ESCOLA NAVAL

talant de bien faire

Hilário Filipe Rocha Araújo

Projeto e desenvolvimento de um veículo autónomo biomimético de subsuperfície

Projeto e desenvolvimento do hardware e software

Dissertação para obtenção do Grau de Mestre em Ciências Militares Navais, na especialidade de Engenharia Naval Ramo de Armas e Eletrónica

Alfeite

2020

Hilário Filipe Rocha Araújo

Projeto e desenvolvimento de um veículo autónomo biomimético de subsuperfície

Projeto e desenvolvimento do hardware e software

Dissertação para obtenção do Grau de Mestre em Ciências Militares Navais, na especialidade de Engenharia Naval Ramo de Armas e Eletrónica

Orientação de: Professor Bruno Damas

Co-orientação de: Professor Vítor Viegas

O Aluno Mestrando,

Hilário Araújo

O Orientador,

Bruno Damas

Alfeite

2020

"Estudar e adaptar princípios utilizados na Natureza não fará de nós menos avançados tecnologicamente, mas sim, mais conscientes das nossas falhas e com novas perspetivas de futuro."

Mara Fernandes

Dedicada à minha família e à minha namorada, porque são o meu porto seguro e sempre acreditaram em mim!

Agradecimentos

A realização da presente dissertação de mestrado contou com o auxílio de inúmeras pessoas e, deste modo expresso o meu total reconhecimento por todo o apoio e tempo disponibilizado.

Á minha família por me incutirem valores, demonstrando-me que o sucesso apenas se alcança com esforço e dedicação e, acima de tudo respeitarem a minha ausência para a conclusão do trabalho.

À minha namorada Catarina que foi, desde cedo, compreensiva pelas horas que passava a desenvolver este projeto e que me deu coragem e força para seguir em frente. E nos momentos dúbios esteve sempre presente, expressando o seu carinho, dedicação e compreensão. Também à sua família que me apoiou e me encorajou desde o início.

Ao meu orientador, Professor Bruno Damas, pelo apoio, compreensão, motivação e supervisão prestados ao longo deste trabalho e pelas horas extraordinárias que me disponibilizava para encontrar possíveis soluções para o problemas encontrados.

Ao meu coorientador, Professor Vítor Viegas, , pelo seu apoio e dedicação, pelas diretrizes fornecidas ao longo de toda a jornada e pelo facto de estar sempre disposto a auxiliar-me nos momentos.

Ao João Gonçalves, pelas horas despendidas para o esclarecimento de dúvidas, amizade e pelos conselhos no desenvolvimento da arquitetura em *Robot Operating System*. Sem a sua ajuda, o caminho percorrido seria muito mais atribulado.

Ao *Amaok*, programador chinês pela pronta disponibilidade para, em parceria, desenvolvermos uma biblioteca I^2C inovadora para o sistema *Linux*.

Ao Destacamento de Mergulhadores Sapadores nº3, especialmente ao Sr. Sargento Barata, pela ajuda e explicação detalhada sobre os *Autonomous Underwater Vehicle* utilizados pela Marinha Portuguesa.

Ao Sr. Engenheiro Albino Pina, que durante o estágio de embarque me ajudou a definir o conceito do projeto e me foi impondo métodos de trabalho para a realização do mesmo. Também pela preocupação demonstrada durante e após o estágio.

Aos membros da minha camarata, Gonçalo Dias de Paiva, Gonçalo Luís Ferreira, Filipe Martins Guilherme, pelo apoio, irmandade, preocupação e carinho que desde o início da Escola Naval demonstraram para comigo.

A todos os membros do curso João Baptista Lavanha, pelas experiências, camaradagem e entre-ajuda durante estes cinco anos.

Aos Professores militares e civis do Departamento de Ciências e Tecnologia da Escola Naval, por ministraram conhecimentos indispensáveis para a realização deste trabalho, nomeadamente os Professores das unidades curriculares: Eletrotecnia, Eletrónica, Arquitetura de Computadores, Sistemas Digitais, Tecnologia e Medidas Elétricas, Máquinas Elétricas, Fundamentos de Telecomunicações e Sistemas de Apoio à Decisão.

Resumo

A presente dissertação de mestrado enquadra-se na continuação do projeto europeu *Swarm of Biomimetic Underwater Vehicles*. Na anterior etapa do projeto, ao nível nacional, foi construído e desenvolvido um veículo biomimético "mãe" apelidado Petinga.

A segunda etapa do projeto, em fase de submissão, pretende desenvolver um *swarm* que contenha outros veículos, com menores capacidades operacionais e de mais baixo custo e que permitam a execução de missões de forma cooperativa. Um desses veículos será o protótipo descrito ao longo desta dissertação. Desta forma será possível utilizar um cardume de veículos biomiméticos em missões operacionais da Marinha Portuguesa, tais como *Intelligence, Surveillance and Reconnaissance*, *Mine Counter Measures*, *Anti-Submarine Warfare*, investigações hidrógrafas e oceanógrafas, entre outras. Nesse contexto, pretende-se construir mais veículos biomiméticos, de baixo custo, que permitam a execução de missões de forma cooperativa.

Neste trabalho pretende-se construir o primeiro protótipo para um desses veículos, nas seguintes áreas: escolha de sensores, sistemas de comunicações e sistema de processamento; concepção da arquitectura de software que equipará o veículo bem como definição do *middleware* a utilizar; elaboração de drivers para os sensores e sistemas de comunicações que equipam o veículo; e integração e testes.

O primeiro protótipo é um veículo biomimético de subsuperfície (em inglês *Biomimetic Underwater Vehicle*) chamado Tobias. Foi construído para ser utilizado na costa portuguesa a uma profundidade máxima de 10 metros, com uma autonomia de 2 horas e com comunicação acústica, WiFi e *Global System for Mobile Communications*. Com perfil biomimético, possui uma cauda, constituída por dois servos e duas barbatanas dorsais que são atuadas por um servo cada uma. Para controlo da flutuabilidade será utilizado um sistema de bexiga artificial. Ao nível dos sensores deverá ser equipado com um sonar frontal para evitar obstáculos, uma câmara, um sensor inercial, um sensor de pressão, um recetor *Global Navigation Satellite System*. Como unidade de comando e controlo utilizar-se-à a placa *Jetson Nano* que correrá

o software. A *framework* adotada para desenvolvimento do software do veículo será o *Robot Operating System*.

Palavras-chave: AUV, Veículos Biomiméticos, *Software*, *Hardware*, ROS

Abstract

This Master's dissertation is part of the continuation of the European project Swarm of Biomimetic Underwater Vehicles. In the previous stage of the project, on the Portuguese side, was built and developed a biomimetic vehicle "mother" nicknamed Petinga.

The second stage of the project, in submission phase, intends to develop a swarm that contains other vehicles, with lower operational capacities and lower cost, that allow the execution of missions in a cooperative way. One of these vehicles will be the prototype described throughout this dissertation. This way it will be possible to use a biomimetic vehicles in operational missions of the Portuguese Navy, such as Intelligence, Surveillance and Reconnaissance, Mine Counter Measures, Anti-Submarine Warfare, hydrographic and oceanographic research, among others. In this context, it is intended to build more biomimetic vehicles, at low cost, that allow the execution of missions in a cooperative way.

This work will deal with the construction of the first prototype for one of those vehicles, in the following areas: choice of sensors, communication systems and processing system; design of the software architecture that will equip the vehicle as well as definition of the middleware to be used; elaboration of drivers for the sensors and communication systems that equip the vehicle; integration and tests.

The first prototype is a Biomimetic Underwater Vehicle (BUV) called Tobias. It was built to be used on the Portuguese coast at a maximum depth of $10m$, with an autonomy of $2h$ and with acoustic communication, Wireless Fidelity and Global System for Mobile Communications. With a biomimetic profile, it has a tail, consisting of two servos, and two dorsal fins that are actuated by one servos each. For buoyancy control an artificial bladder system will be used. In sensory terms it will have to be equipped with a front sonar to avoid obstacles, a camera, an inertial sensor, a pressure sensor, an Global Navigation Satellite System receiver. As a command and control unit, the software will run the Jetson Nano board. The framework adopted to develop the vehicle software will be the Robot Operating System.

Keywords: AUV, Biomimetic Vehicles, Software, Hardware, ROS

Índice

Lista de Figuras

Lista de Tabelas

Lista de Código Fonte

Lista de Acrónimos

AHRS	*Attitude and Heading Reference System.*
AMCL	*Adaptive Monte Carlo Localization.*
ASPOF	Aspirante a Oficial.
ASV	*Autonomous Surface Vehicle.*
ASW	*Anti-Submarine Warfare.*
AT	*ATtention.*
AUG	*Autonomous Underwater Glider.*
AuRA	*Autonomous Robot Architecture.*
AUV	*Autonomous Underwater Vehicle.*
BAUV	*Biomimetic Autonomous Underwater Vehicle.*
BCU	*Buoyancy Control Unit.*
BDS	*BeiDou Navigation Satellite System.*
BMT	*British Maritime Technology.*
BUUV	*Biomimetic Unmanned Underwater Vehicle.*
BUV	*Biomimetic Underwater Vehicle.*
CARMEN	*Carnegie Mellon Navigatio.*
CINAV	Centro de Investigação Naval.
CLARAty	*Coupled Layer Architecture for Robotic Autonomy.*
CMake	*Cross Plataform Make.*
CN3	*Communication/Navigation Network Nodes.*
CPG	*Central Pattern Generator.*
CPU	Unidade Central de Processamento (*Central Processing Unit*).
CRIC	*Chief of Naval Operations' Rapid Innovation Cell.*
CSI	*Camera Serial Interface.*
CTD	*Conductivity, Temperature and Depth.*

DARPA	*Defense Advanced Research Projects Agency.*
DGM	Destacamento de Guerra de Minas.
DMS3	Destacamento de Mergulhadores Sapadores nº3.
DSEI	*Defence & Security Equipment International.*
DTE	*Data Terminal Equipment.*
DUNE	*DUNE Uniform Navigation Environment.*
DVL	*Doppler Velocity Log.*
EDA	*European Defence Agency.*
EGNOS	*European Geostationary Navigation Overlay Service.*
EN	Escola Naval.
EN-MEC	Engenheiro Naval - Ramo de Mecânica.
ENU	*East, North, Up.*
ESA	*European Space Agency.*
EUA	Estados Unidos da América.
FaaS	*Fish as a Service.*
FEUP	Faculdade de Engenharias da Universidade do Porto.
FOURCC	*Four Character Code.*
FTDI	*Future Technology Devices International.*
FTP	*File Transfer Protocol.*
GAL	*Galileo.*
GFLOPS	*Giga Floating-point Operations Per Second.*
GLONASS	*Global Navigation Satellite System.*
GNSS	*Global Navigation Satellite System.*
GNU	*GNU's Not Unix!.*
GPIO	*General Purpose Input/Output.*
GPRS	*General Packet Radio Service.*
GPS	*Global Positioning System.*
GSM	*Global System for Mobile Communications.*
GUI	*Graphical User Interface.*
HD	*High-Definition video.*

HOV	*Human Occupied Vehicles.*
HSDPA	*High-Speed Downlink Packet Access.*
HTML	*HyperText Markup Language.*
HTTP	*Hypertext Transfer Protocol.*
I&D	Investigação e Desenvolvimento.
I^2C	*Inter-Integrated Circuit.*
IA	Inteligência Artificial.
ID	*Inspection/Identification.*
IDE	*Integrated Development Environment.*
IMRS	*Intelligent Mobile Robot System.*
IMU	*Inertial Measurement Unit.*
INS	*Inertial Navigation System.*
IO	*Information Operations.*
IRNSS	*Indian Regional Navigation Satellite System.*
ISM	*Industrial, Scientific and Medical.*
ISR	*Intelligence, Surveillance and Reconnaissance.*
L-BAUV	*Swarm Leader Biomimetic Autonomous Underwater Vehicle.*
L4T	*Linux for Tegra.*
LA	Limitação de Avarias.
LBL	*Long Baseline Navigation System.*
LSTS	Laboratório de Sistemas e Tecnologia Subaquática.
M-BAUV	*Swarm Member Biomimetic Autonomous Underwater Vehicle.*
MCM	*Mine Counter Measures.*
MDN	Ministério da Defesa Nacional.
MIPI	*Mobile Industry Processor Interface.*
MIT	*Massachusetts Institute of Technology.*
MOOS	*Mission Oriented Operating Suite.*
MRDS	*Microsoft Robotics Developer Studio.*
NASREM	*NASA/NBS Standard Reference Model.*

NATO	*North Atlantic Treaty Organization.*
NED	*North, East, Down.*
NMEA	*National Marine Electronics Association.*
OOP	*Object-Oriented Programming.*
OpenCV	*Open Source Computer Vision Library.*
Orocos	*Open Robot Control Software.*
OTAN	Organização do Tratado do Atlântico Norte.
PAG	*Port Authority of Gijón.*
PC	*Personal Computer.*
PCB	*Printed Circuit Board.*
PCL	*Point Cloud Library.*
PID	*Proportional–Integral–Derivative.*
PLA	Poliácido Láctico.
PoE	*Power over Ethernet.*
PWM	*Pulse-Width Modulation.*
QZSS	*Quasi-Zenith Satellite System.*
RECON	Reconhecimento.
REP	*Rapid Environment Picture.*
ROS	*Robot Operating System.*
ROV	*Remotely Operated Vehicle.*
RTCM	*Radio Technical Commission for Maritime Services.*
RX	*Receiver.*
SABUVIS	*Swarm of Biomimetic Underwater Vehicles.*
SBAS	*Satellite-Based Augmentation System.*
SBL	*Short Baseline Navigation System.*
SBS	*System Breakdown Structure.*
SCL	*Serial Clock Line.*
SDA	*Serial Data Line.*
SHARD	*Sea Hunting Autonomous Reconnaissance Drone.*
SLAM	*Simultaneous Localization and Mapping.*

SMS	*Short Message Service.*
SO	Sistema Operativo.
SoFi	*Soft Robotic Fish.*
SONAR	*Sound Navigation and Ranging.*
SPA	*Sense, Plan and Act.*
SSH	*Secure Socket Shell.*
SSRC	*Ship Stability Research Centre.*
SSS	*Side Scan Sonar.*
STANAG	*NATO Standardization Agreement.*
SVS	*Sound Velocity Sensor.*
TCP	*Transmission Control Protocol.*
TCS	*Time Critical Strike.*
TE	*Terminal Equipment.*
TX	*Transmitter.*
UART	*Universal Asynchronous Receiver-Transmitter.*
UAV	*Unmanned Aerial Vehicle.*
UDP	*User Datagram Protocol.*
UE	União Europeia.
USB	*Universal Serial Bus.*
USBL	*Ultra-Short Baseline Navigation System.*
USSD	*Unstructured Supplementary Service Data.*
UUV	*Unmanned Underwater Vehicles.*
UWSim	*UnderWater SIMulator.*
W-CDMA	*Wide-Band Code-Division Multiple Access.*
WAAS	*Wide Area Augmentation System.*
WHOI	*Woods Hole Oceanographic Institute.*
WiFi	*Wireless Fidelity.*
XML	*Extensible Markup Language.*
YAML	*YAML Ain't Markup Language.*
YARP	*Yet Another Robot Platform.*

Capítulo 1

Introdução

A exploração no alto mar é considerada uma tarefa desafiante devido à complexidade e perigosidade do ambiente subaquático. A crescente indústria de veículos subaquáticos não tripulados estimulou os cientistas e investigadores a superar os desafiantes problemas científicos e de engenharia associados ao ambiente subaquático não estruturado e perigoso. Os veículos não tripulados de sub-superfície (em inglês *Unmanned Underwater Vehicles* (UUV)) têm sido amplamente utilizados na exploração do fundo do mar, reparação, levantamento, inspeção cascos de navios, exploração de petróleo e gás, contra-medida de minas, busca e salvamento e recolha de dados científicos (Akçakaya et al., 2009; M. L. Seto, Paull & Saeedi, 2013; Tan, Sutton & Chudley, 2007; Zhu & Sun, 2013). No entanto, a utilidade dos *Unmanned Underwater Vehicles*s (UUVs) não se limita a aplicações no alto mar, isto é, podem ser aplicados em ambientes de águas rasas, como, por exemplo, rios e lagos para tarefas de inspeção e recolha de amostras. Isto desencadeou o desenvolvimento de projetos conceituais, produção e operação de UUVs de menor escala e de baixo custo (Bellingham et al., 1994; W. H. Wang et al., 2008).

1.1 Motivação

Ao longo dos últimos anos, a utilização de veículos não tripulados para realização de determinadas missões tem sido cada vez mais requisitado. Isto deve-se ao facto de:

- Os veículos não tripulados serem mais baratos de operar e manter do que veículos tripulados;
- A utilização de sensores permitir a monitorização de vários parâmetros simultaneamente;
- A vigilância, por parte destes veículos, ser contínua e persistente;

- As plataformas não tripuladas poderem melhorar a produtividade, permitindo que as pessoas, os bens mais valiosos de qualquer organização, possam ser utilizadas noutras tarefas;
- Os veículos não tripulados garantirem a segurança dos seres humanos e das plataformas tripuladas afastadas do perigo.

Os UUVs têm assumido uma elevada importância e uma mais valia para as marinhas militares devido à versatilidade das missões que podem desempenhar (U.S. Navy, 2004), tais como:

- *Intelligence, Surveillance and Reconnaissance* (ISR), como é o caso do *Remotely Operated Vehicle* (ROV) SHRIMP (Figura 1.1) (Muljowidodo et al., 2009);
- *Mine Counter Measures* (MCM), como é o caso do *Autonomous Underwater Vehicle* (AUV) REMUS (Figura 1.2) (Stokey et al., 2001);
- *Anti-Submarine Warfare* (ASW) (Nicolai, 2002);
- *Inspection/Identification* (ID) [1] como é o caso do UUV DRIP (Figura 1.3) (Miller, 1996);
- Oceanografia e hidrografia (Evans et al., 1999; Oliveira, 2018);
- Transporte de cargas (C. Brown & Clark, 2010);
- *Time Critical Strike* (TCS) (Lala & Harper, 1994);
- *Communication/Navigation Network Nodes* (CN3) (Thomas D. Barron, 1998) [2];
- *Information Operations* (IO) [3].

[1] Consiste na inspeção, localização e identificação de objetos em áreas confinadas tais como cais, portos e fundeadouros (U.S. Navy, 2004). O objetivo destas operações é reconhecer rapidamente áreas de preocupação e para detetar e investigar possíveis objetos explosivos que impõem uma ameaça para as forças militares, vias navegáveis, ativos de alto valor e segurança interna.

[2] Os sistemas CN3, foram adotados pela primeira vez pela *US Navy* e têm como principal objetivo estabelecer a conectividade em várias plataformas, tripuladas e não tripuladas, além de dar assistência à navegação (U.S. Navy, 2004).

[3] Consiste em operações militares que visam explorar, enganar, deter e perturbar determinados alvos inimigos (U.S. Navy, 2004).

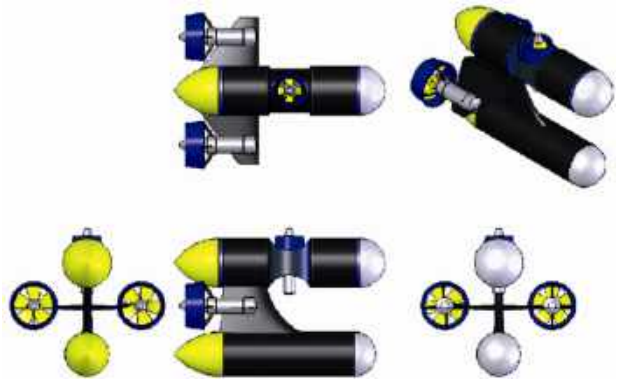


FIGURA 1.1: Visão 3D do UUV SHRIMP.
Fonte: retirado de Muljowidodo et al. (2009).

FIGURA 1.2: Visão 3D do AUV REMUS.
Fonte: retirado de Stokey et al. (2001).

FIGURA 1.3: Visão 3D do UUV DRIP.
Fonte: retirado de Miller (1996).

1.2 Porquê um veículo subaquático biomimético?

Os sistemas baseados em hélices dos UUVs existentes são projetados eficientemente para uma missão (por exemplo, mapeamento do fundo do mar, levantamento hidrográfico) que exija uma velocidade de cruzeiro elevada. No entanto, em certas aplicações, o sistema baseado na hélice pode ser ineficiente, por se observar que este sistema tem um mau desempenho de manobra (isto é, não permite manobrar o veículo a baixa velocidade e tem um elevado raio de giração). Por este motivo

alguns UUVs começaram a ser configurados com múltiplos propulsores à volta do veículo para permitir que este manobre a baixas velocidades (Abreu et al., 2016; Allotta et al., 2015; Chocron & Delaleau, 2019; He et al., 2013; Vega et al., 2015; X. Wang & Liang, 2019). No entanto, esta configuração aumenta o consumo de energia do veículo, que, por sua vez, reduz a sua autonomia.

A procura por tecnologia energicamente mais eficiente (Guo et al., 1998; Rossi et al., 2011), menos ruidosa e com melhor manobrabilidade (Liu & Hu, 2005) inspirou os cientistas e engenheiros a aplicar os princípios biomiméticos dos peixes ao design dos UUVs, onde as hélices são substituídas por propulsão biomimética, isto é, por uma cauda que imita o movimento ondulatório de um peixe. Desta forma, os UUV biomiméticos são mais silenciosos (mais difíceis de detetar) que os sistemas clássicos. podem ter maior autonomia que os sistemas clássicos e, ao terem movimentos semelhantes a peixes, são mais difíceis de detetar quando comparados com os sistemas clássicos.

No mundo militar, apresentam diversas vantagens inerentes às suas capacidades quando comparadas com as dos humanos, designadamente: podem operar em situações perigosas sem pôr a vida humana em risco, podem ser utilizados em situações de desastres químicos/radiológicos, têm custo relativamente baixo, uma vez que não necessitam de mantimentos e não despendem recursos, ao nível dos vencimentos, são versáteis, podem operar em diversos meios, entre outras. Para além disso, com a redução de militares no ativo na Marinha Portuguesa e com a redução das unidades navais disponíveis, torna-se necessário desenvolver um sistema capaz de patrulhar a costa portuguesa com baixo custo de manutenção e com capacidade e comunicação com outros veículos e unidades navais.

1.3 *Swarm of Biomimetic Underwater Vehicles* (SABUVIS)

O projeto *Swarm of Biomimetic Underwater Vehicles* (SABUVIS) foi aprovado pela *European Defence Agency* (EDA), cujo principal objetivo é desenvolver e utilizar os *Biomimetic Underwater Vehicle*s (BUVs) em missões de *stealth data collection* e *surveillance*, onde Portugal, Polónia e Alemanha fazem parte do consórcio.

A Escola Naval (EN) - Centro de Investigação Naval (CINAV) tem desenvolvido um projeto de veículos autónomos de subsuperfície em cooperação com a

Laboratório de Sistemas e Tecnologia Subaquática (LSTS) - Faculdade de Engenharias da Universidade do Porto (FEUP), a *OceanScan - Marine Systems and Technology* no âmbito do projeto internacional SABUVIS. Este protocolo de cooperação foi assinado a 21 de Agosto de 2017 entre as instituições supra mencionadas, o Ministério da Defesa Nacional (MDN) e a EN - CINAV.

O projeto *Swarm of Biomimetic Underwater Vehicles* (SABUVIS) enquadra-se na procura de uma solução para os problemas evidenciados anteriormente e tem como objetivo projetar e desenvolver veículos autónomos biomiméticos de subsuperfície.

Este projecto teve como objetivos:

1. Desenvolver tecnologia e *know-how* que permita a uma empresa portuguesa alargar a sua oferta de veículos submarinos para incluir veículos biomiméticos.
2. Desenvolver técnicas de aprendizagem por reforço para controlar veículos autónomos submarinos.
3. Desenvolver um simulador de UUV biomimético e ambiente de desenvolvimento para facilitar o desenvolvimento de técnicas de aprendizagem por reforço.
4. Realizar testes e desenvolver conceitos de operação para veículos biomiméticos.
5. Dotar as Forças Armadas de um protótipo de um UUV biomimético que possa ser usado operacionalmente.
6. Identificar mais-valias operacionais deste tipo de sistemas e quais os desenvolvimentos futuros a perseguir.
7. Estreitar os laços de cooperação entre a Escola Naval (Escola de Ensino Superior Universitário Militar, integrada no Instituto Universitário Militar) com uma das maiores escolas congéneres a Escola Naval Polaca.

1.3.1 SABUVIS I

O projeto SABUVIS I foi concluído em 2019 e dele resultaram três tipos de veículos:

- *BUV 1* - Um veículo autónomo subaquático inspirado numa foca, figura 1.4. O veículo do tipo foca é composto por duas laterais e uma cauda composta por duas barbatanas menores, tentando simular a modalidade de bruços da natação. Este veículo foi desenvolvido pela Escola Naval Polaca;

FIGURA 1.4: *BUV 1* do projeto SABUVIS em preparativos para testes.
Fonte: obtido no *website* da *Hisutton*[4].

- *BUV 2* - Este robô é também um veículo autónomo subaquático, no entanto, simula um peixe, figura 1.5. O veículo tipo peixe é integrado por duas barbatanas laterais e por uma cauda, composta por apenas um membro seccionado em duas partes para dar mais fluidez ao movimento. Este BUV Foi desenvolvido pela Universidade de Cracóvia;

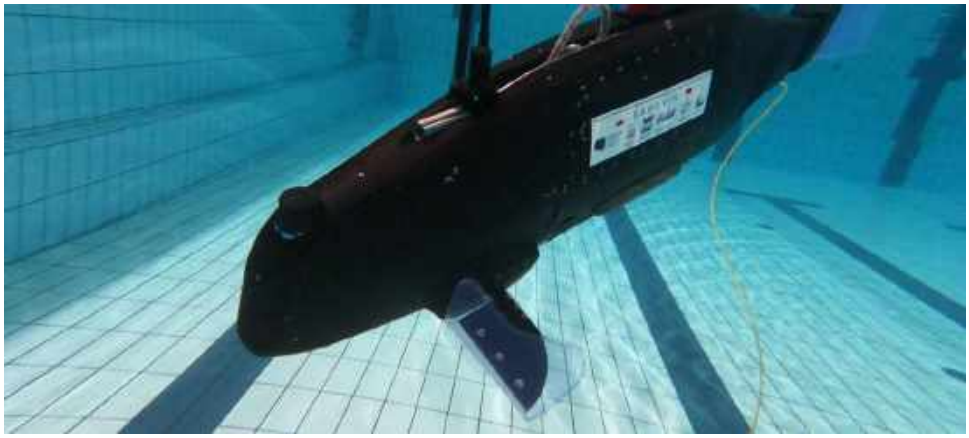


FIGURA 1.5: *BUV 2* do projeto SABUVIS em preparativos para testes.
Fonte: obtido do *website* *Naval News - SABUVIS I*[5].

- *BUV 3* - Este veículo subaquático tem formato semelhante aos dois supramencionados, como se pode observar na figura 1.6, mas a sua cauda é intercambiável entre uma cauda tipo "peixe" (figura 1.7a) e uma cauda tipo "foca" (figura

[4] *Website* da *Hisutton*: http://www.hisutton.com/Biomimetic_Autonomous_Underwater_Vehicles.html.
[5] *Website* da *Naval News - SABUVIS I*: https://www.navalnews.com/naval-news/2019/10/eda-expands-swarming-biomimetic-uuv-development-program/.

1.7b). A grande diferença do trabalho desenvolvido no BUV 3, em compararão aos outros dois, foi a tentativa de aplicação de técnicas de aprendizagem por reforço para o controlo do movimento do veículo. Este foi desenvolvido pelas diferentes instituições portuguesas suprareferenciadas, onde a *OceanScan* foi a responsável pela construção mecânica do veículo, o LSTS pelo desenvolvimento da instrumentação e sensores do veículo e o CINAV por desenvolver os controladores de baixo nível e a Inteligência Artificial (IA) do veículo. Neste veículo também existe a possibilidade de encaixar também uma das caudas tradicionais que equipam os *SeaCon* da Marinha (para comparar propulsão biomimética com propulsão clássica a hélice).

FIGURA 1.6: *BUV 3* do projeto SABUVIS em exposição na EN.

(A) Cauda em forma de peixe. (B) Cauda em forma de foca.

FIGURA 1.7: Tipos de caudas do *BUV 3*.

1.3.2 SABUVIS II

O principal objetivo do projeto SABUVIS II é desenvolver tecnologia para o controlo de um cardume de veículos de subsuperfície biomiméticos heterogéneos e fortemente cooperativos (*Research Technical Proposal - SABUVIS II*, 2018).

Os principais requisitos que este projeto pretende satisfazer são: disponibilizar um sistema de baixo custo, de reduzidas dimensões e baixo peso. A nível técnico seria de intervenção fácil, no que diz respeito a aspetos como expansão, interação e manutenção. Como este veículo será o primeiro protótipo que se pretende desenvolver possui requisitos operacionais "suaves", isto é, a profundidade, autonomia e velocidade máximas do veículo serão reduzidas.

Independentemente da tarefa, assume-se que a constituição do cardume consista em pelo menos dois tipos de veículos, ou seja, os líderes (*Swarm Leader Biomimetic Autonomous Underwater Vehicle* (L-BAUV)) e os membros do cardume (*Swarm Member Biomimetic Autonomous Underwater Vehicle*s (M-BAUVs)). Os líderes serão equipados com sistemas de navegação precisos e sistemas de comunicação de longo alcance, trabalhando principalmente no modo *output*, isto é, irão apenas distribuir a informação da missão e tarefas pelos restantes membros do cardume. Estes últimos deverão possuir um sistema de navegação mais simples e de baixo custo, um sistema de localização relativa dentro do cardume e o um sistema de comunicação de baixo alcance, trabalhando principalmente no modo *input*, isto é, deverão apenas receber a informação da missão e tarefas dos líderes e executá-las. Como o nome indica, os líderes serão utilizados principalmente para liderar todo o cardume por um determinado caminho, enquanto a tarefa dos membros é seguir a ordem dos líderes.

Seguir um trajeto mantendo a formação ou disposição dos veículos é a condição necessária para lidar com o cardume, sendo que esta tarefa é a principal dos veículos pertencentes a este projeto. Em caso de êxito na execução desta tarefa, outras tarefas mais específicas podem ser consideradas consideradas, como por exemplo, uma parte da missão MCM pode ser delegar uma parte do cardume para inspecionar um objeto detectado. Após a inspeção, os veículos devem retornar ao cardume principal e, de seguida, todos os veículos devem continuar a missão interrompida. Outro exemplo de tarefa específica é distribuir todos os membros do cardume para procurar possíveis alvos da missão de combate. Nesse caso, a etapa final da missão é realizada separadamente por todos os membros do cardume. Para reduzir o custo do projeto, supõe-se que membros do cardume, cujos equipamentos em condições reais

devam ser ajustados à tarefa específica (os líderes são responsáveis pela navegação enquanto que os membros são responsáveis pela tarefa) carregam apenas sensores e dispositivos necessários para formar um cardume.

O objetivo da próxima fase do projeto é preparar todos os componentes individuais de hardware e software do cardume para integração e testes. A lista de ações nesta fase é apresentada abaixo:

- O hardware:
  - Análise dos sensores e dispositivos disponíveis em termos de usabilidade no cardume e, posterior, seleção final de sensores e dispositivos;
  - Análise das plataformas subaquáticas existentes, em termos da sua aplicabilidade enquanto líderes do cardume, projeto e implementação de possíveis modificações e testes iniciais de todos os sistemas separados;
  - Projeto e implementação dos membros do cardume, projeto mecânico e elétrico, integração com sensores e dispositivos selecionados e testes iniciais de todos os sistemas separados;
  - Concepção e implementação de bases de pesquisa estacionárias para testar sistemas de comunicação/localização acústica e ótica e testes dos sistemas.
- O software:
  - Concepção e implementação de um ambiente de simulação para testar a funcionalidade do software de alto nível *Biomimetic Autonomous Underwater Vehicle* (BAUV), responsável pela formação de cardumes e outras tarefas de cardume;
  - Projetar, implementar e testar o software de alto nível do BAUV, em condições de simulação, responsável pela operação de cardume;
  - Projetar e implementar o controlo de baixo nível para membros do cardume, testes em separação;
  - Adaptação do software de alto nível do BAUV, implementado no SABUVIS I, em testes separados;
  - Adaptação e testes do sistema de comunicação implementado no SABUVIS I, projeto e implementação de sistema de comunicação entre membros do cardume;

- Concepção e implementação de sistema de navegação para os membros de cardume e ensaios em separação;
- Projeto, implementação e testes do sistema de localização dentro do cardume;
- Adaptação e testes do sistema de prevenção de colisões (passivo e ativo) implementados no SABUVIS I e testes em separação;
- Adaptação da estação de comando e controle projetada no SABUVIS I.

1.4 Objetivo da dissertação

O projeto SABUVIS II possui duas vertentes, sendo a primeira a do hardware e software, e a segunda a da vertente do projeto da plataforma e design.

Esta dissertação de mestrado enquadra-se na primeira vertente do projeto SABUVIS II e tem como objetivo a:

- Análise e seleção de sensores, sistemas de comunicações e sistemas de processamento;
- Definição do *middleware* e linguagem de programação a utilizar;
- Conceção da arquitetura de software que equipará o veículo;
- Elaboração de drivers para os sensores e sistemas de comunicação;
- Integração dos diversos sensores e testes.

A segunda vertente resume-se ao desenho mecânico, projeto das condições de estanqueidade e flutuabilidade, seleção da eletrónica de potência, baterias e motores, e foi desenvolvido pelo camarada Aspirante a Oficial (ASPOF) Engenheiro Naval - Ramo de Mecânica (EN-MEC) Dias de Paiva. O relatório do desenvolvimento do seu trabalho poderá ser consultado na sua dissertação de mestrado.

1.5 Estrutura da Dissertação

A organização do documento procura refletir todas as fases do desenvolvimento do trabalho realizado. Após a introdução elabora-se uma análise do Estado de Arte (capítulo 2), no âmbito da robótica submarina, focando, principalmente, os BUVs. Neste capítulo ainda são definidos conceitos e evidenciados levantamentos de robôs civis e militares que contribuíram para a definição do protótipo. Depois

segue-se a Definição do protótipo (capítulo 3) que descreve o conceito do BUV desenvolvido, isto é, o seu *design* e requisitos que deverá cumprir, e, por último, os módulos constituintes do mesmo. Posteriormente segue-se o Projeto de hardware (capítulo 4), onde se é relatado o equipamento adquirido para capacitar o veículo, a distribuição de energia e as conexões do hardware. De seguida descreve-se o Projeto da arquitetura de software (capítulo 5), onde é feita uma pequena contextualização da arquitetura robótica e escolhido o software e arquitetura a implementar no veículo. Por conseguinte, segue-se o Desenvolvimento (capítulo 6), onde se explica o desenvolvimento do código da camada funcional do veículo. São ainda descritos os Testes e experiências (capítulo 7), por forma a testar e verificar os algoritmos desenvolvidos. Por fim, apresenta-se as conclusões do projeto e sugestões de desenvolvimento e implementação futura.

Capítulo 2

Estado de Arte

A tecnologia dos veículos não tripulados submarinos, conhecidos também por UUVs, em inglês, tal como a robótica em geral, encontra-se nos dias de hoje em estado de desenvolvimento com considerável avanço. Como exemplo disso são os vários veículos já desenvolvidos, provenientes das mais variadíssimas áreas, nomeadamente, na cientifica, comercial e militar. Estes sistemas beneficiam também do advento das novas tecnologias no campo dos materiais, técnicas de fabrico, sistemas computacionais, atuadores, sensores, redes de comunicação, sistemas de navegação, localização e de todo o desenvolvimento geral no contexto dos últimos anos. Fazendo uma análise do estado da arte no campo da robótica submarina, predominantemente direcionada para os BUVs, encontram-se diversos modelos construídos, normalmente, associados às entidades que lhe deram origem, com diferentes características estruturais, aplicações e tecnologias empregues. O presente capítulo aborda os assuntos anteriormente referidos, procurando abranger os vários temas relacionados com o desenvolvimento de robôs submarinos, nomeadamente, os BUVs. Ao longo desta abordagem serão mencionados e descritos os elementos fundamentais que constituem um UUV. Quando um veículo utiliza especificamente uma dada tecnologia em discussão, será dado esse BUV como exemplo concreto. Através da descrição e comparação dessas características predominantes, pretende-se tomar conhecimento do estado da arte, que respeita ao projeto e desenvolvimento de um BUV.

2.1 Considerações gerais

A Natureza, nos seus 3.8 mil milhões de anos de existência, conseguiu desenvolver-se, adaptar-se e evoluir por forma a sobreviver às diversas adversidades, como as catástrofes naturais. Dela fazem parte um conjunto diversificado de formas, estruturas, sistemas, padrões e até cores, que funcionam e coexistem em harmonia. Desta forma poderá utilizar-se todo este conhecimento das suas diversas

formas como ferramentas úteis para os robôs conseguirem ultrapassar os problemas que hoje em dia se deparam (Fernandes, 2013).

2.1.1 Inspiração biológica e biomimética

A palavra biomimética foi usada pela primeira vez, na literatura científica, em 1962. O uso da palavra foi crescendo, em particular no seio dos cientistas de materiais na década de 80 (Pawlyn, 2011). Mais tarde, a cientista Janine Benyus no livro *Biomimicry: Innovation Inspired by Nature*, de 1997, afirmou que a biomimética é: "(...) a nova ciência que estuda os modelos da natureza e imita-os visando resolver os problemas humanos" e afirmou que olhava para a natureza como um "Modelo, Medida e Mentor", sugerindo que o principal objetivo da biomimética fosse a sustentabilidade (Benyus, 1997). A biomimética traduz-se na forma mais brilhante e genial de se procurar soluções sustentáveis para os diversos desafios humanos, através da imitação e modelação das analogias da natureza, dos fenómenos e padrões. Os biólogos estão familiarizados com as diversas adaptações específicas nos animais, podendo estas ser do interesse dos engenheiros. O avanço tecnológico permitiu aos biólogos a identificação de novos subsistemas que constituem os seres vivos, enquanto que para os engenheiros a identificação de novos subsistemas é necessária para explorar e desenvolver veículos biomiméticos. Esta sinergia entre os biólogos e os engenheiros pode ser benéfica no avanço da tecnologia, tendo como farol a natureza para fornecer soluções para os problemas atuais. Para as tecnologias marítimas, a abordagem biomimética mantém, particularmente, a promessa de desempenho aprimorado e de maior eficiência na operação do domínio aquático (Fish & Kocak, 2011).

Nos últimos anos, através de pesquisas extensivas, tornou-se possível incorporar as características funcionais dos peixes nos projetos de veículos não tripulados e robôs (Chowdhury et al., 2014; Clapham & Hu, 2015; Yu et al., 2014; Yu et al., 2016). Estas implementações levaram a duas filosofias de design robótico, inspiração biológica e biomimética.

A inspiração biológica, traduz-se no uso das características funcionais dos organismos biológicos incorporados num robô, através da abordagem de projeto *top-down*[6], isto é, desenvolvem-se sistemas mecânicos simples e eficazes para contornar a complexidade dos subsistemas biológicos observados na natureza.

[6]Filosofia de *design* robótico composto pelas fases: *design* do problema, pesquisa e comparação biológica, identificação dos princípios apropriados, abstração e afastamento do modelo biológico, teste, análise e *feedback* e por último desenho da solução (Aziz & El sherif, 2016)

A biomimética, por outro lado, mimetiza a anatomia biológica com a agregação de unidades de morfologia semelhante às células biológicas e, assim, confere as funcionalidades biológicas à plataforma robótica. Desta forma desempenha um papel muito importante na área emergente da robótica - *ethorobotics*[7] (Krause, Winfield & Deneubourg, 2011), onde os robôs biomiméticos são usados para o estudo dos sistemas biológicos através da imitação, observação disfarçada e interação pretendida. Algumas experiências iniciais mostram que as diferentes implementações biomiméticas provocam reações diferenciadas entre os animais-robôs (Miklosi & Gerencser, 2012). No entanto, a biomimética é difícil de ser implementada devido à complexidade proibitiva das células biológicas. Desta forma, a inspiração biológica é considerada uma abordagem de projeto pragmática e é usada extensivamente para projetar robôs inspirados em peixes (Raj & Thakur, 2016).

2.1.2 Soft robotics

Nos últimos anos, através de diversos desenvolvimentos dos materiais utilizados para a construção robótica, quer pelo desenvolvimento dos próprios sensores e atuadores, surgiu uma área chamada de *soft robotics* que procura explorar o projeto, atuação e controlo dos corpos macios e flexíveis para alcançar uma maior flexibilidade e adaptabilidade, quando comparado com as plataformas rígidas das tradicionais estruturas robóticas (Katzschmann, 2018). Os *soft robots* são parcialmente ou totalmente construídos a partir de materiais altamente compatíveis, semelhantes aos que são encontrados nos organismos vivos. Estes robôs são inspirados nos movimentos dos organismos vivos e na forma como eles se adaptam ao ambiente. Os corpos dos seres vivos apresentam um movimento suave e facilmente adaptável, o que inspira o *design* dos *soft robots*. Pesquisas sobre os *soft robots* visam especialmente os aspetos de flexibilidade, adaptabilidade e agilidade, necessárias nas tarefas robóticas, proporcionando ao mesmo tempo interações mais seguras e eficazes com a natureza (Laschi, Mazzolai & Cianchetti, 2016; T. J. Roberts, 2016; Rus & Tolley, 2015). O projeto, planeamento do movimento, cinemática inversa e controlo, ainda são limitadas e não exploram a capacidade total dos módulos de locomoção flexíveis. As pesquisas nesta área concentram-se principalmente na estrutura, onde também são incluídos os atuadores e sensores, que juntos, constituem o corpo do sistema robótico.

[7] Área da robótica social, onde se procura que os robôs tenham funções sociais em termos de capacidade de incorporação, comportamento e resolução de problemas (cognitivos), para conseguirem interagir com outros seres nos seus *habitats* específicos (Miklósi et al., 2017)

Apesar do sistema de locomoção ser deveras importante, a capacidade do controlador, que representa o cérebro do robô, não fica atrás, e deve permitir o desenvolvimento de novas funcionalidades, por forma a que os *soft robots* possam atingir todo o seu potencial. Embora sejam necessários atuadores e sensores de alta resolução para a locomoção do robô ser suave e flexível (apenas alcançado com atuadores de movimentos precisos e dinâmicos), também se torna necessária a criação de modelos e controladores mais eficientes. Uma das principais motivações dos desenvolvedores de *soft robots* é a esperança de conseguirem melhorar os seus protótipos, no que diz respeito aos movimentos dinâmicos e às interações compatíveis com o meio ambiente (Katzschmann, 2018). Assim como todos os organismos biológicos, que possuem um cérebro e sistema nervoso, têm diversas habilidades motoras, as quais são treinadas e duram a vida toda, e memórias das suas mais diversas experiências, os protótipos robóticos biomiméticos também devem ter a capacidade de se adaptar e aprender com o meio que os rodeia.

Alguns exemplos de *soft robots* desenvolvidos até então são mostrados na figura 2.1.

(A) *Soft Rehabilitation Glove.* Fonte: retirado de Laschi, Mazzolai e Cianchetti (2016).

(B) *Soft robot X-RHex.* Fonte: retirado de Laschi, Mazzolai e Cianchetti (2016).

(C) *Soft robot OCTOPUS.* Fonte: retirado de Laschi, Mazzolai e Cianchetti (2016).

(D) *Soft robotic fish.* Fonte: retirado de Laschi, Mazzolai e Cianchetti (2016).

(E) *Soft robot PoseiDrone.* Fonte: retirado de Laschi, Mazzolai e Cianchetti (2016).

(F) *Soft robot Gripper.* Fonte: obtido do *website* da Festo - Automação[8].

FIGURA 2.1: Algumas das diversas aplicações dos *Soft Robots*.

