INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ

CAMPUS TERESINA CENTRAL

CURSO BACHARELADO EM ENGENHARIA MECÂNICA

MATHEUS ARAUJO DANTAS

APLICAÇÃO DE UM ROBÔ ASPIRADOR COMO PLATAFORMA ROBÓTICA PARA PRÁTICAS EM LABORATÓRIO DA ENGENHARIA MECÂNICA DO INSTITUTO FEDERAL DO PIAUÍ

TERESINA

2022

MATHEUS ARAUJO DANTAS

APLICAÇÃO DE UM ROBÔ ASPIRADOR COMO PLATAFORMA ROBÓTICA PARA PRÁTICAS EM LABORATÓRIO DA ENGENHARIA MECÂNICA DO INSTITUTO FEDERAL DO PIAUÍ

Trabalho de Conclusão de Curso (monografia) apresentado como exigência parcial para obtenção do diploma do Curso de Bacharelado em Engenharia Mecânica do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central.
Orientador: Prof. Me. Francisco Marcelino Almeida de Araújo.

TERESINA

2022

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD

Dantas, Matheus Araújo

D192a Aplicação de um robô aspirador como plataforma robótica do Instituto Federal do Piauí / Matheus Araújo Dantas. - 2022.

50 f.: il. color.

Trabalho de Conclusão de Curso (Bacharelado em Engenharia Mecânica) - Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central, 2022.

Orientador : Prof Me. Francisco Marcelino Almeida de Araújo.

1. Robótica. 2. Roomba. 3. Engenharia. 4. ROS. I.Título.

CDD - 620

Elaborado por Maria Rosismar Farias CRB 3/631

MATHEUS ARAUJO DANTAS

APLICAÇÃO DE UM ROBÔ ASPIRADOR COMO PLATAFORMA ROBÓTICA PARA PRÁTICAS EM LABORATÓRIO DA ENGENHARIA MECÂNICA DO INSTITUTO FEDERAL DO PIAUÍ

Trabalho de Conclusão de Curso (monografia) apresentado como exigência parcial para obtenção do diploma do Curso de Bacharelado em Engenharia Mecânica do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central.

Aprovado (a) em: _____/_____/_____.

BANCA EXAMINADORA

Prof. Me. Francisco Marcelino Almeida de Araújo (Orientador)
Instituto Federal de Educação, Ciência e Tecnologia do Piauí (IFPI)

Prof. Me. Janiel Fontineles Silva
Instituto Federal de Educação, Ciência e Tecnologia do Piauí (IFPI)

Prof. Dr. Nuno Miguel Fonseca Ferreira
Instituto Superior de Engenharia de Coimbra (ISEC)

Prof. Dr. Ranulfo Plutarco Bezerra Neto
Tohoku University (TU)

Eng. Me. Carlos Erlan Olival Lima
Toyohashi University of Technology (TUT)

A Deus.

Aos meus pais.

Aos mestres.

A todos que acreditaram em mim.

AGRADECIMENTOS

A Deus, por tudo.

A minha família, pelo incentivo.

Aos integrantes do LABIRAS por terem me acolhido desde o início dessa jornada. Por todas as oportunidades e ensinamentos que eu nunca alcançaria se não fossem vocês. Em especial o professor Francisco Marcelino e os colegas, Caio Vilanova, Paulo Roberto e Igor Gabriel.

“Ele vai ser útil para muita gente”

(Tadashi Hamada, 2014)

RESUMO

Este trabalho apresenta a adaptação de um robô aspirador comercial em uma plataforma robótica móvel. O intuito é criar uma plataforma que possa ser utilizada em sala de aula para o ensino de robótica do curso de bacharelado em engenharia mecânica do Instituto Federal do Piauí. Para isso o robô Roomba® 614 iRobot foi adquirido devido a execução do projeto Robótica Aplicada para Engenharias. Esse robô em específico foi escolhido, pois a empresa iRobot utiliza o mesmo como base para o robô educacional Create 2, desenvolvida pela mesma, que infelizmente não está a venda em território nacional. Utilizando esta base, foi possível criar um robô que se comunica com o *Robot Operating Sistem* (ROS), via micro-ROS, permitindo assim acesso aos dados dos sensores embarcados e a operação do mesmo. Foi criada uma base impressa em 3D onde podem se fixar diversos sensores e adapta-lo a cenários variados. Por fim, foi criada uma documentação, contendo todas as informações necessárias para operar e modificar o robô.

Palavras-chave: Robótica. Roomba. Engenharia. ROS. micro-ROS.

ABSTRACT

This work presents the adaptation of a commercial vacuum robot in a mobile robotic platform. The aim is to create a platform that can be used in the classroom for teaching robotics in the mechanical engineering course at the Instituto Federal do Piauí. For this, the Roomba 614 iRobot robot was acquired due to the execution of the Applied Robotics for Engineering project. This specific robot was chosen because the company iRobot uses it as a basis for the educational robot Create 2, developed by the same company, which unfortunately is not for sale in the country. Using this base, it was possible to create a robot that communicates with the Robot Operating System (ROS), via micro-ROS, thus allowing access to data from embedded sensors and its operation. A 3D printed base was created, where different sensors can be fixed and adapted to different scenarios. Finally, a documentation was created, containing all the information necessary to operate and modify the robot.

Keywords: Robotic. Roomba. Engineering. ROS. micro-ROS.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

LABIRAS	Laboratório de Inovação e Pesquisa Aplicada
IFPI	Instituto Federal de Educação, Ciência e Tecnologia do Piauí
ROS	*Robot Operating System*
PPC	Proposta Pedagógica Curricular
TCC	Trabalho de Conclusão de Curso
ICC	*Instantaneous Center of Curvature*
RMAD	Robô Móvel de Acionamento Diferencial
PCL	*Point Cloud Library*
Open-CV	*Open Source Computer Vision*
OSRF	*Open Source Robotics Foundation*
DDS	*Data Distribution Service*
API	*Application Programming Interfaces*
NASA	*National Aeronautics and Space Administration*
VIPER	*Volatiles Investigating Polar Exploration Rover*
XRCE	*Extremely Resource Constrained Environments*
RTOS	*Real Time Operating Sistem*
POSIX	*Portable Operating System Interface*
SLAM	*Simultaneous Localization And Mapping*
IDE	*Integrated Development Environment*
SBC	*Single Board Computer*

LISTA DE SÍMBOLOS

® Marca registrada

° Graus

SUMÁRIO

1 INTRODUÇÃO

> “Para a civilização ocidental o conceito de evolução humana está diretamente associado ao grau de desenvolvimento tecnológico adquirido ao longo do tempo [...]. Portanto, a motivação de se criar máquinas que possam substituir o homem na realização de tarefas, é uma característica da própria cultura ocidental” (ROMANO; DUTRA, 2002).

Os avanços tecnológicos impactam diretamente em uma indústria que busca minimizar custos adotando diferentes modelos de produção. Robôs industriais são máquinas reprogramáveis que podem realizar diversos movimentos e se adaptar às necessidades das tarefas que executam, por isso são amplamente utilizados em processos de automação.

O termo robô foi originalmente utilizado em 1921 pelo dramaturgo checo Karen Capek, na sua peça de teatro "Os Robôs de Russum" como referência a um autômato que acaba rebelando-se contra o ser humano. A palavra robô é derivada de "robota" de origem eslava que significa "trabalho forçado". O termo robótica foi criado por Asimov para designar a ciência que se dedica ao estudo destas máquinas (SCIAVICCO; SICILIANO, 1995).

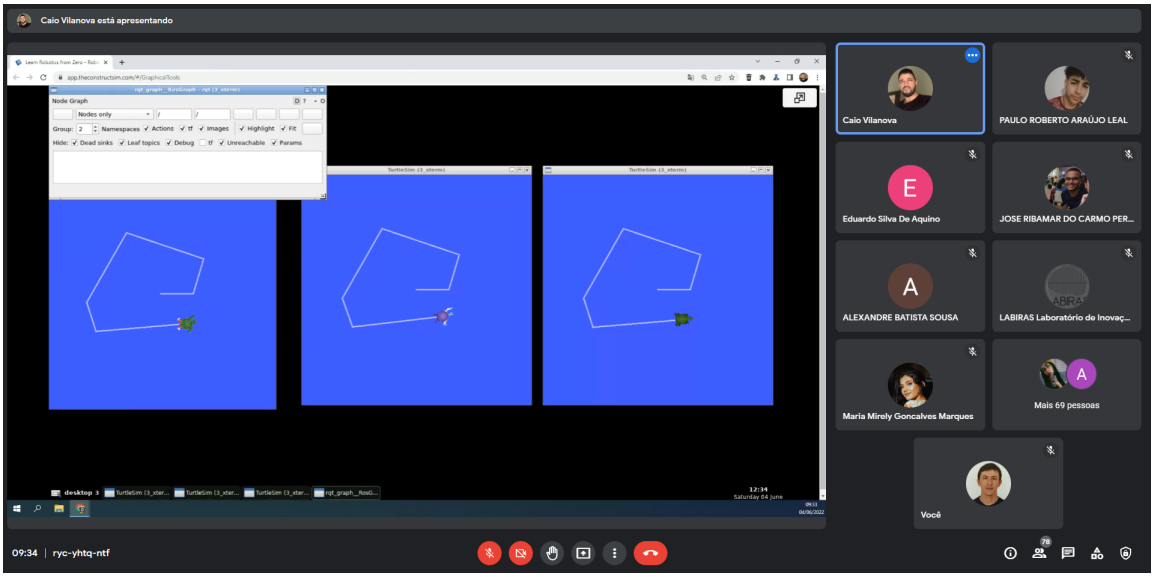
Robôs podem ser definidos ainda como mecanismos atuadores programados com um grau de autonomia para performar locomoção, manipulação ou posicionamento sendo dotados de um sistema de controle. Alguns exemplos de estruturas mecânicas robóticas são manipuladores, plataformas móveis e robôs vestíveis (STANDARTIZATION, 2012).

"Em 2020 foi registrado o recorde de 2.7 milhões de robôs industriais operando em fábricas ao redor do globo.[...] Na América do Sul o Brasil é o número um em estoque operacional com quase 15.300 unidades"(ROBOTICS, 2020). Nesse cenário, o 49º Congresso Brasileiro de Educação em Engenharia foi realizado de 28 a 30 de setembro de 2021, com o tema central “A formação em Engenharia: Tecnologia, Inovação e Sustentabilidade”. O evento discutiu as habilidades esperadas dos graduados de hoje. “[...] o mercado demanda cada vez mais profissionais que desenvolvam as novas tecnologias digitais – como inteligência artificial, *big data*, *machine learning* e robótica avançada.”(BOSCO, 2021).

De 14 de maio a 03 de setembro de 2022 foi ministrado o curso **Robótica Aplicada para Engenharias** (Figura 1) pelo Laboratório de Inovação e Pesquisa Aplicada (LABIRAS), o curso partiu de um projeto proposto e aprovado no edital 3/2021 - PROEX/REI/IFPI, de 29 de abril de 2021. O público alvo do curso foram discentes e docentes das áreas de engenharia do Instituto

Federal de Educação, Ciência e Tecnologia do Piauí (IFPI) campus Teresina Central e Zona Sul. O curso foi online e teve como objetivo ensinar o *Robot Operating System* (ROS) e programação em Python com o foco em entender como funcionam os padrões internacionais do ROS e como ele pode ser aplicado.

Figura 1 – Aula quinzenal do Robótica para Engenharias



Fonte: Elaborado pelo Autor.

Houveram 149 pessoas inscritas para o curso de Robótica Aplicada para Engenharia, sendo 44 desses colaboradores de empresas privadas que atuam diretamente em ambiente industrial. Em virtude disso, além do conteúdo programático do curso ministrado pela equipe de monitores, o projeto acrescentou uma capacitação suplementar em Python, com certificado da *Cisco Academy*® , por meio de parceria com o projeto Piauí Conectado (IFPI, 2022).

Durante o curso foram mostrados exemplos de aplicações industriais do ROS e também os robôs do LABIRAS sendo controlados através do ROS 2, que é a versão mais recente. O curso cobre a parte teórica básica do ROS 2 (conceitos de nós, tópicos, serviços) e fica pendente a parte de aplicação com entidades físicas.

Do curso, 51 alunos concluíram com êxito todo o conteúdo programático e receberam o certificado. Foi proposto para esses alunos que futuramente haveriam outros dois cursos. O primeiro abordando ROS com robôs aéreos e o segundo com robôs móveis terrestres. Diante desse contexto, esse trabalho propõe a utilização de uma metodologia ativa para possibilitar a prática dos conhecimentos adquiridos com o projeto Robótica para Engenharias, utilizando robôs aspiradores comerciais como recurso educacional em laboratório.

