que els eixos del ROV per poder avaluar els moviments d'aquest. Els valors més interessants que s'obtindran d'aquest sensor són el roll, el pitch i el yaw, és a dir, la rotació del robot.

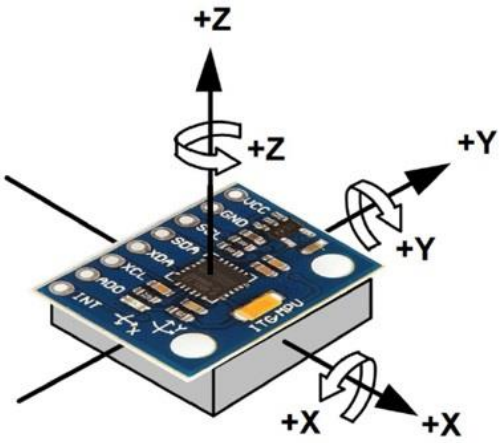

Figura 31. Giroscopi i acceleròmetre MPU6050 amb els eixos que avalua.

Aquest sensor es comunica per I^2C, igual que l'anterior sensor de profunditat BAR30. S'alimenta a 3,3V, la seva adreça és 0x68 i el seu consum de 50 µA. Les resistències internes de pull up són de 4k7 ohms, com es pot observar a l'esquema de la següent figura. Aquestes dades s'han de tenir en compte a l'hora de fer les connexions.

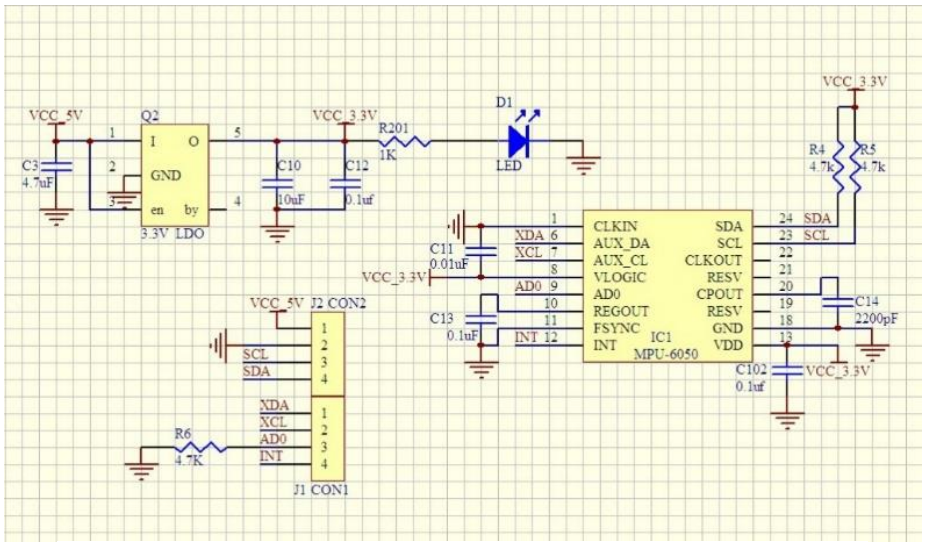

Figura 32. Esquema del giroscopi i acceleròmetre MPU6050.

S'escull aquest sensor, ja que es comunica amb el mateix protocol que la resta de sensors, actuant com a esclau i la Raspberry de mestre. El seu consum és baix i l'adreça

no coincideix amb cap dels altres dispositius connectats al bus. Les resistències de pull up són prou grans per poder fer el paral·lel amb les dels altres sensors connectats al mateix bus, sinó sempre hi ha l'opció de dessoldar-les. Amb els valors obtinguts d'aquest sensor es podrà avaluar l'estabilitat del ROV.

3.10. Brúixola HMC5883L

El sensor HMC5883L consisteix en una brúixola digital, basant-se en els camps magnètics que crea la Terra. Aquest tornarà el nombre de graus de desviament que tenim respecte al nord magnètic, sent 0 el nord i 180 el sud. Caldrà doncs vigilar amb aquest sensor, ja que els camps magnètics creats pels motors o altres components poden desviar les seves lectures. S'haurà de posar al màxim aïllat dels altres components.

Figura 33. Brúixola HMC5883L.

Per a la transmissió de dades, utilitza la transmissió per bus I2C, com els altres sensors. S'alimenta a 3,3V, la seva adreça és 0x1E i el seu consum de 100 µA. Les resistències internes de pull up són de 10 quilo ohms, com es pot observar a l'esquema de la següent figura. Aquestes dades s'han de tenir en compte a l'hora de fer les connexions.

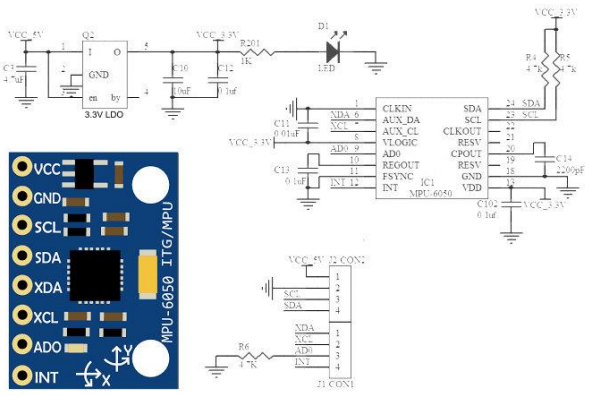

Figura 34. Esquema de la brúixola HMC5883L.

S'escull aquest sensor, ja que es comunica amb el mateix protocol que la resta de sensors, actuant com a esclau i la Raspberry de mestre. El seu consum és baix i l'adreça no coincideix amb cap dels altres dispositius del bus. Les resistències de pull up són prou grans per poder fer el paral·lel amb les dels altres sensors connectats al mateix bus, sinó sempre hi ha l'opció de dessoldar-les. Amb els valors obtinguts d'aquest sensor es podrà avaluar la direcció en la qual està apuntant el Girona 25, quan s'estigui utilitzant i la visibilitat de fora l'aigua sigui reduïda.

3.11. Sensor de consum INA219

El sensor de consum que s'utilitza per a mesurar el corrent instantani i el nivell de voltatge de la bateria per saber la càrrega d'aquesta. Per fer-ho aquest sensor es basa en la caiguda de tensió en una resistència shunt que porta incorporada.

Figura 35. Sensor de consum INA219.

El problema d'aquest sensor és que, encara que pot mesurar voltatges entre 0 i 26V, el corrent màxim que admet la resistència shunt de 0,1 ohms és de 3,2A. És per aquest motiu, que es s'ha de dessoldar aquesta resistència i connectar als terminals d'entrada directament la resistència shunt desitjada, en aquest de 50A i 75mV. Caldrà utilitzar aquests nous valors en el codi i no els que porta per defecte el programa per llegir les dades d'aquest sensor.

Figura 36. Resistència shunt 50A 75 mV.

Per a la transmissió de dades, utilitza la transmissió per bus I2C, com els altres sensors. S'alimenta a 3,3V, la seva adreça és 0x40 i el seu consum de 1 mA. Les resistències internes de pull up són de 10k ohms. Aquestes dades s'han de tenir en compte a l'hora de fer les connexions.

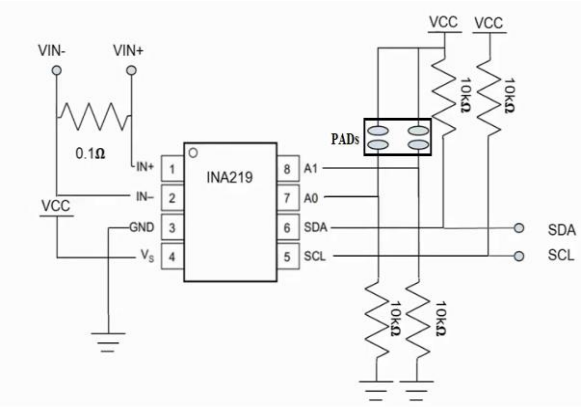

Figura 37. Esquema del sensor de consum INA219.

S'escull aquest sensor, ja que es comunica amb el mateix protocol que la resta de sensors, actuant com a esclau i la Raspberry de mestre. El seu consum és baix i l'adreça no coincideix amb cap dels altres dispositius del bus. Les resistències de pull up són prou grans per poder fer el paral·lel amb les dels altres sensors connectats al mateix bus, sinó sempre hi ha l'opció de dessoldar-les. Amb els valors obtinguts d'aquest sensor es pot saber la càrrega de la bateria, el corrent i la potència que està consumint en cada instant el Girona 25.

3.12. Bateria LiPo 11,1V

El ROV està alimentat des de l'interior d'aquest mateix, dins l'encapsulat inferior, per aquest motiu s'escull una bateria LiPo, feta de polímer de liti, ja que és una bateria molt compacta amb una gran densitat energètica molt utilitzada en projectes RC. Aquest tipus de bateria pesen menys, i la velocitat de descàrrega és molt més elevada. L'inconvenient és que són bastant sensibles, sobretot cal vigilar la temperatura, intentant deixar un temps des de la utilització fins quan es carrega amb el sistema per carregar la bateria sense obrir els encapsulats. En tot cas, es farà servir la temperatura que dona, per exemple, el sensor MPU6050 o la mateixa Raspberry, per vigilar que la temperatura dins l'encapsulat no sigui perillosa, ja que sinó també es pot produir el fenomen "thermal throttling", que reduiria la potència del processador, que també s'intenta reduir amb el dissipador i ventilador per la Raspberry.

La bateria és de 3S, és a dir, tres cel·les, que equival a un voltatge nominal de 11,1V, ja que cada cel·la és de 3,7V. Cal tenir en compte que el voltatge d'aquesta bateria pot arribar a 12,6V quan està totalment carregada, ja que aquest tipus de bateria té un perfil de voltatge bastant ampli. Cal vigilar que totes les cel·les tinguin la mateixa tensió, per això es carreguen amb carregadors que balancegen les cel·les, obtenint informació del connector petit que tenen aquestes bateries. És recomanable no baixar de la tensió nominal per cel·la, però si la tensió disminueix per sota els 3V, la cel·la deixa de ser estable i és perillós.

La capacitat de la bateria s'ha escollit en funció de les dimensions d'aquesta, el consum, principalment dels motors, i la relació pes energia respecte el Girona 500. El primer aspecte que cal tenir en compte és que la bateria i la resta de components puguin cabre a dins dels encapsulats, és a dir, cal seleccionar la bateria tenint en compte el disseny del mòdul de l'electrònica, ja que és l'element que més ocupa. També cal vigilar el pes d'aquesta a l'hora d'aconseguir la flotabilitat nul·la i l'estabilitat.

El consum és el que realment marca la capacitat que ha de tenir la bateria, tenint en compte les hores que es vol que duri aquesta. Per calcular el consum dels motors, es té en compte que a mitja càrrega consumeixen 1,5 A cadascun i que normalment estaran funcionant 3 motors a la vegada, per tant, el consum aproximat d'aquests és de 4,5 A. Els llums, el següent element amb més consum, es preveu com a molt un consum de mitja càrrega, és a dir, de 0,5 A per mòdul LED, en conseqüència, un consum d'1 A. LA Raspberry i la resta de sensors i elements de control tenen un consum molt baix, es considera d'1 A, com a molt. En conclusió, es preveu un consum mitjà de 5,5 A i la capacitat escollida és de 22Ah, per tant la bateria té una duració aproximada de 4 hores.

En relació al Girona 500, aquest AUV té un pes de 150kg i la capacitat de la bateria és de 2,9kWh, la seva relació pes energia és de 19,3 Wh/kg. Mentre que el Girona 25 té un pes de 11kg i la capacitat de la bateria és de 244Wh, obtenint una relació pes energia de 22,2 Wh/kg, millor que la del seu antecessor. Aquesta relació pes energia no és determinant, ja que el consum també està relacionat amb els dispositius que consumeixen de cada robot, encara que en el Girona 500 són més que en aquest ROV.

Figura 38. Bateria LiPo 11,1V 22000mAh 25C.

La capacitat C en aquesta bateria és de 22000mAh, com ja s'ha comentat, i la capacitat màxima de descàrrega és de 25C, és a dir, la bateria pot subministrar teòricament fins a 550 A. Per tant, la bateria pot subministrar prou corrent, encara que es volgués fer servir els motors a màxima potència o cobrir pics de consum quan s'engeguen els motors. Normalment es càrrega la bateria a una velocitat de càrrega de 1C.

3.13. Regulador de tensió 5V

Aquest robot a sota l'aigua s'alimenta només per la bateria anteriorment esmentada, però no tots els components funcionen a 11,1V. És per aquest motiu, que es necessita un regulador de tensió de 5V, per alimentar elements com la Raspberry que funciona amb aquest voltatge i no porta regulador de tensió intern.

Figura 39. Regulador de tensió 5V 6A de Bluerobotics.

S'ha seleccionat aquest regulador de tensió, ja que és el que es fa servir en molts ROV i es sap que funciona correctament. En projectes anteriors s'havien detectat problemes en fer servir reguladors de tensió, ja que, quan la Raspberry s'inicia, té pics de corrent de fins a 3 A. Aquest regulador subministra fins a 6 A amb un rang d'entrada de voltatge de 7 a 26 V, és a dir, per a bateries de 2S a 6S. L'empresa que el subministra és, també, Bluerobotics.

3.14. Fusible 50A

El fusible, amb el corresponent porta fusibles, és un element de protecció que s'utilitza es cas que es produís algun pic de corrent que pogués fondre els cables o cremar alguns components. El valor de 50 A s'ha seleccionat tenint en compte els consums esmentats, els pics que es produeixen quan s'engeguen els motors i deixar un marge per no estar contínuament fonent aquest component, però vigilant de no fondre els altres ni cremar cap cable. Els cables 10 AWG aguanten fins a un corrent de 55 A, per tant, amb el fusible

de 50 A quedaran correctament protegits. Per falta d'espai, no s'han posat fusibles a cadascun dels drivers dels motors per evitar que es fonguin en cas de curtcircuit o qualsevol altre problema.

Figura 40. Fusible de 50 A amb el corresponent porta fusibles.

Cal tenir en compte que si es fon el fusible el robot deixa de funcionar, és a dir, que si aquest fet succeeix a sota l'aigua es pot perdre el robot durant un cert temps, ja que com que la flotabilitat és una mica positiva acabaria sortint una altra vegada a la superfície. També es podria arribar a estirar del cable si es pot accedir a la boia.

3.15. Interruptors per engegar i apagar

Per tal de no haver d'obrir l'encapsulat del robot cada vegada que es vol engegar o apagar, s'ha dissenyat un sistema basat en interruptors magnètics i un relé. Aquest sistema és molt semblant al que incorporen l'Sparus II i el Girona 500.

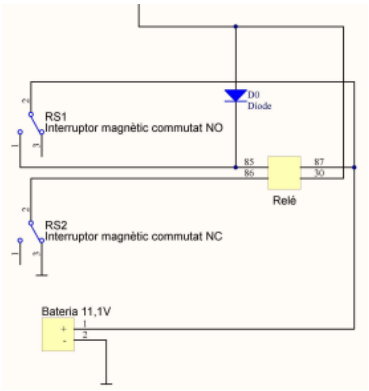

Figura 41. Esquema electrònic del sistema d'alimentació.

El circuit es basa en l'esquema de la figura anterior. És un circuit bàsic de marxa i paro, amb dos interruptors magnètics commutats, un connectat com un contacte normalment obert i l'altre normalment tancat, que s'activaran des de l'exterior de l'encapsulat amb imans. Així doncs, quan s'activi l'interruptor magnètic RS1, es donarà tensió a la bobina del relé i l'interruptor es tancarà deixant circular intensitat pel circuit. La realimentació permet que el robot no hagi de dur l'iman a sobre durant el seu funcionament, però cal

posar un díode per evitar que en el moment d'arrencar, el corrent no es condueixi a través del cable que fa la realimentació, podent cremar-lo. Quan es vulgui apagar el robot, s'ha de posar l'iman prop de l'interruptor normalment tancat, d'aquesta manera s'obrirà el circuit que alimenta la bobina, l'interruptor del relé s'obrirà i no es tornarà a activar perquè ja no hi haurà la realimentació activa.

Figura 42. Relé instal·lat amb al resta de l'electrònica.

El relé escollit és el v23134-b1052-c642, un relé de potència bastant utilitzat en l'automoció. Aquest relé permet treballar a 12V en contínua i fins a una intensitat màxima de 60 A. Cal tenir en compte que la potència necessària per activar el relé és de 1,6W, que aproximadament a 12V, per tant, pels interruptors magnètics passarà un corrent de 133,33 mA. Els interruptors magnètics escollits aguanten fins a 1A i 20W, el sistema és més que suficient.

Cal tenir en compte que de la manera que s'ha muntat el robot, l'interruptor amb cable de color verd és el normalment obert, que quan s'activa engega el robot i l'interruptor amb cable de color blau és el normalment tancat, que quan s'activa apaga el robot.

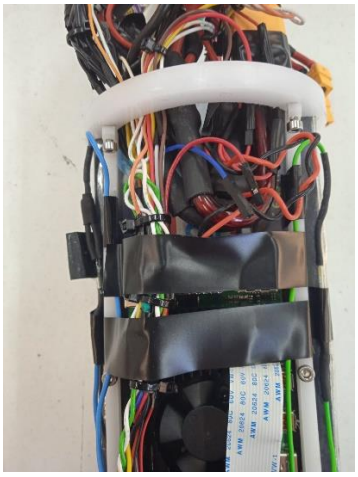

Figura 43. Situació dels interruptors magnètics.

3.16. Sistema per carregar la bateria

El sistema per carregar dissenyat de manera que no s'hagi d'obrir l'encapsulat és possible gràcies al connector subconn. El connector del balanceig, de 4 pins a causa de les 3 cel·les, i un connector paral·lel a la sortida de potència, de 2 pins, positiu i negatiu, de la bateria es connecten als 6 pins que té el connector subconn. De manera que quan es vol carregar, es connecta un cable a la sortida fet expressament per portar cadascun dels pins de manera correcta i ordenada fins al carregador de bateries i carregar-la. Quan es vol utilitzar el robot o ja no es vol carregar, es pot desconnectar aquest cable i connectar el corresponent tap.

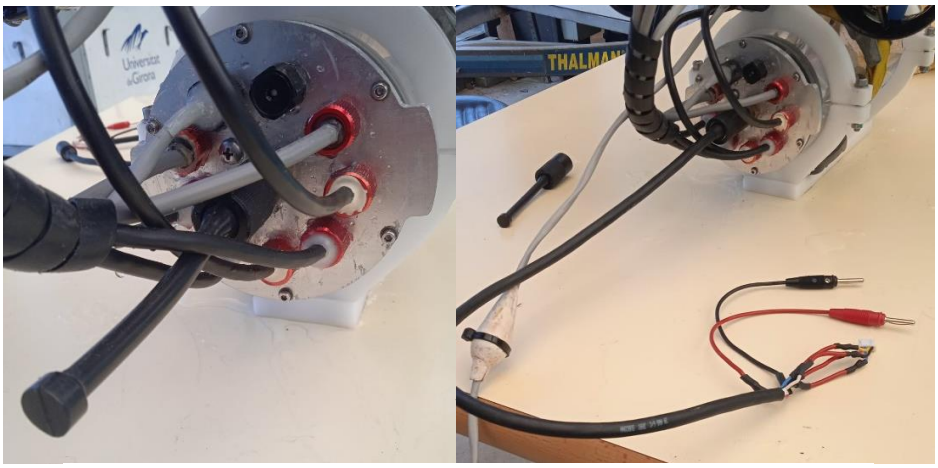

Figura 44. Canvi del tap pel cable per carregar la bateria.

Cal tenir en compte que encara que la capacitat de la bateria és de 22 Ah, el connector subconn només permet carregar 5 A per contacte amb un màxim de 20 A per connector. Tot i això, sempre és millor carregar la bateria a baixa potència per tal de cuidar-la i mantenir les seves prestacions el màxim temps possible.

El carregador utilitzat per carregar la bateria del robot és el Ultimate 1000W que permet carregar tota mena de bateries. Per iniciar el procés de càrrega, es connecten cadascun dels connectors en el corresponent lloc, es selecciona una bateria de tipus LiPo, per la càrrega balancejant les cel·les i una intensitat entre 3 i 5 A.

Figura 45. Carregador de bateries Ultimate 1000W.

Per alimentar aquest carregador, no es fa directament de la xarxa, sinó des d'una font de contínua Expert Power Supply 40A. Amb una tensió de 13,8 V i un corrent màxim de 40 A s'alimenta el carregador, per a complir la seva funció.

Figura 46. Font d'alimentació per alimentar el carregador de bateries.

3.17. Cable Ethernet

Sota l'aigua, les ones electromagnètiques s'atenuen ràpidament, per aquest motiu, la manera més econòmica i fiable de comunicar-se amb el robot submergible és mitjançant l'ús de cables. Es podrien utilitzar altres sistemes com els mòdems acústics. L'inconvenient del sistema escollit, és que la capacitat d'immersió del robot depengui la longitud del cable de comunicació.

El cable escollit per al ROV és el cable de xarxa Ethernet CAT5E, que es connecta a l'entrada Ethernet que té la Raspberry i a la boia de la superfície on es troba el router inalàmbric. També hi ha la possibilitat de connectar aquest cable directament a l'ordinador, però llavors la distància queda reduïda i per connectar el mòbil s'ha de fer a través del WiFi d'aquest. La distància de cable utilitzada és de 28 metres, però es podria arribar a utilitzar un cable de fins a 100 metres.

Figura 47. Cable CAT5E de fins a 100 metres.

Cal tenir en compte que aquest cable podria arrossegar el robot cap enrere, ja que és allà on està connectat, per aquest motiu es solen posar mòduls de flotació enganxats a aquest per evitar-ho. Igual que també s'utilitza una peça anomenada "Tether cable thimble" on s'enrotlla el cable, per evitar que les estrebades que es puguin produir en aquest, repercuteixin directament en el connector.

Figura 48. Utilització del "Tether cable thimble" i el mòdul de flotació en el cable CAT5E del ROV.

3.18. Router inalàmbric

A la boia, que es troba a la superfície, per crear la xarxa WiFi, s'utilitza un router. Aquest router, necessita una entrada per un connector RJ45, per poder-hi el cable de xarxa del ROV. El router que s'utilitza en aquest projecte és el TP-Link TL-WR902AC. Aquest s'utilitzarà com a mode client, de manera que els diferents clients que es connectin a la xarxa es podran comunicar amb el senyal Ethernet que arriba de la Raspberry, dins del ROV, sota l'aigua. D'aquesta manera, qualsevol dispositiu amb connexió WiFi podrà interactuar amb el Girona 25.

Figura 49. Router inalàmbric TP-Link TL-WR902AC.

Aquest dispositiu es trobarà a la superfície dins d'una caixa estanca de material acrílic, conjuntament amb la bateria portàtil de 5V que l'alimenta, gràcies a les seves reduïdes dimensions. Aquest conjunt tindrà un mòdul de flotació per evitar que s'enfonsi i que s'atenuïn les ones del router. La potència de fins a 733 Mbps que té el dispositiu, permet connectar-se al ROV des de qualsevol dispositiu que estigui dins un rang d'uns 20 metres de la boia.

3.19. Bateria portàtil 5V

La bateria que s'utilitza per alimentar el router és de 5V amb una capacitat de 10400 mAh. El consum del router és de 1A, per tant aquesta bateria permet poder fer funcionar el router durant més de 10 hores, més que l'autonomia del ROV.

Figura 50. Bateria portàtil 5V 10400 mAh.

Aquesta bateria és capaç de donar fins a 3 A a un voltatge de 5-6 V, és a dir, fins a 18 W, gràcies a la tecnologia Quik Charge 3.0 i Power Delivery, més que suficient per a les prestacions que ha de donar. La connexió que es farà és servir és la de l'USB i es connectarà al port mini-USB del router, per alimentar-lo. Les dimensions són més grans que el router, però suficientment petites per cabre dins la caixa de 10x10 cm.

3.20. Connectors

Aquest ROV s'ha dissenyat perquè l'electrònica es pugui muntar i desmuntar fàcilment, per, per exemple, poder canviar algun component si és necessari. Això s'aconsegueix utilitzant els connectors semblants al que porta la mateixa bateria, anomenats XT90 i XT60. D'aquesta manera no s'ha d'estar soldant i dessoldant tants de cables quan algun component deixa de funcionar o es vol substituir.

La bateria és l'element que alimenta la resta de components, per aquest motiu s'utilizen connectors de bateria paral·lels XT90 per a poder arribar a tots els dispositius que s'han d'alimentar a aquesta tensió de 11,1V: els motors, el regulador de tensió i els llums. Aquests connectors ja porten un sistema de seguretat perquè a l'hora de fer les connexions no es connecti el negatiu amb el positiu i viceversa. Poden aguantar un

corrent de 90 A i una tensió de 12 a 24 V en continua, més que suficient per a la funció que han de realitzar. Cal tenir en compte que s'hauran de treure els connectors que venen de sèrie amb els dispositius i canviar-los per aquests.

Figura 51. Connectors paral·lels XT90.

Per a la connexió dels controladors amb els corresponents motors, com que hi ha 3 fases, 3 cables, s'utilitzaran uns connectors molt semblants als anteriors, que es basen en el mateix tipus de funcionament, anomenats MR30, que ja són especials per a aquest tipus de connexions. D'aquesta manera es podrà separar fàcilment l'electrònica dels motors i no caldrà desconnectar també els controladors. Aquests connectors aguanten corrents de fins a 15 A i 500V en continua, més que suficients per a la càrrega prevista.

Figura 52. Connectors MR30 per a connectar motors i controladors.

Pel cable d'Ethernet, s'utilitzarà el connector habitual RJ45, que també és molt fàcil de connectar i desconnectar. Comentar que a diferència dels altres, aquest connector no es solda, sinó que s'utilitzen unes alicates especials per a premsar-los quan els cables estan col·locats correctament.

3.21. Cables

El cable escollit per a realitzar les connexions, principalment la dels llums i la dels propulsors, és AWG 10, ja que és el mateix que porta la bateria i el que correspon amb els connectors XT90. Aquest cable és de 5,26 mm^2 de secció i aguanta un amperatge de 52 A, per aquest motiu s'escull el fusible de 50 A per a evitar que es fongui el cable en cas que es produeixi un pic d'intensitat sobtat. També es redueix la caiguda de tensió en els cables, gràcies a la secció d'aquests.

Els cables que s'utilitzen per a connectar els diferents sensors són els anomenats cables Dupont, que ja porten el seu propi connector. Com els anteriors, són connectors amb els que les connexions es realitzen de forma fàcil i ràpida.

Figura 53. Cables Dupont amb els corresponents connectors incorporats.

3.22. Anàlisi electrònica

A l'hora de seleccionar els components i triar allà on aniran connectats cal tenir en compte les especificacions de cadascun dels components que s'han anat comentat en aquest capítol. Alguns aspectes, com la duració de les bateries, ja s'han comentat en el corresponent apartat.

Els sensors tenen un consum i s'alimenten tots a través del pin de 3,3 V de la Raspberry. El corrent que pot donar aquest pin és de més de 500 mA i el consum dels sensors no supera els 5 mA, per tant no hi ha problema, el disseny és correcte.

També cal tenir en compte és quins sensors es connecten a cada bus I^2C i quants busos d'aquest tipus es fan servir. Es fan servir dos busos, ja que, si es vol evitar dessoldar

resistències de pull up, és millor no tenir més de 3 sensors en un sol bus. A més s'utilitza un sol bus pel sensor BAR30, ja que és el més sensible, no es poden dessoldar les resistències de pull up i el fabricant no proporciona gaire informació relativa a l'adreça del sensor. Els altres tres sensors tenen adreces diferents i el paral·lel de les resistències de pull up és suficient per al correcte funcionament del bus.

Un altre aspecte a considerar, és on es situa la resistència shunt del sensor de consum. Aquesta es podria situar abans de la càrrega o després de la càrrega, en aquest cas s'ha optat per situar-la a continuació del fusible, és a dir, abans de la càrrega, ja que no interessa que tota la càrrega quedi sobre un petit nivell de voltatge que produeix aquesta resistència. Es prefereix que el voltatge sigui fins a 75 mV més baix, ja que la tensió de la bateria ja és bastant variable i aquesta diferència no serà significativa.

Per últim, escollir a quins pins de la Raspberry es connecta cada entrada o sortida d'aquesta, ja que no tots són iguals ni poden fer el mateix. Aquest tema es soluciona amb la llibreria que es descriu en el següent capítol, que permet utilitzar qualsevol GPIO com a sortida de senyal PWM, que és el problema més destacat. La utilització d'un segon bus I^2C implica buscar dos pins més especials per a aquesta funció, dels quals l'ordinador de control ja disposa.

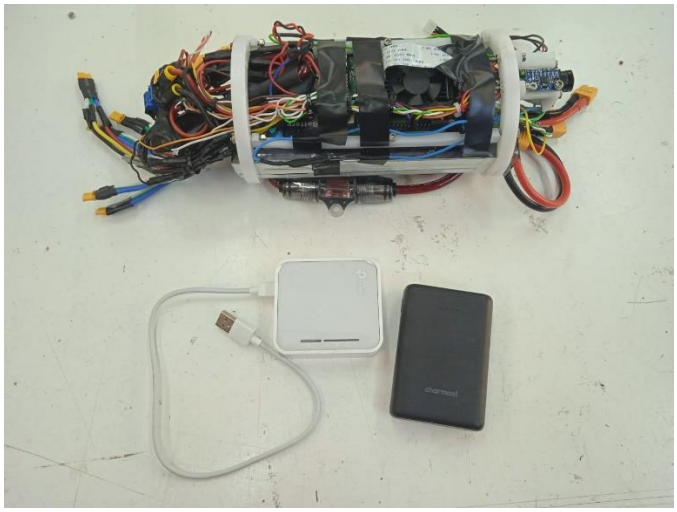

Figura 54. Tota l'electrònica utilitzada en el ROV Girona 25.

4. PROGRAMARI I COMUNICACIONS

Per a controlar el robot es fan servir diferents llenguatges i interfícies, programades amb diferents entorns, per a utilitzar-los en diferents components com poden ser mòbils i ordinadors. També es realitzen comunicacions per mitjà d'Ethernet i WiFi amb els components descrits anteriorment, que cal definir.

Cal comentar que la programació del Girona 25 pren com a base la dels anteriors projectes, millorant-la i afegint les noves funcionalitats corresponents.

4.1. Comunicacions

Les comunicacions són l'aspecte diferencial del robot submarí, ja que si no es dissenyen de forma correcta no es podrà controlar el Girona 25. Cal tenir en compte que l'aigua atenua les ones i que, sota l'aigua, cal que les comunicacions siguin per cable. És quan aquest cable, que cal seleccionar amb cura perquè tingui la longitud desitjada, arriba a la superfície on es poden implementar les comunicacions sense fils.

4.1.1. MQTT Protocol

MQTT, inicials de Message Queuing Telemetry Transport, és un protocol de missatgeria publicació/subscripció, molt lleuger, obert, simple i dissenyat per ser fàcil d'implementar. Aquestes característiques el fan ideal per una gran varietat de situacions, incloent entorns limitats com pot ser en la comunicació màquina a màquina (M2M) i Internet de les coses (IoT).

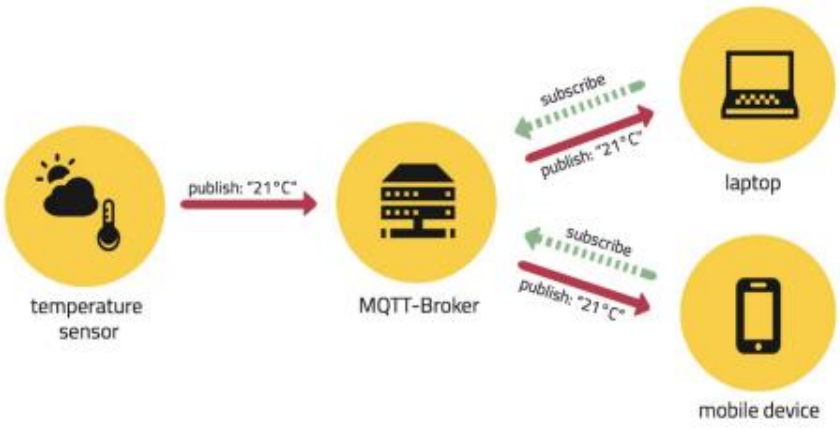

Figura 55. Representació gràfica del funcionament del protocol MQTT.

L'arquitectura del protocol està formada per dos dispositius. Un és el broker o servidor central al qual es connecten els clients, s'encarrega de filtrar i distribuir els missatges publicats entre els subscriptors d'acord amb els tòpics als quals s'ha subscrit. El segon són els ja esmentats clients, que poden publicar missatges a un tòpic i alhora poden subscriure's a un tòpic i així rebre missatges publicats en aquell tòpic.

En aquest projecte s'utilitza aquest protocol de comunicació per a comunicar el ROV amb els dispositius de la superfície. Es pot utilitzar de broker la Raspberry o també un dispositiu que estigui executant el Mosquitto MQTT Broker de Eclipse Foundation, com pot ser un ordinador.

4.1.2. Ethernet i WiFi

Del robot surt un cable Ethernet provinent del connector de la Raspberry de manera que comparteix totes les dades a través d'aquest cable fins a la superfície, connectant-se com un client a través del cable Ethernet. A la superfície, a través del router en mode client es comparteix aquesta connexió mitjançant la xarxa sense fils WiFi, funció per a la qual ha estat dissenyat aquest dispositiu. Qualsevol dispositiu amb aquest tipus de connexió es pot connectar a aquest client, és a dir, amb el Girona 25.