2.1.3 *Unmanned Underwater Vehicles* (UUV)

Os veículos de subsuperfície são divididos em duas categorias que dependem da ocupação humana dos robôs: veículos tripulados (*Human Occupied Vehicles* (HOV)) que foram concebidos para a desempenharem missões com ocupação humana e os veículos não tripulados (UUV) que foram concebidos para executar missões sem ocupação humana. Por sua vez, os veículos não tripulados de subsuperfície, ou UUVs, são divididos em três categorias distintas tendo em conta o seu modo de operação: os *AUVs*, os *ROVs* e os *Autonomous Underwater Gliders (AUGs)*, como se poderá ver na figura 2.2.

[8] *Website* da Festo - Automação: https://www.festo.com/cms/pt_pt/72883.htm.

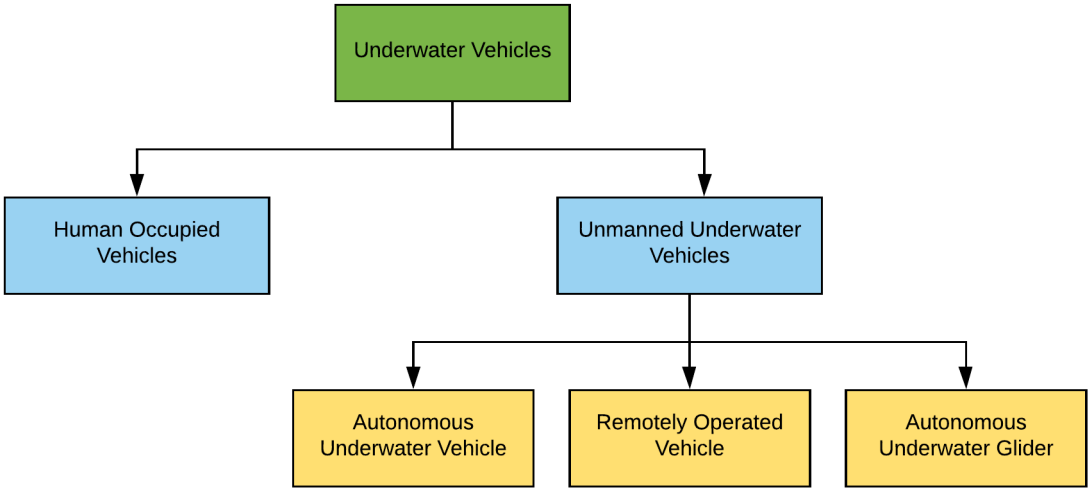


FIGURA 2.2: Diagrama dos veículos de subsuperfície (*Underwater Vehicles*).

Os AUVs são veículos autónomos, isto é, não necessitam de nenhum comando humano para executarem a missão. São controlados por um computador pré-programado e têm incorporados diversos sensores, instrumentos e a energia necessária para que se consigam deslocar sobre a água sob o comando de si próprios (Ocean Explorer, 2015). Numa grande parte dos AUVs, o movimento longitudinal é essencial para permitir a imersão (mergulho da superfície para o fundo) e a emersão (mergulho do fundo para a superfície). A grande maioria dos AUVs têm variadíssimos dispositivos, como sensores, câmaras de vídeo, dispositivos de recolha de amostras e acústicos para realizarem um grande leque de tarefas do tipo mapeamento/batimetria, inspeção, análises e Investigação e Desenvolvimento (I&D) em geral, como se pode observar no AUV HarborScan na figura 2.3. Este tipo de robôs provêm das mais variadíssimas áreas, desde a científica à comercial, passando pela área militar. Estes sistemas beneficiam também do advento das novas tecnologias no campo dos materiais, técnicas de fabrico, sistemas computacionais, atuadores, sensores, redes de comunicação, sistemas de navegação, localização e todo o desenvolvimento geral no contexto dos últimos anos (Gonçalves, 2016).

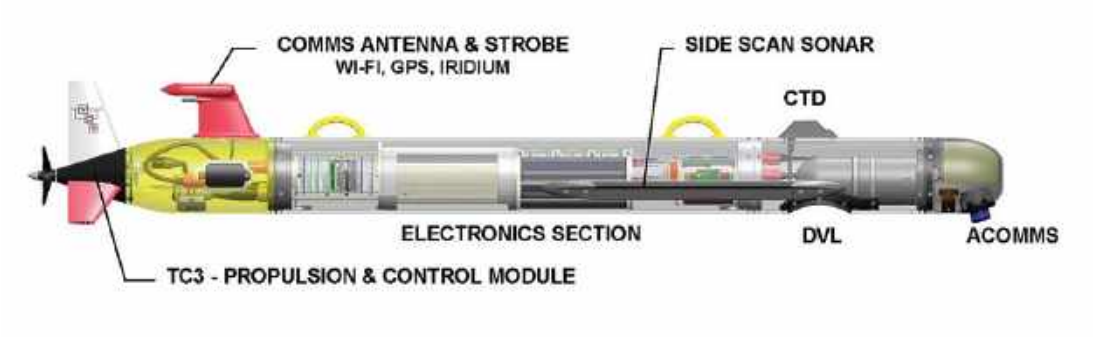


FIGURA 2.3: Constituição geral do AUV HarborScan.
Fonte: retirado de Vehicle Control Technologies (2015).

Na sua maioria, os AUVs são acompanhados por um software de comando e controlo, que permite a interface entre o operador e o veículo. O diagrama da figura 2.4 caracteriza a estrutura detalhada do sistema (em inglês *System Breakdown Structure* (SBS)) genérico de um AUV, permitindo a visualização de todos os sistemas que este tipo de veículos pode integrar (Azevedo, 2013).

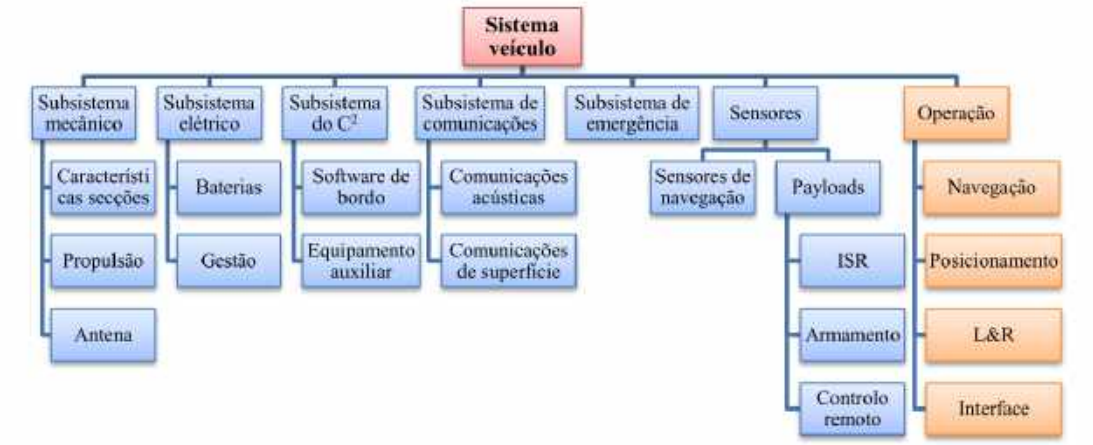


FIGURA 2.4: SBS de um AUV genérico.
Fonte: retirado de Azevedo (2013).

Redes sem fios e *Global Positioning System* (GPS) só funcionam debaixo de água em muito curtas distâncias e diretamente relacionadas com a potência de emissão e frequência, devido à grande atenuação destes sinais de rádio de alta frequência em meios subaquáticos. Assim, a localização no meio subaquático dos AUVs pode ser determinada utilizando sensores inerciais como a *Inertial Measurement Unit* (IMU), a *Attitude and Heading Reference System* (AHRS) e o *Inertial Navigation System* (INS). A IMU é um sistema independente que mede o movimento linear e angular, utilizando, geralmente um trio de giroscópios e um trio de acelerómetros (XSENS,

2020). A AHRS fornece orientação 3D integrando os dados giroscópios com os dados do acelerómetro e com os dados do magnómetro. Com a fusão destes sensores, o erro da integração dos giroscópios é compensada por vetores de referência, como a gravidade, e o campo magnético da terra. O INS é um sistema autónomo de navegação que permite obter e determinar a posição em relação a ponto de partida em latitude e longitude, por meio do processamento de dados fornecidos por giroscópios, acelerómetros e magnómetros. Pode ser ainda complementado com os dados de um sistema GPS. Podem também utilizar sistemas como *Long Baseline Navigation System* (LBL), *Ultra-Short Baseline Navigation System* (USBL) ou *Short Baseline Navigation System* (SBL), que são sistemas acústicos de posicionamento subaquático. Quando o ponto de referência para a comunicação é uma rede de *transponders* no fundo do mar, o sistema é chamado de LBL. Quando o ponto de referência está à superfície, como em um navio de apoio, o sistema é chamado de USBL ou de SBL, que utiliza o sistema de posicionamento do navio, como o GPS, para calcular posição do AUV.

Os AUVs podem ser classificados em três categorias, consoante a área onde operam e, consequentemente, as suas dimensões:

- *Shallow Water Survey AUVs*: veículos de pequenas dimensões para pesquisa em águas pouco profundas (até aos 100 metros de profundidade);
- *Middle Water AUVs*: veículos para operar em águas intermédias (500 a 1000 metros de profundidade);
- *Deep Water AUVs*: veículos para operar em águas profundas (mais de 1000 metros de profundidade).

Os ROVs são veículos que são controlados remotamente por um operador. O movimento do veículo pode ser autónomo, por via direção lógica, ou por controlo remoto do operador, que dependem da capacidade do veículo e do grau de entrada de comandos do operador. A potência do veículo pode ser interna (quando alimentada por baterias, que por sua vez alimentam o motor), exterior (quando a é fornecida por um cabo de ligação, como por exemplo um cabo de jato de água) ou híbrido (por exemplo, bateria de bordo é alimentada por energia transmitida remotamente através de um cabo). De forma simplista, um ROV poderá ser uma câmara montada num compartimento à prova de água, com propulsores para manobras, conectados a um cabo na superfície sobre a qual um sinal de vídeo e telemetria são transmitidos (Figura 2.5) (Christ & Wernli, 2013).

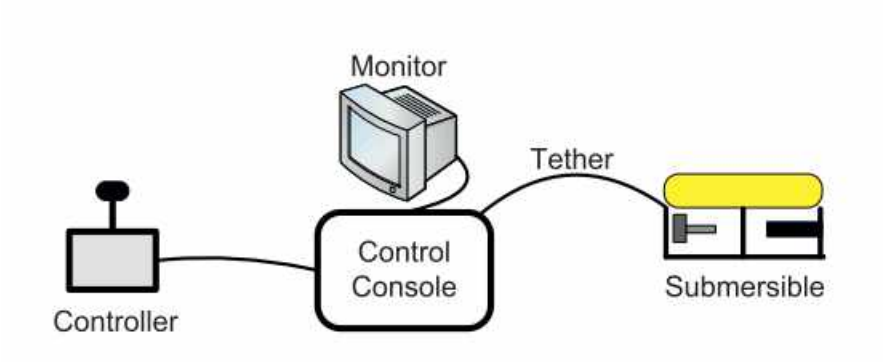


FIGURA 2.5: Constituição básica de um ROV.
Fonte: retirado de Christ e Wernli (2013).

Os AUGs, ou também planadores subaquáticos, são robôs que não possuem um módulo de propulsão (motores e hélice) para adquirirem seguimento. Estes veículos conseguem deslocar-se no meio aquático utilizando a própria variação de flutuabilidade, isto é, convertem a energia vertical das variações de flutuabilidade (efeito das correntes marítimas) em energia horizontal (direção do movimento). Apesar de menos complexos a nível de software, estes veículos podem incorporar vários sensores para efetuarem recolhas de dados meteorológicos e oceanográficos, bem como para o posicionamento (GPS). Estes veículos possuem a capacidade de permanecerem largos períodos em operação e percorrer grandes distâncias, não estando condicionados pelo carregamento das baterias (alguns *gliders* possuem painéis solares (Azevedo, 2013), como demonstrado na figura 2.6).

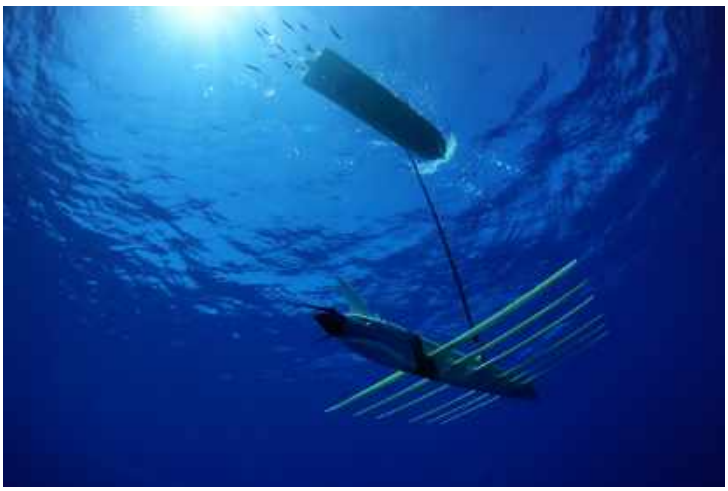


FIGURA 2.6: *Wave Glider*.
Fonte: obtido do *website* da *Composite Approach*[9].

[9] *Website* da *Composite Approach*: https://compositeapproach.com/sv3-marine-test/.

Existem ainda alguns modelos que são híbridos, isto é, podem ser AUV e ROV, como por exemplo o P-SURO II (figura 2.7).

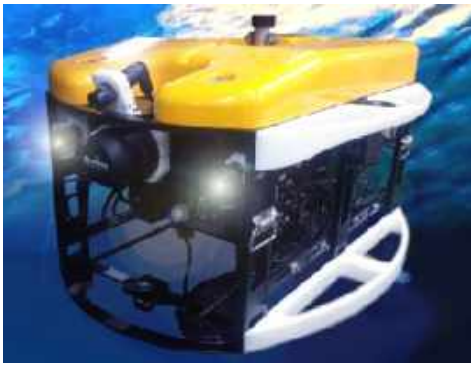
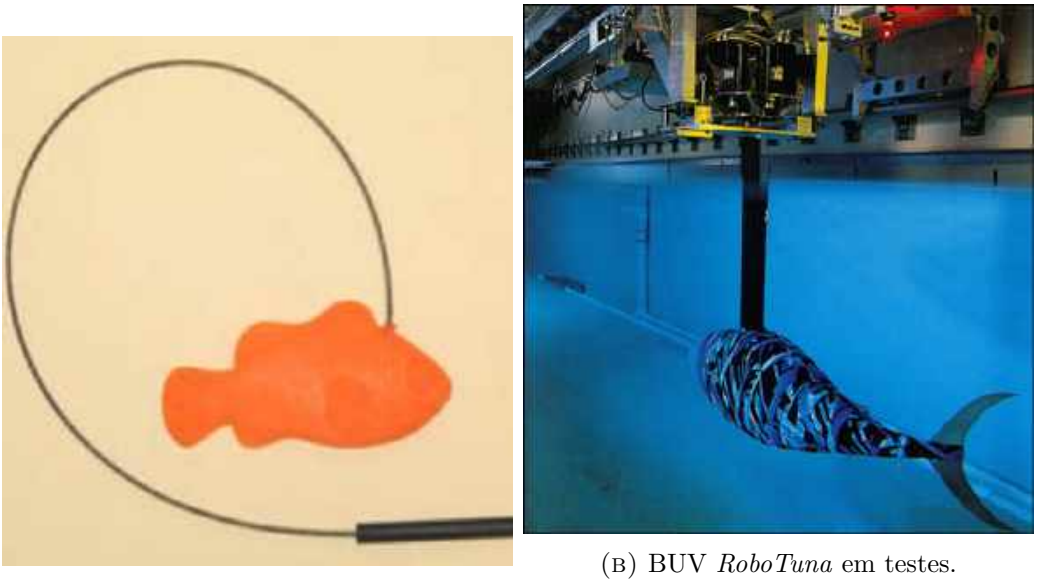


FIGURA 2.7: *P-SURO II.*
Fonte: retirado de Li et al. (2013).

2.1.4 *Biomimetic Underwater Vehicle* (BUV)

A "vida", como conhecemos hoje, evoluiu nos oceanos, onde animais complexos prosperaram por mais de 600 milhões de anos. Existem representantes marinhos de todos os principais filos animais. Os animais marinhos sobrevivem em ambientes tão diversos quanto os recifes de coral tropical, oceanos com gelo polar e as profundezas abissais sem luz. A diversidade de habitats disponíveis nos sistemas marinhos levou a uma vasta gama de projetos corporais e mecanismos fisiológicos e comportamentais. Essas adaptações que evoluíram em animais são usadas para superar os desafios bióticos e abióticos no oceano. Para lidar com os rigores do ambiente marinho, os animais desenvolveram sistemas sensoriais especializados (por exemplo, ecolocalização, eletrorrecepção), mecanismos para lidar com a pressão (por exemplo, controle de flutuabilidade), estratégias para economizar energia (por exemplo, design de corpo fusiforme, escolaridade, natação com rajada e deslizamento), armaduras (por exemplo, escamas ósseas, conchas de moluscos), mecanismos de estabilidade (por exemplo, barbatanas emparelhadas e medianas), manobrabilidade (por exemplo, corpos flexíveis, empuxo vetorial), velocidade (por exemplo, proporção de aspecto alta propulsores oscilatórios, propulsão a jato), discrição (por exemplo, camuflagem, baixa assinatura acústica) e uso de materiais compatíveis (por exemplo, colágeno, borrachas de proteínas, mucosas). Ao usar essas especializações para melhorar sua própria sobrevivência, os animais tentam funcionar de maneira a minimizar seu orçamento total de energia e maximizar o desempenho da especialização (Kottapalli et al., 2016).

Os BUVs, também chamados de *biorobotics* (Bandyopadhyay, 2005), são AUVs inspirados no mundo animal marinho. A locomoção, a inteligência, os sentidos - visão e audição, e a convivência com outros seres são inspirados num determinado animal. Essas características são conseguidas por utilização de determinados sensores e atuadores. Os primeiros robôs inspirados em peixes relatados foram o *TwiddleFish*, figura 2.8a desenvolvido pela *Duke University* (Long, J.H., 2011) e o *RoboTuna*, figura 2.8b, desenvolvido pelo *Massachusetts Institute of Technology* (MIT) (Triantafyllou & Triantafyllou, 1995) (Raj & Thakur, 2016).



(A) BUV *TwiddleFish*.
Fonte: retirado de Fish e Kocak (2011).

(B) BUV *RoboTuna* em testes.
Fonte: retirado de Triantafyllou e Triantafyllou (1995).

FIGURA 2.8: Primeiros BUVs desenvolvidos.

2.2 AUV e BUV desenvolvidos

Ao longo dos últimos anos, têm vindo a ser desenvolvidos diversos veículos autónomos por várias instituições académicas e empresariais. Para ser possível elaborar um protótipo capaz de possuir as características pretendidas deve-se considerar todo o estudo desenvolvido por estas entidades.

2.2.1 *Soft Robotic Fish* (SoFi)

O MIT desenvolveu um BUV chamado de *Soft Robotic Fish* (SoFi), capaz de se mover nos três eixos com a ajuda de uma cauda com atuador hidráulica, duas barbatanas dorsais e um sistema de controle da flutuabilidade (Katzschmann et al., 2018). Na figura 2.9 pode-se ver este veículo no seu habitat com controlo remoto,

utilizando as comunicações acústicas para transmissão de comandos do *joystick* para o BUV.

FIGURA 2.9: BUV SoFi.
Fonte: retirado de Katzschmann et al. (2018).

Na figura 2.10 pode-se analisar a constituição física do sistema desenvolvido. Em cima, da esquerda para a direita, tem-se o BUV e o módulo de controlo remoto, em baixo, da esquerda para a direita, os subcomponentes do sistema: cauda de *elastomer*[10] (visão de corte), a bomba externa de engrenagem, a *Buoyancy Control Unit* (BCU), as duas barbatanas dorsais, o controlador incluindo o recetor acústico e a câmara.

[10]O *elastomer* é um polímero com propriedades "elásticas", tem a capacidade de suportar grandes deformações antes da rutura. Os materiais *elastomers* tal como os pneumáticos são constituídos por uma base de borracha natural (NR) e por borracha sintética (Gadag & Shetty, 2014).

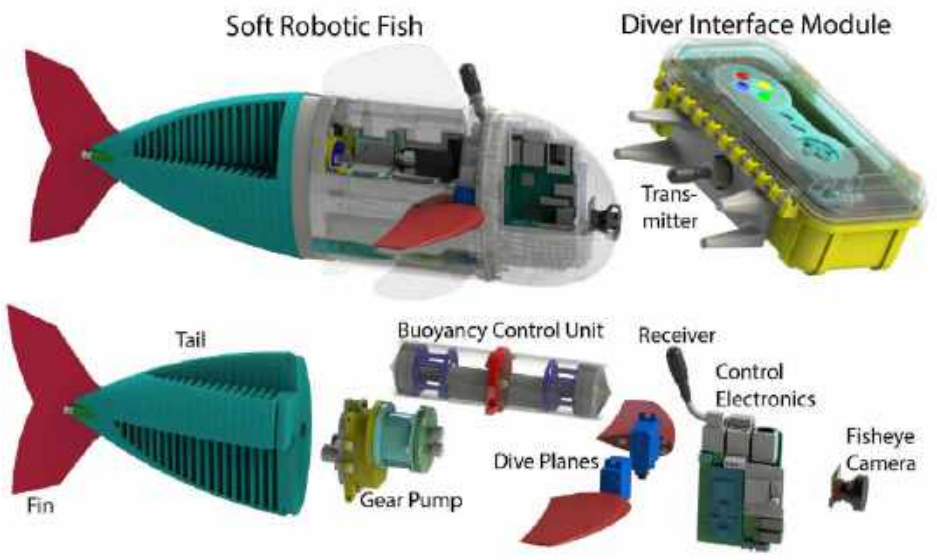


FIGURA 2.10: Visão geral do sistema do SoFi.
Fonte: retirado de Katzschmann et al. (2018).

2.2.2 *CyberFish*

A Universidade da Cracóvia tem vindo a desenvolver diversos protótipos de *Biomimetic Unmanned Underwater Vehicle* (BUUV) no âmbito do projeto de *CyberFish* (Morawski, Malec & Zajac, 2014) como se pode ver na figura 2.11. A primeira versão teve grande sucesso, uma vez que conseguiu cumprir os objetivos propostos inicialmente: movimento biomimético, fluatibilidade neutra e estanqueidade. Contém uma placa de circuito eletrónico baseada num microcontrolador como sistema de controle de baixo nível, baterias, tanque de flutuação, dois servomecanismos, módulos de comunicação, sensores de proximidade simples e câmara de vídeo sem fio. Dispõe de três segmentos que formam a cauda. Cada segmento tem acoplado um servomecanismo e engrenagens separadas, conferindo portanto, quatro graus de liberdade no total (figura 2.11a). É controlado remotamente.

Na segunda versão foi implementada uma IMU VectorNAV VN-100 para conferir ao veículo capacidade de navegação inercial e dois módulos de rádio (*Industrial, Scientific and Medical* (ISM) $868Mhz$), sendo um para transmitir as mensagens de controlo do robô e o outro para transmitir os dados da IMU para o computador (figura 2.11b).

A terceira versão em termos de hardware é igual à segunda. A diferença está no upgrade da capacidade da bateria e no controlo da flutuabilidade (figura 2.11c).

Na quarta versão a estrutura foi alterada adicionado mais um segmento à cauda conferindo, assim, seis graus de liberdade ao veículo e ao nível do hardware foram adicionados dois sensores de pressão (figura 2.11d).

A quinta versão voltou aos três segmentos e foi otimizado o algoritmo da terceira versão (figura 2.11e). Este BUUV pode ainda ser coberto por uma pele artificial que imita um peixe (figura 2.11f).

(A) Versão 1 do *CyberFish* (B) Versão 2 do *CyberFish* (C) Versão 3 do *CyberFish*

(D) Versão 4 do *CyberFish* (E) Versão 5 do *CyberFish* (F) Versão 6 do *CyberFish*

FIGURA 2.11: Versões do *CyberFish*.
Fonte: retirado de Morawski, Malec e Zajac (2014).

2.2.3 *Ichthus V5.5*

Este veículo, figura 2.12 foi desenvolvido por uma parceria entre a *Korea Institute of Industrial Technology* (*Cheonan*, Coreia do Sul), *Imperial College London* (Londres, Reino Unido) e *School of Computer Science and Electronic Engineering* (*Colchester*, Reino Unido) (Ryuh et al., 2015).

Este BUV é constituído por diversos sensores de navegação, tais como sensor infravermelhos (IR), *Micron Sonar* da *Tritech*, sensor GPS, sensor de pressão da água, INS e USBL. Deste modo, o sensor GPS é utilizado para localização quando

o robô está à superfície. Caso contrária, os dados do GPS não estarão acessíveis. Portanto, o veículo precisa de utilizar o sensor de pressão da água para deteção de profundidade e INS para estimar a sua posição. Para compensar o erro do INS, os dados de localização são transmitidos através de um modem acústico à superfície utilizando a USBL. Com estes recursos, pode mover-se autonomamente no ambiente subaquático e à superfície da água. Ainda possui sensores de deteção da qualidade da água que são usados para medir a temperatura, a condutividade elétrica e o *pH* da água.

FIGURA 2.12: BUV *Ichthus V5.5*.
Fonte: retirado de Ryuh et al. (2015).

2.2.4 BlueROV2

A *Blue Robotics* é uma empresa da Califórnia fundada em 2014 , cuja missão é melhorar a acessibilidade da robótica marítima, oferecendo produtos de alta qualidade a baixo custo, desenvolveu diversos ROVs. O último a ser desenvolvido (BlueRobotics, 2018), e que contém alguns sensores que serão utilizados no protótipo desta dissertação, é o *BlueROV2* (BlueRobotics, 2018), como se pode observar na figura 2.13.

FIGURA 2.13: Visão 3D do *BlueROV2*.
Fonte: retirado de BlueRobotics (2018).

O *BlueROV2* usa o software *ArduSub*[11] de código aberto e o piloto automático PixHawk para fornecer recursos autónomos necessários para a navegação do veículo (BlueRobotics, 2018). Pode ser operado com piloto automático ou remotamente, como se pode observar na figura 2.14. Pode ir até uma profundidade de 130m. Tem como sensores de navegação uma IMU *3-DOF*, um barómetro interno, um sensor de pressão *Blue Robotics Bar 30 Pressure/Depth* e como sensores de deteção de avarias contém um sensor de temperatura externo, um sensor de corrente e tensão e um sensor *Leak Detection*.

FIGURA 2.14: *BlueROV2* em missão no ambiente aquático remotamente controlado por fibra ótica.
Fonte: obtido do *website* da *Digital Trends - BlueRov2*[12].

2.2.5 *Fish as a Service* (FaaS)

O FaaS é um BUV constituído por três secções: a cabeça, o tronco e a cauda. Este foi desenvolvido pela empresa americana *Aquaai Corporation*. A estabilidade do veículo é alcançada através das barbatanas dorsais hidrodinâmicas e de uma lâmina estabilizadora, que impede o rolamento durante manobras de alta e baixa velocidade. A locomoção de alta velocidade utiliza uma unidade de propulsão a jato, enquanto que a de baixa velocidade utiliza uma combinação da oscilação da cauda e propulsão intermitente a jato. A inclinação é controlada pelo controlo da

[11] O projeto *ArduSub* é uma solução de código aberto com todos os recursos para veículos subaquáticos operados remotamente ROVs e AUVs. Pode ser acedido no site: https://www.ardusub.com/

[12] *Website* da *Digital Trends - BlueRov2*: https://www.digitaltrends.com/cool-tech/bluerov2-underwater-drone/.

flutuabilidade na bexiga artificial e do controlo do ângulo das barbatanas dorsais (Pieterkosky, Cavanaugh & Thompson, 2017).

A nível de hardware, figura 2.15, utiliza como computador central o *ODroid XU-4 (HardKernel, South Korea)*, como sensores de navegação utiliza o sensor inercial *AHRS+* da *HardKernel*, o GPS da *ASV Global* e um sonar para evitar obstáculos *Micron MK3* da *Tritech*. Para além destes inclui uma sonda *Conductivity, Temperature and Depth* (CTD) para registo da condutividade, temperatura e pressão da água, bem como o oxigénio dissolvido através do módulo *Glider Payload* da *SeaBird Electronics*. Todos os dados dos sensores hidrográficos podem ser anexados automaticamente às imagens gravadas pela câmara *High-Definition video* (HD) *Hero* da *GoPro* (Pieterkosky, Cavanaugh & Thompson, 2017).

FIGURA 2.15: Descrição dos sistemas constituintes do FaaS.
Fonte: retirado de Pieterkosky, Cavanaugh e Thompson (2017).

2.2.6 *RoboShoal*

O *RoboShoal*, figura 2.16, é um BUV desenvolvido num projeto de pesquisa europeu, denominado *Shoal* dirigido pelo *British Maritime Technology* (BMT), criado em 2012. Este projeto é constituído pela *School of Computer Science and Electronic Engineering of the University of Essex*, o *Tyndall National Institute*, o *Ship Stability Research Centre* (SSRC) da *University of Strathclyde*, a *Thales Safare SA* e o *Port Authority of Gijón* (PAG) (BMT, 2012).

FIGURA 2.16: *RoboShoal* em testes no mar.
Fonte: obtido do *website* *Boiled Beans* - SHOAL[13].

O *Shoal* tem como objetivo desenvolver um número de peixes robóticos que trabalharão mutuamente para monitorizar e identificar a poluição dos portos e de outras áreas aquáticas. Os métodos tradicionais de monitorização da poluição envolvem a obtenção de amostras de alguma forma (mergulhadores) e o envio dessas amostras ao laboratório para serem testadas. Todo este processo leva tempo e torna as informações de poluição, em tempo real, muito aquém do se pretende. O *Shoal* tem como objetivo tornar este processo de análise mais rápido, através de BUVs com sensores químicos (BMT, 2012).

Tem uma autonomia de $8h$ e o sistema de navegação depende de quatro *pingers* espalhados na área de missão que funcionam como satélites de GPS para os veículos. Um cardume de *RoboShoal* pode cobrir uma área marinha até 1 km^2 até uma profundidade máxima de 30m. Se um peixe detetar poluição numa área, pode chamar os outros para criarem um mapa detalhado de altas e baixas concentrações ao seu redor, auxiliando as autoridades portuárias a localizar a fonte exata do poluente (BMT, 2012).

2.2.7 *Bionic Manta Ray Robot*

O *Bionic Manta Ray Robot*, figuras 2.17a e 2.17b, foi desenvolvido por uma equipa da *School of Marine Science and Technology* da *Northwestern Polytechnical University* situada na República Popular da China (Zhang et al., 2018).

Foram desenvolvidos vários protótipos com tamanhos diferentes. O mais pequeno tem um comprimento de asas de $0.8m$ e tem uma velocidade máxima de

[13] *Website* da *Boiled Beans - SHOAL*: http://www.boiledbeans.net/2012/11/04/next-time-it-rains-and-everything-floods-here/.

1*nó*. Os protótipos maiores foram construídos com um comprimento de asas de 2 e 3*m* (Zhang et al., 2018).

(A) Em preparação para testes. (B) Durante os testes.

FIGURA 2.17: *Bionic Manta Ray Robot.*
Fonte: obtido do *website* *Northwestern Polytechnical University*[14].

Em termos de hardware possui um micro PC, um módulo de *wireless*, um sensor de pressão, e um *Central Pattern Generator* (CPG) utilizado para gerar um movimento biomimético rítmico e uma câmara. Na figura 2.18 poderá ser consultada o diagrama de blocos da estrutura do hardware (Zhang et al., 2018).

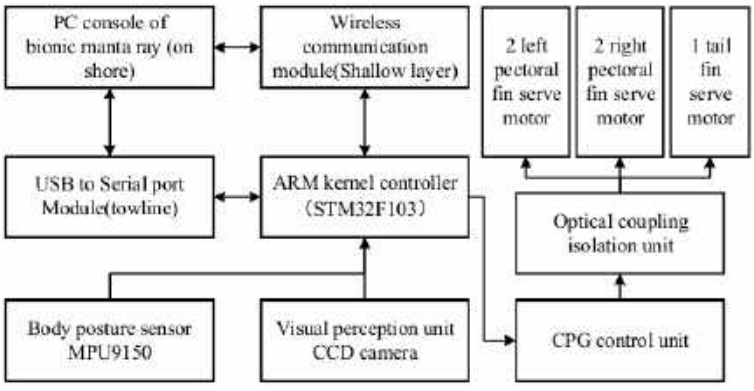


FIGURA 2.18: Diagrama de blocos de estrutura de hardware do *Bionic Manta Ray Robot.*
Fonte: retirado de Zhang et al. (2018).

2.2.8 *Robot Squid*

O BUV *Robot Squid*, também descrito como *squid-like aquatic-aerial vehicle*, é um *soft robot* que está a ser desenvolvido pela *Beihang University* na China.

[14] *Website* da *Northwestern Polytechnical University*: https://en.nwpu.edu.cn/info/1175/3096.htm.

Como o próprio nome indica, é inspirado na lula que se desloca no mundo marinho utilizando jatos de água. Em alguns animais da espécie são potentes o suficiente para que, não só possam saltar para fora da água, como também alcançar um vôo controlado por um breve período 2.19.

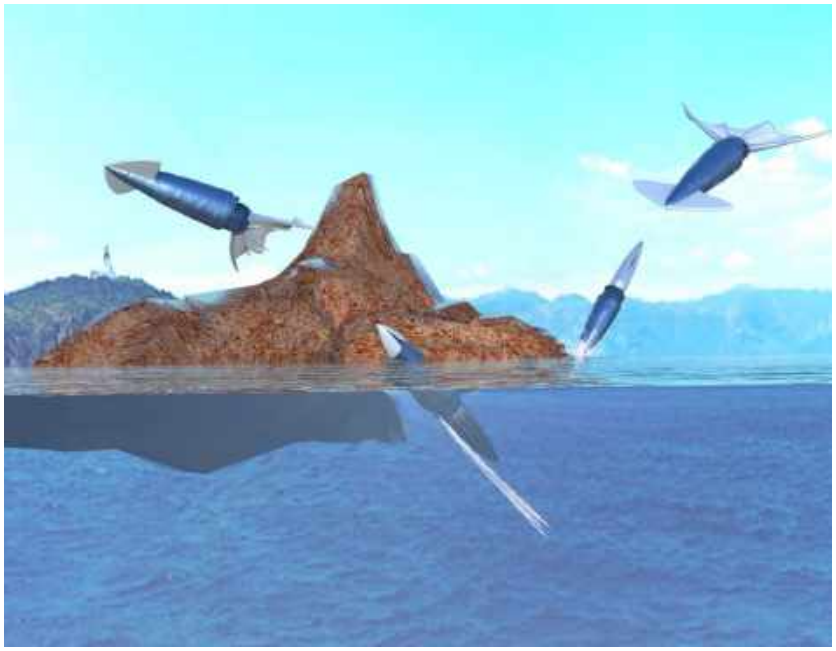


FIGURA 2.19: Multi-locomoção das lulas voadoras no lançamento, disparo de jato, planagem e mergulho.
Fonte: retirado de Hou et al. (2019).

Este *soft robot* utiliza ar comprimido armazenado um cilindro interior de gás, duas alhetas e dois braços - que são controlados por atuadores pneumáticos para imitar na integra uma lula real capaz de voar $10m$ (Hou et al., 2019). A constituição do veículo pode ser vista na figura 2.20.

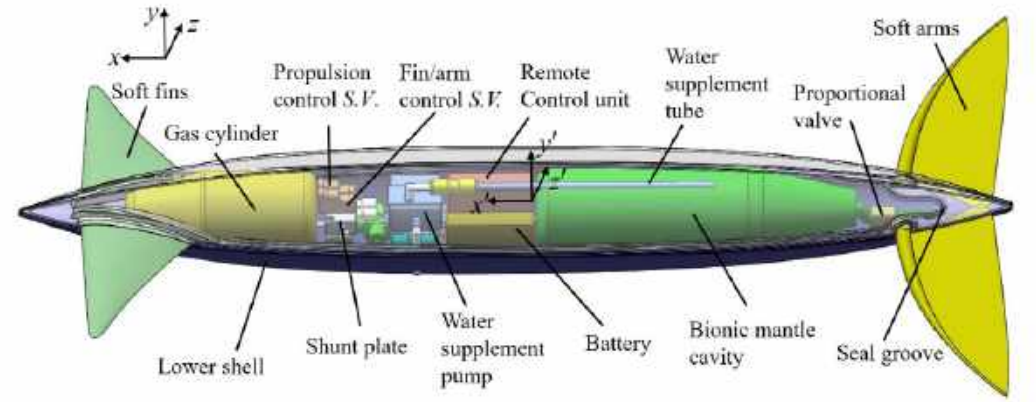


FIGURA 2.20: Constituição do BUV *Robot Squid*.
Fonte: retirado de Hou et al. (2019).

A figura 2.21 foi capturada a partir de um vídeo da experimentação do veículo que poderá ser consultado no *Youtube:* https://www.youtube.com/watch?v=kePnamaDSBw.

FIGURA 2.21: BUV *Robot Squid* em experimentação.

2.3 Alguns UUVs militares

2.3.1 *BIOSwimmer*

O *BIOSwimmer*, figura 2.22, foi desenvolvido pela *Boston Engineering Corporation, Advanced Systems Group Waltham* para o *United States Department of Homeland Security* (DHS) (Conry et al., 2013). Foi inspirado no atum e pode ser utilizado em modo automático, podendo ser-lhe atribuídas diversas missões, ou em modo semi-automático controlado remotamente.

FIGURA 2.22: Visão 3D do *BioSwimmer*.
Fonte: retirado de Rufo (2013).

As missões que o *BIOSwimmer* pode realizar são: inspeções ao casco de navios, missões de busca e salvamento e verificação de porões de carga que possam conter fluidos tóxicos. As especificações do veículo podem ser consultadas na figura 2.23.

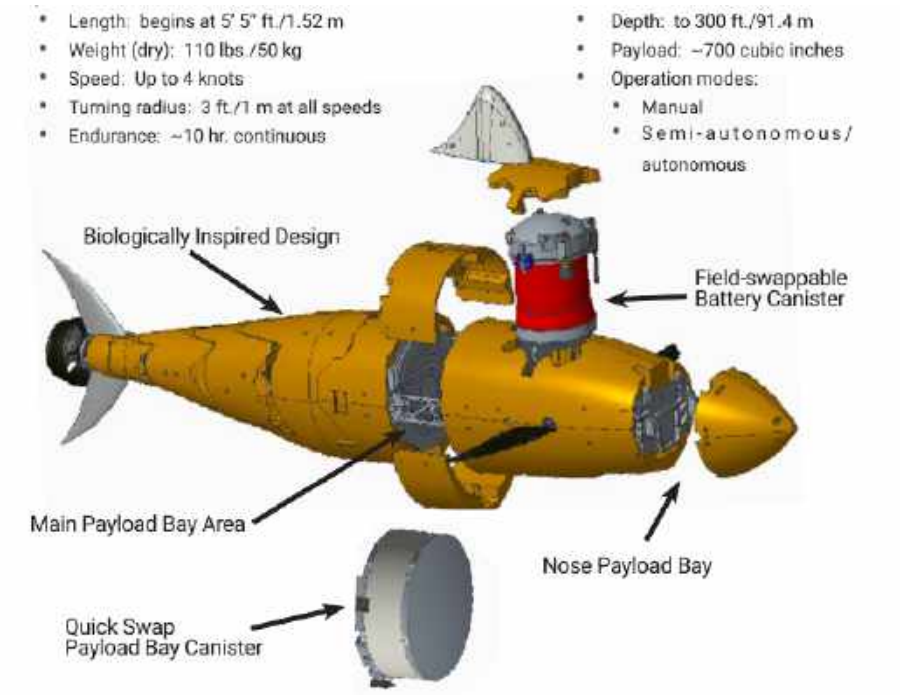


FIGURA 2.23: Especificações do *BIOSwimmer*.
Fonte: retirado de Boston Engineering (2019).

Esta empresa, em parceria com a *Chief of Naval Operations' Rapid Innovation Cell* (CRIC), no âmbito do projeto *Silent NEMO*, desenvolveu ainda uma versão chamada *GhostSwimmer*, figura 2.24, inspirada num tubarão para a *United States Navy*. O objetivo deste BUV é fornecer a segurança adicional em ambientes de baixa visibilidade nas missões de ISR e nas inspeções aos cascos dos navios da frota americana (Inside Unmanned Systems, 2014). Este robô mede cerca de $1.5m$ e pesa quase $45Kg$. A profundidade de operação é entre os $0.25m$ e os $91m$.

FIGURA 2.24: *GhostSwimmer* no mar.
Fonte: obtido do *website* da *New Atlas - GhostSwimmer*[15].

2.3.2 *Mini CyberSeal*

O *Mini CyberSeal* está a ser desenvolvido pela Escola Naval Polaca no âmbito do projeto SABUVIS II (Szymak et al., 2017). Este veículo é uma versão mais pequena do *CyberSeal*, desenvolvido pela Escola Naval Polaca, *Cracow University of Technology, Industrial Institute of Automatics and Measurement* e *Forkos Company* no âmbito do projeto SABUVIS I, como se pode observar na figura 2.25.

FIGURA 2.25: *CyberSeal* no mar.
Fonte: retirado de Szymak et al. (2017).

[15] *Website* do *New Atlas - GhostSwimmer*: https://newatlas.com/ghostswimmer-shark-robot/35286/.

O *Mini CyberSeal*, figura 2.26, encontra-se em fase de testes de implementação das barbatanas de locomoção e funciona apenas por controlo remoto (Szymak et al., 2017).

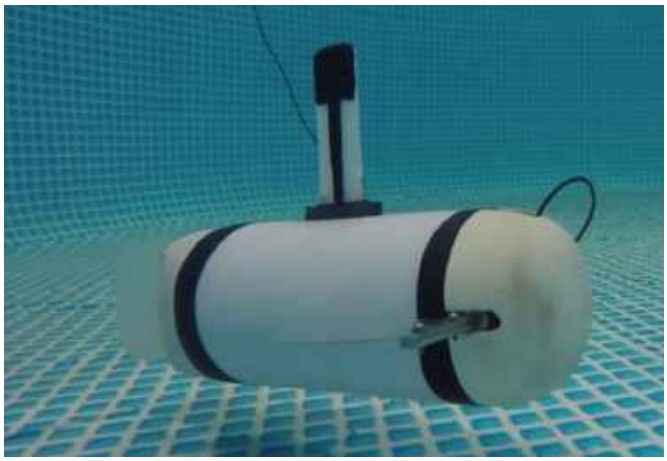


FIGURA 2.26: *Mini CyberSeal* em testes na piscina de experimentação da Escola Naval Polaca.
Fonte: retirado de Szymak et al. (2017).

Na figura 2.27 pode-se observar quais os componentes de hardware implementados no veículo.

