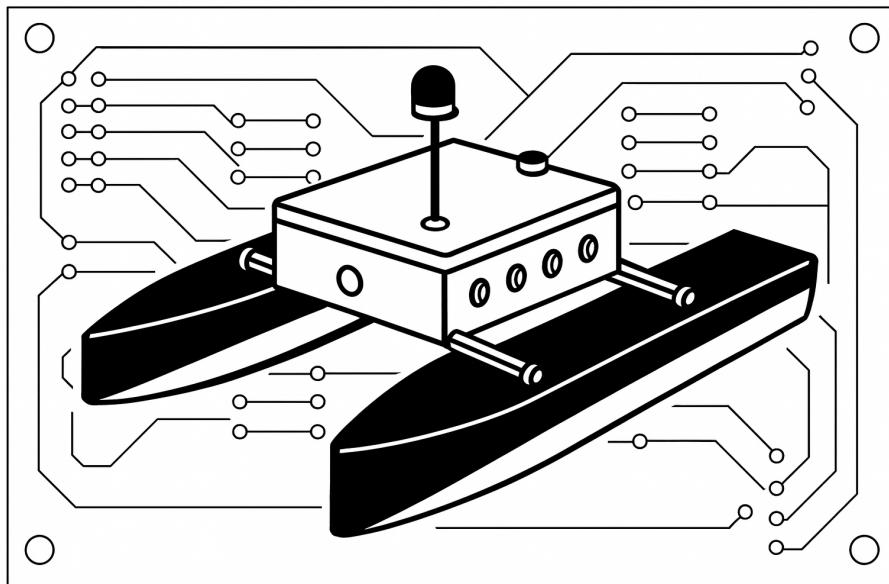




# ISEL



## Sistema Ciberfísico para Controlo Autónomo de Embarcações Unmanned Surface Vehicle

ALEXANDRE MIGUEL CALEJO DE FIGUEIREDO  
Licenciado em Engenharia Informática e Multimédia

Dissertação para obtenção do grau de mestre em Engenharia Informática e Multimédia

Orientadores:

Professor Doutor Carlos Jorge de Sousa Gonçalves  
Professor Doutor Pedro Daniel Dinis Teodoro

Júri:

Presidente: Grau e Nome

Vogais:

Grau e Nome

Grau e Nome

Setembro de 2025



# **Sistema Ciberfísico para Controlo Autónomo de Embarcações Unmanned Surface Vehicle**

**ALEXANDRE MIGUEL CALEJO DE FIGUEIREDO**

Licenciado em Engenharia Informática e Multimédia

Dissertação para obtenção do grau de mestre em Engenharia Informática e Multimédia

Orientador(es):

Professor Doutor Carlos Jorge de Sousa Gonçalves, ISEL

Professor Doutor Pedro Daniel Dinis Teodoro, ENIDH

Júri:

Presidente: Grau e Nome, Afiliação

Vogais:

Grau e Nome, Afiliação

Grau e Nome, Afiliação

**Setembro de 2025**



*"O cientista não procura um resultado imediato. Ele não espera que as suas ideias avançadas sejam prontamente aceites. O seu trabalho é lançar a base para aqueles que virão."*

— Nikola Tesla



# Agradecimentos

Em primeiro lugar, quero expressar a minha profunda gratidão ao Instituto Superior de Engenharia de Lisboa (ISEL), instituição que foi a minha casa ao longo de cinco anos (Licenciatura e Mestrado). Aqui encontrei não só os maiores desafios académicos da minha vida, mas também oportunidades únicas de crescimento pessoal e profissional. Foram muitas noites mal dormidas, momentos de frustração e superação, mas também conquistas que culminaram na realização desta TFM.

Agradeço igualmente à Escola Superior Náutica Infante D. Henrique (ENIDH), que nos últimos meses se tornou a minha segunda casa, disponibilizando não apenas as condições necessárias para o desenvolvimento deste trabalho, mas também a embarcação e os espaços para ensaios. Graças a esta colaboração foi possível participar no exercício Naval-REX25, um subevento do REPMUS25, validando o protótipo em ambiente real.

Quero também agradecer à Marinha Portuguesa e à Escola Naval pela oportunidade de integrar o REPMUS25 e o Naval-REX25, onde este trabalho pôde ser colocado à prova em cenários operacionais exigentes. Esta experiência representou um marco na validação prática do sistema e um reconhecimento da importância de iniciativas académicas em contextos reais.

Um agradecimento muito especial ao Professor Carlos Gonçalves, pela orientação incansável ao longo da licenciatura e do mestrado. O seu rigor, dedicação e espírito de exigência foram fundamentais para a conclusão deste percurso. Recordo as inúmeras noites no ISEL, que se prolongavam até tarde, e que foram decisivas para transformar ideias em resultados concretos.

Agradeço igualmente ao Professor Pedro Teodoro, pelo apoio na compreensão de conceitos ligados às embarcações e pela disponibilização da infraestrutura e embarcação necessárias para a realização dos testes.

À minha família, em particular aos meus pais, agradeço por estarem sempre presentes e pelo apoio incondicional em todos os momentos. Sem o vosso incentivo e confiança, nada disto teria sido possível.

Aos colegas e amigos que me acompanharam ao longo desta jornada académica, deixo também um sincero agradecimento. Foram muitas as vezes em que tive de recusar convites ou abdicar de momentos convosco devido às exigências deste mestrado, mas o vosso apoio e compreensão foram fundamentais para chegar até aqui.

Um agradecimento também à minha empresa, Critical Techworks, pela flexibilidade e pelo apoio concedido durante os dois anos de mestrado. A possibilidade de conciliar a vida profissional com os estudos foi crucial para tornar este percurso viável.

Finalmente, agradeço a mim próprio. Conciliar um mestrado com uma carreira profissional

como trabalhador-estudante foi uma tarefa exigente e, por vezes, desgastante. No entanto, este trabalho é também prova de que, com esforço, dedicação e resiliência, é possível alcançar qualquer objetivo. Agradeço a mim por acreditar, por não desistir e por ser fiel a este caminho até ao fim.

*"Conseguimos! Conseguimos, Portugal, Lisboa! Esperávamos, desejávamos, conseguimos! Vitória!"*

— Professor Doutor Marcelo Rebelo de Sousa

# **Declaração de integridade**

Declaro que esta dissertação é o resultado da minha investigação pessoal e independente. O seu conteúdo é original e todas as fontes listadas nas referências bibliográficas foram consultadas e estão devidamente mencionadas no texto. Mais declaro que todas as referências científicas e técnicas relevantes para o desenvolvimento do trabalho estão devidamente citadas e constam das referências bibliográficas.

O autor

---

Lisboa, 30 de setembro de 2025



# Resumo

O trabalho final de mestrado apresentado neste relatório descreve o desenvolvimento de um sistema ciberfísico para controlo autónomo de embarcações do tipo *Unmanned Surface Vehicle* (USV). O sistema foi concebido de forma modular, expansível e orientado para a integração de múltiplos motores, sensores e interfaces de comunicação, assegurando robustez e adaptabilidade a diferentes cenários operacionais.

A arquitetura proposta baseia-se na utilização de um microcontrolador ESP32 com suporte nativo a I2C, UART e LoRa, para controlo dos diferentes módulos, incluindo sensores e atuadores. Entre os principais módulos incluem-se: i) propulsores *brushless* controlados por *Electronic Speed Controller* (ESC), capazes de fornecer empuxo bidirecional; ii) um *Inertial Measurement Unit* (IMU), que fornece medições de *yaw*, *pitch* e *roll* para garantir a estabilidade e a correção de trajetória; iii) um receptor *Global Positioning System* (GPS), responsável pela navegação precisa através de coordenadas geoespaciais; e iv) um módulo de comunicação *Long Range* (LoRa), que assegura telemetria de longo alcance e receção de rotas em tempo real com baixo consumo energético.

Do ponto de vista de *software*, a solução foi estruturada em bibliotecas independentes que encapsulam funções de propulsão, sensorização, comunicação e controlo. Esta abordagem promove a reutilização de código, facilita a manutenção e garante a escalabilidade do sistema. A comunicação foi otimizada através do uso de *Protocol Buffers* (*Protobuf*), reduzindo significativamente o tempo de transmissão (ToA) em mensagens *Long Range* (LoRa), o que contribui para maior eficiência energética, menor probabilidade de colisões e aumento do alcance efetivo.

A validação experimental incluiu testes incrementais de *hardware* e *software*, desde o controlo isolado de propulsores até à integração de todos os módulos num protótipo funcional. Os resultados demonstraram a capacidade do sistema em seguir rotas definidas por waypoints *Global Positioning System* (GPS), armazenar dados de telemetria e operar em modo manual ou automático.

Conclui-se que o sistema desenvolvido constitui uma base sólida para futuras evoluções em *Unmanned Surface Vehicle* (USV), nomeadamente a aplicação de controladores avançados (*Proportional-Integral-Derivative* (PID) ou *Linear Quadratic Regulator* (LQR)), a integração em plataformas marítimas de maior escala, e a utilização em cenários reais de monitorização ambiental, investigação científica e operações costeiras.

## Palavras chave

Palavras-chave: USV; Sistemas Ciberfísicos; LoRa; ESP32; Navegação Autónoma.



# Abstract

This master's dissertation presents the development of a cyber-physical system for the autonomous control of Unmanned Surface Vehicles (*Unmanned Surface Vehicle (USV)*). The system was designed to be modular, extensible, and oriented towards the integration of multiple motors, sensors, and communication interfaces, ensuring robustness and adaptability to different operational scenarios.

The proposed architecture is based on an ESP32 microcontroller with native support for I2C, UART, and LoRa, responsible for controlling various modules, including sensors and actuators. The main modules include: i) brushless thrusters controlled by *Electronic Speed Controller (ESC)*, capable of providing bidirectional thrust; ii) an *Inertial Measurement Unit (IMU)*, supplying yaw, pitch, and roll measurements to guarantee stability and trajectory correction; iii) a *Global Positioning System (GPS)* receiver, enabling accurate navigation through geospatial coordinates; and iv) a LoRa-based communication module, providing long-range telemetry and real-time route updates with low energy consumption.

From the *software* perspective, the solution was structured into independent libraries that encapsulate propulsion, sensing, communication, and control functionalities. This modular approach fosters code reusability, simplifies maintenance, and ensures system scalability. Communication efficiency was enhanced by employing *Protocol Buffers (Protobuf)*, which significantly reduced the ToA of LoRa messages, leading to higher energy efficiency, lower collision probability, and increased effective range.

Experimental validation included incremental *hardware* and *software* testing, ranging from isolated thruster control to the full integration of all modules into a functional prototype. The results demonstrated the system's ability to follow routes defined by *Global Positioning System (GPS)* waypoints, store telemetry data, and operate in both manual and autonomous modes.

It is concluded that the developed system provides a solid foundation for future advancements in USV, particularly in the application of advanced controllers (*Proportional-Integral-Derivative (PID)* or *Linear Quadratic Regulator (LQR)*), integration into larger-scale maritime platforms, and deployment in real-world scenarios such as environmental monitoring, scientific research, and coastal operations.

## Key words

USV; Cyber-Physical Systems; LoRa; ESP32; Autonomous Navigation.



# Índice de Conteúdos

<b>Agradecimentos</b>	iii
<b>Declaração de integridade</b>	v
<b>Resumo</b>	vii
<b>Índice de Conteúdos</b>	xii
<b>Índice de Figuras</b>	xiv
<b>Índice de Tabelas</b>	xv
<b>Índice de Listagens</b>	xvii
<b>Índice de Acrónimos</b>	xx
<b>1 Introdução</b>	1
1.1 Motivação . . . . .	1
1.2 Contextualização . . . . .	2
1.3 Organização do Documento . . . . .	2
<b>2 Estado da Arte</b>	5
2.1 Trabalho Relacionado . . . . .	5
2.2 Background . . . . .	8
<b>3 Modelo Proposto</b>	11
3.1 Requisitos . . . . .	11
3.2 Abordagem . . . . .	12
<b>4 Arquitetura do Sistema</b>	15
4.1 Estrutura . . . . .	15
4.2 Motores e Alimentação . . . . .	16
4.3 <i>Electronic Speed Controller</i> (ESC) . . . . .	18
4.4 Sensores . . . . .	19
4.4.1 <i>Inertial Measurement Unit</i> (IMU) . . . . .	20
4.4.2 <i>Global Positioning System</i> (GPS) . . . . .	23

4.5	Interfaces de Comunicação . . . . .	24
4.5.1	<i>Long Range (LoRa)</i> . . . . .	25
4.5.2	<i>Inter-Integrated Circuit (I2C)</i> . . . . .	26
4.5.3	<i>Universal Asynchronous Receiver/Transmitter (UART)</i> . . . . .	27
4.6	Sumário . . . . .	28
<b>5</b>	<b>Implementação</b>	<b>29</b>
5.1	Plataforma de Desenvolvimento . . . . .	29
5.1.1	Microcontrolador <i>Espressif32 (ESP32)</i> . . . . .	29
5.1.2	Ambiente de Programação . . . . .	31
5.2	Implementação de <i>Hardware</i> . . . . .	32
5.2.1	Integração dos Propulsores . . . . .	33
5.2.2	Sensores . . . . .	33
5.2.3	Comunicação . . . . .	36
5.2.4	<i>Printed Circuit Board (PCB)</i> . . . . .	36
5.3	Implementação de <i>Software</i> . . . . .	38
5.3.1	Integração do Sistema . . . . .	38
5.3.2	Arquitetura do Código . . . . .	40
5.3.3	Controlo dos Motores . . . . .	45
5.3.4	Leitura e Processamento dos Sensores . . . . .	47
5.3.5	Comunicação LoRa . . . . .	47
5.4	Sumário . . . . .	48
<b>6</b>	<b>Validação e Testes</b>	<b>51</b>
6.1	Testes Incrementais de <i>Hardware</i> . . . . .	51
6.2	Integração e Testes de Sistema . . . . .	52
6.3	Validação em Ambiente Real . . . . .	52
6.4	Sumário . . . . .	53
<b>7</b>	<b>Conclusões e Trabalho Futuro</b>	<b>55</b>
<b>A</b>	<b>Esquemáticas PCB</b>	<b>57</b>
<b>B</b>	<b>Diagramas de classes</b>	<b>61</b>
	<b>Bibliografia</b>	<b>67</b>

# Índice de Figuras

2.1	USV Liquid Robotics Wave Glider . . . . .	6
2.2	USV Saildrone . . . . .	6
2.3	SeaRobotics SR-Surveyor . . . . .	6
2.4	ASV Global C-Cat 3 . . . . .	7
2.5	USV do projeto Sea2Future . . . . .	7
2.6	Arquitetura do Robô Didático . . . . .	8
2.7	Estrutura final do Robô Didático . . . . .	9
3.1	Diagrama da arquitetura modular do USV . . . . .	12
4.1	Estrutura do USV . . . . .	15
4.2	Propulsores <i>brushless</i> utilizados no USV . . . . .	16
4.3	Propulsores Endura C2 . . . . .	17
4.4	Integração elétrica dos propulsores . . . . .	18
4.5	Exemplo de sinal PWM para o controlo de velocidade e direção num ESC . . . . .	19
4.6	Representação do <i>yaw</i> , <i>pitch</i> e <i>roll</i> numa embarcação . . . . .	20
4.7	Representação do <i>yaw</i> numa embarcação . . . . .	21
4.8	Representação do <i>pitch</i> numa embarcação . . . . .	22
4.9	Representação do <i>roll</i> numa embarcação . . . . .	22
4.10	Triangulação do sinal GPS . . . . .	23
5.1	ESP32 TTGO LoRa32 . . . . .	30
5.2	IMU ICM-20948 utilizada no protótipo . . . . .	34
5.3	Módulo GPS BN-880 utilizado no protótipo e respetivo <i>pinout</i> . . . . .	35
5.4	Comparação entre a esquemática (A) e a implementação física (B) da PCB . . . . .	37
5.5	Versão montada da PCB desenvolvida para o protótipo do USV . . . . .	37
5.6	Arquitetura simplificada do sistema USV . . . . .	41
6.1	Ferramenta de visualização de <i>waypoints</i> . . . . .	53
6.2	Demonstração do USV em funcionamento automático e manual (clique na imagem para abrir o vídeo no YouTube). . . . .	53
A.1	Esquemática do microcontrolador . . . . .	57
A.2	Esquemática dos propulsores . . . . .	58

A.3	Esquemática do IMU . . . . .	58
A.4	Esquemática do controlo dos motores (expansor) . . . . .	59
A.5	Esquemática do GPS . . . . .	59
B.1	Diagrama de classes - Versão simplificada . . . . .	61
B.2	Diagrama de classes - Versão intermédia I . . . . .	62
B.3	Diagrama de classes - Versão intermédia II . . . . .	63
B.4	Diagrama de classes - Versão completa . . . . .	64

# **Índice de Tabelas**

4.1 Comparação entre Tecnologias de Comunicação . . . . .	26
---	----



# Índice de Listagens

5.1	Exemplo de ficheiro <code>platformio.ini</code> utilizado no projeto . . . . .	31
5.2	Estrutura principal do programa do USV . . . . .	39
5.3	Comparação entre representação textual e binária com <i>Protobuf</i> . . . . .	42
5.4	Exceto da classe <code>Control</code> demonstrando a lógica automática de controlo . . . . .	43
5.5	Parte da classe <code>Thruster</code> . . . . .	46
5.6	Método <code>setDutyCycle()</code> da classe <code>Expander</code> . . . . .	46



# Índice de Acrónimos

**ASV** *Autonomous Surface Vehicle.* xiii, 7

**DC** *Direct Current.* 9, 16, 17

**ENIDH** Escola Superior Náutica Infante D. Henrique. iii, 2

**ESC** *Electronic Speed Controller.* vii, ix, xi, xiii, 9, 15–19, 28, 33, 36, 40, 45, 46, 51

**ESP-IDF** *Espressif IoT Development Framework.* 30, 32

**ESP32** *Espressif32.* vii, ix, xii, xiii, 29–32, 34–36

**GPS** *Global Positioning System.* vii, ix, xi, xiii, xiv, 1, 7, 9, 12, 20, 23, 24, 27, 28, 33, 35, 36, 39–41, 47, 51, 52, 55, 59

**HTML** *Hyper Text Markup Language.* 52

**I2C** *Inter-Integrated Circuit.* vii, ix, xii, 9, 11, 24, 26–28, 30, 32, 34–37, 40, 47, 51

**IMU** *Inertial Measurement Unit.* vii, ix, xi, xiii, xiv, 1, 7, 9, 12, 20, 21, 24, 26, 28, 33–37, 39–41, 47, 51, 52, 55, 58

**ISEL** Instituto Superior de Engenharia de Lisboa. iii, 2

**LED** *Light-Emitting Diode.* 37, 43, 45

**LoRa** *Long Range.* vii, ix, xii, xiii, 1, 12, 24–26, 28–31, 36, 38, 40, 42, 43, 48, 51, 52, 55

**LQR** *Linear Quadratic Regulator.* vii, ix, 43, 45, 55

**Naval-REX25** *Naval-Robotics Exercise 2025.* iii, 2, 52

**NMEA** *National Marine Electronics Association.* 35, 40, 47, 51

**OLED** *Organic Light-Emitting Diode.* 30, 51

**PCB** *Printed Circuit Board.* xii, xiii, 29, 32, 36, 37, 48, 57

**PID** *Proportional-Integral-Derivative.* vii, ix, 43, 45, 55

**PWM** *Pulse Width Modulation*. xiii, 13, 16–19, 28, 30, 32, 33, 36, 40, 41, 45–47, 51

**REPMUS25** *Robotic Experimentation and Prototyping using Maritime Uncrewed Systems 2025*. iii, 2, 3, 52

**SPI** *Serial Peripheral Interface*. 26, 27, 30

**TFM** Tese Final de Mestrado. iii, 1, 2, 9, 11, 15, 16, 23, 24, 28, 29, 32, 38, 45, 52

**ToA** *Time on Air*. vii, ix, 42, 43, 48

**UART** *Universal Asynchronous Receiver/Transmitter*. vii, ix, xii, 24, 27, 28, 30, 35, 36, 40, 47, 51

**UML** *Unified Modeling Language*. 41

**USV** *Unmanned Surface Vehicle*. vii, ix, xiii, xvii, 1, 2, 5–7, 9, 11–13, 15–21, 23, 24, 26–28, 30, 31, 33–41, 51–53, 55, 56

**Wi-Fi** *Wireless Fidelity*. 30

**ZEE** Zona Económica Exclusiva. 2

# Capítulo 1

## Introdução

O presente trabalho tem como objetivo o desenvolvimento de um sistema ciberfísico para o controlo de veículos autónomos, concebido de forma genérica para abranger tanto plataformas terrestres como marítimas, sendo o caso de estudo desta Tese Final de Mestrado (TFM) a aplicação a um *Unmanned Surface Vehicle* (USV) com capacidade de navegação autónoma. O sistema a ser desenvolvido deverá ser capaz de controlar até quatro motores de forma independente, garantindo a manobrabilidade necessária para diferentes cenários de operação. Além disso, será integrado com um conjunto de sensores essenciais, incluindo: i) sensores de temperatura e humidade, para monitorização ambiental; ii) um *Inertial Measurement Unit* (IMU), responsável por medir e ajustar a orientação e aceleração do USV; e iii) um *Global Positioning System* (GPS), para navegação precisa através de coordenadas geoespaciais.

Para permitir a comunicação de longo alcance, o sistema incluirá uma interface *Long Range* (LoRa), que viabilizará o envio de dados de telemetria e a receção de novas rotas de navegação de forma eficiente, mesmo em áreas remotas.

O protótipo desenvolvido deverá ser capaz de seguir autonomamente uma rota constituída por coordenadas GPS previamente definidas. Durante a navegação, o sistema armazenará os dados de telemetria, incluindo a posição e o estado da embarcação, em um cartão de memória, permitindo não apenas a posterior avaliação do desempenho do veículo, mas também o estudo detalhado da trajetória efetivamente seguida em comparação com a rota planeada.

### 1.1 Motivação

A motivação para o desenvolvimento deste trabalho surge da necessidade crescente de soluções tecnológicas que permitam explorar, monitorizar e intervir em ambientes marítimos de forma autónoma, segura e eficiente. Os USV representam uma alternativa promissora a embarcações tripuladas em missões de risco ou em operações que requerem elevada precisão, contribuindo para áreas como a investigação científica, a monitorização ambiental, a defesa e a segurança marítima.

Para além desta motivação aplicada, o trabalho insere-se num percurso académico e científico que começou com o desenvolvimento de um robô didático [1], concebido para fins pedagógicos. Esse projeto permitiu consolidar conhecimentos fundamentais sobre controlo de movimento, integração

de sensores e comunicação sem fios. Com base nessa experiência, emergiu a necessidade de validar, em contexto real e com maior complexidade, se os princípios teóricos e arquiteturais aplicados em ambientes laboratoriais são escaláveis e eficazes em sistemas ciberfísicos de maior dimensão. Assim, esta TFM não só visa propor uma solução tecnológica inovadora, como também demonstrar, através de validação experimental, que os conceitos previamente adquiridos podem ser aplicados com sucesso a um USV.

## 1.2 Contextualização

O desenvolvimento de veículos marítimos autónomos tem registado um crescimento acentuado a nível mundial, impulsionado pela necessidade de monitorização de ecossistemas marinhos, exploração de zonas remotas e apoio a operações de defesa. Em particular, Portugal, com a sua extensa Zona Económica Exclusiva (ZEE), apresenta um contexto altamente relevante para a adoção de tecnologias de robótica marítima.