Per seguir unes pautes i no tenir problemes amb les comunicacions, cal deixar un temps perquè la Raspberry i el router s'inicialitzin i el router assigni l'adreça IP 192.168.0.100 al broker, és a dir, a la Raspberry. A La resta de dispositius el router els hi assignarà una IP automàticament, sempre evitant que dos dispositius no tinguin la mateixa adreça.

Si es connecta la Raspberry al router, cal connectar el dispositiu que es vulgui fer servir per controlar el ROV a aquest WiFi amb el SSID i la contrasenya predeterminats, que es troben a l'etiqueta del router. A continuació, almenys la primera vegada, cal anar al navegador i entrar a la següent web: http://tplinkwifi.net, per configurar-lo. S'ha de configurar com a mode client, la resta es recomana deixar la configuració predeterminada. D'aquesta manera s'aconsegueix connectar el dispositiu o dispositius i el robot per mitjà d'una xarxa sense fils.

Si es connecta la Raspberry directament a un ordinador per Ethernet, és aconsellable que el broker sigui l'ordinador, ja que no es podrà connectar cap altre dispositiu al ROV. Si es vol connectar algun altre dispositiu, llavors cal crear un servidor, com pot ser

Apache24, per compartir la xarxa i la interfície web, i un punt d'accés WiFi des d'aquest ordinador, al que s'hauran de connectar els altres dispositius. Cal comentar que el mateix robot comparteix la interfície web amb aquest programa i hi pot accedir qualsevol dispositiu connectat a la mateixa xarxa escrivint la IP del ROV al navegador.

El comandament Logitech Gamepad F310 s'utilitza per a comandar el robot, encara que es pot fer directament des de l'ordinador o des del mòbil, on apareix en pantalla una palanca de control virtual i diferents polsadors per a controlar-lo. Aquest dispositiu es connecta a l'ordinador per USB, encara que també es podrien utilitzar altres comandaments com el de la Play Station 4, que es pot connectar per Bluetooth.

Figura 56. Comandament Logitech Gamepad F310.

Per a la utilització de les ulleres de realitat virtual és necessari un telèfon mòbil adequat per a poder utilitzar amb unes ulleres d'aquest tipus. Per tant, serà necessari connectar aquest dispositiu amb el ROV i accedir a la interfície web. Aquest sistema està pensat per utilitzar el comandament per a controlar el ROV i les ulleres per a visualitzar les imatges captades per la càmera. D'aquesta manera s'aconsegueix crear una experiència similar a la que té un submarinista, aconseguint arribar a una profunditat semblant i travessant les barreres físiques que a algunes persones els impedeix realitzar aquest esport. També s'utilitza el giroscopi del telèfon mòbil per a comandar el ROV, és a dir, el Girona 25 es mou sobre ell mateix igual que el cap de la persona que porta les ulleres de realitat virtual, fent més realista l'experiència.

Figura 57. Exemple d'utilització d'ulleres de realitat virtual amb un comandament.

4.2. Raspberry ROV

La Raspberry és l'ordinador encarregat de dirigir el ROV, per tant, el programa de comandament d'aquest haurà d'executar-se des d'aquesta computadora. Per a dissenyar aquests programes s'utilitza un dels llenguatges de programació oficials per programar aquest tipus de dispositius, Python.

Figura 58. Logotip del llenguatge de programació Python.

Abans de començar la programació cal tenir clar quina llibreria s'utilitza per controlar els pins de la Raspberry. En aquest projecte s'utilitza la llibreria GPIO Zero, ja que permet controlar aquests de manera fàcil i intuïtiva. Un dels avantatges d'aquesta llibreria resideix en el fet que internament empra llibreries que ja existien per programar els pins com la RPi.GPIO, una de les més comunes, i la PiGPIO, idònia pel control PWM des de qualsevol GPIO pin.

Un altre aspecte important a tenir en compte és com es farà el comportament autònom, ja que també és necessària una llibreria per a poder-lo desenvolupar de forma òptima. La llibreria que s'utilitza és OpenCV, una llibreria de visió per computador de codi obert desenvolupada per la companyia Intel i llançada per primer cop l'any 1999. Aquesta biblioteca conté més de 2500 algoritmes, que poden ser utilitzats per detecció i reconeixement facial, identificació d'objectes, rastrejar objectes en moviment, extreure models 3D d'objectes, etc.

Per a controlar el ROV, es fa servir un protocol de missatgeria anomenat MQTT, que s'explica en el corresponent apartat. Per a fer-lo servir, és molt convenient utilitzar la llibreria Eclipse Paho MQTT Python, o Paho MQTT. Es tracta d'un projecte de Eclipse Foundation, que implementa les versions 5.0, 3.1.1, i 3.1 del protocol MQTT. Aquesta biblioteca proveeix una classe client, la qual permet que les aplicacions puguin connectar-se a un servidor MQTT i poder publicar missatges, subscriure's a tòpics i rebre els missatges publicats.

El programa es divideix en diferents subprogrames dirigits per un programa principal, que és el que es subscriu als tòpics per obtenir els missatges que s'hi envien.

4.2.1. Programa per executar el principal

Per aconseguir que quan s'engegui la Raspberry, sense donar cap ordre, s'executi el programa principal s'ha creat un "service", que executa un petit programa que el que fa és executar constantment el programa principal, de manera que si es produeix algun error en aquest, es torni a executar i poder continuar utilitzant el robot. S'utilitza aquest sistema, ja que el programa que crea el broker, del protocol MQTT, a la Raspberry funciona de la mateixa manera i s'aconsegueix que un programa pugui estar funcionant en un segon pla de manera automàtica des del moment en què s'engega l'ordinador de control.

4.2.2. Programa principal: MQTT client

Aquest programa és el principal, el que s'executa a la Raspberry i controla tots els moviments i sensors del ROV. Com ja s'ha dit, és el que es subscriu als tòpics per obtenir els missatges que s'hi envien i poder executar les ordres que es desitgen, des d'aquest programa també es poden enviar missatges als diferents tòpics.

El primer que fa és connectar-se al broker, la IP del qual està definida en el programa 'config.py', com a client, retornant el valor a la variable "rc". Un cop connectat passa a analitzar els tòpics i els continguts de cadascun, per determinar quines accions són les que s'han de fer. Aquest programa no té un procés seqüencial, es treballa amb diferents tòpics i es realitzen accions a partir dels missatges que s'hi envien.

Un d'aquests tòpics és el de "Motors", que com el seu nom indica, és on s'envien els missatges del que es vol que facin els propulsors del Girona 25. S'envia el missatge indicant cap a quina direcció es vol que vagi el ROV: up, down, left... juntament amb la consigna de la velocitat a la qual s'han de moure els motors.

Un altre tòpic és el de "Camera". En aquest tòpic és el client, la Raspberry, qui penja missatges. El que fa és executar el programa que es comentarà a continuació anomenat "take_photos.py" i publicarà en aquest tòpic "fotografia feta", missatge que processarà la interfície web.

El següent tòpic que es processa és el "Stream". Quan rep el missatge "play", executa el programa, que també es comentarà a continuació, "streaming.py" i enviarà el missatge corresponent a que el programa s'està executant. Si es rep el missatge de "pausa", es tancarà el programa que s'estava executant i s'enviarà el missatge corresponent a aquesta acció.

En el tòpic "Light" depenent de si es rep el missatge "on" o "off" s'executarà una classe que farà la corresponent acció, és a dir, encendrà o apagarà els llums. També enviarà un missatge al tòpic conforme ha realitzat la corresponent acció.

L'altre tòpic que s'utilitza és el de "Sensors". En aquest, és el ROV com a client qui recull les dades dels diferents sensors i les envia en format de missatge en el tòpic, en l'ordre corresponent per poder ser interpretades per la interfície web.

Per últim, el tòpic "Follow" depenent del missatge que es rep s'activa el mode de seguiment, és a dir, el comportament autònom de seguiment d'objectes, o es surt d'aquest mode.

4.2.3. Processament de les ordres dels motors

Aquest programa processa les dades obtingudes en el programa anterior, només les que es refereixen als motors, i envia les ordres als propulsors que han d'actuar en cada cas. Aquest programa s'anomena "orders.py".

Depenent del missatge i la consigna rebuda, aquest programa crea les classes que determinen en quins motors actuen en funció del missatge rebut, en quin sentit s'han de moure i a quina velocitat. També és capaç d'aturar aquests propulsors.

4.2.4. Funcionament dels motors

En el programa "motor.py" es determina com funciona un propulsor, és a dir, quin senyal PWM cal enviar al controlador perquè compleixi les ordres que es desitja que compleixi el ROV.

Per aquest programa, és importar tenir clar el funcionament d'aquests motors. Cal tenir en compte que la senyal PWM es pot enviar per qualsevol GPIO gràcies a la llibreria utilitzada, utilitzant la funció "PWMOutputDevice", i que abans de poder passar ordres als motors, cal inicialitzar-los, és a dir, enviar una senyal PWM de 1500 microsegons, que equival al senyal de stop.

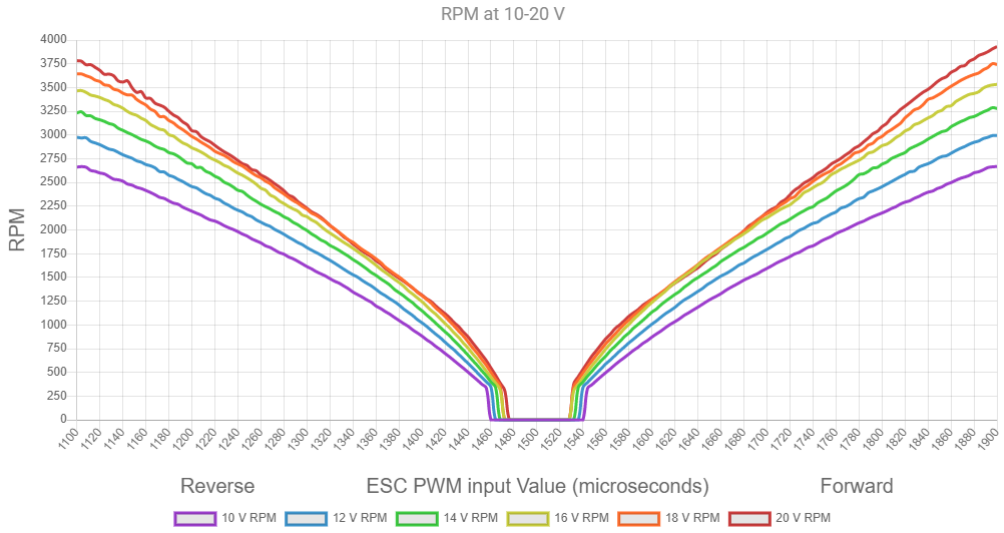

Figura 59. Velocitat dels propulsors en funció de la senyal PWM.

4.2.5. Vídeo captat per la càmera

Quan des del programa principal es rep el missatge de "play" en el tòpic "Stream", s'executa aquest programa anomenat "streaming.py". Aquest programa posa en marxa el programa mjpg-streamer, per iniciar la transmissió en directe del que veu la càmera del robot.

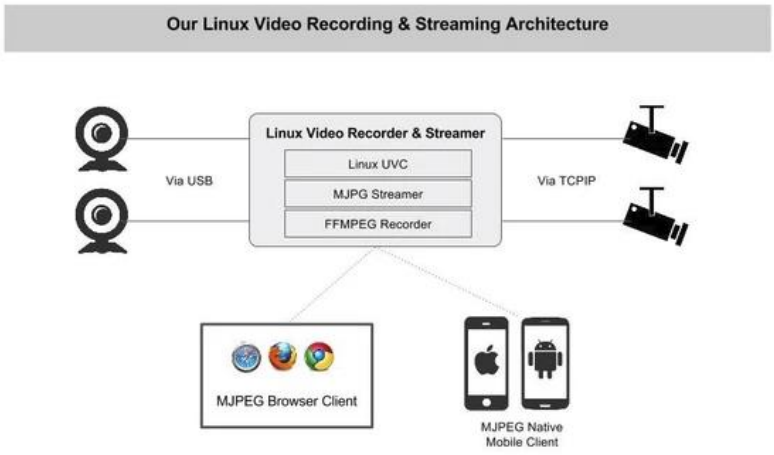

Figura 60. Principi de funcionament del software MJPEG.

Per a poder retransmetre el vídeo que capta la càmera, s'utilitza un programa de codi lliure anomenat mjpg-streamer. Aquest programa el que fa és copiar els fotogrames JPEG (Joint Photographic Experts Group) d'una o més entrades donades, com pot ser la càmera de la Rapsberry, i els envia en temps reals mitjançant el protocol HTTP. Per visualitzar el vídeo, el software utilitzat ha de poder suportar el format MJPEG (Motion JPEG), com la majoria de dispositius. El fet que els fotogrames estiguin en format JPEG i el protocol per transmetre'l sigui HTTP, permet que aquest pugui ser visualitzat amb HTML sense problemes.

4.2.6. Capturar imatges

Per capturar imatges, com s'ha comentat anteriorment, s'utilitza aquest programa anomenat "take_photos.py". Quan s'executa aquest codi, captura una imatge i la guarda al directori /Photos en format ISO, dins la Raspberry.

4.2.7. Els llums

En el programa "light.py", com en el programa "motor.py", es defineix com funcionen els llums. En funció de les ordres que es reben del programa principal, s'executen unes accions, apagar o encendre els llums. Com en els motors, cal definir quin pin o pins de sortida s'utilitzen en el programa principal i importar la classe en aquest.

4.2.8. Dades dels sensors

Aquest conjunt de programes recullen les dades dels diferents sensors creant classes perquè des del fitxer principal es puguin cridar i obtenir les dades dels diferents aparells perquè puguin ser monitoritzades des de la interfície web.

El fitxer "hmc5883l.py" és l'encarregat de llegir les dades del sensor que dona el nom al fitxer, la brúixola. Aquest és l'encarregat d'establir la connexió amb el sensor amb la corresponent adreça, demanar-li dades i llegir-les, és a dir, utilitzar el protocol per obtenir les dades d'aquest sensor.

El fitxer "ina219.py" té la mateixa funció que l'anterior, però amb el sensor INA219. L'única diferència és que amb aquest cas, cal tenir en compte que la resistència shunt

no és la que va amb el sensor, sinó que s'ha modificat i és una altra. Cal canviar els paràmetres corresponents per poder obtenir els valors correctes de càrrega de la bateria.

El fitxer "mpu6050.py" també llegeix les dades d'aquest sensor utilitzant el bus I2C, però a més, llegeix les dades de temperatura que aporta el mateix integrat, obtenint així, la temperatura interior de l'encapsulat.

El fitxer "ms5837.py" és l'encarregat de llegir les dades del BAR30, que porta aquest sensor en forma de circuit integrat a l'interior. Cal tenir en compte que aquest sensor no està connectat al mateix bus que els altres tres, per tant, cal configurar-lo de forma correcta.

Per aquests programes són necessaris els busos I^2C, per tant cal tenir-los definits prèviament a la Raspberry que s'utilitzarà i permetre l'accés a aquests.

4.2.9. Comportament autònom

El comportament autònom d'aquest ROV, es basa en el seguiment d'objectes. Aquest comportament pot servir, per exemple, per a seguir submarinistes o altres robots com el Girona 500. El programa encarregat de fer aquest seguiment és el "tracking.py". Aquest programa detecta una regió en la imatge d'un color i mida determinats, i posiciona el robot perquè aquesta regió es mantingui al centre de l'imatge. Depenent de la mida de la regió, el Girona 25 és capaç de mantenir-se a una certa distància quan s'apropa i de seguir-la quan s'allunya.

Aquest seguiment es realitza per segmentació per color, és a dir, depenent del color de l'objecte. Per escollir el color, en comptes d'utilitzar el tradicional mètode RGB (Red, Green, Blue), s'utilitza el model HSV, inicials que es refereixen al matís (Hue), saturació (Saturation) i brillantor (Value). Això es deu al fet que els colors a sota l'aigua canvien i agafant tonalitats que són més fàcils de distingir amb aquest nou mètode.

Un cop definit el color que es vol seguir, en aquest cas es tria el groc, ja que és un color que sota l'aigua canvia poc i té bastanta saturació, es desenvolupa el codi escrit amb Python i la llibreria OpenCV donant ordres als motors depenent de la posició del centre

de l'objectiu respecte de la càmera i el numero de píxels d'aquest obejcte, basant-se en els contorns captats del color escollit.

La llibreria OpenCV té l'algorisme que s'ha d'aplicar inclòs, per tant, el codi és bastant lleuger i l'execució és ràpida, permetent al ROV seguir als objectes encara que es moguin a certa velocitat.

El programa 'tracking.py' primer analitza la imatge binària obtinguda de la implementació de la funció cv2.connectedComponents. A partir d'aquí, determina quina area tenen els contorns dels grups de píxels i els filtra a partir d'una mida establerta. A continuació, calcula el seu centre i en determina la posició x i y. Depenent d'aquests dos valors es pot determinar a quin sector es troba el centre de l'objecte i s'envien les ordres als motors del robot perquè el ROV realitzi el moviment desitjat.

Per a determinar el color que es vol buscar, segons el model HSV, el valor del matís (H) representa el color que es vol triar, el groc va de 15 a 45 graus. El valor de la saturació (S) representa el grau de blancor que es vol escollir, és a dir, com més alt sigui el valor tindrà més color i com més baix més decolorat, es selecciona un rang de 100 a 255. Per últim, el valor de brillantor (V) representa el rang de negror que es vol escollir, és a dir, com més alt sigui el valor tindrà més color i com més baix més fosc, un rang de 20 a 255. El color vermell, doncs, queda delimitat dins el rang HSV (15-45, 100-255, 20-255).

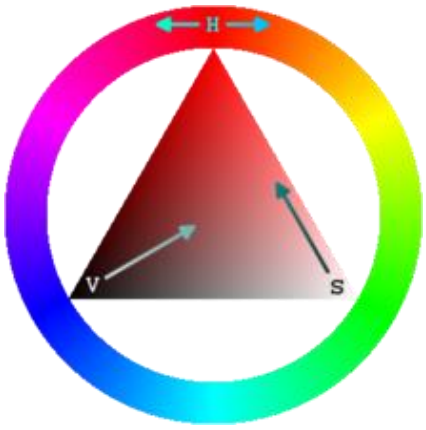

Figura 61. Model HSV usat per a determinar els colors.

4.3. Interfície web

Per tal de que es pugui controlar el Girona 25 des de qualsevol tipus de dispositiu, sense importar sistema operatiu ni altres especificacions, el control es realitza des d'una

interfície web. Cal tenir en compte que des de la interfície no es demana cap contrasenya ni identificació per connectar-se al robot, ja que el WiFi ja demana una contrasenya, si es configura d'aquesta manera. Per accedir-hi es pot fer a través dels documents que s'adjunten en el mateix treball o des de la següent pàgina web, que és un servidor que deixa penjar-hi la teva web i poder-hi accedir: https://girona25.000webhostapp.com/.

Aquesta interfície ha estat dissenyada amb el llenguatge de marcat HTML, de l'anglès HyperText Markup Language, el llenguatge de disseny gràfic CSS, de l'anglès Cascading Style Sheets, i el llenguatge de programació interpretat JavaScript. Aquest tres llenguatges són molt populars en el disseny de pàgines web i tots els navegadors actuals són capaços d'interpretar el codi en JavaScript, permetent d'aquesta manera que la interfície web sigui multiplataforma i pugui ser utilitzada en Android, Linux, Windows, iOS entre altres sistemes.

Els navegadors web no tenen suport pel protocol MQTT. Per solucionar aquest inconvenient, s'ha utilitzat la llibreria Paho JavaScript Client, de Eclipse Foundation, i permetre així comunicar-se amb el broker de MQTT. Aquesta llibreria client, utilitza el protocol WebSocket per així poder comunicar-se amb el broker MQTT.

4.3.1. Interfície web HTML

La pàgina web és un document HTML anomenat "index.html". Aquest fitxer es pot obrir com a pàgina web o com a fitxer de text, que és on s'edita i es programa.

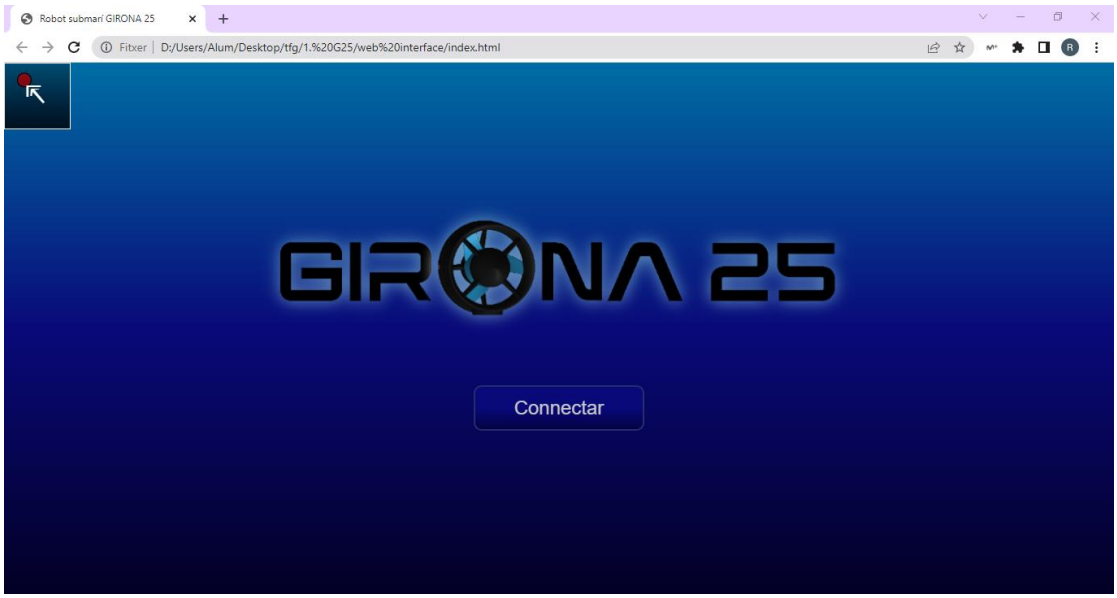

Figura 62. Pàgina principal de la interfície web del Girona 25.

La primera pantalla d'aquest programa és la de la figura anterior. Com es pot observar s'obre en una pàgina web del navegador. Es pot observar el nom del ROV i un botó que permet establir-hi la connexió. També es pot passar a pantalla completa i hi ha una rodona vermella, que es torna de color verd quan es connecta un comandament, com el Logitech Gamepad F310. Un cop connectat al WiFi, tant el robot com el dispositiu, s'ha de clicar el botó connectar i s'obrirà la pantalla de control, que es mostra en la següent figura, des d'on es poden donar totes les indicacions que es desitgin al ROV.

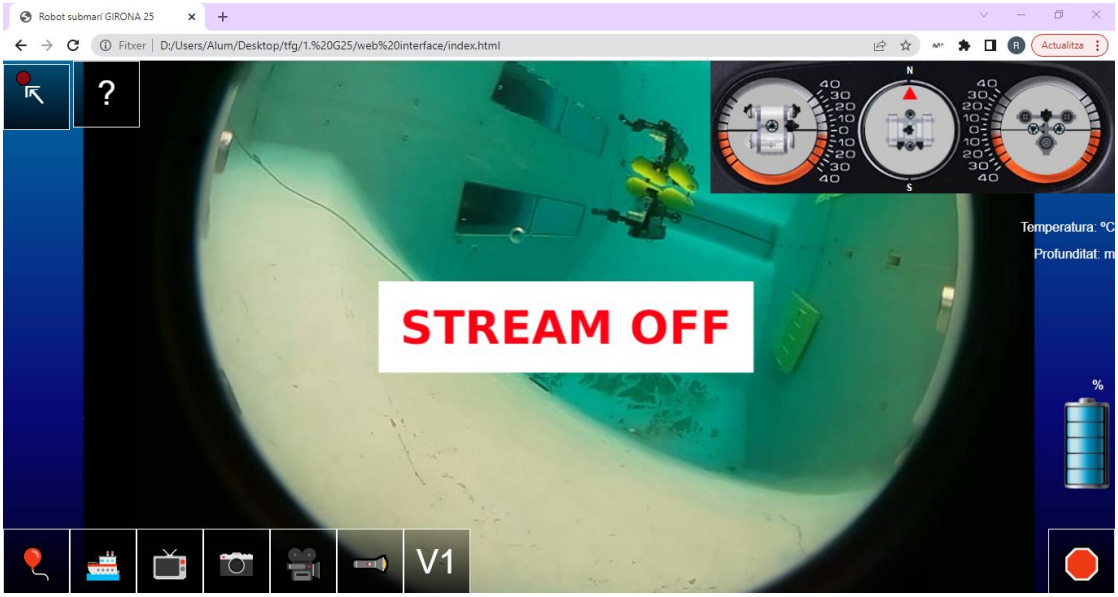

Figura 63. Pàgina de control de la interfície web del Girona 25.

Cadascun dels botons que apareixen a la pantalla té una funció en concret. El primer, el globus vermell, com es pot interpretar, activa i desactiva el seguiment d'objectes. El segon, el vaixell, activa el mode de mantenir la profunditat, quan s'activa apareix una àncora per indicar que aquest mode està activat. Quan es detecta que la profunditat ha augmentat o disminuït s'envia una ordre per fer-lo pujar o baixar i mantenir aquesta profunditat inicial, mitjançant un controlador PI, sintonitzat empíricament.

El tercer botó, la televisió, inicia o pausa la transmissió de vídeo. La càmera fa una captura del vídeo en forma d'imatge. La càmera de vídeo permet gravar la pantalla, per fer un vídeo del que està veient el robot.

La llanterna encén els llums del ROV, mentre aquests estan engegats, el símbol canvia a una bombeta que fa llum per indicar-ho. El següent botó, el V1, indica la velocitat del robot. Hi ha dues velocitats V1, més lenta, i V2, més ràpida, s'alterna polsant-lo.

El senyal vermell atura per complet el robot, per poder-li desconnectar l'alimentació sense perill de malmetre l'ordinador de comandament. Apareix una pestanya on s'ha de confirmar que es vol realitzar aquesta acció, per evitar que es premi el polsador sense voler i el robot s'apagui quan no es desitja. El botó amb el símbol d'interrogant mostra en pantalla la informació del control per teclat.

Cal comentar que les emoticones per als botons són caràcters UTF-8, i per tant pot ser que es vegin diferents depenent del dispositiu en el qual s'utilitzi.

El control del robot per teclat es realitza de forma molt intuïtiva, les fletxes i les tres lletres W, A, S i D serveixen per a comandar-lo en el pla en què es trobi. La Q fa que el robot emergeixi i la E que s'enfonsi. Cal tenir en compte que el robot es mourà mentre la tecla estigui pressionada, en el moment en què es deixi de polsar, el robot atura el seu moviment. Els botons de la pantalla també es poden accionar per teclat, aquests comandaments queden explicats a la pantalla d'informació, que també es mostra amb la lletra H.

El control per teclat no és gaire eficient, per aquest motiu s'implementa el control a través del comandament Logitech Gamepad F310, on amb les palanques de control es mou el robot i amb els diferents polsadors es poden activar els botons de la pantalla.

A la pantalla, es pot observar el roll i el pitch del robot, per tal de poder comprovar que estigui en la posició correcta. També la direcció a la qual apunta, per si en algun moment es perd de vista el robot i l'orientació a través de la càmera no és suficient. A sota, es mostra la temperatura interior del robot, cal controlar-la per evitar sobreescalfaments, i la profunditat a la qual es troba. Per últim, també es mostra el nivell de bateria, per evitar quedar-se sense.

La interfície anterior és la que es mostra quan el dispositiu és un ordinador, però quan és un telèfon mòbil no es pot fer el comandament per teclat ni connectar un comandament. Per aquest motiu, quan s'obre la interfície web des d'un telèfon mòbil apareixen més botons i una palanca de control virtual per fer el comandament per pantalla.

La web per telèfons mòbils incorpora a més el polsador per utilitzar el mòbil amb ulleres de realitat virtual. Quan s'activa aquest botó, desapareixen la resta de botons, es

comença a emetre vídeo, la pantalla passa a pantalla completa i la imatge queda tractada per tal de funcionar amb les ulleres de realitat virtual. En aquest mode, quan es mou el cap el robot fa els mateixos moviments que la persona que porta les ulleres, llegint el giroscopi que porta integrat el telèfon mòbil. A continuació, es pot observar com queda la interfície per a telèfons mòbils.

Figura 64. Pàgina de control de la interfície web del Girona 25 per telèfons mòbils.

Aquest fitxer està escrit en HTML, reuneix i decideix quan s'executen les funcions definides en els fitxers escrits en Javascript a través de la pantalla que mostra, basada en els estils dels fitxers CSS. Per començar importa tots els fitxers que s'utilitzen en l'apartat anomenat "head". A la part del "body" és on defineix totes les pantalles, botons i funcions que s'executen quan es produeixen determinades accions.

4.3.2. Programació JavaScript

En aquests fitxers es troben les funcions que ha de fer la interfície web per a poder comandar el Girona 25. Aquests fitxers són els que utilitzen la llibreria Paho JavaScript Client, per comunicar-se amb el broker.

Els fitxers "gpcon.js", "libgp.js" i "libhtml.js" són els fitxers encarregats de fer funcionar el comandament que es connecta a l'ordinador. El "gpcon.js" és el fitxer principal, mentre que els altres dos són llibreries que utilitza aquest fitxer principal per a poder crear les funcions encarregades de processar les dades que s'envien des del comandament USB.

Els programes "intro_host.js" i "mqtt.js" són els encarregats d'establir la comunicació amb el broker a través de la IP que es proporciona a la pantalla inicial del programa. Utilitzen la llibreria anteriorment esmentada, Paho JavaScript Client, per a connectar-se com a client al broker MQTT.

Figura 65. Logotip de la llibreria Paho JavaScript Client.

Per a poder crear una palanca de control virtual s'utilitzen les funcions escrites dins el fitxer "joystick.js". Aquest programa dibuixa la palanca i obté els valors de la posició d'aquesta perquè després puguin ser processats.

Quan es fa servir l'ordinador i es vol controlar el ROV utilitzant el teclat, el programa anomenat "keyboard.js" és l'encarregat d'aconseguir traspassar les ordres del teclat al programa principal perquè funcioni tot correctament.

Per enviar els missatges adequats als tòpics en funció de les accions realitzades, sobretot els que fan referència als motors, s'utilitza el fitxer "motor.js". En aquest fitxer hi ha les funcions específiques per a enviar a cada tòpic el missatge adequat.

El fitxer "stream.js" conté les funcions referents al vídeo captat per la càmera de la Raspberry. Conté la programació necessària tant per activar i desactivar el vídeo com per processar la imatge que arriba. També és el fitxer que permet fer les captures del vídeo.

Per últim, el fitxer "ui.js" conté les funcions més bàsiques que s'utilitzen en la interfície. També determina quins elements es mostren en funció del tipus de dispositiu que s'està utilitzant.

4.3.3. CSS

Aquest tipus de fitxers defineixen com són els diferents elements que es mostren en pantalla, és a dir, l'aparença. S'utilitzen diferents fitxers per a fer-ho.

El fitxer que determina com són els diferents botons de la interfície web és el "button.css". Es determina el color, com la forma, la mida i tots els aspectes relacionats amb l'aparença.

Encara que no es mostri en pantalla, l'equivalent de les palanques i botons del comandament USB també tenen una aparença determinada en el fitxer "gpcon.css", aquests s'han utilitzat per fer proves i comprovar el correcte funcionament del comandament escollit.

Per a determinar l'aparença i les animacions que es produeixen quan la pàgina està carregant, per exemple, quan es connecta al broker, s'utilitza el fitxer "loading.css".

L'arxiu "style.css" determina la resta d'aparences de la interfície web, és a dir, a quina posició es situa cadascun dels botons, el color del fons de pantalla, la mida de la lletra, si l'objecte és fix o no...

4.3.4. Imatges

Les imatges que s'utilitzen a la interfície web és necessari tenir-les descarregades en el mateix directori on hi ha la pàgina principal "index.html", perquè aquest programa les pugui utilitzar. Hi ha la imatge que es mostra quan encara no es mostra el vídeo, la imatge que s'ensenya quan no hi ha senyal de vídeo, les que s'utilitzen per als nivells de bateria i les del robot per controlar el roll, el pitch i la brúixola.

5. POSADA EN MARXA

En aquest apartat es descriuen totes les proves que s'han realitzat amb el robot per tal de comprovar el seu correcte funcionament.

5.1. Prova fora l'aigua

Aquesta és la primera prova que es va realitzar per a comprovar que tota l'electrònica i tots els sistemes funcionessin correctament. Cal tenir en compte que els motors van lubricats amb aigua i els llums s'han de refrigerar amb l'aigua, per tant, la prova no podia ser de gran durada.

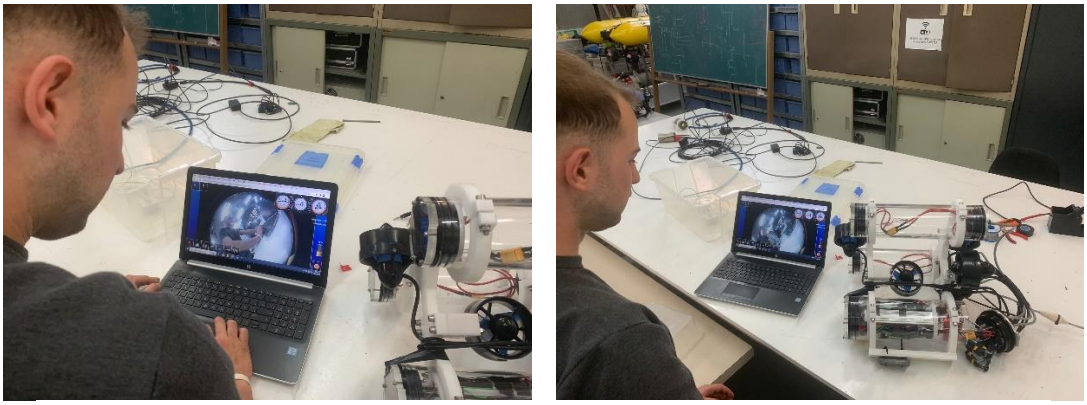

Figura 66. Prova fora l'aigua.

