Dissertação de Mestrado

Omnino: plataforma aberta de robô omnidirecional para enxame

de **José Henrick Viana Ramalho**

orientado por
Prof. Dr. Heitor Judiss Savino
Prof. Dr. Erick de Andrade Barboza

Universidade Federal de Alagoas
Instituto de Computação
Maceió, Alagoas
31 de agosto de 2020

UNIVERSIDADE FEDERAL DE ALAGOAS
Instituto de Computação

OMNINO: PLATAFORMA ABERTA DE ROBÔ OMNIDIRECIONAL PARA ENXAME

Dissertação de Mestrado submitida ao Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do grau de Mestre em Informática.

José Henrick Viana Ramalho

Orientador: Prof. Dr. Heitor Judiss Savino
Coorientador: Prof. Dr. Erick de Andrade Barboza

Banca Avaliadora:

Ícaro Bezerra Queiroz de Araújo	Prof. Dr., IC-UFAL
Armando Alves Neto	Prof. Dr., DELT-UFMG

Maceió, Alagoas
31 de agosto de 2020

UNIVERSIDADE FEDERAL DE ALAGOAS
Instituto de Computação

OMNINO: PLATAFORMA ABERTA DE ROBÔ OMNIDIRECIONAL PARA ENXAME

Dissertação de Mestrado submitida ao Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovado em 31 de agosto de 2020:

Heitor Judiss Savino,
Prof. Dr., Orientador

Erick de Andrade Barboza,
Prof. Dr., Coorientador

Ícaro Bezerra Queiroz de Araújo,
Prof. Dr., IC-UFAL

Armando Alves Neto,
Prof. Dr., DELT-UFMG

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

R165o Ramalho, José Henrick Viana.
Omnino : plataforma aberta de robô omnidirecional para exame / José Henrick Viana Ramalho. – 2020.
63 f. : il.

Orientador: Heitor Judiss Savino.
Co-orientador: Erick de Andrade Barboza.
Dissertação (mestrado em Informática) - Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2020.

Bibliografia: f. 60-63.

1. Educação em robótica. 2. Robôs - Baixo custo. 3. Simulação (Computadores). I. Título.

CDU: 004.896

Dedicatória

Dedico este trabalho a minha família.

Minha esposa Aline, meu filho Vitor, minha mãe Wirla e meus irmãos Ernani e Júlio.

Agradecimentos

Ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal de Alagoas - UFAL que disponibilizou a oportunidade de alcançar novos aprendizados.

Ao meu orientador Heitor Judiss Savino e coorientador Erick de Andrade Barbosa, que me guiaram no desenvolvimento deste trabalho com atenção, respeito e compreensão, sem vocês nada disso seria possível, vocês são especiais.

E a todos aqueles que me deram incentivo e apoio durante esta jornada.

Maceió, 31 de Julho de 2020

As coisas que fazemos sem luxuria ou ambições são as mais puras ações que podemos realizar.

Alan Moore

Resumo

Este trabalho apresenta uma plataforma aberta de um robô para atividades de pesquisa e educacionais. A plataforma é definida como um robô omnidirecional, que é um dos primeiros passos no currículo de robótica quando se deseja projetar um controlador, dado que este não impõe restrições ao movimento num plano. Os robôs são projetados para serem de baixo custo e com acesso aberto ao *software e hardware*. A maior parte do projeto mecânico também é fácil de imprimir em impressoras 3D comumente usadas. O robô é integrado com a plataforma *ROS - Robot Operating Systems*, amplamente utilizada na comunidade de robótica, e o modelo de simulação em ROS-Gazebo é fornecido para permitir uma passagem fácil do ambiente de simulação ao ambiente experimental. Experimentos reais com os robôs propostos são mostrados, assim como a plataforma de simulação com múltiplos robôs.

***Palavras-chave*: *Robótica educacional*; *robôs de baixo custo*; *robôs omnidirecionais*; *projeto de robôs*; *simulação de robôs*.**

Abstract

This work presents an open robotic platform for research and educational activities. The platform is defined as an omnidirectional robot, which is one of the first steps in the robotics curriculum when a controller is to be designed, as it does not impose restrictions for moving on a plane. These robots are designed to be low cost and have open access to software and hardware. Most of the mechanical design is also easy to print on commonly used 3D printers. The robot is integrated with ROS - Robot Operating Systems platform, widely used in the robotics community, and the ROS-Gazebo simulation model is provided to allow an easy transition from the simulation environment to the experimental environment. Real experiments with the proposed robots are shown, as well as the simulation platform with multiple robots.

Keywords*: *Educational robotics*; *low cost robots*; *omnidirectional robots*; *robot design*; *robot simulation.

Lista de Figuras

Lista de Abreviaturas

ASCII American Standard Code for Information Interchange.

AUV Autonomous Underwather Vehicle.

CC Corrente Contínua.

FCEM Força Contra Eletromotriz.

FDM Fused Deposition Modeling.

FPS Frame Por Segundo.

GPIO General Purpose Input/Output.

IC Instituto de Computação.

I/O Input/Output.

IR Infravermelho.

LED Light Emitting Diode.

PID Proporcional, Integral, Derivativo.

PLA Polylactic Acid.

PPR Pulso Por Revolução.

PWM Pulse Width Modulation.

RAM Random Access Memory.

RGB Red, Green, Blue.

RGB-D Red, Green, Blue - Depth.

ROS Robot Operating Systems.

RPM Revoluções por Minuto.

SLAM Simultaneous Localization and Mapping.

UART Universal Asynchrounous Receiver/Transmiter.

UAV Unmanned Aerial Vehicle.

UFAL Universidade Federal de Alagoas.

UGV Unmanned Ground Vehicle.

USB Universal Serial Bus.

WLAN Wireless Local Area Network

Sumário

Capítulo 1

Introdução

Os robôs estão se tornando um componente integral de nossa sociedade atual e tem conquistado grande potencial em ser utilizado como tecnologia educacional e ferramenta de pesquisa em ambientes escolares e universitários. Independente da natureza da atividade de aprendizagem, um robô pode assumir uma série de funções neste campo de desenvolvimento do aprendizado. No trabalho Han et al. (2008), foi demonstrado que estudantes tiveram um maior desempenho em exames de pós-aprendizagem e tiveram maior interesse no aprendizado quando robôs foram utilizados como ferramenta didática de ensino e pesquisa.

1.1 Justificativa

Nos anos de 1960, os primeiros robôs podiam ser vistos em pouquíssimas indústrias existentes em países muito desenvolvidos, como Japão e Estados Unidos. No entanto, a redução de custos relacionadas aos avanços da informática e microeletrônica, propiciaram a geração de grandes benefícios para a implementação da automação robótica nas indústrias, tornando-os mais acessíveis.

Na indústria, a automação possibilita um grande incremento na produtividade e na qualidade do produto final, uniformizando a produção e reduzindo os custos causados por desperdícios e perdas. Os fatores impeditivos da total implantação da automação industrial estão envolvidas em seu custo de implementação e os impactos sociais que a demissão em massa de empregados pode causar à sociedade, pois, muitas vezes, apenas um único robô pode substituir a mão de obra de dezenas de trabalhadores. Mas quando se fala em desemprego de trabalhadores, os menos qualificados é que acabam sendo os maiores prejudicados nesta situação. É necessário salientar que estes empregos sofrem uma mutação e o mercado de trabalho passa a absorver mão de obra mais preparada para os desafios que estas tecnologias impõem, nos quais o trabalhador deve ter capacitação técnico/educacional para lidar com problemas complexos e trazer novas soluções.

Em 1982, o estudioso inglês Tom Stonier (Stonier and Thornton, 1982) ressalta a

necessidade dos governos investirem massivamente na educação gratuita em todos os níveis, aumentando a capacidade dos indivíduos que estão inseridos em uma sociedade na qual robótica e automação são parte do cotidiano.

A robótica já tem se tornado conteúdo em instituições de ensino públicas e privadas. Kits educacionais comerciais como o Lego Mindstorm e o Modelix, são plataformas que oferecem uma introdução aos conceitos da robótica. No entanto, são bastante limitados por seu hardware e software fechados, o que muitas vezes reduz a capacidade de aplicação de conceitos mais avançados nesta área de estudo, além disso o alto custo de aquisição dessas plataforma contribuem para este cenário.

Em ambientes de ensino técnico e superior, estudantes e pesquisadores da área de robótica desenvolvem projetos mais sofisticados. As plataformas anteriormente citadas, por serem limitadas acabam por não propiciar uma abordagem apropriada para suas pesquisas. Para isso, a Festo oferece, por exemplo, uma plataforma denominada Robotino, um robô omnidirecional com tecnologias de visão computacional e sensoriamento embarcados, que possibilitam a realização das mais variadas tarefas de automação, mas que infelizmente, seu custo elevado torna impeditiva a aplicação de tal plataforma em instituições e pesquisas que sofrem de recursos limitados, principalmente em projetos que envolvem enxames de robôs e robótica cooperativa.

Os avanços recentes da microeletrônica e dos sistemas de comunicação têm possibilitado a construção de agentes cada vez mais compactos, porém com alta capacidade computacional. Além disso, os avanços na área de computação, especificamente em sistemas distribuídos, tem incentivado o uso de múltiplos agentes na realização de tarefas de forma cooperativa. Estes agentes, chamados genericamente de enxames de robôs, trazem vantagens à execução de tarefas como robustez, expansibilidade e flexibilidade. A terminologia “enxame” baseia-se na ordem organizacional-biológica das abelhas e formigas que agem coletivamente para construção de colônias, busca de alimentos e defesa.

Constitui-se, como principal desafio da área, o desenvolvimento de sistemas de agentes que sejam escaláveis, de forma que grupos possam ser: controláveis de forma eficiente; capazes de realizar tarefas com informações locais; adaptáveis à adição ou remoção de membros; e que se mantenham robustos, mesmo diante de falhas individuais de seus agentes.

Neste sentido, a Indústria atenta-se para as pesquisas que buscam minimizar estes problemas. Conforme Cao et al. (1997), tem-se desenvolvido uma série de aplicações em que um enxame de agentes autônomos pode controlar o tráfego de veículos autônomos Marcolino and Chaimowicz (2009), limpeza de áreas contaminadas Altshuler et al. (2006) e modelar estruturas Wei et al. (2010).