A utilização dessa plataforma robótica não fica limitada somente às práticas do curso

proposto, já que na Proposta Pedagógica Curricular (PPC) do curso de engenharia mecânica do IFPI é previsto a disciplina de robótica como uma das 23 disciplinas que podem ser abordadas no grupo de optativas de projeto. Nela é prevista por semana 3 horas de aula teórica e 1 hora de aula prática (LOPES *et al.*, 2018). Desse modo, tanto o material desenvolvido como a plataforma podem ser utilizados pelos alunos durante as aulas práticas da graduação.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver uma plataforma robótica móvel terrestre como ferramenta para ensino de robótica industrial para os alunos do curso de bacharelado em engenharia mecânica do IFPI.

2.2 OBJETIVOS ESPECÍFICOS

- Descrever ROS 2 como *framework* de robótica para indústria;
- Operar o robô real utilizando ROS 2;
- Receber dados dos sensores embarcados do robô;
- Adaptar o hardware para receber novos sensores;
- Padronizar a forma que ele recebe e envia as mensagens;
- Documentar a utilização do robô e o funcionamento do mesmo.

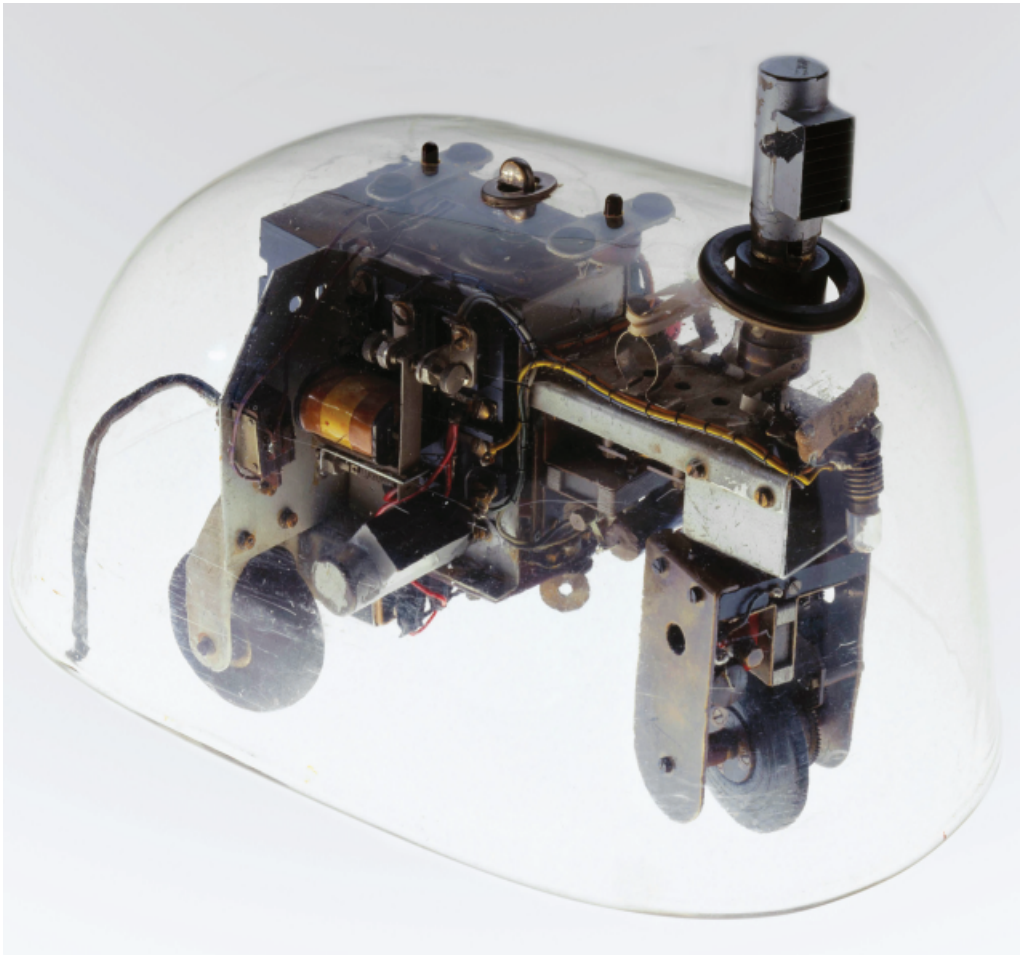
3 FUNDAMENTAÇÃO TEÓRICA

Para melhor compreender as decisões de projeto tomadas durante o desenvolvimento da solução proposta por este trabalho é necessário entender alguns conceitos teóricos relacionados a Robôs móveis e ROS.

3.1 ROBÔS MÓVEIS

De acordo com NILOY *et al.* "Robôs que podem se mover de forma autônoma e podem tomar decisões inteligentes percebendo seus ambientes e objetos ao redor são conhecidos como robôs móveis autônomos". Um dos primeiros exemplos disso é o robô *Tortoise*, ilustrado na Figura 2, criado por W. Grey Walter.

Figura 2 – "Tortoise"robô móvel



Fonte: (MARSH, 2020)

O *Tortoise* foi projetado com dois sensores. O primeiro é uma célula fotoelétrica, que consiste em um dispositivo que produz um sinal elétrico como resposta à exposição à radiação

eletromagnética. Essa célula rotacionava e funcionava como o "olho"da máquina, varrendo os arredores até encontrar uma fonte de luz. O robô parava de escanear e andava em direção a fonte de luz ou, se a fonte fosse muito forte, se distanciava dela. O segundo é um interruptor de contato externo, sensível ao toque. Que faz o robô recuar quando encontra obstáculos. O *Tortoise* retorna à base de carregamento quando sua bateria está baixa.

Em seu trabalho intitulado "*Critical design and control issues of indoor autonomous mobile robots: A review*", NILOY *et al.* afirma que os robôs móveis precisam trabalhar em qualquer ambiente incluindo terrenos acidentados, perigosos, desnivelados, rodeados por obstáculos e áreas restritas. A locomoção é fundamental para o robô se movimentar no espaço de trabalho e não só depende apenas da natureza do ambiente ou do caminho que percorrerá, mas também de fatores vitais como a habilidade de ser controlado, o grau de manobrabilidade, eficiência e estabilidade em terrenos diferentes (RUBIO; VALERO; LLOPIS-ALBERT, 2019). Robôs móveis de interior são projetados para se mover, correr ou andar para poderem completar suas tarefas específicas, então podem ser equipados com pernas, rodas ou até mesmo asas (os que possuem rodas se provam os mais eficientes, como será demonstrado na sessão seguinte) (DUBEY *et al.*, 2015). Neste trabalho o foco será um robô móvel de interior movido por rodas.

3.1.1 Locomoção por rodas

A locomoção por rodas é um dos mecanismos mais populares por ser o método mais simples de deslocamento para robôs móveis já que ele é usado em vários veículos no geral. Por conta da estabilidade, fácil controle e o design mecânico relativamente simples o uso de rodas é bem eficiente e de fácil implementação. Existem 4 tipos diferentes de locomoção por rodas, são eles:

a) Roda padrão: dois graus de liberdade e a rotação ao redor do eixo e ponto de contato.

b) Roda rodízio: dois graus de liberdade e o giro ocorre ao redor da junta de direção deslocada.

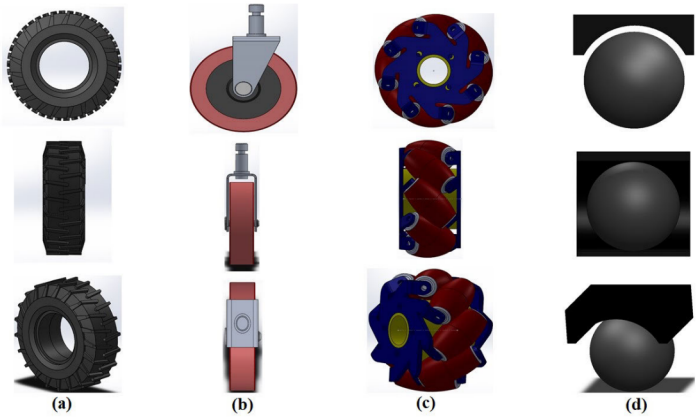
c) Roda omnidirecional: três graus de liberdade e a rotação ocorre ao redor dos rolos, eixo da roda e do ponto de contato.

d) Roda esférica: tem mais área superficial que a roda de rodízio que é cilíndrica.

A Figura 3 mostra as quatro categorias de rodas utilizadas em robôs móveis. Dependendo do ambiente o arranjo de rodas pode utilizar mais de um tipo ou várias de um mesmo tipo. Tanto

o tipo de roda, quanto a geometria são características fundamentais no funcionamento do robô determinando a manobrabilidade, controlabilidade e estabilidade.

Figura 3 – Os quatro tipos de rodas



Fonte: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

Geralmente para ter uma boa estabilidade são usadas três rodas no robô com o centro de gravidade dentro da área formada pelo contado das rodas com o chão, como exemplifica o próprio *Tortoise*.

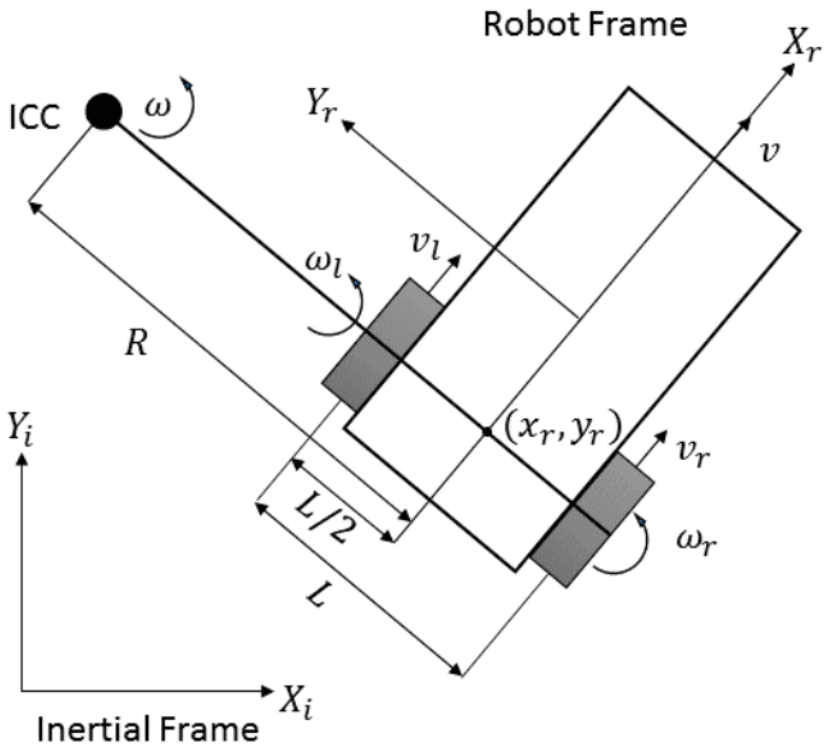
Manobrabilidade é a habilidade de manobrar em qualquer direção gradualmente (DUBEY *et al.*, 2015) e é uma das principais preocupações no projeto do robô. Uma fórmula bem conhecida é utilizar duas rodas padrões e uma roda esférica ou de rodízio. Essa configuração permite bastante manobrabilidade e uma estabilidade aceitável para robôs mais simples.

3.1.2 Cinemática de um robô móvel de acionamento diferencial

Robô Móvel de Acionamento Diferencial (RMAD) é um robô terrestre móvel com rodas que possui três graus de liberdade e dois atuadores, por isso é descrito como um robô não holonômico (ISMAIEL; FILIMONOV, 2021). Robôs holonômicos possuem o mesmo número de graus de liberdade controláveis que os graus de liberdade do próprio robô. RMAD consiste de duas rodas padrões que podem ser acionadas individualmente na esquerda e na direita do chassi do robô e uma roda rodízio na frente para estabilizar a estrutura do robô. A movimentação vai se basear no sentido e velocidade de rotação das rodas padrões combinadas. O RMAD é comumente utilizado em pesquisas robóticas para testar diferentes algoritmos de navegação e desvio de obstáculos, pois sua implementação mecânica e mobilidade são simples (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Cinemática é o estudo do movimento do robô móvel e o comportamento mecânico

Figura 4 – Posição e orientação do RMAD



Fonte: (ISMAIEL; FILIMONOV, 2021)

dele (MALU; MAJUMDAR, 2014). É importante para entender como o robô se movimenta e configurar os parâmetros para controlá-lo. Os principais parâmetros de controle do RMAD são:

- velocidade da roda direita;
- velocidade da roda esquerda;
- posição das rodas no espaço.

