UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

ALYSSON RODRIGO DE OLIVEIRA

DESENVOLVIMENTO DE UMA BIBLIOTECA EM PYTHON PARA CONTROLE DO BLUEROV2 VIA PROTOCOLO MAVLINK

Joinville
2024

ALYSSON RODRIGO DE OLIVEIRA

DESENVOLVIMENTO DE UMA BIBLIOTECA EM PYTHON PARA CONTROLE DO BLUEROV2 VIA PROTOCOLO MAVLINK

Trabalho apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica, no Curso de Engenharia Mecatrônica, do Centro Tecnológico de Joinville, da Universidade Federal de Santa Catarina.

Orientador: Dr. Roberto Simoni

Coorientador: Patrick José Pereira, Bacharel em Engenharia Eletrônica

Joinville

2024

AGRADECIMENTOS

Agradeço a Deus, pelo sustento durante esta jornada.
Aos meus pais, Edelson e Miriam, pelo apoio em todos os momentos.
Ao meu orientador, Professor Roberto Simoni, pela orientação e paciência.
Ao coorientador Patrick Pereira, da Blue Robotics e graduado pela UFSC, pela contribuição essencial ao trabalho.
Aos meus amigos Edilânia, Joener e Maíta, pelo suporte significativo durante o trabalho.
A todos os amigos e colegas que fizeram parte da minha trajetória na faculdade.

“Conhecemos melhor a superfície de Marte do que as profundezas dos nossos próprios oceanos.”
— Frase amplamente discutida por cientistas e exploradores.

RESUMO

Este trabalho tem como objetivo desenvolver uma biblioteca de funções em Python para controle do veículo subaquático BlueROV2, utilizando o protocolo MAVLink. O problema central abordado está relacionado aos desafios de comunicação e controle em ROVs, especialmente no uso do protocolo MAVLink. A metodologia incluiu o estudo da arquitetura de hardware e software do BlueROV2, simulações para validação de comandos MAVLink e testes laboratoriais com coleta de telemetria e detecção de marcadores visuais Aruco. Os resultados demonstram a viabilidade da biblioteca desenvolvida para aprimorar o controle e a eficácia operacional do BlueROV2 em missões de coleta de dados e exploração subaquática.

Palavras-chave: BlueROV2. MAVLink. Veículos subaquáticos. Controle operacional. Pymavlink.

ABSTRACT

This work aims to develop a Python function library for controlling the underwater vehicle BlueROV2 using the MAVLink protocol. The central problem addressed involves the challenges of communication and control in ROVs, particularly in utilizing the MAVLink protocol. The methodology included studying the hardware and software architecture of the BlueROV2, performing simulations to validate MAVLink commands, and conducting laboratory tests for telemetry collection and Aruco marker detection. The results demonstrate the feasibility of the developed library to enhance the control and operational effectiveness of the BlueROV2 in data collection and underwater exploration missions.

Keywords: BlueROV2. MAVLink. Underwater vehicles. Operational control. Pymavlink.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACK	Acknowledgment – Confirmação de recebimento em comunicação.
AUV	Autonomous Underwater Vehicle – Veículo subaquático autônomo.
CV	Computer Vision – Visão computacional.
CURV	Cable-Controlled Underwater Recovery Vehicle – Veículo subaquático controlado por cabo para recuperação.
DVL	Doppler Velocity Log – Log de velocidade por efeito Doppler.
ESC	Electronic Speed Controller – Controlador eletrônico de velocidade.
GCS	Ground Control Station – Estação de controle terrestre.
GPS	Global Positioning System – Sistema de Posicionamento Global.
HD	High Definition – Alta definição.
HITL	Hardware In The Loop – Hardware no circuito de simulação.
IMU	Inertial Measurement Unit – Unidade de Medição Inercial.
IP	Internet Protocol – Protocolo de Internet.
JSON	JavaScript Object Notation – Notação de Objetos JavaScript.
MAV	Micro Air Vehicle – Veículo aéreo micro.
MAVLink	Micro Air Vehicle Link – Protocolo de comunicação para veículos aéreos e aquáticos.
OS	Operating System – Sistema operacional.
PID	Proportional-Integral-Derivative – Controlador Proporcional-Integral-Derivativo.
QGC	QGroundControl – Estação de controle e monitoramento.
ROV	Remotely Operated Vehicle – Veículo operado remotamente.
ROS	Robot Operating System – Sistema Operacional de Robôs.
SITL	Software In The Loop – Simulação de software no circuito.
SSH	Secure Shell – Protocolo de rede para acesso remoto seguro.

TCP	Transmission Control Protocol – Protocolo de Controle de Transmissão.
UDP	User Datagram Protocol – Protocolo de Datagrama de Usuário.
UMV	Unmanned Marine Vehicle – Veículo marítimo não tripulado.
USB	Universal Serial Bus – Barramento Serial Universal.
USV	Unmanned Surface Vehicle – Veículo de superfície não tripulado.
UUV	Unmanned Underwater Vehicle – Veículo subaquático não tripulado.
UV	Unmanned Vehicle – Veículo não tripulado.
VLC	VideoLAN Client – Software multimídia.
XML	Extensible Markup Language – Linguagem de Marcação Extensível.

SUMÁRIO

1 INTRODUÇÃO

Veículos Remotamente Operados (ROVs) e Veículos Autônomos Subaquáticos (AUVs) têm revolucionado nossa habilidade de realizar missões em ambientes de difícil acesso ao ser humano. As possibilidades de aplicação crescem dia a dia e englobam áreas como militar, indústria, pesquisa científica, exploração e até mesmo lazer. A crescente demanda por tecnologias que operem em ambientes desafiadores, como materiais resistentes a grandes pressões e sensores capazes de coletar dados de forma confiável, torna os ROVs e AUVs subaquáticos um tema interessante para pesquisa e desenvolvimento (WYNN et al., 2014).

Um ROV é comumente conectado por cabo, pelo qual é controlado e monitorado. Este método é prático e de baixo custo, porém apresenta limitações como comprimento limitado dos cabos e risco de emaranhamento, o que pode dificultar operações em ambientes complexos. O protocolo de comunicação utilizado também influencia a taxa de transmissão, podendo ser relativamente baixa, como no caso do RS-232, ou mais alta com o uso de ethernet, que suporta distâncias maiores e taxas de dados mais rápidas.

A utilização de algum tipo de comunicação sem fio torna-se interessante para superar essas limitações físicas dos cabos. Entretanto, a comunicação sem fio enfrenta desafios devido à alta absorção de ondas de rádio e à propagação lenta de ondas acústicas, o que dificulta uma comunicação confiável no ambiente aquático. Além disso, tecnologias alternativas, como a comunicação acústica e óptica, costumam ser caras, aumentando os custos de desenvolvimento e operação (XUAN et al., 2022).

O BlueROV2, utilizado neste trabalho, possui seis propulsores que proporcionam excelente manobrabilidade e estabilidade em ambientes subaquáticos, além de uma câmera de alta definição e diversos sensores para captura de imagens e coleta de dados (Blue Robotics, 2022). Com o objetivo de enfrentar os problemas relacionados ao controle e à comunicação dos ROVs, este trabalho visa adicionar funções autônomas ao robô subaquático, permitindo que ele execute tarefas de forma independente e opere como um AUV. A integração de funcionalidades autônomas ao BlueROV2 permitirá aumentar sua eficiência em missões de coleta de dados e navegação, possibilitando a exploração de áreas mais distantes sem a necessidade de intervenção constante do operador.

O ArduPilot é um projeto *open-source* de um sistema de piloto automático para ROVs e AUVs de todos os tipos (ArduPilot Development Team, 2024). Baseado nele, foi desenvolvido, também de maneira aberta, o software ArduSub, voltado para a autonomia e telemetria de veículos subaquáticos, e que é utilizado no BlueROV2

(ArduSub Development Team, 2024). Para implementar funções de movimento autônomo e coleta de dados, será utilizado o protocolo de transmissão MAVLink, que é compatível com o ArduSub e permite a comunicação entre ROVs e estações de comando, facilitando o envio de comandos e a coleta de dados de telemetria (XUAN et al., 2022). O QGroundControl, uma estação de controle também de código aberto, será utilizada para monitorar e controlar ROVs através de mensagens MAVLink, oferecendo uma interface intuitiva para a configuração e visualização das missões.

A ampla gama de aplicações possibilitada por essas tecnologias motiva o desenvolvimento de funções para enviar mensagens do protocolo MAVLink ao BlueROV2, da Blue Robotics, com o intuito de controlá-lo e dar os primeiros passos na exploração de algoritmos de navegação autônoma. Por conta da familiaridade com a linguagem Python, optou-se pelo uso da biblioteca *pymavlink*, implementada nessa linguagem, para trabalhar com o protocolo MAVLink, desenvolvendo funções de movimento, coleta de telemetria e visão computacional.

1.1 Objetivo Geral

Desenvolver uma biblioteca de funções em Python, utilizando *pymavlink* para enviar comandos via protocolo MAVLink, permitindo controlar o BlueROV2.

1.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Mapear e selecionar os comandos do protocolo MAVLink relevantes para o controle e monitoramento de ROVs;
- Realizar simulações do BlueROV2 para verificar a comunicação com o QGroundControl via MAVLink;
- Validar, em laboratório, o funcionamento dos comandos MAVLink aplicados ao BlueROV2;
- Coletar e analisar dados de telemetria do BlueROV2 durante os testes com o protocolo MAVLink;
- Capturar imagens da câmera do veículo e desenvolver algoritmos baseados em visão computacional.

1.3 Metodologia de Desenvolvimento do Trabalho

Para o desenvolvimento da biblioteca de controle do BlueROV2, adotou-se uma metodologia composta por etapas de estudo, implementação e validação, englobando simulação e testes em ambiente real. As etapas foram organizadas conforme descrito a seguir.

Para alcançar os objetivos do trabalho, foi essencial compreender a arquitetura de *software* e *hardware* do BlueROV2 da Blue Robotics, o funcionamento de uma GCS (*Ground Control Station*), métodos de comunicação de redes e a estrutura do protocolo MAVLink. Os conhecimentos requeridos para superar as barreiras técnicas iniciais dessas áreas incluem: informática avançada, programação de alto e baixo nível, eletrônica e robótica.

1.3.1 Estudo da Arquitetura de Hardware e Software do BlueROV2 e Ferramentas para Controle

O BlueROV2 da Blue Robotics é descrito pela fabricante como o ROV de alta performance mais acessível do mercado, com aplicações recomendadas em inspeções, pesquisas e atividades de lazer. Suas configurações eletrônicas e de *software* são de código aberto, permitindo diversas possibilidades de expansão e personalização, tanto em *hardware* quanto em *software* (Blue Robotics, 2022).

A montagem e operação do BlueROV2 podem ser realizadas por usuários com conhecimentos básicos de eletrônica para a conexão dos dispositivos e de informática para o uso dos *softwares* de controle, seguindo os tutoriais disponibilizados pela empresa. No entanto, para a personalização das ferramentas e o uso de protocolos como o MAVLink para controle do ROV, é recomendável que o usuário tenha experiência com programação de baixo nível e um conhecimento mais avançado de eletrônica.

O site da Blue Robotics oferece tutoriais detalhados sobre a montagem e configuração do BlueROV2, além de fóruns com interação constante de usuários e suporte técnico da companhia, onde são discutidos temas relacionados a aplicações, personalizações e problemas comuns. Além disso, vários artigos e estudos exploram os possíveis usos do ROV em diferentes áreas. Esses recursos fornecem uma base sólida para iniciar o desenvolvimento de aplicações com o BlueROV2.

1.3.2 Utilização do protocolo MAVLink em simulação e testes em laboratório

Um dos principais objetivos deste trabalho é entender o funcionamento do protocolo MAVLink, um sistema de comunicação leve e eficiente otimizado para MAVs (*micro air vehicles*), e selecionar os principais comandos para controlar e monitorar o BlueROV2. O Capítulo 3 aborda mais detalhadamente a estrutura do protocolo e seu funcionamento.

Para a implementação prática, o primeiro passo foi realizar a simulação do BlueROV2 em um ambiente virtual, com o objetivo de elucidar conceitos como a conexão de rede do veículo com a estação de controle e iniciar os testes utilizando o protocolo MAVLink para coletar dados de telemetria e enviar comandos de movimento ao ROV. O MAVLink possui várias bibliotecas em diversas linguagens; para este

trabalho, foi escolhida a *pymavlink*, desenvolvida em Python, que contém as funções necessárias para implementar os comandos do protocolo.

Após a validação dos testes em ambiente simulado, o trabalho avançará para a execução de testes em laboratório com o veículo físico. Esses testes serão realizados com o ROV em bancada, utilizando os códigos da biblioteca para acioanr os propulsores e obter dados de telemetria do BlueROV2, avaliando a eficácia da implementação.

1.3.3 Lista de Programas e Recursos Utilizados

O desenvolvimento do projeto contou com ferramentas e recursos específicos que foram utilizados tanto na simulação quanto na implementação prática do controle do BlueROV2. Os principais programas e recursos foram listados abaixo:

- BlueSim: ambiente de simulação e modelo virtual do BlueROV2, fornecido pela Blue Robotics;
- QGroundControl: estação de controle e monitoramento de ROVs e AUVs, para planejamento de missões e visualização de dados de telemetria;
- MAVLink: protocolo de comunicação por mensagem para ROVs, utilizado para envio de comandos e coleta de dados;
- Pymavlink: implementação de funções do protocolo MAVLink em Python;

1.4 Estrutura do Texto

Este trabalho é composto por quatro capítulos principais, além da Introdução.

O segundo capítulo apresenta a revisão bibliográfica, que aborda os principais desafios no desenvolvimento de veículos subaquáticos não tripulados. São explorados os avanços tecnológicos, a competição RoboSub e as especificações do BlueROV2, além de uma análise dos métodos de comunicação e controle para esses veículos.

O terceiro capítulo descreve a fundamentação teórica, detalhando os princípios dos veículos subaquáticos operados remotamente (ROVs) e autônomos (AUVs), incluindo suas classificações, componentes e a arquitetura de hardware e software do BlueROV2.

O quarto capítulo discute a implementação prática do trabalho, abrangendo a utilização do protocolo MAVLink para simulação, os testes laboratoriais e a coleta de dados com o BlueROV2. São apresentados os resultados das simulações realizadas e dos experimentos em ambiente controlado.

Por fim, o último capítulo apresenta as Considerações Finais e Trabalhos Futuros, destacando as contribuições acadêmicas deste trabalho e as principais dificuldades enfrentadas durante seu desenvolvimento. Além disso, são discutidos os pontos a serem considerados em pesquisas futuras. Em seguida, são listadas as referências utilizadas no desenvolvimento deste trabalho.

2 REVISÃO BIBLIOGRÁFICA

Considerando os objetivos deste trabalho, este capítulo apresentará uma contextualização dos projetos envolvendo desenvolvimento e aplicação de ROVs e AUVs, detalhes e cenários de uso do veículo subaquátrico BlueROV2 e o protocolo MAVLink.

2.1 Desafios no Desenvolvimento de Veículos Subaquáticos Não-Tripulados

Vários estudos abordam as dificuldades encontradas no projeto e desenvolvimento de veículos autônomos subaquáticos, destacando a necessidade de avanços tecnológicos para aprimorar a eficácia da navegação e controle desses veículos. Essas dificuldades incluem desde problemas de comunicação em ambientes submersos até desafios de estabilidade e manobrabilidade em águas turbulentas e sob alta pressão. Artigos como os de Bae e Hong (2023) e Wibisono et al. (2023) abordam o estado da arte dos veículos aquáticos autônomos e controlados, e discutem o que vem sendo desenvolvido na área, bem como as tendências futuras para este campo.

É muito comum esses e outros estudos utilizarem uma variedade de siglas para se referirem aos diferentes tipos de veículos subaquáticos e de superfície. A Tabela 1 resume os principais tipos abordados na literatura no que se refere ao campo de interesse estudado neste trabalho, facilitando a compreensão das siglas e suas respectivas funções.

Tabela 1 – Principais Siglas para Veículos Autônomos e Não Tripulados Marinhos

Sigla	Nome Completo	Descrição Básica
UV	Unmanned Vehicle	Termo genérico para veículos não tripulados.
ROV	Remotely Operated Vehicle	Veículo subaquático operado remotamente.
AUV	Autonomous Underwater Vehicle	Veículo subaquático capaz de navegar de forma independente.
UUV	Unmanned Underwater Vehicle	Termo geral utilizado para UVs subaquáticos.
UMV	Unmanned Marine Vehicle	UV que opera em ambientes marinhos, pode ser subaquático ou de superfície.
USV	Unmanned Surface Vehicle	Veículo de superfície não tripulado, podendo ser operado remotamente ou autonomamente.

Fonte: Adaptado de Bae e Hong (2023) e Wibisono et al. (2023).

