Somma - Revista Científica do Instituto Federal de Educação, Ciência e Tecnologia do Piauí

ISSN: 2447-701X
e181125
[Artigo]

Plataforma robótica com uso de robô aspirador, micro-ROS e platformIO IDE como ferramenta de desenvolvimento

Paulo Roberto Araújo Leal[1], Francisco Marcelino Almeida de Araújo[1], Halyson Itallo da Cunha Pimentel[1], Matheus Araújo Dantas[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Piauí – IFPI Campus Teresina Central, Teresina, Piauí, Brasil.

RESUMO

Introdução. A robótica móvel, um campo em constante evolução, exige frequentemente plataformas robóticas para testes e validação de hipóteses científicas. Contudo, a obtenção ou montagem dessas plataformas pode ser complexa, dispendiosa e demandar tempo significativo, muitas vezes não diretamente relacionado ao foco do estudo, além da dificuldade em encontrar robôs móveis robustos, acessíveis, bem documentados e reprodutíveis. **Objetivo**. O presente trabalho visa solucionar esses problemas desenvolvendo uma plataforma robótica móvel de baixo custo, modular, expansível e *open source*, utilizando ROS 2, micro-ROS, ESP32 e o robô aspirador iRobot Roomba® 614. **Material e Métodos**. A plataforma foi desenvolvida com base em equipamentos relativamente baratos e na arquitetura *open source* e modular. Foi realizada a integração do robô aspirador iRobot Roomba® 614 com ROS 2, micro-ROS e ESP32. **Resultados e Discussão**. Com a plataforma em funcionamento, foi possível executar testes de locomoção (*walktests*) e estabelecer um padrão para as leituras dos dados provenientes dos sensores disponíveis no Roomba 614. Essa padronização permite o uso direto desses dados em futuras aplicações e pesquisas. **Considerações Finais**. A plataforma robótica desenvolvida provê uma solução acessível e funcional para a pesquisa em robótica móvel, superando os desafios de custo e complexidade de plataformas tradicionais e demonstrando sua eficácia na padronização da leitura de dados sensoriais do Roomba 614.

PALAVRAS-CHAVE: Robótica móvel. ROS. PlatformIO. ESP32. Roomba.

Robotic platform using robot vacuum cleaner, micro-ROS and IDE platform as a development tool

ABSTRACT

Introduction. Mobile robotics, a constantly evolving field, often requires robotic platforms for testing and validating scientific hypotheses. However, acquiring or assembling such platforms can be complex, costly, and time-consuming, frequently diverting focus from the core objectives of the study. Additionally, there is a lack of robust, affordable, well-documented, and reproducible mobile robots. **Objective**. This work aims to address these issues by developing a low-cost, modular, expandable, and open-source mobile robotic platform using ROS 2, micro-ROS, ESP32, and the iRobot Roomba® 614 vacuum robot.

Correspondência dos Autores

[1a]ORCID: 0000-0002-6173-3123. Email: pauloraraujoleal@gmail.com (Autor correspondente)
[1b]ORCID: 0000-0001-8928-0077. Email: francisco.marcelino@ifpi.edu.br
[1c]ORCID: 0000-0003-4019-6848. Email: halyssonpimentell@gmail.com
[1d]ORCID: 0000-0002-5718-9137. Email: matheussdantas19@gmail.com

Artigo submetido a sistemas de verificação de similaridade

Submetido 24/11/2023 – **Aceito** 29/07/2025 – **Publicado** 26/12/2025

Materials and Methods. The platform was developed based on relatively inexpensive hardware and an open-source, modular architecture. The integration of the iRobot Roomba® 614 with ROS 2, micro-ROS, and ESP32 was carried out. **Results and Discussion**. With the platform in operation, it was possible to perform locomotion tests (walk tests) and establish a standard for reading data from the sensors available on the Roomba 614. This standardization enables the direct use of these data in future applications and research. **Final Considerations**. The developed robotic platform provides an accessible and functional solution for mobile robotics research, overcoming the cost and complexity challenges of traditional platforms and demonstrating its effectiveness in standardizing the reading of sensory data from the Roomba 614.

KEYWORDS: Mobile robotics. ROS. PlatformIO. ESP32. Roomba.

1 INTRODUÇÃO

Para a civilização ocidental, o conceito de evolução humana está diretamente relacionado ao nível de desenvolvimento tecnológico alcançado ao longo do tempo (Romano; Dutra, 2002). Assim, a motivação para criar máquinas que possam substituir os humanos na execução de tarefas é característica da própria cultura ocidental. Diante dessa perspectiva, em 2020 foi registrado o recorde de 2.7 milhões de robôs industriais operando em fábricas ao redor do mundo, sendo que, na América do Sul, o Brasil é o país líder com estoque operacional com quase 15.300 unidades (Robotics, 2020).

Diante disso, a robótica móvel é um dos segmentos da robótica que mais está ganhando notoriedade, pois além do crescimento do espaço da robótica na indústria, muitos esforços estão sendo colocados na robótica educacional, com uma abordagem mais acadêmica e de pesquisa. Assim, é necessária uma plataforma robótica móvel de baixo custo, *open source*, modular e expansível para testar hipóteses de uma pesquisa científica na prática (Cruz *et al*., 2022). Ademais, também é apontado pelos autores supracitados que a montagem de plataformas robóticas pode ser estruturalmente complexa ou não ser diretamente relacionada ao escopo do estudo, como a implementações de funcionalidades de software, resultando em uma significativa perda de tempo, enquanto preços elevados podem desencorajar a compra. Além disso, encontrar robôs móveis replicáveis, baratos e bem documentados não é uma tarefa fácil.

Nessa perspectiva, a fim de realizar práticas testando o conhecimento adquirido no campo da robótica, popularizou-se o hábito de utilizar aspiradores automatizados como base experimental, com o objetivo de utilizar equipamentos confiáveis, sem a necessidade de projetar e fabricar equipamentos do zero. Esses aspiradores robóticos têm sido utilizados mundialmente como base diferencial móvel, como pode ser visto em Williams *et al*. (2018), Araújo *et al*. (2015) e Ruiz *et al*. (2013). Entretanto, mesmo com a possibilidade da aquisição de uma base móvel pronta, muitos outros conhecimentos são requisitados na elaboração de um projeto robótico. Sob essa óptica, Romano e Dutra (2002) descrevem que um projeto de robô é interdisciplinar e que utiliza de diversas áreas clássicas como engenharia mecânica, engenharia elétrica e engenharia eletrônica, teoria de controle e ciência da computação. Também destacam que é preciso considerar aspectos como dimensionamento de atuadores, mecanismos, circuitos eletrônicos, unidades de controle e potência, cálculos estruturais, simulação e modelagem, desenvolvimento de técnicas de programação para o sistema de controle, sistema operacional, diagnóstico de sistemas/componentes e comunicação ao operador, dentre outros.