Sendo o espaço é uma superfície bidimensional (HELLSTRÖM, 2011). A posição e orientação são descritas por dois referenciais como mostrado na Figura 4: *Inertial Frame* (Referencial de Inércia) X_i, Y_i e o *Robot Frame* (Referencial do Robô) X_r, Y_r. O primeiro referencial é fixo e a posição do RMAD é dada em coordenadas (x_i, y_i) e a orientação em θ_i. O segundo referencial é móvel e é ligado ao robô, as coordenadas de posição são (x_r, y_r) e a orientação é θ_r. O espaço entre as rodas direita e esquerda possui tamanho L. A frente do RMAD é em direção do eixo X_r. A matriz de transformação para converter de um referencial para outro é dada por:

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} \tag{3.1}$$

Os outros parâmetros são usados para derivar as equações do modelo cinemático. As velocidades angulares ω_r e ω_l são respectivamente velocidade angular da roda direita e velocidade angular da roda esquerda. E v_r e v_l são respectivamente a velocidade linear da roda direita e velocidade linear da roda esquerda. Assumindo que as rodas possuem velocidade angular diferentes $\omega_r \neq \omega_l$, cada roda está em contato com o solo em um único ponto e estão rolando sem nenhum escorregamento em relação ao chão. Nesse caso, as duas rodas seguirão uma trajetória circular com um ponto central comum chamado *Instantaneous Center of Curvature* (ICC).

O RMAD se move com velocidade linear v e velocidade angular ω ao longo da trajetória circular de raio R em torno do ICC. R é a distância entre o ICC e o centro do eixo entre as rodas esquerda e direita do RMAD. Conseguimos a velocidade linear quando ele completa uma volta ao redor do ICC durante o tempo T:

$$v = \frac{2\pi R}{T} \tag{3.2}$$

$$\omega = \frac{2\pi}{T} \tag{3.3}$$

A partir das equações 3.2 e 3.3, pode-se extrair a velocidade linear como uma função da velocidade angular.

$$v = \omega R \tag{3.4}$$

As velocidades de cada roda podem ser calculadas associando o raio circular da trajetória correspondente de cada roda na equação 3.4.

$$v_r = \omega(R + \frac{L}{2}) \tag{3.5}$$

$$v_l = \omega(R - \frac{L}{2}) \tag{3.6}$$

A relação da velocidade de cada roda pode ser escrita também em função apenas da velocidade linear e angular na equação 3.7 e 3.8.

$$v_r = v + \frac{L}{2}\omega \tag{3.7}$$

$$v_l = v - \frac{L}{2}\omega \tag{3.8}$$

A velocidade linear de cada roda pode ser representada por sua velocidade angular e o raio da roda da seguinte forma $v_r = \omega_r r$ e $v_l = \omega_l r$. A partir das equações 3.5 e 3.6 calculamos o raio da trajetória circular que o RMAD irá seguir na equação 3.9. A velocidade linear e angular do RMAD são representadas pelas velocidades angulares das duas rodas como demonstrado na equação 3.11, enquanto a equação 3.10 é derivada das equações 3.4, 3.5 e 3.6.

$$R = \frac{rL}{2}\frac{(\omega_r + \omega_l)}{(\omega_r - \omega_l)} \tag{3.9}$$

$$v = \frac{r}{2}(\omega_r + \omega_l) \tag{3.10}$$

$$\omega = \frac{r}{L}(\omega_r - \omega_l) \tag{3.11}$$

Por fim as componentes de velocidade do referencial do robô são mostrados na equação 3.12. Substituindo na equação 3.1 temos a equação do modelo cinemático do RMAD que descreve a velocidade dele em função da velocidade angular das rodas na equação 3.13.

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2} & \frac{-r}{2} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta & \frac{-r}{2}\sin\theta \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{3.13}$$

3.2 ROS

Robot Operating System (ROS) ou Sistema Operacional Robótico é um kit de desenvolvimento de software de código aberto para aplicações de robótica. Ele oferece uma plataforma de software padrão para desenvolvedores de todos os setores que os levarão desde a pesquisa e a prototipagem até a implantação e a produção (ROBOTICS, 2022). ROS é um sistema de código aberto e meta-operacional, que provê os serviços que você normalmente espera de um sistema operacional, tais como (MACENSKI *et al.*, 2022) (QUIGLEY *et al.*, 2009) (KOUBÂA *et al.*, 2017):

- abstração de hardware;
- controle de dispositivo de baixo nível;
- implementação de funcionalidade comumente usada;
- troca de mensagens entre processos;
- gerenciamento de pacotes.

O ROS é o sistema operacional destinado a sistemas robóticos, permite comunicar softwares que utilizam diferentes linguagens de programação, bem como ligar os diferentes dispositivos de entrada ou saída de forma trivial e, além desta particularidade, reúne os esforços de vários desenvolvedores nesta área (QUIGLEY; GERKEY; SMART, 2015). São mais de 7000 pacotes desenvolvidos para as várias distribuições do ROS (LALANCETTE; POUBEL, 2022). Por mais de 10 anos, o projeto ROS produziu um vasto ecossistema de software para robótica, alimentando uma comunidade global de milhões de desenvolvedores e usuários que contribuem e melhoram esse software (FOOTE, 2022).

Mesmo que houvesse pesquisas de robótica acontecendo antes do ROS, a maior parte do software era exclusiva para seus próprios robôs. Seu software pode ser de código aberto, mas é muito difícil de reutilizar. Comparando o ROS com sistemas robóticos existentes, ele desempenha bem melhor nos seguintes aspectos (JOSEPH, 2017):

- **Desenvolvimento colaborativo**: Como discutimos, o ROS é de código aberto e gratuito para uso em indústrias e pesquisas. Os desenvolvedores podem expandir as funcionalidades do ROS adicionando pacotes. Quase todos os pacotes de ROS funcionam em uma camada de abstração de hardware, portanto, podem ser reutilizados facilmente para outros robôs.

- **Suporte a linguagem**: O *framework* de comunicação ROS pode ser facilmente implementado em qualquer linguagem moderna. Ele já suporta linguagens populares como C++ e Python, e possui bibliotecas que são mantidas por membros da comunidade que dão suporte a linguagens como Ada, C, JVM, Android, Node.js, Rust, entre outras.

- **Integração de bibliotecas**: O ROS possui uma interface para muitas bibliotecas robóticas de terceiros, como *Open Source Computer Vision* (Open-CV), *Point Cloud Library* (PCL), Open-NI, Open-Rave e Orocos. Os desenvolvedores podem trabalhar com qualquer uma dessas bibliotecas sem muito trabalho.

- **Integração de simuladores**: O ROS também está vinculado a simuladores de código aberto, como Gazebo e Webots, e possui uma boa interface com simuladores proprietários, como CoppeliaSim.

- **Teste de código**: O ROS oferece uma estrutura de teste embutida chamada rostest para verificar a qualidade do código e *bugs*.

- **Escalabilidade**: A estrutura ROS foi projetada para ser escalonável. Pode-se realizar tarefas de grande custo computacional com robôs usando ROS, que podem ser colocados na nuvem ou em *clusters* heterogêneos.

- **Customização**: Como já discutimos, o ROS é totalmente de código aberto e gratuito, portanto, pode-se personalizar essa estrutura de acordo com os requisitos do robô. Se quisermos apenas trabalhar com a plataforma de mensagens ROS, podemos remover todos os outros componentes e usar apenas isso. Pode-se até personalizar o ROS para um robô específico para melhor desempenho.

- **Comunidade**: O ROS é um projeto voltado para a comunidade, liderado principalmente pela *Open Source Robotics Foundation* (OSRF). O grande apoio da comunidade é uma grande vantagem para o ROS, e pode-se facilmente iniciar o desenvolvimento de aplicativos de robótica.

Sob esse panorama, em razão de todas as vantagens presentes explicitadas, o ROS é um dos principais *frameworks* e essencialmente é aplicado no desenvolvimento de sistemas robóticos e de automação ao redor do mundo.

3.2.1 Conceitos básicos

O *node* é uma unidade de código executável, ele troca informação com outros *nodes* através de *messages* formando um grande programa. O conceito chave é o método de comunicação por *message* entre os *nodes*. A conveniência é que cada *node* executa um simples propósito, para facilitar a reusabilidade e o desenvolvimento (PYO *et al.*, 2017).

Um *package* é unidade básica do ROS. As aplicações do ROS são desenvolvidas baseadas em *packages*, ele contém os arquivos de configuração para executar outros *packages* ou *nodes*. O *package* também contém todos os arquivos necessários para execução, incluindo bibliotecas de dependência do ROS para executar vários processos (PYO *et al.*, 2017). Por exemplo, o ROS kinect que foi a distribuição mais popular por bastante tempo, conta com uma biblioteca com 2772 *packages* (LALANCETTE; POUBEL, 2022).

A *message* é um conjunto de dados que são enviados e recebidos pelos *nodes*. *Messages* possuem variáveis como inteiros, floats e booleanos. Uma *message* pode ser estruturada contendo várias outras *messages* dentro dela.

Os *topics* são um elemento vital do gráfico ROS que atuam como um barramento para os *nodes* trocarem *messages*. Um *node* pode publicar dados para qualquer número de *topics* e, simultaneamente, ter assinaturas para qualquer número de *topics*.

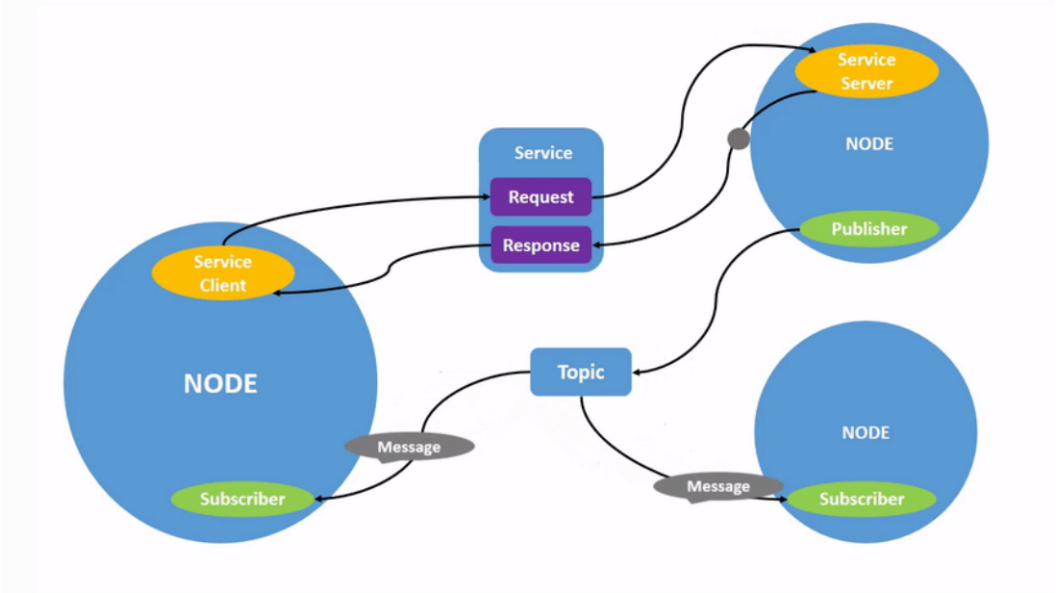
O *service* é a comunicação bidirecional síncrona entre o cliente de serviço que solicita um serviço referente a uma tarefa específica e o servidor de serviço que é responsável por responder às solicitações.

As *actions* são um dos tipos de comunicação, exclusivas do ROS 2, são destinadas a tarefas de longa duração. Eles consistem em três partes: uma meta, retorno e um resultado. As *actions* são construídas em *topics* e *services*. Sua funcionalidade é semelhante à dos *services*, exceto que as *actions* são preemptivas (você pode cancelá-las durante a execução). Eles também fornecem retorno constante, ao contrário de *services* que retornam uma única resposta.

O *namespace* é um mecanismo para encapsular grupos de execução no ROS. É bastante útil quando é necessário executar mais de uma vez o mesmo *node* e separar as informações entre eles, evitando conflitos de nomes.