ROVs e AUVs são frequentemente utilizados para exploração do oceano, detecção de minas submarinas e vigilância militar (WIBISONO et al., 2023). Além dos desafios técnicos, essas aplicações apresentam barreiras significativas devido aos ambientes adversos, que incluem a presença de correntes marítimas, alta pressão nas profundidades, corrosividade da água do mar e limitações de comunicação por sinais de rádio ou GPS (BAE; HONG, 2023). O estudo de Wibisono et al. (2023) também destaca a necessidade do desenvolvimento das tecnologias que dão suporte à navegação deste tipo veículo, uma vez que perder o veículo durante a execução de uma missão também é uma possibilidade real.

Nicholson e Healey (2008) apresenta em seu estudo as tecnologias que compõe um AUVs e ainda ressalta que "as capacidades atuais dos AUVs são limitadas pelo *design* e a maturidade das várias tecnologias que equipam o veículo". Entre essas tecnologias, destacam-se:

- *Design* Modular: ser reconfigurável traz a vantagem de fazer alterações de custo mais baixo quando novas tecnologias de *software* e *hardware* surgem ou quando missões requerem uma mudança específica no veículo, como a inclusão de novos sensores, por exemplo. O uso de baterias modulares também reduzem o tempo que o UV fica parado.
- Navegação: sonares, sensores inerciais, DVL (*Doppler Velocity Log*) e UGPS (*Underwater GPS)*[1] são geralmente utilizados para obter informações de posição de UVs, visto que "o ambiente de operação subaquático não permite acesso contínuo à sinais GPS."
- Comunicação: enquanto submergido, o AUV tem sua comunicação limitada e deve vir à suberfície para atualizações de missão. A água impõe uma barreira natural na comunicação, deixando o AUV inoperável e as oportunidades de comunicação com a central de comando são determinadas pelo veículo.
- Alimentação: baterias de lítio, recarregáveis ou substituíveis, são comumente empregadas em AUVs, o que limita a duração da missão autônoma devido à descarga da bateria.
- Sensores: uma vasta gama de sensores para oceanografia pode ser aplicada em AUVs devido aos seus tamanhos geralmente compactos. Câmeras de alta resolução, sonares para navegação, magnetômetros e sensores químicos e biológicos são exemplos de instrumentos utilizados em AUVs.
- Autonomia: um exemplo mais básico deste componente consiste no controle da posição, velocidade e altitude do veículo e a determinação de *waypoints*. Em um próximo nível, o veículo pode vir a detectar e desviar de obstáculos. Esses níveis progressivos de autonomia permitem que o AUV tome decisões mais complexas e independentes durante as missões.

[1] Mais informações: https://waterlinked.com/underwater-gps

O estudo de Nicholson e Healey (2008) também destaca que o avanço dos veículos autônomos subaquáticos depende do desenvolvimento de melhores soluções para alimentação e autonomia. A criação de baterias mais duráveis ou métodos de carregamento durante a operação possibilitaria missões de maior duração. Além disso, veículos mais autônomos e independentes de intervenção humana tendem a ser mais eficientes em suas tarefas e correm menos riscos de danos ou naufrágios, ao passo que coletam dados mais confiáveis e relevantes para suas missões.

2.2 RoboSub - Competição de AUVs

Um dos principais fomentadores do desenvolvimento no campo dos veículos autônomos subaquáticos são as competições estudantis. O RoboSub é uma competição internacional de estudantes, iniciada em 1998, que visa cultivar e incentivar contribuições para esta área por meio de prêmios e eventos anuais. A competição permite que os participantes projetem, construam e testem sistemas completamente autônomos, adquirindo experiência em engenharia e trabalho em equipe (RoboSub, 2023).

A equipe de competição *Team Inspiration*, responsável pelo AUV nomeado *Græy and Orange*, exibido na Figura 1, foi uma das participantes do RoboSub em 2020. A equipe desenvolveu o projeto mecânico do veículo, feito de alumínio extrudado, de forma a permitir a acomodação e o rearranjo dos sensores para diferentes tarefas. O projeto elétrico foi realizado com o objetivo de otimizar o desempenho com base em competições passadas. Os algoritmos de navegação do veículo foram desenvolvidos com ROS (*Robot Operating System*) (ROS Blog, 2023) e MAVROS (um conjunto de ferramentas de *middleware* que facilita a comunicação entre o MAVLink e o ROS) (ROS Wiki, 2023) para controlar o veículo, além de uma gama de sensores de velocidade e posição, como o DVL (*Doppler Velocity Logger*), CV (*Computer Vision*) e hidrofones, uma espécie de sonar. Além disso, a equipe também implementou uma rede neural para reconhecimento de objetos subaquáticos (SZETO et al., 2020).

Em 2022, a equipe *NYUAD Robosub* participou da competição utilizando o BlueROV2 da Blue Robotics. O time utilizou ROS para coletar dados de telemetria e realizar ecolocalização, aproveitando a característica de customização do ArduSub, software nativo do AUV, para implementar algoritmos de navegação autônoma com filtros de Kalman e controladores PID para velocidade e posição. Em combinação com o ROS, a equipe utilizou a biblioteca Python *pymavlink* para implementar a autonomia do veiculo e planejar missões. Eles obtiveram êxito no treinamento de uma rede neural para reconhecer objetos durante as missões com o veículo. Um diferencial desta equipe foi utilizar comunicação sem fio entre as diferentes placas de controle do veículo (Raspberry Pi, PixHawk e Jetson TX2) para a troca de informações entre si (NHAM et

Figura 1 – Græy and Orange: AUV da Team Inspiration

Fonte: (SZETO et al., 2020)

al., 2022). A versão do ROV BlueROV2 adaptado pela equipe pode ser vizualizada na Figura 2.

Figura 2 – BlueROV2 adaptado pela NYUAD Robosub

Fonte: (NHAM et al., 2022)

2.3 BlueROV2: Especificações e Aplicações

O BlueROV2 é um veículo subaquático lançado em 2016 pela Blue Robotics, uma empresa sediada na Califórnia, EUA. Ele é apresentado pela empresa como o ROV de alta performance mais acessível do mercado e pode ser adquirido diretamente no site da companhia. Suas principais aplicações incluem inspeções marinhas, pesquisas científicas e atividades de lazer (Blue Robotics, 2022). Esta seção tem como objetivo apresentar as principais especificações do BlueROV2, veículo alvo de estudo neste trabalho e exibido na Figura 3.

O BlueROV2 é equipado com 6 propulsores T200, que proporcionam estabilidade e manobabilidade ao veículo. Patentado pela Blue Robotics, este propulsor possui um *design* inovador e é composto por um motor *brushless* trifásico do tipo *outrunner*, totalmente inundado com enrolamentos encapsulados e estator, além de ímãs e rotor revestidos. Seu corpo é feito de plástico policarbonato resistente e aço inoxidável marinho 316. Essa configuração permite que o motor seja resfriado e as buchas de plástico sejam lubrificadas pela água, eliminando a necessidade de selos de eixo, acoplamentos magnéticos e compartimentos cheios de ar ou óleo, o que torna o propulsor naturalmente resistente à pressão. O propulsor opera em conjunto com um controlador de velocidade, o Basic ESC, e é alimentado por baterias de 16 volts (Blue Robotics, 2024c).

Figura 3 – BlueROV2

Fonte: (Blue Robotics, 2024c)

O ROV da Blue Robotics possui um *design* modular, contando com seis penetradores de cabos livres, permitindo a adição de sensores e equipamentos para personalização do veículo conforme as necessidades da missão. Essa flexibilidade é um dos fatores que o tornam atraente para pesquisadores e profissionais que necessitam de um ROV adaptável a diferentes condições e objetivos de trabalho. Ele também possui uma câmera com transmissão de vídeo ao vivo em 1080p HD e

iluminação de intensidade ajustável. Dependendo da versão, o BlueROV2 pode operar a profundidades de até 100 metros ou até 300 metros.

Equipado com o software de código aberto ArduSub e o piloto automático Navigator, o BlueROV2 oferece capacidades de automação que o destacam entre os ROVs de sua categoria. Além disso, ele possui uma série de sensores integrados, como giroscópio, acelerômetro, magnetômetro e sensores de pressão e temperatura, que auxiliam no controle de estabilidade e na coleta de dados ambientais. Essas funcionalidades expandem o potencial do BlueROV2 para operações subaquáticas complexas, permitindo o desenvolvimento de funções de navegação autônoma e a realização de missões de monitoramento em ambientes dinâmicos.

2.4 Comunicação e Controle para Veículos Subaquáticos Remotamente Operados

O artigo de Xuan et al. (2022) apresenta uma pesquisa desenvolvida em 2022 que propõe o desenvolvimento de técnicas de comunicação e controle para ROVs. O veículo utilizado no estudo tem formato de torpedo e se comunica com uma GCS (*Ground Control Station*), localizada em solo ou embarcada, utilizando uma boia que serve como ponto de acesso para o sinal sem fio, conectando a GCS ao ROV. A boia, conectada ao veículo via cabo ethernet, oferece alta taxa de transmissão e suporte para longas distâncias, além de garantir uma comunicação mais estável e de baixo custo ao combinar o sinal sem fio com o uso de cabo para comunicação direta.

Para a comunicação entre a estação de comando e o ROV, os autores utilizaram o protocolo MAVLink, desenvolvido para possibilitar tanto comunicações ponto a ponto quanto em *multicast*. Equipado com um sistema de dupla verificação (*checksum*), o protocolo assegura a confiabilidade e integridade das mensagens trocadas, especialmente importantes em ambientes subaquáticos onde a comunicação é mais suscetível a falhas (XUAN et al., 2022).

Os autores desenvolveram comandos com o protocolo para coletar dados, como estado dos sensores, nível das baterias e informações do sistema, incluindo ângulo e velocidade de movimento, além de enviar comandos de movimento e retorno à superfície em caso de emergência. As mensagens MAVLink são definidas utilizando arquivos XML (*Extensible Markup Language*), que permitem a geração de código em C++, Python ou outras linguagens de programação. Essas mensagens são utilizadas em sistemas embarcados, como a GCS, o computador auxiliar com Linux e o microcontrolador do ROV. Além disso, as mensagens MAVLink enviam respostas de reconhecimento, conhecidas como ACK (*Acknowledgement*), para informar aos usuários se um comando foi executado com sucesso, está em execução ou falhou.

3 FUNDAMENTAÇÃO TEÓRICA

Diante do cenário apresentado na revisão bibliográfica, o desenvolvimento de veículos subaquáticos autônomos e o uso de protocolos de comunicação avançados, como o MAVLink, demonstram grande potencial para superar desafios em ambientes aquáticos. Este capítulo abordará os fundamentos teóricos que sustentam essas tecnologias, explorando aspectos históricos, técnicos e operacionais essenciais para implementar as soluções propostas neste trabalho.

3.1 Fundamentos de Veículos Remotamente Operados e Autônomos Subaquáticos

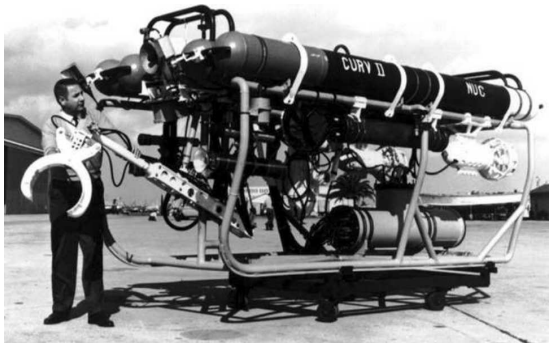
A literatura divide os veículos subaquáticos em tripulados e não tripulados. Dentro dos veículos não tripulados, encontram-se os veículos remotamente operados (ROVs - *Remoted Operated Vehicles*), geralmente conectados via cabo, e os veículos autônomos (AUVs - *Autonomous Underwater Vehicles*). Conforme *The Navy UUV Master Plan* apud Christ e Wernli (2013), um UUV (*Underwater Unmanned Vehicle)* é um submersível com propulsão própria cuja operação é totalmente autônoma (pré-programada ou com controle de missão adaptativo em tempo real) ou realizada com controle supervisório mínimo, e que não possui cabos, exceto cabos de dados. O termo UUV é comumente usado como sinônimo de AUV.

3.1.1 Os primeiros ROVs

A criação do primeiro ROV é atribuída por alguns a Dimitri Rebikoff, em 1953, com um veículo chamado *POODLE*. No entanto, foi a Marinha Americana que iniciou o desenvolvimento de um sistema subaquático completo e operacional. Enquanto o veículo de Rebikoff era utilizado para pesquisas arqueológicas, o problema proposto pela Marinha Americana era a recuperação de torpedos lançados no fundo do mar. A partir de uma parceria público-privada, desenvolveu-se o projeto CURV (*Cable-Controlled Underwater Research Vehicle*) e seus sucessores (CORRÊA, 2021).

Estes modelos vieram a alcançar feitos extraordinários, atraindo atenção global. Em 1966, na costa da Espanha, o CURV recuperou uma bomba atômica a 869 metros de profundidade, o que era além de seu limite operacional. Posteriormente, em 1973, na costa irlandesa, o modelo CURV III conectou um cabo de resgate ao veículo submersível tripulado PISCES III (vide Figura 5), que estava preso a 480 metros de profundidade e com baixo nível de oxigênio. Uma antiga fotografia do seu precursor, o CURV II, pode ser vista na Figura 4. A atuação do ROV foi um sucesso e salvou a vida

Figura 4 – CURV II: precursor CURV III



Fonte: (CHRIST; WERNLI, 2013).

Figura 5 – O resgate da PISCES III

Fonte: (MOORHOUSE, 2015).

dos dois tripulantes a bordo (CHRIST; WERNLI, 2013).

Segundo Christ e Wernli (2013), até meados da década de 1970, a maioria dos ROVs era patrocinada pelo governo. Em 1973, existiam vinte veículos construídos, dos quais apenas três eram de iniciativa privada. A partir do ano seguinte, os avanços tecnológicos necessários para a construção de veículos subaquáticos chegaram ao mercado, especialmente na área de eletrônicos, e grande parte dos UVs passou a ser construída por indústrias privadas. Na década de 1990, os ROVs batiam recordes de profundidade, beneficiando-se dos avanços na engenharia de seus componentes.

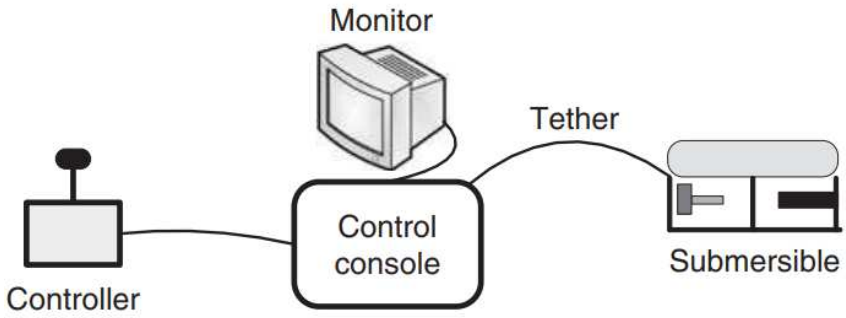
Hoje, estima-se que a superfície de Marte seja mais conhecida que o solo oceânico. Diversos governos realizam operações militares subaquáticas e exploram recursos naturais, enquanto cientistas estudam a fauna e flora marinhas, ainda pouco conhecidas. Isso demonstra que a história dos ROVs e AUVs continua sendo escrita.

3.1.2 Componentes de um ROV

Conforme Christ e Wernli em *The ROV Manual* (2013), um ROV é, por uma perspectiva simplificada, uma câmera acoplada a um invólucro impermeável, equipada com propulsores para manobrabilidade e um cabo de dados para transmissão do sinal de vídeo. Uma representação deste esquema pode ser vista na Figura 6. Nicholson e Healey (2008) exploram em mais detalhes componentes como alimentação, sensores, ferramentas de navegação, autonomia e modularidade. Nesta seção, serão descritos os principais componentes que compõem os veículos subaquáticos.

O invólucro impermeável do ROV permite abrigar com segurança seus componentes eletrônicos, como placas de controle, sensores e baterias. As conexões entre esses dispositivos também devem ser feitas com cabos vedados de forma especial, para evitar a entrada de água no compartimento. É interessante utilizar métodos de vedação que tornem o veículo modular, possibilitando a adição de novos sensores e atuadores, por exemplo.

Figura 6 – Esquema Básico de um ROV

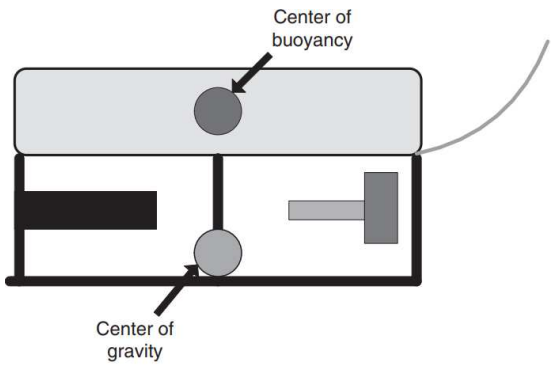


Fonte: (CHRIST; WERNLI, 2013).