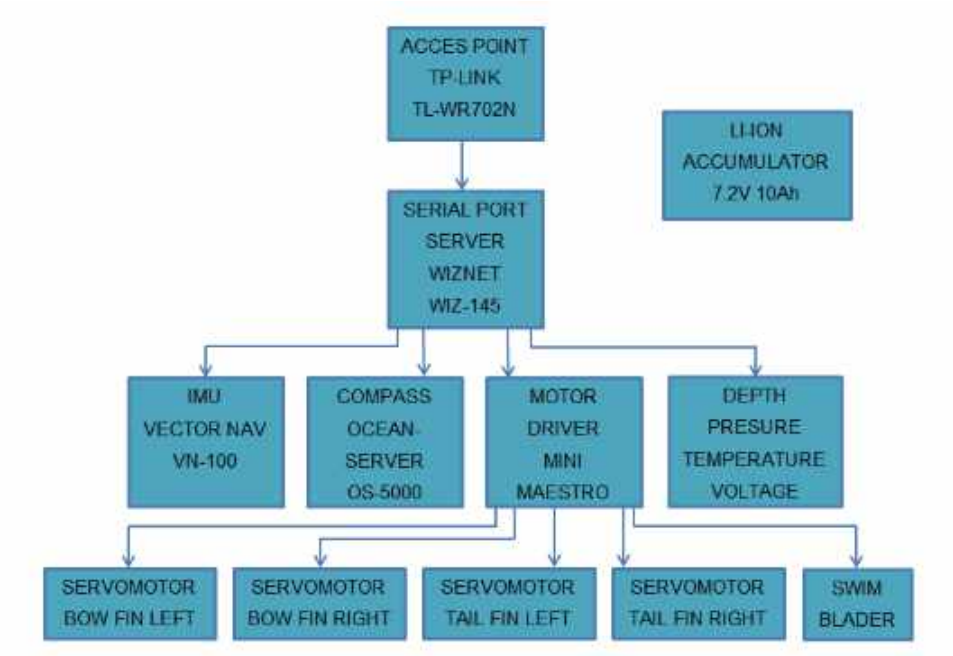


FIGURA 2.27: Diagrama de blocos dos sistemas do *Mini CyberSeal*.
Fonte: retirado de Szymak et al. (2017).

2.3.3 *Sea Hunting Autonomous Reconnaissance Drone* (SHARD)

O SHARD é um BUV inspirado numa lula, também descrito como um *Soft Robotic Cephalopod*, desenvolvido pela empresa militar fabricante de armas australiana *DefendTex*. Este veículo, conforme a figura 2.28, ainda está numa fase de desenvolvimento. Foi apresentado recentemente no *Defence & Security Equipment International* (DSEI) 2019 em Londres.

Move-se dentro de água com auxilio de oito tentáculos. Foi projetado para navegar autonomamente e atacar inimigos submarinos utilizando o seu próprio corpo para desencadear uma explosão. Pode também ser detonado remotamente pelos operadores. Futuramente será desenvolvida a capacidade para atuar em cardume. Poderá ser ainda utilizado em missões de ISR e MCM.

A empresa afirma que o SHARD desloca-se dentro de água com auxilio de oito tentáculos ligados a um motor. Pode ainda, recarregar-se, prendendo-se no fundo do oceano, por forma a permitir que as correntes oceânicas façam girar o gerador interno que carregará a bateria.

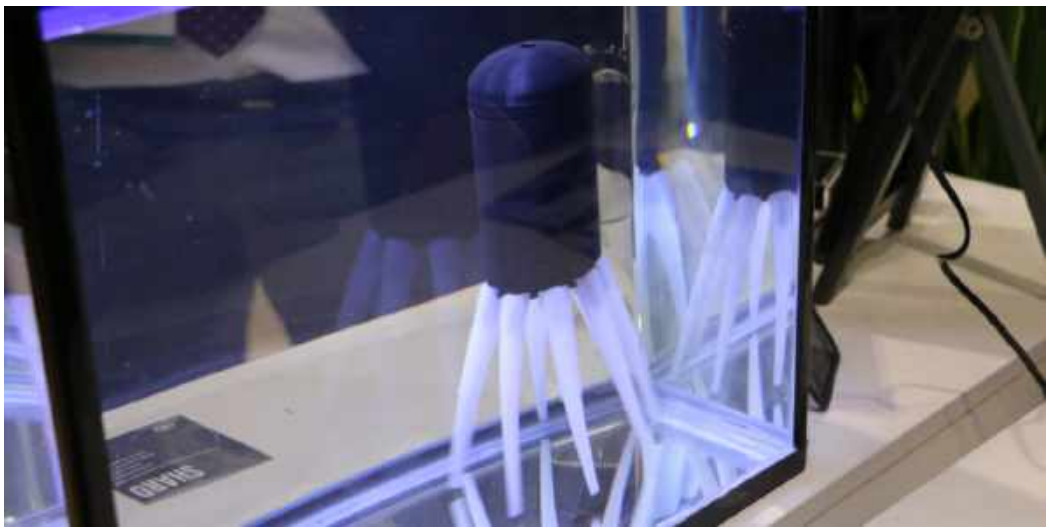


FIGURA 2.28: SHARD em exposição no DSEI 2019.
Fonte: obtido do *website* *Business Insider - Bill Bostock*[16].

2.3.4 *Robo-Shark*

O *Robo-Shark*, figura 2.29, é um BUV militar inspirado num tubarão desenvolvido pela *RoboSea*, empresa fundada em 2015 em *Beijing* na China.

[16] *Website* do *Business Insider - Bill Bostock*: https://www.businessinsider.com/video-squid-military-drone-is-sea-mine-2019-9.

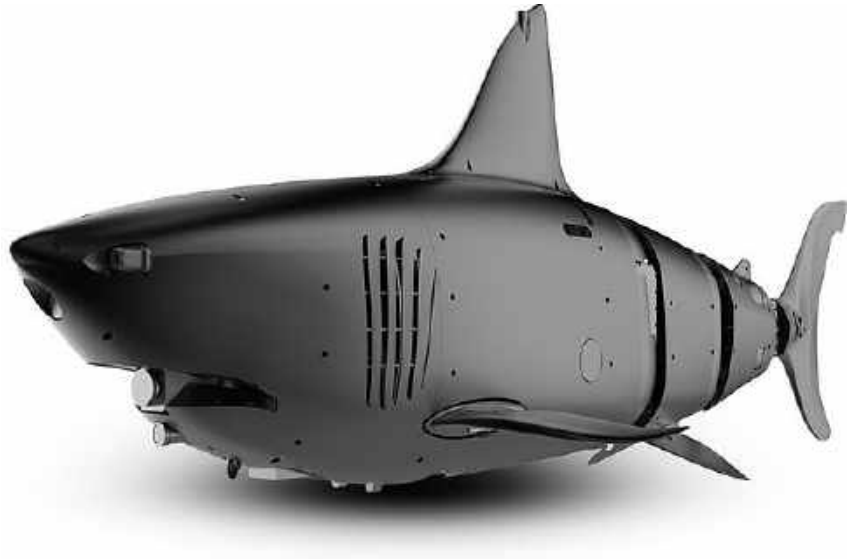


FIGURA 2.29: Visão do BUV *Robo-Shark* em 3D.
Fonte: obtido do *website* *RoboSea - Robo-Shark*[17].

Possui um sistema omnidirecional para evitar obstáculos a bordo. A cauda do robô é constituída por três segmentos, conferindo-lhe três graus de liberdade. Tem como referência uma velocidade máxima de $10 nós$.

O *Robo-Shark* pode ser usado para ISR, levantamentos do fundo do mar, pesquisa de objetos de interesse ou comunicações. Com as especificações do veículo, presentes na tabela 2.1, poderá ser utilizado para realizar intercetações de aproximação rápida, presumivelmente de possíveis mergulhadores inimigos, patrulha de alta mobilidade e rastrear alvos subaquáticos em alta velocidade.

TABELA 2.1: Especificações do BUV *Robo-Shark*.
Fonte: adaptado do *website* *RoboSea - Robo-Shark*.

Características	
Comprimento	$2.2m$
Largura	$85cm$
Altura	$93cm$
Peso	$75Kg$
Velocidade máxima	10 nós
Carga máxima	$20Kg$
Autonomia	8 horas

[17] *Website* da *RoboSea - Robo-Shark*: http://www.robosea.org/sanguanjie.html.

2.3.5 *RoboLobster (BUR-001)*

O *RoboLobster (BUR-001)*, figura 2.30, é um BUV militar desenvolvido pelo *Department of Biology and Marine Science Centre* da *Northeastern University* (Estados Unidos da América (EUA)) em 2006.

O veículo é composto por oito pernas, com três graus de liberdade cada uma, e por duas superfícies de controlo de estabilização hidrodinamicas das pernas posteriores e anteriores. O veículo é alimentado por uma bateria recarregável de *NiMH* ou *Li-Ion* (Ayers & Witting, 2007). A Marinha dos Estados Unidos adquiriu alguns protótipos deste veículo.

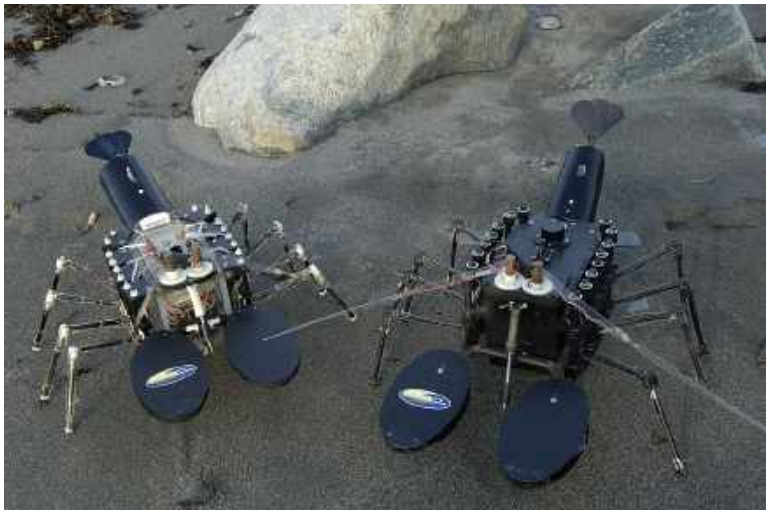


FIGURA 2.30: BUV *RoboLobster*.
Fonte: obtido do *website* *StrategyPage - Military Photo: RoboLobster*[18].

O controlo do veículo é baseado num circuito neuronal que implementa um conjunto de comportamentos obtidos por engenharia inversa a partir sequências de ações das lagostas que se adaptam ao ambiente na procura por objetos (Ayers & Witting, 2007; Young Jun Lee et al., 2005).

O veículo foi projetado para operar em águas rasas que apresentam corrente e obstáculos marinhos.

[18] *Website* da *StrategyPage - Military Photo: RoboLobster*: https://www.strategypage.com/military_photos/military_photos_2006432323235823.aspx.

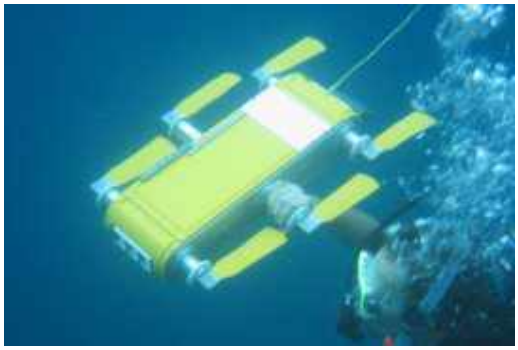
2.3.6 *AQUA2*

O *AQUA2* é um BUV militar desenvolvido pela empresa *Independent Robotics Inc.*, em parceria com a *McGill University* e a *York University* num projeto financiado pela agência americana *Defense Advanced Research Projects Agency* (DARPA), em 2011 (ROBOTS, 2019).

Este veículo proveio da versão original robô *Rhex* (figura 2.31a) desenvolvido em entre 1999 e 2001 (Saranli, Buehler & Koditschek, 2001), onde foram também desenvolvidos os robôs *Rugged RHex* (figura 2.31b), *AQUA* (figura 2.31c), *EduBot* (figura 2.31d) e *Desert RHex* (figura 2.31e).

(A) BUV original *Rhex*.
Fonte: retirado de Prahacs et al. (2011).

(B) *Rugged RHex*.
Fonte: retirado de Prahacs et al. (2011).



(C) *AQUA*.
Fonte: retirado de Prahacs et al. (2011).

(D) *EduBot*.
Fonte: retirado de Galloway, Clark e Koditschek (2013).

(E) *Desert RHex*.
Fonte: retirado de S. Roberts e Koditschek (2016).

FIGURA 2.31: Versões do *Rhex*.

O *AQUA2*, figura 2.32, é composto por seis barbatanas independentes que lhe conferem uma capacidade de manobra elevada. O veículo tem $64cm$ de comprimento, $44cm$ de largura, $13cm$ de altura, pesa $16.5Kg$, tem uma autonomia de $5h$ e $30m$ e pode ir até uma profundidade máxima de $30m$ (Adept Technology & Independent Robotics, 2011). Pode ser controlado por três modos distintos: filo guiado, por um mergulhador (controlo remoto) ou autónomo.

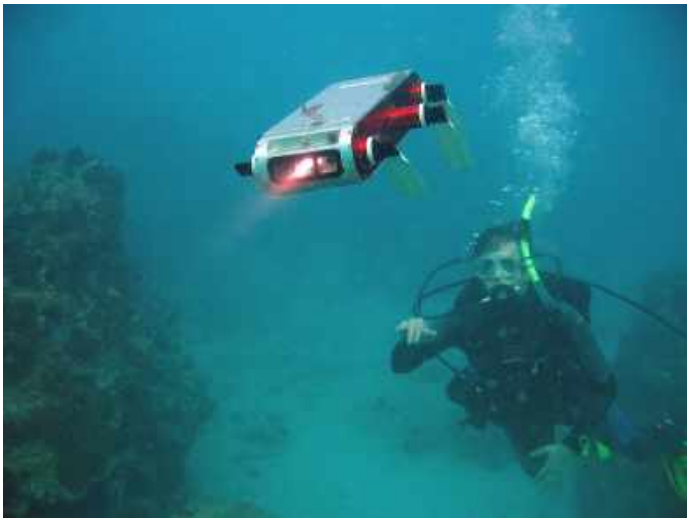


FIGURA 2.32: BUV *AQUA2*.
Fonte: retirado de (Adept Technology & Independent Robotics, 2011).

Este veículo tanto pode ser utilizado em terra, mesmo em ambientes secos, lamacentos, com areia ou neve, como no mundo aquático de água doce ou salgada. Em terra tem uma velocidade máxima de $0.7\mathrm{m\,s^{-1}}$ e no mar de $1.0\mathrm{m\,s^{-1}}$. As baterias são de *Li-Ion* com uma tensão de $28.8V$ e com capacidade de 7.2A h. Utiliza o *middle software Robot Operating System* (ROS) como arquitetura de controlo num *PC104+* (Adept Technology & Independent Robotics, 2011).

A empresa disponibiliza três configurações de hardware diferentes presentes na figura 2.33.

Mobility Base	Basic Vision	Complete Vision
Features	All Mobility Base features, ***plus***	All Mobility Base and Basic Vision features, ***plus***

• Robot shell and ballast system
• Actuator system (motors and driver)
• On-board power supply
• Two batteries and two slow chargers
• Control stack (only)
• Half-circle land legs for walking
• Flippers for swimming
• Robot Control Software Package
• Computer breakout connector
• 'Wired' Ethernet link
  • *Robot GigE to fiber converter*
  • *Fiber Cable (10 m)*
  • *Operator GigE to fiber converter*
	• Wireless Ethernet link
	  • *Robot 802.11 transceiver*
	  • *External antenna*
	• Vision Stack
	  • *Intel® ATOM D525 1.4GHz Processor*
	• 2 USB video cameras
	• Downward facing mirror
	• Image collection/display software
	• IMU (Roll-Pitch-Yaw, XYZ acceleration, +magnetometer)
	• Depth sensor
	  • *Configurable for fresh or seawater*
		• 1 additional USB camera (3 total)
		• Preconfigured Operator Control Unit: ruggedized waterproof laptop, magnesium alloy case rated for salt/fog operation
		• 50 meter armored cable
		• 200 meter armored cable
		• Shipping case
		• Accessories shipping case
		• Maintenance Toolkit
		• Additional Ballasting
		• AC/DC adapter for bench top operation
		• Customized clothing with IRI/Aqua2 logo

Functions	Additional Functions	Additional Functions
• Teleoperated on land or water (line of sight only) • Internal state data collection • Communicate underwater (fiber optic data link) or on land (Ethernet)	• Communicate via wireless data link (Ethernet) on land • Observe and collect live video data • Measure inertial parameters and depth	• LCD display from robot to nearby operator for use underwater • Completely accessorized for indoor/outdoor/remote operation

FIGURA 2.33: Configurações de hardware do BUV *AQUA2*.
Fonte: retirado de (Adept Technology & Independent Robotics, 2011).

2.3.7 *ACM-R5H*

O *ACM-R5H* ou *amphibious snake-like robot*, figura 2.34, é um BUV inspirado numa cobra e foi desenvolvido pela empresa *HiBot*, *spin-off* do *Tokyo Institute of Techonology* do Japão em 2010.

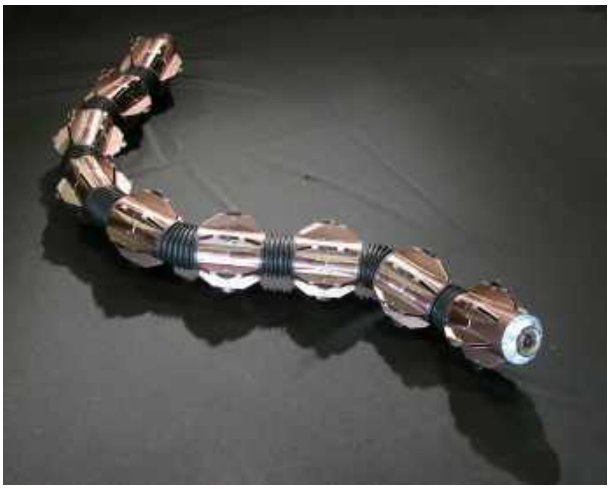


FIGURA 2.34: BUV *ACM-R5H*.
Fonte: obtido de HiBot (2017).

O veículo foi projetado para se poder deslocar, tanto em terra, como no mar (figura 2.35). Pode executar missões de inspeção em espaços confinados e/ou inundados, vigilância de portos e áreas costeiras, inspeção a cascos de navios e missões

de busca e salvamento em ambientes perigosos (HiBot, 2017). É constituído por módulos onde cada um é composto por um pack de baterias, mede 20cm de comprimento e 8cm de diâmetro e pesa 800g. Cada módulo possui dois graus de liberdade de movimento. No total, o veículo é composto por nove módulos pesando 7.95Kg e tem um comprimento máximo de 1.75m. Os módulos podem ser personalizados com vários sensores e mecanismos. O módulo da cabeça é equipado com uma câmara de transmissão por *wireless*. O veículo consegue deslocar-se a uma velocidade máxima de 1.44km/h.

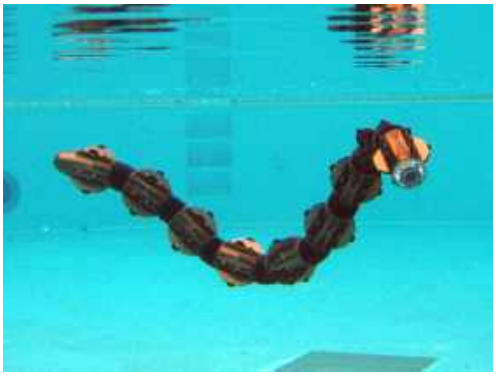


Figura 2.35: BUV *ACM-R5H* na água.
Fonte: obtido de HiBot (2017).

2.4 UUVs da Marinha Portuguesa

Na Marinha Portuguesa existem dois tipos de AUVs: os *SEACon* e os *Gavia 18 e 19*. Estes veículos são operados pelo Destacamento de Mergulhadores Sapadores nº3 (DMS3) - Destacamento de Guerra de Minas (DGM).

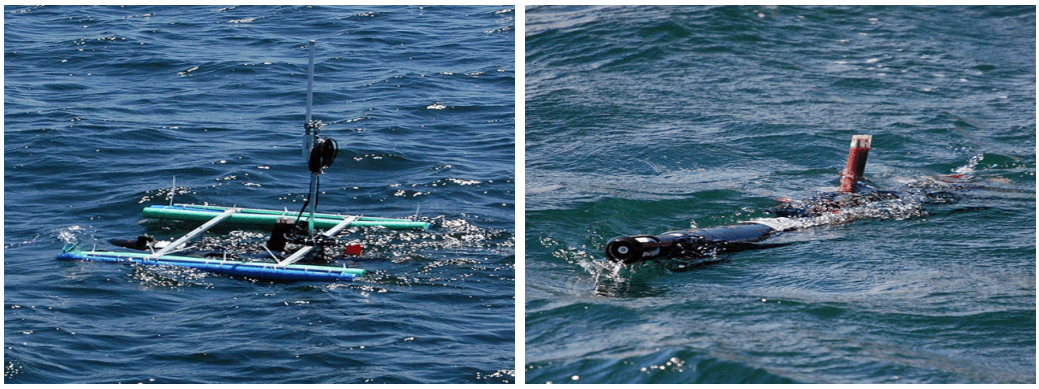
2.4.1 *SEACon* AUVs

Os *SEACon*, figuras 2.36a e 2.36b, foram desenvolvido pelo LSTS da FEUP em colaboração com a Marinha, no âmbito do projeto SeaCon financiado pelo MDN. Este projeto teve como objetivo o desenvolvimento de três veículos autónomos submarinos heterogéneos de baixo custo apoiados por um sistema de comando e controlo, para permitir o treino e formação dos elementos do DGM (Afonso, de Sousa & Martins, 2012).

(A) Passagem por cima de uma mina. (B) Depois da passagem por cima de uma mina.

FIGURA 2.36: *SEACon* em missão.
Fonte: disponibilizadas pelo DMS3.

Os veículos podem ser configurados para dois diferentes modos mediante as missões que irão realizar: AUV para missões de subsuperfície e *Autonomous Surface Vehicle* (ASV) para missões à superfície, como se pode observar na figura 2.37. No âmbito deste projeto foram entregues três exemplares à Marinha Portuguesa.



(A) *SEACon* no modo ASV. (B) *SEACon* no modo AUV.

FIGURA 2.37: Configurações do *SEACon* no *Rapid Environment Picture* (REP)[19]12.
Fonte: obtido de Pinto et al. (2012).

Os três veículos têm diferentes capacidades de carga (*payloads*). O *SEACon 1* é equipado com uma camara fotográfica e um *Side Scan Sonar* (SSS) da *Imaginex*. O *SEACon 2* é equipado com o sonar *SSS Imaginex* e com o sonar multifeixe *837B Delta T - Multibeam* da *Imaginex*. Já o *SEACon 1* é equipado com o sonar *SSS Klein* e com um *Doppler Velocity Log* (DVL) *NavQuest 600 Micro* da *LinkQuest*

[19]Exercício anual de robótica internacional co-organizado pela LSTS e pela Marinha Portuguesa desde 2010 (Laboratório de Sistemas e Tecnologia Subaquática & Marinha Portuguesa, 2020).

Inc. que é um aparelho com dois modos de operação: *bottom-lock*, capaz de medir a velocidade inercial do veiculo e *water-lock*, capaz de medir a velocidade do veiculo em relação à corrente da água. Pode ainda ser utilizado para medir perfis sonoros e levantamentos batimétricos (Afonso, de Sousa & Martins, 2012).

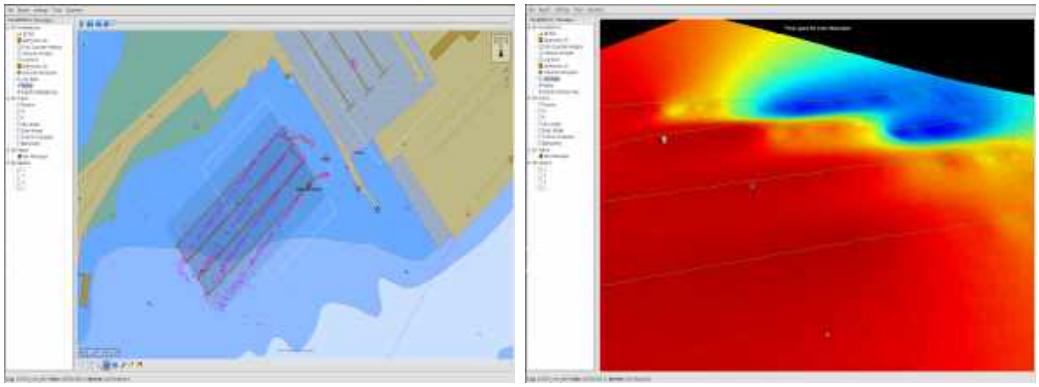
Os sensores comuns destes veículos são a unidade inercial que é uma IMU *HG1700* da *Honeywell*, um GPS *EVK-5* da *ublox*, um bússola *3DM-GX3-25 OEM* da *MicroStrain*, um sensor de pressão *PAA 33X* da *Keller AG* e um sensor CTD *SW1000* da *Global Monitoring Instrumentation Danmark*. Como micro PC são equipados com o *AIM104-COM8* (módulo PC/104) da *Arcom Control Systems* e como modem acústico são equipados com o *Micro Modem* do *Woods Hole Oceanographic Institute* (WHOI). O módulo de bateria é constituído por um grupo de sete baterias de *Li-Ion MP 176065 Integration* da *Saft - Specialty Battery Group*.

As especificações técnicas dos veículos podem ser ser consultadas na tabela 2.2.

Tabela 2.2: Especificações do AUV *SEACon*.
Fonte: adaptado de Afonso, de Sousa e Martins (2012).

Especificações técnicas	
Profundidade máxima	$50m$
Comunicações	*Wireless Fidelity* (WiFi), GSM/HSDPA e *Micro-modem* acústico
Navegação	IMU, LBL e GPS
Sensores	CTD, *Side Scan Sonar*
Outras especificações	
Comprimento	$1.10m$
Diâmetro	$16cm$
Peso	$18Kg$
Velocidade máxima	4 nós
Velocidade de operação	2 nós
Raio de giração	$7 \sim 10m$
Autonomia	8 horas

Para o sistema de comando e controlo foi desenvolvido um software chamado de *Neptus C2*, figura 2.38, composto por módulos para apoio a todas as fases de missão: planeamento, treino, supervisão e análise da missão (figura 2.38a), que pode ser corrido em *Personal Computer*s (PCs) e telemóveis *Android*. Neste software também existe a possibilidade de se extrair os dados obtidos dos sonares incorporados nos veículos, de acordo com a figura 2.38b.



(A) Análise de uma missão do *SEACon*. (B) Gráfico de profundidade de uma missão.

FIGURA 2.38: Software *Neptus C2*.
Fonte: retirado de Afonso, de Sousa e Martins (2012).

Este sistema utiliza um *gateway MANTA*, figura 2.39, desenvolvido pela LSTS com uma autonomia de $8h$. Contém uma antena WiFi, uma antena *Global System for Mobile Communications* (GSM)/*High-Speed Downlink Packet Access* (HSDPA)[20] e o modem acústico *Micro Modem* para estabelecer comunicação com os veículos.

FIGURA 2.39: *Gateway MANTA* em operação no REP 17.
Fonte: obtido do *website* do REP 17[21].

Este veículo encontra-se em fase de aperfeiçoamento (parceria de I&D *SeaConII*). Esta solução permitirá dotar a Marinha de capacidades únicas a nível

[20] O HSDPA é um protocolo de comunicações telefónicas móveis, também chamado de $3.5G$. O é um serviço de transmissão de pacotes de dados que opera dentro do *Wide-Band Code-Division Multiple Access* (W-CDMA), tecnologia líder do $3G$, que permite a transmissão de dados até $14,4Mbit/s$ numa banda de $5MHz$ (Kolding, Frederiksen & Mogensen, 2002). Neste sentido, possibilita a transmissão de multimédia em banda larga em telemóveis móveis.

[21] *Website* do *REP 17*: https://rep17.lsts.pt/systems.

europeu, sendo que está condicionado pela capacidade de desenvolvimento nacional nesta área. Estas capacidades complementam as capacidades únicas da nova arma submarina da Marinha e contribuirão para melhor as adaptar a novos ambientes de operação.

2.4.2 *Gavia* AUVs

Os *Gavia*,figura 2.40, foram desenvolvidos pela empresa *Teledyne Marine*, que é uma das vinte e três empresas de tecnologia submarina de ponta pertencentes ao grupo *Teledyne Technologies Inc.* Em 2008, a Marinha Portuguesa adquiriu um *Gavia* 18 e um *Gavia* 19.

FIGURA 2.40: *Gavia* em preparativos para testes.
Fonte: obtido do *website* da *Marine Teledyne - Gavia*[22].

A estrutura destes AUVs é composta por seis módulos (figura 2.41):

- Propulsão: onde estão integrados os atuadores de propulsão;
- Controlo: onde estão integrados o micro PC *Cool LiteRunner-LX800* (módulo PC/104) da *LiPPERT Embedded Computers* e um *EdgeTech 2205 Side Scan Sonar* de $900/1800KHz$ da *Marine Sonic Technology*. Está acopolado, a este módulo, uma torre de antenas de GPS (*Wide Area Augmentation System*

[22] *Website* da *Marine Teledyne - Gavia*: http://www.teledynemarine.com/gavia-auv/?BrandID=9.

(WAAS)[23]/*European Geostationary Navigation Overlay Service* (EGNOS)[24]), *Iridium* da *NAL Research Corp.*, *Wireless*, um transdutor acústico *ATM-900* da *Teledyne Benthos*, um sensor de pressão *Keller*, um sensor de CTD da *SeaBird*, um *Sound Velocity Sensor* (SVS) *miniSVS* da *Valeport Limited* e um farol visual;

- INS: composto por um INS *T-24INS* da *Kearfott*;
- DVL: composto por um sensor DVL *RDI 1200kHz* da *Teledyne Marine*;
- Bateria: que é composto por várias baterias de *Li-Ion* independentes. Tem uma capacidade total de $1.2kWh$;
- Nariz: onde estão colocados um sonar de desvio de obstáculos 852 *Ultra Miniature Echo Sounder* da *Imagenex* com alcance máximo de $50m$ e uma câmara de baixa luminosidade.

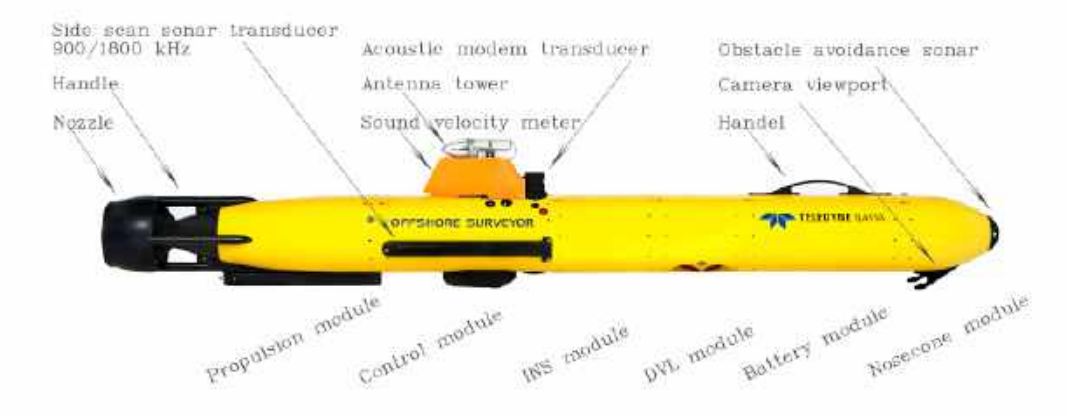


FIGURA 2.41: Módulos da constituição do *Gavia*.
Fonte: retirado de Teledyne Gavia (2017).

Os módulos do *Gavia* podem ser movidos livremente entre veículos (Teledyne Gavia, 2017). As especificações do veículo podem ser consultadas na tabela 2.3.

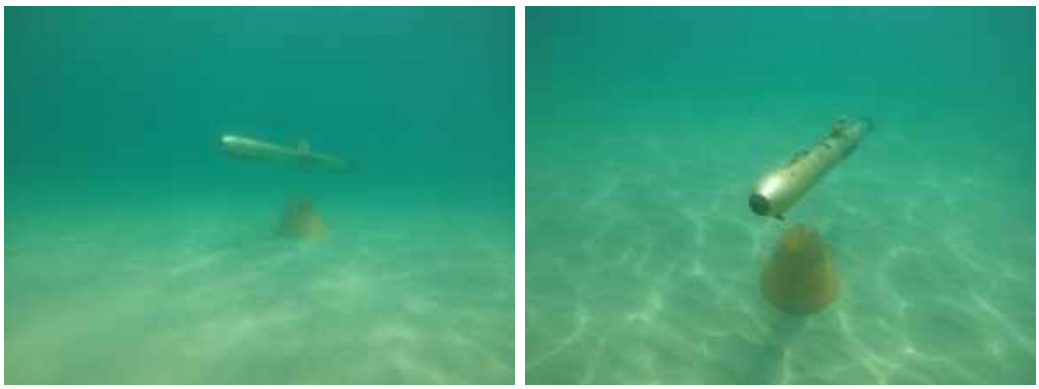
[23]O WAAS foi o primeiro sistema de correções por satélite (*Satellite-Based Augmentation System* (SBAS)) do mundo a fornecer navegação horizontal e vertical para abordagens de precisão de aeronaves. Encontra-se operacional desde 2003 (Flight Global, 2006).

[24]O EGNOS é um sistema SBAS desenvolvido pela *European Space Agency* (ESA) que utiliza três satélites geoestacionários (*Inmarsat's Indian Ocean Region*, *Atlantic Ocean Region-East satellites* e *ESA's Artemis spacecraft*) para transmitir os dados de correção de erros para as suas 34 estações de monitorização da integridade e alcance, localizadas entre Lisboa e Turquia (Flight Global, 2006).

TABELA 2.3: Especificações do AUV *Gavia*.
Fonte: adaptado de Teledyne Gavia (2017).

Especificações técnicas	
Profundidade máxima	$200m$
Comunicações	WiFi, modem acústico, satélite (*Iridium*)
Navegação	INS, DVL e GPS
Sensores	CTD, *Side Scan Sonar*, SVS
Outras especificações	
Comprimento	1.8 - 4.5m (depende da configuração)
Diâmetro	$20cm$
Peso	$48 \sim 100Kg$ (depende da configuração)
Velocidade máxima	5.5 nós
Velocidade de operação	3 nós
Raio de giração	$15 \sim 20m$
Autonomia	6 horas (1 bateria a 3 nós) 10 horas (2 bateria a 3 nós)

Os modos de operação em missão de exploração do fundo do mar podem ser *Bottom Track*, conforme figuras 2.42a e 2.42a, onde o veículo mantém uma profundidade constante em relação ao fundo do mar definida pelo operador ou *Constant Depth*, sendo que o veículo mantém uma profundidade constante em relação à superfície do mar.



(A) Momento de elevação. (B) Movimento de deteção de objeto.

FIGURA 2.42: *Gavia* em missão no modo *Bottom Track*.
Fonte: disponibilizadas pelo DMS3.

Capítulo 3

Definição do protótipo

O acesso aos meios subaquáticos envolve por vezes operações complicadas e perigosas do ponto de vista humano. Torna-se extremamente importante desenvolver robôs que executem tais tarefas de forma autónoma, preferencialmente. Assim evitam-se as limitações que advêm das ligações físicas aos robôs para controlo remoto (tipo ROV) e do alcance das comunicações eletromagnéticas debaixo de água.

O projeto do BUV Tobias baseou-se numa série de requisitos operacionais que fundamentaram a construção do design, estrutura e equipamento constituinte do veículo. O sistema foi organizado em módulos que servirão de base para o seu futuro desenvolvimento.

3.1 Características e especificações

No capítulo anterior foram analisados diversos BUVs desenvolvidos pelas mais diversas organizações, retirando, assim, a informação necessária para iniciar o projeto do veículo que será descrito neste capítulo.

O que se pretende para este veículo é um BUV de pequenas dimensões e, como primeiro protótipo, deverá possuir requisitos pouco exigentes, como por exemplo apenas deverá ser capacitado para mergulhar numa piscina de profundidade máxima de $10m$. Poderá ainda executar manobras nessa piscina e a operar em pequenos rios e lagos. Posteriormente, e para protótipos futuros, deverão ser capaz de operar na costa portuguesa.

A escolha de hardware depende essencialmente do orçamento disponibilizado para o projeto SABUVIS II e das especificações pretendidas para este BUV. Desta forma foi necessário, inicialmente, definir quais seriam as especificações do protótipo com base nas especificações requeridas pelo projeto SABUVIS II.

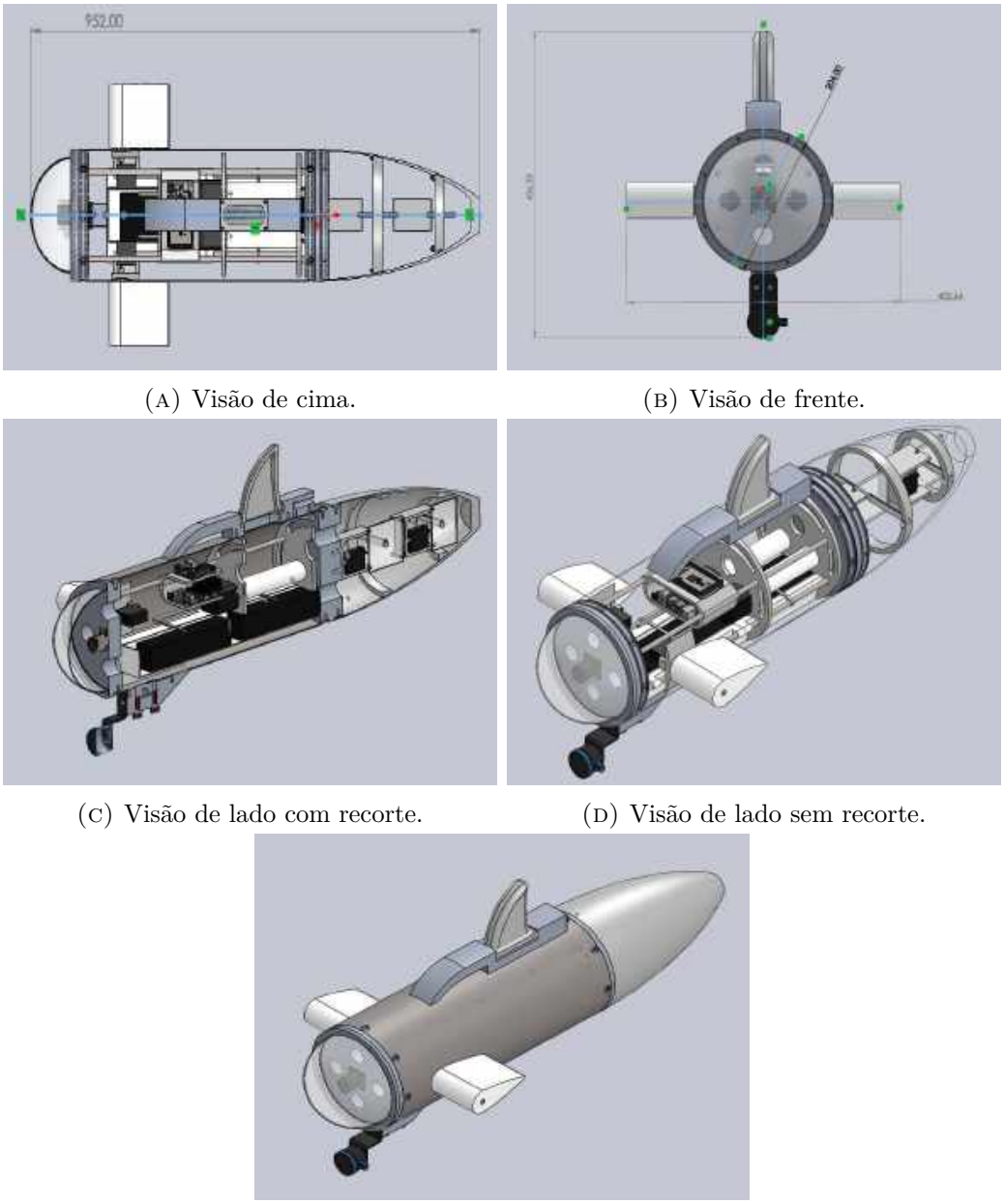
As especificações estipuladas para o primeiro protótipo foram:

- Velocidade horizontal máxima: 1 m/s;
- Autonomia: 2 h (a uma velocidade média de 0.5 m/s);
- Profundidade máxima: $10m$;
- Velocidade vertical máxima: 0.5 m/s;
- Comunicações: constante troca de informação entre a *ground station* e o veículo;
- Águas navegáveis: costa portuguesa (água salgada entre os 5 e 27 $°C$)

3.2 Design e estrutura

O ASPOF EN-MEC Dias de Paiva, na sua dissertação de mestrado, desenvolveu a estrutura e o design do veículo, tendo como base os requisitos e a escolha do hardware que será descrito ao longo desta dissertação. Na figura 3.1 poderá observar-se vários desenhos do design deste primeiro veículo.

O BUV, que foi apelidado de Tobias, tem $952mm$ de comprimento, $436,23mm$ de altura e $405,64mm$ de largura. Pesa cerca de $21Kg$ e tem um volume de $8.4dm^3$. O veículo pode ser dividido em três partes: nariz, corpo e cauda. O nariz é um domo de diâmetro interno $176mm$ com comprimento de $90mm$ em acrílico (espessura de $2mm$) com estrutura de suporte em alumínio, o corpo é um cilíndrico de uma liga de aço com $204mm$ de diâmetro e $500mm$ de comprimento e a cauda com comprimento de $339mm$, construída em ecoflex (membrana num tipo de silicone impermeabilizante), constituída por dois elos (construídos em Poliácido Láctico (PLA)).



(A) Visão de cima.

(B) Visão de frente.

(C) Visão de lado com recorte.

(D) Visão de lado sem recorte.

(E) Visão real.

FIGURA 3.1: Desenhos 3D do BUV Tobias.
Fonte: disponibilizados pelo ASPOF EN-MEC Dias de Paiva.

3.3 Organização do sistema robótico

Após definidas as especificações de capacidade e operação do veículo Tobias, foi necessário organizar todo o sistema robótico, dividindo-o em vários módulos: navegação, Reconhecimento (RECON), locomoção, potência, Limitação de Avarias (LA) e comunicações. Cada módulo pode ser constituído por um ou mais equipamentos/sistemas e foi projetado para satisfazer os requisitos predeterminados. Esta

organização encontra-se esquematizada na figura 3.2.

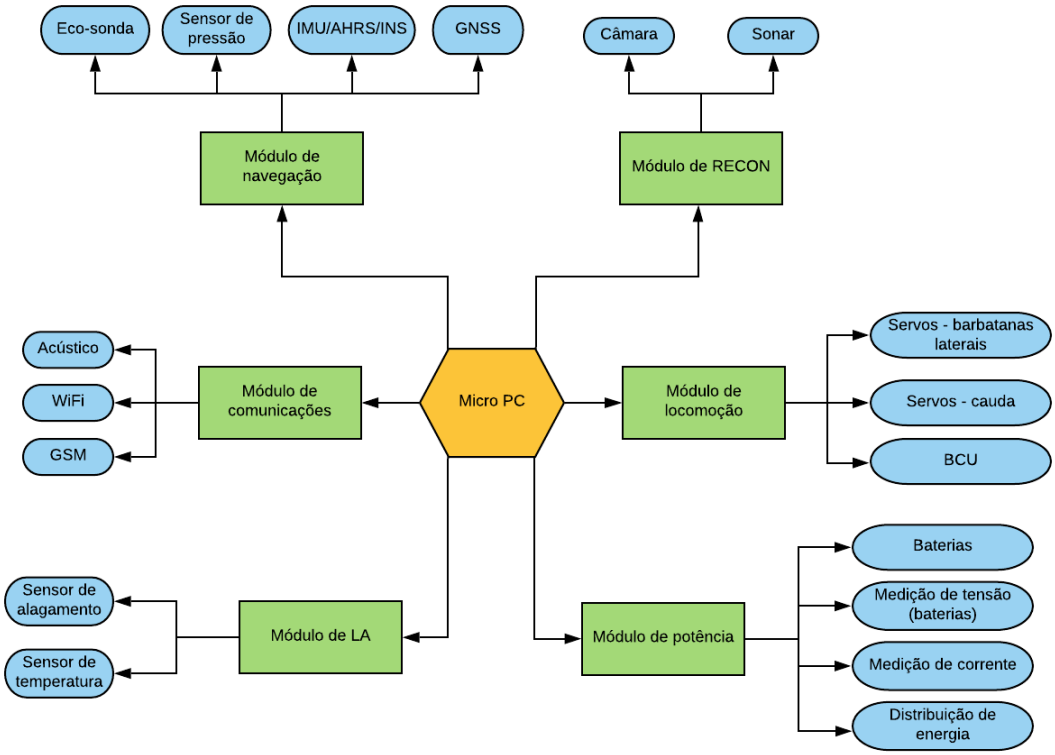


FIGURA 3.2: Divisão do sistema em módulos (hardware).

