ESCOLA NAVAL

talante bien faire

Afonso Jorge Cardoso Lopes

Desenvolvimento do Sistema "Swarm Navigation"

Implementação do modelo "Rolling Leader"

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

Alfeite

2023

Afonso Jorge Cardoso Lopes

Desenvolvimento do Sistema "Swarm Navigation"
Implementação do modelo "Rolling Leader"

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

Orientação de: Víctor Sousa Lobo
Co-orientação de: Bruno Damas

O Aluno Mestrando, *O Orientador,*

Assinado por: VICTOR JOSÉ DE ALMEIDA E SOUSA LOBO
Num. de Identificação: 09302569
Data: 2023.08.29 19:48:08+01'00'

Afonso Lopes Víctor Lobo

Alfeite
2023

"In any moment of decision, the best thing you can do is the right thing, the next best thing is the wrong thing, and the worst thing you can do is nothing."

Theodore Rosevelt

Dedicada aos meus pais, pois estes são o meu porto seguro e à minha irmã, que sempre acreditou em mim e foi sempre o meu exemplo a seguir.

Agradecimentos

A realização desta dissertação de mestrado é o resultado de cinco anos de trabalho, dedicação e sacrifício que contaram com o apoio e suporte de diversas pessoas e, desta maneira expresso o seu reconhecimento por todo este apoio.

Aos meus pais, responsáveis pela pessoa que sou hoje, por me incutirem os valores de responsabilidade, respeito e dedicação, por me apoiarem durante todo o caminho que tenho percorrido e em todas as escolhas feitas.

À minha irmã, que embora fisicamente longe durante esta última etapa, será sempre uma pessoa que posso chamar de amiga, que posso observar como um exemplo a seguir e sei que posso contar sempre com ela.

Ao meu orientador, Professor Vítor Lobo, pelo seu apoio, motivação e ajuda nos momentos de dificuldade em encontrar diversas soluções para os problemas apresentados.

Ao meu coorientador, Professor Bruno Damas, pelo seu apoio, dedicação, motivação e disponibilidade prestada durante a realização deste trabalho e pelas horas extraordinárias que disponibilizava para o auxilio na resolução dos problemas encontrados.

Ao membros da minha classe, Vicente Gomes, Matilde Vieira, Frederico Gonçalves e Francisco Rodrigues, pela relação criada durante todos estes anos e pelo tempo passado durante estes cinco anos.

Aos professores, militares e civis, do Departamento de Ciências e Tecnologia da Escola Naval, por proporcionarem conhecimentos indispensáveis para a realização deste trabalho.

Resumo

A dissertação de mestrado apresentada insere-se no desenvolvimento do atual projeto europeu, *Swarm of Biomimetic Underwater Vehicles*. Este divide-se em duas etapas, tendo sido a primeira etapa representada pela construção de um veículo biomimético e a segunda pelo desenvolvimento de um cardume de veículos de baixo custo.

A segunda etapa do projeto SABUVIS, iniciou-se com a construção do primeiro protótipo de um desses veículos, denominado de Tobias. O foco principal desta etapa é a construção de um modelo de *swarm*, composto por diversos veículos, com capacidades operacionais idênticas ao protótipo já desenvolvido e de baixo-custo. Este modelo tem como foco possibilitar a concretização de operações com uma tipologia benéfica para a Marinha Portuguesa, como a recolha de dados em missões do tipo reconhecimento e vigilância, a monitorização e reconhecimento de determinadas áreas ou a guerra anti-submarina.

O trabalho apresentado pretende realizar o desenvolvimento de um modelo de *swarm*, inicialmente com veículos de superfície, que futuramente possa ser inserido nos protótipos realizados. No modelo construído, são utilizados veículos de superfície diferenciais,isto é veículos apenas com dois veios e sem leme, devido à possibilidade de utilização de um maior número de veículos deste tipo e devido ao menor nível de complexidade perante veículos de sub-superfície.

A construção deste modelo é realizada em diversas etapas como, a preparação e adaptação de um simulador realista já existente, denominado de USVSim através do desenvolvimento de algoritmos de controlo de trajetória e seguimento, com a realização dos seus respetivos testes.

Após esta adaptação, foi concebido um sistema de controlo descentralizado, através de uma ferramenta de comunicações descentralizadas, o Multimaster, sendo os seus testes o controlo da formação dos veículos e a atribuição de tarefas especificas a cada um dos membros do grupo.

Pretendia-se implementar esses algoritmos num pequeno *swarm* de USVs, a construir para o efeito, mas devido a problemas na obtençao no material, tal não

foi possível, sendo o trabalho desenvolvido inteiramente em simulação. Para tal, foi utilizado um simulador realista que opera em ambiente ROS. No entanto foi feito o levantamento do hardware necessário para construção futura desses veículos.

O modelo desenvolvido foca-se na coordenação e planeamento entre os indivíduos do grupo, permitindo a implementação em veículos com características distintas, permitindo um maior número de áreas de atuação.

Palavras-chave: Controlo, Planeamento, Coordenação, Swarm

Abstract

The master's thesis presented is part of the development of the current European project, *Swarm of Biomimetic Underwater Vehicles*. This is divided into two stages, the first stage is represented by the construction of a biomimetic vehicle, the BUV3 and the second by the development of a swarm of low cost vehicles.

The second stage of the SABUVIS project started with the construction of the first prototype of one of these vehicles, called Tobias. The main focus of this stage is the construction of a low-cost model of a *swarm*, composed of several vehicles, with operational capabilities identical to the prototype already developed and at low cost. This model is focused on making it possible to carry out operations with a beneficial typology for the Portuguese Navy, such as data collection, monitoring and reconnaissance of certain areas, anti-submarine warfare, or of a more scientific typology, such as hydrographic surveys or oceanographic studies.

The work presented aims to develop a model of the *swarm*, initially with surface vehicles, which in the future can be inserted in the prototypes made. In the model built, differential surface vehicles are used, i.e. vehicles with only two shafts and no rudder, due to the possibility of using a greater number of vehicles of this type and due to the lower level of complexity compared to sub-surface vehicles.

It was intended to implement these algorithms on a small *swarm* of USVs, to be built for this purpose, but due to problems in obtaining the material, this was not possible and the work was carried out entirely in simulation. To this end, a realistic simulator operating in the ROS environment was used. However, a survey was carried out of the hardware needed to build these vehicles in the future.

The model developed focuses on coordination and planning between individuals in the group, allowing the characteristics of the vehicles where it is implemented not to be seen as a limiting factor.

Keywords: Control, Planning, Coordination, Swarm

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AuRA	**A**utonomous **R**obot **A**rchiteture
BUV	**B**iomimetic **U**nderwater **V**ehicle
CINAV	**C**entro de **I**nvestigação **N**aval
CoCoRo	**C**ollective **C**nognitive **R**obots
EDA	**E**uropean **D**efence **A**gency
EN	**E**scola **N**aval
ESC	**E**letronic **S**peed **C**ontroller
EUSPA	**E**uropean **U**nion **A**gency for **S**pace **P**rogramme
FEUP	**F**aculdade **E**ngenharias da **U**niversidade do **P**orto
GLONASS	**G**lobal'naya **N**avigatsionnaya **S**putnikovaya **S**istema
GNSS	**G**lobal **N**avigation**S**atelite **S**istem
GRPC	**G**oogle **R**emote **P**rocedure **C**all
IA	**I**nteligência **A**rtificial
IMU	**I**nertial **M**easurement **U**nit
L-BAUV	**L**eader **B**iomimetic **U**nderwater **V**ehicle
LAUV	**L**ight **A**utonomous **U**nderwater **V**ehicle
LED	**L**ight-**E**miting **D**iode
LSTS	**L**aboratório de **S**istemas e **T**ecnologia **S**ubaquática
M-BAUV	**M**ember **B**iomimetic **U**nderwater **V**ehicle
NASA	**N**ational **A**eronautics and **S**pace **A**dministration
PVC	**P**olyvinyl **C**hloride
ROS	**R**obot **O**perating **S**system
SABUVIS	**S**warm of **B**iomimetic **U**nderwater **V**ehicles
SAGA	**S**warm of **R**obotics for **A**gricultural **A**pplications
SLAM	**S**imultaneous **L**ocalization and **M**apping
UAV	**U**nmanned **A**erial **V**ehicle
UBEC	**U**niversal **B**attery **E**limination **C**ircuit
UGV	**U**nmanned **G**round **V**ehicle
USV	**U**nmanned **S**urface **V**ehicle
UUV	**U**nmanned **U**nderwater **V**ehicle
VM	**V**irtual **M**achine

WI-FI **Wi**reless **Fi**delity

Capítulo 1

Introdução

O desenvolvimento de enxames de veículos autónomos baseia-se no estudo do comportamento de indivíduos relativamente simples, sem comando estabelecido, que realizam ações e tarefas consoante regras pré-definidas ou eventos exteriores. O estudo destes modelos tem como inspiração sociedades de animais, como por exemplo, enxames de abelhas ou cardumes de peixes, onde se podem observar indivíduos independentes, sem ligação física entre eles, a realizarem movimentos e decisões independentes um dos outros, sendo que cada ação individual leva à concretização de tarefas, que executadas individualmente seriam impossíveis (Cheraghi et al. 2021). Para isso, é necessário perceber a definição de enxame, como sendo um grupo de indivíduos que interage entre si com objetivos comuns. Embora num grupo de indivíduos todos executem tarefas individuais, que primeiramente parecem ser completamente independentes de qualquer outro individuo, uma hierarquia pode ser implementada consoante a importância concedida a cada elemento, que irá delimitar as ações dos restantes (Qin et al. 2017). O uso deste modelo de enxames com veículos autónomos pode ser observado em diversos campos de estudo, como levantamentos hidrográficos, recolha de dados acústicos (Boeck et al. 2014), ou em ambientes de aquacultura (D. Sousa et al. 2019).

1.1 Motivação

Nos últimos tempos, a procura de veículos autónomos não tripulados de menores dimensões tem tido um aumento significativo para a realização de determinadas missões e tarefas em relação a meios como o ser humano ou unidades singulares de maiores dimensões. Esta procura aumentada e preferência de emprego destas soluções autónomas, deve-se a alguns fatores como:

- O menor custo de operação e manutenção relativamente a veículos tripulados, fazendo com que sejam mais vantajosos a nível monetário;

- A facilidade de mudança de sensores consoante a finalidade atribuída ou dependendo da cada missão designada;
- A possibilidade de utilização de elevado número de indivíduos, permitindo uma maior diversidade de recolha de parâmetros;
- A segurança do ser humano, em áreas inóspitas, pela utilização de veículos autónomos não tripulados.

A utilização de enxames de veículos autónomos tem estado presente tanto a nível operacional e de investigação, como a nível pessoal e lúdico, tendo assumido uma elevada importância em determinadas áreas, em situações tais como:

- Drones Kamikaze (Huet 2022);
- Seleção de alvos terrestres de defesa aérea (Johnson 2020);
- Localização e seguimento de alvos dispersos, como lançadores de misseis móveis (Johnson 2020);
- Escolta a navios de maiores dimensões (Hsu 2014);
- Procura cooperativa de alvos usando enxames de veículos aéreos não tripulados (UAV)(Altshuler et al. 2005);
- Utilização em espetáculos visuais (Comittee 2021)(Figura 1.1);

FIGURA 1.1: Demonstração da INTEL

Fonte: Martinez 2016.

Atualmente o número de modelos de veículos não tripulados é bastante elevado, sendo estes empregues em diversos cenários, todos com certas semelhanças mas também com as sua características específicas, espalhados por diversos países, como em Portugal, com o desenvolvimento do LAUV (*Light Autonomous Underwater Vehicle*) pelo Laboratório de Sistemas e Tecnologia Subaquática (A. Sousa et al. 2012). Este tem como objetivo ser um veículo acessível, com elevada operacionalidade que serve como ferramenta de observação e recolha de dados.

1.2 Atualidade

O avanço realizado na área da inteligência artificial, na robótica e nos seus componentes tem acelerado o desenvolvimento e construção de veículos autónomos, permitindo que estes abranjam grandes áreas de aplicação, sendo a sua maior atração a sua possibilidade de executar tarefas eficazmente sem a intervenção humana.

Diversos países estabelecem-se como líderes na construção de veículos autónomos, através de demonstrações da sua tecnologia e dos seus recursos, havendo o aparecimento de dois grandes concorrentes nesta corrida, os Estados Unidos e a China, o que despoleta o aparecimento de uma tensão crescente entre estes dois.

Muitos são os modelos apresentados por estes dois países, podendo ser alguns que demonstram um maior interesse, o veículo autónomo não tripulado dos Estados Unidos, o *Sea Hunter*, desenvolvido pela agência de investigação DARPA no âmbito de investigação do seu programa de veículos de guerra anti-submarina (figura 1.2). É um veículo de 39 metros, com a capacidade de operar durante diversos meses sem uma guarnição (Vincent 2016).

FIGURA 1.2: Veículo não tripulado *Sea Hunter*.
Fonte: Gooley 2021.

Outro modelo de grande interesse para o tema, é o modelo denominado de *Orca XLUUV*, também desenvolvido pelos Estados Unidos (figura 1.3). Como o nome indica, é um veiculo não tripulado subaquático (UUV), com a especial característica da possibilidade de ser modular na sua utilização e construção, com o objetivo de ser empregue em cenários de desenvolvimento e estudo de veículos, trabalhos de integração de cargas MCM (*Mine Countermeasures*) e na luta na guerra anti-submarina, tendo potencial para ser empregue em teor defensivo como ofensivo.

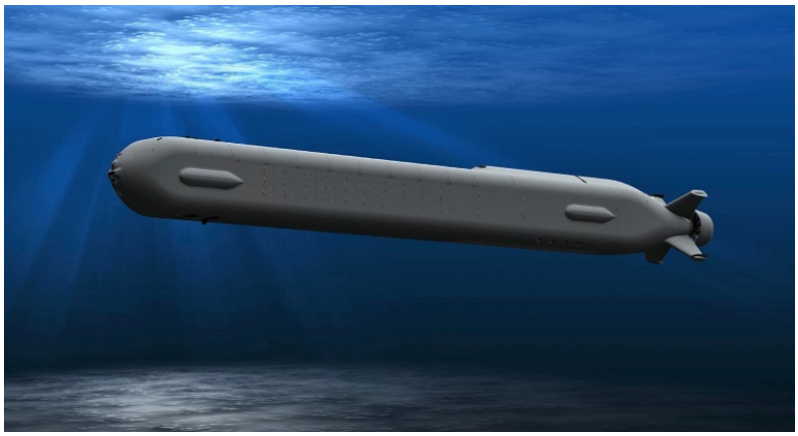


FIGURA 1.3: Veículo não tripulado *Orca XLUUV*.
Fonte: Vavasseur 2022.

1.3 *Swarm of Biomimetic Underwater Vehicles* (SABUVIS)

O projeto *Swarm of Biomimetic Underwater Vehicles* (SABUVIS) é um projeto da *European Defence Agency* (EDA), que tem como principal objetivo a construção de veículos autónomos não tripulados, para aplicação em missões de recolha de dados e de controlo de áreas. De modo a tentar encontrar uma solução para o tipo de missões enunciadas e para a construção deste tipo de veículos, este projeto tem os seguintes objetivos definidos, como referidos por :

1. Desenvolver tecnologia *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 *Unmanned Underwater Vehicles* (UUV) biominético 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.

Este projeto está dividido em duas partes: SABUVIS I, que teve como finalidade a criação de três protótipos de veículos sub-aquáticos autónomos não tripulados, e SABUVIS II, que tem como objetivo a criação e controlo de um cardume de veículos de sub-superfície.

1.3.1 SABUVIS II

A segunda parte do projeto SABUVIS (SABUVIS II), tem como principal objetivo o desenvolvimento de um cardume de veículos de subsuperfície autónomos não tripulados, heterogéneos e fortemente cooperativos, sendo a adaptação a veículos de superfície discutida mais à frente.

O cardume, ou enxame como referido anteriormente, irá ser constituído por indivíduos com diferentes hierarquias. O nível hierárquico mais elevado será o líder, que será o individuo com grande parte da responsabilidade atribuída, maiores capacidades, a nível de tomada de decisão e controlo, e com um maior número de sensores. Será o líder que irá realizar a navegação e coordenação do grupo e receber o objetivo final da tarefa. Para realizar tais tarefas, bem como transmitir informação para os seus seguidores, o líder necessita de sistemas de comunicação de curto e longo alcance. Para além do líder existem os seguidores, que serão os restantes membros do grupo, que irão realizar e executar funções consoante as indicações recebidas do líder. Esta arquitetura será aplicada à movimentação dos elementos, onde o líder seguirá uma determinada rota e os restantes elementos movimentar-se-ão consoante os movimentos designados pelo líder.

O seguimento de uma determinada rota pré-definida consoante uma formatura, realização de manobras e evoluções entre todos os elementos consoante um determinado elemento (líder), são condições necessárias para a consistência, coesão e êxito do grupo. Se todos os pontos referidos forem bem sucedidos, poderemos desenvolver tarefas mais exigentes, como a movimentação propositada de determinados elementos para fora do trajeto planeado com a finalidade de realizar uma tarefa "extra", retomando às posições relativas iniciais.

1.4 Objetivo da Dissertação

O presente projeto a ser desenvolvido está inserido no projeto SABUVIS II, conferido na parte de projeto e desenvolvimento físico de uma plataforma de um modelo de *Swarm*. O desenvolvimento da presente dissertação tem como principais objetivos enunciados seguidamente:

- Seleção do modelo de linguagem e modelos físicos a utilizar;
- Adaptação do modelo físico fornecido por Costa et al. 2016;
- Implementação de um ambiente de simulação inicial individual;
- Implementação de um ambiente de simulação com múltiplos indivíduos e a respetiva coordenação hierárquica;
- Elaboração de tarefas de manobras e evoluções com uma evolução de dificuldade gradual;
- Implementação do modelo de simulação a cada um dos indivíduos;
- Testes e avaliações no mundo real.

Também irá ser desenvolvido um modelo de "*rolling leader*", onde as funções de líder e seguidores não estarão fixas a determinados elementos e não haverá uma cadeia hierárquica, estando todos os elementos no mesmo patamar. Trazendo desta forma uma vantagem ao modelo, pois caso aconteça algo a um dos elementos, o normal desenvolvimento da missão ou tarefa mantém-se, fazendo com que os outros elementos assumam o papel do elemento em falta.

Embora inicialmente estivessem presentes planos para a construção física de um pequeno grupo de veículos de sub-superfície onde iriam ser implementados os algoritmos desenvolvidos, tal não foi possível devido a problemas já mencionados. Consequentemente, todo o trabalho foi realizado em simulação, empregando um simulador realista no ambiente ROS (*Robot Operating System*). Paralelamente, foram identificados os componentes de hardware necessários para a futura construção desses veículos.

É importante destacar que, embora o contexto esteja embutido no projeto SABUVIS, a ênfase principal não recai sobre os veículos de sub-superfície propostos por este. No entanto, o foco está na criação de algoritmos avançados de controlo de formação aplicados a um swarm, com comportamentos de seguimento do líder e implementação de mecanismos de robustez em caso de falha,

1.5 Estrutura da Dissertação

O presente documento pretende demonstrar todas as etapas do desenvolvimento do projeto realizado, estando dividido em seis capítulos. No segundo capítulo, Estado da Arte (capítulo 2), elabora-se uma análise de projetos desenvolvidos na atualidade com elevado interesse para o tema, focando-se em projetos que tenham por base a inspiração em seres vivos ou nos seus comportamentos, através de uma apresentação destes. Posteriormente apresenta-se o Projeto de Hardware (capítulo 3), onde se lista o material necessário para a construção dos veículos, de modo a ser possível aplicar o que foi aprendido no capítulo anterior. Este é seguido pelo Ambiente de Simulação (capítulo 4), onde é feita uma análise de diversos programas existentes na atualidades que servem como ferramenta de apoio à previsão de comportamentos de veículos autónomos. De seguida, temos a Arquitetura de Software (capítulo 5) , onde são demonstrados todos os passos na criação do ambiente de simulação e a criação dos respetivos modelos dos veículos. São ainda apresentados e analisados todos os resultados obtidos na realização dos Testes (capítulo 6). Por fim, apresta-se as conclusões retiradas durante o desenvolvimento deste projeto e sugestões futuras (capítulo 7).

Capítulo 2

Estado da Arte

A presença de veículos autónomos não tripulados no quotidiano tem demonstrado um crescimento acentuado ao longo do século, tanto a nível lúdico como a nível operacional e principalmente na vertente de investigação e recolha de dados. Como tal, a sua utilidade tornou-se uma realidade indispensável em determinado tipo de tarefas, que outrora, poderiam pôr em risco a vida humana, estando presentes em diversas áreas. Os modelos de enxame, tradução da palavra *Swarms*, são criados principalmente pela observação da vida animal, tais como os enxames de abelhas, os cardumes de peixes ou até mesmo as colónias de formigas, sendo esta a base de alguns autores (Cheraghi et al. 2021). Fazendo uma análise do estado da arte e do desenvolvimento destes sistemas, podemos afirmar que atualmente o conjunto de veículos não tripulados com maior popularidade será o de veículos aéreos, também conhecidos por *Unmanned Aerial Vehicles* (UAV), pois têm vindo a despoletar grande interesse na parte do lazer. A utilização de "enxames"de veículos autónomos de superfície em engenharia marítima, também conhecidos por *Unmanned Surface Vehicles* (USV), é caracterizada pela presença de grande números de veículos relativamente pequenos e de baixo custo, ao invés de um único veículo de maiores dimensões, trazendo assim um maior número de vantagens, tanto a nível de obtenção de dados, como a nível monetário (Costa et al. 2016).

Este capítulo procura rever o estado da arte ao nível de enxames de veículos autónomos, nomeadamente os que estão presentes no meio marítimo, dando a conhecer algumas especificidades e características comuns. Essencialmente dar a conhecer os pontos essenciais na formação de um conjunto de USV's.

2.1 Arquitetura dos Sistemas Autónomos

O desenvolvimento da arquitetura de um sistema autónomo aparece inicialmente na década de 80, intitulada de "*The Autonomous Robot Architeture*"(AuRA)(Arkin

1987), que consistia em cinco elementos principais:

- **O planeamento** (*Planning*): elemento responsável pela navegação e pelo objetivo da missão/tarefa;
- **O cartográfico** (*Cartographic*): elemento responsável por armazenar informação recolhida em memória de longo ou curto termo e de a fornecer quando necessitada pelos sensores;
- **O percetivo** (*Perception*): elemento responsável pela recolha da informação do ambiente pelos diversos sensores;
- **O motor** (*Motor*): elemento responsável pela atuação nos meios de interação com o ambiente exterior;
- **O controlo interno** (*Homeostatic Subsystem*): elemento responsável pelo controlo de um ambiente seguro de funcionamento dos meios do individuo.

Este tipo de arquitetura pode ser demonstrada como um cruzamento de diferentes tipo de controlo, que pode ser observado na figura 2.1:

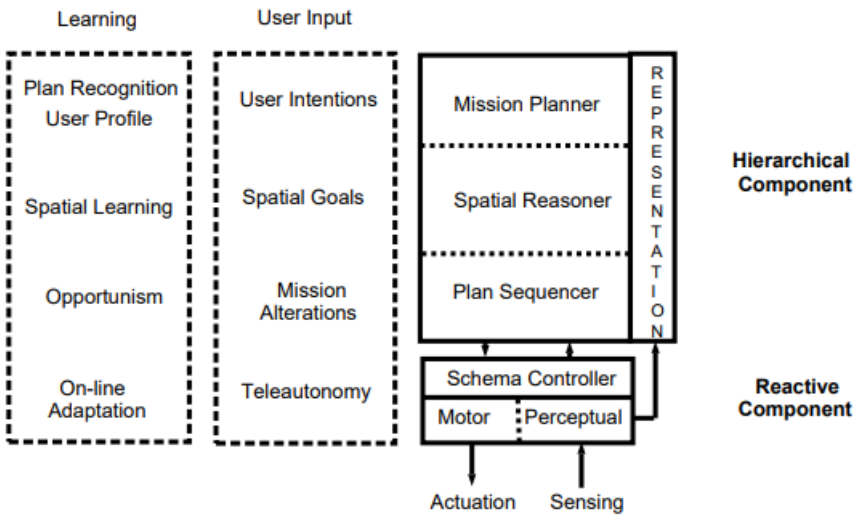


FIGURA 2.1: Arquitetura de AuRA.

Fonte: Arkin e Balch 1970.

2.2 Inspiração na vida animal

O uso e o desenvolvimento de grupos de veículos autónomos têm tido como inspiração a vida animal e as interações sociais ocorrentes entre os seus elementos. Estes modelos baseiam-se nos comportamentos que os seres vivos demonstram perante algumas dificuldades, ou que realizam consoante a finalidade da sua tarefa.

Alguns exemplos que servem de base para o desenvolvimento destes modelos podem ser os enxames de abelhas (figura 2.2a), que com o seu elevado número conseguem proteger a sua colmeia contra predadores, ou as colónias de formigas (figura 2.2c), que conseguem criar estruturas maiores que elas, como por exemplo pontes. Em ambos estes casos, cada ser individual realiza determinadas funções e interage com os restantes elementos consoante as tarefas a realizar, sem terem noção do objetivo final das suas ações. Outros exemplos (figura 2.2), de grandes conjuntos de animais podem ser os cardumes de peixes, que se juntam de modo a tentarem criar uma imagem fictícia de um ser com maior dimensões para proteção, ou um bando de aves que se junta para facilitar os longos voos que realizam (Cheraghi et al. 2021).

(A) Colmeia

(B) Cardume

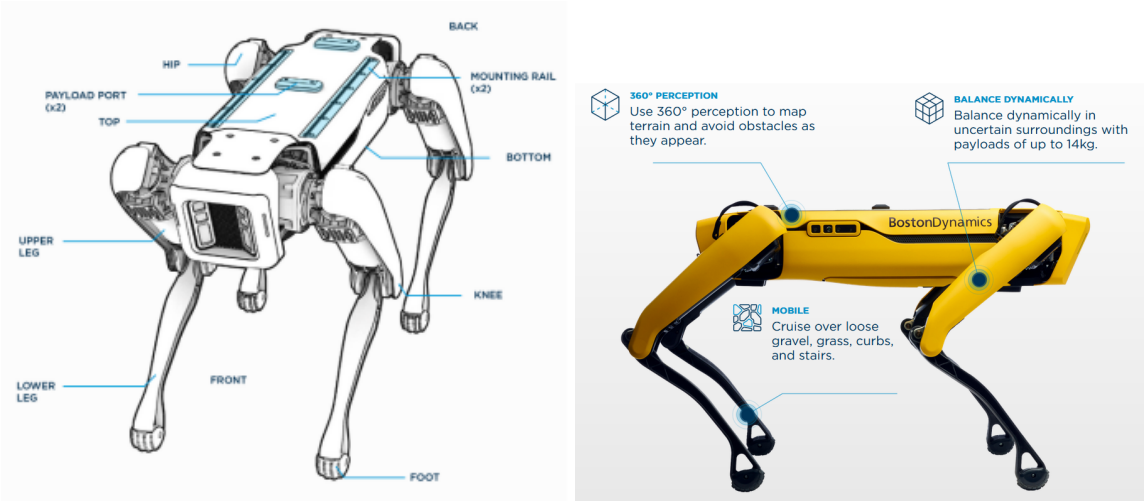
(C) Ponte

(D) Bando

FIGURA 2.2: Exemplos de tarefas realizadas por múltiplos indivíduos

2.2.1 *Spot Classic*

No dia 23 de Junho de 2016, a *Boston Dynamics* anunciou um novo modelo, o *Spot Classic* (figura 2.3), um modelo quadrúpede e com elevada mobilidade, que tem como inspiração seres canídeos, tendo um peso de aproximadamente trinta quilos e uma altura de aproximadamente um metro. Este modelo dispõe de cinco câmaras, duas dianteiras, uma em cada lado do modelo e uma na parte traseira. Relativamente à sua locomoção, este movimenta-se e mexe-se como um cão, tendo articulações flexíveis em todos os quatro apoios (International 2021).



(A) (B)

FIGURA 2.3: Modelo do Spot Classic.

Fonte: Morenville et al. 2022 e *Boston Dynamics*.

Relativamente aos seus sensores, tem um sistema de visualização tridimensional incorporado com *SLAM* e com capacidade de evitar obstáculos, sensores de percepção geográfica incorporados com câmeras visuais, sensores inerciais IMU e sensores de força em todos os membros. O seu software baseia-se numa biblioteca em *Python* e com uma interface de programação baseada em GRPC (*Google Remote Procedure Call*: código de fonte aberta).

2.2.2 *Mini Cyberseal*

O modelo *Mini Cyberseal* (figura 2.4) é um modelo físico de um *Biomimetic Underwater Vehicle* (BUV) que tem como inspiração, a tentativa de reprodução do movimento de um foca a escala reduzida. Este projeto tem como objetivo principal: testar um novo modelo de propulsão com o uso de barbatanas. A construção da *Mini Cyberseal* é baseada num tubo PVC que engloba todos os equipamentos e sensores.

Este modelo está equipado com uma girobússola digital OS-5000 da *Ocean Serve*, um sensor de profundidade A-10 da *Wika*, servomotores controlados pelo POLOLU-1353 miniMaestro e a sua comunicação é realizada via Wifi (Przybylski et al. 2020).

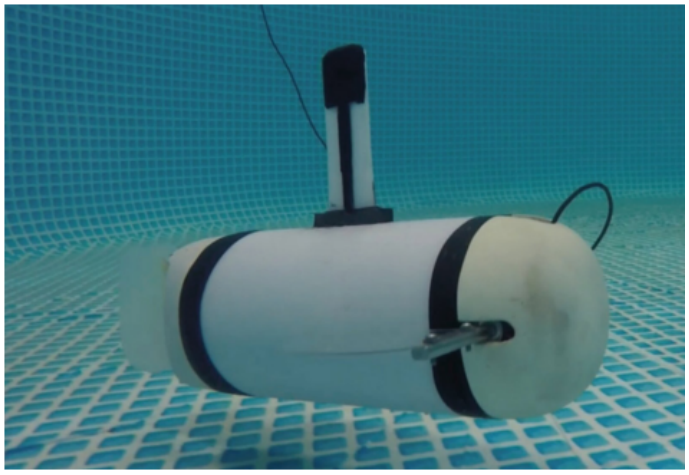


FIGURA 2.4: Modelo da *Cyber Seal*.