Outro componente importante dos ROVs são suas baterias. Esses veículos geralmente utilizam baterias de lítio, recarregáveis ou não. Nicholson e Healey (2008) ressaltam que avanços no desenvolvimento de baterias mais duráveis podem permitir aos ROVs a execução de missões mais longas, e destaca a preferência por substituir baterias ao invés de desativar o veículo para recarga, o que aumenta seu tempo em uso. Uma opção inovadora mencionada em seu artigo é a subida do ROV à superfície para recarregar sua bateria com energia solar, trazendo novas possibilidades em termos de duração das missões e minimizando a necessidade de docagem para recarregamento das baterias.

Os propulsores geralmente consistem em motores elétricos com hélices estrategicamente distribuídas para permitir movimentos precisos em várias direções. Além dos propulsores, o sistema de flutuabilidade é um elemento importante, pois garante que o ROV mantenha uma posição estável e controlada em diferentes profundidades. Geralmente, esses veículos possuem um centro de flutuabilidade elevado e um centro de gravidade mais baixo, proporcionando estabilidade à sua câmera, conforme mostrado na Figura 7. Esse sistema é ajustado com materiais flutuantes ou compartimentos de ar, que ajudam a manter o veículo suspenso na água, evitando que ele afunde ou suba rapidamente de maneira indesejada.

Figura 7 – Centro de flutuabilidade e centro de gravidade de um ROV



Fonte: (CHRIST; WERNLI, 2013).

Outros componentes incluem ferramentas de navegação, como sonares de posicionamento e sensores de profundidade e inclinação, que auxiliam na estabilização e no monitoramento da orientação do veículo. Diversos tipos de sensores são integrados ao ROV para coletar dados ambientais e monitorar seu próprio estado. Sensores de temperatura, campo magnético, pressão e turbidez ajudam a avaliar as condições subaquáticas, enquanto câmeras de alta resolução registram imagens. Um sensor crítico para a segurança do sistema é o sensor de vazamento, posicionado no interior do invólucro impermeável para detectar qualquer entrada de água, protegendo os componentes eletrônicos e garantindo a durabilidade do ROV. Esse sensor permite ao ROV alertar o operador sobre um vazamento ou tomar uma ação de segurança de forma autônoma.

3.1.3 Classificação de ROVs

Existem diferentes maneiras de classificar ROVs na literatura. Christ e Wernli (2013) apresentam três classes de veículos baseado em sua funcionalidade: observação, trabalho e uso especial. A **classe de observação** é equipada com câmeras, sensores e ferramentas para coleta de dados, além de instrumentos para carga útil de equipamentos de intervenção ou de apoio à navegação. A **classe de trabalho** possui uma carenagem maior, múltiplos manipuladores, propulsão e atuadores hidráulicos, sendo adequada para lidar com equipamentos e tarefas subaquáticas pesadas. Por fim, a **classe de uso especial** abrange veículos projetados para finalidades específicas, como o enterramento de cabos no solo oceânico para telecomunicações.

O Hellenic Centre for Marine Research (2020) aprensenta cinco classes de ROV baseadas em tamanho e função. A **Classe I** são veículos pequenos equipados apenas com câmeras, luz e sonar. A **Classe II** abrange veículos com capacidade para carga adicional, duas câmeras, sonar e múltiplos sensores. Veículos grandes com diversos sensores e atuadores compõe a **Classe III**. A **Classe IV** é formada por veículos que realizam trabalhos no solo oceânico e se movimentam com rodas ou sistemas de tração. Por último, a **Classe V** inclui veículos que são protótipos ou estão em desenvolvimento.

3.2 Arquitetura de *Hardware* e *Software* do BlueROV2

Conforme descrito pela Blue Robotics (2024b), o BlueROV2 utiliza dois recursos principais: o *Navigator Flight Controller* e o *Companion Computer*. O *Navigator* (Figura 8) é uma placa de expansão HAT (*Hardware Attached on Top*) desenvolvida pela Blue Robotics, que substituiu o *Pixhawk* (Figura 9) e atua em conjunto com microcontrolador Raspberry Pi na versão R4 do veículo. Criado para ROVs e outras aplicações robóticas, o *Navigator* é equipado com uma unidade de medida inercial (IMU) e um magnetômetro,

responsáveis por medir a orientação e o rumo da bússola, além de um sensor de vazamento que alerta sobre qualquer entrada de água.

A placa do *Navigator* também possui dezesseis saídas, que podem ser conectadas a propulsores, sistemas de iluminação, controladores, atuadores e outros acessórios, além de contar com portas de comunicação serial I^2C para integração com diversos sensores. Esse recurso torna o BlueROV2 um veículo customizável e expansível, adequado para uma variedade de aplicações subaquáticas e robóticas.

Figura 8 – Controlador do BlueROV2 a partir de 2022

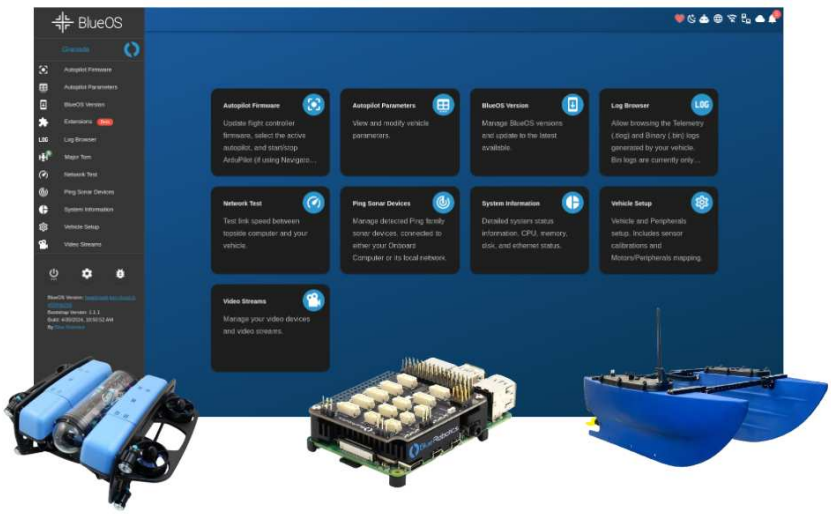
Fonte: (Blue Robotics, 2024b).

Figura 9 – Antigo controlador do BlueROV2

Fonte: (Blue Robotics, 2024b).

O *Companion Computer* é um microcomputador Raspberry Pi 4 que atua como interface entre o *Navigator* e a estação de comando. Ele gerencia vários recursos do ROV, como a execução do ArduSub, o controle de câmeras e a conexão de dados via cabo Ethernet. O *BlueOS*, exibido na Figura 10, uma plataforma *web* executada no *Companion Computer*, foi projetado para proporcionar maior customização ao veículo, permitindo ao usuário adicionar funcionalidades ao BlueROV2 e atualizar seu *firmware*.

Figura 10 – Plataforma *Web* - BlueOS



Fonte: Blue Robotics (2024a).

3.2.1 ArduSub

O projeto ArduPilot, inicialmente desenvolvido para controle de drones aéreos, evoluiu ao longo do tempo para oferecer suporte a outros tipos de veículos remotamente controlados, dando origem a diversos subprojetos, como o ArduRover, para veículos terrestres, o ArduCopter, para multirrotores e helicópteros, o ArduPlane, para veículos de asa fixa, e, por fim, o ArduSub, para veículos subaquáticos (ArduPilot Development Team, 2024).

O ArduSub é um *firmware open-source* derivado do ArduCopter que é executado no Raspberry Pi para controlar o BlueROV2. Ele oferece funcionalidades específicas para ROVs, como *feedback* de controle de estabilidade e controle de profundidade, além de algumas funções compartilhadas com outros tipos de drones, como a coleta de telemetria dos veículos e o planejamento de missão (ArduSub Development Team, 2024).

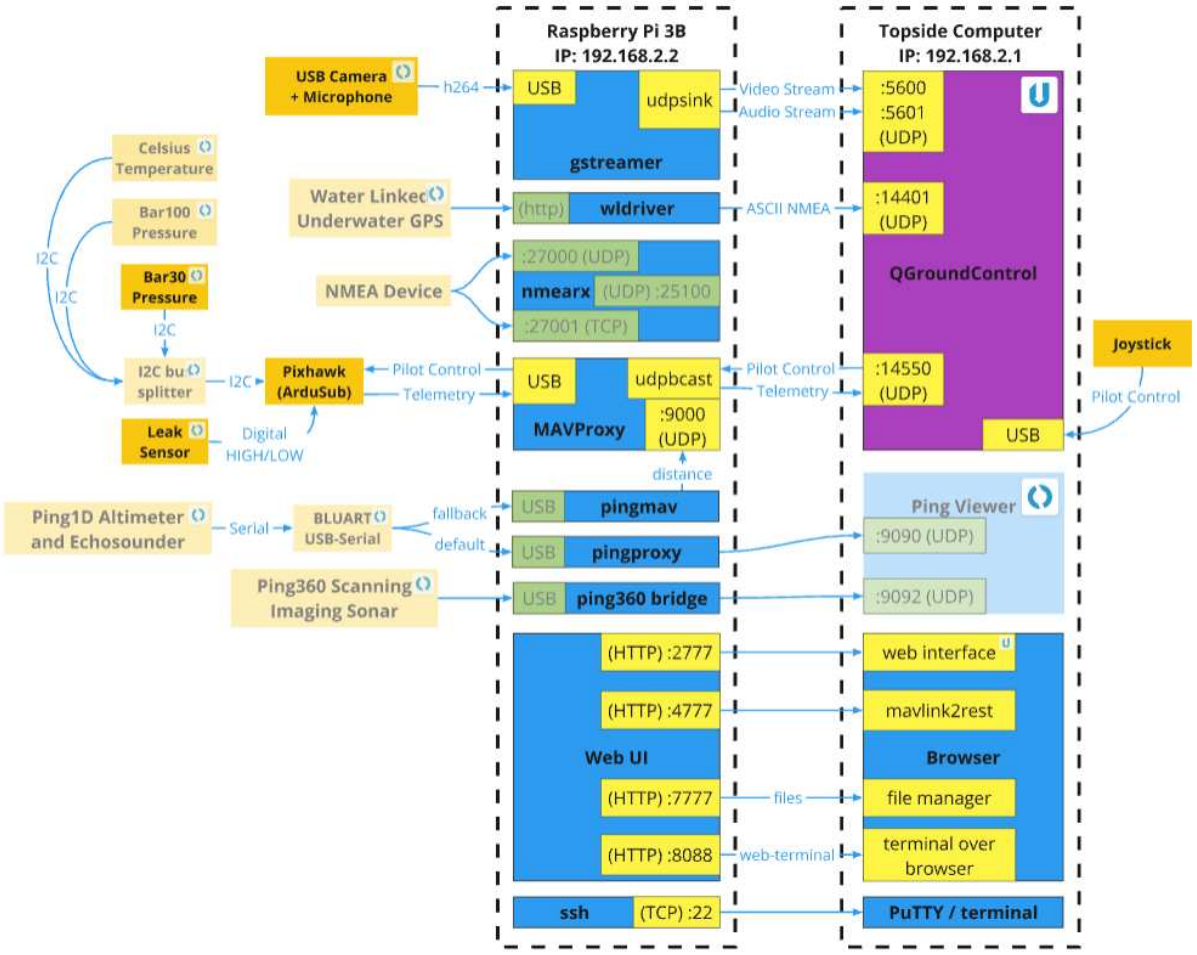
O ArduSub Development Team (2024) apresenta um diagrama, exibido na Figura 11, que detalha a interação entre os componentes *software* e *hardware* da arquitetura do BlueROV2. No diagrama, são exibidas as conexões estabelecidas entre os componentes do sistema, incluindo os protocolos de comunicação e as portas utilizadas para telemetria, controle e transmissão de vídeo. Por exemplo, a porta `14550` é usada para telemetria, enquanto a porta `5600` é responsável pela transmissão de vídeo. Além disso, o endereço IP padrão do BlueROV2 é `192.168.2.2`, exigindo que o computador de controle seja configurado com um endereço IP na faixa `192.168.2.x` e uma máscara de sub-rede `255.255.255.0`. Compreender essas interações foi essencial para o desenvolvimento do trabalho, pois isso permitiu configurar os computadores para que estivessem na mesma rede do veículo, garantir o envio adequado de mensagens do protocolo MAVLink pelas portas designadas e possibilitar a captura de vídeo e telemetria provenientes do ROV.

O entendimento do ArduSub também foi essencial para compreender as funcionalidades suportadas para o controle do BlueROV2 e os protocolos de comunicação compatíveis. Este e outros recursos *open-source* são imensamente valiosos por oferecerem uma grande quantidade de materiais disponíveis *online*, criados por comunidades acadêmicas, científicas e até mesmo privadas, com foco em pesquisa e desenvolvimento. A própria empresa criadora do BlueROV2 patrocinou a criação do projeto ArduSub e a autoria do livro *online* com sua documentação.

A análise da documentação do ArduSub revelou vários aspectos importantes que suportam a transformação do BlueROV2 em um veículo altamente customizável e com possibilidades de operação autônoma. Entre os principais pontos, destacam-se:

- Há suporte para a estrutura vetorizada do BlueROV2, com propulsores dispostos lado a lado e sistema de vetorização de empuxo.

Figura 11 – Diagrama de Comunicação - *Hardware* e *Software*



Fonte: ArduSub Development Team (2024).

- Existe compatibilidade com o protocolo MAVLink, bem como uma documentação estruturada da biblioteca *pymavlink* para ArduSub, permitindo adicionar funcionalidades autônomas aos ROVs e transformá-los em AUVs.
- O ArduSub é compatível e opera de maneira satisfatória com a estação de comando *open-source* QGroundControl.
- O projeto ArduPilot possui um simulador (SITL - *software-in-the-loop*) que permite validar os testes com o veículo antes de testar em campo. Mais informações sobre o SITL podem ser encontradas na Seção 3.5.

Com os conceitos fundamentais do ArduSub estabelecidos, o próximo passo envolve o estudo do protocolo MAVLink e sua aplicação no controle do BlueROV2 para alcançar o objetivo deste trabalho.

3.3 MAVLink: *Micro Air Vehicle Link*

A documentação apresentada pelo ArduSub Development Team (2024) informa que o ArduSub se comunica utilizando um protocolo chamado MAVLink, com o qual é possível coletar informações e comandar o veículo. Esta seção aborda as principais características técnicas do protocolo.

3.3.1 Introdução ao Protocolo MAVLink

MAVLink é a abreviação de *Micro Aerial Vehicle Link* e é o protocolo que define a comunicação entre o veículo não tripulado, que roda com ArduPilot ou *firmwares* derivados, e a estação de comando. A sigla MAV faz referência a um drone aéreo e foi utilizada para nomear o protocolo, mas ele também apresenta muitas funcionalidades compatíveis com veículos subaquáticos e outras categorias oriundas do projeto ArduPilot.

Conforme abordado na pesquisa de Koubâa et al. (2019), o MAVLink é um protocolo de serialização de mensagens leves, criado por Lorenz Meier em 2009. O protocolo foi projetado como uma biblioteca de *marshalling*, que significa transformar as mensagens de estado e de comando de um sistema em uma sequência de *bytes*. O MAVLink se tornou o protocolo mais difundido em sistemas não tripulados devido às suas características vantajosas: mensagens pequenas em comparação com outras técnicas de serialização, como XML e JSON, comunicação bidirecional entre o sistema e a estação de comando e dupla verificação por *checksum* para assegurar a integridade dos dados, além de permitir a conexão de até 255 sistemas de forma simultânea.

3.3.2 Trasmissão e Comunicação

O protocolo MAVLink define a estrutura das mensagens e como será feita sua serialização para as camadas de transporte e física. Devido ao seu tamanho reduzido, essas mensagens podem ser transmitidas por vários meios, como WiFi, Ethernet e outros canais de telemetria serial de baixa largura de banda, como Bluetooth. Segundo Koubâa et al. (2019), a taxa de transmissão pode chegar a até 250 kbps, e o alcance máximo pode chegar a 500 metros em condições sem interferência. Para comunicação sem fio ou por cabo, geralmente utiliza-se conexão UDP (*User Datagram Protocol*), que é rápida, mas não assegura a entrega da mensagem, ou TCP (*Transmission Control Protocol*), que oferece maior confiabilidade na entrega, mas pode introduzir atrasos devido à sua gestão. A escolha dos tipos de conexão da camada de transporte depende dos requisitos da aplicação (KOUBÂA et al., 2019).

3.3.3 Estrutura da Mensagem

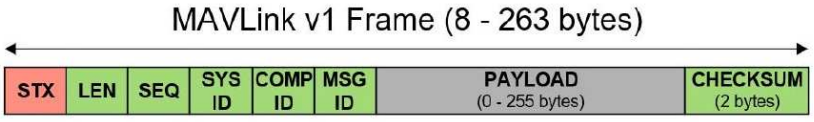
As mensagens MAVLink são sequências de *bytes* serializados que o receptor desserializa para reconstituir a mensagem. Cada mensagem é composta por um cabeçalho, uma carga de dados (ou *payload*) e um *checksum*. O cabeçalho contém informações sobre a mensagem, enquanto o *payload* armazena os dados transmitidos. O protocolo possui duas versões principais: a versão 1.0, que utiliza um cabeçalho de 8 *bytes*, e a versão 2.0, lançada em 2017, que utiliza um cabeçalho de 14 *bytes*. Ambas incluem 2 *bytes* de *checksum* (KOUBÂA et al., 2019).