O transporte cooperativo de cargas, por exemplo, é uma aplicação da robótica que pode ser tratada como um grupo de robôs móveis mantendo uma formação em volta do objeto que se deseja deslocar. Essa abordagem subdivide-se em: Prensável ou Não-

Prensável.

A abordagem prensável considera a ligação entre os robôs do enxame e a carga como uma ligação rígida. A manipulação e o controle do objeto podem ser aplicados de modo que o enxame seja controlado por um algoritmo baseado em seguir o líder. Neste modo, um dos agentes do enxame determina, ou conhece, o sentido e rotação da carga a ser transportada e todos os outros agentes mantêm a formação geométrica realizando mudanças em sua localização e ângulo. Um exemplo deste tipo de aplicação pode ser visto no trabalho de Groß et al. (2006). Em abordagens não-prensáveis, os robôs apenas se posicionam em volta do objeto que se deseja deslocar, empurrando-o ou rolando-o (Chen et al., 2015). A manutenção de formação, como aquela que se deseja atingir em torno de um determinado objeto visando a manipulação cooperativa, será a aplicação escolhida para demonstrar o comportamento em enxame deste trabalho.

No entanto, o desenvolvimento de pesquisa nestas áreas ainda encontra entraves pela falta de recursos que podem afetar regiões ainda em desenvolvimento. Os robôs móveis encontrados no laboratório de robótica do Instituto de Computação da UFAL são, por exemplo, dois QBots da Quanser, custando USD 8000. Com isto, o foco principal deste trabalho de dissertação é a utilização de ferramentas e tecnologias acessíveis para o desenvolvimentos e confecção de robôs móveis holonômicos, para fins de aplicação de métodos de controle distribuído e enxames de robôs conforme demonstrados na Figura 1.1, assim como para compor o currículo de cursos de robótica tanto em nível técnico como superior.

O software apresentado é integrado à plataforma de middleware ROS (Robotics Operating System), gerando uma nova plataforma de desenvolvimento de baixo custo para futuros trabalhos desenvolvidos dentro do grupo de pesquisa e fácil de ser replicado por parceiros institucionais.

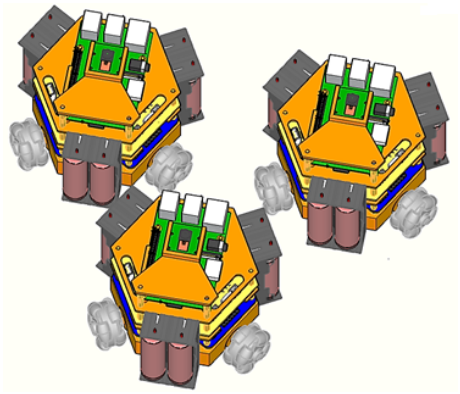

Figura 1.1: Enxame de robôs colaborativos controle de formação.

1.2 Objetivos geral e específicos

1.2.1 Geral

O objetivo geral deste trabalho é produzir uma plataforma de robô holonômico visando aplicações de enxame de robôs, disponibilizando de forma aberta todo o *hardware e software* necessários para sua construção e utilização com fins educacionais. O sistema será testado numa aplicação de controle de formação utilizando técnica de controle distribuído baseado em consenso.

1.2.2 Específicos

Para que o objetivo geral seja atingido, foram definidos os seguintes objetivos específicos:

- Projeto mecânico e modelagem cinemática do robô com três rodas omnidirecionais;
- Desenvolvimento do chassi e disponibilização aberta para impressoras 3D;
- Projeto e sintonia de controladores PID com microcontroladores dedicados para o controle de velocidade de cada roda;
- Seleção e desenvolvimento do *software* de localização utilizando câmeras e marcadores em ROS;
- Projeto de *software* e seleção do dispositivo de controle de alto nível levando em consideração sua capacidade de processamento e custo;
- Realização de experimentos aplicando técnicas de controle distribuído baseadas em consenso.

1.3 Relevância da proposta

Realizar a construção de uma plataforma robótica omnidirecional, visando experimentação prática de algoritmos desenvolvido em pesquisa de robótica móvel. Sabe-se que, geralmente, simulações não refletem as circunstancias em que o sistema será inserido no mundo real, incluindo distúrbios gerados pela própria característica construtiva de sistemas reais como folga em engrenagens, atrito e ondulações na superfície de contato. No trabalho de Bayındır (2016), o autor destaca a grande falta de trabalhos que utilizem enxames de robôs no mundo real e que sua aplicação traria novas ideias de como os fatores externos afetariam a operação de um enxame e estimularia novas pesquisas. Estão disponíveis no mercado uma variedade de kits de robótica móvel mas que, para a realidade da maioria dos pesquisadores e instituições, são inviáveis financeiramente. Além disto, estes

sistemas são geralmente fechados, o que dificulta sua aplicação em modelos desenvolvidos em pesquisa.

Percebendo esta precariedade, este trabalho tem o intuito de desenvolver um plataforma omnidirecional de hardware e software abertos validando a utilização de um algoritmo de controle de formação baseado em consenso para enxames de robôs, para que seja possível aos pesquisadores terem acesso a um sistema de capacidade e desempenho semelhantes aos demais kits de robótica para pesquisa e experimentação de robôs móveis.

1.4 Metodologia

O desenvolvimento do trabalho compreende:

- levantamento bibliográfico e estudo dos modelos de cinemática de robôs holonômicos de 3 eixos;
- modelagem do robô omnidirecional proposto;
- desenvolvimento e impressão dos chassis em impressoras 3D;
- projeto e sintonia de controladores PID de velocidade para cada motor;
- seleção e desenvolvimento de dispositivo de localização para os agentes do enxame;
- seleção do sistema a ser embarcado nos robôs móveis, sendo estes microcontroladores com núcleo ARM 32 Bits ou PC cards como o Rasperry Pi;
- Integrar o sistema ao ROS;
- execução do controlador em modelo real, coletando e aplicando calibrações ao sistema;
- apresentação dos resultados coletados.

Capítulo 2

Fundamentação teórica

Este capítulo apresenta a fundamentação teórica necessária para o desenvolvimento deste trabalho. Os equipamentos utilizados na plataforma Omnino e seu funcionamento são explicados em detalhes. Os softwares desenvolvidos para a pesquisa são igualmente apresentados e descritos neste capítulo. Fundamentos como modelagem cinemática, enxames de robôs e métodos de localização também são expostos aqui.

2.1 Robôs Móveis

Graças à evolução da microeletrônica, sistemas embarcados tiveram seus custos reduzidos e suas capacidades de processamento aumentadas. Hoje, sensores lasers, câmeras e atuadores têm se tornado mais eficientes e acessíveis, sendo facilmente adicionados em projetos de laboratório de robótica. Além disso, têm adquirido maior precisão, robustez e menor consumo de energia, tornando possível a implementação destes dispositivos em sistemas com fornecimento de energia limitados e desconectados das redes convencionais de energia. Tudo isso promove o crescente uso de agentes autônomos móveis, onde uma maior capacidade de geração de informações que um robô pode coletar do ambiente, quando interpretados por seus algoritmos de controle, possibilitam a realização de tarefas complexas sem a supervisão humana.

A criação de agentes capazes de interagir com o ambiente e seres humanos de forma autônoma tem sido objetivo de pesquisadores e desenvolvedores na área da robótica. As pesquisas têm colaborado no ambiente humano desde a execução de tarefas mais simples, como a limpeza de ambientes residenciais com aspiradores de pó inteligentes, até a exploração de lugares inóspitos como a Lua e Marte.

Em robótica móvel, os robôs devem ter a capacidade de interagir com o ambiente e tomar decisões precisas para que obtenham êxito em suas tarefas. Estes robôs móveis podem ser classificados conforme o ambiente em que são inseridos com diferentes nomenclaturas:

- Robôs subaquáticos: *Autonomous Underwather Vehicle* (AUV);

Entre os robôs subaquáticos pode-se destacar o REMUS, um robô capaz de executar tarefas como: levantamento oceanográfico; mapeamento de áreas profundas; pesquisa de turbulência; e levantamento hidrográfico. O REMUS (Figura 2.1) foi o agente robótico responsável pela localização dos destroços do avião da Air France Flight 447, e todo seu trabalho de escaneamento e localização pode ser acompanhado no artigo de Purcell et al. (2011).

Figura 2.1: AUV REMUS 6000

- Robôs aéreos: *Unmanned Aerial Vehicle* (UAV); No trabalho realizado por Neumann et al. (2017), é apresentado um agente robótico aéreo, chamado REGAS (Figura 2.2) capaz de detectar e localizar fontes de vazamento de gás, propiciando uma maior segurança para seres humanos, reduzindo os riscos de exposição e acidentes.

Figura 2.2: UAV REGAS

- Robôs terrestres: *Unmanned Ground Vehicle* (UGV);

  Dentre as subcategorias de robôs móveis terrestres com rodas se destacam, em trabalhos de pesquisa e utilização comercial, os seguintes tipos:

  – Robôs móveis diferenciais (Figura 2.3): definidos como não-holonômicos por terem restrições de movimento geradas por sua configuração construtiva. No caso de robôs móveis com acionamento diferencial, a restrição é imposta pela impossibilidade do robô locomover-se em qualquer direção sem a alteração de seu ângulo.

  – Robôs móveis omnidirecionais (Figura 2.4): este tipo de robô possui a vantagem de ter os movimentos de translação e rotação desacoplados, isto é, pode

se mover em qualquer direção sem a necessidade de se reorientar. Sendo assim, robôs omnidirecionais são considerados robôs holonômicos, pois, possuem capacidade de execução de tarefas sem restrições cinemáticas.