Atualmente o ROS está na sua segunda versão o ROS 2. Normalmente quando o termo ROS é citado está sendo referenciado a comunidade e o projeto como um conceito geral. Para diferenciar as versões será utilizado o termo ROS 1 e ROS 2 quando for necessário apontar características específicas de cada um. A principal diferença entre as duas versões é a arquitetura

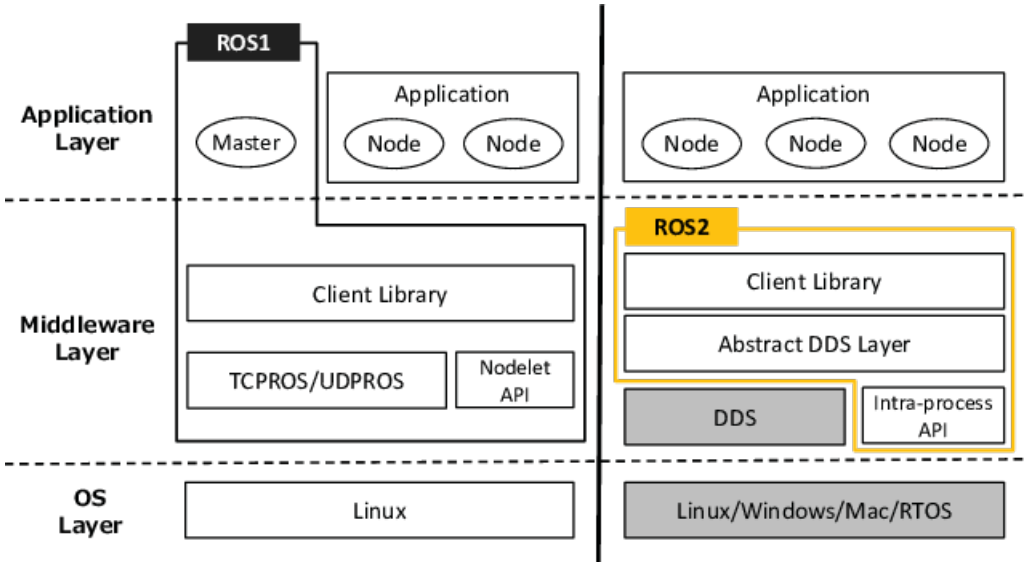
Figura 5 – Comunicação entre *nodes*



Fonte: (WEON, 2022)

de como é feita a comunicação entre os *nodes*. No ROS 1 era necessário ter um *node* chamado *Master* que gerenciava as comunicações e utilizava o protocolo TCP/UDP para comunicação. No ROS 2 isso é feito na camada de *middleware* e o protocolo é o *Data Distribution Service* (DDS) (MARUYAMA; KATO; AZUMI, 2016), como mostrado na Figura 6.

Figura 6 – Arquitetura ROS 1/ROS 2



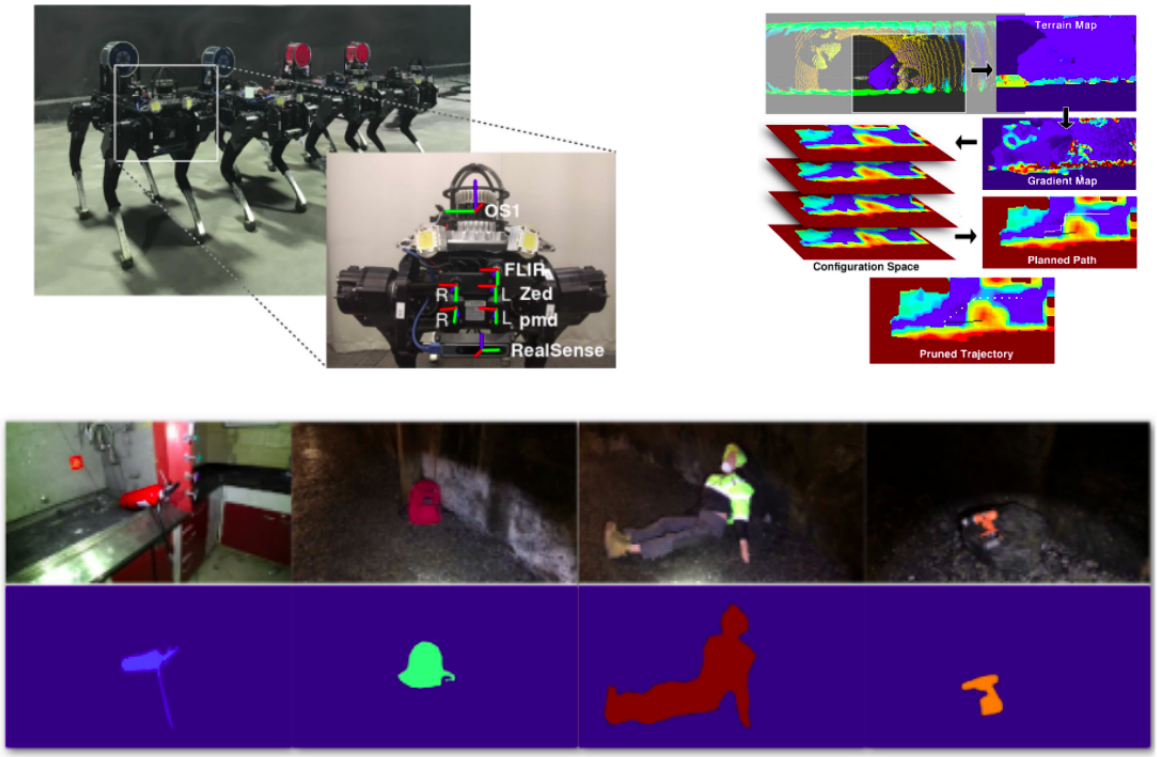
Fonte: (MARUYAMA; KATO; AZUMI, 2016)

3.2.2 Casos de uso

Recentemente foi divulgada uma lista pública com quase 400 empresas conhecidas que utilizam o ROS 1 ou ROS 2, ou qualquer de suas ferramentas de desenvolvimento, para criar produtos ou oferecer serviços. Nessa lista constam desde empresas centenárias como a Bosch, Boeing, BMW Group, Jhon Deere, KUKA, até empresas recentes Toyota Woven Planet, TechnoYantra, Tangram Vision, Stogl Robotics (VILCHES, 2022). Inclusive algumas delas trabalham especificamente prestando consultoria do uso de ROS.

Nesse contexto, em ambientes terrestres, Miller *et al.* (2020) apresenta em sua pesquisa a utilização de múltiplos robôs quadrúpedes da empresa Ghost Robotics para a exploração em túneis de minas subterrâneas (Figura 7). O ROS foi incorporado ao sistema, dando suporte ao controle da missão, permitindo o mapeamento e retorno. Durante a missão, o sistema identifica e cruza informações com o banco de dados, em seguida é possível montar e executar cada ação necessária, isso tudo através de *Application Programming Interfaces* (API) que se utilizam da interface *publisher-subscriber*.

Figura 7 – Robôs quadrúpedes para exploração e mapeamento em minas subterrâneas

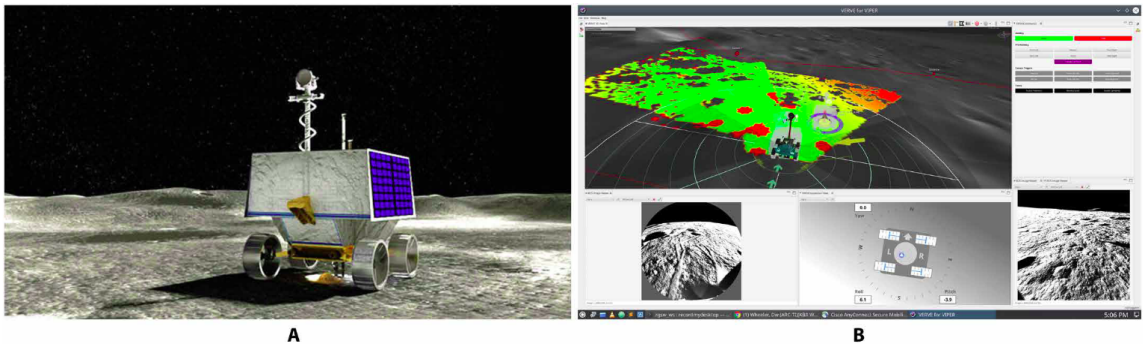


Fonte: (MILLER *et al.*, 2020)

Como outro exemplo de destaque, pode-se citar a *National Aeronautics and Space*

Administration (NASA), ela promoverá uma missão chamada *Volatiles Investigating Polar Exploration Rover* (VIPER), programada para ser lançada em novembro de 2023, com o intuito de explorar a região polar da lua e buscar gelo e outros recursos em uma missão de 100 dias de duração. Várias das ferramentas de operações baseadas na Terra, módulos de cálculo e simulações de alta fidelidade são baseadas no ROS 2 e Gazebo. Como mostrado na Figura 8, em "A"temos o robô *VIPER* na superfície da lua (renderizado), e em "B"o *software* de operação e comandos (MACENSKI *et al.*, 2022).

Figura 8 – Robô VIPER da NASA



Fonte: (MACENSKI *et al.*, 2022)

3.2.3 Micro-ROS

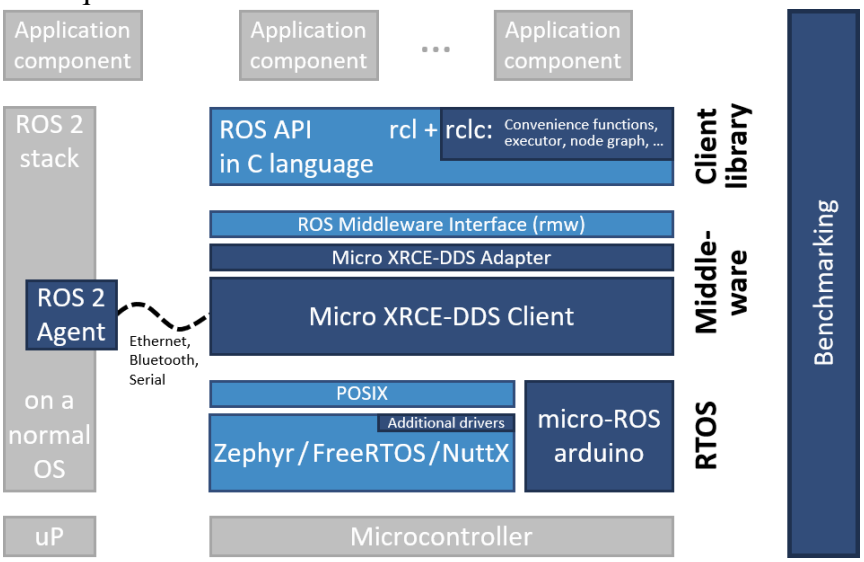
De acordo com ROS (2022) a missão do time que desenvolve o micro-ROS é preencher o espaço entre microcontroladores com recursos restritos e processadores mais potentes em aplicações robóticas que são baseadas no ROS. Com o micro-ROS é possível fazer com que microcontroladores, mais baratos e simples que computadores e notebooks, utilizem o ROS. Esses microcontroladores são usados em quase todos os robôs, os principais motivos são o acesso ao hardware, baixa latência e economia de energia.

O micro-ROS segue a arquitetura do ROS 2 e toma proveito da capacidade de conexão do *middleware* para utilizar *Data Distribuiton Service - Extremely Resource Constrained Environments* (DDS-XRCE), que é otimizado para microcontroladores. Entretanto ele utiliza o *Real Time Operating Sistem* (RTOS) baseado em *Portable Operating System Interface* (POSIX) ao invés de Linux.

A Figura 9 demonstra como funciona a arquitetura do micro-ROS. Em azul escuro estão os componentes desenvolvidos especificamente para o micro-ROS. Azul claro são os componentes padrões do ROS 2 que foram utilizados. O **ROS 2** ***Agent*** é o nó responsável por

fazer a ponte de comunicação entre o microcontrolador e o computador com mais poder de processamento.

Figura 9 – Arquitetura do micro-ROS



Fonte: (ROS, 2022)

4 TRABALHOS RELACIONADOS

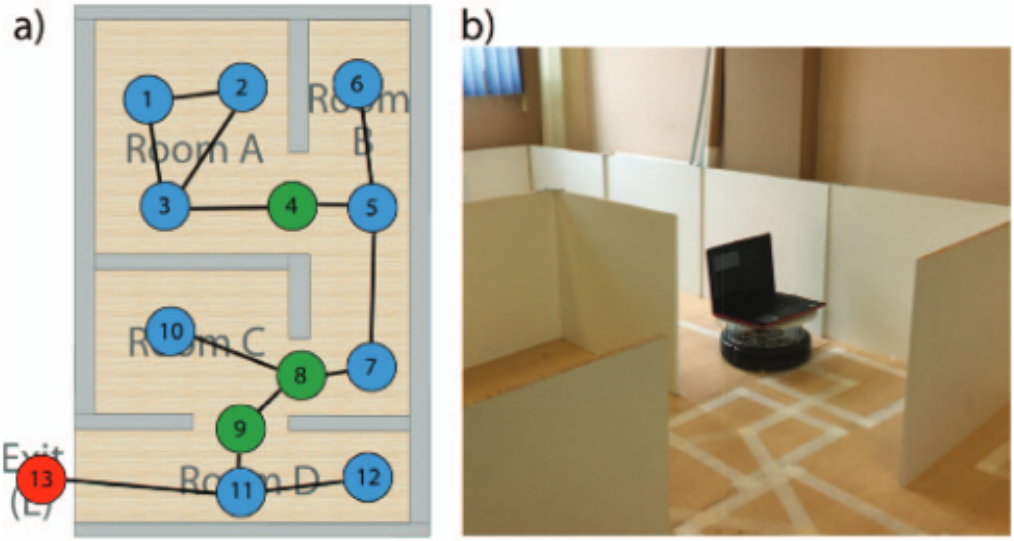
Neste capítulo se encontram trabalhos que utilizam o Roomba como plataforma robótica para teste de algoritmos, ferramenta educacional e como plataforma robótica em geral.