Fonte: Przybylski et al. 2020.

2.2.3 AQUA2

AQUA2 é o principal projeto desenvolvido pela *Independent Robotics*. Este modelo é um AUV biomimético que utiliza seis barbatanas de controlo independente que lhe fornece uma elevada mobilidade em ambiente aquático e não apresenta perigos para a vida marinha externa, que pode ser observado na figura 2.5.

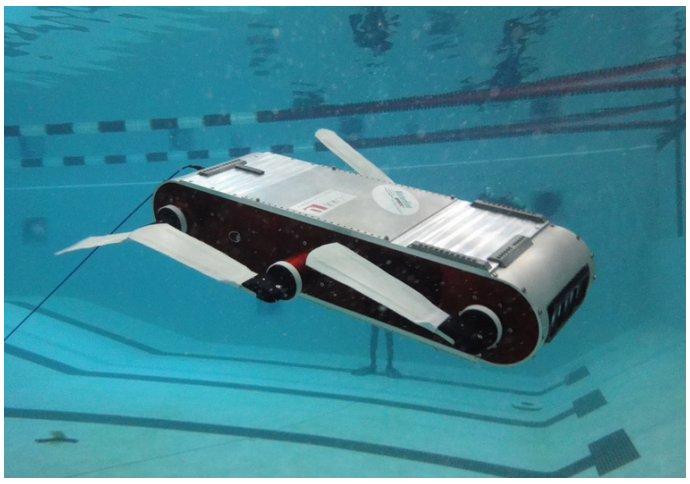


FIGURA 2.5: Modelo AQUA2.

Fonte: IEEERobots 2010.

Este modelo é operado sem plataformas de apoio e pode ser armazenado em contentores de transporte do tamanho de uma mala média de transporte de bagagens

(Tabela 2.1) e pode ser implementado diretamente no meio (AUVAC 2014). As suas missões principais podem ser de controlo de determinada área, inspeção de cascos de veículos maiores, recolha de informação do meio marinho, entre muitas outras.

Características	
Altura	13 cm
Comprimento	64 cm
Largura	44 cm
Peso	16.5 kg
Velocidade	3.6 km/h

TABELA 2.1: Características do AQUA2

2.2.4 *RoboLobster*

Este modelo de uso militar tem como inspiração lagostas e é designado para andar em águas de pouca profundidade e ao longo da costa. Tem como principal objetivo a deteção de minas sub-aquáticas e transmissão de informação. É um dos primeiros modelos construídos que utiliza músculo artificial, denominado de *Nitinol*, proporcionando o seu movimento físico semelhante ao do ser, bem como a resposta a diversas condições do meio(Figura 2.6).

FIGURA 2.6: Modelo RoboLobster

Fonte: Carinecs 2018.

2.2.5 Projeto SABUVIS

Como referido anteriormente o projeto SABUVIS é um projeto que envolve diversos países como Portugal, Polónia e Alemanha, tendo como objetivo principal a criação de BUV's para o emprego em missões de recolha de dados.

No âmbito deste projeto, a Escola Naval (EN) em conjunto com o Centro de investigação Naval (CINAV), com a Faculdade de Engenharias da Universidade do

Porto (FEUP) e com o Laboratório de Sistemas e Tecnologia Subaquática (LSTS) tem desenvolvido diversos modelos de veículos autónomos de superfície e de sub-superfície, com os objetivos presentes anteriormente.

SABUVIS I

A primeira parte deste projeto resultou na construção de três diferentes modelos de BUV's biomiméticos, cada um com características específicas.

BUV 1: veículo representado na figura 2.7, desenvolvido pela Universidade de Cracóvia, teve como inspiração um peixe na sua construção, com duas barbatanas laterais e uma cauda.

Figura 2.7: Modelo do BUV 1.

Fonte: Sutton 2019.

BUV 2: veículo representado na figura 2.8, desenvolvido pela Escola Naval Polaca, teve como inspiração uma foca na sua construção, constituído por duas barbatanas laterais e uma cauda constituída por duas barbatanas de menores dimensões.

Figura 2.8: Modelo do BUV 2.

Fonte: Sutton 2019.

BUV 3: veículo representado na figura 2.9, desenvolvido pela OceanScan, em colaboração com o CINAV e com a FEUP, tendo como inspiração os dois modelos

apresentados para a sua construção, constituído por duas barbatanas laterais e uma cauda que poderá ser constituída por uma ou duas barbatanas.

FIGURA 2.9: Modelo do BUV 3.
Fonte: Araújo 2020.

SABUVIS II

O objetivo principal da segunda parte do projeto SABUVIS é o desenvolvimento da tecnologia para o controlo de um grupo de indivíduos heterogéneos com um elevado grau de cooperação, focando-se num sistema de baixo custo e dimensões. De uma perspetiva técnica, este sistema tem que apresentar as características de um *swarm*, exibindo expansibilidade, flexibilidade e robustez.

Independentemente do tipo de tarefa atribuído ao grupo, é expectável que este apresente pelo menos dois tipos de indivíduos: os líderes, denominados de L-BAUV (*Swarm Leader Biomimetic Autonomous Underwater Vehicle*) e os restantes membros, ou seguidores, denominados de M-BAUV (*Swarm Member Biomimetic Autonomous Vehicle*). Embora se pretenda uma parecença em todos os veículos, os líderes estariam equipados com sistemas de navegação e comunicação e sensores superiores aos restantes membros, de modo a transmitirem apenas informação pertinente relativamente aos seus objetivos aos restantes membros do grupo. Os restantes membros, estariam equipados com sistemas de navegação e comunicação

mais simples e de menor alcance, com objetivo de se focarem apenas no grupo em si.

Manter a formação e coordenação dentro do grupo, são tarefas de elevada importância, e serão nestas que o desenvolvimento do projeto se baseia, pois com uma coordenação sólida entre veículos, há a possibilidade de atribuição de diferentes tarefas a cada veículo, como a patrulha de uma determinada área, com o objetivo de deteção de minas, e posteriormente retornar ao grupo, entre outras.

A fase seguinte deste projeto visa desenvolver os componentes de *software* e o *hardware* para integração e teste em meios físicos.

2.3 *SWARMS*

Como já foi referido anteriormente, pode-se determinar que o conceito geral de um *Swarm* é um conjunto de indivíduos, veículos ou seres, todos com as mesmas características que se agrupam com um objetivo comum e que podem ou não realizar tarefas distintas através de sucessivas interações. Este conceito observa-se maioritariamente na natureza, entre seres com as mesma características e com o mesmo objetivo final, como por exemplo numa manada de animais que migra de uma localidade para a outra, quanto maior for o número de elementos da manada maior é a segurança individual e maior será o número de indivíduos na nova localidade. O mesmo se pode aplicar ao conceito dos *Swarms*, como por exemplo, quando estes são usados como defesa contra alvos aéreos servindo de engodo contra misseis, quanto maior for o número de indivíduos do grupo, maior será a percentagem de sucesso. Como é o caso dos F-35 da Força Aérea dos Estados Unidos usar exames de drones como defesa anti-míssil (Osborn 2021) e dos caças da sexta geração que irão ser vistos mais como uma plataforma do que um veículo, no aspeto em que estes ao serem empregues, irão ser escoltados por diversos veículos autónomos, de menores dimensões, sendo o seu único objetivo a proteção do operador humano.

2.3.1 Características essenciais

A criação de um modelo de *Swarm* no mundo da robótica baseia-se na observação de interações que acontecem entre indivíduos abrangidos no mesmo grupo. Embora esta observação seja um ponto importante para a criação de um modelo posterior, existem certos fatores que têm que ser estabelecidos inicialmente para que o sucesso do grupo seja efetivo.

Estes fatores podem ser estabelecidos relativamente a muitos pontos de foco, como a organização espacial dos indivíduos, os meios de comunicação entre estes ou tarefas atribuídas a cada elemento individual, entre outras. Muitos autores apresentam diversos fatores que consideram relevantes desde um ponto de partida, contudo alguns fatores comuns presentes nos grupos deverão ser (Brambilla et al. 2013; Cheraghi et al. 2021; Barca e Sekercioglu 2013):

- **Robustez** (*Robustness*): ter capacidade de prosseguir com a tarefa imposta mesmo com a perda ou falha de indivíduos. Esta perda poderá ser resultante de mudanças no ambiente externo ou falhas no ambiente interno. Assim sendo, se o grupo é constituído por indivíduos de baixo custo e sem a possibilidade de realizar tarefas individualmente, a falta de um determinado elemento não deve ter um impacto significativo para o resto do grupo;
- **Flexibilidade** (*Flexibility*): ter capacidade de se adaptar ao meio ambiente, às mudanças sucessivas que nele acontecem e às respetivas tarefas implementadas, independentemente do leque de exigências que estas requerem. Para as diferentes tarefas exigidas o grupo deve ter a capacidade de coordenação, para criar diferentes soluções e cooperação, para atribuir diferentes indivíduos consoante as necessidades.
- **Expansibilidade** (*Scalability*): ter a capacidade de operar no ambiente e nas tarefas atribuídas, independentemente do tamanho e número de indivíduos no grupo. A atribuição ou exclusão de indivíduos, não deverá ter um impacto relevante na operacionalidade do grupo. Este deverá manter a capacidade de cooperação em grupos com elevado número de indivíduos e deverá manter a eficiência em grupos com reduzido número de indivíduos.

2.3.2 Tipos de Controlo

Grupos de veículos autónomos com a finalidade de construirem um *Swarm*, têm a tendência de aumentar em tamanho e importância com a constante adição de novos elementos com diversas funções. Incrementando o número de indivíduos num grupo, aumenta o nível de coordenação entre cada um e consecutivamente aumenta também a necessidade de uma elevada eficácia do controlo do grupo.

Relativamente ao controlo de grupos de elevado número de indivíduos, este pode ser implementado de duas maneiras diferentes: 1) através de um controlo centralizado, onde o controlo está atribuído a um indivíduo singular (denominado de líder); ou 2) ou através de um controlo descentralizado, onde todos os indivíduos

têm o mesmo nível de hierarquia, sendo este modelo um pouco mais robusto que o primeiro consoante a sua aplicação (figura 2.10). Segundo (Bayindir 2016), pode-se fazer a seguinte análise entre estes tipos de controlo:

- **Controlo Centralizado**: na aplicação deste tipo de controlo, existe uma unidade ou sistema central que tem o controlo absoluto sobre as ações dos restantes elementos do *Swarm*. Esta unidade normalmente recolhe informação de cada elemento, e toma as decisões ou transmite ordens de maneira a controlar o comportamento do grupo (Iocchi et al. 2001). Algumas características deste tipo de controlo são as seguintes:
  - Toma de decisão global: a unidade central tem uma perceção global do grupo, podendo tomar decisões considerando os objetivos finais do sistema;
  - Atribuição de recursos eficaz: este tipo de controlo permite atribuição eficaz de recursos, pois a unidade central tem completa noção das capacidades de cada indivíduo, podendo otimizar a atribuição de tarefas;
  - Único ponto de falha: a unidade central representa o único ponto de falha do sistema, o que poderá ser considerado um aspeto negativo em diversos casos.
- **Controlo Descentralizado**: na aplicação deste tipo de controlo, o comando e a tomada de decisão estão distribuídos entre todos os indivíduos do grupo. Cada indivíduo pode operar autonomamente, baseando-se em informação e interações com o ambiente local ou outros indivíduos (Chen et al. 2021). Algumas características deste tipo de controlo são as seguintes:
  - Tomada de decisão local: cada indivíduo tomas as suas decisões consoante a sua perceção local e consoante a troca de informação com outros indivíduos nas proximidades;
  - Robustez: este tipo de controlo permite robustez no grupo, fazendo com que a falha de um único indivíduo não leve à falha total do grupo;
  - Coordenação limitada: como neste tipo de controlo os indivíduos apenas se preocupam com o meio e outros indivíduos nas suas proximidades, a realização de maiores tarefas pode apresentar dificuldades.

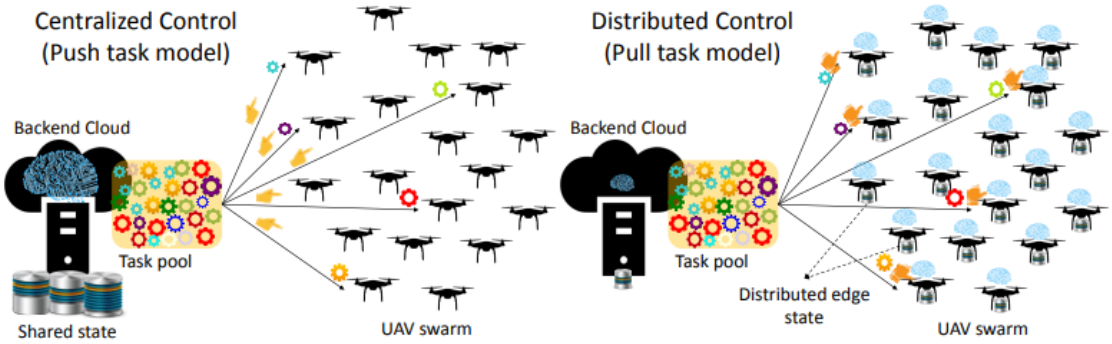


FIGURA 2.10: Diferentes modelos de controlo

Fonte: Hu et al. 2018.

2.3.3 Interações entre indivíduos

Embora a operação de *Swarms* pareça ser um processo quase na sua totalidade autónomo, podendo até ser comparado a um sistema do tipo "*fire and forget*", a interação com um ser humano não deixa de ser um fator essencial no controlo deste sistema. Este é denominado supervisor em vez de operador propriamente dito, pois este não necessita de controlar cada veículo continuamente ou controlar constantemente os movimentos do grupo, necessitando apenas de estabelecer regras e ordens iniciais, havendo assim sempre um elemento de interação com o Humano.

Relativamente às interações entre indivíduos, em diversos projetos, esta é feita utilizando quase sempre o mesmo método (Qin et al. 2017; Liang et al. 2019), podendo ser feita através da atribuição de uma área em torno cada veículo, com uma determinada distância máxima ou mínima, que irá ser denominada como perigo para os restantes veículos (figura 2.11), pois estes podem de certo modo associar o outro USV a um obstáculo, movimentando-se de maneira a evitarem esse obstáculo, evitando que existam colisões entre indivíduos do mesmo grupo.

Outro método utilizado, muito semelhante ao modelo anteriormente apresentado, que também utiliza uma determinada área estabelecida, é o modelo de atribuição de um modelo de uma força repulsiva (figura 2.12), que interage com diversos veículos, que atua no rumo e na velocidade dos veículos caso estes penetrem na área de segurança de qualquer veículo do grupo, sendo consequentemente um modelo eficaz de prevenção de colisões entre indivíduos do mesmo grupo.

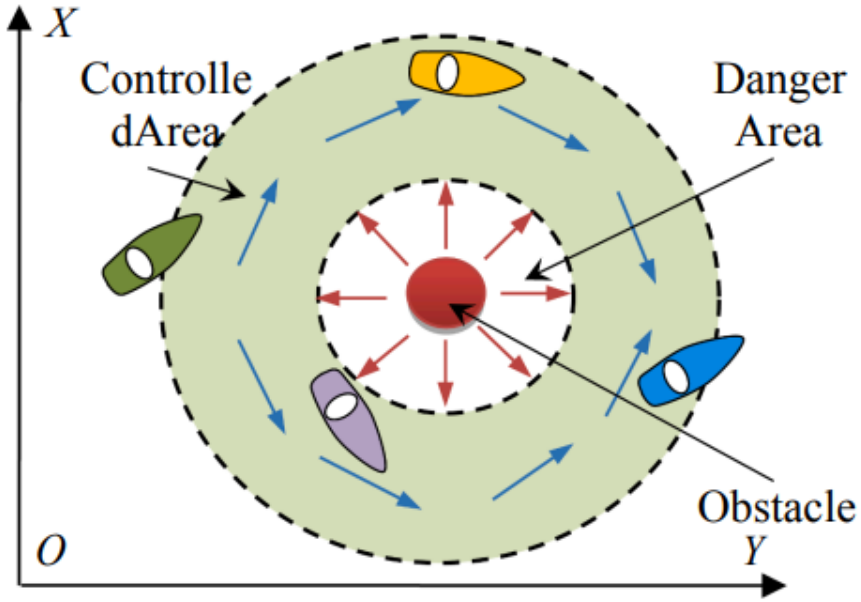


FIGURA 2.11: Área de segurança dos usv's.

Fonte: Liang et al. 2019.

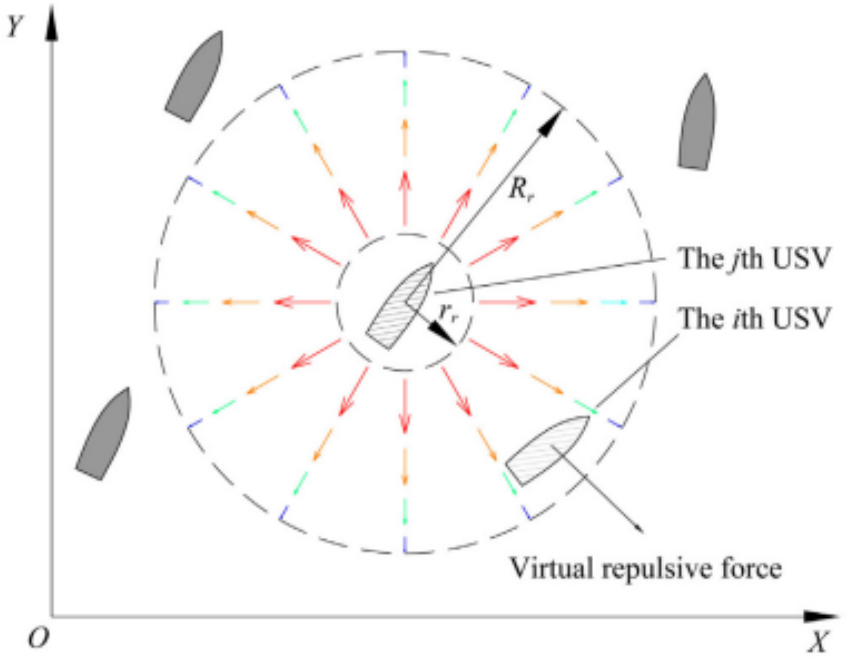


FIGURA 2.12: Área de segurança dos usv's utilizando uma força repulsiva.

Fonte: Qin et al. 2017.

2.3.4 Campos de Aplicação

Atualmente a aplicação de veículos autónomos e de modelos de *Swarm* na robótica está difundida em diversas áreas. A aplicação destes modelos vem com intenção de utilizar meios mais descartáveis e que possam aumentar a segurança do ser humano, permitindo assim que sejam realizadas tarefas com determinados objetivos, que outrora realizadas por meios humanos seriam consideradas impossíveis. Algumas dessas tarefas foram já consideradas, por alguns autores (Khaldi e Cherif 2015 ; Tan 2013):

- Criação de Padrões: capacidade de criação de determinados padrões e formas. Esta aplicação pode ser utilizada para demonstrações lúdicas ou até mesmo transmissão de determinadas mensagens visuais;
- Procura: capacidade de utilizar estes modelos para procurar determinados objetos ou explorar determinadas áreas. A utilização destes meios permite, como por exemplo uma colónia de formiga, procurar algo. Quanto maior e mais distribuídos estiverem os elementos maior será a taxa de sucesso;
- *Scanning* e navegação: possibilidade de fazer *scan* a determinadas áreas e movimentar-se consoante as leituras feitas e dados obtidos. Esta ideia poderá ser utilizada na área de busca e salvamento, onde um conjunto de veículos corre uma determinada área à procura de uma pessoa, fazendo a leitura e navegação no meio constantemente até encontrar algo ou alguém. Um exemplo disto poderá ser a utilização de UAV para identificação de uma pessoa numa floresta, como exemplificado na figura 2.17.

(A) (B) (C)

FIGURA 2.13: Conjunto de Drones a fazer tracking a uma pessoa.

Fonte: Zhou et al. 2022.

2.3.5 Caracterização de grupos

Referindo o tópico anterior, podemos observar que manadas de animais muitas vezes são construídas por indivíduos com características muito semelhantes ou até mesmo idênticas, sendo que isso não é regra absoluta a todos os grupos. Também podem ser observados conjuntos de indivíduos com características muito dispersas ou mesmo indivíduos com qualidades diferentes, tanto a nível visual como a nível objetivo. Assim, segundo Barca e Sekercioglu 2013 podemos fazer uma distinção e avaliação entre grupos homogéneos e grupos heterogéneos, a nível da robótica, apresentando definição breve de cada um:

- Grupo Heterogéneo: constituído por indivíduos com diferentes arquiteturas e funcionalidades, que normalmente se complementam de forma a completar com sucesso as tarefas atribuídas. Poderá ser um grupo constituído por UAV e USV onde os elementos aéreos patrulham e detetam determinadas zonas de investigação e os elementos terrestres investigam essas áreas (Pinciroli, O'grady et al. 2009), como representado na figura 2.14:

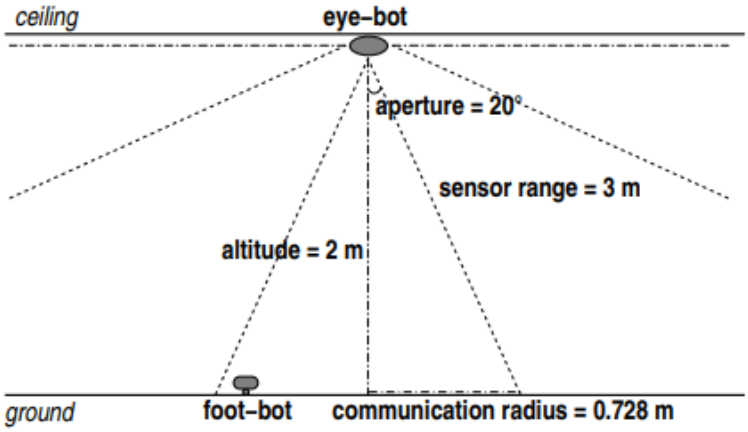


FIGURA 2.14: Grupo heterogéneo com indivíduos aéreos e terrestres.

Fonte: Pinciroli, O'grady et al. 2009.

- Grupo Homogéneo: composto por indivíduos com arquiteturas e funcionalidades semelhantes, como é possível ver na figura 2.15. Isto faz com que a modulação e adaptação entre indivíduos seja facilitada, tornando assim o grupo mais robusto a falhas e permite um maior emprego do grupo devido à aquisição de material idêntico.

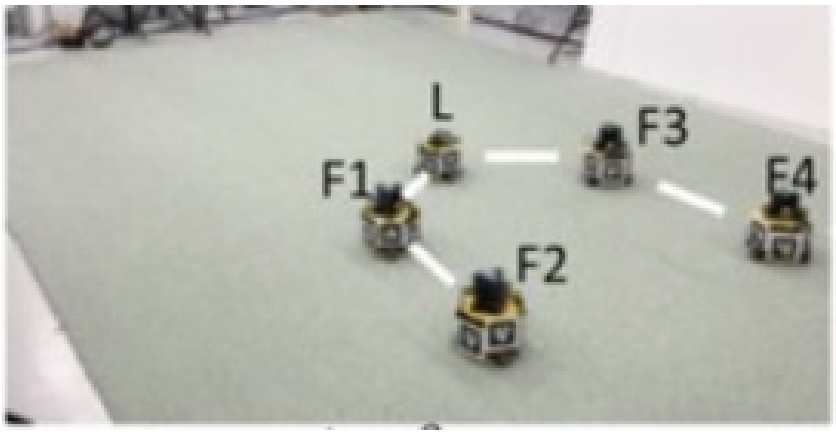


FIGURA 2.15: Grupo homogéneo.

Fonte: Sano et al. 2023.

2.4 *Swarms* desenvolvidos na atualidade

O desenvolvimento de sistemas de *Swarms* encontraram elevado valor em tópicos como problemas de otimização e procura. Como estes são inspirados por comportamentos do meio animal, têm vindo a emergir muitos projetos, criando-se assim algoritmos de otimização de colónias de formigas e otimização de enxames de partículas (Dorigo et al. 1996).

Estes imitam as capacidades de tomada de decisão coletivas e as capacidades de exploração, observados em grupos naturais. Técnicas de otimização de grupos de indivíduos tem sido provadas eficazes na resolução de problemas de otimização complexos, como problemas de logística e localização de recursos.

2.4.1 *Marsbee*

O projeto *Marsbee*, é um projeto desenvolvido pela *NASA* em conjunto com a Universidade do Alabama, com o objetivo de criar um conjunto de indivíduos heterogéneos autónomos (figura 2.16) que facilita a exploração da superfície de Marte (Kang et al. 2019).

Este grupo seria constituído por uma plataforma de maiores dimensões (unidade principal), um rover terrestre (figure 2.17a) que teria como finalidade ser uma estação de recarga e o módulo de comunicação principal, e por indivíduos destacáveis aéreos, as *Marsbee* (figura 2.17b), que teriam como finalidade, a recolha de dados, como imagem vídeo do terreno, recolhida através de uma câmera, permitindo a construção 3D do terreno, para a navegação da base móvel (o rover). Estes

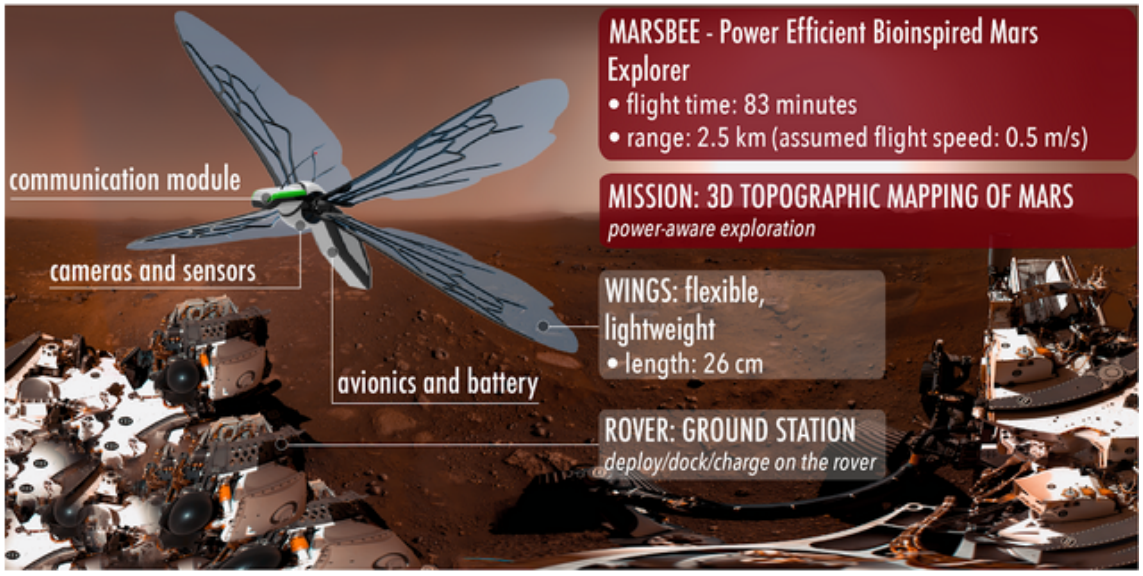
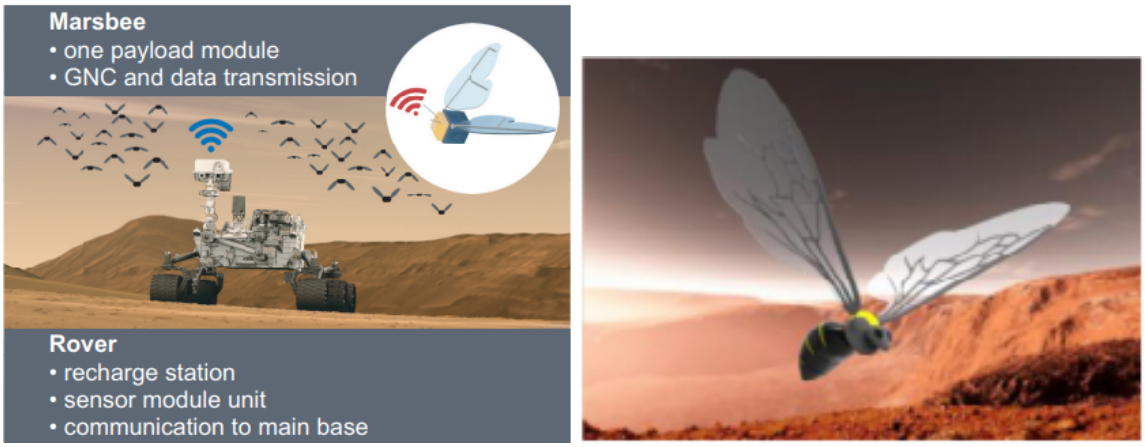


FIGURA 2.16: Modelo do projeto *Marsbee*.

Fonte: Kang et al. 2019.

modelos aéreos poderiam também estar equipados com diversos sensores, que permitissem a recolha da pressão e temperatura do meio, como sensores espectrais, para identificação de minerais.



(A) Rover (B) Modelo aéreo

FIGURA 2.17: Diferentes indivíduos.

Fonte: Kang et al. 2019.

2.4.2 *SWAMR-BOT*

SWARM-BOT é um projeto desenvolvido por Dorigo 2005, na universidade de Bruxelas, que consiste no desenvolvimento de um grupo de veículos autónomos terrestres, denominados de *S-bots*, com capacidades de navegação e perceção física

do ambiente que o rodeia, com capacidade de agarrar objetos e com a capacidade de se interligarem fisicamente (figura 2.18).

Estes modelos movimentam-se através da atuação de rodas e lagartas, permitindo-lhes uma movimentação em ambientes mais agrestes. A base dos modelos é construida por sensores de distância infravermelhos e acelarômetros. Estes também estão equipados com módulos de comunicação, permitindo a comunicação entre indivíduos, como com câmeras omnidirecionais e emissores e recetores sonoros.