5.2. Prova d'estanquitat

Després de resoldre diferents problemes relacionats amb l'estanquitat el robot ha aguantat tota una nit a dins la piscina, sense l'electrònica a l'interior, a una profunditat de 5 metres, sense que li entres aigua.

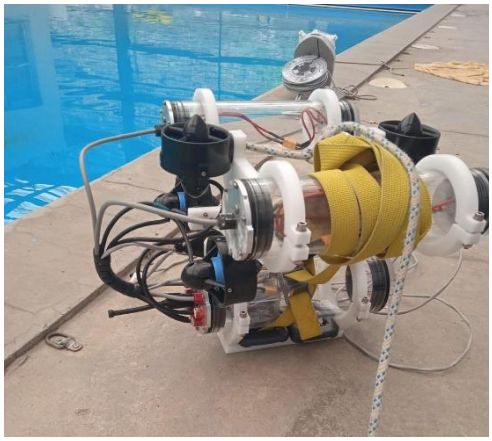

Figura 67. El Girona 25 després de les proves d'estanquitat.

5.3. Prova de flotabilitat i estabilitat

Per comprovar que el ROV sigui estable i que pugui navegar bé, cal que estigui ben tarimat, és a dir, que sigui estable dins de l'aigua. Teòricament, amb el model de l'estructura, ja es va fer aquest estudi, però a l'hora de provar-ho a la realitat es van haver de fer diferents ajustos.

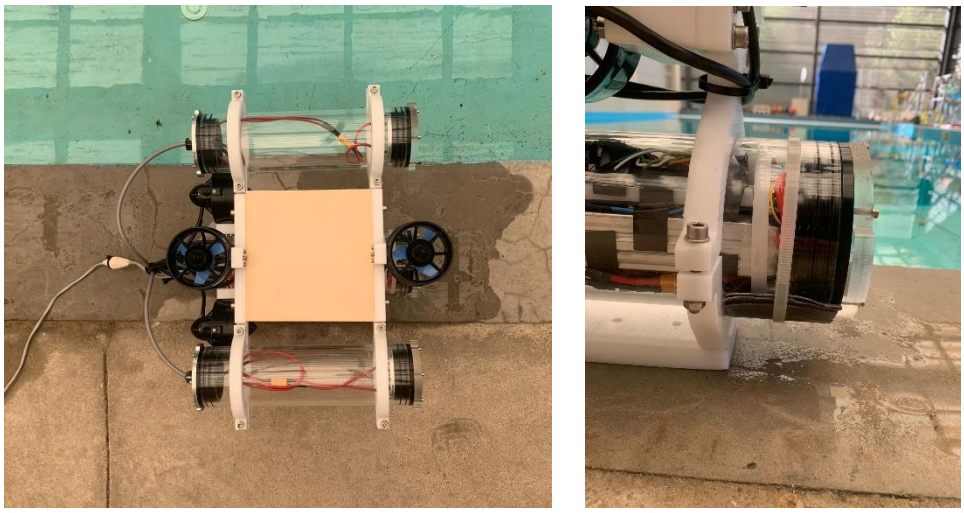

Figura 68. El Girona 25 amb la flotació i els pesos.

El problema era que el robot tenia més pes a darrere que a davant, i això feia que s'inclinés molt. Per solucionar-ho, s'han utilitzat els forats que es van deixar a la part superior per posar-hi flotació HCP 50, que manté la mateixa densitat fins a 100 metres de profunditat, i, a la part inferior, s'han posat unes planxes de plom envoltant la part davantera del cilindre inferior, per fer de contrapès. D'aquesta manera s'aconsegueix eliminar l'efecte dels pesos amb la flotació de la part superior, que faria que el robot s'enfonsés, i amb els pesos s'ha pogut estabilitzar el robot. A més, afegir flotació a la part superior també ha fet que el centre de gravetat i flotació quedin més separats accentuant l'efecte pèndol que anul·la el roll.

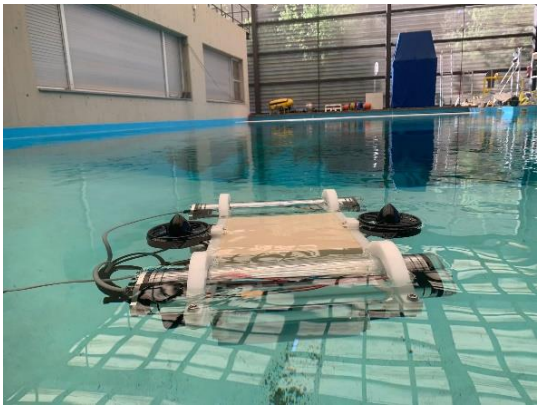

Figura 69. Resultat de l'estabilitat del Girona 25.

5.4. Prova a l'aigua amb l'ordinador

La primera prova de funcionament que s'ha fet és fer-lo navegar per la piscina, provant els diferents sistemes i components, com els motors, els llums i tots els sensors, per comprovar que tot funcionava correctament. Els resultats d'aquesta prova van ser molt bons, el robot tenia una bona navegabilitat, bona estabilitat, els pesos estaven ben ajustats, no entrava aigua i els sistemes d'instrumentació funcionaven correctament. Des de l'ordinador, es podia controlar perfectament amb els comandaments per teclat i la intrerfície web.

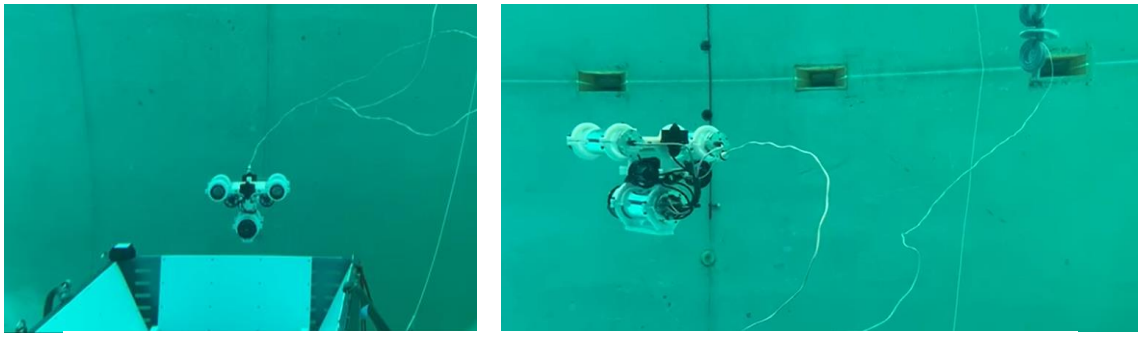

Figura 70. Prova a la piscina del Girona 25, comandat amb l'ordinador.

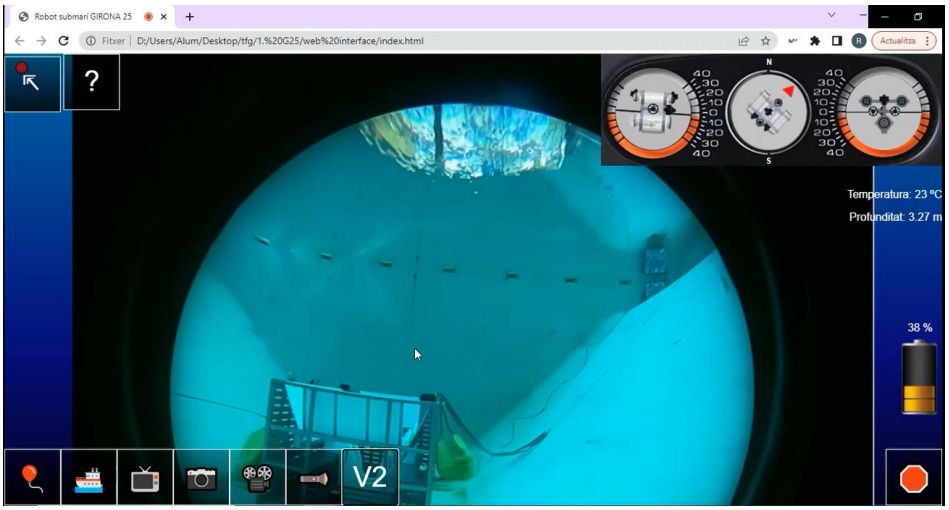

Figura 71. Captura de pantalla de la interfície web de la prova a la piscina.

5.4.1. Prova del comandament Logitech Gamepad F310

Una vegada comprovat que el robot funcionava correctament el comandament per teclat, es va passar a comprovar el funcionament del comandament amb el comandament Logitech Gamepad F310, connectant-lo a l'ordinador. Les proves van donar resultats positius ja que aquest comandament era molt més eficient i intiuïtiu que

el teclat de l'ordinador, el robot es controlava millor i es podien realitzar totes les accions des del mateix comandament, sense haver de tocar la pantalla.

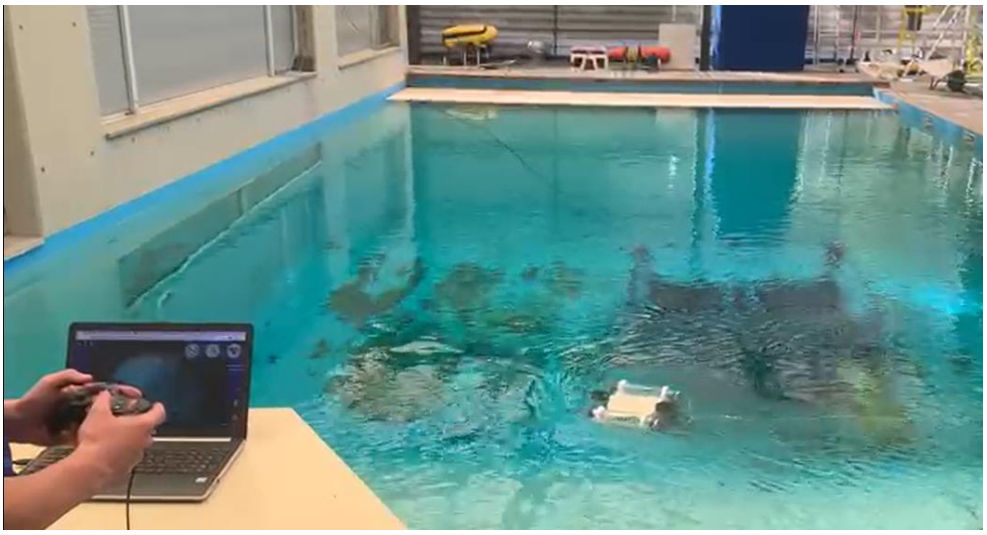

Figura 72. Prova amb el comandament Logitech.

5.5. Prova amb el telèfon mòbil

Després de comprovar que es podia comandar el robot amb l'ordinador, també calia comprovar que es pogués fer servir la interfície web amb el telèfon mòbil. Amb aquest dispositiu es podia comandar el robot d'una manera més fàcil, més compacte i amb més mobilitat, els resultats van ser positius encara que no es comandava tan bé com amb el comandament Logitech Gamepad F310. Aquest fet es deu al fet que amb el telèfon mòbil com que és tot integrat en un sol component, per exemple, costa veure la imatge i controlar el robot a la vegada.

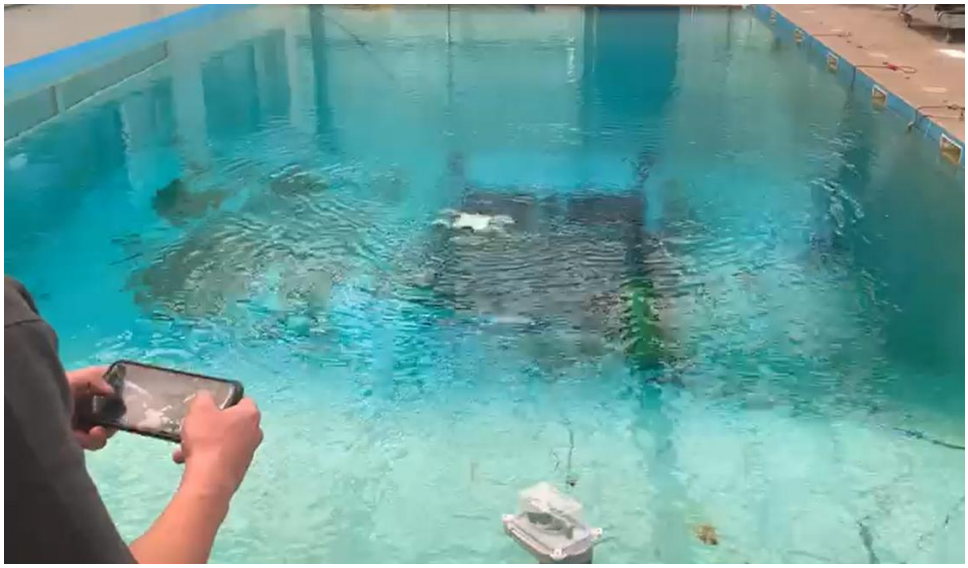

Figura 73. Prova de funcionament comandant amb el telèfon mòbil.

5.5.1. Prova de la realitat virtual

Amb el telèfon mòbil es pot accedir al mode de realitat virtual. Amb aquest mode, la persona qui dirigeix el robot, es posa les ulleres de realitat virtual amb el mòbil al seu interior i pot veure en primera persona el que està veient el robot. Quan la persona mou el cap amunt, avall o cap a algun costat, el robot també ho fa, perquè l'experiència sigui encara més realista. Els resultats d'aquesta prova van ser positius, l'experiència és bastant realista, però és necessari un ordinador i un comandament per a poder complementar-la.

Figura 74. Prova de les ulleres de realitat virtual.

5.6. Prova del mode de mantenir la profunditat

Per tal de mantenir la profunditat s'agafa la dada que dona aquest valor del sensor de profunditat BAR30 que inclou el robot, es passa a la pàgina web i des d'aquesta s'implementa un controlador de tipus PI per a mantenir-la. Es va sintonitzar aquest controlador de forma empírica, és a dir, a base de prova i error, per tant, es van haver de fer diferents proves abans de donar-lo per correcte.

Després de sintonitzar-lo correctament, es van començar a fer les proves amb aquest controlador. El resultat va ser positiu, ja que el robot es mantenia a la mateixa profunditat allà on es deixava.

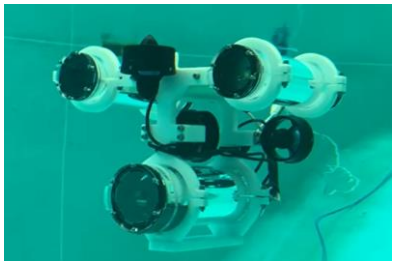

Figura 75. Prova del mode de mantenir la profunditat.

5.6.1. Amb pertorbació

Després de comprovar que funcionava correctament el controlador i el robot quedava allà on es deixava, es va decidir de provar d'incloure una pertorbació al sistema per veure si el robot es tornava a la profunditat establerta. Es va crear una pertorbació fent anar el robot amunt i també enfonsant-lo amb un pal. En ambdós casos, el Girona 25 es va tornar a posar a la profunditat inicial, obtenint un resultat positiu de la prova.

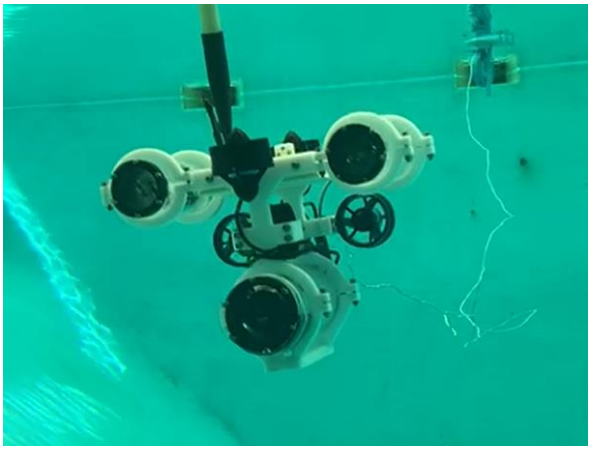

Figura 76. Prova del mode de mantenir la profunditat amb pertorbació.

5.7. Prova del mode autònom

Una vegada desenvolupat el programari del seguiment per color, calia posar-lo a prova. La primera prova que es va fer a dins la piscina va ser lligar una peça groga, el color que segueix, a un pal de manera que es podia posar el pal dins de l'aigua i la peça quedés a dins d'aquesta. Llavors amb el ROV es va activar el mode de seguiment des de la interfície web i movent el pal des de fora, el Girona 25 el seguia. Aquesta prova va donar resultats positius, el robot seguia l'objecte de color groc, sempre mantenint una certa distància.

Figura 77. Prova del mode autònom.

5.7.1. Perseguint el BlueROV

La següent prova que es va fer, a manera d'exemple de per a què es pot utilitzar aquesta eina, va ser posar-li aquesta peça groga a un ROV que hi ha al CIRS, el BlueROV, de manera que el Girona 25 seguiria aquest altre robot. Es va afegir també una planxa groga més grossa, una de les carcasses del Sparus II. Els resultats d'aquesta prova també van ser positius, el Girona 25 aconseguia seguir l'altre robot sense problemes, mantenint una certa distància.

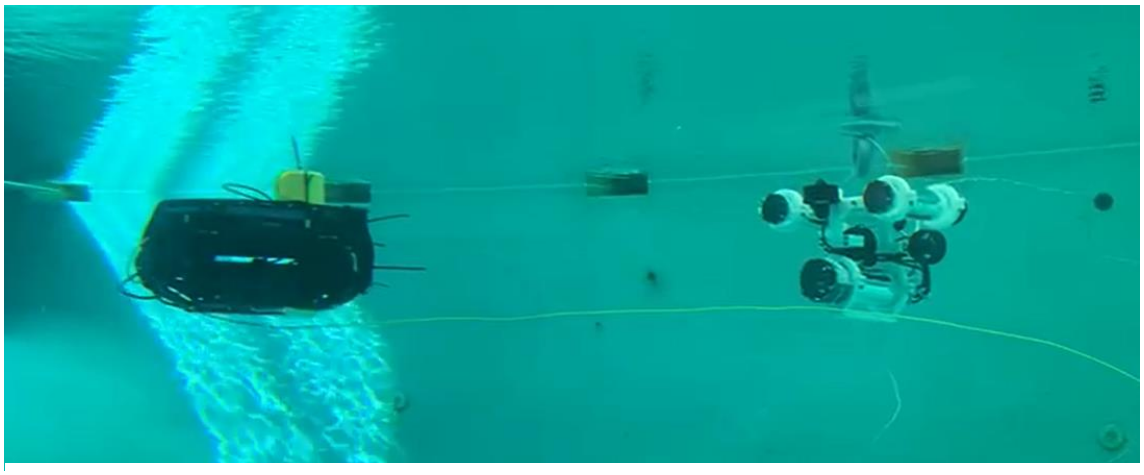

Figura 78. Prova del mode autònom seguint el BlueROV.

5.8. Prova al mar

Per fer la prova al mar, el primer que es va fer és carregar les dues bateries fins al 100% de capacitat i afegir el pes dissenyat per navegar en aigua salada. Es va fer la prova a la platja de Calella de Palafrugell.

Figura 79. Prova al mar amb el Girona 25.

Es va comandar el robot des d'una petita barca i dos bussejadors van gravar el robot des de l'aigua. La prova va ser un èxit, el robot va poder baixar fins a 12,5 metres sense que hi entres aigua i va navegar bé durant tota la prova amb el pes afegit. La visió del robot, però, no era gaire nítida degut a que l'aigua era una mica tèrbola.

Figura 80. Imatge captada pel robot al mar a més de 12 metres de profunditat.

6. RESUM DEL PRESSUPOST

En el document pressupost, s'hi pot trobar el preu unitari de tot allò que es troba a l'estat d'amidaments i s'obté un pressupost final del projecte.

En el pressupost, es conclou que el cost econòmic d'aquest projecte és de dotze mil cinc-cents cinquanta-nou euros i vint-i-set cèntims, sense IVA.

7. CONCLUSIONS

El projecte en qüestió compleix els objectius proposats, inclús els amplia, assolint més especificacions de les plantejades a l'inici del projecte.

El primer objectiu plantejat és dissenyar i construir un robot amb una estructura que faci que el robot pugui navegar sense problemes d'estabilitat i flotabilitat, que es resolen de forma teòrica amb el disseny de l'estructura i s'acaba d'ajustar durant la posada en marxa amb la possibilitat d'afegir pesos i flotació a aquesta. El disseny de l'estructura a semblança del Girona 500, és a dir, amb dos encapsulats a la part superior, que aporten molta flotabilitat i estabilitat, i un encapsulat a la part inferior, més pesat i on es concentra tota l'electrònica, aconsegueix posicionar el centre de masses per sota el centre de flotació, sempre alineats, equilibrant el robot. Amb els pesos i el mòdul de flotació s'aconsegueix que el ROV sigui estable i tingui una bona navegabilitat.

Un altre objectiu que s'ha plantejat és aconseguir millorar la estanquitat. Amb els encapsulats, penetrators i un epoxi més bo i aplicat de forma correcte, s'aconsegueix assolir aquest objectiu. Els nous motors també ajuden a que no es filtri aigua a través d'aquests, ja que són de més qualitat. Les diferents proves fetes durant més de 2 mesos amb el robot han demostrat que es Girona 25 és robust i estanc.

Seguint amb els nous motors, l'objectiu de millorar la navegabilitat amb la nova configuració també s'assoleix gràcies a la incorporació d'un cinquè motor que permet moure's lateralment amb més facilitat. La potència dels motors també ha augmentat i l'estabilitat del ROV també ajuda a que es pugui fer anar més ràpid, encara que el robot sigui més gran i pesat que els anteriors.

Pel que fa al comportament autònom, el software detecta millor els colors, fa els càlculs de les posicions de manera més eficient i s'hi han donat més aplicacions, ja que realment és una eina molt potent.

Referent a la instrumentació, els sensors aporten dades rellevants i precises a l'usuari que li permeten saber l'estat del ROV en tot moment, podent evitar situacions perilloses com pot ser que entri aigua dins l'encapsulat o que es quedi sense bateria.

Els experiments fets durant la posada en marxa del projecte també han estat un èxit. Tots s'han pogut dur a terme i s'han obtingut resultats positius, podent arribar a comandar el robot tan amb l'ordinador, amb el comandament Logitech, amb el telèfon mòbil i amb les ulleres de realitat virtual. El mode de seguiment i el de mantenir la profunditat també han donat bons resultats. La prova al mar també va ser molt profitosa, es van poder gravar molts de vídeos i el robot va funcionar correctament.

També s'assoleixen objectius no comentats a l'inici del treball, per exemple, la utilització de les ulleres de realitat virtual per a controlar el ROV o la connexió al Girona 25 per mitjà de xarxes sense fils.

En conclusió, l'objectiu principal d'aquest TFG, de redissenyar i construir un nou robot submarí utilitzant com a base els seus predecessors, corregint els errors comesos i millorant-ne les prestacions, s'ha assolit. Encara que el preu del projecte sigui de quinze mil euros, fet que es comenta en l'apartat anterior, cal tenir en compte que continua sent un robot de baix cost, si es compara amb el que pot valer el seu predecessor, el Girona 500. Aquest ROV podrà ser utilitzat amb fins educatius, ja que es pot mostrar als grups d'alumnes que fan cursos al CIRS i, a més, la Raspberry també es pot programar amb Scratch, un llenguatge senzill molt utilitzat per iniciar els alumnes en centres de primària i secundària a la programació.

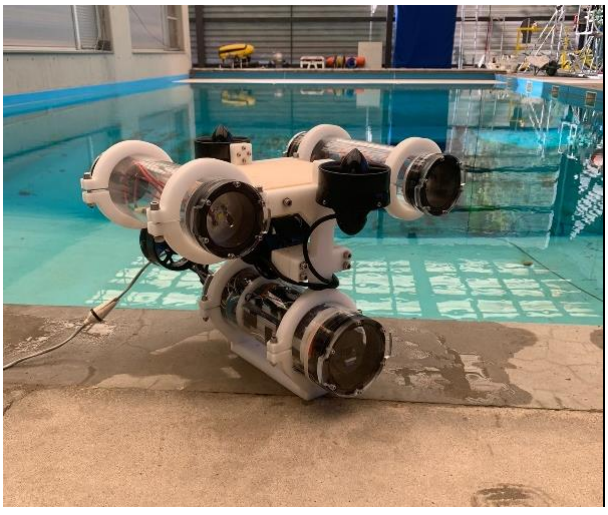

Figura 81. ROV Girona 25.

Es pot afirmar que el projecte ha assolit els objectius proposats a l'inici d'aquest, inclús els ha ampliat. Tot i així, sempre hi ha aspectes a millorar. A curt termini, es podria millorar gràficament la interfície web perquè sigui més agradable a la vista o dissenyar una placa per integrar alguns dels components, com els controladors dels motors, i

optimitzar l'espai de l'interior de l'encapsulat inferior. També es podria preparar el robot perquè els alumnes de secundària que venen al CIRS a fer els tallers puguin programar amb Scratch una petita missió. A llarg termini, es podrien explorar nous sistemes de ventilació dins de l'encapsulat per a regular o dissipar la temperatura, per exemple, utilitzant els nous tubs de Bluerobotics d'alumini anoditzat, que a més, incorporen corda de sallat. També es podria implementar la programació amb el Robot Operating System (ROS), una estructura de desenvolupament de programari per robots, que és el que implementen els robots del CIRS. Una altra possibilitat seria implementar una càmera a cadascun dels encapsulats superiors, per poder crear una visió en tres dimensions i utilitzar-la, per exemple, amb les ulleres de realitat virtual. En cas que es disposi de més pressupost, es podrien posar tots els connectors de tipus subconn, d'aquesta manera es redurien substancialment els problemes d'estanquitat.

ROGER FELIU I SERRAMITJA - DNI 41592276L

Signat digitalment per ROGER FELIU I SERRAMITJA - DNI 41592276L
Data: 2023.05.30 13:02:15 +02'00'

Roger Feliu Serramitja

Graduat en Enginyeria Electrònica Industrial i Automàtica

La Cellera de Ter, 30 de maig de 2023

8. RELACIÓ DE DOCUMENTS

El projecte consta dels següents documents: Memòria, Plànols, Plec de condicions, Estat d'amidaments i Pressupost.

9. BIBLIOGRAFIA

CHARMAST. Charmast 10400mAh 18W Mini Power Bank (https://www.charmast.com/products/charmast-smallest-10000-pd-quick-chargeportable-charger-ultra-compact-10400mah-usb-c-power-delivery-qc-3-0-powerbank-lightest-external-battery-pack-compatible-with-iphone-samsung-google-pixel, 10 de novembre de 2022).

CUFÍ XAVIER, EL FAKDI ANDRES. Attracting talent to increase interest for engineering among secondary school students. IEEE Engineering Education Conference. "Learning Environments and Ecosystems in Engineering Education" (IEEE - EDUCON'2011). Amman (Jordània), abril de 2011.

CUFÍ XAVIER, EL FAKDI ANDRES. Team-based building of a remotely operated underwater robot, an innovative method of teaching engineering. Journal of Intelligent Robotic Systems (special issue on Teaching Robotics), Vol. 81 p.51-61, gener de 2016.

CUFÍ XAVIER, EL FAKDI ANDRES. Team-based building of a remotely operated underwater robot as a method to increase interest for engineering among secondary school students. 4th International Conference on Education and New Learning Technologies. Barcelona, Juliol 2012.

CUFÍ XAVIER, VEGA DAURA. "Manual de Implementación de Arduino y Scratch para el Control de ROVs". Plataforma Oceánica de Canarias (PLOCAN). Gran Canaria, 2016.

CUFÍ XAVIER, VEGA DAURA. "Taller de Robótica Submarina (Manual de Construcción de un ROV)". Plataforma Oceánica de Canarias (PLOCAN). Gran Canaria, 2014.

LIFEWIRE. What is the HSV (Hue, Saturation, Value) Color Model? (https://www.lifewire.com/what-is-hsv-in-design-1078068, 20 de novembre de 2021)

MICHALNG. mjpg-streamer. (https://github.com/jacksonliam/mjpg-streamer, 17 de novembre de 2022)

OPENCV. Introduction to OpenCV-Python Tutorials.
(https://docs.opencv.org/4.x/d0/de3/tutorial_py_intro.html, 4 de novembre de 2022)

RENESAS. What are Brushless DC Motors
(https://www.renesas.com/us/en/support/engineer-school/brushless-dc-motor-01-overview , 12 de novembre de 2022)

STEVE. How MQTT Works - Beginners Guide (http://www.steves-internet-guide.com/mqtt-works/, 2 de novembre de 2022)

10. GLOSSARI

AUV: Autonomous Underwater Vehicle

CIRS: Centre d'Investigació en Robòtica Submarina

CSS: Cascading Style Sheets

ESC: Electronic Speed Controler.

GPIO: General-purpose input/output.

HTML: Hyper Text Markup Language.

I^2C: Inter-Integrated Circuit

JPEG: Joint Photographic Experts Group

LED: Díode Emissor de Llum. (Light Emitting Diode)

MQTT: Message Queuing Telemetry Transport

PWM: Pulse-Width Modulation

RC: Radio Control

ROS: Robot Operating System

ROV: Remote Operated Vehicle

SDA: Serial Data

SCL: Serial Clock

A. PROGRAMA

En aquest annex es presenta el codi utilitzat en el projecte que s'ha comentat anteriorment, el servidor web i el programa intern de la Raspberry.

A.1. Programa de la Raspberry

Com ja s'ha comentat aquest programa es divideix en diferents fitxers escrits en Python. Només cal executar el primer de tots, "mqtt_client.py", ja que engloba tots els altres.