Mesmo os sistemas robóticos mais simples envolvem várias etapas complexas. O gerenciamento de inúmeros dispositivos eletrônicos (*hardware*) diferentes, como por exemplo os microcontroladores, sensores, atuadores, impactam diretamente no desenvolvimento do software, pois exigem do programador um grande grupo de habilidades específicas. Assim, a programação para sistemas embarcados ou embutidos de robôs pode não ser uma tarefa tão trivial, já que a criação de uma arquitetura de *software* do zero pode levar bastante tempo. Desse modo, é nesse meio que emergem estudos sobre uma arquitetura de *software* que conseguem abstrair a complexidade do hardware e oferecer ao desenvolvedor maior modularidade, como pode ser visto em Gutiérrez-Flores e Bachmayer (2022).

Entre as muitas ferramentas que permitem o desenvolvimento de novas utilizações na área da robótica, há o *Robot Operating System* (ROS) (De Oliveira Júnior *et al*., 2022), que, de acordo com o site oficial, "O ROS provê uma estrutura flexível para escrever software robótico. É um conjunto de ferramentas, bibliotecas e esquemas projetados para facilitar a construção de um comportamento robótico robusto e complexo em ampla gama de plataformas de robôs" (ROS, [s.d.]). O ROS é um meta-sistema operacional, com abstração de *hardware*, gerenciamento de dispositivos de baixo nível, implementação de funções postas com frequência, passagem de mensagens entre processos e gerenciamento de pacotes (Quigley *et al*., 2009; Koubaa, 2016; Macenski *et al*., 2022).

O intuito deste trabalho está na utilização de uma plataforma comercial de robôs terrestres relativamente acessíveis para o desenvolvimento de uma estrutura robótica capaz de receber e enviar informações no padrão ROS. Desse modo, é possível realizar atividades de movimentação e coleta de dados baseados em algoritmos programados em um computador por meio do ROS 2, visando tornar mais acessível e prático o ensino, a prática e o desenvolvimento científico no âmbito da robótica móvel. Assim, apresentamos um estudo de caso no qual é empregado o Visual Studio Code, por intermédio da extensão PlatformIO, como ambiente para desenvolvimento com o microcontrolador ESP32 em conjunto com o micro-ROS, a fim de controlar o robô iRobot Roomba® 614, um dispositivo de limpeza doméstico que possui diversos sensores e atuadores, tornando-o uma plataforma ideal para testar e desenvolver sistemas robóticos. Dessa forma, os fundamentos teóricos que se seguem vêm reforçar argumentações sobre a importância de elaborar uma plataforma robótica com ROS que auxilie e sirva de base para fixação e prática dos conhecimentos adquiridos durante o processo de aprendizagem da robótica, bem como para facilitar no desenvolvimento de pesquisas acadêmicas, visto a crescente expansão desse ramo no contexto geral da sociedade.

2 FUNDAMENTOS TEÓRICOS

2.1 Robot Operating System

Robot Operating System (ROS) é um conjunto de ferramentas de *software open source* que ajuda a construir robôs móveis e manipuladores. Ele fornece uma arquitetura de programação para robôs e uma grande variedade de bibliotecas e ferramentas para tarefas comuns, como navegação, planejamento de movimento, reconhecimento de voz e visão, processamento de sensores e gerenciamento de dispositivos. ROS foi desenvolvido pela Willow Garage, uma empresa de robótica de

pesquisa, em 2007 e é atualmente mantido pela Open Robotics. Ele é amplamente utilizado em robótica acadêmica e de pesquisa, bem como em aplicações industriais.

O ROS permite comunicar softwares que utilizam diferentes linguagens de programação, bem como ligar diferentes dispositivos de entrada ou saída de forma trivial e, além desta particularidade, reúne os esforços de vários desenvolvedores desta área (Quigley *et al*., 2015). São mais de 7000 pacotes desenvolvidos para as diversas distribuições do ROS ("ROS Index", [s.d.]). Outrossim, sua arquitetura de nó de controle (*Nodes*) distribuída e fracamente acoplada permite ao projeto ser construído em formato de módulos para obter baixa adesão e alta internalização, além de melhorar a taxa de reutilização de código e a eficiência do desenvolvimento (Alisher; Alexander; Alexandr, 2015). Atualmente o ROS está na sua segunda versão, o ROS 2, sendo que, normalmente quando o termo ROS é citado, está sendo referenciado a comunidade e o projeto como um conceito geral.

2.2 *Microcontroller Robot Operating System*

Em consoante com o ROS (2022), o micro-ROS é uma implementação do sistema operacional ROS para dispositivos embarcados com recursos limitados. Ele é projetado para ser executado em dispositivos com baixo poder de processamento, memória e armazenamento, como microcontroladores e sistemas de tempo real (Gutiérrez-Flores; Bachmayer, 2022). Ele fornece as mesmas funcionalidades do ROS tradicional, mas com uma estrutura de *software* mais leve e adaptada para dispositivos embarcados, permitindo que os desenvolvedores de robôs implementem facilmente a inteligência e o controle em dispositivos embarcados, sem sacrificar desempenho e capacidade de resposta (Micro-ROS - Fiware, 2021).

Em seguimento, o micro-ROS serve de ponte para a comunicação entre a parte do *hardware* e o sistema operacional do ROS 2 para que, dessa forma, os microcontroladores possam ser dedicados para tarefas específicas de *hardware*, com latência baixa, determinística e baixo consumo de energia através da criação de um *link* com processadores maiores e mais aptos a executar o ROS 2, em aplicações robóticas (Features and Architectures, 2023).

2.3 ESP32

O ESP32 é uma série de microcontroladores de baixo custo e baixo consumo de energia desenvolvidos pela Espressif Systems. Ele é baseado na arquitetura de 32 bits Xtensa LX6 e inclui recursos como *WiFi*, *bluetooth*, entradas/saídas digitais e analógicas, e uma variedade de interfaces como SPI, I2C, e UART. Esta placa microcontroladora é amplamente utilizada em projetos de Internet das coisas (IoT) e outros projetos eletrônicos devido às suas especificações serem melhores que as placas Arduino.

2.4 Arduino IDE

Arduino IDE, de acordo com o site oficial, é um ambiente de desenvolvimento integrado (IDE) para programação de microcontroladores Arduino (Arduino IDE, [s.d.]). Ele é gratuito e de código aberto, e permite escrever, compilar e carregar código em dispositivos Arduino, fornecendo uma interface gráfica fácil de usar para acessar seus recursos, como portas seriais e pinos digitais e analógicos. Além disso, o Arduino IDE possui uma ampla biblioteca de exemplos e bibliotecas adicionais que podem ser instaladas para expandir as capacidades do dispositivo Arduino, além de suportar uma variedade de sistemas operacionais, incluindo Windows, MacOS e Linux.

O Arduino IDE se torna uma alternativa para a codificação de microcontroladores, visto que é uma aplicação multiplataforma de confecção de *softwares* livres que permite programar e carregar códigos para diversas placas microcontroladoras, sendo, principalmente, placas Arduino, mas com capacidade de inclusão de novas bibliotecas e placas microcontroladoras a partir da adição de pacotes externos (Peña, 2020). Entretanto, o ambiente de desenvolvimento Arduino IDE não apresenta *snippets*, ou seja, um analisador de código em tempo de desenvolvimento, realce de sintaxe para bibliotecas externas, e recursos como auto completar ou sugestões para complemento de código, o que não agrega ao processo de desenvolvimento, mas impacta na produtividade, além de que a inclusão de bibliotecas externas exige muitos passos, que em sua maioria são poucos intuitivos, agravando o problema de atrasar o processo de desenvolvimento.