Ambas as versões iniciam com um *byte* chamado `STX`, que sinaliza o início da mensagem; no MAVLink 1.0, ele corresponde ao valor `0xFE`. Após este sinalizador, a primeira versão do protocolo inclui os seguintes campos:

- `LEN`: indica o tamanho em *bytes* da mensagem enviada;
- `SEQ`: identifica a ordem da mensagem (variando de 0 a 255), reiniciando a contagem ao atingir o valor máximo;
- `SYS ID`: representa o ID do sistema não tripulado;
- `COMP ID`: identifica o tipo de hardware que está enviando a mensagem;
- `MSG ID`: especifica o tipo da mensagem enviada;
- `PAYLOAD`: contém os dados da mensagem, com limite de 255 *bytes*;
- `CHECKSUM`: composto por dois *bytes* para validação da mensagem.

Esses elementos garantem que o MAVLink 1.0 seja eficiente na comunicação entre o sistema e a estação de controle, mantendo a integridade e a clareza das mensagens transmitidas. A Figura 12 apresenta a estrutura da mensagem na primeira versão do protocolo.

Figura 12 – Estrutura da mensagem MAVLink versão 1.0



Fonte: (KOUBÂA et al., 2019)

O MAVLink 2.0 apresenta algumas diferenças em relação à primeira versão. Seu primeiro *byte* corresponde ao valor `0xFD`. Ele introduz dois novos campos de *flags* após o campo `LEN` e inclui até treze *bytes* de assinatura, além de um `MSG ID` expandido para 24 *bytes*, aumentando significativamente o número de tipos de mensagens suportados. As principais diferenças incluem:

- `Incompatibility Flag`: indica a presença de *bytes* de assinatura anexados ao final da mensagem.
- `Compatibility Flag`: contém uma lista de *flags* que podem ser ignoradas caso não sejam reconhecidas pelo interpretador, como, por exemplo, uma marcação

de alta prioridade na mensagem.

- `Signature`: Os treze *bytes* de assinatura são opcionais e aprimoram a segurança do protocolo. Eles permitem autenticar a mensagem e verificar sua origem, garantindo maior confiabilidade.

Essas modificações fazem do MAVLink 2.0 uma evolução em termos de segurança e flexibilidade, especialmente para aplicações que exigem maior autenticidade e controle das mensagens transmitidas. A Figura 13 apresenta a estrutura da mensagem no formato atualizado da versão 2.0. O protocolo é compatível com ambas as versões, identificando automaticamente qual delas está sendo utilizada por meio do identificador no início de cada mensagem.

Figura 13 – Estrutura da mensagem MAVLink versão 2.0

Fonte: (KOUBÂA et al., 2019)

3.3.4 Tipos de Mensagens MAVLink

A documentação apresentada pelo MAVLink Development Team (2024) define duas categorias principais de mensagens: mensagens de estado e mensagens de comando. As mensagens de estado são enviadas pelo ROV para a estação de comando, fornecendo dados sobre o estado do sistema, como ID, localização, velocidade e altitude. Já as mensagens de comando são emitidas pela estação de comando para o sistema não tripulado, instruindo-o a executar ações específicas, como navegar de um ponto A até um ponto B.

Na versão 1.0 do protocolo, é possível utilizar até 255 tipos de mensagens, cada uma identificada por um `MSG ID` de 8 *bits*. A partir da versão 2.0, o número de mensagens possíveis foi expandido significativamente, permitindo mais de 16 milhões de tipos, devido à expansão do campo `MSG ID` para 24 *bits*. Com isso, comandos com IDs maiores que 255 pertencem exclusivamente à segunda versão. A seguir, são apresentadas cinco mensagens consideradas relevantes para o controle do BlueROV2 neste trabalho.

3.3.4.1 HEARTBEAT

A mensagem `HEARTBEAT` é considerada a mais importante no protocolo MAVLink. Identificada pelo `Msg ID = 0`, ela indica se o sistema está ativo e respondendo. Essa mensagem é transmitida pelo veículo para a estação de comando a cada 1 segundo e contém campos que, nesta ordem, informam: o tipo de veículo, o tipo

de *firmware*, o modo de operação, os modos de operação suportados pelo *firmware*, o estado atual do sistema e a versão do protocolo MAVLink. A Figura 14 ilustra a distribuição de *bits* em cada campo da mensagem.

Figura 14 – Mensagem de *Heartbeat*

type	autopilot	base_mode	custom_mode	system_status	mavlink_version
8 bits	8 bits	8 bits	32 bits	8 bits	8 bits

Fonte: (KOUBÂA et al., 2019)

3.3.4.2 SYS_STATUS

A mensagem com `Msg ID = 1` informa o estado atual do sistema. Ela fornece dados sobre o funcionamento dos componentes conectados ao veículo, como bateria e sensores. Além disso, permite identificar problemas como perda de pacotes MAVLink transmitidos ou erros de comunicação interna. Em situações de baixo nível de bateria ou falhas de comunicação, é possível programar o ROV para executar ações de segurança automaticamente.

3.3.4.3 COMMAND_LONG

Identificada pelo `Msg ID = 76`, a mensagem de comando `COMMAND_LONG` permite o envio de diferentes tipos de instruções ao ROV, transmitindo a enumeração correspondente ao comando e seus parâmetros associados. Os dois primeiros campos da mensagem especificam o sistema e o componente do sistema com o qual a estação de comando está se comunicando, seguidos pela enumeração do comando e o campo de confirmação. Os campos restantes contêm parâmetros específicos que variam de acordo com o tipo de comando enviado. A estrutura da mensagem é apresentada na Figura 15.

Figura 15 – Mensagem `COMMAND_LONG`

target system	target component	command	confirmation	param1	param2	param3	param4	param5	param6	param7
unit8_t	unit8_t	unit16_t	unit8_t	float	float	float	float	float	float	float

Fonte: (KOUBÂA et al., 2019)

Um exemplo da aplicação dessa mensagem é enviar o comando para ativar ou desativar o veículo. De acordo com a documentação do `COMMAND_LONG`, os campos da mensagem devem incluir o ID do sistema, o ID do componente alvo, a enumeração `MAV_CMD_COMPONENT_ARM_DISARM (400)` correspondente ao comando, o número zero no campo de confirmação (que corresponde à primeira tentativa de envio da mensagem), os dois parâmetros do comando (conforme apresentado na Tabela 2)

e o restante dos parâmetros preenchidos com o número zero, já que este comando possui apenas dois parâmetros.

Tabela 2 – MAV_CMD_COMPONENT_ARM_DISARM (400)

Parâmetro	Descrição	Valores
1	0: ativar - 1: desativar	0 ou 1
2	0: respeitar verificação de segurança 21196: sobrepor a verificação de segurança	0 ou 21196

Fonte: Adaptado de MAVLink Development Team (2024)

3.3.4.4 COMMAND_ACK

A mensagem `COMMAND_ACK`, identificada pelo `Msg ID = 77`, é utilizada para fornecer o status da execução de um comando enviado ao ROV. Ela contém dois campos principais: o campo `command`, que identifica o `Command ID` do comando executado, e o campo `result`, que informa um valor inteiro correspondente ao resultado da execução conforme a Tabela 3. Essa mensagem é essencial para monitorar o desempenho do sistema, fornecendo feedback imediato sobre o sucesso, falha ou estado em execução de um comando emitido. (MAVLink Development Team, 2024).

Tabela 3 – MAV_RESULT

Valor	Parâmetro
0	MAV_RESULT_ACCEPTED
1	MAV_RESULT_TEMPORARILY_REJECTED
2	MAV_RESULT_DENIED
3	MAV_RESULT_UNSUPPORTED
4	MAV_RESULT_FAILED
5	MAV_RESULT_IN_PROGRESS
6	MAV_RESULT_CANCELLED
7	MAV_RESULT_COMMAND_LONG_ONLY
8	MAV_RESULT_COMMAND_INT_ONLY
9	MAV_RESULT_COMMAND_UNSUPPORTED_MAV_FRAME

Fonte: Adaptado de MAVLink Development Team (2024)

3.3.4.5 MANUAL_CONTROL

Esta mensagem, com `Msg ID = 69`, envia à estação de comando sinais equivalentes ao de um controlador manual, também conhecido como *joystick*. De acordo com a documentação fornecida pelo MAVLink Development Team (2024), o controle manual possui os seguintes campos: o ID do sistema, o valor normalizado dos eixos X, Y, Z e R, e um *bitfield* correspondente a um botão pressionado pelo *joystick* para algum comando específico, como ativar o veículo, alterar o modo de operação ou ativar sua câmera.

Os eixos são normalizados entre `[-1000, 1000]` e correspondem a uma direção de movimento do veículo ou à sua propulsão. A Tabela 4 mostra o significado de cada um dos deixos.

Tabela 4 – Eixos do comando MANUAL_CONTROL

Eixo	Direção Positiva	Direção Negativa
X	Para frente	Para trás
Y	Para frente	Para trás
Z	Propulsão positiva	Propulsão negativa
R	Rotação anti-horária	Rotação horária

Fonte: Adaptado de MAVLink Development Team (2024).

O comando para controle manual do veículo que emula os sinais emitidos por um *joystick* por ser útil para desenvolver algoritmos de navegação autônoma quando o ROV não possui um sistema de localização embutido, como, por exemplo, GSP, DVL ou outro tipo de sistema.

3.3.5 Pymavlink

O *pymavlink* é uma biblioteca de *software* desenvolvida em Python, disponível no repositório GitHub da ArduPilot[1], que implementa o protocolo MAVLink e permite a comunicação entre veículos não tripulados e estações de comando. Ela fornece uma interface programável para enviar e receber mensagens MAVLink, facilitando o controle, o monitoramento e o desenvolvimento de autonomia para veículos como ROVs e AUVs. Por ser uma implementação *open-source*, o *pymavlink* é amplamente utilizado no meio acadêmico e na indústria para criar sistemas baseados no protocolo MAVLink (MAVLink Development Team, 2024). No presente trabalho, o *pymavlink* desempenha um papel central, sendo utilizado na criação de uma biblioteca de funções para controle do BlueROV2, o que foi essencial para validar os objetivos propostos.

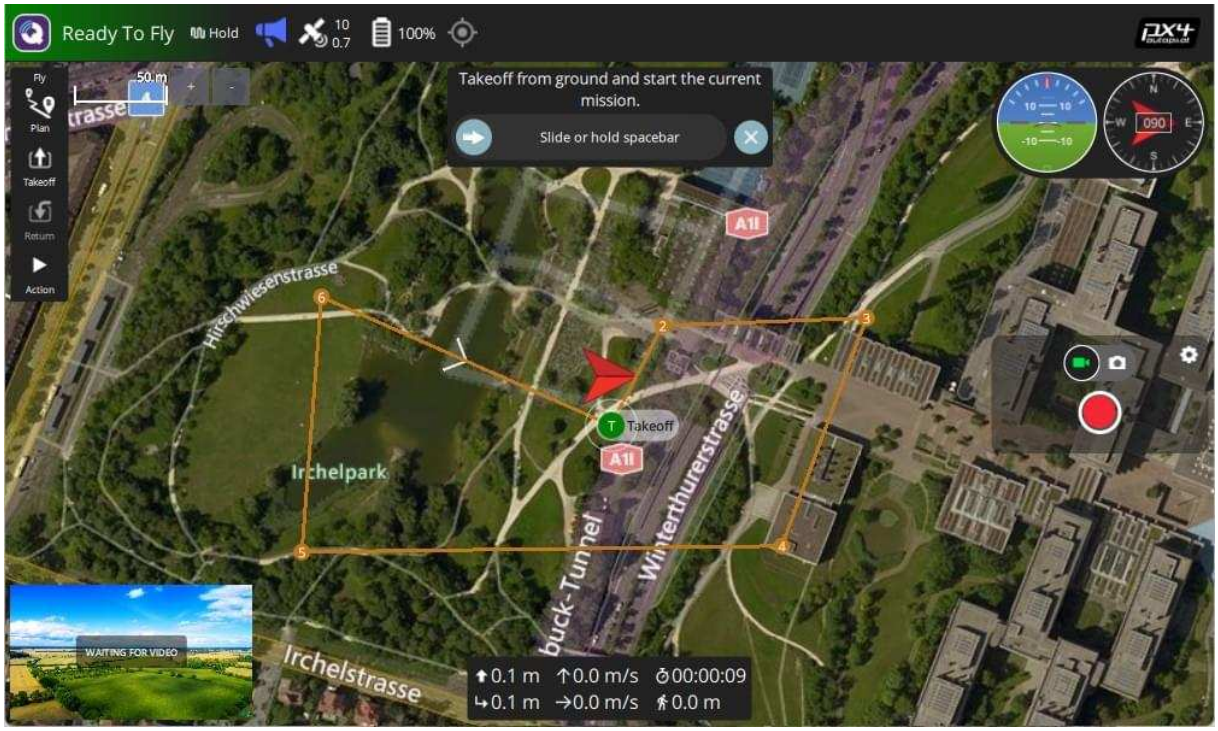
3.4 QGroundControl

O QGroundControl (QGC) é uma das principais estações de comando disponíveis, compatível com veículos baseados no ArduPilot, incluindo o ArduSub, e é distribuído como open-source. Estações de comando, também conhecidas como GCS (*Ground Control Station*), desempenham um papel crucial na navegação de ROVs. O QGC oferece uma interface gráfica para coleta de dados de telemetria, planejamento de missões e configuração do veículo. Através dela, é possível acessar informações detalhadas dos dispositivos do ROV, calibrar componentes, testar o sinal de vídeo do veículo e planejar missões autônomas para sistemas que já contam com essa funcionalidade de navegação (QGroundControl Documentation, 2024).

[1] Disponível em: https://github.com/ArduPilot/pymavlink. Acesso em: 2 de dezembro de 2024.

Uma das funcionalidades importantes do software é a seleção do modo de navegação do veículo. O modo guiado permite o planejamento de missões com o uso de sistemas de navegação, como GPS e DVL, para conduzir o veículo de um ponto a outro ou traçar uma rota completa, como ilustrado na Figura 16. Neste trabalho, utilizou-se o modo de navegação manual, pois a comunicação será feita via mensagens MAVLink, que emulam os sinais deste tipo de controle.

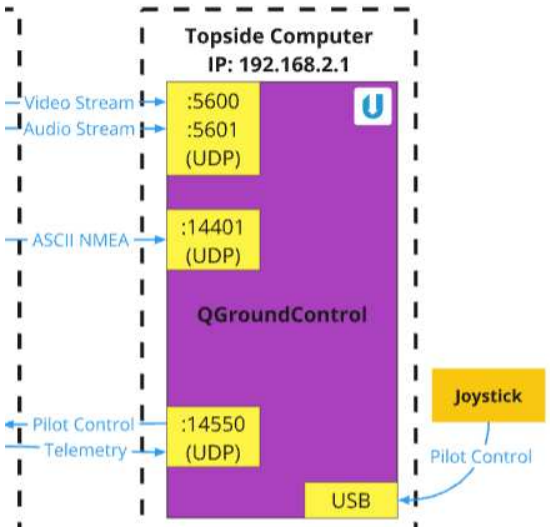
Figura 16 – Interface Gráfica do QGroundControl



Fonte: QGroundControl Documentation (2024)

Quanto à interação com o ArduSub, o QGC funciona como um software intermediário, capturando informações importantes por meio da leitura de portas UDP. A porta `14450` é responsável por receber, via protocolo MAVLink, dados de telemetria do veículo e mensagens de comando para o piloto automático. Já as portas `5600` e `5601` são utilizadas para receber os sinais de vídeo e áudio, respectivamente. O QGC também permite a conexão de um *joystick* ao computador via USB, além do envio de comandos diretamente pelo protocolo MAVLink, utilizando a porta mencionada. Adicionalmente, sinais de GPS podem ser acessados pela porta `14401`. A Figura 17, apresentada pelo ArduSub Development Team (2024), exibe parte da arquitetura de software e *hardware* do ArduSub e a comunicação com a estação de comando.

Figura 17 – Comunicação do QGroundControl com o ROV



Fonte: (ArduSub Development Team, 2024)

3.5 Simulação de ROVs e AUVs

A simulação é uma ferramenta indispensável no desenvolvimento e teste de veículos não tripulados, permitindo validar sistemas em condições controladas antes de sua aplicação em ambientes reais. Para ROVs e AUVs, um dos principais desafios é integrar os componentes de *hardware* e *software* em um ambiente virtual que simule de forma fidedigna as condições físicas e operacionais enfrentadas em cenários subaquáticos.