A.1.1. Programa PROGRAMAPRINCIPAL.py

```
#!/usr/bin/python3
from time import sleep
import subprocess
import os
import mqtt_client

sleep(60)

while True:
    subprocess.run(["python3", "mqtt_client.py"], cwd="~/")
```

A.1.2. Programa mqtt_client.py

```
#!/usr/bin/python3
import paho.mqtt.client as mqtt
from time import sleep
import subprocess
import json
import RPi.GPIO as GPIO
from gpiozero import PWMOutputDevice as POD
from gpiozero.pins.pigpio import PiGPIOFactory
import gpiozero
from motor import Motor
from orders import Actions
from config import host
from mpu6050 import mpu6050
from hmc5883l import hmc5883l
from ina219 import INA219
from ina219 import DeviceRangeError
```

```
import bar30
import time

################################################################################

class NewProcess():
    """
    Aquesta classe s'utilitza per executar i aturar el mode de
    seguiment d'objectes per color.
    """
    def __init__(self, cmd, Proc: subprocess.Popen = None):
        self.cmd = cmd
        self.newProc = Proc

    def start(self):
        self.newProc = subprocess.Popen(self.cmd, cwd="/home/pi/")

    def kill(self):
        self.newProc.kill()

################################################################################

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        client.connected_flag=True
        print("connected OK Returned code=",rc)
        #print(robot.show_values())
        #sleep(2)
    else:
        print("Bad connection Returned code=",rc)
        client.bad_connection_flag=True

def on_message(client, userdata, message):
    """
    Aquesta funcio analitza els topics i el contigut dels
    missatges rebuts i determina quines accions s'ha de
    executar.
    """
    print("message received ", str(message.payload.decode("utf-8")))
    topic_ = str(message.topic)
    if topic_ == "Motors":
        msg = json.loads(message.payload.decode()) # Llista de missatges
rebuts
        if msg[0] == "up":
            spd_l = float(msg[1])
            spd_r = float(msg[2])
            client.publish(topic, "UP")
            robot.up(spd_l, spd_r)
```

```
        elif msg[0] == "dwn":
            spd_l = float(msg[1])
            spd_r = float(msg[2])
            client.publish(topic, "DOWN")
            robot.down(spd_l, spd_r)

        elif msg[0] == "lleft":
            spd = float(msg[1])
            client.publish(topic, "Left")
            robot.left(spd)

        elif msg[0] == "lright":
            spd = float(msg[1])
            client.publish(topic, "Right")
            robot.right(spd)

        elif msg[0] == "fwd":
            spd_l = float(msg[1])
            spd_r = float(msg[2])
            client.publish(topic, "Move forward")
            robot.forward(spd_l,spd_r)

        elif msg[0] == "bkd":
            spd_l = float(msg[1])
            spd_r = float(msg[2])
            client.publish(topic, "Move backward")
            robot.backward(spd_l, spd_r)

        elif msg[0] == "left":
            spd = float(msg[1])
            client.publish(topic, "Turn left")
            robot.turn_left(spd)

        elif msg[0] == "right":
            spd = float(msg[1])
            client.publish(topic, "Turn right")
            robot.turn_right(spd)

        elif msg[0] == "off":
            client.publish(topic, "OFF")
            robot._off()

        elif msg[0] == "stopraspy":
            passw= 'raspberry'
            command= 'shutdown -h now'
            command= command.split()
            cmd1=subprocess.Popen(['echo', passw], stdout=subprocess.PIPE)
            cmd2=subprocess.Popen(['sudo','-S']+command, stdin= cmd1.stdout,
stdout=subprocess.PIPE)
```

```
        elif msg[0] == "stopdalt":
            client.publish(topic,"STOP_DALT")
            robot.stop_dalt()

        elif msg[0] == "stopbaix":
            client.publish(topic,"STOP_BAIX")
            robot.stop_baix()

        else:
            client.publish(topic, "STOP")
            robot.stop_all()

    elif topic_ == "Light":
        msg = str(message.payload.decode("utf-8"))
        if msg == "on":
            GPIO.output(33, 1)
        if msg == "off":
            GPIO.output(33, 0)

    elif topic_ == "Camera":
        try:
            subprocess.run(["python3", "take_photos.py"], cwd="/home/pi/")
            client.publish(topic, "Fotografia feta")
        except:
            client.publish(topic, "Error")
            print("Error")

    elif topic_ == "Stream":
        msg = str(message.payload.decode("utf-8"))
        print("starting")
        if msg == "play":
            subprocess.Popen(["python3", "streaming.py"], cwd="/home/pi/")
            client.publish(topic, "Start stream")

        elif msg == "pause":
            subprocess.run(["sudo", "killall", "mjpg_streamer"])
            client.publish(topic, "Stream finished")

    elif topic_ == "Follow":
        msg = str(message.payload.decode("utf-8"))
        if msg == "start":
            tracking.start()
            client.publish(topic, "Start tracking")
        elif msg == "stop":
            tracking.kill()
            client.publish(topic, "Stop tracking")

    elif topic_ == "Info":
```

```
        print(robot.value)

def connect_mqtt():
    client = mqtt.Client(client_id)
    client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.connect(host, port)
    return client


def subscribe(client: mqtt):
    client.subscribe(topics)
    client.on_message = on_message


########################################################################
mqtt.Client.connected_flag=False

#host = "169.254.10.124"
port = 1883
print(host)
client_id = "Nemo"
username = "Node"
password = "password"

topic = "Server"
topic2 = "Motors"
topic3 = "Camera"
topic4 = "Follow"
topic5 = "Stream"
topic6 = "Sensors"
topic7 = "Light"
topic8 = "Aigua"
topics = [(topic2,0), (topic3, 0), (topic4, 0), (topic5,0), (topic6,0),
(topic7,0), (topic8,0)]

cmd = ["python3", "tracking.py"]
tracking = NewProcess(cmd)

robot = Actions(16, 19, 20, 26, 12)

#####################################################################
aigua=0
def enviaaigua(channel):
     global aigua
     if aigua==0:
         client.publish("Aigua", "Aigua")
     aigua=1
client = connect_mqtt()
```

```
subscribe(client)
rollpitchant=[0,0,0,0]
directionant=0
profant=0
bateriant=0
rollpitch =[0,0,0,0]
direction=0
bateria=0
voltage=0
SHUNT_OHMS = 0.00075
MAX_EXPECTED_AMPS = 100
prof=0
mpu = mpu6050(0x68)
compass = hmc5883l(gauss=4.7, declination =(2,41))
ina = INA219(SHUNT_OHMS, MAX_EXPECTED_AMPS)
ina.configure(ina.RANGE_16V, ina.GAIN_2_80MV)
sensor = bar30.MS5837_30BA()
if not sensor.init():
        print("Sensor could not be initialized")
        exit(1)

# We have to read values from sensor to update pressure and temperature
if not sensor.read():
    print("Sensor read failed!")
    exit(1)

print(("Pressure: %.2f atm  %.2f Torr  %.2f psi") % (
sensor.pressure(bar30.UNITS_atm),
sensor.pressure(bar30.UNITS_Torr),
sensor.pressure(bar30.UNITS_psi)))

print(("Temperature: %.2f C  %.2f F  %.2f K") % (
sensor.temperature(bar30.UNITS_Centigrade),
sensor.temperature(bar30.UNITS_Farenheit),
sensor.temperature(bar30.UNITS_Kelvin)))

freshwaterDepth = sensor.depth() # default is freshwater
sensor.setFluidDensity(bar30.DENSITY_SALTWATER)
saltwaterDepth = sensor.depth() # No nead to read() again
sensor.setFluidDensity(1000) # kg/m^3
print(("Depth: %.3f m (freshwater)  %.3f m (saltwater)") % (freshwaterDepth,
saltwaterDepth))

GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Set pin 10 to be an
input pin and set initial value to be pulled low (off)
GPIO.add_event_detect(18,GPIO.FALLING,callback=enviaaigua)
```

```
GPIO.setup(19, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Set pin 10 to be an
input pin and set initial value to be pulled low (off)
GPIO.add_event_detect(19,GPIO.FALLING,callback=enviaaigua)
GPIO.setup(33, GPIO.OUT)
client.loop_start()
while(1):
    rollpitchant=rollpitch
    directionant=direction
    profant=prof
    bateriant=bateria
    rollpitch=mpu.dadesMPU()
    direction=compass.degrees(compass.heading())
    voltatge= ina.voltage()
    bateria=(voltatge-11.1)/1.5*100
    if sensor.read():
                freshwaterDepth = sensor.depth() # default is freshwater
                sensor.setFluidDensity(bar30.DENSITY_SALTWATER)
                saltwaterDepth = sensor.depth() # No nead to read() again
                sensor.setFluidDensity(1000) # kg/m^3
                prof=sensor.depth()
                prof=round(prof,2)
    if prof<=0:
        prof=0
    if bateria<=0:
        bateria=0
    if abs(bateriant-bateria)>=1 or abs(rollpitch[0]-rollpitchant[0])>= 4 or
abs(rollpitch[1]-rollpitchant[1])>= 4 or abs(directionant-direction)>=5 or
abs(prof-profant)>=0.01:
        msgenv=str([rollpitch[0],rollpitch[1],direction,rollpitch[3], prof,
bateria]).encode("utf-8")
        client.publish("Sensors",msgenv)
```

A.1.3. Programa orders.py

```
from gpiozero import PWMOutputDevice as POD
from gpiozero.pins.pigpio import PiGPIOFactory
import gpiozero
from time import sleep
from motor import Motor


class Actions(Motor):
    """
    orders

    Aquesta clase defineix el moviment del robot.
```

```
Els pins s'han de introduir en el següent ordre:
  1. Motor vertical esquerra
  2. Motor vertical dreta
  3. Motor horitzontal esquerra
  4. Motor horitzontal dreta
  5. Motor lateral

"""
def __init__(self, *pins):
    self.motorVE = Motor(pins[0])
    self.motorVD = Motor(pins[1])
    self.motorHE = Motor(pins[2])
    self.motorHD = Motor(pins[3])
    self.motorC = Motor(pins[4])

def up(self, spd_l=0, spd_r=0):
    self.motorVE.horari(spd_l)
    self.motorVD.antihorari(spd_r)

def down(self, spd_l=0, spd_r=0):
    self.motorVE.antihorari(spd_l)
    self.motorVD.horari(spd_r)

def forward(self, spd_l, spd_r):
    self.motorHE.horari(spd_l)
    self.motorHD.antihorari(spd_r)

def backward(self, spd_l, spd_r):
    self.motorHE.antihorari(spd_l)
    self.motorHD.horari(spd_r)

def turn_right(self, spd):
    self.motorHE.horari(spd)
    self.motorHD.horari(spd)

def turn_left(self, spd):
    self.motorHE.antihorari(spd)
    self.motorHD.antihorari(spd)

def right(self, spd):
    self.motorC.horari(spd)

def left(self, spd):
    self.motorC.antihorari(spd)

def stop_dalt(self):
    self.motorVE.stop()
    self.motorVD.stop()
```

```
        self.motorC.stop()

    def stop_baix(self):
        self.motorHE.stop()
        self.motorHD.stop()

    def stop_all(self):
        """
        Atura tots els motors.
        """
        self.motorVE.stop()
        self.motorVD.stop()
        self.motorHE.stop()
        self.motorHD.stop()
        self.motorC.stop()


    def _off(self):
        """
        Apaga tots els motors.
        """
        self.motorVE.off()
        self.motorVD.off()
        self.motorHE.off()
        self.motorHD.off()
        self.motorC.off()
```

A.1.4. Programa motor.py

```
from gpiozero import PWMOutputDevice as POD
from gpiozero.pins.pigpio import PiGPIOFactory
from time import sleep

################################################

"""
Motor VE:
  - valor > 0.149 --> antihorari. (baixar)
  - valor < 0.149 --> horari.     (pujar)

Motor VD:
  - valor > 0.149 --> antihorari. (pujar)
  - valor < 0.149 --> horari.     (baixar)

Motor HE:
  - valor > 0.149 --> antihorari. (enrera)
```

```
  - valor < 0.149 --> horari.     (endavant)

Motor HD:
  - valor > 0.149 --> antihorari. (endavant)
  - valor < 0.149 --> horari.      (enrera)


El motiu pel qual els motors paral·les giren en sentit contrari quan
el vol moure en linea recta es degut a que tenen les hèlix invertides
i d'aquesta forma els moments dels motors s'anulen entre ells donant
major estabilitat al robot.

"""

class Motor():
    """
    motors

    Aquesta clase defineix l'estat d'un motor.

    Parameters:
      pin :      int
      Número de pin GPIO de la raspberry a utilitzar.

    """

    DEFAULT_SPEED = 0

    def __init__(self, pin):
        # Important especificar la llibreria de pins PiGPIO al usar PWM.
        self.motor = POD(pin=pin, pin_factory=PiGPIOFactory())

        # Procés d'inicialització dels motors.
        self.motor.off()
        sleep(0.5)
        self.motor.on()
        sleep(0.5)

        # Per començar a funcionar, es necessari que el valor inicial del pin
sigui 0.149
        self.motor.value = 0.149
        self.speed = self.DEFAULT_SPEED

    def motor_speed(self, spd=None):
        """
        Determina la velocitat a la que es mou el motor.

        spd : float, entre 0 i 1
```

```
        Retorna el valor de velocitat a la que gira el motor.
        """
        if spd is not None:
            self.speed = spd * 0.006
            return self.speed

        return self.speed

    def horari(self, spd=None):
        """
        Fa girar el motor en sentit horari.
        """
        self.motor.value = 0.147 - self.motor_speed(spd)
        print(self.motor.value)

    def antihorari(self, spd=None):
        """
        Fa girar el motor en sentit antihorari.
        """
        self.motor.value = 0.152 + self.motor_speed(spd)
        print(self.motor.value)

    def stop(self):
        """
        Atura el motor
        """
        self.motor.value = 0.149

    def off(self):
        """
        Apaga el pin
        """
        self.motor.off()
        print('off')


    def on(self):
        """
        Encen el pin
        """
        self.motor.off()
        print("on")


    def show_value(self):
        """
        Mostra el valor del pin.
        """
        print(self.motor.value)
```

A.1.5. Programa streaming.py

```
import subprocess

"""
Al executar aquest codi, es posa en marxa el programa
mjpg-streamer per iniciar la transmissió en directe del
que veu la càmara del robot.
"""

subprocess.run(['/usr/local/bin/mjpg_streamer', "-i", '/usr/local/lib/mjpg-
streamer/input_uvc.so -r 1280x720 -d /dev/video0 -f 30', "-o",
'/usr/local/lib/mjpg-streamer/output_http.so -p 8090 -w /usr/local/share/mpjg-
streamer/www'])
```

A.1.6. Programa take_photos.py

```
from datetime import datetime
from signal import pause
import requests


"""
Al executar aquest codi, captura una imatge i la guarda al
directory /Photos en format ISO.

"""


timestamp = datetime.now().isoformat()
try:
    img = requests.get("http://localhost:8090/?action=snapshot")
    name = ("/home/pi/Photos/%s.jpg" % timestamp)

    with open(name, "wb") as f:
        f.write(img.content)
    f.close()


except:
    from picamera import PiCamera
    PiCamera().capture("/home/pi/Photos/%s.jpg" % timestamp)


print("Picture taken")
```

A.1.7. Programa bar30.py

```
try:
    import smbus
except:
    print('Try sudo apt-get install python-smbus2')

import time
from time import sleep

# Models
MODEL_02BA = 0
MODEL_30BA = 1

# Oversampling options
OSR_256  = 0
OSR_512  = 1
OSR_1024 = 2
OSR_2048 = 3
OSR_4096 = 4
OSR_8192 = 5

# kg/m^3 convenience
DENSITY_FRESHWATER = 997
DENSITY_SALTWATER = 1029

# Conversion factors (from native unit, mbar)
UNITS_Pa     = 100.0
UNITS_hPa    = 1.0
UNITS_kPa    = 0.1
UNITS_mbar   = 1.0
UNITS_bar    = 0.001
UNITS_atm    = 0.000986923
UNITS_Torr   = 0.750062
UNITS_psi    = 0.014503773773022

# Valid units
UNITS_Centigrade = 1
UNITS_Farenheit  = 2
UNITS_Kelvin     = 3


class MS5837(object):

    # Registers
    _MS5837_ADDR            = 0x76
    _MS5837_RESET           = 0x1E
    _MS5837_ADC_READ        = 0x00
```

```
    _MS5837_PROM_READ        = 0xA0
    _MS5837_CONVERT_D1_256   = 0x40
    _MS5837_CONVERT_D2_256   = 0x50

    def __init__(self, model=MODEL_30BA, bus=1):
        self._model = model

        try:
            self._bus = smbus.SMBus(bus)
        except:
            print("Bus %d is not available."%bus)
            print("Available busses are listed as /dev/i2c*")
            self._bus = None

        self._fluidDensity = DENSITY_FRESHWATER
        self._pressure = 0
        self._temperature = 0
        self._D1 = 0
        self._D2 = 0

    def init(self):
        if self._bus is None:
            "No bus!"
            return False

        self._bus.write_byte(self._MS5837_ADDR, self._MS5837_RESET)

        # Wait for reset to complete
        sleep(0.01)

        self._C = []

        # Read calibration values and CRC
        for i in range(7):
            c = self._bus.read_word_data(self._MS5837_ADDR,
self._MS5837_PROM_READ + 2*i)
            c =  ((c & 0xFF) << 8) | (c >> 8) # SMBus is little-endian for
word transfers, we need to swap MSB and LSB
            self._C.append(c)

        crc = (self._C[0] & 0xF000) >> 12
        if crc != self._crc4(self._C):
            print("PROM read error, CRC failed!")
            return False

        return True

    def read(self, oversampling=OSR_8192):
        if self._bus is None:
```

```
            print("No bus!")
            return False

        if oversampling < OSR_256 or oversampling > OSR_8192:
            print("Invalid oversampling option!")
            return False

        # Request D1 conversion (pressure)
        self._bus.write_byte(self._MS5837_ADDR, self._MS5837_CONVERT_D1_256 +
2*oversampling)

        # Maximum conversion time increases linearly with oversampling
        # max time (seconds) ~= 2.2e-6(x) where x = OSR = (2^8, 2^9, ...,
2^13)
        # We use 2.5e-6 for some overhead
        sleep(2.5e-6 * 2**(8+oversampling))

        d = self._bus.read_i2c_block_data(self._MS5837_ADDR,
self._MS5837_ADC_READ, 3)
        self._D1 = d[0] << 16 | d[1] << 8 | d[2]

        # Request D2 conversion (temperature)
        self._bus.write_byte(self._MS5837_ADDR, self._MS5837_CONVERT_D2_256 +
2*oversampling)

        # As above
        sleep(2.5e-6 * 2**(8+oversampling))

        d = self._bus.read_i2c_block_data(self._MS5837_ADDR,
self._MS5837_ADC_READ, 3)
        self._D2 = d[0] << 16 | d[1] << 8 | d[2]

        # Calculate compensated pressure and temperature
        # using raw ADC values and internal calibration
        self._calculate()

        return True

    def setFluidDensity(self, denisty):
        self._fluidDensity = denisty

    # Pressure in requested units
    # mbar * conversion
    def pressure(self, conversion=UNITS_mbar):
        return self._pressure * conversion

    # Temperature in requested units
    # default degrees C
    def temperature(self, conversion=UNITS_Centigrade):
```

```
        degC = self._temperature / 100.0
        if conversion == UNITS_Farenheit:
            return (9.0/5.0)*degC + 32
        elif conversion == UNITS_Kelvin:
            return degC + 273
        return degC

    # Depth relative to MSL pressure in given fluid density
    def depth(self):
        return (self.pressure(UNITS_Pa)-101300)/(self._fluidDensity*9.80665)

    # Altitude relative to MSL pressure
    def altitude(self):
        return (1-pow((self.pressure()/1013.25),.190284))*145366.45*.3048

    # Cribbed from datasheet
    def _calculate(self):
        OFFi = 0
        SENSi = 0
        Ti = 0

        dT = self._D2-self._C[5]*256
        if self._model == MODEL_02BA:
            SENS = self._C[1]*65536+(self._C[3]*dT)/128
            OFF = self._C[2]*131072+(self._C[4]*dT)/64
            self._pressure = (self._D1*SENS/(2097152)-OFF)/(32768)
        else:
            SENS = self._C[1]*32768+(self._C[3]*dT)/256
            OFF = self._C[2]*65536+(self._C[4]*dT)/128
            self._pressure = (self._D1*SENS/(2097152)-OFF)/(8192)

        self._temperature = 2000+dT*self._C[6]/8388608

        # Second order compensation
        if self._model == MODEL_02BA:
            if (self._temperature/100) < 20: # Low temp
                Ti = (11*dT*dT)/(34359738368)
                OFFi = (31*(self._temperature-2000)*(self._temperature-
2000))/8
                SENSi = (63*(self._temperature-2000)*(self._temperature-
2000))/32

        else:
            if (self._temperature/100) < 20: # Low temp
                Ti = (3*dT*dT)/(8589934592)
                OFFi = (3*(self._temperature-2000)*(self._temperature-2000))/2
                SENSi = (5*(self._temperature-2000)*(self._temperature-
2000))/8
                if (self._temperature/100) < -15: # Very low temp
```

```
                    OFFi =
OFFi+7*(self._temperature+1500)*(self._temperature+1500)
                    SENSi =
SENSi+4*(self._temperature+1500)*(self._temperature+1500)
            elif (self._temperature/100) >= 20: # High temp
                Ti = 2*(dT*dT)/(137438953472)
                OFFi = (1*(self._temperature-2000)*(self._temperature-
2000))/16
                SENSi = 0

        OFF2 = OFF-OFFi
        SENS2 = SENS-SENSi

        if self._model == MODEL_02BA:
            self._temperature = (self._temperature-Ti)
            self._pressure = (((self._D1*SENS2)/2097152-OFF2)/32768)/100.0
        else:
            self._temperature = (self._temperature-Ti)
            self._pressure = (((self._D1*SENS2)/2097152-OFF2)/8192)/10.0

    # Cribbed from datasheet
    def _crc4(self, n_prom):
        n_rem = 0

        n_prom[0] = ((n_prom[0]) & 0x0FFF)
        n_prom.append(0)

        for i in range(16):
            if i%2 == 1:
                n_rem ^= ((n_prom[i>>1]) & 0x00FF)
            else:
                n_rem ^= (n_prom[i>>1] >> 8)

            for n_bit in range(8,0,-1):
                if n_rem & 0x8000:
                    n_rem = (n_rem << 1) ^ 0x3000
                else:
                    n_rem = (n_rem << 1)

        n_rem = ((n_rem >> 12) & 0x000F)

        self.n_prom = n_prom
        self.n_rem = n_rem

        return n_rem ^ 0x00

class MS5837_30BA(MS5837):
    def __init__(self, bus=1):
        MS5837.__init__(self, MODEL_30BA, bus)
```

```
class MS5837_02BA(MS5837):
    def __init__(self, bus=1):
        MS5837.__init__(self, MODEL_02BA, bus)
```

A.1.8. Programa hmc5883l.py

```
import smbus
import math
import time
import sys

class hmc5883l:

    __scales = {
        0.88: [0, 0.73],
        1.30: [1, 0.92],
        1.90: [2, 1.22],
        2.50: [3, 1.52],
        4.00: [4, 2.27],
        4.70: [5, 2.56],
        5.60: [6, 3.03],
        8.10: [7, 4.35],
    }

    def __init__(self, port=0, address=0x1E, gauss=1.3, declination=(0,0)):
        self.bus = smbus.SMBus(port)
        self.address = address

        (degrees, minutes) = declination
        self.__declDegrees = degrees
        self.__declMinutes = minutes
        self.__declination = (degrees + minutes / 60) * math.pi / 180

        (reg, self.__scale) = self.__scales[gauss]
        self.bus.write_byte_data(self.address, 0x00, 0x70) # 8 Average, 15 Hz,
normal measurement
        self.bus.write_byte_data(self.address, 0x01, reg << 5) # Scale
        self.bus.write_byte_data(self.address, 0x02, 0x00) # Continuous
measurement

    def declination(self):
        return (self.__declDegrees, self.__declMinutes)

    def twos_complement(self, val, len):
        # Convert twos compliment to integer
        if (val & (1 << len - 1)):
```

```
            val = val - (1<<len)
        return val

    def __convert(self, data, offset):
        val = self.twos_complement(data[offset] << 8 | data[offset+1], 16)
        if val == -4096: return None
        return round(val * self.__scale, 4)

    def axes(self):
        data = self.bus.read_i2c_block_data(self.address, 0x00)
        #print map(hex, data)
        x = self.__convert(data, 3)
        y = self.__convert(data, 7)
        z = self.__convert(data, 5)
        return (x,y,z)

    def heading(self):
        (x, y, z) = self.axes()
        headingRad = math.atan2(y, x)
        headingRad += self.__declination

        # Correct for reversed heading
        if headingRad < 0:
            headingRad += 2 * math.pi

        # Check for wrap and compensate
        elif headingRad > 2 * math.pi:
            headingRad -= 2 * math.pi

        # Convert to degrees from radians
        headingDeg = headingRad * 180 / math.pi
        return headingDeg

    def degrees(self, headingDeg):
        degrees = math.floor(headingDeg)
        minutes = round((headingDeg - degrees) * 60)
        return (degrees)

    def __str__(self):
        (x, y, z) = self.axes()
        return "Axis X: " + str(x) + "\n" \
               "Axis Y: " + str(y) + "\n" \
               "Axis Z: " + str(z) + "\n" \
               "Declination: " + self.degrees(self.declination()) + "\n" \
               "Heading: " + self.degrees(self.heading()) + "\n"

if __name__ == "__main__":
    # http://magnetic-declination.com/Great%20Britain%20(UK)/Harrogate#
BARCELONA
```

```
    compass = hmc5883l(gauss = 4.7, declination = (2,41))
    while True:
        sys.stdout.write("\rHeading: " +
str(compass.degrees(compass.heading())) + "     ")
        sys.stdout.flush()
        time.sleep(0.5)
```

A.1.9. Programa hmc5883l.py

```
import logging
import time
from math import trunc
import Adafruit_GPIO.I2C as I2C


class INA219:
    """Class containing the INA219 functionality."""

    RANGE_16V = 0  # Range 0-16 volts
    RANGE_32V = 1  # Range 0-32 volts

    GAIN_1_40MV = 0  # Maximum shunt voltage 40mV
    GAIN_2_80MV = 1  # Maximum shunt voltage 80mV
    GAIN_4_160MV = 2  # Maximum shunt voltage 160mV
    GAIN_8_320MV = 3  # Maximum shunt voltage 320mV
    GAIN_AUTO = -1  # Determine gain automatically

    ADC_9BIT = 0  # 9-bit conversion time  84us.
    ADC_10BIT = 1  # 10-bit conversion time 148us.
    ADC_11BIT = 2  # 11-bit conversion time 2766us.
    ADC_12BIT = 3  # 12-bit conversion time 532us.
    ADC_2SAMP = 9  # 2 samples at 12-bit, conversion time 1.06ms.
    ADC_4SAMP = 10  # 4 samples at 12-bit, conversion time 2.13ms.
    ADC_8SAMP = 11  # 8 samples at 12-bit, conversion time 4.26ms.
    ADC_16SAMP = 12  # 16 samples at 12-bit,conversion time 8.51ms
    ADC_32SAMP = 13  # 32 samples at 12-bit, conversion time 17.02ms.
    ADC_64SAMP = 14  # 64 samples at 12-bit, conversion time 34.05ms.
    ADC_128SAMP = 15  # 128 samples at 12-bit, conversion time 68.10ms.

    __ADDRESS = 0x40

    __REG_CONFIG = 0x00
    __REG_SHUNTVOLTAGE = 0x01
    __REG_BUSVOLTAGE = 0x02
    __REG_POWER = 0x03
    __REG_CURRENT = 0x04
    __REG_CALIBRATION = 0x05
```

```
__RST = 15
__BRNG = 13
__PG1 = 12
__PG0 = 11
__BADC4 = 10
__BADC3 = 9
__BADC2 = 8
__BADC1 = 7
__SADC4 = 6
__SADC3 = 5
__SADC2 = 4
__SADC1 = 3
__MODE3 = 2
__MODE2 = 1
__MODE1 = 0

__OVF = 1
__CNVR = 2

__BUS_RANGE = [16, 32]
__GAIN_VOLTS = [0.04, 0.08, 0.16, 0.32]

__CONT_SH_BUS = 7

__AMP_ERR_MSG = ('Expected current %.3fA is greater '
                 'than max possible current %.3fA')
__RNG_ERR_MSG = ('Expected amps %.2fA, out of range, use a lower '
                 'value shunt resistor')
__VOLT_ERR_MSG = ('Invalid voltage range, must be one of: '
                  'RANGE_16V, RANGE_32V')

__LOG_FORMAT = '%(asctime)s - %(levelname)s - INA219 %(message)s'
__LOG_MSG_1 = ('shunt ohms: %.3f, bus max volts: %d, '
               'shunt volts max: %.2f%s, '
               'bus ADC: %d, shunt ADC: %d')
__LOG_MSG_2 = ('calibrate called with: bus max volts: %dV, '
               'max shunt volts: %.2fV%s')
__LOG_MSG_3 = ('Current overflow detected - '
               'attempting to increase gain')

__SHUNT_MILLIVOLTS_LSB = 0.01  # 10uV
__BUS_MILLIVOLTS_LSB = 4  # 4mV
__CALIBRATION_FACTOR = 0.04096
__MAX_CALIBRATION_VALUE = 0xFFFE  # Max value supported (65534 decimal)
# In the spec (p17) the current LSB factor for the minimum LSB is
# documented as 32767, but a larger value (100.1% of 32767) is used
# to guarantee that current overflow can always be detected.
__CURRENT_LSB_FACTOR = 32800
```

```
def __init__(self, shunt_ohms, max_expected_amps=None,
             busnum=0, address=__ADDRESS,
             log_level=logging.ERROR):
    """Construct the class.
    Pass in the resistance of the shunt resistor and the maximum expected
    current flowing through it in your system.
    Arguments:
    shunt_ohms -- value of shunt resistor in Ohms (mandatory).
    max_expected_amps -- the maximum expected current in Amps (optional).
    address -- the I2C address of the INA219, defaults
        to *0x40* (optional).
    log_level -- set to logging.DEBUG to see detailed calibration
        calculations (optional).
    """
    if len(logging.getLogger().handlers) == 0:
        # Initialize the root logger only if it hasn't been done yet by a
        # parent module.
        logging.basicConfig(level=log_level, format=self.__LOG_FORMAT)
    self.logger = logging.getLogger(__name__)
    self.logger.setLevel(log_level)

    self._i2c = I2C.get_i2c_device(address=address, busnum=busnum)
    self._shunt_ohms = shunt_ohms
    self._max_expected_amps = max_expected_amps
    self._min_device_current_lsb = self._calculate_min_current_lsb()
    self._gain = None
    self._auto_gain_enabled = False

def configure(self, voltage_range=RANGE_32V, gain=GAIN_AUTO,
              bus_adc=ADC_12BIT, shunt_adc=ADC_12BIT):
    """Configure and calibrate how the INA219 will take measurements.
    Arguments:
    voltage_range -- The full scale voltage range, this is either 16V
        or 32V represented by one of the following constants;
        RANGE_16V, RANGE_32V (default).
    gain -- The gain which controls the maximum range of the shunt
        voltage represented by one of the following constants;
        GAIN_1_40MV, GAIN_2_80MV, GAIN_4_160MV,
        GAIN_8_320MV, GAIN_AUTO (default).
    bus_adc -- The bus ADC resolution (9, 10, 11, or 12-bit) or
        set the number of samples used when averaging results
        represent by one of the following constants; ADC_9BIT,
        ADC_10BIT, ADC_11BIT, ADC_12BIT (default),
        ADC_2SAMP, ADC_4SAMP, ADC_8SAMP, ADC_16SAMP,
        ADC_32SAMP, ADC_64SAMP, ADC_128SAMP
    shunt_adc -- The shunt ADC resolution (9, 10, 11, or 12-bit) or
        set the number of samples used when averaging results
        represent by one of the following constants; ADC_9BIT,
        ADC_10BIT, ADC_11BIT, ADC_12BIT (default),
```

```
            ADC_2SAMP, ADC_4SAMP, ADC_8SAMP, ADC_16SAMP,
            ADC_32SAMP, ADC_64SAMP, ADC_128SAMP
        """
        self.__validate_voltage_range(voltage_range)
        self._voltage_range = voltage_range

        if self._max_expected_amps is not None:
            if gain == self.GAIN_AUTO:
                self._auto_gain_enabled = True
                self._gain = self._determine_gain(self._max_expected_amps)
            else:
                self._gain = gain
        else:
            if gain != self.GAIN_AUTO:
                self._gain = gain
            else:
                self._auto_gain_enabled = True
                self._gain = self.GAIN_1_40MV

        self.logger.info('gain set to %.2fV' % self.__GAIN_VOLTS[self._gain])

        self.logger.debug(
            self.__LOG_MSG_1 %
            (self._shunt_ohms, self.__BUS_RANGE[voltage_range],
             self.__GAIN_VOLTS[self._gain],
             self.__max_expected_amps_to_string(self._max_expected_amps),
             bus_adc, shunt_adc))

        self._calibrate(
            self.__BUS_RANGE[voltage_range], self.__GAIN_VOLTS[self._gain],
            self._max_expected_amps)
        self._configure(voltage_range, self._gain, bus_adc, shunt_adc)

    def voltage(self):
        """Return the bus voltage in volts."""
        value = self._voltage_register()
        return float(value) * self.__BUS_MILLIVOLTS_LSB / 1000

    def supply_voltage(self):
        """Return the bus supply voltage in volts.
        This is the sum of the bus voltage and shunt voltage. A
        DeviceRangeError exception is thrown if current overflow occurs.
        """
        return self.voltage() + (float(self.shunt_voltage()) / 1000)

    def current(self):
        """Return the bus current in milliamps.
        A DeviceRangeError exception is thrown if current overflow occurs.
        """
```

```
        self._handle_current_overflow()
        return self._current_register() * self._current_lsb * 1000

    def power(self):
        """Return the bus power consumption in milliwatts.
        A DeviceRangeError exception is thrown if current overflow occurs.
        """
        self._handle_current_overflow()
        return self._power_register() * self._power_lsb * 1000

    def shunt_voltage(self):
        """Return the shunt voltage in millivolts.
        A DeviceRangeError exception is thrown if current overflow occurs.
        """
        self._handle_current_overflow()
        return self._shunt_voltage_register() * self.__SHUNT_MILLIVOLTS_LSB

    def sleep(self):
        """Put the INA219 into power down mode."""
        configuration = self._read_configuration()
        self._configuration_register(configuration & 0xFFF8)

    def wake(self):
        """Wake the INA219 from power down mode."""
        configuration = self._read_configuration()
        self._configuration_register(configuration | 0x0007)
        # 40us delay to recover from powerdown (p14 of spec)
        time.sleep(0.00004)

    def current_overflow(self):
        """Return true if the sensor has detect current overflow.
        In this case the current and power values are invalid.
        """
        return self._has_current_overflow()

    def reset(self):
        """Reset the INA219 to its default configuration."""
        self._configuration_register(1 << self.__RST)

    def is_conversion_ready(self):
        """Check if conversion of a new reading has occured."""
        cnvr = self._read_voltage_register() & self.__CNVR
        return (cnvr == self.__CNVR)

    def _handle_current_overflow(self):
        if self._auto_gain_enabled:
            while self._has_current_overflow():
                self._increase_gain()
        else:
```

```
            if self._has_current_overflow():
                raise DeviceRangeError(self.__GAIN_VOLTS[self._gain])

    def _determine_gain(self, max_expected_amps):
        shunt_v = max_expected_amps * self._shunt_ohms
        if shunt_v > self.__GAIN_VOLTS[3]:
            raise ValueError(self.__RNG_ERR_MSG % max_expected_amps)
        gain = min(v for v in self.__GAIN_VOLTS if v > shunt_v)
        return self.__GAIN_VOLTS.index(gain)

    def _increase_gain(self):
        self.logger.info(self.__LOG_MSG_3)
        gain = self._read_gain()
        if gain < len(self.__GAIN_VOLTS) - 1:
            gain = gain + 1
            self._calibrate(self.__BUS_RANGE[self._voltage_range],
                            self.__GAIN_VOLTS[gain])
            self._configure_gain(gain)
            # 1ms delay required for new configuration to take effect,
            # otherwise invalid current/power readings can occur.
            time.sleep(0.001)
        else:
            self.logger.info('Device limit reach, gain cannot be increased')
            raise DeviceRangeError(self.__GAIN_VOLTS[gain], True)

    def _configure(self, voltage_range, gain, bus_adc, shunt_adc):
        configuration = (
            voltage_range << self.__BRNG | gain << self.__PG0 |
            bus_adc << self.__BADC1 | shunt_adc << self.__SADC1 |
            self.__CONT_SH_BUS)
        self._configuration_register(configuration)

    def _calibrate(self, bus_volts_max, shunt_volts_max,
                   max_expected_amps=None):
        self.logger.info(
            self.__LOG_MSG_2 %
            (bus_volts_max, shunt_volts_max,
             self.__max_expected_amps_to_string(max_expected_amps)))

        max_possible_amps = shunt_volts_max / self._shunt_ohms

        self.logger.info("max possible current: %.3fA" %
                         max_possible_amps)

        self._current_lsb = \
            self._determine_current_lsb(max_expected_amps, max_possible_amps)
        self.logger.info("current LSB: %.3e A/bit" % self._current_lsb)

        self._power_lsb = self._current_lsb * 20
```

```
        self.logger.info("power LSB: %.3e W/bit" % self._power_lsb)

        max_current = self._current_lsb * 32767
        self.logger.info("max current before overflow: %.4fA" % max_current)

        max_shunt_voltage = max_current * self._shunt_ohms
        self.logger.info("max shunt voltage before overflow: %.4fmV" %
                         (max_shunt_voltage * 1000))

        calibration = trunc(self.__CALIBRATION_FACTOR /
                            (self._current_lsb * self._shunt_ohms))
        self.logger.info(
            "calibration: 0x%04x (%d)" % (calibration, calibration))
        self._calibration_register(calibration)

    def _determine_current_lsb(self, max_expected_amps, max_possible_amps):
        if max_expected_amps is not None:
            if max_expected_amps > round(max_possible_amps, 3):
                raise ValueError(self.__AMP_ERR_MSG %
                                 (max_expected_amps, max_possible_amps))
            self.logger.info("max expected current: %.3fA" %
                             max_expected_amps)
            if max_expected_amps < max_possible_amps:
                current_lsb = max_expected_amps / self.__CURRENT_LSB_FACTOR
            else:
                current_lsb = max_possible_amps / self.__CURRENT_LSB_FACTOR
        else:
            current_lsb = max_possible_amps / self.__CURRENT_LSB_FACTOR

        if current_lsb < self._min_device_current_lsb:
            current_lsb = self._min_device_current_lsb
        return current_lsb

    def _configuration_register(self, register_value):
        self.logger.debug("configuration: 0x%04x" % register_value)
        self.__write_register(self.__REG_CONFIG, register_value)

    def _read_configuration(self):
        return self.__read_register(self.__REG_CONFIG)

    def _calculate_min_current_lsb(self):
        return self.__CALIBRATION_FACTOR / \
            (self._shunt_ohms * self.__MAX_CALIBRATION_VALUE)

    def _read_gain(self):
        configuration = self._read_configuration()
        gain = (configuration & 0x1800) >> self.__PG0
        self.logger.info("gain is currently: %.2fV" % self.__GAIN_VOLTS[gain])
        return gain
```

```
def _configure_gain(self, gain):
    configuration = self._read_configuration()
    configuration = configuration & 0xE7FF
    self._configuration_register(configuration | (gain << self.__PG0))
    self._gain = gain
    self.logger.info("gain set to: %.2fV" % self.__GAIN_VOLTS[gain])

def _calibration_register(self, register_value):
    self.logger.debug("calibration: 0x%04x" % register_value)
    self.__write_register(self.__REG_CALIBRATION, register_value)

def _has_current_overflow(self):
    ovf = self._read_voltage_register() & self.__OVF
    return (ovf == 1)

def _voltage_register(self):
    register_value = self._read_voltage_register()
    return register_value >> 3

def _read_voltage_register(self):
    return self.__read_register(self.__REG_BUSVOLTAGE)

def _current_register(self):
    return self.__read_register(self.__REG_CURRENT, True)

def _shunt_voltage_register(self):
    return self.__read_register(self.__REG_SHUNTVOLTAGE, True)

def _power_register(self):
    return self.__read_register(self.__REG_POWER)

def __validate_voltage_range(self, voltage_range):
    if voltage_range > len(self.__BUS_RANGE) - 1:
        raise ValueError(self.__VOLT_ERR_MSG)

def __write_register(self, register, register_value):
    register_bytes = self.__to_bytes(register_value)
    self.logger.debug(
        "write register 0x%02x: 0x%04x 0b%s" %
        (register, register_value,
         self.__binary_as_string(register_value)))
    self._i2c.writeList(register, register_bytes)

def __read_register(self, register, negative_value_supported=False):
    if negative_value_supported:
        register_value = self._i2c.readS16BE(register)
    else:
        register_value = self._i2c.readU16BE(register)
```

```
        self.logger.debug(
            "read register 0x%02x: 0x%04x 0b%s" %
            (register, register_value,
             self.__binary_as_string(register_value)))
        return register_value

    def __to_bytes(self, register_value):
        return [(register_value >> 8) & 0xFF, register_value & 0xFF]

    def __binary_as_string(self, register_value):
        return bin(register_value)[2:].zfill(16)

    def __max_expected_amps_to_string(self, max_expected_amps):
        if max_expected_amps is None:
            return ''
        else:
            return ', max expected amps: %.3fA' % max_expected_amps


class DeviceRangeError(Exception):
    """Class containing the INA219 error functionality."""

    __DEV_RNG_ERR = ('Current out of range (overflow), '
                     'for gain %.2fV')

    def __init__(self, gain_volts, device_max=False):
        """Construct a DeviceRangeError."""
        msg = self.__DEV_RNG_ERR % gain_volts
        if device_max:
            msg = msg + ', device limit reached'
        super(DeviceRangeError, self).__init__(msg)
        self.gain_volts = gain_volts
        self.device_limit_reached = device_max


SHUNT_OHMS = 0.1
MAX_EXPECTED_AMPS = 0.2


def read():
    ina = INA219(SHUNT_OHMS, MAX_EXPECTED_AMPS, log_level=logging.INFO)
    ina.configure(ina.RANGE_16V, ina.GAIN_2_80MV)

    print("Bus Voltage    : %.3f V" % ina.voltage())
    print("Bus Current    : %.3f mA" % ina.current())
    print("Supply Voltage : %.3f V" % ina.supply_voltage())
    print("Shunt voltage  : %.3f mV" % ina.shunt_voltage())
    print("Power          : %.3f mW" % ina.power())
```

```
if __name__ == "__main__":
    read()
```