2.5 Visual Studio Code

Conforme o site oficial, o *Visual Studio Code* (VS Code) é um editor de código-fonte leve, mas com poderosas ferramentas de desenvolvedor, como conclusão e depuração de código *IntelliSense*. Ele é um ambiente que pode ser executado em sua área de trabalho, estando disponível para sistemas Windows, macOS e Linux, apresentando também suporte para diversas linguagens. Essa ferramenta de desenvolvimento vem com suporte integrado para JavaScript, TypeScript e Node.js e possui um rico ecossistema de extensões para outras linguagens e tempos de execução, como: C++, C#, Java, Python, PHP, Go, .NET, dentre muitos outros (Salamah *et al*., 2021). Ademais, apresenta características como realce de sintaxe, correspondência de colchetes, recuo automático, seleção de caixa, *snippets*, bem como suporte interno para preenchimento de código *IntelliSense*, entendimento e navegação de código semântico avançado e refatoração de código (VISUAL STUDIO CODE, [s.d.]).

2.6 PlatformIO IDE

O PlatformIO IDE é uma plataforma que promove aos desenvolvedores um ambiente moderno e integrado. Ela é voltada para o mercado de sistemas embarcados, funciona em multiplataformas, suporta diferentes kits de desenvolvimento de softwares ou frameworks, e inclui *debugging*, testes de unidade, análises de código automática e gerenciamento remoto (Platformio, [s.d.]). Além disso, é uma ferramenta que possui uma arquitetura descentralizada, e com isso, fornece, para antigos e novos profissionais que trabalham com o uso de mais de uma plataforma para desenvolvimento embarcado, um caminho rápido de integração para produção de produtos prontos para consumo geral, e redução do tempo para inserção do produto no mercado.

Nesse sentido, segundo a documentação oficial da PlatformIO IDE, a plataforma foi criada com o objetivo de maximizar a flexibilidade e escolha dos desenvolvedores, de maneira que eles possam utilizar tanto comandos gráficos, quanto comandos de editores, ou os dois. Desse modo, em uma área tradicionalmente ocupada por ferramentas de software complexas, manuseadas por experientes engenheiros de *hardware*, que normalmente aprenderam com imensa dificuldade, agora essas ferramentas de desenvolvimento embarcado podem ser usufruídas por amadores ou profissionais. Eles podem produzir produtos comercializáveis importando o clássico esboço “Blink” do Arduino ou desenvolvendo programas sofisticados de baixo nível em C.

O PlatformIO IDE é instalado como uma extensão do *VS Code*, o que possibilita, caso se torne necessário usar outras ferramentas de desenvolvimento, como Visual Studio, CLion, etc., integrar esta

plataforma a essas ferramentas de desenvolvimento (Babiuch; Foltýnek, 2021). Desse modo, a integração desses ambientes de desenvolvimento com a PlatformIO IDE torna factível a utilização dessas ferramentas que fornecem para o desenvolvedor um ambiente de maior liberdade de manipulação de código, e com isso, diminuir limitações práticas, a exemplo das previamente citadas que se manifestam no Arduino IDE, e acelerar o processo de conclusão de um projeto baseado em microcontroladores.

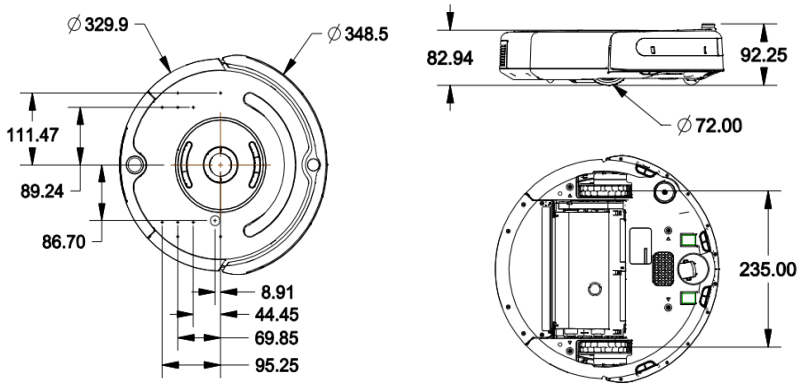
2.7 Locomoção e robôs móveis com acionamento diferencial

Dentro do campo da robótica, a locomoção é um fator primordial para que robôs possam se movimentar no espaço de trabalho e que não só dependa da natureza do ambiente ou do trajeto que percorrerá, mas também de aspectos vitais como a habilidade de ser controlado, o grau de manobrabilidade, eficiência e estabilidade em terrenos diferentes (Rubio *et al*., 2019). Nesta óptica, uma configuração comumente utilizada é o emprego de duas rodas padrões e uma roda esférica ou de rodízio. Essa configuração permite bastante manobrabilidade, que é a habilidade de manobrar em qualquer direção de forma gradual, e uma estabilidade aceitável para robôs mais simples, como os robôs por acionamento diferencial (Dubey; Prateek; Saxena, 2015). Essa categoria de robô é amplamente utilizada em aplicações industriais, como automação de fábrica, transporte de materiais e limpeza automatizada, sendo eles controlados por meio de um sistema de controle baseado em computador, que é responsável por calcular a velocidade e a direção das rodas. Isso é feito usando informações de sensores, como sensores de proximidade e câmeras, para navegar pelo ambiente.

2.8 iRobot Roomba® 614

iRobot Roomba® 614 é um robô aspirador que consegue suportar uma carga de 15 quilogramas, sendo que o robô pesa cerca de 3,5 quilogramas (IROBOT, 2015). Suas dimensões podem ser visualizadas na Figura 1.

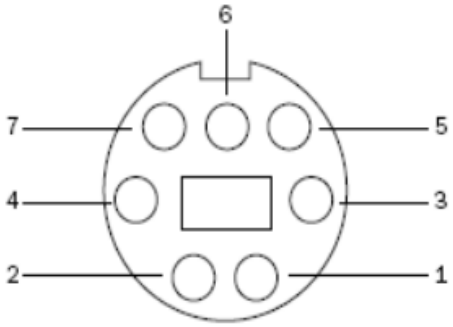
Figura 1: Vista superior, lateral e inferior do robô aspirador da iRobot Roomba® 614. Medidas em milímetros.



Fonte: Retirando de IROBOT, 2015.

Ele dispõe de diversos sensores, sendo eles de detecção de bordas, botões superiores, contadores de roda, nível de bateria, de batida, queda de roda, infravermelho, temperatura, voltagem e corrente elétrica. Diante disso, o robô apresenta um conector serial mini-din, no qual é possível obter e enviar comandos para o dispositivo robótico (Figura 2).

Figura 2: Conector mini-din do Roomba 614.



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: Retirado de IROBOT, 2015.

