Sabrina Araújo Cardoso

Relatório de Estágio Supervisionado Laboratório de Sistemas Embarcados e Computação Pervasiva (Embedded)

Campina Grande, Paraíba

Maio de 2024

Sabrina Araújo Cardoso

Relatório de Estágio Supervisionado
Laboratório de Sistemas Embarcados e Computação Pervasiva (Embedded)

Relatório de Estágio Supervisionado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, *Campus* Campina Grande, como parte dos requisitos necessários para obtenção do grau de Bacharel em Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Orientador: Danilo Freire de Souza Santos, D.Sc

Campina Grande, Paraíba

Maio de 2024

Sabrina Araújo Cardoso

Relatório de Estágio Supervisionado
Laboratório de Sistemas Embarcados e Computação Pervasiva (Embedded)

Relatório de Estágio Supervisionado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, *Campus* Campina Grande, como parte dos requisitos necessários para obtenção do grau de Bacharel em Engenharia Elétrica.

Trabalho aprovado em: 17 de Maio de 2024

Danilo Freire de Souza Santos, D.Sc
Orientador

Gutemberg Gonçalves dos Santos Júnior, Dr.
Professor Convidado

Campina Grande, Paraíba
Maio de 2024

Dedico este trabalho a minha família e à todos que acreditaram em mim.

Agradecimentos

Gostaria de agradecer, primeiramente a Deus, por ter me dado força, coragem, foco e muita perseverança para que eu pudesse encarar todos os obstáculos encontrados durante o período de graduação.

Também quero agradecer imensamente a minha família. Em especial, a minha mãe, Iranilda, que mesmo diante de todas as dificuldades enfrentadas, nunca me deixou desamparada e nunca desacreditou do meu potencial. E a minha irmã, Silvana, que sempre foi minha fonte de inspiração. Elas sempre me apoiaram em todas as minha decisões. E sei que posso contar sempre com elas para o que for preciso. Também quero agradecer a meu sobrinho, Dieguinho, que chegou ao mundo há menos de 4 meses, e foi a luz que me deu forças para não desanimar nessa reta final.

Agradeço a minha namorada Jade, que sempre esteve ao meu lado em todos os momentos de estresse e me deu forças quando nem eu mesma acreditava que conseguiria ter. Sem sua companhia, minha jornada no curso seria mais ardosa. Obrigada por deixar tudo mais leve.

Não posso deixar de agradecer aos meus amigos de curso, em especial Débora e Dhara, que sempre me acompanharam nos momentos de café, estudo e desabafo. Com vocês foi possível descontrair e tentar amenizar toda a carga acompanhada pela graduação. Agradeço também a Beatriz e Felipe, por estarem sempre me dando força no ambiente de trabalho e tornando tudo mais divertido e leve.

Agradeço ao professor Antonio Marcus por ter me acolhido desde o segundo período do curso e sempre ter me dado oportunidade. O senhor foi essencial para o meu crescimento acadêmico. E ao professor Danilo Santos, por ter confiado no meu trabalho e ter aceitado me orientar durante meu período de estágio.

Meus sinceros obrigada à todos!

Resumo

Este relatório descreve as atividades realizadas por Sabrina Araújo Cardoso durante o estágio supervisionado no Laboratório de Sistemas Embarcados e Computação Pervasiva (Embedded) da Universidade Federal de Campina Grande, em parceria com o Núcleo de PDI VIRTUS. Durante o período de estágio, a estagiária focou em desenvolver e executar simulações de robôs utilizando o Sistema Operacional de Robôs (ROS) e o simulador Gazebo, destacando-se na criação de um ambiente virtual para testar interações e mapeamentos. As atividades incluíram a simulação de robôs específicos, como o braço robótico *Universal Robots UR10* e o robô móvel *Turtlebot2i*, e a utilização de técnicas para a sincronização de Gêmeos Digitais e o mapeamento em nuvem de pontos, contribuindo para o desenvolvimento de um artigo científico sobre sincronização de Gêmeos Digitais.

Palavras-chave: ROS, Gazebo, Rviz, Gêmeo Digital, RTAB-Map.

Abstract

This report details the activities carried out by Sabrina Araújo Cardoso during her supervised internship at the Laboratory of Embedded Systems and Pervasive Computing (Embedded) at the Federal University of Campina Grande, in partnership with the VIRTUS R&D Center.Throughout her internship, the intern focused on developing and executing advanced robot simulations using the Robot Operating System (ROS) and Gazebo simulator, excelling in creating a virtual environment to test complex interactions and mappings. The activities included the simulation of specific robots such as the UR10 robotic arm and Turtlebot2i mobile robot, and the use of advanced techniques for Digital Twin synchronization and point cloud mapping, significantly contributing to the development of a scientific article on Digital Twin synchronization.

Keywords: ROS, Gazebo, Rviz, Digital Twin, RTAB-Map.

Lista de ilustrações

Figura 1 – Fotografia do prédio do Laboratório Embedded. 3
Figura 2 – Fotografia da sede do VIRTUS. 4
Figura 3 – Revoluções Indústriais. 5
Figura 4 – Modelo de Gêmeo Digital. 7
Figura 5 – Representação do funcionamento dos Nós no ROS. 10
Figura 6 – Representação do funcionamento dos Tópicos no ROS. 11
Figura 7 – Representação do funcionamento dos Serviços no ROS. 12
Figura 8 – Representação do funcionamento das Ações no ROS. 13
Figura 9 – Linha do Tempo da História do Gazebo Clássico. 14
Figura 10 – GUI do Gazebo. 15
Figura 11 – GUI do Rviz. 17
Figura 12 – GUI do RTAB-Map. 18
Figura 13 – Robô UR10 Real 21
Figura 14 – Robô UR10 no Gazebo e no Rviz 21
Figura 15 – Robô *Turtlebot2i* Real 22
Figura 16 – *Turtlebot2i* no Gazebo e no Rviz 22
Figura 17 – Representação da Sala 03 do Embedded no Gazebo. 23
Figura 18 – À esquerda, visão da câmera no Gazebo; à direita, imagem capturada pela câmera no RViz. 25
Figura 19 – Mapeamento em nuvem de pontos vista no Rviz. 25

Lista de Abreviaturas e Siglas

CEEI	Centro de Engenharia Elétrica e Informática.
UFCG	Universidade Federal de Campina Grande.
Embedded	Laboratório de Sistemas Embarcados e Computação Pervasiva da UFCG.
VIRTUS	Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação.
ROS	*Robot Operating System.*
GD	Gêmeo Digital.
URDF	*Unified Robot Description Format.*
RTAB-Map	*Real-Time Appearance-Based Mapping.*
IoT	Internet das Coisas.
AI	Inteligência Artificial.
GUI	Interface Gráfica do Usuário.
Rviz	ROS *Visualization.*
SLAM	Simultaneous Localization and Mapping

Sumário

1 INTRODUÇÃO . . . 1
1.1 Objetivos . . . 1
1.1.1 Objetivos Específicos . . . 2

2 AMBIENTE DE TRABALHO . . . 3
2.1 Embedded . . . 3
2.2 VIRTUS . . . 4

3 FUNDAMENTAÇÃO TEÓRICA . . . 5
3.1 Indústria 4.0 . . . 5
3.2 Gêmeo Digital . . . 6
3.3 *Robot Operating System* (ROS) . . . 8
3.3.1 *ROS Nodes* (Nós) . . . 9
3.3.2 *ROS Topics* (Tópicos) . . . 10
3.3.3 *ROS Services* (Serviços) . . . 11
3.3.4 *ROS Actions* (Ações) . . . 12
3.4 Gazebo Clássico . . . 13
3.5 *ROS Visualization* (Rviz) . . . 16
3.6 *Real-Time Appearance-Based Mapping* (RTAB-Map) . . . 17
3.7 Metodologias Ágeis: *Scrum* . . . 19
3.8 Linux . . . 19
3.9 Sistema de Controle de Versão: Git . . . 19

4 ATIVIDADES REALIZADAS . . . 20
4.1 Visão Geral . . . 20
4.2 Capacitação em Robótica . . . 20
4.2.1 Desenvolvimento dos URDFs . . . 20
4.2.1.1 Robô *Universal Robots UR10* . . . 21
4.2.1.2 Robô *Turtlebot2i* e Câmera *Astra Pro Plus* . . . 21
4.2.1.3 Sala 03 do Embedded . . . 22
4.3 Sincronização do Gêmeo Digital do UR10 . . . 23
4.4 Desenvolvimento do Mapeamento Utilizando o RTAB-Map . . . 24
4.4.1 Outras atividades . . . 26

5 CONSIDERAÇÕES FINAIS . . . 27

REFERÊNCIAS 28

1 Introdução

A elaboração, apresentação e avaliação da disciplina Estágio Supervisionado ou Integrado são obrigatórias para os alunos concluintes, conforme previsto no Projeto Pedagógico do Curso de Engenharia Elétrica do Centro de Engenharia Elétrica e Informática (CEEI) da Universidade Federal de Campina Grande (UFCG). Eles compõe a grade curricular obrigatória para obtenção do grau de Bacharel em Ciências no domínio da Engenharia Elétrica. O objetivo deste é proporcionar aos alunos concluintes uma experiência profissional, onde seja possível conhecer e executar atividades associadas à Engenharia Elétrica, consolidando, assim, os conhecimentos adquiridos durante a graduação, além de possibilitar o contato com o mercado de trabalho.

O presente relatório é o documento de descrição das principais atividades realizadas durante a execução da disciplina de Estágio Supervisionado pela aluna Sabrina Araújo Cardoso, no curso de graduação em Engenharia Elétrica na UFCG. O estágio teve carga horária de 23 horas semanais e ocorreu durante o período de 06 de dezembro de 2023 à 21 de fevereiro de 2024, totalizando 256 horas, no Embedded (Laboratório de Sistemas Embarcados e Computação Pervasiva da UFCG), em parceria com o VIRTUS (Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação), sob orientação do professor Danilo Freire de Souza Santos e supervisão do professor Gutemberg Gonçalves dos Santos Júnior.