A.1.10. Programa mpu6050.py

```
import smbus
import time
import math



class mpu6050:

    # Global Variables
    GRAVITIY_MS2 = 9.80665
    address = None
    bus = None

    # Scale Modifiers
    ACCEL_SCALE_MODIFIER_2G = 16384.0
    ACCEL_SCALE_MODIFIER_4G = 8192.0
    ACCEL_SCALE_MODIFIER_8G = 4096.0
    ACCEL_SCALE_MODIFIER_16G = 2048.0

    GYRO_SCALE_MODIFIER_250DEG = 131.0
    GYRO_SCALE_MODIFIER_500DEG = 65.5
    GYRO_SCALE_MODIFIER_1000DEG = 32.8
    GYRO_SCALE_MODIFIER_2000DEG = 16.4

    # Pre-defined ranges
    ACCEL_RANGE_2G = 0x00
    ACCEL_RANGE_4G = 0x08
    ACCEL_RANGE_8G = 0x10
    ACCEL_RANGE_16G = 0x18

    GYRO_RANGE_250DEG = 0x00
    GYRO_RANGE_500DEG = 0x08
    GYRO_RANGE_1000DEG = 0x10
    GYRO_RANGE_2000DEG = 0x18

    # MPU-6050 Registers
    PWR_MGMT_1 = 0x6B
    PWR_MGMT_2 = 0x6C

    ACCEL_XOUT0 = 0x3B
```

```
    ACCEL_YOUT0 = 0x3D
    ACCEL_ZOUT0 = 0x3F

    TEMP_OUT0 = 0x41
    TEMP_OUT1 = 0x42

    GYRO_XOUT0 = 0x43
    GYRO_YOUT0 = 0x45
    GYRO_ZOUT0 = 0x47

    ACCEL_CONFIG = 0x1C
    GYRO_CONFIG = 0x1B

    def __init__(self, address, bus=0):
        self.address = address
        self.bus = smbus.SMBus(bus)
        # Wake up the MPU-6050 since it starts in sleep mode
        self.bus.write_byte_data(self.address, self.PWR_MGMT_1, 0x00)

    # I2C communication methods

    def read_i2c_word(self, register):
        # Read the data from the registers
        high = self.bus.read_byte_data(self.address, register)
        low = self.bus.read_byte_data(self.address, register + 1)

        value = (high << 8) + low

        if (value >= 0x8000):
            return -((65535 - value) + 1)
        else:
            return value

    def set_accel_range(self, accel_range):
        # First change it to 0x00 to make sure we write the correct value
later
        self.bus.write_byte_data(self.address, self.ACCEL_CONFIG, 0x00)

        # Write the new range to the ACCEL_CONFIG register
        self.bus.write_byte_data(self.address, self.ACCEL_CONFIG, accel_range)

    def read_accel_range(self, raw = False):
        raw_data = self.bus.read_byte_data(self.address, self.ACCEL_CONFIG)

        if raw is True:
            return raw_data
        elif raw is False:
            if raw_data == self.ACCEL_RANGE_2G:
                return 2
```

```
            elif raw_data == self.ACCEL_RANGE_4G:
                return 4
            elif raw_data == self.ACCEL_RANGE_8G:
                return 8
            elif raw_data == self.ACCEL_RANGE_16G:
                return 16
            else:
                return -1

    def get_accel_data(self, g = False):
        x = self.read_i2c_word(self.ACCEL_XOUT0)
        y = self.read_i2c_word(self.ACCEL_YOUT0)
        z = self.read_i2c_word(self.ACCEL_ZOUT0)

        accel_scale_modifier = None
        accel_range = self.read_accel_range(True)

        if accel_range == self.ACCEL_RANGE_2G:
            accel_scale_modifier = self.ACCEL_SCALE_MODIFIER_2G
        elif accel_range == self.ACCEL_RANGE_4G:
            accel_scale_modifier = self.ACCEL_SCALE_MODIFIER_4G
        elif accel_range == self.ACCEL_RANGE_8G:
            accel_scale_modifier = self.ACCEL_SCALE_MODIFIER_8G
        elif accel_range == self.ACCEL_RANGE_16G:
            accel_scale_modifier = self.ACCEL_SCALE_MODIFIER_16G
        else:
            print("Unknown range-accel_scale_modifier set to
self.ACCEL_SCALE_MODIFIER_2G")
            accel_scale_modifier = self.ACCEL_SCALE_MODIFIER_2G

        x = x / accel_scale_modifier
        y = y / accel_scale_modifier
        z = z / accel_scale_modifier

        if g is True:
            return {'x': x, 'y': y, 'z': z}
        elif g is False:
            x = x * self.GRAVITIY_MS2
            y = y * self.GRAVITIY_MS2
            z = z * self.GRAVITIY_MS2
            return {'x': x, 'y': y, 'z': z}

    def set_gyro_range(self, gyro_range):
        # First change it to 0x00 to make sure we write the correct value
later
        self.bus.write_byte_data(self.address, self.GYRO_CONFIG, 0x00)

        # Write the new range to the ACCEL_CONFIG register
        self.bus.write_byte_data(self.address, self.GYRO_CONFIG, gyro_range)
```

```
def read_gyro_range(self, raw = False):
    raw_data = self.bus.read_byte_data(self.address, self.GYRO_CONFIG)

    if raw is True:
        return raw_data
    elif raw is False:
        if raw_data == self.GYRO_RANGE_250DEG:
            return 250
        elif raw_data == self.GYRO_RANGE_500DEG:
            return 500
        elif raw_data == self.GYRO_RANGE_1000DEG:
            return 1000
        elif raw_data == self.GYRO_RANGE_2000DEG:
            return 2000

        else:
            return -1

def get_gyro_data(self):
    x = self.read_i2c_word(self.GYRO_XOUT0)
    y = self.read_i2c_word(self.GYRO_YOUT0)
    z = self.read_i2c_word(self.GYRO_ZOUT0)

    gyro_scale_modifier = None
    gyro_range = self.read_gyro_range(True)

    if gyro_range == self.GYRO_RANGE_250DEG:
        gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_250DEG
    elif gyro_range == self.GYRO_RANGE_500DEG:
        gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_500DEG
    elif gyro_range == self.GYRO_RANGE_1000DEG:
        gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_1000DEG
    elif gyro_range == self.GYRO_RANGE_2000DEG:
        gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_2000DEG
    else:
        gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_250DEG

    x = x / gyro_scale_modifier
    y = y / gyro_scale_modifier
    z = self.read_i2c_word(self.GYRO_ZOUT0)

    gyro_scale_modifier = None
    gyro_range = self.read_gyro_range(True)

    if gyro_range == self.GYRO_RANGE_250DEG:
        gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_250DEG
    elif gyro_range == self.GYRO_RANGE_500DEG:
        gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_500DEG
```

```
        elif gyro_range == self.GYRO_RANGE_1000DEG:
            gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_1000DEG
        elif gyro_range == self.GYRO_RANGE_2000DEG:
            gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_2000DEG
        else:
            gyro_scale_modifier = self.GYRO_SCALE_MODIFIER_250DEG

        x = x / gyro_scale_modifier
        y = y / gyro_scale_modifier
        z = z / gyro_scale_modifier

        return {'x': x, 'y': y, 'z': z}

    def get_all_data(self):
        temp = self.get_temp()
        accel = self.get_accel_data()
        gyro = self.get_gyro_data()

        return [accel, gyro, temp]

    def get_temp(self):
        """Reads the temperature from the onboard temperature sensor of the
MPU-6050.
        Returns the temperature in degrees Celcius.
        """
        # Get the raw data
        raw_temp = self.read_i2c_word(self.TEMP_OUT0)

        # Get the actual temperature using the formule given in the
        # MPU-6050 Register Map and Descriptions revision 4.2, page 30
        actual_temp = (raw_temp / 340) + 36.53

        # Return the temperature
        return actual_temp

    def dadesMPU(self):
       accel_data = mpu.get_accel_data()
       gyro_data = mpu.get_gyro_data()
       temp = mpu.get_temp()
       radiansy = math.atan2(accel_data['x'], math.sqrt(accel_data['y'] *
accel_data['y'] + accel_data['z'] * accel_data['z']))
       roll =  math.degrees (radiansy)
       radiansx = math.atan2(accel_data['y'], math.sqrt(accel_data['x'] *
accel_data['x'] + accel_data['z'] * accel_data['z']))
       pitch =  math.degrees (radiansx)
       pitch = int(pitch)
       roll = int(roll)
       temp = int(temp)
       return [roll, pitch, 0, temp]
```

```
mpu = mpu6050(0x68)
roll = 0
pitch = 0
if __name__ == "__main__":
    while(1):
        try:
           rollant = roll
           pitchant = pitch
           accel_data = mpu.get_accel_data()
           gyro_data = mpu.get_gyro_data()
           temp = mpu.get_temp()
           radiansy = math.atan2(accel_data['x'], math.sqrt(accel_data['y'] *
accel_data['y'] + accel_data['z'] * accel_data['z']))
           roll =  math.degrees (radiansy)
           radiansx = math.atan2(accel_data['y'], math.sqrt(accel_data['x'] *
accel_data['x'] + accel_data['z'] * accel_data['z']))
           pitch =  math.degrees (radiansx)
        except KeyboardInterrupt:
            break

        time.sleep(0.5)
```

A.1.11. Programa ms5837.py

```
try:
    import smbus2 as smbus
except:
    print('Try sudo apt-get install python-smbus2')

from time import sleep

# Models
MODEL_02BA = 0
MODEL_30BA = 1

# Oversampling options
OSR_256  = 0
OSR_512  = 1
OSR_1024 = 2
OSR_2048 = 3
OSR_4096 = 4
OSR_8192 = 5

# kg/m^3 convenience
DENSITY_FRESHWATER = 997
```

```
DENSITY_SALTWATER = 1029

# Conversion factors (from native unit, mbar)
UNITS_Pa     = 100.0
UNITS_hPa    = 1.0
UNITS_kPa    = 0.1
UNITS_mbar   = 1.0
UNITS_bar    = 0.001
UNITS_atm    = 0.000986923
UNITS_Torr   = 0.750062
UNITS_psi    = 0.014503773773022

# Valid units
UNITS_Centigrade = 1
UNITS_Farenheit  = 2
UNITS_Kelvin     = 3


class MS5837(object):

    # Registers
    _MS5837_ADDR            = 0x76
    _MS5837_RESET           = 0x1E
    _MS5837_ADC_READ        = 0x00
    _MS5837_PROM_READ       = 0xA0
    _MS5837_CONVERT_D1_256  = 0x40
    _MS5837_CONVERT_D2_256  = 0x50

    def __init__(self, model=MODEL_30BA, bus=1):
        self._model = model

        try:
            self._bus = smbus.SMBus(bus)
        except:
            print("Bus %d is not available."%bus)
            print("Available busses are listed as /dev/i2c*")
            self._bus = None

        self._fluidDensity = DENSITY_FRESHWATER
        self._pressure = 0
        self._temperature = 0
        self._D1 = 0
        self._D2 = 0

    def init(self):
        if self._bus is None:
            "No bus!"
            return False
```

```
        self._bus.write_byte(self._MS5837_ADDR, self._MS5837_RESET)

        # Wait for reset to complete
        sleep(0.01)

        self._C = []

        # Read calibration values and CRC
        for i in range(7):
            c = self._bus.read_word_data(self._MS5837_ADDR,
self._MS5837_PROM_READ + 2*i)
            c =  ((c & 0xFF) << 8) | (c >> 8) # SMBus is little-endian for
word transfers, we need to swap MSB and LSB
            self._C.append(c)

        crc = (self._C[0] & 0xF000) >> 12
        if crc != self._crc4(self._C):
            print("PROM read error, CRC failed!")
            return False

        return True

    def read(self, oversampling=OSR_8192):
        if self._bus is None:
            print("No bus!")
            return False

        if oversampling < OSR_256 or oversampling > OSR_8192:
            print("Invalid oversampling option!")
            return False

        # Request D1 conversion (pressure)
        self._bus.write_byte(self._MS5837_ADDR, self._MS5837_CONVERT_D1_256 +
2*oversampling)

        # Maximum conversion time increases linearly with oversampling
        # max time (seconds) ~= 2.2e-6(x) where x = OSR = (2^8, 2^9, ...,
2^13)
        # We use 2.5e-6 for some overhead
        sleep(2.5e-6 * 2**(8+oversampling))

        d = self._bus.read_i2c_block_data(self._MS5837_ADDR,
self._MS5837_ADC_READ, 3)
        self._D1 = d[0] << 16 | d[1] << 8 | d[2]

        # Request D2 conversion (temperature)
        self._bus.write_byte(self._MS5837_ADDR, self._MS5837_CONVERT_D2_256 +
2*oversampling)
```

```
        # As above
        sleep(2.5e-6 * 2**(8+oversampling))

        d = self._bus.read_i2c_block_data(self._MS5837_ADDR,
self._MS5837_ADC_READ, 3)
        self._D2 = d[0] << 16 | d[1] << 8 | d[2]

        # Calculate compensated pressure and temperature
        # using raw ADC values and internal calibration
        self._calculate()

        return True

    def setFluidDensity(self, denisty):
        self._fluidDensity = denisty

    # Pressure in requested units
    # mbar * conversion
    def pressure(self, conversion=UNITS_mbar):
        return self._pressure * conversion

    # Temperature in requested units
    # default degrees C
    def temperature(self, conversion=UNITS_Centigrade):
        degC = self._temperature / 100.0
        if conversion == UNITS_Farenheit:
            return (9.0/5.0)*degC + 32
        elif conversion == UNITS_Kelvin:
            return degC + 273
        return degC

    # Depth relative to MSL pressure in given fluid density
    def depth(self):
        return (self.pressure(UNITS_Pa)-101300)/(self._fluidDensity*9.80665)

    # Altitude relative to MSL pressure
    def altitude(self):
        return (1-pow((self.pressure()/1013.25),.190284))*145366.45*.3048

    # Cribbed from datasheet
    def _calculate(self):
        OFFi = 0
        SENSi = 0
        Ti = 0

        dT = self._D2-self._C[5]*256
        if self._model == MODEL_02BA:
            SENS = self._C[1]*65536+(self._C[3]*dT)/128
            OFF = self._C[2]*131072+(self._C[4]*dT)/64
```

```
            self._pressure = (self._D1*SENS/(2097152)-OFF)/(32768)
        else:
            SENS = self._C[1]*32768+(self._C[3]*dT)/256
            OFF = self._C[2]*65536+(self._C[4]*dT)/128
            self._pressure = (self._D1*SENS/(2097152)-OFF)/(8192)

        self._temperature = 2000+dT*self._C[6]/8388608

        # Second order compensation
        if self._model == MODEL_02BA:
            if (self._temperature/100) < 20: # Low temp
                Ti = (11*dT*dT)/(34359738368)
                OFFi = (31*(self._temperature-2000)*(self._temperature-
2000))/8
                SENSi = (63*(self._temperature-2000)*(self._temperature-
2000))/32

        else:
            if (self._temperature/100) < 20: # Low temp
                Ti = (3*dT*dT)/(8589934592)
                OFFi = (3*(self._temperature-2000)*(self._temperature-2000))/2
                SENSi = (5*(self._temperature-2000)*(self._temperature-
2000))/8
                if (self._temperature/100) < -15: # Very low temp
                    OFFi =
OFFi+7*(self._temperature+1500)*(self._temperature+1500)
                    SENSi =
SENSi+4*(self._temperature+1500)*(self._temperature+1500)
            elif (self._temperature/100) >= 20: # High temp
                Ti = 2*(dT*dT)/(137438953472)
                OFFi = (1*(self._temperature-2000)*(self._temperature-
2000))/16
                SENSi = 0

        OFF2 = OFF-OFFi
        SENS2 = SENS-SENSi

        if self._model == MODEL_02BA:
            self._temperature = (self._temperature-Ti)
            self._pressure = (((self._D1*SENS2)/2097152-OFF2)/32768)/100.0
        else:
            self._temperature = (self._temperature-Ti)
            self._pressure = (((self._D1*SENS2)/2097152-OFF2)/8192)/10.0

    # Cribbed from datasheet
    def _crc4(self, n_prom):
        n_rem = 0

        n_prom[0] = ((n_prom[0]) & 0x0FFF)
```

```
        n_prom.append(0)

        for i in range(16):
            if i%2 == 1:
                n_rem ^= ((n_prom[i>>1]) & 0x00FF)
            else:
                n_rem ^= (n_prom[i>>1] >> 8)

            for n_bit in range(8,0,-1):
                if n_rem & 0x8000:
                    n_rem = (n_rem << 1) ^ 0x3000
                else:
                    n_rem = (n_rem << 1)

        n_rem = ((n_rem >> 12) & 0x000F)

        self.n_prom = n_prom
        self.n_rem = n_rem

        return n_rem ^ 0x00

class MS5837_30BA(MS5837):
    def __init__(self, bus=1):
        MS5837.__init__(self, MODEL_30BA, bus)

class MS5837_02BA(MS5837):
    def __init__(self, bus=1):
        MS5837.__init__(self, MODEL_02BA, bus)
```

A.1.12. Programa tracking.py

```
import cv2
import numpy as np
import paho.mqtt.client as mqtt
import json
from time import sleep
from config import host


#################################################################
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        client.connected_flag=True
        print("connected OK Returned code=",rc)
    else:
        print("Bad connection Returned code=",rc)
```

```
        client.bad_connection_flag=True

def connect_mqtt():
    client = mqtt.Client(client_id)
    client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.connect(host, port)
    return client

#################################################################
port = 1883

client_id = "stalker"
username = "Stalker"
password = "password"
topic = "Motors"
mbaix=False
mdalt=False
#################################################################
client = connect_mqtt()
client.loop_start()

#cap = cv2.VideoCapture("http://localhost:8090/?action=stream")
grocBajo = np.array([15,100,20],np.uint8)
grocAlto = np.array([45,255,255],np.uint8)
msg_cache= None
while True:
  cap = cv2.VideoCapture("http://localhost:8090/?action=stream")
  ret,frame = cap.read()
  if ret==True:
    frameHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(frameHSV,grocBajo,grocAlto)
    contornos,hierachy = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    #cv2.drawContours(frame, contornos, -1, (255,0,0), 3)
    areagran=False
    for c in contornos:
      area = cv2.contourArea(c)
      if area > 500:
        M = cv2.moments(c)
        if (M["m00"]==0): M["m00"]=1
        x = int(M["m10"]/M["m00"])
        y = int(M['m01']/M['m00'])
        cv2.circle(frame, (x,y), 7, (0,255,0), -1)
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame, '{},{}'.format(x,y),(x+10,y), font,
0.75,(0,255,0),1,cv2.LINE_AA)
        nuevoContorno = cv2.convexHull(c)
        cv2.drawContours(frame, [nuevoContorno], 0, (255,0,0), 3)
```

```
        if (area>30000):
            msg = "bkd"
            client.publish(topic, json.dumps([msg, 0.15, 0.15]))
            print('enrera')
        elif (area>10000):
            msg = "stop"
            client.publish(topic, json.dumps([msg]))
            print('stop')
        elif (x>850):
            msg = "fwd"
            client.publish(topic, json.dumps([msg, 0.3, 0.15]))
            print('dreta')
            mbaix=True
        elif (x<500):
            msg = "fwd"
            client.publish(topic, json.dumps([msg, 0.15, 0.3]))
            print('esquerra')
            mbaix=True
        elif (y>600):
            msg = "dwn"
            client.publish(topic, json.dumps([msg, 0.15, 0.15]))
            print('avall')
            mdalt=True
        elif (y<300):
            msg = "up"
            client.publish(topic, json.dumps([msg, 0.15, 0.15]))
            print('amunt')
            mbaix=True
        else :
            msg = "fwd"
            client.publish(topic, json.dumps([msg, 0.15, 0.15]))
            print('endavant')
            mdalt=True
        areagran=True
        break

    if areagran==False or (mbaix==True and mdalt==True):
      msg = "stop"
      client.publish(topic, json.dumps([msg]))
      print("stop no troba")
      mdalt=False
      mbaix=False
    #cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('s'):
      break

cap.release()
cv2.destroyAllWindows()
```

A.1.13. Programa config.py

```
host = "169.254.10.124" # IP of the broker MQTT device.
#host = 'localhost'     # Local broker.
```

A.2. Interfície web

Per a programar la interfície web es necessiten diferents llenguatges amb diferents programes cadascun. Només cal executar el primer fitxer "index.html" per a fer funcionar aquesta interfície i comunicar-se amb el Girona 25.

A.2.1. Programa "index.html"

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="author" content="Roger Feliu Serramitja">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Robot submarí GIRONA 25</title>

    <!-- CSS -->
    <link rel="stylesheet" href="css/style.css">
    <link rel="stylesheet" href="css/button.css">
    <link rel="stylesheet" href="css/loading.css">
    <link rel="stylesheet" href="css/gpcon.css">

    <!-- JS -->
    <script src="js/paho.javascript-1.0.3/paho-mqtt.js"
type="text/javascript"></script>
    <script src="js/intro_host.js" type="text/javascript"></script>
    <script src="js/motor.js"></script>
    <script src="js/ui.js" type="text/javascript"></script>
    <script src="js/keyboard.js" type="text/javascript"></script>
    <script src="js/joystick.js" type="text/javascript"></script>
    <script src="js/mqtt.js" type="text/javascript"></script>
    <script src="js/stream.js"></script>
    <script src="js/libhtml.js"></script>
    <script src="js/libgp.js"></script>
    <script src="js/gpcon.js"></script>
    <!--
```

```
    <style>
      #ultim + div{
          visibility: hidden;
      }
  </style>
    -->
  </head>
  <body>
    <div class="loading-overlay" id="loading-overlay">
      <div class="center">
        <div style="margin-top: 3em">
          <div class="loading-container">
            <div class="loading"></div>
            <div id="loading-text">Connecting</div>
          </div>
        </div>
      </div>
    </div>

    <div class="button-panel top-left estat-cont" id="estat-cont">
      <div class="center">
        <div id="estat" class="estat">
          A message goes here. にゃー
        </div>
      </div>
    </div>

    <div class="pantalla center" style="display: flex">
      <div>
        <h1>Robot submarí</h1>
        <div>Loading ^.^</div>
      </div>
    </div>

    <div class="pantalla" id="intro">
      <div class="center intro">
        <div class="login-cont">
          <img src="images/LOGO.png" style="top: 10%; left: 29%; position:
fixed; z-index:-1; width: 500 px; height: 200px;"/>

          <div style="height: 2em"></div>
          <div style="height: 2em"></div>

          <input class= "hide" type="text" id="host-ip" placeholder="Host o IP
del submarí"> <!-- class= "hide" -->

          <div style="height: 2em"></div>
          <div style="height: 2em"></div>
          <div style="height: 2em"></div>
```

```
          <div style="height: 2em"></div>

          <button class="login" id="submit-ip" onclick="start_connection()">
            Connectar
          </button>
        </div>
      </div>
      <div class="button-panel top-left">
         <button class="motor" id="fullscreen" onclick="change_fullscreen()"
title="FullScreen" style="font-size: 2.5vw; padding-left: .1vw;"
value="close">⇱</button>
      </div>
    </div>

    <div class="main-content pantalla" id="panel">
      <div>
        <div class="stream" oncontextmenu="disable_contextmenu()">
              <img
            class="interface"
            id="streaming"
            src="images/stream_off.png"
            alt="Not image available"
            onerror="stream_error()"
          />
          <img
            class="hide interfaceDrt"
            id="streaming1"
            src="images/stream_off.png"
            alt="Not image available"
            onerror="stream_error()"
          />
          <img
            class="hide interfaceEsq"
            id="streaming2"
            src="images/stream_off.png"
            alt="Not image available"
            onerror="stream_error()"
          />
        </div>
        <div class="button-panel center-left mobile">
          <div>
            <div class="space">
            </div>
            <button style="margin-bottom: 10px;" class="motor move ull"
id="up" ontouchstart="emerge()" ontouchend="stop()" title="Emerge">↑</button>
          </div>
          <button class="motor move ull" ontouchstart="left()"
ontouchend="stop()" title="Left">←</button>
          <div class="space2">
```

```
          </div>
          <button class="motor move ull" ontouchstart="right()"
ontouchend="stop()" title="Right">→</button>
          <div>
            <div class="space">
            </div>
               <button style="margin-top: 10px;" class="motor move ull"
id="down" ontouchstart="immerse()" ontouchend="stop()"
title="Immerse">↓</button>
               </div>
          </div>

        <div class="button-panel bottom-left horitzontal">
    <button class="motor mobile" id= "ulleres" onclick="realitat()"
title="VR">□</button>
         <button class="motor follow ull" id="follow-button" onclick="follow()"
title="Follow">🎈</button>
    <button class="motor hold move ull" id="hold-button"
onclick="depth_hold()" title="Depth hold">🚢</button>
         <button class="motor streaming ull" id="stream-button"
onclick="stream()" title="Play/pause Stream">📺</button>
         <button class="motor camera ull" onclick="stream_screenshot()"
title="Capture shot">📷</button>
    <button class="motor video ull" id="video-button" onclick="stream_video()"
title="Film">📽</button>
    <button class="motor ull" id= "light-button" onclick="light()"
title="Light">🔦</button>
    <button class="motor move ull" id= "velocity-button" onclick="velocity()"
title="Velocitat">V1</button>
        </div>
        <div class="button-panel bottom-right horitzontal">
    <button class="motor stop move ull" id= "stop" onclick="stop_raspy()"
title="Stop">⬤</button>
        </div>

        <div class="button-panel center-right move mobile">
          <canvas id="joystick" class="joystick ull">
            Wops. Canvas unsuported.
          </canvas>
        </div>

        <div class="button-panel top-left">
         <button class="motor ull" id="fullscreen"
onclick="change_fullscreen()" title="FullScreen" style="font-size: 2.5vw;
padding-left: .1vw;" value="close">⇱</button>
    <button class="motor ull" id="helpbut" onclick="ui_help()"
title="Help">?</button>
        </div>
```

```
    <div class="help" id="help">
      <div class="center">
        <div class="helpbox">
          <h1 class="banner">GIRONA 25</h1>
          <div>Dreceres de teclat:</div>
          <div id="help-keyboard" class="help-keyboard">
          </div>
        </div>
      </div>
    </div>

    <div>
<img class= "imgA1" src="images/INC_BRU.PNG" id="imagetot">
<img class= "imgA2" src="images/ROLL.png" id="imageroll">
<img class= "imgA3" src="images/BRUIX.png" id="imagebruix">
<img class= "imgA4" src="images/PITCH.png" id="imagepitch">
<h1 class= "Temp">  Temperatura: <span id="temp"></span> °C </h1>
<h1 class= "deep">  Profunditat: <span id="deep"></span> m </h1>
<h1 class= "BAT">  <span id="Nbat"></span> % </h1>
<img class= "imgB0" src="images/Bateria0.png" id="imageB0">
<img class= "imgB0" src="images/Bateria25.png" id="imageB25">
<img class= "imgB0" src="images/Bateria50.png" id="imageB50">
<img class= "imgB0" src="images/Bateria75.png" id="imageB75">
<img class= "imgB0" src="images/Bateria100.png" id="imageB100">
    </div>

  </div>

</div>

<div id="sol" class="nodisp"></div>

<div id="prompt" class="nodisp">●</div>

<div id="main" class="nodisp">
  <div class="motor">□</div>
    <div id="button-bar">
        <div id="button-bar-box" class="nodisp"></div>
    </div>
    <div id="gamepad-container">
    </div>
</div>

<!-- Templates are copied for use when needed -->
<div id="templates">
    <input id="template-button" type="button" class="selector-button">