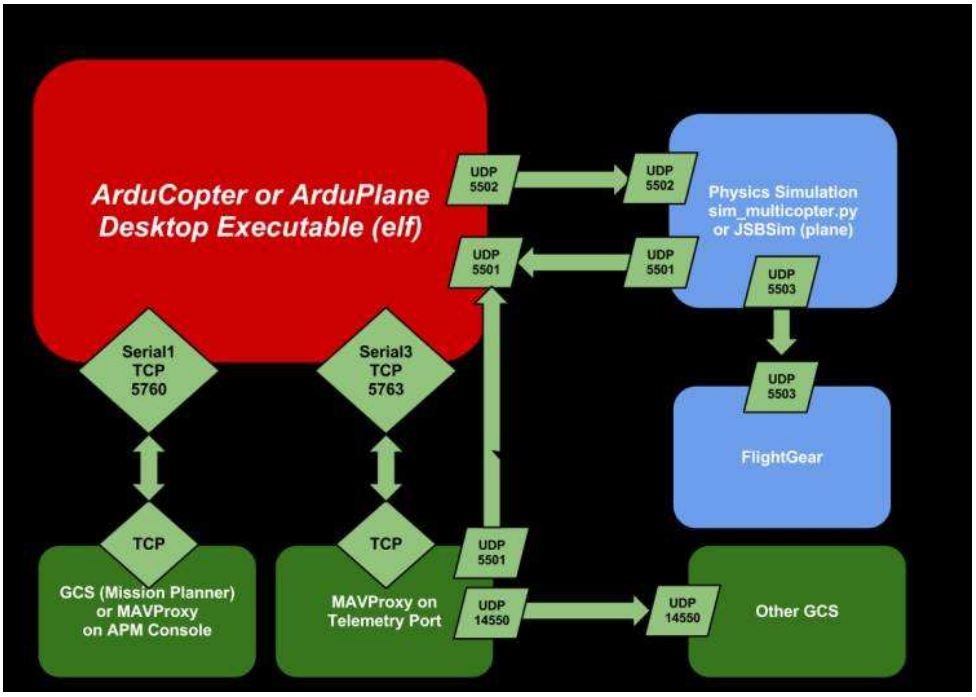
Duas abordagens amplamente utilizadas para a simulação de sistemas não tripulados são o SITL (*Software-in-the-Loop*) e o HITL (*Hardware-in-the-Loop*). O SITL simula o comportamento do veículo em um ambiente controlado usando exclusivamente *software*, enquanto o HITL incorpora componentes reais de *hardware* ao processo de simulação. Considerando que testes com veículos reais são frequentemente custosos e complexos, essas abordagens oferecem uma alternativa econômica e prática para o desenvolvimento e validação de sistemas. Este trabalho utilizou apenas o SITL.

3.5.1 SITL - *Software in the Loop*

De acordo com Qays, Jumaa e Salman (2020), o SITL possibilita a realização de testes com ROVs e AUVs de diferentes tipos, eliminando a necessidade de componentes físicos, como controladores (por exemplo, o Navigator ou Pixhawk, no caso do BlueROV2) ou atuadores. Além disso, o SITL viabiliza o teste de códigos para navegação autônoma, como aqueles baseados no protocolo MAVLink, o que o torna uma ferramenta valiosa para o desenvolvimento e validação de tecnologias voltadas para sistemas não tripulados.

O ArduPilot Development Team (2024) apresenta um diagrama que ilustra a configuração das portas de comunicação na simulação SITL e sua interação com a estação de comando. Na Figura 18, observa-se que duas portas TCP, `5760` e `5763`, são destinadas à comunicação, sendo necessário configurá-las corretamente para garantir a troca eficiente de mensagens MAVLink, possibilitando a realização de testes em tempo real, como a verificação de telemetria e envio de comandos ao veículo virtual.

Figura 18 – Conexões do ArduPilot SITL



Fonte: ArduPilot Development Team (2024)

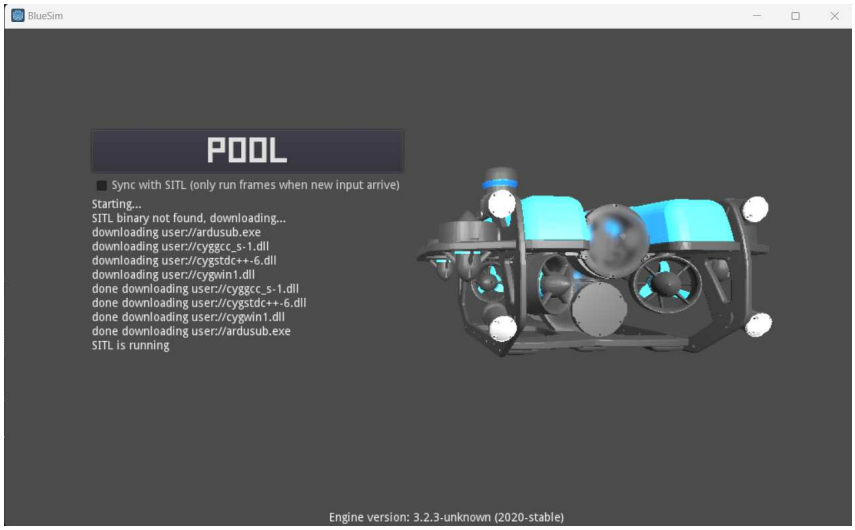
3.6 BlueSim

O BlueSim é um ambiente de simulação para o BlueROV2 disponibilizado no repositório GitHub da Blue Robotics[2]. Ele utiliza o Godot Engine[3], uma plataforma *open-source* para desenvolvimento de aplicações 2D e 3D. O BlueSim executa sua própria instância de SITL, permitindo testes de funcionalidades do ROV em um ambiente virtual controlado. Além disso, ele pode ser integrado a outras instâncias de *Software-in-the-loop*, caso necessário. A Figura 19 mostra a tela inicial do *software*.

Após iniciar a aplicação, é necessário configurar a estação de comando para conectar via protocolo TCP/IP. Esse procedimento é detalhado no Capítulo 4, onde são descritos os passos para estabelecer a comunicação entre o BlueSim e a estação de comando, bem como os resultados obtidos na simulação.

[2] Disponível em: https://github.com/bluerobotics/bluesim. Acesso em: 2 de dezembro de 2024.
[3] Disponível em: https://docs.godotengine.org/en/stable/index.html. Acessado em: 2 de dezembro de 2024.

Figura 19 – Tela inicial do BlueSim



Fonte: Captura de tela realizada pelo autor no *software* BlueSim.

3.7 Níveis de Autonomia de AUVs

A autonomia de veículos não tripulados é amplamente discutida na literatura, com abordagens que analisam diferentes níveis de independência e interação humana. Nicholson e Healey (2008) enfatiza a autonomia como a tecnologia que possibilita enfrentar os vários desafios da comunicação subaquática. É comum classificar um ROV como um AUV após a adição de funcionalidades autônomas, mas essa transição envolve nuances operacionais e tecnológicas. De acordo com Huang (2004) apud Christ e Wernli (2013), os modos de operação para veículos não tripulados podem ser categorizados em quatro níveis principais: *completamente autônomo, semi-autônomo, tele-operado e remotamente controlado*. Cada nível reflete o grau de autonomia e a necessidade de interação humana:

Completamente autônomo — O veículo realiza sua missão de forma independente, sem qualquer intervenção humana durante a operação.

Semi-autônomo — Algumas tarefas do veículo requerem interação humana para serem executadas, enquanto outras são realizadas de forma autônoma.

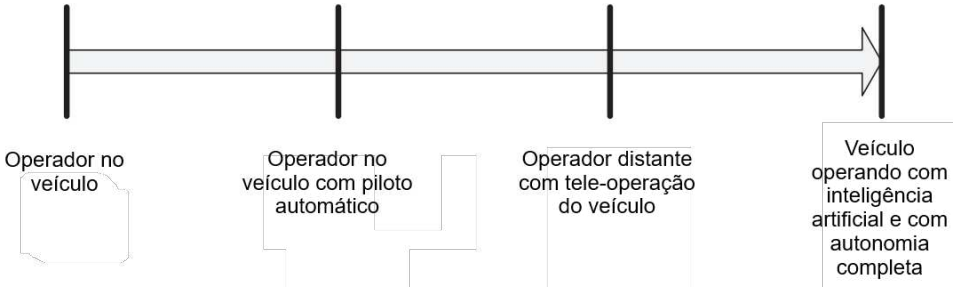
Tele-operado — O operador utiliza informações dos sensores e vídeo do veículo para enviar comandos remotamente, seja por cabo ou outros meios de comunicação. O veículo pode executar ações limitadas pré-programadas, como evitar obstáculos ou coletar dados.

Remotamente controlado — Neste modo, o veículo é completamente dependente do operador humano, sem qualquer iniciativa própria. Toda a operação exige interação contínua com o operador.

Os níveis de autonomia podem ser representados também graficamente, conforme apresentado na Figura 20. O avanço da autonomia pode ser visto como

um caminho a ser percorrido, onde as funcionalidades autônomas substituem cada vez mais os comandos realizados pelo operador, exigindo sistemas robustos e complexos, que incluem sensores avançados, algoritmos de inteligência artificial e mecanismos de tomada de decisão autônoma.

Figura 20 – Níveis de Autonomia



Fonte: Adaptado de Christ e Wernli (2013).

4 TESTES DE COMUNICAÇÃO E CONTROLE DO BLUEROV2 EM SIMULAÇÃO E LABORATÓRIO

Neste capítulo, são apresentados os testes realizados para validar os conceitos e ferramentas discutidos nos capítulos anteriores. Esses testes foram conduzidos em dois contextos distintos: um ambiente de simulação, utilizando o BlueSim, e um ambiente físico controlado, com o BlueROV2 em laboratório. O objetivo principal foi avaliar a comunicação e o controle do veículo por meio do protocolo MAVLink, além de testar a funcionalidade da biblioteca de funções desenvolvida e explorar o uso da câmera integrada.

A primeira parte deste capítulo aborda a configuração das ferramentas utilizadas no ambiente virtual e os testes realizados para compreender o funcionamento das mensagens de comando e estado do protocolo MAVLink. Em seguida, são detalhadas as etapas para configurar o computador para conexão com o BlueROV2 e os experimentos realizados em laboratório com o veículo. Por fim, são apresentados os resultados da coleta de telemetria e as análises feitas a partir das imagens capturadas.

4.1 Configuração do BlueSim

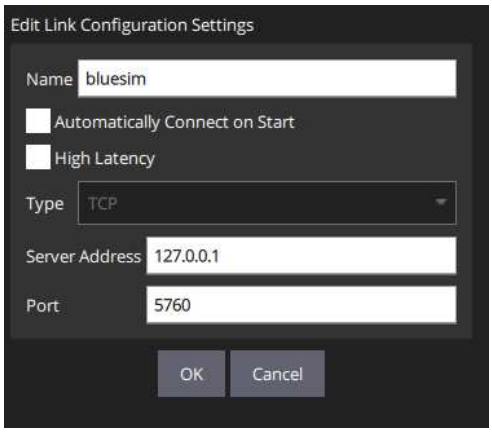
Conforme descrito no repositório do GitHub do BlueSim, a conexão ao SITL é realizada utilizando o protocolo TCP/IP com o endereço IP `127.0.0.1`, ou *localhost*, que aponta para o computador onde o *software* está sendo executado. A porta `5760` é destinada à estação de comando, enquanto a porta `5763` é reservada para o envio de mensagens MAVLink, conforme ilustrado na Figura 18.

Os seguintes passos foram realizados para configurar o ambiente utilizando a estação de comando QGroundControl:

1. Abrir o QGroundControl.
2. Acessar a opção *Application Settings*.
3. Na aba *Comm Links*, clicar em *Add*.
4. Selecionar o tipo de conexão TCP, inserir `127.0.0.1` no campo *Server Address* e `5760` em *Port* (vide Figura 21).
5. Executar o aplicativo BlueSim e clicar em *Pool* para iniciar a simulação.
6. Selecionar a conexão criada e clicar em *Connect*.
7. Controlar o veículo utilizando o teclado do computador, um *joystick*, ou comandos MAVLink.

Uma vez que a comunicação seja estabelecida corretamente, o QGroundControl exibirá o status de conexão como *Ready to Fly* e o modo de operação como *Manual*. A localização inicial do ROV será exibida em um mapa, posicionando-o

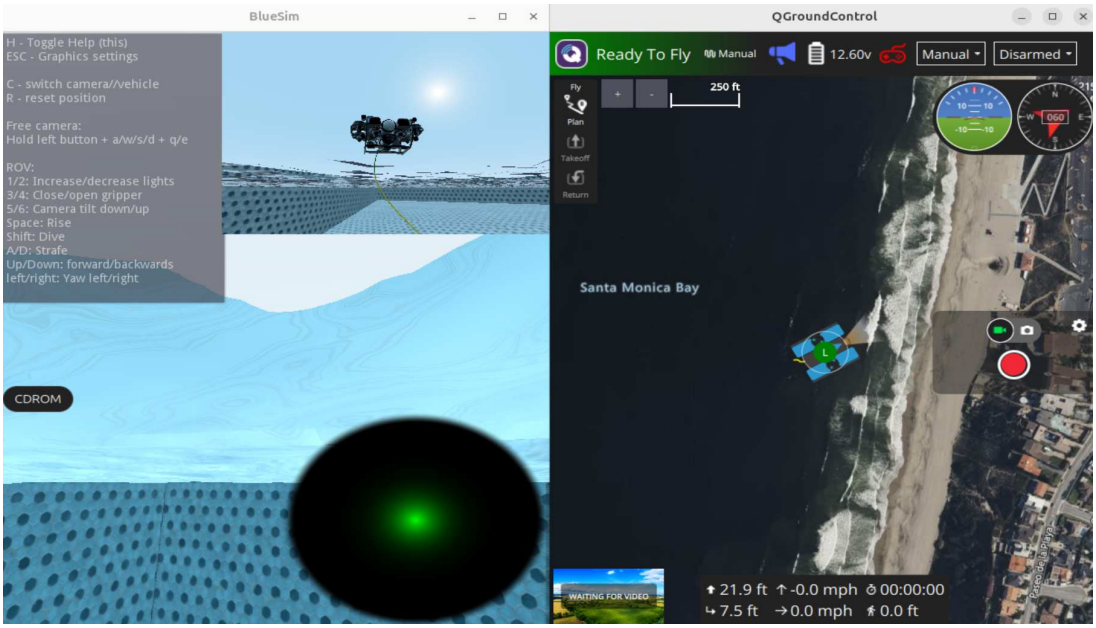
Figura 21 – Criação de conexão TPC entre o BlueSim e o QGroundControl



Fonte: Captura de tela realizada pelo autor no *software* QGroundControl.

no mar, próximo a uma região costeira. Além disso, qualquer movimentação realizada no BlueSim, seja manual ou por comandos MAVLink, será refletida em tempo real no mapa da estação de comando. A Figura 22 apresenta a interface dos *softwares* após a conexão.

Figura 22 – Conexão estabelecida entre o BlueSim e o QGroundControl



Fonte: Captura de tela realizada pelo autor nos *softwares* BlueSim e QGroundControl.

Com o ambiente de simulação devidamente configurado, o próximo passo foi realizar testes utilizando o protocolo MAVLink no BlueSim. Esses testes permitiram avaliar a comunicação entre o simulador e a estação de comando, bem como validar o envio e a recepção de mensagens relevantes para o controle do veículo, tema abordado na próxima seção.

4.2 Testes de Envio de Mensagens MAVLink no BlueSim

O BlueSim, por executar sua própria instância de *Software-in-the-Loop* (SITL), permite realizar testes que simulam o comportamento do ROV físico em um ambiente virtual controlado. O objetivo principal desses testes é verificar o funcionamento do protocolo MAVLink, utilizando a biblioteca *pymavlink*, implementada em Python, para enviar e receber comandos MAVLink. A versão do Python utilizada nos experimentos foi a 3.12.

Antes de iniciar os testes, é necessário instalar o *pymavlink* e importar a biblioteca *mavutil*[1], que fornece funções utilitárias de baixo nível para comunicação MAVLink. Essa biblioteca facilita a criação de conexões com o veículo, o envio de mensagens e o processamento de respostas.

Para instalar o *pymavlink*, recomenda-se a criação de um ambiente virtual, que centraliza todas as dependências do projeto, evitando conflitos com outros ambientes no sistema. A instalação pode ser realizada globalmente ou no ambiente virtual com o comando:

Listing 4.1 – Comando para instalar a biblioteca pymavlink

```
1    pip install pymavlink
```

Após a instalação, a biblioteca *mavutil* pode ser incluída no início do código com o seguinte comando:

Listing 4.2 – Importação da biblioteca mavutil

```
1    from pymavlink import mavutil
```

Com essa configuração, as funções utilitárias da biblioteca estão disponíveis para estabelecer a comunicação e executar comandos no ROV.