2.9 Trabalhos relacionados

O trabalho vigente teve como inspiração predominante o robô Create 3 iRobot (Shamlian, 2021). Entretanto, infelizmente, o mesmo não encontra-se disponível para compra no Brasil. Sob esse viés, uma alternativa viável é a utilização do robô da série Roomba como base para desenvolver uma versão própria. Como exposto a seguir, essa metodologia já foi implementada anteriormente, visto que o emprego do Roomba facilita o desenvolvimento da robótica, possibilitando pular algumas etapas da criação do robô. Deste modo, pode-se verificar o uso do Roomba como alternativa factível de plataforma de robótica para fins variados nos trabalhos de Blake *et al*. (2022), Lemon, Anglea e Wang (2019), Williams *et al*. (2018), Rojas-Fernández *et al*. (2018), Romero-Martí *et al*. (2016), Nattharith e Guzel (2014), dentre outras.

3 MATERIAL E MÉTODOS

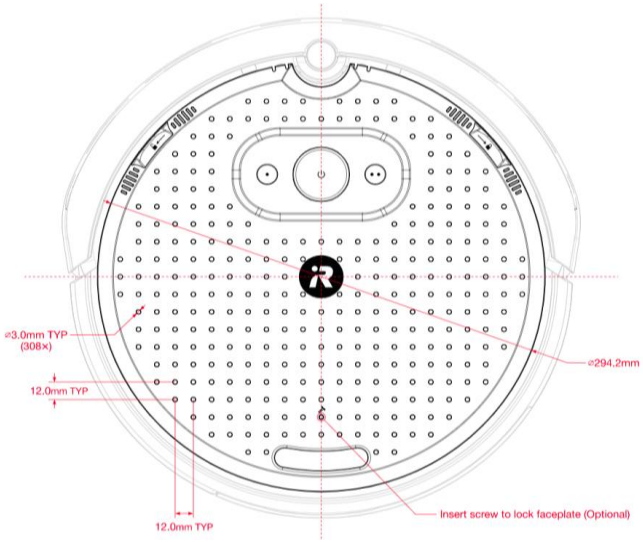
Para a realização deste trabalho, foram seguidas as sequências de ações listadas abaixo. Inicialmente, foram realizadas pesquisas bibliográficas com o objetivo de procurar trabalhos semelhantes que utilizassem as tecnologias abordadas neste trabalho.

3.1 Placa de fixação

No presente trabalho, como já supracitado, o robô é capaz de transportar uma carga útil de 15 kg. Diante disso, a área superior do robô pode ser aproveitada para a criação de uma base que possa servir para abrigar vários sensores e modificações. Sob essa perspectiva, será adotado como referência

o modelo iRobot Create 3 para a criação de uma base superior para fixação de módulos externos à plataforma. O padrão utilizado é composto por diversos furos de 3 mm de diâmetro, conhecidos como M3, com um espaçamento entre eles de 12 mm, ao qual apresentam distribuição em um padrão retangular, como mostrado na Figura 3.

Figura 3: Dimensões da placa do iRobot Create 3.



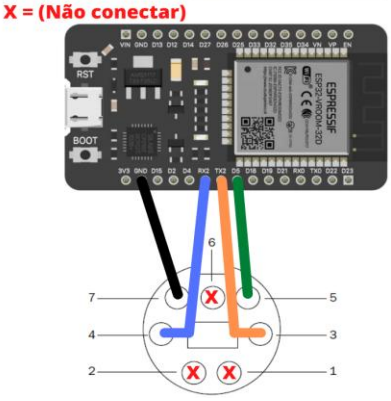
Fonte: Retirado de Shamlian, 2022.

3.2 Hardware

Neste trabalho foram utilizados o robô aspirador iRobot Roomba® 614 como plataforma robótica móvel e o microcontrolador ESP-WROOM-32 para enviar comandos via serial para o Roomba através do conector mini-din do robô, como apresentado na Figura 4, que expõe a conexão dos pinos.

Figura 4: Pinagem para conexão ESP32 e Roomba 614.

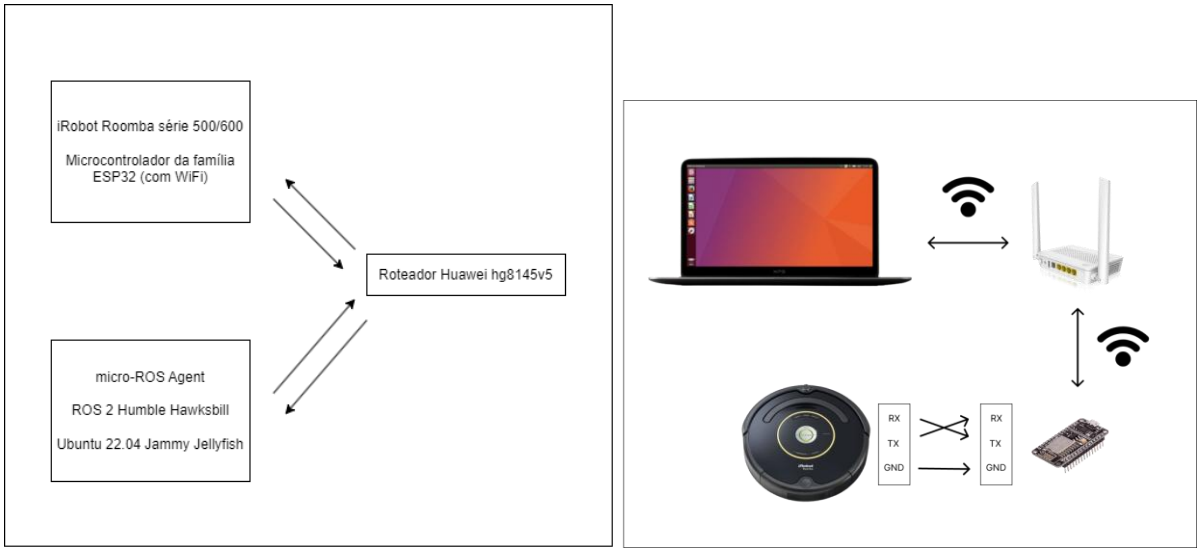
Pin	Name	Description
1	Vpwr	Roomba battery + (unregulated)
2	Vpwr	Roomba battery + (unregulated)
3	RXD	0 – 5V Serial input to Roomba
4	TXD	0 – 5V Serial output from Roomba
5	DD	Device Detect input (active low) – used to wake up Roomba from sleep
6	GND	Roomba battery ground
7	GND	Roomba battery ground



Fonte: Elaborado pelos autores, 2023.

O ESP32 se conecta à rede Wi-Fi do roteador Huawei hg814v5, onde recebe as mensagens padronizadas de acordo com ROS 2 *Humble Hawksbill*, isto é, o sistema operacional subjacente no computador é o Ubuntu 22.04 LTS, codinome Jammy Jellyfish, em que este computador está conectado à mesma rede Wi-Fi, como apresentado pelo diagrama na Figura 5.

Figura 5: Estrutura e diagrama de comunicação do projeto, mostrando a interação entre o robô, o microcontrolador ESP32, o agente micro-ROS em ambiente ROS 2 e a rede Wi-Fi utilizada para a troca de dados.



Fonte: Elaborado pelos autores, 2023.