Os USV baseados em propulsão elétrica, equipados com sensores e módulos de comunicação de longo alcance, têm-se afirmado como plataformas versáteis para missões de curta e média duração. No entanto, o elevado custo associado às soluções comerciais limita a sua utilização em contextos académicos e de investigação aplicada.

Neste enquadramento, o presente trabalho resulta da colaboração entre o Instituto Superior de Engenharia de Lisboa (ISEL) e a Escola Superior Náutica Infante D. Henrique (ENIDH), instituições que partilham o objetivo de promover soluções inovadoras e acessíveis no domínio da robótica marítima. Esta parceria não só viabilizou o desenvolvimento de um protótipo de baixo custo, modular e escalável, como também criou condições privilegiadas para a realização de ensaios práticos em ambiente real. Foi nesse âmbito que o sistema pôde ser testado no *Robotic Experimentation and Prototyping using Maritime Uncrewed Systems 2025* (REPMUS25), mais concretamente no subevento *Naval-Robotics Exercise 2025* (Naval-REX25), da Marinha Portuguesa e Escola Naval, permitindo validar conceitos avançados de navegação autónoma e comunicação sem fios em condições operacionais próximas das que se encontram em missões reais.

Todo o código desenvolvido nesta TFM encontra-se em um repositório remoto público (*open source*) no GitHub [2]. Este processo assegurou a rastreabilidade das alterações, a organização das diferentes versões do código e a possibilidade de colaboração futura.

## 1.3 Organização do Documento

Este documento encontra-se estruturado em sete capítulos principais. O Capítulo 1 apresenta a introdução ao trabalho, incluindo a motivação, a contextualização e a organização da TFM. O Capítulo 2 descreve o estado da arte, explorando soluções existentes e antecedentes relevantes, incluindo o projeto do robô didático que serviu de base conceptual. O Capítulo 3 define o modelo proposto, os requisitos do sistema e a abordagem metodológica adotada. O Capítulo 4 detalha a arquitetura do sistema, abrangendo os módulos de *hardware* e *software* necessários para a operação do USV. O Capítulo 5 aborda a implementação prática, descrevendo o processo de integração e as

soluções desenvolvidas. O Capítulo 6 apresenta a validação e os testes realizados, desde ensaios incrementais até à validação em ambiente real no exercício internacional REPMUS25. Por fim, o Capítulo 7 apresenta as conclusões obtidas e as perspetivas para trabalho futuro, destacando potenciais evoluções e aplicações do sistema.



# **Capítulo 2**

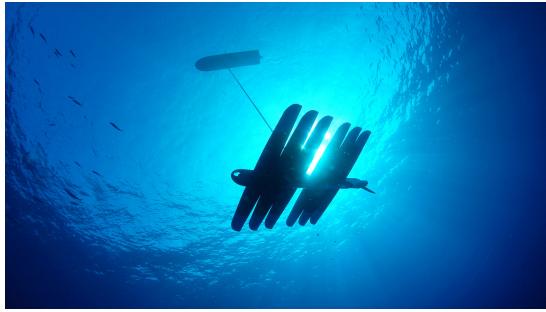
## **Estado da Arte**

O estado da arte apresentado neste capítulo tem como objetivo contextualizar o desenvolvimento de USV no panorama atual da robótica marítima. São discutidas as principais soluções de propulsão, desde abordagens baseadas em energias renováveis até plataformas motorizadas de alto desempenho, destacando as respetivas vantagens, limitações e áreas de aplicação. Adicionalmente, é feita uma ponte com o trabalho realizado durante a licenciatura, o qual serviu de base conceptual e experimental para a presente dissertação. Esta análise fornece o enquadramento necessário para compreender a evolução tecnológica nesta área e justifica as opções adotadas no desenvolvimento do protótipo proposto.

### **2.1 Trabalho Relacionado**

O desenvolvimento de USV tem atraído crescente atenção devido à sua relevância em operações de monitorização, recolha de dados e exploração de ambientes marítimos. Estas plataformas permitem realizar missões de forma segura e eficiente em áreas de difícil acesso ou potencialmente perigosas para embarcações tripuladas, constituindo uma solução cada vez mais valorizada em contextos científicos, ambientais e de defesa.

Entre as alternativas de propulsão destacam-se soluções que dispensam motores convencionais. O *Wave Glider*, desenvolvido pela Liquid Robotics, utiliza a energia das ondas para se deslocar de forma contínua e passiva, sendo particularmente adequado para missões de longa duração com baixo consumo energético. A Figura 2.1 ilustra o princípio de funcionamento do mecanismo de propulsão por ondas (parte submersa) e a estrutura de superfície responsável por garantir flutuabilidade e suporte para a instalação de sensores e antenas.



(A) Mecanismo de propulsão por ondas



(B) Estrutura da superfície

**Figura 2.1** USV Liquid Robotics Wave Glider [3]

Outra abordagem alternativa é o *Saildrone* [4], representado na Figura 2.2, que recorre à propulsão à vela. Este tipo de USV tem sido amplamente utilizado em campanhas de monitorização oceânica de larga escala, tirando partido da elevada autonomia oferecida pela energia eólica.



**Figura 2.2** USV Saildrone [4]

Apesar das vantagens associadas à utilização de fontes renováveis, as USV motorizadas oferecem maior controlo de navegação, velocidade e manobrabilidade, características essenciais em ambientes de correntes variáveis ou condições marítimas adversas. Um exemplo relevante é o SeaRobotics SR-Surveyor [5], amplamente utilizado em levantamentos hidrográficos e mapeamento subaquático, equipado com sonar multifeixe para operações precisas em ambientes costeiros e portuários.



**Figura 2.3** SeaRobotics SR-Surveyor [6]

A Figura mostra outro exemplo, o *Autonomous Surface Vehicle* (ASV) (subclasse de USV) Global's C-Cat 3 [7], é um USV compacto movido a motor e projetado para missões em águas pouco profundas, aplicável em tarefas como monitorização ambiental e batimetria.



**Figura 2.4** ASV Global C-Cat 3 [7]

Também em Portugal têm surgido iniciativas relevantes, como o projeto Sea2Future [8, 9], desenvolvido pela Escola Superior Náutica Infante D. Henrique, que, conforme ilustrado na Figura 2.5, explora o potencial dos USV motorizados em cenários de resgate e recolha de dados ambientais.



**Figura 2.5** USV do projeto Sea2Future [8, 9]

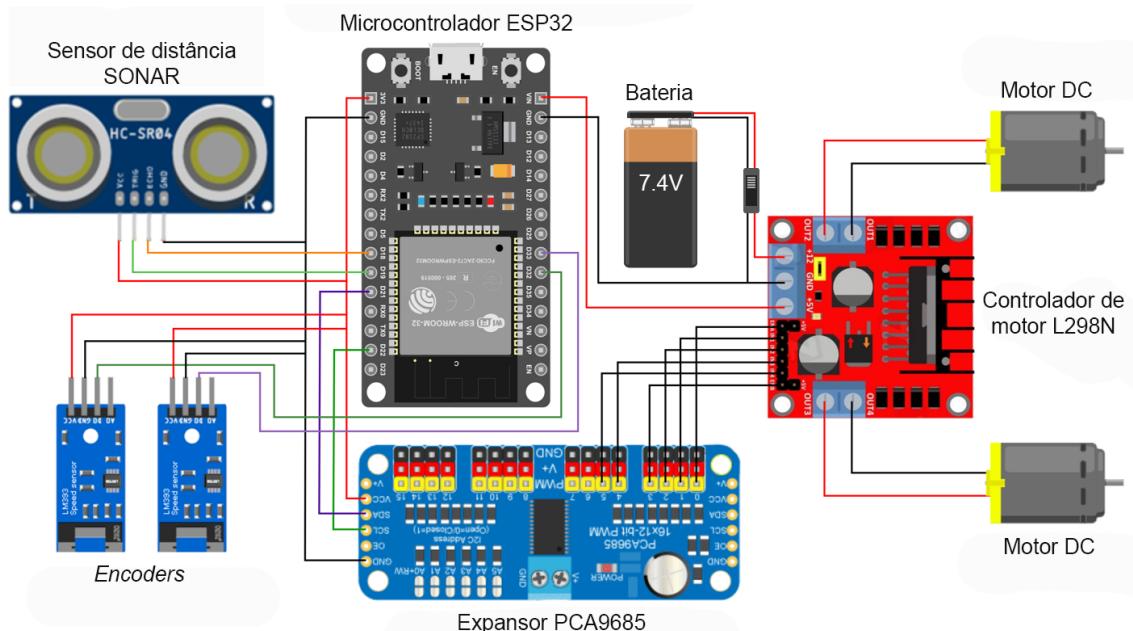
Embora tecnologicamente avançadas, estas soluções comerciais apresentam custos elevados, frequentemente na ordem de centenas de milhares de euros. A aquisição e manutenção de plataformas como o SR-Surveyor ou o C-Cat 3 podem atingir valores próximos de meio milhão de euros, dependendo do conjunto de sensores e da robustez estrutural requerida para operar em ambientes marítimos adversos.

Nos últimos anos, os avanços tecnológicos em sistemas de controlo e sensorização têm impulsionado o desenvolvimento de USV mais acessíveis e modulares. Destaca-se o controlo independente de múltiplos motores, aliado à integração de sensores como GPS, IMU e sensores ambientais, que tornam estas plataformas cada vez mais autossuficientes e precisas, abrindo caminho para aplicações em larga escala em monitorização costeira, exploração científica e operações de segurança.

## 2.2 Background

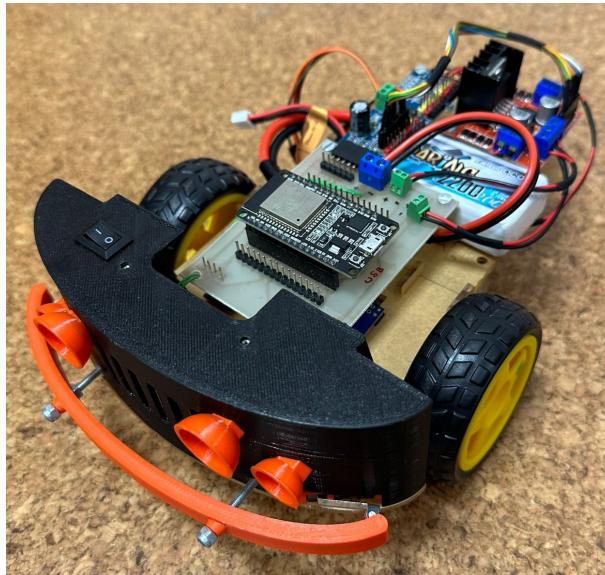
Durante o percurso académico da licenciatura, foi desenvolvido um projeto descrito em [1], cujo objetivo principal consistiu na criação de um ambiente didático orientado para a exploração de conceitos fundamentais em robótica, como o controlo de movimento, a comunicação sem fios e a programação em Java. O resultado deste trabalho foi um protótipo funcional de robô didático, concebido como uma ferramenta pedagógica eficaz.

A arquitetura do sistema, apresentada em [1] e ilustrada na Figura 2.6, foi concebida segundo uma abordagem modular e expansível. Esta estrutura permitia a integração de até quatro motores (localizados na parte direita da figura), incluía os módulos de comunicação (representados no centro) e os módulos de sensorização (à esquerda), possibilitando a implementação e validação de diferentes configurações de forma flexível e adaptável.



**Figura 2.6** Arquitetura do Robô Didático [1]

O projeto culminou na construção de um protótipo físico, ilustrado na Figura 2.7, que materializou a arquitetura proposta e demonstrou em ambiente real a integração de motores, sensores e módulos de comunicação. Esta validação experimental confirmou a viabilidade da abordagem modular, destacando a facilidade de substituição ou expansão de componentes.



**Figura 2.7** Estrutura final do Robô Didático [1]

Um dos elementos centrais desta arquitetura era o controlador de motores, que assegurava a ligação entre a bateria principal e os diferentes atuadores. Este módulo não só suportava uma ampla gama de tensões de entrada, adequada para motores de diferentes características, como também fornecia uma saída regulada para o microcontrolador, simplificando a gestão energética do sistema. A presença de um interruptor dedicado entre a bateria e o controlador permitia desligar rapidamente todo o sistema em situações de teste ou emergência, reforçando a segurança operacional.

A presente TFM surge como uma evolução natural deste projeto, aplicando e expandindo os conceitos adquiridos para o desenvolvimento de um USV com capacidade de navegação autónoma. Embora a lógica base da arquitetura tenha sido mantida, nomeadamente a utilização do barramento *Inter-Integrated Circuit* (I2C) como interface principal de comunicação, foram realizadas adaptações relevantes. Os motores *Direct Current* (DC) do robô didático foram substituídos por propulsores *brushless*, controlados por *Electronic Speed Controller* (ESC), mais adequados ao ambiente marítimo.

Adicionalmente, foram removidos sensores como o sonar e os encoders, que no contexto terrestre permitiam estimar a distância percorrida a partir do número de rotações da roda. No caso de um USV, esta abordagem revela-se inviável, pois fatores externos como correntes, vento ou turbulência influenciam a relação entre rotações do propulsor e distância efetivamente percorrida. Para superar esta limitação, os encoders foram substituídos por um IMU e por um GPS, que fornecem medições absolutas de aceleração e orientação, permitindo estimar com maior precisão o movimento do veículo em ambiente marítimo.

Assim, o trabalho desenvolvido na licenciatura serviu de alicerce conceptual e experimental para a presente TFM, onde a arquitetura foi adaptada a um contexto mais exigente, integrando sensores e atuadores adequados às particularidades da navegação marítima.



# **Capítulo 3**

## **Modelo Proposto**

O modelo proposto nesta TFM consiste no desenvolvimento de um sistema ciberfísico para o controlo autónomo de embarcações do tipo USV, concebido de forma modular e expansível. A modularidade é garantida pela utilização do barramento I2C de comunicação entre os diferentes módulos de *hardware*. A escolha do I2C como base do sistema justifica-se pela sua simplicidade de implementação, baixo número de ligações necessárias e pela possibilidade de integrar múltiplos dispositivos no mesmo barramento através da atribuição de endereços únicos. Esta abordagem garante escalabilidade, permitindo a integração de novos sensores, atuadores, ou até mesmo diferentes microcontroladores, desde que estes suportem I2C, sem alterações profundas na arquitetura.

De forma a apresentar esta proposta de forma estruturada, este capítulo encontra-se organizado em duas secções principais. Na Secção 3.1 são descritos os requisitos funcionais e não funcionais que orientam o desenvolvimento do sistema, identificando as funcionalidades essenciais e as propriedades globais de desempenho e escalabilidade. Já na Secção 3.2 é detalhada a abordagem modular seguida, explicitando os principais módulos de *hardware* e *software* que compõem o sistema, bem como as suas interações. Esta estrutura permite compreender de forma clara tanto as bases conceptuais como a implementação prática do modelo proposto.

### **3.1 Requisitos**

Os requisitos funcionais correspondem a descrições formais e detalhadas das funcionalidades que o sistema deve disponibilizar, especificando as ações que este deve ser capaz de executar e a forma como deve responder às diferentes interações do utilizador ou de outros sistemas.

Por sua vez, os requisitos não funcionais referem-se a propriedades globais do sistema que não estão diretamente associadas a uma funcionalidade específica, mas que condicionam o seu comportamento e qualidade de operação. Entre estes incluem-se aspetos como o desempenho, a segurança, a usabilidade, a escalabilidade e a fiabilidade, que são determinantes para garantir a robustez e adequação da solução em cenários reais de utilização.

Os requisitos principais definidos para o sistema são:

1. O sistema deve ser expansível através do barramento I2C, assegurando a fácil integração de novos módulos. Requisito não funcional, uma vez que não descreve uma funcionalidade direta

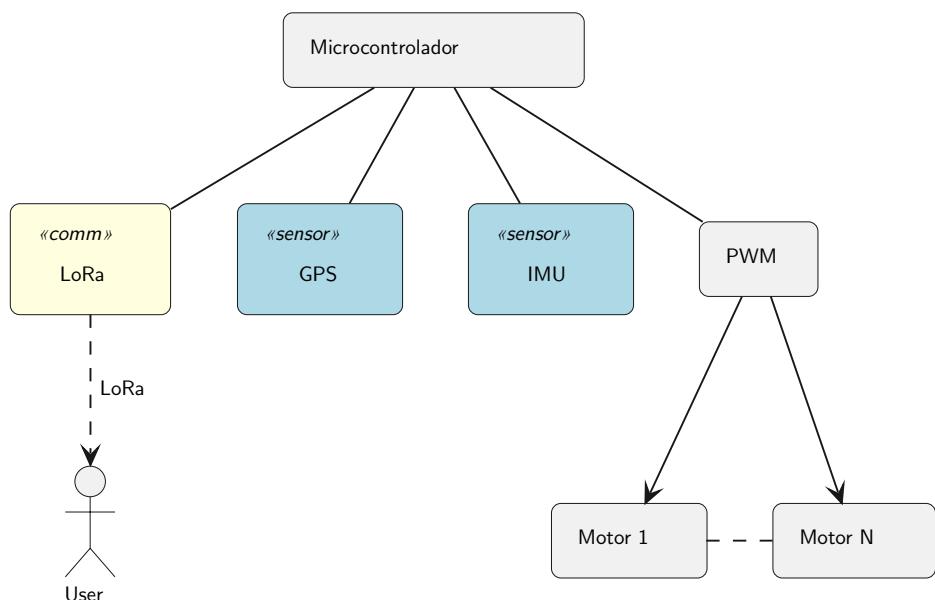
do sistema, mas sim uma característica de extensibilidade e escalabilidade, que garante a adaptabilidade futura da arquitetura.

- O sistema deve utilizar comunicação de longo alcance baseada em tecnologia LoRa, de forma a permitir telemetria e atualização de rotas em tempo real, mesmo em cenários remotos. Requisito funcional, dado que descreve explicitamente uma ação que o sistema deve executar, isto é, a capacidade de estabelecer comunicação sem fios de longo alcance para troca de dados críticos com a estação de controlo.

Desta forma, observa-se que o primeiro requisito reflete uma propriedade estrutural do sistema, relacionada com a sua capacidade de evolução e integração modular, enquanto o segundo requisito define uma funcionalidade central para a operação do veículo, indispensável para o cumprimento dos objetivos de navegação autónoma e monitorização remota.

## 3.2 Abordagem

Para assegurar a operação autónoma, o sistema integra diferentes módulos, tal como apresentado na Figura 3.2.



**Figura 3.1** Diagrama da arquitetura modular do USV

O *Global Positioning System* (GPS) é responsável pela determinação da posição geográfica da embarcação, fornecendo coordenadas de latitude e longitude necessárias para o seguimento de rotas. O *Inertial Measurement Unit* (IMU), por sua vez, mede aceleração, rotação e orientação da embarcação nos eixos tridimensionais, permitindo a correção de trajetória e a compensação de efeitos de correntes e ondas. A interface de comunicação *Long Range* (LoRa) é utilizada para o envio de dados de telemetria, a receção de novas rotas e a indicação do estado de autonomia (manual ou automático). No modo manual, possibilita ainda a transmissão de comandos de movimento (frente,

trás, esquerda e direita), garantindo uma comunicação fiável em cenários de operação prolongada e em zonas sem infraestrutura de telecomunicações.

Adicionalmente, foi integrado um módulo de armazenamento em cartão SD, cuja função é registar localmente os dados de telemetria recolhidos durante a operação. Este armazenamento persistente assegura que a informação relativa à trajetória, parâmetros de navegação e estados do sistema possa ser posteriormente analisada, mesmo em situações em que a comunicação de longo alcance esteja temporariamente indisponível. Esta funcionalidade é essencial para a validação experimental, análise pós-missão e suporte a algoritmos de diagnóstico e melhoria contínua.

O sistema deverá ainda ser capaz de controlar até quatro motores de forma independente, recorrendo a sinais *Pulse Width Modulation* (PWM) gerados pelo microcontrolador. O controlo distribuído de motores oferece flexibilidade na navegação e aumenta a manobrabilidade do USV.

A abordagem modular proposta assegura que a arquitetura não se limita aos sensores e interfaces inicialmente integrados, permitindo a evolução futura do sistema com a adição de novos módulos, como sensores de ambiente, câmaras ou interfaces de comunicação alternativas. Desta forma, o modelo apresentado constitui uma solução escalável, robusta e alinhada com os requisitos de flexibilidade e adaptabilidade para missões em ambientes marítimos complexos.



# Capítulo 4

## Arquitetura do Sistema

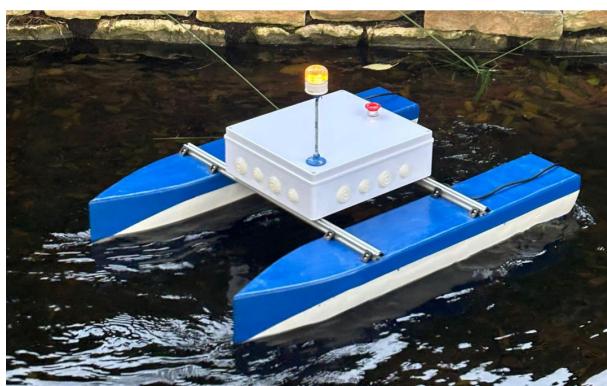
A arquitetura desenvolvida para o USV proposto nesta TFM apresenta diversas semelhanças com a arquitetura do Robô Didático, descrita em [1], no Capítulo 4. A experiência obtida nesse trabalho serviu como base conceptual e prática para a definição da presente solução, permitindo a reutilização de princípios estruturais e de integração de módulos.

Este capítulo encontra-se organizado em quatro secções principais. Na Secção 4.1 é apresentada a estrutura física do protótipo, destacando a configuração adotada e as opções construtivas. A Secção 4.2 descreve os motores e o sistema de alimentação que asseguram a propulsão do veículo. Na Secção 4.3 analisa-se o papel do ESC no controlo da velocidade e direção dos propulsores. Por fim, a Secção 4.4 aborda a integração dos principais sensores do sistema, culminando na apresentação das interfaces de comunicação que asseguram a ligação entre os diferentes módulos.

### 4.1 Estrutura

A definição da estrutura física constituiu o primeiro passo para a construção do protótipo do USV. A seleção do *chassi* foi orientada por três objetivos principais: garantir a flutuabilidade, sustentar todos os módulos necessários e assegurar a robustez da embarcação em cenários de teste.

Optou-se por uma configuração em catamarã, composta por dois flutuadores fabricados em impressão 3D, ilustrado na Figura 4.1.



**Figura 4.1** Estrutura do USV [10]

Para este processo foi utilizada uma impressora Ender 3 de primeira geração, modificada com

a adição de um eixo vertical de 1,5 metros, de forma a possibilitar a produção de peças de maior dimensão. O material escolhido foi PLA, posteriormente reforçado com fibra de vidro, garantindo maior resistência mecânica e durabilidade em ambiente marítimo.

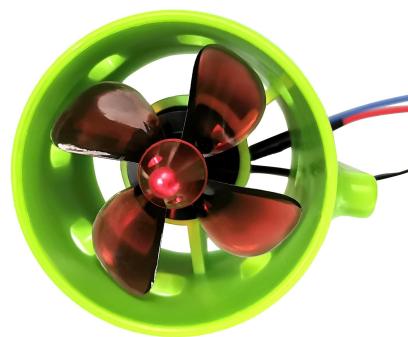
No centro da estrutura foi instalada uma caixa estanque, destinada a albergar a eletrónica responsável pelo controlo do USV, incluindo sensores, interfaces de comunicação e unidades de processamento. Esta caixa integra todos os elementos eletrónicos necessários ao funcionamento do sistema.