FIGURA 2.18: Ligação entre indivíduos.
Fonte: Dorigo 2005.

2.4.3 Projeto SAGA

Segundo (Albani et al. 2017), a agricultura apresenta um domínio grande e em crescimento, constituindo um dos maiores mercados com mais impacto para a aplicação da indústria robótica. Assim sendo, um dos pontos com maior interesse e preocupação é a monitorização de espécies evasivas na manutenção diária das colheitas.

O projeto SAGA (*Swarm Robotics for Agriculture Applications*), usa como base uma plantação de beterraba, que tem como principal preocupação, o aparecimento e crescimento de batatas de colheitas anteriores, pois estas são o principal fator de transmissão de doenças e de destruição dos solos, através da libertação de componentes nocivos, que eventualmente acabam por destruir as colheitas.

Este projeto tem como propósito a monitorização de campos semeados, através do mapeamento de determinadas áreas, passando a informação a sistemas de exterminação terrestre. Esta monitorização e mapeamento é realizado através

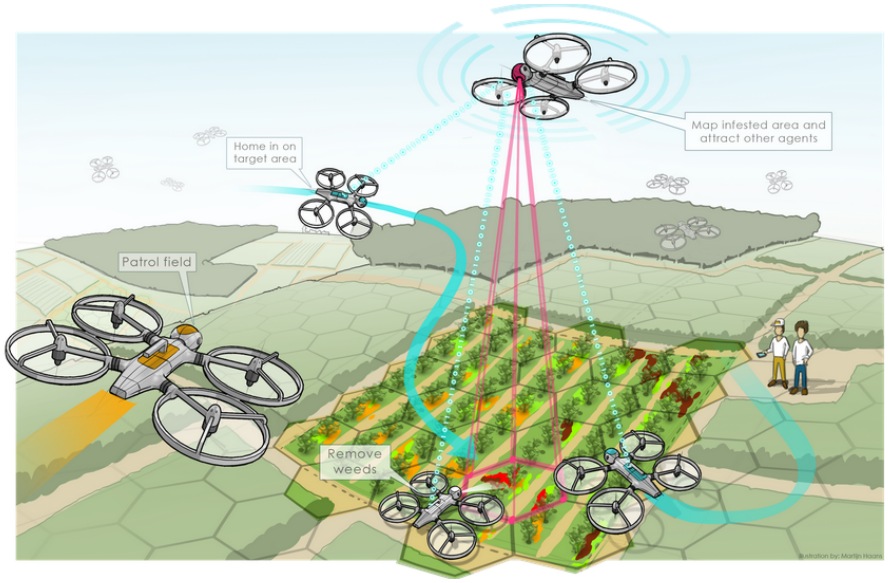


FIGURA 2.19: Esquema do projeto SAGA

Fonte: Albani et al. 2017.

de um conjunto de veículos autónomos aéreos (como representados na figura 2.19), que ao cobrir uma determinada área da colheita, consegue identificar quais rebentos da espécie não desejada (como visto na figura 2.20) e determinar quais as áreas de atuação mais urgente (Cheein e Carelli 2013).

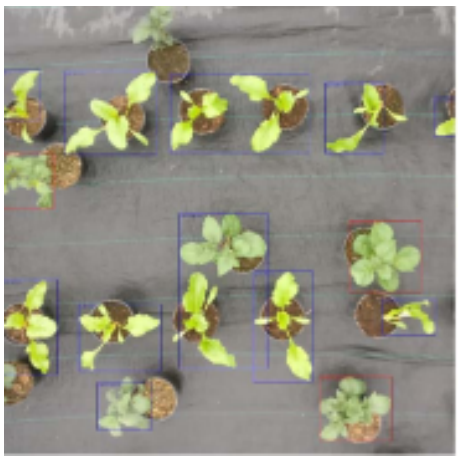


FIGURA 2.20: Deteção das diferentes espécies

Fonte: Albani et al. 2017.

2.4.4 Jasmine III

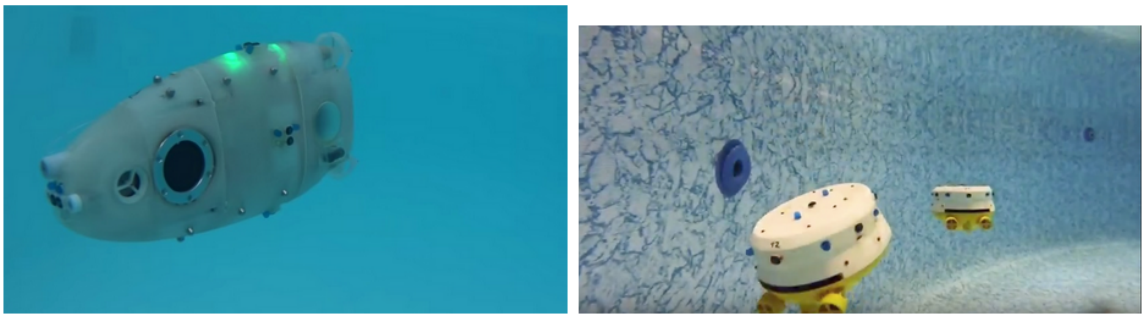
Jasmine III é uma plataforma de hardware e software *open-source* com um tamanho relativamente pequeno, de aproximadamente 3 cm (figura 2.21), com o principal objetivo de possibilitar a criação de um conjunto de indivíduos até mesmo em casa.

Este projeto permite a criação de grupos de 100 ou mais indivíduos, para o desenvolvimento de organização, navegação e agrupamento de múltiplos veículos. É um modelo modular, ou seja, pode ser integrado com diversos sensores consoante a finalidade pretendida (Gerndt e Krupop 2010).

FIGURA 2.21: Modelo singular de um veículo do projeto Jasmine III.
Fonte: Gerndt e Krupop 2010.

2.4.5 Projeto CoCoRo

CoCoRo (*Collecctive Cognitive Robotics*) é um projeto fundado pela União Europeia, que contou com o apoio de vários países, como Áustria, Itália, Reino Unido e Alemanha, que pretende estudar um grupo heterogéneo de veículos bio-miméticos relativamente ao seu comportamento e monitorização de determinadas áreas (Schmickl et al. 2011). Este grupo é constituído por 41 veículos autónomos com diversas características e objetivos. Estes estão divididos em duas categorias, os denominados de ***Jeff***, veículos mais ágeis e rápidos, com o objetivo de navegar e mapear determinadas áreas (figura 2.22a) e os denominados de ***Lily***, veículos de menores dimensões, com o objetivo de funcionarem como veículos de comunicação (figura 2.22b), passando a informação recolhida pelos *Jeff* a um terceiro elemento, a estação-base, que é o elemento de ligação com o grupo e com o ponto de controlo terrestre.



(A) Modelo **Jeff** (B) Modelo **Lily**

FIGURA 2.22: Diferentes modelos.

Fonte: Schmickl et al. 2011.

Ao contrário de outros projetos que utilizam veículos autónomos de subsuperfície, a comunicação e coordenação de indivíduos neste projeto não é feita através de sensores sonares, mas sim através de sensores óticos, através da utilização de LED's, e campos elétricos. Estes permanecem num aglomerado através de flashes de uma luz azul, que são emitidos pela estação-base (figura 2.23).

Como este modelo é coordenado e controlado, principalmente, através de sensores óticos, o meio ambiente tem um grande impacto na eficácia do conjunto, pois meios com grandes alterações de densidade ou de partículas flutuantes, irão afetar drasticamente os resultados pretendidos, daí a utilização de campos elétricos, que limitam a dispersão dos indivíduos no meio, fazendo com que estes não se afastem demasiado da estação base, permitindo uma constante coesão entre indivíduos.

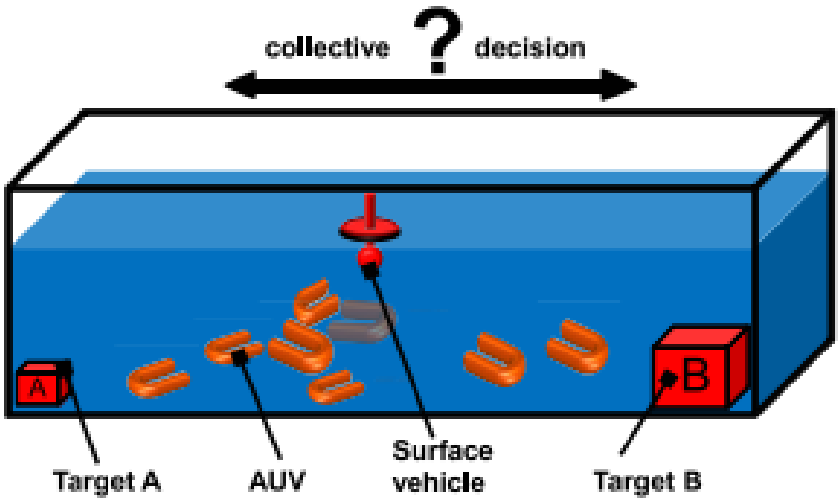


FIGURA 2.23: Dispersão do conjunto no meio, consoante a estação-base

Fonte: Schmickl et al. 2011.

Capítulo 3

Projeto de *Hardware*

A disponibilidade de veículos autónomos na atualidade, é um fator que já não apresenta grandes barreiras para o seu desenvolvimento e aplicação, devido à quantidade de modelos presentes no mercado e à versatilidade que estes apresentam, podendo este também ser um fator limitativo, pois cada modelo apresenta características diferentes apresentando em média um custo relativamente elevado.

O desenvolvimento de múltiplos veículos, de baixo custo e baixa complexidade, são pontos de extrema importância e necessidade para a construção de um sistema de *Swarm*, características tais presentes em diversos modelos apresentados anteriormente e no modelo em estudo.

Assim, neste capítulo é feita uma escolha e análise dos componentes a serem utilizados na construção dos modelos físicos, de modo a manter o foco no baixo custo de construção, estando também sempre presente o fator de fiabilidade que estes apresentam..

3.1 Modelo base

Pela análise e observação realizada no capítulo anterior, é possível recolher informação necessária para a criação de um conjunto de veículos autónomos, possibilitando também a perceção de algumas limitações e possibilidades que estes modelos apresentam.

Como se trata de um modelo inicial de um *Swarm*, as exigências para este são de algum modo baixas, focando-se inicialmente na presença de múltiplos indivíduos no meio, na movimentação individual de cada um e posteriormente na navegação coordenada do grupo, sendo estes os pontos de maior foco relativamente aos veículos individuais.

Relativamente à construção dos veículos autónomos, os modelos a serem utilizados serão baseados no projeto do Instituto Universitário de Lisboa (Costa et al. 2016). Estes modelos, são relativamente simples e de baixo custo, com a possibilidade de modularidade, ou seja, têm a possibilidade de implementar diversos sensores ou equipamentos consoante a sua aplicação e serviram como base inicial para o desenvolvimento do projeto em estudo.

3.2 Características do Casco

O casco utilizado nestes modelos é um mono-casco (figura 3.1), com as características físicas apresentadas na tabela 3.1. O material utilizado para a sua criação é, espuma de poliestireno, que é um material de baixo custo, com boa flutuabilidade, o que permite adicionar componentes mantendo a sua estabilidade. Por se tratar deste tipo de material, permite uma melhor moldagem manual consoante a necessidade de adição de qualquer sensor ou componente extra.

Casco	
Altura	15 cm
Comprimento	65 cm
Largura	40 cm
Peso (total)	3 kg

Tabela 3.1: Características do Casco

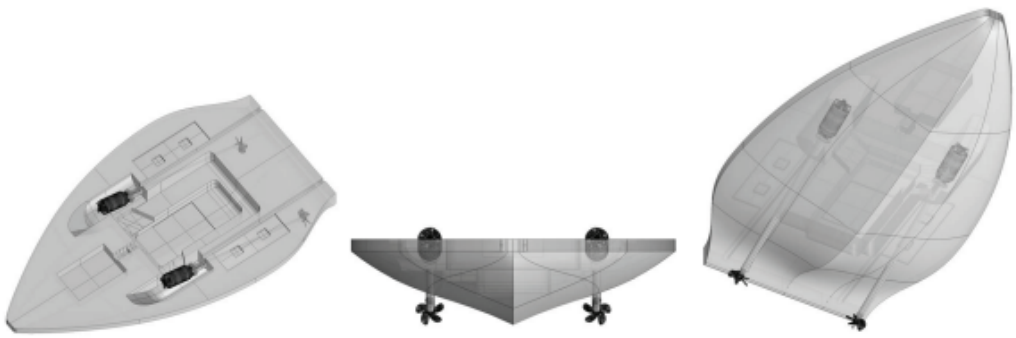


Figura 3.1: Modelo do Casco.

Fonte: Costa et al. 2016.

3.3 Módulo de Propulsão

Nesta secção são referidos todos os componentes pedidos, que atuam na interação do navio com o meio, relativamente ao seu deslocamento.

3.3.1 Motores

Por se tratar de um veículo de superfície que tem por base o controlo do seu movimento através de uma atuação diferencial entre dois veios e duas hélices, devido à ausência de uma pá de leme, a sua movimentação não será realizada através da interação entre o seguimento do navio e a atuação do leme, mas será feita através do controlo do número de rotações e da força aplicada às duas hélices independentemente.

Para possibilitar tal ação, é necessário ter uma atuação em cada veio através de motores individuais. Os modelos propostos são os motores *PROPDRIVE v2 28-30A 800Kv Brushless Outrunner Motor* (figura 3.2).

FIGURA 3.2: Modelo do Motor.
Fonte: HobbyKing s.d.

É um motor construído sem a presença de um veio externo, o que facilita a sua montagem e uso em diversas aplicações, sem modificações de grande impacto. Como a sua construção é realizada de raiz, existe um certo patamar de fiabilidade presente na utilização deste tipo de motor. As características estão apresentadas na tabela 3.2.

3.3.2 ESC

Com a aquisição, montagem e operação de diversos componentes, poderão haver certos problemas que incidem na comunicação e atuação entre componentes,

Características	
Kv (rpm)	800.00
Resistência (mOhm)	161.00
Potência (W)	270.00
Comprimento (mm)	31.00
Comprimento invólucro (mm)	19.10
Corrente máxima (A)	20.00
Tensão máxima (V)	17.00
Veio parcial (mm)	3.18
Diâmetro (mm)	28.00
Comprimento total (mm)	45.10

TABELA 3.2: Características PROPDRIVE v2.

Fonte: HobbyKing s.d.

havendo necessidade de utilização de duas ESC (*Eletronic Speed Control*), que recebem o sinal de comando do microprocessador e que atuam na corrente aplicada aos motores em função desse sinal de comando.

Para a realização desta correta passagem de sinal, será utilizado de um ESC, componente que regula a potência proveniente das baterias para atuação dos motores, tendo que ser escolhido consoante a corrente exigida pelos motores. Assim sendo, o ESC utilizado será o *HobbyKing 50A Boat ESC 4A UBEC* (figura 3.3), que para além do controlo dos motores, tem também uma UBEC (*Universal Battery Elimination Circuit*) de 4A, que fornece uma tensão regulada para a eletrónica utilizada.

FIGURA 3.3: Modelo do ESC.

Fonte: HobbyKing s.d.

3.3.3 Veio flexível e Hélice

O componente final nos veios que permite a movimentação do veículo, serão as hélices, sendo o modelo utilizado, uma hélice de 3 pás metálicas (figura 3.4).

Características	
Diâmetro (mm)	3
Pitch	P1.4
Largura veio (mm)	4

FIGURA 3.4 & TABELA 3.3: Caracteristicas hélice.

Fonte: HobbyKing s.d.

A interação entre os motores e as hélices é feita através de um veio. Para o modelo em estudo, é utilizado um veio flexível, que permite um maior grau de liberdade relativamente à sua construção, e uma maior impermeabilidade do modelo.

3.3.4 Baterias

De modo a permitir uma alimentação a todos os equipamentos propostos, é necessário a utilização de baterias com uma capacidade considerável e que apresentem também um menor tamanho possível.

Deste modo, as baterias escolhidas para utilização serão as ZIPPY Compact 8000mAh 2S1P 30C Lipo Pack (figura 3.5), que apresentam as características enunciadas na tabela 3.4.

FIGURA 3.5: Modelo das baterias.

Fonte: HobbyKing s.d.

Características	
Capacidade (mAh)	8000
Tensão (V)	2 células / 7.4
Peso (g)	390
Dimensões (mm)	166x69x18

Tabela 3.4: Características das baterias.

Fonte: HobbyKing s.d.

3.4 Módulo de Sensores e Comunicações

Nesta secção serão abordados os sensores e os componentes utilizados para a transmissão de dados e comunicação com o veículo.

3.4.1 GPS

Segundo a *European Union Agency for Space Programme* (EUSPA) o GNSS (*Global Navigation Satellite System*) refere-se a uma constelação de satélites que transmite sinais do espaço, fornecendo informação espacial e temporal aos seus utilizadores, que por definição fornece cobertura global.

Alguns exemplos de GNSS são o Europeu *Galileu*, o do Estados Unidos da América *NAVSTAR GPS*, o sistema Russo *GLONASS* (*Russia's Global'naya Navigatsionnaya Sputnikovaya Sistema*) e o sistema Chinês *BeiDou Navigation Satellite System*, retirados do site a EUSPA (EUSPA 2021).

A EUSPA também define quatro critérios essenciais para avaliar o desempenho destes sistemas, sendo estes:

- Precisão (*Accuracy*): a diferença entre posição, tempo e velocidade medida e real recebida por um utilizador;
- Integridade (*Integrity*): a capacidade de um sistema fornecer um determinado nível de confiança, e no evento de uma anomalia, um alarme;
- Continuidade (*Continuity*): a capacidade de um sistema funcionar sem interrupção;
- Disponibilidade (*Availability*): a percentagem de tempo que um determinado sinal satisfaz os critérios anteriores.

Tendo em conta, a disponibilidade material e financeira, o recetor GPS escolhido para ser implementado no veículo será o módulo *GPS NEO-7M-UART* com antena cerâmica, que apresenta as seguintes características presentes na tabela 3.5:

Características modelos GPS		
Modelo	NEO-7N	NEO-7M
Tipo	GPS / QZSS GLONASS	GPS / QZSS GLONASS
Alimentação	2.7V - 3.6V	1.65V - 3-6V
Interface	UART, USB, SPI, DDC	UART, USB, SPI, DDC
Exatidão Gps	2.5 m	
Exatidão GLONASS	4 m	
Temperatura de funcionamento	-40ºC a +85ºC	
Dimensões (mm)	12.2 x 16.0 x 2.4	
Vantagens	Otimizada para limitações financeiras	Melhor desempenho e integração RF

Tabela 3.5: Caracretristicas do GPS modelo NEO-7.

Fonte: *NEO-7 Datasheet* 2015.

3.4.2 Adaptador WiFi

A comunicação e a transmissão de informação será feita através do adaptador USB TP-Link TL-WN722N 2.0 (figura 3.6) é um adaptador Wi-fi de baixo custo, que pode ser ligado a um computador, fornecendo ligações rápidas e fiáveis. A sua antena externa de ganho elevado permite aumentar o alcance e a estabilidade do seu sinal. Juntamente com diversos sistemas operativos, permite a este adaptador ter uma grande versatilidade de aplicações.

Alguns aspetos menos positivos deste adaptador, poderão incluir a sua capacidade de operar apenas na banda de frequências de 2.4 GHz, podendo ser mais propenso a interferências externas, e as dimensões físicas da antena, que poderá ser um fator limitativo na sua aplicação.

Figura 3.6: Modelo do Adaptador Wi-fi.
Fonte: GlobalData s.d.

3.4.3 IMU

Devido a este projeto incidir no uso de veículos autónomos de baixo custo, a IMU (*Inertial Measurement Unit*) escolhida foi uma com 10 graus de liberdade (figura 3.7) e de baixo consumo do site BotnRoll. Embora o seu baixo custo, esta placa demonstra ser fiável na aquisição da dados relativamente à sua posição, à sua altura e à sua temperatura.

Esta IMU está designada para ser utilizada em diversas aplicações, apresentando inúmeras características que a classificam como fiável, incluindo:

- **10 Graus de Liberdade**: esta placa está equipada com um acelarómetro, um giroscópio, um magnetómetro, ambos com 3 eixos de liberdade e um barómetro, permitindo assim calcular aceleração linear, velocidade angular, campos magnéticos e pressão atmosférica;
- **Consumo**: uma das características que leva a esta placa a ser escolhida é o seu baixo consumo, sendo apenas de 3.3 V a 5 V;
- **Precisão**: a tabela 3.6 apresenta os valores máximos e de oepração da placa;
- **Tamanho**: a placa apresenta dimensões compactas, de 20mm x 16mm, permitindo a sua fácil aplicação em veículos de menores dimensões;
- **Integração**: apresenta uma fácil integração com múltiplos microcontroladores e placas de desenvolvimento, incluindo Arduino, Raspberry Pi e Jetson Nano;
- **User-friendly**: a placa está integrada com uma biblioteca que poderá ser aplicada a diversos projetos, permitindo uma recolha de dados inicial rápida.

Especificações	Características
Alimentação	3.3 V - 5 V
Acelarometro	Resolução: 16 bit Alcance: 2g, 4g, 8g, 16g Operação: 450 uA
Giroscópio	Resolução: 16 bit Alcance: 250, 500, 1000, 2000º/s Operação: 3.2 mA
Magnetómetro	Resolução: 14 ou 16 bit Alcance: 4800 uT Operação: 280 uA
Barómetro	Resolução: 16∼19 bit Alcance: 300∼1100hPa (Altitude: 9000m ∼-500m) Precisão: 0.02hPa (0.17 m)

Tabela 3.6: Características da IMU.

Fonte: BotnRoll s.d.

FIGURA 3.7: Modelo do IMU.

Fonte: BotnRoll s.d.

3.4.4 Câmara

Outro sensor que poderá apresentar grandes vantagens para a construção de um modelo de um *Swarm*, é um sensor ótico, pois este permitirá ter uma perceção visual do ambiente e de todos os outros veículos do ponto de vista mais específico.

O sensor escolhido foi uma câmara de pequenas dimensões, representada na figura 3.8.

FIGURA 3.8: Modelo da câmara.

Fonte: BotnRoll s.d.

3.5 Módulo de Processamento

Micro PC's ou micro-processador, são o equivalente a computadores de dimensões relativamente pequenas que são utilizados para o controlo de funções específicas. Estes são tipicamente incorporados em sistemas de maiores dimensões, como projetos de automação ou robótica, podendo ser encontrados numa grande variedade de aplicações, tais como sistemas automóveis, sistemas industriais ou até eletrónica comum, sendo desenhados para serem de baixo custo, versáteis com um baixo consumo energético e com facilidade de utilização, permitindo a sua adaptação a uma grande variedade de projetos (Inc. s.d.).

Figura 3.9: Modelo da Jetson Nano.
Fonte: NVIDIA s.d.

Atualmente existe uma grande variedade de micro-controladores disponíveis ao público, cada um com o seu próprio conjunto de características e propriedades. Alguns dos controladores mais comuns na atualidade poderão ser o Arduino, o Raspberry Pi e o STM32, que são reconhecidos devido à sua versatilidade, custo reduzido e facilidade de utilização (Circuits s.d.).

3.5.1 Jetson Nano

Um micro-controlador que tem vindo a aumentar a sua popularidade nos últimos tempos é a placa desenvolvida pela NVIDIA, a *Jetson Nano*. Este micro-controlador apresenta grande capacidades de computação, sendo o seu foco principal de aplicação inteligência artificial e robótica (NVIDIA s.d.).

Este micro-controlador está equipado com um CPU *quad-core ARM Cortex-A57* e com uma placa gráfica 128-core NVIDIA *Maxwell GPU*, o que permite uma

elevada performance em deep learning e tarefas visuais. Está também equipada com 4 GB de memória LPDDR4, possibilidade de ethernet na casa dos gigabit e possibilidade de integração de múltiplos periféricos (NVIDIA s.d.).

A comparação entre a Jetson Nano e outros micro-controladores existentes atualmente no mercado, pode ser observada na tabela 3.7:

	Jetson Nano	PC/104 Plus	Raspberry PI 4	Latte Panda
GPU	128 Cores NVIDIDA 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 1.5 Ghz	Intel Cherry Trail Z8350 Quad-core 1.8 GGz
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 (C)	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 GPIO's	26-pin GPIO's 6 fichas
Portas	4 USB 3.0, USB 2.0, Micro-USB	3 USB 2.0, 2 RS-232/485	2 USB 3.0, 2 USB 2.0	USB 3.0, 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)

Tabela 3.7: Comparação de vários Micro-Controladores.

Fonte: Araújo 2020.

Capítulo 4

Ambiente de simulação

Recentemente, a atenção para o desenvolvimento de veículos autónomos tem vindo a aumentar entre os grupos de investigação em diversos campos. Um dos principais e primeiros desafios na sua construção e desenvolvimento é assegurar que estes conseguem operar em segurança e eficientemente no meio do mundo real. Deste modo, o ambiente de simulação tem vindo a demonstrar ser uma ferramenta crucial no desenvolvimento de autómatos, principalmente no desenvolvimento de veículos não tripulados autónomos (Corke 2011; Liarokapis e White 2013).

Ambientes de simulação permitem a criação de ambientes virtuais, que simulam, com um elevado detalhe e rigor, cenários do mundo real. Podem ser usados para teste e avaliação do desempenho de veículos autónomos em diversas condições meteorológicas, como tempo limpo e com boa claridade ou até mesmo um tempo mais chuvoso e agreste, como diferentes tipos de meio (aéreo, aquático ou terrestre). Estes ambientes virtuais também permitem realizar uma observação das interações existentes entre os veículos e o meio ambiente, permitindo um ajuste e uma otimização mais facilitada destes.

Algumas principais vantagens dos ambientes de simulação podem ser sua a flexibilidade e a sua relação custo-eficácia. Relativamente à sua flexibilidade, estes permitem uma facilidade de modificação e alteração de parâmetros da simulação, como o tipo de veículo a ser utilizado, o ambiente que o rodeia, a tarefa que este tem que realizar, permitindo assim ter um conjunto de informação relativa a diversas variáveis do desempenho do veículo, possibilitando assim a identificação dos melhores modelos e das duas respostas antes da aplicação no mundo real (Koos et al. 2013). Relativamente à sua relação custo-eficácia, a construção de modelos e realização de testes no mundo real pode ser um processo demorado e com um elevado custo monetário associado, assim sendo, estes meios virtuais diminuem esses aspetos, podendo até mesmo eliminar alguns desses fatores não desejados.

A limitação da presença de material físico e o tempo de espera para a aquisição do mesmo, foi um dos grandes fatores que influenciou a utilização de um simulador para o desenvolvimento deste projeto. Deste modo, neste capítulo é feita uma análise de programas de simulação atuais e uma escolha daquele que indica ser o mais adequado para o desenvolvimento deste projeto.

4.1 Escolha do simulador

4.1.1 Atualidade

Atualmente o grande leque de programas de simulação de robótica permite ter uma maior liberdade de escolha consoante as preferências do utilizador, o nível de precisão esperado, o número de ferramentas necessárias para a operação dos veículos e a facilidade de utilização. Alguns exemplos destes programas que têm uma maior popularidade podem ser:

- *WeBots*: um programa de simulação desenvolvido pela *Cyberbotics Ltd.*, com um ágil ambiente de modelação, programação e simulação, que tem o seu foco principal em veículos autónomos. Inclui bibliotecas internas que permitem a implementação de programas de controlo do próprio utilizador, possibilitando a modificação completa de um individuo, ou até mesmo a possibilidade de modificar e simular múltiplos indivíduos no mesmo ambiente de simulação. Para cada objeto é possível a modulação do seu aspecto fisíco, a sua textura, a sua massa e a sua fricção, entre outros, sendo possível a utilização de qualquer ambiente de desenvolvimento escolhido pelo utilizador (Michel 2004).
- *USARsim*: é um simulador open source, inicialmente desenvolvido pelo exército americano, que pode ser utilizado tanto no âmbito da investigação como no âmbito da educação e ensino. Tem como objetivo principal o emprego de veículos autónomos para a exploração em áreas urbanas e missões de resgate. Este simulador oferece diversas características que o diferencia dos inúmeros simuladores existentes, como a possibilidade de utilizar diversos cenários e a utilização de modelos físicos avançados (Carpin et al. 2007).
- *Gazebo*: é um simulador 3D open source, utilizado para o desenvolvimento robótico, projetos de investigação e no âmbito da educação. Fornece um ambiente virtual que permite simular diversos modelos e as suas interações com o ambiente, com determinados sensores e atuadores. Suporta o desenvolvimento de modelos robóticos complexos, permitindo a simulação de um variedade de

sensores e atuadores, como por exemplo, a utilização de câmeras, *lidars* (Light detection and ranging) e pinças. Suporta também a utilização de diversos modelos físicos, permitindo várias interações físicas, como colisões, fricção e gravidade. Apresenta uma arquitetura extensa e flexível, permitindo aos utilizadores a criação dos seus próprios modelos, fazendo com que este simulador seja uma potente ferramenta.

- *UWSim*: é um simulador open source, modular e de alta fidelidade, para robótica sub-aquática. Fornece um ambiente de simulação sub-aquático e permite que investigadores testem diversas aplicações, como mapeamento sub-aquático, inspeção e intervenção. Este simulador é construído utlizando a linguagem ROS (*Robotic Operating System*) e utiliza o Gazebo como o seu simulador físico (Prats et al. 2012).

4.1.2 Limitações

Existem muitas aplicações na simulação de *Swarms* que têm por base a utilização dos programas acima referidos, aumentando assim o número de oportunidades na criação de ambientes virtuais para este tema. Atualmente a simulação de grupos de veículos autónomos pode ser auxiliada através de diversos simuladores, embutidos já com esta tarefa. Alguns destes simuladores podem ser o *TeamBots* (SCHLEI 2002), o *ArGOs* (Pinciroli, Trianni et al. 2011) ou o *SwarmLab* (Soria et al. 2020).