Os diversos módulos podem ser descritos da seguinte forma:

O módulo de Micro PC consiste numa unidade computacional capaz de processar os dados que provêm dos mais diversos sensores, de receber e transmitir informação, e tendo em conta a missão a executar, atuar sobre os atuadores. Basicamente é o centro nevrálgico do sistema e onde são tomadas as decisões de alto nível.

O módulo de comunicações A profundidade de operação do protótipo trará um desafio para as comunicações. As ondas eletromagnéticas a partir dos $5m$ abaixo da linha de superfície deixam de ser eficazes (Jiang & Georgakopoulos, 2011). Desta forma, torna-se necessário a utilização de ondas acústicas para estabelecer a comunicação subaquática entre a *ground station* ou com outros veículos, o que é conseguido através da integração de um modem acústico no BUV.

Para além da comunicação acústica será necessário utilizar um modem de GSM para possibilitar a troca de informação a longo alcance. Para a fase de

testes, controlo remoto e envio de comandos de forma mais expedita deverá ser utilizado um modem WiFi.

Em suma, o módulo de comunicações deverá possuir três tipos de equipamentos para transmitir e receber informações com objetivos diferentes: GSM - mensagens de emergência e rotina quando afastado da estação em terra cerca de $300m$, WiFi - telemetria e modem acústico - comunicação com a estação em terra, quando se encontrar debaixo de água, e com outros veículos.

O módulo de navegação deverá ser constituído por diversos sensores para que o BUV possa ser autónomo e tenha a capacidade de estimar o seu estado mediante os dados que recebe dos sensores, ou seja, para fornecer uma estimativa o mais precisa possível da localização, orientação e velocidade do veículo.

Existem três tipos de sensores inerciais diferentes atualmente no mercado:

- A *Inertial Measurement Unit* (IMU) é um sistema que faz a medição de aceleração linear e velocidade angular, e é geralmente composto por um giroscópio com três graus de liberdade, um acelerómetro com três graus de liberdade e um magnetómetro com com três graus de liberdade, originado nove graus de liberdade. A IMU, por si só, não fornece nenhum tipo de solução de navegação (posição, velocidade e atitude), apenas atua como um sensor.
- O *Attitude and Heading Reference System* (AHRS) é um sistema que incorpora uma IMU combinada com um processador integrado que cria um sistema de três eixos capaz de medir os ângulos da proa (*yaw*), *pitch* e *roll* de um objeto em movimento no espaço *3D*. Este sistema pode ainda integrar um filtro *Kalman*[25] para calcular a solução de orientação das leituras das medidas. A principal diferença para a IMU, é que a AHRS já realiza o processamento no *chip* que permite obter a pose do veículo (só com a IMU este processamento teria de ser feito externamente).
- O *Inertial Navigation System* (INS) é constituído por uma IMU, por forma a criar um sistema de navegação independente que usa medições fornecidas pela IMU para rastrear a posição, velocidade e orientação de um objeto em relação a um ponto de partida. Pode utilizar um sensor

[25]Método matemático criado por *Rudolf Kalman* que utiliza medições de grandezas realizadas ao longo do tempo (contaminadas com ruído e outras incertezas) para gerar resultados que tendam a aproximar-se dos valores reais das grandezas medidas e dos valores associados (R. G. Brown & Hwang, 2012).

Global Navigation Satellite System (GNSS)[26] para estabilizar o desvio do giroscópio e fornecer uma estimativa mais precisa do vetor de aceleração inercial, isto é, uma versão melhorada do AHRS.

Na sua constituição deverá conter quatro sensores: um sensor de pressão para receber a variável da pressão a que o veículo se encontra, que através do processamento dessa informação dará o valor da profundidade; um sensor inercial que tanto poderá ser uma IMU, um AHRS ou um INS que será utilizado para fazer a estima da posição e orientação quando o veículo não estiver à superfície, isto é, quando, por motivos de cobertura do sinal, não consegue estabelecer contacto com os satélites de GNSS; um sistema de GNSS para determinar a sua posição correta quando à superfície utilizando satélites; e uma eco-sonda que servirá para avaliar o panorama de contactos ao redor do veículo, isto é, servirá para detetar contactos submarinos, utilizando ondas acústicas e por conseguinte servirá para o veículo poder desviar-se de obstáculos que se encontrem na sua rota.

O módulo de LA surge da necessidade do veículo possuir capacidade para detetar se existe algum perigo interno, quer seja causado pela entrada de água quer pela elevada temperatura interna, para evitar avarias ou mesmo danos irreversíveis.

Como o protótipo, na grande parte da sua utilização, estará submerso no ambiente aquático, poderá correr o risco de danificar os seus circuitos elétricos pela entrada de água, que poderá ser provocada por embates em certos objetos, má vedação do casco e dos elementos exteriores ao mesmo ou por contacto com outros seres vivos, como aconteceu com um AUV *REMUS SharkCam* em 2013 no México, figura 3.3. Desta forma, o robô deverá ser equipado com um sistema de deteção de entrada de água.

[26] O GNSS, ou sistema de navegação global por satélite, pode ser definido como um sistema capaz de proporcionar o posicionamento a nível global usando uma constelação de satélites. O Norte Americano GPS, o Russo GLONASS e o Europeu Galileo são alguns exemplos (Vallejo, Sanguino & Rodrigues, 2014).

FIGURA 3.3: Ataque de um tubarão branco a um AUV *REMUS SharkCam*. Fonte: obtido do *website* WHOI - Robotic Vehicles Offer a New Tool in Study of Shark Behavior[27].

Com o processamento interno de informação, pela dissipação de energia em forma de calor da distribuição de energia por todos os componentes elétricos do veículo do BUV e a possível existência de um curto circuito poderá levar a um aumento de temperatura interna, que poderá levar falhas no processamento e na transmissão de informação, bem como, a danos irreparáveis nos módulos. Desta forma o veículo deverá ter algum sensor capaz de monitorizar constantemente a temperatura interna.

O módulo de RECON deverá conter sensores e equipamentos eletrónicos com a capacidade de análise e obtenção de dados para a compilação do cenário situacional da missão.

Todos os navios da Marinha que executam missões de ASW e MCM, como é o caso dos submarinos e das fragatas da classe Vasco da Gama, são equipados com um *Sound Navigation and Ranging* (SONAR). Este equipamento fornece aos navios a capacidade de deteção e seguimento de sinais acústicos a profundidades elevadas gerados por outros navios (se for passivo) ou gerados pelo próprio SONAR do navio (se for ativo).

A câmara é um dos equipamentos, geralmente, adicionados a qualquer AUV para permitir ao operador a visualização em imagem real do trajeto percorrido pelo veículo, onde se encontram gravados todos os contactos visuais.

O módulo de locomoção é constituído por três subgrupos: o submódulo dos servos das barbatanas laterais, constituído por dois servos, um em cada borda,

[27] *Website WHOI - Robotic Vehicles Offer a New Tool in Study of Shark Behavior*: https://www.whoi.edu/press-room/news-release/sharkcam-paper/.

que são fundamentais para controlar o ângulo de *pitch* (mergulhar e emergir à superfície) e também podem servir para ajudar na propulsão, travagem e guinadas do veículo. Para além disso, mesmo que o ângulo de *roll* seja automaticamente e mecanicamente estabilizado, podem também ser usadas para o controlar durante as manobras do veículo; o submódulo dos servos da cauda, constituído por dois servos, conferindo dois graus de liberdade ao veículo para se poder deslocar com movimento idêntico ao dos peixes; e o submódulo BCU, que controlará a flutuabilidade do veículo permitindo que este possa emergir e imergir com mais facilidade.

O módulo de potência é responsável pelo fornecimento de energia a todo o sistema, pelo que, deverá ser constituído por um conjunto de baterias, um sistema de monitorização de tensão e capacidade das baterias, um sistema de distribuição de energia, e um sistema de monitorização de corrente das ligações críticas da instalação elétrica.

Na primeira versão deste protótipo optou-se, por razões do custo associado, em prescindir quer do sonar como *payload* quer do modem acústico como canal de comunicação. Apesar de comunicações debaixo de água serem essenciais para a cooperação de um *swarm*, este aspeto será deixado para uma versão posterior do protótipo, mesmo que no capítulo seguinte se faça uma proposta de equipamento deste tipo (sonar e modem acústico) para este veículo.

Capítulo 4

Projeto de hardware

Neste capítulo é descrito o projeto de hardware, onde são selecionados os componentes eletrónicos de cada módulo descritos no capítulo anterior e evidenciados na figura 3.2. Para isso foram analisados e comparados diversos sensores e *modems* dentro de cada tipo de equipamento. Foram consultadas várias empresas e foram pedidos vários orçamentos. Foi também realizado um levantamento energético do conjunto desses equipamentos para ser possível a seleção das baterias.

O capitulo começa com a escolha do hardware, seguido do levantamento das conexão do hardware selecionado ao Micro PC e acaba com um resumo em forma de tabela com todos os equipamentos adquiridos.

4.1 Micro PC

Existem muitas alternativas populares de micro PCs, como por exemplo, as placas *PC/104* que podem correr Linux ou mesmo Windows (PC/104 Consortium, 2020), o *LattePanda* que corre apenas Windows e tem uma interface para o Arduíno (LattePanda, 2020) e o *Raspberry Pi* que corre Linux (RaspberryPi, 2020). Recentemente, a NVIDIA desenvolveu uma placa de baixo custo, a *Jetson Nano*, com alto desempenho computacional voltada para o desenvolvimento de aplicações de alta performance e procura servir de base para projetos nas áreas de robótica, visão computacional, assistentes domésticos, drones e veículos autónomos (NVIDIA, 2019b). Tem a capacidade para executar várias redes neuronais em paralelo para aplicações como classificação de imagens, deteção de objetos, segmentação e processamento de fala. Como se pretende construir veículos de baixo custo a visão computacional é uma excelente opção, pois uma câmara aliada a um bom processamento de imagem podem ser a alternativa a sensores mais dedicados e muito mais caros, como por exemplo os *SideScan* sonares. Nesse contexto existe todo o interesse em utilizar uma placa que tenha boas capacidades computacionais no domínio do

processamento de imagem, onde a *Jetson Nano* tem uma grande vantagem sobre os concorrentes que apresenta.

As diferentes especificações técnicas de cada equipamento podem ser consultadas na tabela 4.1.

TABELA 4.1: Especificações dos *mini PCs* mais populares para projetos de AUVs.

	Jetson Nano	**PC/104 Plus**	**Raspberry Pi 4**	**Latte Panda**
GPU	128-cores NVIDIA Maxwell	Intel HD Graphics Gen 7 Engine	Broadcom BCM2711	Intel HD Graphics
CPU	Quad-core ARM A57 1.43 GHz	Intel Celeron 1.58 GHz	Quad core Cortex-A72 1.5GHz	Intel Cherry Trail Z8350 Quad Core 1.8GHz
Memória	4 GB 64-bit	4 GB 64-bit	4 GB 64-bit	4 GB 64-bit
Peso $[g]$	136	120	65	100
Dimensões (C x L) $[mm]$	69 x 45	90 x 96	85 x 56	88 x 70
Temperatura de operação $[ºC]$	0 a +50	-20 a +60	0 a +50	-10 a +50
Tensão de alimentação $[V]$	5	5	5	5
Corrente máxima $[A]$	2.5	1.7	3	2
Conexões	40-Pin GPIO, Socket M.2 Key	ISA & PCI support	40 pin GPIOs	26 pin GPIOs e 6 fichas
Portas	4 USB 3.0, 1 USB 2.0 Micro-B	3 USB 2.0, 2 RS-232/485	2 USB 3.0 e 2 USB 2.0	1 USB 3.0 e 2 USB 2.0
Performance AI	472 GFLOPS	326 GFLOPS	8 GFLOPS	2.72 GFLOPS
Preço [€]	88,61 (SparkFun)	273,60 (Mouser Eletronics)	59,99 (Robert Mauser Lda)	187.07 (DFRobot)

O micro PC escolhido foi a Jetson Nano com um cartão de memória *flash microSD* de $64GB$ no qual foi instalado o Ubuntu 18.4 como Sistema Operativo (SO), figura 4.1, uma vez que apresenta um baixo custo, alto desempenho do Unidade

Central de Processamento (*Central Processing Unit*) (CPU) e para futuras implementações, possui uma boa capacidade para IA (472 *Giga Floating-point Operations Per Second* (GFLOPS), medida de alto desempenho do CPUs para operações com pontos flutuantes), como por exemplo implementação de IA para deteção de objetos utilizando uma câmara, criação de um mundo virtual, aprendizagem automática para o controlo das barbatanas, etc.

FIGURA 4.1: Visão 3D da *Jetson Nano Developer Kit*.
Fonte: obtido de NVIDIA (2019b).

4.2 Módulo de comunicações

Como foi referido no capítulo anterior, é necessário selecionar um *modem* GSM, um *modem* acústico e WiFi.

4.2.1 GSM

Existem diversos módulos e *shields* com GSM, alguns deles incluem também uma interface para antena GPS. Foram analisados diversos módulos com e sem a interface GPS. Por motivos de simplificação do processamento e do tratamento dos dados, optou-se por se escolher um módulo sem interface GPS, para assim se poder adquirir à parte um sensor GPS dedicado, com melhor exatidão. O módulo escolhido foi o *SIM800L GSM-GPRS 5V V2.0*, figura 4.2, desenvolvido pela *SIMCom* e distribuído em Portugal pela *Box Eletrónica* (SIMCom, 2013).

FIGURA 4.2: Visão 3D do *SIM800 L GSM-GPRS 5V V2.0*.
Fonte: obtido do *website* da Box Electŕonica[28].

Este módulo possui capacidade para os serviços GSM, *General Packet Radio Service* (GPRS), *Short Message Service* (SMS), *Unstructured Supplementary Service Data* (USSD) e para os serviços de voz e internet utiliza *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), *Hypertext Transfer Protocol* (HTTP) e *File Transfer Protocol* (FTP). A comunicação do módulo com o microcontrolador utiliza um *Universal Asynchronous Receiver-Transmitter* (UART)[29] embutido no módulo. As especificações do módulo podem ser consultadas na tabela 4.2.

[28] *Website* da Box Electrónica: https://www.boxelectronica.com/pt/gsm-gprs-gps/1089-modulo-gprs-gsm-wireless-sim800l-5v-v20.html.

[29] Um UART é um dispositivo de hardware para efetuar a comunicação série assíncrona com outro periférico em que o formato de dados e velocidades de transmissão são configuráveis. É geralmente um circuito integrado individual ou parte de um (https://www.circuitbasics.com/basics-uart-communication/).

Tabela 4.2: Características do módulo *SIM800L GSM-GPRS 5V V2.0*.

Especificações técnicas	
Baud Rate	1200*bps* a 115200*bps*
Frequência	*Quad-Band* GSM 850/900/1800/1900 MHz
Suportes	GSM, GPRS, SMS, USSD, TCP, UDP, HTTP e FTP
Outras especificações	
Potência	2*W*
Tensão de alimentação	4.6 a 5.2*V*
Corrente de alimentação	1 a 2.6*A*
Temperatura de operação	−40 a +85ž*C*
Conexão	*UART*
Dimensões (C x L x H) [*mm*]	27 x 39 x 10
Peso	18*g*

4.2.2 Modem Acústico

No projeto anterior SABUVIS, foi escolhido como módulo acústico o *WHOI Micromodem*, conforme a figura 4.3, por questões de compatibilidade com os AUVs da Marinha Portuguesa *SeaCon* do DMS3. Durante o projeto foram utilizados os *drivers* já então desenvolvidos para os *SeaCon* em *DUNE Uniform Navigation Environment* (DUNE), tanto para o *BUV 3* como para a *Ground Station*. Os modems, muitas vezes, não são compatíveis entre si e como o objetivo final do projeto SABUVIS II é dotar a Marinha Portuguesa de um cardume de veículos biomiméticos que executam missões em cooperação, todos os veículos submarinos da Marinha devem comunicar entre si, isso implica a utilização do *Micromodem* para se poder falar com o BUV3, que será o líder do *swarm*. Desta forma, pelo conhecimento desenvolvido e por questões de compatibilidade, o modem acústico escolhido foi o *WHOI Micromodem*.

FIGURA 4.3: *WHOI Micromodem.*
Fonte: obtido de Acoustic Communications Group (2020).

4.2.3 WiFi

Atualmente existem diversos *modems* WiFi no mercado dos mais diversos tamanhos para diferentes objetivos. Foram analisados o *Ubiquiti PicoStation M Series* da Ubiquiti (NetWifiWorks, 2020) utilizado no *BUV 3* do projeto anterior SABUVIS, entretanto descontinuado, o *Intel Dual Band Wireless-Ac 8265* da Intel (Intel, 2020) e o *WiFi Bullet BM2HP* da Ubiquiti (Ubiquiti, 2019) utilizado no *BUV 2* do projeto anterior SABUVIS.

Por aconselhamento da Escola Naval Polaca e tendo em conta o baixo custo, o grande alcance e a alto desempenho em altitudes neutras foi escolhido o *Bullet BM2HP*, conforme a figura 4.4, como modem WiFi. A alimentação deste módulo é conseguida por *Power over Ethernet* (PoE). As características do equipamento encontram-se na tabela 4.3.

FIGURA 4.4: Visão 3D do modem *WiFi Bullet BM2HP*.
Fonte: obtido de Ubiquiti (2019).

Tabela 4.3: Características do modem *WiFi Bullet BM2HP*.

Especificações técnicas	
Alcance	$50km$ [30]
Frequência	$2.4GHz$
Velocidade de transmissão	$100 + Mbps$
Outras especificações	
Potência	$8W$
Tensão de alimentação	5 a $24V$ PoE
Temperatura de operação	-40 a $+80$žC
Conexão	*Ethernet*
Dimensões (C x L x H) $[mm]$	152 x 37 x 31
Peso	$180g$

4.3 Módulo de navegação

Nesta secção é descrita a análise e seleção do sensor de pressão, sensor inercial, sensor GNSS e a eco-sonda.

4.3.1 Sensor de pressão

A profundidade a que o AUV se encontra é uma informação de extrema importância pois é indispensável para o estabilizar na coluna de água. A leitura da profundidade consegue-se através da medição da pressão da água junto ao veículo. Para isso é utilizado, então, o sensor de pressão.

O sensor de pressão deverá ser implementado no casco do BUV para ser capaz de medir a pressão da água que é proporcional à profundidade que o veículo se encontra. Desta forma, o sensor deverá fornecer medidas de pressão absoluta.

Foram analisados diversos sensores, no entanto, os que se evidenciaram foram: o *DST800 Smart* da *AIRMAR* capaz de medir a velocidade, a temperatura e a profundidade (AIRMAR, 2020) mas com um custo de €225.07 e o *Bar30 High-Resolution 300m Depth/Pressure Sensor* da *BlueRobotics*, utilizado no *BlueROV2*

[30] Alcance máximo com recurso a antenas direcionais com elevado ganho (https://www.ui.com/airmax/bulletm/).

(subsecção 2.2.4), com uma resolução de $2mm$ com um custo de €61,72 (BlueRobotics, 2020a).

O sensor escolhido foi o *Bar30 High-Resolution 300m Depth/Pressure Sensor*, figura 4.5, devido ao seu baixo custo face aos outro sensores dedicados à aplicação em robôs submarinos e às suas principais características, presentes na tabela 4.4.

FIGURA 4.5: Visão 3D do *Bar30 High-Resolution 300m Depth/Pressure Sensor*.
Fonte: obtido de BlueRobotics (2020a).

TABELA 4.4: Características do *Bar30 High-Resolution 300m Depth/Pressure Sensor*.

Especificações técnicas	
Resolução	$2mm$
Máxima pressão mecânica	$50bar$
Pressão de operação	0 - $30bar$
Precisão	$+/-200mbar$ $(2.04m)$
Profundidade máxima de operação	$300m$
Outras especificações	
Corrente de alimentação	$1.25mA$
Tensão de alimentação	2.5 - $5.5V$
Temperatura de operação	-20 a $+85$žC
Conexão	*Inter-Integrated Circuit* (I^2C)
Dimensões (C x L x H) $[mm]$	30 x 30 x 37
Peso	$30g$

4.3.2 IMU/AHRS/INS

Na fase inicial tencionava-se adquirir um INS pelas suas caraterísticas e exatidão, mas, através de vários orçamentos pedidos a diversas empresas, chegou-se à conclusão que o custo de aquisição deste sistema seria alto para o orçamento disponível. Desta forma optou-se por um AHRS ou uma IMU que apresentam um custo de aquisição mais baixo e com características de localização inercial precisas e suficientes para as especificações desejadas para o BUV.

Foram analisados e comparados os sensores *VN-100T AHRS* da *VectorNav Technologies*, o *Orientus* da *Advanced Navigation*, *STIM300 IMU* e *STIM318 IMU* da *Sensonor* e o *Ellipse 2 Micro AHRS* da *SBG Systems*. Na tabela 4.5 pode comparar-se as diferentes especificações e características de cada sensor.

TABELA 4.5: Especificações dos AHRS e IMU revistos.

	VN-100T	***Orientus***	***STIM300***	***STIM318***	***Ellipse 2 Micro***
Tipo	AHRS	AHRS	IMU	IMU	AHRS
Gyro Bias Instability	$5° \, h^{-1}$	$3 \, ° \, h^{-1}$	$0.3 \, ° \, h^{-1}$	$0.3 \, ° \, h^{-1}$	$7 \, ° \, h^{-1}$
Accelerometer Bias Instability	$< 0.04mg$	$< 20ug$	$< 0.5mg$	$< 0.003mg$	$< 5mg$
Roll & Pitch Accuracy (Static)	$0.5 \, ° \, h^{-1}$	$0.2 \, ° \, h^{-1}$	N/D	N/D	$0.05 \, ° \, h^{-1}$
Heading Accuracy (Static)	$1.0 \, ° \, h^{-1}$	$0.5 \, ° \, h^{-1}$	N/D	N/D	$0.4 \, ° \, h^{-1}$
Roll & Pitch Accuracy (Dynamic)	$1.0 \, ° \, h^{-1}$	$0.6 \, ° \, h^{-1}$	N/D	N/D	$0.1 \, ° \, h^{-1}$
Heading Accuracy (Dynamic)	$2.0 \, ° \, h^{-1}$	$1.0 \, ° \, h^{-1}$	N/D	N/D	$0.8 \, ° \, h^{-1}$
Preço [€]	€998	€1129.05	€6500	€7500	€1700
Potência	$220mW$	$325mW$	1.5 a $2W$	1.8 a $2.5W$	$400mW$
Peso	15	25	55	57	10
Dimensões (C x L x H) $[mm]$	36 x 33 x 9	30 x 40.6 x 24	44.8 x 38.6 x 21.5	44.8 x 38.6 x 21.5	26.8 x 18.8 x 9.5
Protocolo de comunicação	RS-232, UART	RS-232	RS-232	RS-232	RS-232, RS422, USB

Tendo em conta a comparação presente na tabela 4.5, face aos valores de aquisição e precisão, o sensor escolhido foi o *VN-100T* presente na figura 4.6. Para além de apresentar o valor de aquisição mais baixo, apresenta uma exatidão para o *pitch* e *roll* aceitáveis para o tempo máximo que o protótipo deverá estar em missão.

FIGURA 4.6: AHRS *VN-100T Development Kit*.

4.3.3 GNSS

O sistema de GNSS é definido por constelações de satélites que permitem determinar a posição e a localização de qualquer objeto no globo terrestre. Dentro destas constelações fazem parte o GPS dos Estados Unidos, *BeiDou Navigation Satellite System* (BDS) da China, *Quasi-Zenith Satellite System* (QZSS) do Japão, *Global Navigation Satellite System* (GLONASS) da Rússia, *Indian Regional Navigation Satellite System* (IRNSS) da Índia e o *Galileo* (GAL) da União Europeia (UE). A performance deste sistema poderá ser melhorada pelos SBAS regionais que melhoram a precisão e a confiança das informações, corrigindo os erros de medição do sinal e fornecendo as informações sobre a precisão, integridade, continuidade e disponibilidade dos seus sinais (GSA, 2018). Diversos países implementaram seu próprio SBAS, por exemplo, na Europa, o EGNOS abrange a maioria da UE, juntamente com alguns países e regiões vizinhas.

Foram analisados e comparados os sensores *Hornet ORG1518-R01* e *Hornet ORG1518-R02* da *Origin GPS*, o *OEM7600* da *NovAtel*, *NEO-M8N* da *M5Stack* e o *U-Blox NEO-7M* da *u-blox*. Na tabela 4.6 pode comparar-se as diferentes especificações e características de cada sensor.

TABELA 4.6: Especificações dos sensores de GNSS.

	Hornet ORG1518-R01	***Hornet ORG1518-R02***	***NovAtel OEM7600***	***M5Stack NEO-M8N***	***U-Blox NEO-7M***
Constelações	GPS, GLONASS	GPS, GLONASS, BDS, GAL	GPS, GLONASS, GAL, IRNSS, BDS	GPS, GLONASS, QZSS, GAL e BDS	GPS, GLONASS, QZSS, SBAS
Precisão	$< 1.5m$	$< 1.5m$	$< 1.5m$	$< 2.5m$	$< 4m$
Preço [€]	€283.21	€280.98	€872	€40.45	€32.42
Temperatura de Operação	-40 a $+85žC$	-40 a $+85žC$	-40 a $+85žC$	-40 a $+85žC$	-40 a $+85žC$
Potência	$15mW$	$15mW$	$0.9W$	$15mW$	$15mW$
Tensão de Alimentação	$1.8V$	$3.3V$	$3.3V$	2.7 a $3.6V$	3.5 a $5V$
Peso	$2.5g$	$8g$	$31g$	$43g$	$15g$
Dimensões (C x L x H) $[mm]$	17 x 17 x 6.7	17 x 17 x 6.7	35 x 55 x 11	54.2 x 54.2 12.8	40 x 25 x 15
Protocolo de comunicação	UART - *National Marine Electronics Association* (NMEA)	UART - NMEA	USB ou Ethernet	UART - NMEA	USB ou UART

Após a comparação dos diversos sensores, o módulo de GNSS escolhido foi o *Velleman VMA430* equipado com o *U-Blox NEO-7M*, figura 4.7, por ter um valor de aquisição baixo, por ter uma tensão de alimentação de $5V$, ao contrário do *NEO-M8N*, que apesar de ser de uma geração mais recente, não possui uma tensão compatível com os reguladores de tensão de $5V$ que serão implementados no veículo, e pela média precisão de localização, que para o protótipo inicial é suficiente.

FIGURA 4.7: Sensor GNSS *Velleman VMA430* equipado com o *U-Blox NEO-7M*.
Fonte: obtido do *website* da *Mauser PT*[31].

4.3.4 Eco-sonda

A eco sonda, neste projeto, será utilizada para evitar contactos (*contact avoidance*), isto é, tendo em conta a direção a que o veículo se está a deslocar ela irá fazer a procura de contactos nesse azimute e fornecerá esses dados ao computador central.

No mercado, as mini eco-sondas para o meio aquático apresentam um custo de aquisição não compatível com a filosofia de baixo custo que se pretende seguir na construção do veículo. A única alternativa a esta situação com um custo aceitável foi o *Ping Sonar Altimeter and Echosounder* da *BlueRobotics*, figura 4.8.

FIGURA 4.8: Visão 3D da sonda *Ping Sonar Altimeter and Echosounder*.
Fonte: obtido de BlueRobotics (2020b).

[31] *Website* da *Mauser PT*: https://mauser.pt/catalog/product_info.php?cPath=1667_2604_2607&products_id=096-6118.

Possui um alcance máximo de $30m$, custo de aquisição de €253.22 e as principais características estão evidencias na tabela 4.7 (BlueRobotics, 2020b). Esta eco sonda é um sonar de feixe único (*single-beam*), constituído por um transdutor piezoelétrico que envia um pulso acústico ultrassónico para a água e, em seguida, recebe os ecos de retorno. Através da análise dos ecos, é possível determinar a distância até o eco mais forte, que geralmente é o fundo do oceano ou um objeto de grandes dimensões. A empresa disponibiliza uma interface de utilizador de código aberto e as bibliotecas de desenvolvimento para *Arduino*, `C++` e *Python*.

TABELA 4.7: Características da sonda *Ping Sonar Altimeter and Echosounder*.

Especificações acústicas	
Frequência	$115kHz$
Largura de feixe de medição	30ž
Resolução de alcance a $30m$	$15cm$
Resolução de alcance a $2m$	$1cm$
Resolução de alcance	0.5% do alcance
Taxa de refrescamento do horizonte	1 *a* 30 $leituras/sec$
Especificações técnicas	
Tipo de sonar	*single-beam*
Alcance máximo	$30m$
Alcance mínimo	$0.5m$
Profundidade máxima de operação	$300m$
Outras especificações	
Corrente de alimentação	$100mA$
Tensão de alimentação	$5V$
Temperatura de operação	-20 a $+85$žC
Conexão	Serial UART
Dimensões (C x L x H) $[mm]$	59.5 x 48 x 34.5
Peso	$135g$

4.4 Módulo de LA

Nesta seleção foi selecionado o sensor de alagamento e estudado uma forma de obter o valor da temperatura, através de outros equipamentos selecionados para o veículo.

4.4.1 Sensor de alagamento

O sensor de alagamento servirá para detetar se existe alguma entrada de água para o interior do veículo autónomo, e caso isso aconteça, este deverá regressar à superfície a fim de evitar possíveis avarias ou mesmo danos irreversíveis.

Atualmente, no mercado existem diversos sistemas para este efeito. Como se pode observar no capítulo 2, existem veículos que utilizam um sensor de pressão interno, outros um sensor de nível de água e outros uma *Printed Circuit Board* (PCB) com ligações elétricas ao casco que testa a condutividade do mesmo.

O sensor escolhido para este fim foi o *SOS Leak Sensor* da *BlueRobotics*, figura 4.9. Utiliza quatro pontas de prova reutilizáveis com a ponta de esponja que ajudam a monitorizar todos os cantos do interior de um recipiente (BlueRobotics, 2020d). Quando uma entrada de água é detectada, a situação é assinalada alterando o valor lógico de um sinal digital de saída do sensor. As características deste equipamento podem ser consultadas na tabela 4.8.

Figura 4.9: Placa eletrónica do *SOS Leak Sensor*.
Fonte: obtido de BlueRobotics (2020d).

TABELA 4.8: Características do *SOS Leak Sensor*.

Especificações técnicas	
Até 4 pontas de prova Integração em equipamentos de $3.3V$ a $5V$	
Outras especificações	
Corrente de alimentação	$20mA$
Tensão de alimentação	3.3 - $5V$
Conexão	3 pinos
Dimensões (C x L x H) $[mm]$	24.5 x 13 x 9.5
Peso	$10g$

4.4.2 Sensor de temperatura

O sensor de temperatura encontra-se embutido em diferentes equipamentos como na *Jetson Nano* e *VN-100T*. Através desses sensores e utilizando software apropriado pode fazer-se a aquisição dos valores de temperatura interna do veículo.

4.5 Módulo de RECON

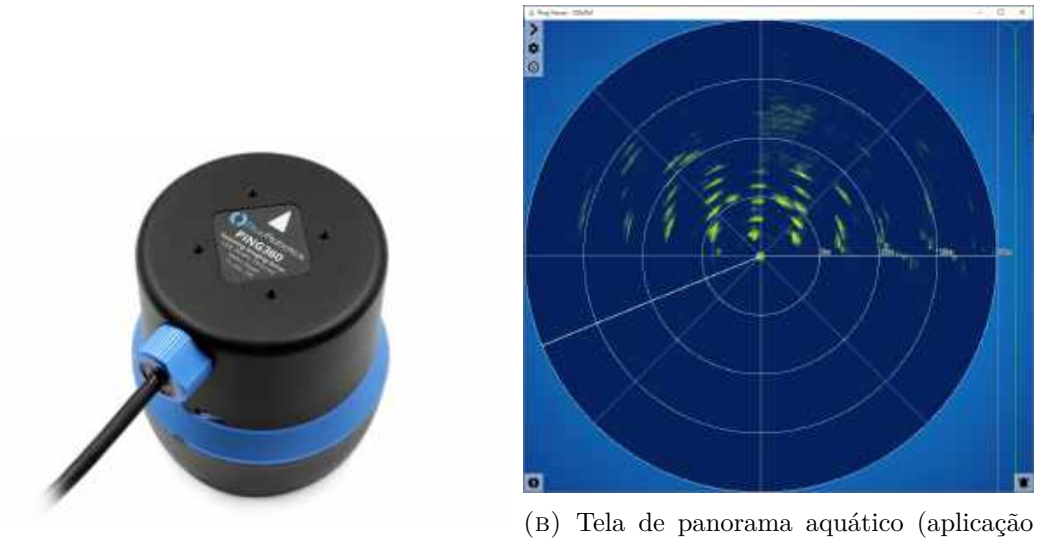
Nesta secção foram analisados diferentes sonares e foi proposta uma recomendação para futura aquisição. Também foram analisadas duas câmaras e selecionada apenas uma que fará parte deste módulo.

4.5.1 SONAR

Foram pedidos orçamentos a várias empresas de sonares, e, apenas dois, apresentaram um valor de aquisição razoável: o *Micron - Mechanical Scanning Sonar* da *Tritech* a €6739 e o *Ping360 Scanning Imaging Sonar* da *BlueRobotics* a €1356,85. O primeiro é amplamente utilizado nos AUVs como por exemplo no BUV1 e BUV2, *Ichthus V5.5*, já o segundo é utilizado no *BlueRov2*.

Tendo em conta o preço e as características de cada um dos sonares, recomenda-se o *Ping360 Scanning Imaging Sonar*, figura 4.10, que é um sonar de varredura (*scanning sonar*), onde dentro do sonar existe um transdutor acústico que envia um feixe acústico estreito para a água e depois escuta os ecos provenientes da reflexão. Ao transdutor é acoplado um motor que o faz girar de um em um grau e, ao

fazer isso, gera uma imagem circular dos contactos sonar com um alcance máximo de $50m$.



(A) Visão 3D.

(B) Tela de panorama aquático (aplicação *open source Ping Viewer*).

FIGURA 4.10: *Ping360 Scanning Imaging Sonar.*
Fonte: obtido de BlueRobotics (2020c).

As características encontram-se na tabela 4.9. O sonar pode ser ligado à aplicação *Ping-Viewer* de código aberto para controlo e exibição de dados. Esta aplicação poderá ser executada no *Windows*, *Mac* e *Linux* (BlueRobotics, 2020c). Pode ainda ser integrado noutros sistemas e receber os dados diretamente do sonar através da utilização de bibliotecas para *Arduino* e *Python* fornecidas pela *BlueRobotics*.

TABELA 4.9: Características do *Ping360 Scanning Imaging Sonar*.

Especificações acústicas	
Frequência	$115kHz$
Largura de feixe horizontal	2ž
Largura de feixe vertical	25ž
Setor digitalizado	Variável até 360
Resolução de alcance a $50m$	$4.1cm$
Resolução de alcance a $2m$	$1.6cm$
Resolução de alcance	0.08% do alcance
Velocidade de varrimento a $2m$	9 sec/360ž
Velocidade de varrimento a $50m$	35 sec/360ž
Especificações técnicas	
Tipo de sonar	*Mechanical scanning sonar*
Tipo de varrimento	*Horizontal*
Alcance máximo	$50m$
Alcance mínimo	$0.75m$
Profundidade máxima de operação	$300m$
Outras especificações	
Potência	$5W$
Tensão de alimentação	11 - $25V$
Conexão	USB, *Ethernet*, RS485
Dimensões (C x L x H) $[mm]$	77 x 77 x 83
Peso	$510g$

4.5.2 Câmara

Na maioria dos BUV desenvolvidos foi utilizada uma câmara de *streaming* e de gravação de vídeo de empresas desportivas como é o caso da *GoPro*. No entanto, este tipo de equipamentos impossibilita o processamento de imagem.

Para este projeto deverá utilizar-se uma câmara com *Camera Serial Interface* (CSI), interface de transmissão de dados de alta velocidade entre uma câmara e um processador *host*, isto é, basicamente é uma ligação direta ao processador que evita a sobrecarga de dados, como é o caso das câmaras *Universal Serial Bus*

(USB). Desta forma, poderá ser possível, no futuro, a utilização da IA da *Jetson Nano* no processamento e análise de imagem em tempo real. Já existem vários projetos e bibliotecas desenvolvidas que permitem a deteção de objetos, a classificação de imagens e a análise de distâncias a objetos (NVIDIA, 2019b).

Foram analisadas duas câmaras (tabela 4.10): a *Low-Light HD USB Camera* da *Blue Robotics* utilizada no AUV *BlueRov2* (Blue Robotics, 2017) e a *Arducam 8 MP Sony IMX219 with M12 lens LS40136* da *Arducam* utilizada em vários projetos de processamento de imagem no *Raspberry Pi* e na *Jetson Nano* (RobotShop, 2018).

Tabela 4.10: Especificações das câmaras revistas.

	Low-Light HD	***Arducam 8 MP Sony IMX219***
Resolução	$2.24MP$	$8MP$
Campo de Visão Horizontal	80ž	64ž
Campo de Visão Vertical	64ž	70ž
Suporte de Vídeo	$1080p$	$1080p30$, $720p60$ e $640x480p90$
Preço [€]	€80.51	€73.68
Corrente de Alimentação Máxima	$0.220A$	$0.4A$
Tensão de Alimentação	$5V$	$5V$
Peso	$10g$	$54g$
Dimensões (C x L x H) $[mm]$	32 x 32 x 23.35	36 x 36 x 20
Protocolo de comunicação	USB	CSI-2

A câmara escolhida foi *Arducam 8 MP Sony IMX219 with M12 lens LS40136*, figura 4.11, pela sua resolução de imagem e pelo protocolo de comunicação CSI-2 útil para futura implementação de IA no processamento de imagem.

FIGURA 4.11: *Arducam 8 MP Sony IMX219 with M12 lens LS40136.*
Fonte: obtido de RobotShop (2018).

4.6 Módulo de locomoção

Os três subgrupos deste módulo, submódulo dos servos das barbatanas laterais, submódulo dos servos da cauda e o submódulo BCU, deverão ser controlados por uma placa de desenvolvimento composta por um microcontrolador por uma questão de simplicidade porque senão seria necessário desenvolver mais hardware e boards próprias, o que complicaria o desenvolvimento. A placa *Arduino* escolhida, que receberá comandos do micro PC, foi o *Mega 2560*, figura 4.12, uma vez que possui 54 pinos de entradas e saídas digitais onde 15 destes podem ser utilizados como saídas *Pulse-Width Modulation* (PWM). Possui ainda 16 entradas analógicas e 4 portas de comunicação série (https://store.arduino.cc/arduino-mega-2560-rev3). Para além das características de pinos, não utiliza o processador *AT MEGA 328*, que, segundo vários fóruns de desenvolvimento[32], apresenta problemas de compatibilidade com interface serial entre o ROS (*middleware* que será apresentado no capítulo seguinte) e os microcontoladores *Arduino*. Na tabela 4.11 pode comparar-se todos os microcontroladores analisados.

[32] http://wiki.ros.org/rosserial_arduino, https://forum.arduino.cc/index.php?topic=329438.0 e https://forum.arduino.cc/index.php?topic=326017.0.

Tabela 4.11: Comparação entre as diversas placas de desenvolvimento *Arduino*.

	UNO	**Mega 2560**	**Leonardo**	**Due**	**Mega ADK**	**Nano**	**Pro mini**	**Esplora**
Processador	*AT MEGA 328*	*AT MEGA 2560*	*AT MEGA 32u4*	*AT 91SAM 3X8E*	*AT MEGA 2560*	*AT MEGA 328*	*AT MEGA 168*	*AT MEGA 32u4*
Portas digitais	14	54	20	54	54	14	14	N/D
Portas PWM	6	15	7	12	15	6	6	N/D
Portas analógicas	6	16	12	12	16	8	8	N/D
Clock	$16Mhz$	$16Mhz$	$16Mhz$	$84Mhz$	$16Mhz$	$16Mhz$	$8Mhz$	$16Mhz$
Memória $[Kbytes]$	32	256	32	512	256	32	16	32
Tensão de alimentação $[V]$	5	5	5	3.3	5	5	3.3	5
Conexões	USB	USB	Micro USB	Micro USB	USB	USB Mini-B	Série	Micro USB

Figura 4.12: *Arduino Mega 2560*.
Fonte: obtido do *website* *Arduino Mega*[33].

4.6.1 Servos das barbatanas laterais e cauda

O ASPOF EN-MEC Dias de Paiva, no âmbito da sua dissertação de mestrado, através de estudos estruturais, obteve a potência necessária para o veículo se deslocar e selecionou o tipo e modelo de servos ideais para o projeto. Por uma questão de compatibilidade optou por utilizar o mesmo tipo de servos para a cauda

[33] *Website* do *Arduino Mega*: https://store.arduino.cc/arduino-mega-2560-rev3.

e para as barbatanas laterais. O servo escolhido foi o *JX BLS-12V7146*, figura 4.13, distribuído pela *Banggood* com um preço unitário de €60.13. As especificações técnicas podem ser consultadas na tabela 4.12.

FIGURA 4.13: Servo *JX BLS-12V7146*.
Fonte: obtido do *website* *JX Servo*[34].

TABELA 4.12: Características do servo *JX BLS-12V7146*.

Especificações técnicas	
Tipo de motor	*Brushless*
Frequência de trabalho	1520µs/330Hz
Binário (11.1V)	38 kg cm^{-1}
Binário (14V)	47 kg cm^{-1}
Velocidade de operação (11.1V)	0.12s/60°
Velocidade de operação (14V)	0.10s/60°
Outras especificações	
Tensão de alimentação	$11.1 - 15V$
Conexão	*JR* 265mm
Dimensões (C x L x H) [mm]	40 x 20 x 37
Peso	71g

Neste sentido foram adquiridos quatro servos: dois para a cauda e dois para as barbatanas laterais, um para cada bordo.

Quando os servos estão inativos, isto é, quando não executam nenhum movimento, consumem uma corrente residual. Por forma a evitar que haja consumo de energia desnecessário, nesta situação, foi adquirido e implementado um módulo Relé 5V 4 Canais (*SRD-05VDC-SL-C*), figura 4.14. Este equipamento permite o controlo da ligação elétrica entre a fonte de alimentação (reguladores de tensão) e

[34] *Website* da *JX Servo*: http://www.jx-servo.com/en/Product/SERVO/12Vbs/462.html.

os servos. O controlo será efetuado a partir do *arduino*. A tabela 4.13 descreve as características do módulo.

FIGURA 4.14: Módulo Relé 5V 4 Canais
Fonte: obtido do *website* FILIPEFLOP[35].

TABELA 4.13: Características do Módulo Relé 5V 4 Canais.

Especificações técnicas	
Tempo de resposta	$5 \sim 10\ ms$
Tensão de entrada máxima	$220V\ AC$ e $30V\ DC$
Corrente de saída máxima	$10A$
Outras especificações	
Tensão de alimentação	$5V$
Corrente típica de operação	$15 \sim 20\ mA$
Conexão	4 pinos digitais
Dimensões (C x L x H) $[mm]$	80 x 60 x 20
Peso	$100g$

4.6.2 *Buoyancy Control Unit* (BCU)

Ainda no âmbito da dissertação do ASPOF EN-MEC Dias de Paiva, pretende-se desenvolver um sistema de controlo de flutuabilidade baseada na utilização de bexigas artificiais. Através da entrada e saída de água de uma seringa, o veículo aumenta e diminui de peso, o que por sua vez fará com que o veículo imirja ou emirja. Estas duas bexigas serão colocados no centro do veículo, uma em cada bordo.