O DualShock 4 conectado ao computador via *Bluetooth* foi usado como acionador das entradas para controlar o robô através do pacote *teleop_twist_joy*, também havendo a possibilidade do uso do pacote *teleop_twist_keyboard* para movimentação do robô via teclado. O ROS 2, por sua vez, recebe os *inputs* do controle e transforma essas informações em comandos de velocidade padrão do ROS 2, para posterior envio ao ESP32 por intermédio da conexão estabelecida pelo micro-ROS via Wi-Fi e, deste modo, a transformação deste dado em comando serial para mover os motores do Roomba 614 (Figura 5).

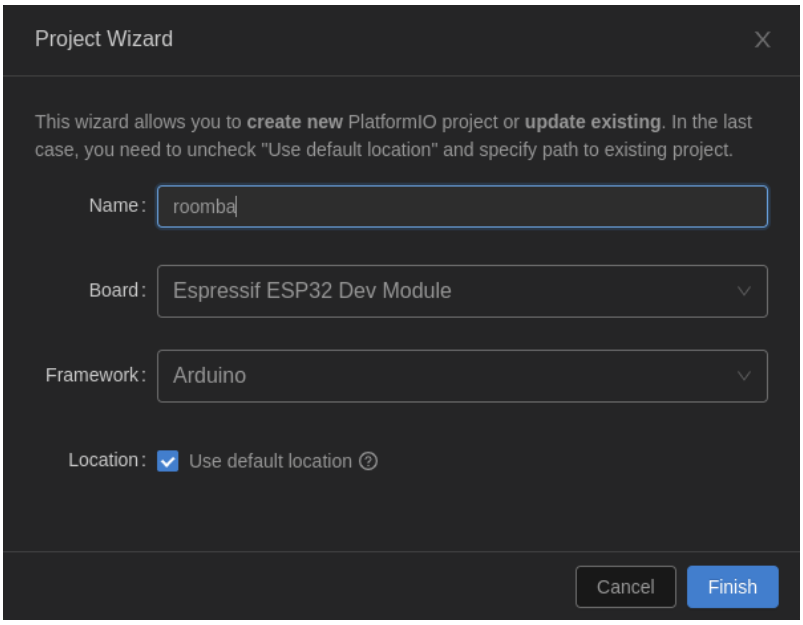
O padrão básico de mensagens utilizado para a movimentação de robôs é composto de dois vetores de velocidade, sendo eles: Linear [x, y, z] (m/s) e Angular [x, y, z] (rad/s), dado que, geralmente, esse padrão de mensagens é nomeado de *cmd_vel*, onde apresenta interface *geometry_msgs/msg/Twist*.

3.3 Software

Para o desenvolvimento do algoritmo configurado na placa ESP32 foi utilizado a plataforma PlatformIO IDE como uma extensão do *VS Code* (Platformio, [s.d.]), que foi necessário, primeiramente, a preparação do ambiente de desenvolvimento, ao qual é preciso a instalação da extensão PlatformIO IDE no *VS Code*. Com a utilização dessa extensão, passa a ser possível a criação de projetos e *upload* de código para diversas placas de microcontroladores, de forma simples e prática.

Posteriormente, foi preciso realizar a criação de um projeto para a placa microcontroladora ESP32, como exposto na Figura 6.

Figura 6: Criação do projeto para a placa ESP32.



Fonte: Elaborado pelos autores, 2023.

Não obstante, tornou-se necessária a realização das pré-configurações do projeto para os microcontroladores ESP32 por meio do arquivo *platformio.ini*, localizado na pasta raiz do projeto. Nesse arquivo são definidos parâmetros essenciais para o funcionamento do sistema, tais como a taxa de transmissão da placa (*baudrate*), a velocidade de atualização do monitor serial (*monitor_speed*), o link da dependência do projeto *micro_ros_platformio* (*lib_deps*), a versão do ROS 2 a ser utilizada (*board_microros_distro*) e o método de comunicação entre o microcontrolador e o ROS (*board_microros_transport*), que pode ocorrer via Wi-Fi ou comunicação serial. Adicionalmente, de forma opcional, é possível especificar a porta de destino para o envio do código (*upload_port*) nos casos em que existam múltiplos dispositivos conectados ao computador; contudo, quando apenas o microcontrolador está conectado, a própria plataforma realiza automaticamente o reconhecimento da porta correspondente. A Figura 7 apresenta essa pré-configuração realizada.

Figura 7: Pré-configuração do arquivo *platformio.ini*.

```
11  [env:esp32dev]
12  platform = espressif32
13  board = esp32dev
14  framework = arduino
15  lib_deps = https://github.com/micro-ROS/micro_ros_platformio
16  board_microros_distro = foxy
17  board_microros_transport = wifi
18  monitor_speed = 115200
19  upload_speed = 921600
20  upload_port = /dev/ttyUSB0
```

Fonte: Elaborado pelos autores, 2023.

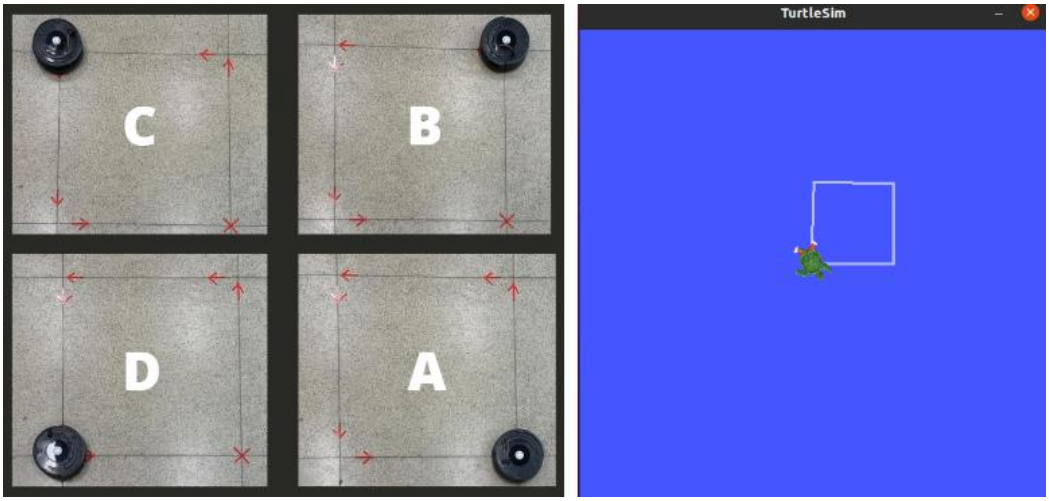
Por fim, a padronização de coleta das informações dos dados dos sensores foi realizada por intermédio do uso do pacote *create_msgs* (Perron, 2022) para posterior envio dos dados coletados para o ROS.

4 RESULTADOS E DISCUSSÃO

Primeiramente, foi constatado que o *hardware* do robô aspirador recebeu integralmente, por meio da conexão de rede sem fio, os comandos que foram enviados por pacotes padrões do ROS 2 que estavam sendo executados no computador. A plataforma robótica pôde executar, de maneira satisfatória, os movimentos aos quais foi programado no *software* de controle, completando o teste de validação de movimentação proposto pela equipe.

Na Figura 8 é apresentado um trajeto (quadrado) criado para a realização dos testes de movimento da plataforma, no qual foram registrados os movimentos executados simultaneamente nos dois ambientes: real e virtual.