Este TCC teve como inspiração principal o robô Create 3 iRobot (SHAMLIAN, 2021). Infelizmente o mesmo não está disponível para compra no Brasil. Com isso em vista, pode-se utilizar o mesmo robô Roomba como base para desenvolver uma versão própria. Como visto a seguir, não é a primeira vez que isso foi feito. A utilização do Roomba facilita o desenvolvimento da robótica, permitindo pular algumas etapas da criação de um robô.

No trabalho de Rojas-Fernandez *et al.* (2018), é feita a comparação de performance de algoritmos de *Simultaneous Localization And Mapping (SLAM)*. Para tal, é utilizado o Roomba 654 como plataforma robótica de direção diferencial. Foram testados três algoritmos, sendo todos pacotes do ROS, o Gmapping, HectorSLAM e CRSM SLAM.

Romero-Marti *et al.* (2017) em seu trabalho aplica navegação e planejamento de trajetória usando aprendizado por reforço em um robô Roomba. No trabalho, ele cria um ambiente controlado e usa o Roomba como base para navegar no ambiente (Figura 10). Em "a)"temos o mapa de navegação usado pelo robô e em "b)"o roomba navegando pelo ambiente controlado.

Figura 10 – Navegação autônoma com roomba

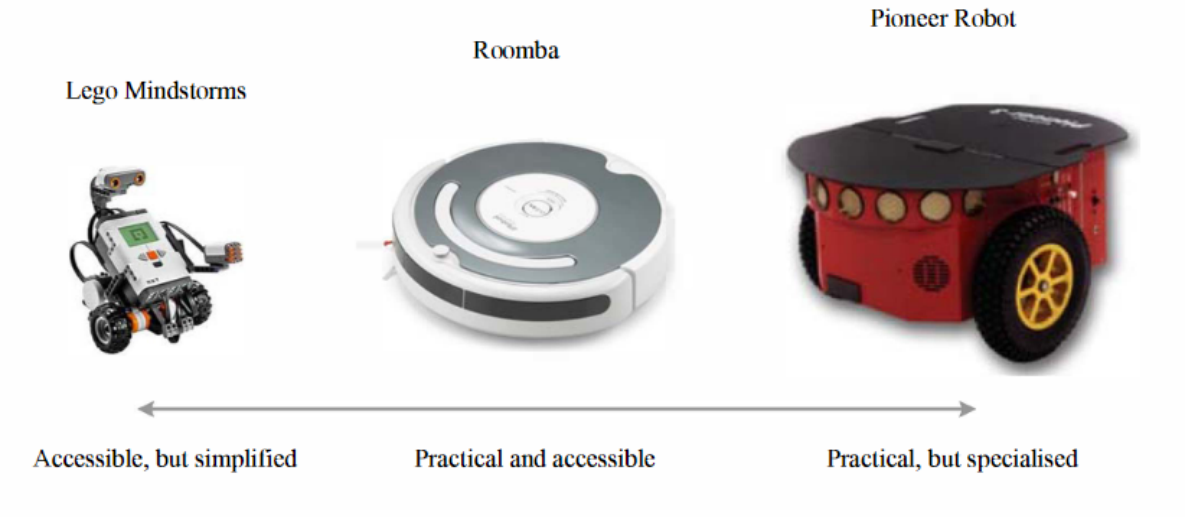


Fonte: (ROMERO-MARTI *et al.*, 2017)

Na área educacional, Nattharith e Guzel (2014) em seu trabalho desenvolveu uma plataforma de baixo custo para pesquisa de robótica utilizando o Roomba. É demonstrado como

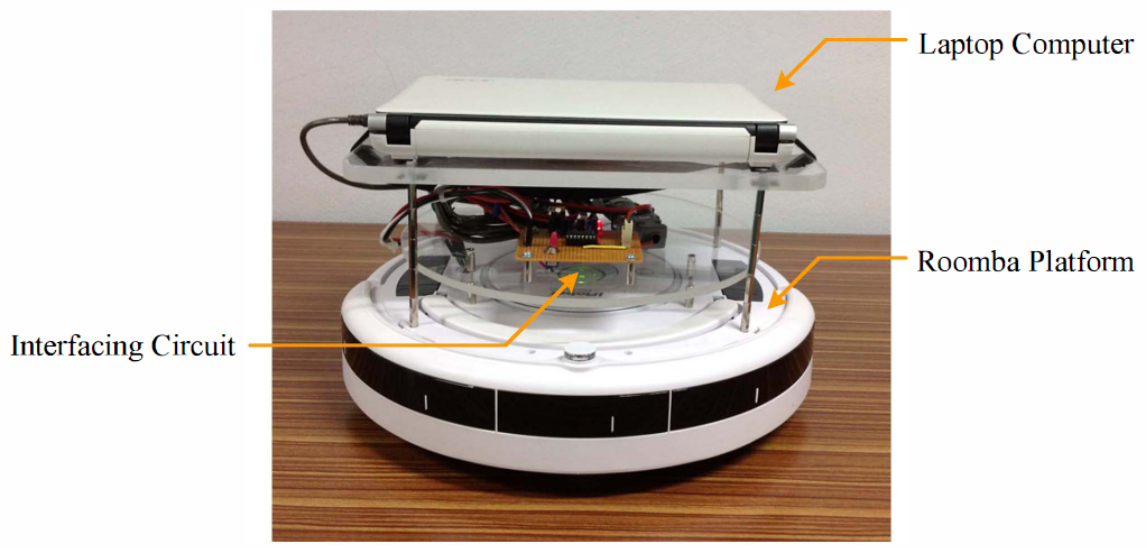
o Roomba se qualifica entre outras plataformas robóticas disponíveis no mercado (Figura 11). Para avaliar a plataforma, foi desenvolvido e implementado um sistema para navegar em um ambiente controlado. Ao Roomba foram adicionados sensores infravermelho e uma base para suportar um notebook (Figura 12).

Figura 11 – Espectro de robôs de pesquisa educacionais



Fonte: (NATTHARITH; GUZEL, 2014)

Figura 12 – Plataforma modificada do Roomba



Fonte: (NATTHARITH; GUZEL, 2014)

Exitem vários outros exemplos de utilização do Roomba como plataforma robótica para fins variados. Alguns desses são:

- *Towards the development of Junkyard Hacks: Networked Robotics Applications* (WILLI-

AMS *et al.*, 2018): que utiliza robôs aspiradores, que seriam descartados como lixo por não conseguirem fazer sua função de aspirador, como plataformas para robótica.

- *Experimental Study of Decentralized Robot Network Coordination* (LEMON; ANGLEA; WANG, 2019): faz o uso de 4 Roombas para estudar técnicas de navegação e sincronização para redes de robôs.
- *Robot Navigation Using Ultra-Wideband Indoor Localization and Dead Reckoning Algorithms* (BLAKE *et al.*, 2022): utiliza o Roomba para desenvolver maneiras para navegar em interiores com baixo custo.

5 METODOLOGIA

Para a realização deste trabalho será seguida a sequência de ações listada abaixo, podendo seguir diferentes consultas a assuntos similares, caso haja a necessidade.

5.1 PESQUISA BIBLIOGRÁFICA

Foi realizada uma pesquisa bibliográfica com o intuito de observar aplicações similares ou possíveis de serem feitas com o Roomba e ROS. Os resultados dessa pesquisa se condensam no capítulo 4, o qual é denominado "**TRABALHOS RELACIONADOS**".

5.2 OPERAÇÃO DO ROBÔ

Para realização do trabalho, foram utilizados equipamentos de informática disponíveis no ambiente de laboratório do LABIRAS, acrescido de robôs aspiradores Roomba® 614 iRobot (Figura 13), adquiridos para a execução do projeto Robótica Aplicada para Engenharias, aprovado por meio do EDITAL 3/2021 - PROEX/REI/IFPI, de 29 de abril de 2021. Para a operação do robô, foi utilizado o manual *Create 2 Open Interface* que possui especificações de todas interfaces de entrada e saída de robôs da série Roomba® 600.

Figura 13 – Roomba® 614 iRobot

Fonte: (IROBOT, 2020)

5.2.1 Software

Para interfacear a comunicação com o robô, foi utilizado o ESP32 Dev Module. A programação foi feita em linguagem C++. O código foi desenvolvido utilizando o *Integrated Development Environment* (IDE) *Visual Studio Code*, que possui várias extensões que auxiliam no desenvolvimento de software. Foi utilizado a extensão platformIO que auxilia a fazer o envio do código para dentro do microcontrolador, além de auxiliar no gerenciamento de bibliotecas.

Para as funções de controle do Roomba foi utilizada a biblioteca *iRobot-Create2-Arduino*, uma biblioteca de arduino para o Create 2 (AMATO, 2016). Porém foi necessária uma adaptação da mesma para o uso com a placa ESP32 devido a diferença da arquitetura da placa com o arduino. Para adaptação foi necessário trocar algumas bibliotecas que não eram compatíveis.

Além disso foi refeita a função de coleta de dados dos sensores. Antes a função conseguia coletar o dado de um sensor por chamada, agora é possível passar uma lista de sensores e a função retorna uma lista com o valor de cada sensor na mesma ordem da primeira lista. Ainda assim, não é possível coletar o dado de todos os sensores ao mesmo tempo, devida a limitação do Roomba via conexão serial. Na sua taxa de transmissão mais rápida (115200 *baud*) um máximo de 172 bytes podem ser enviados em 15 ms (esse é o tempo que demora a coleta dos valores dos sensores). Se mais dados forem requisitados o fluxo será corrompido.

Tendo isso em vista, para este trabalho foram selecionados os sensores que estão listados na Tabela 1. Nela está a lista na ordem de requisição dos sensores, o nome do pacote do sensor (igual está em iRobot (2015)), o número que deve ser enviado na serial para requisição do mesmo e por fim o número de bytes que os pacotes devolvem. Para outros projetos, podem ser selecionados sensores diferentes que sejam relevantes, desde que não ultrapasse o limite de bytes mencionado anteriormente.

Foram utilizadas as mensagens do pacote *create_msgs* (PERRON, 2022), para enviar os dados coletados para o ROS. Ao pacote foram adicionadas mensagens *Cliff*, que contém todas as informações do índice 2 ao 5, *Battery*, que contém do índice 7 ao 11, e *Wheels*, que contém do índice 12 ao 15.

Para realizar a coleta desses valores o ESP32 comunica via serial com o Roomba. A comunicação com o ROS 2 é feita via *WiFi* em rede local. Para configurar a comunicação, assim que o ESP32 é energizado ele cria uma rede *WiFi* na qual quando conectada abre uma página HTML onde deve ser definida a rede *WiFi* desejada, a senha da rede, o endereço IP da máquina que está executando o *Agent* e o *namespace* que o robô utilizará (esse último campo

não é obrigatório). Após isso, o ESP32 faz a conexão com o *Agent* e começa a enviar e receber as mensagens nos tópicos especificados.