[35] *Website* da FILIPEFLOP: https://www.filipeflop.com/produto/modulo-rele-5v-4-canais/.

O desenvolvimento do módulo BCU foi dividido em duas partes distintas: o projeto do módulo e o projeto do hardware. No projeto do módulo, o ASPOF EN-MEC Dias de Paiva, relatado na sua dissertação, calculou a potência necessária que deveria ser fornecida para expulsar uma determinada quantidade de água a uma pressão máxima (pressão sentida à profundidade máxima do veículo), criou toda a estrutura mecânica e escolheu o motor passo a passo (*stepper motor*) que mais se adequava ao estudo elaborado. No projeto do hardware, relatado na presente dissertação de mestrado, foi idealizado o hardware necessária para integração dos motores passo a passo no veículo, por forma a ser possível o controlo da flutuabilidade pelo Micro PC.

O motor passo a passo selecionado foi o *LA421S14-B-TJCA* da *Nanotec*, figura 4.15 que requer uma corrente de $1.4A$ por fase.

FIGURA 4.15: Motor passo a passo *LA421S14-B-TJCA*.
Fonte: obtido do *website* *Nanotec*[36].

Por forma a ser possível a implementação do motor sem fim, foi necessária a aquisição de um driver para o motor passo a passo (em inglês, *stepper motor driver*). O driver escolhido foi o *MP6500* com controlo de corrente por um potenciómetro da *Pololu*, figura 4.16. A distribuidora oficial da marca em Portugal é a *PT Robotics*. As características deste driver podem ser consultadas na tabela 4.14.

FIGURA 4.16: *Driver* do motor passo a passo *MP6500*.
Fonte: obtido do *website* *Pololu*[37].

[36] *Website* da *Nanotec*: https://en.nanotec.com/products/2412-la421s14-b-tjca/.
[37] *Website* da *Pololu*: https://www.pololu.com/product/2966.

Tabela 4.14: Características do *driver* do motor passo a passo *MP6500*.

Especificações técnicas	
Tempo de resposta	$1\mu s$
Tipos de resolução de passo	1, 1/2, 1/4 e 1/8 do passo do motor
Tensão de entrada máxima	$220V$ AC e $30V$ DC
Pico de corrente por fase	$2.5A$
Corrente contínua máxima por fase	$1.5A$
Outras especificações	
Tensão lógica de funcionamento	3.3 e $5V$
Tensão de entrada	4.5 a $35V$
Corrente consumida sem ativação	$1 \sim 5$ mA
Tipo de controlo do limite de corrente	Potenciómetro
Conexão	3 pinos digitais (mínima configuração)
Sistemas de proteção	Temperaturas elevadas, curto-circuito, baixa tensão, baixa corrente
Dimensões (C x L x H) $[mm]$	9.8 x 4.5 x 0.8
Peso	$1.4g$

4.7 Módulo de potência

A escolha das baterias assentou na carga necessária para que o veículo pudesse cumprir os requisitos de autonomia de 2 h (0.5 m/s) e a tensão requerida pelos diversos equipamentos integrados no BUV. Desta forma e para se alimentar todo o sistema elétrico com a devida tensão e corrente, foi necessário fazer-se o levantamento energético do veículo (tabela 4.15), isto é a tensão, corrente e potência de alimentação de todo o hardware. O consumo dos motores depende da sua utilização e da potência mecânica que lhes é pedida.

TABELA 4.15: Levantamento energético do hardware.

	Tensão de Alimentação	**Corrente de Alimentação Máxima**	**Potência Consumida Típica**
Jetson Nano	$5V$	$2.5A$	$5-10W$
Arduino Mega	$5V$ (pino *Vin**)	$80mA$	$1W$
Módulo SIM800L	$5V$	$1A$	$5W$
WiFi Bullet2HP	$5-24V\,PoE$	$1.4A$	$7W$
Bar30	$2.5-5V$	$1.25mA$	$6.25mW$
AHRS *VN-100T*	$5V$	$44mA$	$220mW$
U-Blox NEO-7M	$3.5-5V$	$3mA$	$15mW$
Ping 1D	$5V$	$100mA$	$500mW$
SOS Leak Sensor	$3.5-5V$	$20mA$	$100mW$
MP6500	$4.5-35V$	$1.5A$	N/D
Motor passo a passo	$4.5-24V$	$1.4A$	N/D
ArduCam 8 MP Sony IMX219	$5V$	$0.4A$	$2W$
Servos *JX BLS-12V7146*	$12V$	$2A$	$24W$

*Passa pelo regulador interno do *Arduino*.

Com esta tabela conclui-se que a tensão do sistema deverá ser superior a $12V$ ($+2V$ de margem da tensão mais elevada dos sensores), a corrente máxima deverá ser superior a $3.5A$ ($2.5A$ (valor máximo de corrente) + $1A$ (margem para pico de corrente)) e uma potência de consumo teórico total de $210W$.

4.7.1 Baterias

Com base na tabela 4.15, e em parceria com o ASPOF EN-MEC Dias de Paiva, foram escolhidas as baterias *TATTU Bateria Lipo 4S1P 15C* da *Gens ace &Tattu* de $14.8V$ com capacidade de $16000mAh$, presente na figura 4.17. Cada bateria é constituída por um pacote de quatro células de $3.7V$ ligadas em série. Na sua dissertação, o ASPOF EN-MEC calculou a potência teoricamente consumida pelo

hardware em durante duas horas, chegando à conclusão que para as baterias fornecerem a energia necessária ao sistema durante duas horas deveriam de ser utilizadas duas baterias de $16000mAh$ ligadas em paralelo.

A empresa portuguesa HP Modelismo, especializada em comercialização e reparação de artigos de rádio modelismo, aconselhou que a escolha da tensão das baterias deveriam ser superiores a $2V$ da tensão máxima necessária para o veículo ($12V$) para que se utilizasse a total capacidade elétrica das baterias. Desta forma, foi escolhida a tensão de 14.8 para as baterias. A marca recomenda que não se deixe a tensão de cada célula alcançar os $3V$[38], o recomendado é que a descarga de cada célula pare nos $3.5V$, correspondendo à tensão de $14V$ da bateria.

Através da equação 4.1, e substituindo o valor taxa máxima de descarga da bateria que é $30C$, obteve-se a corrente máxima que pode ser drenada da bateria.

$$\begin{aligned} Corrente\ máxima\ drenada(A) &= c\_bat \times tax\_desc \\ &= 16 \times 30 = 480A \end{aligned} \tag{4.1}$$

Onde a *c_bat* representa a capacidade da bateria (em Ah) e a *tax_desc* representa a taxa de descarga máxima (em C).

A corrente máxima de descarga da bateria é de $480A$, que é um valor muito acima da corrente típica e esperada que será consumida pelo veículo $18.4A$ (somatório da corrente de alimentação máxima de todos os equipamentos do veículo).

Como a taxa de carregamento máxima é de $1C$, substituindo os valores na equação 4.2, obteve-se a corrente de carregamento máxima.

$$\begin{aligned} Corrente\ carregamento\ máxima(A) &= c\_bat \times tax\_max\_carr \\ &= 16 \times 1 = 16A \end{aligned} \tag{4.2}$$

Onde a *c_bat* representa a capacidade da bateria (em Ah), a *tax_carr* representa a taxa de carregamento máxima (em C).

A corrente de carregamento máxima para estas baterias é de $16A$, isto é, as baterias apenas poderão ser carregadas a uma corrente máxima de alimentação de $16A$.

[38] Disponível nos *websites*: https://www.genstattu.com/bw e https://www.genstattu.com/tattu-16000mah-4s1p-15c-lipo-battery-pack.html.

O carregador de baterias adquirido, por recomendação da *HP Modelismo*, foi o *SKYRC Ultimate Duo 400W 20A AC/DC*.

FIGURA 4.17: Bateria *TATTU 4S1P 15C*.
Fonte: obtido do *website* HP Modelismo[39].

4.7.2 Distribuição de energia

O sistema de distribuição de energia deverá regular a tensão das baterias ($14.8V$) para a tensão dos diferentes equipamentos elétricos, assegurar a monitorização da corrente elétrica nas ligações críticas da instalação elétrica e ainda assegurar a monitorização da carga das baterias.

Com base na tabela 4.15, chegou-se à conclusão que seria necessário regular a tensão das baterias para duas tensões diferentes: $5V$ e $12V$.

Atualmente, no mercado, existem diversos reguladores de tensão para $5V$ com rendimento energético acima de 90% com corrente de saída variável entre 0 a $5A$. Como é o caso do *Buck Converter ND1805TA 5V* da *Eletechsup*, figura 4.18a e o *Buck Step Down Converter AJ20 V7.0* , figura 4.18b.

O regulador de tensão de $12V$ utilizado foi o *DROK Mini Electric Voltage Converter 15A Module*, figura 4.18c, por fornecer uma elevada corrente, de forma, a conseguir-se alimentar dois servos no mesmo circuito.

[39] *Website* da HP Modelismo: https://www.hpmodelismo.com/pt/lipo/5160-bateria-lipo-gens-ace-15c-148v-16000mah-4s1p.html?search_query=16000&results=6.

(A) *Buck Converter ND1805TA 5V.* (B) *Buck Step Down Converter AJ20 V7.0.* (C) *DROK Mini Electric Voltage Converter 15A Module.*

FIGURA 4.18: Reguladores de tensão utilizados..
Fonte: obtidos do *website* da *Banggood*[40] e da *Amazon*[41].

Na tabela 4.16 constam as características destes três reguladores de tensão utilizados no projeto.

TABELA 4.16: Características dos reguladores de tensão utilizados.

	ND1805TA 5V	***AJ20 V7.0***	***DROK Mini 15A***
Tensão de entrada $[V]$	$6 - 18$	$4.5 - 24$	$4 - 32$
Tensão de saída $[V]$	5	$0.8 - 12$	$1.2 - 32$
Corrente máxima de saída $[A]$	5	3	15
Corrente continua de saída $[A]$	4	2.1	10
Corrente estática $[mA]$	$16 - 18$	< 45	N/D
Eficiência energética	Até 93%	Até 97.5% ($5V$ a $0.7A$)	Até 98%
Proteção curto-circuito	Sim	Sim	Sim
Dimensões (C x L x H) $[mm]$	27 x 18 x 5	17 x 10 x 2	60 x 51 x 22

A figura 4.19 representa a distribuição de energia de todo o sistema, desde as baterias até aos sensores/atuadores, implementando os reguladores de tensão. A lógica da organização das ligações dos equipamentos aos conversores está relacionada com a organização do veículo por módulos e com a gestão do consumo de corrente. Desta forma são necessários dois conversores de $4A$ (menor eficência que o de $2.1A$) uma vez que a *Jetson Nano* requer uma corrente de alimentação superior a $2.1A$ e o modem WiFi que poderá necessitar de mais corrente que $2.1A$; dois conversores

[41] *Website* da *Banggood*: https://pt.banggood.com/Mini-5A-DC-DC-Converter-Step-Down-Module-Voltage-Regulator-Buck-Board-4_5V-18V-to-3V-3_3V-3_7V-4_2V-5V-6V-p-1626234.html e https://pt.banggood.com/5pcs-4_5V-24V-To-0_8V-12V-DC-DC-Buck-Step-Down-Converter-Adjustable-Power-Module-p-1644957.html.

[41] *Website* da *Amazon*: https://us.amazon.com/Converter-DROK-Adjustable-Step-down-Transformer/dp/B00C9UUFHC?_encoding=UTF8&psc=1.

de 2.1A; dois conversores de 12V porque apenas um provavelmente não aguentaria a potência solicitada pelos servos; e dois *stepper motor drivers*, uma vez que cada motor necessita de 1.4A por fase.

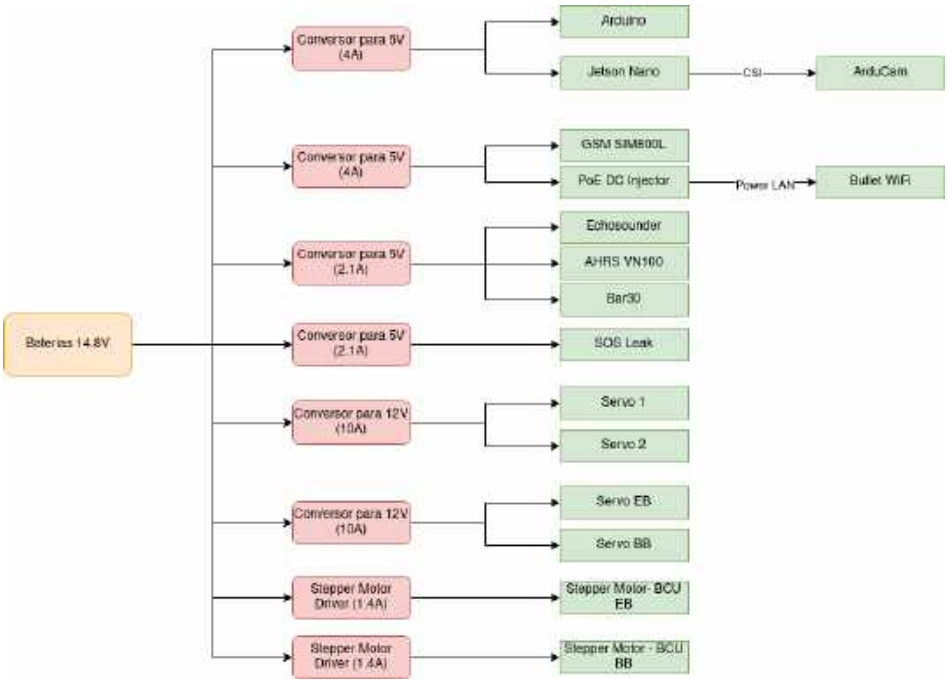


FIGURA 4.19: Diagrama da distribuição de energia do veículo.

4.7.3 Medição de corrente

O sistema de medição de corrente elétrica utilizará quatro sensores de medição de corrente *ACS712 ELC 30A*, figura 4.20, que suporta uma corrente máxima de 30A e é alimentado com uma tensão de 5V. Este sistema servirá para medir a corrente nas ligações críticas da instalação elétrica, isto é, deve medir constantemente a corrente de entrada para cada servo (quatro - dois da cauda e dois das barbatanas laterais) e para cada motor passo a passo (dois). Caso haja um aumento significativo do valor de corrente, o sistema deverá efetuar adotar um determinado movimento de emergência, como por exemplo voltar à superfície. Este sistema pode, posteriormente, ser útil para cálculos de rendimento de energia, isto é, sabendo o consumo dos motores, mais tarde, os controladores da cauda e barbatanas poderão ser desenhados de forma a otimizar a eficiência energética da locomoção do veículo.

FIGURA 4.20: Sensor de medição corrente *ACS712 30A*.
Fonte: obtido do *website* Mixtrónica[42].

4.7.4 Medição de tensão

O sistema de medição da tensão das baterias utilizará um sensor de medição de tensão *B25*, figura 4.21, que é capaz de medir a uma tensão de entrada até $25V$ e é alimentado com uma tensão de $5V$. O princípio de funcionamento do sensor é baseado em divisores de tensão até cinco vezes o valor da alimentação. Isto é, se forem utilizados $5V$ como VCC (tensão de alimentação positiva) a tensão de leitura máxima suportada é $5 \times 5 = 25V$, mas se forem utilizados $3.3V$ como VCC a tensão de leitura máxima suportada é $3.3 \times 5 = 16.5V$.

FIGURA 4.21: Sensor de medição de tensão *B25*.
Fonte: obtido do *website* *ElectroFun*[43].

4.8 Conexão do hardware

Foi também necessário efetuar-se um levantamento das conexões necessárias para ser possível determinar se a *Jetson Nano* teria disponível todas as portas/conexões necessárias para cada equipamento. Com base levantamento na tabela 4.17 serão necessárias: quatro conexões UART, uma CSI-2, uma I^2C e duas USB.

[42] *Website* da Mixtrónica: https://mixtronica.com/corrente-sensores/29818-mxacs712-30a-sensor-corrente-acs712-30a.html.
[43] *Website* da *ElectroFun*: https://www.electrofun.pt/sensores-arduino/modulo-sensor-tensao-b25.

TABELA 4.17: Levantamento das conexões dos sensores.

	Conexões
Arduino Mega	USB ou I^2C
Módulo SIM800L	UART
WiFi Bullet2HP	*Ethernet*
Bar30 Depht/Pressure Sensor	I^2C
AHRS *VN-100T*	RS232 ou UART
U-Blox NEO-7M	UART ou USB
Ping Sonar Altimeter and Echosounder	UART
SOS Leak Sensor	1 pino digital
Ping360 Scanning Imaging Sonar	USB, *ethernet* ou RS485
ArduCam 8 MP Sony IMX219	CSI-2

A *Jetson Nano* é caracterizada por possuir diversos pinos de conexão agrupados em *expansion headers*, como se pode observar na figura 4.22. O *J41* é constituído por quarenta pinos, onde quatro são de energia não comutáveis (dois de $3.3V$ e dois de $5V$) e os restantes são de interface de nível de sinal de $3.3V$ (disposição e função de cada pino no anexo I). Por padrão, todos os pinos de interface são configurados como *General Purpose Input/Output*s (GPIOs), exceto os pinos 3 e 5 e os pinos 27 e 28, que são I^2C *Serial Data Line* (SDA) e *Serial Clock Line* (SCL), e os pinos 8 e 10, que são UART *Transmitter* (TX) e *Receiver* (RX). O *Linux for Tegra* (L4T)[44] disponibiliza uma biblioteca em *Python*, *Jetson.GPIO*, para controlar GPIOs. No apêndice A pode ser consultado como foi instalada esta biblioteca.

[44]O L4T, ou *NVIDIA® Jetson ™ Linux Driver Package*, é uma distribuição de software de sistema baseada em *GNU's Not Unix!* (GNU)/*Linux* da *Nvidia* para a série de processadores *Nvidia Tegra* utilizados nas séries de placas *Nvidia Jetson*.

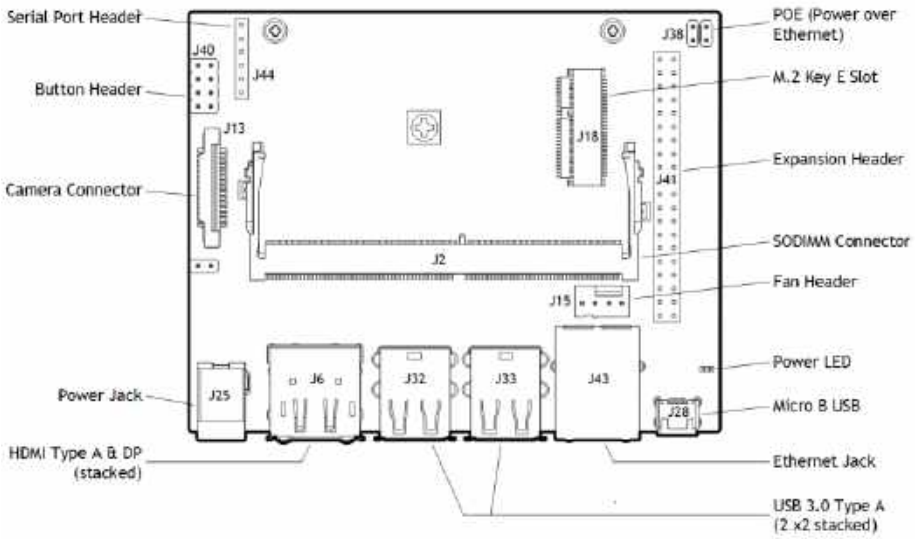


FIGURA 4.22: *Expansion headers* da *Jetson Nano Developer Kit*.
Fonte: retirado de NVIDIA (2019a).

A *jetson* apenas possui duas portas UART possíveis de serem ligadas, no entanto, para a implementação do hardware são necessárias quatro portas. Foi criado um tópico no fórum de desenvolvimento da *NVIDIA* (https://forums.developer.nvidia.com/t/uart-communication-i-need-four-uarts-in-jetson/117541/8) para descobrir qual era a melhor solução para o problema. Através de vários contributos e pesquisa optou-se por utilizar um *hub 4PORT TTL Adapter* da *YIDAMA-TONGLE* com o *chip FTDI FT4232H* que converte quatro portas UART em USB (figura 4.23). Este equipamento permite utilizar sinais digitais de $3.3V$ e $5V$ e é disponibilizado suporte para o protocolo *RS232*. Apenas é distribuído pelo *AliExpress* e *Amazon*.

FIGURA 4.23: *Hub 4 Port TTL Adapter*.
Fonte: obtido do *website* *AliExpress*[45].

[45] *Website* do *AliExpress*: https://pt.aliexpress.com/i/32994110442.html.

4.9 Resumo

Na tabela 4.18, pode ser consultado um resumo de todo o hardware escolhido para capacitar o protótipo aos requisitos definidos. Como referido no capítulo anterior, neste primeiro protótipo, optou-se por deixar de fora a integração de um modem acústico e de um sonar.

Tabela 4.18: Resumo do hardware.

Módulo de hardware	**Tipo de equipamento**	**Equipamento**	**Empresa produtora**
Micro PC	PC principal	*Jetson Nano Development Kit*	*NVIDIA*
Comunicações	GSM	*SIM800L GSM-GPRS 5V V2.0*	*RoHS*
Comunicações	Modem Acústico	*Micromodem*	WHOI
Comunicações	WiFi	*Bullet BM2HP*	*Ubiquiti*
Navegação	Sensor de pressão	*Bar30*	*BlueRobotics*
Navegação	AHRS	*VN-100T Development Kit*	*VectorNav*
Navegação	GNSS	*NEO-7M*	*Ublox*
Navegação	Eco-sonda	*Ping Sonar Altimeter and Echosounder (Ping1D)*	*BlueRobotics*
LA	*Sensor de alagamento*	*SOS Leak Sensor*	*BlueRobotics*
RECON	*Sonar*	*Ping360 Scanning Imaging Sonar*	*BlueRobotics*
RECON	*Câmara*	*8mP Sony IMX219*	*Arducam*
Locomoção	*Microcontrolador*	*Mega 2560*	*Arduino*
Locomoção	*BCU Stepper Driver*	*MP6500*	*Pololu*
Locomoção	*Módulo de relés*	*Módulo Relé 5V 4 Canais*	*Hong Wei*
Potência	*Sensor de medição de corrente*	*ACS712T*	*MicroSystems*
Potência	*Sensor de medição de tensão*	*B25*	*Eieship*

Nos capítulos seguintes será explicado como será realizada a integração deste equipamento no veículo.

Capítulo 5

Projeto da arquitetura de software

O projeto da arquitetura de software tem como objetivo habilitar um veículo autónomo a operar no seu ambiente utilizando os seus recursos físicos e computacionais. Deve assegurar o cumprimento das tarefas do robô de forma estável e robusta, a flexibilidade e modularidade para lidar com os vários tipos de tarefas, bem como a possibilidade de alteração ou adição de novos módulos de hardware ou software. Segundo Medeiros (1998) as características fundamentais de uma arquitetura de software para veículos autónomos são:

- Reatividade: o sistema robótico deve ser reativo para conseguir reagir a mudanças súbitas do ambiente e também deve ser capaz de levar em conta eventos externos, de modo a executar de forma correta, segura e eficiente as suas tarefas.
- Comportamento inteligente: as reações do sistema robótico aos estímulos externos devem ser guiadas pelos objetivos da sua tarefa principal;
- Integração de múltiplos sensores: a limitação de precisão, confiabilidade e aplicabilidade dos sensores individuais pode ser compensada pela integração de vários sensores complementares;
- Resolução de múltiplos objetivos: no caso dos sistemas robóticos autónomos ou semi-autónomos exitem situações que requerem ações concorrentes conflituosas. Uma arquitetura de software deve prover de meios para o cumprimento desses múltiplos;
- Robustez: o robô deve manipular leituras imperfeitas, eventos inesperados e falhas súbitas;
- Confiabilidade: é a habilidade de operar sem falhas ou degradação de desempenho num certo período de tempo;

- Modularidade: é a habilidade de dividir toda a arquitetura de software em módulos que podem ser separados e incrementalmente desenvolvidos, implantados e mantidos;
- Programabilidade: um sistema robótico utilizável deve ser capaz de realizar múltiplas tarefas que sejam descritas num certo nível de abstração;
- Flexibilidade: sistemas robóticos experimentais requerem mudanças contínuas durante a fase de desenvolvimento. Consequentemente, são requeridas estruturas de controlo flexíveis, de modo a permitir o desenvolvimento da arquitetura de software;
- Capacidade de expansão: é necessário algum tempo para desenvolver, construir e testar os componentes individuais de um sistema robótico. Consequentemente, uma arquitetura expansível é desejável de modo a ser possível a construção incremental do sistema;
- Adaptabilidade: como o estado do mundo tecnológico altera rápida e imprevisívelmente, o sistema robótico deve ser capaz de se adaptar ao ambiente, alternando entre diferentes estratégias de controlo;
- Raciocínio global: é necessário um agente tomador de decisões global, responsável pelo entendimento da situação geral, para se corrigir erros introduzidos por interpretações erróneas de dados sensoriais e também para fundir as informações.

5.1 Conceitos

Para o entendimento desta dissertação de mestrado e para a fase de implementação da arquitetura torna-se necessário a definição de alguns conceitos sobre a arquitetura robótica.

5.1.1 Arquitetura robótica

Um robô é uma máquina composta por sensores, atuadores e uma unidade de computação. O cérebro do robô será a unidade computacional, que pode ser um micro PC ou um micro-controlador. A tomada de decisão e as ações do robô dependem completamente do programa que o cérebro da máquina executa (Joseph, 2018). A construção de todas as sinapses entre o cérebro da máquina e todos os membros e sistemas do seu corpo (sensores e atuadores) designa-se por arquitetura robótica.

Existem muitas abordagens na literatura sobre a arquitetura robótica como as apresentadas, por exemplo, por R. Brooks (1985), Arkin (1998) e Nourbakhsh e Siegwart (2004). A arquitetura robótica é uma área do conhecimento dedicada ao projeto de *robots* altamente especializados por meio de integração de blocos de construção de software. A arquitetura robótica é a abstração do sistema de controlo e o sistema de controle é a própria realização da arquitetura. A organização da arquitetura pode ser horizontal, na qual as tarefas do sistema de controlo são divididas em várias sub-tarefas baseadas nas suas funcionalidades, vertical, nas quais a divisão das tarefas é dividida em camadas de abstração de tarefas, e híbrida, na qual as tarefas são integras em componentes verticais e horizontais.

De acordo com Siegwart, Nourbakhsh e Scaramuzza (2011), os componentes básicos da arquitetura robótica podem ser classificados em três grupos distintos:

- **Perceção**: envolve as atividades de interpretação e integração dos sensores;
- **Planeamento**: envolve o planeamento das tarefas, a sincronização e a monitorização da execução de todas as atividades do veículo;
- **Atuação**: envolve as atividades de execução dos movimentos, ações do robô e o controlo dos atuadores.

Para os UUVs, no «Standard Guide for UUV Autonomy and Control» (2006), são considerados três componentes na aferição das necessidades da autonomia para as missões: consciência da situação (*situation awareness*), tomada de decisão, planeamento e controlo (*decision-making, planning, and control*) e interação externa (*external interaction*). Estes três componentes são ortogonais e inter-relacionados, pelo que abrangem o "espaço de autonomia" do UUV. Existem várias escalas que descrevem os níveis de autonomia de um UUV. Varia desde a operação remota até aos sistemas totalmente autónomos (Huang, 2008; «Standard Guide for UUV Autonomy and Control», 2006).

Paradigmas

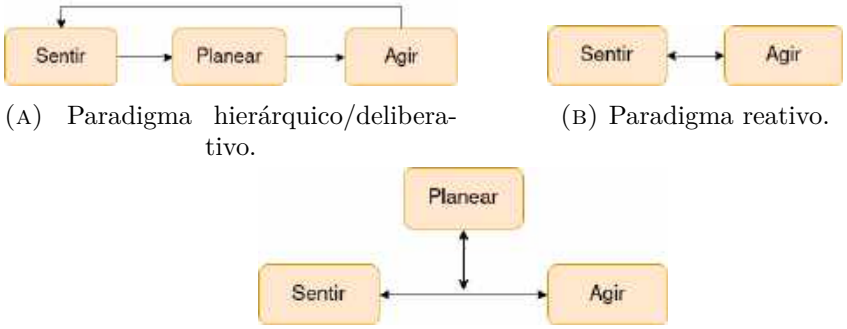
Na arquitetura robótica existem três tipos de paradigmas[46]: paradigma hierárquico/deliberativo, paradigma reativo e o paradigma híbrido deliberativo/reativo (Murphy, 2000). Não existe uma única solução para cada tipo de problema mas sim, melhores ou piores soluções para o mesmo problema.

[46]Conjunto de suposições e/ou técnicas que caracterizam uma abordagem para uma classe de problemas genéricos (Murphy, 2000)

O paradigma hierárquico/deliberativo é o mais antigo e consiste num modelo centralizado que recolhe informações do meio ambiente utilizando os sensores, cria um modelo virtual desse ambiente (define-se mundo), planeia o próximo movimento e executa a ação. É um sistema do tipo *Sense, Plan and Act* (SPA) ou *top-down* (figura 5.1a). A abordagem deliberativa é limitada a ambientes previsíveis porque os agentes atuam com pouca autonomia e precisam de modelos simbólicos explícitos dos seus ambientes. É utilizado nas arquiteturas dos modelos *Albus* (Albus, 1991), *NASA/NBS Standard Reference Model* (NASREM) (Albus et al., 1989), o *Intelligent Mobile Robot System* (IMRS) (Saridis & Valavanis, 1988; F. Y. Wang et al., 1991), e o *MARIUS (AUV)* (Fryxell et al., 1996).

Contrariamente o paradigma reativo não emprega o modelo do mundo, e minimiza a informação sobre os seus estados internos e sobre o ambiente, isto é, organiza a inteligência em camadas verticais onde o sistema é programado para agir através da ativação de uma coleção de comportamentos primitivos de baixo nível (figura 5.1b). Existe uma ligação robusta entre a perceção do ambiente (sensores) e as ações (atuadores). É utilizado nas arquiteturas dos modelos esquema motor (Arbib, 1992) e subsunção (R. a. Brooks, 1987).

O paradigma híbrido deliberativo/reativo pode ser descrito na forma de planear, sentir-agir. Isto é, o robô primeiramente planeia como irá cumprir a missão e depois executa diferentes comportamentos (sentir-agir) colecionados num conjunto pré-definido para executar o planeamento (figura 5.1c). Procura fundir as vantagens da IA dos robôs deliberativos com a simplicidade dos robôs reativos. Atualmente, os robôs com arquitetura híbrida predominam uma vez que, em muitas das aplicações robóticas mais complexas, as formas de conhecimento do mundo na arquitetura robótica permitem que a navegação do robô seja mais flexível, eficiente e geral. Algumas arquiteturas híbridas são a *Autonomous Robot Architecture* (AuRA) (Arkin, Riseman & Hanson, 1988), *AUV Phoenix* (Healey, Marco & McGhee, 1996), *Coupled Layer Architecture for Robotic Autonomy* (CLARAty) (Volpe et al., 2001), *Stanley* (Montemerlo et al., 2006), Liu, Hu e Gu (2006) (arquitetura híbrida aplicada a um BUV) e a arquitetura de três camadas (Murphy, 2000).



(A) Paradigma hierárquico/deliberativo.

(B) Paradigma reativo.

(C) Paradigma híbrido deliberativo/reativo.

FIGURA 5.1: Diagrama comparativo de paradigmas.
Fonte: adaptado de Murphy (2000).

Arquitetura híbrida

Segundo Murphy (2000), apesar das arquiteturas híbridas variarem na forma como a função deliberativa atua no sistema, os componentes implementados são semelhantes, sendo eles:

- **Organizador de sequências (ou *sequencer*)**: agente que gera um conjunto de comportamentos a serem adotados para realizar uma determinada subtarefa e determina quaisquer sequência e condições de ativação;
- **Gestor de recursos (ou *resource manager*)**: agente que aloca os recursos aos comportamentos, isto é, está diretamente ligado à parte sensorial do robô e que deve verificar qual é o sensor mais adequado para cada situação/missão do robô;
- **Cartógrafo (ou *cartographer*)**: agente responsável por criar, armazenar e atualizar o modelo do mundo virtual do veículo;
- **Planeamento de missão (ou *mission planner*)**: agente que interage com o utilizador, traduz possíveis comandos do utilizador para a linguagem do robô e constrói o planeamento da missão;
- **Monitorização de desempenho e solução problemas (ou *performance monitoring and problem solving*)**: agente que permite que o robô possa perceber se está a fazer progressos ao longo da tarefa.

O modelo da arquitetura híbrida de três camadas é a mais utilizada para robôs de grande complexidade como os AUV (Healey, Marco & McGhee, 1996; Kortenkamp & Simmons, 2008; Liu, Hu & Gu, 2006; Murphy, 2000; Ridao et al., 2000).

É organizada em três camadas baseadas no estado de conhecimento: planeamento, executivo e funcional.

Utilizando a definição de Murphy (2000), a camada do planeamento é composta pelos componentes planeamento de missão e cartógrafo. É responsável por gerar os objetivos e os planos estratégicos enviando-os, posteriormente, para a camada intermédia executiva. Nesta camada poderá ser integrados sistemas de cooperação entre diversos robôs (cardume, ou em inglês *swarm*) (M. Seto, Paull & Saeedi, 2013).

A camada executiva é composta pelo organizador de sequências e pela monitorização de desempenho e solução problemas, e mediante o objetivo do robô, poderá integrar o gestor de recursos. Utiliza as técnicas de planeamento reativo para selecionar um conjunto de comportamentos de uma biblioteca de comportamentos e desenvolve uma rede de tarefas (*task network*) especificando a sequência de execução para os comportamentos a um sub-objetivo particular. Isto é, é o responsável por traduzir os paneamentos de alto nível da camada de planeamento em comportamentos de baixo nível (camada funcional).

A camada funcional encontra-se no nível mais baixo da arquitetura, estando ligada diretamente aos sensores e atuadores. Esta camada é responsável por receber e tratar a informação provinda dos sensores (estimadores) e executar os comportamentos ordenados pela camada executiva (controladores). Neste nível deverá residir as teorias de controlo tradicionais, tais como, filtros de *Kalman*, controlos *Proportional–Integral–Derivative* (PID), entre outros.

A composição de cada camada da arquitetura híbrida de três camadas, pelos diversos componentes de Murphy (2000), podem ser observada na figura 5.2.

FIGURA 5.2: Camadas e componentes da arquitetura híbrida das três camadas.

5.1.2 *Middleware* e linguagens de programação

O programa que o robô irá "correr" pode ser um *firmware* a correr num micro-controlador, ou um código `C++` ou *Python* a correr num micro PC. A programação do robô é o processo de escrever um programa para o "cérebro" deste, para que, futuramente, possa receber a informação dos sensores, tomar decisões e atuar conforme desejado. Ou seja, é o programa que implementa as camadas/componentes/abstrações descritas na secção anterior. Atualmente existe uma variedade de linguagens de programação utilizadas em robôs como é o caso do `C++` , *Python, Java, C# e Wiring* (baseada em `C++` utilizada nos *Arduinos*).

Middleware

A maior parte dos veículos autónomos desenvolvidos utilizam uma camada de abstração especial que simplifica a comunicação e a gestão dos dados das entidades computacionais (robôs, computadores, sensores, atuadores, entre outros) em aplicações distribuídas. Esta camada reside entre o sistema operativo e as aplicações de software (Elkady & Sobh, 2012), como se pode ver na figura 5.3. Foi projetado para gerir a heterogeneidade do hardware, melhorar a qualidade da aplicação, simplificar o *design* do software e reduzir os custos de desenvolvimento. Ao utilizar um

middleware, apenas é necessário criar a lógica ou algoritmos dos componentes, após os quais poderão ser combinados e integrados com outros componentes existentes.

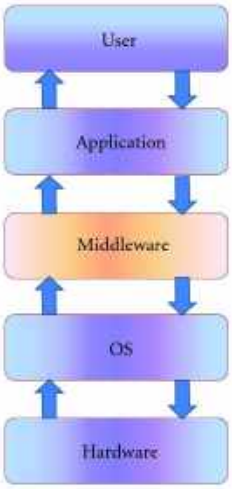


FIGURA 5.3: Camadas de *middleware*.
Fonte: retirado de Elkady e Sobh (2012).

Nos últimos anos foram desenvolvidos diversos *middlewares* por faculdades, centros de investigação, organizações e empresas para desenvolverem os seus próprios robôs. Alguns desses exemplos são: o DUNE (desenvolvido pela LSTS e utilizado nos veículos *SeaCon* e no BUV 3), *Player, Orca, Miro*, *Open Robot Control Software* (Orocos), CLARAty, *Microsoft Robotics Developer Studio* (MRDS), *Yet Another Robot Platform* (YARP), *Carnegie Mellon Navigatio* (CARMEN), *Mission Oriented Operating Suite* (MOOS) e o *NATO Standardization Agreement* (STANAG) 4586 (interface padronizada do sistema de controlo dos *Unmanned Aerial Vehicle* (UAV) para interoperabilidade de veículos da Organização do Tratado do Atlântico Norte (OTAN) (*North Atlantic Treaty Organization* (NATO))).

O ROS também é um *middleware open-source* desenvolvido por uma equipa de investigação robótica liderado por Morgan Quigly na *Stanford University* (EUA) em 2007. Mais tarde, uma *startup* de investigação robótica chamada *Willow Garage* (http://www.willowgarage.com/) assumiu o projeto e deu-lhe o nome ROS, em 2007. Este software é classificado como um *meta-operating system* (Joseph, 2018), uma vez que fornece os serviços esperados de um sistema operativo, incluindo a abstração de hardware, controlo de baixo nível, transmissão de mensagens entre processos e controlo de pacotes. Fornece ferramentas e bibliotecas para obter, criar, escrever e executar código em vários sistemas. No site do projeto (https://www.ros.org/) encontra-se vários tutoriais sobre a instalação do software e a utilização do mesmo no desenvolvimento de robôs.

Atualmente, a maior parte dos veículos desenvolvidos utilizam o ROS como sistema *middleware* (Rösmann, Hoffmann & Bertram, 2017; Vaz et al., 2018) uma vez que possui:

- Interface de transmissão de mensagens entre processos. O ROS fornece uma interface de transmissão de mensagens para a comunicação entre dois programas ou processos. Por exemplo, um sensor GNSS obtém a posição do robô, essa posição é enviada para um processo de navegação para determinar se o robô chegou à posição desejada. O processo de navegação, posteriormente, envia ordens para os motores para o veículo se deslocar para a posição;
- Recursos semelhantes ao sistema operativo;
- Suporte e ferramentas de linguagem de programação de alto nível. O ROS suporta as linguagens de programação *C ++, Python* e *Lisp*. Há suporte experimental para linguagens como *C#, Java, Node.js*, entre outras (http://wiki.ros.org/Client%20Libraries);
- Disponibilidade de bibliotecas de terceiros. A estrutura do ROS pode integrar bibliotecas de terceiros, como por exemplo, o *Open Source Computer Vision Library* (OpenCV) (https://opencv.org) que é utilizado na visão robótica e o *Point Cloud Library* (PCL) (http://pointclouds.org) que é utilizado na percepção 3D do robô;
- Algoritmos prontos para utilização. O ROS implementou algoritmos populares de robótica, como o PID (http://wiki.ros.org/pid), *Simultaneous Localization and Mapping* (SLAM) (http://wiki.ros.org/gmapping) e planeadores de caminhos como *A*, Dijkstra* (http://wiki.ros.org/global_planner) e *Adaptive Monte Carlo Localization* (AMCL) (http://wiki.ros.org/amcl). A lista de implementações de algoritmos no ROS continua. Os algoritmos disponíveis no mercado reduzem o tempo de desenvolvimento do controlo de um robô;
- Facilidade na construção de pacotes de software. Uma vantagem do ROS são os algoritmos "fora da caixa" e aliado a isso o ROS possui pacotes que podem ser facilmente reutilizados com qualquer outro robô. Por exemplo, pode criar-se um pacote rápido para um determinado robô personalizando um pacote existente disponível no repositório ROS. Os pacotes do repositório do ROS podem ser utilizados porque a maioria dos pacotes é de código aberto e reutilizável para fins comerciais e de investigação. Desta forma, reduz-se o tempo de desenvolvimento do software do robô:

- Suporte da comunidade. A principal razão para a popularidade e desenvolvimento do ROS é o apoio da comunidade. Os desenvolvedores de ROS encontram-se espalhados por todo o mundo, desenvolvendo e mantendo ativamente os pacotes ROS. O *ROS Answers* é uma plataforma para consultas relacionadas ao ROS (https://answers.ros.org/questions/). O *ROS Discourse* é um fórum online no qual os utilizadores do ROS discutem vários tópicos e publicam notícias relacionadas ao ROS (https://discourse.ros.org);
- Extensas ferramentas e simuladores. O ROS é construído com muitas ferramentas de linha de comandos e *Graphical User Interface* (GUI) para ser possível a depuração, visualização e simulação de aplicações robóticas. Essas ferramentas são muito úteis para trabalhar com um robô. Por exemplo, a ferramenta *Rviz* (http://wiki.ros.org/rviz) é usada para visualização de câmaras, scanners a laser, unidades de medida inercial, etc. Para trabalhar com simulações de *robot* , pode ser utilizado o simulador *Gazebo* (http://gazebosim.org).

Em ROS existem sete elementos fundamentais (Mahtani et al., 2018): mestre, servidor de parâmetros, nós, tópicos, serviços, mensagens e *bags*. Na enumeração abaixo, estes elementos estão hierarquicamente organizados, do mais alto para o mais baixo nível.

1. Mestre ou *master*: fornece o registo de nomes e o serviço de pesquisa para os restantes nós para além de configurar as conexões entre os nós. Num sistema distribuído, poderá executar nós noutros computadores;
2. Servidor de parâmetros ou *parameter server*: é um programa que normalmente é executado com o mestre do ROS . O utilizador pode armazenar vários parâmetros ou valores neste servidor a que todos os nós podem aceder. O utilizador também pode definir a privacidade do parâmetro. Se for um parâmetro público, todos os nós terão acesso, se for privado, apenas um nó específico poderá aceder ao parâmetro;
3. Nós ou *nodes*: são processos ou módulos de software. Para que um processo possa interagir com outros nós, é necessário criar um nó com esse processo por forma a ligá-lo à rede ROS. Os nós são gravados utilizando uma biblioteca do cliente ROS, por exemplo, *roscpp* (`C++`) ou *rospy* (*Python*);
4. Tópicos ou *topics*: são os rótulos das mensagens para que estas possam ser identificadas pela rede ROS. Quando um nó envia dados publica um tópico. Os nós podem receber tópicos de outros nós se subscreverem o tópico, mesmo que não haja outros nós a publicar esse tópico específico;

5. Serviços ou *services*: é uma função que pode ser chamada sempre que um nó cliente envia uma solicitação. O nó que cria uma chamada de serviço é chamado nó servidor e quem chama o serviço é chamado nó cliente;

6. Mensagens ou *messages*: são utilizadas para que os nós comunicarem. Contêm dados que fornecem informações para outros nós. O ROS possui muitos tipos de mensagens padronizadas, mas o programador poderá criar novos tipos, chamadas mensagens personalizadas (*custom messages*);

7. *Bags*: métodos para armazenar e reproduzir dados de mensagens do ROS. Os dados são registados num formato de ficheiro *.bag* criado pela ferramenta *rosbag*. Estes dados poderão ainda ser processados, analisados e visualizados utilizando outras ferramentas como *rqt_bag*.

