UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Sistemas de Informação
Patrick de Paula Barroso

CUSTOMIZAÇÃO DE UM AMBIENTE DE SIMULAÇÃO VIRTUAL BASE UTILIZANDO AS TECNOLOGIAS ROS2 E WEBOTS PARA SIMULAR MÚLTIPLOS VANTS

Diamantina
2024

Patrick de Paula Barroso

CUSTOMIZAÇÃO DE UM AMBIENTE DE SIMULAÇÃO VIRTUAL BASE UTILIZANDO AS TECNOLOGIAS ROS2 E WEBOTS PARA SIMULAR MÚLTIPLOS VANTS

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação, como parte dos requisitos exigidos para a conclusão do curso.

Orientador: Prof. Dr. Rafael Santin

Diamantina
2024

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Patrick de Paula Barroso

CUSTOMIZAÇÃO DE UM AMBIENTE DE SIMULAÇÃO VIRTUAL BASE UTILIZANDO AS TECNOLOGIAS ROS2 E WEBOTS PARA SIMULAR MÚLTIPLOS VANTS

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisitos parcial para conclusão do curso.

Orientador: Prof. Dr. Rafael Santin

Aprovado em 4 de julho de 2024

BANCA EXAMINADORA

Prof. Dr. Alessandro Vivas Andrade
Faculdade de Ciências Exatas - DECOM - UFVJM

Profª. Drª. Luciana Pereira de Assis
Faculdade de Ciências Exatas - DECOM - UFVJM

Prof. Dr. Rafael Santin
Faculdade de Ciências Exatas - DECOM - UFVJM

Documento assinado eletronicamente por **Rafael Santin**, **Servidor (a)**, em 11/07/2024, às 18:38, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Luciana Pereira de Assis**, **Servidor (a)**, em 12/07/2024, às 16:20, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Alessandro Vivas Andrade**, **Servidor (a)**, em 12/07/2024, às 19:05, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1473624** e o código CRC **05464306**.

Referência: Processo nº 23086.006867/2024-57 SEI nº 1473624

RESUMO

Veículos Aéreos Não Tripulados (VANTs) são utilizados em diversos setores da sociedade, graças às inúmeras inovações tecnológicas das últimas décadas, que tornaram os VANTs mais acessíveis e versáteis. Ferramentas como o *Robot Operating System* (ROS) e seu sucessor, ROS2, são essenciais no desenvolvimento de VANTs. Essas plataformas de software livre oferecem componentes fundamentais para a robótica, facilitando o desenvolvimento de controles para robôs, como os VANTs, e sendo amplamente utilizadas na indústria. Juntamente de plataformas de simulação virtual, como Webots e Gazebo, permitem testes em ambientes virtuais com grande fidelidade ao mundo real. A realização desses testes reduz custos e facilitam a validação de funcionalidades, minimizando desperdícios e riscos em testes reais. O objetivo do presente trabalho é adaptar um modelo de simulação base para um único drone para que ele seja capaz de simular múltiplos drones, com quantidade escolhida pelo usuário, e possibilitar que cada drone na simulação seja controlado separadamente.

Palavras-chave: ROS2, VANTS, DRONES, WEBOTS, SIMULAÇÃO

ABSTRACT

Unmanned Aerial Vehicles (UAVs) are used in various sectors of society, thanks to numerous technological innovations of recent decades, which have made UAVs more accessible and versatile. Tools such as the Robot Operating System (ROS) and its successor, ROS2, are essential in the development of UAVs. These open-source software platforms provide fundamental components for robotics, facilitating the development of controls for robots, such as UAVs, and are widely used in the robotic industry. Alongside with virtual simulation platforms like Webots and Gazebo, they allow testing of robots in virtual environments with great fidelity to the real world. Conducting these tests reduces costs and facilitates the validation of functionalities, minimizing waste and risks in real tests. The objective of the present work is to adapt a simulation model based on a single drone so that it can simulate multiple drones, with the quantity chosen by the user, and enable each drone in the simulation to be controlled separately.

Keywords: ROS2. UAVS. WEBOTS. SIMULATION.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Atualmente, o escopo de utilização de Veículos Aéreos Não Tripulados (VANTs) abrange diversos setores da sociedade, tais como agricultura, segurança pública, missões de resgate, monitoramento ambiental, logística e entretenimento. Essa diversidade é possível graças a um longo histórico de inovações tecnológicas nas últimas décadas que permitiram a produção de VANTs cada vez mais acessíveis e versáteis que atraem diversos investimentos para pesquisas relacionadas ao seu desenvolvimento. Com grande capacidade de coleta de dados e facilidade em acessar áreas antes de difícil acesso ao ser humano, os VANTs estão se consolidando como ferramentas essenciais nesses setores, impulsionando a eficiência e melhorando os resultados das operações em que são empregados (TELLI *et al.*, 2023).

Nesse sentido, é cada vez mais importante a utilização de ferramentas que auxiliem o desenvolvimento de VANTs. O *Robot Operating System (ROS)* e seu sucessor *Robot Operating System 2 (ROS2)* são conjuntos de ferramentas de software livre que oferecem todos os componentes essenciais para o desenvolvimento de aplicações de robótica. Com menos de duas décadas de existência, o *ROS* se popularizou e teve seu sucessor lançado em 2022 com o objetivo de corrigir suas limitações de segurança e escalabilidade. Ambas versões são amplamente utilizados em toda a indústria robótica e sua aplicação é relevante desde o uso em projetos individuais a grandes projetos corporativos. Dentre as inúmeras aplicações que são possíveis desenvolver com o uso dessa tecnologia, se destacam aplicações para o controle de VANTs, visto a facilitação que a abstração trazida pelo *ROS* e *ROS2* trás no desenvolvimento de controles para robôs complexos (MACENSKI *et al.*, 2023).

Além disso, plataformas de simulação virtual são ferramentas indispensáveis no desenvolvimento de VANTs. Isso porque os custos relacionados a criação e obtenção de VANTs são geralmente altos e os ambientes de teste no mundo real devem respeitar diversos requisitos, o que restringe os testes no que tange a viabilidade de locais e profissionais operadores (CHEN *et al.*, 2023). Simuladores como *Webots* e *Gazebo* possibilitam que robôs sejam testados em ambientes virtuais com grande fidelidade ao mundo real graças a seus motores de física e renderizadores gráficos. Ambos simuladores possuem diversos componentes que auxiliam na construção das simulações, tais como bibliotecas de elementos prontos como modelos tridimensionais de robôs, sensores, materiais e objetos. Isso possibilita grande customização e variedade de aplicações. A utilização de simuladores virtuais permite, então, que os robôs sejam testados sem as restrições do mundo real, o que agiliza a detecção de comportamentos indesejados e auxilia a validação de seu funcionamento (MICHEL, 2004).

Em conjunto, ferramentas de desenvolvimento de software para robôs e ambientes de simulação possuem grande potencial, visto que todo o desenvolvimento do software relacionado ao controle do robô pode ser feito usando a interface do *ROS2* e a simulação, teste e validação podem ser feitos com simuladores como *Webots* e *Gazebo*. Isso garante que quando o software for eventualmente testado em VANTs no mundo real, as chances de desperdício de recursos, prejuízo financeiro ou ocorrência de fracasso sejam minimizadas.

Nesse contexto, o objetivo principal deste trabalho é customizar o modelo de simulação base para o drone *Mavic 2 PRO* integrado ao ROS2 e disponibilizado pela *Cyberbotics*, organização criadora e responsável pelo simulador *Webots*, para comportar a simulação de múltiplos VANTs controlados individualmente.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Customizar o ambiente de simulação virtual base para um único drone com o objetivo de permitir ao usuário a simulação de múltiplos drones controlados individualmente.

1.1.2 OBJETIVOS ESPECÍFICOS

- Possibilitar ao usuário da simulação escolher de forma simples, no início da execução, a quantidade de drones desejados para o cenário simulado.
- Permitir que o usuário consiga controlar os drones da simulação individualmente, de forma separada.