Embora estas ferramentas sejam um ponto essencial no desenvolvimento de um grupo de indivíduos, pois dão a possibilidade de ter indivíduos com características únicas, que podem ser adaptadas consoante as tarefas atribuídas, conseguindo implementar sensores externos a pedido do utilizar, a grande maioria destes simuladores apenas simulam ambientes com indivíduos terrestres (UGV), aéreos (UAV) ou sub-aquáticos (UUV), havendo uma limitação de simuladores de veículos autónomos de superfície (USV), dificultando assim a escolha do simulador adequado para o presente projeto. Esta limitação deve-se principalmente aos fatores de atuação e interação que cada veículo tem com o meio que o rodeia, isto é, influência de atuações externas no comportamento de cada veículo. Enquanto determinados simuladores apresentam boas representações gráficas do meio ambiente (Koenig e Howard 2004), outros apresentam boas simulações de partículas do meio aquático e permitem a introdução de diversos modelos de veículos (Prats et al. 2012), podendo ser expansíveis, com facilidade de adição de addons ou plugins (Rohmer et al. 2013).

4.2 USVSim

Após a análise das limitações impostas anteriormente, a escolha do simulador teria que se basear num que pudesse integrar características e ferramentas de um determinado simulador, por exemplo a demonstração gráfica e física presentes no Gazebo, com ferramentas de outro ambiente de simulação, como por exemplo as partículas e a interação com o meio aquático presentes no UWSim. Assim o simulador escolhido e posteriormente utilizado foi o USVSim de Paravisi et al. 2019, pois é um simulador que tem presente diversas características e funcionalidades de diversos simuladores e plugins, fazendo a ligação entre estes, de modo a criar um simulador de veículos de superfície que conseguia ter atuações acima, abaixo e no nível do meio aquático.

A arquitetura deste simulador cria uma ligação de troca de informação entre o Gazebo e o UWSim, figura 4.1, sendo as caixas a azul módulos de simulação mais recentes ou personalizados, criando assim a possibilidade de ter um ambiente com veículos que sofram interações com o meio, através da corrente, ondulação e vento. O Gazebo é utilizado como motor principal de simulação, com a adição de dois plugins extras, o *Foil Dynamic Plugin* responsável pelas atuações no casco e do leme, e o *Improved Free Floating Plugin* uma versão atualizada de um plugin existente, responsável pela flutuabilidade, ventos e correntes, enquanto o UWSim é utilizado como meio de visualização principal.

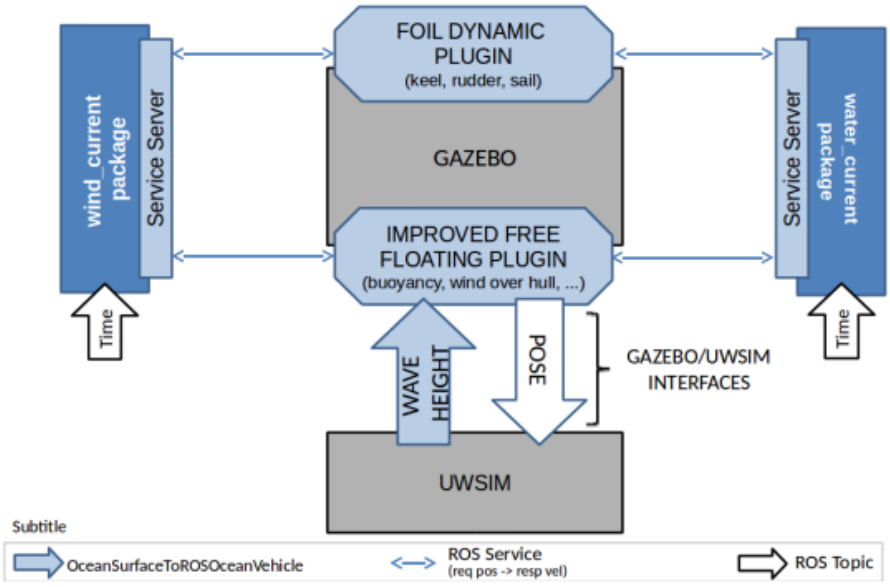


FIGURA 4.1: Arquitetura do simulador USVSim.

Fonte: Paravisi et al. 2019.

Está sumarizada a comparação de vários simuladores na tabela 4.1, relativamente a alguns aspetos e ferramentas necessárias para a criação de um simulador de veículos de superfície, reforçando a ideia de utilização do USVsim, devido às ferramentas presentes neste mesmo. Na tabela constam os pontos de maior relevância para o simulador, sendo estes:

- Ondulação: simulação da ondulação e interação com o veículo (✓✓); apenas simulação visual (✓); incapacidade de simulação (X);
- Flutuabilidade: o veículo segue o movimento da ondulação e atua nos seus graus de liberdade (✓✓); apenas se move consoante o eixo vertical do veículo e da ondulação (✓); sem flutuabilidade (X);
- Corrente: uma força constante num período de tempo e espaço consoante a dinâmica de fluídos (✓✓); uma força constante num período de tempo e espaço(✓); sem atuação da corrente (X);
- Vento: semelhante ao ponto anterior, mas aplicado acima da linha de água;
- Atuação abaixo da linha de água nos propulsores: simula os efeitos dinâmicos das pás dos propulsores (✓✓); atuação apenas de uma força linear no veículo (✓); não aplicável (X);
- Atuação acima da linha de água nos propulsores: semelhante ao ponto anterior, mas aplicado acima da linha de água;
- Atuações no casco: simula atuações de elevação e arrasto no casco do veículo (✓✓); não simula estas atuações (X).

Simulador	Ondulação	Flutuabilidade	Corrente	Vento	Atuação acima da linha de água	Atuação abaixo da linha de água	Casco
UWSim	✓	✓	X	X	✓	X	X
Gazebo	X	X	X	X	✓✓	✓✓	X
Freefloating Gazebo	✓	✓	✓	X	✓✓	✓✓	X
VREP	✓	✓	X	X	✓	✓✓	X
RobotX Simulator	✓	✓✓	X	✓	✓✓	✓✓	X
USVSim	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓

Tabela 4.1: Comparações entre diversos simuladores.

Fonte: Paravisi et al. 2019.

4.2.1 *Foil Dynamics Plugin*

Este plugin é utilizado para calcular os efeitos e as respostas que as forças de sustentação (*lift*) e de arrasto (*drag*) têm nos componentes do veículo, nomeadamente no leme, no casco e/ou na vela, consoante o modelo de embarcação utilizado.

Para estimar essas forças, calcula a velocidade aparente entre os componentes do veículo e o fluído (seja este água ou ar), presentes no trabalho desenvolvido por Paravisi et al. 2019. Embora o plugin original, LiftDrag, calcule as forças tendo em conta apenas a velocidade do veículo, esta versão mais atualizada tem em conta a velocidade do fluído onde este se encontra também, permitindo que o utilizador apenas tenha que indicar as características físicas do modelo e o ponto onde iram ser aplicadas as forças.

4.2.2 *Improved Free Floating Gazebo*

Atualmente, o Gazebo não tem a possibilidade de simular correntes e ondulação, assim este simulador cria um elo de ligação entre o Gazebo e o UWSim, que apresenta uma biblioteca denominada de *OSGOcean*, que permite a simulação de ondulação através da transformada rápida de Fourier. Como esta informação não é passada ao Gazebo, este utiliza um meio de comunicação entre os dois, passando a altura das ondas ao simulador, sendo que cria uma linha com os valores que são passados como a altura da onda. Como a junção destes pontos recebidos apenas cria uma linha, qualquer modelo adicionado só iria estar a respeitar esta altura consoante um ponto definido neste, podendo haver parte do modelo debaixo de água e parte acima, como se pode ver na figura 4.2.

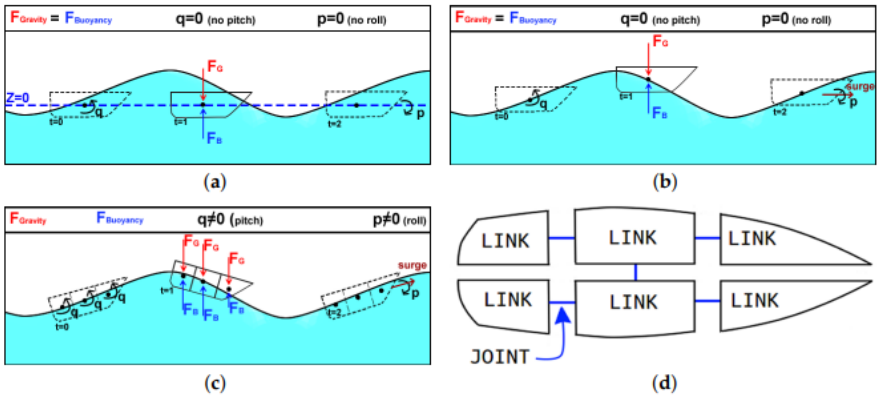


FIGURA 4.2: Representação dos modelos na ondulação.

Fonte: Paravisi et al. 2019.

Assim, os modelos das embarcações são divididos em 6 partes principais, denominados de links, que estão ligados por juntas (*joints*), permitindo uma representação mais real do comportamento dos modelos. Este plugin também adiciona a leitura de múltiplos graus de liberdade aos modelos, como o balanço, cabeceamento e guinada, melhor conhecidos como *roll*, *pitch* e *yaw*.

4.2.3 Modelos das embarcações

O simulador já apresenta diversos modelos implementados, representados na figura 4.3, o que permite uma facilidade de escolha do modelo a utilizar e a possibilidade de múltiplas aplicações.

Os modelos embutidos no simulador são quatro, sendo estes um modelo de um aerobarco, um modelo motorizado com um veio e um leme, um modelo diferencial apenas com dois veios e um modelo à vela com leme.

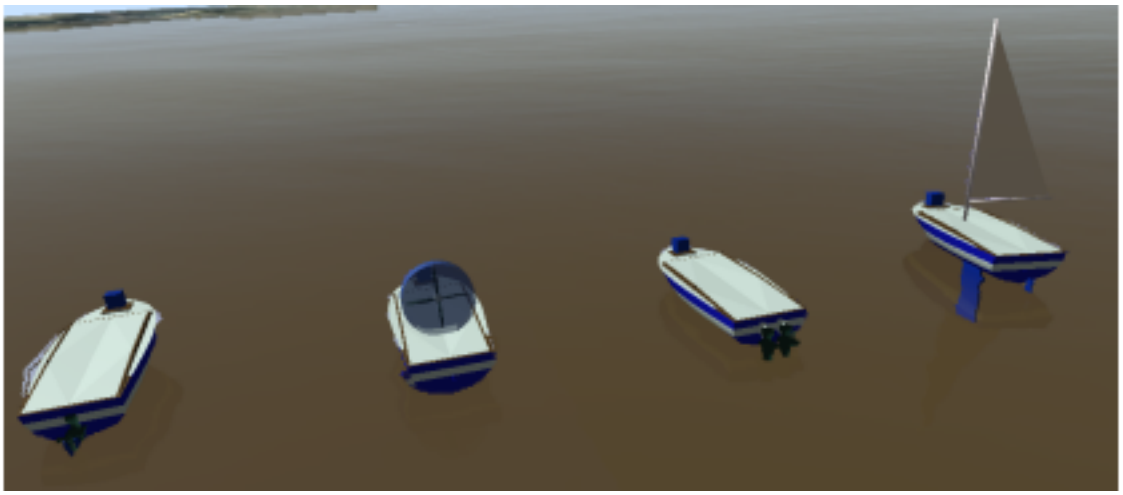


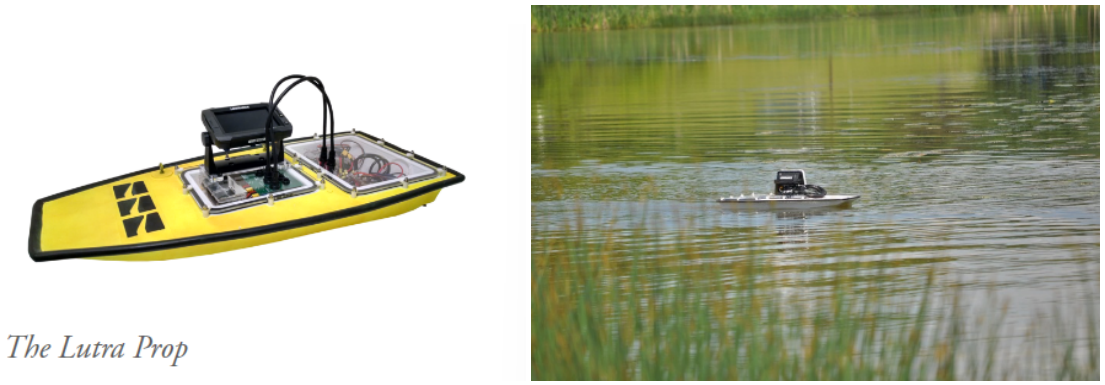
FIGURA 4.3: Diversos modelos implementados.

Fonte: Paravisi et al. 2019.

Existem diversos sensores prontos a utilizar nestes modelos, como sensores de posição, unidades de medição de inércia (IMU) e deteção de alcances, que têm a possibilidade de serem adaptados com múltiplos sensores graças ao Gazebo. Como pode ser observado através da sua página de tutoriais (Gazebo s.d.), podem ser sensores de temperatura, leitores do ph da água, sensores de correntes, implementação de sensores óticos como o lidar, implementação de câmaras, leitores e sensores de binário, sensores térmicos ou até mesmo microfones.

O modelo com maior foco e utilizado neste projeto será o modelo diferencial, pois o modelo físico a implementar será também o de um veículo diferencial de dimensões idênticas à do modelo utilizado no simulador. As características do modelo utilizado no simulador são baseadas num modelo pré-existente, com a nomenclatura de Lutra Prop (figura 4.4), desenvolvido pela Platypus (Platypus s.d.), que pode ser empregue em grandes massas de água parada ou de baixa corrente. Pode ser utilizado em superfícies com apenas 30 cm de altura de água, cobrindo áreas com elevada rapidez e estabilidade, devido ao seu pequeno perfil, o que lhe

permite diminuir o efeito da resistência do ar. Este modelo tem as dimensões físicas representadas na tabela 4.2.



(A) (B)

FIGURA 4.4: Modelo Lutra Prop.

Fonte: Platypus s.d.

Dimensões do Lutra Prop	
Comprimento	106 cm
Largura	48 cm
Altura	45 dm
Peso	9 Kg
Carga extra	3 Kg
Velocidade Máxima	0.67 m/s

TABELA 4.2: Características físicas do modelo Lutra Prop.

Fonte: Platypus s.d.

Capítulo 5

Arquitetura de *Software*

Neste capítulo vão ser apresentados todos os passos desenvolvidos durante a fase de trabalho e utilização do simulador, juntamente com uma descrição da utilização e modificação dos elementos presentes, possibilitando assim a utilização desta aplicação em projetos ou estudos futuros.

A componente de simulação neste projeto apresentou uma elevada relevância, devido à falta de material para a construção e aplicação dos modelos no mundo real. A simulação permitiu desenvolver um ambiente virtual, com veículos com características específicas e aplicação direcionada para o mundo real, permitindo assim que o ambiente de simulação possa ser aplicado de uma maneira quase direta num meio real. O ambiente de simulação também permite realizar alterações sucessivas ao modelos e ao meio, com um impacto temporal e material relativamente reduzido, diminuindo o impacto de uma simulação menos sucessiva a nível comportamental ou de interação com outros veículos ou com o próprio meio.

Assim, um meio experimental de simulação anterior à aplicação física dos modelos, permite-nos aperfeiçoar todos os aspetos necessários, facilitando a implementação posterior dos veículos em determinados ambientes.

5.1 Pré-requisitos

Para possibilitar a utilização do simulador, foi necessário que este fosse instalado no sistema operativo Linux Ubuntu 16.04, visto que este utilizava uma versão mais antiga do ROS (*Robot Operating System*) para correr. A versão do ROS utilizada, é o ROS Kinetic, versão lançada em 2016, tornando-se uma escolha popular na comunidade robótica, graças à sua versatilidade e extensa compatibilidade do múltiplas plataformas.

Esta versão foi desenhada para permitir o desenvolvimento de aplicações modulares e flexíveis, oferecendo uma vasta quantidade de bibliotecas, ferramentas e recursos, fornecendo uma estrutura que simplifica a integração de sensores, atuadores e algoritmos, facilitando a criação de sistemas complexos.

A instalação do simulador foi feita utilizando os passos descritos na página do Github (`https://github.com/disaster-robotics-proalertas/usv_sim_lsa`), onde estão presentes todos os requisitos necessários para a utilização do simulador, como os passos necessários para a sua instalação, que provou ser de nível bastante reduzido e com facilidade de aplicação. Este simulador, inicialmente tem a possibilidade de implementar os quatro modelos de veículos anteriormente descritos (capítulo 4) a três cenários possíveis exemplificados na figura 5.1, todos eles com interações físicas entre o modelo e o meio, sendo estes:

- **Cenário 1**: o veículo navega no meio de um corredor de boias;
- **Cenário 2**: o veículo navega entre três boias evitando colidir com estas mesmo:
- **Cenário 3**: o veículo navega numa determinada área pré-definida.

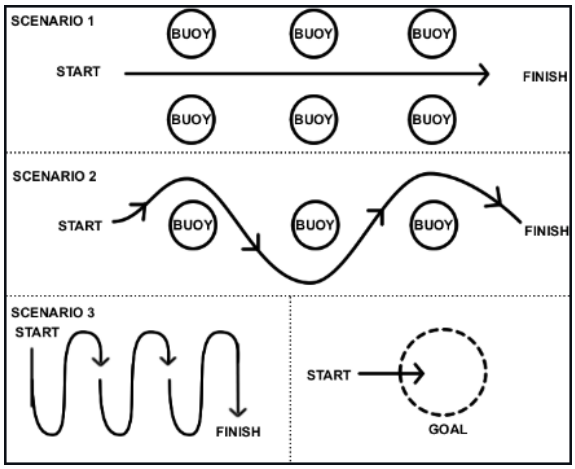


FIGURA 5.1: Diversos cenários do simulador.

Fonte: `https://github.com/disaster-robotics-proalertas/usv_sim_lsa`.

A habituação ao simulador foi um processo demorado devido à limitação de documentação de operação do simulador, o que demonstrou ser um fator de elevada aplicação temporal. Este processo baseou-se assim num modelo de operação de tentativa e erro, tentando perceber de que maneira seria possível alterar os cenários, criar modelos de veículos consoante certas especificações e operá-los no simulador. A a estrutura e o funcionamento dos modelos já existentes também dispunham de uma falta de documentação de apoio à sua utilização.

5.1.1 Criação de um novo ambiente

Após a utilização do simulador estar mais cimentada e ser um processo um pouco mais corrente e de maior facilidade, passou-se à fase de modificação dos cenários e dos modelos dos veículos existentes de modo a moldar o simulador para a aplicação do presente projeto.

As modificações feitas no simulador e nas ferramentas que este apresenta além de terem como objetivo preferências pessoais relativamente ao projeto, também têm como objetivo a implementação de uma linguagem consistente e limpa, de modo a que a sua interpretação, manutenção e compreensão seja facilitada e com possibilidade de reutilização e implementação por futuros membros do projeto.

5.1.2 A criação do novo modelo

Para se começar a criar um ambiente de trabalho que permitisse utilizar todas as ferramentas dispostas pelo simulador, inicialmente foi necessário a criação de um novo modelo de um veículo diferencial, que teria como base as características e funcionamento idêntico ao modelo previamente apresentado no simulador, sendo que para esta ação ter sucesso seria necessário entender a arquitetura do modelo, bem como as interações que existiam entre o simulador e os plugins presentes. Ao iniciar um modelo num determinado cenário, era nos apresentado uma interface gráfica, que exibia o ambiente de simulação com o veículo presente neste (figura 5.2) e a partir daí a simulação era iniciada a partir do Gazebo.

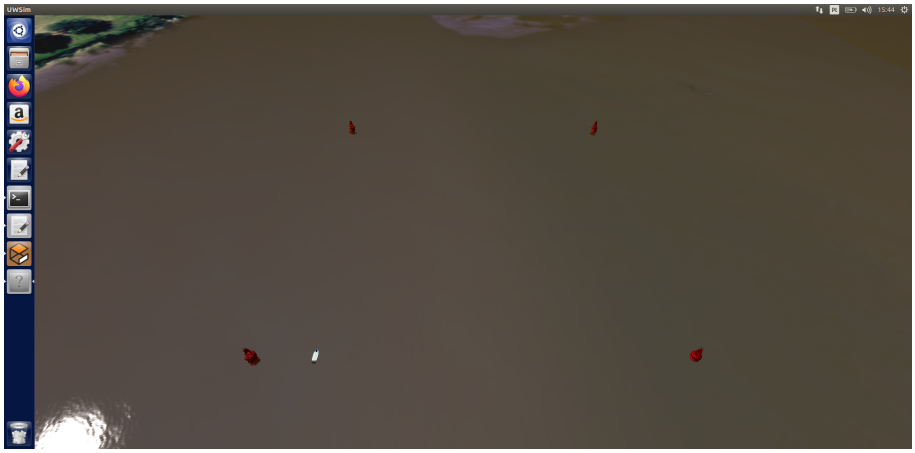


FIGURA 5.2: Interface gráfico do simulador, do cenário 3.

Após correr a simulação, através da ferramenta do ROS, a **rqt_graph**, é possível observar as ligações existentes entre os nós presentes na constituição do veículo, quais os tópicos transmitidos entre estes, e a interação que cada um tem com

os plugins externos (figura 5.3). Pela observação do esquema, foi possível determinar que o modelo do veículo existente (denominado de *diffboat*) estava segmentado em diversos nós essenciais para o seu funcionamento, tais como:

- **/patrol**: nó responsável por transmitir os waypoints para o nó de controlo do veículo;
- **/heading_patrol**: nó com maior impacto no funcionamento do veículo, pois é neste nó que é realizado todo o processamento do controlo do veículo, desde o erro de rumo do veículo com o azimute do waypoint, atribuição de valores de rotação aos motores, apresentando uma elevada possibilidade de falha, pois caso este nó falhasse, não iria haver movimentação do veículo;
- **/gazebo**: nó responsável pelo decorrer da simulação e do cálculo de determinados fatores físicos;
- **/uwsim**: nó responsável pela apresentação da interface gráfica, e pelas interações do veículo com o meio aquático.

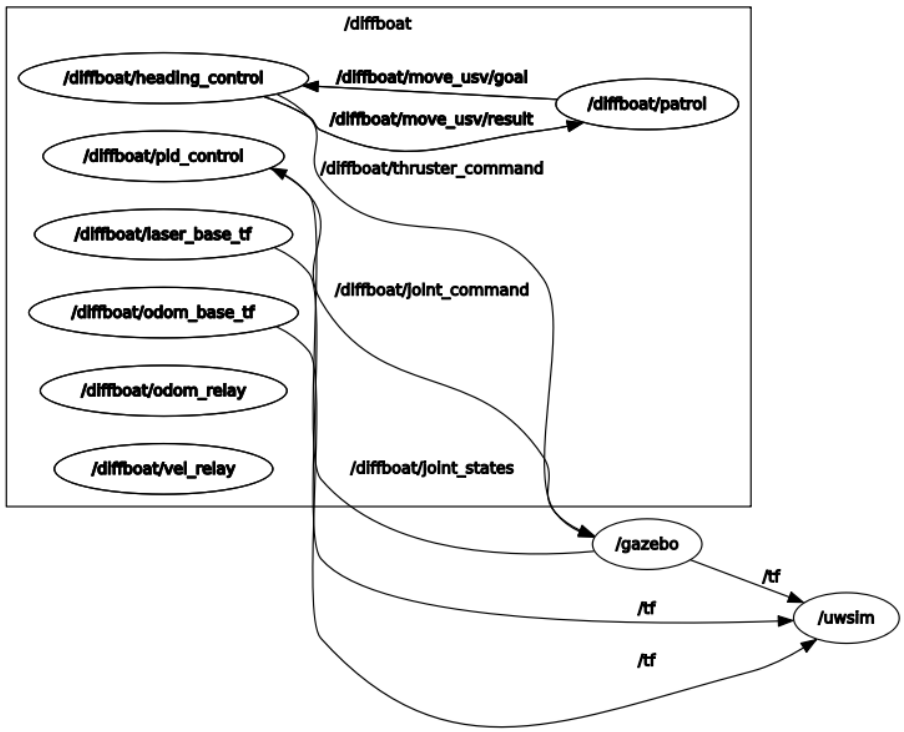


Figura 5.3: Esquema das interações do ROS no modelo diffboat.

Pela observação da arquitetura do veículo, pode-se concluir que esta focava-se essencialmente em dois nós específicos, o responsável pela atribuição e transmissão dos waypoints para o veículos, e o nó responsável por todo o controlo deste mesmo, desde o cálculo do erro de rumo, ao número de rotações a atuar em cada motor de

modo a movimentar o veículo. Esta arquitetura, embora seja relativamente simples, apresentava um problema, tinha uma maior probabilidade de falha devido à sua reduzida taxa de modularidade.

Assim, foi esta mesma arquitetura que foi implementada no novo modelo a ser criado, um modelo que mantivesse à mesma uma simplicidade reduzida, mas que apresentasse uma vertente modular, que permitisse a distribuição de tarefas por um maior número de nós, fazendo com que estes tivessem assim um nível de simplicidade ainda mais reduzido, o que facilita a futura aplicação no mundo real, onde cada veículo tem um módulo especifico para cada componente operacional presente neste.

Com a alteração da estrutura da arquitetura do veículo, também foram alterados os nomes dos tópicos e dos nós de modo a que futuramente, caso seja necessário a utilização do código ou outro tópico, a sua perceção e interpretação seja facilitada, passando a arquitetura do novo modelo a seguinte:

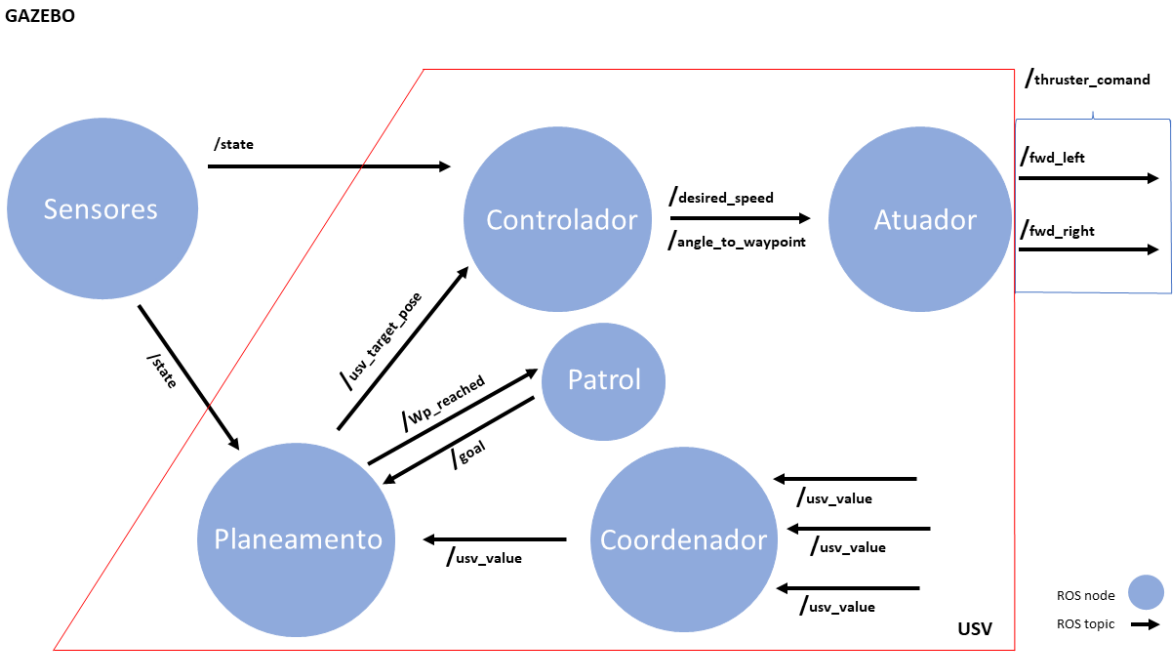


FIGURA 5.4: Esquema das interações do ROS no novo modelo.

Neste esquema simplificado da arquitetura dos novos modelos, este é constituído por quatro nós principais:

- nó de Planeamento: responsável por ir transmitindo o ponto alvo ao veículo, consoante uma lista de waypoints pré-definida e consoante a sua posição no grupo;

- nó de Coordenação: como o nome indica, responsável pela coordenação e atribuição das respetivas posições no grupo dos respetivos veículos;
- nó de Controlo: responsável pelo funcionamento do tipo de controlador do veículo, isto é, pelo cálculo do rumo e da velocidade que o veiculo tem que adquirir;
- nó dos Atuadores: através dos valores passados pelo nó de controlo, é responsável pela cálculo e atribuição das RPM's que cada um dos motores.

Embora esta arquitetura tenha características idênticas à arquitetura de um modelo real, como a presença de nós separados responsáveis pelo funcionamento de cada componentes do veículo, existem mesmo assim fatores que têm que ser alterados caso seja implementada num modelo real tais como a aquisição da sua posição, que neste modelo é realizada de forma direta pelo Gazebo, pois o foco principal deste dissertação não é o desenvolvimento de módulos de navegação. Esta tem que ser realizada através de um único nó que recebe os valores transmitidos pelos sensores presentes de modo a obter uma estimativa do estado do veículo.

Algo essencial ao qualquer veículo autónomo, fazendo parte de um sistema GNC (*Guidance, Navigation and Control*), que no atual modelo é representado pelos nós de Planeamento / Patrol (*Guidance*) e do Controlador (*Control*).

5.2 Nó de Controlo

O funcionamento deste nó é baseado no nó inicial responsável pela execução do controlador do modelo apresentado no simulador, sendo que a grande diferença é a diminuição de tarefas (Anexo A).