Durante o período de vigência, as principais atividades desenvolvidas foram a atuação em uma das equipes de desenvolvimento de projeto, contribuindo para o desenvolvimento na área da robótica, atuando, especialmente, no desenvolvimento de simulação de dois robôs diferentes, utilizando o *Robot Operating System* (ROS), no português Sistema Operacional de Robôs, e o simulador Gazebo. Foi realizada, também, a simulação de câmeras de profundidade com o objetivo de executar o mapeamento da simulação e transformar em nuvem de pontos. Além de participar de pesquisa e investigação, análise e organização de uma lista de requisitos e especificações técnicas e documentação de processos realizados no projeto, bem como no desenvolvimento de um artigo científico.

1.1 Objetivos

O estágio supervisionado, realizado no Laboratório Embedded, teve como principal objetivo o desenvolvimento e execução de simulações envolvendo dois robôs distintos: o braço robótico *Universal Robots UR10* e o robô móvel *Turtlebot2i*. Essas simulações foram projetadas para prever o comportamento desses robôs em cenários reais. Adicionalmente, foi realizada a simulação de uma câmera com sensor de profundidade, em uma

tentativa de deixá-la o mais próximo possível do modo de funcionamento da Câmera *Astra Pro Plus*, visando gerar um mapeamento em nuvem de pontos do ambiente simulado. Este trabalho culminou na contribuição ao desenvolvimento de um artigo científico que explorava métodos para a sincronização de um Gêmeo Digital (GD).

1.1.1 Objetivos Específicos

Para alcançar este objetivo principal, o estágio foi estruturado em tarefas específicas, que incluíam:

1. Ajustar e validar os arquivos *Unified Robot Description Format* (URDF) dos robôs *Universal Robots UR10* e *Turtlebot2i* para garantir simulações precisas no Gazebo.
2. Implementar a simulação de uma câmera com sensor de profundidade no Gazebo para capturar dados tridimensionais necessários ao mapeamento do ambiente.
3. Desenvolver uma réplica virtual do laboratório em URDF, facilitando a integração e a simulação de um GD.
4. Utilizar o *Real-Time Appearance-Based Mapping* (RTAB-Map) para mapear o ambiente com base nos dados coletados pela câmera simulada, acompanhando o movimento dos robôs.
5. Auxiliar na elaboração de um artigo científico focado em métodos de sincronização de GD, baseando-se nas experiências e resultados obtidos durante o estágio.

Essas atividades não apenas visaram à aplicação prática dos conhecimentos adquiridos durante o curso, mas também à inserção da estagiária no contexto profissional e acadêmico da Engenharia Elétrica e da Robótica.

No presente capítulo foi apresentado uma breve introdução e os objetivos do estágio, juntamente com uma visão geral da organização deste relatório. Nos próximos capítulos, serão abordados os seguintes temas: Capítulo 2, apresentação do Embedded e o VIRTUS, além de um breve resumo de suas histórias; no Capítulo 3, será feita uma fundamentação teórica necessária para embasar os conceitos utilizados nas atividades; Capítulo 4, descrição das atividades desenvolvidas; e, por fim, no Capítulo 5 apresentam-se as considerações finais.

2 Ambiente de Trabalho

Este capítulo apresenta uma descrição detalhada do Laboratório de Sistemas Embarcados e Computação Pervasiva (Embedded) e do VIRTUS, Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação, enfatizando suas infraestruturas e equipes. Estes grupos desempenham papéis fundamentais para o progresso tecnológico e para as pesquisas avançadas na área de tecnologia da informação.

2.1 Embedded

O Laboratório Embedded, parte integrante do Centro de Engenharia Elétrica e Informática (CEEI) da Universidade Federal de Campina Grande (UFCG), está situado em Campina Grande, Paraíba, e foi fundado em dezembro de 2005. O laboratório ocupa uma área de 600 metros quadrados no campus universitário. Uma fotografia da fachada do prédio é apresentada na Figura 1. (EMBEDDED, 2023)

Figura 1 – Fotografia do prédio do Laboratório Embedded.

Fonte: embedded.ufcg.edu.br.

Com uma equipe composta por aproximadamente 60 colaboradores, incluindo docentes e pesquisadores do CEEI, além de alunos de graduação, mestrado e doutorado dos cursos de Engenharia Elétrica e Ciência da Computação, o laboratório visa desenvolver novas soluções e melhorias para as tecnologias de sistemas embarcados e computação pervasiva. Essas tecnologias são projetadas para integrar-se de maneira imperceptível no dia a dia dos usuários.

A missão do Embedded é beneficiar a sociedade ao aplicar esses avanços tecnológicos, mantendo um equilíbrio entre as demandas acadêmicas e as necessidades do mercado por meio de parcerias estratégicas com grandes empresas.

O laboratório foca na concepção, implementação e experimentação de soluções inovadoras para sistemas embarcados e computação pervasiva, proporcionando tecnologias que se integram de forma invisível ao cotidiano dos usuários. As atividades do estágio foram primariamente conduzidas na Sala 03, um espaço dedicado a projetos de pesquisa e desenvolvimento de hardware.

2.2 VIRTUS

Fundado em 2014, o VIRTUS é um órgão suplementar da UFCG vinculado ao CEEI. Localizado também em Campina Grande, Paraíba, ele é um centro fundamental para o avanço da pesquisa e da inovação tecnológica em colaboração com o setor industrial, abrangendo áreas como tecnologia da informação, comunicação e automação. Este núcleo realiza projetos amparados por programas de incentivo como Lei de Informática, EMBRAPII e ANP. (VIRTUS, 2023)

A equipe do VIRTUS é diversificada, contando com professores, pesquisadores, estudantes e profissionais externos, trabalhando conjuntamente em várias linhas de pesquisa. As principais áreas de foco incluem Sistemas Embarcados, Aprendizagem de Máquina, Internet das Coisas (IoT) e Computação em Nuvem, o que permite ao VIRTUS enfrentar desafios tecnológicos complexos e contribuir com soluções inovadoras para o campo da engenharia e tecnologia.

Figura 2 – Fotografia da sede do VIRTUS.

Fonte: virtus.ufcg.edu.br.

3 Fundamentação Teórica

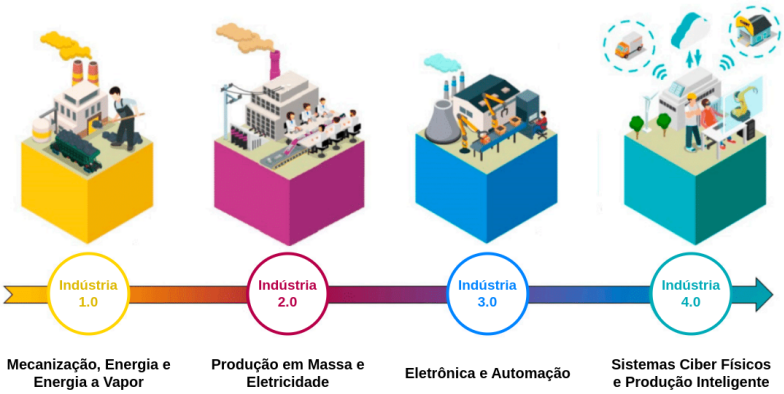
Este capítulo detalha os conceitos-chave que formam a base teórica essencial para as atividades desenvolvidas durante o estágio. A compreensão aprofundada desses conceitos é crucial para entender como diversas tecnologias e métodos foram aplicados ao longo deste período. Estes fundamentos são essenciais para a utilização eficaz das ferramentas disponíveis e para enfrentar os desafios técnicos encontrados, facilitando a busca por soluções adequadas e enriquecendo o aprendizado prático e teórico.

3.1 Indústria 4.0

A Indústria 4.0, também conhecida como Quarta Revolução Industrial, simboliza uma transformação fundamental nas operações industriais, marcada pela introdução de automação avançada e pela integração de tecnologias digitais. Este conceito emergiu da estratégia de alta tecnologia do governo alemão em 2011 e ganhou destaque global quando adotado pelo Fórum Econômico Mundial em 2016 (DELOITTE, 2020).

Este movimento tem suas raízes na Primeira Revolução Industrial, que revolucionou a fabricação com a mecanização. As evoluções subsequentes trouxeram a produção em massa facilitada pela eletricidade e mais tarde a automação proporcionada por computadores e eletrônica. A Quarta Revolução Industrial expande esses avanços anteriores, elevando a produção a patamares inéditos de digitalização e inteligência operacional.

Figura 3 – Revoluções Indústriais.



Fonte: (BENOTSMANE; KOVÁCS; DUDÁS, 2019).

A composição da Indústria 4.0 inclui tecnologias cruciais como a Internet das Coisas (*IoT*), *Big Data*, Inteligência Artificial (*AI*), Aprendizado de Máquina (*Machine Lear-*

ning), Robótica Avançada, Realidade Aumentada e Sistemas Ciber-físicos. Essas inovações permitem a emergência de fábricas inteligentes onde sistemas e máquinas comunicam-se e analisam dados para otimizar operações autônomas e aumentar a eficiência (IBM, 2020).

Os Gêmeos Digitais (GDs), destacados na seção 3.2, representam um pilar central da Indústria 4.0. Essas réplicas virtuais de objetos físicos ou sistemas facilitam a simulação, monitoramento e controle operacional. Sua integração com outras tecnologias do setor, não só impulsiona o desenvolvimento de produtos, como também otimiza processos de manutenção e propicia inovações em modelos de negócios (NEDASHKOVSKIY et al., 2020; AZHEGANOVA et al., 2020).

Consequentemente, a Indústria 4.0 está reformulando os alicerces da produção industrial, apresentando desafios e abrindo portas para novas oportunidades. Empresas por todo o mundo estão se adaptando a essa nova era, redefinindo o conceito de inteligência no ambiente industrial e explorando como os dados podem ser empregados para fomentar inovações contínuas. Com os GDs desempenhando um papel crucial nesse processo, a conexão entre elementos físicos e digitais nunca foi tão íntima (GUPTA et al., 2020).