1.2 ORGANIZAÇÃO

No capítulo 1, é introduzida a proposta do trabalho e seus objetivos. Já no capítulo 2, é realizado um estudo com relação aos temas pertinentes ao contexto do ambiente de simulação virtual desenvolvido no trabalho, tais como os VANTs e as ferramentas utilizadas, sendo elas o Robot *Operating System (ROS)* e o simulador *Webots*. No capítulo 3, é detalhada a metodologia utilizada para o desenvolvimento do ambiente de simulação, além das motivações por trás das escolhas realizadas para construção do trabalho. No capítulo 4, é descrita a jornada de customização da simulação e o funcionamento do sistema desenvolvido. Por fim, no capítulo 5, são apresentadas as conclusões obtidas e sugeridas ideias de trabalhos futuros.

2 REVISÃO DE LITERATURA

Este capítulo aborda assuntos relacionados ao desenvolvimento da simulação apresentada pelo presente trabalho. Inicialmente, são explorados os VANTs, destacando suas aplicações, história e importância para a sociedade humana. Em seguida, discutimos o *Robot Operating System 2 (ROS2)*, um *framework* flexível que facilita a comunicação e integração entre diferentes componentes robóticos. Posteriormente, são examinadas plataformas de simulação, em especial o Gazebo e Webots, opções avaliadas para construção do presente trabalho.

2.1 VEÍCULOS AÉREOS NÃO TRIPULADOS

Os Veículos Aéreos Não Tripulados (VANTs), ou simplesmente drones, tem uma história que remonta ao século XIX, quando os primeiros veículos aéreos não tripulados foram utilizados pelos austríacos. Nessa ocasião, balões cheios de explosivos foram empregados como bombas, marcando o início do uso de veículos aéreos sem piloto a bordo (KARDASZ1 *et al.*, 2016). Porém, de acordo com Tice (2023) foi no século XX que os VANTs tomaram força, de início em operações militares visto que, por não ter tripulantes, eram mais adequados para missões perigosas. Somente na década de 1980 foi produzido o primeiro drone civil. De imediato, já haviam diferenças significativas entre os drones civis e os militares em termos de tamanho e propulsão, sendo geralmente menores e movidos por motores elétricos, enquanto os militares usam motores de combustão interna (KARDASZ1 *et al.*, 2016).

A definição atual de VANTs, conforme a Organização Internacional da Aviação Civil ICAO (2011), é simplesmente uma aeronave operada sem um piloto humano a bordo. O avanço da Indústria 4.0 aumentou significativamente as capacidades dos drones, expandindo ainda mais suas aplicações. Recentemente, a aplicação de drones observou um rápido crescimento durante a pandemia global de *COVID-19*. Devido às medidas restritivas de distanciamento social, grandes cadeias de varejo e empresas de entrega de pacotes foram forçadas a melhorar suas operações logísticas. Além disso, várias empresas online começaram a oferecer entregas no mesmo dia, aumentando as expectativas dos clientes por entregas rápidas. Empresas como *Amazon*, *DHL* e *FedEx* têm adotado drones para a entrega na última milha (*LMD*), visando entregas mais rápidas e eficazes, além de aumentar a lucratividade (LI *et al.*, 2023).

De acordo com Telli *et al.* (2023) a demanda por VANTs reflete sua capacidade de executar a ampla gama de tarefas e operações que podem realizar, desde atividades cotidianas a operações militares e de resgate. Nesse sentido, é possível observar o grande impacto causado pelos VANTs em múltiplas áreas da sociedade.

2.2 ROBOT OPERATING SYSTEM

De acordo com Macenski *et al.* (2022) o objetivo de um *framework* de Robótica é fornecer uma arquitetura que decomponha softwares complexos de controle de robôs em partes menores e mais fáceis de gerenciar, promovendo com isso reusabilidade de componentes, eficiência de execução e manutenção, alta escalabilidade e boas práticas de desenvolvimento.

Diversas tentativas de desenvolver uma plataforma desse tipo foram promovidas nas últimas décadas, em especial o *Robot Operating System (ROS),* criado em 2007, que é

considerado o maior ecossistema de robótica e de código aberto disponível para utilização (MACENSKI *et al.*, 2022). Porém, de acordo com Maruyama, Kato e Azumi (2022), apesar de proporcionar um grande aumento na produtividade de desenvolvimento de aplicações para robôs graças à seu sistema de comunicação de pacotes editor/assinante e múltiplas bibliotecas de componentes, o *ROS* ainda carece de requisitos essenciais para utilização em robôs de verdade. Em especial, o sistema de comunicação do *ROS* depende de um processo mestre, o que diminui a tolerância a falhas de um sistema desenvolvido com essa ferramenta. Além disso, o *ROS* não é um *framework* multiplataforma, pois só pode ser usado em computadores *Linux*. De acordo com Hasnain e Rafi (2019) com base em coletas de formulário com comunidades de desenvolvedores de software, apenas cerca de 1/4 dos usuários preferem máquinas com sistema operacional Linux para trabalhar em projetos de desenvolvimento de software. Com isso, não ser multiplataforma reduz a abrangência do *ROS* e dificulta sua difusão e usabilidade por parte da maioria dos usuários.

Diante das limitações do *ROS*, foi desenvolvido, então, o *Robot Operating System 2 (ROS2),* cuja proposta é continuar o legado do *ROS* como poderoso *framework* de desenvolvimento de aplicações para robôs, mas também trazer correções e tratativas para suas principais falhas com relação à segurança, confiabilidade e usabilidade. O *ROS2* trás um novo sistema de comunicação baseado em *Data Distribution Service (DDS)*, que é ideal para sistemas críticos devido a sua variedade de serviços para comunicação de componentes. Ao invés de depender de um processo mestre, como seu antecessor, o sistema de comunicação do *ROS2* possui uma camada de abstração que lida com envio e recebimento de mensagens de forma descentralizada, o que torna o sistema mais confiável e resistente à falhas. Além disso, o *ROS2* foi desenvolvido para ser um framework multiplataforma, podendo ser usado em computadores com os sistemas operacionais *Linux*, *Windows* e *Mac*, o que facilita sua difusão na comunidade de desenvolvimento de software (MACENSKI *et al.*, 2022). A diferença entre o *ROS* e o *ROS2* no que tange a sua arquitetura descentralizada pode ser observada na Figura 1.

2.3 PLATAFORMAS DE SIMULAÇÃO VIRTUAL DE ROBÔS

A utilização de simulações virtuais é uma prática valiosa no desenvolvimento de novos algoritmos e funcionalidades para robôs, antes da construção e execução de códigos em sistemas robóticos físicos. Isso porque é possível validar o comportamento dos robôs em uma grande variedade de cenários simulados e identificar falhas rapidamente e sem prejuízos, dado que o custo de simular robôs virtualmente é mínimo comparado ao teste em robôs físicos (FARLEY; WANG; MARSHALL, 2022).

Nos últimos anos, diversas plataformas de simulação virtual para robôs foram desenvolvidas e disponibilizadas para utilização de forma gratuita, por exemplo, *Webots*, *Gazebo*, *Simbad*, *CoppeliaSim*, entre outras. Essas plataformas fornecem diversas características cruciais para o teste de funcionalidades de robôs em ambientes virtuais, tais como bibliotecas imbutidas, motores físicos sofisticados e precisos e gráficos agradáveis onde é possível visualizar a simulação (CHEN *et al.*, 2023).