4.2.1 Teste de Conexão e *Heartbeat*

O primeiro teste realizado foi a conexão ao veículo simulado no BlueSim. A comunicação ocorre por meio de uma conexão TCP com o endereço `127.0.0.1` (também chamado de *localhost*), utilizando a porta `5763`. Essa configuração é necessária, pois a porta `5760` já está sendo usada pela estação de comando (QGroundControl). O código abaixo ilustra como estabelecer a conexão:

Listing 4.3 – Estabelecimento de conexão com o BlueSim

```
1    connection = mavutil.mavlink_connection("tcp:127.0.0.1:5763")
```

A variável `connection` é criada para armazenar um objeto que representa as informações da conexão estabelecida, como os IDs do sistema e do componente do

[1] Fonte: https://github.com/ArduPilot/pymavlink/blob/master/mavutil.py

sistema que está comunicando-se com o computador. Após estabelecer a conexão, o próximo passo é verificar o recebimento da mensagem *HEARTBEAT*, que confirma que o veículo está ativo e pronto para comunicação. O seguinte código permite aguardar e capturar essa mensagem:

Listing 4.4 – Recebimento do Heartbeat

```
1    connection.wait_heartbeat()
2    print("Conexao␣estabelecida!␣Heartbeat␣recebido.")
```

Se o *Heartbeat* for recebido corretamente, a conexão está configurada, e o veículo simulado pode ser controlado utilizando os comandos do protocolo MAVLink. Além disso, os campos dessa mensagem, representados na Figura 14, podem ser consultados utilizando a função `recv_match()` da biblioteca `mavutil`. Essa função permite receber e filtrar mensagens MAVLink específicas. Abaixo, é apresentado um exemplo de como verificar os campos da mensagem *HEARTBEAT* e acessar outros dados de telemetria:

Listing 4.5 – Campos da mensagem Heartbeat

```
1    msg = connection.recv_match(type="HEARTBEAT", blocking=True,
        timeout=5)
2    print(msg)
```

No exemplo acima, o parâmetro `type` define o tipo da mensagem que desejamos filtrar, enquanto `blocking=True` impede a execução do programa até que a mensagem especificada seja recebida. A execução do código exibe os campos da mensagem *HEARTBEAT*, que é enviada pelo ROV a cada segundo. A Figura 23 apresenta o resultado gerado pelo terminal:

Figura 23 – Saída da função `recv_match()` filtrando a mensagem *HEARTBEAT*

```
(venv) alysson@alysson:~/bluerov2-TCC/codes/telemetry$ python3 listen.py
HEARTBEAT {type : 12, autopilot : 3, base_mode : 81, custom_mode : 19, sys
tem_status : 3, mavlink_version : 3}
```

Fonte: Captura de tela realizada pelo autor no terminal de comando.

Se o `timeout` for configurado, como no exemplo acima, o programa aguardará pela mensagem por até cinco segundos antes de prosseguir. Esse parâmetro é especialmente útil para avaliar a estabilidade da conexão entre a estação de comando e o veículo simulado. Caso a mensagem não seja recebida no período especificado, é possível implementar lógicas para diagnosticar falhas de comunicação, como alertas ou tentativas de reconexão.

4.2.2 Comandos de Ativar e Desativar do ROV

Dois comandos essenciais para o controle de ROVs são os que permitem ativar e desativar o veículo. Estes comandos são necessários para ativar ou desativar os propulsores, seja para iniciar operações, seja por razões de segurança, como evitar danos em situações de emergência. Para implementá-los, utilizaremos a mensagem *COMMAND_LONG*, conforme especificado no protocolo MAVLink. Adicionalmente, como discutido na Seção 3.3.4.4, a mensagem *COMMAND_ACK* pode ser utilizada para verificar se os comandos foram executados com sucesso. O código a seguir ilustra a implementação desses comandos no ambiente simulado:

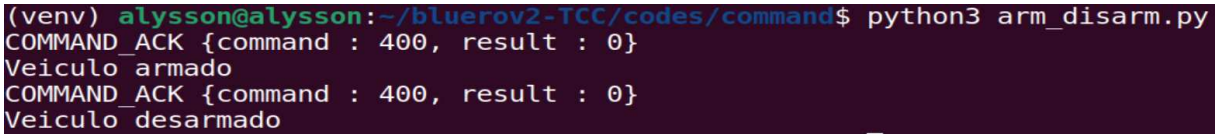
Listing 4.6 – Comando para ativar e desativar o veículo

```
    #Ativando veiculo
    connection.mav.command_long_send(connection.target_system,
    connection.target_component,
    mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM, #enumeracao
    0, #confirmacao
    1, #1 para ativar
    0, #0 para nao forcar ativacao
    0, 0, 0, 0, 0) #parametros ignorados

    ack_msg = connection.recv_match(type="COMMAND_ACK", blocking=
        True)
    print(ack_msg)
    time.sleep(2)

    #Desativando veiculo
    connection.mav.command_long_send(connection.target_system,
    connection.target_component,
    mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM, #enumeracao
    0, #confirmacao
    0, #0 para desativar
    0, #0 para nao forcar desativacao
    0, 0, 0, 0, 0) #parametros ignorados

    ack_msg = connection.recv_match(type="COMMAND_ACK", blocking=
        True)
    print(ack_msg)
```

O código acima utiliza o objeto `connection`, criado no momento da conexão com o veículo, para incluir os parâmetros necessários no comando. A função `recv_match()` é usada para capturar e exibir a mensagem *COMMAND_ACK*, enviada

pelo veículo para confirmar o sucesso ou falha do comando executado. A execução do código pode ser visualizada na Figura 24:

Figura 24 – Saída da função `recv_match()` filtrando a mensagem *COMMAND_ACK*



Fonte: Captura de tela realizada pelo autor no terminal de comando.

A mensagem *COMMAND_ACK* retornada pelo veículo contém dois valores principais: `command: 400`, que representa a enumeração do comando, e `result: 0`, indicando que o comando foi executado com sucesso (verificar Tabela 3). Em uma situação hipotética, o valor `2` para o campo `result` significaria "comando negado", enquanto o valor `4` indicaria "falha na execução do comando". Além disso, o QGroundControl reflete essas ações, exibindo mensagens visuais e emitindo um alerta sonoro ao ativar ou desativar o veículo.

4.2.3 Teste de Movimentação do ROV

O protocolo MAVLink oferece comandos de movimento que dependem de sistemas de localização, como GPS. Quando o QGroundControl identifica dispositivos desse tipo, o modo de operação guiado (*guided*) pode ser ativado, permitindo o planejamento de missões baseadas em coordenadas geográficas. Como o BlueSim possui integração com um sinal GPS simulado, foi possível testar esses comandos de movimentação em ambiente virtual. Nesta seção, serão apresentados os testes realizados com comandos de navegação guiada no BlueSim. Já os testes de controle manual, realizados com o veículo físico, serão abordados nas seções subsequentes.

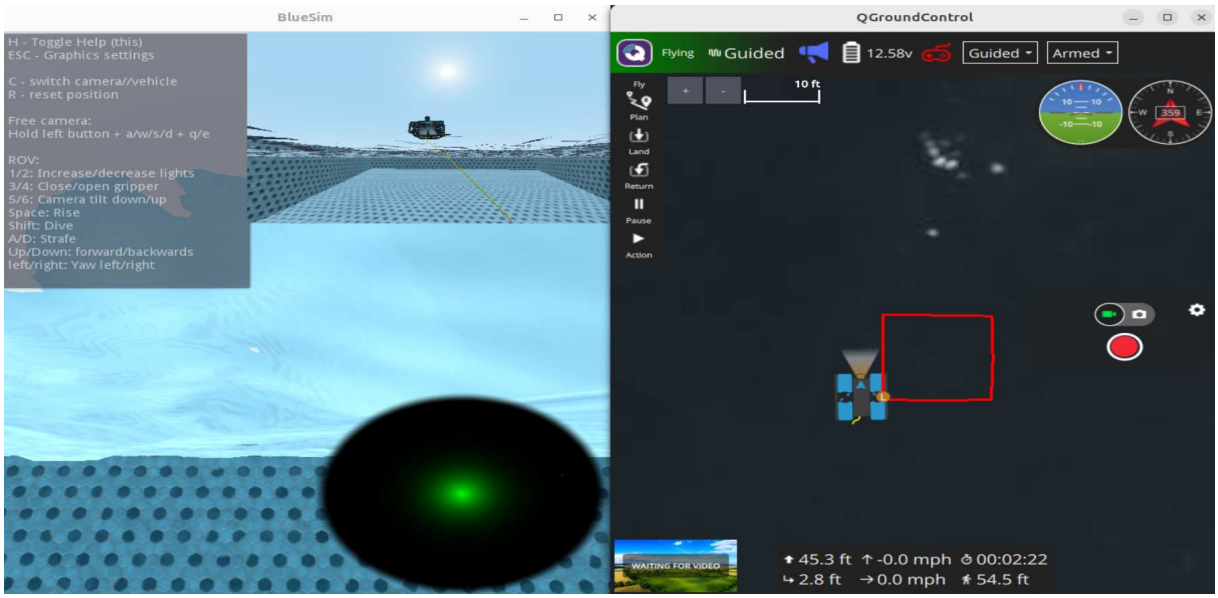
Listing 4.7 – Comando de movimento com coordenadas NED

```
1    connection.mav.send(mavutil.mavlink.
        MAVLink_set_position_target_local_ned_message(
2    10, #tempo desde o inicio do sistema em milissegundos
3    connection.target_system, connection.target_component,
4    mavutil.mavlink.MAV_FRAME_LOCAL_OFFSET_NED, #sistema de
        referencia
5    int(0b110111111000), #mascara de bits para ativar funcoes
6    north, east, down, #mudanca de posicao ativadas com os bits 0s
7    0, 0, 0, #mudanca de velocidade desativada
8    0, 0, 0, #mudanca de aceleracao desativada
9    0, 0)) #mudanca de rotacao desativada
```

O código acima utiliza a função `send` para enviar a mensagem de comando de

movimento que utiliza o sistema de coordenadas NED. Este sistema envia comandos nas direções norte, leste e para baixo ao veículo. O objetivo deste teste foi conhecer as possibilidades e tipos de comandos existentes, mas sua explicação não será aprofundada pois não foi utilizada neste trabalho com o veículo físico que não possui um sistema de comunicação. Apenas para esclarecimento, os campos do código acima `north`, `east` e `down` receberam o comando de navegar cinco metros em cada direção cardinal (norte, leste, sul e oeste) e o resultado pode ser visto na Figura 25 que mostra o registro de movimento pelo QGC.

Figura 25 – Teste de movimento com o sistema NED



Fonte: Captura de tela realizada pelo autor nos *softwares* BlueSim e QGroundControl.

Após concluir os testes no ambiente simulado, as próximas seções exploram os experimentos realizados com o veículo físico em laboratório. Embora existam limitações inerentes ao ambiente controlado, esses testes têm como objetivo validar os resultados obtidos no BlueSim e demonstrar a viabilidade prática dos comandos e funcionalidades desenvolvidos. Além disso, os experimentos em laboratório fornecem uma perspectiva mais próxima da operação em campo, mesmo sem reproduzir integralmente as condições reais.

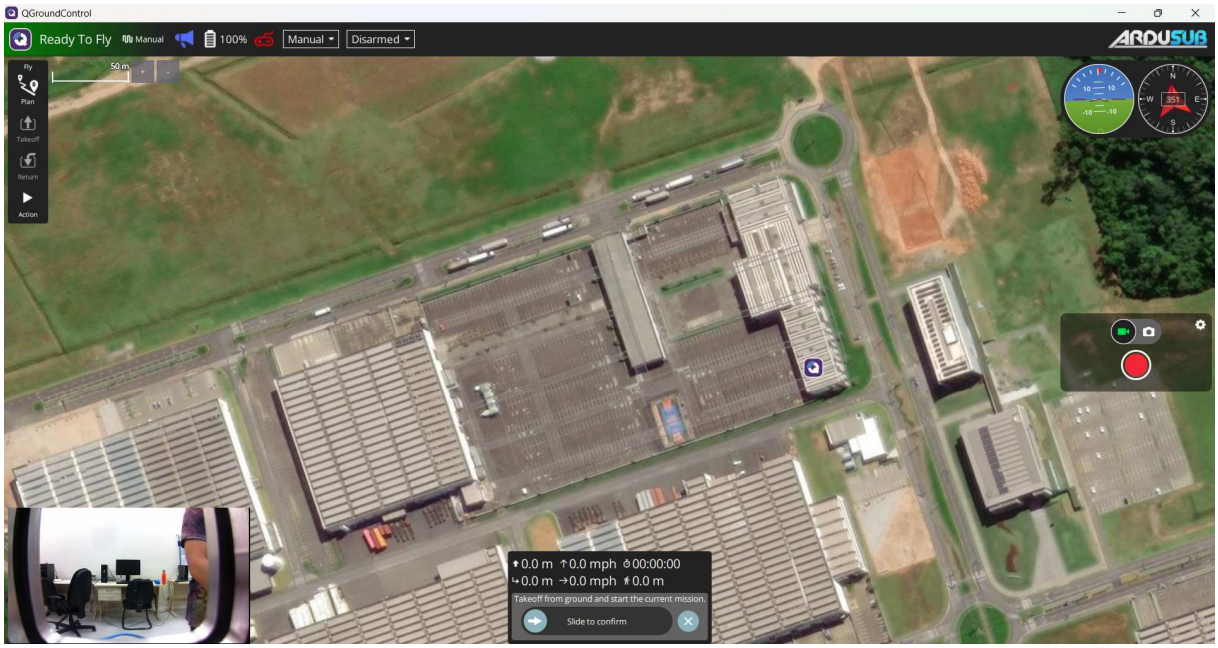
4.3 Configuração de Rede para Comunicação com o BlueROV2

Para estabelecer comunicação com o BlueROV2, é necessário garantir que o computador e o veículo estejam na mesma faixa de rede. O *software* Companion, executado em uma placa Raspberry Pi e responsável pelo gerenciamento das conexões de rede do BlueROV2, utiliza o endereço IP padrão `192.168.2.2`. Para configurar o computador, deve-se atribuir um endereço IP dentro da mesma faixa, como `192.168.2.X`, e definir a máscara de sub-rede como `255.255.255.0`. Neste projeto, o

IP do computador foi configurado como `192.168.2.1` para realizar os testes.

Após realizar essa configuração de rede, a estação de comando QGroundControl deve ser capaz de detectar automaticamente o veículo ao ser iniciada. O QGC também oferece suporte a *joysticks*, permitindo o controle manual do veículo, que estará pronto para operar. A interface exibida no QGroundControl ao estabelecer a conexão é apresentada na Figura 26.

Figura 26 – Reconhecimento automático do BlueROV2 pelo QGroundControl



Fonte: Captura de tela realizada pelo autor no *software* QGroundControl.

Além da configuração de rede, a conexão física entre o computador e o BlueROV2 é feita por meio de um cabo Ethernet. Esse cabo conecta o computador à placa LX200v20[2], que serve como interface de comunicação entre o Raspberry Pi e o computador. A LX200v20 gerencia a troca de dados e garante a conectividade com o *software* Companion, permitindo o envio e recebimento de mensagens entre os dispositivos.

Como mencionado no capítulo anterior, a Figura 11 apresenta o diagrama da arquitetura de *software* do BlueROV2, que auxilia na compreensão dos endereços IP e portas utilizados para comunicação. Diferentemente do BlueSim, que utiliza comunicação TCP, o BlueROV2 utiliza o protocolo UDP para envio e recebimento de mensagens. A porta `14550` é usada para coleta de dados de telemetria, enquanto a porta `5600` é destinada à coleta de imagens transmitidas pela câmera do veículo. A próxima seção abordará a utilização dessas portas e configurações na prática, demonstrando o envio de comandos MAVLink e a coleta de dados durante os testes realizados.

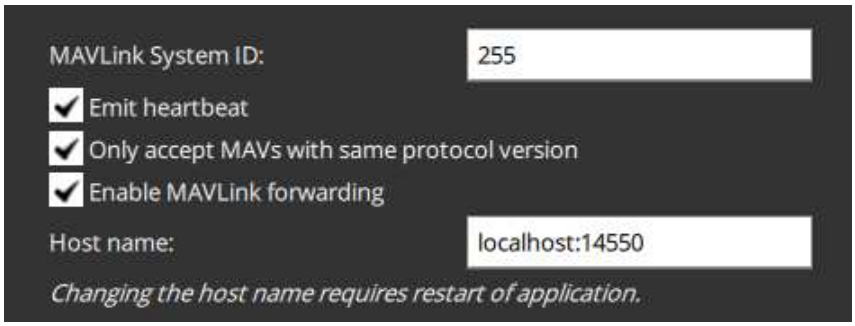
[2] Datasheet: https://bluerobotics.com/wp-content/uploads/2022/07/LX200V20-Datasheet-v1.2.pdf

4.4 Testes de Comunicação e Controle com o BlueROV2

Após a configuração apresentada na seção anterior, o BlueROV2 está pronto para operação, podendo ser controlado manualmente ou, como neste trabalho, validado para testes com o protocolo MAVLink previamente executados em ambiente simulado.

O QGroundControl desempenha um papel essencial na execução dos testes, servindo como intermediário para a emissão do *Heartbeat* do ROV e habilitação do envio de comandos pela porta UDP `14550`. Para garantir o funcionamento correto dos testes, o usuário deve acessar a opção *Application Settings*, navegar até a aba *MAVLink*, e habilitar as opções *Emit heartbeat* e *Enable MAVLink forwarding*. A Figura 27 ilustra a configuração necessária no QGroundControl para viabilizar a comunicação com o BlueROV2.