Tabela 1 – Lista com sensores que tiveram os dados coletados

Índice	**Nome do pacote**	**Número do pacote**	**Nº de bytes**
0	*Bumps and Wheel Drops*	7	1
1	*Light Bumper*	45	1
2	*Cliff Left*	9	1
3	*Cliff Front Left*	10	1
4	*Cliff Front Right*	11	1
5	*Cliff Right*	12	1
6	*Charging State*	21	1
7	*Voltage*	22	2
8	*Current*	23	2
9	*Temperature*	24	1
10	*Battery Charge*	25	2
11	*Battery Capacity*	26	2
12	*Requested Right Velocity*	41	2
13	*Requested Left Velocity*	42	2
14	*Left Encoder Counts*	43	2
15	*Right Encoder Counts*	44	2

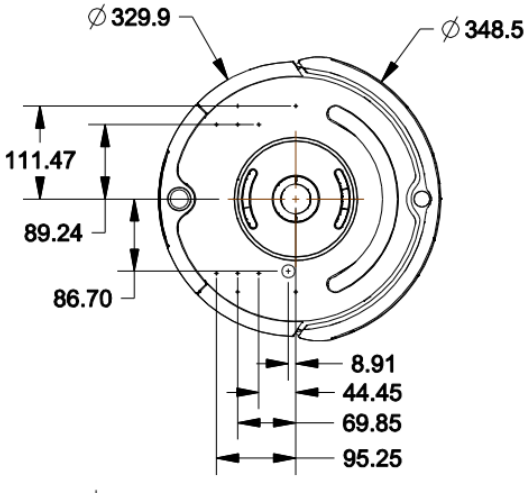
5.2.2 Hardware

As dimensões do robô estão ilustradas nas Figuras 14 e 15 em milímetros. A área do topo foi aproveitada para fixar uma base que possa futuramente receber sensores ou atuadores variados.

O roomba consegue suportar uma carga de 15 quilogramas, somente o robô pesa por volta de 3,5 quilogramas (IROBOT, 2015). Ele possui sensores de detecção de beirada, botões superiores, contadores de roda, nível de bateria, de batida, queda de roda, infravermelho, temperatura, voltagem e corrente elétrica. Como saídas ele possui, dois motores de passo, alto-falantes e LEDs.

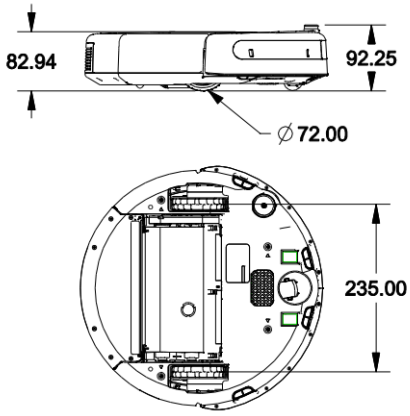
Como mencionado anteriormente, a conexão do ESP32 com o Roomba é feita via serial através do conector mini-din do robô (Figura 16).

Figura 14 – Vista superior



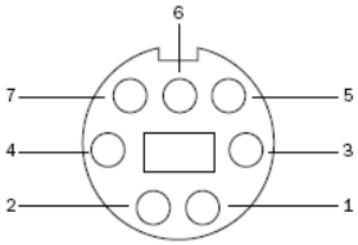
Fonte: iRobot (2015)

Figura 15 – Vista lateral e inferior



Fonte: (IROBOT, 2015)

Figura 16 – Conector mini-din do Roomba



Pin	Name	Description
1	Vpwr	Roomba battery + (unregulated)
2	Vpwr	Roomba battery + (unregulated)
3	RXD	0 – 5V Serial input to Roomba
4	TXD	0 – 5V Serial output from Roomba
5	BRC	Baud Rate Change
6	GND	Roomba battery ground
7	GND	Roomba battery ground

Fonte: (IROBOT, 2015)

5.3 PADRONIZAÇÃO

Uma das grandes vantagens de usar o ROS para desenvolvimento de robôs é a facilidade de reutilizar os *packages* criados por outras pessoas. Para tal, é necessário que haja uma padronização na forma que são enviadas e recebidas as mensagens entre esses *packages*.

Já existem vátios repositórios com pacotes que trabalham com os diferentes modelos de Roomba, que são ou foram comercializados. Foi realizado uma busca nos repositórios de projetos de código aberto no GitHub, que é uma plataforma de hospedagem de código fonte e arquivos com controle de versão usando a ferramenta git, e todos os repositórios que foram encontrados utilizam um computador ou um *Single Board Computer* (SBC) para comunicar e controlar o Roomba via ROS.

Nenhum dos repositórios encontrados faz uso do micro-ROS para realizar essa função. Isso não impossibilita a utilização, basta manter o padrão de troca das mensagem. Dito isso, como referência foi utilizado o padrão desenvolvido pela AutonomousLab para o Create 1, Create 2, e Roombas das séries 400 à 900 (PERRON, 2022).

5.4 DOCUMENTAÇÃO

Por fim, ao concluir o desenvolvimento da plataforma móvel, foi criada uma documentação. Nela contém a ficha técnica do robô, os padrões utilizados, instruções de utilização e boas práticas para desenvolvimentos futuros.

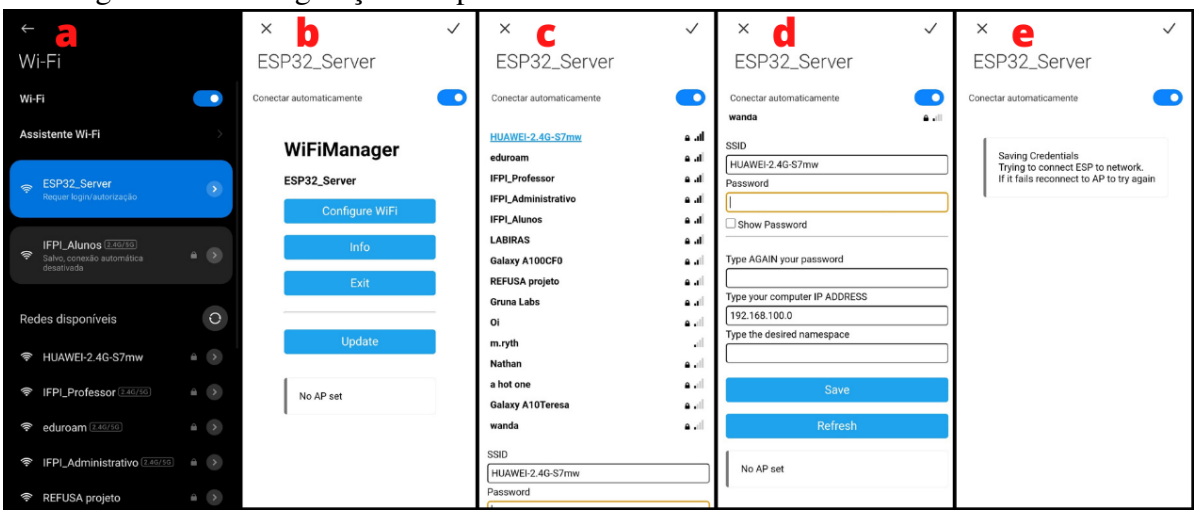
6 RESULTADOS

Neste capítulo será descrito como o robô está funcionando atualmente, como testar os possíveis comandos e visualizar os dados que o robô disponibiliza. Também será apresentado as mudanças físicas que foram feitas para futuros melhoramentos ou adaptações.

6.1 OPERAÇÃO DO ROBÔ

Começando pela inicialização do robô, na Figura 17 temos os passos necessários para inicia-lo. Seguindo a sequência temos a) Conectar na rede ESP32_Server, b) Selecionar opção *Configure WiFi*, c) Selecionar a rede desejada, d) Preencher os campos (somente o último campo não é obrigatório), e) Aguardar a página fechar automaticamente (se ela não fechar é necessário reiniciar o ESP32 e reiniciar o processo).

Figura 17 – Configuração dos parâmetros do robô



Fonte: Elaborado pelo autor.

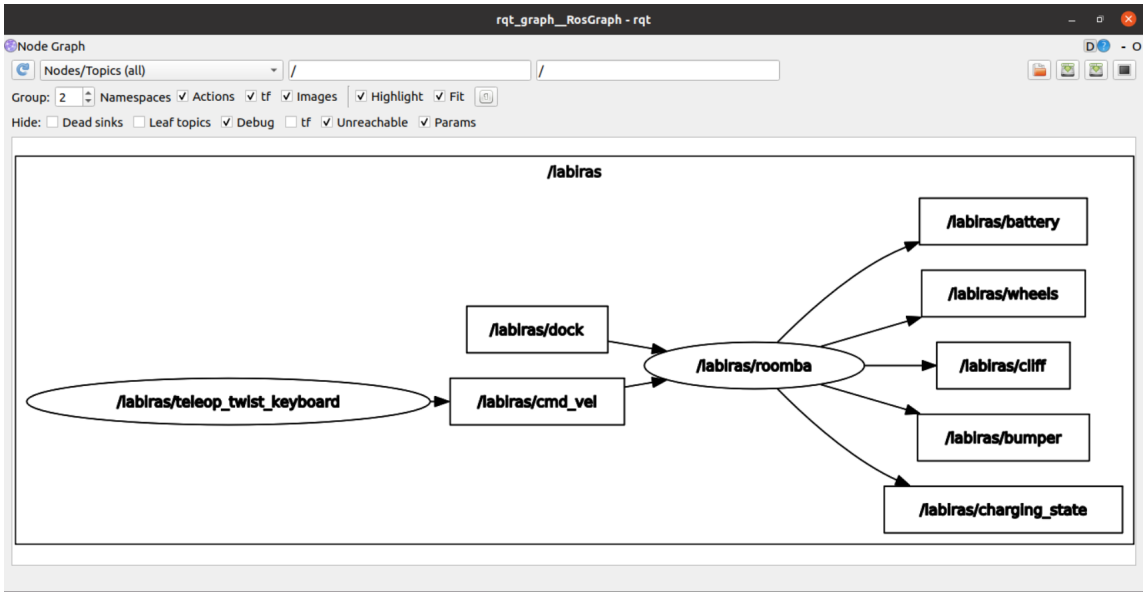
O *micro-ROS Agent* precisa estar executando na máquina, a qual foi inserida o endereço IP, ele pode ser instalado criando um *ROS workspace* e clonando o *package micro_ros_setup* e compilando manualmente, ou utilizando uma imagem *Docker* como mostra Finocchiaro (2020). A segunda opção inclusive é a recomendada e a que foi utilizada no desenvolvimento deste trabalho.

Após a conexão, o *Agent* cria o *node* e os *topics*. Pode ser visualizado usando a ferramenta *rqt_graph* do ROS (Figura 18). São sete *topics* e um *node*, todos possuem o prefixo do *namespace* que foi definido na inicialização do robô. Nesse caso o *namespace* é **/labiras**.

Para publicar em **/labiras/cmd_vel** foi utilizado o *node* teleop_twist_keyboard (Figura

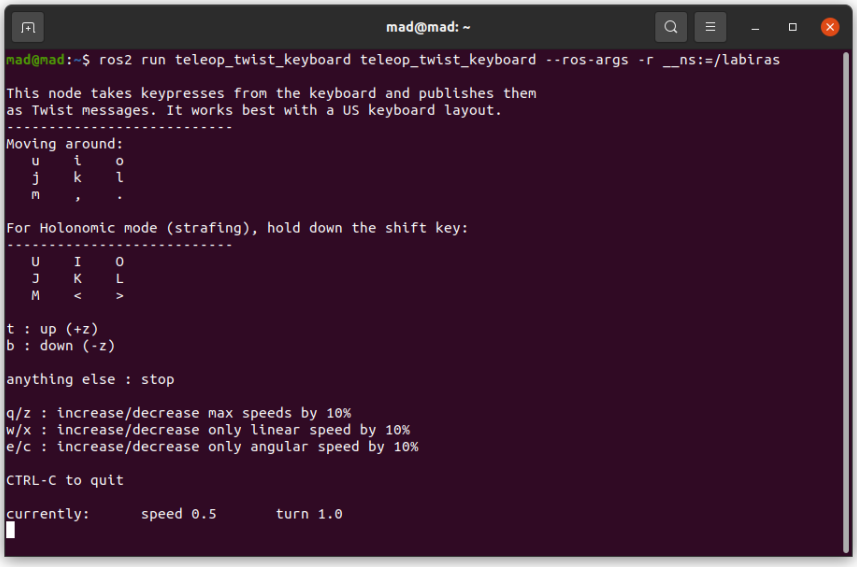
19), que faz parte de um *package* padrão que vem instalado juntamente com a versão *desktop* do ROS 2.

Figura 18 – rqt mostrando o *node* e *topics* gerados pelo micro-ROS e o *node* de teleoperação



Fonte: Elaborado pelo autor.

Figura 19 – *Node* teleop_twist_keyboard executando



Fonte: Elaborado pelo autor.

Para movimentação o ESP32 espera mensagens do tipo **geometry_msgs/msg/Twist** no *topic* **/labiras/cmd_vel**. A mensagem é definida por dois vetores de velocidade, um linear com x, y e z, e um angular também com x, y e z. O robô só consegue se mover no plano, andando

na direção x e rotacionando em z. Sendo assim, só as componentes linear x e angular z serão necessárias. O ESP32 espera esses valores e comanda os motores de acordo. Os valores estão em unidade de metros por segundos. Usando as equações 3.7 e 3.8 podemos comandar a velocidade de cada roda recebendo apenas velocidade linear e angular (a velocidade máxima de cada roda é de 0,5 m/s, qualquer valor que ultrapasse isso o robô irá manter a velocidade máxima e ignorar o excedente).