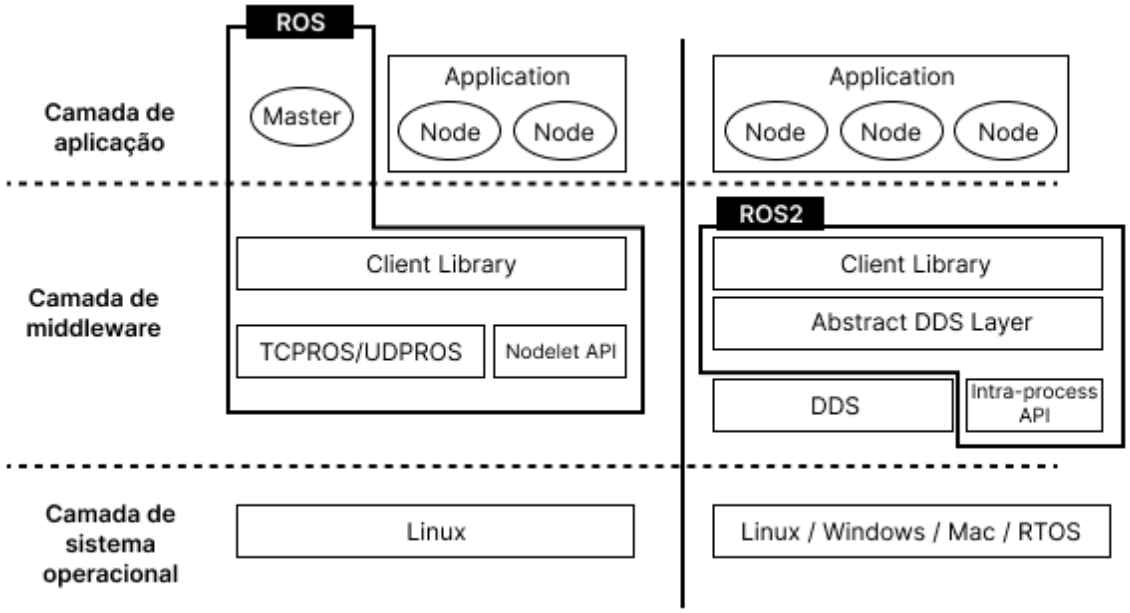

Figura 1 – Comparação das arquiteturas do *ROS* e *ROS2*

De acordo com Farley, Wang e Marshall (2022), que realizaram uma pesquisa de feedback de usuários, focada em roboticistas que trabalham principalmente com controle de robôs móveis, identificou-se a precisão, o código aberto e a capacidade de usar o mesmo código tanto para robôs reais quanto simulados como os critérios mais importantes na escolha de um simulador e a qualidade da visualização e a carga computacional foram citadas como critérios secundários. A precisão de um simulador depende principalmente de seu motor de física, que calcula as dinâmicas de contato essenciais para simulações fisicamente precisas.

2.3.1 GAZEBO

O Gazebo é um software avançado de código aberto para construção de simulações, utilizadas no campo da pesquisa e desenvolvimento de robótica. O Gazebo possibilita a construção de simulações de dinâmicas externas com alta fidelidade, sendo possível modelar cenários detalhados ao ar livre com um forte foco em realismo. Sua natureza de código aberto estimula o crescimento de um ecossistema vibrante de contribuidores que continuamente aprimoram a plataforma, adaptando-a para atender às necessidades de pesquisa em constante evolução (KOENING; HOWARD, 2004).

Além disso, Koening e Howard (2004) apontam que o Gazebo se destaca na reprodução de ambientes dinâmicos que os robôs podem encontrar, incorporando atributos como massa, velocidade e fricção para interações realistas com objetos e terrenos. Essa capacidade é crucial para qualquer simulador confiável, pois na maioria dos testes, é necessário que o ambiente virtual seja preciso em termos de física para que os testes sejam válidos.

No entanto, apesar de sua robustez e capacidade de realizar simulações complexas e eficientes, o Gazebo demanda hardware consideravelmente avançado para boa execução. Além

disso, possui suporte a menos linguagens de programação que seus concorrentes no mercado, como *Webots* e *CoppeliaSim*. O *framework* de desenvolvimento de robótica *ROS* é compatível com o Gazebo, porém, possui integração arduosa com a ferramenta, visto que não há conexão nativa entre os dois softwares (FARLEY; WANG; MARSHALL, 2022).

2.3.2 WEBOTS

O *Webots* é um software de código aberto e multiplataforma usado para criação de simulações virtuais de robôs com grande destaque na indústria. Ele provê um ambiente completo para modelar, programar e simular diversos tipos de robôs. Uma das características mais significaticas do Webots é sua portabilidade, visto que possui bibliotecas que permitem a transferência de componentes modelados e testados no simulador para robôs reais (MICHEL, 2004).

Além disso, o *Webots* é altamente compatível com os sistemas operacionais mais populares do mercado, *Linux*, *Windows* e *Mac*, e suporta diversas linguagens de programação no desenvolvimento de suas simulações, tais quais C, C++, Python, Matlab, Java, entre outras. Ele também possui diversos modelos nativos de robôs e objetos, o que permite vasta personalização do cenário e dos robôs simulados, sendo possível submeter os robôs a diversas condições climáticas e fazer com que robôs diferentes interajam no mesmo cenário (CHEN *et al.*, 2023).

3 MATERIAIS E MÉTODOS

O desenvolvimento do ambiente de simulação proposto por esse trabalho teve como primeira etapa a definição de seu escopo. Foi decidido que se trataria de um ambiente virtual de simulação para múltiplos drones usando as ferramentas *ROS2* e *Webots*. O presente trabalho apresenta uma versão inicial, que demanda a instalação do código fonte da simulação e sua execução pelo terminal.

Com isso, se iniciou a etapa de estudos e avaliação das ferramentas, que não são usuais e demandaram tempo para se acostumar. Em especial, o *ROS2*, pois se trata de um *framework* que possui práticas e etapas definidas para construção de operações.

Com o conhecimento aprendido na etapa de estudo, foi realizada a customização do modelo de simulação base para um único drone do modelo *DJI Mavic 5*, com o objetivo de permitir que ele aceite múltiplos drones, com quantidade definida pelo usuário, e para que os drones possam ser controlados independentemente.

3.1 FERRAMENTAS E TECNOLOGIAS

O sistema operacional utilizado para desenvolvimento foi o Ubuntu 22.04 (*Jammy Jellyfish*), um Sistema Operacional de código aberto construído a partir do núcleo GNU/Linux, baseado em Debian e que utiliza GNOME como ambiente gráfico. O hardware utilizado para testes da simulação foi um notebook Lenovo Ideapad S145, com processador AMD Ryzen 5-3500U, placa de vídeo AMD Radeon RX Vega 8, com 8 gigas de memória RAM e SSD Kingston M.2 2280 PCIe.

A ferramenta de desenvolvimento de software para robôs escolhida foi o *Robot Operating System (ROS2)*, dada sua confiabilidade e ampla utilização na indústria robótica. Apesar da curva de aprendizado inicialmente alta, o *ROS2* possui etapas de desenvolvimento bem definidas visto sua natureza como *framework*. O *ROS2*, apesar de seu recente lançamento em 2022, foi escolhido ao invés do *ROS* devido à iminente obsolência do *ROS*, que, devido à suas limitações, terá vida útil menor que o *ROS2*. A escolha do *ROS2* permite futura expansão e escalonamento da simulação para incorporar diversas outras funcionalidades, como salvamento de imagens capturadas pelos drones na simulação, atribuição de algoritmos de controle de rotas para os drones, entre outros.

O ROS2 possui compatibilidade com diversas linguagens de programação, como C, C++, Java, Python, entre outras. Dentre as possibilidades, a linguagem de programação escolhida para o projeto foi o Python, pela sua simplicidade, facilidade de uso, produtividade e grande diversidade de materiais e pacotes já implementados que reduzem a dificuldade em construir aplicações complexas.

Dentre os softwares de simulação virtual, foram considerados para a simulação o *Gazebo* e o *Webots*. Foi escolhido o *Webots* por conta da sua facilidade de instalação, riqueza de componentes nativos e integração com o *ROS2* menos complexa que o *Gazebo*, possibilitada por meio de pacotes disponibilizados pela *Cyberbotics*.