`C++` e *Python*

A linguagem `C++` derivou de uma proposta de acrescento de características à linguagem *C* (desenvolvida em 1972 pelo norte americano Dennis Ritchie para o sistema operativo *Unix*), pelo matemático cientista de computação dinamarquês Bjarne Stroustrup, para que se tornasse algumas tarefas de programação mais simples. Esta primeira adaptação da linguagem, em 1979, foi designada por *C with Classes*. Mais tarde, em 1983, após sucessivos aperfeiçoamentos, evoluiu de um "mero rebento conatural da *C* para uma linguagem jovem e emancipada, com identidade e compilador próprios" apelidada de `C++` (Loureiro, 2019). Esta linguagem é classificada como uma linguagem de programação orientada a objetos (*Object-Oriented Programming* (OOP)) puramente compilada [47] e de código aberto gerida pela *Standard C++ Foundation* ou muitas vezes também designada por *Standard CPP Foundation* (https://isocpp.org/).

A linguagem *Python* é uma linguagem de programação de alto nível, interpretada [48] e orientada a objetos. O *Python* foi criado, em 1989, por Guido van Rossum e o lançada em 1990. É um software de código aberto gerido pela *Python Software Foundation*, empresa sem fins lucrativos (www.python.org/). A principal

[47] Linguagem compilada: linguagem que em que o código fonte é executado diretamente no SO ou pelo processador, após ser traduzido, por um compilador, para uma linguagem de baixo nível, como a linguagem de montagem ou código de máquina. Exemplos desta linguagem são `C++`, *C, Pascal, Delphi, Visual Basic* e *Java*.

[48] Linguagem interpretada: é uma linguagem de programação em que o código fonte é executado por um programa de computador chamado interpretador, que é executado pelo SO ou processador. Mesmo que uma fonte de código passe pelo processo de compilação, a linguagem pode ser considerada interpretada se o programa resultante não for executado diretamente pelo SO. Exemplos desta linguagem são o *JavaScript, PHP, R, Ruby* e *Wolfram Language*.

filosofia de design do *Python* é a legibilidade do código e sintaxe, que permite aos programadores expressarem os seus conceitos em menos linhas de código. Em aplicações robóticas, o *Python* é geralmente utilizado quando existe pouca necessidade computação, como por exemplo gravar dados num dispositivo usando protocolos de comunicação série, registar dados de um sensor, criar uma interface de utilizador, etc (Joseph, 2018).

O `C++` e *Python* são as linguagens mais comuns utilizadas em programação robótica (Joseph, 2018). Segundo Joseph (2018) se a preferência do programador for o desempenho, deverá ser utilizado o `C++`, executa uma aplicação mais rapidamente utilizando menos recursos de computação, mas se a prioridade for a facilidade na programação, deverá ser utilizado *Python*, a construção da aplicação é mais rápida mas pode utilizar mais recursos de computação. Basicamente, a escolha da linguagem de programação para aplicação robótica é debruçada na decisão entre o desempenho e o tempo de desenvolvimento.

5.2 Arquitetura robótica implementada

Foi escolhido o paradigma híbrido uma vez que o BUV proposto necessita de aliar o comportamento reativo da rede de sensores-atuadores com as atividades deliberativas necessárias para realizar tarefas mais complexas.

Foi tido como base do projeto da arquitetura robótica híbrida o modelo das três camadas. No entanto, fez-se algumas alterações na organização dos componentes principais da arquitetura. Foram adicionados alguns componentes presentes em outras arquiteturas (Gallimore, Stokey & Terrill, 2018; Huang, 2008; Pinto et al., 2013; Siegwart, Nourbakhsh & Scaramuzza, 2011):

- Camada de planeamento:
  - Supervisor do veículo (ou *vehicle supervisor*): responsável por monitorizar o estado do veículo, isto é, o nível de baterias, os sensores de LA e os níveis de corrente das ligações críticas. Verifica também se os restantes sensores estão operacionais (GPS, IMU, comunicações, etc). Este elemento, se verificar que as condições de emergência foram estabelecidas, deverá comunicar, diretamente, com o piloto para emergir o veículo. Faz parte deste elemento o *Vehicle Supervisor Node*;

  - Controlador de missão (ou *mission controller*): responsável pela interface com o operador, isto é, faz o acompanhamento e o *feedback* das missões ao operador e faz o *upload* das missões criadas pelo operador;
  - Coordenador de missão (ou *mission planner*): responsável por armazenar o planeamento de missões com data e hora para a sua execução e, quando chegar à hora da realização da missão, ordenar à camada executiva a execução da missão;
  - Cartógrafo (ou *cartographer*): responsável por criar, atualizar e fazer a manutenção dos mapas ou modelos do mundo;
  - Sistema de cooperação multi-BUV's (ou *multi-BUV cooperation*): responsável pelo interface entre os diversos veículos que farão parte de uma determinada missão.

- Camada executiva:
  - Navegador (ou *navigation planner*): responsável por gerar o planeamento de navegação para a missão com base na informação disponibilizada pelo coordenador de missão, como por exemplo *waypoints*, tempo entre cada *waypoint*, velocidade máxima de missão, entre outros;
  - Piloto (ou *pilot*): é o organizador de sequências responsável por gerar um conjunto de comportamentos a serem adotados para realizar uma determinada sub-tarefa e determina quaisquer sequência e condições de ativação. Recebe a informação da missão pelo coordenador de planeamento, o planeamento de navegação pelo navegador e o modelo do mundo pelo cartógrafo. Com base nestes dados executa um determina comportamento, calculando os *setpoints* e enviando-los aos controladores dos atuadores. Em caso de emergência executa um comportamento predefinido hierarquicamente superior, isto é, deixa de seguir as ordens da camada de planeamento;
  - Gestor de recursos (ou *resource manager*): responsável por alocar os recursos aos comportamentos, isto é, está diretamente ligado à parte sensorial de RECON do BUV e que deve verificar qual é o sensor mais adequado para cada situação/missão do robô;
  - Monitorização de execução (ou *performing monitoring*): responsável pela monitorização de todas as tarefas que estão a ser executadas no BUV e,

em caso de falha, reúne o *log* do erro. Abaixo da sua hierarquia existe o Este componente está em sincronia com o controlador de missão.

- Camada funcional:
  - Sensores (ou *sensors*): *drivers* e tarefas associadas a cada sensor responsáveis pelas atividades de interpretação e integração dos sensores. Fazem parte deste grupo o *Bar30 Node*, *AHRS Node*, *Ublox GPS Node* e o *Echosounder Node*;
  - Atuadores (ou *actuators*): *drivers* de hardware desenvolvidos em `C++` e controlados por *nodes* que permitem que o veículo se desloque no meio aquático. Fazem parte deste grupo o BCU *Node*, *Lateral Fin Node* e o *Fin Node*;
  - Comunicadores (ou *comms*): *drivers* e tarefas associadas a cada módulo de comunicação, responsáveis pelo envio e receção de mensagens entre o veículo e o operador. Fazem parte deste grupo o *GSM Node, Acustic Node* e o *WiFi Node*;
  - Estimadores (ou *estimators*): responsáveis pelo tratamento da informação proveniente dos sensores, aplicando os filtros necessários para obtenção de todas as variáveis do sistema (filtro de *Kalman*). Faz parte deste grupo o *Localization Node*;
  - Controladores (ou *controllers*): responsáveis pela tradução dos comandos provenientes da camada executiva para os atuadores (controladores PID) e para os sensores de RECON. Fazem parte deste grupo o *Control Node, Sinusoidal Generator Node* e o *RECON Control Node*;
  - Interpretador (ou *interpreters*): responsável pela interpretação e tradução de mensagens. Compila a informação a ser enviada e envia a mensagem para o comunicador, selecionado pelo *gestor de recursos*, enviar para o operador. Na receção, recebe a mensagem provinda do comunicador e traduz enviando as intenções do operador ao controlador da missão.

Na figura 5.4 pode observar-se a organização da arquitetura robótica do veículo implementada. Na presente dissertação apenas foi desenvolvida e aplicada a arquitetura da camada funcional. Para o desenvolvimento de toda a camada funcional foi utilizado o ROS como sistema *middleware*. Optou-se por utilizar o `C++` para desenvolver os *drivers* de todo o hardware e os sistemas de controlo por executar uma aplicação mais rapidamente utilizando poucos recursos de computação

e o *Python* para as aplicações de RECON por ser mais rápida a construção da aplicação.

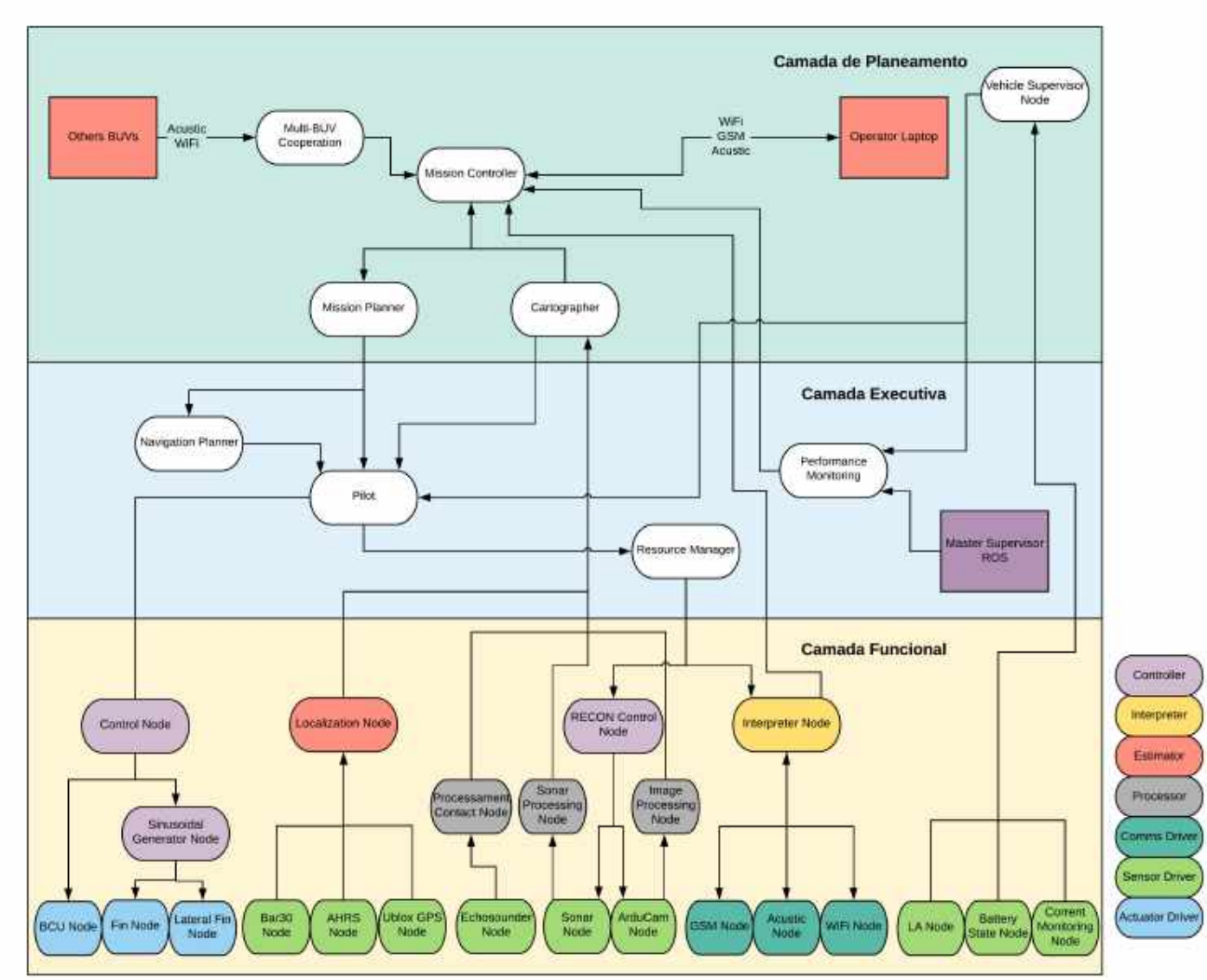


FIGURA 5.4: Arquitetura de robótica do veículo.

O estilo de codificação deve ser limpo e consistente para que seja possível a sua interpretação, depuração e manutenção. O código não deve apenas ser funcional também deve ser elegante para que possa ser reutilizado e aprimorada por outros membros do projeto. Desta forma fez-se uma análise das recomendações das melhores práticas de codificação utilizados em ROS para desenvolvimento em `C++`, que se encontra presente no *website* http://wiki.ros.org/CppStyleGuide. Optou-se por utilizar o estilo da *Google* presente no *website* https://google.github.io/styleguide/cppguide.html. Todo o código desenvolvido foi construído com este estilo.

Capítulo 6

Desenvolvimento

Neste capítulo é relatado todo o desenvolvimento da camada funcional da arquitetura proposta no capítulo anterior. A camada funcional do veículo foi desenvolvida em ROS e organizada hierarquicamente por módulos, sendo eles os drivers, estimadores, comunicadores e controladores.

Desta forma foi criado um projeto ROS da camada funcional, chamado *tobias_ros*. Este é constituído por *metapackages*, que contêm diversos módulos, constituídos por pacotes de software em ROS. Este software encontra-se disponível, à comunidade cientifica, no repositório do *GitHub* https://github.com/hilarioaraujo/tobias_ros. Através do acesso ao repositório pode-se descarregar o software bem como aceder a toda a documentação (em linguagem *Markdown*) presente nos ficheiros *README.md* dos pacotes ROS desenvolvidos.

6.1 Configuração inicial

Inicialmente foi necessário configurar a *Jetson Nano* para que fosse possível utilizá-la e para se poder implementar a *framework* ROS. No apêndice A está presente um pequeno guia que foi elaborado para a preparação da *Jetson Nano*.

6.2 *tobias_drivers*

O *metapackage tobias_drivers* é constituído pelos pacotes de ROS com as drivers de operação entre os sensores, atuadores e comunicadores. Para todos pacotes de drivers foi elaborado um ficheiro de configuração em *YAML Ain't Markup*

Language (YAML)[49], com os diversos parâmetros de configuração da ligação à *jetson*, como por exemplo as portas de conexão, *baudrate* e *timeout* da comunicação, com os diversos parâmetros de configuração do equipamento como por exemplo a frequência de trabalho, os *plugins* ativos ou desativos entre outros, e os parâmetros de *debug* (parâmetro criado para ser possível o *debug* aprofundado de cada driver). Para inicializar cada driver é utilizada a ferramenta do ROS *roslaunch* que recebe um ou mais arquivos de configuração *Extensible Markup Language* (XML) (com a extensão *.launch*) que especificam os parâmetros a serem configurados e os nós a serem iniciados, bem como as máquinas nas quais eles devem ser executados. Isto é, serve para inicializar os vários nós do ROS localmente ou via *Secure Socket Shell* (SSH) podendo carregar os vários parâmetros do ficheiro de configurações para o servidor de parâmetros. Desta forma, o utilizador poderá alterar o ficheiro de configurações e executar os pacotes dos drivers sem que seja necessária a construção dos pacotes.

O pacote ROS de cada driver é constituído por diversas pastas e ficheiros:

- Subpastas:
  - *include*: contém os *header files* com a definição de macros e classes;
  - *src*: contém os *source files* com o código que implementa as classes;
  - *launch*: contém o ficheiro *.launch* que irá inicializar o driver e carregar os parâmetros do ficheiro de configuração;
  - *config*: contém os ficheiros de configuração em formato YAML;
  - *datasheet*: contém os *datasheets* e manuais de utilização do equipamento respetivo.
- *CMakeList.txt*: ficheiro de configuração obrigatório para o *catkin build* ou *catkin_make*, que são os compiladores de pacotes ROS, por compilarem o pacote com recurso ao *Cross Plataform Make* (CMake)[50]. Neste ficheiro devem ser colocadas as dependências de bibliotecas, outros pacotes ROS (como por exemplo *roscpp* ou *rospy*), o diretório dos *header files* e *source codes*, e se existirem, os ficheiros executáveis;

[49] A YAML é uma linguagem de codificação de dados para humanos inspirada em linguagens como *XML, C, Python, Perl.* A YAML foi desenvolvido por Clark Evans em 2001 em conjunto com Ingy döt Net e Oren Ben-Kiki (*website* oficial: https://yaml.org/). É utilizada no ROS para definir os *rosparams*.

[50] O CMake é um software de código aberto multi-plataforma desenhado para construir, testar e criar pacotes de software. É utilizado para controlar o processo de compilação utilizando ficheiros simples de configuração independentes da plataforma e do compilador (https://cmake.org/).

- *package.xml*: ficheiro de manifesto do pacote ROS obrigatório que deve fazer parte da pasta raiz de qualquer pacote ROS compatível com o *catkin*. Neste ficheiro são definidas as propriedades do pacote como o nome, número da versão, autores, *maintainers* e as dependências de outros pacotes *catkin*;
- *readme.md*: ficheiro no formato *Markdown*[51] que fornece uma breve descrição do pacote ROS em inglês. No *GitHub*, o texto deste ficheiro é carregado automaticamente, mostrando a descrição formatada na respetiva pasta de cada pacote.

6.2.1 Sensor de alagamento - *SOS Leak Sensor*

Este pacote ROS é o driver para o *SOS Leak Sensor*. O sensor é constituído por quatro pontas de prova posicionadas ao longo do veiculo. Cada ponta de prova é constituída por dois pinos que em contacto com a água estabelecem um curto-circuito. Quando isto acontece o sensor envia um sinal *hight* para a *jetson nano*.

A conexão entre o sensor e o driver é realizada através de um pino GPIO ao qual o utilizar irá fazer a ligação, que por defeito é o pino 7 mas poderá ser alterado no ficheiro de configuração (parametro *pin_input*).

Com recurso à biblioteca *JetsonGPIO* (guia de instalação na secção A.3 do apêndice A), desenvolvida em `C++`, é configurado esse pino como pino de entrada digital. Através de um *timer*, onde a sua duração pode ser alterada no ficheiro de configuração (*update_duration*), o sistema verifica o estado do sensor com o determinado intervalo, e em caso positivo (valor de entrada $hight = 1$), publica no tópico *leak* verdadeiro.

O pacote *vehicle_supervisor* é o subscritor deste tópico, e em caso positivo adotará um comportamento de emergência.

6.2.2 Sensor de pressão - *Bar30*

O sensor de pressão *Bar30* incorpora o sensor de pressão *MS5837-30BA* da *TE Connectivity* que fornece um valor digital de 24 bits de temperatura e de pressão. Através destes valores possível calcular a profundidade do veículo. Este

[51] *Markdown* é uma linguagem simples de formatação de texto desenvolvida pelo John Gruber. Permite que seja convertida em vários formatos de saída, mas a ferramenta original com o mesmo nome apenas suporta HTML. É a linguagem utilziada para formatar os ficheiros *readme* no *GitHub* https://www.markdownguide.org/.

sensor utiliza o protocolo I^2C simples, isto é, sem registos internos, para comunicar com outros equipamentos.

Antes de se proceder ao desenvolvimento do driver foi estabelecida e testada a conexão entre o sensor e o micro PC, utilizando o comando $ i2cdetect -y -r 1 (o número "1" refere-se à porta um I^2C) sem sucesso. Foi consultada o fórum da comunidade *BlueRobotics* BlueRobotics sobre este problema[52], ao qual a solução seria utilizar o comando *i2cdump*. Desta forma foi testada a conexão com o comando $ i2cdump 1 0x76 onde o resultado foi positivo.

A *BlueRobotics* disponibiliza uma biblioteca em *python* para *Raspberry Pi* com interface para o sensor de pressão[53]. No entanto, como se pretendia desenvolver todos os drivers em `C++` por uma questão de optimização de software, seria necessário desenvolver uma biblioteca em `C++` capaz de comunicar com o sensor utilizando o protocolo I^2C. Foi encontrada uma biblioteca em `C++` para sistemas *Linux*[54] que, infelizmente, não estava preparada para funcionar com o sensor, por este não utilizar registos internos. Foi estabelecido o contacto com o desenvolvedor, e através de algumas reuniões por *Skype* foi desenvolvida uma biblioteca capaz de comunicar com o sensor.

O *datasheet* BlueRobotics (2020a), do sensor *MS5837-30BA*, descreve os comandos e sequências de leituras para desenvolver o driver, que foi desenvolvido em `C++` com todas as funcionalidades descritas no mesmo. Fazem parte destas funcionalidades a calibração do sensor, a leitura da pressão, temperatura da água do mar e profundidade.

No ficheiro de configuração deverá atualizar-se a densidade do fluído (parâmetro *fluid_density*, por defeito é 1029 Kg/m^3), a unidade de pressão (parâmetro *pressure_unit*, por defeito é o *mbar*) e a porta de conexão I^2C (parâmetro *i2c_bus*, que por defeito é a 1 (pinos 3 (SDA) e 5 (SCL)).

Através de um *timer*, cuja duração pode ser alterada no ficheiro de configuração (*update_duration*), é realizada a leitura do sensor e publicados os tópicos *sea_temperature*, *pressure* e *depth* com os respetivos valores da temperatura do mar (em žC), pressão (em *mbar*) e profundidade (em metros).

[52]Tópicos das respostas na página: https://discuss.bluerobotics.com/search?q=i2cdump
[53]Disponível no repositório: https://github.com/bluerobotics/ms5837-python
[54]Disponível no repositório: https://github.com/amaork/libi2c

6.2.3 Eco-sonda - *Ping1D*

A *BlueRobotics* disponibiliza um software gráfico de interface com a eco-sonda *Ping1D* apenas para sistemas de arquitetura x86 de 64 bits chamado *Ping Viewer*[55]. No entanto, o sistema da *Jetson Nano* é *arm64*, por esse motivo não foi possível testar o equipamento diretamente no Micro PC.

A ecosonda utiliza o ping-protocol[56] que é um protocolo de comunicação desenvolvido pela empresa e projetado para comunicações síncronas entre o equipamento e o computador através de uma rede mestre e escravo. O escravo (ecosonda) apenas envia os dados quando for solicitado pelo mestre, permitindo que o protocolo seja utilizado num barramento *half-duplex* como o *RS485*.

Este protocolo é disponibilizado pela empresa em `C++`[57] ou em *Python*[58]. Foi utilizada uma versão do protocolo *ping-cpp* específica para desenvolvimento de uma nova aplicação disponível no *deployment branch* do repositório *GitHub ping-cpp*.

O pacote ROS do driver do sensor foi desenvolvido com base no guia do sensor[59]. Existe uma opção descrita por *contuinous start and stop*, mas não foi implementada por uma questão de redução do fluxo de informação. Desta forma apenas é enviada a distância a pedido da *Jetson Nano*.

O pacote depende do pacote *serial*[60], que é uma biblioteca com interface RS-232 para portas série escrita em `C++` desenvolvida pelo William Woodall.

No ficheiro de configurações poderão ser alteradas as definições da porta série (*serial_port*), *baudrate* (*baud_rate*), velocidade do som (*sound_velocity*, em mm/s), ativação do *ping* (*ping_enable*) e o modo de operação (*mode*). O modo manual ($mode = 0$) permite a alteração do alcance mínimo (*scan_start*, em mm), alcance máximo (*scan_length*, em mm) e do ganho (*gain_setting*, que pode variar entre $0 \rightarrow 0.6$ a $6 \rightarrow 144$ dbm). Já o modo automático ($mode = 1$) define automaticamente o alcance mínimo, máximo e o ganho.

[55] Guia de instalação e de utilização disponível no *website*: https://docs.bluerobotics.com/ping-viewer/.
[56] Documentação sobre o protocolo disponível no *website*: https://docs.bluerobotics.com/ping-protocol/.
[57] Repositório *ping-cpp*: https://github.com/bluerobotics/ping-cpp.
[58] Repositorio *ping-python*: https://github.com/bluerobotics/ping-python.
[59] Disponível no *website*: https://bluerobotics.com/learn/ping-sonar-technical-guide/.
[60] Disponibilizado no *website*: http://wiki.ros.org/serial.

Foi criada uma *custom message*[61] (*PingSimpleDistance.msg*) para permitir que fosse publicado no mesmo tópico a distância (*float32*) e a confiança da medição (*uint8*).

Inicialmente, quando o programa é executado, faz a configuração inicial da ecosonda, isto é, carrega todos os parâmetros do ficheiro de configuração e envia essas configurações para a ecosonda através do *ping-protocol*. Posteriormente, através de um *timer* com duração definida no ficheiro de configuração (*update_duration*), o ROS envia uma solicitação para a ecosonda para receber a distância ao objeto mais próximo, a sua confiança, a temperatura do CPU e da placa. E, quando a eco-sonda envia a resposta com os dados solicitados, são publicados nos tópicos *ping_simple_distance*, *temperature_cpu* e *temperature_board*, respetivamente.

Foram criadas duas funções *callback* que são ativadas de forma diferente:

- *setPingCallback*: ativada quando o tópico *echosounder_ping* sofre alteração (verdadeiro ou falso). Quando o *echososunder_ping* é verdadeiro o *ping* do equipamento é ativado, quando é falso é desativado;
- *setSpeedSound*: ativada quando o tópico *echosounder_sound_velocity* sofre alteração para um determinado valor. Este valor é a velocidade do som na água, que depende da salinidade, pressão e temperatura da água. Quando este valor é alterado, esta função envia o comando de alteração do valor da velocidade do som para a ecosonda.

6.2.4 Sensor GSM - *SIM800L*

O sensor *SIM800L GSM-GPRS 5V V2.0* requer conexão série e utiliza comandos *ATtention* (AT)[62] como protocolo de comunicação entre um computador ou microcontrolador. Existem muitas aplicações deste módulo em projetos de *Arduino*, no entanto, pela pesquisa efetuada, não existe aplicação ainda desenvolvida para ROS. Desta forma foi desenvolvido um driver de origem com recurso ao manual *AT Commands*[63] e ao *datasheet*[64] do *chip SIM800L*.

[61] Tipo de mensagem ROS criada pelo utilizador (http://wiki.ros.org/ROS/Tutorials/DefiningCustomMessages).

[62] Os comandos AT são instruções usadas para controlar um modem. Todas as linhas de comandos começam com "AT" ou "at". É por isso que os comandos do modem são chamados de comandos AT.

[63] Disponível no *website*: https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf.

[64] Disponível no *website*: https://img.filipeflop.com/files/download/Datasheet_SIM800L.pdf.

O driver utiliza o pacote *serial* para a interface série. Através desta interface são enviados os comandos AT para o módulo. Segundo o manual *AT Commands*, na interface série, o driver é o *Data Terminal Equipment* (DTE) e o módulo é o *Terminal Equipment* (TE).

Foram desenvolvidas as seguintes funções específicas para envio e receção de comandos AT:

- *setFunctionalityMode*: envia o comando para alterar o modo de funcionamento. Existem três modos: modo mínimo (sem acesso à internet), modo de vôo (radio frequência desligada) e modo completo. Pode ser alterado no parâmetro *functionality_mode* do ficheiro de configurações ou durante a execução através da publicação do modo no tópico *gsm_functionality*;
- *setDebugCmee*: envia o comando para alterar a ativação do relato do erro do equipamento móvel. Isto é em caso de erro, o TE envia a informação detalhada do erro. Pode ser alterado no parâmetro *debug_cmee* do ficheiro de configurações;
- *setPin*: envia o comando de definição do *pin* do cartão SIM. Pode ser alterado no parâmetro *pin* do ficheiro de configurações;
- *setSleepMode*: envia o comando para alteração do modo de hibernação do equipamento. Isto é, caso o valor do *sleep_mode_* seja verdadeiro, coloca o equipamento na funcionalidade da operação mínima e reduz a velocidade do *clock* do equipamento. Pode ser alterado no parâmetro *sleep_mode* do ficheiro de configurações ou durante a execução através da publicação do modo no tópico *gsm_sleep*;
- *getSignalQuality*: através da resposta ao comando *AT+CSQ*, recebe a qualidade de sinal e publica-a no tópico *gsm_quality_signal*. O valor recebido varia entre $0 \rightarrow -115\ dBm\ ou\ menos$ e $31 \rightarrow -52\ dBm\ ou\ mais$;
- *sendSms*: envia um determinado conjunto de strings publicadas no tópico *send_message* para o número definido no parâmetro *control_number* do ficheiro de configurações;
- *readSms*: recebe as SMS do módulo e publica-as no tópico *received_message_gsm*.

A verificação da chegada de novas mensagens é controlada por um *timer*, cuja duração pode ser alterada no ficheiro de configuração (*update_duration*). Para além desta verificação também é atualizado o valor da qualidade de sinal.

No ficheiro de configuração podem ainda ser alterados os parâmetros da porta série (*serial_port*), *baudrate* (*baud_rate*) e do *timeout* (*time_out_read_serial*).

6.2.5 Sensor GNSS - *VMA430*

O sensor GNSS *VMA430* possui duas interfaces para se estabelecer ligação com o micro PC: *MicroUSB* e UART. Em ambas as interfaces podem ser utilizados três protocolos de comunicação diferentes: NMEA, *Radio Technical Commission for Maritime Services* (RTCM) e *UBX* (protocolo específico de proprietário - *u-blox*) (U-blox, 2018). Para a configuração do sensor GNSS apenas se pode utilizar o protocolo do proprietário.

Por uma questão de rentabilização dos recursos de comunicação da *Jetson* optou-se por utilizar a interface UART.

A empresa *Velleman* disponibiliza uma biblioteca para *Arduino* para operação do sensor de GNSS *VMA430* no repositório do *GitHub*: https://github.com/Velleman/VMA430_GPS_Module (Velleman, 2019).

Inicialmente tentou-se adaptar a biblioteca *Arduino* para ROS, mas optou-se por adaptar um package ROS já existente ao hardware em questão. Foi encontrado um pacote chamado *ublox*[65] adaptado pela *Kumar Robotics*, *Vijay Kumar LAb of School of Engineering and Applied Science* (EUA). Este pacote disponibiliza o *ublox_gps node* para recetores de GNSS, mensagens e pacotes de série utilizando o protocolo *UBX*.

Este pacote está configurado, por padrão, para o sensor de GNSS *C94-M8P*. Foi necessário criar um novo ficheiro de configuração YAML, presente no apêndice B, com as configurações interpretadas dos manuais U-blox (2014, 2018) e e alterar o ficheiro *launch* para carregar o ficheiro de configuração correto. Estas configurações poderão ser vistas no B.1.

O pacote é constituído por três subpastas:

- *ublox_gps*: contém todos os ficheiros de configuração, o ficheiro *launch*, os *headers* e *source files*;
- *ublox_msgs*: constituído por várias *custom messages* para ser possível a integração de todos os dados provenientes do sensor no ROS, e pelos *header* e *source files* que carregam essas mesmas mensagens;

[65] Disponível no repositório: https://github.com/KumarRobotics/ublox.

- *ublox_serialization*: contém os *header files* necessários para estabelecer comunicação com a porta série. Isto é, o pacote não necessita de bibliotecas exteriores para a comunicação série.

O sensor envia para a *Jetson* os dados de latitude, longitude e altitude, esta, por sua vez, publica-os no tópico */ublox_gps/fix*. Para isso é utilizada uma mensagem pré-formatada *sensor_msgs/NavSatFix Message*[66]. Para além da posição também são publicados os dados de velocidade do quadro local de *East, North, Up* (ENU) no tópico */ublox_gps/fix_velocity* que utiliza a mensagem pré-formatada *geometry_msgs/TwistWithCovarianceStamped*[67].

6.2.6 Sensor AHRS - *VN100*

A *VN-100* é um AHRS de gama média que providencia, não só os dados do acelerómetro, giroscópio e magnetómetro, como também a estimativa de orientação por representação baseada em quaterniões, que permite que o sensor funcione em qualquer orientação livre de problemas associados ao bloqueio de *Cardan*.

A empresa disponibiliza um software, *VectorNav Control Center*[68], para *Windows* que permite uma ampla variedade de ferramentas para visualizar e examinar os dados disponibilizados pelo sensor. Para além deste software também disponibiliza várias bibliotecas para desenvolvimento em `C++` e MATLAB[69].

Foi desenvolvido um pacote ROS que providencia a interface de comunicação entre o sensor e a *Jetson Nano*. Foram utilizadas as bibliotecas em `C++` da empresa para máquinas *Linux*.

Em comparação com os quaterniões, os ângulos de *Euler* são simples e intuitivos e prestam-se bem a uma análise e controlo simples. Por outro lado, os ângulos de *Euler* são limitados por um fenómeno chamado *"gimbal lock"*, que os impede de medir a orientação quando o ângulo de inclinação se aproxima de $+/-90$ graus. Apesar dos quaterniões serem menos intuitivos do que os ângulos de *Euler* e da sua matemática poder ser um pouco mais complicada, fornecem uma técnica de medição alternativa que não sofre do fenómeno *"gimbal lock"* (CHRobotics, 2012;

[66] Mensagem pré-formatada disponível no *website*: http://docs.ros.org/melodic/api/sensor_msgs/html/msg/NavSatFix.html.

[67] Mensagem pré-formata que representa uma variação estimada com o quadro de coordenadas de referência com registo de data e hora (http://docs.ros.org/jade/api/geometry_msgs/html/msg/TwistWithCovarianceStamped.html)

[68] Disponível para *download* no *website*: https://www.vectornav.com/support/control-center.

[69] Disponíveis no *website*: https://www.vectornav.com/support/downloads.

Narváez, Árbito & Proaño, 2018). Desta forma, todos os cálculos efetuados para a aquisição do *yaw, pitch* e *roll* são efetuados em quaterniões.

No ficheiro de configurações poderá ser alterado a porta série da conexão (*serial_port*), o *baudrate* (*serial_baud*), taxa da saída assíncrona em Hz (*async_output_rate*), a orientação do sensor no referencial local, isto é ENU ou *North, East, Down* (NED) (*tf_ned_to_enu*), as matrizes das covariâncias da aceleração linear (não produzida pelo sensor - *linear_accel_covariance*), velocidade angular (não produzida pelo sensor - *angular_vel_covariance*), orientação sobrescrita pelo condutor (*orientation_covariance*) e a unidade de saída do *yaw, pitch* e *roll* do tópico */vectornav/Ypr*, que poderá ser em graus ou radianos (*ypr_data_type*).

Este pacote é constituído por sete tópicos que publicam diversos tipos de dados provenientes da AHRS:

- */vectornav/GPS*: tópico que publica a posição GPS da AHRS no formato de mensagem *sensor_msgs/NavSatFix*. No entanto apenas publica esses dados se houver alguma antena GPS ligada ao sensor;
- */vectornav/IMU*: tópico que publica os dados da orientação (x, y, z, w, isto é em quarterniões), velocidade angular (em rad/s), covariância da velocidade angular, aceleração linear (em m/s^2) e covariância da aceleração linear, publicando-os no tipo de mensagem *sensor_msgs/Imu*[70];
- */vectornav/Mag*: tópico que publica a informação referentes aos campos magnéticos x, y e z em unidades *Gauss*[71]. Utiliza a mensagem *sensor_msgs/MagneticField*[72];
- */vectornav/Odom*: tópico que representa uma estimativa da posição e velocidade no espaço livre. Utiliza a mensagem *nav_msgs/Odometry*[73]. Pode ser chamado o serviço *reset_odom* para redefinir a posição inicial para atual;

[70]Mensagem pré-formatada disponível no *website*: http://docs.ros.org/api/sensor_msgs/html/msg/Imu.html.

[71]Um *Gauss* é igual a 10^{-4} *Tesla* (unidade do sistema internacional da densidade de fluxo magnético).

[72]Mensagem pré-formatada disponível no *website*: http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Mag\protect\discretionary{\char\hyphenchar\font}{}{}ne\protect\discretionary{\char\hyphenchar\font}{}{}tic\protect\discretionary{\char\hyphenchar\font}{}{}Fi\protect\discretionary{\char\hyphenchar\font}{}{}eld.html.

[73]Mensagem pré-formatada disponível no *website*: http://docs.ros.org/melodic/api/nav_msgs/html/msg/Odometry.html.

- */vectornav/Pres*: tópico que publica o valor da pressão barométrica registado pela AHRS com recurso à mensagem *sensor_msgs/FluidPressure*[74];
- */vectornav/Temp*: tópico que publica o valor de temperatura registada pelo sensor no formato de mensagem *sensor_msgs/Temperature*[75];
- */vectornav/Ypr*: publica os ângulos do *yaw, pitch* e *roll* utilizando a mensagem *geometry_msgs/Vector3*[76]. A unidade pode ser alterada entre graus e radianos fazendo a alteração da variável *ypr_data_type* no ficheiro de configuração.

6.2.7 Câmara - *Arducam IMX219*

A *Jetson Nano Developer Kit* possui um conector *Mobile Industry Processor Interface* (MIPI)-CSI, *J13*, que é compatível com a câmara *Arducam IMX219* e os drivers de comunicação já vêm instalados no *SDK JetPack*. No entanto foi desenvolvido um guia de instalação destes drivers para futuros trabalhos e implementações, caso haja algum problema de compatibilidade com este equipamento. Este guia encontra-se na secção A.6 do apêndice A e foi desenvolvido tendo por base o guia da *ArduCam* para interface com a *Jetson Nano*[77].

A *Jetson Nano* utiliza uma biblioteca *GStreamer*[78] para assegurar o *streaming* de vídeo das câmaras CSI (*JetsonHacks*[79]).

Através do comando exposto no código-fonte A.5 do apêndice A foi testada a interface da *Jetson* com a camara. Este código utiliza o *GStreamer* para abrir o fluxo de vídeo da câmara, com largura de 3820 píxeis, comprimento de 2464 píxeis a 21 *frames* por segundo e mostra-o numa janela de largura 1920 píxeis e de comprimento 1080 píxeis. O *flip-method* pode ser utilizado para alterar a orientação da câmara e pode tomar vários valores de rotação.

[74] Mensagem pré-formatada disponível no *website*: http://docs.ros.org/melodic/api/sensor_msgs/html/msg/FluidPressure.html.

[75] Mensagem pré-formatada disponível no *website*: http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Temperature.html.

[76] Mensagem pré-formatada disponível no *website*: http://docs.ros.org/api/geometry_msgs/html/msg/Vector3.html.

[77] Disponível no website: https://www.arducam.com/docs/camera-for-jetson-nano/mipi-camera-modules-for-jetson-nano/driver-installation/.

[78] O *GStreamer* é uma biblioteca de código aberto e multi-plataforma utilizada na construção e manipulação de componentes de vídeo e imagens. As aplicações desta biblioteca passam por reprodução simples de *Ogg/Vorbis*, *streaming* de áudio/vídeo, processamento complexo de áudio e vídeo (edição não linear) (https://gstreamer.freedesktop.org/).

[79] Disponível no *website*: https://www.jetsonhacks.com/2019/04/02/jetson-nano-raspberry-pi-camera/.

A *JetsonHacks*[80] disponibiliza alguns exemplos de código em *Python* e `C++` para teste da câmara em aplicações externas.

Foi necessário criar um *metapackage* chamado *camara_csi* por forma a ser possível incluir dois pacotes na mesma pasta. Estes pacotes são o *gscam* e o *jetson_csi_cam*.

Para integrar a câmara no ROS, foi necessário adaptar e utilizar um pacote ROS *camera_driver*[81] chamado *GSCam*[82], pacote que cria uma interface entre o *GStreamer* e o ROS. Para ser possível a utilização deste pacote foi necessário instalar algumas bibliotecas do *GStreamer* adicionais (guia de instalação no apêndice A.6).

O pacote *gscam* utiliza as bibliotecas de ROS *image_pipeline*[83] projetada para preencher a lacuna entre as entradas de vídeo e imagem não processadas e o processamento de visão de alto nível. Por sua vez, esta biblioteca utiliza o OpenCV[84] para aplicar os diversos algoritmos de visão computacional.

O pacote ROS chamado *jetson_csi_cam* foi desenvolvido para simplificar a integração da câmara CSI e das aplicações multimédia da *Nvidia* no pacote ROS *gscam*. É constituído por um ficheiro *.launch*, que configura corretamente o *gscam* para funcionar com o hardware da *Jetson*.

O ficheiro *jetson_csi_cam.launch* define o valor da variável *GSCAM_CONFIG* para operar com o *plugin nvarguscamerasrc* que foi criado pela *NVidia* para a codificação de vídeo da conexão CSI. Para além disso cria os tópicos *csi_cam_0/image_raw* e *csi_cam_0/camara_info*. O primeiro publica a imagem bruta (matriz de pixeis) utilizando a mensagem ROS *sensor_msgs/Images*, o segundo publica a informação da câmara (nome da câmara, caminho e calibração) utilizando a mensagem ROS *sensor_msgs/SetCameraInfo* . Neste ficheiro também são inicializados os nós *image_saver*, que irá guardar as imagens, e *video_recorder* que gravará o vídeo.

[80] Disponível no repositório: https://github.com/JetsonHacksNano/CSI-Camera.

[81] Pacote ROS de drivers para as câmaras desenvolvidos pela comunidade (http://wiki.ros.org/camera_drivers).

[82] Disponível no repositório: https://github.com/ros-drivers/gscam.

[83] Disponível no *website*: https://github.com/ros-perception/image_pipeline.

[84] O OpenCV é uma biblioteca de código aberto e multi-plataforma desenvolvida para aplicações de visão computacional e aprendizagem de máquina. Foi originalmente desenvolvida em 2000 pela *Intel*. O OpenCV possui módulos de processamento de imagem e Video I/O, estrutura de dados, álgebra linear e algoritmos de visão computacional como filtros de imagem, calibração de câmara, reconhecimento de objetos, análise estrutural e outros (https://opencv.org/).

A calibração da câmara é efetuada com recurso ao pacote *camera_calibration* da biblioteca *image_pipeline*. Para ser possível testar a driver foi efetuada uma calibração inicial com recurso ao comando evidenciado no código fonte A.4 do apêndice A.

Os nós *image_saver* e *video_recorder* pertencem ao pacote *image_view*[85] da biblioteca *ros-perception/image_pipeline*. Foi criado um ficheiro de configuração para cada nó onde cada um tem parâmetros distintos:

- *image_saver*:
  - *filename_format*: define o caminho e o nome do ficheiro onde serão guardadas as imagens. A *string '% 04i'* serve para alterar o nome do ficheiro com o número da sequência e a *string '% s'* define o formato do ficheiro, que poderá ser definido para *jpg, png* ou *pgm*;
  - *encoding*: define o espaço de cores da imagem;
  - *save_all_image*: se for definido como falso as imagens só são guardas quando o serviço *save* é chamado (*rosservice call /image_saver/save*). Caso contrário as imagens são guardadas sempre que o tópico *csi_cam_0/image_raw* sofre alterações.
- *video_recorder*:
  - *filename_format*: define o caminho e o nome do vídeo;
  - *fps*: define a *framerate* do vídeo, deverá de ser igual à *framerate* do argumento *fps* do ficheiro *jetson_csi_cam.launch*;
  - *codec*: define o *codec* de vídeo *Four Character Code* (FOURCC)[86];
  - *encoding*: define o espaço de cores da imagem.

Para visualizar o *streaming* da câmara pode ser utilizado o comando *rqt_image_view*. Este comando abre uma janela de visualização de vídeo, onde pode ser selecionada a entrada de vídeo, por defeito é */csi_cam_0/image_raw*.

[85] Informação sobre o pacote disponível no *website*: http://wiki.ros.org/image_view.
[86] Tipos de *codecs* disponíveis no *website*: http://www.fourcc.org/mjpg/.

6.2.8 Atuadores e sensores de potência - *arduino_drivers*

Este pacote é um *metapackage* que é constituído pelo *metapackage rosserial* (composto pelo *rosserial_arduino* e pelo *rosserial_python*), pelo pacote *arduino_driver* e pela pasta *arduino_codes* que juntos constituem o software do sistema de propulsão e de potência (sensores) do veículo. Este pacote foi desenvolvido para garantir a troca de informação entre a *Jetson Nano* e o *Arduino Mega*.

Existem algumas abordagens para a conexão entre placas de desenvolvimento com sistema operativo *Linux* e *Arduino* utilizando o *middleware* ROS, como por exemplo, o *rosserial_arduino*[87] e *ros_arduino_bridge*[88]. Este último pertence ao pacote *rosserial*[89], que é um conjunto de *wrappers* que estabelece um protocolo de comunicação, envio e receção de mensagens ROS, entre dois dispositivos diferentes com recurso à comunicação série ou através de um *socket* de rede.