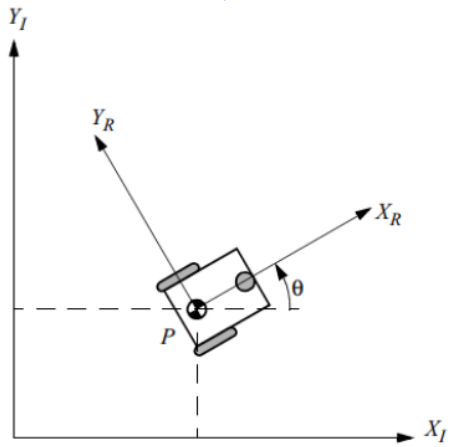

Figura 2.3: Robô móvel diferencial (Siegwart et al., 2011)

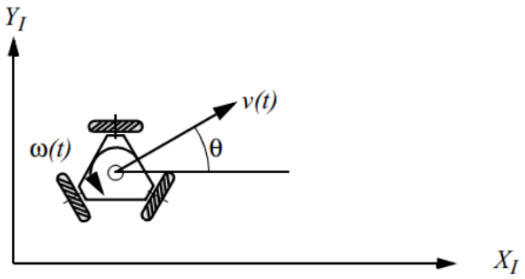

Figura 2.4: Robô móvel omnidirecional (Siegwart et al., 2011)

Diversos trabalhos foram realizados na modelagem de robôs omnidirecionais, proporcionando estudos de caso vastos na área de robótica móvel. No trabalho de Liu et al. (2003), foi desenvolvido um controle de linearização de trajetória, tendo como base os modelos cinemático e dinâmico de um robô omnidirecional de 3 rodas, demonstrando estabilidade e robustez para diferentes modelos de trajetórias. No trabalho de Li and Zell (2009), é proposto o método de Inverse Input-Output para realizar a modelagem cinemática de um robô omnidirecional de 3 rodas, garantindo estabilidade no controle de trajetória do robô em trajetórias de circuito fechado.

Na atualidade, os modelos omnidirecionais têm sido utilizados para empregar maior manobrabilidade em sistemas autônomos móveis. Com modificações mecânicas relativamente simples em seus projetos, o desenvolvedor promove a holonomia em agentes autônomos.

No trabalho de Doroftei et al. (2007) são relatadas as vantagens que plataformas omnidirecionais tem em relação a modelos de deslocamento como Ackerman e diferencial, pois são capazes de se movimentar e virar-se em locais estreitos, deslocando-se lateralmente e

seguindo trajetórias complexas. Esses robôs podem executar tarefas em ambientes estáticos e dinâmicos com a vantagem de realizar menos movimentos para alcançar posições desejadas quando comparados com robôs do tipo Ackerman e diferencial.

2.2 Roda Omnidirecional

As rodas Suecas, ou omnidirecionais, mostradas na Figura 2.5, foram desenvolvidas por Bengt Ilon em 1970. A rotação da roda é feita pelo eixo principal, mas pode se mover com pouco atrito em qualquer direção devido a pequenos roletes que constituem toda a estrutura. O uso de rodas suecas não impõem restrições cinemáticas ao robô, pois permite ao robô se locomover em todas as direções devido aos graus de liberdade internos da roda (Dudek and Jenkin, 2010).

De acordo com Siegwart et al. (2011), a roda Sueca e a roda esférica são modelos menos restritivos do que as rodas padrões convencionais. A roda Sueca funciona como uma roda normal, mas provém baixa resistência na direção perpendicular à direção convencional. Os roletes conectados ao redor da circunferência da roda são passivos e o eixo principal da roda serve como a única junta atuada ativamente. A vantagem principal deste modelo é que, apesar da rotação da roda ser atuada apenas ao longo do único eixo principal, a roda pode se mover cineticamente com fricção muito baixa ao longo de várias trajetórias possíveis, e não simplesmente para frente e para trás.

Figura 2.5: Rodas Suecas ou omnidirecionais

As desvantagens deste modelo de roda omnidirecionais podem ser causadas por seus roletes, que podem causar derrapagens, prejudicando a odometria.

2.3 Motor de Corrente Contínua

Motores de corrente contínua com escovas convertem a corrente aplicada em seu indutor em movimento mecânico no eixo de saída. Motores com escovas requerem um sinal de *feedback* para que tenham uma operação estável. Eles devem ser usados em um circuito de controle de malha fechada, o que torna seu sistema de controle mais complexo do que, por exemplo, no uso de motores de passo. No entanto, motores com escovas tem torque

mais alto e pode alcançar velocidades maiores do que motores de passo Kurfess (2018). O modelo visto na Figura 2.6, foi empregado para a locomoção da plataforma Omnino, suas características são: tensão nominal 9Vcc, corrente de trabalho 130mA, *encoder* 200PPR e 90RPM.

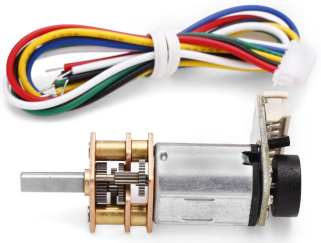

Figura 2.6: Motor CC Omnino

O *encoder* fixado neste motor propicia o fechamento da malha de controle para a construção do controle de velocidade.

2.4 Ponte H

Para controlar um motor não é possível apenas ligar o fio do motor em um pino de I/O do microcontrolador. Um motor drena uma corrente muito maior do que qualquer microcontrolador consegue fornecer em um pino de saída. O motor utilizado neste projeto, tem uma corrente nominal de 130mA, valor muito acima do que a porta do microcontrolador consegue fornecer, que é no máximo 25mA. Caso essa ligação seja feita diretamente, na tentativa de acionar o motor a corrente drenada poderá causar a queima do pino de I/O ou até mesmo do próprio microcontrolador, o controle de condução da corrente através do motor para determinar seu sentido de rotação também deve ser realizado, para isso, um circuito externo ao sistema de controle será necessário, para tal tarefa, o uso de uma ponte H é a solução mais empregada.

A ponte H é um circuito composto por transistores, funcionando como chave, que se acionados na base por um sinal digital permite a passagem de corrente entre os terminais de coletor e emissor e, consequentemente, a passagem de corrente elétrica pelo motor.

O esquema representativo do circuito de uma ponte H está mostrado na Figura 2.7. Em (1), todos os transistores estão abertos e mantém o motor parado. Em (2), dois transistores permitem a passagem de corrente pelo motor, da esquerda para a direita, fazendo o motor rotacionar em um sentido. Em (3), acontece o inverso de (2), e logo o motor é acionado no sentido contrário.

O circuito integrado L9110 foi selecionado para o controle de sentido do motor. Ele apresenta uma tensão de trabalho que varia de 2,5Vcc até 12Vcc, 800mA de corrente

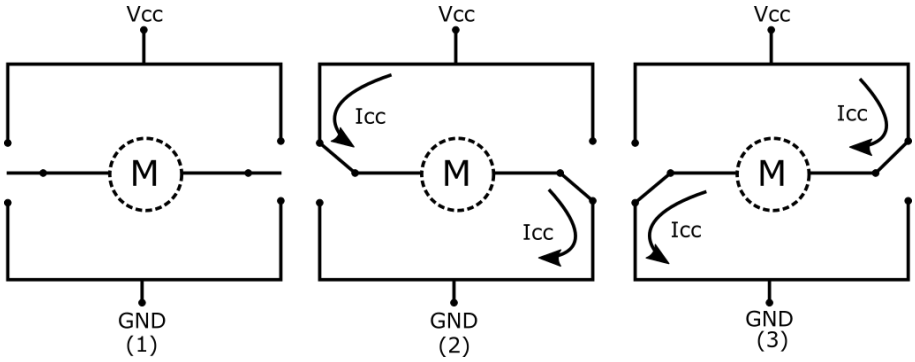

Figura 2.7: Ponte H

nominal e circuito de fácil implementação.

2.5 Plataforma Holonômica Omnino

A plataforma de robô proposta, Omnino, é baseada no modelo proposto por Siegwart et al. (2011), conforme representado na Figura 2.8. O desenvolvimento do Omnino foi projetado em *software* de desenho 3D e confeccionado em impressora 3D, trazendo maior acessibilidade e também precisão na sua construção.

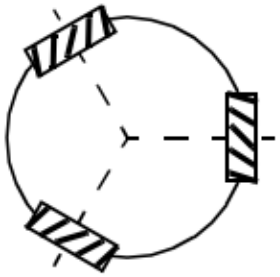

Figura 2.8: Modelo Holonômico de três Rodas Siegwart et al. (2011)

Em robôs omnidirecionais, a exatidão em que os eixos são distribuídos influencia diretamente no desempenho do sistema de locomoção e na modelagem de seu sistema. Por isso, a escolha de sistemas mais sofisticados como a impressão 3D tem se tornado uma opção bastante recorrente em desenvolvimento de protótipos robóticos, como pode ser visto nos trabalhos de Zou et al. (2018) e Tavakoli et al. (2013). Na Figura 2.9, é mostrado o modelo base do chassi desenvolvido para o robô omnidirecional Omnino.

O robô omnidirecional Omnino utiliza três rodas suecas distribuídas uniformemente em ângulos de 120°. A cada roda são ligados motores de corrente contínua com redução e *encoders* acoplados, para que seja possível medir e controlar a velocidade empregada em cada uma de suas rodas.

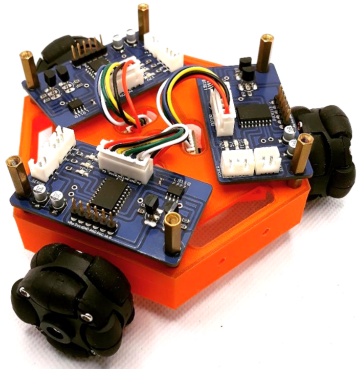

Figura 2.9: Modelo base Omnino

2.6 Modelo Cinemático

A cinemática do robô lida com a configuração dos robôs em seu espaço de trabalho e as relações entre seus parâmetros construtivos e as restrições impostas em suas trajetórias. As equações cinemáticas dependem da estrutura geométrica do agente. Em um robô móvel pode-se ter diversas rodas com ou sem restrições em seu movimento.

Na análise do movimento deste robô, é importante notar que cada roda tem uma contribuição direta com o movimento geral do robô. Considerando o robô omnidirecional da Figura 2.10, este robô possui três rodas suecas de 90°, dispostas radialmente a cada 120°, com rolos perpendiculares em cada roda. Na Figura 2.10, Vel1, Vel2 e Vel3 representam as velocidades lineares exercidas por cada uma das rodas, w representa a velocidade de rotação em radianos/segundo e R é a distancia do centro do robô ao centro das rodas.

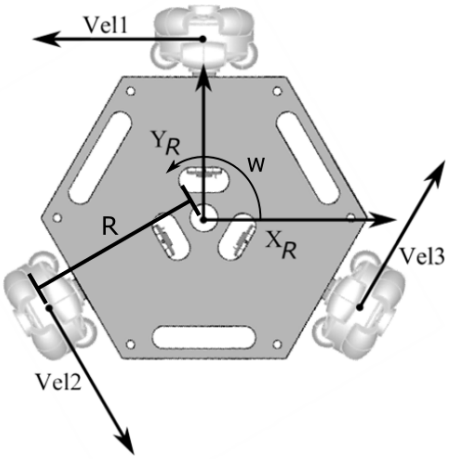

Figura 2.10: Robô móvel omnidirecional

Para a modelagem cinemática do sistema, os vetores velocidade foram deslocados para a origem geométrica do robô. Sabendo que a soma vetorial de cada uma das velocidades das rodas deve resultar em um vetor que aponte para o local desejado, decompondo estes vetores em suas resultantes nos eixos X_R e Y_R. Este procedimento pode ser acompanhado

na Figura 2.11. Desta representação, podem ser obtidas as Equações 2.1 que representam a cinemática direta do agente.