Para versionamento de código, foi utilizado o *Git* e o repositório *GitHub*, onde também estão definidas as instruções para instalação e utilização do ambiente de simulação.

3.2 ESCOLHA DO ROBOT OPERATING SYSTEM 2

O *ROS2*, ou *Robot Operating System 2*, foi escolhido para o projeto de construção de um ambiente de simulação para múltiplos drones por uma série de razões. Em primeiro lugar, sua arquitetura modular e flexível oferece uma base consistente para desenvolver e integrar sistemas complexos como simulações de drones. Com seus diversos componentes, é possível adaptar o sistema às necessidades específicas do projeto, desde a comunicação entre drones até a interação com sensores, atuadores virtuais e controle por meio de algoritmo. Essa característica também permite que o sistema, desenvolvido com o *ROS2*, cresça em complexidade e tamanho sem prejudicar a capacidade de manutenção ou o desempenho da aplicação, características cruciais para sistemas nos dias atuais.

Além disso, a comunidade *ROS2* é bastante ativa, o que proporciona um amplo suporte e uma vasta gama de recursos e pacotes prontos para uso. Há diversos pacotes que encapsulam comportamentos específicos dos robôs, tais como seu deslocamento em um plano tridimensional, o que evita que o desenvolvedor despendie muitos esforços em detalhes que não são relacionados ao contexto em que está trabalhando e possa, com isso, focar seus esforços na resolução do problema em si. Isso não apenas acelera o desenvolvimento do projeto, mas também proporciona o acesso a soluções comprovadas e atualizadas para os desafios encontrados durante a customização do ambiente de simulação.

Além disso, o *ROS2* oferece suporte para comunicação distribuída e escalável, essencial para simular múltiplos drones de forma eficiente. Sua abordagem orientada a mensagens facilita a troca de dados entre os drones simulados e outros componentes do sistema, permitindo uma simulação realista e precisa do comportamento cooperativo dos drones em diferentes cenários. Essa capacidade de escalabilidade é crucial para garantir que a simulação possa crescer em complexidade à medida que o projeto avança, mantendo um desempenho consistente e confiável.

3.3 ESCOLHA DO SIMULADOR

Na busca por um simulador adequado, foram consideradas duas opções amplamente utilizadas na indústria e no meio acadêmico, os softwares *Gazebo* e *Webots*. Ambos os simuladores possuem suas próprias vantagens e desvantagens, tornando a escolha uma decisão crítica para a eficácia e eficiência do projeto.

O *Gazebo* é um simulador de código aberto bastante robusto, amplamente adotado na comunidade de robótica e com credibilidade consolidada. Suas principais vantagens incluem a capacidade de simular ambientes complexos e variados, a compatibilidade com uma vasta gama de sensores e atuadores, e a possibilidade de integração com o *ROS (Robot Operating System),* também amplamente utilizado. Além disso, a grande quantidade de recursos e tutoriais disponíveis torna o *Gazebo* uma escolha atraente para pesquisadores e desenvolvedores. No entanto, sua curva de aprendizado é íngreme para iniciantes, e a configuração inicial pode

ser demorada, especialmente para simulações mais complexas. Isso porque não há abstração da integração entre o *Gazebo* e o *ROS2*, que deve ser feita manualmente e contém diversas dependências e etapas para ser bem sucedida. Outro ponto a considerar é o seu alto consumo de memória, que pode exigir hardware robusto para funcionar de maneira eficiente. Isso ocorre por conta da complexidade dos motores gráficos e de física que ele implementa.

Por sua vez, o *Webots* é um simulador também de código aberto, mas com uma interface mais amigável e intuitiva, apesar de ser menos utilizado que o Gazebo na indústria e ser preferido em meios acadêmicos. Com instalação simples, ele oferece uma vasta biblioteca de modelos de robôs pré-construídos, o que facilita o início rápido dos projetos. Ainda assim, o *Webots* é conhecido por sua capacidade de renderizar gráficos de alta qualidade e fornecer simulações precisas em tempo real, com credibilidade e equivalência suficiente ao mundo real para garantir a qualidade de simulações desenvolvidas com ele. Além disso, a curva de aprendizado do *Webots* é baixa, sendo possível fazer simulações poderosas e complexas sem muito tempo de experiência com a ferramenta. Isso é possível graças a documentação do software, que possui tutoriais simples, diretos, atualizados e que cobrem todas as funcionalidades da ferramenta. Por fim, o Webots possui pacotes, em sua maioria desenvolvidos pela comunidade, que facilitam a utilização do *ROS* nas simulação, como o *webotsros2*, que não apresenta grande complexidade de instalação.

Diante disso e da natureza temporal do presente trabalho, que teve como tempo para desenvolvimento alguns meses, foi decidido o uso do *Webots* devido à facilidade de integração com o *ROS* por meio do pacote *webotsros2* e dos *plugins* já existentes, além da curva de aprendizado menos íngreme se comparada ao *Gazebo*. A biblioteca *webotsros2* é peça chave para a decisão, pois oferece uma ponte eficiente entre o simulador e o *ROS2*, simplificando o desenvolvimento e a experimentação de algoritmos de robótica em um ambiente simulado. Apesar de ainda possuir alto consumo de memória, a facilidade de integração alinhada a qualidade das simulações do *Webots* foram determinantes para a escolha final.

3.3.1 PACOTE WEBOTSROS2

O pacote *webotsros2* é um conjunto de ferramentas que facilita a integração entre o simulador *Webots* e o *ROS 2 (Robot Operating System).* Ele permite que desenvolvedores criem e testem algoritmos de robótica de maneira mais eficiente, simplificando a complexa etapa de configuração presente para muitos simuladores disponíveis, como o *Gazebo*. Essa integração é útil porque os dados dos sensores simulados no *Webots* podem ser facilmente publicados como tópicos *ROS2*, e os comandos de controle gerados por nós *ROS2* podem ser enviados diretamente para os atuadores simulados no *Webots*.

Uma das principais vantagens do *webotsros2* é a presença de interfaces padronizadas para essa comunicação bidirecional. Isso simplifica o desenvolvimento de algoritmos de controle e percepção, permitindo uma experimentação mais rápida e eficiente. Além disso, o repositório inclui uma série de *plugins* pré-construídos que suportam uma ampla gama de robôs e sensores, facilitando o início dos projetos sem a necessidade de criar modelos de simulação do zero.

Os *plugins* disponíveis abrangem desde robôs móveis simples até manipuladores industriais complexos, proporcionando uma base sólida para diversas aplicações de robótica. Outro benefício significativo do *webotsros2* é a documentação extensa e os exemplos fornecidos, que são especialmente úteis para iniciantes. Esses recursos oferecem um ponto de partida claro e instruções detalhadas sobre como configurar e usar o sistema.

4 DESENVOLVIMENTO

Este capítulo apresenta as etapas da construção do ambiente de simulação virtual para múltiplos drones individuais construído no presente trabalho. O projeto tomou como base um modelo de simulação já existente e disponível para utilização e customização e, a partir dele, foram feitas modificações que permitissem a simulação de múltiplos drones independentes.

4.1 AMBIENTE DE SIMULAÇÃO BASE PARA O DRONE MAVIC DJI

O *Webots* permite validar algoritmos de controle de drones, como o *Mavic DJI*. O plugin *webotsros2*, mencionado anteriormente, possui um modelo base para testes com esse drone, que pode ser clonado e posteriormente personalizado graças à característica de código livre da biblioteca.