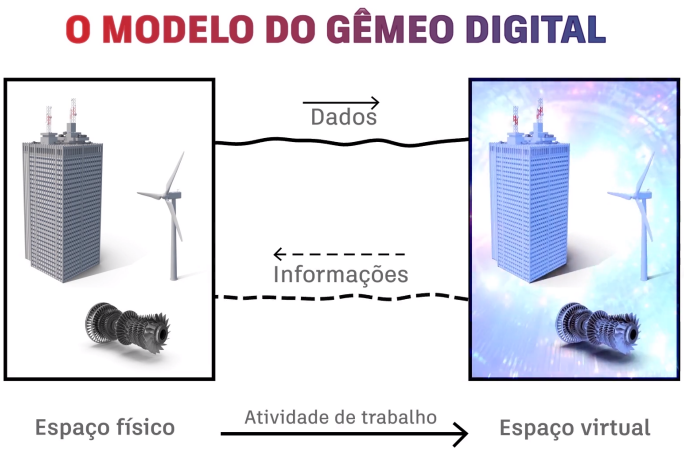
3.2 Gêmeo Digital

Gêmeos Digitais são definidos como réplicas virtuais precisas de objetos físicos, proporcionando a simulação, monitoramento e análise detalhada de produtos e processos em diversos setores industriais.

Introduzidos no contexto acadêmico por (GRIEVES, 2014), esses modelos digitais são projetados para replicar os estados e comportamentos dos seus correspondentes físicos ao longo de todo o ciclo de vida. O desenvolvimento dessas representações tem se aprimorado significativamente com o avanço das tecnologias de coleta de dados e simulação, alcançando uma fidelidade que os torna quase indistinguíveis de seus pares materiais.

O conceito de GD surgiu durante um curso sobre *Gestão do Ciclo de Vida do Produto* na Universidade de Michigan, destacando-se como uma ferramenta poderosa para a análise, simulação e otimização de produtos de maneira virtual. Inicialmente, essas representações eram mais simples, mas hoje, graças ao progresso tecnológico, elas oferecem uma riqueza de detalhes impressionante.

Figura 4 – Modelo de Gêmeo Digital.



Fonte: (Autodesk, 2024).

Conforme detalhado por (NEDASHKOVSKIY et al., 2020), um GD é composto por três componentes principais:

- **Espaço Físico:** O local onde os objetos ou processos reais estão situados, equipados com sensores que coletam dados operacionais e ambientais.
- **Espaço Virtual:** Uma plataforma digital que hospeda o modelo digital, capaz de replicar e simular o comportamento do objeto físico com precisão em tempo real.
- **Dados e Conexões:** Elementos que formam as pontes entre o espaço físico e virtual, permitindo um fluxo contínuo de informações bidirecionais que são essenciais para atualizações e análises aprofundadas.

Estes componentes colaboram para oferecer *insights* detalhados sobre o desempenho, comportamento e manutenção de objetos físicos. A capacidade de simular cenários em um ambiente controlado virtualmente, apresenta-se como uma das principais vantagens dos GDs, abrindo novos caminhos para otimização e inovação.

Os GDs têm aplicações vastas, desde melhorar a eficiência de linhas de produção até prever falhas em equipamentos complexos. No âmbito educacional e de treinamento, eles servem como plataformas seguras e eficazes para capacitação técnica, minimizando riscos ao equipamento real.

À medida que a tecnologia evolui, a integração dos Gêmeos Digitais nas práticas empresariais modernas torna-se cada vez mais prevalente. Com o crescimento da IoT e a emergência de novas tecnologias, espera-se que os GDs desempenhem um papel ainda mais crucial na Indústria 4.0, facilitando a integração inédita entre o mundo físico e o digital (NEDASHKOVSKIY et al., 2020).

3.3 *Robot Operating System* (ROS)

O *Robot Operating System* (ROS), no português Sistema Operacional de Robôs, é uma coleção de bibliotecas de *software* e ferramentas projetadas para ajudar no desenvolvimento de aplicações robóticas. Composto por *drivers*, algoritmos avançados e instrumentos poderosos para desenvolvedores, este *framework* é essencial para a robótica moderna.

O ROS opera como um metasistema operacional de código aberto, facilitando serviços típicos de um sistema operacional, incluindo abstração de *hardware*, controle de dispositivos de baixo nível, além de oferecer funcionalidades amplamente utilizadas, como a passagem de mensagens entre processos e o gerenciamento de pacotes. Além disso, ele também fornece bibliotecas e ferramentas para adquirir, construir, escrever e executar códigos em diversos computadores.

Embora não seja um sistema operacional real, ROS funciona como uma estrutura robusta que simula as funcionalidades de um sistema operacional num *cluster* de computadores heterogêneo. Sua aplicabilidade não se limita apenas a robôs, pois a maioria de suas ferramentas é focada em trabalhar com *hardware* periférico.

Sua principal característica é como o *software* é executado e a maneira como se comunica com o *hardware*, sem a necessidade de entendê-lo profundamente. Isso é alcançado por meio da conexão de uma rede de processos, conhecidos como **nós**.

Os métodos para criar a rede incluem fornecer serviços solicitáveis ou definir conexões de *publicador/assinante* com outros nós. Estes métodos utilizam tipos de mensagens específicos, que podem ser providenciados pelos pacotes principais ou definidos por pacotes individuais. Ou seja, um sistema ROS é composto por vários nós independentes, cada um comunicando-se com os outros utilizando um modelo de mensagens de *publicador/assinante*. Por exemplo, o *driver* de um sensor específico pode ser implementado como um nó que publica dados do sensor num fluxo de mensagens. Estas podem ser consumidas por um número variado de outros nós, incluindo filtros, registradores e sistemas de nível superior, como sistemas de orientação e localização de trajetos.

Adicionalmente, o ROS é independente de linguagem, o que permite a implementação de um nó em Python e outro em C++, com ambos se comunicando eficientemente. Estas são as linguagens mais comuns, mas outras podem ser utilizadas por meio de algumas bibliotecas para criar novos nós, possibilitando a realização de comunicações entre diferentes máquinas, empregando diferentes linguagens de programação.

Normalmente, os aplicativos do ROS se comunicam através de interfaces de um dos três tipos: **tópicos, serviços ou ações**, permitindo uma troca de informações fluida e eficiente.

As seções subsequentes abordam mais detalhes sobre essas interfaces de comunicação.

3.3.1 *ROS Nodes* (Nós)

Um nó no ROS é um processo computacional que desempenha uma função específica dentro de uma aplicação robótica. Implementado como um programa executável, opera dentro de um pacote de *software*.

Eles são elementos fundamentais de qualquer projeto em ROS, organizados em um gráfico onde se comunicam entre si por meio de *tópicos, serviços ou ações*, conforme a necessidade da interação.

Cada nó atua como um participante ativo neste gráfico, utilizando bibliotecas clientes para facilitar a comunicação com outros nós. Esta interação pode ocorrer dentro do mesmo processo, entre processos distintos ou até mesmo entre diferentes máquinas, proporcionando uma flexibilidade significativa no *design* dos sistemas robóticos.

Além disso, eles podem publicar informações em tópicos específicos para compartilhar dados com outros nós, ou se inscrever em tópicos para receber dados de outros nós. Com isso, eles podem funcionar como clientes ou servidores de serviços, permitindo que computações específicas sejam solicitadas e executadas. Para tarefas que demandam mais tempo, os nós também podem atuar como clientes ou servidores de ações, facilitando operações complexas e prolongadas com a possibilidade de fornecer *feedback* contínuo.

Os nós geralmente combinam vários componentes, incluindo:

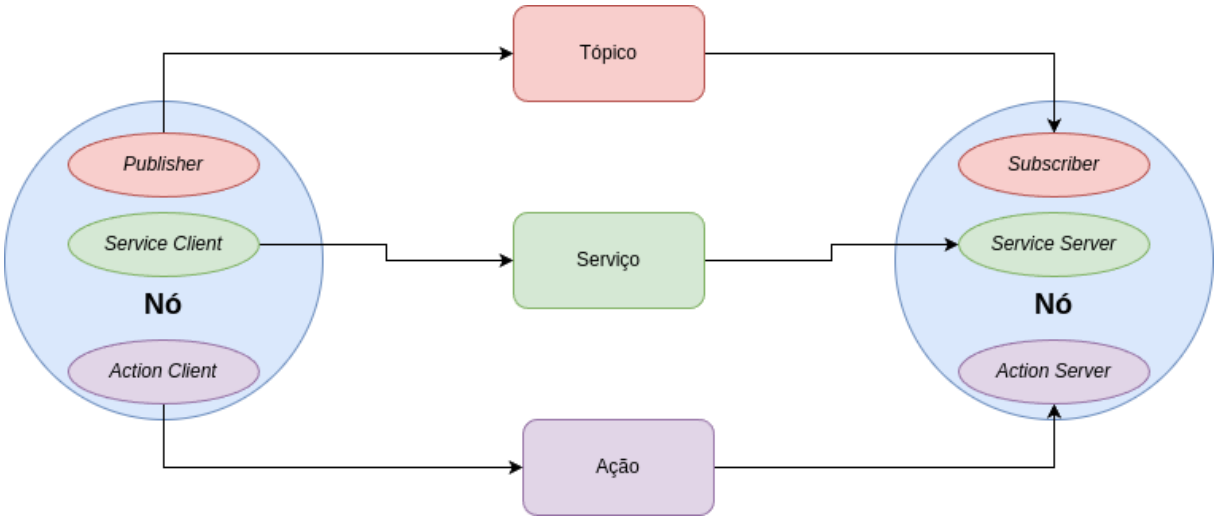
- ***Publisher* (Publicador)**: Responsável por enviar dados para tópicos específicos.
- ***Subscriber* (Assinante)**: Recebe dados de tópicos específicos.
- ***Service Client* (Cliente de Serviço)**: Solicita serviços a outros nós.
- ***Service Server* (Servidor de Serviço)**: Oferece serviços a outros nós.
- ***Action Client* (Cliente de Ação)**: Inicia operações de longa duração.
- ***Action Server* (Servidor de Ação)**: Executa operações de longa duração.