Figura 11: Área marcada para visualização do trajeto esperado juntamente com os testes de movimentação em ambiente real e virtual.



Fonte: Elaborado pelos autores, 2023.

Por conseguinte, houve a categorização da padronização das mensagens do micro-ROS para captação dos sensores que estão à disposição no robô aspirador usando as ferramentas gráficas do ROS e o uso das informações no sistema do ROS 2. Para este trabalho, foram reunidos os sensores que estão listados na Tabela 1. Nela está a lista na ordem de requisição dos sensores, o nome do pacote do sensor, o número enviado na serial para a requisição do pacote do sensor e, por fim, o número de *bytes* que são devolvidos pelos pacotes. É passado por parâmetro de uma função o índice que evoca na matriz um código de operação para determinado sensor, que é limitado ao número máximo de 172 *bytes* em 15 ms, devido a taxa de transmissão máxima do Roomba ser 115200 de *baudrate*. Para outros projetos, podem ser selecionados sensores diferentes que sejam relevantes, desde que não ultrapassem o limite de *bytes* mencionados anteriormente, visto que, caso mais dados forem requisitados, o fluxo será corrompido.

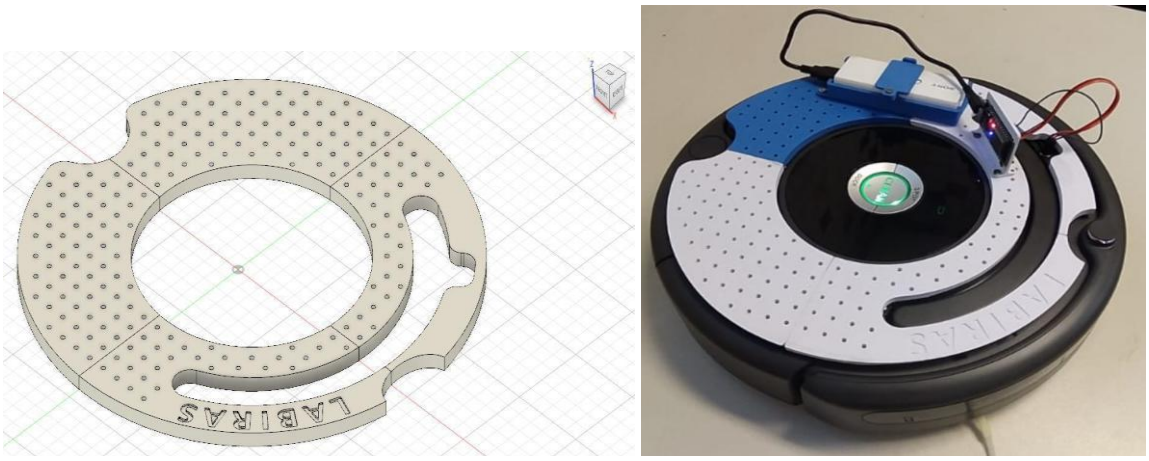
Tabela 1: Lista com os sensores que tiveram os seus dados coletados, com seus respectivos identificadores de pacote e tamanho dos dados coletados.

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

Fonte: Elaborado pelos autores, 2023.

Outrossim, a área superior do robô foi aproveitada para criar uma base que possa servir para abrigar vários sensores e modificações, atualmente, ela serve para fixar o ESP32 e sua bateria. A estrutura foi modelada usando o *software* Fusion 360, impresso em 3D e montado no Roomba (Figura 12).

Figura 12: Modelo 3D para adição de sensores e Roomba montado com ESP32, bateria e a base para adição de sensores variados.



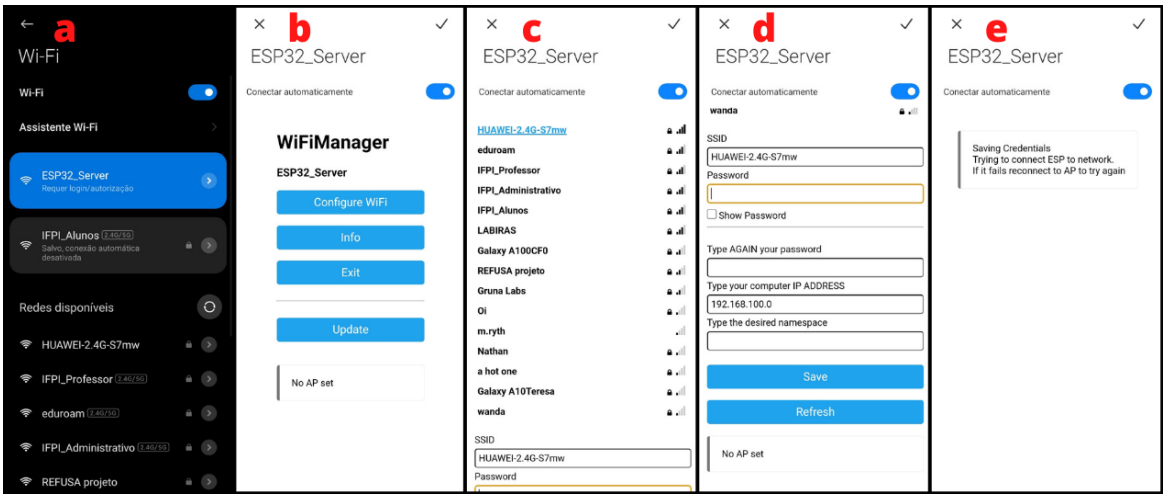
Fonte: Elaborado pelos autores, 2023.

Além disso, averiguou-se que o uso do Visual Studio Code juntamente com a extensão PlatformIO IDE agregou para um aumento significativo da produtividade durante todo o processo de desenvolvimento do algoritmo citado neste trabalho, sendo recursos como os *snippets* e o *autocomplete* grandes diferenciais em relação a outras ferramentas, como a já citada Arduino IDE. Não obstante, para a inserção dos parâmetros de rede necessários para a realização da comunicação entre o ESP32, com micro-ROS, e o computador, com o ROS 2, como nome do *WiFi*, senha e ipv4 do computador na rede, utilizou-se a biblioteca *WiFi Manager*, que possibilitou inseri-los através de uma página *web*. Nessa

página também é possível inserir parâmetros opcionais como *namespace*, que é usado para adicionar uma identificação adicional no ROS 2.

O passo a passo necessário para iniciar a operação do robô e inclusão dos parâmetros são detalhados na Figura 13: a) Conectar o *smartphone* na rede ESP32_Server; b) Clicar na opção *Configure WiFi*; c) Clicar no nome da rede desejada; d) Preencher os campos em branco (senha da rede *WiFi*, endereço IPV4 do computador na rede e o *namespace*, sendo este opcional); e) Aguardar a página fechar automaticamente (caso não feche, reinicie o ESP32 e retorne para o passo "a").

Figura 13: Configuração dos parâmetros do robô.



Fonte: Elaborado pelos autores, 2023.

Os testes finais demonstraram que o ESP32 com micro-ROS é uma solução viável para sistemas robóticos. O estudo de caso apresentado demonstrou que o ESP32 é capaz de controlar o Roomba 614 da iRobot de forma precisa e estável, além de ser capaz de coletar e fornecer acesso fácil aos dados dos sensores, o que amplia a gama de novas possibilidades de pesquisa e aplicação dessa plataforma.