Chamado de *webotsros2mavic*, a simulação base inclui um modelo detalhado de um único Mavic 2 PRO, um drone popular da DJI. Esse modelo é baseado em um arquivo *URDF (Unified Robot Description Format)* que descreve a geometria, cinemática e dinâmica do drone. O Webots carrega esse modelo e simula o comportamento do drone no ambiente virtual, permitindo uma representação precisa dos movimentos e respostas do Mavic 2 PRO às condições simuladas. Na Figura 2, é mostrado o modelo do Mavic 2 PRO, que conta com suas hélices, sensores, câmera e fidelidade com a aparência de seu modelo físico. Já na Figura 3, uma captura de tela da simulação base, é possível observar o Mavic 2 PRO planando a 1 metro de distância do chão e o ambiente de simulação ao seu redor.

Figura 2 – Modelo tridimensional do drone Mavic 2 PRO

Para controlar o Mavic 2 PRO, o *webotsros2* utiliza um controlador PID (Proporcional-Integral-Derivativo) simples. Esse controlador recebe comandos de velocidade linear e angular (*geometrymsgs/Twist*) e ajusta as velocidades das hélices para manter o drone estável e cumprir os comandos dados. Além disso, o modelo do Mavic inclui sensores simulados,

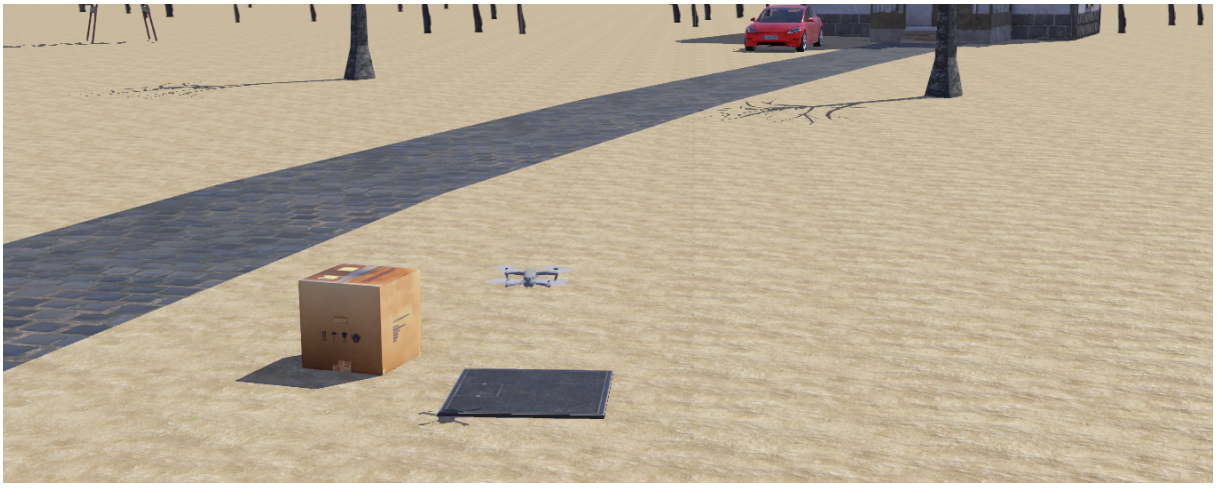

Figura 3 – Modelo de simulação base para um único Mavic 2 PRO disponível pelo *webotsros2*

como câmeras, IMU (Unidade de Medida Inercial) e GPS, que fornecem dados essenciais para a navegação e controle, replicando os sensores reais usados no drone físico.

A interação com o *ROS2* no *webotsros2* é facilitada pela capacidade de publicar comandos de velocidade e receber dados dos sensores do Mavic 2 PRO no ambiente de simulação. É possível também visualizar o drone no *RViz*, um visualizador 3D, para depuração e análise detalhada. Na Figura 4 é possível visualizar a captura de imagem pela câmera do Mavic 2 PRO, possibilitada pelo *RVIZ*. Essa integração abrangente não só acelera o desenvolvimento de algoritmos de controle, mas também proporciona uma plataforma segura e eficaz para testes, garantindo que as soluções desenvolvidas sejam otimizadas e seguras antes de serem implementadas em drones reais.

4.2 CUSTOMIZAÇÃO DO AMBIENTE DE SIMULAÇÃO BASE

O desenvolvimento desse trabalho utilizou como base a simulação para o drone Mavic 2 PRO disponível no *GitHub* sob licença Apache, o que significa que o código pode ser usado, modificado e distribuído gratuitamente por qualquer pessoa, desde que sejam cumpridos os termos da licença. Entre as principais obrigações estão a necessidade de incluir um aviso de copyright original, uma cópia da licença, e um aviso de mudanças quando modificações forem feitas. Além disso, a licença Apache concede uma patente explícita, protegendo os usuários de futuras reivindicações de patentes dos contribuidores. Isso promove a inovação e colaboração, garantindo segurança legal e incentivando a participação da comunidade no desenvolvimento do software (CYBERBOTICS, 2024).

Sendo assim, o repositório *webotsros2mavic* foi clonado para máquina local para permitir a customização necessária para o trabalho apresentado. Inicialmente, o mundo de simulação incluía apenas um drone, sem a possibilidade de escolher a quantidade de drones na simulação e controlá-los separadamente. Esses são os principais objetivos deste trabalho: expandir a funcionalidade do simulador para suportar múltiplos drones e controle individualizado. O repositório contém diversas pastas e arquivos cruciais para o funcionamento da aplicação,

Figura 4 – Conteúdo captado pela câmera do Mavic 2 PRO no ambiente simulado, capturado pelo Rviz.

porém destacam-se três: o arquivo do mundo virtual, o arquivo de piloto dos drones e o arquivo de inicialização da aplicação, que serão editados para a construção do presente trabalho.

4.2.1 ALTERAÇÃO DO MUNDO VIRTUAL BASE PARA COMPORTAR MÚLTIPLOS VANTS

Um arquivo *.wbt* é um arquivo de mundo do *Webots*. Arquivos desse tipo definem o ambiente ou ”mundo”onde os robôs operam, contendo descrições do ambiente 3D, incluindo objetos, terrenos e configurações de robôs, escritos em uma sintaxe específica que o Webots interpreta. O conteúdo normalmente inclui configurações do mundo, como intervalo de tempo e gravidade, descrições detalhadas de todos os objetos dentro do mundo, incluindo suas formas, tamanhos, posições e propriedades físicas, bem como configurações de robôs que especificam sua estrutura, sensores, atuadores e parâmetros de controle. Além disso, são definidas características ambientais, como terrenos, obstáculos e luzes.

Inicialmente a simulação possuia apenas um modelo do Mavic 2 PRO, sem possibilidade de adicionar outros drones sem editar diretamente o arquivo do mundo. Para permitir a inserção de múltiplos drones no mundo, teve-se de pensar em uma forma de editar e inserir drones no arquivo sem precisar abrir diretamente o arquivo do mundo e fazer a edição manualmente.

Para isso, a simulação proposta pelo trabalho foi remover o drone existente no modelo base e criado um objeto genérico no código em Python do *launcher* da simulação para ser adicionado ao mundo.

4.2.2 PERSONALIZAÇÃO DA QUANTIDADE DE VANTS NO INÍCIO DA SIMULAÇÃO

Em um projeto *ROS2*, o arquivo de inicialização, também chamado de *launcher*, é um *script* em Python que realiza a execução de nós e a configuração do ambiente de simulação ou aplicação robótica. Esse arquivo define e automatiza a execução de múltiplos nós, parâmetros e outras configurações necessárias.