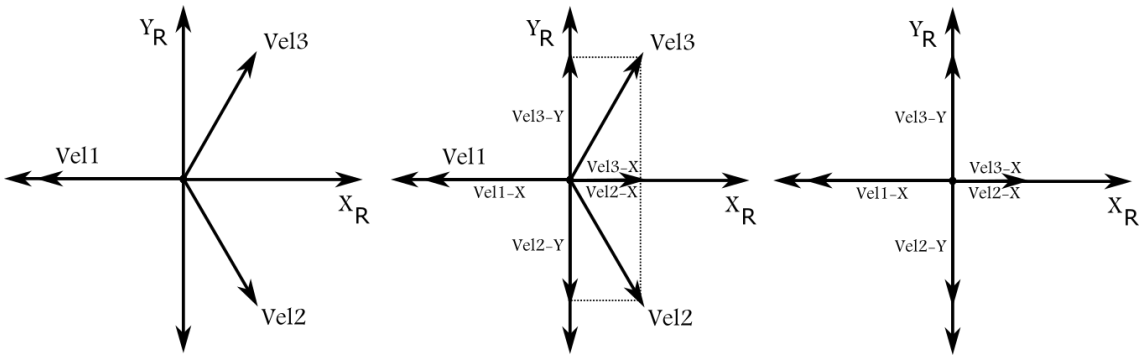

Figura 2.11: Decomposição vetorial

$$\begin{bmatrix} Vx \\ Vy \\ \omega \end{bmatrix} = \begin{bmatrix} \text{Vel1}\cos(\pi) & \text{Vel2}\cos(5\pi/3) & \text{Vel3}\cos(\pi/3) \\ \text{Vel1}\sin(\pi) & \text{Vel2}\sin(5\pi/3) & \text{Vel3}\sin(\pi/3) \\ \text{Vel1}/R & \text{Vel2}/R & \text{Vel3}/R \end{bmatrix} \tag{2.1}$$

Colocando as velocidades da matriz em evidencia como na Equação 2.2, é possível verificar que sabendo-se as velocidades empregadas em cada uma das rodas torna-se praticável determinar o vetor direção que o robô irá se deslocar.

$$\begin{bmatrix} Vx \\ Vy \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\pi) & \cos(5\pi/3) & \cos(\pi/3) \\ \sin(\pi) & \sin(5\pi/3) & \sin(\pi/3) \\ R & R & R \end{bmatrix} \cdot \begin{bmatrix} \text{Vel1} \\ \text{Vel2} \\ \text{Vel3} \end{bmatrix} \tag{2.2}$$

A determinação da posição e orientação final do robô se torna possível por cinemática direta quando as variáveis de velocidade em cada roda, até o instante atual, são conhecidas. Isso quer dizer que as variáveis de entrada na equação são as velocidades, mas se for necessário colocar o robô em uma posição e orientação desejadas, é necessário saber quais devem ser as velocidades a serem empregadas em cada uma de suas rodas. Para que o agente seja capaz de realizar esta tarefa o controlador deve ter como variável de entrada a posição final.

A fim obter este resultado é preciso executar a cinemática inversa da Equação 2.2. Em aplicações reais, a equação de cinemática inversa tem caráter muito importante, já que o controlador atua diretamente nas velocidades das rodas. A Equação 2.3 representa o modelo cinemático inverso do robô Omnino.

$$\begin{bmatrix} \text{Vel1} \\ \text{Vel2} \\ \text{Vel3} \end{bmatrix} = \begin{bmatrix} -\frac{2}{3} & 0 & \frac{1}{3R} \\ \frac{1}{3} & -\frac{\sqrt{3}}{3} & \frac{1}{3R} \\ \frac{1}{3} & \frac{\sqrt{3}}{3} & \frac{1}{3R} \end{bmatrix} \begin{bmatrix} Vx \\ Vy \\ \omega \end{bmatrix} \tag{2.3}$$

Com a Equação 2.3 é possível analisar o movimento do robô multiplicando a equação

da cinemática inversa pela matriz posição. Exemplo: deseja-se deslocar o robô ao longo do eixo Y, aplicando a matriz de movimento ao modelo cinemático do robô, Equação 2.4, será obtido como resposta do modelo para as velocidades em cada uma das rodas do robô, Equação 2.5.

$$\begin{bmatrix} \text{Vel1} \\ \text{Vel2} \\ \text{Vel3} \end{bmatrix} = \begin{bmatrix} -\frac{2}{3} & 0 & \frac{1}{3R} \\ \frac{1}{3} & -\frac{\sqrt{3}}{3} & \frac{1}{3R} \\ \frac{1}{3} & \frac{\sqrt{3}}{3} & \frac{1}{3R} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} \text{Vel1} \\ \text{Vel2} \\ \text{Vel3} \end{bmatrix} = \begin{bmatrix} 0 \\ -0{,}58 \\ 0{,}58 \end{bmatrix} \tag{2.5}$$

O motor com velocidade Vel1 tem seu valor igual a zero e permanecerá parado. O motor relacionado à velocidade Vel2 tem o valor de -0,58, consequentemente terá seu sentido de rotação invertido. E o motor com Vel3, com valor de 0,58, mantém a mesma direção com a mesma velocidade do motor Vel2, fazendo com que o robô se desloque sobre o eixo Y. Como os rolamentos da roda omnidirecional são livres no ângulo de 90°, a roda do Motor Vel1 não gera restrições ao deslocamento do robô. Na Figura 2.12, pode-se notar os vetores velocidade das rodas e o vetor deslocamento resultante.

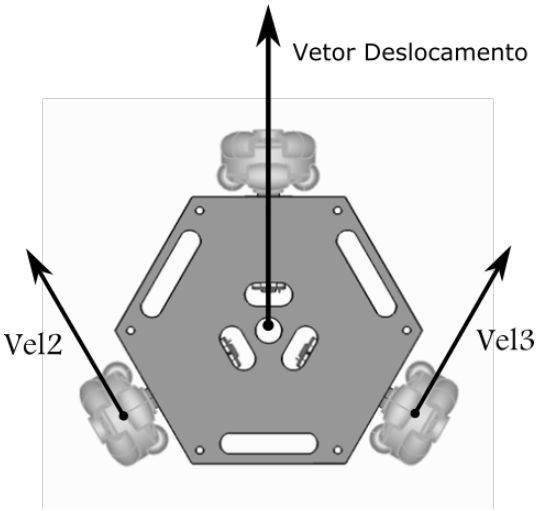

Figura 2.12: Deslocamento resultante de um exemplo

2.7 Enxame de robôs

Um enxame de robôs é um grupo de três ou mais robôs que executam tarefas de forma cooperativa com supervisão limitada ou inexistente de operadores humanos Arnold et al. (2019). Além disso, comportamento inteligente pode emergir de interações limitadas de robôs simples (Brooks et al., 1999). O conceito de enxame de robôs se inspirou na ca-

pacidade de auto-organização de formigas, que faz uso de suas mandíbulas e patas para se conectarem umas as outras para a criação de barcos e pontes (Anderson et al., 2002), no modo como os pássaros voam em formação, e em como cardumes de peixes se agrupam de forma organizada para evitar predadores. Embora a maior parte dos sistemas que aplicam robôs móveis utilize um único robô para a realização de tarefas, para muitos desenvolvedores a distribuição de tarefas em um número maior de agentes, de construção simples, provou ter propriedades muito interessantes como robustez, flexibilidade e capacidade de resolver problemas complexos, que envolvem paralelismo e auto-organização. Isso tem levado a um crescente desenvolvimento da área, com diversos desafios a serem solucionados.

No trabalho de Ardakani et al. (2017), foi desenvolvida uma nova abordagem para que um enxame de robôs desloque uma placa elástica em um ambiente desconhecido, realizando somente as leituras de atrito entre os robôs e as placas. O sistema de localização utilizado pelos robôs eram baseados em *encoders*, que se mostraram insatisfatórios, sendo substituídos por um sistema de processamento de imagens. Isto diminuiu a estimativa de erro de localização dos agentes do enxame em comparação aos sensores odométricos embarcados. Um detalhe do trabalho de Ardakani et al. (2017) é que todo o processamento dos controladores é feito de forma centralizada. Outros trabalhos com a temática de transporte de cargas por enxames também podem foram descritos por Chen et al. (2015) e Kube and Bonabeau (2000), onde são abordadas estratégias de controle e simulação para este tipo de aplicação.

2.7.1 Plataformas de robôs omnidirecionais e para enxame

Existe uma grande variedade de plataformas robóticas voltadas para o desenvolvimento de pesquisas em enxames. As plataformas Alice Caprari and Siegwart (2005), E-Puck Mondada et al. (2009) e Jasmine Kernbach et al. (2009), vistas respectivamente na Figura 2.13, são exemplos de trabalhos que foram realizados no desenvolvimento de micro-robôs voltados para essa aplicação. Todos eles apresentam locomoção diferencial e são equipados com sensores de proximidade e receptores infravermelhos (IR) usados para comunicação com vizinhos. Seus núcleos de processamento e interfaces de comunicação são baseados em microcontroladores de 8bits e 32bits, o que limita seu poder computacional tornando-os sistemas dependentes de agentes externos que realizem o processamento mais ”pesado”do enxame, como por exemplo, a localização por visão computacional e o controle de movimento dos agentes.