Figura 27 – Habilitação do Hearbeat e Envio de Comandos MAVLink



Fonte: Captura de tela realizada pelo autor no *software* QGroundControl.

Desta forma, o QGroundControl estará intermediando a conexão, permitindo iniciar a comunicação de maneira similar à realizada na simulação com o BlueSim. A única diferença está no tipo de conexão, que deve ser configurado como `udp` ou `udpin`, e na porta de comunicação, que deve ser definida como `14550` (`localhost` e `127.0.0.1` são endereços equivalentes):

Listing 4.8 – Início da conexão no BlueROV2

```
1  from pymavlink import mavutil
2
3  connection=mavutil.mavlink_connection("udpin:localhost:14550")
4  connection.wait_heartbeat()
5  print("Conexao␣estabelecida");
```

O código acima aguarda indefinidamente pela mensagem *HEARTBEAT* enviada pelo BlueROV2 para confirmar que a conexão foi estabelecida. Caso deseje limitar o tempo de espera, é possível utilizar o parâmetro `timeout=10`, configurando, por exemplo, um tempo de espera de dez segundos (ou outro intervalo conforme necessário). A função `wait_heartbeat()` da biblioteca *mavutil* utiliza internamente `recv_match()` para filtrar e receber exclusivamente a mensagem *HEARTBEAT*. Esse

método assegura que a comunicação com o veículo seja iniciada apenas após o recebimento da mensagem de inicialização, permitindo iniciar os testes explicados a seguir.

4.4.1 Coleta dos Dados de Telemetria

Durante a simulação, utilizamos a função `recv_match()` para filtrar a mensagem *HEARTBEAT*. Essa função permite receber uma das muitas mensagens enviadas pelo BlueROV2 para a estação de comando. No entanto, ao executá-la sem especificar um filtro dentro de um *loop while*, uma grande variedade de dados é exibida continuamente conforme os pacotes são recebidos, o que pode dificultar a visualização e interpretação. Quando utilizada fora de um *loop*, a função retorna apenas a primeira mensagem recebida após a execução do programa. A próxima seção discutirá a aplicação de filtros e destacará quais dados foram considerados mais relevantes para coleta, com base nas características do veículo.

Para fins de demonstração, a Figura 28 exibe a saída da função em execução única, capturando apenas uma mensagem aleatória durante o processo. Já a Figura 29 apresenta a saída da função `recv_match()` em execução contínua dentro de um *loop while*, exibindo diversas mensagens recebidas.

Figura 28 – Saída da função `recv_match()` ao capturar uma única mensagem

```
PS C:\Users\alyss\OneDrive\Documentos\TCC\pymavlink> python .\listen.py
SCALED_PRESSURE2 {time_boot_ms : 25812, press_abs : 1009.8999633789062, press_diff : -0.699999988079071,
temperature : 2066}
```

Fonte: Captura de tela realizada pelo autor no terminal de comando.

Figura 29 – Saída contínua da função `recv_match()` em execução com *loop*

```
PS C:\Users\alyss\OneDrive\Documentos\TCC\pymavlink> python .\listen.py
MISSION_ACK {target_system : 1, target_component : 1, type : 1}
ATTITUDE {time_boot_ms : 351697, roll : 0.034545429050922394, pitch : 0.00942760705947876, yaw : -0.17425
790429115295, rollspeed : 0.002605913206934929, pitchspeed : -0.0002697559248190373, yawspeed : -0.008318
223059177399}
MISSION_ACK {target_system : 1, target_component : 1, type : 13}
AHRS2 {roll : 0.037314146757125854, pitch : 0.0064020222052931786, yaw : -0.20812220871448517, altitude :
 0.0, lat : 0, lng : 0}
AHRS3 {roll : 0.034545429050922394, pitch : 0.00942760705947876, yaw : -0.17425790429115295, altitude : 0
.0, lat : 0, lng : 0, v1 : 0.0, v2 : 0.0, v3 : 0.0, v4 : 0.0}
MISSION_ACK {target_system : 1, target_component : 1, type : 1}
MISSION_ACK {target_system : 1, target_component : 1, type : 1}
MISSION_ACK {target_system : 1, target_component : 1, type : 1}
MISSION_ACK {target_system : 1, target_component : 1, type : 13}
MISSION_ACK {target_system : 1, target_component : 1, type : 1}
ATTITUDE {time_boot_ms : 463972, roll : 0.030462879687547684, pitch : 0.009489759802818298, yaw : -0.1727
2859811782837, rollspeed : 0.0019275068771094084, pitchspeed : 0.0008386272238567472, yawspeed : -0.00783
9813828468323}
```

Fonte: Captura de tela realizada pelo autor no terminal de comando.

4.4.2 Comandos de Movimento

Como o BlueROV2 não possui um sistema de navegação integrado, utilizaremos a mensagem de comando *MANUAL_CONTROL* para simular os controles de um *joystick* e acionar os propulsores do veículo. A Subseção 3.3.4.5 apresenta em detalhe os parâmetros necessários para enviar este comando e controlar o veículo de maneira manual.

Uma limitação importante dos testes realizados em laboratório está relacionada ao tempo de acionamento dos propulsores. Devido às características físicas dos motores, que dependem de água para lubrificação, não é possível operá-los por períodos prolongados ou com potência elevada. Dessa forma, os testes foram conduzidos com restrições apropriadas, priorizando a segurança do equipamento e evitando danos aos componentes.

O teste consistiu nos seguintes passos:

1. Ativar o BlueROV2;
2. Comandar o veículo para se mover para frente;
3. Comandar o veículo para se mover para trás;
4. Comandar o veículo para se mover para a esquerda;
5. Comandar o veículo para se mover para a direita;
6. Comandar o veículo para realizar uma rotação;
7. Desativar o veículo.

Cada um destes passos deve ser acompanhado pela mensagem *COMMAND_ACK*, indicando o sucesso na execução do comando. Além disso, o acionamento dos propulsores foi observado visualmente no laboratório para confirmar o comportamento esperado do veículo. O código utilizado para o teste é apresentado a seguir:

Listing 4.9 – Código para ativacao e comando de movimento do BlueROV2

```
1    #Ativando veiculo
2    connection.mav.command_long_send(connection.target_system,
3    connection.target_component,
4    mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM, #enumeracao
5    0, #confirmacao
6    1, #1 para ativar
7    0, #0 para nao forcar armamento
8    0, 0, 0, 0, 0) #parametros ignorados
9
10   #Aguarda sucesso na ativacao
11   ack_msg = connection.recv_match ( type = "COMMAND_ACK" ,
        blocking =True)
12
```

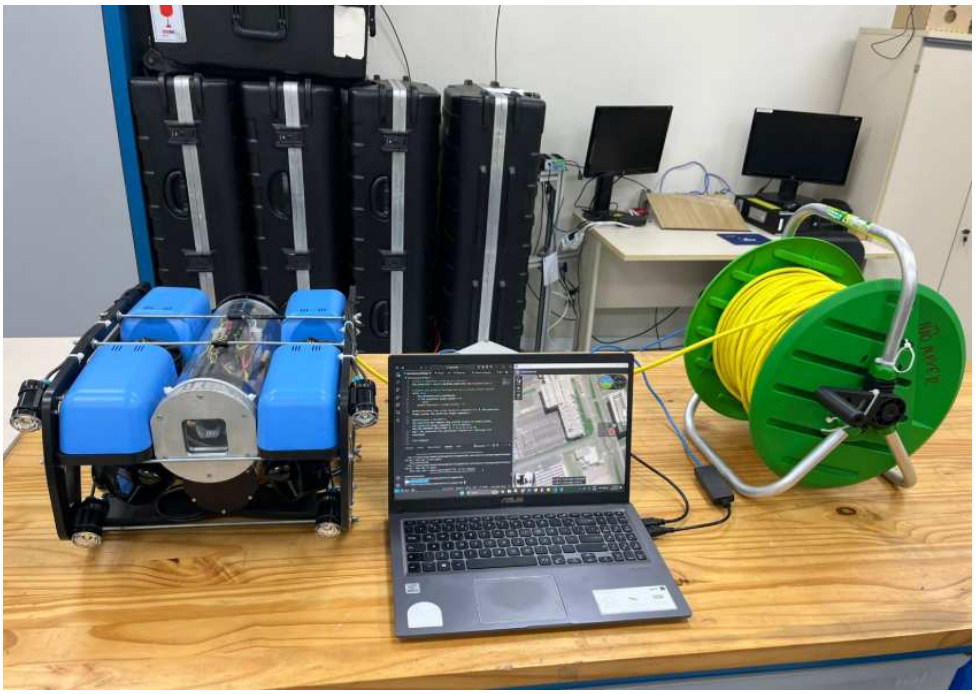
```
13      #Comando para movimento o ROV
14      connection.mav.send(mavutil.mavlink.
           MAVLink_manual_control_message(the_connection.target_system,
15      pitch, #para frente: 0 a 1000 - para tras: -1000 a 0
16      roll, #para direita: 0 a 1000 - para esquerda: -1000 a 0
17      thurst, #propulsao positiva: 0 a 1000 - negativa: -1000 a 0
18      yaw, #rotacao anti horario: 0 a 1000 - horario: -1000 a 0
19      0)) #outros botoes
```

A execução deste código resulta na ativação e no acionamento dos propulsores do veículo, conforme os seguintes comandos:

- Movimento para frente: `pitch: 500, roll: 0, thrust: 500, yaw: 0`
- Movimento para trás: `pitch: -500, roll: 0, thrust: 500, yaw: 0`
- Movimento para a direita: `pitch: 0, roll: 500, thrust: 500, yaw: 0`
- Movimento para a esquerda: `pitch: 0, roll: -500, thrust: 500, yaw: 0`
- Rotação no eixo: `pitch: 0, roll: 0, thrust: 500, yaw: 500`

O valor constante de `thrust: 500` foi configurado para manter o veículo na mesma altitude durante os movimentos (um valor baixo ou alto desta variável faz o veículo perder ou ganhar altitude). Os testes foram realizados em bancada, aplicando pulsos curtos de comando, de modo a preservar a integridade dos propulsores e evitar riscos ao equipamento.

Figura 30 – Teste realizado em bancada com o BlueROV2



Fonte: Imagem capturada pelo autor durante os testes.

4.5 Demonstração e Resultados da Coleta de Telemetria

O protocolo MAVLink envia continuamente dados de telemetria do ROV para a estação de comando, permitindo monitorar seu desempenho em tempo real. Essas mensagens podem ser coletadas utilizando a função `recv_match()` da biblioteca *mavutil*. Conforme apresentado na seção anterior, cada execução da função captura uma única mensagem enviada pelo ROV, e várias delas podem ser obtidas executando a função em *loop*. Para obter maior controle sobre os dados recebidos e tomar decisões mais assertivas com base neles, é possível filtrar tanto as mensagens quanto os campos específicos de cada mensagem.

Após uma análise inicial das mensagens recebidas, e com o auxílio de um algoritmo de análise textual para organizar os dados gerados, foi decidido configurar a função para capturar os seguintes parâmetros:

- `voltage_battery`: carga atual da bateria em milivolts.
- `current_battery`: corrente atual na bateria em centi-ampères.
- `roll`: inclinação lateral do veículo.
- `pitch`: inclinação longitudinal do veículo.
- `yaw`: grau de rotação em torno do eixo vertical.
- `press_abs`, `press_diff`: valores de pressão absoluta e diferencial.

A escolha dos campos pode variar dependendo da aplicação e dos objetivos específicos do projeto. Em algumas situações, a integração de novos sensores pode ser necessária para complementar os dados de telemetria disponíveis. O código abaixo demonstra como filtrar as mensagens para exibir apenas os campos de interesse, utilizando o método `get_type` para identificar o tipo da mensagem:

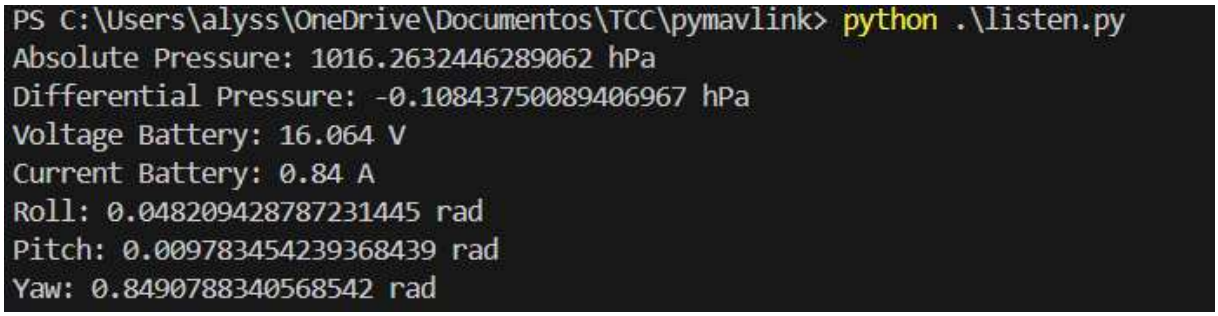
Listing 4.10 – Código de coleta de dados de telemetria filtrados

```
1   while True:
2       telemetry = connection.recv_match(blocking=True)
3
4       if telemetry.get_type() == "SYS_STATUS":
5           print(f"Voltage Battery: {telemetry.voltage_battery / 
                1000.0} V")
6           print(f"Current Battery: {telemetry.current_battery / 
                100} A")
7
8       elif telemetry.get_type() == "ATTITUDE":
9           print(f"Roll: {telemetry.roll} rad")
10          print(f"Pitch: {telemetry.pitch} rad")
11          print(f"Yaw: {telemetry.yaw} rad")
12
13      elif telemetry.get_type() == "SCALED_PRESSURE":
```

```
14                print(f"Absolute Pressure: {telemetry.press_abs} hPa")
15                print(f"Differential Pressure: {telemetry.press_diff} hPa")
```

A execução do código acima é ilustrada na Figura 31. É possível observar que as mensagens são exibidas na ordem *SCALED_PRESSURE*, *ATTITUDE* e *SYS_STATUS*. Essa ordem pode variar conforme os pacotes chegam à estação de comando, dependendo da prioridade e do intervalo de envio de cada mensagem pelo ROV.

Figura 31 – Execução do código de filtragem de dados de telemetria



Fonte: Captura de tela realizada pelo autor no terminal de comando.

Com a coleta dos dados de telemetria bem-sucedida, foi possível validar a comunicação entre o ROV e a estação de comando. Esses dados demonstram o potencial do MAVLink para monitoramento em tempo real para operações veículos não tripulados.

A próxima seção explora a coleta e o processamento de imagens utilizando a câmera do BlueROV2, com foco na detecção de marcadores Aruco e na integração desses dados para controle do veículo.

4.6 Coleta de Imagem e Detecção de Marcadores Aruco

A última etapa do trabalho consistiu em acessar a câmera do BlueROV2 e implementar um algoritmo de visão computacional utilizando o OpenCV. O objetivo foi testar a integração entre o processamento de imagem e o movimento do veículo, configurando-se como uma aplicação prática e adicional do sistema. Foram utilizados marcadores ArUco disponíveis no laboratório onde encontrava-se o veículo.

A seguir, serão abordados os procedimentos para configuração do veículo e o código utilizado para o processamento de imagens.

4.6.1 Configuração do acesso à câmera

O primeiro passo foi compreender como o vídeo é transmitido pelo BlueROV2. O *software* Companion, executado na placa Raspberry Pi do veículo, gerencia a

câmera e transmite o sinal de vídeo pela porta UDP `5600` utilizando o GStreamer[3]. Portanto, foi necessário modificar a configuração do Companion para permitir que o GStreamer enviasse o sinal de vídeo para duas portas UDP: uma para ser utilizada pelo QGroundControl e outra para ser acessada diretamente pelo código de processamento de imagem.

O acesso à placa foi realizado por meio do protocolo SSH, utilizando o endereço IP do ROV: `192.168.2.2`. O arquivo de configuração responsável pela transmissão de vídeo, `start_video.sh`, foi editado com o editor de texto `nano`. O código original do GStreamer, que enviava o vídeo apenas para a porta `5600`, foi alterado para incluir uma segunda porta, `4777`. O comando original era:

Listing 4.11 – Configuração Original do GStreamer

```
1  gst-launch-1.0 -v fdsrc ! h264parse ! rtph264pay config-
       interval=10 pt=96 !
2  udpsink host=192.168.2.1 port=5600
```

A configuração foi modificada para:

Listing 4.12 – Configuração Modificada do GStreamer

```
1  gst-launch-1.0 -v fdsrc ! h264parse ! rtph264pay config-
       interval=10 pt=96 !
2  multiudpsink clients=192.168.2.1:5600,192.168.2.1:4777
```

Após a alteração e reinicialização do veículo, o sinal de vídeo da câmera do BlueROV2 passou a ser transmitido simultaneamente para as portas `5600` e `4777`, permitindo que tanto o QGroundControl quanto o código de processamento pudessem acessar o vídeo.