Conforme a solução proposta, o modelo do Mavic 2 PRO foi removido do mundo simulado e foi implementado um método no arquivo de inicialização que, ao ser chamado, solicita do usuário a quantidade de drones desejada e, dada resposta, adiciona a quantidade solicitada de drones no arquivo do mundo. Isso é realizado abrindo-se o arquivo do mundo .wbt e inserindo o modelo do drone repetidamente até que a quantidade solicitada de drones seja atingida. Na Figura 5 consta a tela do terminal após execução da simulação, na qual é solicitado ao usuário que informe a quantidade desejada de drones no cenário. Já na Figura 6, é possível visualizar um cenário de simulação em que foram inseridos três drones no mundo virtual.

```
patrick@ubuntu-patrick:~/Documents/ros/drone-simulation_ws$ ros2 launch mavic_simulation robot_launch.py
[INFO] [launch]: All log files can be found below /home/patrick/.ros/log/2024-06-27-16-47-08-837466-ubuntu-patrick-94724
[INFO] [launch]: Default logging verbosity is set to INFO
WARNING: No valid Webots directory specified in `ROS2_WEBOTS_HOME` and `WEBOTS_HOME`, fallback to default installation folder /usr/local/webots.

**************************************
How many drones do you want to simulate?
**************************************
R:
```

Figura 5 – Solicitação, no terminal, de que o usuário informe quantidade desejada de drones no cenário simulado.

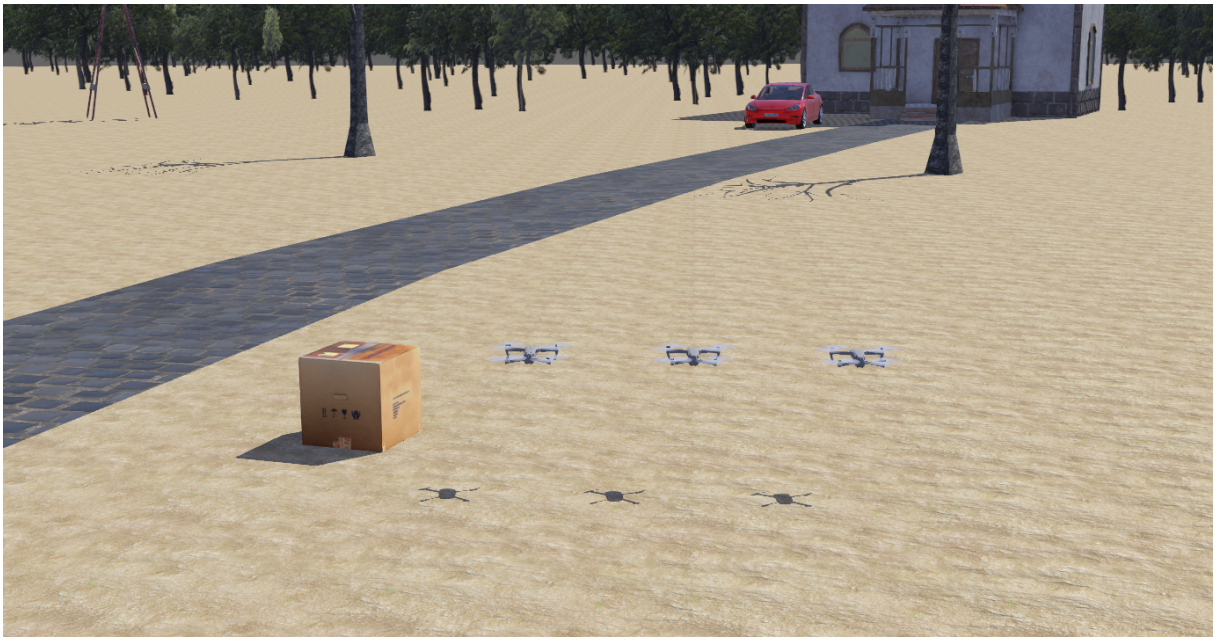

Figura 6 – Simulação com três drones Mavic 2 PRO

Porém, somente a inclusão dos drones no mundo não os torna funcionáveis, visto que cada drone adicionado à simulação também requer a inclusão de um *plugin* de controle específico. Esses *plugins* são responsáveis por fornecer a lógica de controle e interação necessária para que cada drone possa ser controlado durante a simulação. No contexto da aplicação deste trabalho,

o plugin é um nó do *ROS2*. Um nó é uma unidade fundamental de execução responsável por realizar uma tarefa específica em um sistema robótico distribuído, no presente contexto, controlar a movimentação dos drones. Cada nó é um processo independente que pode se comunicar com outros nós através de uma infraestrutura de comunicação baseada em tópicos, serviços e ações.

Com a associação dos nós de controle aos drones adicionados ao mundo, eles se tornam entidades manipuláveis e interativas dentro do ambiente virtual, permitindo seu controle e navegação no espaço simulado. Porém, nessa etapa todos os controles recebem mensagens do mesmo tópico, ou seja, eles são todos controlados simultaneamente.

O *ROS2* possui uma ferramenta chamada *rqtgraph* que permite aos usuários visualizar graficamente a topologia do sistema de nós em execução. Ele fornece uma representação visual das conexões entre os nós, tópicos e serviços, facilitando a compreensão da estrutura e comunicação do sistema robótico. Na Figura 7, é possível observar a existência de um nó para cada um dos drones, em um cenário com três drones, e três controles correspondentes. Porém, todos os controles recebem mensagens do mesmo tópico, o *cmdvel*, que registra velocidades geométricas que indicam o posicionamento dos drones na simulação.

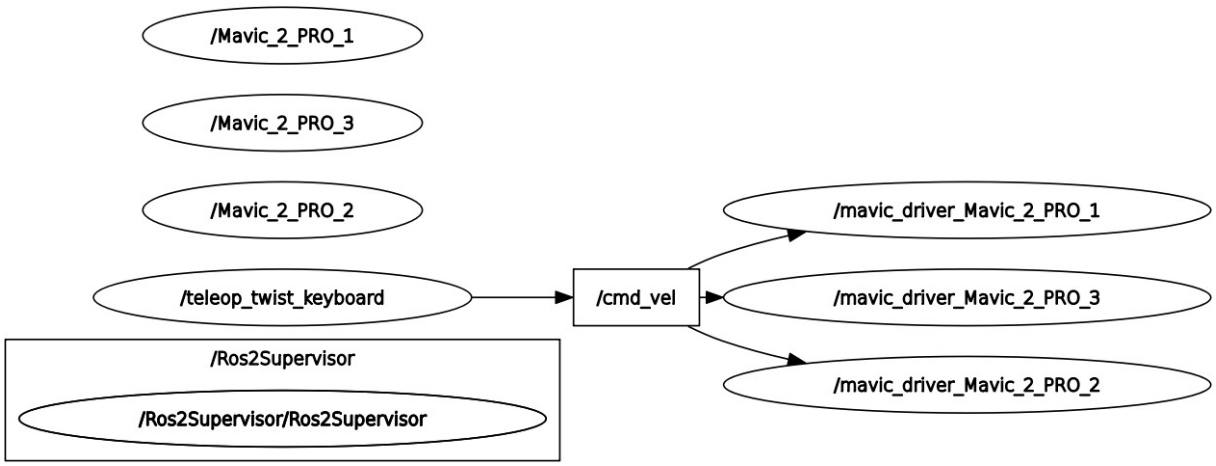

Figura 7 – Topologia dos nós de controle dos drones na simulação no cenário em que todos os drones recebem comandos do mesmo controle obtida pelo *rqtgraph*.

4.2.3 INDIVIDUALIZAÇÃO DOS CONTROLES DOS VANTS

No contexto do *webotsros2mavic*, o controle dos drones é facilitado pelo uso das mensagens do pacote *teleoptwist*, que permite ao usuário controlar os drones por meio de comandos de velocidade linear e angular, publicados no tópico genérico *cmdvel*. Inicialmente, todos os drones compartilham o mesmo controle, ou seja, são movidos de maneira igual, proporcionando uma experiência não muito flexível. O ideal seria que cada drone fosse independente e pudesse se movimentar sem dependência com os demais drones da simulação.

Para individualizar o controle de cada drone, foi necessário alterar o arquivo de *driver* ou simplesmente controle utilizado nos drones. Ao invés de associar todos os drones com o mesmo tópico *cmdvel*, como por padrão, foi gerado para cada um deles um tópico diferente

baseado no *cmdvel* original, da qual somente o drone correspondente recebe mensagens. O *ROS2* permite que, ao criar um tópico, seja feito uma cópia independente dele caso um nome diferente seja passado para a instância. Com isso, cada drone pode ser controlado separadamente, desde que os comando de velocidade angular sejam enviados ao controle adequado. Na Figura 8, é possível observar a existência de um nó de controle para cada drone, que publica em um tópico exclusivo que só é acessado pelo drone equivalente.