Entre os trabalhos mais recentes com enxames de robôs de pequena dimensão, o Zooids desenvolvido por Le Goc et al. (2016), se destaca por ter 2,6cm de diâmetro com duas rodas não colineares. São robôs rápidos que chegam a velocidades de 74cm/s. Zooids são equipados com sensores de toque para interação humano-robô e utiliza uma câmera

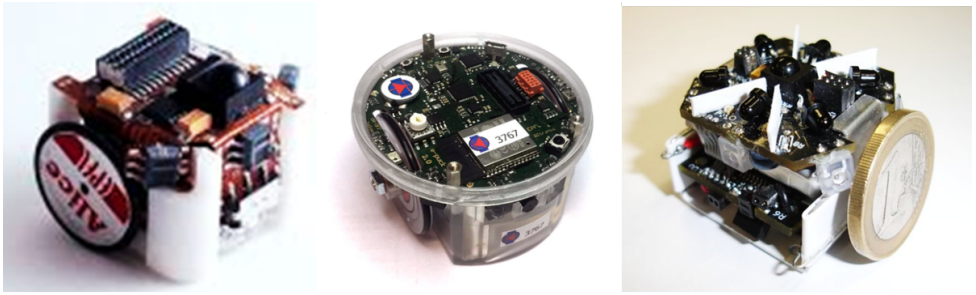

Figura 2.13: Plataformas para Enxame, Alice, E-Puck e Jasmine, respectivamente

externa para determinação da posição de cada um dos agentes do enxame.

O trabalho desenvolvido por Rezeck et al. (2017), a plataforma HeRo, Figura 2.14, apresenta uma nova plataforma para enxame de baixo custo, em torno de $18 dólares, fácil de montar usando componentes prontos para uso e integrada ao ROS. A plataforma robótica é totalmente aberta, composta por corpo impresso em 3D e software open source.

Figura 2.14: Plataforma HeRo

As plataformas citadas utilizam softwares dedicados para o controle do enxame ou mesmo controles remotos, o que limita suas possibilidades de replicação e modificação em outros ambientes de pesquisa de forma aberta, a necessidade de processamento externo ao enxame também é uma característica restritiva que estas plataformas apresentam, pois a falha do sistema de processamento central irá comprometer o funcionamento de todos os agentes do enxame, sendo este um ponto crítico do sistema.

Esta categoria de robôs não se enquadra no escopo em que o robô Omnino deseja se destacar, o objetivo do Omnino é oferecer: uma plataforma robótica integrada com o middleware ROS; processamento 100% embarcado; e capacidade computacional para diversas aplicações de controle autônomo e distribuído, sem qualquer necessidade de sistemas de processamento externo que comprometa sua autonomia e execução de tarefas.

Palm Pilot Robot

O kit *Palm Pilot Robot* é uma plataforma para um robô totalmente autônomo, fácil de construir e controlado por um computador de mão. Esse projeto foi criado por dois

grupos de pesquisa do Carnegie Mellon Robotics Insitute, a Toy Robots Initiative e o Manipulation Lab, com a intenção de permitir que praticamente qualquer pessoa comece a criar e programar robôs móveis a um custo modesto. Ele tem um *PalmTop* como controlador em um tamanho pequeno, funciona com baterias e pode exibir gráficos e uma interface de usuário interativa. Este robô é capaz de se movimentar e sentir o ambiente próximo. A base usa três rodas omnidirecionais que permitem dirigir em qualquer direção com controle independente de rotação, o que significa que ele se move holonomicamente no plano. A base também possui três sensores de alcance óptico para detectar obstáculos no ambiente próximo a até um metro de distância.

OuijaBot

A plataforma de robô omnidirecional OuijaBot, que reúne capacidades de detecção e atuação para implementar o controladores de translação e rotação. O OuijaBot contém quatro rodas omnidirecionais simetricamente colocadas, com rolos livres ao longo do seu perímetro que permite movimentos laterais. As rodas são conduzidas independentemente por quatro motores de corrente contínua, cada um deles classificado com uma corrente máxima de 5A abaixo de 12V, gerando um torque máximo de 0,78Nm. *Encoders* são montados com nos motores para medir sua velocidade. A plataforma possui sensores de controle e processamento, incluindo o *encoders*, monitoramento de corrente do motor, acelerômetro, giroscópio e magnetômetro. Um Raspberry Pi é usado como controlador e para executar ROS. O OuijaBot tem uma massa de 2,7 kg e pode se mover até 1m/s.

Rovio

Esta plataforma é projetada para ser um sistema de telepresença móvel, ela é capaz de se comunicar via Wifi, armazenar localizações e pode se guiar mesmo em localidades escuras. Também é capaz de se dirigir ao sistema de energia para recarregar suas baterias de forma autônoma. O Rovio é equipado com o seguinte conjunto de sensores: uma câmera VGA móvel, um microfone bidirecional, sensores infravermelhos para posicionamento, detecção de colisão e monitoramento de carga da bateria. A API do Rovio é baseado no protocolo HTTP, tornando possível controlar a plataforma utilizando qualquer dispositivo habilitado para a Web: PC ou Mac, telefone celular, smartphone, PDA ou até em consoles de videogame. Por ser uma plataforma bastante versátil, o porte para ROS foi prontamente realizado por usuários.

Robotino

O Robotino (Figura 2.16) é uma plataforma robótica omnidirecional desenvolvida pela FESTO. Robotino usa unidade omnidirecional para mover, que permite ao sistema mover-se livremente em um plano sem alterar seu angulo, tem um sensor de pára-choque em seu

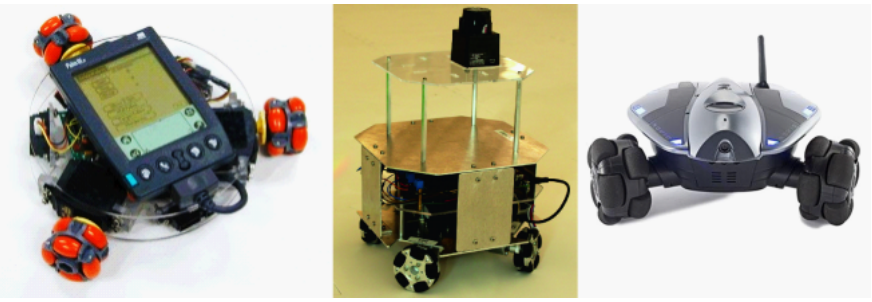

Figura 2.15: Plataforma PalmPilot Robot, OuijaBot e Rovio, respectivamente.

entorno, sensores de distância infravermelhos, uma câmera colorida e controle externo via PC. Através de um link WLAN, o Robotino pode enviar todas as leituras do sensor para um PC externo. Os comandos de controle também podem ser emitidos pelo PC externo na forma de controle remoto. Dessa forma, os programas de controle podem ser executados no PC externo ou no Robotino diretamente. O Robotino pode ser programado em seu ambiente de programação "nativo" RobotinoView II que é um software de simulação de robôs, que respeita o modelo de movimento do robô em ambiente virtual 3D. Também são disponibilizadas interfaces para programação em C, C ++, Python, Java, .NET, Matlab, Simulink, Labview, ROS e Microsoft Robotics Developer Studio.

Figura 2.16: Plataforma Robotino

A Tabela 2.1 disponibiliza algumas características e custos dos robôs apresentados.

Observando a Tabela 2.1, pode ser verificado que a maior parte das plataformas comerciais tem um custo elevado, muitas vezes impeditivos para grande parte dos pesquisadores e instituições.

No design da plataforma proposta Omnino, busca-se um robô móvel omnidirecional de tamanho pequeno e baixo custo. A eletrônica é baseada no Raspberry Pi 3B+, amplamente presente em diferentes laboratórios, permitindo a implementação de algoritmos

Tabela 2.1: Comparação modelos Omnidirecionais

Comparação entre diferentes plataformas				
Modelo	Custo	Maior Dimensão	Locomoção/Rodas	Integração ROS
Palm Pilot Robot	$ 299	12cm	Omnidirecional/3	Não
Ouijabot	$4000	48cm	Omnidirecional/4	Sim
Rovio Robot	$ 285	34cm	Omnidirecional/3	Sim
Robotino	$5130	45cm	Omnidirecional/3	Sim
Omnino	$100	14cm	Omnidirecional/3	Sim

distribuídos personalizados para enxame e alta integração com o ROS para controle, monitoramento e simulação.

2.8 Consenso

Um dos métodos mais usuais em sistemas multiagentes é chamado de consenso. Segundo Savino (2016), o significado do problema de consenso é fazer com que todos os agentes de um sistema alcancem um acordo sobre uma variável de interesse, assumindo que cada agente possa compartilhar e/ou adquirir informações dentro de um subconjunto de outros agentes, chamados vizinhos.

A lei de controle baseada em consenso é denominada protocolo de consenso, e é executada de forma distribuída entre os agentes do sistema para permitir a execução da tarefa cooperativa. Um protocolo de consenso para manter formação será aplicado na parte experimental para permitir aos robôs se deslocarem enquanto mantém formação com um robô líder.

2.9 Localização

Determinar a localização de um agente autônomo móvel no espaço de trabalho tem sido grande desafio para os desenvolvedores na área de robótica. Dentre as funções relativas de um agente autônomo, a navegação é um pré-requisito para que tarefas mais complexas sejam possíveis de ser executadas por um robô. Em alguns casos, o agente autônomo tem conhecimento prévio do ambiente, como um mapa, e deve alcançar os pontos de destino desejados. Para que isso seja possível, o robô deve saber onde está localizado para que assim possa planejar sua trajetória até seu ponto de destino.

Estimar correntemente esta posição é um dos grandes desafios da robótica móvel. Em um problema de auto-localização as possibilidades de implementação são inúmeras, cada uma conforme a necessidade e os custos envolvidos. A métrica para essa escolha baseia-se na tolerância aos erros de localização para a execução de cada tarefa e a capacidade computacional do sistema que será embarcado.

2.9.1 Localização por Odometria

Uma das maneiras de realizar esta tarefa pode ser o emprego de odometria. Consiste em determinar a posição do robô a partir da quantidade de voltas que suas rodas realizaram. Para isso são utilizados sensores do tipo *encoder* acoplados ao eixo do motor. O emprego dessa técnica é encarada como ”grosseira”, pois sua estimativa de localização decresce enquanto seus erros se acumulam. Fontes de erros como diâmetro desigual das rodas, desalinhamento do eixo de deslocamento, baixa resolução dos sensores odométricos, escorregamento nas rodas, pisos irregulares, entre outros também contribuem com a ineficiência deste tipo de sensor.