A comunicação entre os nós é estabelecida por meio de um processo de descoberta distribuída. Quando um nó é iniciado, ele anuncia sua presença na rede ROS, permitindo que outros nós reconheçam sua existência e estabeleçam conexões apropriadas para comunicação futura.

Este processo de descoberta é contínuo, garantindo que novas entidades sejam descobertas e integradas ao sistema mesmo após o período inicial de configuração. Ou

seja, os nós têm a capacidade de anunciar quando estão saindo da rede, facilitando a gestão de recursos e a manutenção do sistema. A Fig. 5, demonstra, por meio de um diagrama, o funcionamento dos nós.

Figura 5 – Representação do funcionamento dos Nós no ROS.



Fonte: Autoria própria.

3.3.2 *ROS Topics* (Tópicos)

Os tópicos desempenham um papel fundamental ao funcionar como um barramento onde os nós trocam mensagens. Este sistema divide aplicações complexas em múltiplos nós modulares, cada um desempenhando funções específicas e comunicando-se através de tópicos.

As mensagens são o método pelo qual um nó envia dados pela rede sem esperar uma resposta. Por exemplo, um nó que monitora a temperatura pode publicar essas informações em um tópico utilizando uma mensagem de temperatura. Outros nós podem assinar este tópico para receber atualizações sempre que novos dados forem publicados. Esta funcionalidade é crucial para manter os sistemas robóticos informados e responsivos em tempo real.

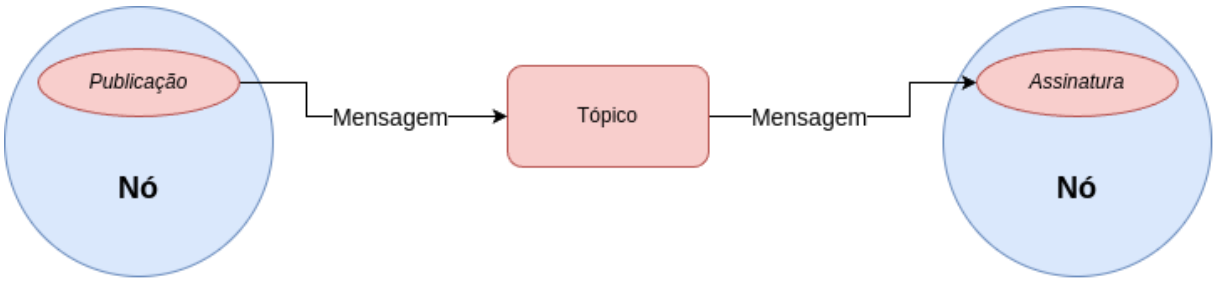
Os tópicos permitem que qualquer nó publique e se inscreva em múltiplos tópicos, facilitando uma comunicação eficiente e descentralizada entre diferentes partes do sistema robótico. Assim, eles são ideais para o fluxo contínuo de dados, como leituras de sensores ou estados dinâmicos de robôs.

O ROS opera como um sistema de publicação/assinatura fortemente tipado e anônimo. Isso significa que enquanto os produtores de dados (publicadores) e os consumidores de dados (assinantes) interagem através de tópicos nomeados, os assinantes geralmente não precisam saber a identidade dos publicadores. Esta arquitetura promove uma maior

flexibilidade e modularidade, permitindo que componentes sejam adicionados ou removidos sem perturbar o sistema como um todo.

Para gerenciar tópicos no ROS, são utilizadas várias ferramentas como `ros topic list`, `ros topic info`, e `ros topic pub`, que ajudam os desenvolvedores a listar, obter informações e publicar em tópicos, respectivamente. A Fig. 6, demonstra, por meio de um diagrama, o funcionamento dos tópicos.

Figura 6 – Representação do funcionamento dos Tópicos no ROS.



Fonte: Autoria própria.

3.3.3 *ROS Services* (Serviços)

No ambiente do ROS, os serviços constituem um método fundamental de comunicação entre nós, baseando-se em um modelo de chamada (*request*) e resposta (*response*). Diferentemente dos tópicos, que permitem subscrições a fluxos contínuos de dados, os serviços são invocados especificamente por um cliente que necessita de uma resposta imediata a uma solicitação específica.

Cada serviço é caracterizado por um tipo específico que define a estrutura das mensagens de solicitação e resposta. Essencialmente, um tipo de serviço é composto por duas partes principais: uma definição de solicitação e uma definição de resposta, facilitando assim, a realização de chamadas de procedimento remoto. No ROS, essa característica é essencial para operações que requerem uma interação síncrona entre nós, onde um nó cliente envia uma solicitação e espera que o nó servidor processe essa solicitação e retorne um resultado de forma eficiente.

Os serviços são identificados por nomes únicos, que são distintos dos nomes de tópicos. A comunicação por meio de serviços é bidirecional e estritamente definida, proporcionando um controle mais direto e imediato sobre as operações realizadas.

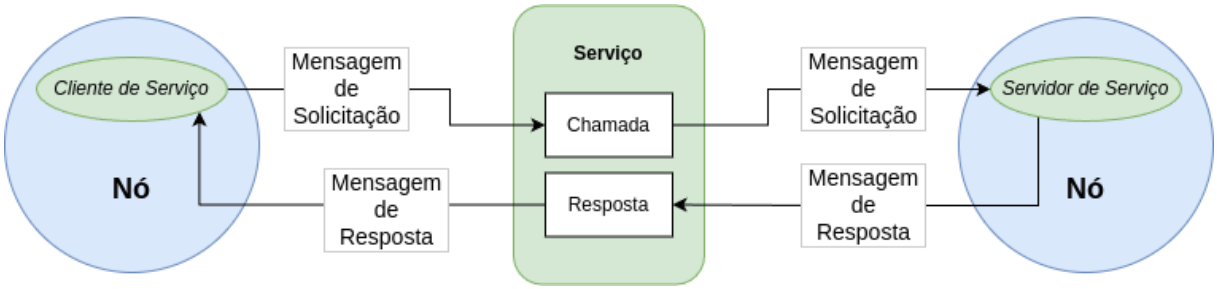
Um serviço ROS é composto por duas entidades principais:

- **Servidor de Serviço:** É a entidade que processa a solicitação recebida. Aceita as requisições de procedimento remoto e executa as operações necessárias, retornando um resultado ao cliente.

- **Cliente de Serviço:** Faz a solicitação de um serviço específico. Pode haver múltiplos clientes para um único servidor de serviço, cada um solicitando a execução de uma operação e aguardando o resultado correspondente.

Devido a essa necessidade de resposta rápida, os serviços geralmente não são adequados para processos que demandam um tempo de execução prolongado. Para tarefas mais longas, onde interrupções e *feedback* contínuo podem ser necessários, recomenda-se o uso de ações, que será apresentado adiante, em vez de serviços. A Fig. 7, demonstra, por meio de um diagrama, o funcionamento dos serviços.

Figura 7 – Representação do funcionamento dos Serviços no ROS.



Fonte: Autoria própria.

3.3.4 *ROS Actions* (Ações)

No ROS, as ações representam um tipo avançado de comunicação destinado a operações de longa duração que requerem interação contínua, *feedback* durante a execução e a capacidade de cancelamento. Uma ação é composta por três componentes principais: uma meta, um fluxo de *feedback* e um resultado final.

Elas combinam aspectos dos tópicos e dos serviços. Sendo baseadas em um modelo de cliente-servidor, mas com a capacidade adicional de cancelamento e fornecimento contínuo de *feedback*, diferenciando-as significativamente dos serviços que oferecem apenas uma única resposta. Essa capacidade as tornam ideais para tarefas que não só levam mais tempo para serem concluídas, mas também aquelas que podem necessitar de ajustes durante sua execução.

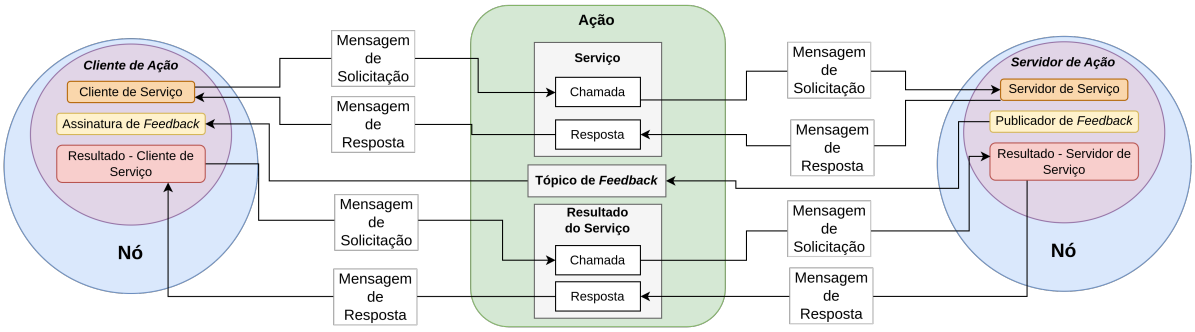
Um exemplo típico de seu uso é na navegação robótica, onde uma ação pode ser configurada para mover um robô a uma posição específica. Durante este processo, o robô, através do nó servidor de ação, pode enviar atualizações regulares sobre seu progresso, como distância percorrida ou obstáculos encontrados, antes de finalmente enviar uma resposta quando o destino for alcançado. Este método assegura que operações complexas sejam monitoradas e ajustadas conforme necessário, aumentando a eficácia e segurança da operação robótica.

A estrutura de uma ação inclui dois componentes chave:

- **Servidor de Ação:** Este é responsável por receber a solicitação de ação, processá-la e fornecer *feedback* regular ao cliente. Além disso, o servidor de ação gerencia solicitações de cancelamento ou alterações durante a execução da tarefa.
- **Cliente de Ação:** O cliente solicita a execução de uma ação e pode continuar a receber *feedback* até a conclusão da tarefa. Os clientes podem iniciar, modificar ou cancelar ações conforme necessário, proporcionando flexibilidade na execução de tarefas complexas.