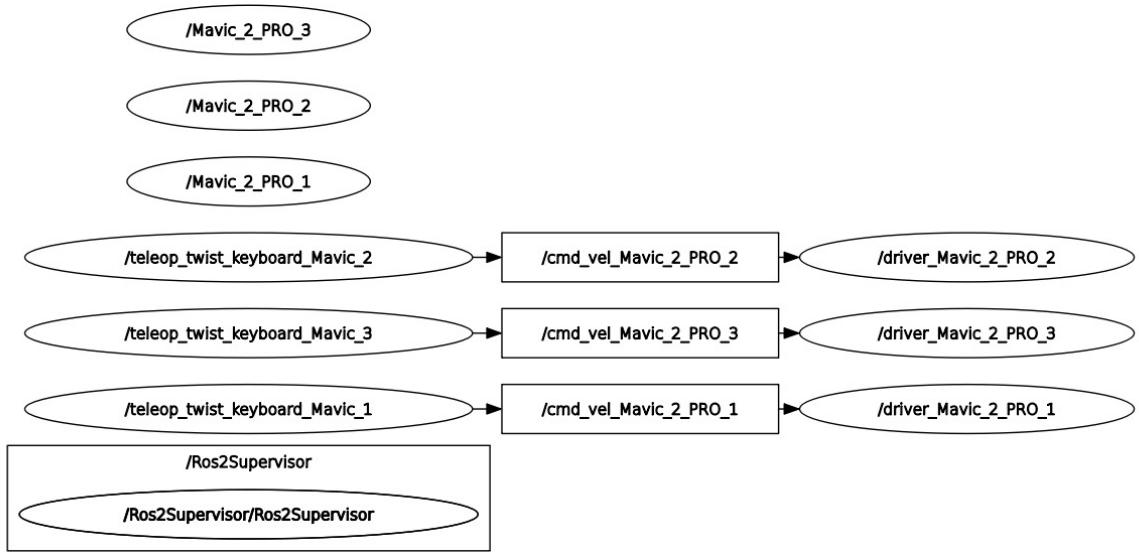

Figura 8 – Topologia dos nós de controle dos drones na simulação no cenário em que cada drone recebe comandos de um controle individual obtida pelo *rqtgraph*.

Para utilizar esse sistema de controle individualizado, o usuário precisa executar tanto o nó do *publisher*, ou nó editor, quanto o nó do *subscriber*, ou nó assinante, correspondente ao drone que deseja controlar. Essa abordagem oferece uma flexibilidade adicional ao permitir que o usuário escolha quais drones controlar e como interagir com eles durante a simulação, tornando a experiência de desenvolvimento mais interativa e dinâmica.

A Figura 9 apresenta um fluxograma que demonstra os passos para executar e utilizar a simulação. Por sua vez, a Figura 10 mostra um fluxograma com o fluxo de execução dos componentes no projeto durante a execução da simulação.

4.3 DESEMPENHO DA SIMULAÇÃO

A quantidade máxima de drones na simulação é teoricamente ilimitada, porém é importante considerar que o aumento no número de drones impacta diretamente no desempenho da aplicação. Conforme mais drones são adicionados, a aplicação torna-se mais pesada e a execução, tanto para iniciar a simulação, quanto para controlá-la, pode ficar consideravelmente mais lenta.

Para o presente trabalho, foram executadas sete instâncias de testes com diferentes quantidades de drones, em que eles foram colocadas para voar no ambiente simulado e controlados pelo usuário com o fim de avaliar o desempenho em diversas condições.

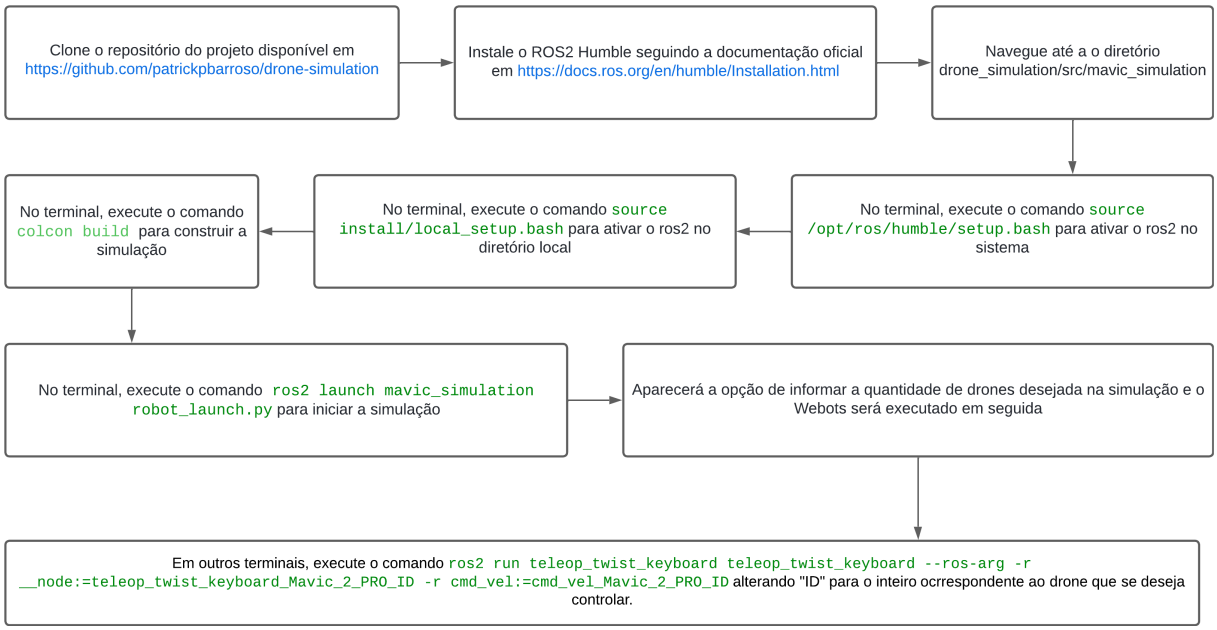

Figura 9 – Fluxograma com o passo a passo para executar e utilizar a simulação.