A escolha da estrutura apresentada na Figura 4.1 revela vantagens logísticas significativas. O USV-enautica1, embora robusto, apresenta dimensões e peso consideráveis, o que dificulta o seu transporte e limita a sua utilização em testes frequentes. Em contraste, a adoção de um catamarã mais pequeno e leve permite a realização de ensaios de navegação e validação de algoritmos de forma prática e acessível, assegurando maior portabilidade e facilidade de operação.

É importante salientar que o USV-enautica1 utiliza o mesmo tipo de motores PWM integrados no protótipo desenvolvido nesta TFM, garantindo assim compatibilidade total com os procedimentos de controlo e validação aqui descritos. Esta correspondência tecnológica assegura que os resultados obtidos no protótipo são representativos e transferíveis para sistemas de maior escala.

## 4.2 Motores e Alimentação

Tal como o robô didático descrito em [1], o USV desenvolvido neste trabalho recorre a motores para a sua propulsão. No entanto, em vez de motores DC utilizados em protótipos educacionais, optou-se pela integração de propulsores subaquáticos de maior robustez, adequados ao ambiente de operação marítima. Estes propulsores, demonstrados na Figura 4.2, também conhecidos como *thrusters*, são motores elétricos *brushless* acoplados a hélices, cuja função é gerar empuxo ao movimentar o fluido envolvente (água), permitindo o deslocamento da embarcação.



**Figura 4.2** Propulsores *brushless* utilizados no USV

Para este projeto foram selecionados os propulsores U01 [11], concebidos para aplicações em veículos de superfície e submersíveis de pequena escala. Estes motores operam numa faixa de tensão de 12 a 16 V, com uma potência nominal de 200 W, proporcionando até 2 kgf de empuxo em ambos os sentidos de rotação, uma vez que o seu controlador eletrónico (ESC) permite controlo bidirecional. Esta característica é essencial para o USV, uma vez que garante a manobrabilidade necessária tanto para navegação em linha reta como para rotações em torno do próprio eixo permitindo mudar a

direção da embarcação.

Para aplicações que exijam maior potência, podem ser utilizados motores mais robustos, como o Endura C2 [12] presentes no USV-enautica1 da Sea2Future [8] [9] e ilustrados tanto na Figura 2.5 como na Figura 4.3.



**Figura 4.3** Propulsores Endura C2 [12]

Os Endura C2 requerem 30A e são também alimentados externamente por uma bateria de 12V.

Em termos de integração elétrica, como ilustrado na Figura 4.4, cada motor é alimentado por uma bateria de 12 V de elevada capacidade, de forma a suportar a potência exigida e garantir autonomia de operação durante missões prolongadas. A escolha de propulsores *brushless* deve-se a várias vantagens face aos motores DC convencionais: maior eficiência energética, menor desgaste mecânico devido à ausência de escovas, capacidade de fornecer binário elevado a baixas rotações e maior fiabilidade em condições ambientais adversas, como a exposição contínua à água.

Adicionalmente, o sistema de controlo utiliza sinais de PWM gerados pelo microcontrolador principal, que são interpretados pelo ESC de cada motor. Este mecanismo permite ajustar de forma contínua a velocidade e o sentido de rotação, possibilitando manobras precisas, tais como mudanças bruscas de direção ou movimentos de baixa velocidade em ambientes restritos.

A adoção de dois propulsores oferece um equilíbrio entre simplicidade e manobrabilidade, permitindo que o USV execute translações e rotações sem necessidade de sistemas de direção adicionais. Contudo, a arquitetura do sistema foi concebida para ser escalável, suportando até quatro propulsores, caso seja necessário aumentar a estabilidade, a potência de propulsão ou a capacidade de operação em cenários marítimos mais exigentes.



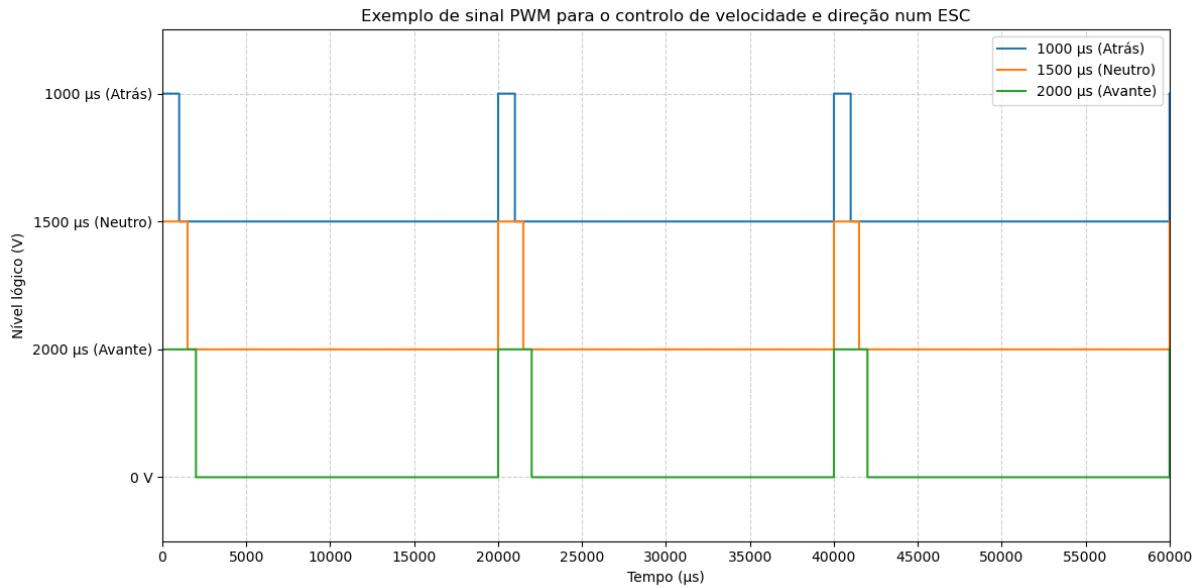
**Figura 4.4** Integração elétrica dos propulsores

### 4.3 Electronic Speed Controller (ESC)

O *Electronic Speed Controller* (ESC) é o componente responsável por controlar o funcionamento dos motores *brushless* utilizados no USV. Este dispositivo converte o sinal de comando proveniente do microcontrolador em impulsos elétricos adequados para alimentar as bobinas do motor, garantindo a regulação da sua velocidade e direção de rotação.

A Figura 4.5 apresenta a forma de onda do sinal de PWM aplicado ao ESC para controlar os *thrusters*. A Figura 4.3 apresenta a forma de onda do sinal de PWM descrito.

O princípio de funcionamento baseia-se na utilização de sinais de PWM enviados pelo microcontrolador. Estes sinais consistem em pulsos periódicos cuja largura (duração em microssegundos) determina a resposta do motor [1]. Tipicamente, um sinal com largura de  $1000\ \mu s$  (linha azul da Figura 4.5) corresponde à velocidade mínima no sentido inverso,  $1500\ \mu s$  (linha laranja da Figura 4.5) corresponde ao ponto neutro, no qual o motor permanece parado, e  $2000\ \mu s$  (linha verde da Figura 4.5) corresponde à velocidade máxima no sentido direto. A partir deste ponto central, a largura do pulso varia de forma simétrica: valores inferiores a  $1500\ \mu s$  comandam rotações no sentido inverso com velocidade crescente à medida que a largura do pulso diminui, enquanto valores superiores a  $1500\ \mu s$  comandam rotações no sentido direto com velocidade crescente à medida que a largura do pulso aumenta. A frequência habitual do sinal é de  $50\ Hz$ , o que significa que cada período tem  $20000\ \mu s$ . Este padrão segue a mesma lógica dos sinais de controlo usados em servomotores, o que facilita a sua implementação e utilização de controladores *standard* de radio controle utilizado em modelismo.



**Figura 4.5** Exemplo de sinal PWM para o controlo de velocidade e direção num ESC

Antes de os motores poderem ser utilizados, o ESC necessita de ser armado, este processo consiste na aplicação de uma sequência de sinais de inicialização que confirmam a ligação entre o controlador e o motor, garantindo que este não entra em funcionamento acidentalmente. No caso em estudo, este procedimento envolve enviar inicialmente um sinal de largura mínima (por exemplo,  $1000\ \mu s$ ) durante 2 segundos, seguido da transição para a largura correspondente ao ponto neutro ( $1500\ \mu s$ ). Apenas após esta sequência o ESC considera o sistema pronto a operar, emitindo normalmente sinais sonoros característicos que confirmam o estado de prontidão.

O controlo bidirecional da rotação é assegurado pelo próprio ESC, que interpreta os sinais de PWM acima e abaixo do ponto neutro. Assim:

- Pulsos inferiores a  $1500\ \mu s$  comandam a rotação no sentido inverso (atrás);
- Pulsos superiores a  $1500\ \mu s$  comandam a rotação no sentido direto (avante).

Este mecanismo elimina a necessidade de sistemas mecânicos de inversão de polaridade, uma vez que o ESC efetua digitalmente a comutação das fases do motor *brushless*. A grande vantagem desta abordagem é a elevada precisão no controlo da velocidade, associada a uma resposta rápida e eficiente, fatores críticos para a manobrabilidade do USV em cenários marítimos complexos.

Em síntese, o ESC desempenha um papel central no sistema de propulsão: traduz os comandos enviados pelo microcontrolador em potência controlada para os motores, garante a segurança durante a inicialização através do processo de armamento, e possibilita a rotação bidirecional através da modulação do sinal PWM.

## 4.4 Sensores

A operação autónoma de um USV depende fortemente da disponibilidade de sensores capazes de fornecer medições fiáveis sobre a sua posição, movimento e orientação. Estes dados constituem a base

para o controlo de navegação, permitindo não apenas o seguimento rigoroso de rotas previamente definidas, mas também a correção dinâmica de desvios causados por correntes, ondas ou vento.

No protótipo desenvolvido, a arquitetura de sensorização foi concebida de forma modular, possibilitando a integração e substituição de sensores de maneira independente, sem comprometer a funcionalidade global. Esta abordagem garante escalabilidade e flexibilidade, permitindo adaptar o sistema a diferentes cenários operacionais ou evoluir para missões com requisitos mais complexos.

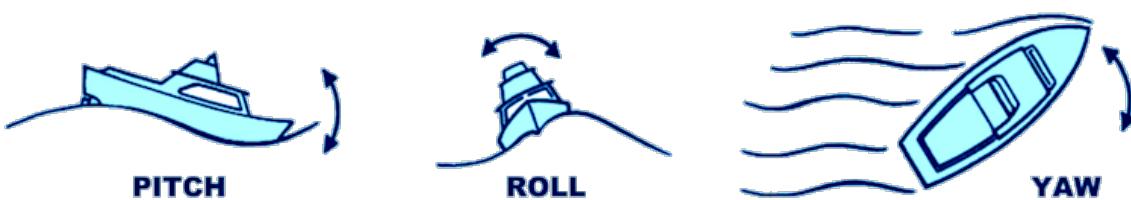
Entre os sensores integrados, destacam-se dois elementos centrais. O primeiro é a *Inertial Measurement Unit* (IMU), responsável por medir aceleração, velocidade angular e intensidade do campo magnético, permitindo calcular com elevada precisão a orientação do veículo nos três eixos principais (*yaw*, *pitch* e *roll*). O segundo é o recetor *Global Positioning System* (GPS), que fornece a posição geográfica absoluta em coordenadas de latitude e longitude, constituindo a referência essencial para a definição e seguimento de *waypoints*.

O uso combinado destes dois sensores permite mitigar as limitações individuais de cada um: enquanto o IMU fornece dados de orientação com elevada taxa de atualização, mas sofre de erro acumulado, o GPS disponibiliza medições absolutas de posição, embora com menor frequência e suscetíveis a erros momentâneos. Assim, a fusão de dados provenientes de ambos os sensores garante maior precisão, robustez e fiabilidade na estimativa da trajetória do USV.

As subseções seguintes apresentam em detalhe a integração da *Inertial Measurement Unit* (IMU) (Subsecção 4.4.1) e do recetor *Global Positioning System* (GPS) (Subsecção 4.4.2), descrevendo os princípios de funcionamento, a configuração adotada e o contributo de cada sensor para a arquitetura global de navegação.

#### 4.4.1 Inertial Measurement Unit (IMU)

O IMU é um dispositivo essencial para medir a orientação e o movimento do USV. Este sensor combina acelerômetros, giroscópios e, em alguns casos, magnetómetros, fornecendo dados sobre aceleração linear, taxas de rotação e direção magnética. Estas medições são processadas para determinar os ângulos de guinada (*yaw*), arfagem (*pitch*) e rotação (*roll*), que descrevem a orientação do veículo no espaço tridimensional, tal como ilustrado na Figura 4.6.



**Figura 4.6** Representação do *yaw*, *pitch* e *roll* numa embarcação [13]

O *yaw* representa a rotação em torno do eixo vertical e é essencial para controlar a direção do USV. O *pitch* mede a inclinação em torno do eixo horizontal, sendo crítico para avaliar o impacto de ondas e alterações na estabilidade longitudinal. Já o *roll* refere-se à oscilação lateral em torno do eixo longitudinal, desempenhando um papel importante na manutenção da estabilidade transversal

da embarcação.

Para este projeto, foram avaliadas diversas opções de IMU culminando em duas opções: o MPU6050 e o MPU9250. O MPU6050 é um sensor de 6 eixos que integra um acelerómetro e um giroscópio, permitindo medições básicas de orientação e movimento. Apesar da sua simplicidade e baixo custo, a ausência de um magnetômetro limita a sua capacidade de fornecer orientação absoluta, o que pode ser um fator crítico em qualquer aplicação que necessite dos conceitos anteriormente descritos. Por outro lado, o MPU9250 é um sensor de 9 eixos que, além do acelerómetro e do giroscópio, inclui um magnetômetro. Este último componente possibilita medições do campo magnético terrestre, fornecendo informações sobre a direção absoluta, algo indispensável para sistemas de navegação autónoma que dependem de dados precisos para evitar desvios ou erros acumulados.

A escolha do MPU9250 (ou equivalente) foi motivada pela sua superioridade técnica em comparação com o MPU6050. A inclusão de um magnetômetro permite compensar erros acumulados pelo giroscópio, garantindo medições mais estáveis e precisas em condições dinâmicas. Além disso, o suporte a técnicas de fusão de sensores (*sensor fusion*), que combinam dados do acelerómetro, giroscópio e magnetômetro, aumenta significativamente a fiabilidade das estimativas de orientação. Apesar de o MPU9250 ter um custo ligeiramente superior e exigir maior esforço de calibração, as suas vantagens, como a capacidade de medição absoluta de orientação, justificam plenamente a sua integração no sistema.

Com o MPU9250, o sistema é capaz de calcular e ajustar a orientação do USV em tempo real, permitindo a correção de rota em resposta a forças externas, como correntes marítimas e ondas. Para contextualizar melhor este processo, apresentam-se de seguida os conceitos fundamentais de *yaw*, *pitch* e *roll*, que descrevem a orientação da embarcação nos três eixos principais.

### **Yaw ( $\Psi$ )**

O ângulo de Yaw ( $\Psi$ ) representa a rotação em torno do eixo vertical (z) (Figura 4.7) e é calculado utilizando os dados do magnetômetro, corrigidos pela inclinação com base nos ângulos de *pitch* e *roll*.



**Figura 4.7** Representação do yaw numa embarcação (Adaptada de [13])

Primeiramente, calcula-se  $m'_x$  utilizando a Equação 4.1.

$$m'_x = m_x \cos(\Theta) + m_z \sin(\Theta) \quad (4.1)$$

Em seguida,  $m'_y$  é obtido com base na Equação 4.2.

$$m'_y = m_x \sin(\Phi) \sin(\Theta) + m_y \cos(\Phi) - m_z \sin(\Phi) \cos(\Theta) \quad (4.2)$$

Finalmente, o ângulo de *yaw* ( $\Psi$ ) é determinado através da Equação 4.3.

$$\Psi = \arctan 2(-m'_y, m'_x) \quad (4.3)$$

Aqui,  $m_x$ ,  $m_y$  e  $m_z$  são as medições do magnetómetro nos eixos  $x$ ,  $y$  e  $z$ , respetivamente, e os ângulos  $\Theta$  (pitch) e  $\Phi$  (roll) corrigem a inclinação do sensor.

### Pitch ( $\Theta$ )

O ângulo de *pitch* ( $\Theta$ ) representa a inclinação do sensor para frente e para trás (Figura 4.8).



**Figura 4.8** Representação do *pitch* numa embarcação (Adaptada de [13])

Sendo calculado diretamente com base nas medições do acelerómetro, conforme a Equação 4.4.

$$\Theta = \arcsin \left( \frac{-a_x}{\sqrt{a_x^2 + a_y^2 + a_z^2}} \right) \quad (4.4)$$

onde  $a_x$ ,  $a_y$  e  $a_z$  correspondem às acelerações medidas nos eixos  $x$ ,  $y$  e  $z$ , respetivamente.

### Roll ( $\Phi$ )

O ângulo de *roll* ( $\Phi$ ) representa a inclinação lateral (esquerda/direita) (Figura 4.9) e é derivado das medições do acelerómetro, como descrito na Equação 4.5.



**Figura 4.9** Representação do *roll* numa embarcação (Adaptada de [13])

$$\Phi = \arctan 2(a_y, a_z) \quad (4.5)$$

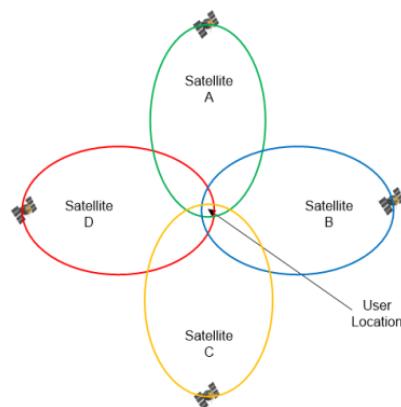
onde  $a_y$  e  $a_z$  são as acelerações medidas nos eixos  $y$  e  $z$ , respetivamente. A função  $\arctan 2$  é usada para garantir que o ângulo esteja no quadrante correto.

#### 4.4.2 Global Positioning System (GPS)

O GPS é uma tecnologia amplamente utilizada para a determinação da posição geográfica de objetos ou indivíduos em qualquer ponto do globo. Este sistema, desenvolvido inicialmente pelo Departamento de Defesa dos Estados Unidos, é constituído por uma constelação de satélites que transmitem sinais de rádio para dispositivos receptores na Terra. Estes sinais são processados para calcular coordenadas geográficas com elevada precisão, fornecendo informações cruciais em aplicações de navegação, monitorização e controlo de sistemas móveis, como veículos autónomos.

O funcionamento do GPS baseia-se no princípio da triangulação, tal como se pode observar na Figura 4.10), onde pelo menos quatro satélites são necessários para determinar com precisão a posição de um receptor no espaço tridimensional (latitude, longitude e altitude). Cada satélite transmite sinais codificados contendo informações sobre a sua posição e o tempo exato em que o sinal foi enviado. O receptor GPS calcula o tempo que o sinal levou a chegar a partir de múltiplos satélites, determinando assim a distância de cada um. Com base nestes dados, a posição do receptor é triangulada.

No caso do USV, apenas são necessários três satélites, pois não é necessário determinar a altitude. Como o USV opera na superfície da água, a sua posição é restrita a um plano bidimensional, considerando apenas latitude e longitude. Desta forma, a triangulação pode ser realizada com precisão utilizando apenas três satélites, simplificando o cálculo em relação a sistemas que requerem coordenadas tridimensionais.



**Figura 4.10** Triangulação do sinal GPS [14]

O sistema GPS depende de relógios atómicos precisos instalados nos satélites para garantir a sincronização temporal. Além disso, algoritmos avançados compensam erros causados por fatores como a atmosfera terrestre e o desvio orbital dos satélites.

Entre as principais vantagens do GPS, destacam-se a sua elevada precisão, que em condições ideais permite obter localizações com margens de erro de poucos metros, e a cobertura global, que garante funcionalidade em qualquer lugar do mundo, independentemente de fronteiras geográficas. Adicionalmente, o receptor GPS opera de forma passiva, recebendo sinais sem necessidade de os transmitir, o que contribui para a eficiência energética. A sua facilidade de integração em sistemas autónomos, como os USV descritos nesta TFM, é também uma característica relevante.

Por outro lado, o GPS apresenta algumas limitações, como a dependência de visibilidade direta

com os satélites, sendo o sinal frequentemente bloqueado por obstáculos como edifícios, montanhas, vegetação ou nuvens densas. As condições atmosféricas, incluindo tempestades solares e interferências atmosféricas, podem comprometer a precisão do sistema. Além disso, o consumo energético do receptor, embora moderado, pode ser um desafio em dispositivos com baterias de capacidade limitada. Por fim, o GPS é vulnerável a interferências e bloqueios intencionais (*jaming*), o que reduz a sua fiabilidade em situações críticas.

Apesar da possibilidade de utilizar apenas um IMU para estimar a posição e orientação do USV, esta abordagem revela-se insuficiente em aplicações que exigem elevada precisão em trajetos extensos. O IMU utiliza sensores como acelerômetros e giroscópios para calcular deslocações relativas, mas está sujeito a deriva acumulada ao longo do tempo devido a erros nos sensores e no processamento de dados. A combinação com um GPS é, portanto, fundamental, uma vez que este fornece medições absolutas que corrigem a deriva inerente ao IMU, assegurando uma navegação fiável e precisa.

Importa salientar que receptores GPS são tipicamente configurados e utilizados através de portas série (*Universal Asynchronous Receiver/Transmitter* (UART)), o que representa um desafio quando se pretende integrar múltiplos módulos num sistema que privilegia a comunicação via I2C. No contexto desta TFM, foi definido como requisito que todos os módulos comunicassem preferencialmente através do barramento I2C, de modo a simplificar a arquitetura e reduzir o número de interfaces independentes no microcontrolador. Para alcançar este objetivo, foi integrado um conversor I2C-UART (SC16IS750), que atua como uma ponte entre os dois protocolos. Esta solução permitiu expandir o número de interfaces série disponíveis sem comprometer a modularidade da arquitetura, garantindo a integração eficiente do GPS e mantendo a uniformidade da comunicação entre os diferentes módulos do sistema.

Concluída a descrição dos sensores principais, torna-se agora pertinente apresentar as interfaces de comunicação adotadas no sistema, que asseguram não apenas a ligação entre sensores e atuadores, mas também a transmissão de dados com a estação remota. Entre estas destacam-se a comunicação de longo alcance via LoRa, o barramento I2C como espinha dorsal do sistema modular, e a utilização pontual do UART para integração de dispositivos específicos.

## 4.5 Interfaces de Comunicação

Nesta secção são apresentadas as principais interfaces de comunicação utilizadas no desenvolvimento do sistema, fundamentais para assegurar a integração entre sensores, atuadores e a estação remota. A escolha das tecnologias de comunicação teve em consideração critérios de alcance, eficiência energética, simplicidade de implementação e escalabilidade da arquitetura. São analisadas três interfaces complementares: i) o LoRa, responsável pela comunicação de longo alcance e pela transmissão de telemetria em tempo real; ii) o I2C, que constitui o barramento principal para a integração modular de sensores e controladores; e iii) o UART, utilizado em casos específicos, como na ligação de receptores GPS, através de conversores dedicados.