É mostrado, na Fig. 8, o diagrama de funcionamento das ações.

Figura 8 – Representação do funcionamento das Ações no ROS.



Fonte: Autoria própria.

3.4 Gazebo Clássico

O Gazebo é um simulador robótico 2D/3D de código aberto, desenvolvido inicialmente em 2002 na Universidade do Sul da Califórnia por Dr. Andrew Howard e seu aluno Nate Koenig. O simulador foi criado para atender à necessidade de testar robôs em ambientes externos sob diversas condições atmosféricas, com um alto grau de fidelidade (ROBOTICS, 2024b).

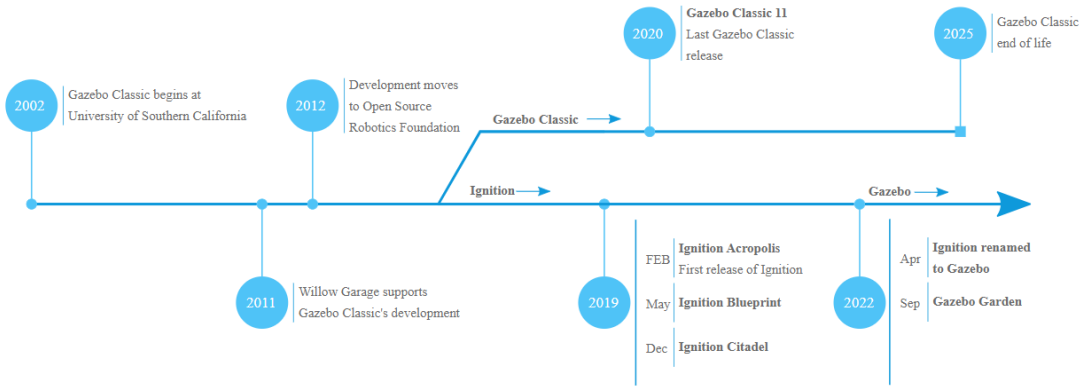
Ele oferece uma abordagem inovadora na simulação robótica, incorporando um conjunto completo de ferramentas de desenvolvimento e serviços na nuvem, além de poder modelar sensores que "veem" o ambiente simulado, como medidores de distância a *laser*, câmeras, entre outros. Isso facilita a simulação de projetos físicos em ambientes realistas, com simulação de alta fidelidade e teste de estratégias de controle em um ambiente seguro (ROBOTICS, 2024a).

O *software* encapsula um vasto conjunto de bibliotecas de desenvolvimento de código aberto que incluem funcionalidades essenciais como tipos de dados matemáticos comuns, registro de atividades, gerenciamento de malhas 3D e passagem de mensagens assíncronas. Além disso, Gazebo suporta uma variedade de funcionalidades:

- **Simulação:** Simulador avançado de robôs para pesquisa, design e desenvolvimento.
- ***Plugins*:** Possibilidade de desenvolver *plugins* personalizados para controle de robôs e ambientes simulados.
- **Transporte:** Passagem de mensagens assíncrona distribuída.
- **Sensores:** Conjunto extenso de modelos de sensores e ruídos.
- **Física:** Interface baseada em *plugins* para motores de física.
- **Renderização:** Interface baseada em *plugins* para motores de renderização.
- **Modelos de Robôs:** Disponibilidade de diversos modelos de robôs e suporte para construção de modelos customizados usando *Spatial Data File* (SDF).

Em 2017, o desenvolvimento do Gazebo se dividiu em duas vertentes: o "*Gazebo Classic*", que manteve a arquitetura monolítica original, e o "*Ignition*", que representa uma evolução para uma coleção de bibliotecas pouco acopladas, adaptando-se melhor às novas exigências da robótica (WIKIPEDIA, 2024).

Figura 9 – Linha do Tempo da História do Gazebo Clássico.



Fonte: (ROBOTICS, 2024a).

O Gazebo se estabeleceu como uma ferramenta essencial para roboticistas, proporcionando um ambiente de simulação robusto, que permite a realização de testes de regressão, design de robôs e treinamento de sistemas de *AI* em cenários altamente realistas. Sua natureza de código aberto e a comunidade ativa contribuem significativamente para seu contínuo desenvolvimento e aprimoramento.

De acordo com (ROBOTICS, 2014), a Interface Gráfica do Usuário (GUI) do Gazebo, possui vários componentes:

1. **Cena:** Onde os objetos simulados são animados e com os quais você interage.

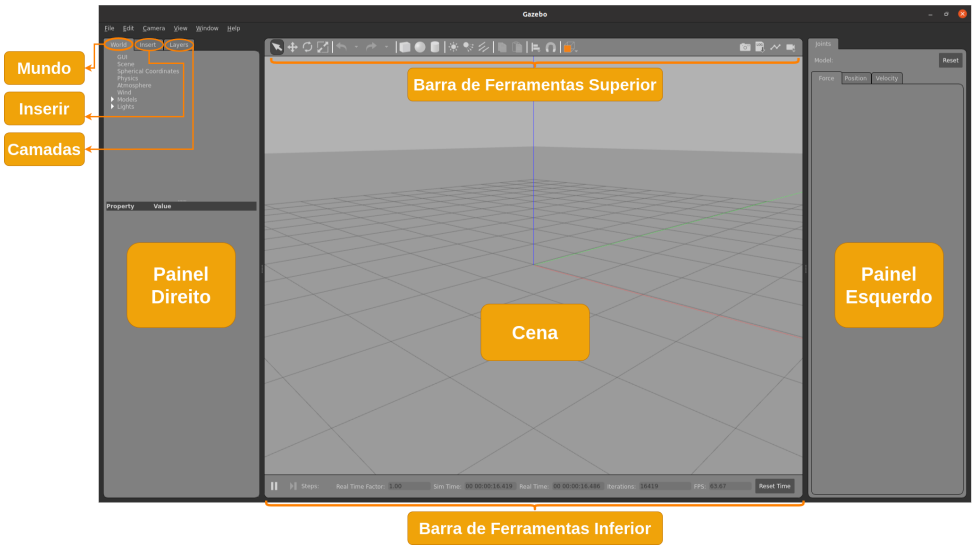
2. **Painéis Laterais:** Painéis à direita e à esquerda que podem ser exibidos, ocultados ou redimensionados. O painel direito pode ser usado para interagir com as partes móveis de um modelo selecionado, como por exemplo, as juntas de um robô. O painel à esquerda possui três abas:

    - **Mundo (*World*)**: Exibe os modelos que estão atualmente na cena, permitindo a visualização e modificação dos parâmetros do modelo, como sua pose.
    - **Inserir (*Insert*)**: Adiciona novos objetos (modelos) à simulação.
    - **Camadas (*Layers*)**: Organiza e exibe os diferentes grupos de visualização disponíveis na simulação. Uma camada pode conter um ou mais modelos. Ativar ou desativar uma camada exibirá ou ocultará os modelos nessa camada.

3. **Barra de Ferramentas Superior:** Inclui opções para interagir com a simulação, como selecionar, mover, rotacionar e escalar objetos, além de inserir formas simples e manipular luzes.

4. **Barra de Ferramentas Inferior:** Exibe informações sobre o tempo de simulação e sua relação com o tempo real.

Figura 10 – GUI do Gazebo.



Fonte: Autoria Própria.

Essas ferramentas permitem uma manipulação detalhada e interativa dos modelos e ambientes dentro do simulador Gazebo.

3.5 *ROS Visualization* (Rviz)

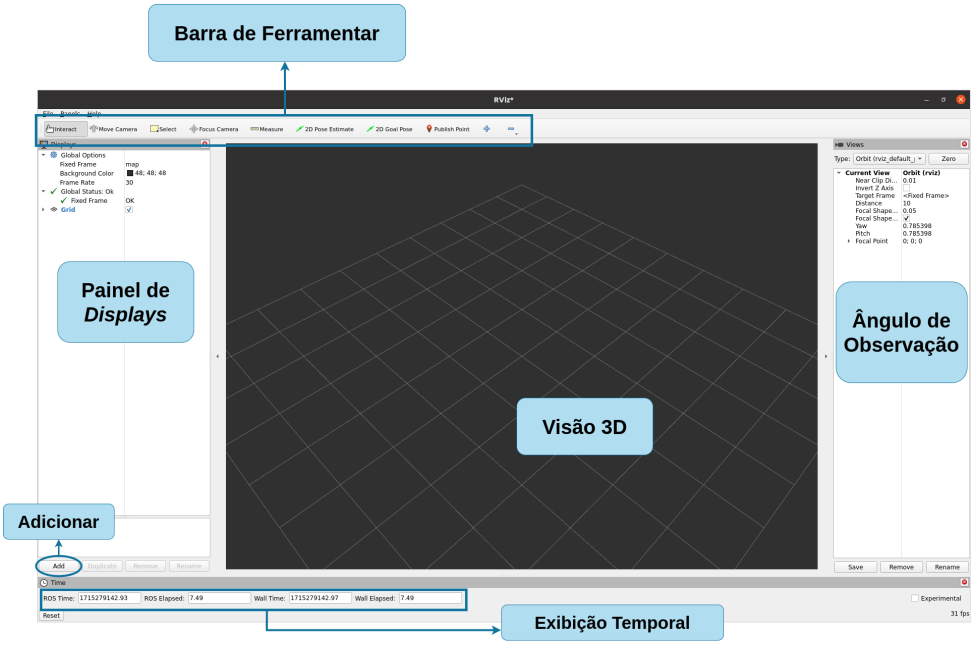
O RViz é uma ferramenta de visualização 3D essencial para a representação de robôs, sensores e algoritmos. Sua principal função é permitir a visualização da percepção do estado do robô em seu mundo, tanto real quanto simulado, fornecendo assim uma representação detalhada do ambiente operacional do robô (SEARS-COLLINS, 2020).

Ele se trata de uma plataforma integrada ao ROS que permite tanto a visualização de dados externos quanto o envio de comandos de controle a objetos. Em seu ambiente, o usuário pode interagir com a visualização 3D através de um conjunto diversificado de ferramentas e opções.