Erros de odometria devem ser minimizados, pois caso contrário o robô não conseguirá realizar nenhuma tarefa que envolva interação com uma área de trabalho. No trabalho desenvolvido por Thrun et al. (2001), uma abordagem estatística chamada Mixture Monte Carlo Localization, com base no casamento entre dados coletados pelos sensores odométricos e informações contidas no mapa, foi possível minimizar os erros de localização e degradação para pequenos conjuntos de amostras dos sensores odométricos, demonstrando eficiência, versatilidade e robustez. A técnica desenvolvida no trabalho de Borenstein and Feng (1996) utiliza a fusão de sensores por filtro de Kalman para mitigar os erros inerentes a qualquer sistema real de sensoriamento, acoplando giroscópios e *encoders*, com a técnica denominada Gyrodometry, que reduziu os erros de localização odométricos em até 18 vezes.

Nem sempre é possível que o agente autônomo móvel tenha disponível um mapa do ambiente em que ele será inserido, em casos como esse, o robô deverá ter mecanismos que possibilitem a criação do mapa da região de trabalho em tempo de execução, pois assim o agente será capaz de traçar os caminhos possíveis que o levam até uma posição desejada evitando os obstáculos existentes. Em casos como esse, o robô constrói um mapa usando a informação coletada por seus sensores, mas para isso ele precisa que sua localização no ambiente esteja correta, enquanto o próprio mapa ainda está em construção. Este problema é denominado como Mapeamento e Auto-localização Simultâneos (Simultaneous Localization and Mapping - SLAM). No trabalho desenvolvido por Wolf and Sukhatme (2005), pontos de referência foram criados para contribuir na localização do agente móvel, tornando-o capaz de mapear toda região de trabalho e detectar até mesmo objetos dinâmicos no ambiente, atualizando seu mapa durante a execução, a cada interação realizada.

Encoders

Encoders são sensores muito populares na robótica móvel. Eles são sensores responsáveis pela propriocepção do robô, fornecendo uma estimativa da distância percorrida pelo agente autônomo e consequentemente sua localização. Um odômetro do tipo *encoder* é composto por um disco acoplado ao eixo do motor e um dispositivo que gera pulsos elé-

tricos conforme a quantidade de giros que esse disco realiza. Essa onda quadrada gerada serve para medir o deslocamento do robô. Em geral, os *encoders* utilizados em robótica móvel medem de 20 a 2500 pulsos por revolução, *encoders* industriais podem medir até 10000 pulsos por revolução. Os *encoders* do tipo magnético utilizados no robô Omnino podem ser vistos na figura 2.17. O mesmo já está acoplado ao eixo do motor, onde o disco é equipado com um ímã, e sensores de efeito hall percebem a passagem deste, gerando assim uma saída quadrada com pico de 5Vcc. Um detalhe interessante deste *encoder* é que ele gera apenas 7 pulsos por revolução, mas por conta da caixa de redução presente no motor, essa relação se altera para um total de 200 pulsos por revolução da roda.

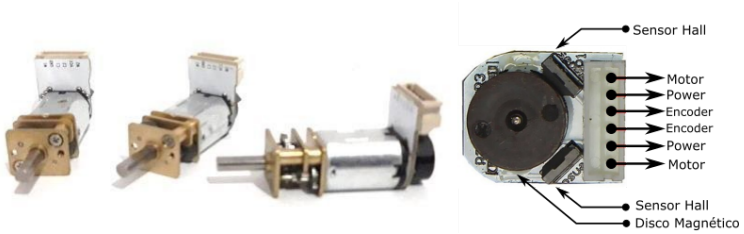

Figura 2.17: Motor CC com redução e Encoder

No Omnino, os *encoders* não foram utilizados para gerar a localização do robô, mas teve papel primordial para o controle de velocidade das rodas do agente. Contudo, a informação de velocidade pode ser obtida pelo controle central para implementar tais funções de odometria através desta leitura. No tópico controlador de velocidade será apresentado seu uso neste projeto.

2.9.2 Localização por visão

A utilização de câmeras tem sido utilizada por pesquisadores da área de robótica pela riqueza de informações fornecidas, que vão desde o reconhecimento de padrões geométricos, cores e movimentos até a profundidade dos elementos em uma cena com o auxílio de câmeras do tipo estéreo, para criação de mapas tridimensionais. Hoje, um dos principais exponenciais deste equipamento tem sido o sistema denominado Kinect. O Kinect foi desenvolvido pela Microsoft para controlar um dos seus videogames, XBox, e que é capaz de gerar dados do tipo RGB-D [1].

Atualmente as câmeras se tornaram bastante compactas, leves e acessíveis para serem empregadas em robótica móvel. Portanto, é conveniente utilizar esse tipo de equipamento para auxiliar agentes autônomos em sua localização. Marcadores naturais ou artificiais podem colaborar na localização de um robô através da utilização da visão computacional e seus algoritmos. Se o sistema visual contém apenas uma câmera, será necessário que o agente capture duas fotos do marcador durante sua movimentação, assim ele será capaz de calcular a relação trigonométrica e se localizar no espaço de trabalho tendo como

[1] Técnica capaz de prover os dados de cor e profundidade de um pixel

referencia o marcador. Tipos artificias de marcadores 2D possibilitam a determinação da localização do agente autônomo com apenas uma captura de imagem, conhecendo os padrões de calibração da câmera é possível obter a translação e rotação do robô em relação ao marcador ou a uma referência global. Muitos tipos de sistemas de marcadores 2D já são disponíveis para aplicação, como ARSTudio, ARToolkit e ARTag na Figura 2.18, estes modelos foram desenvolvidos para aplicações em realidade aumentada.

Figura 2.18: Tags de localização (1)Intersense, (2)ARSTudio, (3)ARToolkit, (4)ARTag

Para simplificar o processo de localização, o rastreamento baseado em marcadores 2D pode ser usado em várias aplicações. Com marcadores, a informação da posição e orientação pode ser adquirida com precisão e alta velocidade. Neste trabalho, é utilizado o método de localização baseado em marcadores ArUco (Figura 2.19) por ser um dos marcadores mais populares e prontamente disponíveis para integração no ROS.

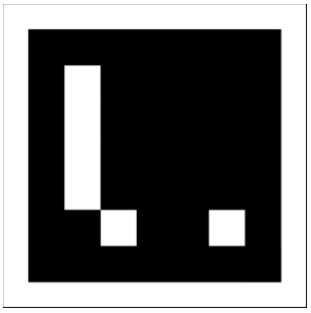

Figura 2.19: Tag ArUco

Os marcadores no sistema ArUco são definidos como uma matriz de quadrados 7x7. Cada um dos quadrados é preto ou branco, mas as duas linhas e colunas externas são pretas (Figura 2.19). Para uma detecção bem-sucedida do marcador, o sistema assume que o tamanho do marcador e os parâmetros de calibração da câmera são conhecidos. No trabalho realizado por Bacik (2017), foi demonstrada a praticidade e robustez do uso de marcadores ArUco no sistema operacional ROS para a realização de tarefas de navegação em agentes autônomos aéreos, obtendo bons resultados até mesmo para hardware de baixo custo, o que contribuiu na escolha desta metodologia de localização para a plataforma Omnino.

O pacote de localização ArUco[2], fornece um sistema que permite que um robô determine sua posição e orientação, observando vários marcadores fiduciais que são fixos no

[2]http://wiki.ros.org/aruco

ambiente de trabalho do agente. Inicialmente, a posição de um marcador é especificada ou determinada automaticamente. Depois disso, um mapa é criado pela observação de pares de marcadores fiduciais e pela determinação da translação e rotação entre eles.

O sistema de localização usa vários marcadores fiduciais de tamanho conhecido para determinar a posição do robô. Para cada marcador visível na imagem, é produzido um conjunto de vértices nas coordenadas da imagem. Como os parâmetros intrínsecos de calibração da câmera e o tamanho do fiducial são conhecidos, a posição do fiducial em relação à câmera pode ser estimada.

O diagrama da Figura 2.20, mostra o sistema de coordenadas de um marcador fiducial, que tem um comprimento em 2D. Uma vez impressos, eles podem ser afixados no ambiente. Eles não precisam ser colocados em nenhum padrão específico, mas a densidade deve ser tal que dois ou mais marcadores possam ser vistos pela câmera do robô, para que o mapa possa ser construído. Colocá-los no teto reduz os problemas de oclusão. As coordenadas da imagem (x, y) de cada vértice correspondem a um raio de distância da câmera. O código de estimativa de pose resolve um conjunto de equações lineares para determinar a coordenada global (x, y, z) de cada um dos vértices.

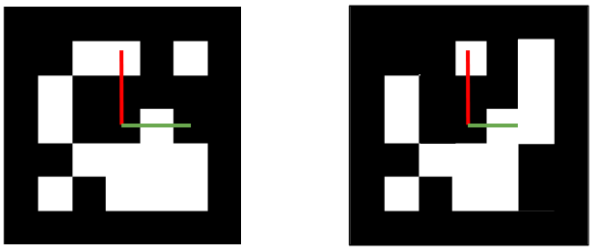

Figura 2.20: Fiducials ArUco

A partir disso, obtemos a transformação do sistema de coordenadas do marcador fiducial para o sistema de coordenadas da câmera. Dessa maneira, o mapa é construído à medida que mais pares de fiduciais são observados. Várias observações são combinadas, para produzir uma estimativa da pose de cada marcador fiducial, gerando assim um mapeamento das posições de cada um dos marcadores.

2.10 Método de Ziegler/Nichols e Sintonia PID

Controladores Proporcional Integral Derivativo (PID) são amplamente utilizados em sistemas de controle industriais por ter um número reduzido de parâmetros a serem ajustados. Eles proporcionam sinais de controle conforme o valor do erro entre o valor desejado, ou *set-point*, e o sinal de saída real da variável de processo. Em sistemas de controle de processos, sabe-se que o controle PID confere um controle satisfatório, embora em muitas situações eles podem não proporcionar um controle ótimo (Ogata and Severo, 1998).

Os métodos de sintonização desenvolvidos por Ziegler e Nichols, foram responsáveis por grande aceitação na indústria, pois através de métodos experimentais a sintonização de sistemas de controle foi simplificada, sem que operadores tivessem a total necessidade de modelos matemáticos do comportamento de seus sistemas. Os métodos de Ziegler-Nichols foram introduzidos em 1942 e hoje são considerados clássicos. Estes métodos continuam a ser largamente aplicados até hoje.