#### 4.5.1 Long Range (LoRa)

No contexto de sistemas ciber-físicos e veículos autónomos, a seleção da interface de comunicação é um elemento essencial para garantir a troca eficiente e confiável de dados entre os diferentes módulos do sistema. As interfaces de comunicação desempenham um papel fundamental na transmissão de telemetria, atualizações de rota e integração de sensores em tempo real. Neste projeto, foi escolhido o uso de LoRa como tecnologia principal, devido às suas características únicas, especialmente em aplicações que requerem comunicações de longa distância com baixo consumo energético.

O LoRa é uma tecnologia de comunicação sem fios desenvolvida especificamente para sistemas de longa distância e baixo consumo de energia. Utilizando a modulação de espalhamento espectral (*chirp spread spectrum*), o LoRa é capaz de transmitir dados a distâncias que podem ultrapassar 10 km em linha de vista, enquanto consome uma fração da energia necessária para tecnologias de maior largura de banda, como LTE [15] ou 4G [16, 17].

Comparado com outras tecnologias, como LTE/4G e XBee [18], o LoRa apresenta vantagens específicas para sistemas remotos e energeticamente eficientes. O LTE/4G oferece velocidades de transmissão muito superiores, sendo ideal para aplicações que exigem elevada largura de banda, como *streaming* de vídeo ou comunicação em tempo real. No entanto, o seu elevado consumo energético e dependência de infraestrutura de telecomunicações tornam-no inadequado para dispositivos móveis em áreas remotas [17]. Já o XBee, uma solução de curto alcance baseada em Zigbee, destaca-se pela simplicidade de integração e baixo consumo energético, mas a sua limitada cobertura geográfica restringe significativamente o seu uso em comunicações de longa distância [19, 20]. Assim, o LoRa apresenta-se como uma alternativa equilibrada, unindo alcance estendido e eficiência energética, especialmente relevante para este projeto.

Entre as vantagens do LoRa, destacam-se o alcance de comunicação longo, que pode superar 10 km em condições ideais, e o baixo consumo energético, essencial para prolongar a autonomia de dispositivos alimentados por bateria. A modulação utilizada oferece robustez contra interferências e ruídos, garantindo comunicação confiável em ambientes adversos, e o custo de implementação, tanto em termos de *hardware* como de infraestrutura, é significativamente inferior ao de outras soluções como LTE/4G. Por outro lado, o LoRa também possui limitações, como a baixa largura de banda, que o torna inadequado para transmissão de grandes volumes de dados, e uma maior latência na entrega de mensagens, que pode ser problemática para aplicações que exigem respostas em tempo real. Adicionalmente, a infraestrutura de comunicação pode necessitar de gateways específicos para integração com redes de maior escala, o que pode ser um desafio em cenários de maior complexidade.

Em termos de alcance, a tecnologia LoRa tem demonstrado um grande potencial (cerca de 45 vezes o alcance inicialmente proposto pela empresa Semtech, a criadora do LoRa). Recentemente, em Portugal, foi estabelecido um novo recorde de distância com LoRa, atingindo os 1336 km [21]. Este feito foi realizado no âmbito do projeto Custodian, com a instalação de trackers LoRa num barco de pesca e nas suas boias na costa de Sesimbra, Portugal. O tracker estabeleceu comunicação com um portal nas Ilhas Canárias, localizando-se a mais de 1.300 km de distância. Este recorde foi alcançado ao nível do mar, o que elimina variáveis potenciais introduzidas por altitudes variadas e fornece uma medida mais padronizada das capacidades da tecnologia. Este marco é particularmente

notável, pois demonstra a robustez da tecnologia LoRa para comunicações de longa distância em condições reais, e sublinha a sua capacidade de ultrapassar distâncias que antes eram consideradas inatingíveis para outras tecnologias de comunicação sem fios.

A Tabela 4.1 apresenta as diferenças de desempenho entre as tecnologias de comunicação avaliadas para este projeto.

**Tabela 4.1** Comparação entre Tecnologias de Comunicação

Tecnologia	Alcance (m)	Consumo Energético (mW)	Largura de Banda (kbps)
LoRa	Até 1 336	10–50	0.3–50
LTE/4G	Até 20 000	1 000–2 000	1 000–100 000
XBee (Zigbee)	Até 100	40–60	20–250

O LoRa foi escolhido devido ao seu baixo consumo energético, crucial para otimizar a autonomia em sistemas ciber-físicos alimentados por bateria, e ao seu longo alcance, que garante comunicações robustas em ambientes remotos sem a necessidade de infraestrutura complexa. A escolha do LoRa como interface de comunicação foi uma decisão estratégica, uma vez que permite a transmissão de dados de telemetria e a atualização de rotas em locais onde tecnologias como LTE/4G ou XBee seriam inviáveis, seja por questões práticas ou económicas. Assim, a aplicação do LoRa neste sistema ciber-físico assegura o cumprimento dos objetivos de eficiência energética e robustez operacional, oferecendo uma solução confiável e alinhada com os desafios reais enfrentados por USV.

#### 4.5.2 Inter-Integrated Circuit (I2C)

O protocolo I2C é um dos mais comuns para comunicação entre dispositivos periféricos em sistemas embebidos, como o USV. A sua principal vantagem reside na capacidade de utilizar apenas dois fios para comunicação: um para o relógio (*SCL*) e outro para dados (*SDA*), o que reduz significativamente a complexidade e o número de conexões no sistema. Este protocolo permite a comunicação entre múltiplos dispositivos com um único controlador mestre e vários escravos, sendo cada dispositivo identificado por um endereço único. Comparado com tecnologias como o *Serial Peripheral Interface* (SPI), o I2C apresenta uma largura de banda menor e é adequado para distâncias mais curtas.

A utilização do I2C neste projeto é fundamental para a integração de sensores e módulos que requerem comunicação de dados de forma eficiente e com baixo consumo energético. A flexibilidade do I2C permite expandir facilmente o número de dispositivos conectados ao sistema, com a adição de novos expansores de I/O, como descrito em [1] Secção 4.4, que podem ser utilizados para controlar sensores adicionais ou atuadores, como motores e atuadores de direção. O protocolo I2C é amplamente utilizado em módulos de sensores como o MPU9250 (IMU), que requerem comunicação constante para garantir medições em tempo real da orientação e movimento do USV.

Apesar das suas vantagens, o I2C apresenta algumas limitações que devem ser consideradas no desenho do sistema. Em primeiro lugar, trata-se de um protocolo *half-duplex*, dado que a linha de dados (*SDA*) é partilhada para envio e receção, impossibilitando transmissões simultâneas bidirecio-

nais. Além disso, a especificação original suporta até 127 dispositivos endereçáveis, mas na prática este número é frequentemente inferior devido a restrições elétricas e à possibilidade de colisões de endereços entre diferentes módulos. Outro aspeto limitador é o alcance físico do barramento, que tipicamente não deve exceder alguns metros devido à capacidade das linhas, tornando o I2C adequado apenas para sistemas compactos. Finalmente, a sua taxa de transferência, que pode variar entre 100 kbit/s (modo standard) e 3,4 Mbit/s (modo *high speed*), é consideravelmente inferior à de protocolos como SPI, o que pode ser uma limitação em sistemas que requerem alta largura de banda.

#### 4.5.3 Universal Asynchronous Receiver/Transmitter (UART)

O protocolo UART é um dos métodos mais simples e amplamente utilizados para comunicação série em sistemas embebidos. Diferente do I2C, em que um dos fios é dedicado ao sinal de relógio (*SCL*) e o outro transporta os dados (*SDA*), o UART utiliza duas linhas independentes: uma para transmissão (*TX*) e outra para receção (*RX*). Esta característica elimina a necessidade de um sinal de relógio externo e permite que a comunicação seja assíncrona. Para que dois dispositivos comuniquem corretamente, é necessário que concordem previamente sobre parâmetros como a taxa de transmissão (*baud rate*), o número de bits por *byte*, a paridade e os bits de *stop*.

Uma diferença fundamental entre os dois protocolos é que o UART é tipicamente *full-duplex*, dado que possui linhas independentes para transmissão e receção, possibilitando o envio e receção de dados em simultâneo. Já o I2C funciona em modo *half-duplex*, uma vez que todos os dispositivos partilham o mesmo canal de dados (*SDA*), sendo necessária a coordenação por parte do mestre, que inicia a comunicação, enquanto os escravos apenas respondem quando endereçados. Esta distinção tem impacto direto no desempenho: o UART garante maior fluidez em transmissões contínuas ponto-a-ponto, enquanto o I2C favorece cenários com múltiplos dispositivos num barramento partilhado.

Em sistemas como USV, o UART é frequentemente utilizado para comunicação com módulos de sensores que não requerem a complexidade do I2C, mas que necessitam de uma ligação direta e contínua. Um exemplo é a comunicação com receptores de GPS, que enviam dados em fluxo constante. Também pode ser aplicado em sensores de temperatura ou outros periféricos que exigem comunicação bidirecional simples. Nestes casos, o UART apresenta-se como uma alternativa acessível e eficiente.

Em contextos mais avançados, variantes como RS-232 e RS-485 são também relevantes. O RS-485, em particular, é amplamente utilizado em aplicações industriais por permitir comunicação a distâncias maiores e com múltiplos dispositivos. Ao contrário do UART tradicional, que se limita a ligações ponto-a-ponto, o RS-485 possibilita comunicação multi-endereço através de um único par de fios, tornando-se ideal para sistemas distribuídos em que diversos sensores e atuadores se encontram dispersos ao longo da embarcação.

Uma vantagem do UART em relação ao I2C é a possibilidade de comunicação a maiores distâncias, especialmente quando combinado com transmissores de maior potência. Contudo, não suporta de forma nativa a ligação de múltiplos dispositivos no mesmo barramento, o que limita a sua utilização em arquiteturas complexas. Por esse motivo, a escolha entre I2C e UART depende diretamente da aplicação: enquanto o I2C se destaca pela simplicidade e modularidade em barramentos partilhados,

o UART garante maior eficiência em transmissões contínuas ponto-a-ponto.

Finalmente, o UART é frequentemente utilizado em conjunto com conversores USB-Série, que facilitam a integração com computadores e plataformas de monitorização remota. Esta abordagem permite que os dados enviados pelo USV sejam facilmente analisados em tempo real ou armazenados para tratamento posterior, assegurando a interoperabilidade entre o sistema embarcado e os sistemas de apoio em terra.

## 4.6 Sumário

A arquitetura do sistema desenvolvida para o USV nesta TFM assenta em quatro pilares fundamentais: a estrutura física, o sistema de propulsão, controlo, os módulos de sensorização e comunicação. A estrutura em configuração de catamarã, construída com recurso a impressão 3D e reforçada com fibra de vidro, garante simultaneamente flutuabilidade, robustez e facilidade de integração dos módulos eletrónicos, através de uma caixa estanque central. Esta solução assegura condições adequadas para testes experimentais e validação de algoritmos de navegação, sem comprometer a compatibilidade com o USV-enautical.

Relativamente à propulsão, foram selecionados propulsores *brushless* U01, que se distinguem pela sua eficiência energética, robustez e capacidade de fornecer empuxo bidirecional, controlado através de ESC. O sistema foi concebido para operar com dois propulsores, equilibrando simplicidade e manobrabilidade, mas mantendo a escalabilidade até quatro unidades para cenários mais exigentes. A gestão energética é assegurada por baterias de 12 V de elevada capacidade, permitindo missões prolongadas.

No que respeita ao controlo, o ESC assume um papel central, interpretando os sinais de PWM provenientes do microcontrolador para regular a velocidade e o sentido de rotação. O processo de armamento garante segurança na inicialização, enquanto a capacidade de operar em ambos os sentidos de rotação proporciona ao USV elevada manobrabilidade em cenários marítimos complexos.

A integração sensorial recorre a dispositivos complementares: o IMU assegura medições contínuas de orientação (*yaw*, *pitch* e *roll*), essenciais para a estabilidade da embarcação, enquanto o GPS fornece posicionamento absoluto, compensando a deriva acumulada do IMU. Esta fusão de sensores garante navegação precisa e robusta, mesmo em trajetos longos ou em presença de perturbações externas. Para suportar a integração de múltiplos sensores, o sistema inclui expansores de comunicação que permitem a utilização combinada de I2C e UART, aumentando a flexibilidade da arquitetura.

Por fim, a interface de comunicação LoRa foi adotada como tecnologia principal para telemetria e receção de rotas em tempo real, devido ao seu baixo consumo energético e elevado alcance, já demonstrado em cenários reais de operação marítima. Em complemento, a utilização de protocolos como I2C e UART assegura a interoperabilidade entre os diversos módulos, permitindo escalabilidade e adaptabilidade da solução a diferentes configurações.

Em síntese, a arquitetura proposta conjuga robustez estrutural, eficiência energética, precisão sensorial e fiabilidade comunicacional, estabelecendo uma base sólida para a implementação de um sistema ciberfísico autónomo aplicável a missões de monitorização e exploração em ambientes marítimos.

# **Capítulo 5**

## **Implementação**

Neste capítulo, a implementação é analisada em duas vertentes complementares. Na primeira parte, dedicada ao *hardware*, descreve-se a plataforma de desenvolvimento na Secção 5.1, seguida da integração dos propulsores na Subsecção 5.2.1, da utilização dos sensores na Subsecção 5.2.2, dos módulos de comunicação e, por fim, da conceção da *Printed Circuit Board* (PCB) na Subsecção 5.2.4. Na segunda parte, referente ao *software*, detalham-se a arquitetura modular do código na Subsecção 5.3.2, os mecanismos de controlo de motores na Subsecção 5.3.3, a aquisição e processamento de dados dos sensores na Subsecção 5.3.4 e as estratégias de comunicação via LoRa na Subsecção 5.3.5. Esta organização permite compreender de forma clara e sistemática como os diferentes componentes foram concebidos e integrados para dar origem ao sistema ciberfísico proposto.

### **5.1 Plataforma de Desenvolvimento**

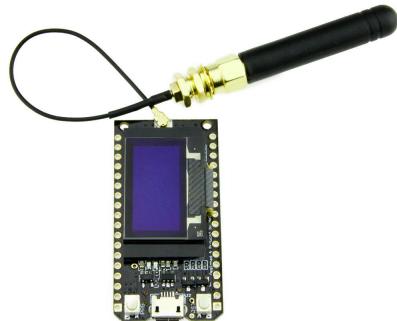
A plataforma de desenvolvimento utilizada neste trabalho constitui a base para a implementação tanto de *hardware* como de *software*, garantindo a integração eficiente de todos os módulos do sistema. Nesta secção são apresentados dois elementos fundamentais: em primeiro lugar, o microcontrolador *Espressif32* (ESP32), descrito na Subsecção 5.1.1, que assegura o processamento central e a coordenação das interfaces de comunicação; e, em segundo lugar, o ambiente de programação, detalhado na Subsecção 5.1.2, que fornece as ferramentas necessárias para a compilação, gestão de bibliotecas e controlo de versões do código. A análise conjunta destes componentes evidencia como a escolha da plataforma contribuiu para a robustez, modularidade e escalabilidade do protótipo desenvolvido.

#### **5.1.1 Microcontrolador Espressif32 (ESP32)**

O microcontrolador *Espressif32* (ESP32) é um sistema de elevado desempenho e reduzidas dimensões, dotado de um processador *dual-core* com frequência até 240 MHz e suporte a vírgula flutuante. Esta capacidade de processamento é largamente superior às necessidades do presente projeto, garantindo margem para futuras expansões e algoritmos mais exigentes em termos de cálculo.

Para a implementação descrita nesta TFM foi utilizado o módulo TTGO LoRa32, representado na Figura 5.1, que integra num único dispositivo o microcontrolador ESP32, um transceptor (trans-

mite e recebe sinais) LoRa e um *display Organic Light-Emitting Diode* (OLED) de 0.96 polegadas. Esta integração reduz significativamente a complexidade do hardware, uma vez que combina num só módulo os principais elementos de computação e comunicação necessários para o funcionamento do USV.



**Figura 5.1** ESP32 TTGO LoRa32

Do ponto de vista das interfaces de comunicação, o ESP32 suporta nativamente I2C, UART e SPI, além da geração de sinais PWM, o que o torna adequado para aplicações de controlo de atuadores e aquisição de dados de sensores heterogéneos. O módulo disponibiliza também conectividade sem fios através de *Wireless Fidelity* (Wi-Fi) e *bluetooth*, que, embora não tenham sido utilizados diretamente neste trabalho, representam uma mais-valia para cenários futuros, como a monitorização remota via redes locais ou a configuração simplificada de parâmetros através de dispositivos móveis.

O *display* OLED incorporado revelou-se particularmente útil durante a fase de testes e validação, ao permitir monitorizar em tempo real o estado do sistema. Através dele foi possível confirmar a receção de dados provenientes de sensores, a correta execução de comandos de controlo e a integridade das mensagens trocadas via LoRa. Este recurso facilitou o processo de *debug* e reduziu a dependência de sistemas de monitorização externos, acelerando a fase de desenvolvimento.

Outro aspecto relevante é a vasta comunidade de suporte e o ecossistema de bibliotecas disponíveis para o ESP32, tanto no ambiente Arduino como em *frameworks* mais avançadas, como a *Espressif IoT Development Framework* (ESP-IDF), que constitui a framework oficial de desenvolvimento disponibilizada pela Espressif programável em C. Esta característica contribui para uma maior fiabilidade no desenvolvimento, reduzindo riscos de integração e permitindo concentrar o esforço na implementação da arquitetura modular proposta para o USV.

Em síntese, o módulo TTGO LoRa32 constituiu uma plataforma robusta, versátil e de fácil integração, adequando-se não apenas às exigências do desenvolvimento deste protótipo de USV, mas também oferecendo capacidade de evolução para cenários mais complexos, como a execução de algoritmos de navegação autónoma em tempo real ou a integração com sistemas de monitorização distribuídos.

### 5.1.2 Ambiente de Programação

O desenvolvimento do *software* para o protótipo do USV foi realizado recorrendo ao ambiente de programação PlatformIO, integrado no editor Visual Studio Code. Esta escolha deve-se às vantagens oferecidas pelo PlatformIO, nomeadamente a gestão centralizada de bibliotecas e o controlo de versões das mesmas, a configuração automatizada do ambiente de compilação e a possibilidade de integração com múltiplos *frameworks* de desenvolvimento.

Um exemplo claro destas funcionalidades pode ser observado no ficheiro de configuração `platformio.ini`. Este ficheiro define todas as dependências e parâmetros necessários para compilar e carregar o projeto, assegurando que qualquer programador envolvido no desenvolvimento utiliza exatamente as mesmas versões de bibliotecas e definições de compilação. A Listagem 5.1 mostra um exemplo de um ficheiro de configuração do PlatformIO.

**Listagem 5.1** Exemplo de ficheiro `platformio.ini` utilizado no projeto

```
1 ; PlatformIO Project Configuration File
2 [env:ttgo-lora32-v1]
3 platform = espressif32@~5.0.0
4 board = ttgo-lora32-v1
5 framework = arduino
6 lib_ldf_mode = deep+
7 lib_deps =
8     bodmer/TFT_eSPI@^2.5.43
9     adafruit/Adafruit SSD1306@^2.5.13
10    adafruit/Adafruit GFX Library@^1.11.11
11    sandeepmistry/LoRa@^0.8.0
12    mikalhart/TinyGPSPlus@^1.1.0
13    plerup/EspSoftwareSerial@^8.2.0
14    alexmaurer-madis/SC16IS7X0@^1.0.1
15    sparkfun/SparkFun 9DoF IMU Breakout - ICM 20948 - Arduino Library@
16        ^1.3.0
17    adafruit/Adafruit PWM Servo Driver Library@^3.0.2
18    paulstoffregen/Time@^1.6.1
19    madhephaestus/ESP32Servo@^3.0.6
20    arduino-libraries/Madgwick@^1.2.0
21    nanopb/Nanopb@^0.4.91
22 monitor_speed = 115200
23 monitor_filters = direct
24 check_tool = clangtidy
```

Neste caso, a secção `[env:ttgo-lora32-v1]` especifica o ambiente de compilação direcionado para a placa TTGO LoRa32, fixando a versão da plataforma ESP32 (`espressif32@~5.0.0`) e o *framework* (`arduino`). A diretiva `lib_deps` lista todas as bibliotecas necessárias ao projeto, juntamente com a versão exata de cada uma. Assim, bibliotecas como Adafruit SSD1306, LoRa ou TinyGPSPlus são automaticamente descarregadas e mantidas consistentes em qualquer máquina de desenvolvimento, evitando problemas de incompatibilidade entre versões.

Este mecanismo garante que o projeto é reproduzível e portável, pois basta clonar o repositório e executar a compilação para obter um ambiente idêntico, sem necessidade de configurar manualmente as bibliotecas ou dependências.

No âmbito deste projeto, foi utilizada a *framework* Arduino do PlatformIO, pela sua simplicidade e pela ampla disponibilidade de bibliotecas compatíveis com o ESP32. Alternativamente, o ESP-IDF poderia ter sido utilizado, oferecendo maior controlo de baixo nível e otimizações de desempenho, mas a sua maior complexidade tornaria o ciclo de desenvolvimento menos ágil. Assim, a opção pelo ecossistema Arduino mostrou-se mais adequada à prototipagem rápida e à integração de múltiplos sensores e interfaces de comunicação.

Para além destas ferramentas, foi também utilizado o sistema de controlo de versões Git, com integração em repositório remoto público (*open source*) no GitHub [2]. Este processo assegurou a rastreabilidade das alterações, a organização das diferentes versões do código e a possibilidade de colaboração futura.

Embora o desenvolvimento principal tenha sido efetuado no Visual Studio Code com o suporte do PlatformIO, foram também realizados testes complementares na Arduino IDE, devido à sua simplicidade na programação inicial do ESP32 e na verificação rápida da compatibilidade das bibliotecas utilizadas.

Em síntese, a combinação destas ferramentas permitiu estabelecer um ambiente de desenvolvimento robusto, flexível e adaptado às exigências do projeto, ao mesmo tempo que garantiu escalabilidade e reproduzibilidade para trabalhos futuros.

## 5.2 Implementação de Hardware

A implementação do hardware do sistema desenvolvido nesta TFM assenta numa arquitetura modular, representada de forma esquemática na Figura ???. Tal como no projeto realizado em [1], procurou-se manter uma estrutura clara e escalável, onde cada componente desempenha um papel bem definido e comunica com a unidade central de forma eficiente.

O núcleo do sistema é constituído pelo microcontrolador ESP32, responsável por coordenar a execução das tarefas de controlo, comunicação e aquisição de dados. Através do barramento I2C, o ESP32 comunica com um expansor, responsável pela distribuição dos sinais de controlo e pela interface com os diferentes periféricos PWM. Este dispositivo disponibiliza 16 canais adicionais de saída a partir de apenas dois pinos de comunicação I2C (SDA e SCL), sendo ainda possível encadear múltiplos expansores para aumentar o número de canais disponíveis. Esta solução permite ultrapassar a limitação do número de pinos físicos do microcontrolador, assegurando a escalabilidade do sistema e reduzindo simultaneamente a complexidade das interligações elétricas, o que simplifica a integração de novos módulos funcionais.

Esta representação global serve como ponto de partida para a descrição detalhada apresentada nas subseções seguintes, onde são abordadas individualmente a integração dos propulsores, a utilização dos sensores, os módulos de comunicação e a conceção da PCB.