O *topic* **/labiras/dock** é do tipo **std_msgs/msg/Bool**, por ele passam *messages* do pacote padrão do ROS do tipo boleano. O *node* /labiras/roomba fica aguardando uma *message* com o valor *true* para comandar o Roomba a voltar para a base de carregamento.

Os outros cinco *topics* recebem os valores dos sensores que foram selecionados previamente e disponibilizam para o ROS. A seguir será mostrado como é a estrutura de cada uma das *messages*. Uma característica que todas possuem em comum é iniciar a mensagem com um cabeçalho que é um campo chamado *header*. Ele é um tipo de *message* do pacote de std_msgs, possui basicamente três campos.

- **frame_id:** Quadro de transformação ao qual esses dados estão associados.
- **stamp:** Timestamp de dois inteiros que é expresso como segundos e nanossegundos. É separado em dois campos.
  - **sec:** O componente de segundos, válido sobre todos os valores int32.
  - **nanosec:** O componente de nanossegundos, válido no intervalo [0, 10^9].

Começando por **/labiras/battery**, que informa os dados disponíveis sobre o estado da bateria do robô. Possui cinco campos do tipo float32, eles informam os seguintes dados:

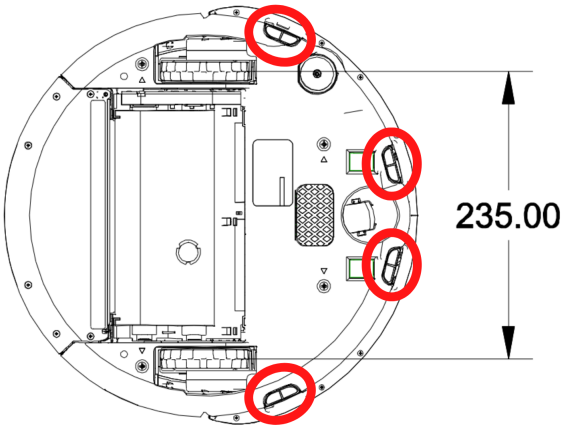
- ***voltage*:** Voltagem da bateria em mV.
- ***current*:** Corrente da bateria em mA.
- ***temperature*:** Temperatura da bateria em °C.
- ***charge:*** Carga atual da bateria em mAh.
- ***capacity:*** Capacidade máxima de carga da bateria em mAh.

O *topic* **/labiras/wheels** é onde são publicados as informações referentes as rodas do Roomba. Possui seis campos para as rodas, três para roda esquerda e três para a direita, mais o *header*. Os campos das rodas são:

- ***drop_left:*** Informa em boleano se a roda esquerda está fora do chão ou não.
- ***encoder_counts_left:*** Número de contagens do codificador da roda esquerda.
- ***velocity_left:*** Velocidade da roda esquerda em mm/s.
- ***drop_right:*** Informa em boleano se a roda direita está fora do chão ou não.
- ***encoder_counts_right:*** Número de contagens do codificador da roda direita.
- ***velocity_right:*** Velocidade da roda direita em mm/s.

O **/labiras/cliff** é onde são publicados as informações referentes aos sensores de beirada. São quatro sensores (Figura 20) que retornam o valor em boleano, se tiver algum desnível o sensor retorna o valor como *true*.

Figura 20 – Posição dos sensores de beirada



Fonte: (IROBOT, 2015)

- ***is_cliff_left:*** Retorna *true* se o sensor da esquerda detectar um desnível.
- ***is_cliff_front_left:*** Retorna *true* se o sensor frontal da esquerda detectar um desnível.
- ***is_cliff_right:*** Retorna *true* se o sensor da direita detectar um desnível.
- ***is_cliff_front_right:*** Retorna *true* se o sensor frontal da direita detectar um desnível.

O **/labiras/bumper** é onde são publicados as informações referentes ao parachoque. Ele consegue detectar se está sendo pressionado na direita, esquerda, ou os dois ao mesmo tempo. Além disso, possui seis sensores infravermelho distribuídos pelo *bumper* que conseguem detectar se há algum obstáculo e retorna o valor da detecção em boleano. Também é possível verificar a

força do sinal que está retornando dos sensores, mas para este trabalho esses dados não foram coletados, seriam os pacotes 46 ao 51. Os campos que foram utilizados, exceto o *header*, são:

- ***is_left_pressed:*** Retorna *true* se o lado esquerdo do *bumper* está pressionado.
- ***is_right_pressed:*** Retorna *true* se o lado direito do *bumper* está pressionado.
- ***is_light_left:*** Retorna *true* se o sensor de luz da esquerda detectar um obstáculo.
- ***is_light_front_left:*** Retorna *true* se o sensor de luz frontal da esquerda detectar um obstáculo.
- ***is_light_center_left:*** Retorna *true* se o sensor de luz central da esquerda detectar um obstáculo.
- ***is_light_center_right:*** Retorna *true* se o sensor de luz central da direita detectar um obstáculo.
- ***is_light_front_right:*** Retorna *true* se o sensor de luz frontal da direita detectar um obstáculo.
- ***is_light_right:*** Retorna *true* se o sensor de luz da direita detectar um obstáculo.

Por fim, o **/labiras/charging_state** é onde são publicados as informações referentes ao atual estado de carregamento da bateria. Possui apenas um campo que retorna um número de zero a cinco que corresponde aos estados mostrados na Figura 21.

Figura 21 – Estados de carregamento da bateria

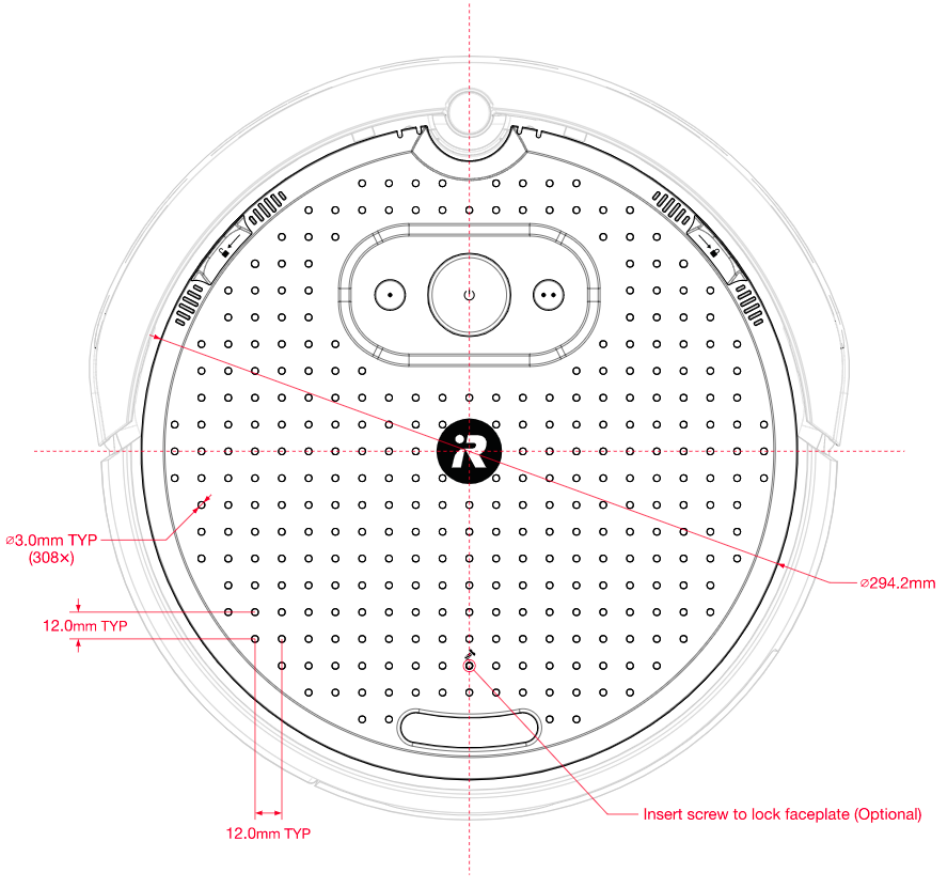
Code	Charging State
0	Not charging
1	Reconditioning Charging
2	Full Charging
3	Trickle Charging
4	Waiting
5	Charging Fault Condition

Fonte: (IROBOT, 2015)

6.2 MODIFICAÇÕES FÍSICAS

Como mencionado anteriormente, o robô consegue suportar um *payload* de 15 kg. E foi aproveitada a área do topo do robô para criar uma base que sirva no futuro para receber diferentes sensores e modificações. Atualmente a base é utilizada para fixar o ESP 32 e a bateria de alimentação do mesmo. Ela foi modelada usando o software Fusion 360 (Figura 23), impressa em 3D e depois fixada no Roomba (Figura 24). A base segue o padrão de furação do modelo Create 3 da iRobot. São furos de 3mm com 12mm de espaçamento entre eles, como mostra a Figura 22.

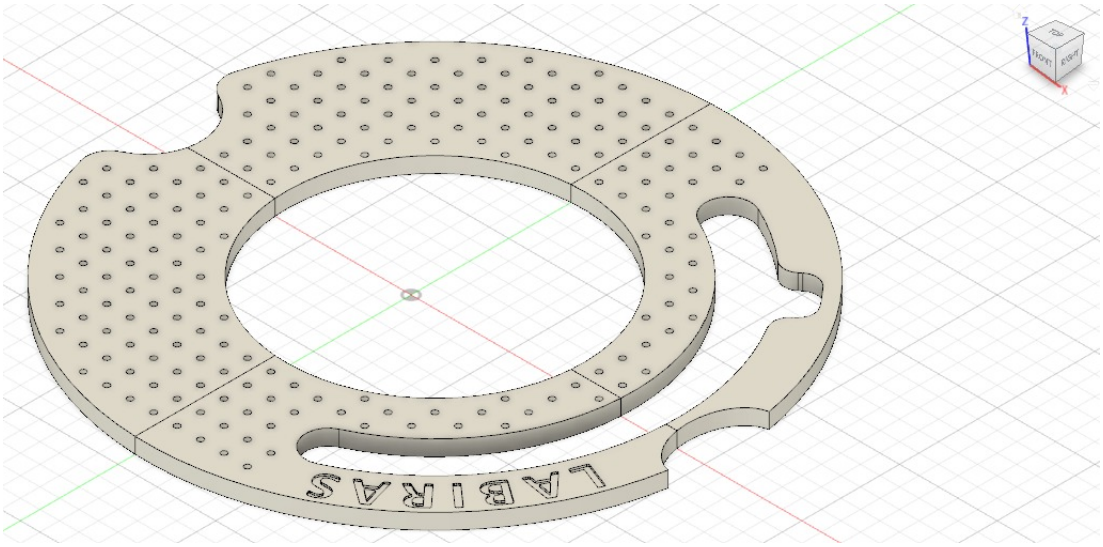
Figura 22 – Dimensões da placa do Create 3



Fonte: (SHAMLIAN, 2022)

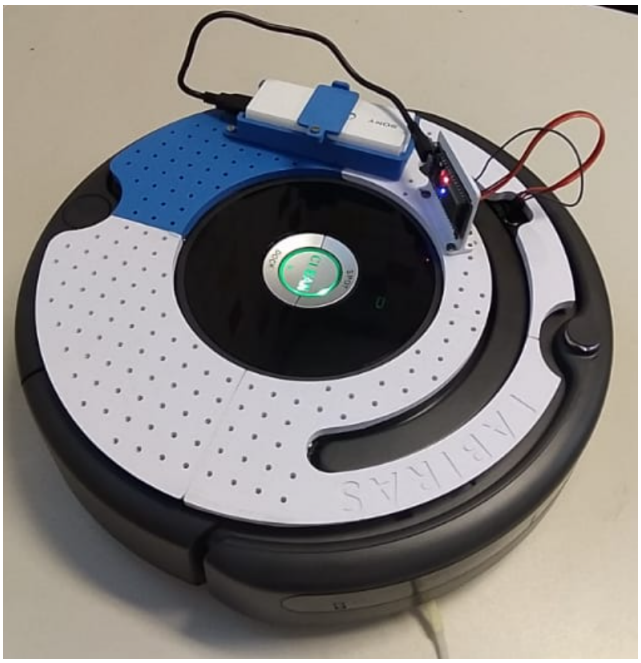
Manter o padrão de furação do Create 3 possibilita utilizar os projetos já existentes de suportes impressos em 3D para os sensores mais comuns. Caso não exista um suporte para o sensor desejado, quando for criado para esse robô ou para o Create 3, será compatível com os dois modelos.