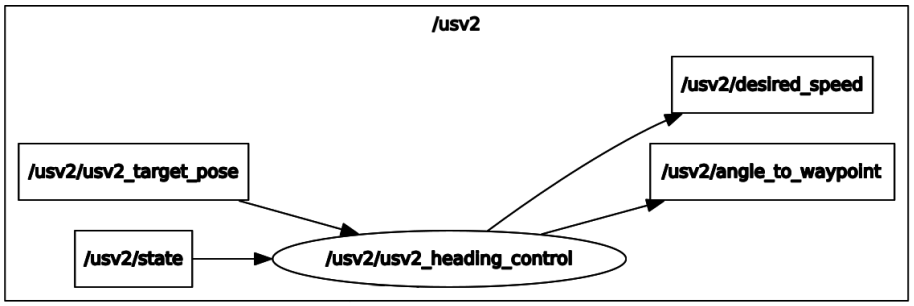


FIGURA 5.5: Tópicos de entrada e saída do nó do controlador.

Este apresenta um número igual de tópicos de entrada e de tópicos de saída, sendo ambos os tópicos de entrada dois tópicos com estrutura semelhante, sendo uma parte a sua posição física (x,y,z) e a restante a sua orientação (*roll, pitch e yaw*). Os tópicos de saída saõ apenas dois valores numéricos, transmitidos através de um valor inteiro, que indicam a velocidade desejada e o ângulo para o próximo waypoint, que iram ser lidos pelo nó dos atuadores.

A grande responsabilidade deste nó, é o cálculo do rumo e da velocidade que o veículo tem que adquirir de modo a deslocar-se para o waypoint que lhe é imposto. Esta ação é realizada através da aquisição da posição atual do veículo, que é transmitida pelo Gazebo através do tópico **/state**, onde é indicada a sua posição espacial (x,y,z) bem como a sua rotação nestes mesmos eixos (*roll, pitch e yaw*), e através da posição do waypoint, que é transmitida pelo nó de patrulha, pelo tópico **/state**, do mesmo modelo que o tópico de posição.

Neste novo algoritmo do controlador existe um desacoplamento entre o controlo do rumo, através da implementação discreta de um controlador do tipo PD e o controlo da velocidade do veículo, através do uso de um filtro de 1ª ordem e uma relação com a distância ao alvo.

O cálculo do erro de rumo é realizado através da equação 5.2.1 e do valor da implementação discreta do controlador é feito pela equação 5.2.2:

$$e(t) = r(t) - y(t) \tag{5.2.1}$$

$$u(t) = K_p \cdot e(t) + K_d \cdot \frac{e(t) - e(t-1)}{T} \tag{5.2.2}$$

Onde:

$u(t)$ é o valor do controlador no instante t.

$r(t)$ é a o ângulo entre o veículo e o ponto alvo no instante t, em graus.

$y(t)$ é o valor do rumo do veículo no instante t, em graus.

$e(t)$ é o erro calculado no instante t.

K_p é o ganho proporcional.

K_d é o ganho derivativo.

T é o intervalo de tempo entre amostras.

No novo controlador é utilizado um controlador do tipo PD para o cálculo do rumo, devido às propriedades benéficas que este apresenta, como a imposição de estabilidade do veículo através de constantes correções de rumo e devido à sua rápida resposta, permitindo que o veículo adquira um rumo mais estável e como menos variações entre wapoints. A mudança do controlador utilizado, de um do tipo PI para um um do tipo PD, deve-se ao facto de tentar mitigar o excesso de desvio relativamente ao planeamento, devido a uma resposta mais lenta do controlador e também pela facto do tempo de resposta de um controlador do tipo PD ser menor que um controlador do tipo PI, apresentado uma maior robustez relativamente a distúrbios externos.

O cálculo da velocidade pretendida para o veículo é feito da seguinte forma:

$$V_{\text{des}} = \alpha \times V_{\text{cruise}} \times f_{\text{dist}}(k) + (1 - \alpha) \times V_{\text{des(k-1)}} \tag{5.2.3}$$

Onde $f_{\text{dist}}(k) = \gamma \| P_{\text{des}(k-1)} - p(k) \|$, onde $P_{\text{des}(k)}$ é a posição do alvo atual e $p(k)$ é a posição atual do veículo.

Nesta equação certas variáveis apresentadas são adicionadas de modo a poder influenciar as variações de velocidade do veículo, principalmente:

- $\boldsymbol{\alpha}$: é um valor entre 0 e 1, que tem como objetivo atuar na velocidade de cruzeiro de modo a que esta não seja aumentada instantaneamente;
- $\boldsymbol{\gamma}$: é o valor atribuído, escolhido entre 0 e 0.1 de modo a atuar na distância que o veículo está do waypoint, fazendo com que a velocidade diminua à medida que este se aproxima cada vez mais do alvo, de modo a evitar que este se desvie do trajeto ideal o mínimo possível.

5.3 Nó dos Atuadores

A criação deste nó, tem como objetivo ter um módulo separado, exclusivamente responsável pela transmissão para os motores dos valores de rotação dos veios (figura 5.6). Estes valores irão ser calculados com base nos valores transmitidos pelo nó de controlo, sendo estes a velocidade pretendida e o erro angular para o waypoint.

Este apresenta dois tópicos de entrada, sendo estes os dois tópicos de saída do nó descrito anteriormente (nó de Controlo), mas apenas um tópico de saída, com uma estrutura matricial de 2x1, contendo dois valores inteiros, que representam as rotação a impor em cada um dos motores. No caso deste projeto, como é realizado apenas em ambiente de simulação, estes valores são lidos diretamente pelo Gazebo com atuação direta nos componentes dos modelos.

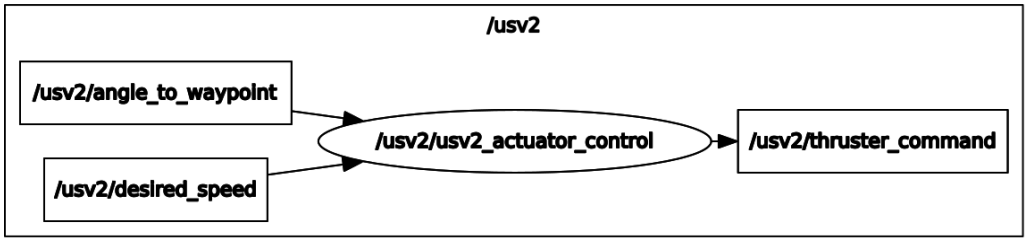


FIGURA 5.6: Tópicos de entrada e saída do nó do atuador.

A atuação no motor da esquerda e no motor da direita é feita através de uma relação linear, por meio do cálculo das rotações a aplicar a cada um destes, utilizando os valores transmitidos para este nó provenientes do nó de controlo, da seguinte forma:

$$motorBB = velocidadecomum + velocidadediferencial$$
$$motorEB = velocidadecomum - velocidadediferencial$$

A mudança de variáveis representada tem como função principal ajustar a velocidade média de rotação dos dois motores em função da velocidade desejada e da velocidade deste, calculada anteriormente no nó de controlo. Está presente também o fator de erro do rumo desejado e rumo atual, que atua sobre o comando de velocidade diferencial.

Ao calcular o rumo do veículo e o ângulo para o waypoint, é utilizada um função que delimita os ângulos que a proa do navio pode ter entre -180º e 180º, para a esquerda e para a direita respetivamente, como demonstrado na figura 5.7 por $\boldsymbol{\alpha}$. À conta desta atribuição de valores, nesta figura dá a entender o porquê de somar

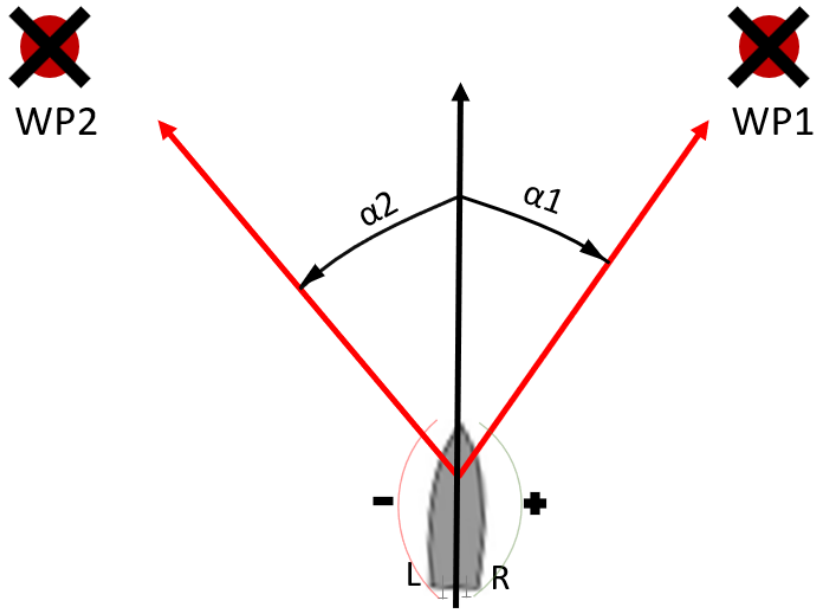


FIGURA 5.7: Atribuição de valores pelo Gazebo.

ou subtrair os valores das velocidade, consoante a posição do waypoint em relação ao veículo.

5.4 Nó de Patrulha

O nó de patrulha, tem como objetivo o armazenamento de um lista de waypoints definida pelo utilizador e ir transmitindo por ordem, os valores da sua posição (x,y,z) à medida que o veículo vai informando que está no waypoint, com as seguintes características:

$$[(281.0, 95.0, 0.0), (0.0, 0.0, 0.0, 1.0)]$$

Onde o primeiro aglomerado de valores representação as suas coordenadas físicas (x,y,z) e o segundo aglomerado representa o quaternião relativo aos ângulos (*roll, pitch e yaw*) bem como a orientação inicial do veículo.

A mensagem que indica que o veículo se encontra no waypoint, é transmitida através do nó de planeamento, que perante uma determinada distância estabelecida, assume que o veículo se encontra no waypoint, e transmite o valor de 1, através do tópico **/waypoint_reached**, que permite ao nó de patrulha começar a transmitir o próximo waypoint. Posteriormente foi adicionado um valor booleano de mudança de valor, o que aumenta a eficácia do nó, pois caso houvesse alguma falha na passagem

de tópicos, haveria sempre uma segunda maneira e permitia também a adição do mesmo tópico de outros veículos.

Este nó pode ser separado do lançamento inicial do veículo, de modo a que futuramente possa ser aplicado a uma estação de controlo, que estaria responsável pela transmissão dos waypoints a cada veículo independentemente, permitindo haver sempre um elo de comunicação entre os veículos e uma estação com um supervisor.

Relativamente aos tópicos subscritos e publicado, este publica o tópico **/goal** com o formato apresentado anteriormente para ser lido pelo nó de planeamento, e subscreve o tópico **/waypoint_reached** de cada um dos veículos de modo a proceder à transmissão do waypoint seguinte.

5.5 Nó de Coordenação

Uma das características mais importantes num *Swarm* de veículos autónomos, é a presença de um fator vinculativo entre todos os veículos de modo a poderem ser denominados de um grupo e não apenas indivíduos singulares independentes.

É exatamente neste fator que este nó se baseia, na criação de um canal de comunicação entre os veículos do mesmo grupo, de modo a acrescentar um elemento de coesão a este mesmo (figura 5.8). Este tem como objetivo a criação de uma ligação de hierarquia entre os indivíduos do grupo, de forma a criar uma coordenação e com esta, a atribuição de tarefas consoante o seu nível hierárquico e a possibilidade de implementação do modelo de *rolling leader*.

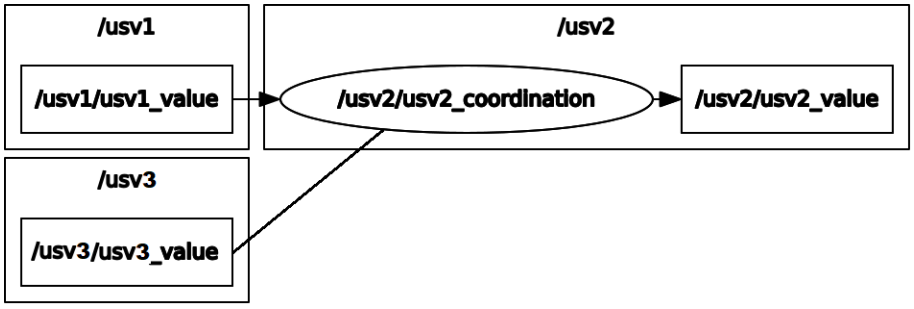


Figura 5.8: Tópicos de entrada e saída do nó de coordenação.

Relativamente aos tópicos de entrada e saída deste nó, em cada um dos nós de cada um dos veículos, este apresenta dois tópicos de entrada, que são as posições relativas dos restantes veículos do grupo, através do tópico **/usvX_value** e calcula e transmite a posição do repsetivo veículo, através de um número inteiro, que irá

ser lida e subscrita pelos restantes nós de coordenação e planeamento dos restantes veículos.

A comunicação entre veículos é feita através da transmissão de uma mensagem, através de um tópico de ROS, por parte de todos os veículos com o objetivo de indicar que estes estão a funcionar, como uma mensagem de a indicar que estes estão vivos. A mensagem que cada um transmite é recebida por todos os veículos do grupo, como indicado na figura 5.9.

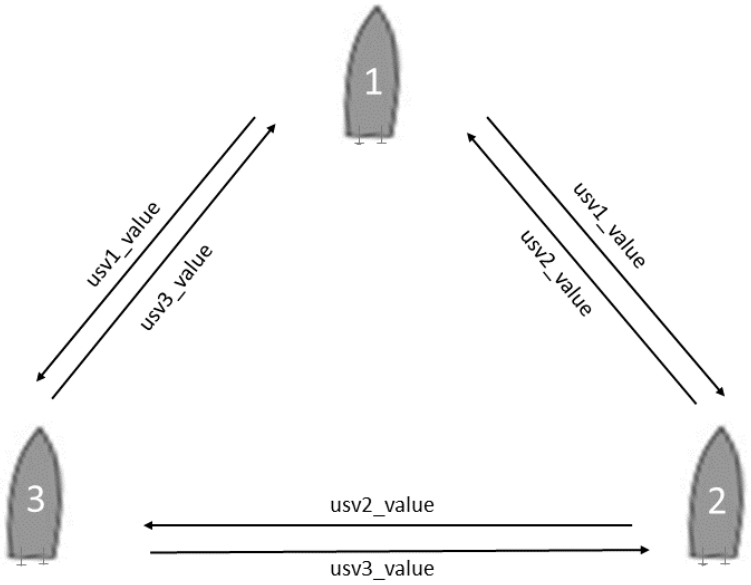


FIGURA 5.9: Comunicação entre indivíduos.

A receção e transmissão desta mensagem permite que cada veículo calcule a sua posição dentro do grupo, isto é, consoante as mensagens que cada um recebe, consegue atribuir um valor a ele próprio e consoante esse valor, realizar determinadas tarefas ou neste caso, colocar-se numa posição relativa aos restantes elementos do grupo. O cálculo do valor da posição do grupo é feito através das mensagens recebidas, começando o líder inicial a transmitir a mensagem, que é apenas o valor 1, e consoante a hierarquia, cada veículo assume a sua posição.

Partindo de que a hierarquia do grupo está definida pela seguinte ordem: USV1, USV2, USV3, etc, o cálculo e atribuição da posição dentro do grupo permite uma ordenação e reordenação em caso de falha de algum veículo, através do seguinte modelo:

- Pos_usv1 –> Líder inicial –> Transmite o valor de 1

- Pos_usv2 = máx(Pos_usv1) + 1 –> A sua posição no grupo passa a ser a posição 2
- Pos_usv3 = máx(Pos_usv1 , Pos_usv2) + 1 –> A sua posição no grupo passa a ser a posição 3, pois será o valor da Pos_usv2 incrementada de 1
- Pos_usv4 = máx(Pos_usv1 , Pos_usv2 , Pos_usv3) + 1 –> A sua posição no grupo passa a ser a posição 3, pois será o valor da Pos_usv3 incrementada de 1

Esta ordenação simplesmente recebe os valores atribuídos aos veículos anteriores, destacando o maior valor recebido. Após ter determinado esse maior valor, incrementa o valor de um, de modo a atribuir essa nova posição ao respetivo veículo.

Ao atribuir esta esquemática ao grupo, é nos possibilitado a integração de um número ilimitado de veículos, tocando assim num ponto importante nas características básicas de um *Swarm*, a expansibilidade. A atribuição deste modelo permite a aquisição de posições de um nível hierárquico mais elevado, caso exista a falha de veículos no grupo (5.10) e acrescenta um nível de robustez, pelo facto de se existir a falha de diversos veículos, o grupo consegue concluir os objetivos impostos.

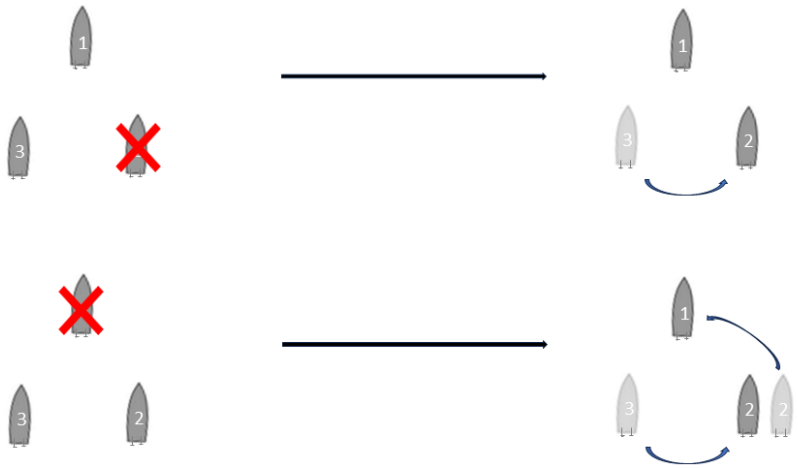


FIGURA 5.10: Aquisição de novas posições no grupo.

Na figura 5.10 estão representados dois exemplos de acontecimentos que podem ocorrer dentro do grupo, devido a perdas ou atribuição de outras tarefas a cada um dos veículos. No primeiro caso existe a perda do USV2, o que faz com que o USV3 assuma essa posição, passando a ser o segundo elemento do grupo, o que também pode ser observado no segundo exemplo, onde existe a perda do líder, e ambos os USV's sobem na cadeia hierárquica, assumindo as suas novas posições.

5.6 Nó de Planeamento

Com a criação de um nó de coordenação entre os veículos, é possibilitada a atribuição de tarefas ou formas de navegação consoante os valores obtidos para a posição de cada veículo, estando os tópicos subscritos e publicados por este nó na figura 5.11, sendo este o ponto de foco deste nó.

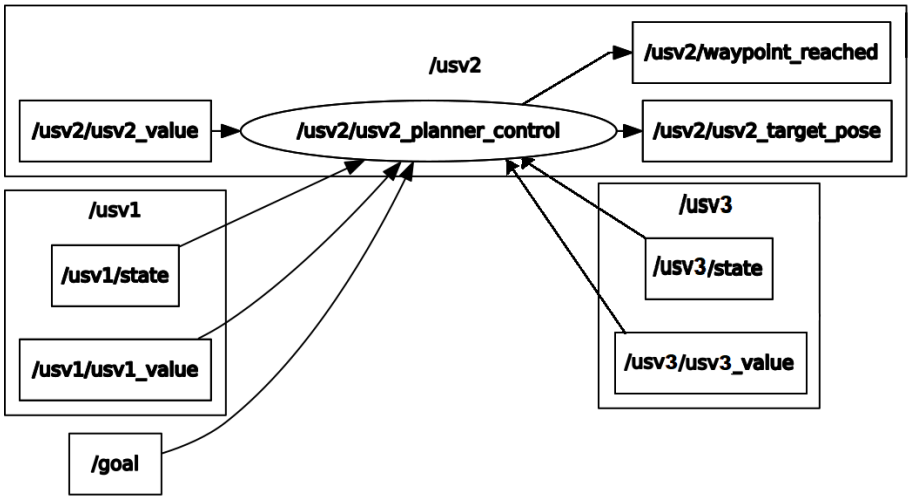


FIGURA 5.11: Tópicos de entrada e saída do nó de planeamento.

Este tem como objetivo a atribuição de tarefas ou formaturas específicas consoante a quantidade de veículos operacionais, como pode ser visto na figura 5.12, bem como o cálculo de uma distância de segurança, de modo a evitar colisões entre veículos.

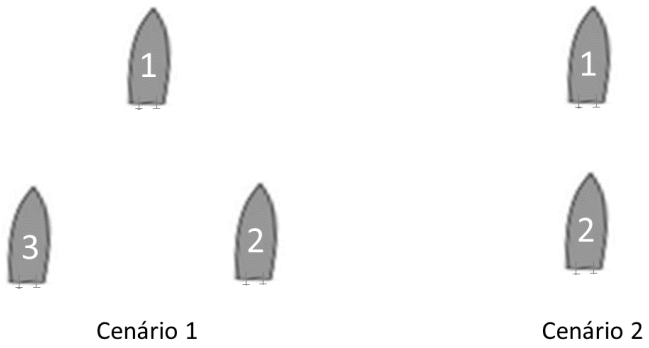


FIGURA 5.12: Possíveis cenários de navegação.

Estes dois cenários são um exemplo da capacidade deste nó, sendo que no cenário 1 temos a presença de três veículos operacionais, o que permite a navegação em formatura destes mesmos, consoante o atual líder. No cenário 2, apenas com a presença de dois veículos, faz com que o segundo veículo navegue atrás do líder em linha. Estes são apenas dois exemplos dos diversos cenários que se podem criar, consoante o número de veículos presente no grupo.

Neste nó, estão presentes seis tópicos de entrada, sendo estes a posição de cada um dos veículos (tópico descrito anteriormente), os valores relativos ao grupo de cada um dos veículos, passados pelos respetivos nós de coordenação e o tópico **/goal**, passado pelo nó de patrulha, de modo a que este nó consiga atribuir tarefas e trajetórias especificas e individuais, a todos os elementos do grupo.

5.7 Adição de múltiplos veículos

Após todas as modificações no modelo terem sido realizadas e ter sido implementada uma arquitetura de funcionamento modular, foi necessário testar o correto funcionamento do modelo no simulador, de modo a confirmar que todos os nós e tópicos tinham sido criados e havia uma correta passagem de informação e as interações necessárias com o UWSim estavam a ser realizadas corretamente. Deste modo, foi criado um novo cenário e transmitida uma lista de waypoints para o veículo, confirmando assim o correto funcionamento deste mesmo.

Realizados os testes com sucesso ao novo modelo do veículo, o próximo passo seria a criação de um novo ambiente de simulação com múltiplos modelos deste mesmo veículo. Para ser possível a criação do novo ambiente, foram criados três modelos com as mesmas características, denominados de **USV1**, **USV2** e **USV3**, através da criação de todos os ficheiros necessários para o seu correto funcionamento e seguindo todos os passos necessários, presentes no apêndice Anexo A.

5.7.1 1º Ambiente de simulação

Numa parte inicial, foi criado um novo ambiente com a inclusão dos três novos modelos dos veículos, como demonstrado na figura 5.13, com todas as interações e respostas corretas com o meio envolvente. Cada modelo era constituído pelos nós descritos anteriormente na criação de um novo modelo e todos partilhavam uma lista de waypoints comuns.

Com o correr da simulação, foram observados vários comportamentos que tinham de ser alterados e afinados, sendo destes relativamente à sua movimentação, pois estes apresentavam uma trajetória em ziguezague, o que indicaria que teria que haver um pequeno ajuste nos valores das constantes do controlador de cada um do veículos, e estes colidiam uns com os outros à chegada dos waypoints, devido ao facto destes serem iguais para todos o veículos, o que fazia com que estes tomassem rotas desnecessárias.

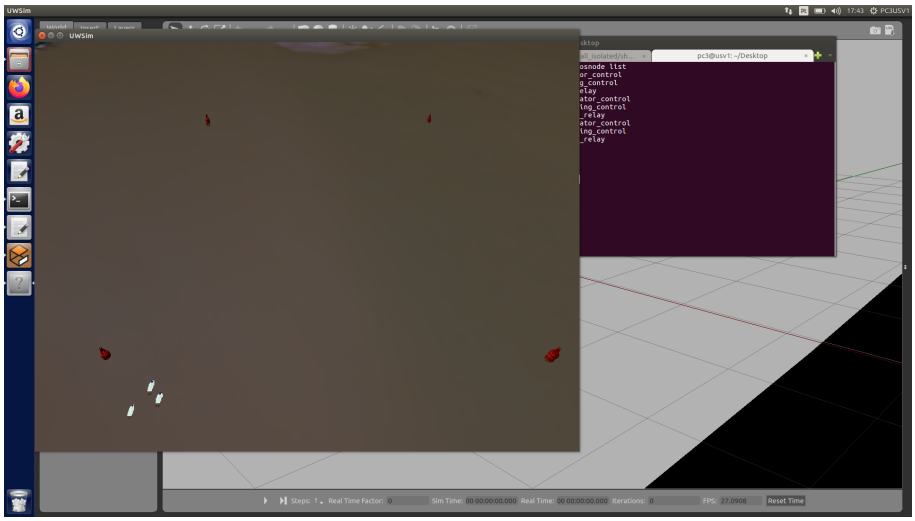


FIGURA 5.13: Novo ambiente de simulação com múltiplos veículos.

Feitas as observações e os ajustes aos valores das constantes dos controladores de cada veículo, foi adicionado ao nó de controlo uma função que recebia não só a posição atual do veículo, mas também iria receber a posição atual dos restantes veículos, de modo a calcular a distância entre veículos, e caso essa distância fosse inferior a um valor pré-definido, os veículos guinariam para o rumo oposto ao azimute do outro veículo, de modo a aumentarem essa distância. A partir do momento em que essa distância fosse maior que a distância pré-definida, o veículo voltaria ao rumo em direção ao waypoint. Com a adição desta função foi possível observar que os veículos, no momento de chegada aos waypoints, já não colidiam uns com os outros, desviando-se assim do caminho de cada um.

Embora o ambiente de simulação corresse de forma correta e apresentava os resultados esperados, ainda não estava criado um grupo coeso de veículos, pois neste ambiente apenas se tinha três veículos independentes, a navegarem com a mesma lista de waypoints e a desviarem-se uns dos outros, sem um elo de coordenação ou hierarquia entre estes.

Primeiro elo de comunicação

O primeiro elemento de comunicação / coordenação entre os veículos, foi a adição de uma bandeira boleana correspondente a cada um dos veículos, a cada nó de patrulha existente. A presença desta opção nestes nós permitia a mudança para o próximo waypoint, para todos os veículos, no momento em que qualquer um dos veículos chegasse ao waypoint, ou seja, independentemente de qualquer que fosse o veículo, caso algum dos outros por diversas razões, chegasse primeiro ao waypoint, os restantes começavam-se a dirigir, a partir daquele momento, para o próximo valor da lista.

Assim, com a presença deste valor booleano e com a presença da função de distância de segurança entre indivíduos, o nível de coesão começaria a estar num patamar superior, começando a existir uma certa estrutura no grupo.

5.8 Sistema de comunicações

Um grande desafio relativamente à troca de informação entre modelos de veículos reais, é o facto destes necessitarem de uma ligação física entre si, podendo esta estar limitada consoante o meio envolvente em que estes se encontram.

Alguns aspetos que se tem que ter em conta quando nos referimos às comunicações entre veículos autónomos são:

- **Alcance**: ligações físicas como o Wi-fi, para veículos de superfície, ou comunicações acústicas, para veículos de sub-superfície, têm grandes limites relativamente ao seu alcance, o que poderá influenciar o emprego e a eficácia destes tipos de veículos pois existe uma necessidade destes permanecerem relativamente perto um dos outros ou da sua estação;
- **Largura de banda**: devido ao baixo custo de construção destes modelos, o fator largura de banda pode ser muito limitativa, pois esta é muito reduzida e a informação passada ter que ser apenas a essencial. Outro fator de elevada importância em comunicações acústicas é o meio envolvente, bem como possível ruído presente neste;
- **Segurança**: a comunicação entre veículos deve ser segura e protegida contra ataques ou interferência externa, que pode por em causa a estrutura do grupo, ou até mesmo da tarefa;
- **Latência**: este fator é muito ambíguo, pois a grande influência relativamente a este, é o meio em que os veículos operam, pois se estivermos na presença de

um grupo heterogéneo, com veículos de superfície e de sub-superfície, poderá existir algum atraso relativamente à comunicação realizada entre estes.

Em modelos reais, podem estar presentes nós de comunicação dedicados, responsáveis pela troca de informação limitada entre os veículos do grupo.

5.9 Comunicações em ambiente de simulação

No seguimento dos fatores limitativos referidos anteriormente e pelo fato deste modelo estar a ser testado em ambiente de simulação, o método utilizado para realizar a comunicação entre os diversos computadores, foi através da utilização do pacote do ROS, denominado de Multimaster, ferramenta esta que permite a limitação da informação passada entre veículos, neste caso através de tópicos, que poderá simular o fator de limitação de largura de bando num modelo real.

5.9.1 Multimaster

O próprio simulador utilizava o Gazebo como base de ambiente de simulação, com a adição de determinados plugins, de modo a ser possível a criação de um ambiente de veículos de superfície, utilizando ferramentas e bibliotecas do ROS para a operação os veículos. Posto isto, caso fosse necessário aplicar este projeto futuramente, em veículos de superfície reais, cada um dos veículos necessitaria de ter presente o seu próprio núcleo operacional, o seu próprio sistema operativo ROS, algo que não era alcançado com o correr do ambiente de simulação referido anteriormente, pois ao iniciar este, apenas um núcleo de ROS era inicializado, sendo este responsável pelos três veículos.

Para iniciar o ROS num computador ou num micro-controlador, é necessário iniciar um núcleo de ROS, denominado de *roscore*, que inicia todas as suas bibliotecas e ferramentas para uso, sendo possível apenas iniciar um núcleo por unidade computacional, sendo necessário a utilização de outros computadores para a utilização independente de cada um dos veículos. O número de computadores a utilizar não seria fator limitativo, devido ao fato de inicialmente se estar a utilizar uma máquina virtual para correr o simulador, criando-se assim uma máquina para cada veículo.

Com a utilização de diversas máquinas virtuais, diversos ROS e de diversos Gazebos's, era necessário haver um meio de comunicação entre todos estes computadores, de modo a ser possível a passagem de informação e tópicos entre cada

veículo, que a partir deste momento estariam separados fisicamente, cada um com o seu meio de simulação.