    <div id="template-gamepad" class="gamepad nodisp">
        <div class="gamepad-title"></div>
```

```
            <div class="gamepad-mapping"></div>
            <div class="gamepad-id"></div>
            <div class="gamepad-controls-center">
                <div class="gamepad-controls">
                    <div class="gamepad-buttons-box"></div>
                    <div class="gamepad-axes-box"></div>
                </div>
            </div>
        </div>

        <div id="template-gamepad-button-container" class="gamepad-button-
container">
            <div class="gamepad-button"></div>
            <div class="gamepad-button-label"></div>
        </div>

        <div id="template-gamepad-axis-pair-container" class="gamepad-axis-
pair-container">
            <div class="gamepad-axis-pair">
                <div class="gamepad-circle nodisp"></div>
                <div class="gamepad-axis-pip"></div>
                <div class="gamepad-axis-crosshair">
                    <div class="gamepad-axis-crosshair-v"></div>
                    <div class="gamepad-axis-crosshair-h"></div>
                </div>
            </div>
            <div class="gamepad-axis-pair-label"></div>
            <div class="gamepad-axis-pair-value"></div>
        </div>
    </div>
        <!--<div id="ultim"></div>-->
  </body>
</html>
```

A.2.2. Programa gpcon.js

```
(function () {
    "use strict";

    // Imports
    let template = htmlLib.template;
    let qs = htmlLib.qs;

    // Currently visible controller
    let currentVisibleController = null;

    /**
```

```
     * Show a certain controller
     */
    function showController(n) {

        n = n | 0;

        console.log("Selecting gamepad " + n);

        let gamepads = document.querySelectorAll("#gamepad-container
.gamepad");

        for (let i = 0; i < gamepads.length; i++) {
            let gp = gamepads[i];
            let index = gp.getAttribute("data-gamepad-index");

            index = index | 0;

            if (index == n) {
                gp.classList.remove('nodisp');
            } else {
                gp.classList.add('nodisp');
            }
        }

        currentVisibleController = n;
    }

    /**
     * Reconstruct the UI for the current gamepads
     */
    function rebuildUI() {

        // Handle gamepad selector button clicks
        function onButtonClick(ev) {
            let b = ev.currentTarget;
            let gpIndex = b.getAttribute('data-gamepad-index');

            showController(gpIndex);
        }

        let gp = navigator.getGamepads();

        let bbbox = qs("#button-bar-box");
        bbbox.innerHTML = '';

        let gpContainer = qs("#gamepad-container");
        gpContainer.innerHTML = '';
```

```
        let haveControllers = false, curControllerVisible = false,
firstController = null;

        // For each controller, generate a button from the
        // button template, set up a click handler, and append
        // it to the button box
        for (let i = 0; i < gp.length; i++) {

            // Chrome has null controllers in the array
            // sometimes when nothing's plugged in there--ignore
            // them
            if (!gp[i] || !gp[i].connected) { continue; }

            let gpIndex = gp[i].index;

            // Clone the selector button
            let button = template("#template-button",
                {
                    "id": "button-" + gpIndex,
                    "data-gamepad-index": gpIndex,
                    "value": gpIndex
                });

            bbbox.appendChild(button);

            // Add the selector click listener
            button.addEventListener('click', onButtonClick);

            // Clone the main holder
            let gamepad = template("#template-gamepad",
                {
                    "id": "gamepad-" + gpIndex,
                    "data-gamepad-index": gpIndex
                });

            gpContainer.appendChild(gamepad);

            let mapping = gp[i].mapping;
            // Add the buttons for this gamepad
            let j;
            let buttonBox = qs(".gamepad-buttons-box", gamepad)

            for (j = 0; j < gp[i].buttons.length; j++) {
                let buttonContainer = template("#template-gamepad-button-
container",
                    {
                        "id": "gamepad-" + gpIndex + "-button-container-" + j
                    });
```

```
                    qs(".gamepad-button", buttonContainer).setAttribute("id",
"gamepad-" + gpIndex + "-button-" + j);
                    //qs(".gamepad-button-label", buttonContainer).innerHTML =
j;

                buttonBox.appendChild(buttonContainer);
            }


            // Add the axes for this gamepad
            let axesBox = qs(".gamepad-axes-box", gamepad);
            let axesBoxCount = ((gp[i].axes.length + 1) / 2)|0; // Round up
(e.g. 3 axes is 2 boxes)

            for (j = 0; j < axesBoxCount; j++) {
                let axisPairContainer = template("#template-gamepad-axis-pair-
container",
                    {
                        "id": "gamepad-" + gpIndex + "-axis-pair-container-" +
j
                    });

                //qs(".gamepad-axis-pair",
axisPairContainer).setAttribute("id", "gamepad-" + gpIndex + "-axispair-" +
j);

                let pairLabel;

                // If we're on the last box and the number of axes is odd,
just put one label on there
                if (j == axesBoxCount - 1 && gp[i].axes.length % 2 == 1) {
                    pairLabel = j*2;
                } else {
                    pairLabel = (j*2) + "," + ((j*2)+1);
                }
                //qs(".gamepad-axis-pair-label", axisPairContainer).innerHTML
= pairLabel;

                axesBox.appendChild(axisPairContainer);
            }

            // And remember that we have controllers now
            haveControllers = true;

            if (i == currentVisibleController) {
                curControllerVisible = true;
            }

            if (firstController === null) {
```

```
                firstController = i;
            }
        }

        // Show or hide the "plug in a controller" prompt as
        // necessary
        if (haveControllers) {
            qs("#prompt").classList.add("nodisp");
            qs("#main").classList.remove("nodisp");
        } else {
            qs("#prompt").classList.remove("nodisp");
            qs("#main").classList.add("nodisp");
        }

        if (curControllerVisible) {
            //showController(currentVisibleController);
        } else {
            currentVisibleController = firstController;
            showController(firstController);
        }
    }

    /**
     * Update the UI components based on gamepad values
     */
     function updateUI() {

        let gamepads = navigator.getGamepads();

        let mode = 'clamp'; // raw, norm, clamp

        // For each controller, show all the button and axis information
        for (let i = 0; i < gamepads.length; i++) {
            let gp = gamepads[i];
            let j;

            if (!gp || !gp.connected) { continue; }

            let gpElem = qs("#gamepad-" + i);

            // Show button values
            let buttonBox = qs(".gamepad-buttons-box", gpElem);

            for (j = 0; j < gp.buttons.length; j++) {
                let buttonElem = qs("#gamepad-" + i + "-button-" + j,
buttonBox)
                let button = gp.buttons[j];

                // Put the value in there
```

```
                buttonElem.innerHTML = button.value;

                // Change color if pressed or not
                if (gp.buttons[0].pressed) {
                    ui_help();
                } else {
                    ui_help_hide();
                }
                if (gp.buttons[1].pressed) {
                    stream_screenshot();
                } else {

                }
                if (gp.buttons[12].pressed) {
                    emerge();
                } else {

                }
                if (gp.buttons[13].pressed) {
                    immerse();
                } else {

                }

            }


            // Show axis values
            let axesBox = qs(".gamepad-axes-box", gpElem);
            let axesBoxCount = ((gp.axes.length + 1) / 2)|0; // Round up (e.g.
3 axes is 2 boxes)

            for (j = 0; j < axesBoxCount; j++) {
                let axisPairContainer = qs("#gamepad-" + i + "-axis-pair-
container-" + j, axesBox);
                let axisPairValue = qs(".gamepad-axis-pair-value",
axisPairContainer);
                let axisPip = qs(".gamepad-axis-pip", axisPairContainer);
                let axisCross = qs(".gamepad-axis-crosshair",
axisPairContainer);
                let valueX, valueY, valueStr;
                let deadzoneActive = true;

                valueX = gp.axes[j*2];

                // If we're not a single axis in the last box, show the
                // second axis in this box. This handles a last box with
                // a single axis (odd number of axes total).
```

```
                let last_odd_axis = j == axesBoxCount - 1 && gp.axes.length %
2 == 1;

                valueY = last_odd_axis? 0: gp.axes[j*2 + 1];


                if (deadzoneActive) {
                    [valueX, valueY] = gpLib.deadzone(valueX, valueY);
                }

                // Set the value label
                valueStr = valueX.toFixed(2);

                if (!last_odd_axis)
                    valueStr += ',' + valueY.toFixed(2);

                // Position the raw indicator
                axisCross.style.left = (valueX + 1) / 2 * 100 + '%';
                axisCross.style.top = (valueY + 1) / 2 * 100 + '%';


                // Position the pip, clamping if necessary
                let clampCircle = qs(".gamepad-circle", axisPairContainer);

                if (mode == 'clamp') {
                    // Clamp
                    let clampX, clampY;
                    [clampX, clampY] = gpLib.clamp(valueX, valueY, mode);
                    axisPip.style.left = (clampX + 1) / 2 * 100 + '%';
                    axisPip.style.top = (clampY + 1) / 2 * 100 + '%';
                    joystick_move(valueX,valueY);

                    clampCircle.classList.remove("nodisp");

                    // Overwrite the value string with clamped values
                    valueStr = clampX.toFixed(2)


                    if (!last_odd_axis)
                        valueStr += ',' + clampY.toFixed(2);
                } else {
                    // Raw
                    axisPip.style.left = axisCross.style.left;
                    axisPip.style.top = axisCross.style.top;

                    clampCircle.classList.add("nodisp");
                }

                // Show coordinates
```

```
                axisPairValue.innerHTML = valueStr;
            }

        }
    }

    /**
     * Render a frame
     */

    function onFrame() {
        let conCheck = gpLib.testForConnections();

        // Check for connection or disconnection
        if (conCheck) {
            console.log(conCheck + " new connections");

            // And reconstruct the UI if it happened
            rebuildUI();
        }

        // Update all the UI elements
        updateUI();

        requestAnimationFrame(onFrame);
    }
    /**
     * onload handler
     */
    function onLoad() {
        if (gpLib.supportsGamepads()) {
            rebuildUI();
            requestAnimationFrame(onFrame);
        } else {
            qs("#sol").classList.remove("nodisp");
        }
    }
    // Initialization code
    window.addEventListener('load', onLoad);
}());
```

A.2.3. Programa libgp.js

```
gpLib = (function () {

    /**
     * Test gamepad support
```

```
 */
function supportsGamepads() {
    return !!(navigator.getGamepads);
}

/**
 * Test for new or removed connections
 */
let testForConnections = (function() {

    // Keep track of the connection count
    let connectionCount = 0;

    // Return a function that does the actual tracking
    //
    // The function returns a positive number of connections,
    // a negative number of disconnections, or zero for no
    // change.
    return function () {
        let gamepads = navigator.getGamepads();
        let count = 0;
        let rv;

        for (let i = gamepads.length - 1; i >= 0; i--) {
            let g = gamepads[i];

            // Make sure they're not null and connected
            if (g && g.connected) {
                count++;
            }
        }

        // Return any changes
        rv = count - connectionCount;

        connectionCount = count;

        return rv;
    }
}());

/**
 * Clamp X and Y gamepad coordinates to length 1.0
 *
 * @param {Number} x
 * @param {Number} y
 *
 * @return {Array} The clamped X and Y values
 */
```

```
    function clamp(x, y) {
        let m = Math.sqrt(x*x + y*y); // Magnitude (length) of vector

        // If the length greater than 1, normalize it (set it to 1)
        if (m > 1) {
            x /= m;
            y /= m;
        }


        return [x, y];
    }


    /**
     * Given a 2D gamepad axis value, normalize it so there's a deadzone
     *
     * @param {Number} x The x axis value
     * @param {Number} y The y axis value
     * @param {Number} deadzone [optional] The deadzone radius, 0 to 0.999
     *
     * @return [{Number},{Number}] The deadzone value of the axis
     */
    function deadzone(x, y, deadzone=0.2) {
        let m = Math.sqrt(x*x + y*y);

        if (m < deadzone)
            return [0, 0];

        let over = m - deadzone;  // 0 -> 1 - DEADZONE
        let nover = over / (1 - deadzone);  // 0 -> 1

        let nx = x / m;
        let ny = y / m;

        return [nx * nover, ny * nover];

    }


    // Exports
    return {
        clamp: clamp,
        deadzone: deadzone,
        supportsGamepads: supportsGamepads,
        testForConnections: testForConnections
    };

}());
```

A.2.4. Programa libhtml.js

```
htmlLib = (function () {
    /**
     * Wrapper around querySelector
     */
    function qs(s, p) {
        if (p) {
            return p.querySelector(s);
        }
        return document.querySelector(s);
    }

    /**
     * Clone an HTML template
     */
    function template(id, attribs, subs) {
        let t = qs(id);
        let html = t.outerHTML;
        let key;

        // Do the substitutions in the HTML
        if (subs) for (key in subs) {
            let val = subs[key];
            html = html.replace(new RegExp(key, 'g'), val);
        }

        // Make a dummy element to hold the cloned HTML
        let wrapper = document.createElement('div');
        wrapper.innerHTML = html;
        let clone = wrapper.querySelector(id);

        // Set the new attributes
        if (attribs) for (key in attribs) {
            let val = attribs[key];
            clone.setAttribute(key, val);
        }

        return clone;
    }

    // Exports
    return {
        qs: qs,
        template: template
    };

}());
```

A.2.5. Programa intro_host.js

```
window.MQTT_PORT = 9001;


function start_connection() {
  setTimeout(ui_connecting_animation.bind(null, true), 0); // async

  window.host = host;

  window.mqtt = new MQTT(
    window.host,
    MQTT_PORT,
    console.log,
    (err) => { // On error message:
      setTimeout(ui_connecting_animation.bind(null, false), 0); // async
      message(err);
  });
  window.mqtt.on_connect(() => {
    ui_connecting_animation(false);
    pantalla('panel');
    message('Connected to robot ✓');
  });
  window.mqtt.connect();
}
```

A.2.6. Programa joystick.js

```
function Joystick(elem, callback) {
  // Elem must be a canvas.
  // callback(rel_x, rel_y) {...}

  if (this === window)
    return new Joystick(elem, callback);

  this.callback = callback || function(){};
  this.canvas = elem;
  this.ctx    = null;
  this.width  = 0;
  this.height = 0;
  this.radius = 100;
  this.off_x  = 0;
  this.off_y  = 0;
  this.x_org  = 0;
  this.y_org  = 0;
```

```
  this.x      = 0;
  this.y      = 0;
  this.paint = false;

  this.colors = {
    // Default colors. If you wanna change these on runtime, use:
    //                 Joystick.set_background and Joystick.set_foreground
    bg: '#CCCCCC',
    fg: '#22AA22',
    fr: '#22AA2280', // foreground radius
  };
  this.fr_radius = 10; // foreground radius radius in px
  this.bg_radius_extra = 40;
  this.joy_size = 1; // Multiply (zoom) the joystick size. Bigger → Bigger dot

  this.zoom = 4; // Reversed zoom (higher values means smaller)

  this.setup();
}

Joystick.prototype.set_callback = function(callback) {
  // Set/change callback function.
  this.callback = callback;
  return this;
};

Joystick.prototype.set_background = function(bgcolor) {
  // Set the background color.
  this.colors.bg = bgcolor;
  this.draw();
  return this;
};

Joystick.prototype.set_foreground = function(fgcolor, fgradius) {
  // Set the background color.
  this.colors.fg = fgcolor;
  this.colors.fr = fgradius || fgcolor + '80';
  this.draw();
  return this;
};

Joystick.prototype.set_background_radius = function(r_extra) {
  // Set extra radius in pixels for the background.
  this.bg_radius_extra = r_extra;
  this.draw();
  return this;
};

Joystick.prototype.set_joystick_size = function(level) {
```

```
  // Set joystick zoom size. Bigger → Bigger dot
  this.joy_size = level;
  this.draw();
  return this;
};

Joystick.prototype.set_zoom = function(level) {
  // Set zoom level, bigger values → bigger joystick.
  this.zoom = 1/(level/4);
  this.resize();
  return this;
};

Joystick.prototype.setup = function() {
  // Setup canvas and events.
  this.ctx = this.canvas.getContext('2d');
  this.resize();

  this.canvas.addEventListener('mousedown', this.start_drawing.bind(this));
  document.addEventListener('mouseup', this.stop_drawing.bind(this));
  document.addEventListener('mousemove', this.move.bind(this));

  this.canvas.addEventListener('touchstart', this.start_drawing.bind(this));
  document.addEventListener('touchend', this.stop_drawing.bind(this));
  document.addEventListener('touchcancel', this.stop_drawing.bind(this));
  document.addEventListener('touchmove', this.move.bind(this));
  window.addEventListener('resize', this.resize.bind(this));
};

Joystick.prototype.launch_callback = function() {
  const rel = this.get_relative();
  this.callback(rel.x, rel.y);
};

Joystick.prototype.resize = function() {
  // Resize joystick.
  const size = this._get_elem_size(this.canvas);
  this.width  = size.width;
  this.height = size.height;
  this.off_x  = size.x;
  this.off_y  = size.y;
  this.ctx.canvas.width  = this.width;
  this.ctx.canvas.height = this.height;
  this.radius = Math.min(this.width, this.height) / this.zoom;
  this.x_org = this.width / 2;
  this.y_org = this.height / 2;
  this.goto_center();
};
```

```
Joystick.prototype.background = function() {
  // Draw the background
  this.ctx.beginPath();
  this.ctx.arc(this.x_org, this.y_org, this.radius + this.bg_radius_extra, 0,
Math.PI * 2, true);
  this.ctx.fillStyle = this.colors.bg;
  this.ctx.fill();
};

Joystick.prototype.joystick = function(width, height) {
  this.ctx.beginPath();
  this.ctx.arc(width, height, this.radius * this.joy_size, 0, Math.PI * 2,
true);
  this.ctx.fillStyle = this.colors.fg;
  this.ctx.fill();
  this.ctx.strokeStyle = this.colors.fr;
  this.ctx.lineWidth = this.fr_radius;
  this.ctx.stroke();
};

Joystick.prototype.get_position = function(event) {
  // Calculate position from event.
  const mouse_x = event.clientX || event.touches[0].clientX;
  const mouse_y = event.clientY || event.touches[0].clientY;
  this.x = mouse_x - this.off_x;
  this.y = mouse_y - this.off_y;
};

Joystick.prototype.get_relative = function() {
  // Returns {x: 0 ↔ 1, y: 0 ↔ 1} in the plane - ↑→ +
  let x = (this.x - this.x_org) / (this.width / 2);
  let y = - (this.y - this.y_org) / (this.height / 2);
  return {
    x: x > 0 ? Math.min(1, x) : Math.max(-1, x),
    y: y > 0 ? Math.min(1, y) : Math.max(-1, y),
  };
};

Joystick.prototype.in_circle = function() {
  // Return wheather the mouse pointer is in the circle or not.
  const current_radius = Math.sqrt(Math.pow(this.x - this.x_org, 2)
                                   + Math.pow(this.y - this.y_org, 2));
  return (this.radius >= current_radius);
};

Joystick.prototype.goto_center = function() {
  this.x = this.width / 2;
  this.y = this.height / 2;
  this.draw();
```

```
};

Joystick.prototype.start_drawing = function(event) {
  // Start drawing the joystick move.
  this.paint = true;
  this.get_position(event);
  if (this.in_circle()) {
    this.ctx.clearRect(0, 0, this.width, this.height);
    this.background();
    this.joystick(this.x, this.y);
    this.draw();
  }
};

Joystick.prototype.stop_drawing = function() {
  // Stop drawing the joystick move.
  if (this.paint) {
    this.goto_center();
  }
  this.paint = false;
  window.stop();
}

Joystick.prototype.move = function(event) {
  // Process the mouse pointer movement.
  if (!this.paint)
    return;
  this.get_position(event);
  this.draw();
  this.launch_callback();
};

Joystick.prototype.draw = function() {
  // Draw the joystick.
  this.ctx.clearRect(0, 0, this.width, this.height);
  this.background();
  const angle = Math.atan2((this.y - this.y_org), (this.x - this.x_org));

  if (this.in_circle())
    this.joystick(this.x, this.y);
  else {
    let x = this.radius * Math.cos(angle) + this.x_org;
    let y = this.radius * Math.sin(angle) + this.y_org;
    this.joystick(x, y);
  }
};

Joystick.prototype._get_elem_size = function(elem) {
  // Get an element size.
```

```
  // Returns DOMRect { x: 0, y: 0, width: 1920, height: 1080, top: 0, right:
1920, bottom: 0, left: 0 }
  return elem.getBoundingClientRect();
};
```

A.2.7. Programa keyboard.js

```
const KEYBOARD_SHORTCUTS = {
  // {Key : {down: callback, up: callback, description: "My description ;-p"]}
  // {Key : "alias"} → example: {a: "ArrowLeft"}
  ArrowUp: {
    down        : keyboard_start_move.bind(null, 'ArrowUp'),
    up          : keyboard_stop_move.bind(null, 'ArrowUp'),
    description : "Mou el submarí cap endevant",
  },
  ArrowLeft: {
    down        : keyboard_start_move.bind(null, 'ArrowLeft'),
    up          : keyboard_stop_move.bind(null, 'ArrowLeft'),
    description : "Rota el submarí ↺ sobre si mateix",
  },
  ArrowDown: {
    down        : keyboard_start_move.bind(null, 'ArrowDown'),
    up          : keyboard_stop_move.bind(null, 'ArrowDown'),
    description : "Mou el submarí cap endarrera",
  },
  ArrowRight: {
    down        : keyboard_start_move.bind(null, 'ArrowRight'),
    up          : keyboard_stop_move.bind(null, 'ArrowRight'),
    description : "Rota el submarí ↻ sobre si mateix",
  },
  w: "ArrowUp",
  a: "ArrowLeft",
  s: "ArrowDown",
  d: "ArrowRight",
  W: "ArrowUp",
  A: "ArrowLeft",
  S: "ArrowDown",
  D: "ArrowRight",
  q: {
    down        : (() => { emerge(); }),
    up          : (() => { stop(); }),
    description : "Mou el submarí cap amunt",
  },
  Q: "q",
  e: {
    down        : (() => { immerse(); }),
```

```
    up          : (() => { stop(); }),
    description : "Mou el submarí cap avall",
  },
  E: "e",
  x: {
    down        : keyboard_kill,
    description : "Força l'aturada de tots els events",
  },
  X: "x",
  f: {
    up          : (() => { follow(); }),
    description : "Activa/desactiva el mode de seguiment",
  },
  h: {
    down        : (() => { ui_help(); }),
    up          : (() => { ui_help_hide(); }),
    description : "Mostra el missatge d'ajuda",
  },
  v: {
    up          : (() => { stream(); }),
    description : "Activa/desactiva el video en streaming",
  },
  l: {
    up          : (() => { light(); }),
    description : "Activa/desactiva les llums",
  },
  L: "l",
  o: {
    up          : (() => { stream_screenshot(); }),
    description : "Save a shot of the stream",
  },
};

window._keyboard_down = {}; // Currently pressed keys

function keyboard_install() {
  document.body.addEventListener('keydown', keyboard_handle_down);
  document.body.addEventListener('keyup', keyboard_handle_up);
}

function keyboard_disable() {
  document.body.removeEventListener('keydown', keyboard_handle_down);
  document.body.removeEventListener('keyup', keyboard_handle_up);
}

function keyboard_kill() {
  // Force to stop any active action
  for (const [key, down] of Object.entries(_keyboard_down))
    if (down)
```

```
      keyboard_handle_up({key: key});
}

function keyboard_descriptions() {
  // Returns [[keys], "description"]
  // Example: [["←", "a", "A"], "Move to the left"]
  function translate_key(k) {
    // Change some key names.
    if (k == "ArrowUp")    return '↑';
    if (k == "ArrowLeft")  return '←';
    if (k == "ArrowDown")  return '↓';
    if (k == "ArrowRight") return '→';
    return k;
  }

  // GroupBy description
  let desc;
  let keys_by_desc = {} // {"description": ["key1", "key2"]}
  for (const key of Object.keys(KEYBOARD_SHORTCUTS)) {
    desc = (key_to_action(key)||{}).description;
    if (!keys_by_desc[desc])
      keys_by_desc[desc] = [];
    keys_by_desc[desc].push(translate_key(key));
  }

  // Create result format
  let res = [];
  for (const [desc, keys] of Object.entries(keys_by_desc))
    res.push([keys, desc]);

  return res;
}

function key_to_action(key) {
  // Returns {up: callback, down: callback, description: "arst"} or null
  const kval = KEYBOARD_SHORTCUTS[key];
  if (! kval)
    return null;
  if (typeof kval === 'string')
    return key_to_action(kval);
  return kval;
}

function keyboard_handle_down(event) {
  const key    = event.key;
  const action = key_to_action(key);
  if (! action || event.repeat) // Is the key for us?
    return;
  window._keyboard_down[key] = true;
```

```
  setTimeout(action.down || function(){}, 0); // Do action
}

function keyboard_handle_up(event) {
  const key    = event.key;
  const action = key_to_action(key);
  if (! action) // Is the key for us?
    return;
  window._keyboard_down[key] = false;
  setTimeout(action.up || function(){}, 0); // Do action
}

function keyboard_start_move(direction) {
  // Direction must be in ["ArrowUp", "ArrowLeft", "ArrowDown", "ArrowRight"]
  if (direction == "ArrowUp") {
    if (window._keyboard_down["ArrowLeft"])
      move_forward(0.2, 0.5);
    else if (window._keyboard_down["ArrowRight"])
      move_forward(0.5, 0.2);
    else if (window._keyboard_down["ArrowDown"])
      return;
    else
      move_forward(0.4, 0.4);
  } else if (direction == "ArrowDown") {
    if (window._keyboard_down["ArrowLeft"])
      move_backward(0.2, 0.5);
    else if (window._keyboard_down["ArrowRight"])
      move_backward(0.5, 0.2);
    else if (window._keyboard_down["ArrowUp"])
      return;
    else
      move_backward(0.4, 0.4);
  } else if (direction == "ArrowLeft") {
    if (window._keyboard_down["ArrowUp"])
      move_forward(0.2, 0.5);
    else if (window._keyboard_down["ArrowDown"])
      move_backward(0.2, 0.5);
    else if (window._keyboard_down["ArrowRight"])
      return;
    else
      turn_left();
  } else if (direction == "ArrowRight") {
    if (window._keyboard_down["ArrowUp"])
      move_forward(0.5, 0.2);
    else if (window._keyboard_down["ArrowDown"])
      move_backward(0.5, 0.2);
    else if (window._keyboard_down["ArrowLeft"])
      return;
    else
```

```
      turn_right();
  }
}

function keyboard_stop_move(direction) {
  // Direction must be in ["ArrowUp", "ArrowLeft", "ArrowDown", "ArrowRight"]
  if (window._keyboard_down['ArrowUp']
      || window._keyboard_down['ArrowLeft']
      || window._keyboard_down['ArrowDown']
      || window._keyboard_down['ArrowRight'])
    keyboard_start_move(direction);
  else
    stop();
}
```

A.2.8. Programa motor.js

```
var mqtt;
let angle=0;
var reconnectTimeout = 2000;
//window.host = "169.254.10.125";
var port = 9001;
window.TOPICS = ["Server", "Motors", "Camera", "Follow", "Stream", "Sensors",
"Light", "Aigua"];
var username = 'Java';
var password = 'Script';

function sub_mqtt_msg(callback, error) {
  callback = callback || function(){};
  error    = error    || message;

  function onConnect() {
    callback();
    client.subscribe(TOPICS[0]);
    client.subscribe(TOPICS[1]);
    client.subscribe(TOPICS[2]);
    client.subscribe(TOPICS[3]);
    client.subscribe(TOPICS[4]);
    client.subscribe(TOPICS[5]);
    client.subscribe(TOPICS[6]);
    client.subscribe(TOPICS[7]);
    message("Waiting for " + TOPICS[0]);
  }

  // Send an MQTT message
  client = new Paho.MQTT.Client(window.mqtt.host, port, "C");
```

```
  client.onMessageArrived = onMessageArrived;
  client.onConnectionLost = error.bind(null, "Connection lost :v");
  client.connect({onSuccess:onConnect});


  document.getElementById("estat").innerText = "Trying to connect...";

}
let prof = 0;
function onMessageArrived(message) {
  let topic = message.destinationName;
  let result = message.payloadString.split(",");
  if (topic == "Sensors"){
    //ROLL
    const rotated=document.getElementById('imageroll');
    let rotateROLL = parseInt(result[0].slice(1));
    rotated.setAttribute("style", "transform: rotate(" + rotateROLL + "deg)");
    //PITCH
    const rotatedp=document.getElementById('imagepitch');
    let rotatePITCH = parseInt(result[1]);
    rotatedp.setAttribute("style", "transform: rotate(" + rotatePITCH +
"deg)");
    //BRUIXOLA
    const rotatedb=document.getElementById('imagebruix');
    let rotateBRUIX = parseInt(result[2]);
    rotatedb.setAttribute("style", "transform: rotate(" + rotateBRUIX +
"deg)");
    //TEMPERATURA
    let temperatura = parseInt(result[3]);
    document.getElementById('temp').innerHTML = temperatura ;
    //PROFUNDITAT
    let profunditat = parseFloat(result[4]);
    prof = profunditat;
    document.getElementById('deep').innerHTML = profunditat ;
    //BATERIA
    let bateria = parseInt(result[5]);
    document.getElementById('Nbat').innerHTML = bateria;
  if (bateria==0){
    document.getElementById('imageB0').style.visibility = "visible";
    document.getElementById('imageB25').style.visibility = "hidden";
    document.getElementById('imageB50').style.visibility = "hidden";
    document.getElementById('imageB75').style.visibility = "hidden";
    document.getElementById('imageB100').style.visibility = "hidden";
  }
  if (bateria>=1 && bateria<=25){
    document.getElementById('imageB25').style.visibility = "visible";
    document.getElementById('imageB0').style.visibility = "hidden";
    document.getElementById('imageB50').style.visibility = "hidden";
    document.getElementById('imageB75').style.visibility = "hidden";
```

```
    document.getElementById('imageB100').style.visibility = "hidden";
  }
  if (bateria>=26 && bateria<=50){
    document.getElementById('imageB50').style.visibility = "visible";
    document.getElementById('imageB0').style.visibility = "hidden";
    document.getElementById('imageB25').style.visibility = "hidden";
    document.getElementById('imageB75').style.visibility = "hidden";
    document.getElementById('imageB100').style.visibility = "hidden";
  }
  if (bateria>=51 && bateria<=75){
    document.getElementById('imageB75').style.visibility = "visible";
    document.getElementById('imageB0').style.visibility = "hidden";
    document.getElementById('imageB25').style.visibility = "hidden";
    document.getElementById('imageB50').style.visibility = "hidden";
    document.getElementById('imageB100').style.visibility = "hidden";
  }
  if (bateria>=76 && bateria<=100){
    document.getElementById('imageB100').style.visibility = "visible";
    document.getElementById('imageB0').style.visibility = "hidden";
    document.getElementById('imageB25').style.visibility = "hidden";
    document.getElementById('imageB50').style.visibility = "hidden";
    document.getElementById('imageB75').style.visibility = "hidden";
  }
  }
  if (topic == "Aigua"){
    window.alert("S'ha detectat aigua dins el GIRONA 25!");
  }
}

// Movement actions

function emerge() {
    let action = "up";
    let speed = 0.2 * velocitat;
    const msg = JSON.stringify([action, speed, speed]);
    window.mqtt.send(TOPICS[1],msg);
}

function immerse() {
    let action = "dwn";
    let speed = 0.2 * velocitat;
    const msg = JSON.stringify([action, speed, speed]);
    window.mqtt.send(TOPICS[1],msg);
}
function left() {
  let action = "lleft";
  let speed = 0.4 * velocitat;
  const msg = JSON.stringify([action, speed, speed]);
  window.mqtt.send(TOPICS[1],msg);
```

```
}

function right() {
  let action = "lright";
  let speed = 0.4 * velocitat;
  const msg = JSON.stringify([action, speed, speed]);
  window.mqtt.send(TOPICS[1],msg);
}

function move_forward(speed_left, speed_right) {
    let action = "fwd";
    const msg = JSON.stringify([action, speed_left, speed_right]);
    window.mqtt.send(TOPICS[1],msg);
}

function move_backward(speed_left, speed_right) {
    let action = "bkd";
    const msg = JSON.stringify([action, speed_left, speed_right]);
    window.mqtt.send(TOPICS[1],msg);
}

function turn_left() {
    let action = "left";
    let speed = 0.2 * velocitat;
    const msg = JSON.stringify([action, speed]);
    window.mqtt.send(TOPICS[1],msg);
}

function turn_right() {
    let action = "right";
    let speed = 0.2 * velocitat;
    const msg = JSON.stringify([action, speed]);
    window.mqtt.send(TOPICS[1],msg);
}

function stop() {
    let action = "stop";
    const msg = JSON.stringify([action, 0]);
    window.mqtt.send(TOPICS[1],msg);
}

function stop_dalt() {
  let action = "stopdalt";
  const msg = JSON.stringify([action, 0]);
  window.mqtt.send(TOPICS[1],msg);
}

function stop_baix() {
  let action = "stopbaix";
```

```
  const msg = JSON.stringify([action, 0]);
  window.mqtt.send(TOPICS[1],msg);
}

function stop_raspy() {
  apaga= confirm("Desitja aturar el GIRONA 25?");
  if (apaga) {
    let action = "stopraspy";
    const msg = JSON.stringify([action, 0]);
    window.mqtt.send(TOPICS[1],msg);
    document.getElementById("stop").innerText = "⚠";
  }
}

let Kp=0.5;
let Ki=0.001;
let error=0;
let output=0;
let integral=0;
let integral_prior=0;

let funciodepth;
let depth_hold_func= 1;
function start_hold(){
  depth_hold_func=1;
  let holdprof = prof;
  error=0;
  output=0;
  integral=0;
  integral_prior=0;
  funciodepth = setInterval(profun,100,holdprof);
}
function profun(holdprof){
  error= prof-holdprof;
  integral=  integral_prior + error;
  output=Kp*error+Ki*integral;
  integral_prior=integral;
  if (output>=0.5){
    output=0.5;
  }
  if (output<=-0.5){
    output=-0.5;
  }
  console.log(output);
  if (output>0){
    let action = "up";
    let speed = output * velocitat;
    const msg = JSON.stringify([action, speed, speed]);
    window.mqtt.send(TOPICS[1],msg);
```