### 5.2.1 Integração dos Propulsores

A propulsão do USV é assegurada por dois propulsores elétricos independentes, cada um alimentado por uma bateria dedicada de 12 V. Esta configuração garante uma fonte de energia estável e suficientemente dimensionada para fornecer a corrente necessária ao funcionamento contínuo dos motores, reduzindo simultaneamente o risco de sobrecarga caso ambos fossem alimentados por uma única bateria.

A descrição detalhada das características dos motores e do respetivo sistema de controlo eletrónico encontra-se apresentada na Secção 4.2, enquanto o princípio de funcionamento dos ESC é explicado na Secção 4.3. Aqui, foca-se a forma como esses componentes foram efetivamente integrados no protótipo desenvolvido.

Cada propulsor encontra-se ligado ao respetivo ESC, responsável por converter os sinais de controlo recebidos em impulsos elétricos adequados ao motor *brushless*. Tal como descrito anteriormente, o controlo é realizado por sinais de PWM, sendo que, no protótipo implementado, o propulsor esquerdo foi ligado ao canal 14 do expansor, enquanto o propulsor direito foi associado ao canal 15. Apesar desta atribuição, qualquer outro canal disponível poderia ser utilizado, uma vez que a correspondência é definida ao nível do *software*.

O funcionamento dos motores segue o princípio explicado na Secção 4.3, em que a largura do pulso de PWM define o sentido e a intensidade da rotação. Esta abordagem permite um controlo independente de cada propulsor, possibilitando tanto movimentos de translação em linha reta como manobras de rotação em torno do próprio eixo, aumentando significativamente a manobrabilidade da embarcação.

No que respeita aos testes, cada motor foi inicialmente verificado de forma individual, em conjunto com o seu respetivo ESC, assegurando a correta resposta a diferentes larguras de pulso, a ausência de falhas de comunicação e a estabilidade térmica durante o funcionamento contínuo. Apenas após a validação isolada de cada propulsor se procedeu à sua integração conjunta no sistema, confirmando a coerência do funcionamento em paralelo e a capacidade de executar manobras de coordenação.

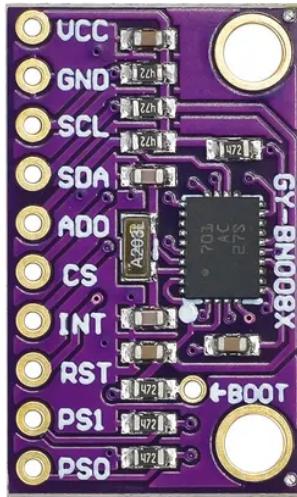
Esta estratégia de integração modular e progressiva assegura que cada propulsor pode ser controlado e validado de forma independente, ao mesmo tempo que garante a escalabilidade do sistema, permitindo a futura adição de mais motores ou a substituição dos atuais sem alterações significativas à arquitetura global.

### 5.2.2 Sensores

Nesta subseção são descritos os principais sensores integrados no USV, responsáveis pela recolha de dados de navegação e monitorização ambiental. São apresentados o IMU, para determinação do *yaw*, *pitch* e *roll*, e o módulo GPS, para aquisição de posição geográfica. Adicionalmente, discutem-se aspectos de integração elétrica e física relevantes para o funcionamento estável do sistema.

## Inertial Measurement Unit (IMU)

Para a determinação do *yaw*, *pitch* e *roll* do USV, foi utilizado um IMU baseado no sensor ICM-20948, ilustrado na Figura 5.2. Este dispositivo integra num único chip um acelerómetro de três eixos, um giroscópio de três eixos e um magnetómetro de três eixos, totalizando nove graus de liberdade (9-DOF *gyro-stabilized eCompass* [22]).



**Figura 5.2** IMU ICM-20948 utilizada no protótipo

A combinação destes sensores permite medir acelerações lineares, taxas de rotação e intensidade do campo magnético terrestre, fornecendo assim dados fundamentais para o cálculo do *yaw*, *pitch* e *roll*.

Em termos de integração física, o módulo IMU foi ligado ao microcontrolador ESP32 através do barramento I2C, utilizando os pinos SDA e SCL. No protótipo, foram atribuídos os pinos GPIO 21 (SDA) e GPIO 22 (SCL), respetivamente, recorrendo a uma frequência de comunicação de 400 kHz (*I2C Fast Mode*). Esta interface garante uma comunicação fiável e de elevada velocidade, adequada para a leitura periódica dos dados dos nove eixos.

O ICM-20948 suporta dois endereços I2C distintos (0x68 e 0x69), selecionáveis via configuração de *hardware*, o que possibilita a integração simultânea de múltiplos módulos no mesmo barramento, caso seja necessário. No presente projeto, foi utilizada a configuração por omissão (0x68).

A ligação elétrica do módulo foi realizada diretamente à linha regulada de 3.3 V fornecida pelo ESP32, estabelecendo-se também a referência comum de massa (GND). Esta configuração garante compatibilidade elétrica entre os dispositivos e elimina a necessidade de conversores de nível lógico, uma vez que tanto o ESP32 como o ICM-20948 operam nativamente a 3.3 V.

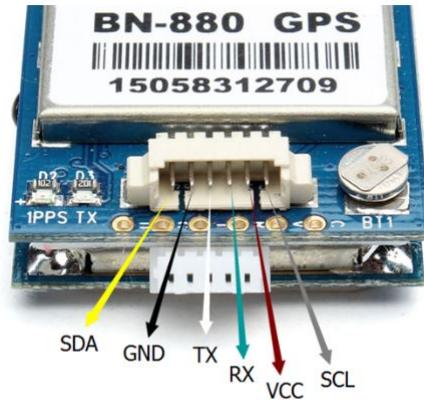
O módulo foi montado em proximidade com a unidade de processamento central, dentro da caixa estanque, de forma a reduzir o comprimento dos cabos e minimizar a suscetibilidade a ruídos elétricos. A posição física do sensor foi escolhida de modo a manter o alinhamento dos seus eixos com a estrutura da embarcação, simplificando assim a interpretação dos dados brutos.

Embora o processo de calibração e compensação de erros seja abordado posteriormente na Secção 5.3, é importante referir que, a nível de *hardware*, a disposição do sensor e a utilização de ligações curtas e estáveis contribuem para a redução de erros sistemáticos, nomeadamente ruído

eletromagnético e interferência cruzada entre sinais.

### Global Positioning System (GPS)

Para a determinação da posição geográfica do USV, foi utilizado o módulo BN-880 GPS, ilustrado na Figura 5.3. Este receptor integra um chip de posicionamento GNSS e uma antena cerâmica incorporada, sendo amplamente utilizado em aplicações embarcadas devido à sua elevada sensibilidade e ao baixo tempo de aquisição de sinal.



**Figura 5.3** Módulo GPS BN-880 utilizado no protótipo e respetivo *pinout*

O módulo GPS disponibiliza os dados em formato *National Marine Electronics Association* (NMEA) [23], um padrão amplamente adotado para a transmissão de coordenadas de latitude, longitude, velocidade e tempo. Estes dados são enviados através de uma interface série UART, que estabelece comunicação direta entre o módulo e o microcontrolador.

No entanto, dado que a porta série principal do ESP32 foi reservada para tarefas de *debug* e monitorização via USB, recorreu-se a um conversor I2C-UART (SC16IS750), que permitiu a criação de uma porta série adicional a partir do barramento I2C. Esta solução garantiu a integração do GPS sem comprometer a disponibilidade da interface série nativa do ESP32, mantendo a arquitetura modular e expansível do sistema.

Um aspecto crítico na integração deste módulo prende-se com a compatibilidade elétrica. Embora o BN-880 seja alimentado a 5 V, o ESP32 opera a 3.3 V nos seus pinos de entrada/saída digitais. Para evitar danos no microcontrolador e assegurar a comunicação fiável, foi introduzido um conversor de nível lógico, responsável por adaptar os sinais da interface UART entre os dois dispositivos.

A alimentação do módulo é assegurada diretamente pela linha de 5 V do sistema, enquanto as ligações de dados (TX e RX) passam pelo conversor de nível lógico antes de chegar ao ESP32. Esta configuração garante a integridade do sinal e a proteção dos módulos. As ligações elétricas detalhadas podem ser observadas no Anexo A, na Figura A.5, que apresenta a esquemática correspondente.

Tal como o IMU, o módulo GPS foi instalado no interior da caixa estanque, com especial atenção à orientação da antena integrada, de forma a manter visibilidade direta com o céu e reduzir obstruções ao sinal.

### 5.2.3 Comunicação

A comunicação entre o USV e a estação remota é assegurada pela tecnologia *Long Range* (LoRa), integrada no próprio ESP32 TTGO LoRa32, que incorpora o transceptor Semtech SX1276. Esta solução permitiu simplificar a arquitetura de hardware, eliminando a necessidade de módulos externos dedicados de rádio e garantindo ao mesmo tempo baixo consumo energético e elevada robustez em cenários de operação remota.

A configuração do transceptor LoRa foi realizada de acordo com a regulamentação europeia, utilizando uma frequência de operação de 866 MHz.

O sistema foi configurado para operar com um *Spreading Factor* de 12, que maximiza o alcance da comunicação ao custo de um *bitrate* mais baixo. Apesar de não ter sido modificada no protótipo, a biblioteca LoRa permite também ajustar parâmetros como a potência de transmissão, a largura de banda e o esquema de codificação, o que poderá ser explorado em trabalhos futuros para otimizar o compromisso entre alcance, robustez e eficiência energética.

A comunicação pode ser utilizada para dois propósitos principais:

1. Comandos remotos, que foram implementados no âmbito deste trabalho, consistem na receção de instruções provenientes da estação remota. Estes comandos permitem tanto a atualização de rotas sob a forma de *waypoints* como a navegação manual, através de ordens de movimento direto (avante e atrás), lateral (esquerda e direita) e paragem. A implementação destes comandos garante a possibilidade de alternar entre controlo autónomo e intervenção manual sempre que necessário.
2. Telemetria, que consiste no envio periódico de dados essenciais para a monitorização do sistema, incluindo posição GPS, orientação obtida pelo IMU, estado da bateria e modo de controlo ativo (manual ou automático). Esta funcionalidade não foi implementada na presente fase do trabalho, encontrando-se prevista para desenvolvimentos futuros.

### 5.2.4 Printed Circuit Board (PCB)

Com o objetivo de integrar todos os módulos do sistema de forma compacta e fiável, foi desenvolvida uma PCB dedicada, que reúne numa única placa todos os elementos de controlo, aquisição e comunicação do protótipo do USV. A centralização numa única PCB reduz significativamente a complexidade da cablagem, minimiza possíveis falhas de ligação e facilita a montagem, manutenção e escalabilidade do sistema.

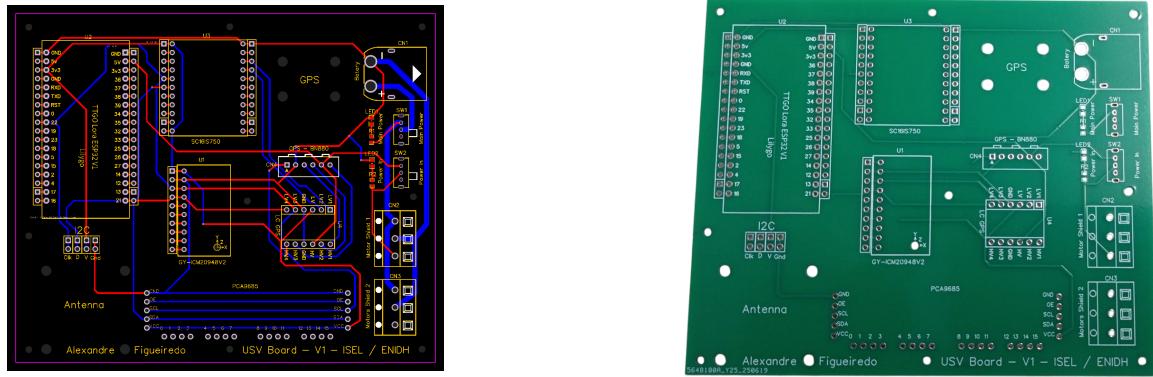
Na sua conceção, a placa incorpora diferentes blocos funcionais. O processamento central é assegurado pelo módulo TTGO LoRa32, baseado no microcontrolador ESP32, que tem a responsabilidade de gerar sinais PWM, gerir a comunicação via I2C, UART e LoRa, e garantir a ligação ao sistema através de uma antena dedicada. O controlo dos motores é realizado por um expansor PCA9685, que disponibiliza até 16 canais de PWM, dos quais dois foram configurados para comandar os propulsores através dos respetivos ESC.

No que respeita à navegação, o recetor GPS BN-880 foi integrado no sistema através de um conversor SC16IS750, que atua como ponte entre a interface UART do módulo e o barramento

I<sub>2</sub>C, permitindo a sua comunicação com o microcontrolador sem comprometer a arquitetura modular. Complementarmente, o sensor IMU ICM-20948, também conectado por I<sub>2</sub>C, assegura medições contínuas de aceleração, velocidade angular e intensidade do campo magnético nos três eixos, fundamentais para a estimativa da orientação do veículo.

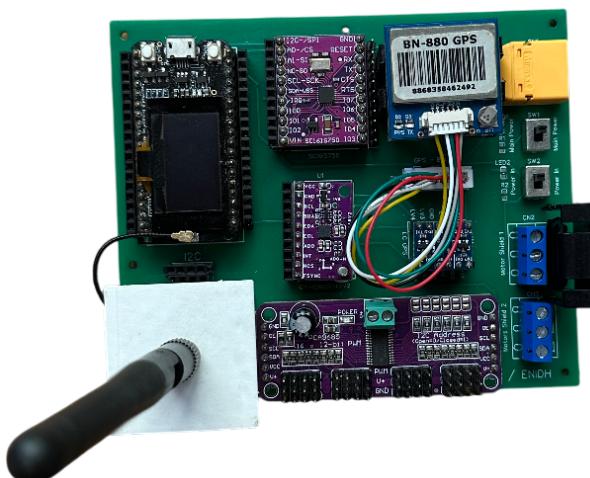
Por fim, a gestão de energia foi contemplada através da inclusão de conectores específicos para a entrada de energia principal a 12 V, interruptores de corte que permitem desligar o sistema de forma segura e *Light-Emitting Diode* (LED) indicadores que fornecem feedback visual sobre o estado de funcionamento.

A Figura 5.4 ilustra a comparação entre a esquemática desenvolvida e a implementação física da placa.



**Figura 5.4** Comparação entre a esquemática (A) e a implementação física (B) da PCB

As esquemáticas completas encontram-se documentadas no Anexo A, permitindo uma análise detalhada de cada ligação elétrica. A versão montada da PCB encontra-se representada na Figura 5.5, onde é possível identificar a disposição dos principais módulos e as camadas de interligação.



**Figura 5.5** Versão montada da PCB desenvolvida para o protótipo do USV

Em suma, a integração de todos os módulos numa única PCB permitiu não só simplificar o

design global do sistema, mas também garantir maior fiabilidade elétrica e reduzir a probabilidade de falhas por mau contacto ou cablagem incorreta. Esta abordagem aumenta a robustez do protótipo e facilita a sua replicação em versões futuras.

## 5.3 Implementação de Software

A implementação de *software* constituiu a camada lógica do sistema desenvolvido, responsável por coordenar a interação entre os diferentes módulos de hardware, assegurar o processamento em tempo real dos dados de sensores, gerir a comunicação sem fios e aplicar os algoritmos de navegação. Esta camada foi concebida de forma modular, permitindo que cada componente seja desenvolvido, testado e substituído de forma independente, sem comprometer a integridade do sistema global.

O código foi estruturado para garantir três requisitos principais: funcionamento não bloqueante, escalabilidade e reutilização. O primeiro requisito assegura que todas as tarefas críticas, como leitura de sensores, controlo dos motores e receção de comandos, são executadas sem interrupções prolongadas, preservando a responsividade do sistema. O segundo requisito garante que novos módulos ou algoritmos de controlo podem ser facilmente integrados, acompanhando a evolução do protótipo. O terceiro requisito promove a portabilidade do código para outros projetos semelhantes, mantendo a mesma filosofia modular que guiou a implementação do Robot Didático [1].

Importa ainda destacar que todo o código desenvolvido no âmbito desta TFM foi disponibilizado em regime de *open source*, encontrando-se acessível num repositório público de controlo de versões na plataforma GitHub [2]. Esta decisão assegura não apenas a transparência e reproduzibilidade científica, mas também a possibilidade de reutilização, extensão e validação por parte de outros estudantes, investigadores, profissionais da área e qualquer curioso.

Ao longo desta secção são descritos os principais aspetos da implementação de *software*, incluindo a integração dos subsistemas no ciclo principal na Subsecção 5.3.1, a arquitetura modular do código na Subsecção 5.3.2, o controlo dos motores na Subsecção 5.3.3, a leitura e processamento dos sensores na Subsecção 5.3.4 e a comunicação via LoRa com mensagens estruturadas em *Protobuf* na Subsecção 5.3.5. Estes elementos, em conjunto, constituem a base funcional que permite ao veículo operar de forma autónoma ou manual, com monitorização e controlo em tempo real.

### 5.3.1 Integração do Sistema

A integração do sistema foi concebida de forma modular, com o objetivo de simplificar o desenvolvimento e permitir que novos utilizadores ou projetos derivados possam reutilizar o código com um esforço mínimo. A interface de utilização resume-se à inicialização do objeto principal USV e à execução do método `loop()`, que coordena todos os subsistemas. Os sensores disponibilizam métodos dedicados para aceder aos dados mais recentes (e.g. `getGPSData()`, `getIMUData()`), garantindo uma utilização intuitiva e não bloqueante.

## Estrutura do Código Principal

O código principal (contido no ficheiro `main_usv.cpp`) organiza-se em duas funções fundamentais: `setup()`, responsável pela inicialização de todos os módulos (sensores, comunicação, propulsores e controladores), e `loop()`, que executa repetidamente o ciclo de controlo. A execução do `loop()` do objeto USV é suficiente para garantir a atualização contínua de sensores, o envio de telemetria, a receção de comandos e o controlo dos motores. A lógica interna foi desenhada para operar de forma assíncrona, evitando chamadas bloqueantes que pudessem comprometer a responsividade do sistema.

A Listagem 5.2 apresenta o código completo do ficheiro `main_usv.cpp`, ilustrando a simplicidade da interface de controlo.

### Listagem 5.2 Estrutura principal do programa do USV

```
1 #include "USV/USV.h"
2
3 GPS_BN880 gps;
4 IMU_ICM_20948 imu;
5 Expander expander(0x40);
6 ThrusterController leftController(14, 1000, 2000, 1500, 50, "left");
7 ThrusterController rightController(15, 1000, 2000, 1500, 50, "right");
8 Led yellow(expander, 7);
9 Led green(expander, 11);
10 USV usv = USV(expander, leftController, rightController, green, yellow);
11
12 void setup()
13 {
14     gps.setup();
15     imu.setup();
16
17     usv.begin();
18 }
19
20 void loop()
21 {
22     gps.loop();
23     imu.loop();
24
25     usv.getLoRaProto().receive();
26
27     GPSData gpsData = gps.getGPSData();
28     IMUData imuData = imu.getIMUData();
29     usv.loop(gpsData, imuData);
30 }
```

É importante destacar que os dados provenientes do GPS e da IMU são processados de forma independente. A obtenção destas informações é realizada explicitamente através das chamadas aos

métodos `getGPSData()` e `getIMUData()`, nas linhas 27 e 28 do código, respetivamente. Esta abordagem reflete uma decisão arquitetural relevante, pois garante que o objeto USV não está rigidamente acoplado a implementações específicas de sensores, mas antes a interfaces genéricas que fornecem dados estruturados de posicionamento e orientação.

Desta forma, a substituição ou atualização de sensores pode ser efetuada sem necessidade de alterar a lógica central do sistema, preservando a modularidade e a escalabilidade do *software*. A profundidade desta estratégia de abstração e independência sensorial é analisada em maior detalhe na Subsecção 5.3.4, onde se descrevem os mecanismos de aquisição e processamento de dados dos sensores.

### Sincronização de Sensores, Motores e Comunicação

A atualização dos sensores é realizada periodicamente, com cada módulo a gerir internamente a sua própria leitura. O GPS, através da classe `GPS_BN880`, descodifica continuamente as mensagens NMEA recebidas via ponte I2C-UART, enquanto o IMU (`IMU_ICM_20948`) atualiza os valores de aceleração, rotação e orientação utilizando o filtro de Madgwick [24]. Os propulsores são controlados via PWM gerado pelo expansor, com os valores definidos pela lógica de navegação implementada na classe `Control`. Paralelamente, a comunicação LoRa, encapsulada nas classes `LoRaDuplex` e `LoRaProto`, trata de forma não bloqueante tanto o envio periódico de telemetria como a receção de pacotes de controlo. Esta abordagem garante que nenhum módulo interrompe a execução do ciclo principal, assegurando a fluidez da operação em tempo real.

### Testes Incrementais

A validação do sistema foi realizada de forma incremental, seguindo uma estratégia de integração progressiva. Numa primeira fase foram testados os motores de forma isolada, validando a geração de sinais PWM, o processo de armamento dos ESC e a resposta dos propulsores. Posteriormente, os sensores foram integrados individualmente, confirmando a calibração do IMU e a receção estável de coordenadas GPS. Numa fase seguinte, foi estabelecida a comunicação via LoRa, assegurando a transmissão de telemetria e a receção de comandos em tempo real. Finalmente, todos os módulos foram integrados no sistema completo, testando a coerência do funcionamento conjunto, incluindo a execução de rotas automáticas baseadas em *waypoints* e a alternância entre modos manual e automático.

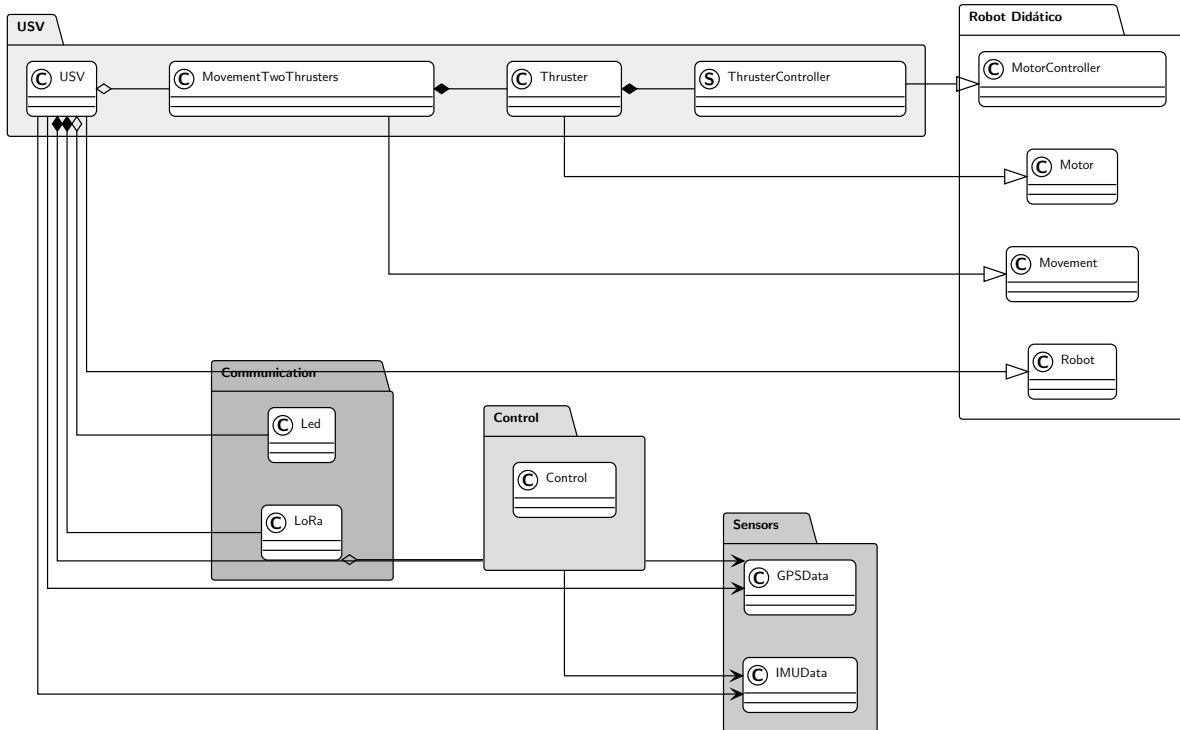
Em síntese, a arquitetura de *software* foi desenhada para ser modular, escalável e de fácil utilização, permitindo que qualquer utilizador possa controlar o USV com apenas algumas chamadas de alto nível, enquanto a lógica interna garante a sincronização e integração de todos os subsistemas de forma transparente.