Sua GUI possui vários componentes:

- **Painel de *Displays*:** Localizado à esquerda, inclui uma lista de *plugins* para visualizar dados de sensores e informações de estado do robô. *Plugins* adicionais podem ser incorporados usando o botão "Adicionar".
- **Barra de Ferramentas:** Situada no topo, oferece acesso a ferramentas multifuncionais que facilitam a manipulação dos dados visualizados.
- **Visão 3D:** A área central do RViz, onde os dados são exibidos em três dimensões. Configurações como cor de fundo, quadro fixo e parâmetros da grade são ajustáveis nas opções globais.
- **Área de Exibição Temporal:** Localizada abaixo da visão 3D, mostra informações temporais relevantes, incluindo o tempo do sistema e do ROS.
- **Configuração do Ângulo de Observação:** À direita, permite ao usuário ajustar o ângulo de visualização para otimizar a perspectiva observada.

Figura 11 – GUI do Rviz.



Fonte: Autoria Própria.

Com isso, ele é uma ferramenta fundamental dentro da comunidade ROS para diagnósticos e visualização de dados em 3D. Embora possua limitações como interface de controle, sua capacidade de expandir visualizações personalizadas é inestimável para desenvolvedores e pesquisadores em robótica.

3.6 *Real-Time Appearance-Based Mapping* (RTAB-Map)

O RTAB-Map, ou Mapeamento Baseado na Aparência em Tempo Real, é uma tecnologia utilizada em robótica para ajudar máquinas a entender e navegar por seu ambiente de forma autônoma. O termo SLAM (*Simultaneous Localization and Mapping*), que significa Mapeamento e Localização Simultâneos, é frequentemente associado ao RTAB-Map, pois descreve sua capacidade de criar um mapa de um ambiente desconhecido enquanto simultaneamente se localiza dentro desse mapa.(INTROLAB, 2024)

O RTAB-Map utiliza três principais tecnologias de sensores para perceber o ambiente:

- **RGB-D (*Red, Green, Blue - Depth*)**: Câmeras que capturam informações de cor e profundidade.
- ***Stereo***: Uso de duas ou mais câmeras para capturar imagens tridimensionais.
- ***Lidar***: Tecnologia que usa laser para medir distâncias e criar representações precisas do ambiente.

Um aspecto crítico do RTAB-Map é o "fechamento de laço", no inglês, *loop closure detector*. Este processo refere-se à habilidade do sistema de reconhecer um local previamente visitado, usando um método conhecido como "abordagem saco de palavras", também conhecido como *bag-of-words approach*. Nessa abordagem, ele examina as imagens capturadas para verificar se a nova imagem capturada parece com alguma que já foi registrada anteriormente e compara descrições visuais das imagens para determinar semelhanças.

Quando é reconhecido um local já visitado, o sistema adiciona uma nova "restrição" ao grafo do mapa. Essas restrições ajudam a corrigir o mapa, minimizando erros através de um processo chamado otimização de grafo (*graph optimization*), que reajusta as relações e distâncias entre os pontos mapeados para garantir a precisão. Isso é crucial para corrigir erros no mapa que podem acumular com o tempo devido ao movimento contínuo do robô.

O RTAB-Map incorpora uma estratégia de gestão de memória que permite limitar o número de locais armazenados, essencial para manter a eficiência em ambientes grandes e complexos. Isso garante que o sistema possa operar em tempo real, sem perder desempenho. Além disso, ele possui uma GUI que permite ao usuário visualizar os mapas 3D gerados, bem como acompanhar em tempo real a trajetória e os pontos de referência detectados pelo sistema de mapeamento.

Figura 12 – GUI do RTAB-Map.

Fonte: Autoria Própria.

Esta tecnologia é altamente versátil e pode ser aplicada em vários contextos, desde pequenos robôs de pesquisa até veículos autônomos, sendo uma ferramenta poderosa na robótica moderna, permitindo que dispositivos naveguem e interajam com precisão em ambientes desconhecidos.

3.7 Metodologias Ágeis: *Scrum*

Scrum é uma metodologia ágil de gerenciamento de projetos, particularmente adequada para projetos de desenvolvimento de *software* que exigem rapidez e flexibilidade. Operando através de ciclos de trabalho chamados *sprints*, ele promove um processo iterativo e incremental que dura de duas a quatro semanas. As *sprints* facilitam o planejamento rápido, a execução e a revisão de tarefas, permitindo ajustes frequentes com base no *feedback* das partes interessadas e mudanças no escopo do projeto. Fundamentalmente, *Scrum* apoia-se em três pilares: transparência, inspeção e adaptação,o que facilita a integração e o autogerenciamento da equipe.(SCHWABER; SUTHERLAND, 2020)

3.8 Linux

Linux, introduzido por Linus Torvalds em 1991, é um sistema operacional tipo *Unix* de código aberto. Distingue-se pela sua compatibilidade com diversas plataformas de *hardware*, de computadores pessoais, a supercomputadores e dispositivos embarcados. Sua arquitetura modular e o modelo de desenvolvimento liderado pela comunidade permitem atualizações e melhorias contínuas, assegurando uma operação segura e eficiente. Além disso, Linux é amplamente celebrado por sua robustez em segurança, que é fortalecida por uma comunidade ativa que rapidamente identifica e corrige vulnerabilidades.(TORVALDS; DIAMOND, 1997)

3.9 Sistema de Controle de Versão: Git

Desenvolvido, também, por Linus Torvalds em 2005, Git[1] é um sistema de controle de versão distribuído que permite aos desenvolvedores gerenciar eficientemente grandes projetos de *software*. Ele é projetado para proporcionar velocidade, integridade de dados e suporte para fluxos de trabalho distribuídos e não lineares. Uma de suas características mais notáveis é a capacidade de se ramificar e se mesclar rapidamente, facilitando o desenvolvimento paralelo de recursos, correções de *bugs* e experimentação. Além disso, Git oferece suporte para trabalhar *offline* e pode sincronizar as mudanças entre diferentes repositórios quando reconectado à rede.(TORVALDS, 2005)

[1] <https://git-scm.com/>

4 Atividades Realizadas

Nesta seção são descritas as principais atividades realizadas durante o período de estágio, fornecendo uma visão geral abrangente. Além disso, são relatadas as etapas de treinamento e o processo de desenvolvimento de simulações de robôs.

4.1 Visão Geral

Durante o estágio no Laboratório Embedded, a estagiária participou ativamente da equipe de desenvolvimento de projetos focados em robótica. A sua contribuição central foi no desenvolvimento de simulações para dois tipos distintos de robôs: o braço robótico *Universal Robots UR10* e o robô móvel *Turtlebot2i*, ambos simulados no Gazebo através do ROS. Também foi executada a simulação de uma câmera de profundidade, realizada com o objetivo de executar o mapeamento tridimensional do ambiente simulado e convertê-lo em uma nuvem de pontos.

Além das simulações, o envolvimento da estagiária incluiu pesquisas e investigações. Com isso, ela participou da análise e da organização de uma lista de requisitos e especificações técnicas necessárias para o avanço do projeto. Realizando, assim, documentações dos processos desenvolvidos durante o projeto e participação em atividades diárias de trabalho junto com a equipe, incluindo a realização de reuniões diárias (*dailys*), revisões (*review*) e planejamento (*plainning*) com o time. A estagiária contribuiu para a academia, participando no desenvolvimento de um artigo científico que aborda métodos para a sincronização de GDs.

4.2 Capacitação em Robótica

A fim de capacitar a estagiária no desenvolvimento de uma prova de conceito (*PoC*, do inglês *Proof of Concept*) com foco no desenvolvimento de simulações de robôs e sensores, foram realizadas as seguintes atividades:

4.2.1 Desenvolvimento dos URDFs

O URDF é um formato de arquivo *Extensible Markup Language* (XML), que desempenha um papel crucial na robótica ao descrever todos os elementos de um robô. Esse formato abrange desde a geometria, cinemática e dinâmica do robô até suas propriedades visuais e sensores, sendo amplamente adotado em ambientes de simulação.

A importância do URDF se estende a vários aspectos, como:

- **Modelagem Completa:** O URDF facilita a definição da estrutura física do robô, incluindo os *links* (partes do robô) e *joints* (articulações), o que é essencial para o cálculo de movimentos e interação com o ambiente.
- **Simulação de Sensores:** Através do URDF, sensores como *Lidar*, câmeras e sensores táteis podem ser integrados ao modelo do robô. Isso possibilita a simulação da interação dos sensores com o ambiente, coleta de dados e sua utilização em navegação e outras tarefas operacionais.

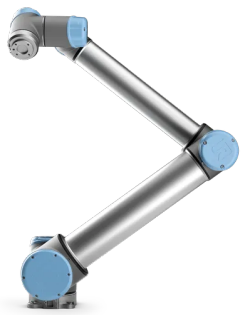
Durante o estágio, foram desenvolvidos 4 arquivos URDFs, com o objetivo de descrever os robôs, a sala em que o projeto foi executado e a câmera.

4.2.1.1 Robô *Universal Robots UR10*

A elaboração do arquivo URDF para o braço robótico *Universal Robots UR10* foi baseada na documentação fornecida pela fabricante, que está disponível no *GitHub*[1]. A empresa *Universal Robots* criou um pacote específico para facilitar a simulação de seus braços robóticos no Gazebo e Rviz, integrados ao ROS. Esse recurso foi utilizado pela estagiária, permitindo-lhe economizar tempo e otimizar o processo, ao invés de desenvolver um modelo do zero que já existia.

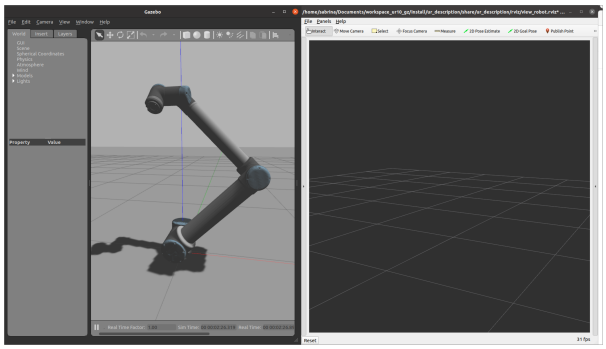
Como ilustração do sucesso deste trabalho, a Fig.13 apresenta o robô UR10 em seu ambiente operacional real e a Fig.14 sua representação virtual no Gazebo e no Rviz.