5 CONSIDERAÇÕES FINAIS

Conclui-se que a utilização de um robô aspirador comercial como plataforma de validação para algoritmos de movimento de ação programados em Sistema Operacional Robótico é uma maneira viável para a prática em laboratório, visto que a praticidade de não ser necessária a construção de um *hardware* do zero para testar as hipóteses desenvolvidas, acelerou o processo de análise e execução da movimentação utilizando os comandos programados no software. A utilização do Visual Studio Code com a extensão PlatformIO IDE acelerou o processo de desenvolvimento e de depuração do código, por meio de funcionalidades como os *snippets* e o *autocomplete*, recursos estes que não estão presentes no Arduino IDE. A PlatformIO IDE se mostrou uma opção mais eficaz e produtiva que o Arduino IDE, pois além das vantagens citadas anteriormente, ela possui uma gestão de dependências mais objetiva e prática. Por conseguinte, com o concluído neste trabalho pode-se citar trabalhos futuros como desenvolvimento de algoritmos para execução de atividades de deslocamento e aplicações para criação de algoritmos de robôs *Micromouse*, um pequeno robô móvel com o objetivo de solucionar labirintos o mais rápido possível, normalmente usados em atividades competitivas.

AGRADECIMENTOS. Ao IFPI e ao laboratório do IFPI, o LABIRAS (*Laboratory of Intelligent Robotics, Automation and Systems*) pelo espaço e pelo suporte para realização dos experimentos.

CONTRIBUIÇÃO DOS AUTORES. **FMAA**: Conceituação; **PRAL, HICP, MAD**: Metodologia, Software, Curadoria de dados, Análise formal, Redação – rascunho original, Visualização, Investigação; **PRAL, HICP, MAD**: Redação – revisão e edição, Aquisição de financiamento, Supervisão. Todos os autores participaram ativamente da discussão dos resultados, revisaram e aprovaram a versão final do artigo.

CONFLITO DE INTERESSE. Os autores declaram que não têm interesse comercial ou associativo que represente um conflito de interesses em relação ao manuscrito.

APROVAÇÃO ÉTICA. Não se aplica.

REFERÊNCIAS

ALISHER, K.; ALEXANDER, K.; ALEXANDR, B. Control of the mobile robots with ROS in robotics courses. **Procedia Engineering**, v. 100, p. 1475-1484, 2015. Doi: 10.1016/j.proeng.2015.01.519. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877705815005469. Acesso em: 16 dez. 2025.

ARAÚJO, A.; PORTUGAL, D.; COUCEIRO, M. S.; ROCHA, R. P. Integrating Arduino-based educational mobile robots in ROS. **Journal of Intelligent & Robotic Systems**, v. 77, p. 281-298, 2015. Doi: 10.1007/s10846-013-0007-4. Disponível em: https://link.springer.com/article/10.1007/s10846-013-0007-4. Acesso em: 16 dez. 2025.

ARDUINO IDE. **Arduino IDE**. Disponível em:https://docs.arduino.cc/software/ide/. Acesso em: 15 dez. 2025.

BABIUCH, M.; FOLTÝNEK, P. Creating a mobile application with the ESP32 Azure IoT Development Board using a cloud platform. *In*: INTERNATIONAL CARPATHIAN CONTROL CONFERENCE (ICCC), 22., 2021. **Anais** [...]. IEEE, 2021. p. 1-4. Doi: 10.1109/ICCC51557.2021.9454607. Disponível em: https://ieeexplore.ieee.org/abstract/document/9454607/?casa_token=KgQ7sZomfgAAAAAA:vBpL4Rv85QzKUhq-Phzghmxx-uCYggCP5aY8OVjT94GCGEMkLCIonYPghHxO6nO-LRVKwbtSCrrMOgE. Acesso em: 16 dez. 2025.

BLAKE, K.; HUEY, R.; MENEZES, D.; ROOT, J.; TRAN, L.; CHEN, H. Robot navigation using ultra-wideband indoor localization and dead reckoning algorithms. *In*: OPPORTUNITY RESEARCH SCHOLARS SYMPOSIUM (ORSS), 2022. **Anais** [...]. IEEE, 2022. p. 12-15. Doi: 10.1109/ORSS55359.2022.9806034. Disponível em: https://ieeexplore.ieee.org/abstract/document/9806034/?casa_token=CsPxw9V4hUMAAAAA:L-nQOr9cXG22chRJp9Lj2HQSOMFU8GmFhfZD0CSoKp978VP3q5c_ey7ZoLXSx5YrjSueDj0tPDKKB-s. Acesso em: 16 dez. 2025.

CRUZ, O. A. R. **Desenvolvimento de um Robô Móvel Open Source Baseado em ROS 2 para Pesquisa em Robótica Cooperativa**. 98 f. 2022. Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica e de Telecomunicações) – Universidade Federal de Uberlândia, Patos de Minas, 2022. Disponível em: https://repositorio.ufu.br/handle/123456789/34172. Acesso em: 13 dez. 2025.

DE OLIVEIRA JÚNIOR, A.; DUTRA DE ASSUMPÇÃO, H.; QUEIROZ, J.; PIADI, L.; LEITÃO, P.; PÉREZ-PONS, M. E.; PARRA DOMÍNGUEZ, J. Ensino de robótica móvel através da realização de um hackathon em ROS. 2022. In: RAMOS, C.; MARREIROS, G.; PARRA DOMÍNGUEZ, J. (eds.). **Proceedings of the IV Workshop on Disruptive Information and Communication Technologies for Innovation and Digital Transformation**. Salamanca: Ediciones Universidad de Salamanca, 2022. Doi: 10.14201/0AQ031587104. Disponível em: https://gredos.usal.es/handle/10366/149417. Acesso em: 15 dez. 2022.

DUBEY, S.; PRATEEK, M.; SAXENA, M. Robot locomotion – a review. **International Journal of Applied Engineering Research**, v. 10, n. 3, p. 7357-7369, 2015. Disponível em: https://eprints.whiterose.ac.uk/id/eprint/210874/. Acesso em: 16 dez. 2025.

FEATURES AND ARCHITECTURE. **micro.ros.org**. Disponível em: https://micro.ros.org/docs/overview/features/. Acesso em: 11 jan. 2023.

GUTIÉRREZ-FLORES, P. A.; BACHMAYER, R. Concept development of a modular system for marine applications using ROS2 and micro-ROS. *In*: IEEE/OES AUTONOMOUS UNDERWATER VEHICLES SYMPOSIUM (AUV), 2022. **Anais** [...]. IEEE, 2022. p. 1-6. Doi: 10.1109/AUV53081.2022.9965867. Disponível em: https://ieeexplore.ieee.org/abstract/document/9965867/?casa_token=ki0SAgdO1Y4AAAAA:eWfu9Z1A5QVfvo0DyoeJiDQ33u0DzKtOK8Wu-oJCUbyzyMIIFs74EP7_s3MIB5szMIULYudX1FWLWhs. Acesso em: 16 dez. 2025.

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: 15 nov. 2022.