### 5.3.2 Arquitetura do Código

A arquitetura de *software* foi concebida de forma modular, seguindo princípios de encapsulamento e separação de responsabilidades. Cada componente do sistema foi implementado numa biblioteca própria, organizada em *packages* correspondentes às suas funções principais, como propulsão, sensores,

comunicação e controlo. Esta abordagem modular facilita a manutenção, promove a reutilização do código e assegura a escalabilidade do sistema, permitindo que novos sensores ou atuadores possam ser integrados com alterações mínimas à base existente.

A Figura 5.6 apresenta um diagrama *Unified Modeling Language* (UML) simplificado que ilustra a arquitetura do sistema e as interações entre as suas classes principais (para diagramas mais complexos ver Anexo B).



**Figura 5.6** Arquitetura simplificada do sistema USV

Tal como se pode observar, o desenvolvimento partiu da biblioteca USV, que estende o projeto do robot didático descrito em [1]. Esse robot foi concebido de forma a ser genérico e extensível, suportando diferentes tipos de robôs móveis, incluindo USV. No presente trabalho, a classe USV herda de Robot e encapsula toda a lógica específica do protótipo, integrando a gestão dos propulsores, a leitura de sensores e os mecanismos de comunicação. A sua utilização é simplificada através de métodos de alto nível como `begin()`, responsável pela inicialização do *hardware*, e `loop()`, que coordena de forma não bloqueante a atualização de todos os módulos.

O controlo da propulsão é implementado pela classe `MovementTwoThrusters`, que herda de `Movement` e abstrai a complexidade da utilização de dois motores independentes. Esta classe fornece métodos como `left()` e `right()` que permitem executar manobras diferenciais sem necessidade de interação direta com os sinais PWM. Cada propulsor é representado pela classe `Thruster`, que estende de `Motor` e disponibiliza métodos de controlo direto, como `front(speed)`, `back(speed)` e `stop()`. O comportamento de cada propulsor é parametrizado através da estrutura `ThrusterController`, que herda de `MotorController` e define parâmetros como os pinos de PWM, os pulsos mínimos, máximos e neutros, bem como a frequência do sinal.

No que respeita aos sensores, foram criados pacotes independentes para GPS e IMU. A arqui-

tetura foi desenhada de forma a desacoplar a lógica de aquisição de dados da lógica de utilização, permitindo que qualquer sensor capaz de preencher as estruturas `GPSData` ou `IMUData` seja integrado sem modificações na camada de aplicação. Esta característica garante flexibilidade na substituição de sensores e assegura a compatibilidade futura com diferentes modelos de *hardware*.

A comunicação foi organizada no pacote `Communication`. A classe `LoRaDuplex` encapsula a interface direta com o transceptor LoRa integrado no TTGO LoRa32, implementando a lógica de envio e receção de pacotes em formato texto ou binário. Por cima desta camada foi desenvolvida a classe `LoRaProto`, responsável pela serialização e desserialização de mensagens com recurso a *Protobuf* [25]. A utilização de *Protobuf* permite definir de forma clara e estruturada os formatos de mensagem, ao mesmo tempo que reduz significativamente o tamanho dos pacotes transmitidos. Por exemplo, uma mensagem `StateMessage` representada em formato textual ocupa 33 bytes, enquanto a mesma mensagem em formato binário ocupa apenas 6 bytes, como ilustrado na Listagem 5.3.

---

#### Listagem 5.3 Comparação entre representação textual e binária com *Protobuf*

---

```

1 -- String (JSON-like):
2 {"state": "MANUAL", "manual": "FORWARD"}
3 Tamanho: 33 bytes
4
5 -- Protobuf (binario):
6 08 01 12 02 08 01
7 Tamanho: 6 bytes

```

---

Em comunicações LoRa, uma métrica crítica é o tempo no ar (*Time on Air* (ToA)), que corresponde à duração de transmissão de um pacote. O *Time on Air* (ToA) depende do tamanho da mensagem ( $N_{payload}$ ), do fator de espalhamento (SF), da largura de banda (BW) e do código de correção de erros (CR). O cálculo é realizado em três etapas principais.

Primeiro, calcula-se a duração de um símbolo ( $T_{sym}$ ), que depende diretamente do fator de espalhamento e da largura de banda:

$$T_{sym} = \frac{2^{SF}}{BW} \quad (5.1)$$

A Equação 5.1 mostra que o aumento do SF (maior robustez contra ruído) provoca símbolos mais longos, aumentando o tempo de transmissão. Em contrapartida, quanto maior for a largura de banda, menor será a duração de cada símbolo.

De seguida, determina-se o número de símbolos necessários para transportar a carga útil ( $N_{payload}$ ):

$$N_{payload} = 8 + \max \left( \left\lceil \frac{8 \cdot PL - 4 \cdot SF + 28 + 16 \cdot CRC - 20 \cdot H}{4 \cdot (SF - 2 \cdot DE)} \right\rceil \cdot (CR + 4), 0 \right) \quad (5.2)$$

Na Equação 5.2,  $PL$  representa o tamanho da carga útil (em bytes),  $CRC$  indica se o mecanismo de verificação de erros está ativo,  $H$  distingue entre cabeçalho implícito ou explícito e  $DE$  refere-se à otimização para baixas taxas de dados. Este cálculo permite estimar quantos símbolos adicionais são necessários, para além da pré-codificação e redundância.

Finalmente, obtém-se o tempo total de transmissão de um pacote ( $T_{\text{packet}}$ ), somando a duração do preâmbulo ao número de símbolos da carga útil:

$$T_{\text{packet}} = (N_{\text{preamble}} + 4.25 + N_{\text{payload}}) \cdot T_{\text{sym}} \quad (5.3)$$

Na Equação 5.3,  $N_{\text{preamble}}$  corresponde ao número de símbolos de sincronização enviados no início de cada transmissão, enquanto o fator 4.25 resulta de um ajuste especificado pelo protocolo LoRa [26]. Multiplicando pela duração de cada símbolo, obtém-se o tempo efetivo que o pacote permanece no ar.

Aplicando estas fórmulas ao exemplo descrito, com SF12, BW de 125 kHz e CR = 4/5, a transmissão de uma mensagem de 33 bytes apresenta um ToA aproximado de 1.81 segundos. Já para uma mensagem de apenas 6 bytes, o valor reduz-se para 0.99 segundos. Esta redução de aproximadamente 45% traduz-se numa menor probabilidade de colisões, numa diminuição da taxa de erro de pacotes e num consumo energético inferior.

Assim, ainda que o alcance teórico permaneça constante (pois depende apenas de SF, BW e CR), o alcance efetivo em cenários práticos é superior para mensagens mais curtas. Isto deve-se ao facto de pacotes mais compactos reduzirem a probabilidade de erros acumulados, a taxa de colisões e o tempo de ocupação do canal, contribuindo para uma comunicação mais robusta e eficiente.

Ainda no pacote de comunicação encontra-se a classe `Led`, que abstrai o controlo visual do sistema através de LED de estado. Esta classe disponibiliza métodos simples como `on()`, `off()` e `blink(duration, interval)`, sendo que o comportamento intermitente é implementado de forma não bloqueante mediante a chamada periódica do método `update()`.

Finalmente, a lógica de navegação e controlo do veículo foi encapsulada no pacote `Control`. A classe `Control` é responsável por gerir a alternância entre o modo manual e o modo automático, utilizando os dados dos sensores e os comandos recebidos via comunicação para determinar as ações a aplicar nos propulsores. No modo manual, os comandos são traduzidos diretamente em ações sobre os motores. No modo automático, foi implementado um controlo simples baseado em lógica `on/off`: sempre que o erro de rumo (*bearing*) é positivo, o veículo corrige para a esquerda, e quando o erro é negativo, corrige para a direita. Apesar de simples, este método revelou-se eficaz para um protótipo de dois propulsores. No entanto, a arquitetura modular permite a substituição futura deste mecanismo por controladores mais avançados, como *Proportional-Integral-Derivative* (PID) ou *Linear Quadratic Regulator* (LQR), sem impacto significativo no restante sistema.

A Listagem 5.4 apresenta a implementação da lógica de controlo responsável pela alternância entre os modos automático e manual de navegação.

#### Listagem 5.4 Excerto da classe `Control` demonstrando a lógica automática de controlo

```

1 void Control::control()
2 {
3     if (state == StateMessage_State_AUTOMATIC)
4     {
5         if (lastState != state)
6         {
7             automaticLed.on();
8         }
9     }
10 }
```

```

8         manualLed.off();
9         lastState = state;
10    }
11
12    automaticControl();
13 }
14 else if (state == StateMessage_State_MANUAL)
15 {
16     if (lastState != state)
17     {
18         automaticLed.off();
19         manualLed.on();
20         lastState = state;
21     }
22
23     manualControl(manualState);
24 }
25 }
26
27 void Control::automaticControl()
28 {
29     if (currentWaypoint >= waypoints.size() && waypoints.size() > 0)
30     {
31         automaticLed.blink(2000, 100);
32         movement.stop();
33         return;
34     }
35     setCourse(lastBearingError);
36 }
37
38 void Control::setCourse(int bearingError)
39 {
40     if (bearingError > 0)
41     {
42         movement.right(100, 0, bearingError);
43     }
44     else if (bearingError < 0)
45     {
46         movement.left(100, 0, -bearingError);
47     }
48     else
49     {
50         movement.stop();
51     }
52 }
```

O método `control()` inicia verificando o estado atual do sistema. Quando este se encontra em modo automático, o LED correspondente é ativado e o LED manual é desligado, fornecendo assim um *feedback* visual imediato ao operador. De forma análoga, quando o modo manual é selecionado, o LED manual é ligado e o LED automático é desligado. Esta alternância simples assegura que o estado de operação do veículo é sempre claramente indicado.

No modo automático, o método `automaticControl()` verifica se todos os *waypoints* definidos foram percorridos. Caso isso ocorra, o LED automático entra em modo intermitente e o movimento é interrompido, sinalizando o fim da missão. Caso contrário, é chamado o método `setCourse()`, que recebe como parâmetro o erro de rumo acumulado (`lastBearingError`) e executa a correção necessária.

A lógica implementada em `setCourse()` baseia-se diretamente no sinal desse erro: valores positivos correspondem a desvios que requerem uma correção para a direita, valores negativos exigem correções para a esquerda e, no caso de o erro ser nulo, o movimento é interrompido com a chamada a `movement.stop()`. Trata-se, portanto, de um controlador simples do tipo *on/off*, que se revelou eficaz para o seguimento de trajetórias definidas por *waypoints*. Apesar de básico, este mecanismo é suficiente para o protótipo descrito nesta TFM, mantendo no entanto a possibilidade de evolução para controladores mais sofisticados, como *Proportional-Integral-Derivative* (PID) ou *Linear Quadratic Regulator* (LQR), graças à natureza modular do código.

Em síntese, a arquitetura de *software* concebida garante robustez, clareza e expansibilidade, constituindo uma base sólida não apenas para o funcionamento do protótipo, mas também para evoluções futuras em sistemas mais complexos.

### 5.3.3 Controlo dos Motores

Como discutido na Secção 4.3, o controlo dos motores é efetuado através de um ESC, que requer sinais de PWM com larguras de pulso específicas, expressas em microssegundos, e não apenas uma variação do *duty cycle*. O intervalo típico situa-se entre  $1000 \mu\text{s}$  e  $2000 \mu\text{s}$ , correspondendo respetivamente às rotações máximas nos sentidos inverso e direto, sendo  $1500 \mu\text{s}$  o ponto neutro em que o motor permanece parado. Torna-se assim necessário mapear valores percentuais de velocidade para larguras de pulso compatíveis com o ESC.

Para esse fim, a classe `Thruster` inclui o método `setDirection(clockwise)`, que define o sentido de rotação (horário ou anti-horário) e ajusta os limites mínimo, máximo e neutro do sinal PWM. Com esta abstração, funções de mais alto nível, como `front(speed)` e `back(speed)`, permitem ao programador fornecer apenas uma percentagem de velocidade, sendo a conversão para o sinal em microssegundos tratada internamente.

A ponte entre o domínio temporal, expresso em microssegundos, e o domínio do controlador PCA9685, expresso em *ticks* de 12 bits, é assegurada pelo método `pulseToTicks(microseconds)`. A relação matemática implementada encontra-se expressa na Equação 5.4.

$$n_{ticks} = \frac{t_{pulse} \cdot N_{ticks}}{10^6 / f_{PWM}} \quad (5.4)$$

em que  $t_{pulse}$  representa a largura de pulso em microssegundos,  $N_{ticks} = 4096$  é a resolução

interna do PCA9685 em ticks e  $f_{PWM} = 20000$  a frequência de operação configurada em microssegundos. Esta conversão garante que os valores de pulso mínimo, máximo e neutro são corretamente interpretados pelo ESC.

A Listagem 5.5 apresenta parte da implementação da classe Thruster, ilustrando a lógica de mapeamento dos sinais PWM para cada direção de rotação.

#### Listagem 5.5 Parte da classe Thruster

---

```

1 void Thruster::setDirection(bool clockwise)
2 {
3     if (clockwise)
4     {
5         _expander.setMinPwmOn(pulseToTicks(controller.PWM_STOP_PULSE));
6         _expander.setMaxPwmOn(pulseToTicks(controller.PWM_MIN_PULSE));
7     }
8     else
9     {
10        _expander.setMinPwmOn(pulseToTicks(controller.PWM_STOP_PULSE));
11        _expander.setMaxPwmOn(pulseToTicks(controller.PWM_MAX_PULSE));
12    }
13 }
14
15 uint16_t Thruster::pulseToTicks(int microseconds)
16 {
17     return (uint16_t)((microseconds * 4096L) / (1000000 / controller.
18     PWM_FREQ));

```

---

Na função `pulseToTicks()`, o parâmetro `PWM_FREQ` corresponde à frequência do sinal PWM ( $f_{PWM}$ ), expressa em hertz. O fator  $10^6$  é utilizado para converter a frequência em período (em microssegundos), dado que  $1\text{ s} = 10^6\ \mu\text{s}$ . Desta forma, o denominador da Equação 5.4 ( $10^6/f_{PWM}$ ) representa o período de um ciclo PWM, assegurando a coerência entre a formulação matemática e a implementação em código.

Após este mapeamento, o método `setDutyCycle()` da classe Expander, apresentado na Listagem 5.6, traduz o valor normalizado de *duty cycle* no número correspondente de *ticks* do PCA9685, assegurando que a saída do controlador corresponde ao pulso pretendido. Esta camada adicional de abstração permite que o controlo seja realizado em percentagens de velocidade, ao mesmo tempo que o sistema converte internamente esses valores nos pulsos temporizados requeridos pelo ESC.

#### Listagem 5.6 Método `setDutyCycle()` da classe Expander

---

```

1 int Expander::setDutyCycle(byte channel, float duty_cycle)
2 {
3     int pwm = map(duty_cycle, _MIN_DUTY_CYCLE, _MAX_DUTY_CYCLE,
4                   _MIN_PWM_ON, _MAX_PWM_ON);
5     _board.setPWM(channel, _PWM_OFF_POINT, pwm);
6     return pwm;

```

Esta implementação retoma e generaliza a lógica desenvolvida no projeto do Robô Didático [1], em que já haviam sido testados motores com diferentes requisitos de controlo. O resultado é um sistema flexível e extensível, capaz de controlar não apenas os motores *brushless* utilizados neste trabalho, mas também qualquer outro motor que dependa de sinais PWM baseados em temporização em vez de simples variação do *duty cycle*.

### 5.3.4 Leitura e Processamento dos Sensores

A leitura e processamento dos sensores foi implementada de forma modular e não bloqueante, garantindo que o `loop()` principal do sistema permanece responsivo mesmo em cenários de elevada carga de processamento. Cada sensor possui uma classe dedicada responsável pela aquisição e atualização periódica dos dados, mantendo internamente uma estrutura de estado que pode ser acedida a qualquer momento pelo restante sistema. Assim, a cada iteração do ciclo principal, a última leitura válida encontra-se disponível sem necessidade de esperar por novas medições.

No caso do IMU, a classe `IMU_ICM_20948` realiza leituras periódicas do acelerómetro, giroscópio e magnetómetro integrados no chip ICM-20948, armazenando os valores em objetos do tipo `IMUData`. Esta classe integra também o algoritmo de fusão de sensores de Madgwick, responsável por combinar as medições dos diferentes sensores para fornecer estimativas robustas dos ângulos de *yaw*, *pitch* e *roll*. A utilização deste filtro garante maior estabilidade e precisão, reduzindo a acumulação de erros proveniente do giroscópio e compensando variações transitórias. O resultado é um vetor de orientação continuamente atualizado, acessível através do método `getIMUData()`.

Já para o GPS, a classe `GPS_BN880` é responsável por gerir a comunicação via interface I2C-UART com o receptor BN-880. As mensagens NMEA recebidas são descodificadas em tempo real utilizando a biblioteca `TinyGPSPlus` [27], sendo posteriormente armazenadas numa estrutura `GPSData`. Esta estrutura disponibiliza informações completas como latitude, longitude, velocidade, direção e número de satélites em uso, bem como indicadores de qualidade do sinal. Tal como no caso do IMU, o acesso a estes dados é realizado de forma assíncrona através do método `getGPSData()`, garantindo que o programa principal pode obter a posição mais recente sem necessidade de reexecutar o processo de descodificação.

Como previamente referido na Subsecção 5.2.2, esta abordagem, baseada em estruturas de dados desacopladas da implementação específica do sensor, permite substituir ou adicionar novos módulos de forma transparente. Por exemplo, a utilização futura de um receptor GNSS multiconselação ou de um IMU com maior precisão poderá ser facilmente incorporada sem modificações no código que consome os dados, uma vez que a interface pública das classes `GPSData` e `IMUData` permanecerá inalterada.

### 5.3.5 Comunicação LoRa

O envio de dados é realizado através dos métodos `sendPacket()`, disponíveis em duas versões: uma para mensagens do tipo `String` e outra para blocos de dados binários (`uint8_t*`). A receção

de pacotes segue o mesmo princípio, podendo ser feita como texto simples ou diretamente em *buffer* binário, permitindo flexibilidade no tratamento de mensagens estruturadas. Para garantir compatibilidade e reduzir ambiguidades, as mensagens foram estruturadas utilizando *Protocol Buffers* (*Protobuf*), definidos previamente em ficheiros .proto.

O *Protobuf* consiste num método de serialização binária altamente eficiente, desenvolvido pela Google, que permite representar estruturas de dados complexas em mensagens compactas e de fácil interpretação. Neste projeto, definiu-se no ficheiro USV.proto as mensagens StateMessage e WaypointsMessage. A primeira descreve o estado atual do sistema, incluindo o modo de controlo (manual ou automático) e, no caso manual, o comando em execução (frente, trás, esquerda, direita ou paragem). A segunda define listas de *waypoints*, representados por pares latitude/longitude, que podem ser transmitidos a partir de uma estação remota para atualizar rotas.

O principal benefício da utilização de *Protobuf* reside na sua eficiência: para além da redução no tamanho das mensagens, já apresentada na Secção 5.3.2, a serialização binária permite reduzir o tempo de transmissão (ToA) e, consequentemente, o consumo energético do sistema. Além disso, a definição explícita do formato das mensagens assegura compatibilidade entre diferentes versões do *software* e facilita a integração futura com outras plataformas de controlo ou monitorização.

Durante a fase de testes, verificou-se a correta inicialização do transceptor e a integridade dos pacotes transmitidos e recebidos. Foram validados contadores internos de envio e receção, disponíveis na classe LoRaDuplex, que permitem avaliar de forma contínua o desempenho da comunicação. Os ensaios confirmaram a estabilidade em cenários de linha de vista, assegurando que o sistema mantém comunicação fiável em condições típicas de operação em ambiente marítimo.

Em síntese, a integração da comunicação LoRa no TTGO LoRa32, em conjunto com a abstração fornecida pelas classes LoRaDuplex e LoRaProto, resultou numa solução eficiente, escalável e robusta, permitindo não apenas o envio de telemetria em tempo real, mas também a receção de comandos de controlo com latência mínima.

## 5.4 Sumário

A implementação do sistema ciberfísico descrito neste capítulo seguiu uma abordagem modular, tanto ao nível do *hardware* como do *software*. No que respeita ao hardware, foi realizada a integração progressiva dos propulsores, sensores, módulos de comunicação e gestão de energia numa PCB dedicada, assegurando fiabilidade, simplicidade de montagem e escalabilidade futura.

Ao nível do software, a arquitetura foi organizada em pacotes independentes, promovendo o encapsulamento de funcionalidades e a reutilização de código. Foram desenvolvidas bibliotecas próprias para o controlo dos motores, aquisição e processamento de dados de sensores, comunicação sem fios via LoRa e navegação autónoma baseada em *waypoints*. Esta modularidade garante não só a clareza e robustez do sistema, mas também a sua capacidade de evolução, permitindo a integração de novos sensores, algoritmos de controlo mais avançados ou protocolos de comunicação alternativos.

A utilização de técnicas de programação não bloqueantes assegurou a execução fluida do ciclo principal, conciliando a leitura de sensores, o envio de telemetria, a receção de comandos e o controlo dos propulsores em tempo real. A adoção de *Protobuf* para a serialização das mensagens reforçou a

eficiência da comunicação, reduzindo o tempo de transmissão e o consumo energético.

Em síntese, a implementação resultou num sistema funcional, robusto e escalável, capaz de realizar navegação autónoma com telemetria em tempo real, representando uma base sólida para futuras iterações e aplicações em cenários reais de veículos de superfície não tripulados.



# **Capítulo 6**

## **Validação e Testes**

A validação do sistema desenvolvido foi conduzida de forma incremental, começando por testes unitários a cada componente de *hardware*, passando pela integração em ambiente controlado e culminando na experimentação em cenários reais. Esta abordagem permitiu não apenas garantir o correto funcionamento de cada subsistema de forma isolada, mas também identificar precocemente potenciais falhas de integração, reduzindo o risco de insucesso em fases mais avançadas.

### **6.1 Testes Incrementais de Hardware**

Antes da integração completa no protótipo do USV, foram realizados testes individuais a cada módulo de *hardware*, recorrendo a programas de validação dedicados. Estes testes encontram-se documentados no repositório público associado a este projeto [2], na diretoria `examples`. Cada exemplo foi concebido para verificar a operação de um subsistema específico, assegurando que todos os módulos funcionavam corretamente de forma isolada.