Figura 13 – Robô UR10 Real



Fonte: (Universal Robots, 2024)

Figura 14 – Robô UR10 no Gazebo e no Rviz



Fonte: Autoria própria.

4.2.1.2 Robô *Turtlebot2i* e Câmera *Astra Pro Plus*

O desenvolvimento do arquivo URDF para o robô móvel *Turtlebot2i* e da Câmera *Astra Pro Plus* também se baseou na documentação disponibilizada pela fabricante no *GitHub*[2]. A empresa *Interbotix* elaborou um pacote específico que facilita a simulação

[1] <https://github.com/UniversalRobots/Universal_Robots_ROS2_Driver>
[2] <https://github.com/Interbotix/turtlebot2i>

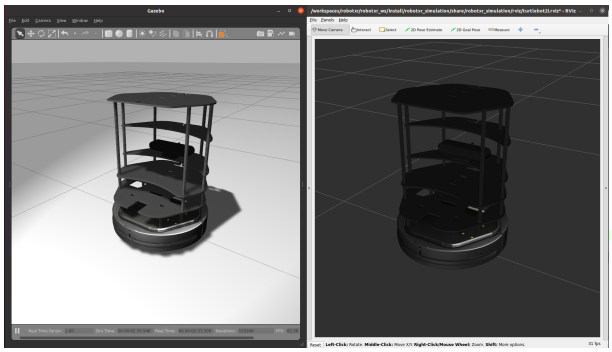
deste robô e no Gazebo e Rviz, utilizando o ROS. Ela também disponibilizou o URDF da câmera. O braço robótico e a câmera superior do robô não foram utilizados na aplicação do estágio.

A Fig.15 exibe o robô *Turtlebot2i* em sua forma física e a Fig.16 mostra representação simulada no Gazebo e no Rviz.

Figura 15 – Robô *Turtlebot2i* Real

Fonte: (TurtleBot, 2024)

Figura 16 – *Turtlebot2i* no Gazebo e no Rviz



Fonte: Autoria própria.

4.2.1.3 Sala 03 do Embedded

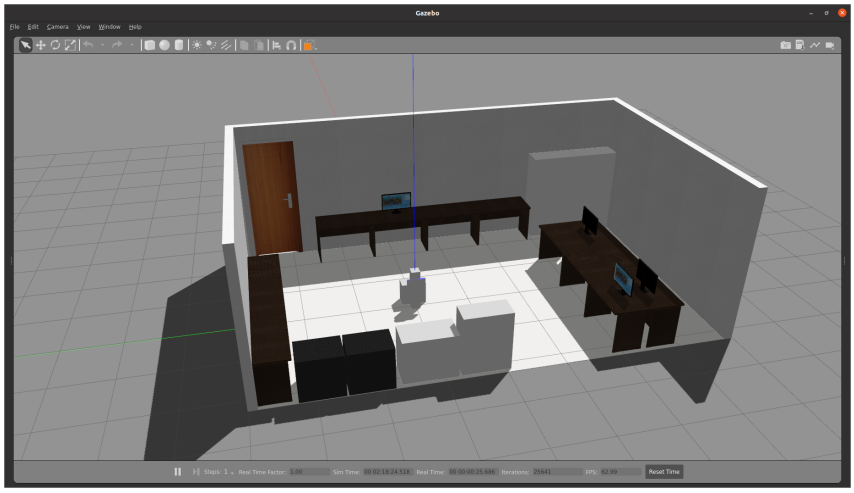
Para criar o modelo URDF da sala 03 do Laboratório Embedded, foi essencial medir todos os objetos presentes, incluindo paredes, portas, mesas, cadeiras e a disposição espacial entre eles. Essas medições precisas asseguram que a simulação reproduza de maneira fiel o ambiente real.

No desenvolvimento da parte visual, optou-se por utilizar modelos de paredes, pisos, mesas e armários, além de monitores de computador disponibilizados pela comunidade do Gazebo no *GitHub*[3]. Com esses modelos à disposição, procedeu-se à construção do arquivo URDF, posicionando cada elemento virtual conforme sua localização na sala real.

O resultado desse trabalho é ilustrado na Fig.17.

[3] <https://github.com/leonhartyao/gazebo_models_worlds_collection>

Figura 17 – Representação da Sala 03 do Embedded no Gazebo.



Fonte: Autoria própria.

4.3 Sincronização do Gêmeo Digital do UR10

Durante o período de estágio, a estagiária atuou auxiliando no desenvolvimento de um artigo científico que propõe uma estratégia de sincronização em tempo real, que utiliza controle cinemático para otimizar a interação entre um Gêmeo Digital (GD) e seu correspondente físico, concentrando-se especificamente em um braço robótico do modelo *Universal Robots UR10*. Ao explorar os princípios de controle cinemático, o objetivo é harmonizar os estados do UR10 e seu GD em tempo real. Esta estratégia de sincronização incorpora o ROS, aproveitando suas funcionalidades de Qualidade de Serviço (QoS) para o controle eficiente de GDs.

A sincronização em tempo real destaca a capacidade do GD de refletir o status contínuo e as atividades do sistema físico que simula. Para um GD de um robô, a sincronização em tempo real é crucial para simulações precisas, análises preditivas, supervisão remota, operação e treinamento. Ela estabelece uma conexão robusta entre os reinos virtual e físico, aumentando a eficácia, segurança e eficiência dos sistemas robóticos.

A implementação do GD foi realizada utilizando o ROS e a simulação foi conduzida no ambiente do Gazebo. A sincronização foi efetivada através de um algoritmo de controle cinemático e *feedback* de estado, avaliando a eficiência do nosso método com um braço UR10 e ROS no Gazebo.

A estratégia de sincronização foi aplicada a partir da configuração inicial dos parâmetros necessários para a simulação, incluindo ajustes de tempo e localização, usando *scripts* em Python para configurar o ambiente de lançamento no ROS. O GD do braço robótico UR10 foi controlado com parâmetros de calibração adquiridos do robô real, garantindo que a simulação refletisse com precisão o comportamento do braço físico. O

trabalho completo pode ser encontrado em (NASCIMENTO et al., 2023).

4.4 Desenvolvimento do Mapeamento Utilizando o RTAB-Map

Para efetuar o mapeamento utilizando o RTAB-Map, foi crucial simular uma câmera de profundidade que se assemelhasse ao máximo à Câmera *Astra Pro Plus* da *Orbbec*. Por meio do *plugin* `libgazebo_ros_camera.so`, integrou-se esta simulação ao ambiente Gazebo juntamente com o ROS, configurando a câmera para espelhar as especificações requeridas para uma representação acurada dentro da simulação. O *plugin* serve como uma conexão entre o simulador Gazebo e o ROS, facilitando o acesso a dados de sensores simulados através das interfaces padrão do ROS.

Suas principais características são:

- **Simulação de Câmeras:** O *plugin* simula uma câmera no ambiente Gazebo, gerando dados de imagem que imitam aqueles que uma câmera real capturaria. Isso inclui a geração de imagens RGB e, dependendo da configuração, dados de profundidade.
- **Publicação de Imagens e Informações no ROS:** As imagens capturadas pela câmera simulada são publicadas em um tópico do ROS, permitindo que outros nós consumam essas imagens para processamento ou controle. Informações detalhadas da câmera também são publicadas, o que é fundamental para várias tarefas de visão computacional.
- **Configuração Flexível:** É possível configurar a simulação da câmera usando parâmetros do ROS, ajustando aspectos como taxa de atualização, resolução da imagem e campo de visão, o que permite adaptar a simulação a diferentes tipos de câmeras e exigências.

A simulação da câmera foi ajustada para refletir as características técnicas do modelo real, conforme apresentado na Tabela 1.

Tabela 1 – Especificações técnicas da câmera *Astra Pro Plus*.

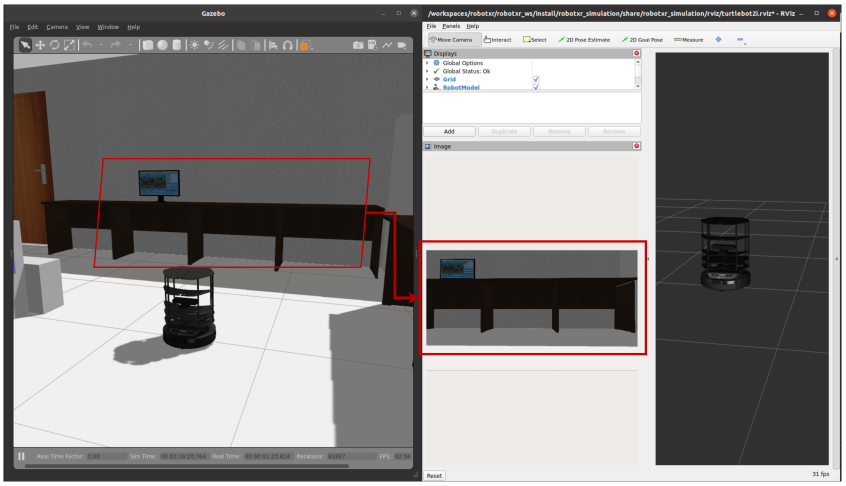
Característica	**Valor**
Campo de Visão (FOV)	73° horizontal
Resolução	1920x1080
Taxa de Quadros	30 fps
Distância de Visualização	0.6m a 8m

Fonte: (Orbbec, 2024)

Na Fig.18, à esquerda, é possível ver o robô *Turtlebot2i* no Gazebo. Ele possui o URDF da câmera *Astra Pro Plus* e foi inserido o *plugin* para simulá-la. A visão da câmera

são as mesas (demarcado pelo losango vermelho). À direita, é possível ver a imagem que capturada no Gazebo no RViz.