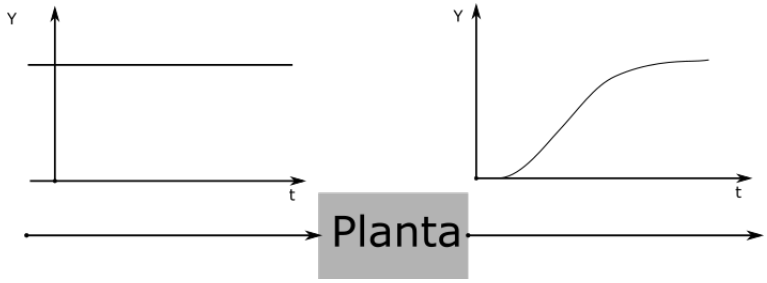

Figura 2.21: Resposta ao degrau unitário

No método de malha aberta, ou primeiro método de Ziegler e Nichols, foi obtido de forma experimental a resposta da planta a um degrau unitário, como mostra a Figura 2.21. Para que o sistema esteja adequado para a aplicação desse método, a resposta da planta não deve ser integradora ou oscilatória, mas sim em forma de S, ou sigmoidal. Esta curva pode ser gerada de forma experimental ou com a utilização de simulação dinâmica da planta Ogata and Severo (1998).

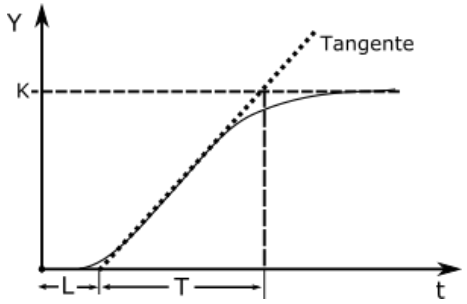

Figura 2.22: Sintonia por malha aberta

A curva sigmoidal pode ser caracterizada por duas constantes, a constante de tempo T e o atraso L. Ambos são determinados desenhando-se uma linha tangente no ponto de inflexão da curva sigmoidal, sendo possível determinar o ponto de intersecção da linha tangente com o eixo do tempo, assim como com a linha K, como mostra a Figura 2.22.

Apos a determinação dos valores da constante de tempo (T), do atraso (L) e do valor desejado (K), esses valores são substituídos na tabela 2.2 conforme o tipo de controlador que se deseja obter, (P) proporcional, (PI) proporcional-integral ou (PID) proporcional-integral-derivativo.

Em conclusão, é necessário enfatizar que o projeto que utilizam as técnicas de Ziegler e Nichols para a sintonia de controladores PID em que as dinâmicas da planta não são

Tabela 2.2: Sintonização P, PI e PID de Ziegler-Nichols

	K_p	K_i	K_d
P	T/L		
PI	$0{,}9T/L$	$0{,}3L$	
PID	$1{,}2T/L$	$2L$	$0{,}5L$

precisamente conhecidas. Por muitos anos, essas regras se mostraram muito úteis. Em conclusão, faz-se necessário realizar diversas tentativas e comparar os respectivos desempenhos para a escolha do melhor controlador, validando, através de experimentos reais ou simulações, o controlador selecionado. Além disso, a definição de critérios de desempenho adequados devem ser atendidos, fator de amortecimento, pico máximo e tempo de estabilização são alguns deles, para o desenvolvimento de um controlador adequado.

Capítulo 3

Projeto de Hardware

Omnino é um robô omnidirecional para pesquisa em enxame de agentes inteligentes e para os primeiros passos em educação de robótica móvel. Os requisitos definidos são baixo custo, hardware acessível e fácil disponibilização, integrável com plataformas de *software* difundidas. Baseando-se em impressão 3D de partes específicas e chassi, o projeto considera duas restrições principais: três rodas omnidirecionais para locomoção, que podem ser adquiridas a baixo custo ou impressas; e a Raspberry Pi 3 como base para a eletrônica, por ser uma plataforma de placa única comumente difundida em laboratórios.

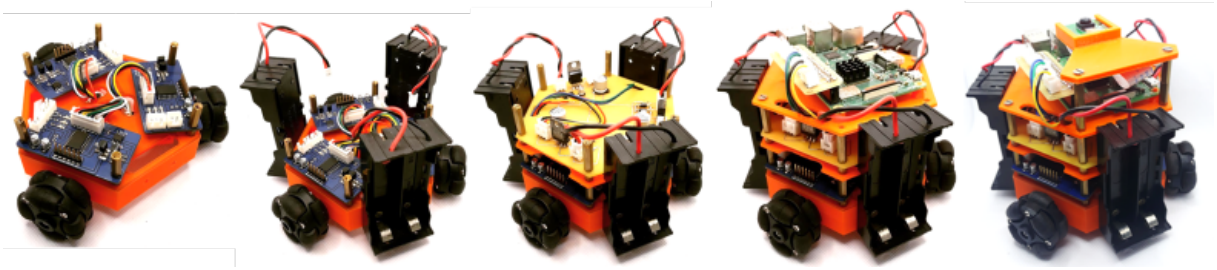

Figura 3.1: Projeto de Hardware do Omnino

Baseado nestes requisitos, o projeto mecânico é mostrado na Figura 3.1. Da esquerda para a direita, mostra-se a camada base, um hexágono com as rodas colocadas em três lados, e inclui também os controladores dos motores sobre a base. Os suportes das baterias também são posicionados nas laterais, podendo ser adaptados à bateria desejada. Na sequência da terceira imagem, mostra-se a segunda camada, que inclui o controle de potência para fornecer carga suficiente para motores e sensores, assim como carga estável para o circuito de controle. Em seguida, a terceira camada inclui o controlador baseado na Raspberry Pi 3. Por fim, a quarta camada possui uma adaptação de câmera integrada ao controlador para implementar um sistema de localização baseado em processamento de imagens. O projeto visa expansões em camadas adicionais para sensores e atuadores.

Os blocos amarelos no diagrama da Figura 3.2 representam o sistema de locomoção. São utilizados três motores, Motor 1, Motor 2 e Motor 3, acoplados a rodas omnidirecionais, que possuem roletes ao longo de seu perímetro para propiciar à roda um grau extra

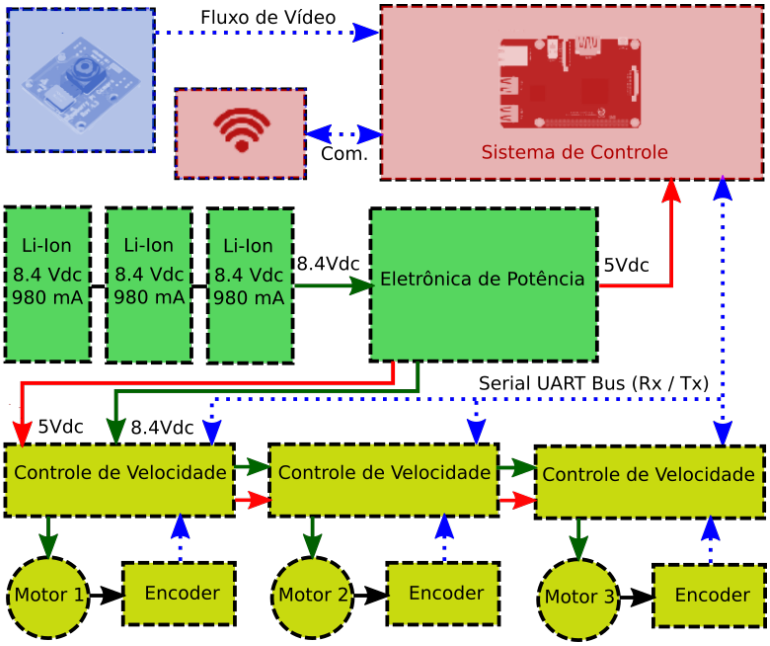

Figura 3.2: Projeto de Hardware do Omnino

de liberdade ao longo do eixo de rotação da roda. Esta roda permite que o robô possa se locomover em qualquer direção num plano, sem a necessidade de reorientação. Cada motor possui em sua estrutura um *Encoder* acoplado no eixo, isto é, um disco magnético e sensores de efeito *hall*, onde a cada giro realizado pelo eixo do motor gera um pulso quadrado em sua saída, permitindo a medição da velocidade. A seta em cor preta indica o sinal físico medido pelo Encoder, e a seta pontilhada em azul indica o pulso de velocidade usado pelo Controle de Velocidade.

As placas de Controle de Velocidade se comunicam com a Raspberry Pi, no Sistema de Controle, pelo barramento serial UART (setas pontilhadas em azul indicando sinais de controle). Embora esta placa tenha sido projetada para se comunicar com o Raspberry Pi 3B+ do Omnino, ela também pode ser usada com qualquer outro sistema que tenha o mesmo barramento de comunicação, como os microcontroladores da família PIC, Atmel, Arduino, Notebooks e PC's com conversor USB/Serial. O controle de velocidade está embarcado num microcontrolador PIC16F628A, que de acordo com o sinal de controle recebido pela UART (setas azuis pontilhadas de controle), gera o sinal de potência PWM para o motor (setas verdes de sinal de potência), leitura de pulsos do Encoder (setas pontilhadas azuis da medição de velocidade).

A placa de potência do robô Omnino é descrita no diagrama da Figura 3.2 na cor verde. Possui proteção contra inversão de polaridade nas baterias, capacidade de condução de corrente de até 3A para a tensão regulada de 5Vcc, LED's indicadores de funcionamento para cada um dos *packs* de baterias, filtro capacitivo e proteção contra FCEM que podem ser geradas pelo acionamento de sistemas indutivos como motores. As conexões da placa utilizam três *packs* de baterias Li-Ion como fonte de alimentação. A saída da placa

de Eletrônica de Potência fornece tensão regulada para alimentação dos circuitos que demandam potência (setas em verde) em 8,4 Vdc, assim como a fonte de tensão para os circuitos de controle ligados à Raspberry Pi (setas em vermelho) em 5 Vdc.

O Sistema de Controle representado pelos blocos em vermelho, se baseiam na plataforma Raspberry Pi 3 B+, um computador com o tamanho reduzido, desenvolvido para aplicações educacionais. Entre suas especificações técnicas as que se destacam são: a frequência de processamento de 1,4GHz; memória RAM de 1GB; um barramento GPIO de 40 pinos; conexão via Wi-fi e Bluetooth 4.2, o que torna a Raspberry também responsável pelo sistema de comunicação. Graças a seu pequeno tamanho e preço acessível, é amplamente adotado por entusiastas e "makers"que desejam criar projetos com capacidade de processamento maior que um simples microcontrolador. Apesar de serem mais lentos que um computador Desktop ou Notebook, a Raspberry Pi pode embarcar um sistema operacional Linux com uma diversidade de recursos de comunicação e processamento. O *software* de controle de alto nível é implementado neste Sistema de Controle, e é responsável pela comunicação com os demais sensores, como a câmera em azul, e com os controladores de velocidade.