4.6.2 Detecção de distância utilizando marcadores Aruco

Por ser uma biblioteca de visão computacional *open-source*, o OpenCV possui uma ampla gama de exemplos e aplicações disponíveis, incluindo uma documentação específica para detecção de marcadores ArUco[4]. Esses exemplos serviram como base para o desenvolvimento dos códigos desta seção.

Após configurar o acesso à câmera, a transmissão do vídeo foi acessada pela porta UDP utilizando um arquivo *Session Description Protocol (SDP)*, que descreve o caminho do fluxo de dados do vídeo. Para isso, foi utilizada a biblioteca *python-vlc*[5], que integra funcionalidades do reprodutor de mídia VLC[6] com a linguagem Python.

[3] Mais informações disponíveis em: https://gstreamer.freedesktop.org/
[4] Mais informações disponíveis em: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
[5] Mais informações disponíveis em: https://pypi.org/project/python-vlc/
[6] Mais informações disponíveis em: https://www.videolan.org/

O arquivo SDP criado para ler o vídeo na porta `5600`, considerando as características descritas pelo GStreamer, foi o seguinte:

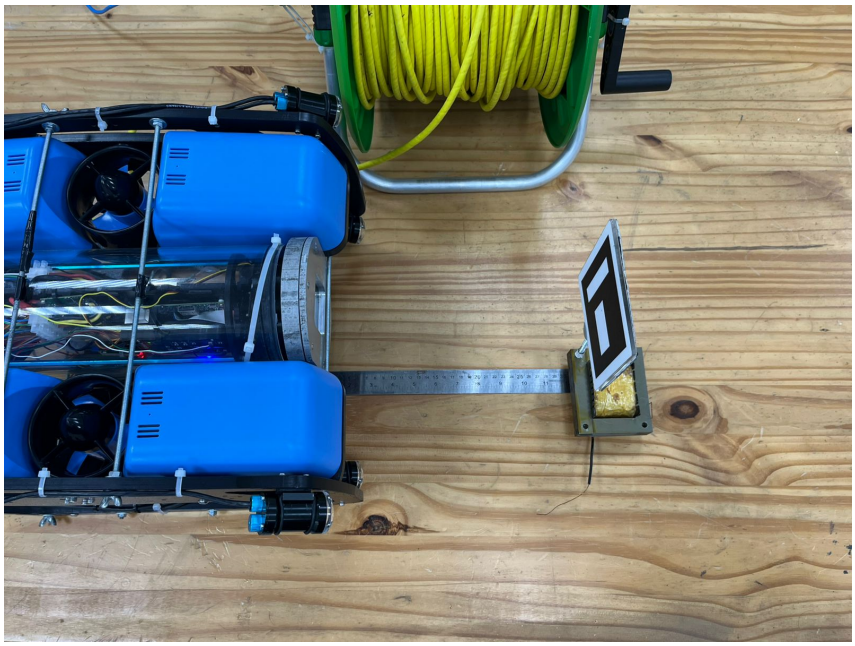
Listing 4.13 – Arquivo SDP para acesso à porta UDP 5600

```
v=0
o=- 0 0 IN IP4 192.168.2.2
s=Stream
c=IN IP4 192.168.2.2
t=0 0
m=video 5600 RTP/AVP 96
a=rtpmap:96 H264/90000
```

Com a configuração concluída, o VLC foi utilizado para acessar o fluxo de vídeo e salvar o primeiro *frame*. Esse *frame* foi então carregado pela função `cv2.imread()` do OpenCV e processado utilizando a biblioteca específica para marcadores ArUco. Além de identificar os marcadores, essa biblioteca é capaz de estimar a distância entre a câmera e o marcador após passar por um processo de calibração. No experimento, utilizou-se uma matriz de calibração ideal, que desconsidera erros de distorção da câmera, permitindo uma estimativa aproximada da distância.

Para o teste, um marcador ArUco foi posicionada a 30 centímetros da câmera, e o código foi executado para capturar e analisar a imagem. A configuração do experimento pode ser visualizada na Figura 32, enquanto o resultado do processamento é mostrado na Figura 33.

Figura 32 – Configuração experimental com marcador ArUco

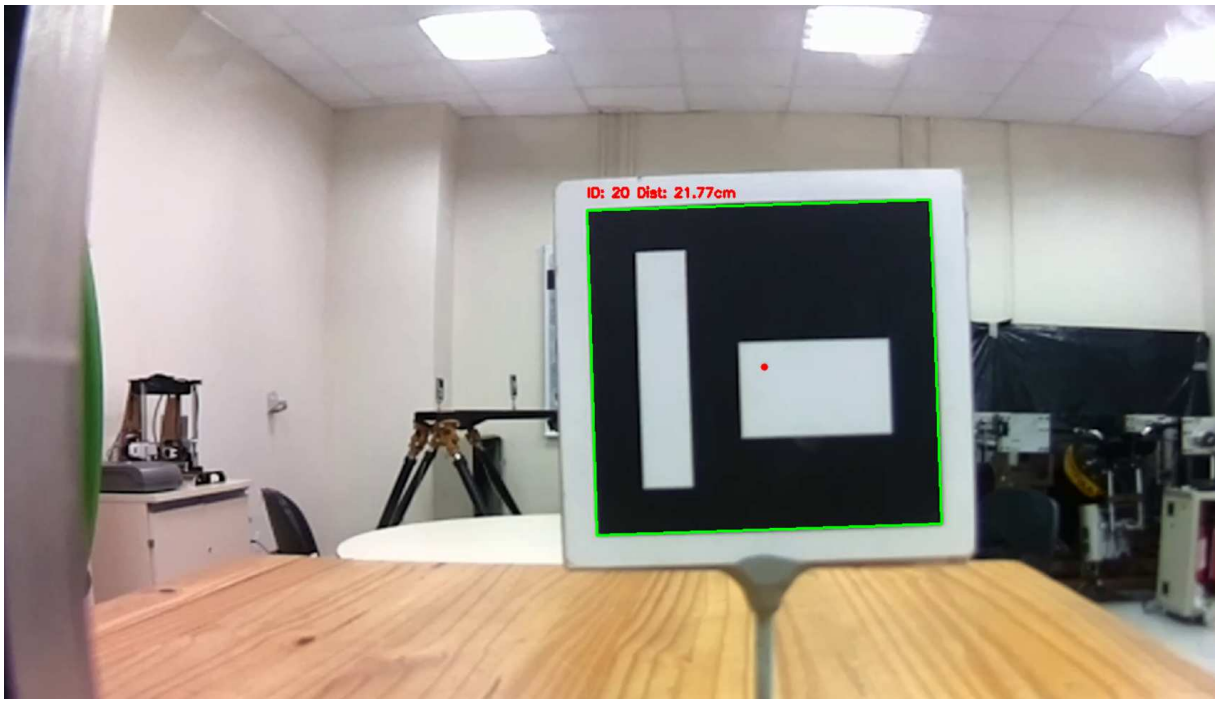


Fonte: Imagem capturada pelo autor.

A Figura 33 apresenta o ID do marcador ArUco detectada pelo programa e

a distância estimada de 21,77 centímetros. O erro de 8,23 centímetros em relação à posição real do marcador evidencia a importância de um processo de calibração adequado para aplicações que demandem alta precisão.

Figura 33 – Resultado da detecção de marcador ArUco com erro de distância



Fonte: Imagem capturada pelo autor.

Pensando em aplicações de navegação autônoma com o BlueROV2, o código desenvolvido para detecção da distância pode ser integrado ao comando MAVLink *MANUAL_CONTROL*, testado anteriormente, para aproximar o veículo do marcador ArUco de forma autônoma. Os códigos utilizados para este experimento, bem como os testes realizados e o processamento de imagem, serão disponibilizados no repositório GitHub do autor para consulta e replicação.

5 CONCLUSÕES

Este capítulo apresenta as considerações finais deste trabalho, destacando as principais contribuições e limitações encontradas, além de propor sugestões para futuros desenvolvimentos e pesquisas na área.

5.1 Considerações finais

Este trabalho explorou a aplicação do protocolo MAVLink no controle do BlueROV2, visando iniciar a exploração de algoritmos de navegação autônoma para o veículo. Foi desenvolvida uma biblioteca de códigos para controle do veículo e coleta de dados de telemetria, validada por meio de testes realizados tanto em ambiente de simulação quanto em laboratório, na Universidade Federal de Santa Catarina - Campus Joinville.

Os resultados obtidos foram satisfatórios, comprovando a eficácia da biblioteca desenvolvida para executar comandos básicos de movimento no ROV. As funções de coleta de dados de telemetria mostraram-se ferramentas úteis para o monitoramento em tempo real do veículo, permitindo o acompanhamento de variáveis relevantes para suas missões. Além disso, o algoritmo de visão computacional com marcadores ArUco demonstrou-se promissor como base para futuras aplicações que integrem essa tecnologia à navegação autônoma.

Como contribuição, este trabalho fornece uma base sólida para o uso do protocolo MAVLink em projetos similares, servindo como ferramenta de consulta e aprendizagem. Por abordar aspectos técnicos fundamentais, como comunicação entre dispositivos, configuração de *softwares* e exemplos de códigos, ele pode ser um ponto de partida para quem deseja se aprofundar na área. Além disso, a biblioteca desenvolvida será disponibilizada no repositório GitHub do autor[1] para a comunidade acadêmica e demais interessados.

Esses resultados destacam a relevância do protocolo MAVLink em aplicações subaquáticas e estabelecem uma base para o desenvolvimento de soluções mais avançadas, que possam ampliar as capacidades de veículos não tripulados.

5.2 Desafios e Limitações Encontradas

Um dos principais desafios enfrentados ao lidar com tecnologias *open-source*, especialmente aquelas em fase de desenvolvimento, é garantir a compatibilidade entre as ferramentas utilizadas. Durante este trabalho, problemas foram encontrados

[1] Repositório GitHub do Autor: https://github.com/alyshowoliveira12/bluerov2-TCC/

na aquisição de imagens da câmera do BlueROV2 devido ao uso de uma versão mais antiga do *software* Companion. Embora o GStreamer, parte da arquitetura de *software* do ROV, seja amplamente recomendado em tutoriais, não foi possível encontrar compatibilidade completa entre todos os recursos necessários, fazendo necessário o uso alternativo do VLC.

Além disso, o tempo investido para solucionar essa questão inviabilizou o desenvolvimento de um algoritmo de calibração da câmera, resultando em imprecisões na detecção de distância com marcadores ArUco. Embora essas imprecisões sejam aceitáveis para testes preliminares, destaca-se a importância da calibração adequada para aplicações que requeiram maior precisão.

Por fim, um defeito no cabo da bateria do veículo exigiu reparos e impossibilitou testes submersos. Dessa forma, todos os experimentos foram realizados em laboratório, o que permitiu validar as funcionalidades desenvolvidas, mas limitou a análise prática em condições reais. Apesar disso, o trabalho demonstrou avanços significativos, oferecendo bases sólidas para trabalhos futuros, como detalhado na próxima seção.

5.3 Trabalhos futuros

Este trabalho abre diversas oportunidades de pesquisa e desenvolvimento nas áreas de drones, ROVs e AUVs. Algumas sugestões para trabalhos futuros incluem:

- **Teste submerso**: realizar experimentos em ambiente aquático para validar a execução dos comandos de movimento do ROV em condições reais, complementando os testes realizados no BlueSim e em laboratório.
- **Calibração avançada da câmera**: implementar algoritmos de calibração que considerem a distorção óptica da câmera, visando aumentar a precisão na detecção de marcadores ArUco e na estimativa de distâncias.
- **Planejamento de missão em modo guiado**: explorar o modo guiado, utilizando sinais GPS para navegação em superfície, ou integrar tecnologias como DVL (*Doppler Velocity Log*), sonares e filtros de Kalman para missões submersas.
- **Integração com inteligência artificial**: desenvolver algoritmos de aprendizado de máquina para reconhecimento de objetos, permitindo que o AUV execute missões autônomas, como evitar colisões, coletar amostras ou realizar operações de recuperação.

Como destacado no início deste trabalho, conhecemos mais sobre a superfície de Marte do que sobre o fundo dos oceanos da Terra. Essa realidade reflete o vasto potencial de exploração subaquática, onde ROVs e AUVs desempenham um papel crucial em expandir nosso conhecimento e realizar aplicações práticas. Os resultados obtidos, as contribuições realizadas e as oportunidades futuras aqui apresentadas ressaltam a relevância e o impacto do tema explorado neste trabalho.

REFERÊNCIAS

ArduPilot Development Team. **ArduPilot Developer Documentation**. 2024. Acesso em 16 de Out. de 2024. Disponível em: https://ardupilot.org/dev/index.html.

ArduSub Development Team. **ArduSub Documentation**. 2024. Acesso em 16 de Out. de 2024. Disponível em: https://www.ardusub.com/.

BAE, I.; HONG, J. Survey on the developments of unmanned marine vehicles: intelligence and cooperation. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 23, n. 10, p. 4643, 2023.

Blue Robotics. **BlueROV2 Datasheet - Rev 2022-R4ROV**. 2022. Accesso em 16 de Out. de 2024. Disponível em: https://www.dropbox.com/scl/fi/q0p579cn3131yqdtd9pv5/br_bluerov2_datasheet_rev2022-R4ROV.pdf?rlkey=5h6mavui1s7y4ctu4s5w1c1im&e=1&dl=0.

Blue Robotics. **BlueOS: Cloud Platform for ROVs**. 2024. Accesso em 26 de Nov. de 2024. Disponível em: https://blueos.cloud/.

Blue Robotics. **BlueROV2 Product Page**. 2024. Accesso em 7 de Nov. de 2024. Disponível em: https://bluerobotics.com/store/rov/bluerov2/.

Blue Robotics. **T200 Thruster**. 2024. Acesso em 29 de Out. de 2024. Disponível em: https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/.

CHRIST, R.; WERNLI, R. **The ROV Manual: A User Guide for Remotely Operated Vehicles**. Butterworth-Heinemann, 2013. ISBN 9780080982915. Disponível em: https://books.google.com.br/books?id=c8ix1dHSx5kC.

GRAÇAS CORRÊA, F. das. Inovação em tecnologias submarinas na exploração oceânica: passado e presente. **Revista InterAção**, v. 12, n. 2, p. 13–27, 2021.

Hellenic Centre for Marine Research. **ROV Types**. 2020. Accessed: 2023-11-7. Disponível em: https://www.eurofleets.eu/wp-content/uploads/2020/07/ROV-Types.pdf.

KOUBÂA, A. et al. Micro air vehicle link (mavlink) in a nutshell: A survey. **IEEE Access**, IEEE, v. 7, p. 87658–87680, 2019.

MAVLink Development Team. **MAVLink Protocol Documentation**. 2024. Accesso em 15 de Nov. de 2024. Disponível em: https://mavlink.io/en/.

MOORHOUSE, P. A modern history of the manned submersible. **Marine Technology Society Journal**, Marine Technology Society, v. 49, n. 6, p. 65–78, 2015.

NHAM, K. M. Q. et al. **Nyuad robosub: Automated underwater vehicle adaptation and control**. [S.l.]: California: Team Inspiration. https://robonation. org/app/uploads/sites/4 . . . , 2022.

NICHOLSON, J.; HEALEY, A. The present state of autonomous underwater vehicle (auv) applications and technologies. **Marine Technology Society Journal**, Washington, DC: Marine Technology Society, v. 42, n. 1, p. 44–51, 2008.

QAYS, H. M.; JUMAA, B. A.; SALMAN, A. D. Design and implementation of autonomous quadcopter using sitl simulator. **IRAQI JOURNAL OF COMPUTERS, COMMUNICATIONS, CONTROL AND SYSTEMS ENGINEERING**, University of Technology, v. 20, n. 1, p. 1–15, 2020.

QGroundControl Documentation. **QGroundControl User Guide**. 2024. Accesso em 16 de Out. de 2024. Disponível em: https://docs.qgroundcontrol.com.

RoboSub. **Why RoboSub?** 2023. Accesso em 31 de Out. de 2024. Disponível em: https://robosub.org/why-robosub/.

ROS Blog. **Why ROS? - ROS.org**. 2023. Accesso em 31 de Out. de 2024. Disponível em: https://www.ros.org/blog/why-ros/.

ROS Wiki. **MAVROS - ROS Wiki**. 2023. Accesso em 31 de Out. de 2024. Disponível em: https://wiki.ros.org/mavros.

SZETO, C. et al. **The Design of Team Inspiration's 2020 AUVs**. [S.l.]: California: Team Inspiration. https://robonation. org/app/upload s/sites/4 . . . , 2020.

WIBISONO, A. et al. A survey on unmanned underwater vehicles: Challenges, enabling technologies, and future research directions. **Sensors**, MDPI, v. 23, n. 17, p. 7321, 2023.

WYNN, R. B. et al. Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience. **Marine Geology**, v. 352, p. 451–468, jun. 2014. ISSN 00253227. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0025322714000747.

LE XUAN, T. et al. Communication and control for remotely operated underwater vehicles. In: HUYNH, D. V. K. et al. (Ed.). **Proceedings of the 2nd Vietnam Symposium on Advances in Offshore Engineering**. Singapore: Springer Singapore, 2022. p. 216–221. ISBN 978-981-16-7735-9.