Por simplicidade no desenvolvimento da driver, optou-se por utilizar a biblioteca *rosserial_arduino*, *wrapper* que permite a comunicação entre a *Jetson Nano* e o *Arduino Mega* utilizado no módulo de locomoção, por forma a ser possível atuar nos servos (barbatanas laterais e/ ou cauda), motores passo a passo (BCU) e receber as informações de tensão e corrente provenientes dos sensores ligados a esta placa. Este pacote disponibiliza ainda um *setup* para configurações avançadas[90] onde se pode ajustar todo o software do *wrapper*.

Foi desenvolvido um pacote ROS chamado *arduino_driver* que executa o carregamento de vários parâmetros de configuração presentes na pasta *config* para o servidor ROS. Através *rosserial_python* esses parâmetros são comutados para o *Arduino*, que por sua vez, solicita ao servidor ROS os parâmetros de configuração necessários para, posteriormente, executar a inicialização dos sensores. Desta forma é possível editar os pinos de conexão de cada sensor/atuador ao *Arduino* sem a necessidade de compilar de novo o código.

No ficheiro de configurações poderão ser alterados o *baudrate*, quais os servos ativos (*servo1-4_is_connected*), quais os pinos de ligação aos quatro canais do módulo de relés (*pin_rele_servo1-4*), qual o estado inicial de cada servo (*power_servo1-4*), quais os pinos PWM do *Arduino* a que estão conectados os servos (*pin_servo1-4*),

[87] Disponível no *website*: https://wiki.ros.org/rosserial_arduino.
[88] Disponível no *website*: http://wiki.ros.org/ros_arduino_bridge.
[89] Disponível no *website*: https://wiki.ros.org/rosserial.
[90] Tutorial disponível no *website*: http://wiki.ros.org/rosserial_arduino/Tutorials/NodeHandle%20and%20ArduinoHardware.

qual o tamanho do pulso mínimo e máximo de cada servo (*min_pulse_width_servo1-4* e *max_pulse_width_servo1-4*) e quais os ângulos máximos e mínimos de cada servo (max_angle_servo1-4 e min_angle_servo1-4).

O software do *Arduino* foi desenvolvido por sistemas, isto é, foram elaborados dois *scripts* que contêm o código desenvolvido para o módulo de propulsão e para o módulo de potência. No futuro estes dois scripts deverão apenas formar um *script*. Assim sendo, dentro da pasta *arduino_codes* pode encontrar-se a subpasta *power_system* e a *propulsion_system*.

No sistema de propulsão, o software desenvolvido apresenta cinco funções *callback* principais:

- *releSubscriver*: chamada quando o tópico *propulsion_system/power_rele* é alterado, e tendo em conta essa alteração, é encarregada de ligar ou desligar os servos fazendo uso do módulo de reles de quatro canais. Este tópico utiliza uma *custom message* chamada *rele.msg* e que é constituída por quatro variáveis *booleans* que correspondem aos quatro servos;
- *setAngleServo1* à *setAngleServo4*: chamada quando o tópico *propulsion_system/servos/angle_servo1* (à *propulsion_system/servos/angle_servo4*) é alterado. O tópico pode tomar um valor inteiro que corresponde ao ângulo da próxima posição do servo em questão.

No sistema de potência, são publicadas as mediação de corrente, fazendo uso de quatro sensores *ACS712T* que deverão ser ligados às saídas dos atuadores. Essas medições são publicadas no tópico*/power_system/corrent_info* sob formato de um *array* de *floats*.

Capítulo 7

Testes e experiências

Como por altura da finalização da dissertação o veículo ainda não tinha sido construído, não foi possível integrar o hardware no mesmo, tendo sido, por isso, bastante limitados os testes realizados, relativamente às funcionalidade dos diversos equipamentos para os quais se produziu software. Mesmo assim, procurou-se fazer um plano de testes que, ainda antes da fase de integração, permitissem aferir a funcionalidade do software desenvolvido.

Para gravar os tópicos em ficheiro *.bag* utilizou-se a linha de comandos do *rosbag*[91] e a aplicação *rqt_bag*[92]. Este último para além de gravar tópicos também permite a visualização e processamento dos tópicos gravados, utilizando *PyQtGraph*[93] e *MatPlotLib*[94].

7.1 Teste ao driver GPS

Foi desenvolvido um código em *Python* chamado *ros_bag_map* que carrega um ficheiro *.bag* com a gravação do tópico */ublox_gps/fix*, que mostra o percurso realizado pelo veículo num *Google Map*. Esta aplicação utiliza a biblioteca *gmplot* que utiliza a *Maps JavaScript API*, por forma a construir uma camada superior com o *plot* das coordenadas. Desta aplicação resulta um ficheiro *HyperText Markup Language* (HTML) que deve ser aberto no *browser* para ser possível observar o mapa com o trajeto do veículo.

[91] Guia de utilização disponível no *website*: http://wiki.ros.org/rosbag/Commandline.

[92] Guia de utilização disponível no *website*: http://wiki.ros.org/rqt_bag.

[93] Biblioteca gráfica e GUI de *Python* construída em PyQt4/PySide e numpy. Destina-se a ser utilizado em aplicações matemáticas/científicas/engenharia. O *PyQtGraph* é distribuído sob a licença de código aberto do MIT (http://www.pyqtgraph.org/).

[94] Biblioteca abrangente para a criação de visualizações estáticas, animadas e interactivas em *Python* (https://matplotlib.org/).

Para validar o pacote *ublox*, estabeleceu-se a ligação do sensor GNSS - *VMA430* à *Jetson Nano* e utilizou-se uma antena padrão [95] de ganho $28dBi$ com conexão SMA-C e extensão de cabo de $5m$.

Numa varanda desenhou-se um triângulo virtual com a antena na mão enquanto se gravava o tópico */ublox_gps/fix*, com recurso ao comando $ rosbag record ublox_gps/fix. De seguida abriu-se o ficheiro *.bag* na aplicação de *Python* desenvolvida. O resultado está evidente na figura 7.1, onde a vermelho representa o movimento que se fez com a antena, e a azul representa o *plot* dos dados provenientes do GPS.

FIGURA 7.1: Mapa de trajeto do Tobias durante teste ao GPS

Desta forma afere-se que o driver *ublox* está a funcionar corretamente, no entanto, o sensor apresentou um erro de precisão de $71m$ (distância medida no *Google Maps* entre triângulo vermelho e o centro do trajeto azul). Para descobrir se o erro provinha da precisão da antena, foi desligada a antena extensiva e utilizada a antena de cerâmica incluída no equipamento, deste modo, teve se se colocar o sensor no

[95] Adquira à *MauserPT*, pode ser consultada no *website*: https://mauser.pt/catalog/product_info.php?cPath=576_2853_849&products_id=824-8126.

exterior fazendo uso de uma extensão USB de $5m$. O resultado deste último teste é evidenciado na figura 7.2, que como se poderá observar, o erro é praticamente zero.

FIGURA 7.2: Mapa de trajeto do Tobias durante teste ao GPS

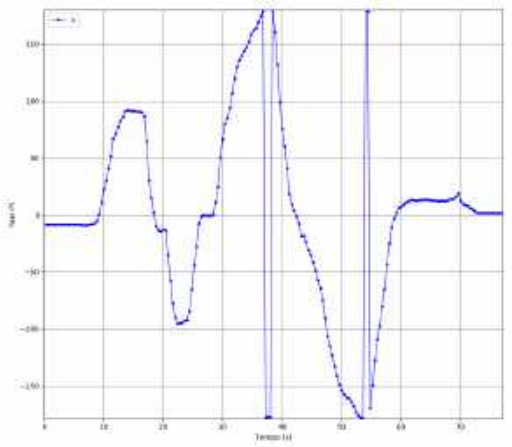
Com isto, concluiu-se que a baixa precisão da posição GPS do primeiro teste deveu-se à antena externa ativa utilizada, para colmatar esse problema, no futuro, será necessário a aquisição de uma antena GPS com melhor precisão e ganho.

7.2 Teste ao driver da AHRS

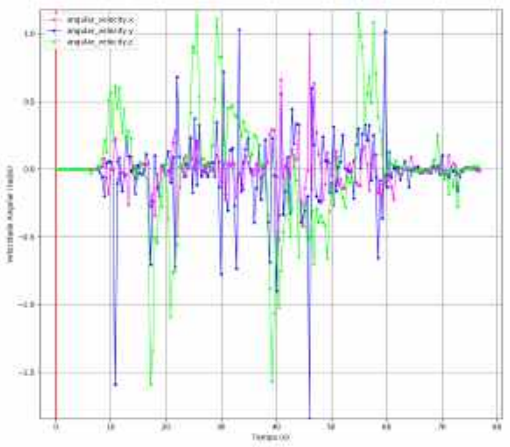
Para validar o driver e os dados fornecidos pela AHRS, foram realizados três testes, e por sua vez, gravados três ficheiros *.bag* diferentes.

No primeiro teste, figura 7.3, gravou-se todos os tópicos do driver com o objetivo de se registar a variação do *yaw*. Durante este teste o sensor foi rodado com intervalos de 90ž. Essa variação está reproduzida no gráfico da figura 7.3a.

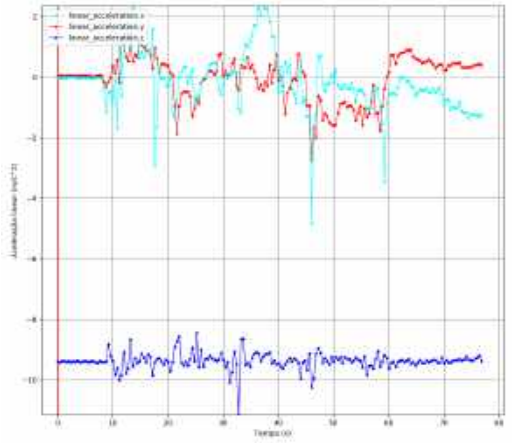
Para além do *yaw*, também reproduziu-se num gráfico a velocidade angular ((figura 7.3b)), a aceleração linear (figura 7.3c) e a densidade de fluxo do campo magnético (figura 7.3d).



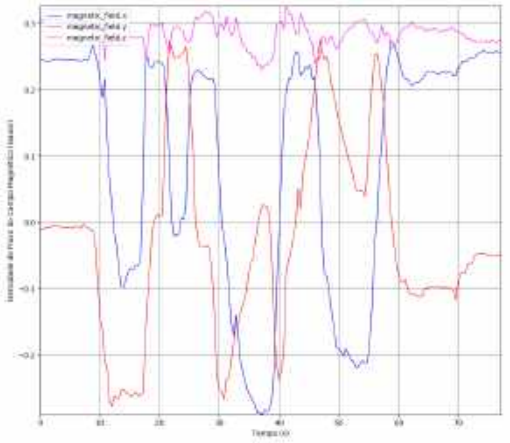
(A) *Yaw* (ž) Vs Tempo (s).



(B) Velocidade (rad/s) Vs Tempo (s).



(C) Aceleração linear (m/s^2) Vs Tempo (s).



(D) Densidade de fluxo magnético ($Gauss$) Vs Tempo (s).

FIGURA 7.3: Gráficos resultantes do primeiro teste ao driver da AHRS.

De seguida testou-se a variação do *pitch*, figura 7.4, começando na posição 0, de seguida na posição 90, 0, -90 e 0 de novo.

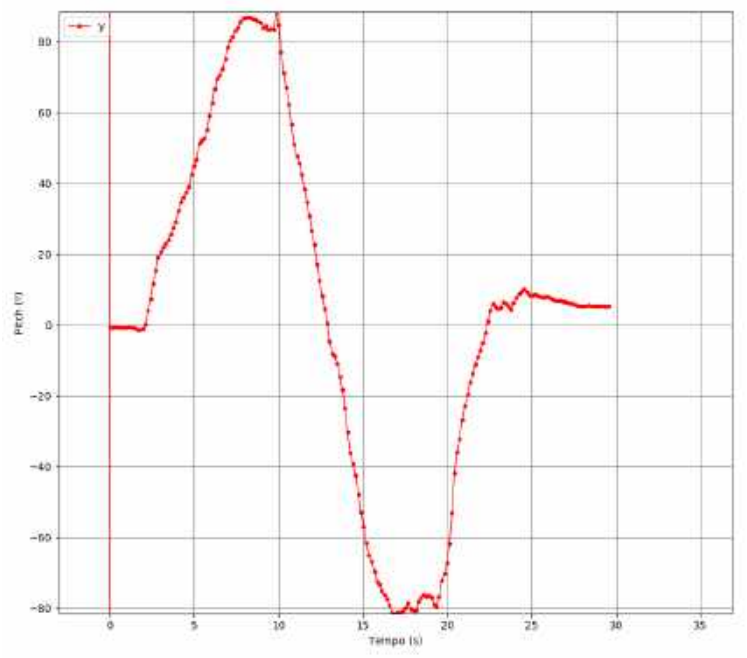


FIGURA 7.4: Gráfico do terceiro teste: *Pitch* (ž) Vs Tempo (s).

No terceiro testou-se a variação do *roll*, figura 7.5, começando na posição 0, de seguida 180, 0, -180 e 0 de novo.

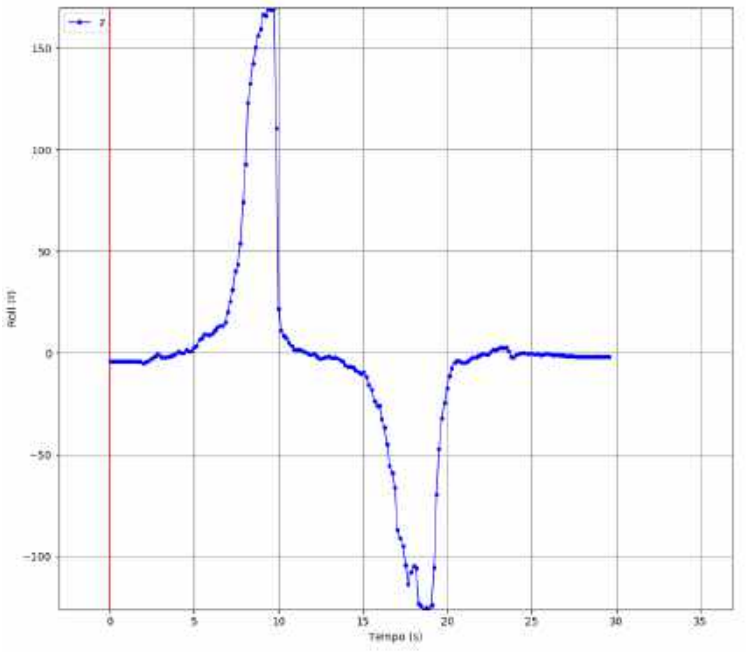


FIGURA 7.5: Gráfico do terceiro teste: *Roll* (ž) Vs Tempo (s).

Todos os dados provenientes do sensor foram validados podendo afirmar assim que o driver está a funcionar corretamente.

7.3 Teste ao driver da ecosonda

Por forma a validar o driver e a capacidade da ecosonda gravou-se o tópico *ping_simple_distance* durante a um teste que foi realizado com a calibração do sensor para a velocidade do som no ar ($350m/s$) com distâncias a um contacto previamente registadas. O sensor foi fixado num suporte no chão a uma distância de $1.5m$ da parede, tal e qual como se pode observar na figura 7.6.

FIGURA 7.6: Ambiente de teste da ecosonda.

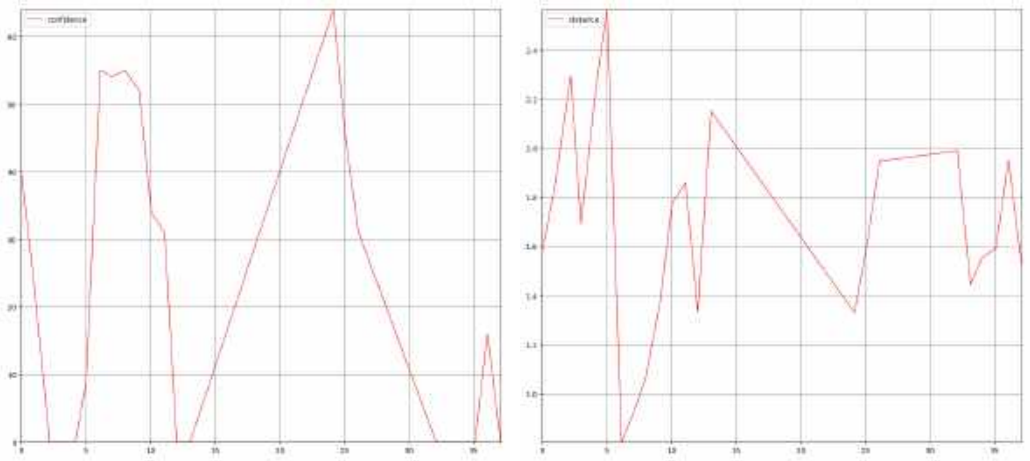
De seguida gravou-se o tópico *ping_simple_distance* durante cerca de 30 segundos. Posteriormente, através do *rqt_bag*, foi visualizado os dados do tópico e foram reproduzidos nos gráficos 7.7a e 7.7b, onde o primeiro representa a confiança da medição ao objeto e o segundo representa o valor absoluto, em metros, da medição da distância entre o sensor e a parede. Ambos os gráficos são resultantes da interpolação linear de cerca de trinta pontos obtidos a partir de um período de aquisição de $35s$.

Após a análise dos resultados e, ao comparar-se os dois gráficos pôde concluir-se que as medições são próximas ao expectável. No entanto existem pontos *outliers* que apresentam valores muito diferentes. Isto deve-se ao facto de o teste ter sido

realizado num ambiente com algumas paredes e outros objetos, que produziu várias reflexões.

O nível de confiança está relacionado com a certeza do sensor sobre a medição efetuada, isto é, quanto maior for a percentagem de confiança mais confiável é a medição. Os valores aproximados de $1.4m$ são obtidos apenas com a taxa de confiança de 50%.

Desta forma, para futuro desenvolvimento do algoritmo de desvio de obstáculos deverá de apenas aceitar valores de distâncias apenas com percentagem de confiança acima dos 50%.



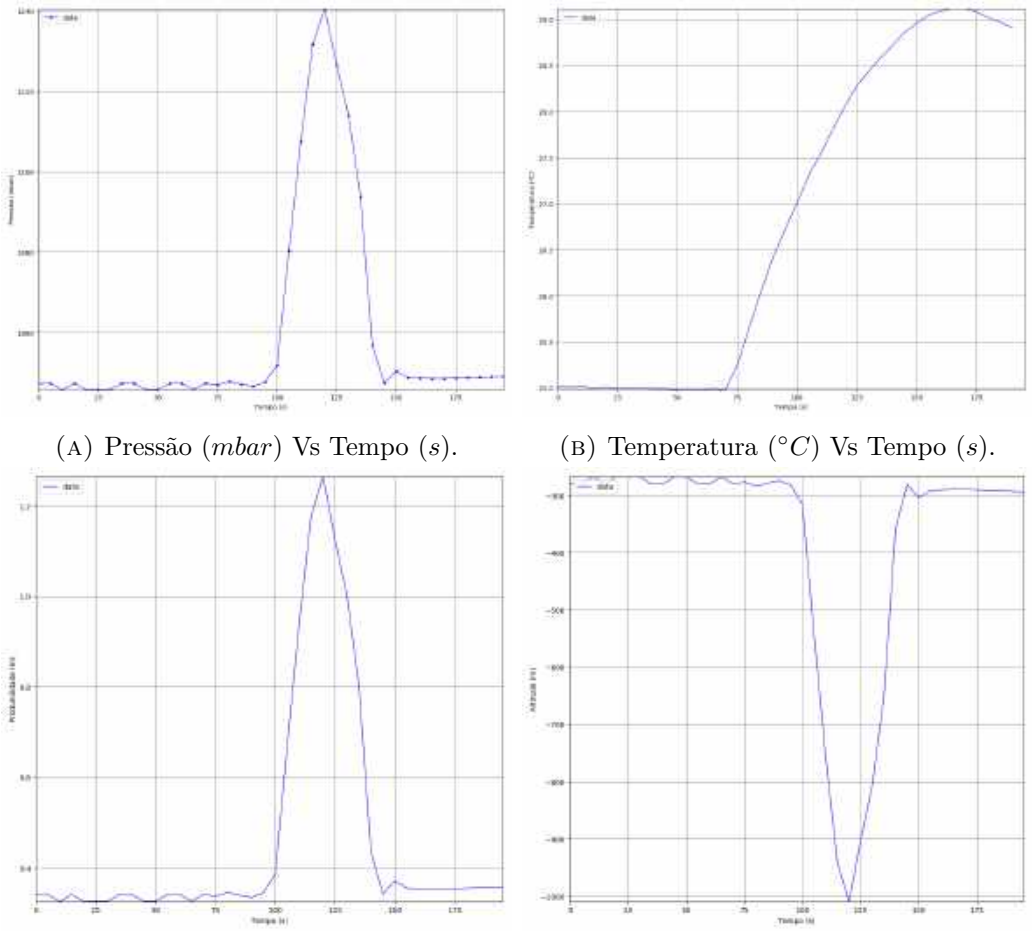
(A) Confiança de medição (%) Vs Tempo (s). (B) Distância ao objeto (m) Vs Tempo (s).

FIGURA 7.7: Gráficos resultantes do teste do driver da ecosonda.

7.4 Teste ao driver do sensor de pressão

O driver do sensor de pressão deveria ser testado dentro de água, o que não foi possível tendo em conta o estado de desenvolvimento atual do veículo. Por essa razão foi testado num ambiente controlado em que a temperatura média era de 29žC. Simulou-se o aumento da pressão na água à medida que a profundidade aumenta, pressionando o sensor com força, aumentado, assim, a pressão medida. Na figura 7.8 pode observar-se os gráficos dos resultados obtidos deste teste.

O gráfico da altitude, figura 7.8d, apenas se encontra evidenciado neste teste porque o sensor foi testado em contacto com o ar e não se poderia aplicar a fórmula de cálculo da profundidade.



(A) Pressão ($mbar$) Vs Tempo (s).

(B) Temperatura ($°C$) Vs Tempo (s).

(C) Profundidade (m) Vs Tempo (s).

(D) Altitude (m) Vs Tempo (s).

FIGURA 7.8: Gráficos resultantes do teste ao driver do sensor de tensão.

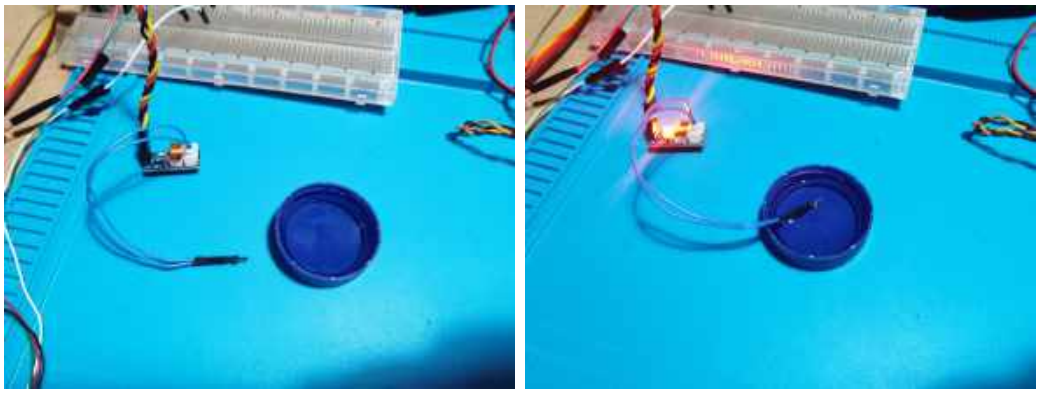
Com estes resultados pode concluir-se que o software em ROS está a funcionar corretamente. Esta conclusão deve-se aos factos de a pressão variar com o aumento da força aplicada ao sensor (figura 7.8a), a temperatura aumenta rapidamente e diminui lentamente (figura 7.8b), a profundidade (7.8c) e a altitude (figura 7.8d) variam com a variação da pressão.

7.5 Teste ao driver *SOS Leak Sensor*

Para testar a operacionalidade do driver para o *SOS Leak Sensor* foi simulada a entrada de água no veículo, utilizando para isso, uma rolha de plástico coberta com água.

Para se proceder ao teste mergulhou-se uma ponta de prova na rolha de plástico coberta com água o que acionou o led vermelho do sensor e publicou o valor

verdadeiro no tópico *leak* no ROS da *Jetson Nano* (figura 7.9b). Quando a ponta de prova foi retirada da rolha, o tópico tomou o valor falso e o led vermelho do sensor apagou-se (figura 7.9a).



(A) Sem entrada de água. (B) Com entrada de água.

FIGURA 7.9: Teste do driver *SOS Leak Sensor*.

Desta forma conclui-se que o driver está operacional e que, no futuro, apenas se terá de desenvolver um determinado comportamento, que é chamado quando o tópico *leak* tomar o valor verdadeiro, para que o veículo venha à superfície.

7.6 Teste ao driver *camara_csi*

O driver para a câmara *Arducam IMX219* de interface CSI permite fazer a gravação de vídeo e imagem, *streaming* e gravação dos pixeis da imagem em tópicos. Para testar este driver foram seguidas três procedimentos diferentes.

Inicialmente estabeleceu-se a coneção do sensor com a *Jetson Nano* e fez-se apenas o *streaming* da câmara utilizando o comando *rqt_image_view*. A figura 7.10 é um *screenshot* da desse *streaming*.

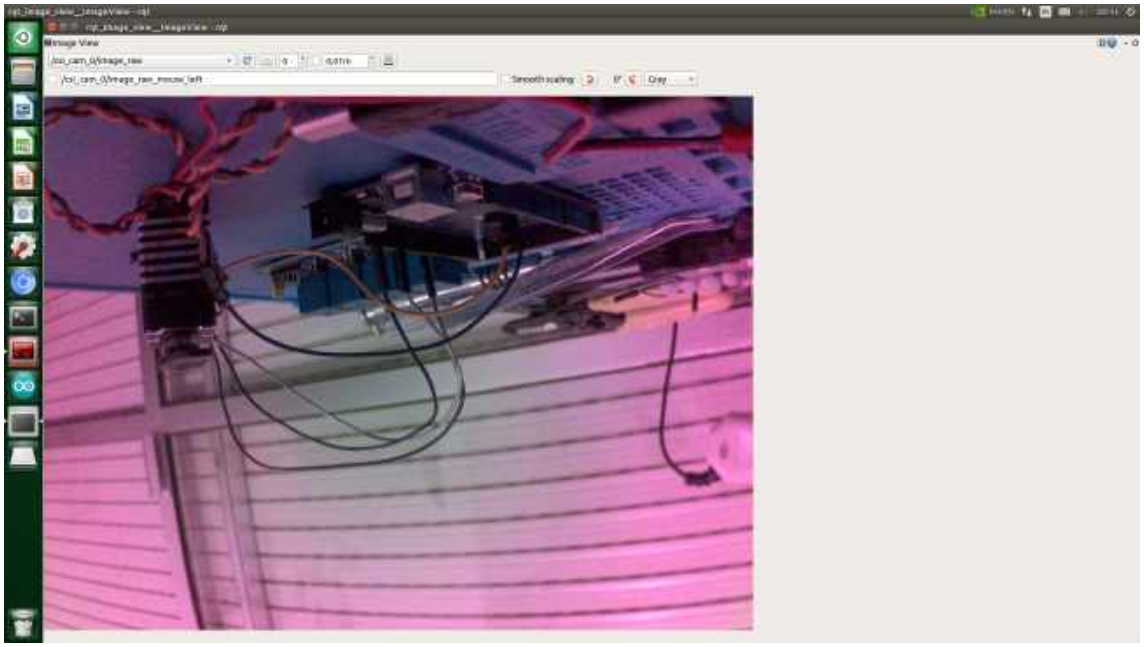


FIGURA 7.10: *Streaming* de imagem (*rqt_image_view*).

Seguidamente gravou-se o tópico */csi_cam_0/image_raw* num ficheiro *.bag* para, posteriormente, reproduzir a imagem utilizando a interface *rqt_bag*. Como se pode observar na figura 7.11, a reprodução da imagem do ficheiro *.bag* funcionou corretamente.

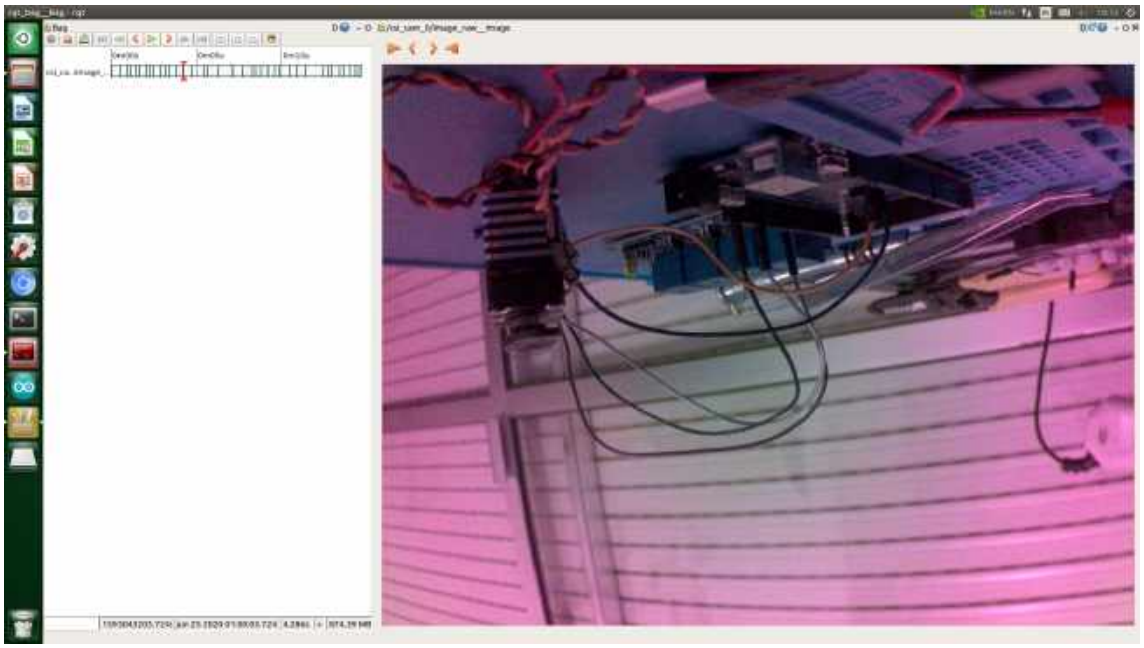


FIGURA 7.11: Reprodução de imagem através de um ficheiro bag (*rqt_bag*).

Por último gravou-se com sucesso um vídeo do *streaming* e uma imagem, figura 7.12, este último com recurso ao *rosservice* (comando $ rosservice call /image_saver/save).

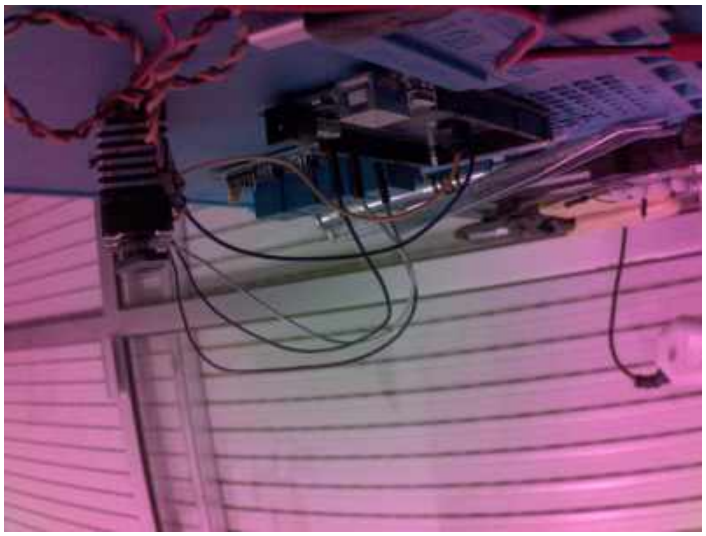


FIGURA 7.12: Imagem guardada utilizando o *rosservice call /image_saver/save*.

7.7 Teste ao driver GSM

A falta de compatibilidade entre cartões atuais e o dispositivo, a necessidade de utilização de um cartão SIM com determinadas características, a requisição de um desses cartões ao serviço de informática ao serviço de informática da EN, e o facto deste ainda não ter sido disponibilizado contribuíram para que não se pudesse realizar os testes pretendidos. Apenas foi possível testar a operacionalidade do driver enquanto interface entre o sensor GSM e a *Jetson Nano*, o qual foi concluído com sucesso.

Caso tivesse sido disponibilizado o cartão, a intenção seria testar a receção e envio de uma mensagem de texto para um número de telemóvel conhecido e, consequentemente, garantir a comunicação do veículo com a *ground station* através da rede móvel.

7.8 Teste ao driver do *Arduino*

O driver do *Arduino* (arduino_driver) foi testado com dois objetivos distintos. O primeiro foi validar o carregamento dos parâmetros ROS no *Arduino* e o

segundo foi validar o envio de comandos da Jetson Nano para o *Arduino* utilizando tópicos ROS.

Para validar o primeiro objetivo foram colocadas no ficheiro de configurações alguns pinos de conexão aos servos e módulo de relés de quatro canais e variáveis de controlo iniciais e verificou-se que essas variáveis foram carregadas no *Arduino*. Assim sendo este teste foi concluído com sucesso e desta forma, para futuros desenvolvimentos com o *Arduino*, poderá colocar-se todas as definições num ficheiro de configurações.

Por forma a verificar o cumprimento do segundo objetivo foram elaborados três testes diferentes. Inicialmente testou-se o envio de ângulos para o servo 1, através do comando $ rostopic pub /propulsion_system/servos/angle_servo1 std_msgs/UInt16 50. Após a publicação deste comando, o servo foi para a posição pretendida.

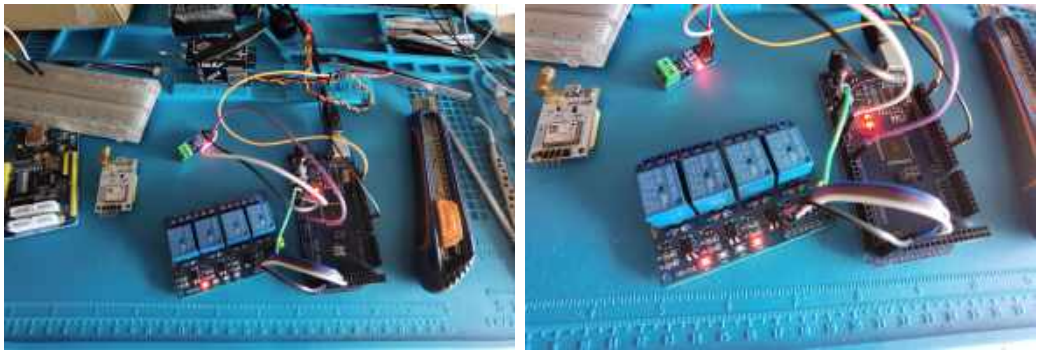
Seguidamente, e pelo sucesso do teste anterior, desenvolveu-se um pacote ROS chamado *sinusoidal_generator*, que cria uma onda sinusoidal para ser descrita no servo. Este pacote desenvolvido corresponde a um primeiro desenvolvimento do controlador *Sinusoidal Generator Node*, membro da arquitetura de software deste veículo. Com este pacote pode fazer-se um teste para validar o driver e a interface desenvolvida para a posterior implementação do movimento ondulatório biomimético da cauda do veículo. No link do *Youtube* https://youtu.be/KUSW3dHcnwo pode observar-se um vídeo, gravado no momento que foi realizado este teste, que mostra o movimento sinusoidal descrito pelo servo com a interface ROS entre a *Jetson Nano* e o *Arduino*.

Por último testou-se a ativação ou desativação dos servos, processo controlado pelo módulo de relés de quatro canais, com recurso ao comando *rostopic pub*. Através dos comandos abaixo mencionados, respetivamente, ativou-se o canal 1 (corresponde ao servo 1) do módulo de relés e posteriormente ativou-se o canal 1 e 4 (corresponde ao servo 1 e 4, respetivamente).

$ rostopic pub /propulsion_system/power_rele propulsion_driver/rele "header: seq: 0 stamp: secs: 0, nsecs: 0 frame_id: ” servo1: true servo2: false servo3: false servo4: false"

$ rostopic pub /propulsion_system/power_rele propulsion_driver/rele "header: seq: 0 stamp: secs: 0, nsecs: 0 frame_id: ” servo1: true servo2: false servo3: false servo4: true"

O sucesso do teste deste último teste pode ser comprovado pela figura 7.13, sendo constituída por duas sub-figuras. A sub-figura 7.13a representa o resultado da publicação da primeira alteração que fez com que o canal 1 fosse ativado e consequentemente acendeu-se o led vermelho do canal 1. A sub-figura 7.13b representa o resultado da segunda alteração ao tópico, que fez com que o canal 1 e 4 fossem ativados e consequentemente acendeu-se o lev vermelho do canal 1 e 4.



(A) Canal 1 (servo 1) ativado. (B) Canal 1 e 4 (servo 1 e 4) ativado.

FIGURA 7.13: Teste do *driver_arduino* no módulo de relés.

Em suma, os dois objetivos, estabelecidos em cima para este driver, foram cumpridos com sucesso.

Capítulo 8

Conclusão

O veículo biomimético submarino ou BUV, Tobias, desenvolvido ao longo desta dissertação foi motivado pela necessidade de se obter um veículo biomimético, por forma a operar em diversas missões e cooperação com outros veículos semelhantes.

Depois de analisado o Estado da Arte concluiu-se que seria uma mais valia para a Marinha Portuguesa desenvolver um robô com este formato e quais os tipos de sensores que deveriam equipar o veículo. Desta forma, fez-se a definição dos requisitos do sistema, que, por conseguinte, definiu-se o conceito do BUV a ser desenvolvido. Partindo da definição de conceito definiu-se a arquitetura de hardware e selecionou-se os equipamentos (sensores e atuadores) e baterias, que deveriam ser incorporadas no veículo para cumprir com os requisitos pré-determinados. Com a definição do hardware procedeu-se à definição da arquitetura robótica de software, tendo sido esta dividida em três camadas. A camada funcional foi projetada utilizando o *framework* ROS. Ainda se desenvolveu todos os drivers dos equipamentos elétricos selecionados no projeto de hardware, utilizando pacotes do ROS. Por último fez-se alguns testes aos drivers individualmente, que serviram para validar o código desenvolvido e, por sua vez, a camada funcional.

Em suma, todos os objetivos estabelecidos no início para esta dissertação foram cumpridos, designadamente: a projeção do hardware e software para um BUV de menor dimensão e custo do que os veículos desenvolvidos no projeto SABUVIS I, e ainda o desenvolvimento da camada funcional com integração em ROS dos diversos equipamentos adquiridos. No entanto, testes mais rigorosos e desafiantes precisam de ser efetuados, nomeadamente, a integração de todos os sistemas e a validação de todo o sistema debaixo de água. Também é necessário desenvolver o sistema de propulsão e o comando e controlo do veículo.

A nível pessoal foi adquirida experiência, conhecimento e sensibilidade para o desenvolvimento de sistemas robóticos, nomeadamente, AUVs.

Todo o código desenvolvido encontra-se acessível e devidamente documentado em https://github.com/hilarioaraujo/tobias_ros. Os drivers desenvolvidos poderão ser utilizados individualmente para outro tipo de sistemas.

Dificuldades

Ao longo da dissertação foram surgindo alguns contratempos característicos do processo de desenvolvimento de um sistema robótico deste nível. Desde logo identifica-se como dificuldades a inexperiência em desenvolver drivers na *framework* ROS, a criação de uma interface I^2C entre a *Jetson Nano* e o sensor de pressão, a escolha de alguns componentes de hardware e implementação destes sensores num veículo submarino.

Para além disto, devido à declaração do Estado de Emergência, no âmbito da pandemia da doença *COVID-19*, a encomenda do hardware selecionado sofreu alguns constrangimentos, que resultaram num atraso de entrega do equipamento e, consequentemente, num começo tardio no desenvolvimento dos drivers para os mesmos.

Apesar das dificuldades sentidas durante o desenvolvimento desta dissertação de mestrado, estas foram ultrapassadas com sucesso.

Trabalho Futuro

Como recomendações para investigações futuras, propõe se o desenvolvimento de temáticas ao nível do hardware e do software.

No que respeita ao hardware sugere-se a realização dum estudo sobre a viabilidade da utilização de comunicação óptica entre veículos a distâncias curtas, onde basicamente são utilizados foto-transístores para capturar a luz emitida por um led de cor azul/verde. Este meio de comunicação de sub-superfície possui um custo de aquisição relativamente baixo quando comparado a um modem acústico. Na área das comunicações submarinas sugere-se, ainda, a implementação do módulo acústico no veículo, conferindo-lhe a capacidade de comunicar debaixo de água. Aconselha-se a implementação de um sonar no veículo para lhe conferir capacidade de reconhecimento e exploração subaquática e de um sensor de salinidade para que

seja calculado o perfil da velocidade do som (importante para alguns sensores e cálculos processamento de dados). É a única variável que falta ao veículo para poder ser dotado dessa capacidade.

Quanto ao software recomenda-se o desenvolvido do sistema de propulsão e a respetiva performance no modo de deslocação do veículo, uma vez que sem veículo é impossível testar a propulsão, usando actuações ondulatórias nos servos da cauda e das barbatanas laterais; do sistema de localização do veículo utilizando um filtro de *Kalman* para poder fundir a informação proveniente da IMU e do GPS numa única variável de estado coerente; da camada executiva; e da camada de planeamento e de um modelo de dinâmica do veículo e do simulação do veículo. Neste último sugere-se a utilização do programa de simulação robótica chamado *Gazebo*, uma vez que tem compatibilidade total com o ROS ou do *UnderWater SIMulator* (UWSim), especializado em simulação subaquática. No entanto, o modelo dinâmico de um veículo com propulsão ondulatória está longe de ser trivial (é bastante mais complexo do que os modelos dos veículos com propulsão convencional a hélice).

Bibliografia

Abreu, P. C., Botelho, J., Góis, P., Pascoal, A., Ribeiro, J., Ribeiro, M., Rufino, M., Sebastião, L. & Silva, H. (2016). The MEDUSA class of autonomous marine vehicles and their role in EU projects. *OCEANS 2016 - Shanghai*. https://doi.org/10.1109/OCEANSAP.2016.7485620

Acoustic Communications Group. (2020). WHOI Micromodem. Obtido 4 janeiro 2020, de https://acomms.whoi.edu/micro-modem/

Adept Technology & Independent Robotics. (2011). AQUA2 - Brochure. http://www.independentrobotics.com/aqua2-robot-brochure.pdf

Afonso, C., de Sousa, J. B. & Martins, R. (2012). SEACON - Sistema de Múltiplos Veículos Autónomos Submarinos. *Revista da Armada*, *460*(2), 8–11.

AIRMAR. (2020). DST800 Smart. Obtido 4 janeiro 2020, de http://www.airmar.com/productdescription.html?id=110

Akçakaya, H., Yildiz, H. A., Sağlam, G. & Gürleyen, F. (2009). Sliding mode control of autonomous underwater vehicle. *ELECO 2009 - 6th International Conference on Electrical and Electronics Engineering*, (December).

Albus, J. S. (1991). Outline for a theory of intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, *21*(3), 473–509. https://doi.org/10.1109/21.97471

Albus, J. S., Lumina, R., Fiala, J. & Wavering, A. (1989). NASREM - The NASA/NBS Standard Reference Model for Telerobot Control System Architecture. *In Proceedings of the 20th International Symposium on Industrial Robots*.

Allotta, B., Costanzi, R., Gelli, J., Pugi, L. & Ridolfi, A. (2015). Design of a modular propulsion system for MARTA AUV. *MTS/IEEE OCEANS 2015 - Genova: Discovering Sustainable Ocean Energy for a New World*, (May). https://doi.org/10.1109/OCEANS-Genova.2015.7271397

Arbib, M. (1992). Schema Theory Schemas : History and Comparisons. *The Encyclopedia of Artificial Intelligence*, 1–16.