No caso do IMU, foi implementada uma leitura contínua das acelerações, rotações e campo magnético, permitindo validar a calibração inicial e a estabilidade dos valores registados. Para o módulo GPS BN-880, procedeu-se à descodificação de mensagens NMEA e à aquisição de coordenadas em tempo real, confirmando tanto a receção de satélites como a fiabilidade da posição estimada. Relativamente aos propulsores, foram gerados sinais PWM direcionados aos ESC, testando o processo de armamento e a resposta do sistema a diferentes larguras de pulso.

A comunicação LoRa foi também alvo de verificação, com o envio e receção de mensagens simples que permitiram avaliar a estabilidade do canal e a correta configuração do transceptor. Por fim, o *display* OLED integrado foi utilizado para apresentar mensagens de *debug*, demonstrando a sua utilidade na monitorização em tempo real da receção de dados provenientes dos sensores.

Estes ensaios confirmaram o funcionamento básico de todos os módulos de forma independente, assegurando que as interfaces de comunicação (I2C, UART, PWM) e as camadas de abstração implementadas estavam operacionais antes da integração no sistema completo.

## 6.2 Integração e Testes de Sistema

Após a validação individual de cada componente, procedeu-se à integração progressiva do sistema. Inicialmente, os propulsores foram testados em conjunto, verificando a coerência do movimento diferencial e a execução de manobras básicas (frente, trás e rotações). Posteriormente, o IMU e o GPS foram integrados com o sistema de controlo, validando a fusão de dados de orientação e navegação com a lógica de *waypoints*.

Em paralelo, a comunicação LoRa foi testada para garantir a transmissão de telemetria e a receção de comandos remotos, sem comprometer o funcionamento do ciclo principal. A utilização de mensagens estruturadas com *Protobuf* demonstrou vantagens significativas na redução do tempo de transmissão e na fiabilidade da comunicação.

Todos os ensaios desta fase foram realizados em ambiente controlado de laboratório, permitindo afinar parâmetros de *software* e validar a robustez da implementação antes da realização de testes em condições reais.

## 6.3 Validação em Ambiente Real

A validação final do sistema ocorreu no âmbito do exercício internacional REPMUS25, considerado o maior exercício mundial de robótica marítima [28, 29]. O protótipo desenvolvido foi testado no subevento Naval-REX25, organizado pela Escola Naval [30], que visa promover a experimentação tecnológica e a cooperação em cenários operacionais simulados, no contexto das operações não tripuladas.

Este teste constituiu um marco relevante na validação do sistema, uma vez que permitiu avaliar o desempenho do USV em condições operacionais reais, incluindo a comunicação de longo alcance via LoRa e a execução de rotas predefinidas com recurso ao GPS e ao IMU.

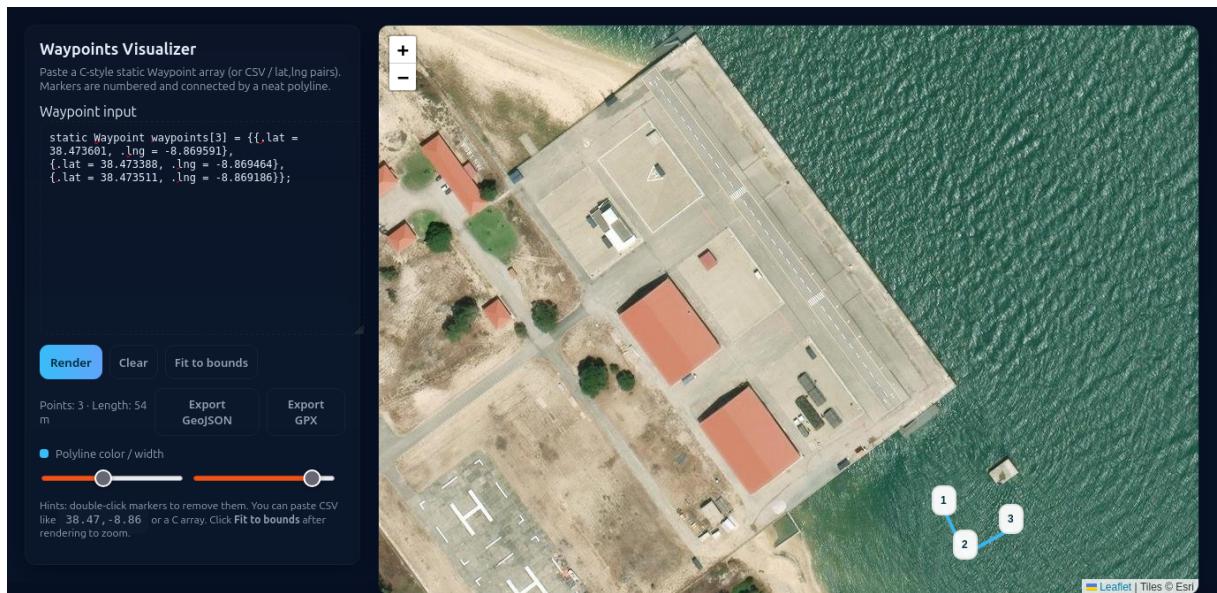
A participação no Naval-REX25, integrado no REPMUS25, demonstrou a aplicabilidade prática da solução desenvolvida e reforçou o contributo da Escola Náutica e da investigação nacional no domínio dos sistemas ciberfísicos aplicados à robótica marítima. Esta validação permitiu comprovar a robustez, escalabilidade e relevância prática do sistema, consolidando o trabalho desenvolvido no âmbito desta TFM.

Durante o processo de validação identificou-se a necessidade de complementar os testes com uma ferramenta de apoio à visualização geográfica dos *waypoints*, de forma a avaliar em tempo real a coerência e a precisão das rotas definidas. Para tal, foi desenvolvida uma aplicação em *Hyper Text Markup Language* (HTML) que, recorrendo a um servidor de *map tiling*, permite representar graficamente a posição geográfica de cada *waypoint* no globo. Esta ferramenta revelou-se essencial para a interpretação e validação prática das trajetórias, uma vez que forneceu uma perspetiva espacial clara do percurso programado e da sua correspondência com o ambiente real.

Para além da sua utilidade imediata na fase de testes, esta aplicação possui elevado potencial para ser expandida em trabalhos futuros, nomeadamente através da integração com sistemas de monitorização remota, planeamento dinâmico de rotas e visualização em tempo real do estado do USV. Desta forma, poderá constituir-se como um módulo complementar para a exploração e gestão

de missões autónomas em cenários operacionais complexos.

A Figura 6.1 apresenta uma captura de ecrã da aplicação desenvolvida, evidenciando a sua utilidade como suporte complementar à experimentação em campo.



**Figura 6.1** Ferramenta de visualização de waypoints

A Figura 6.2 contém um link para um vídeo na plataforma YouTube, que demonstra o USV a se movimentar.



**Figura 6.2** Demonstração do USV em funcionamento automático e manual (clique na imagem para abrir o vídeo no YouTube).

## 6.4 Sumário

A estratégia de testes adotada demonstrou-se eficaz, permitindo a deteção e resolução de problemas de forma faseada e garantindo que o sistema final apresentava fiabilidade e robustez. Os testes unitários asseguraram a operação correta de cada módulo, os ensaios de integração confirmaram a coerência da arquitetura modular, e a validação em ambiente real comprovou a aplicabilidade do protótipo em cenários operacionais complexos. Em conjunto, estes resultados confirmam o cumprimento dos objetivos estabelecidos e a viabilidade da solução proposta para o controlo autónomo de veículos de superfície não tripulados.



## Capítulo 7

# Conclusões e Trabalho Futuro

O trabalho desenvolvido permitiu conceber e validar um sistema ciberfísico modular para o controlo autónomo de embarcações do tipo USV. O protótipo integrou propulsores *brushless*, sensores de navegação (GPS e IMU), comunicação de longo alcance via LoRa, bem como uma arquitetura de software estruturada em módulos reutilizáveis. Os resultados obtidos demonstraram a viabilidade da solução em cenários reais, confirmando a robustez, a escalabilidade e o potencial de evolução do sistema.

Conclui-se que a abordagem proposta constitui uma base sólida para futuras investigações e aplicações práticas no domínio da robótica marítima com recurso a veículos de superfície não tripulados. O sistema desenvolvido provou ser capaz de operar em modo manual e automático e seguir rotas definidas por *waypoints* GPS, representando um contributo relevante para a área.

No que respeita ao trabalho futuro, identificam-se várias direções de evolução. Em primeiro lugar, destaca-se a possibilidade de integração do sistema no USV-enautica1 da Escola Superior Náutica Infante D. Henrique, permitindo validar o desempenho em cenários de maior escala. Outra vertente importante consiste na implementação de um sistema de controlo remoto manual, quer através de um comando dedicado semelhante aos utilizados em rádio controlo, quer por meio da integração com o comando desenvolvido em [10], assegurando assim a compatibilidade com plataformas operacionais.

A expansão para guardar telemetria num cartão SD constitui igualmente um caminho relevante. Do ponto de vista do controlo, será pertinente implementar e avaliar estratégias mais avançadas, nomeadamente controladores PID, LQR, de controlo ótimo ou até técnicas baseadas em aprendizagem por reforço, de modo a aumentar a precisão e a robustez da navegação.

Outra linha de evolução consiste na exploração da navegação cooperativa com múltiplos USV, o que implica a coordenação distribuída entre embarcações através de LoRa ou tecnologias alternativas de comunicação em rede. Neste mesmo sentido, a integração de sensores adicionais, como GNSS multiconstelação, câmaras ou radar de baixo custo, permitirá reforçar a precisão da navegação e a resiliência do sistema em ambientes mais complexos.

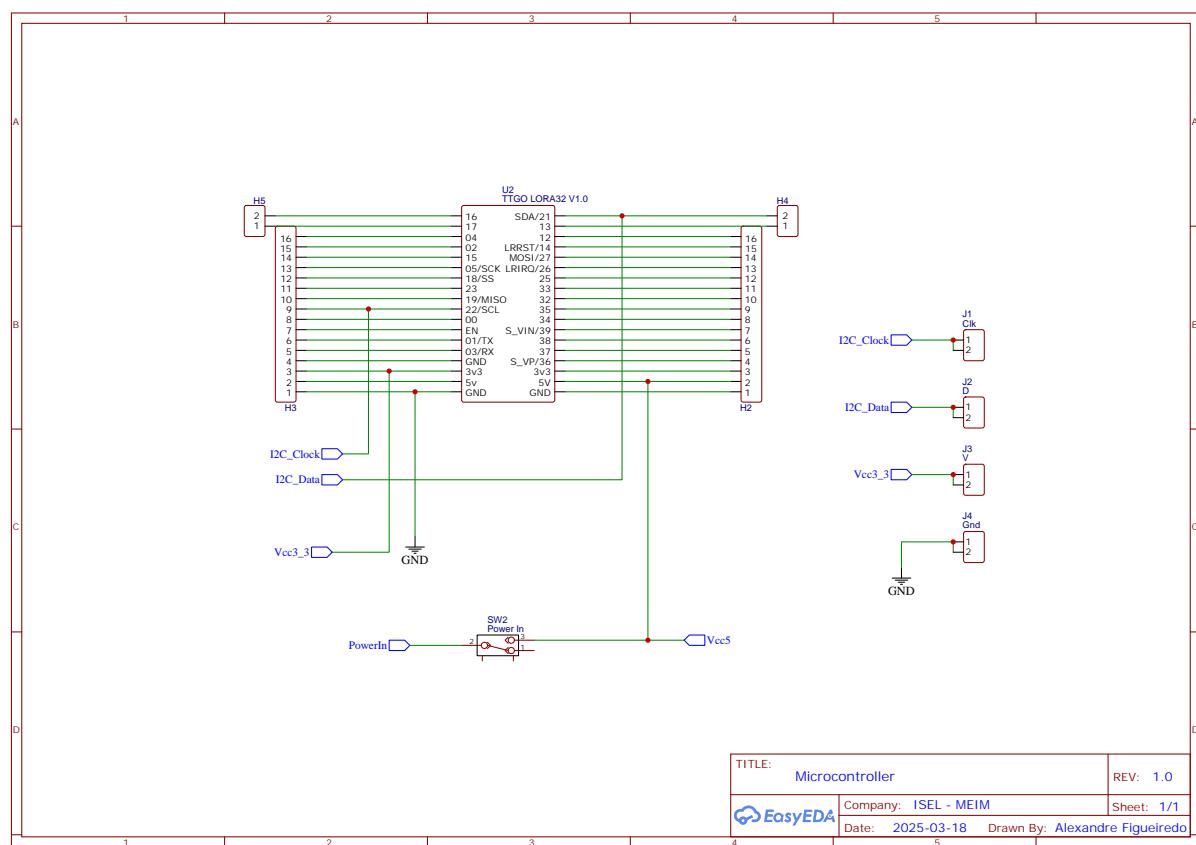
A sustentabilidade energética poderá ser reforçada através da integração de painéis solares ou de sistemas híbridos de alimentação, aumentando a autonomia em missões prolongadas. Paralelamente, o desenvolvimento de uma interface de monitorização remota, acessível via aplicação web ou móvel,

poderá facilitar o acompanhamento em tempo real e o registo histórico das missões realizadas. Finalmente, a realização de ensaios em condições ambientais adversas permitirá avaliar a resiliência e a fiabilidade do sistema em cenários reais de maior exigência.

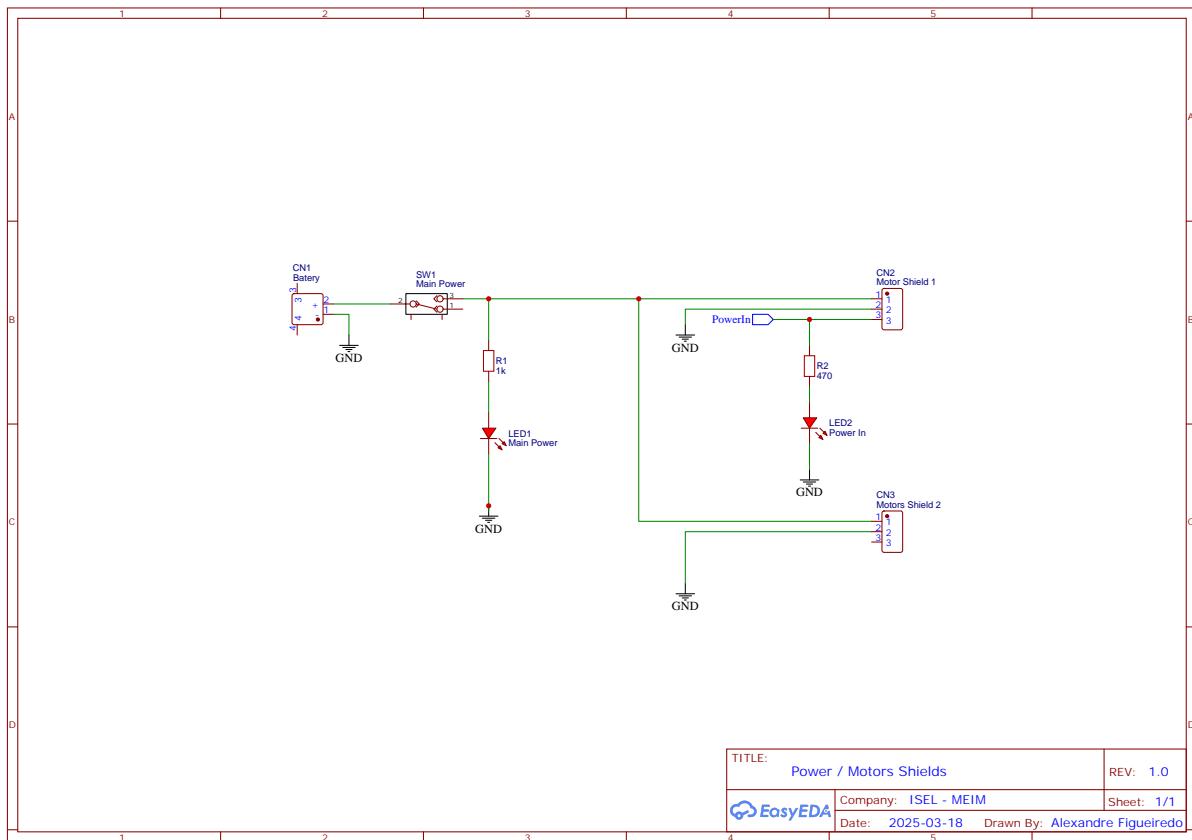
Em síntese, a evolução do sistema deverá orientar-se para o aumento da autonomia, da fiabilidade e da aplicabilidade prática, consolidando o seu papel enquanto plataforma experimental e operacional para investigação e desenvolvimento na área dos veículos marítimos não tripulados USV.

## Anexo A

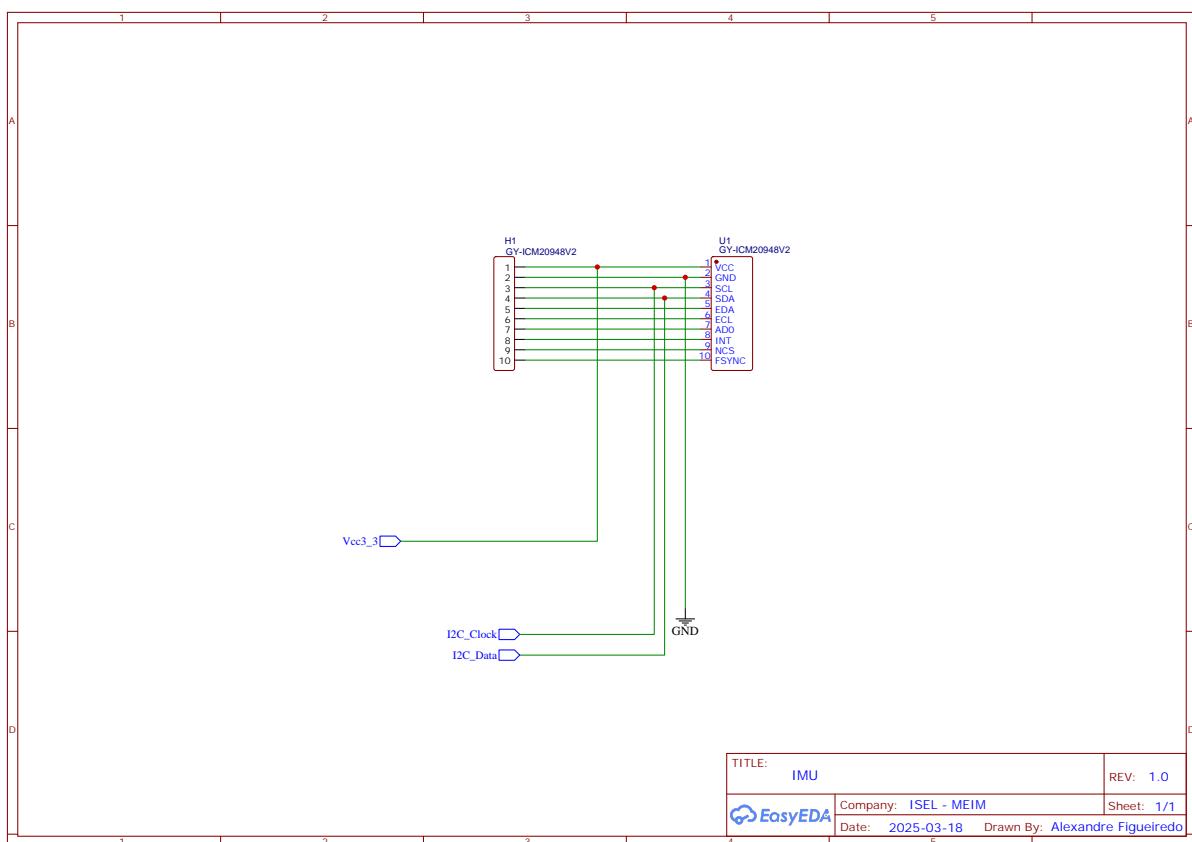
# Esquemáticas PCB



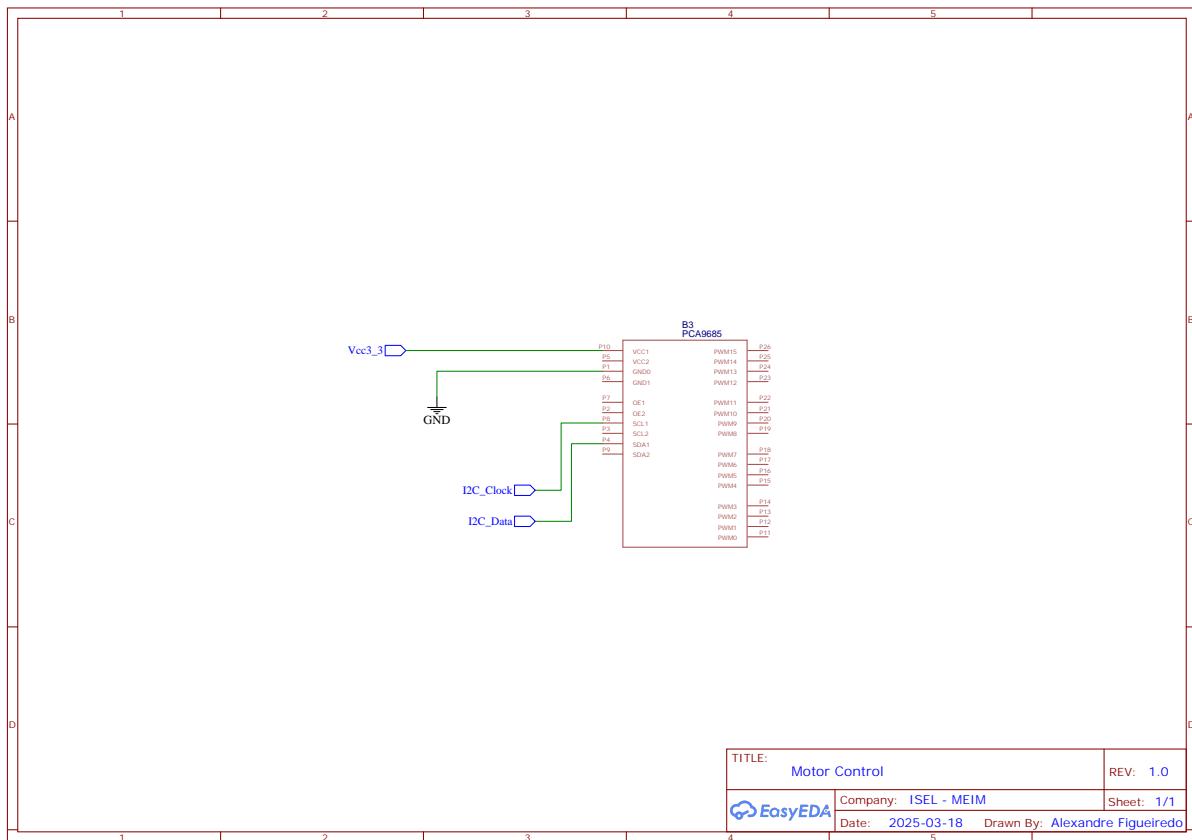
**Figura A.1** Esquemática do microcontrolador