```
    console.log("puja");
  } else if(output<0){
    let action = "dwn";
    let speed = - output * velocitat;
    const msg = JSON.stringify([action, speed, speed]);
    window.mqtt.send(TOPICS[1],msg);
    console.log("enfonsa");
  } else {
    console.log("aguanta");
    stop();
  }
}
/*function profun(holdprof){
if(holdprof <= (prof+0.05) && holdprof >= (prof-0.05)){
  console.log("aguanta");
  stop();
} else if (holdprof >= (prof+0.05)){
  //immerse();
  let action = "dwn";
  let speed = 0.1 * velocitat;
  const msg = JSON.stringify([action, speed, speed]);
  window.mqtt.send(TOPICS[1],msg);
  console.log("enfonsa");
} else if (holdprof <= (prof-0.05)){
  //emegre();
  let action = "up";
  let speed = 0.1 * velocitat;
  const msg = JSON.stringify([action, speed, speed]);
  window.mqtt.send(TOPICS[1],msg);
  console.log("puja");
}
}
*/
const vaclxant= 0;
const vaclyant= 0;
let amunt = 0;
let avall = 0;
let dreta = 0;
let esquerra = 0;
function handleMotion(event) {
  const aclx = event.acceleration.x;
  //document.getElementById('temp').innerHTML = aclx ;
  const acly = event.acceleration.y;
  //document.getElementById('deep').innerHTML = acly ;
  const aclz = event.acceleration.z;
  if (aclx >= 0.2 && vaclxant == 0 && esquerra == 0){
    turn_left();
    dreta = 1;
  } if (aclx <= -0.2 && dreta == 1){
```

```
    stop();
    dreta = 0;
  } if (aclx <= -0.2 && vaclxant == 0 && dreta == 0){
    turn_right();
    esquerra = 1;
  } if (aclx >= 0.2 && esquerra == 1){
    stop();
    esquerra = 0;
  } if (acly >= 0.2 && vaclyant == 0 && amunt == 0){
    emerge();
    avall = 1;
  } if (acly <= -0.2 && avall == 1){
    stop();
    avall = 0;
  } if (acly <= -0.2 && vaclyant == 0 && avall == 0){
    immerse();
    amunt = 1;
  } if (acly >= 0.2 && amunt == 1){
    stop();
    amunt = 0;
  }
  vaclxant=aclx;
  vaclyant=acly;
}
// Other actions
//Realitat virtual
function realitat(){
  let is_vr=document.getElementById("ulleres").following;
  let imgprin= document.getElementById("streaming");
  let imgdrt= document.getElementById("streaming1");
  let imgesq= document.getElementById("streaming2");
  if (!is_vr){
    document.getElementById("ulleres").following = true;
    button_toggle('ulleres', true);
    disable_controls_1();
    open_fullscreen();
    stream();
    imgprin.classList.add("hide");
    imgdrt.classList.remove("hide");
    imgesq.classList.remove("hide");
    window.addEventListener("devicemotion", handleMotion);
  }
  else {
    document.getElementById("ulleres").following = false;
    button_toggle('ulleres', false);
    enable_controls_1();
    //close_fullscreen();
    stream();
    imgprin.classList.remove("hide");
```

```
    imgdrt.classList.add("hide");
    imgesq.classList.add("hide");
    window.removeEventListener("devicemotion", handleMotion);
  }
}

// Autonomous mode
function follow() {
  let is_follow = document.getElementById("follow-button").following;
  if (!is_follow) {
    document.getElementById("follow-button").following = true;
    button_toggle('follow-button', true);
    disable_controls();
    window.mqtt.send(TOPICS[3], "start");
  }
  else {
    document.getElementById("follow-button").following = false;
    button_toggle('follow-button', false);
    enable_controls();
    window.mqtt.send(TOPICS[3], "stop");
  }
}

function light(){
  let lighton = document.getElementById("light-button").following;
  if (!lighton) {
    document.getElementById("light-button").following = true;
    button_toggle('light-button', true);
    window.mqtt.send(TOPICS[6], "on");
    document.getElementById("light-button").innerText = "💡";
  }
  else {
    document.getElementById("light-button").following = false;
    button_toggle('light-button', false);
    window.mqtt.send(TOPICS[6], "off");
    document.getElementById("light-button").innerText = "🔦";
  }
}
let velocitat = 0.5;
function velocity(){
  let velocitaton = document.getElementById("velocity-button").following;
  if (!velocitaton) {
    document.getElementById("velocity-button").following = true;
    button_toggle('velocity-button', true);
    velocitat = 1;
    document.getElementById("velocity-button").innerText = "V2";
  }
  else {
    document.getElementById("velocity-button").following = false;
```

```
    button_toggle('velocity-button', false);
    velocitat = 0.5;
    document.getElementById("velocity-button").innerText = "V1";
  }
}
function depth_hold(){
  let depthon = document.getElementById("hold-button").following;
  if (!depthon) {
    document.getElementById("hold-button").following = true;
    button_toggle('hold-button', true);
    document.getElementById("hold-button").innerText = "⚓";
    start_hold();
  }
  else {
    document.getElementById("hold-button").following = false;
    button_toggle('hold-button', false);
    document.getElementById("hold-button").innerText = "🚤";
    clearInterval(funciodepth);
    funcio= null;
  }
}
function stream_video(){
  let videoon = document.getElementById("video-button").following;
  if (!videoon) {
    document.getElementById("video-button").following = true;
    button_toggle('video-button', true);
    document.getElementById("video-button").innerText = "📽";
    start_video();
  }
  else {
    document.getElementById("video-button").following = false;
    button_toggle('video-button', false);
    document.getElementById("video-button").innerText = "📽";
    stop_video();
  }
}
// Take a picture
function capture_photo() {
  window.mqtt.send(TOPICS[2], "capture");
}

// Finish
function Finish() {
  window.mqtt.send("Follow", "stop");
  window.mqtt.send("Stream", "pause");
}

// window.load = null;
window.onunload = Finish();
```

A.2.9. Programa mqtt.js

```
function MQTT(host, port, log, error) {
  if (this === window)
    return new MQTT(host, log, error);

  if (!host || !port)
    throw "MQTT: Invalid host";

  this._host  = host;
  this._port  = port;
  this.error  = error || alert;
  this.log    = log   || console.log;
  this.client = null;

  this.callback = function(){};
  this.on_connect_callback = function(){};
}

// Constants
MQTT.TIMEOUT = 30;
MQTT.TIMEOUT = 2; // WARNING: DEBUG

// Getters

MQTT.prototype = {
  get host() {
    return this._host;
  },
  set host(x) {
    throw "Can't change host after instanciating MQTT";
  }
};


// Methods

MQTT.prototype.send = function(topic, action, callback) {
  this.set_callback(callback);
  message = new Paho.MQTT.Message(action);
  message.destinationName = topic;
  this.client.send(message);
};

MQTT.prototype.set_callback = function(callback) {
  // Callback for any message
  this.callback = callback || function(){};
```

```
  return this;
};


MQTT.prototype.on_message = function(callback) {
  // Callback for any message
  return this.set_callback(callback);
};

MQTT.prototype.on_connect = function(callback) {
  this.on_connect_callback = callback || function(){};
  return this;
};

MQTT.prototype.connect = function() {
  console.log("new Paho.MQTT.Client(",this._host, ",",this._port,', "App")');
    this.client = new Paho.MQTT.Client(this._host, this._port, "App");
    //console.log(this._host);
  this.client.onMessageArrived = this.new_message.bind(this);
  this.client.onConnectionLost = this.connection_lost.bind(this);
  this.client.connect({
    onSuccess: this.connected.bind(this),
    onFailure: this.fail.bind(this),
    timeout: MQTT.TIMEOUT,
  });
};

MQTT.prototype.new_message = function(payload) {
  let msg = message.payloadString;
  this.log(msg);
  this.callback(msg);
};

MQTT.prototype.connection_lost = function(payload) {
  this.error("Connection lost: " + payload.errorMessage);
};

MQTT.prototype.fail = function(payload) {
  this.error("Failed comunication: " + payload.errorMessage);
};

MQTT.prototype.connected = function() {
  this.client.subscribe(TOPICS[0]);
  this.client.subscribe(TOPICS[1]);
  this.client.subscribe(TOPICS[2]);
  this.client.subscribe(TOPICS[3]);
  this.client.subscribe(TOPICS[4]);
  this.client.subscribe(TOPICS[5]);
  this.client.subscribe(TOPICS[6]);
```

```
  this.client.subscribe(TOPICS[7]);
  this.client.onMessageArrived = onMessageArrived;
  this.log("Connected succesfully");
  this.on_connect_callback();
};
```

A.2.10. Programa stream.js

```
var STREAM_PORT = 8090;
var RASPY_IP = "169.254.10.125"

function stream() {
  let elem = document.getElementById("streaming");
  let playing = !!elem.playing;
  if (playing) {
    elem.playing = false;
    elem.classList.remove("streaming-on");
    button_toggle('stream-button', false);
    elem.src = "images/stream_off.png";
    window.mqtt.send(window.TOPICS[4], "pause");
  }
  else {
    elem.playing = true;
    elem.retry = 100;
    elem.classList.add("streaming-on");
    button_toggle('stream-button', true);
    stream_refresh_img();
    window.mqtt.send(window.TOPICS[4], "play", () => {stream_refresh_img();});
  }
}

function stream_refresh_img() {
  let elem = document.getElementById("streaming");
  if (! elem.playing)
    return;
    elem.src = "http://"+RASPY_IP+":"+STREAM_PORT+"/?action=stream";
}

function stream_error() {
  let elem = document.getElementById('streaming');

  // Show nosignal image
  elem.src='images/nosignal.png';

  // Auto-retry in 2x time
  elem.retry = Math.min((elem.retry||100) * 2, 10000); // Up to 10s
```

```
  setTimeout(stream_refresh_img, elem.retry);
}

function stream_screenshot() {
  let url = document.getElementById('streaming').src.replace('=stream',
'=snapshot');
  download_image(url, (new Date()).toISOString()+'.jpg');
}

function download_image(src, name) {
  let a = document.createElement('a');
  a.style.position = 'fixed';
  a.style.top      = '-800px';
  a.style.left     = '-800px';
  a.href     = src;
  a.download = name;
  a.target   = '_blank';
  document.body.appendChild(a);
  a.click();
  setTimeout(() => { document.body.removeChild(a); }, 2000);
}

const data = {
    capture: false,
    stream: null,
    buffer: [],
    iMediaRecorder: null,
}

async function start_video(){
  const displayMediaOptions = {
    video: {
      cursor: "never",
    },
    audio: false
  };
  try {
    data.buffer = [];
    data.stream = await
navigator.mediaDevices.getDisplayMedia(displayMediaOptions);
    data.iMediaRecorder = new MediaRecorder(data.stream, { mimeType:
'video/webm;codecs=h264' });
    data.iMediaRecorder.start();
    data.iMediaRecorder.ondataavailable = function (e){
      data.buffer.push(e.data);
    }
    data.iMediaRecorder.onstop = function (){
      let blob = new Blob(data.buffer, {type: "video/webm"});
      data.buffer = [];
```

```
      download(blob);
    }
  } catch(err) {
    console.log("Error: " + err);
  }
  function download (blob){
    let link = document.createElement("a");
    link.href = window.URL.createObjectURL(blob);
    link.setAttribute("download", "G25_video.mp4");
    link.style.display ="none";
    document.body.appendChild(link);
    link.click();
    link.remove();
  }
}
function stop_video(){
  data.iMediaRecorder.stop()
}
```

A.2.11. Programa ui.js

```
const MESSAGE_DISPLAY_TIME = 10000; // ms


function pantalla(id) {
  for (const e of document.getElementsByClassName('pantalla'))
    e.style.display = 'none';
  document.getElementById(id).style.display = 'block';
  pantalla_rules(id);
}

function pantalla_rules(id) {
  // Rules/functions to apply when an specific screen is showed
  if (id == 'panel') {
    keyboard_install();
    setup_joystick();
  } else {
    keyboard_disable();
  }
}

function message(text) {
  // Show a message to the user.
  if (!text || text == "")
    document.getElementById("estat-cont").style.display = 'none';
  else {
```

```
    document.getElementById("estat-cont").style.display = 'block';
    document.getElementById("estat").innerText = text;
    try {
      clearInterval(window._ui_message_interval);
    } catch (err) {}
    window._ui_message_interval = setTimeout(message, MESSAGE_DISPLAY_TIME);
  }
}

function ui_connecting_animation(show) {
  document.getElementById('loading-overlay').style.display = ['none',
'block'][+!!show];
}

function button_toggle(id, state) {
  let elem = document.getElementById(id);
  console.log(elem, state);
  if (state)
    elem.classList.add('active');
  else
    elem.classList.remove('active');
}

function disable_contextmenu() {
    return false;
}

function disable_controls() {
  for (const but of document.getElementsByClassName('move'))
    but.classList.add("hide");
}

function enable_controls() {
  for (const but of document.getElementsByClassName('move'))
    but.classList.remove("hide");
}

function disable_controls_1() {
  for (let elem of document.getElementsByClassName("ull"))
      elem.classList.add("hide");
}

function enable_controls_1() {
  for (let elem of document.getElementsByClassName("ull"))
      elem.classList.remove("hide");
}

function setup_joystick() {
  if (window._joystick) {
```

```
    // Already installed, force to recalculate it.
    window._joystick.resize();
    return;
  }
  // Install it
  let elem = document.getElementById('joystick');
  if (elem)
    window._joystick = (
      Joystick(elem, joystick_move)
        .set_background("#00000040")
        .set_foreground("#FFFFFF")
        .set_background_radius(20)
        .set_joystick_size(.75)
    );
}

function joystick_move(x, y) {
    const last = window._last_move || new Date(1999, 9, 6);
    const now = Date.now();
    if ((now - last) > 500) { // Milliseconds
    window._last_move = now;
    if (y >= 0) {
       if (x <= 0.2 && x >= -0.2) {
           let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
           window.move_forward(spd, spd);
       }
       else if (x > 0.2) {
           let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
           let spd_l = spd;
           let spd_r = spd - x;
           window.move_forward(spd_l, spd_r);
       }
       else if (x < -0.2) {
           let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
           let spd_l = spd + x;
           let spd_r = spd;
           window.move_forward(spd_l, spd_r);
       }
    }
    else if (y <= -0) {
       if (x <= 0.2 && x >= -0.2) {
           let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
           window.move_backward(spd, spd);
       }
       else if (x > 0.2) {
           let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
           let spd_l = spd;
           let spd_r = spd - x;
           window.move_backward(spd_l, spd_r);
```

```
      }
      else if (x < -0.2) {
          let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
          let spd_l = spd + x;
          let spd_r = spd;
          window.move_backward(spd_l, spd_r);
      }
    }
    else {
      window.stop();
    };
    console.log("joystick: ", x, y);
  }
}

function joystick_move_controller(x, y) {
  x=x/2;
  y=y/2;
  const last = window._last_move || new Date(1999, 9, 6);
  const now = Date.now();
  if ((now - last) > 500) { // Milliseconds
  window._last_move = now;
  if (y <= -0.01) {
  if (x <= 0.2 && x >= -0.2) {
    let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
    window.move_forward(spd, spd);
  }
  else if (x > 0.2) {
    let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
    let spd_l = spd;
    let spd_r = spd - x;
    window.move_forward(spd_l, spd_r);
  }
  else if (x < -0.2) {
    let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
    let spd_l = spd + x;
    let spd_r = spd;
    window.move_forward(spd_l, spd_r);
  }
  }
  else if (y >= 0.01) {
  if (x <= 0.2 && x >= -0.2) {
    let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
    window.move_backward(spd, spd);
  }
  else if (x > 0.2) {
    let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
    let spd_l = spd;
    let spd_r = spd - x;
```

```
    window.move_backward(spd_l, spd_r);
  }
  else if (x < -0.2) {
    let spd = Math.min(Math.sqrt(x**2 + y**2), 1) * velocitat;
    let spd_l = spd + x;
    let spd_r = spd;
    window.move_backward(spd_l, spd_r);
}
  }
  else {
    window.stop_baix();
      };
      console.log("joystick: ", x, y);
    }
}
// Help overlayer

function ui_help() {
  document.getElementById('help').style.display = 'inline-block';
  setTimeout(()=> {document.getElementById('help').style.display =
'none'},3000)
}
function install_ui_keyboard_descriptions() {
  let elem = document.getElementById('help-keyboard');
  elem.innerHTML = "";
  keyboard_descriptions().forEach(([keys, description]) => {
    let row = document.createElement("div");
    row.className = "row";

    // Description
    let desc = document.createElement("div");
    desc.className = "description";
    desc.innerText = description;
    row.appendChild(desc);

    // Keys
    let keycont = document.createElement('div');
    keycont.className = 'keys';
    for (const key of keys) {
      let keyelem = document.createElement('div');
      keyelem.className = 'key';
      keyelem.innerText = key;
      keycont.appendChild(keyelem);
    }
    row.appendChild(keycont);
    elem.appendChild(row);
  });
}
```

```
/* Display fullscreen */
function open_fullscreen() {
  var elem = document.documentElement;
  if (elem.requestFullscreen) {
    elem.requestFullscreen();
  }
  else if (elem.mozRequestFullScreen) { /* Firefox */
    elem.mozRequestFullScreen();
  }
  else if (elem.webkitRequestFullscreen) { /* Chrome and Safari */
    elem.webkitRequestFullscreen();
  }
  else if (elem.msRequestFullscreen) { /* IE/Edge */
    elem = window.top.document.body;
     elem.msRequestFullscreen();
  }
    document.getElementById("fullscreen").value = "open";
    document.getElementById("fullscreen").innerText = "⇲";
}

/* Close fullscreen */
function close_fullscreen() {
  if (document.exitFullscreen) {
    document.exitFullscreen();
  }
  else if (document.mozCancelFullScreen) {
    document.mozCancelFullScreen();
  }
  else if (document.webkitExitFullscreen) {
    document.webkitExitFullscreen();
  }
    document.getElementById("fullscreen").value = "close";
    document.getElementById("fullscreen").innerText = "⇱";
}

/* Enable/disable fullscreen */
function change_fullscreen() {
    var screen = document.getElementById("fullscreen").value;
    if (screen == "close") {
       open_fullscreen();
    }
    else if (screen == "open") {
       close_fullscreen();
    }
}


function apply_mobile() {
  for (let elem of document.getElementsByClassName("mobile"))
```

```
      elem.classList.add("hide");
}

function apply_notmobile() {
  var imagegran = document.getElementById("imagetot");
  imagegran.classList.remove("imgA1");
  imagegran.classList.add("imgA1M");
  var image1 = document.getElementById("imageroll");
  image1.classList.remove("imgA2");
  image1.classList.add("imgA2M");
  var image2 = document.getElementById("imagebruix");
  image2.classList.remove("imgA3");
  image2.classList.add("imgA3M");
  var image3 = document.getElementById("imagepitch");
  image3.classList.remove("imgA4");
  image3.classList.add("imgA4M");
  var image4 = document.getElementById("logo");
  image4.classList.add("logoM");
}


function amagarjoy() {
  for (let elem of document.getElementsByClassName("gamepad"))
      elem.classList.add("hide");
}

// Main

function main() {
  message();

  // One time installs:
  install_ui_keyboard_descriptions();
  amagarjoy();

  // Find platform
  if (!/Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera
Mini/i.test(navigator.userAgent) ) {
    apply_mobile();
  } else{
    apply_notmobile();
    window.screen.orientation.lock("landscape");
  }
  // Boot:
  pantalla('intro');
}

window.onload = main;
```

A.2.12. Programa button.css

```
button.login {
  font-size: 1.6em;
  font-weight: bold;
  background: rgb(0,66,0);
  background: linear-gradient(0deg, rgba(0,0,50,1) 0%, rgba(0,0,0,1) 100%);
  color: white;
  padding: .5em 2em .5em 2em;
  border-style: solid;
  border-width: 2px 2px 2px 2px;
  border-color: #333;
  width: 100%;
  cursor: pointer;
  transition: .4s all;
}

button.login:hover {
  box-shadow: #fff 0px 0px 10px;
}

button.login:active {
  background: linear-gradient(180deg, rgba(0,66,0,1) 0%, rgba(0,210,0,1)
100%);
}


/* Panel */

button.motor {
  color: white;
  font-size: 3vw;
  width: 6vw;
  min-width: 64px;
  height: 6vw;
  min-height: 64px;
  padding: .5vw;
  background: var(--button-background);
  border-style: solid;
  border-color: white;
  border-width: 1px;
  cursor: pointer;
  user-select: none;
  -webkit-user-select: none;
  -ms-user-select: none;
}

button.motor:hover {
```

```
  color: red;
  border-color: red;
}

button.motor:active {
  filter: invert();
}
.active {
  background: linear-gradient(0deg, rgba(0,0,0,.5) 0%, rgba(0,0,0,.2) 100%)
!important;
}
```

A.2.13. Programa gpcon.css

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
    display: block;
}
body {
    line-height: 1;
}
ol, ul {
    list-style: none;
}
blockquote, q {
    quotes: none;
}
blockquote:before, blockquote:after,
```

```
q:before, q:after {
    content: '';
    content: none;
}
table {
    border-collapse: collapse;
    border-spacing: 0;
}

/* end of reset */


body {
    font-family: sans-serif;
    padding: 14px;
}

i {
    font-style: italic;
}

#con {
    width: 100%;
    height: 100%;
}

#templates {
    display: none;
}

.nodisp {
    display: none;
}

#button-bar {
    display: inline-block;
}

#button-bar-box > input {
    margin-right: 2px;
    cursor: pointer;
}

#mode-select-box, #deadzone-box {
    display: inline-block;
    margin-left: 25px;
}

.gamepad-title {
```

```
    font-weight: bold;
    margin-bottom: 1ex;
    display: inline-block;
}

.gamepad-mapping {
    display: inline-block;
}

.gamepad-id {
    height: 7ex;
}
.gamepad-button-container {
    width: 24px;
    display: inline-block;
    margin: 0px 1px 0px 0px;
    text-align: center;
    font-size: 12px;
}

.gamepad-button {
    width: 24px;
    height: 24px;
    line-height: 24px;
    border-radius: 12px;
    background: #003000;
    color: white;
    margin-bottom: 0px;
}

.gamepad-button.pressed {
    background: green;
}

.gamepad-button-label {
    color: gray;
    margin-top: 0px;
}

.gamepad-controls {
    display: inline-block;
}

.gamepad-controls-center {
    text-align: center;
}

.gamepad-buttons-box, .gamepad-axes-box {
    text-align: center;
```

```
}

.gamepad-axis-pair-container {
    display: inline-block;
    text-align: center;
    font-size: 12px;
}

.gamepad-axis-pair {
    width: 80px;
    height: 80px;
    background: gray;
    border: 2px solid black;
    display: inline-block;
    position: relative;
    margin: 5px 5px 0px 5px;
}

.gamepad-circle {
    width: 79px;
    height: 79px;
    border: 1px solid #ddd;
    border-radius: 40px;
}

.gamepad-axis-pair.normalized {
    border-radius: 40px;
}

.gamepad-axis-pair-label {
    color: gray;
    margin-top: 0px;
}

.gamepad-axis-pip {
    width: 7px;
    height: 7px;
    border-radius: 3px;
    background: red;
    position: absolute;
    border: 1px solid white;
    left: 50%;
    top: 50%;
    transform: translateX(-4px) translateY(-4px);
}

.gamepad-axis-crosshair {
    width: 7px;
    height: 7px;
```

```
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translateX(-3px) translateY(-3px);
}

.gamepad-axis-crosshair-v {
    position: absolute;
    background: blue;
    width: 1px;
    height: 100%;
    left: 3px;
}

.gamepad-axis-crosshair-h {
    position: absolute;
    background: blue;
    width: 100%;
    height: 1px;
    top: 3px;
}
```

A.2.14. Programa loading.css

```
:root {
  --loading-size: 20vw;
  --loading-font-size: 2em;
  --loading-border-size: 4px;
}


.loading-overlay {
  position: fixed;
  top: 0;
  left: 0;
  background-color: rgba(0, 0, 0, .7);
  width: 100vw;
  height: 100vh;
  z-index: 7000;
  display: none;
}

@keyframes rotate-loading {
  0%  {transform: rotate(0deg);-ms-transform: rotate(0deg); -webkit-transform:
rotate(0deg); -o-transform: rotate(0deg); -moz-transform: rotate(0deg);}
```

```
  100% {transform: rotate(360deg);-ms-transform: rotate(360deg); -webkit-
transform: rotate(360deg); -o-transform: rotate(360deg); -moz-transform:
rotate(360deg);}
}

@-moz-keyframes rotate-loading {
  0%  {transform: rotate(0deg);-ms-transform: rotate(0deg); -webkit-transform:
rotate(0deg); -o-transform: rotate(0deg); -moz-transform: rotate(0deg);}
  100% {transform: rotate(360deg);-ms-transform: rotate(360deg); -webkit-
transform: rotate(360deg); -o-transform: rotate(360deg); -moz-transform:
rotate(360deg);}
}

@-webkit-keyframes rotate-loading {
  0%  {transform: rotate(0deg);-ms-transform: rotate(0deg); -webkit-transform:
rotate(0deg); -o-transform: rotate(0deg); -moz-transform: rotate(0deg);}
  100% {transform: rotate(360deg);-ms-transform: rotate(360deg); -webkit-
transform: rotate(360deg); -o-transform: rotate(360deg); -moz-transform:
rotate(360deg);}
}

@-o-keyframes rotate-loading {
  0%  {transform: rotate(0deg);-ms-transform: rotate(0deg); -webkit-transform:
rotate(0deg); -o-transform: rotate(0deg); -moz-transform: rotate(0deg);}
  100% {transform: rotate(360deg);-ms-transform: rotate(360deg); -webkit-
transform: rotate(360deg); -o-transform: rotate(360deg); -moz-transform:
rotate(360deg);}
}

@keyframes rotate-loading {
  0%  {transform: rotate(0deg);-ms-transform: rotate(0deg); -webkit-transform:
rotate(0deg); -o-transform: rotate(0deg); -moz-transform: rotate(0deg);}
  100% {transform: rotate(360deg);-ms-transform: rotate(360deg); -webkit-
transform: rotate(360deg); -o-transform: rotate(360deg); -moz-transform:
rotate(360deg);}
}

@-moz-keyframes rotate-loading {
  0%  {transform: rotate(0deg);-ms-transform: rotate(0deg); -webkit-transform:
rotate(0deg); -o-transform: rotate(0deg); -moz-transform: rotate(0deg);}
  100% {transform: rotate(360deg);-ms-transform: rotate(360deg); -webkit-
transform: rotate(360deg); -o-transform: rotate(360deg); -moz-transform:
rotate(360deg);}
}

@-webkit-keyframes rotate-loading {
  0%  {transform: rotate(0deg);-ms-transform: rotate(0deg); -webkit-transform:
rotate(0deg); -o-transform: rotate(0deg); -moz-transform: rotate(0deg);}
```

```
  100% {transform: rotate(360deg);-ms-transform: rotate(360deg); -webkit-
transform: rotate(360deg); -o-transform: rotate(360deg); -moz-transform:
rotate(360deg);}
}

@-o-keyframes rotate-loading {
  0%  {transform: rotate(0deg);-ms-transform: rotate(0deg); -webkit-transform:
rotate(0deg); -o-transform: rotate(0deg); -moz-transform: rotate(0deg);}
  100% {transform: rotate(360deg);-ms-transform: rotate(360deg); -webkit-
transform: rotate(360deg); -o-transform: rotate(360deg); -moz-transform:
rotate(360deg);}
}

@keyframes loading-text-opacity {
  0%  {opacity: 0}
  20% {opacity: 0}
  50% {opacity: 1}
  100%{opacity: 0}
}

@-moz-keyframes loading-text-opacity {
  0%  {opacity: 0}
  20% {opacity: 0}
  50% {opacity: 1}
  100%{opacity: 0}
}

@-webkit-keyframes loading-text-opacity {
  0%  {opacity: 0}
  20% {opacity: 0}
  50% {opacity: 1}
  100%{opacity: 0}
}

@-o-keyframes loading-text-opacity {
  0%  {opacity: 0}
  20% {opacity: 0}
  50% {opacity: 1}
  100%{opacity: 0}
}
.loading-container,
.loading {
  height: var(--loading-size);
  position: relative;
  width: var(--loading-size);
  border-radius: 100%;
}

.loading-container {
```

```
  /* にゃー！ */
}

.loading {
  border: var(--loading-border-size) solid transparent;
  border-color: transparent #fff transparent #FFF;
  -moz-animation: rotate-loading 1.5s linear 0s infinite normal;
  -moz-transform-origin: 50% 50%;
  -o-animation: rotate-loading 1.5s linear 0s infinite normal;
  -o-transform-origin: 50% 50%;
  -webkit-animation: rotate-loading 1.5s linear 0s infinite normal;
  -webkit-transform-origin: 50% 50%;
  animation: rotate-loading 1.5s linear 0s infinite normal;
  transform-origin: 50% 50%;
}


.loading-container .loading {
  -webkit-transition: all 0.5s ease-in-out;
  -moz-transition: all 0.5s ease-in-out;
  -ms-transition: all 0.5s ease-in-out;
  -o-transition: all 0.5s ease-in-out;
  transition: all 0.5s ease-in-out;
}

#loading-text {
  -moz-animation: loading-text-opacity 2s linear 0s infinite normal;
  -o-animation: loading-text-opacity 2s linear 0s infinite normal;
  -webkit-animation: loading-text-opacity 2s linear 0s infinite normal;
  animation: loading-text-opacity 2s linear 0s infinite normal;
  color: #ffffff;
  font-family: "Helvetica Neue, "Helvetica", ""arial";
  font-size: var(--loading-font-size);
  margin-top: calc(var(--loading-size)/2 - var(--loading-border-size) - .5em);
  font-weight: bold;
  opacity: 0;
  position: absolute;
  text-align: center;
  text-transform: uppercase;
  top: 0;
  width: 100%;
}
```

A.2.15. Programa style.css

```
:root {
```

```
    --button-background: linear-gradient(0deg, rgba(0,0,0,.8) 0%,
rgba(0,0,0,.3) 100%);
}

body {
    background: rgb(2,0,36);
    background: linear-gradient(0deg, rgba(2,0,36,1) 0%, rgba(9,9,121,1) 35%,
rgba(0,110,164,1) 100%);
    background-size: 100vw 100vh;
    height: 100vw;
    width: 100vw;
    margin: 0;
    overflow-x: hidden;
    overflow-y: hidden;
}


.banner {
    font-size: 3em;
}

.pantalla {
    display: none;
    top: 0;
    left: 0;
    position: fixed;
    width: 100vw;
}

.pantalla > div {
    height: 100vh;
}

div.center {
    display: flex;
    justify-content: center;
    align-items: center;
}

.login-cont {
    color: white;
}

/* Inputs */

input {
    font-size: 1.5em;
    width: calc(100% - 2em);
    padding: .3em 1em .3em 1em;
```

```
}

div.checkbox {
    display: block;
    height: 1.3em;
    font-size: 1.3em;
    justify-content: flex-start;
}
div.checkbox > input {
    width: .6em;
    height: .6em;
}
div.checkbox > label {
    display: inline;
}

/* State message */

.estat-cont {
    width: 100vw;
    display: block;
    justify-content: center;
    align-items: center;
    z-index: 9999;
}
.estat {
    display: inline-block;
    font-size: 1.2em;
    color: white;
    padding: .3em 1em .3em 1em;
    background: var(--button-background);
    border-radius: 0 0 .5em .5em;
}

/* Streaming video interface */

.interface {
    position: fixed;
    top: 0%;
    left: 0;
    width: 100vw;
    height: 100vw;
    overflow-x: hidden;
    overflow-y: hidden;
}
#streaming {
    object-fit: contain;
    width: 100%;
    height: 100vh;
```

```
}

.interfaceEsq {
    position: fixed;
    top: 0;
    right: 50%;
    width: 50%;
    height: 100vh;
    object-fit: cover;
    object-position: 30% 0;
}
#streaming2 {
    height: 100vh;
}

.interfaceDrt {
    position: fixed;
    top: 0;
    left: 50%;
    width: 50%;
    height: 100vh;
    object-fit: cover;
    object-position: 70% 0;
}
#streaming1 {
    height: 100vh;
}



/* Panel button areas */

.button-panel {
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
}

.top-right {
    top: 0;
    right: 0;
}

.top-left {
    top: 0;
    left: 0;
}
```

```
.center-left {
    top: 50%;
    left: 0;
    transform: translate(0, -50%);
}

.center-right {
    top: 50%;
    right: 0;
    transform: translate(0, -50%);
}

.bottom-left {
    bottom: 0;
    left: 0;
}

.bottom-right {
    bottom: 0;
    right: 0;
}

.horitzontal {
    display: flex;
}

/* Joystick */

.joystick {
    width: 20vw;
    height: 20vw;
}

/* Misc */

.botons.vert {
    position: relative;
    border: solid 1px;
    text-align: center;
    width: 20%;
    height: 100%;
}

.botons.horitz {
    position: relative;
    border: solid 1px;
    text-align: center;
    width: 20%;
    height: 100%;
```

```
}

/* Imatge */

img.streaming-on {
    /*transform: rotate(180deg);*/
    transform: rotate(0deg);
}
img.logoM{
    width: 400px;
    height: 100px;
}
img.imgA1{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 0px;
    right: 0px;
    z-index: 1;
}
img.imgA1M{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 0px;
    right: 0px;
    z-index: 1;
    width: 198px;
    height: 65px;
}
img.imgA2{
    position: absolute;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 30px;
    right: 30px;
    z-index: 3;
    width: 110px;
    height: 110px;
}
img.imgA2M{
    position: absolute;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 11px;
```

```
    right: 11px;
    z-index: 3;
    width: 45px;
    height: 45px;
}
img.imgA3{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 30px;
    right: 197px;
    z-index: 3;
    width: 110px;
    height: 110px;
}
img.imgA3M{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 11px;
    right: 78.5px;
    z-index: 3;
    width: 45px;
    height: 45px;
}
img.imgA4{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 30px;
    right: 366px;
    z-index: 3;
    width: 110px;
    height: 110px;
}
img.imgA4M{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 12px;
    right: 145px;
    z-index: 3;
    width: 45px;
    height: 45px;
}
```

```
.Temp{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 30%;
    right: 0%;
    z-index: 3;
    font-size: 1em;
    color: white;
}
.deep{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    top: 35%;
    right: 0%;
    z-index: 3;
    font-size: 1em;
    color: white;
}
.BAT{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    bottom: 38%;
    right: 1%;
    z-index: 3;
    font-size: 1em;
    color: white;
}
img.imgB0{
    position: fixed;
    user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
    bottom: 18%;
    right: -5%;
    z-index: 3;
    width: 10%;
    height: 20%;
}
/* Help overlayer */

.help {
    display: none;
```

```
    position: absolute;
    top: 0;
    left: 0;
    width: 100vw;
    color: white;
}

.helpbox {
    background: var(--button-background);
    padding: .5em 2em 2em 2em;
    border-radius: 1em;
    margin-top: 3em;
    z-index: 5;
}

.help-keyboard {
    /* yup */
}

.help-keyboard > .row {
    display: flex;
    margin-top: .3em;
    align-items: center;
}

.help-keyboard > .row > .keys {
    display: flex;
}

.key {
    border-style: solid;
    border-color: white;
    border-width: 1px;
    margin-right: .3em;
    padding: .2em;
    min-width: 1.2em;
    font-family: monospace;
    text-align: center;
}

.help-keyboard > .row > .description {
    width: 100%;
    margin-left: 1em;
}

.hide {
    display: none !important;
}
```

```
@media only screen and (orientation:portrait) {
    body {
      height: 100vw;
      transform: rotate(90deg);
    }
}
```

B. PROCÉS I ADAPTACIONS

Per tal de dissenyar i dur a terme aquest projecte s'ha seguit un procés d'aprenentatge, a base de preguntar, prova i error, veure altres robots... En aquest annex s'explica tot aquest procés al llarg del temps.