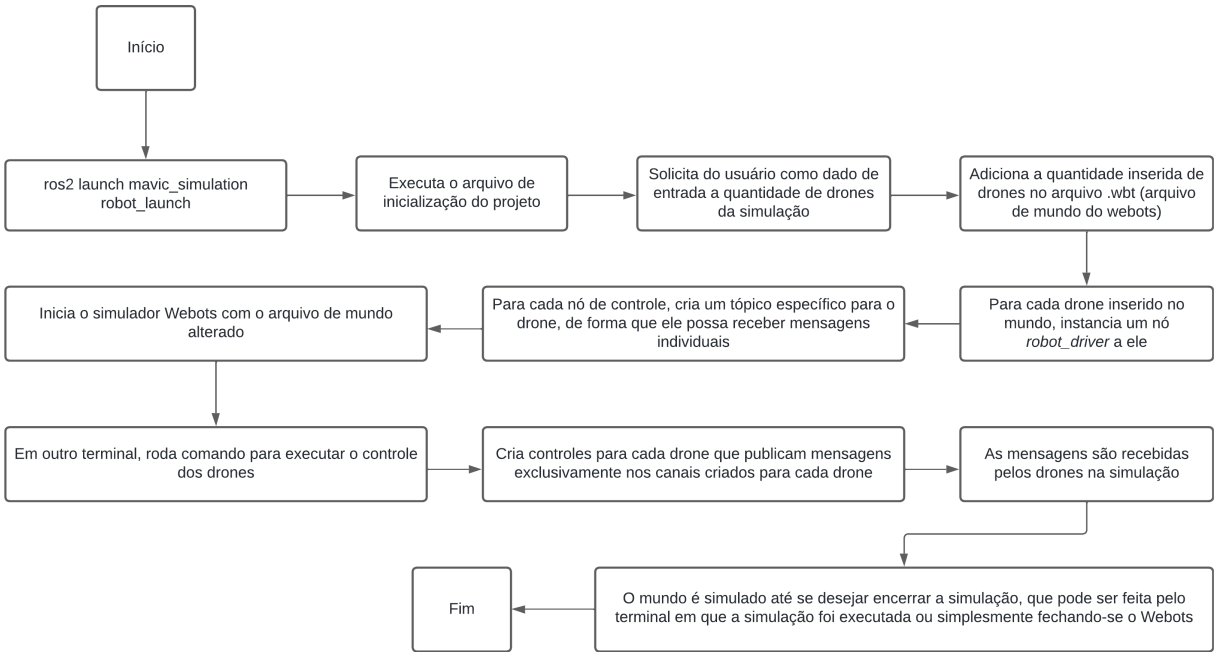

Figura 10 – Fluxograma com o fluxo de execução dos componentes do projeto durante a execução da simulação.

Os resultados dos testes estão apresentados na tabela a seguir e revelaram que o tempo necessário para iniciar a aplicação aumentou significativamente à medida que mais drones foram adicionados.

Quantidade de drones	**Tempo de inicialização (segundos)**
1	18
2	20
5	27
10	68
50	95
100	291
150	377

Ademais, foi observado nos testes que, para o hardware utilizado, simulações com até 20 drones foram executadas sem gargalos, apresentando um bom desempenho e sem quaisquer problemas de lentidão. Já com uma instância de 50 drones, foi observada maior latência e lentidão na execução da simulação, indicando um maior esforço computacional necessário para processar essa quantidade de drones e seus controles, porém ainda factível de se manipular. A partir de 100 drones, a simulação se tornou consideravelmente lenta, com tempos de resposta mais longos e uma experiência de simulação menos fluida.

Para os testes executados, cada drone foi controlado individualmente por meio de seu controle correspondente. Porém, como cada drone está associado a um controle específico, que recebe mensagens de um tópico exclusivo, há possibilidade de implementação de algoritmos que definam as rotas para os drones e publiquem em seus respectivos canais, qual direção devem seguir. Isso é uma funcionalidade poderosa, pois permite a aplicaçao de diversos algoritmos de inteligência artificial e otimização que definam, de forma inteligente, a trajetória dos drones no mundo simulado.

5 CONCLUSÃO E TRABALHOS FUTUROS

Com o grande avanço em inovações de tecnologias relacionadas a veículos aéreos não tripulados (VANTS), essas máquinas são cada vez mais utilizadas em diversos setores da sociedade. Os drones se tornaram pilares para realização de diversas tarefas, como resgate, monitoramento ambiental, agricultura, logística, entre outras.

Nesse sentido, surge também a necessidade de desenvolver softwares cada vez mais potentes para os VANTs e, dados os custos envolvidos no processo de teste e desenvolvimento, formas seguras, baratas e confiáveis de realizar os testes são essenciais.

O presente trabalho propôs a customização de um modelo base de simulação que permite ao usuário testar determinada quantidade desejada de drones em um espaço virtual e controlá-los separadamente. Isso representa o primeiro passo de um projeto que pode evoluir de diversas formas e contribui para melhorar a eficiência e produtividade de simulações para múltiplos drones. O código do projeto foi disponibilizado no *GitHub* juntamente com as instruções de instalação (BARROSO, 2024). Seguindo os comandos, todas as dependências devem ser instaladas e o usuário deve ser capaz de informar quantos VANTs deseja controlar na simulação. Uma vez que o *Webots* tenha carregado a simulação, o usuário, pelo terminal, deve poder controlar cada um dos VANTs via controle correspondente.

Para futuros trabalhos, recomenda-se a implementação de uma parametrização para definir a localização inicial de cada drone. Também recomenda-se o desenvolvimento de uma interface gráfica para o *launcher* da simulação, tornando a interação com o usuário mais intuitiva e acessível, sem a necessidade de execução de comandos verbosos no terminal e facilitando a parametrização. Além disso, a captura e armazenamento das imagens registradas pelos drones por meio do RViz pode ser explorada, oferecendo uma ferramenta valiosa para análise e visualização dos dados gerados durante a simulação.

Outra possibilidade de pesquisa é a implementação de algoritmos pré-determinados no lugar do controle manual, permitindo a execução de testes automatizados e a comparação de diferentes estratégias de controle em cenários variados. Com os diversos sensores disponíveis nos drones, como infra-vermelho, é possível implementar algoritmos poderosos que evitem colisões, percorram o mundo simulado, e muito mais.

Essas melhorias e extensões contribuirão para a evolução e aprimoramento do simulador, ampliando suas capacidades e tornando-o uma ferramenta ainda mais poderosa para o desenvolvimento e validação de sistemas de drones.

REFERÊNCIAS

BARROSO, P. **Projeto de simulação para múltiplos drones individuais**. 2024. Https://github.com/patrickpbarroso/drone-simulation.

CHEN, Z.; YAN, J.; MA, B.; SHI, K.; YU, Q.; YUAN, W. A survey on open-source simulation platforms for multi-copter uav swarms. **MDPI**, 2023. ISSN 1545-598X.

CYBERBOTICS. **Código fonte do pacote webotsros2mavic**. 2024. Urlhttps://github.com/cyberbotics/webots$_r os$2.

FARLEY, A.; WANG, J.; MARSHALL, J. A. How to pick a mobile robot simulator: A quantitative comparison of coppeliasim, gazebo, morse and webots with a focus on accuracy of motion. **Elsevier**, 2022. ISSN 1545-598X.

HASNAIN, S. G. M.; RAFI, F. A. Windows, linux, mac operating system and decision making. **International Journal of Computer Applications**, 2019. ISSN 1545-598X.

ICAO. Unmanned aircraft systems (uas). **International Civil Aviation Organization**, 2011. ISSN 1545-598X.

KARDASZ1, P.; DOSKOCZ1, J.; HEJDUK2, M.; WIEJKUT3, P.; ZARZYCKI4, H. Drones and possibilities of their use. **Journal of Civil and Environmental Engineering**, 2016. ISSN 1545-598X.

KOENING, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. **International Conference on Intelligent Robots and Systems**, 2004. ISSN 1545-598X.

LI, X.; TUPAYACHI, J.; SHARMIN, A.; FERGUSON, M. M. Drone-aided delivery methods, challenge, and the future: A methodological review. **MDPI**, 2023. ISSN 1545-598X.

MACENSKI, S.; FOOTE, T.; GERKEY, B.; LALANCETTE, C.; WOODALL, W. Robot operating system 2: Design, architecture, and uses in the wild. 2022. ISSN 1545-598X.

MACENSKI, S.; SORAGNA, A.; CARROLL, M.; GE, Z. Impact of ros 2 node composition in robotic systems. **Systems**, 2023. ISSN 1545-598X.

MARUYAMA, Y.; KATO, S.; AZUMI, T. Exploring the performance of ros2. 2022. ISSN 1545-598X.

MICHEL, O. A comprehensive review of recent research trends on unmanned aerial vehicle (uavs). **International Journal of Advanced Robotic Systems**, 2004. ISSN 1545-598X.

TELLI, K.; KRAA, O.; HIMEUR, Y.; OUAMANE, A.; BOUMEHRAZ, M. A comprehensive review of recent research trends on unmanned aerial vehicle (uavs). **Systems**, 2023. ISSN 1545-598X.

TICE, B. P. Unmanned aerial vehicles: The force multiplier of the 1990s. **Airpower**, 2023. ISSN 1545-598X.