Este meio de comunicação entre núcleos de ROS e computadores seria possível através da utilização do pacote *multimaster_fkie* (Cotarelo e Juan 2015). Este pacote é constituído essencialmente por dois nós responsáveis:

- **master_discovery**:que tem como características principais:
  - periodicamente manda uma mensagem multi-canal para a rede, de forma a que outros master's detetem a sua presença e detetando outros master's disponíveis;
  - verifica se o núcleo local sofreu alterações na rede e notifica todos os outros master's na rede comum sobre essas mudanças.
- **mater_sync**:que tem como características principais:
  - transmite informação e tópicos recolhidos a outros master's;
  - utiliza a informação passada por outros master's para atualizar os tópicos do seu núcleo;
  - pode selecionar quais os master's, informação ou tópicos pode ignorar.

À semelhança do que acontece num sistema ROS normal, assim que os tópicos e serviços são registados noutros sistemas ROS, é criada uma ligação ponto-a-ponto para fazer uma ligação direta entre os nós, como demonstrado na figura 5.14, não sendo utilizando até haver uma mudança desses valores. A instalação deste pacote foi feita em cada um dos computadores, de forma a testar a sua conexão e correto funcionamento.

Este pacote apresenta duas opções de funcionamento, sendo a primeira a utilização de diversos computadores, cada um com o seu núcleo de ROS, na mesma rede comum (figura 5.15), e a segunda a utilização de diversas redes com diversos computadores, a comunicarem numa outra rede, fazendo com que a seja a primeira opção a ser aplicada, devido a cada veículo ter o seu núcleo de ROS, independente, mas estarem a comunicar na mesma rede comum.

Com a aplicação da primeira opção, houve a necessidade de mudança dos valores IPv4 de cada um dos computadores, para valores fixo, de modo a que a comunicação entre estes fosse sempre possível, independente da rede utilizada para a sua comunicação.

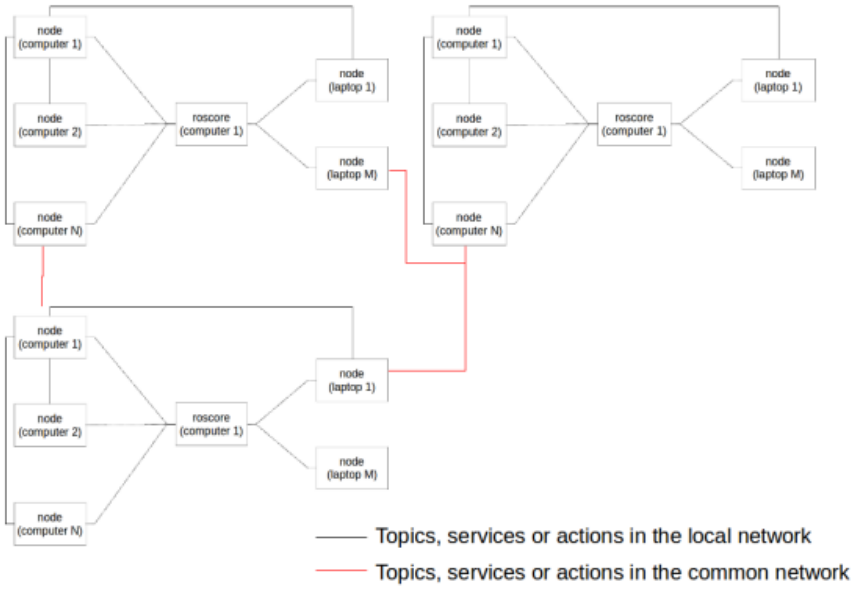


FIGURA 5.14: Esquema de conexão entre diversos nós.

Fonte: Cotarelo e Juan 2015.

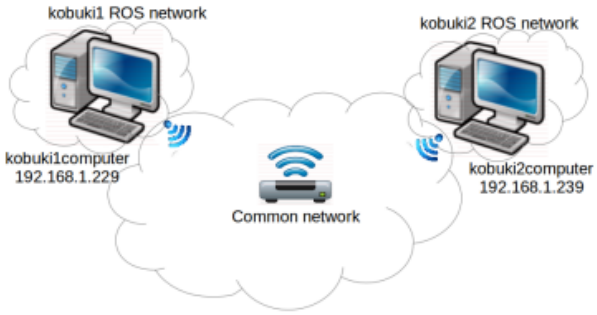


FIGURA 5.15: Exemplo de configuração de apenas uma rede comum.

Fonte: Cotarelo e Juan 2015.

A mudança dos endereços de cada computador permite a comunicação e troca de informações caso fosse utilizada em rede local. Após esta mudança ser efetuada, pingou-se todos os computadores de forma a confirmar o sucesso da conexão e começar a utilizar ambientes de simulação independentes para cada veículo.

Ao iniciar ambientes de simulação para cada veículo, houve a necessidade de mudança de nomes de todos os nós de cada veículo, de forma a estes serem únicos e relativos a apenas um veículo específico, bem como a mudança dos nós base do Gazebo (figura 5.16), havendo necessidade de alteração dos seus ficheiros de lançamento iniciais, de modo a que cada veículo e ambiente de simulação estivesse conectado a um nó único, de forma a ser possível a utilização de diversos ambientes

de simulação simultaneamente.

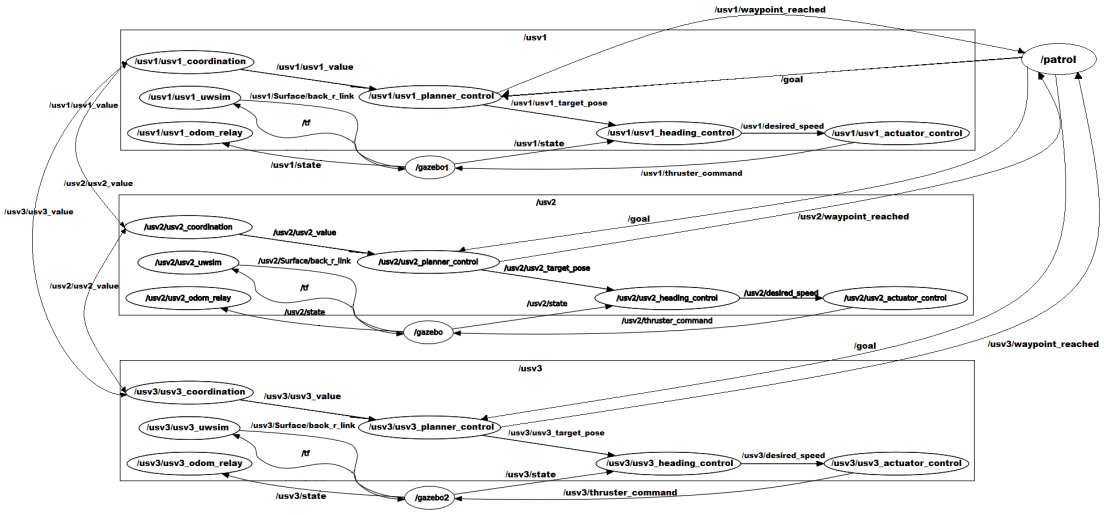


FIGURA 5.16: Configuração múltiplos ambientes de simulação, através da ferramenta rqt_graph.

Como pode ser visto, a utilização do nó de patrulha no computador de controlo e a sucessiva transmissão do tópico **/goal** para todos os veículos, permite a alteração dos waypoints durante o correr da simulação caso seja necessário, ficando a ser o elemento de interação com o elemento humano, sendo mais um passo para o aumento da coesão do grupo.

Todas as alterações e o código realizado durante este capítulo, foram realizado de forma a serem o mais simples possível e a necessitarem do menor número de ferramentas do Gazebo, de modo a facilitar a futura implementação nos modelos físicos.

Capítulo 6

Resultados experimentais

No desenvolver de projetos, um fator essencial para o seu sucesso é a fase de testes e experiências, pois é nesta que todos os componentes físicos e digitais são postos à prova, como uma forma de certificar o seu correto funcionamento e que estes estão prontos para serem implementados.

Os testes realizados focaram-se essencialmente em:

- testar o funcionamento do simulador, de modo a determinar o seu correto funcionamento e criar uma base estável para o trabalho desenvolvido;
- testar a operação dos veículos, consoante o tipo de controlador imposto;
- testar o ambiente de simulação com a imposição de diversos veículos;
- experimentar a comunicação entre os diversos computadores (VM) com a utilização do *multimaster*;
- experimentar os diversos cenários de coordenação entre os veículos.

Devido ao fator limitativo da falta de material para a construção dos veículos, não foi possível a sua integração ou construção, o que revelou ser um barreira considerável ao trabalho desenvolvido, pois não permitiu aferir o correto funcionamento destes modelos no mundo real. Deste modo, foram realizados testes, com o maior realismo possível.

6.1 Teste do ambiente

Um dos primeiros passos a realizar foi a criação de um novo ambiente computacional, no sistema operativo Linux, pois o simulador utilizado é desenvolvido utilizando a linguagem ROS, tendo sido toda a preparação desse ambiente conforma descrito no Anexo A.

Após a instalação do simulador ser bem sucedida, embora o modelo com interesse fosse apenas o veículo diferencial, inicialmente correu-se uma simulação a cada um dos modelos, em cada um dos cenários, com o intuito de aferir possíveis erros de funcionamento.

Cada cenário de simulação é constituído por dois ficheiros essenciais, um ficheiro **.launch** e um ficheiro **.xml** responsáveis pela criação e o arranque do ambiente de simulação. O ficheiro **.xml** é o responsável pela criação dos objetos e veículos presentes no ambiente de simulação, bem como a criação da parte gráfica, enquanto o ficheiro **.launch** é responsável pelo início do UWSim, lançamento do Gazebo e lançamento do ficheiro do modelo do veículo. Neste documento, está ainda presente um nó com o nome *record_usv1*, do tipo **.bag** que guarda a posição do veículo, transmitida pelo tópico */usv1/state*.

6.2 Controlo de um veículo individual

Após a confirmação do correto funcionamento do simulador, passou-se à criação dos novos modelos. Estes foram criados com base nas características do modelo diferencial já existente no simulador, com a alteração do seu *namespace*, passando estes a denominarem-se de **USV1**, **USV2** e **USV3**, e com a alteração de todos os nós extras necessários para o seu funcionamento.

Para criar os novos veículos foi necessário a criação dos respetivos ficheiros: **.yaml**, responsável pela criação das uniões do modelo, para sofrerem alterações pelo Gazebo; **.xml**, responsável pela criação do modelo físico no simulador; **.launch**, responsável pelo arranque do modelo e **.urdf**, responsável pelas interações com o UWSim, de modo às alterações do meio envolvente (vento, ondulação, colisões, etc), produzirem efeitos neste.

Com a criação dos novos modelos, foi necessário correr a simulação de modo a aferir as corretas conexões entre os nós dos novos veículos, bem como o seu comportamento no ambiente de simulação. A figura 6.1 apresenta a nova arquitetura de cada um dos veículos e a figura 6.2 apresenta o trajeto realizado pelo novo veículo, no cenário 2, observando-se o comportamento característico do controlador do tipo PI, com a reação lenta do veículo.

De modo a ser possível ter uma representação gráfica do movimento do veículo, foi criado um ficheiro *Python* denominado de *position_plot*, que utiliza a ferramenta de leitura e criação de ficheiros do tipo **.csv**, de modo a retirar os valores das coordenadas X e Y da mensagem do tipo *Odometry* publicada pelo veículo,

compilando a sua posição durante o correr da simulação e expondo esta, utilizando a biblioteca *matplotlib* do *Python*, numa representação gráfica.

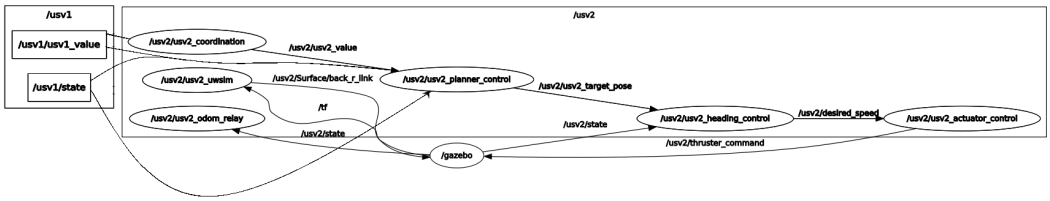


FIGURA 6.1: Esquema final dos modelos.

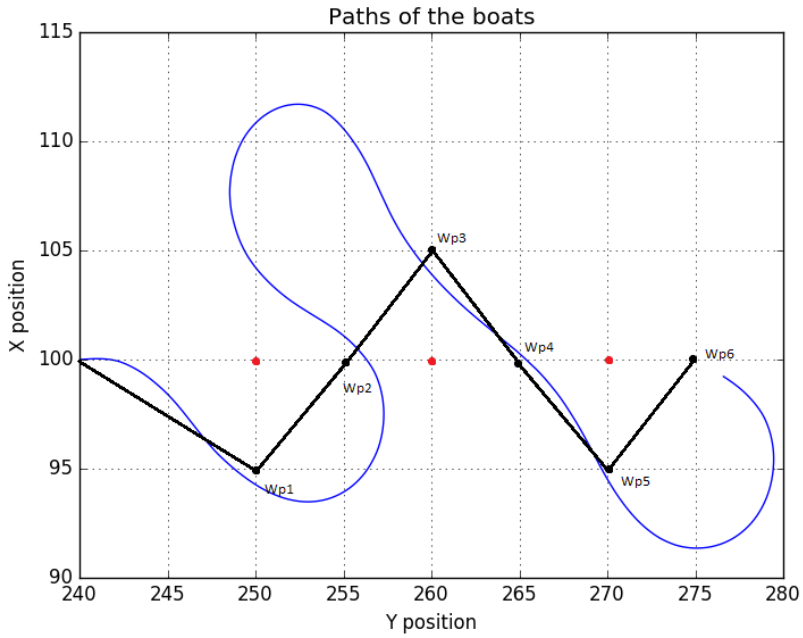
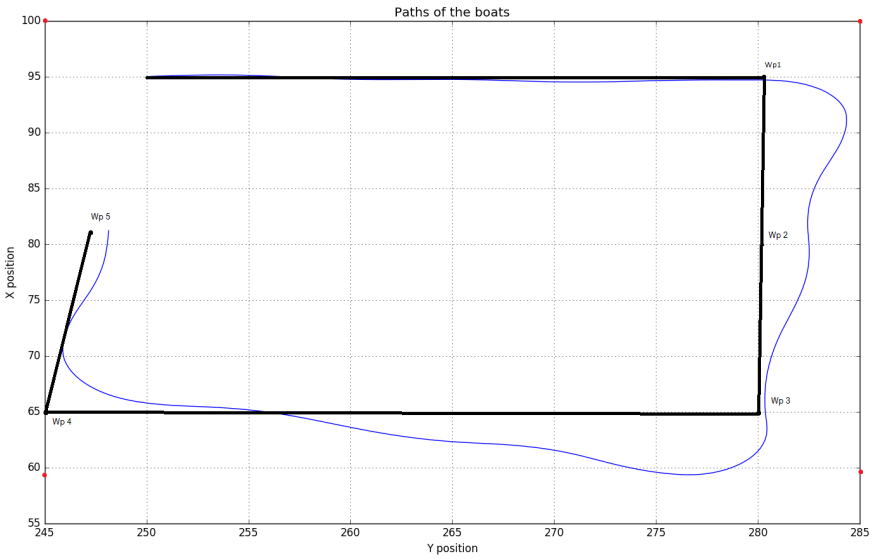


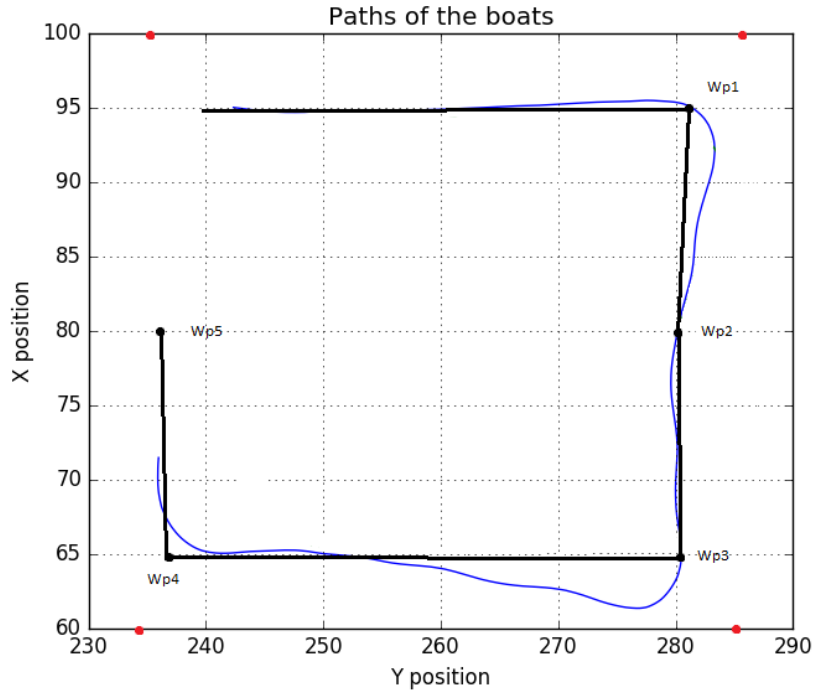
FIGURA 6.2: Trajeto do modelo a realizar o cenário 2, com controlador PI.

Com o correto funcionamento dos novos veículos, foi alterado o tipo de controlador. Devido ao modelo utilizado ser um modelo já presente no simulador, este apresentava inicialmente um controlador do tipo PI, sendo este alterado para um controlador do tipo PD, pois este apresentava características mais benéficas para o comportamento do veículo (figuras 6.3b e 6.3a).

A alteração do tipo de controlador não apresenta grandes alterações visualmente, embora o controlador PD apresente maior estabilidade, como pode ser visto no gráfico entre o wp2 e o wp3 (Figura 6.3b) e apresenta uma reposta mais rápida na chegada aos waypoints, com um *overshoot* menor.



(A) Trajeto do veículo com controlador PI.



(B) Trajeto do veículo com controlador PD.

FIGURA 6.3: Diferença entre o trajeto do veículo com controlador PI e PD.

A afinação dos controladores usados, foi feita manualmente através da tentativa erro, pois apenas seria possível observar estas alterações durante o correr da simulação.

6.2.1 Seguimento de alvos estáticos e móveis

De forma a realizar o teste das rotações relativamente a alvos em movimento, foi criado um cenário com dois modelos, onde o primeiro modelo iria seguir uma lista de waypoints fixos, enquanto que o segundo modelo seguiria esse. Este feito é possível pelo facto do nó de planeamento do segundo modelo receber a posição do primeiro, e através dessa criar um waypoint virtual, que seria transmitido como ponto alvo para o segundo modelo.

Para entender o comportamento que seria esperado dos motores num modelo real, foi criado outro ficheiro, de modo a compilar num gráfico os valores das rotações impostas nos motores do modelo na simulação, não tendo em conta as variações impostas em cada motor independente de modo a controlar o rumo do veículo.

Após a criação deste ficheiro, foi utilizado em dois cenários distintos, onde o trajeto realizado foi o representado na figura 6.3, sendo que no primeiro este apenas seguiria os alvos estáticos e no segundo este seguiria um alvo em movimento a realizar esse trajeto.

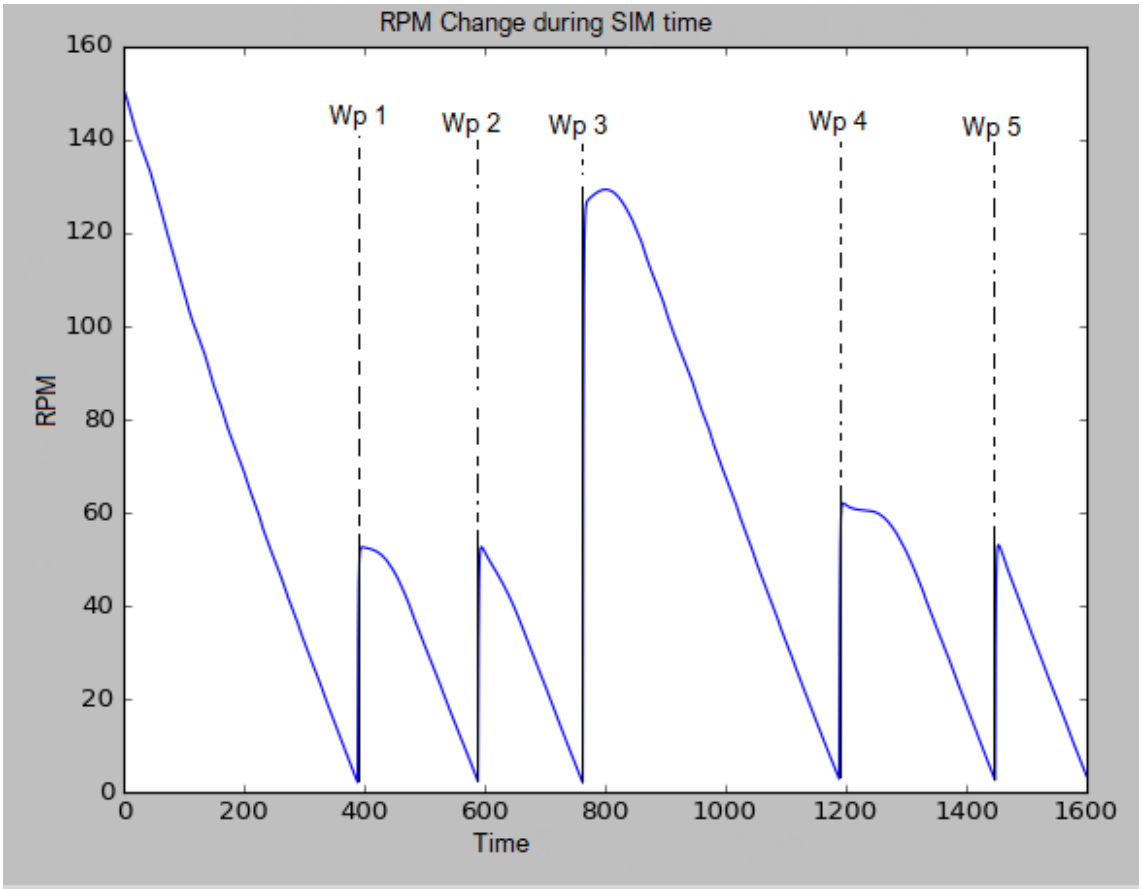


FIGURA 6.4: Rotações dureante o trajeto com alvos fixos.

No gráfico da figura 6.4 podemos observar que o valor das rotações aumenta quando há uma mudança de alvo e à medida que a distância do alvo vai diminuindo, este efeito vai sendo atenuado, até receção do novo alvo. O maior valor atribuído do inicio ao Wp 1 e do Wp 3 para o Wp 4 é devido a estas serem as maior distâncias a serem percorridas pelo veículo, devido à velocidade do veículo estar relacionada com a distância a que este se encontra do alvo.

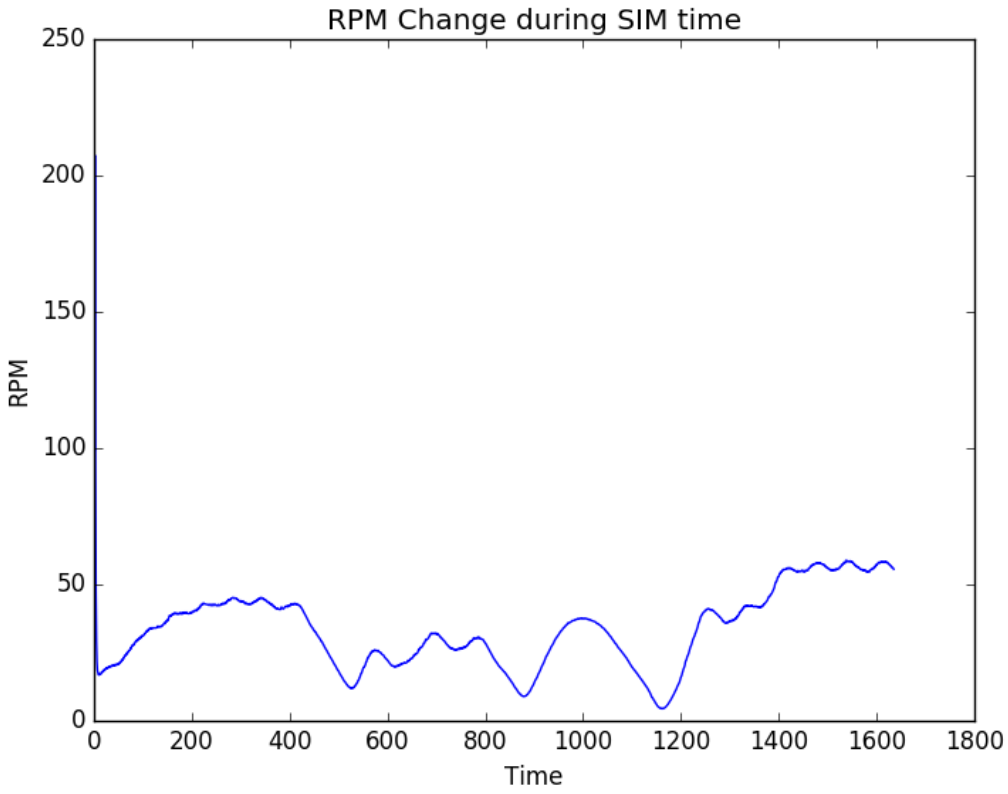


FIGURA 6.5: Rotações durante o trajeto com alvos móveis.

No gráfico da figura 6.5 pode-se observar que não existem a presenças de picos como no gráfico anterior, mas sim uma atribuição de valores mais constante devido a este estar a seguir um alvo em movimento, sendo o seu objetivo aproximar-se deste o melhor possível. Neste gráfico os valores mais constantes devem-se ao seu alvo estar sempre em movimento, ou seja, mantém sempre uma determinada distância do veículo, sendo as barrigas criadas quando o alvo se afasta mais do veículo.

Para ambos os casos, o valor máximo de rotações possível foi escolhido para as 2000 rpm, não por fatores de teste ou de otimização de valores, mas sim pelo facto de no simulador ser possível a atribuição de qualquer valor de rotação, algo que não é praticável num modelo real, dai ser escolhido um valor de carácter informal apenas.

O veículo apresenta valores de rotações baixos, devido ao valor atribuído a γ na equação da velocidade ser um valor baixo. Este valor pode ser alterado consoante a resposta pretendida do veículo, tendo sido inicialmente escolhido um valor baixo, pois era o suficiente para apresentar um representação visual significativa.

6.3 Controlo de formação de um *swarm*

Com os resultados obtidos anteriormente, foram criados os restantes dois modelos, utilizando os mesmos métodos descritos, sendo adicionados ao mesmo ambiente de simulação, criando assim os três veículos consecutivos.

Foi corrida a simulação, obtendo o gráfico com o trajeto dos três veículos (figura 6.6), através da utilização do ficheiro *Python* criado. Com o correr da simulação, pode-se observar que enquanto grupo criavam problemas, que anteriormente não tinham sido observados no comportamento individual.

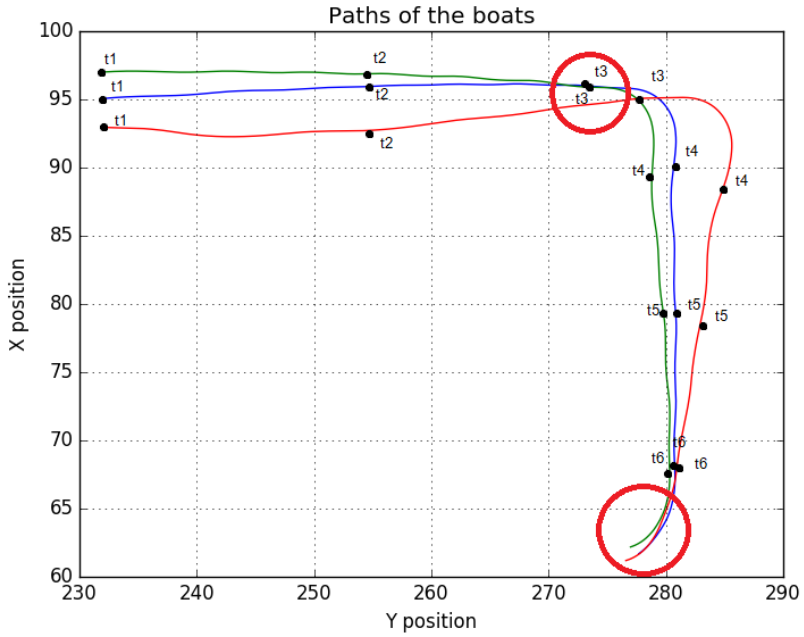


FIGURA 6.6: Trajeto dos três veículos no mesmo ambiente de simulação.

Este erro, identificado no gráfico (pelo círculo vermelho), é derivado do facto de ambos os veículos estarem a navegar para os mesmos waypoints, causando colisões. Em determinadas simulações, este efeito é quase impercetível, pelo facto destes colidirem à chegada ao wayipoint, onde decorre a mudança para o próximo, aliviando o desvio devido à colisão.

Após esta observação, no nó de planeamento foi adicionado um função que calcularia a distância entre cada veículo e caso estes tivessem a uma distância inferior a uma distância de segurança estabelecida, estes começavam a desviar-se, como assinalado na figura 6.7.