B.1. Aprenentatge del projecte anterior

A l'estiu de 2022 es va començar a aprendre com funcionava i com estava fet el projecte anterior a aquest, el robot R2B2, de la mà del seu autor. Durant aquest període es va aprendre com estava programat el robot, com s'havia dissenyat, com s'havia construït, quins eren els punts forts del robot i quins els més febles per poder-los millorar. A més, va ser la primera presa de contacte amb el Centre d'Investigació Robòtica Submarina (CIRS) de la universitat i amb tota la gent que treballa allà.

B.2. Aprenentatge d'altres ROV

A començament del curs 2022-23, es va començar a dissenyar el projecte a partir de les idees obtingudes de l'anterior projecte i de l'experiència adquirida durant l'estiu. A més, durant el mes d'octubre, es va col·laborar en el muntatge d'un robot submarí comprat per la mateixa universitat, el BlueROV2, de l'empresa Bluerobotics, i del qual també se'n van agafar idees i experiència. També va ser molt important poder preguntar a la gent del CIRS com es construeixen i com estan dissenyats els robots que es fan allà.

B.3. Comanda de material i proves amb l'electrònica

Després de recopilar tota aquesta informació es va acabar de dissenyar el robot i es va fer la comanda de material que faltava, ja que la major part del material ja era allà. Aquest fet ha influït en el desenvolupament del projecte, ja que, per exemple, poder fresar el metacrilat per fer les tapes de davant del robot ha fet que no es compressin les cúpules que portava l'anterior R2B2. Una altra de les adaptacions que s'ha fet és no comprar els nous cilindres i porta tòriques noves que ha dissenyat Bluerobotics, que porten corda de sallat, que fan que no s'obrin els encapsulats. Tampoc s'han comprat els nous penetrators de Bluerobotics, que no funcionen amb resina epoxi, sinó que funcionen com premsa estopes.

Durant els mesos de desembre i gener es va muntar tota l'electrònica i es va comprovar que tot funcionava correctament. Durant aquest període no hi va haver molts de problemes, el més complicat va ser aprendre com funcionava la Raspberry i configurar correctament el router perquè funcionés com es desitjava.

B.4. Fresat i muntatge de l'estructura

El mes de febrer es va aprendre com funcionava la fresadora que hi ha al CIRS, es va fresar tota l'estructura i es va muntar el Girona 25. Es van continuar fent proves amb l'electrònica per millorar el seu funcionament, incorporant noves funcionalitats i dissenyant tot el software. Aquest procés no van ser de gran dificultat, tot anava segons el previst. El fet més problemàtic va ser que un dels encapsulats acrílics, un dels superiors, era més petit del compte per un error de fabricació i el porta tòriques no entrava. Es va solucionar aquest problema llimant la part on van aquests porta tòriques fins que es va aconseguir que entressin.

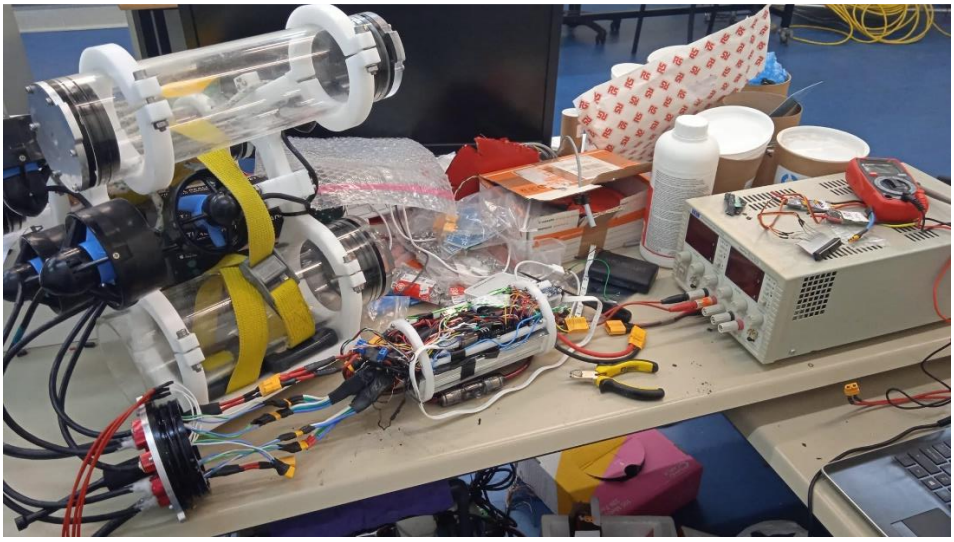

Figura 82. Proves amb el Girona 25 fora l'aigua.

B.5. Proves d'estanquitat

Va ser durant el mes de març quan van començar a sorgir més problemes, relacionats amb l'estanquitat. Es van començar a fer proves del robot a dins de l'aigua sense èxit, sempre sense electrònica a l'interior. Per sort es van poder detectar els problemes, observant just després de posar-lo a l'aigua, passant un paper sec pels llocs on es creia que podria haver entrat o fins i tot posant paper a dins l'encapsulat abans de llençar-lo

a l'aigua per veure per on quedava moll. Després de detectar els problemes, calia troba'ls-hi solució.

Les primeres proves a l'aigua hi havia fugues en dos dels tres encapsulats. El problema principal era que entrava aigua pels penetrators, entre els cables i l'epoxi, això va ser degut a la inexperiència i que la resina epoxi escollida era molt dura i no s'ajustava prou bé als cables. Per solucionar aquest problema es va passar a utilitzar una epoxi més tova, més líquida i de més qualitat que s'adherís millor. La manera de posar-la també es va canviar, es van buscar cables més flexibles i quan es posava epoxi abans es llimava una mica el cable, a la zona on s'havia d'enganxar, per crear una mica de rugositat i que l'expoxi s'hi pogués adherir correctament, després es netejava amb alcohol isopropílic per eliminar les grasses o qualsevol element que pogués fer que l'epoxi no s'agafés correctament. El fet que l'epoxi fos més líquid ajudava a que l'hora de posar-lo arribes a tots els racons i aconseguir que l'aigua no passes, una vegada sec. Inclús es feia un tap amb silicona a la sortida del penetrator per evitar que l'epoxi s'escapés per sota. Era important també que només els cables entressin dins l'encapsulat, per tal d'evitar que algun tall o algun forat a la funda d'aquests pogués fer que l'aigua passés entremig.

Figura 83. Muntatge de pesos per poder fer enfonsar el ROV.

Un cop solucionat el problema dels penetrators, a les següents proves encara entrava aigua a dins l'encapsulat inferior. Es va detectar que pels cables dels motors entrava aigua pel mig del coure. Per solucionar-ho, en comptes d'aïllar els connectors MR30, on sortia l'aigua dels cables dels motors, amb plàstic termoretràctil es va decidir aïllar-los amb epoxi, aconseguint aïllar-los tan elèctricament com evitar que l'aigua passes a dins de l'encapsulat. Una altra manera d'evitar aquest problema hauria pogut ser fer un tall en el cable i fer una soldadura dins el penetrator perquè no hi hagués continuïtat en el coure i l'epoxi retingues la possible aigua que passés per l'interior del cable. Una altra possible solució, més eficient, però més cara, és posar connectors subconn a tots els cables.

L'aigua entrava per dins els motors, però teòricament no hauria de ser així, ja que aquest model de motors són de més qualitat. El problema és que aquests motors ja havien sigut utilitzats en altres aplicacions anteriorment i feia temps que estaven allà al CIRS, fet que segurament els ha malmès.

Figura 84. Proves d'estanquitat a la piscina del CIRS.

El fet que entrés aigua pels connectors, abans de donar-se'n compte que hi havia aquest problema, va produir-ne un de major. Fent proves amb els motors fora l'aigua, en un dels connectors va quedar aigua, provocant un curtcircuit que va incendiar un dels controladors dels motors que es va haver de substituir per un altre. Fins que no es va veure el problema que l'aigua entrava pel coure dels cables dels motors es desconeixia la raó de l'incendi.

Figura 85. Driver cremat durant les proves amb el robot fora l'aigua.

Tot i solucionar aquest problema, encara entrava aigua a dins el mateix encapsulat inferior, era molt poca quantitat, però cal tenir en compte que aquest robot s'ha dissenyat per no haver-lo d'obrir més, per tant no hi pot entrar gens d'aigua. Després de fer bastants proves, es va poder veure que hi havia un problema a la junta de la tapa, ja que al ser fresada allà al CIRS i d'alumini verge i feia temps que era allà, tenia impureses i ratllades que no es veuen a simple vista i la tòrica no era suficient per a aturar l'aigua.

Es van proposar diferents solucions, una d'elles era rebaixar una capa de mig mil·límetre de la tapa amb la fresadora per fer eliminar aquestes impureses, però es va descartar degut a que ja estava tot muntat i soldat i, a més, la fresadora no és prou precisa com perquè quedi bé. La solució va ser utilitzar tefló, politetrafluoroetilè (PTFE), entre les dues parts, a més de la tòrica, per aconseguir tapar les irregularitats del material i que no entres aigua. El tefló és una espècie de cinta adhesiva, que normalment es col·loca entre dues rosques o dues juntes per evitar fugues d'aigua en canonades i claus de pas.

Encara va sorgir un últim problema d'estanquitat. Entre l'epoxi i els cables d'un dels motors continuava entrant aigua, segurament degut a que va quedar un camí pel que encara entrava aigua. Per solucionar-ho es va decidir fer un tall a la funda del cable a la part de fora de la tapa, de manera que quedessin els cables del motor a l'aire, i es va injectar epoxy just en aquest punt per tapar bé el forat i evitar que l'aigua pogués entrar. Per evitar que l'epoxy que es mogués mentre s'assecava, es va posar un tub i es va emplenar fins a dalt de tot. Amb això, es van solucionar tots els problemes d'estanuitat, i ja no entrava més aigua al robot.

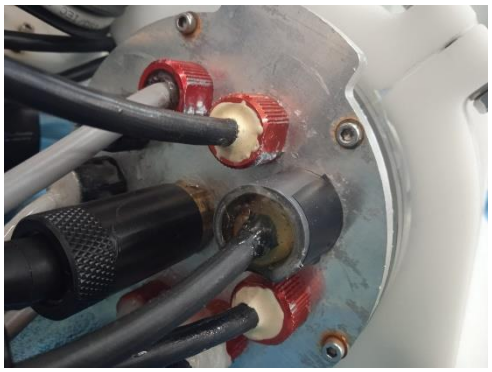

Figura 86. Driver cremat durant les proves amb el robot fora l'aigua.

B.6. Proves de funcionament

A finals d'arbil, una vegada solucionats tots els problemes d'estanquitat, es van començar a fer les proves amb el Girona 25, dins i fora l'aigua, modificant el codi i l'estructura del robot. Aquestes proves no van donar gaires problemes, el més complicat van ser els petits ajustos i modificacions del codi per acabar de millorar el funcionament del ROV. Els resultats de les proves van ser positius i es van gravar els vídeos corresponents, gràcies a l'ajuda de la gent del CIRS i el BlueROV que hi ha allà, ja que era l'única manera de fer una demostració del funcionament del Girona 25. Anar al mar també va ser tota una experiència. Gràcies un dels tutors, en Jordi Freixenet, i la seva família, es va poder posar el robot al mar i comandar-lo des d'una petita barca.

B.7. Oxidació de les tapes no anoditzades

El fet de no comprar tapes d'alumini anoditzades, va sorgir un darrer problema, no tan greu. L'alumini, com que no està anoditzat, es va començar a oxidar. Una possible solució al problema plantejat és la utilització d'un sistema amb ànode. El sistema amb ànode s'utilitza per localitzar l'oxidació d'un conjunt en un element fàcilment substituïble. En aquest cas, es suggereix la utilització d'un ànode de zinc (Zn), essent aquest un material químicament més noble que l'alumini (Al) i per tant, localitzant l'oxidació del conjunt en ell. Considerant, però, que aquesta modificació no és crítica pel desenvolupament dels objectius d'aquest treball, no s'ha considerat necessari la seva aplicació.

Figura 87. Tapa d'alumini oxidada.

C. MANUAL D'USUARI

En aquest document es descriu el funcionament bàsic del Girona 25 per a ser utilitzat per l'usuari de manera fàcil i senzilla.

C.1. Engegar i apagar el Girona 25

El primer que cal fer és engegar el robot i el sistema de comunicació per a poder connectar-s'hi. El sistema per engegar el robot, funciona amb interruptors magnètics. Per encendre'l cal acostar l'imant l'interruptor magnètic que té els cables de color verd, situat a la part esquerra de l'encapsulat inferior, i per apagar-lo, cal acostar-lo a l'interruptor que té els cables de color blau, situat a la part dreta de l'encapsulat inferior. Mentre s'utilitza el robot, no cal posar l'imant enlloc, simplement guardar-lo per a quan es vulgui apagar.

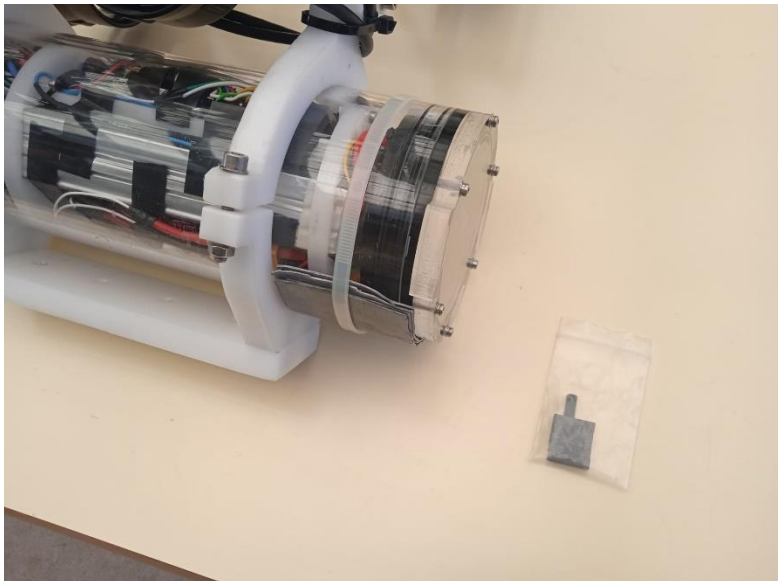

Figura 88. Imant que s'utilitza per engegar i apagar el robot.

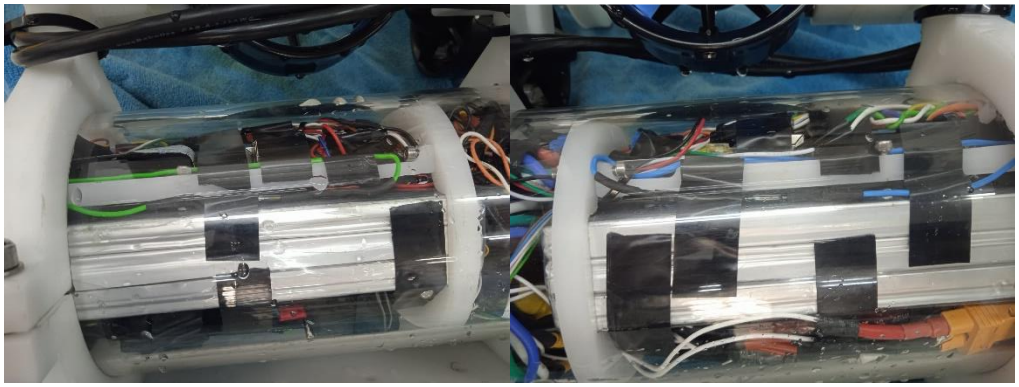

Figura 89. Interruptors magnètics on s'ha de posar l'imant per engegar i apagar el ROV.

El robot fa uns sons que indiquen que els motors tenen alimentació i les llums de l'interior, de la Raspberry i alguns sensors, s'encendran. Al cap d'aproximadament un minut, el programa per comandar el robot s'inicia, els motors tornen a sonar indicant que estan preparats per rebre ordres.

Abans d'apagar el robot, cal abans polsar el botó d'apagar des de la interfície, per tancar de forma correcta tots els programes i l'ordinador de control. Després de confirmar per tancar, el robot ja no es pot comandar des de la interfície i ja es pot acostar l'imant al costat corresponent per treure l'alimentació.

C.2. Engegar i apagar el router

Un cop engegat el robot, cal encendre el router per poder-s'hi connectar. Per fer-ho, cal obrir el taper i connectar el cable que surt de la bateria de la boia al router. A continuació, el WiFi s'inicialitzarà i ja es podrà tornar a tancar el taper.

Figura 90. Taper obert per poder connectar el cable d'alimentació al router.

Cal esperar que el robot s'hi connecti primer, és a dir, esperar que es sentin els sons dels motors que indiquen el programa s'ha iniciat. Això es fa perquè el router assigni al robot la primera IP: 192.168.0.100, i a la resta de dispositius que s'hi connectin, IP correlatives a aquesta, ja que els programes estan dissenyats perquè funcioni d'aquesta manera.

Quan es vulgui apagar el router, simplement cal tornar a obrir el taper i desconnectar el cable d'alimentació i el router s'apagarà. Es recomana desconnectar-lo després d'apagar el robot.

C.3. Carregar el robot

Per carregar el robot cal tenir-lo apagat, és a dir, haver passat abans l'imant per l'interruptor magnètic blau, que no es vegi cap llum encès a dins de l'encapsulat. Aleshores, cal treure el tap del connector subconn i posar-hi el connector que va amb el cable ja dissenyat per poder carregar la bateria. Els connectors que surten d'aquest cable cal connectar-los al corresponent lloc del carregador de bateries que s'utilitzi. A continuació es pot veure un exemple amb el carregador de bateries que hi ha al CIRS.

Figura 91. Carregador de bateries amb els connectors del subconn al corresponent lloc.

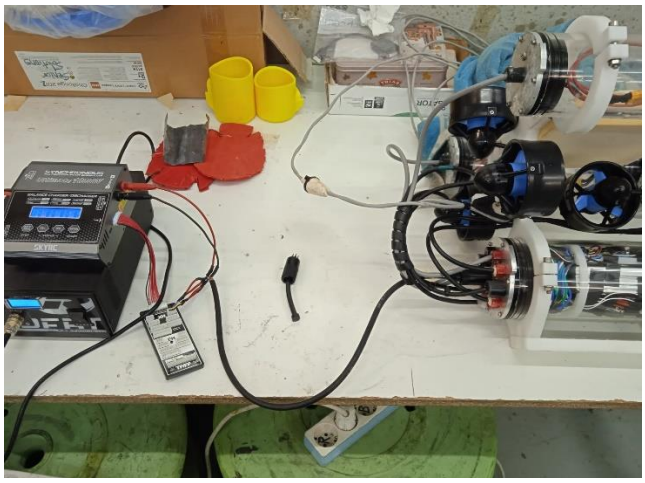

Figura 92. Exemple de com s'ha de carregar la bateria del robot.

Per carregar la bateria cal seleccionar, el tipus de bateria, Polímer de liti (LiPo), el número de cel·les, 3 cel·les, 3S, i la intensitat a la qual es vol carregar, es recomana entre 1 i 2 ampers.

És important connectar abans els cables al carregador de bateries, ja que si no es fa en aquest ordre, els connectors del final del cable poden tocar entre ells i fer curtcircuits. També cal recordar-se de tornar a posar el tap al connector quan s'acaba de carregar, sobretot abans de posar-lo a l'aigua.

C.4. Pes afegit per anar al mar

Si es desitja utilitzar el ROV per navegar al mar, com que l'aigua del mar té més densitat, la força que fa que el robot emergeixi és major. Per tant, cal afegir un pes al robot perquè continuï tenint la mateixa flotabilitat que a l'aigua dolça. Per aquest motiu, s'ha dissenyat un pes de plom que es pot afegir a la part de sota del robot.

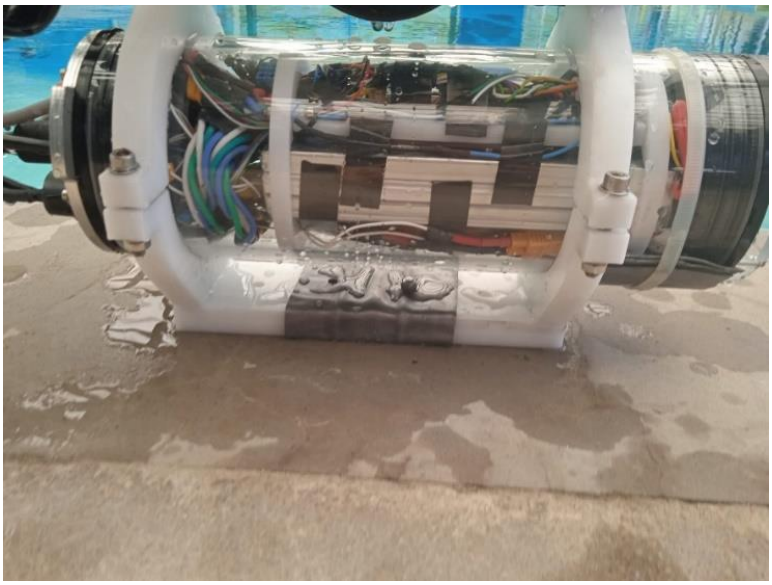

Figura 93. Pes que s'afegeix a l'estructura per navegar al mar.

Aquest pes, pel material que està fet, és molt modelable i se li pot donar forma. Per tant, amb la forma que se li dona i un cargol avellanat, perquè no sobresurti per la part de sota, es pot aguantar fàcilment a l'estructura. Hi ha més forats per si es volgués acollar més la peça o afegir més pesos, però no és necessari.

Figura 94. Pes de plom modelable que s'aguanta amb un cargol avellanat.

C.5. Comandament del robot

Per comandar cal utilitzar un dispositiu que tingui connexió WiFi, ja que el primer que cal fer és connectar-se a la xarxa del router del robot, el nom de la xarxa és el mateix que el del robot: "GIRONA 25". El router demana una contrasenya per accedir-hi, aquesta és "rogerfeliu".

Una vegada connectats a la mateixa xarxa que el robot, cal accedir a la interfície web per poder connectar-se realment al robot i poder-lo utilitzar. Per accedir a la interfície es pot fer de diferents maneres. Una és a través del robot, ja que comparteix la interfície amb el programa Apache24, per tant si s'escriu la IP 192.168.0.100 al navegador es pot accedir a la interfície. Una altra és a través de la pàgina web https://girona25.000webhostapp.com/. Per últim, i la més recomanable, és descarregar-se els arxius al dispositiu i obrir el fitxer "index.html". Es poden descarregar els arxius en el següent enllaç: https://github.com/Girona25/web_interface.io.

C.5.1. Interfície web

Quan s'accedeix a la pàgina web apareix aquesta primera pantalla, on cal prémer el botó de connectar per establir la connexió amb el robot.

Figura 95. Pàgina principal de la interfície web del Girona 25.

En aquesta primera pantalla, igual que en la següent, es pot passar a pantalla completa i hi ha una rodona vermella, que es torna de color verd quan es connecta un

comandament, com el Logitech Gamepad F310. Quan es connecta al robot apareix la següent pantalla:

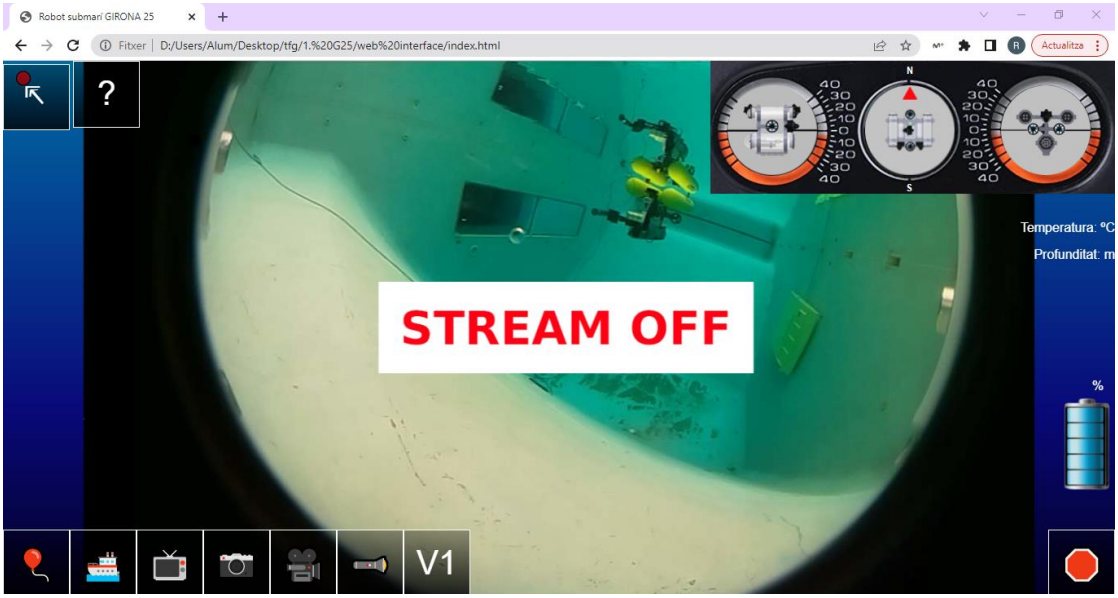

Figura 96. Pàgina de control de la interfície web del Girona 25.

Cadascun dels botons que apareixen a la pantalla té una funció en concret. El primer, el globus, com es pot interpretar, activa i desactiva el seguiment d'objectes de color groc. El segon, el vaixell, activa el mode de mantenir la profunditat, quan s'activa apareix una àncora per indicar que aquest mode està activat. Quan es detecta que la profunditat ha augmentat o disminuït s'envia una ordre per fer-lo pujar o baixar i mantenir aquesta profunditat inicial, mitjançant un controlador PI, sintonitzat empíricament.

El tercer botó, la televisió, inicia o pausa la transmissió de vídeo. La càmera fa una captura del vídeo en forma d'imatge. La càmera de vídeo permet gravar la pantalla, per fer un vídeo del que està veient el robot.

La llanterna encén els llums del ROV, mentre aquests estan engegats, el símbol canvia a una bombeta que fa llum per indicar-ho. El següent botó, el V1, indica la velocitat del robot. Hi ha dues velocitats V1, més lenta, i V2, més ràpida, s'alterna polsant-lo.

El senyal vermell atura per complet el robot, per poder-li desconnectar l'alimentació sense perill de malmetre l'ordinador de comandament. Apareix una pestanya on s'ha de confirmar que es vol realitzar aquesta acció, per evitar que es premi el polsador sense voler i el robot s'apagui quan no es desitja. El botó amb el símbol d'interrogant mostra en pantalla la informació del control per teclat.

Cal comentar que les emoticones per als botons són caràcters UTF-8, i per tant pot ser que es vegin diferents depenent del dispositiu en el qual s'utilitzi.

El control del robot per teclat es realitza de forma molt intuïtiva, les fletxes i les tres lletres W, A, S i D serveixen per a comandar-lo en el pla en què es trobi. La Q fa que el robot emergeixi i la E que s'enfonsi. Cal tenir en compte que el robot es mourà mentre la tecla estigui pressionada, en el moment en què es deixi de polsar, el robot atura el seu moviment. Els botons de la pantalla també es poden accionar per teclat, aquests comandaments queden explicats a la pantalla d'informació, que també es mostra amb la lletra H.

El control per teclat no és gaire eficient, per aquest motiu s'implementa el control a través del comandament Logitech Gamepad F310, on amb les palanques de control es mou el robot i amb els diferents polsadors es poden activar els botons de la pantalla.

A la pantalla, es pot observar el roll i el pitch del robot, per tal de poder comprovar que estigui en la posició correcta. També la direcció a la qual apunta, per si en algun moment es perd de vista el robot i l'orientació a través de la càmera no és suficient. A sota, es mostra la temperatura interior del robot, cal controlar-la per evitar sobreescalfaments, i la profunditat a la qual es troba. Per últim, també es mostra el nivell de bateria, per evitar quedar-se sense.

C.5.2. Interfície web per telèfons mòbils

Figura 97. Pàgina de control de la interfície web del Girona 25 per telèfons mòbils.

La interfície anterior és la que es mostra quan el dispositiu és un ordinador, però quan és un telèfon mòbil no es pot fer el comandament per teclat ni connectar un comandament. Per aquest motiu, quan s'obre la interfície web des d'un telèfon mòbil apareixen més botons i una palanca de control virtual per fer el comandament per pantalla.

La web per telèfons mòbils incorpora a més el polsador per utilitzar el mòbil amb ulleres de realitat virtual. Quan s'activa aquest botó, desapareixen la resta de botons, es comença a emetre vídeo, la pantalla passa a pantalla completa i la imatge queda tractada per tal de funcionar amb les ulleres de realitat virtual. En aquest mode, quan es mou el cap el robot fa els mateixos moviments que la persona que porta les ulleres, llegint el giroscopi que porta integrat el telèfon mòbil.

C.6. Documentació Girona 25

En cas que la informació que apareix en aquest document no sigui suficient, a continuació hi ha l'enllaç que permet accedir a tota la documentació relativa al Girona 25: https://github.com/Girona25/documentacio.

C.7. Raó del manual d'usuari

Per tal de que aquest robot el pugui utilitzat tothom qui ho necesiti al CIRS, s'ha decidit escriure aquest manual d'usuari per a explicar com s'ha d'utilitzar i com funciona aquest ROV. Aquest manual es trobarà en un document que s'hi podrà accedir a través del codi QR de la següent figura i es trobarà enganxat a l'interior del taper de la boia, de manera que es podrà llegir des de fora.

Figura 98. Codi QR per accedir al manual d'usuari.

D. FULL DE CARACTERÍSTIQUES

Nom: Girona 25
Descripció: La funció principal d'aquest robot és poder submergir-se en el medi aquàtic per aplicacions professionals, recerca o amb finalitats educatives. L'estructura modular, l'estabilitat, la potència i la bona navegabilitat permeten a aquest ROV complir amb qualsevol missió que se li proposi.

Altura: 40 cm
Amplada: 50 cm
Llargada: 40 cm
Diàmetre dels encapsulats superiors: 7,5 cm
Diàmetres de l'encapsulat inferior: 10 cm
Pes a l'aire: 10 kg
Profunditat màxima: 25 metres
Energia: 244Wh cel·les de polímer de liti
Duració: aprox. 4 h

Propulsió:
Número de motors: 5
Força màxima de cada motor: 1 kg

Sensors:
Sensor de profunditat, pressió i temperatura
Sensor de consum
Sensors d'orientació
Sensor d'aigua
Càmera
Llums LED

Comunicacions:
WiFi
Ethernet