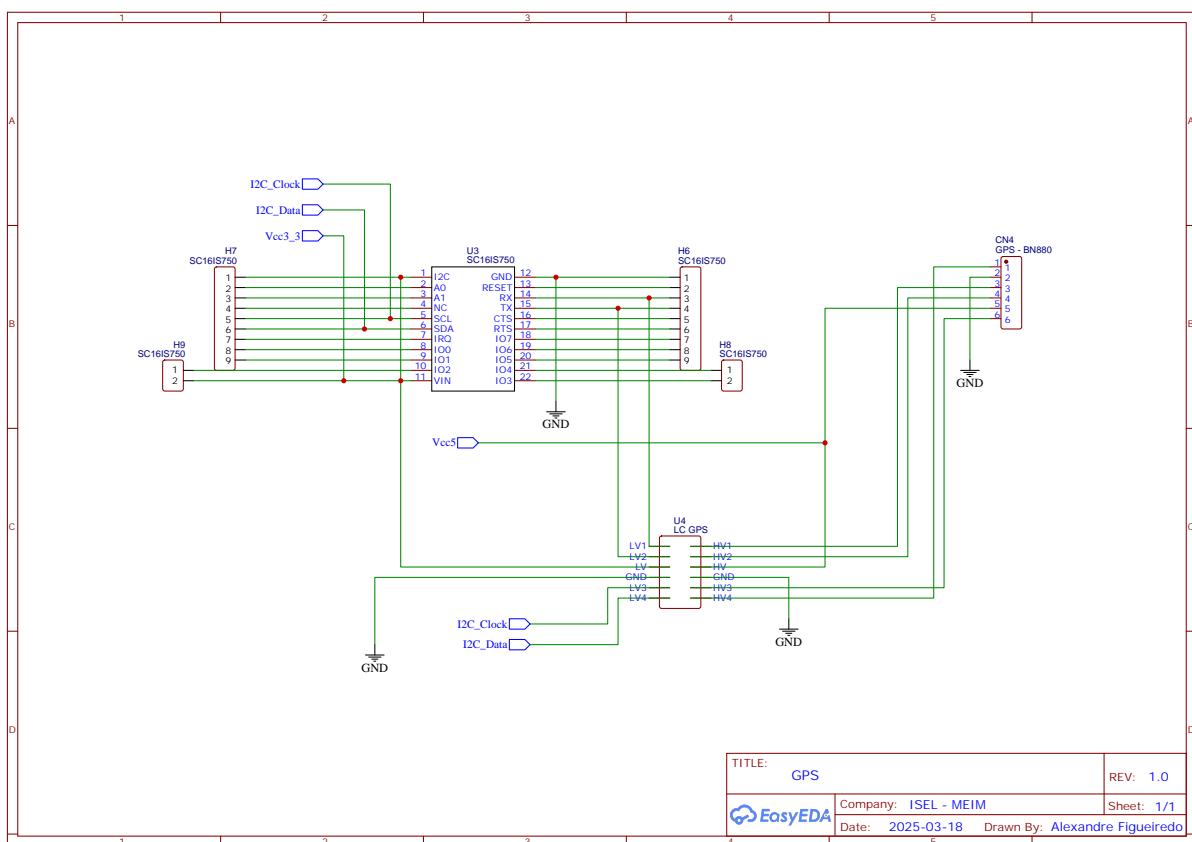
**Figura A.2** Esquemática dos propulsores



**Figura A.3** Esquemática do IMU



**Figura A.4** Esquemática do controlo dos motores (expansor)

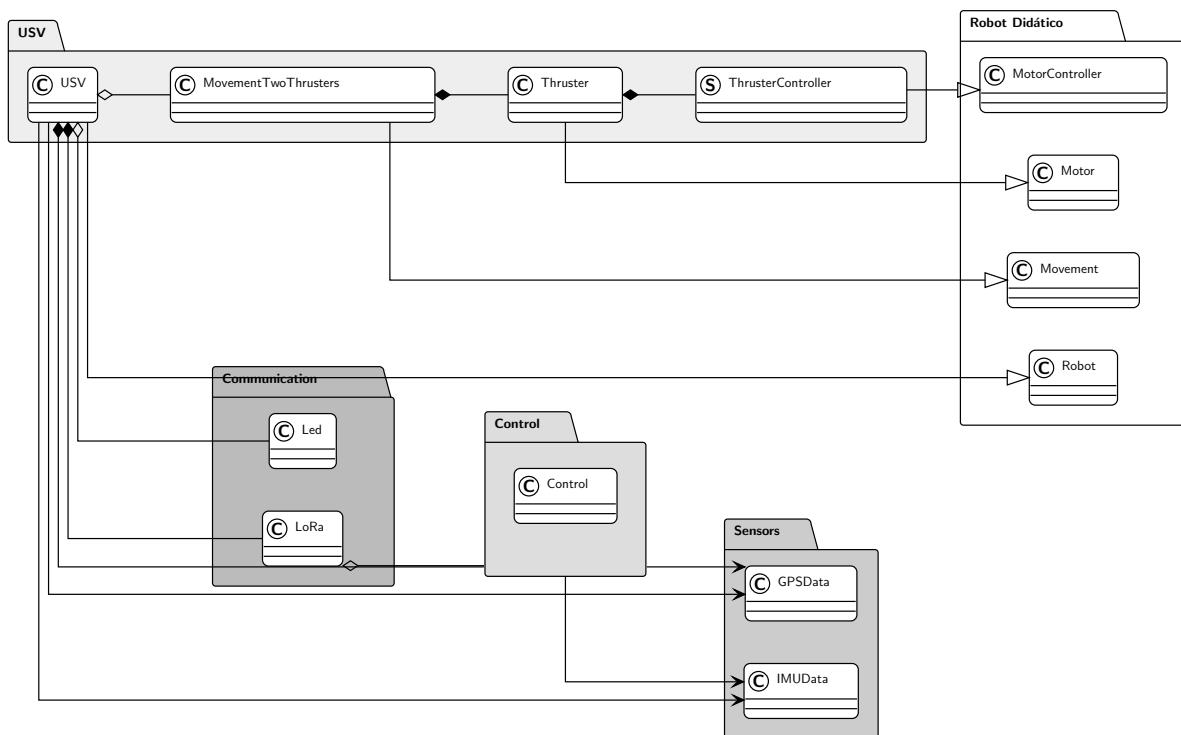


**Figura A.5** Esquemática do GPS

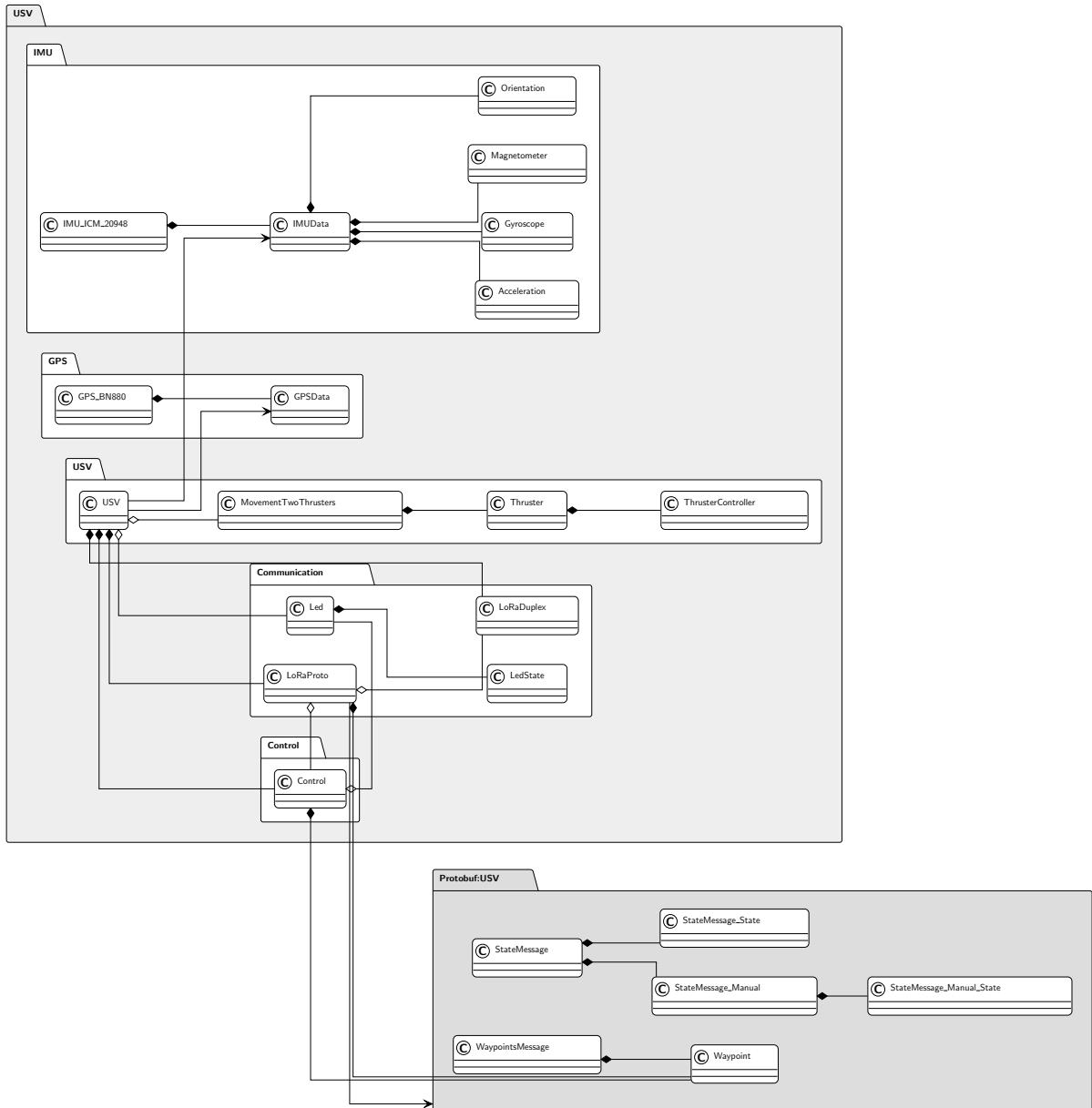


## Anexo B

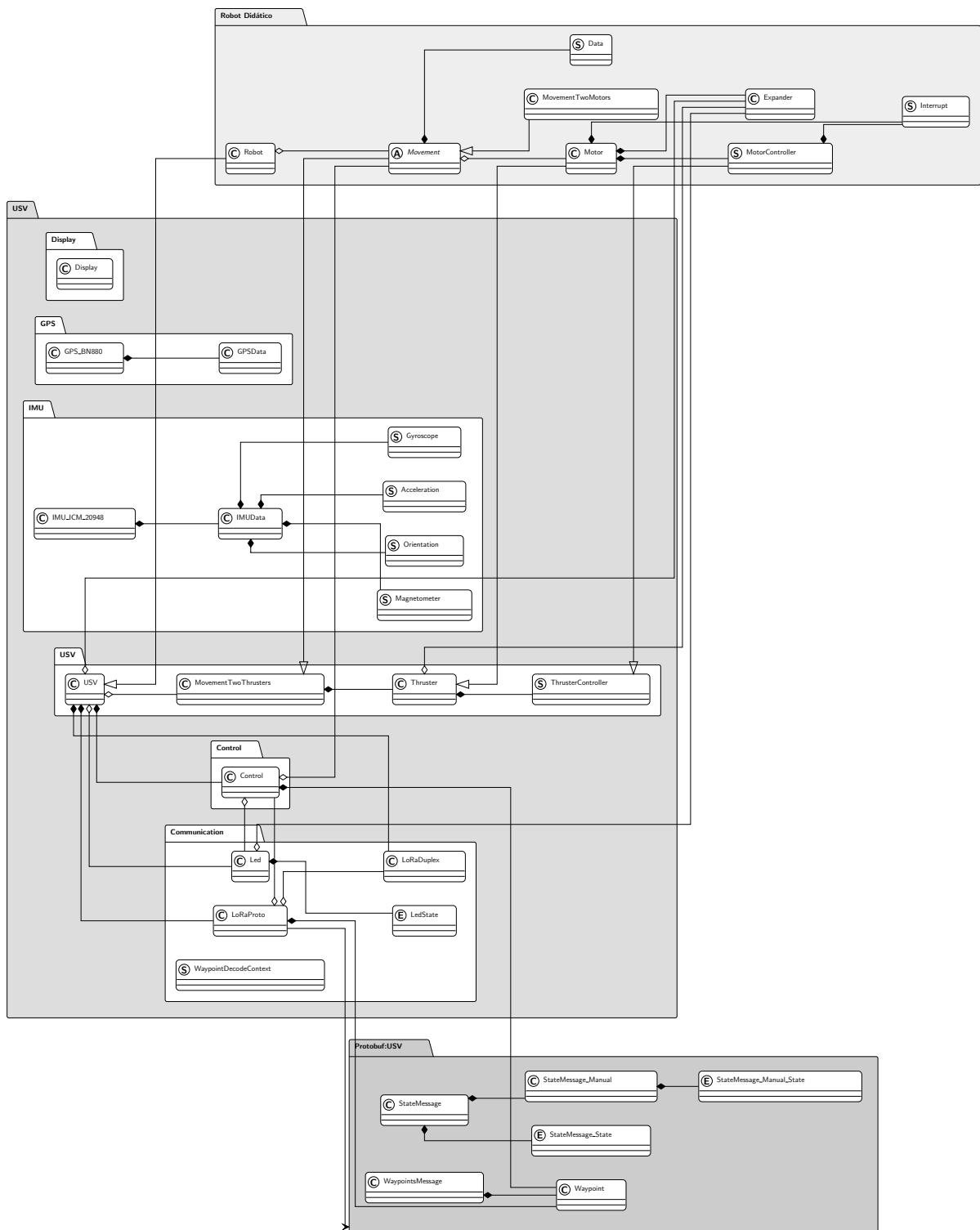
### Diagramas de classes



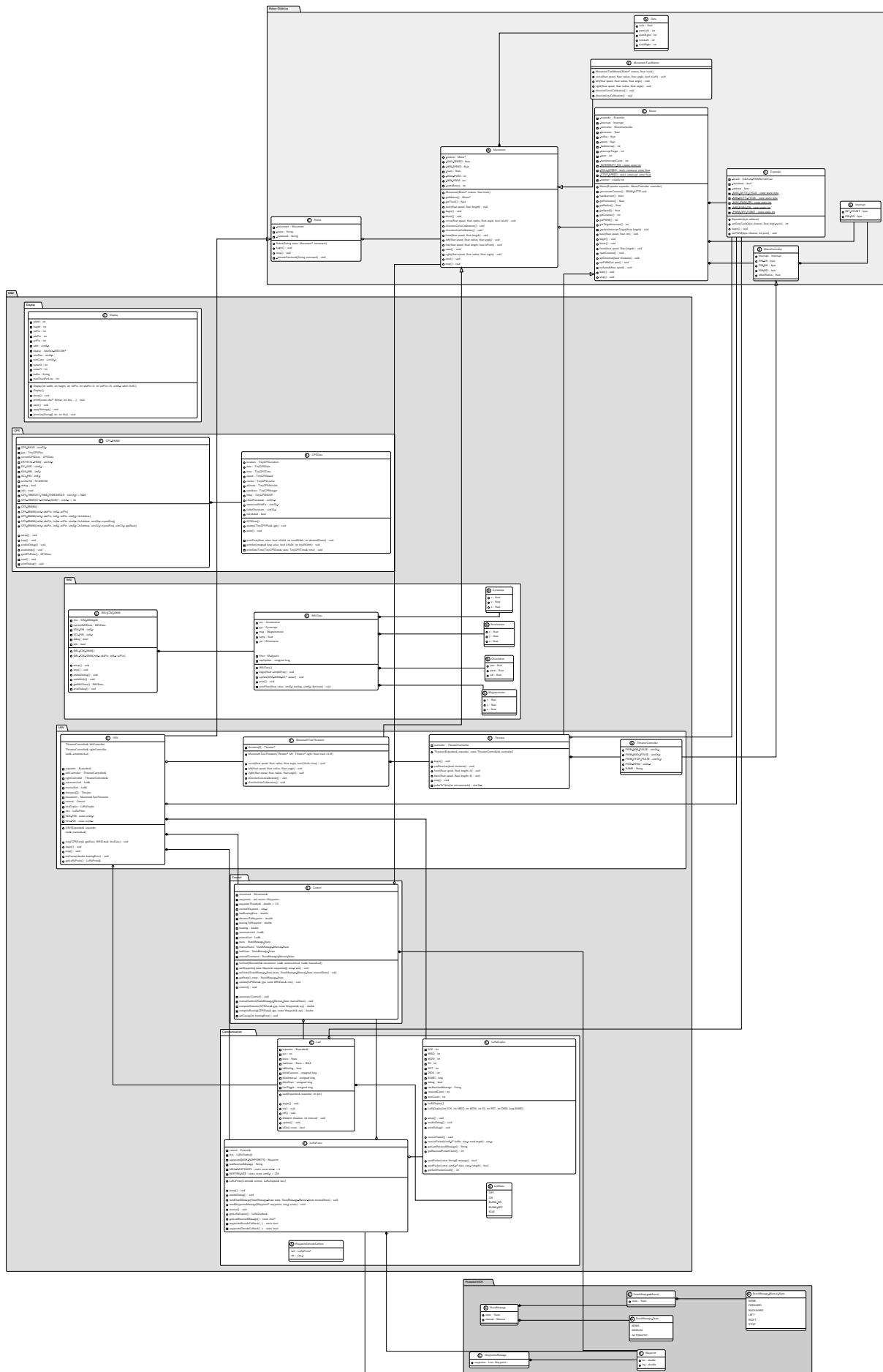
**Figura B.1** Diagrama de classes - Versão simplificada



**Figura B.2** Diagrama de classes - Versão intermédia I



**Figura B.3** Diagrama de classes - Versão intermédia II



**Figura B.4** Diagrama de classes - Versão completa

# Bibliografia

- [1] Alexandre Figueiredo e Daniela Gonçalves. Desenvolvimento de robô didático. 2023.
- [2] Alexandre Figueiredo. Usv. <https://github.com/AlexFigas/USV>, 2025. Accessed: 2025-09-16.
- [3] Liquid Robotics. Reimagine ocean monitoring and operations. uncrewed robots powered by nature. <https://www.liquid-robotics.com/wave-glider/overview/>, 2024. Accessed: 2024-11-04.
- [4] Saildrone. The world's most capable usvs. <https://www.saildrone.com/technology/vehicles>, 2025. Accessed: 2025-09-27.
- [5] SeaRobotics. Sr surveyor class intelligent, integrated and portable. <https://www.searobotics.com/products/autonomous-surface-vehicles/sr-surveyor-class>, 2024. Accessed: 2024-10-22.
- [6] SeaRobotics. <https://www.searobotics.com/images/products/asvs/sr-surveyorm18-truck-bed-700x490.jpg>, 2024. Accessed: 2024-11-04.
- [7] L3Harris Technologies. C-cat 3 asv. <https://www.l3harris.com/all-capabilities/c-cat-3-asv>, 2024. Accessed: 2024-10-12.
- [8] Sea2Future. <https://sea2future.pt/index.html>, 2024. Accessed: 2024-10-16.
- [9] Escola Superior Náutica Infante D. Henrique. Sea2future. <https://www.enautica.pt/pt/projetos-7/projetos-id-186/sea2future/sea2future-1072>, 2024. Accessed: 2024-10-16.
- [10] Ruben Reis e Gabriel Veiga Joaquim Pereira, Mateus Russo. Catamarã telecomandado. 2025.
- [11] APISQUEEN. U01 12v 16v 2kg thrust brushless underwater thruster/propeller/propulsion with bi-directional control esc for rov boat. <https://www.underwaterthruster.com/products/u01-12v-16v-200w-2kg-thrust-brushless-underwater-subsea-thruster-propeller-propulsion-w>, 2025. Accessed: 2025-09-13.
- [12] Minn Kota. Datasheet endura c2. <https://productimageserver.com/literature/ownersManual/397360M.pdf>, 2025. Accessed: 2025-01-02.

- [13] MechanicalElements. Does your trailer have attitude? <https://mechanicalelements.com/trailer-attitude-pitch-yaw-roll/>, 2024. Accessed: 2024-12-02.
- [14] DDKNav. Gnss notes. <https://www.ddknav.com/help-gps/>, 2024. Accessed: 2024-12-02.
- [15] Wikipédia. Long term evolution. [https://pt.wikipedia.org/wiki/Long\\_Term\\_Evolution](https://pt.wikipedia.org/wiki/Long_Term_Evolution), 2025. Accessed: 2025-09-27.
- [16] Wikipédia. 4g. <https://pt.wikipedia.org/wiki/4G>, 2025. Accessed: 2025-09-27.
- [17] Bivocom. Will lora replace 4g lte in iot? <https://www.bivocom.com/industry-trend/will-lora-replace-4g-lte-in-iot>, 2024. Accessed: 2025-09-27.
- [18] Digi International. Digi xbee® rf modules. <https://www.digi.com/products/embedded-systems/digi-xbee>, 2025. Accessed: 2025-09-27.
- [19] Digi International. Digi xbee® zigbee — outdoor / line-of-sight range and power specs. <https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee-zigbee>, 2025. Accessed: 2025-09-27.
- [20] Digi International. Xbee family features comparison — range, power, data rate. [https://media.digikey.com/pdf/other%20related%20documents/digi%20international%20other%20doc/xbee\\_rf\\_features\\_chart.pdf](https://media.digikey.com/pdf/other%20related%20documents/digi%20international%20other%20doc/xbee_rf_features_chart.pdf), 2025. Accessed: 2025-09-27.
- [21] Pplware. Recorde: Comunicação com lora chega aos 1336 km (portugal). <https://pplware.sapo.pt/high-tech/recorde-comunicacao-com-lora-chega-aos-1336-km-portugal/>, 2024. Accessed: 2024-12-10.
- [22] Embedded Computing Design. Basics of 6dof and 9dof sensor fusion. <https://embeddedcomputing.com/technology/analog-and-power/basics-of-6dof-and-9dof-sensor-fusion>, 2025. Accessed: 2025-09-21.
- [23] GPS World. What exactly is gps nmea data? <https://www.gpsworld.com/what-exactly-is-gps-nmea-data/>, 2025. Accessed: 2025-09-21.
- [24] AHRS Documentation. Madgwick filter. <https://ahrs.readthedocs.io/en/latest/filters/madgwick.html>, 2025. Accessed: 2025-09-28.
- [25] Google. Protocol buffers documentation. <https://protobuf.dev/>, 2025. Accessed: 2025-09-28.
- [26] The Things Network. Lora physical layer packet format. <https://www.thethingsnetwork.org/docs lorawan/lora-phy-format/>, 2025. Accessed: 2025-09-28.
- [27] Mikal Hart. Tinygpsplus: A new, improved gps library for arduino. <https://github.com/mikalhart/TinyGPSPlus>, 2025. Accessed: 2025-09-28.
- [28] IDD Portugal. Exercício repmus25. <https://www.iddportugal.pt/agenda/exercicio-repmus25/>, 2025. Accessed: 2025-09-24.

- [29] SAPO. Escola náutica reforça presença portuguesa no maior exercício mundial de robótica marítima. <https://sapo.pt/artigo/escola-nautica-reforca-presenca-portuguesa-no-maior-exercicio-mundial-de-robotica-maritima-2025>. Accessed: 2025-09-24.
- [30] Escola Naval. Naval rex(up) – projeto de investigação integrado no exercício repmus25. [https://escolanaval.marinha.pt/pt/investigacao\\_web/Paginas/Naval-REX\(UP\).aspx](https://escolanaval.marinha.pt/pt/investigacao_web/Paginas/Naval-REX(UP).aspx), 2025. Accessed: 2025-09-24.