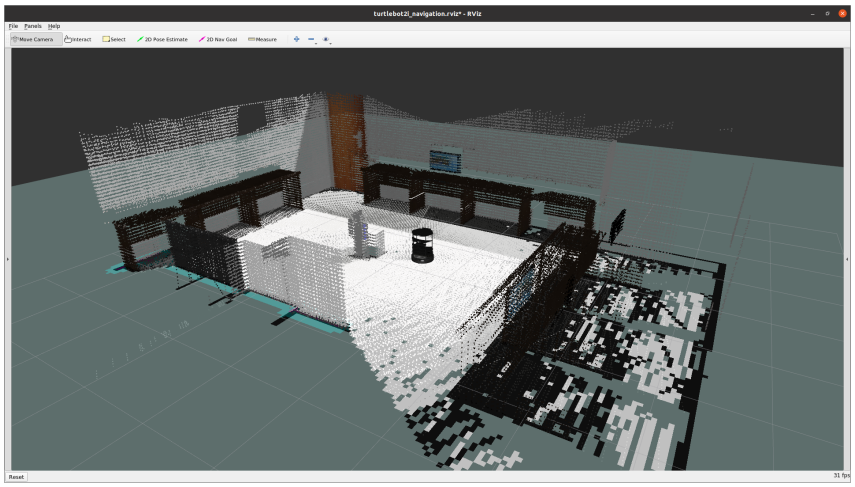
Figura 18 – À esquerda, visão da câmera no Gazebo; à direita, imagem capturada pela câmera no RViz.



Fonte: Autoria própria.

A implementação do RTAB-Map para o ROS iniciou-se com a configuração do ambiente de lançamento, definindo parâmetros essenciais para a simulação. Utilizando arquivos de configuração escritos em Python, foi estabelecido um sistema para capturar e processar dados tridimensionais da câmera simulada, conduzindo ao mapeamento do ambiente. A Fig.19 ilustra o mapeamento realizado.

Figura 19 – Mapeamento em nuvem de pontos vista no Rviz.



Fonte: Autoria própria.

O mapeamento foi realizado por meio de diversos nós do ROS, responsáveis pela odometria RGB-D e pelo mapeamento SLAM. A odometria RGB-D combina informações de câmeras RGB com sensores de profundidade para criar uma representação mais

precisa do ambiente, permitindo ao robô calcular sua posição atual baseada em um local inicialmente conhecido. Com esta técnica é possível ter uma estimativa de movimento e localização do robô em ambientes dinâmicos. O SLAM, por sua vez, permite que o robô crie ou atualize um mapa de um ambiente desconhecido enquanto simultaneamente se localiza neste mapa. Este processo combina fontes de dados sensoriais para construir progressivamente um mapa do entorno e usar esse mesmo mapa para refinar suas estimativas de localização. A visualização e interpretação dos dados coletados foram facilitadas pelo uso do RViz.

Essas atividades não apenas demonstraram a capacidade da estagiária em integrar sistemas e realizar simulações, mas também contribuíram significativamente para seu desenvolvimento profissional na área da robótica.

4.4.1 Outras atividades

Durante o desenvolvimento do projeto, a estagiária identificou a importância vital da documentação para registrar e compartilhar informações pertinentes sobre o projeto. Ela se empenhou em elaborar relatórios de progresso detalhados, que não apenas delineavam as etapas já completadas, mas também discutiam os desafios enfrentados e as soluções encontradas. Esses relatórios foram complementados com a criação de diagramas de arquitetura, esquemas de conexões de hardware e descrições detalhadas de cada componente utilizado, melhorando significativamente o entendimento do projeto e facilitando a colaboração com outros integrantes da equipe.

Para otimizar o gerenciamento do projeto e o controle de versões do código, foi escolhido o uso do *GitLab*, uma plataforma robusta de gerenciamento de código. A estagiária estabeleceu um repositório para o projeto, proporcionando um meio eficaz para controlar versões do código e monitorar alterações feitas. Através do *GitLab*, ela pôde colaborar com outros membros da equipe, revisando e mesclando código de forma eficiente e controlando o avanço das tarefas. Ademais, o repositório incluiu toda a documentação e descrições das atividades desenvolvidas, abarcando todos os testes realizados em cada etapa do projeto.

5 Considerações Finais

Neste relatório foram descritas as atividades realizadas durante o estágio supervisionado no Laboratório Embedded em parceria com o VIRTUS, como parte integrante da disciplina curricular. A estagiária cumpriu a carga horária proposta, executando as tarefas com êxito. As experiências adquiridas ao longo deste período foram fundamentais para a aplicação prática dos conhecimentos teóricos, com destaque para a utilização do ROS em projetos de simulação robótica.

Durante o estágio, foi dada atenção à simulação e sincronização de Gêmeos Digitais usando ROS e Gazebo, que permitiram modelar comportamentos de robôs como o *Universal Robots UR10* e o *Turtlebot2i* em ambientes controlados. Essas simulações foram cruciais para o desenvolvimento de habilidades em programação robótica e manipulação de sensores e atuadores em um contexto virtual, que é um substituto essencial para testes em ambientes reais.

A estagiária não apenas implementou algoritmos de navegação e mapeamento, como também contribuiu para a otimização do uso de câmeras de profundidade com o RTAB-Map com o ROS. Este trabalho demonstrou como ferramentas de simulação podem ser efetivamente utilizadas para melhorar a precisão dos sistemas robóticos e para prever resultados antes da implementação física.

Este estágio também reforçou a importância do aprendizado contínuo e da adaptação às rápidas mudanças tecnológicas que caracterizam o campo da robótica. O uso de *GitLab* para documentação e controle de versão do código assegurou que todos os aspectos do projeto fossem replicáveis e bem documentados.

Em resumo, a experiência adquirida com a simulação ROS e Gazebo forneceu não somente a competência técnica necessária para enfrentar desafios futuros, mas também uma forte compreensão de como teoria e prática se interligam na solução de problemas de engenharia. Este estágio foi uma etapa valiosa na preparação da estagiária para contribuições futuras na área de robótica e automação.

Referências

Autodesk. *Gêmeo Digital: Entenda o que é, como funciona e suas aplicações*. 2024. <https://www.autodesk.com/br/design-make/articles/gemeo-digital>. Acesso em: 09 Maio 2024.

AZHEGANOVA, E. et al. Digital twin: Manufacturing excellence through virtual factory replication. *University of Michigan Executive Course*, 2020.

BENOTSMANE, R.; KOVÁCS, G.; DUDÁS, L. Economic, social impacts and operation of smart factories in industry 4.0 focusing on simulation and artificial intelligence of collaborating robots. *Social Sciences*, MDPI, v. 8, n. 5, p. 143, 2019.

DELOITTE. *Digital twin applications bridging the physical and digital*. 2020. Disponível em: <https://www2.deloitte.com/us/en/insights/focus/tech-trends/2020/digital-twin-applications-bridging-the-physical-and-digital.html>. Acesso em: 20 Abr. 2024.

EMBEDDED. *Laboratório de Sistemas Embarcados e Computação Pervasiva da UFCG*. 2023. Disponível em: <https://www.embedded.ufcg.edu.br/>. Acesso em: 07 Abr. 2024.

GRIEVES, M. Digital twin: Manufacturing excellence through virtual factory replication; a white paper; michael grieves, llc: Melbourne, fl, usa. 2014.

GUPTA, R. et al. Machine learning applications on iot data in manufacturing operations and their interpretability implications: A systematic literature review. *Journal of Manufacturing Systems*, 2020.

IBM. *What is Industry 4.0?* 2020. Disponível em: <https://www.ibm.com/topics/industry-4-0>. Acesso em: 22 Abr. 2024.

INTROLAB. *RTAB-Map: Real-Time Appearance-Based Mapping*. 2024. Disponível em: <https://introlab.github.io/rtabmap/#:~:text=RTAB%2DMap%20(Real%2DTime,location%20or%20a%20new%20location.> Acesso em: 08 Mai. 2024.

NASCIMENTO, F. H. et al. Synchronizing a collaborative arm's digital twin in real-time. In: IEEE. *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*. [S.l.], 2023. p. 230–235.

NEDASHKOVSKIY, A. et al. Project research report: Application of digital twin in industry 4.0. 2020.

Orbbec. *Astra Pro Plus*. 2024. <https://shop.orbbec3d.com/Astra-Pro-Plus>. Acesso em: 10 Maio 2024.

ROBOTICS, O. *Beginner: GUI*. 2014. Disponível em: <https://classic.gazebosim.org/tutorials?cat=guided_b&tut=guided_b2>. Acesso em: 29 Abr. 2024.

ROBOTICS, O. *About Gazebo*. 2024. Disponível em: <https://gazebosim.org/about>. Acesso em: 29 Abr. 2024.

ROBOTICS, O. *Gazebo Simulator*. 2024. Disponível em: <https://gazebosim.org/home>. Acesso em: 29 Abr. 2024.

SCHWABER, K.; SUTHERLAND, J. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. [S.l.]: Scrum.org, 2020.

SEARS-COLLINS, A. *What is the Difference Between RViz and Gazebo?* 2020. Disponível em: <https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/>. Acesso em: 07 Mai. 2024.

TORVALDS, L. *Git*. 2005. <https://git-scm.com/>.

TORVALDS, L.; DIAMOND, D. *Just for Fun: The Story of an Accidental Revolutionary*. [S.l.]: HarperCollins, 1997.

TurtleBot. *TurtleBot2*. 2024. <https://www.turtlebot.com/turtlebot2/>. Acesso em: 10 Maio 2024.

Universal Robots. *CB3 Robot Arm*. 2024. <https://www.universal-robots.com/products/cb3/>. Acesso em: 10 Maio 2024.

VIRTUS. *VIRTUS - Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação*. 2023. Disponível em: <https://www.virtus.ufcg.edu.br/>. Acesso em: 07 Abr. 2024.

WIKIPEDIA. Gazebo simulator. 2024. Disponível em: <https://en.wikipedia.org/wiki/Gazebo_simulator>.