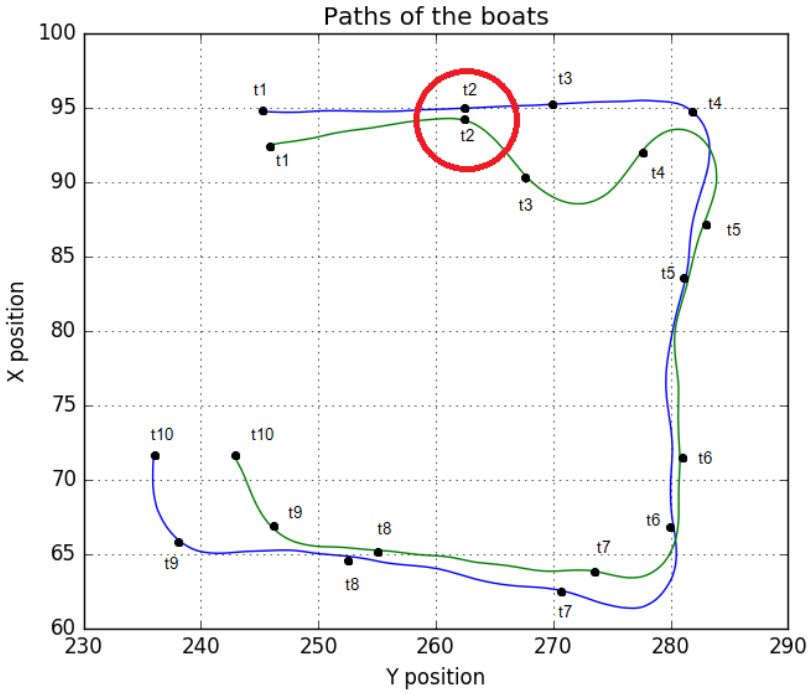


FIGURA 6.7: Trajeto dos veículos com distância de segurança.

Após o primeiro desvio, este ficam com o desfasamento relativamente aos seus trajetos, aumento assim a distância entre os dois, o que faz com que estes não se desviem, como por exemplo nos instantes t5, que os trajetos se cruzam, mas como este já não navegam lado a lado, não necessidade dessa atuação.

6.4 Criação dos outros modelos

Com a empregabilidade dos veículos no mundo real, haveria a necessidade de ter presente um microprocessador em cada um dos veículos, o que implicaria que houvesse um servidor ROS (ou denominado de *roscore*) a correr em cada um desses. De modo a tentar replicar de melhor modo possível esta situação, foram criadas outras VM's, sendo cada uma responsável por um veículo.

Em cada um dos novos computadores foram importados todos os ficheiros para o correto funcionamento individual de cada veículo, e mais uma vez testados em todos os computadores (figura 6.8).

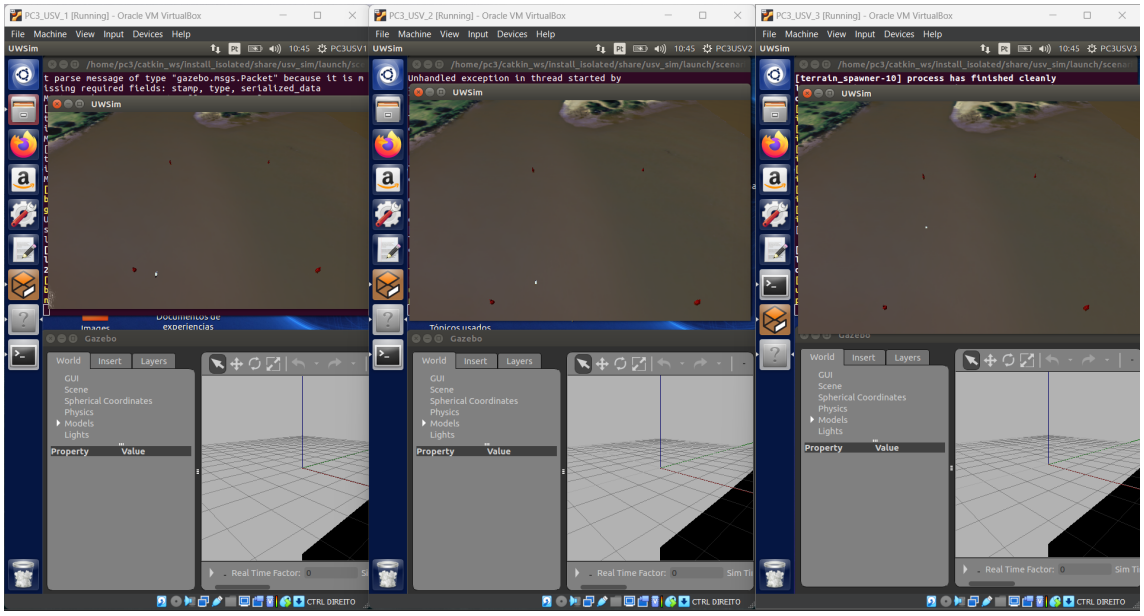


FIGURA 6.8: Diversos ambientes de simulação.

6.4.1 Inicialização do *Multimaster*

Após a criação e separação dos diferentes veículos nas diferentes VM's, utilizou-se o pacote *multimaster* de modo a permitir a comunicação entre as diversas redes do ROS em cada veículo.

A configuração final dos valores IPv4 atribuídos a cada um dos computadores é a representada na tabela 6.1.

IP	Estação
192.169.1.152	Controlo
192.168.1.153	USV1
192.168.1.154	USV2
192.168.1.155	USV3

TABELA 6.1: Configuração dos IPv4 atribuídos a cada um dos computadores.

A utilização deste pacote permitia assim a troca dos tópicos entre os computadores. Realçar que devido a ser um ambiente de simulação, todos os tópicos podem ser passados entre todos os veículos, o que poderá não ser viável num ambiente real, devido à limitação de largura de banda possível, havendo necessidade de os limitar estes ao mínimo indispensável.

Relativamente à distinção de quais tópicos devem ter carácter geral ou não, estes estão referidos no Apêndice A.

6.5 Teste aos diferentes cenários

Após a criação e atribuição dos novos computadores a cada um dos modelos, iniciou-se a fase de testes ao grupo como um todo, criando-se diversos cenários de aplicação de modo a consolidar o trabalho realizado.

6.5.1 Navegação em formatura

O primeiro cenário a ser criado é um cenário simples, sendo apenas a navegação do grupo entre waypoints com o objetivo de analisar o comportamento do grupo. As figuras 6.9a e 6.9b demonstram o comportamento de cada um dos indivíduos em novos cenários. Em cada gráfico estão representados pontos temporais, de modo a entender as posições relativas de cada um dos veículos.

Em ambos os cenários representados, os veículos USV2 e USV3 seguem o líder, USV1, a uma distância de aproximadamente 4 metros a ré deste, para a esquerda e para a direita. Pode-se observar que esta formação é mantida durante a totalidade do percurso, mantendo sempre a distância de segurança entre os veículos.

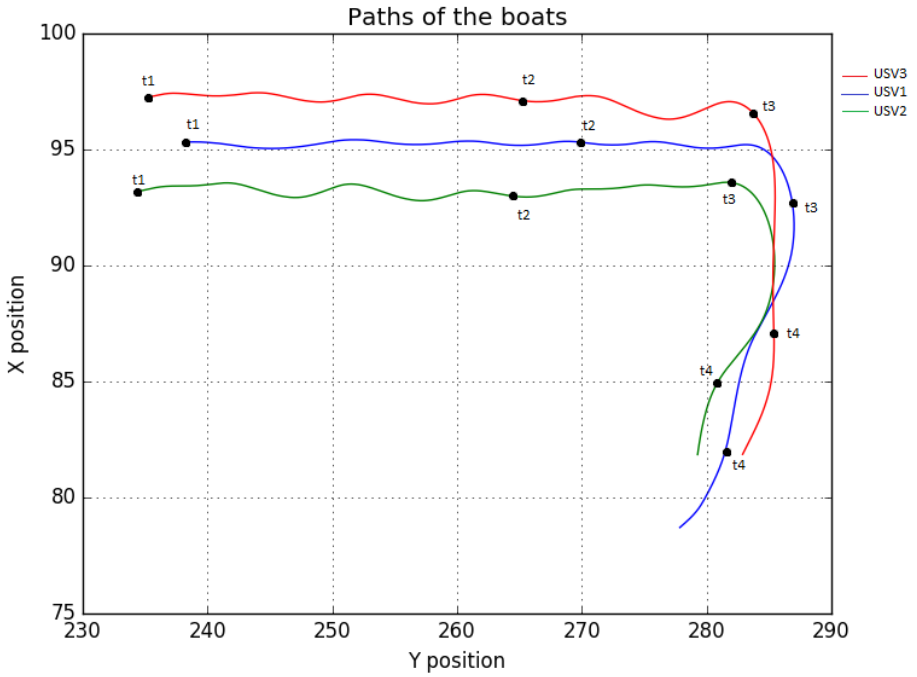
O percursos dos dois veículos seguidores apresenta ser um pouco oscilatório, não devido ao seu controlador, mas sim ao facto de como estes estão a seguir dois pontos virtuais em relação ao líder, a mínima mudança realizada pelo líder no seu percurso, representa-se numa maior atuação nos seguidores.

6.5.2 Perda do líder

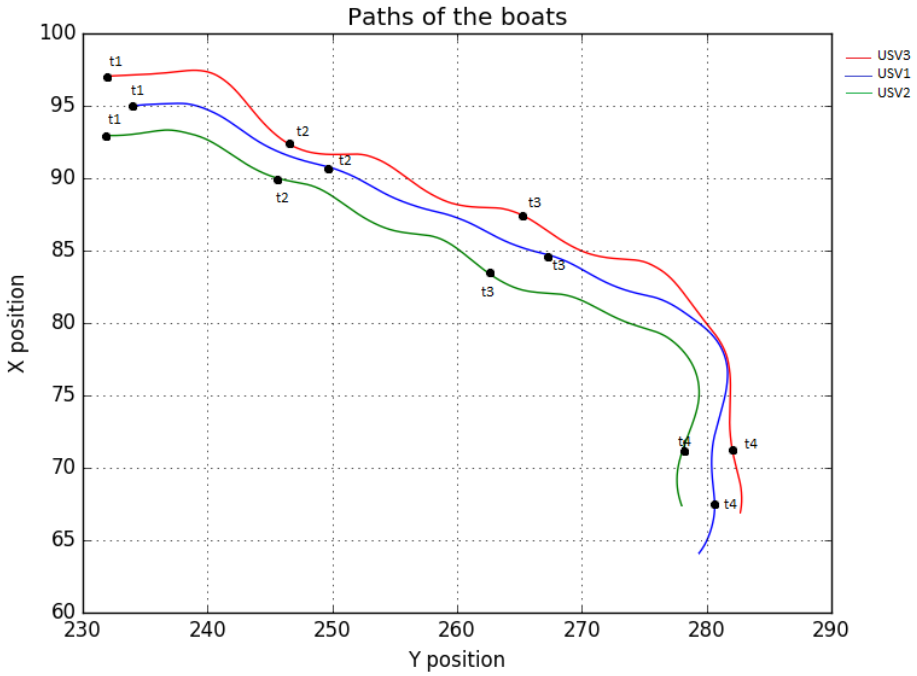
Um dos focos mais importantes num *swarm* é a sua fiabilidade e robustez, ou seja, possibilitar a continuação do grupo mesmo com a perda de diversos indivíduos. Neste cenário é mesmo isso que é testado, se a perda de indivíduos do grupo punha em causa a continuação da tarefa dos restantes elementos. O trajeto dos veículos está representado na figura 6.10.

Neste cenário, o grande acontecimento é a perda do líder durante a navegação do grupo. Como foi referido anteriormente, na estrutura do grupo está imposta uma hierarquia entre os indivíduos, não de um modo de aplicar mais importância a certos veículos, mas de modo a organizar o grupo consoante funções ou posições.

Assinalado no gráfico está o momento em que se dá a perda do atual líder, o USV1. Após o líder parar de transmitir a mensagem de que está vivo, existe um período de espera e só após esse período de espera é que o próximo veículo assume a posição de novo líder. Neste caso, passado esse período, o USV2 assume a posição



(A) Trajeto dos veículos no cenário 1.



(B) Trajeto dos veículos no cenário 2.

FIGURA 6.9: Trajetos realizados pelo grupo.

de líder e o cenário passa a ser apenas o seguimento do líder, por parte do USV3, pelo facto de serem apenas dois veículos, como representado na figura 5.12.

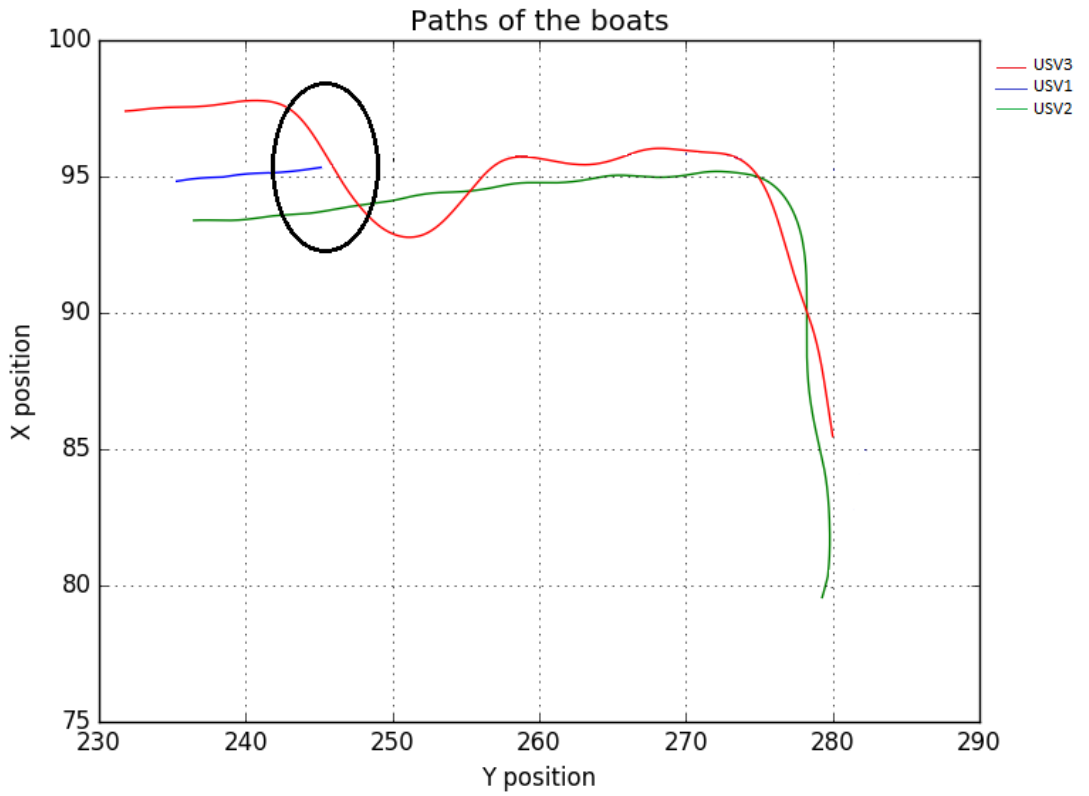


FIGURA 6.10: Cenário de perda do líder.

6.5.3 Atribuição de tarefas específicas

Para aumentar a versatilidade do grupo, este deve permitir que os diversos elementos sejam empregues em determinadas tarefas secundárias, durante o cumprimento do objetivo principal.

Para isso, foi criado outro cenário, onde o objetivo principal do grupo era percorrer um determinado trajeto, através do seguimento de waypoints, mas durante esse percurso um dos veículos iria:

- navegar para uma determinada área;
- entrar na área determinada;
- realizar uma determinada tarefa;
- sair desta área e juntar-se de novo ao grupo.

Este acontecimento está representado na figura 6.11, onde o USV3 passado um tempo de navegação, dirige-se à área marcada no gráfico, sendo a sua tarefa

apenas navegar nesta área durante um período de tempo, e após esse tempo inicia o seu deslocamento de modo a juntar-se novamente com o grupo.

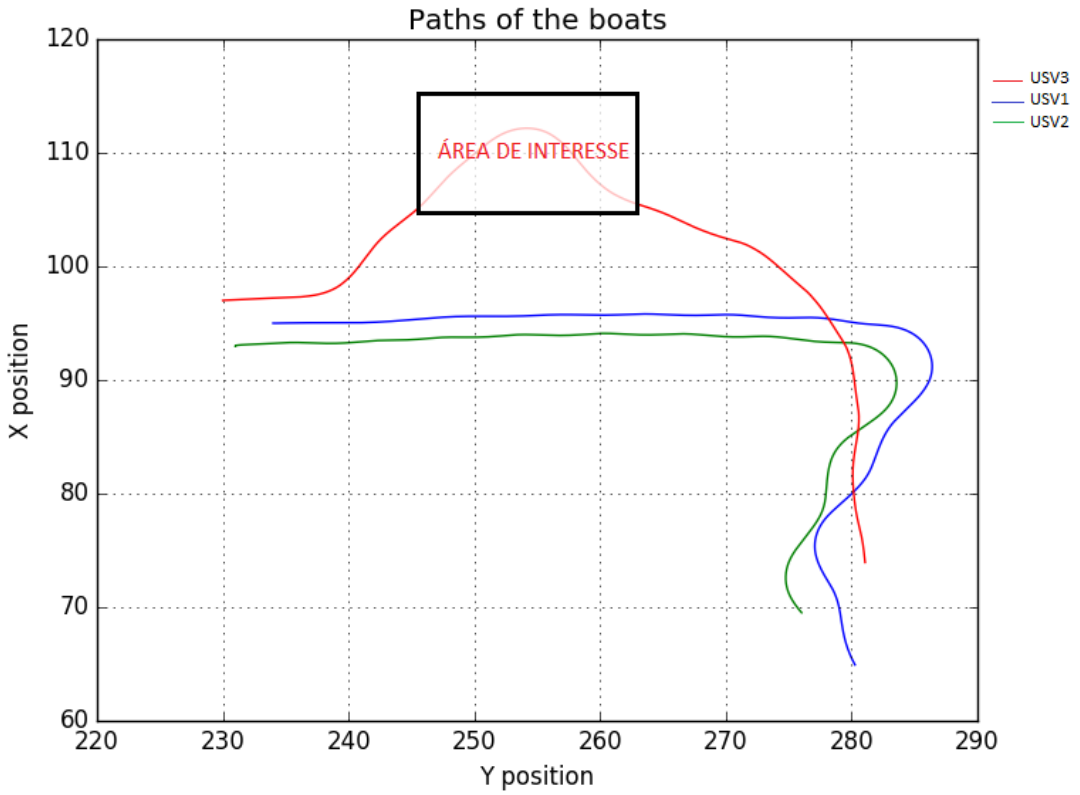


FIGURA 6.11: Cenário de atribuição de tarefas secundárias.

No caso da simulação, o veículo desloca-se a uma determinada área passado um tempo de simulação, contudo num ambiente real, este deslocamento pode ser iniciado através de um comando direto ou devido a um acontecimento exterior.

Capítulo 7

Conclusão

O projeto de um modelo de *swarm* de veículos de superfície em ambiente de simulação desenvolvido no decorrer desta dissertação foi motivado pela necessidade de criação e controlo de um cardume de veículos de subsuperfície, pertencente à segunda parte do projeto SABUVIS.

Após uma análise do estado da arte, foi possível determinar quais as características necessárias para o desenvolvimento de um grupo veículos que pudesse arcar com esta definição, tanto a nível individual como a nível coletivo. Assim conseguiu-se realizar a definição dos requisitos básicos dos sistemas, relativamente ao seu nível material, como a nível estrutural e comportamental do modelo a ser desenvolvido.

Inicialmente era pretendido o desenvolvimento deste modelo para veículos de sub-superfície (foco do projeto SABUVIS), mas como não havia disponibilidade desses veículos, alterou-se o foco para veículos de superfície. Através das características necessárias para estes veículos, fez-se a escolha do *hardware* a ser utilizado em cada veículo, de modo a cumprir com os requisitos de simplicidade e baixo custo. Contudo, devido ao facto do material não ter sido adquirido pela Escola Naval segui-se a construção da arquitetura de *software*, através do desenvolvimento de algoritmos em ambiente simulado, baseados na estrutura ROS, permitindo que esta fosse imposta de igual modo, a todos os modelos existentes e a novos, cumprindo com o fator de expansibilidade necessário num projeto deste tipo. Por último, foram realizados testes aos modelos individualmente e em grupo, em diversas tarefas, com possível aplicação a um modelos destes, de modo a validar o código desenvolvido, como a sua camada funcional.

Resumidamente, a grande maioria dos objetivos propostos no inicio da dissertação foi realizada com sucesso, nomeadamente: a implementação de um ambiente de simulação, inicialmente aos veículos individualmente; implementação de um ambiente de simulação com diversos veículos e uma respetiva coordenação hierárquica.

No entanto, a realização dos objetivos com maior impacto para o desenvolvimento do projeto não foram possível realizar, como a adaptação ao modelo físico e os seus respetivos testes, devido à falta de material fornecido, o que revelou ser um fator bastante negativo, pois este permitia uma melhor aferição de aspetos essenciais ao seu correto funcionamento.

Pessoalmente, foi adquirida bastante experiência para a resolução de problemas, bem como aquisição de conhecimento e sensibilidade para o desenvolvimento de sistemas deste tipo.

7.1 Dificuldades

Durante o desenvolvimento deste projeto foram sendo apresentadas algumas dificuldades relacionadas com o processo de criação de um sistema que utiliza diversas linguagens para o seu funcionamento. Inicialmente identificaram-se dificuldades na operação do sistema operativo Linux, devido ao contacto prévio quase inexistente com este sistema operativo. Embora esta dificuldade não ser algo que pudesse ser extinguida, foi demonstrando um menor nível de impacto ao longo do projeto.

Outra dificuldade, foi a operação com diversas linguagens de programação, como *Python* e a arquitetura do ROS, onde a aprendizagem da sua maneira de funcionar teve que ser feita de forma autónoma, relevando ser um fator de grande impacto temporal.

Com as experiências aos modelos e aos cenários, percebeu-se que ao correr a simulação, o fator temporal com o passar do tempo, começava a apresentar um desfasamento cada vez maior entre o tempo de simulação e o tempo real, o que poderá ser devido às capacidades do computador utilizado, sendo um fator muito limitativo na eficácia do ambiente de simulação.

Como referido anteriormente, a maior dificuldade apresentada foi a ausência de material físico para a construção dos modelos.

Embora serem diversas as adversidades apresentadas durante o desenvolvimento deste dissertação, a maioria destas foi ultrapassada ou mitigada a um ponto de impacto muito reduzido.

7.2 Trabalho futuro

Como trabalhos futuros, recomenda-se o desenvolvimento de temáticas a nível de *hardware* e afinações e desenvolvimento de temáticas a nível de *software*.

Relativamente à temática do *hardware*, propõe-se a aquisição de material de construção dos diversos modelos, o que não deve ser um fator muito limitativo, pois o objetivo da construção destes modelos, é estes serem de baixo-custo. Com a possibilidade da aquisição de material, pode ser feita uma melhor avaliação do material a ser utilizado, como a aquisição de material para a construção de veículos de subsuperfície, pois são nestes que a segunda parte do projeto SABUVIS se foca.

Quanto à temática do *software*, recomenda-se uma possível afinação diversos nós presentes em cada veículo, principalmente aos nós de coordenação, que poderá ser embutido um sistema mais robusto, como a confirmação da posição de um veículo, quando se dá a perda de comunicações, através da confirmação obtida por diversos veículos e não diretamente e do nó de planeamento, bem como a adaptação de realização de situações por comando e não pré-determinadas.

Relativamente ao controlador imposto a cada um dos veículos, este poderá ser afinado e imposto de modo diferente, através de um controlo transversal e longitudinal separado, o que permitirá uma melhor atuação por parte de cada veículo no deslocamento entre alvos.

É necessário também uma adaptação do código construído, de modo a que este possa ser aplicado à placa *Jetson Nano*. No funcionamento do veículo é necessário uma mudança de valores utilizados, pois estes são todos provenientes do Gazebo, havendo a necessidade de mudança da origem destes, sendo alguns destes como a posição, que terá que ser dada por GPS ou pela IMU imposta no veículo; as rotações dos motores, neste projeto são calculadas através de um numeral, não tendo em conta fatores como a corrente, que será determinada pela ESC.

Bibliografia

Albani, Dario, Joris IJsselmuiden, Ramon Haken e Vito Trianni (2017). «Monitoring and mapping with robot swarms for agricultural applications». Em: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, pp. 1–6.

Altshuler, Yaniv, Vladimir Yanovsky, Israel A Wagner e Alfred M Bruckstein (2005). «The cooperative hunters—efficient cooperative search for smart targets using UAV swarms». Em: *The First International Workshop on Multi-Agent Robotic Systems*. Vol. 2. SCITEPRESS, pp. 165–170.

Araújo, Hilário Filipe Rocha (2020). «Projeto e desenvolvimento de um veículo autónomo biomimético de subsuperfície - Projeto e desenvolvimento do hardware e software». Escola Naval, Dissertação para obtenção do Grau de Mestre em Ciências Militares Navais, na especialidade de Engenharia Naval Ramo de Armas e Eletrónica.

Arkin, Ronald Craig (1987). «Towards Cosmopolitan Robots: Intelligent Navigation in extendend man-made environments». University of Massachusetts, in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY.

Arkin, Ronald Craig e Tucker Balch (1970). *AuRA: Principles and Pratice in Review*. Rel. téc. College of Computing Georgia Institute of Technology.

AUVAC (2014). *AUV System Spec Sheet*. `https://auvac.org/179-2/`. [Acedido a : 2023-02-02].

Barca, Jan Carlo e Y. Ahmet Sekercioglu (mai. de 2013). «Swarm robotics reviewed». Em: *Robotica* 31 (3), pp. 345–359. ISSN: 02635747. DOI: `10.1017/S026357471200032X`.

Bayindir, Levent (jan. de 2016). «A review of swarm robotics tasks». Em: *Neurocomputing* 172, pp. 292–321. ISSN: 18728286. DOI: `10.1016/j.neucom.2015.05.116`.

Boeck, Florin, Matthias Golz, Sebastian Ritz e Gerd Holbach (2014). «SMIS - Subsea Monitoring via Intelligent Swarms, Design Challenges of an Autonomous Seabed Station». Em: *The American Society of Mechanichal Engineers*, p. 8.

BotnRoll (s.d.). *10 DOF IMU Sensor*. https://www.botnroll.com/pt/. [Acedido a : 2023-03-22].

Brambilla, Manuele, Eliseo Ferrante, Mauro Birattari e Marco Dorigo (2013). «Swarm Intell Swarm robotics: a review from the swarm engineering perspective». Em: *Swarm Intelligence* 7 (1), pp. 1–41. DOI: 10.1007/s11721-012-0075-2Ãŕ. URL: https://hal.archives-ouvertes.fr/hal-01405919.

Carinecs (2018). *Robolobsters: Bio-mimetic Underwater Robot Program*. https://sprink.carinecsq.com/2018/02/22/robolobsters-bio-mimetic-underwater-robot-program/. [Acedido a : 2023-02-02].

Carpin, Stefano, Mike Lewis, Jijun Wang, Stephen Balakirsky e Chris Scrapper (2007). «USARSim: a robot simulator for research and education». Em: *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1400–1405. DOI: 10.1109/ROBOT.2007.363180.

Cheein, Fernando Alfredo Auat e Ricardo Carelli (2013). «Agricultural robotics: Unmanned robotic service units in agricultural tasks». Em: *IEEE industrial electronics magazine* 7.3, pp. 48–58.

Chen, Ji, Ruojia Sun e Hadas Kress-Gazit (ago. de 2021). «Distributed Control of Robotic Swarms from Reactive High-Level Specifications». Em: vol. 2021-August. IEEE Computer Society, pp. 1247–1254. ISBN: 9781665418737. DOI: 10.1109/CASE49439.2021.9551578.

Cheraghi, Ahmad Reza, Sahdia Shahzad e Kalman Graffi (jan. de 2021). «Past, Present, and Future of Swarm Robotics». Em: URL: http://arxiv.org/abs/2101.00671.

Circuits, All About (s.d.). *Arduino vs. Raspberry Pi vs. STM32: Which Board is Right for Your Project?* https://www.allaboutcircuits.com/news/arduino-vs-raspberry-pi-vs-stm32-which-board-is-right-for-your-project/. [Acedido a : 2023-03-24].

Comittee, International Olimpic (2021). *https://olympics.com/ioc/news/spectacular-intel-drone-light-show-helps-bring-tokyo-2020-to-life-1*. https://olympics.com/ioc/news/spectacular-intel-drone-light-show-helps-bring-tokyo-2020-to-life-1. [Acedido a : 2023-04-15].

Corke, Peter (2011). *Robotics, vision and control: fundamental algorithms in MATLAB*. Springer Science & Business Media.

Costa, Vasco, Miguel Duarte, Tiago Rodrigues, Sancho Moura Oliveira e Anders Lyhne Christensen (jun. de 2016). «Design and development of an inexpensive aquatic swarm robotics system». Em: Institute of Electrical e Electronics Engineers Inc. ISBN: 9781467397247. DOI: 10.1109/OCEANSAP.2016.7485496.

Cotarelo, Sergi Hernandez Juan Fernando Herrero e Sergi Hernandez Juan (2015). *Multi-master ROS systems*. URL: http://www.iri.upc.eduhttp://www.iri.upc.edu/staff/shernand.

Dorigo, M. (2005). «SWARM-BOT: an experiment in swarm robotics». Em: *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* Pp. 192–200. DOI: 10.1109/SIS.2005.1501622.

Dorigo, M., V. Maniezzo e A. Colorni (1996). «Ant system: optimization by a colony of cooperating agents». Em: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1, pp. 29–41. DOI: 10.1109/3477.484436.

EUSPA (2021). *What is GNSS?* https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss. [Acedido a : 2023-03-22].

Gazebo (s.d.). *Gazebo Tutorials*. https://classic.gazebosim.org/tutorials?cat=sensors. [Acedido a : 2023-05-16].

Gerndt, Reinhard e Stefan Krupop (2010). *EcoBe! Mixed Reality Robot Kit-An Entry-Level System for Teaching Cooperative Robotics*. ISBN: 9783000328633. URL: https://www.researchgate.net/publication/228458125.

GlobalData (s.d.). *Wi-fi Adaptor*. https://www.globaldata.pt/adaptador-usb-tp-link-tl-wn722n-wi-fi-n150-high-gain-usb-20-tl-wn722n. [Acedido a : 2023-03-22].

Gooley, Petty Officer 2nd Class Thomas (2021). *Sea Hunter medium displacement unmanned surface vessel launch*. https://www.dvidshub.net/image/6608558/sea-hunter-medium-displacement-unmanned-surface-vessel-launch-during-uxs-ibp-21. [Acedido a : 2023-06-21].