Arkin, R. C. (1998). *Behavior-Based Robotics* (1ª ed.). The MIT Press.

Arkin, R. C., Riseman, E. M. & Hanson, A. R. (1988). *AUR: An Architecture for Vision-Based Robot Navigation* (rel. téc.). USA, University of Massachusetts.

Ayers, J. & Witting, J. (2007). Biomimetic approaches to the control of underwater walking machines. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *365*(1850), 273–295. https://doi.org/10.1098/rsta.2006.1910

Azevedo, D. O. M. (2013). *Conceito de Operação dos Veículos Submarinos Autónomos SeaCon a partir dos Submarinos da Classe Tridente* (tese de doutoramento). Escola Naval. http://comum.rcaap.pt/handle/10400.26/12545

Aziz, M. S. & El sherif, A. Y. (2016). Biomimicry as an approach for bio-inspired structure with the aid of computation. *Alexandria Engineering Journal*, *55*(1), 707–714. https://doi.org/10.1016/j.aej.2015.10.015

Bandyopadhyay, P. R. (2005). Trends in biorobotic autonomous undersea vehicles. https://doi.org/10.1109/JOE.2005.843748

Bellingham, J. G., Goudey, C. A., Consi, T. R., Bales, J. W., Atwood, D. K., Leonard, J. J. & Chryssostomidis, C. (1994). A second generation survey AUV. *Proceedings of IEEE Symposium on Autonomous Underwater Vehicle Technology (AUV'94)*, 148–155. https://doi.org/10.1109/AUV.1994.518619

Benyus, J. M. (1997). *Biomimicry: Innovation Inspired by Nature*. Harper Collins. https://doi.org/10.2307/4450504

Blue Robotics. (2017). Low-Light HD USB Camera. Obtido 24 fevereiro 2020, de https://bluerobotics.com/store/electronics/cam-usb-low-light-r1/

BlueRobotics. (2018). *BlueROV 2 Datasheet* (rel. téc.). BlueRobotics. http://bluerobotics.com/downloads/bluerov2.pdf

BlueRobotics. (2020a). Bar30 High-Resolution 300m Depth/Pressure Sensor. Obtido 4 janeiro 2020, de https://bluerobotics.com/store/sensors-sonars-cameras/sensors/bar30-sensor-r1/

BlueRobotics. (2020b). Ping Sonar Echosounder for Underwater Distance Measurement. Obtido 4 janeiro 2020, de https://bluerobotics.com/store/sensors-sonars-cameras/sonar/ping-sonar-r2-rp/

BlueRobotics. (2020c). Ping360 Scanning Imaging Sonar for ROVs. Obtido 4 janeiro 2020, de https://bluerobotics.com/store/sensors-sonars-cameras/sonar/ping360-sonar-r1-rp/

BlueRobotics. (2020d). SOS Leak Sensor. Obtido 4 janeiro 2020, de https://bluerobotics.com/store/sensors-sonars-cameras/leak-sensor/sos-leak-sensor/

BMT. (2012). Shoal. Obtido 6 fevereiro 2020, de https://www.roboshoal.com/

Boston Engineering. (2019). BIOSwimmer ™ Unmanned Underwater Vehicle (UUV).

Brooks, R. (1985). A Robust Layered Control Architecture for a Mobile Robot. http://dl.acm.org/citation.cfm?id=888753

Brooks, R. a. (1987). How to Build Complete Creatures Rather than Isolated Cognitive Simulators. *Artificial Intelligence*, 225–239.

Brown, C. & Clark, R. P. (2010). Using a novel vehicle conceptual design utility to evaluate a long-range, large payload UUV. *OCEANS 2010 MTS/IEEE SEATTLE*, 1–10. https://doi.org/10.1109/OCEANS.2010.5664540

Brown, R. G. & Hwang, P. Y. C. (2012). *Introduction to Random Signals and Applied Kalman Filtering - with MATLAB Exercises* (4ª ed., Vol. 1). John Wiley & Sons.

Chocron, O. & Delaleau, E. (2019). Trajectory-Based Synthesis of Propulsion Systems for Fixed-Thrusters AUVs BT - ROMANSY 22 – Robot Design, Dynamics and Control. Em V. Arakelian & P. Wenger (Eds.). Springer International Publishing.

Chowdhury, A. R., Kumar, V., Prasad, B., Kumar, R. & Panda, S. K. (2014). Kinematic study and implementation of a bio-inspired robotic fish underwater vehicle in a Lighthill mathematical framework. *Robotics and Biomimetics*, *1*(1), 15. https://doi.org/10.1186/s40638-014-0015-2

Christ, R. D. & Wernli, R. L. (2013). *The ROV manual: a user's guide to remotely operated vehicles*.

CHRobotics. (2012). *AN-1006 - Understanding Quaternions* (rel. téc.). CHRobotics. http://www.chrobotics.com/docs/AN-1006-UnderstandingQuaternions.pdf

Clapham, R. J. & Hu, H. (2015). iSplash: realizing fast Carangiform swimming to outperform a real fish. *Springer Tracts in Mechanical Engineering*, *12*, 193–218. https://doi.org/10.1007/978-3-662-46870-8_7

Conry, M., Keefe, A., Ober, W., Rufo, M. & Shane, D. (2013). BIOSwimmer: Enabling technology for port security. *2013 IEEE International Conference on Technologies for Homeland Security, HST 2013*, 364–368. https://doi.org/10.1109/THS.2013.6699031

Elkady, A. & Sobh, T. (2012). Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography. *Journal of Robotics*, *2012*, 1–15. https://doi.org/10.1155/2012/959013

Evans, J. C., Smith, J. S., Martin, P. & Wong, Y. S. (1999). Beach and nearshore crawling UUV for oceanographic measurements. *Oceans '99. MTS/IEEE. Riding the Crest into the 21st Century. Conference and Exhibition. Conference Proceedings (IEEE Cat. No.99CH37008)*, *3*, 1300–1306. https://doi.org/10.1109/OCEANS.1999.800180

Fernandes, M. (2013). Biomimética e o Design. Obtido 26 janeiro 2020, de https://www.publico.pt/2013/03/20/p3/cronica/biomimetica-e-o-design-1816727

Fish, F. E. & Kocak, D. M. (2011). *Biomimetics and Marine Technology: An Introduction* (B. Bingham, C. Jaskolski, D. Kocak, S. Kraus, D. Lindsay, J. Manley, S. Showalter, J. Stanley, E. Widder & J. Zande, Eds.; Vol. 45). Marine Technology Society. https://doi.org/10.4031/mtsj.45.4.14

Flight Global. (2006). Satellite navigation: What is WAAS/EGNOS? Obtido 12 fevereiro 2020, de https://www.flightglobal.com/satellite-navigation-what-is-waas/egnos/69463.article

Fryxell, D., Oliveira, P., Pascoal, A., Silvestre, C. & Kaminer, I. (1996). Navigation, guidance and control of AUVs: An application to the MARIUS vehicle. *Control Engineering Practice*, *4*(3), 401–409. https://doi.org/10.1016/0967-0661(96)00018-4

Gadag, R. & Shetty, A. N. (2014). Elastomers and Synthetic Rubbers. *Engennering Chemistry* (3ª ed., p. 296). I K International Publishing House.

Gallimore, E., Stokey, R. & Terrill, E. (2018). Robot Operating System (ROS) on the REMUS AUV using RECON. *AUV 2018 - 2018 IEEE/OES Autonomous Underwater Vehicle Workshop, Proceedings*, 1–6. https://doi.org/10.1109/AUV.2018.8729755

Galloway, K. C., Clark, J. E. & Koditschek, D. E. (2013). Variable Stiffness Legs for Robust, Efficient, and Stable Dynamic Running. *Journal of Mechanisms and Robotics*, *5*(1). https://doi.org/10.1115/1.4007843

Gonçalves, C. F. S. (2016). *Projeto e Desenvolvimento de um Veículo Submarino Autónomo* (tese de doutoramento). Faculdade de Engenharia da Universidade do Porto.

GSA. (2018). *GNSS User Technology Report*. European GNSS Agency. https://doi.org/10.2878/743965

Guo, S., Fukuda, T., Kato, N. & Oguro, K. (1998). Development of underwater microrobot using ICPF actuator. *Proceedings - IEEE International Conference on Robotics and Automation*, *2*(May), 1829–1834. https://doi.org/10.1109/ROBOT.1998.677433

He, B., Wang, B. R., Yan, T. H. & Han, Y. Y. (2013). A distributed parallel motion control for the multi-thruster autonomous underwater vehicle. *Mechanics Based Design of Structures and Machines*, *41*(2), 236–257. https://doi.org/10.1080/15397734.2012.726847

Healey, A. J., Marco, D. B. & McGhee, R. B. (1996). Autonomous underwater vehicle control coordination using a tri-level hybrid software architecture.

Proceedings of IEEE International Conference on Robotics and Automation, *3*, 2149–2159 vol.3. https://doi.org/10.1109/ROBOT.1996.506188

HiBot. (2017). ACM-R5H. Obtido 17 fevereiro 2020, de https://www.hibot.co.jp/ecommerce/prod-detail/39

Hou, T., Yang, X., Su, H., Jiang, B., Chen, L., Wang, T. & Liang, J. (2019). Design and Experiments of a Squid-Like Aquatic-Aerial Vehicle with Soft Morphing Fins and Arms. *2019 International Conference on Robotics and Automation (ICRA)*, *2019-May*, 4681–4687. https://doi.org/10.1109/ICRA.2019.8793702

Huang, H.-M. (2008). Autonomy levels for unmanned systems (ALFUS) framework. *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems - PerMIS '07*, *1*(October), 48–53. https://doi.org/10.1145/1660877.1660883

Inside Unmanned Systems. (2014). U.S. Navy Completes Tests on the GhostSwimmer. Obtido 5 fevereiro 2020, de https://insideunmannedsystems.com/u-s-navy-completes-tests-ghostswimmer/

Intel. (2020). IntelDual Band Wireless - AC 8265 Product Specifications. Obtido 4 janeiro 2020, de https://ark.intel.com/content/www/us/en/ark/products/94150/intel-dual-band-wireless-ac-8265.html

Jiang, S. & Georgakopoulos, S. (2011). Electromagnetic Wave Propagation into Fresh Water. *Journal of Electromagnetic Analysis and Applications*, *03*(07), 261–266. https://doi.org/10.4236/jemaa.2011.37042

Joseph, L. (2018). *Robot Operating System for Absolute Beginners*. Apress. https://doi.org/10.1007/978-1-4842-3405-1

Katzschmann, R. K., DelPreto, J., MacCurdy, R. & Rus, D. (2018). Exploration of underwater life with an acoustically controlled soft robotic fish. *Science Robotics*, *3*(16), eaar3449. https://doi.org/10.1126/scirobotics.aar3449
From Duplicate 1 (Exploration of underwater life with an acoustically controlled soft robotic fish - Katzschmann, Robert K.; DelPreto, Joseph; MacCurdy, Robert; Rus, Daniela) SoFi - biomimetic fish of the MIT

Katzschmann, R. K. (2018). Building and controlling fluidically actuated soft robots : from open loop to model-based control. https://dspace.mit.edu/handle/1721.1/119278%7B%5C#%7Dfiles-area
Importante para a base - 2.4.2 Para o paiva - pag 98

Kolding, T., Frederiksen, F. & Mogensen, P. (2002). Performance aspects of WCDMA systems with high speed downlink packet access (HSDPA). *Proceedings IEEE 56th Vehicular Technology Conference*, 477–481. https://doi.org/10.1109/VETECF.2002.1040389

Kortenkamp, D. & Simmons, R. (2008). Robotic Systems Architectures and Programming. Em B. Siciliano & O. Khatib (Eds.), *Springer Handbook of Robotics*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-30301-5

Kottapalli, A. G. P., Asadnia, M., Miao, J. & Triantafyllou, M. S. (2016). *Biomimetic microsensors inspired by marine life* (1ª ed.). Springer Nature. https://doi.org/10.1007/978-3-319-47500-4

Krause, J., Winfield, A. F. & Deneubourg, J. L. (2011). Interactive robots in experimental biology. https://doi.org/10.1016/j.tree.2011.03.015

Laboratório de Sistemas e Tecnologia Subaquática & Marinha Portuguesa. (2020). Home | REP. Obtido 2 fevereiro 2020, de https://rep.lsts.pt/

Lala, J. H. & Harper, R. E. (1994). Architectural principles for safety-critical real-time applications. *Proceedings of the IEEE*, *82*(1), 25–40. https://doi.org/10.1109/5.259424

Laschi, C., Mazzolai, B. & Cianchetti, M. (2016). Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Science Robotics*, *1*(1), 1–12. https://doi.org/10.1126/scirobotics.aah3690

LattePanda. (2020). LattePanda 4G/64GB. Obtido 2 janeiro 2020, de https://www.lattepanda.com/products/3.html

Li, J. H., Lee, M. J., Park, S. H., Kim, J. G., Kim, J. T. & Suh, J. H. (2013). Development of P-SURO II hybrid AUV and its experimental study. *OCEANS 2013 MTS/IEEE Bergen: The Challenges of the Northern Dimension*. https://doi.org/10.1109/OCEANS-Bergen.2013.6608045

Liu, J. & Hu, H. (2005). Mimicry of sharp turning behaviours in a robotic fish. *Proceedings - IEEE International Conference on Robotics and Automation*, *2005*(April), 3318–3323. https://doi.org/10.1109/ROBOT.2005.1570622

Liu, J., Hu, H. & Gu, D. (2006). A hybrid control architecture for autonomous robotic fish. *IEEE International Conference on Intelligent Robots and Systems*, (1), 312–317. https://doi.org/10.1109/IROS.2006.282422

Long, J.H., J. (2011). Biomimetics: Robotics Based on Fish Swimming. *Encyclopedia of Fish Physiology: From Genome to Environment* (1ª ed., pp. 603–612). Elsevier Academic Press: San Diego. https://doi.org/10.1016/B978-0-1237-4553-8.00232-X

Loureiro, H. (2019). *C++ - Guia Moderno de Programação* (1ª ed.). FCA - Editora de Informática. https://www.fca.pt/pt/catalogo/informatica/programacao/c/

Mahtani, A., Sanchez, L., Fernandez, E., Martinez, A. & Joseph, L. (2018). *ROS Programming: Building Powerful Robots* (1ª ed.). Packt Publishing.

Medeiros, A. A. D. (1998). A survey of control architectures for autonomous mobile robots. *Journal of the Brazilian Computer Society*, *4*.

Miklosi, A. & Gerencser, L. (2012). Potential application of autonomous and semi-autonomous robots in the study of animal behaviour. *3rd International Conference on Cognitive Infocommunications (CogInfoCom)*, 759–762. https://doi.org/10.1109/CogInfoCom.2012.6421952

Miklósi, Á., Korondi, P., Matellán, V. & Gácsi, M. (2017). Ethorobotics: A New Approach to Human-Robot Relationship. *Frontiers in Psychology*, *8*(JUN), 1–8. https://doi.org/10.3389/fpsyg.2017.00958

Miller, D. P. (1996). Design of a small, cheap UUV for under-ship inspection and salvage. *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology*, (September), 18–20. https://doi.org/10.1109/auv.1996.532395

Montemerlo, M., Thrun, S., Dahlkamp, H., Stavens, D. & Strohband, S. (2006). Winning the DARPA grand challenge with an AI robot. *Proceedings of the National Conference on Artificial Intelligence*, *1*(Gat 1998), 982–987.

Morawski, M., Malec, M. & Zajac, J. (2014). Development of CyberFish – Polish Biomimetic Unmanned Underwater Vehicle BUUV. *Applied Mechanics and Materials*, *613*, 76–82. https://doi.org/10.4028/www.scientific.net/AMM.613.76

Muljowidodo, K., Adi N., S., Budiyono, A., Prayogo, N., Adi, S., Budiyono, A. & Prayogo, N. (2009). Design of SHRIMP ROV for surveillance and mine sweeper. *Indian Journal of Marine Sciences*, *38*(3), 332–337.

Murphy, R. R. (2000). *Introduction to AI Robotics* (1ª ed.). MIT Press.

Narváez, F., Árbito, F. & Proaño, R. (2018). A Quaternion-Based Method to IMU-to-Body Alignment for Gait Analysis. *Medical oncology and tumor pharmacotherapy* (1ª ed., pp. 217–231). Springer. https://doi.org/10.1007/978-3-319-91397-1_19

NetWifiWorks. (2020). Ubiquiti PicoStation M Series. Obtido 4 janeiro 2020, de https://www.netwifiworks.com/PicoStation-M.asp

Nicolai, L. M. (2002). Anti-submarine warfare uav and method of use thereof.

Nourbakhsh, I. R. & Siegwart, R. (2004). *Autonomous Mobile Robots* (R. C. Arkin, Ed.; 1ª ed.). The MIT Press. http://home.deib.polimi.it/gini/robot/docs/siegwart.pdf

NVIDIA. (2019a). Jetson Nano Developer Kit User Guide. https://developer.nvidia.com/embedded/downloads%7B%5C#%7D?search=Jetson%20Nano

NVIDIA. (2019b). NVIDIA Jetson Nano Developer Kit | NVIDIA Developer. Obtido 2 janeiro 2020, de https://developer.nvidia.com/embedded/jetson-nano-developer-kit

Ocean Explorer. (2015). What is an AUV? Obtido 19 outubro 2019, de https://oceanexplorer.noaa.gov/facts/auv.html

Oliveira, L. M. (2018). Desenvolvimento de Veículos Autônomos Submarinos para Aplicações Oceanográficas.

Pawlyn, M. (2011). *Biomimicry in Architecture* (2ª ed.). Riba Publishing. https://doi.org/1000701603

PC/104 Consortium. (2020). What is PC/104? Obtido 2 janeiro 2020, de https://pc104.org/hardware-specifications/pc104/

Pieterkosky, S., Cavanaugh, C. & Thompson, L. (2017). FaaS - Fish as a service biomimetic fish drone for ocean monitoring. *OCEANS 2017 - Anchorage*, 1–4.

Pinto, J., Dias, P. S., Martins, R., Fortuna, J., Marques, E. & Sousa, J. (2013). The LSTS toolchain for networked vehicle systems. *OCEANS 2013 MTS/IEEE Bergen: The Challenges of the Northern Dimension*. https://doi.org/10.1109/OCEANS-Bergen.2013.6608148

Pinto, J., Sousa, J., Py, F. & Rajan, K. (2012). Experiments with Deliberative Planning on Autonomous Underwater Vehicles. *WREM2012 - Workshop on Robotics for Environmental Monitoring*, (January 2014), 1–6.

Prahacs, C., Saudners, A., Smith, M. K., McMordie, D. & Buehler, M. (2011). TOWARDS LEGGED AMPHIBIOUS MOBILE ROBOTICS. *Proceedings of the Canadian Engineering Education Association*, (January). https://doi.org/10.24908/pceea.v0i0.4043

Raj, A. & Thakur, A. (2016). Fish-inspired robots: design, sensing, actuation, and autonomy—a review of research. *Bioinspiration & Biomimetics*, *11*(3), 031001. https://doi.org/10.1088/1748-3190/11/3/031001

From Duplicate 1 (Fish-inspired robots: design, sensing, actuation, and autonomy - a review of research - Raj, Aditi; Thakur, Atul) Introdução aos peixes robóticos

RaspberryPi. (2020). Buy a Raspberry Pi 4 Model B. Obtido 2 janeiro 2020, de https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

Research Technical Proposal - SABUVIS II (rel. téc.). (2018).

Ridao, P., Yuh, J., Batlle, J. & Sugihara, K. (2000). On AUV control architecture. *IEEE International Conference on Intelligent Robots and Systems*, *2*, 855–860. https://doi.org/10.1109/IROS.2000.893126

Roberts, S. & Koditschek, D. E. (2016). RHex Slips on Granular Media. *University of Pennsylvania Technical Report*, (January 2016), 2–3.

Roberts, T. J. (2016). Contribution of elastic tissues to the mechanics and energetics of muscle function during movement. *Journal of Experimental Biology*, *219*(2), 266–275. https://doi.org/10.1242/jeb.124446

ROBOTS. (2019). Aqua2. Obtido 17 fevereiro 2020, de https://robots.ieee.org/robots/aqua/

RobotShop. (2018). Arducam 8MP Sony IMX219 Camera Module w/ CS lens 2718 (Raspberry Pi). Obtido 24 fevereiro 2020, de https://www.robotshop.com/eu/en/arducam-8mp-sony-imx219-camera-module-cs-lens-2718-raspberry-pi.html

Rösmann, C., Hoffmann, F. & Bertram, T. (2017). *Robot Operating System (ROS)* (A. Koubaa, Ed.; Vol. 707). Springer International Publishing. https://doi.org/10.1007/978-3-319-54927-9

Rossi, C., Colorado, J., Coral, W. & Barrientos, A. (2011). Bending continuous structures with SMAs: a novel robotic fish design. *Bioinspiration & Biomimetics*, *6*(4), 1–15. https://doi.org/10.1088/1748-3182/6/4/045005

Rufo, M. (2013). GhostSwimmer ™ : Tactically Relevant , Biomimetically Inspired , Silent , Highly Efficient and Maneuverable Autonomous Underwater Vehicle.

Rus, D. & Tolley, M. T. (2015). Design, fabrication and control of soft robots. *Nature*, *521*(7553), 467–475. https://doi.org/10.1038/nature14543

Ryuh, Y. S., Yang, G. H., Liu, J. & Hu, H. (2015). A school of robotic fish for mariculture monitoring in the sea coast. *Journal of Bionic Engineering*, *12*(1), 37–46. https://doi.org/10.1016/S1672-6529(14)60098-6

Saranli, U., Buehler, M. & Koditschek, D. E. (2001). RHex: A Simple and Highly Mobile Robot. *International Journal of Robotics Research*, *20*(7), 616–631.

Saridis, G. N. & Valavanis, K. P. (1988). Analytical design of intelligent machines. *Automatica*, *24*(2), 123–133. https://doi.org/10.1016/0005-1098(88)90022-2

Seto, M. L., Paull, L. & Saeedi, S. (2013). Introduction to Autonomy for Marine Robots. *Marine Robot Autonomy* (1ª ed., pp. 1–46). Springer-Verlag New York. https://doi.org/10.1007/978-1-4614-5659-9

Seto, M., Paull, L. & Saeedi, S. (2013). Introduction to Autonomy for Marine Robots. Em M. L. Seto (Ed.), *Marine Robot Autonomy* (pp. 1–382). Springer Science+Busines. https://doi.org/10.1007/978-1-4614-5659-9-1

Siegwart, R., Nourbakhsh, I. R. & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots* (2ª ed., Vol. 53). The MIT Press. https://mitpress.mit.edu/books/introduction-autonomous-mobile-robots-second-edition

SIMCom. (2013). Sim800L Hardware Design. https://doi.org/SIM800L_Hardware_Design_V1.00

Standard Guide for UUV Autonomy and Control. (2006). *ASTM F2541-06*. https://doi.org/10.1520/F2541-06

Stokey, R., Austin, T., Allen, B., Forrester, N., Gifford, E., Goldsborough, R., Packard, G., Purcell, M. & von Alt, C. (2001). Very shallow water mine countermeasures using the REMUS AUV: a practical approach yielding accurate results. *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*, *1*, 149–156 vol.1. https://doi.org/10.1109/OCEANS.2001.968696

Szymak, P., Praczyk, T., Pietrukaniec, L. & Hozyn, S. (2017). *Experimental Robotics* (J. P. Desai, G. Dudek, O. Khatib & V. Kumar, Eds.; Vol. 63). Springer International Publishing. https://doi.org/10.1007/978-3-319-00065-7

Tan, C. S., Sutton, R. & Chudley, J. (2007). An integrated collision avoidance system for autonomous underwater vehicles. *International Journal of Control*, *80*(7), 1027–1049. https://doi.org/10.1080/00207170701286702

Teledyne Gavia. (2017). *Gavia User Manual* (8ª ed.). Teledyne Gavia ehf.

Thomas D. Barron. (1998). Apparatus for interconnecting an underwater vehicle and a free floating communications pod.

Triantafyllou, M. S. & Triantafyllou, G. S. (1995). An Efficient Swimming Machine. *Scientific American*, *272*(3), 64–70. https://doi.org/10.1038/scientificamerican0395-64

Ubiquiti. (2019). Bullet™ M. Obtido 4 janeiro 2020, de https://www.ui.com/airmax/bulletm/

U-blox. (2014). NEO-7 GNSS Datasheet, 1–40.

U-blox. (2018). NEO-7 - Receiver Description.

U.S. Navy. (2004). *The Navy Unmanned Undersea Vehicle (UUV) - Master Plan* (tese de doutoramento July). http://www.navy.mil/navydata/technology/uuvmp.pdf

Vallejo, R., Sanguino, J. & Rodrigues, A. (2014). Posicionamento com GNSS em cenários de multi-constelação.

Vaz, F. C., Portugal, D., Araújo, A., Couceiro, M. S. & Rocha, R. P. (2018). A localization approach for autonomous underwater vehicles: A ROS-Gazebo framework, 2–3. http://arxiv.org/abs/1811.05836

Vega, E. P., Chocron, O., Ferreira, J. V., Benbouzid, M. & Meirelles, P. S. (2015). Evaluation of AUV fixed and vectorial propulsion systems with dynamic simulation and non-linear control. *IECON 2015 - 41st Annual Conference of*

the IEEE Industrial Electronics Society, (December 2016), 944–949. https://doi.org/10.1109/IECON.2015.7392221

Vehicle Control Technologies. (2015). HarborScan. https://auvac.org/configurations/view/235

Velleman. (2019). VMA430 User Manual.

Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R. & Das, H. (2001). The CLARAty architecture for robotic autonomy. *IEEE Aerospace Conference Proceedings*, *1*(March), 1121–1132. https://doi.org/10.1109/aero.2001.931701

Wang, F. Y., Kyriakopoulos, K. J., Tsolkas, A. & Saridis, G. N. (1991). A Petri-net coordination model for an intelligent mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics*, *21*(4), 777–789. https://doi.org/10.1109/21.108296

Wang, W. H., Chen, X. Q., Marburg, A., Chase, J. G. & Hann, C. E. (2008). A Low-Cost unmanned underwater vehicle prototype for shallow water tasks. *2008 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications, MESA 2008*, (May 2014), 204–209. https://doi.org/10.1109/MESA.2008.4735649

Wang, X. & Liang, S. (2019). Maneuverability Analysis of a Novel Portable Modular AUV. *Mathematical Problems in Engineering*, *2019*. https://doi.org/10.1155/2019/1631930

XSENS. (2020). Inertial Sensor Modules. Obtido 5 janeiro 2020, de https://www.xsens.com/inertial-sensor-modules

Young Jun Lee, Jihyun Lee, Yong-Bin Kim & Ayers, J. (2005). Low power CMOS adaptive electronic central pattern generator design. *48th Midwest Symposium on Circuits and Systems, 2005.*, *2*, 1350–1353. https://doi.org/10.1109/MWSCAS.2005.1594360

Yu, J., Wang, K., Tan, M. & Zhang, J. (2014). Design and control of an embedded vision guided robotic fish with multiple control surfaces. *The Scientific World Journal*, *2014*. https://doi.org/10.1155/2014/631296

Yu, J., Wu, Z., Wang, M. & Tan, M. (2016). CPG Network Optimization for a Biomimetic Robotic Fish via PSO. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(9), 1962–1968. https://doi.org/10.1109/TNNLS.2015.2459913

Zhang, Y., Wang, S., Wang, X. & Geng, Y. (2018). Design and Control of Bionic Manta Ray Robot With Flexible Pectoral Fin. *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, *2018-June*, 1034–1039. https://doi.org/10.1109/ICCA.2018.8444283

Zhu, D. & Sun, B. (2013). The bio-inspired model based hybrid sliding-mode tracking control for unmanned underwater vehicles. *Engineering Applications of Artificial Intelligence*, *26*(10), 2260–2269. https://doi.org/10.1016/j.engappai.2013.08.017

Apêndice A - Preparação da *Jetson Nano*

A.1 Instalação da imagem do SO

A *NVIDIA* disponibiliza um guia de iniciação para a *Jetson Nano Developer Kit*[96] onde também disponibiliza a imagem mais recente *SDK JetPack* que inclui um sistema operativo baseado no *Ubuntu 18.04*.

Inicialmente seguiu-se esse guia inicial. Foi instalado num cartão *microSD* a imagem *SDK JetPack 32.3.1* utilizando o *software Etcher - Balena*. Posteriormente foi colocado o cartão no *slot microSD* do micro PC para se proceder ao primeiro *boot*.

A.2 Instalação do ROS

Findada o primeiro *boot* foi instalado o *ROS-Desktop-Full* para o sistema operativo *Ubuntu* executando os comandos presentes no código fonte A.1 baseados no guia de instalação oficial do ROS (http://wiki.ros.org/melodic/Installation/Ubuntu). Esta instalação contém os seguintes pacotes: ROS, *rqt, rviz, robot-generic libraries, 2D/3D simulators* e */3D perception*.

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

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

sudo apt update //Installation
sudo apt install ros-melodic-desktop-full

sudo rosdep init //Initialize rosdep
rosdep update

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

sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential //Install Dependencies for building packages
```

[96] Disponível no *website*: http://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit.

```
15
16 mkdir -p ~/catkin_ws/src //Creat the workspace
17 cd catkin_ws/src
18 catkin_init_workspace
```

Código Fonte A.1: Instalação do *ROS-Desktop-Full* na *Jetson Nano* via linha de comandos (terminal).

A.3 Instalação da Biblioteca GPIO

A biblioteca oficial de GPIO é em *python* e é disponibilizada pela *NVIDIA*. Por forma a instalar a biblioteca GPIO utilizou-se o comando *pip*. Seguidamente, criou-se um novo grupo de usuários *gpio*, adicionou-se o utilizador *jetson* ao grupo e reiniciou-se as regras *udev*, necessário para o utilizador aceder aos *pins* GPIO. Em suma, os comandos utilizados foram:

$ sudo pip install Jetson.GPIO

$ sudo groupadd -f -r gpio

$ sudo usermod -a -G gpio jetson # user-name

$ sudo udevadm control –reload-rules && sudo udevadm trigger

No entanto como os *drivers* foram desenvolvidos em *C++* foi necessária a utilização de uma biblioteca *JetsonGPIO*, não oficial, desenvolvida para o ambiente *C++* disponibilizada no repositório https://github.com/pjueon/JetsonGPIO. Por forma a instalar esta biblioteca executou se os seguintes comandos:

$ git clone https://github.com/pjueon/JetsonGPIO

$ cd JetsonGPIO/build

$ make all

$ sudo make install

$ sudo groupadd -f -r gpio

$ sudo usermod -a -G gpio jetson # user-name

$ sudo udevadm control –reload-rules && sudo udevadm trigger

A.4 Instalação de requisitos para utilização de *Arduino*

Por forma a utilizar o microcontrolador *Arduino* foi necessário instalar o *Arduino Integrated Development Environment* (IDE) através do comando:

$ sudo apt-get install arduino arduino-core

$ sudo apt-get install libcanberra-gtk-module

De seguida foi testado o IDE com um *Arduino UNO*, mas sem sucesso: a *Jetson* não reconhecia o *Arduino UNO* na porta de USB. Depois de alguma pesquisa, chegou-se à conclusão que o problema era causado por falta de *Future Technology Devices International* (FTDI) *Drivers*. Para o corrigir foi necessário instalar o *Grinch Kernel* que aumenta os recursos da *Jetson* adicionando suporte a uma ampla variedade de periféricos externos permitindo ainda diferentes parâmetros de configuração (https://www.jetsonhacks.com/2015/05/26/install-grinch-kernel-for-l4t-21-3-on-nvidia-jetson-tk1/):

$ git clone https://github.com/jetsonhacks/installGrinch.git

$ cd installGrinch

$ ./installGrinch.sh

Depois de reiniciar o micro PC foi testado e compilado o exemplo *Blink* com sucesso.

O ROS disponibiliza um pacote de

$ sudo apt-get install ros-melodic-rosserial-arduino

$ sudo apt-get install ros-melodic-rosserial

A.5 Instalação do *OpenCV*

O OpenCv, por defeito, já vem instalado com a imagem *JetPack*, no entanto por forma a utilizar a versão mais recente tornou-se necessário instalar de novo a biblioteca. Foi consultado o guia de instalação do OpenCV para *Ubuntu*[97] e desenvolvido um novo guia de instalação presente no código-fonte A.2.

```
1 // Install OpenCV from the Ubuntu Repository
```

[97] Disponível no *website*: https://linuxize.com/post/how-to-install-opencv-on-ubuntu-18-04/.

```
2
3 sudo apt update // Upadate your system
4
5 sudo apt install python3-opencv // Install OpenCV library for python3
6
7 python3 -c "import cv2; print(cv2.__version__)" // Test the
    installation with python3. This command will check and print the
    version of opencv
8
9 sudo apt install python-opencv //Install OpenCV library for python2
10
11 python2 -c "import cv2; print(cv2.__version__)" // Test the
    installation with python2. This command will check and print the
    version of opencv
12
13
14 // Installing OpenCV from the Source
15
16 sudo apt install build-essential cmake git pkg-config libgtk-3-dev \
17     libavcodec-dev libavformat-dev libswscale-dev libv4l-dev \
18     libxvidcore-dev libx264-dev libjpeg-dev libpng-dev libtiff-dev \
19     gfortran openexr libatlas-base-dev python3-dev python3-numpy \
20     libtbb2 libtbb-dev libdc1394-22-dev // Install the required
    dependencies
21
22 mkdir ~/opencv_build && cd ~/opencv_build
23 git clone https://github.com/opencv/opencv.git // Clone the OpenCVs
    repository
24 git clone https://github.com/opencv/opencv_contrib.git //Clone OpenCV
    contrib
25
26 cd ~/opencv_build/opencv // Create a temporary build directory, and
    switch to it
27 mkdir build && cd build
28
29 // Set up the OpenCV build with CMake
30 cmake -D CMAKE_BUILD_TYPE=RELEASE \
31     -D CMAKE_INSTALL_PREFIX=/usr/local \
32     -D INSTALL_C_EXAMPLES=ON \
33     -D INSTALL_PYTHON_EXAMPLES=ON \
34     -D OPENCV_GENERATE_PKGCONFIG=ON \
35     -D OPENCV_EXTRA_MODULES_PATH=~/opencv_build/opencv_contrib/modules
    \
36     -D BUILD_EXAMPLES=ON ..
37
```

```
38 make -j4 // Start the compilation process. The number 4 represent the
       number of processor. With the command $nproc you can see this value
       , if it's different change it
39
40 sudo make install // Install OpenCV
41
42 pkg-config --modversion opencv4 // Type the following command and you
       should see the OpenCV version
```

Código Fonte A.2: Instalação do OpenCV na *Jetson Nano* via linha de comandos (terminal).

O ROS disponibiliza uma biblioteca chamada *vision_opencv* que utiliza um pacote chamado *cv_bridge*, ponte que faz a interface entre o *OpenCV* e o ROS. Por defeito, quando é instalado o *ROS-Desktop-Full* esta biblioteca também é instalada. No entanto para a versões superiores a 4.0 do *OpenCV*, o caminho definido no *CMakeLists* da *cv_bridge* está erradamente definido e terá de ser alterado. Isto é, no caminho */usr/include/* existe uma pasta chamada *opencv4* (se for a versão 4) no entanto no ficheiro *cv_bridgeConfig.make* que se encontra na pasta */opt/ros/-melodic/share/cv_bridge/cmake* o caminho definido é */usr/include/opencv/*. Para corrigir este problema foi necessário procurar pelas linhas:

```
1 if(NOT "include;/usr/include;/usr/include/opencv " STREQUAL " ")
2  set(cv_bridge_INCLUDE_DIRS "")
3  set(_include_dirs "include;/usr/include;/usr/include/opencv/")
```

E alterar a última linha para:

```
1  set(_include_dirs "include;/usr/include;/usr/include/opencv4/")
```

Caminho para a versão 4 do *OpenCV*, para outras versões deverá ser alterado para o caminho do *OpenCV* em *usr/include/opencvx*.

A.6 Instalação dos *drivers* da Arducam

Os *drivers* adequados pode ser baixado no repositório da *ArduCAM*[98]. No entanto foi necessário determinar qual era o número de distribuição do L4T e do *Kernel*, desta forma recorreu-se aos comandos:

[98]Disponível no repositório: https://github.com/ArduCAM/MIPI_Camera/tree/master/Jetson/Jetvariety/driver.

```
cat /etc/nv_tegra_release // command to determine the L4T release
    number
output: # R32 (release), REVISION: 3.1, GCID: 18186506, BOARD: t210ref,
     EABI: aarch64, DATE: Tue Dec 10 06:58:34 UTC 2019

uname -a // command to determine the kernel number
output: Linux jetson-desktop 4.9.140-tegra #1 SMP PREEMPT Mon Dec 9
    22:47:42 PST 2019 aarch64 aarch64 aarch64 GNU/Linux
```

Baixou-se, portanto, o driver *arducam-nvidia-l4t-kernel_4.9.140-32.3.1 -20200509173033_arm64.deb* onde 4.9.140-32.3.1 corresponde aos outputs obtidos no código acima descrito (4.9.140 - *Kernel* e 32.3.1 - L4T). De seguida procedeu-se à instalação do driver baixado:

$ sudo dpkg -i arducam-nvidia-l4t-kernel_4.9.140-32.3.1-20200509173033_ arm64.deb

Não ocorreu nenhum erro durante a instalação mas, caso acontecesse, deveria ser desinstalado o *driver* com o comando: $ sudo dpkg -r arducam-nvidia-l4t-kernel

Para ver a lista de câmaras disponíveis na *Jetson* e o respetivo caminho foi utilizar comando $ ls -ltrh /dev/video*. Já para testar e verificar as capacidades da câmara foi instalado a ferramenta *v4l2* através do comando $ sudo apt-get install v4l-utils. O código-fonte A.3 é o *output* do comando $ v4l2-ctl - -list-formats-ext, código utilizado para obter os formatos de imagem da câmara CSI que se encontra no caminho */dev/video0*.

```
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'RG10'
        Name        : 10-bit Bayer RGRG/GBGB
            Size: Discrete 3264x2464
                  Interval: Discrete 0.048s (21.000 fps)
            Size: Discrete 3264x1848
                  Interval: Discrete 0.036s (28.000 fps)
            Size: Discrete 1920x1080
                  Interval: Discrete 0.033s (30.000 fps)
            Size: Discrete 1280x720
                  Interval: Discrete 0.017s (60.000 fps)
            Size: Discrete 1280x720
```

```
15                    Interval: Discrete 0.017s (60.000 fps)
```

Código Fonte A.3: *Output* dos formatos de imagem da câmara *Arducam IMX219*.

Para ser possível a utilização do *GStreamer* no driver em ROS foi necessário instalar algumas bibliotecas do *GStreamer* adicionais, recorrendo ao comando:

$ sudo apt-get install gstreamer1.0-tools libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-good1.0-dev

Para ser possível testar a driver foi efetuada uma calibração inicial com recurso ao comando evidenciado no código fonte A.4 e seguindo o guia de calibração para câmaras monocolares[99].

```
1 rosrun camera_calibration cameracalibrator.py --size 8x6 --square 0.108
    ime:=/csi_cam_0/image_raw camera:=/csi_cam_0
```

Código Fonte A.4: *Rosrun* para calibração da câmara com recurso ao pacote *camera_calibration*.

Onde *–size* corresponde à matriz da folha de calibração, que deverá ser imprimida e está disponível na pasta *calibration* do repositório do driver e o *–square* corresponde ao tamanho da área em metros quadrados.

No código-fonte A.5 encontra-se o comando de teste da câmara com interface da *Jetson*. Este código utiliza o *GStreamer* para abrir o fluxo de vídeo da câmara, com largura de 3820 píxeis, comprimento de 2464 píxeis a 21 *frames* por segundo e mostra-o numa janela de largura 1920 píxeis e de comprimento 1080 píxeis.

```
1 gst-launch-1.0 nvarguscamerasrc ! 'video/x-raw(memory:NVMM), width
    =3280, height=2464, framerate=21/1, format=NV12' ! nvvidconv flip-
    method=2 ! 'video/x-raw,width=1920, height=1080' ! nvvidconv !
    nvegltransform ! nveglglessink -e
```

Código Fonte A.5: *Streaming* da *Arducam* com recurso ao *GStreamer*.

[99] Disponível no *website*: http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration.

Apêndice B - Ficheiro de configuração para o GNSS *VMA430* em YAML

```yaml
# Configuration Settings for NEO-7M device

debug: 1                        # Range 0-4 (0 means no debug statements will print)

save:
  mask: 3103                    # Save I/O, Message, INF Message, Nav, Receiver
                                # Manager, Antenna, and Logging Configuration
  device: 4                     # Save to EEPROM

device: /dev/ttyTHS1
frame_id: NEO7M
rate: 1                         # in Hz
nav_rate: 1                     # [# of measurement cycles], recommended 1 Hz, may
                                # be either 5 Hz (Dual constellation) or
                                # 8 Hz (GPS only)
dynamic_model: sea      # Airborne < 2G, 2D fix not supported (3D only),
                                # Max Alt: 50km
                                # Max Horizontal Velocity: 250 m/s,
                                # Max Vertical Velocity: 100 m/s
fix_mode: auto
enable_ppp: false               # Not supported by NEO-7M
dr_limit: 1

uart1:
  baudrate: 9600              # NEO-7M specific
  in: 17                        # RTCM 3
  out: 17                       # No UART out for rover

gnss:
  glonass: true                 # Supported by NEO-7M
  beidou: true                # Supported by NEO-7M
  qzss: true                  # Supported by NEO-7M

dgnss_mode: 3                   # Fixed mode

inf:
```

```
37   all: true                    # Whether to display all INF messages in
       console
38
39 # Enable u-blox message publishers
40 publish:
41   all: true
42   esf: true
43   aid:
44     hui: false
45   nav:
46     posecef: false
```

Código Fonte B.1: Ficheiro de configuração para o GNSS *VMA430* em YAML.

Anexo I - *Pinout J41 da Jetosn Nano*

Fonte: retirado do *website* da *Jetson Hacks*:
https://www.jetsonhacks.com/nvidia-jetson-nano-j41-header-pinout/

Jetson Nano J41 Header					
Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC *Power*	1	2	**5.0 VDC** *Power*	
	I2C_2_SDA *I2C Bus 1*	3	4	**5.0 VDC** *Power*	
	I2C_2_SCL *I2C Bus 1*	5	6	**GND**	
gpio216	**AUDIO_MCLK**	7	8	**UART_2_TX** */dev/ttyTHS1*	
	GND	9	10	**UART_2_RX** */dev/ttyTHS1*	
gpio50	**UART_2_RTS**	11	12	**I2S_4_SCLK**	gpio79
gpio14	**SPI_2_SCK**	13	14	**GND**	
gpio194	**LCD_TE**	15	16	**SPI_2_CS1**	gpio232
	3.3 VDC *Power*	17	18	**SPI_2_CS0**	gpio15
gpio16	**SPI_1_MOSI**	19	20	**GND**	
gpio17	**SPI_1_MISO**	21	22	**SPI_2_MISO**	gpio13
gpio18	**SPI_1_SCK**	23	24	**SPI_1_CS0**	gpio19
	GND	25	26	**SPI_1_CS1**	gpio20
	I2C_1_SDA *I2C Bus 0*	27	28	**I2C_1_SCL** *I2C Bus 0*	
gpio149	**CAM_AF_EN**	29	30	**GND**	
gpio200	**GPIO_PZ0**	31	32	**LCD_BL_PWM**	gpio168
gpio38	**GPIO_PE6**	33	34	**GND**	
gpio76	**I2S_4_LRCK**	35	36	**UART_2_CTS**	gpio51
gpio12	**SPI_2_MOSI**	37	38	**I2S_4_SDIN**	gpio77
	GND	39	40	**I2S_4_SDOUT**	gpio78