Figura 23 – Modelo 3D da base para sensores



Fonte: Elaborado pelo autor.

Figura 24 – Roomba montado com ESP 32, bateria e a base para adição de sensores variados



Fonte: Elaborado pelo autor.

7 CONCLUSÃO

Utilizando o robô Roomba foi possível construir uma plataforma robótica genérica que comunica com ROS 2, simulando um robô educacional comercial que está indisponível no Brasil e é usado ao redor do mundo para treinamentos e capacitações. Essa plataforma pode ser aplicada em sala de aula para ensino ou pode ser utilizada em projetos de pesquisa para desenvolver projetos de maneira mais rápida, sem a preocupação de criar uma base móvel partindo do zero.

Foi possível operar o robô utilizando os *packages* padrões do ROS, devido a padronização das mensagens. Receber os dados dos sensores selecionados e visualizá-los no computador utilizando as ferramentas gráficas do ROS. É possível utilizar sensores variados ou fazer adaptações dependendo do uso, aproveitando a base impressa em 3D. E todas as informações do projeto estão documentadas e disponíveis para os alunos que utilizarão o laboratório LABIRAS.

Com esta plataforma é possível realizar de forma prática aulas sobre ROS, que é uma ferramenta amplamente utilizada na indústria, e preparar os alunos de engenharia mecânica do IFPI tirando proveito da disciplina de robótica prevista no PPC do curso, não limitado somente a graduação de engenharia mecânica, mas também os cursos de mestrado de materiais e graduação em engenharia civil. Já que até então não existia nenhuma ferramenta do tipo na instituição.

8 TRABALHOS FUTUROS

Este trabalho é apenas o primeiro passo para explorar o vasto campo da robótica. Ainda é possível adicionar muito mais funcionalidades, criar soluções e desenvolver pesquisas com esse robô. A seguir está uma lista com algumas das possibilidades que podem ser desenvolvidas utilizando a plataforma como base:

- Desvio de obstáculos: apenas com os sensores já disponíveis é possível fazer o robô seguir uma rota e desviar de obstáculos autonomamente.
- Solucionador de labirintos: existem competições para solucionar labirintos, como por exemplo o *micromouse*. O Roomba já possui sensores de distância na frente e codificadores nas rodas, que normalmente são os sensores utilizados nessas competições, ele pode ser usado para testar os algoritmos que são utilizados nessas competições.
- Seguidor de linha: adicionar sensores de cor na frente do robô e aprender sobre algoritmos de seguidores de linha.
- SLAM: vários tipos de SLAM existem, usando diferentes sensores, o robô já possui valores de codificadores nas rodas, no LABIRAS existem *lidar*, câmera de profundidade, *webcam*, que podem ser adicionadas ao robô para realizar navegação.
- Base móvel com manipulador: é comum aplicações onde se usa em conjunto um robô móvel com um manipulador acoplado, como exemplo o robô da KUKA youBot. Graças a base impressa em 3D é possível acoplar um braço robótico.
- Múltiplos robôs: Atualmente existem três robôs disponíveis no laboratório, é possível utilizar os três ao mesmo tempo e fazer gerenciamento de frota de robôs ou trabalhos em conjunto com os três.

REFERÊNCIAS

AMATO, D. **iRobot-Create2-Arduino**. 2016. Disponível em: https://github.com/brinnLabs/Create2. Acesso em: 19 set. 2022.

BLAKE, K.; HUEY, R.; MENEZES, D.; ROOT, J.; TRAN, L.; CHEN, H. Robot navigation using ultra-wideband indoor localization and dead reckoning algorithms. In: IEEE. **2022 Opportunity Research Scholars Symposium (ORSS)**. [S.l.], 2022. p. 12–15.

BOSCO, A. **Nova engenharia passa por inteligência artificial e robótica**. 2021. Disponível em: https://www.crea-mg.org.br/comunicacao/sala-de-imprensa/release/Nova%20engenharia%20passa%20por%20intelig%C3%AAncia%20artificial%20e%20rob%C3%B3tica. Acesso em: 21 ago. 2022.

DUBEY, S.; PRATEEK, M.; SAXENA, M. *et al.* Robot locomotion–a review. **Int. J. Appl. Eng. Res**, v. 10, n. 3, p. 7357–7369, 2015.

FINOCCHIARO, F. **micro-ROS porting to ESP32**. 2020. Disponível em: https://micro.ros.org/blog/2020/08/27/esp32/. Acesso em: 06 out. 2022.

FOOTE, T. **Why ROS?** 2022. Disponível em: https://www.ros.org/blog/why-ros/. Acesso em: 19 jul. 2022.

HELLSTRÖM, T. **Kinematics equations for differential drive and articulated steering**. [S.l.]: Department of Computing Science, Umeå University, 2011.

IFPI. **Teresina Central abre inscrições para curso de Robótica para Engenharias**. 2022. Disponível em: https://www.ifpi.edu.br/teresinacentral/noticias/teresina-central-abre-inscricoes-para-curso-de-robotica-para-engenharias. Acesso em: 22 ago. 2022.

IROBOT. **iRobot® Create® 2 Open Interface (OI) Specification based on the iRobot® Roomba® 600**. 2015. Disponível em: https://cdn-shop.adafruit.com/datasheets/create_2_Open_Interface_Spec.pdf. Acesso em: 29 ago. 2022.

IROBOT. **Robô Aspirador de Pó Inteligente Bivolt Roomba® 614 iRobot**. 2020. Disponível em: https://www.irobotloja.com.br/roomba-614-robo-aspirador-de-po-inteligente-bivolt-irobot-p/p. Acesso em: 23 ago. 2022.

ISMAIEL, A.; FILIMONOV, M. Y. A simulation-based study to calculate all the possible trajectories of differential drive mobile robot. In: AIP PUBLISHING LLC. **AIP Conference Proceedings**. [S.l.], 2021. v. 2333, n. 1, p. 070008.

JOSEPH, L. **ROS Robotics Projects**. 1st. ed. Birmingham: Packt Publishing Ltd., 2017.

KOUBÂA, A. *et al.* **Robot Operating System (ROS).** [S.l.]: Springer, 2017. v. 1.

LALANCETTE, C.; POUBEL, L. **ROS Index**. 2022. Disponível em: https://index.ros.org/stats/. Acesso em: 19 jul. 2022.

LEMON, M.; ANGLEA, T.; WANG, Y. Experimental study of decentralized robot network coordination. In: . [S.l.: s.n.], 2019.

LOPES, R. de A.; FORTES, A. F. C.; PEDROSA, A. Í. R.; ARAÚJO, A. G. F. de; ALVES, R. M. **PROJETO PEDAGÓGICO DO CURSO DE ENGENHARIA MECÂNICA NA MODALIDADE PRESENCIAL CAMPUS TERESINA CENTRAL**. 2018. Disponível em: https://www.ifpi.edu.br/cursos/documentos-dos-cursos/ppc/PPCEMETSA.pdf. Acesso em: 23 ago. 2022.

MACENSKI, S.; FOOTE, T.; GERKEY, B.; LALANCETTE, C.; WOODALL, W. Robot operating system 2: Design, architecture, and uses in the wild. **Science Robotics**, American Association for the Advancement of Science, v. 7, n. 66, p. eabm6074, 2022.

MALU, S. K.; MAJUMDAR, J. Kinematics, localization and control of differential drive mobile robot. **Global Journal of Research In Engineering**, 2014.

MARSH, A. The proto-roomba - [past forward]. **IEEE Spectrum**, v. 57, n. 3, p. 50–50, 2020.

MARUYAMA, Y.; KATO, S.; AZUMI, T. Exploring the performance of ros2. In: . [S.l.: s.n.], 2016. p. 1–10.

MILLER, I. D.; CLADERA, F.; COWLEY, A.; SHIVAKUMAR, S. S.; LEE, E. S.; JARIN-LIPSCHITZ, L.; BHAT, A.; RODRIGUES, N.; ZHOU, A.; COHEN, A. *et al.* Mine tunnel exploration using multiple quadrupedal robots. **IEEE Robotics and Automation Letters**, IEEE, v. 5, n. 2, p. 2840–2847, 2020.

NATTHARITH, P.; GUZEL, M. S. An indoor mobile robot development: A low-cost platform for robotics research. In: . [S.l.: s.n.], 2014.

NILOY, M. A.; SHAMA, A.; CHAKRABORTTY, R. K.; RYAN, M. J.; BADAL, F. R.; TASNEEM, Z.; AHAMED, M. H.; MOYEEN, S. I.; DAS, S. K.; ALI, M. F. *et al.* Critical design and control issues of indoor autonomous mobile robots: A review. **IEEE Access**, IEEE, v. 9, p. 35338–35370, 2021.

PERRON, J. **create_robot**. 2022. Disponível em: https://github.com/AutonomyLab/create_robot. Acesso em: 03 out. 2022.

PYO, Y.; CHO, H.; JUNG, R.; LIM, T. **ROS Robot Programming**. 1st. ed. Seoul,: ROBOTIS Co.,Ltd., 2017.

QUIGLEY, M.; CONLEY, K.; GERKEY, B.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A. Y. *et al.* Ros: an open-source robot operating system. In: KOBE, JAPAN. **ICRA workshop on open source software**. [S.l.], 2009. v. 3, n. 3.2, p. 5.

QUIGLEY, M.; GERKEY, B.; SMART, W. D. **Programming Robots with ROS: a practical introduction to the Robot Operating System**. [S.l.]: "O'Reilly Media, Inc.", 2015.

ROBOTICS, I. F. of. **World Robotics 2020 Report**. 2020. Disponível em: http://reparti.free.fr/robotics2000.pdf. Acesso em: 19 jul. 2022.

ROBOTICS, O. **ROS - Robot Operating System**. 2022. Disponível em: https://www.ros.org/. Acesso em: 19 jul. 2022.

ROJAS-FERNANDEZ, M.; MUJICA-VARGAS, D.; MATUZ-CRUZ, M.; LOPEZ-BORREGUERO, D. Performance comparison of 2d slam techniques available in ros using a differential drive robot. In: . [s.n.], 2018. v. 2018-January. Disponível em: https://ieeexplore.ieee.org/abstract/document/8327175. Acesso em: 19 set. 2022.

ROMANO, V.; DUTRA, M. Introdução a robótica industrial. **Robótica Industrial: Aplicação na Indústria de Manufatura e de Processo, São Paulo: Edgard Blücher**, p. 1–19, 2002.

ROMERO-MARTI, D. P.; NUNEZ-VARELA, J. I.; SOUBERVIELLE-MONTALVO, C.; OROZCO-DE-LA-PAZ, A. Navigation and path planning using reinforcement learning for a roomba robot. In: . [s.n.], 2017. Disponível em: https://ieeexplore.ieee.org/abstract/document/7955160. Acesso em: 19 set. 2022.

ROS micro. **micro-ROS puts ROS 2 onto microcontrollers**. 2022. Disponível em: https://micro.ros.org/. Acesso em: 02 ago. 2022.

RUBIO, F.; VALERO, F.; LLOPIS-ALBERT, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. **International Journal of Advanced Robotic Systems**, SAGE Publications Sage UK: London, England, v. 16, n. 2, p. 1729881419839596, 2019.

SCIAVICCO, L.; SICILIANO, B. **Robotica Industriale: modellistica e controllo di manipolatori**. [S.l.]: McGraw-Hill Libri Italia, 1995.

SHAMLIAN, S. **iRobot® Create® 3 Educational Robot**. 2021. Disponível em: https://iroboteducation.github.io/create3_docs/. Acesso em: 26 set. 2022.

SHAMLIAN, S. **iRobot® Create® 3 Mechanical System**. 2022. Disponível em: https://iroboteducation.github.io/create3_docs/hw/mechanical/. Acesso em: 09 out. 2022.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to autonomous mobile robots**. [S.l.]: MIT press, 2011.

STANDARTIZATION, I. O. for. **ISO 8373:2021(en)**. 2012. Disponível em: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-3:v1:en. Acesso em: 22 ago. 2022.

VILCHES, V. M. **ROS Robotics Companies**. 2022. Disponível em: https://github.com/vmayoral/ros-robotics-companies. Acesso em: 19 set. 2022.

WEON, E. S. **Understanding nodes**. 2022. Disponível em: https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html. Acesso em: 02 ago. 2022.

WILLIAMS, E.; ACVECEDO, M.; BHARMAL, A.; FOMITCHEV, V.; NICHOLS, L.; RICKETTS, K.; CHANDRASEKARAN, B.; ELLIBOL, E.; MORRIS, M.; INTEGLIA, R. Towards the development of junkyard hacks: networked robotics applications. In: **31st Florida Conference on Recent Advances in Robotics**. [S.l.: s.n.], 2018.