HobbyKing (s.d.). *HobbyKing Website*. https://hobbyking.com/pt_pt/?___store=pt_pt. [Acesso a : 2023-03-22].

Hsu, Jeremy (2014). *A swarm of autonomous boats could escort larger ships in the future*. https://spectrum.ieee.org/us-navy-robot-boat-swarm. [Acedido a : 2023-04-15].

Hu, Justin, Ariana Bruno, Drew Zagieboylo, Mark Zhao, Brian Ritchken, Brendon Jackson, Joo Yeon Chae, Francois Mertil, Mateo Espinosa e Christina Delimitrou (mai. de 2018). «To Centralize or Not to Centralize: A Tale of Swarm Coordination». Em: URL: http://arxiv.org/abs/1805.01786.

Huet, Natalie (2022). *Switchblade drones: What are these US-made 'kamikaze' weapons being sent to Ukraine?* https://www.euronews.com/next/2022/05/16/switchblade-drones-what-are-these-kamikaze-weapons-and-how-can-they-help-ukraine. [Acedido a : 2023-01-28].

IEEERobots (2010). *Aqua2.* https://robots.ieee.org/robots/aqua/?gallery=photo1. [Acedido a : 2023-02-02].

Inc., Microchip Technology (s.d.). *What is a Microcontroller?* https://www.microchip.com/design-centers/8-bit. [Acedido a : 2023-03-24].

International, Nuclear Engineering (2021). *Spot the robot.* https://www.neimagazine.com/features/featurespot-the-robot-8752895/. [Acedido a : 2023-02-02].

Iocchi, Luca, Daniel Nardi e Massimiliano Salerno (2001). «Reactivity and Deliberation: A Survey on Multi-Robot Systems». Em: *Journal of Robotics and Autonomous Systems*, pp. 9–34.

Johnson, James (fev. de 2020). «Artificial Intelligence, Drone Swarming and Escalation Risks in Future Warfare». Em: *RUSI Journal* 165 (2), pp. 26–36. ISSN: 17440378. DOI: 10.1080/03071847.2020.1752026.

Kang, Chang-kwon, Farbod Fahimi, Rob Griffin, D Brian Landrum, Bryan Mesmer, Guangsheng Zhang, Taeyoung Lee, Hikaru Aono, Jeremy Pohly, Jesse McCain et al. (2019). *Marsbee-Swarm of Flapping Wing Flyers for Enhanced Mars Exploration.* Rel. téc.

Khaldi, Belkacem e Foudil Cherif (2015). *An Overview of Swarm Robotics: Swarm Intelligence Applied to Multi-robotics*, pp. 975–8887. URL: http://gazebosim.org/.

Koenig, N. e A. Howard (2004). «Design and use paradigms for Gazebo, an open-source multi-robot simulator». Em: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566).* Vol. 3, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

Koos, Sylvain, Jean-Baptiste Mouret e Stéphane Doncieux (2013). «The transferability approach: Crossing the reality gap in evolutionary robotics». Em: *IEEE Transactions on Evolutionary Computation* 17.1, pp. 122–145.

Liang, Xiao, Xingru Qu, Ning Wang, Ye Li e Rubo Zhang (nov. de 2019). «Swarm control with collision avoidance for multiple underactuated surface vehicles». Em: *Ocean Engineering* 191. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2019.106516.

Liarokapis, Minas V e Martin White (2013). «Developing intelligent robots using simulation». Em: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, pp. 1857–1862.

Martinez, Kennedy (2016). *Drones Dazzle Around the World*. https://www.dronethusiast.com/around-the-world-people-dazzle-with-drone-lighting-and-art/. [Acedido a : 2023-04-15].

Michel, Olivier (2004). «Cyberbotics ltd. webots™: professional mobile robot simulation». Em: *International Journal of Advanced Robotic Systems* 1.1, p. 5.

Morenville, Achille ; Vermeulen e Lucas (2022). *"Simultaneous localization and mapping and autonomous navigation on SPOT robot from Boston Dynamics"*. URL: http://hdl.handle.net/2078.1/thesis:35688.

NEO-7 Datasheet (2015). URL: www.u-blox.com.

NVIDIA (s.d.). *Jetson Nano Developer Kit*. https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Acedido a : 2023-03-24].

Osborn, Kris (2021). *America's F-35: Soon Armed With Drone Swarms?* https://nationalinterest.org/blog/reboot/americas-f-35-soon-armed-drone-swarms-181976. [Acedido a : 2023-02-03].

Paravisi, Marcelo, Davi H. Santos, Vitor Jorge, Guilherme Heck, Luiz Marcos Gonçalves e Alexandre Amory (2019). «Unmanned surface vehicle simulator with realistic environmental disturbances». Em: *Sensors (Switzerland)* 19 (5). ISSN: 14248220. DOI: 10.3390/s19051068.

Pinciroli, Carlo, Rehan O'grady, Anders Lyhne Christensen e Marco Dorigo (2009). *Self-Organised Recruitment in a Heteregeneous Swarm*.

Pinciroli, Carlo, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle et al. (2011). «ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics». Em: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5027–5034.

Platypus (s.d.). *The Lutra Prop*. http://senseplatypus.com/lutra-prop. [Acedido a : 2023-05-16].

Prats, Miguel, Jose Perez, Jose Javier Fernandez e Pedro Jose Sanz (2012). «An open source tool for simulation and supervision of underwater intervention missions». Em: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2577–2582.

Przybylski, Michał, Piotr Szymak, Zygmunt Kitowski e Paweł Piskur (2020). «Comparison of Different Course Controllers of Biomimetic Underwater Vehicle with Two Tail Fins». Em: vol. 1196 AISC. Springer, pp. 1507–1518. ISBN: 9783030509354. DOI: 10.1007/978-3-030-50936-1_125.

Qin, Zihe, Zhuang Lin, Dongmei Yang e Ping Li (abr. de 2017). «A task-based hierarchical control strategy for autonomous motion of an unmanned surface vehicle swarm». Em: *Applied Ocean Research* 65, pp. 251–261. ISSN: 01411187. DOI: 10.1016/j.apor.2017.04.013.

Rohmer, E., S.P.N. Singh e M. Freese (2013). «V-REP: A versatile and scalable robot simulation framework». Em: *Simulation, Modeling, and Programming for Autonomous Robots* 73.2, pp. 385–400.

Sano, Yoshinori, Takahiro Endo, Takumi Shibuya e Fumitoshi Matsuno (2023). «Decentralized navigation and collision avoidance for robotic swarm with heterogeneous abilities». Em: *Advanced Robotics* 37.1-2, pp. 25–36. DOI: 10.1080/01691864.2022.2117996. URL: https://doi.org/10.1080/01691864.2022.2117996.

SCHLEI, EDSON ELMAR (2002). «Uma linguagem para definiçãoo de estratégias de controle de times de robôs jogares de futebol em um ambiente simulado». Em: *Trabalho de Conclusão de Curso, Universidade Regional de Blumenau. Blumenau*.

Schmickl, Thomas, Ronald Thenius, Christoph Möslinger, Jon Timmis, Andy Tyrrell, Mark Read, James Hilder, Jose Halloy, Cesare Stefanini, Luigi Manfredi, Alexandre Campo, Tobias Dipper, Donny Sutantyo e Serge Kernbach (2011). *CoCoRo-The Self-aware Underwater Swarm*.

Soria, Enrica, Fabrizio Schiano e Dario Floreano (2020). «SwarmLab: A MATLAB drone swarm simulator». Em: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 8005–8011.

Sousa, Alexandre, Luis Madureira, Jorge Coelho, José Pinto, João Pereira, João Borges Sousa e Paulo Dias (2012). «LAUV: The Man-Portable Autonomous Underwater Vehicle». Em: *IFAC Proceedings Volumes* 45.5. 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles, pp. 268–274. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20120410-

3-PT-4028.00045. URL: https://www.sciencedirect.com/science/article/pii/S1474667016306140.

Sousa, Daniela, Diego Hernandez, Francisco Oliveira, Miguel Luís e Susana Sargento (nov. de 2019). «A platform of unmanned surface vehicle swarms for real time monitoring in aquaculture environments». Em: *Sensors (Switzerland)* 19 (21). ISSN: 14248220. DOI: 10.3390/s19214695.

Sutton, H. I. (2019). *Guide to Biomimetic Underwater Vehicles*. http://www.hisutton.com/Biomimetic_Autonomous_Underwater_Vehicles.html. [Acedido a : 2023-06-21].

Tan, Ying (2013). «Swarm Robotics: Collective Behavior Inspired by Nature». Em: DOI: 10.4172/jcsb.1000e106. URL: http://dx.doi.org/10.4172/jcsb.1000e106.

Vavasseur, Xavier (2022). *Here is our first look at the US Navy's Orca XLUUV*. https://www.navalnews.com/naval-news/2022/05/here-is-our-first-look-at-the-us-navys-orca-xluuv/. [Acedido a : 2023-06-21].

Vincent, James (2016). *The US Navy's new autonomous warship is called the Sea Hunter*. https://www.theverge.com/2016/4/8/11391840/us-navy-autonomous-ship-sea-hunter-christened. [Acedido a : 2023-06-21].

Zhou, Xin, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu e Fei Gao (2022). «Swarm of micro flying robots in the wild». Em: *Science Robotics* 7.66, eabm5954. DOI: 10.1126/scirobotics.abm5954. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm5954.

Apêndice A - Tutorial de utilização do simulador

A.1 Introdução

O USVSim é um simulador criado em 2019, com o intuito de ser um simulador de veículos de superfície com interações físicas com o meio envolvente, permitindo assim haver uma plataforma que pudesse simular este tipo de cenários, pois a atual propensão de simuladores está muito restringida a simuladores de veículos aéreos, sub-aquáticos ou terrestres, sendo muito limitada a escolha de um simulador que crie um veículo de superfície com interações físicas com o meio.

Este documento serve assim, como base de apoio para os primeiros passos de utilização do simulador USVSim criado por Paravisi et al. 2019. Baseia-se apenas na utilização corrente do simulador para o desenvolvimento deste projeto, não sendo documentação criada pelos autores deste programa, mas criada com o propósito de poder ajudar na sua utilização, demonstrando alguns pontos de dificuldades encontradas e algumas maneiras de as superar. Pode-se encontrar mais informações na página do GitHub dos criadores do simulador. (`https://github.com/disaster-robotics-proalertas/usv_sim_lsa`)

A.2 Pré-requisitos

Para o simulador poder ser instalado e utilizado corretamente, é necessário utilizar o Ubuntu Linux 16.04 pois este suporta a versão ROS Kinetic. Após o sistema operativo ser o necessário para correr o simulador, uns dos primeiros passos é a instalação do Gazebo, pois este será a base do ambiente de simulação, esta é feita em conjunto com a instalação do ROS. A instalação da versão do ROS Kinetic e algumas das suas dependências é feita copiando as seguintes linhas de comando no terminal:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
  $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-
  latest.list'
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80'
  --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
sudo apt-get update
sudo apt-get install ros-kinetic-desktop-full ros-kinetic-
```

```
    control-*ros-kinetic-osg-markers ros-kinetic-move-base -y
sudo rosdep init
rosdep update
sudo echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Após ter sido corridas estas linhas, sem a presença de erros, passamos ao passo de transferência de dependências do simulador.

```
sudo apt-get install python-rosinstall python-rosinstall-
generator python-wstool build-essential python-rosdep python
-wxtools python-lxml python-pathlib python-h5py python-scipy
python-geolinks python-gdal -y

sudo apt-get install libfftw3-* libxml++2.6-* libsdl-image1.2-
dev libsdl-dev -y
```

Após a realização destes dois passos, os requisitos estão terminados e podemos passar à instalação dita do simulador. Recomenda-se a atenção para a ausência de qualquer mensagem de erro que possa aparecer durante qualquer passo anterior e para os próximos, pois poderá ser um fator impeditivo no funcionamento regular do simulador no futuro. Antes ou após destes passos, é recomendado que a habituação ao sistema Linux seja feita anteriormente ao uso regular do simulador, pois baixará o nível de dificuldade que será a parte do troubleshooting.

A.3 Preparação

Nestes próximos passos, passamos à instalação do simulador e à primeira demonstração do seu funcionamento. De modo a correr o simulador e os seus pacotes é necessário a criação de um ambiente de trabalho *catkin* (mais conhecido como *catkin workspace*).

Um ambiente de trabalho catkin fornece uma maneira conveniente de organizar, construir e gerir pacotes de ROS, permitindo assim maior desenvolvimento

eficiente de software. Catkin é o sistema de construção de pacotes e ferramentas do ROS, criando assim um ambiente dedicado para o seu desenvolvimento.

Para a criação de um ambiente catkin, é necessário correr as seguintes linhas de comando:

```
source /opt/ros/kinetic/setup.bash
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

Para transferir e instalar o simulador nesse ambiente é necessário correr as seguintes linhas de comando:

```
cd ~/catkin_ws/src
git clone
 https://github.com/disaster-robotics-proalertas/usv_sim_lsa.git
cd usv_sim_lsa
git submodule init
git submodule update
```

Correr o documento de instalação:

```
cd ~/catkin_ws/src/usv_sim_lsa
chmod +x ./install_usv_sim
./install_usv_sim
```

Instalar as dependências:

```
rosdep install --from-paths src --ignore-src --rosdistro
  kinetic -y
```

E realizar a sua compilação:

```
cd ~/catkin_ws/
catkin_make_isolated --install
source install_isolated/setup.bash
```

Por fim, correr um dos cenário presentes no simulador de modo a averiguar a instalação e funcionamento eficaz:

```
roslaunch usv_sim airboat_scenario1.launch parse:=true
roslaunch usv_sim airboat_scenario1.launch parse:=false
```

A partir deste momento, se não houveram nenhuns problemas e a simulação iniciou corretamente, para se usar o simulador apenas é necessário correr estas últimas linhas de comando, pois estas realizam as seguintes funções:

- **roslaunch usv_sim airboat_scenario1.launch parse:=true**: cria o ambiente de simulação com o veículo e com o meio envolvente a partir do ficheiro de *scene spawner*;
- **roslaunch usv_sim airboat_scenario1.launch parse:=false**: inicia a simulação e abre a janela do mostrador visual da simulação.

A.4 Funcionalidades

No simulador já estão incluídos alguns cenários pré-existentes (figura A.1), com diversas tarefas a a serem realizadas pelos veículos, como por exemplo a navegação entre boias.

Cada cenário poder ser atribuído a um veículo pré-existente, sendo estes são quatro, demonstrados na figura A.2 por ordem, cada um com características físicas diferentes, sendo estas:

- *Rudderboat*: veículo com um veio propulsor e um leme;
- *Airboat*: veículo movido por uma ventoinha acima do casco;
- *Diffboat*: veículo com propulsão diferencial, com dois veios;

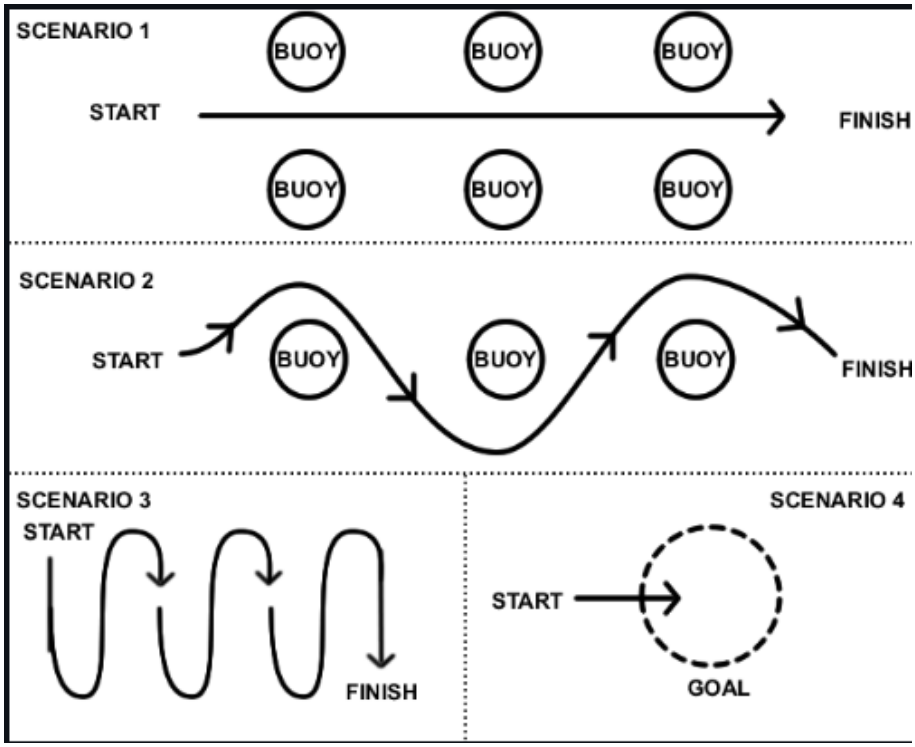


FIGURA A.1: Diferentes cenários pré-existentes no simulador.

Fonte: Paravisi et al. 2019.

- *Sailboat*: veículo com propulsão à vela e com leme.

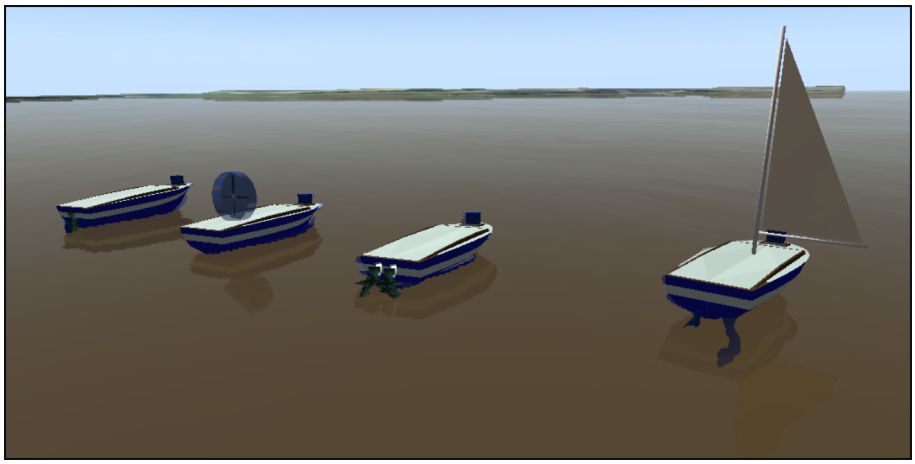


FIGURA A.2: Diferentes modelos pré-existentes no simulador.

Fonte: Paravisi et al. 2019.

A.4.1 Criação de modelos e cenários

A criação e adição de novos modelos também è possível, desde que a adição de todos os ficheiros de funcionamento deste sejam adicionados corretamente e nas devidas dependências. Para a adição, modificação ou criação de novos modelos,

existe um conjunto necessário de ficheiros que têm que estar presentes para o seu correto funcionamento, sendo estes:

- Ficheiro **.xacro**: acessível a partir de: **Home > catkin_ws > install_isolated > share > usv_sim > xacro**. Deve ser adicionado um ficheiro, com o nome do modelo finalizado de .xacro. Este ficheiro adiciona os plugins necessários ou pretendidos para esse mesmo modelo.
- Ficheiro **.yaml**: acessível a partir de: **Home > catkin_ws > install_isolated > share > usv_sim > config**. Deve ser adicionado um ficheiro, com o nome do modelo finalizado de .yaml. Este ficheiro adiciona a configuração do modelo, como a constituição do seu corpo e das suas juntas, para ser possível a comunicação com o Gazebo.
- Ficheiro **.launch**: acessível a partir de: **Home > catkin_ws > install_isolated > share > usv_sim > launch > models**. Deve ser adicionado um ficheiro com o nome *spawn_modelname.launch*, que será responsável pela criação do modelo no cenário. Este ficheiro contém os nós pertencentes ao modelo, como alguns objetos de comunicação com o Gazebo.
- Ficheiro **dummy.urdf**: acessível a partir de: **Home > catkin_ws > install_isolated > share > usv_sim > launch > urdf**. Deve ser adicionado um ficheiro com o nome *modelname_dummy.urdf*, que indica as características do veículo como as suas dimensões e a interação com o meio.
- Ficheiro **.urdf**: acessível a partir de: **Home > catkin_ws > install_isolated > share > usv_sim > launch > urdf**. Criado automaticamente um ficheiro do tipo *modelname_uwsim.urdf*, após a primeira vez que o modelo corre num cenário. Este ficheiro é responsável pelas interações físicas do modelo com o ambiente envolvente, através da interação com o UWSim e pelas dimensões do veículo.

Após a adição destes ficheiros, a criação do modelo está acabada. Para confirmar a criação correta do modelo e que este funciona dentro do previsto. recomenda-se que se crie um cenário novo, de modo a testar o seu funcionamento. Para se criar um novo cenário, é necessário a criação dos seguintes ficheiros:

- Ficheiro **.xml**: acessível a partir de:**Home > catkin_ws > install_isolated > share > usv_sim > scenes**. Ficheiro que cria o cenário (criando o ambiente e o modelo). Neste ficheiro é necessário a adição do modelo e de todas as untas que constituem este.

- Ficheiro **.launch**: acessível a partir de: **Home > catkin_ws > install_isolated > share > usv_sim > launch > scenarios_launch**. Ficheiro responsável por lançar o cenário com o modelo funcional do veículo.

No ficheiro que lança o cenário, a linha presente no nó do veículo que tem a adição de um novo ficheiro, **patrol_pid_scene1.py**, está responsável por relacionar o modelo com os ficheiros de navegação, que podem ser acedidos a partir de: **Home > catkin_ws > install_isolated > share > usv_navegation > scripts**. Nesta pasta, temos os ficheiros responsáveis pela navegação dos modelos nos diferentes cenários.

A.4.2 Exemplo

Servindo de exemplo, criamos um cenário que tenha o nome de **myscenario** com o modelo **mymodel**. Criamos o ficheiro que cria o ambiente e o modelo **mymodel_myscenario.xml** e o ficheiro **mymodel_myscenario.launch**. Neste ficheiro temos que criar um **<include>** e um **<node>** para o modelo utilizado, com a seguinte estrutura:

```
<node name="node" pkg="usv_navigation" type="patrol_pid_scene1.py"
ns="modelname" unless="$(arg gui)">
  <include file="$(find usv_sim)/launch/models/spawn_mymodel.launch">
            <arg name="gui" value="$(arg gui)"/>
            <arg name="spawnGazebo" value="$(arg spawnGazebo)"/>
            <arg name="namespace" value="$(arg mymodel)"/>
            <arg name="windType" value="global"/>
            <arg name="waterType" value="global"/>
        </include>
```

Após serem criados estes ficheiros e adicionadas estas linhas no ficheiro **mymodel_myscenario.launch**, devemos correr as seguintes linhas no terminal de modo a dar início à simulação, e observar se esta corre corretamente:

```
    roslaunch usv_sim mymodel_myscenario.launch parse:=true
    roslaunch usv_sim mymodel_myscenario.launch parse:=false
```

A.5 Problemas

A habituação a um novo sistema operativo e a um simulador com um interface diferente do que se tem nos dias de hoje pode levar ao aparecimento de mensagens de erro, a funcionalidades correm de uma maneira incorreta e programas não abrirem.

Nesta secção são abordados alguns problemas que foram aparecendo durante a habituação ao sistema operativo, ao simulador e à modificação e criação dos ficheiros para outros modelos. Algumas destas irregularidades são as seguintes:

- Linhas de comando no terminal: ter em especial atenção ao funcionamento do terminal e da maneira como são escritas as linhas de comando, pois muitas vezes estas poderão ser copiadas em duplicado, não realizando o esperado.
- Mau funcionamento do Gazebo: como o simulador corre num sistema operativo com algum tempo, quando são iniciados os cenários, existem muitos processos a correr instantaneamente, o que poderá levar à execusão incorreta do Gazebo. Caso este não abra ou corra de maneira não esperada, fechar a simulação, noutra janela do terminal escrever "**Gazebo**", e este deverá abrir corretamente. De seguida fechar, e correr de novo a simulação.
- Execução incorreta de ficheiros: com a adição e modificação dos ficheiros dos modelos no simulador consoante as necessidades do simulador, poderão haver ficheiros que não correm corretamente devido a erros de código.
  - Caso ao correr uma simulação, na janela no terminal aparecer uma mensagem do tipo "**INTENTIFING CONTROL**", este erro está relacionado com o ficheiro **spawner.launch**.
  - Se for o caso de adição de um novo nó ao modelo com a finalidade de uma tarefa, ao executar a simulação, noutra janela do terminal escrever "**rosnode list**"e observar se o nó aparece. Caso não apareça, poderá ser devido a um problema de código.
  - Se for pretendido a passagem de tópicos entre nós e estes não forem devidamente passados, noutra janela do terminal escrever "**rostopic list**"e observar se o tópico pretendido aparece. Caso não apareça, poderá ser devido a um problema de código.

A.6 Recursos

Para a instalação, modificação e utilização correta deste simulador, certa documentação dever ser lida e observada, de modo a auxiliar o seu funcionamento correto. Neste documento são fornecidos alguns links que poderão servir de ajuda para a utilização deste simulador.

Para um enquadramento do Gazebo:

- Gazebo Tutorials: `https://classic.gazebosim.org/tutorials`.
- Gazebo User Guide: `https://gazebosim.org/user_guide`.
- Gazebo API Documentation: `http://osrf-distributions.s3.amazonaws.com/gazebo/api/10.0.0/index.html`.
- Gazebo Plugins: `https://classic.gazebosim.org/tutorials?tut=ros_gzplugins`.
- Gazebo Models: `http://models.gazebosim.org/`.
- Gazebo ROS Wiki: `http://wiki.ros.org/gazebo`.

Para um enquadramento do Linux:

- Linux Documentation Project (LDP): `https://tldp.org/`.
- Linux manual pages: `https://manpages.ubuntu.com/`.
- Ubuntu Documentation: `https://help.ubuntu.com/`.
- Debian Wiki: `https://wiki.debian.org/`.

Para um enquadramento do ROS:

- ROS Wiki: `http://wiki.ros.org/`.
- ROS Documentation: `https://docs.ros.org/`.
- ROS Answers: `https://answers.ros.org/questions/`.
- ROS GitHub Repositories: `https://github.com/ros`.

Para aceder à página dos criadores do simular, aceder: `https://github.com/disaster-robotics-proalertas/usv_sim_lsa`

A.7 Conclusão

Este documento serve como auxílio à operação do simulador USVSim, de um ponto de vista de um utilizador e com algumas ajudas de problemas comuns que foram aparecendo durante a utilização do simulador para o desenvolvimento deste projeto.

Apêndice B - Tabela de Tópicos utilizados

Namespace	Nome do Tópico	Tipo de Mensagem	Origem	Destino	Parametros Passados	Observações
/usv1	/state	nav_msgs/Odometry	/gazebo1	/usv1/usv1_heading_control /usv1/usv1_odom_relay /usv2/usv2_heading_control /usv3/usv3_heading_control	posição espacial do veículo	GERAL
	/wp_reached	std_msgs/Float64	/usv1/usv1_heading_control	/patrol	valor de 0 ou 1	GERAL
	/desired_speed	std_msgs/Float64	/usv1/usv1_heading_control	/usv1/usv1_actuator_control	velocidade desejada dependo da distância ao wp	LOCAL
	/thruster_command	sensor_msgs/JointState	/usv1/usv1_actuator_control	/gazebo1	valor a aplicar em cada motor	LOCAL
	/thruster_use	sensor_msgs/JointState	/gazebo1	-	atuação independente do gazebo	LOCAL
	/usv1_value	std_msgs/Int32	/usv1/usv1_coordination	/usv2/usv2_coordination /usv3/usv3_coordination /usv1/usv1_planner_control	indica o valor do usv1 no grupo	GERAL
/usv2	/state	nav_msgs/Odometry	/gazebo	/usv2/usv2_heading_control /usv2/usv2_odom_relay /usv1/usv1_heading_control /usv3/usv3_heading_control	posição espacial do veículo	GERAL
	/wp_reached	std_msgs/Float64	/usv2/usv2_heading_control	/patrol	valor de 0 ou 1	GERAL
	/desired_speed	std_msgs/Float64	/usv2/usv2_heading_control	/usv2/usv2_actuator_control	velocidade desejada dependo da distância ao wp	LOCAL
	/thruster_command	sensor_msgs/JointState	/usv2/usv2_actuator_control	/gazebo	valor a aplicar em cada motor	LOCAL
	/thruster_use	sensor_msgs/JointState	/gazebo	-	atuação independente do gazebo	LOCAL
	/usv2_value	std_msgs/Int32	/usv2/usv2_coordination	/usv1/usv1_coordination /usv3/usv3_coordination /usv2/usv2_planner_control	indica o valor do usv2 no grupo	GERAL
/usv3	/state	nav_msgs/Odometry	/gazebo2	/usv3/usv3_heading_control /usv3/usv3_odom_relay /usv1/usv1_heading_control /usv2/usv2_heading_control	posição espacial do veículo	GERAL
	/wp_reached	std_msgs/Float64	/usv3/usv3_heading_control	/patrol	valor de 0 ou 1	GERAL
	/desired_speed	std_msgs/Float64	/usv3/usv3_heading_control	/usv3/usv3_actuator_control	velocidade desejada dependo da distância ao wp	LOCAL
	/thruster_command	sensor_msgs/JointState	/usv3/usv3_actuator_control	/gazebo2	valor a aplicar em cada motor	LOCAL
	/thruster_use	sensor_msgs/JointState	/gazebo2	-	atuação independente do gazebo	LOCAL
	/usv3_value	std_msgs/Int32	/usv3/usv3_coordination	/usv2/usv2_coordination /usv1/usv1_coordination /usv3/usv3_planner_control	indica o valor do usv3 no grupo	GERAL
/patrol	/goal	nav_msgs/Odometry	/patrol	/usv1/usv2_heading_control /usv2/usv2_heading_control /usv3/usv3_heading_control	trasnmite os wp para os veículos e vê se algum já chegou ao wp para transmitir o próximo	GERAL

FIGURA B.1: Tabela de tópicos transmitidos no funcionamento do simulador.