KOUBAA, A. **Robot Operating System (ROS)**: The Complete Reference, *1* vol. Cham: Springer, 2016.

LEMON, M.; ANGLEA, T.; WANG, Y. Experimental study of decentralized robot network coordination. *In*: CHINESE CONTROL AND DECISION CONFERENCE (CCDC), 2019. **Anais** [...]. IEEE, 2019. p. 4863-4867. Doi: 10.1109/CCDC.2019.8832498. Disponível em: https://ieeexplore.ieee.org/abstract/document/8832498/?casa_token=U04rOgAOYV8AAAAA:5QvgSgyhyawlJEjRrdLspJTgXKRz3PU7XW36eSMhvvUP1cOf4HZfSSJmXETw_L59tm6uV_iN7cBqx9o. Acesso em: 16 dez. 2025.

MACENSKI, S.; FOOTE, T.; GERKEY, B.; LALANCETTE, C.; WOODALL, W. Robot Operating System 2: Design, architecture, and uses in the wild. **Science Robotics**, v. 7, n. 66, p. eabm6074, 2022. Doi:

10.1126/scirobotics.abm6074. Disponível em: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074. Acesso em: 16 dez. 2025.

MICRO-ROS – FIWARE. **Micro-ROS – FIWARE**. 2021. Disponível em: https://www.fiware.org/about-us/micro-ros/. Acesso em: 20 jan. 2023.

NATTHARITH, P.; GUZEL, M. S. An indoor mobile robot development: a low-cost platform for robotics research. *In*: INTERNATIONAL ELECTRICAL ENGINEERING CONGRESS (IEECON), 2014. **Anais** [...]. IEEE, 2014. p. 1-4. Doi: 10.1109/iEECON.2014.6925835. Disponível em: https://ieeexplore.ieee.org/document/6925835/?casa_token=rbaw3EBQyasAAAAA:1KKPm52ale9xcqw--jCHaK_XxyQCgk9FqFiQIjbTvO1RH4p-IIHjVGJbt2dpTNc5U8iZzAqiOUS9mVU. Acesso em: 16 dez. 2025.

PEÑA, C. **Arduino IDE**: domina la programación y controla la placa. Buenos Aires: RedUsers, 2020.

PERRON, J. **create_robot**. 2022. Disponível em: https://github.com/AutonomyLab/create_robot. Acesso em: 3 out. 2022.

PLATFORMIO. **What is PlatformIO? PlatformIO latest documentation**. Disponível em: https://docs.platformio.org/en/latest/what-is-platformio.html. Acesso em: 6 fev. 2023.

QUIGLEY, M.; CONLEY, K.; GERKEY, B.; FAUST, J.; FOOTE, T.; LEIBS, J.; NG, A. Y. ROS: an open-source Robot Operating System. *In*: **ICRA Workshop on Open Source Software**, 2009. p. 5.

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

ROBOTICS, I. F. **World Robotics 2020 Report**. 2020. Disponível em: http://reparti.free.fr/robotics2000.pdf. Acesso em: 16 jan. 2023.

ROJAS-FERNÁNDEZ, M.; MÚJICA-VARGAS, D.; MATUZ-CRUZ, M.; LÔPEZ-BORREGUERO, D. Performance comparison of 2D SLAM techniques available in ROS using a differential drive robot. *In*: INTERNATIONAL CONFERENCE ON ELECTRONICS, COMMUNICATIONS AND COMPUTERS (CONIELECOMP), 2018. **Anais** [...]. IEEE, 2018. p. 50-58. Doi: 10.1109/CONIELECOMP.2018.8327175. Disponível em https://ieeexplore.ieee.org/abstract/document/8327175/?casa_token=I9o-LjSyXPcAAAAA:naFSwDcuKwsByLFWwRBrh2wOIM0zBoqDzmH8HhOdK1DiMiyPisdW5CCCYW_1AwJ-SIsYkfjbgpnakpw. Acesso em: 16 dez. 2025.

ROMANO, V.; DUTRA, M. Introdução à robótica industrial. *In*: **Robótica industrial**: aplicação na indústria de manufatura *e de processos*. São Paulo: Edgard Blücher, 2002. p. 1-19.

ROMERO-MARTÍ, D. P.; NÚNEZ-VARELA, J. I.; SOUBERVIELLE-MONTALVO, C.; OROZCO-DE-LA-PAZ, A. Navigation and path planning using reinforcement learning for a Roomba robot. *In*: CONGRESSO MEXICANO DE ROBÓTICA, 18., 2016. **Anais** [...]. IEEE, 2016. p. 1-5. Doi: 10.1109/COMROB.2016.7955160. Disponível em: https://ieeexplore.ieee.org/abstract/document/7955160/. Acesso em: 16 dez. 2025.

ROS INDEX. **ROS Index**. Disponível em: https://index.ros.org/stats/. Acesso em: 15 dez. 2025.

ROS, MICRO. **micro-ROS puts ROS 2 into microcontrollers**. 2022. Disponível em: https://micro.ros.org/. Acesso em: 21 jan. 2022.

ROS. **ROS - Robot Operating System**. Disponível em: https://www.ros.org/. Acesso em: 22 jan. 2023.

RUBIO, F.; VALERO, F.; LLOPIS-ALBERT, C. A review of mobile robots: concepts, methods, theoretical framework, and applications. **International Journal of Advanced Robotic Systems**, v. 16, n. 2, p. 1729881419839596, 2019. Doi: 10.1177/1729881419839596. Disponível em: https://journals.sagepub.com/doi/abs/10.1177/1729881419839596. Acesso em: 16 dez. 2025.

RUIZ, E.; ACUÑA, R.; CERTAD, N.; TERRONES, A.; CABRERA, M. E. Development of a control platform for the mobile robot Roomba using ROS and a Kinect sensor. *In*: LATIN AMERICAN ROBOTICS SYMPOSIUM AND COMPETITION, 2013. **Anais** [...]. IEEE, 2013. p. 55-60. Doi: 10.1109/LARS.2013.57. Disponível em: https://ieeexplore.ieee.org/abstract/document/6693270/. Acesso em: 16 dez. 2025.

SALAMAH, Ummy Gusti; ST., S. **Tutorial Visual Studio Code**. Jakarta: Media Sains Indonesia, 2021. Disponível em: https://www.myedisi.com/medsan/644167/tutorial-visual-studio-code. Acesso em: 16 dez. 2025.

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: 9 out. 2022.

VISUAL STUDIO CODE. **Documentation for Visual Studio Code**. Disponível em: https://code.visualstudio.com/docs. Acesso em: 30 jan. 2023.

WILLIAMS, E.; ACVECEDO, M.; BHARMAL, A. M.; FOMITCHEV, V.; NICHOLS, L.; RICKETTS, K.; INTEGLIA, R. Towards the development of junkyard hacks: networked robotics applications. *In*: 31ST FLORIDA CONFERENCE ON RECENT ADVANCES IN ROBOTICS, 31., 2018. **Anais** [...]. 2018. p. 24-26. Disponível em: https://mae.ucf.edu/fcrar2018/wp-content/uploads/2018/05/5_FPU-Networked-Robotics-Williams-Morris-Integlia-cr.pdf. Acesso em: 16 dez. 2025.