A câmera Raspberry Pi Cam possui conexão direta com muitos modelos de Raspberry Pi, uma resolução de 5MP e taxa de gravação de 30fps, dimensões de 25mm x 20mm x 9mm e uma massa de 3g, sendo visada para aplicações em dispositivos móveis em que o tamanho e massa são critérios importantes. O robô Omnino utiliza a Raspberry Pi Cam para realizar a localização do robô no ambiente através de processamento de imagem.

Com isto, no fim deste capítulo está representada na Tabela 3.2, o custo de *hardware* da plataforma apresentada. Esta tabela inclui os custos dos componentes mecânicos e eletrônicos em dólares americanos. Ao final, é fornecido também o custo esperado sem incluir a Raspberry, visto que, por vezes, estas placas se encontram disponíveis num grande número de laboratórios educacionais.

3.1 Controlador de Velocidade

Uma das necessidades básicas para a navegação autônoma de um robô é sua capacidade de determinar as velocidades em cada uma de suas rodas. Para que isso seja possível, projetar um sistema de controle que verifique as variáveis envolvidas no processo é primordial. Para tal tarefa existem basicamente duas configurações de malhas de controle, essas configurações são chamadas de controle em malha aberta e controle em malha fechada.

O controle em malha aberta, visto na Figura 3.3, consiste em aplicar um sinal de *set-point* no controlador e esperar que em um determinado tempo a variável controlada atinja um valor desejado. Neste tipo de sistema de controle a informação de como a saída do processo está se comportando não pode ser mensurada. Mais especificamente, o

sinal de controle não é atualizado a partir de uma medição do erro entre o sinal de saída e seu *set-point*. Sistemas de controle de malha aberta podem operar de forma eficiente em sistemas repetitivos e bem calibrados em ambientes não dinâmicos. Este método não deve ser aplicado em sistemas que necessitam de robustez e confiabilidade a distúrbios, por este motivo a metodologia de malha fechada se enquadra de forma mais eficiente no cumprimento desses critérios.

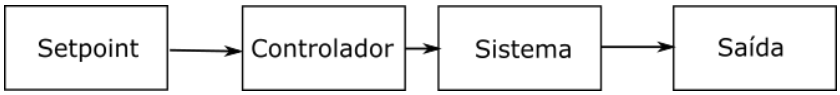

Figura 3.3: Controlador em Malha Aberta

No controle em malha fechada, informações sobre como a saída de controle está se comportando são utilizadas para determinar o sinal de controle que deve ser aplicado ao processo. Isto só é possível a partir do momento em que uma realimentação é feita da saída do sistema para a entrada do controlador. Em geral, a fim de tornar o sistema mais eficaz e sensível a perturbações do processo, o sinal de saída é comparado com um sinal de *set-point* e o erro, resultante da diferença entre esses dados, é utilizado para determinar o sinal de controle que deve ser aplicado ao processo. Assim, o sinal de controle é ajustado de forma a corrigir este erro para aproximá-lo de zero. O dispositivo que utiliza o sinal de erro para calcular o sinal de controle a ser aplicado ao sistema é chamado de controlador. O diagrama básico de um sistema de controle em malha-fechada é mostrado na Figura 3.4.

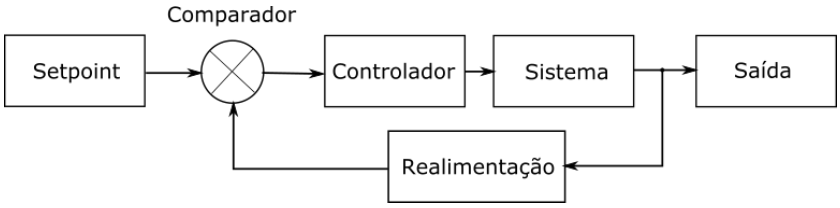

Figura 3.4: Controlador em Malha Fechada

Para o controle de velocidade das rodas do agente Omnino foi utilizada uma arquitetura de controle por realimentação em malha fechada. Nos trabalhos de Kanojiya and Meshram (2012) e Dimeas et al. (2017) é possível verificar sua aplicação em controladores para motores de corrente contínua.

Aplicando o conceito do controlador em malha fechada, o diagrama da Figura 3.5, foi desenvolvido para adequar os componentes eletrônicos responsáveis por cada etapa do processo. O microcontrolador PIC16f628A será o dispositivo que realizará a leitura do *set-point* pelo canal serial. Tendo o valor de *set-point* desejado o microcontrolador fará a comparação entre a variável de *set-point* e a variável de processo, calculando assim o erro do sistema. Aplicando o erro do sistema ao seu controlador, será gerada uma resposta de saída no canal PWM do dispositivo, afim de realizar os ajustes necessários para que

o motor alcance o valor de *set-point* de velocidade, aumentando ou diminuindo seu sinal conforme o erro.

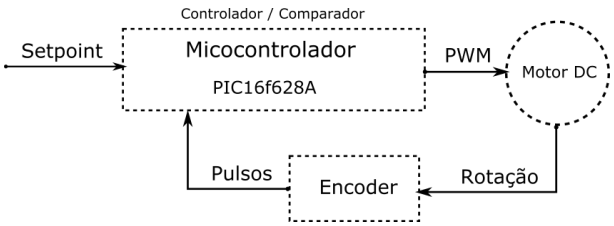

Figura 3.5: Controlador em Malha Fechada de Velocidade

O circuito esquemático pode ser visto na Figura 3.6, onde está representado todo o circuito do controlador de velocidade.

Com o auxílio do *software* de confecção de placas, Eagle PCB Design da empresa Autodesk, foi possível otimizar a criação da placa de circuito impresso, pois esta ferramenta oferece mecanismos como o auto-roteamento do circuito e a configuração do número de camadas que a PCI pode ter. Na Figura 3.7 é mostrado o resultado final do projeto da placa.

O Eagle também gera os arquivos apropriados, denominados GERBER, para enviar o projeto para uma empresa de manufatura de placas de circuito impresso, facilitando a produção e a velocidade de desenvolvimento do projeto Omnino, gerando maior segurança e durabilidade ao circuito. A placa manufaturada pode ser vista na Figura 3.8.

Após a soldagem dos componentes da placa de controle é possível acompanhar o resultado final na Figura 3.9.

A placa de controle de velocidade, foi idealizada para ser acessível, sem componentes restritivos ou que necessitem de técnicas sofisticadas para sua confecção. Além disso, a placa de controle de velocidade não foi projetada exclusivamente para a plataforma Omnino, ela pode ser empregada em quaisquer projetos que necessitem de acionamentos realizados por motores de corrente contínua, que tenham uma interface de comunicação UART com velocidade de 9600bps. Assim, a placa de controle pode ser confeccionada de forma manual, para aplicações em sala de aula e no desenvolvimento de experimentos em laboratório para disciplinas de controle, sintonia, robótica e atividades ”makers”. Na Figura 3.10 é possível verificar a mesma confeccionada de forma artesanal.

Com o *hardware* construído, é necessário fornecer o *software* para permitir seu funcionamento. O microcontrolador é um dispositivo programável, logo, necessita que uma série de instruções sejam gravadas em sua memória de programa, para que assim possa realizar alguma atividade.

A placa de controle de velocidade tem a possibilidade de receber novos métodos de controle e ajustes no PID que podem ser desenvolvidos por alunos e pesquisadores. Assim, foi destinada uma barra de pinos do tipo *Header*, para que o operador possa utilizar um gravador que respeite a sequencia de pinos do PicKit3 e realize a atualização do *Firmware*

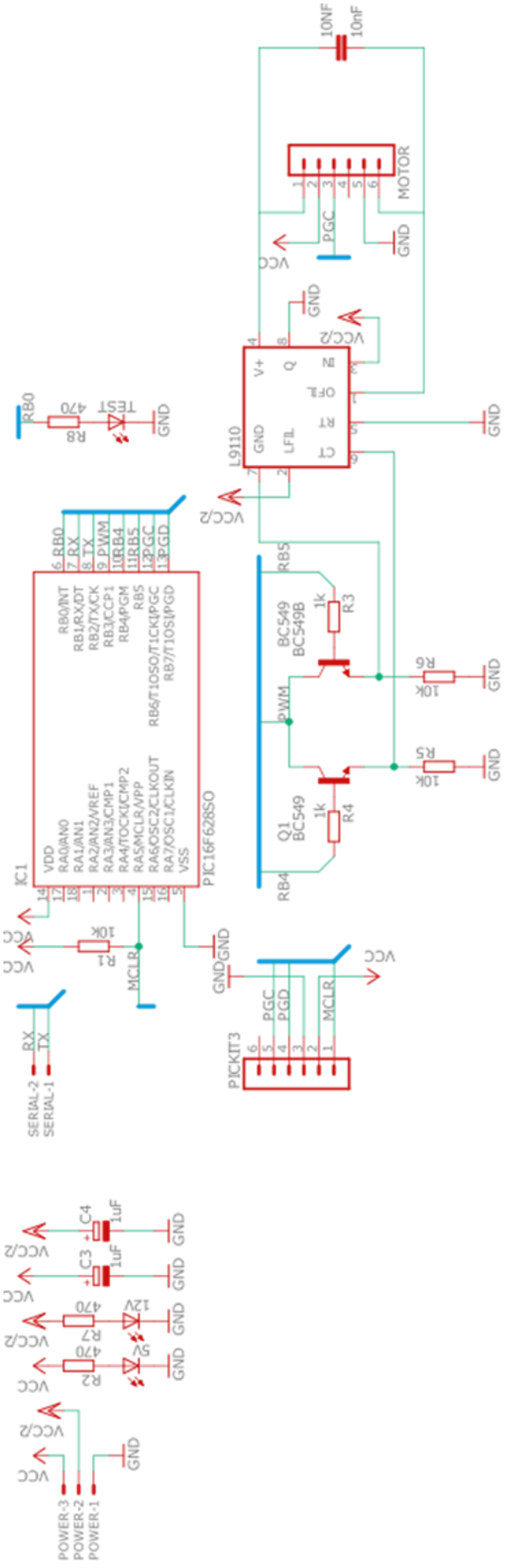

Figura 3.6: Diagrama Esquemático Controlador de Velocidade PID

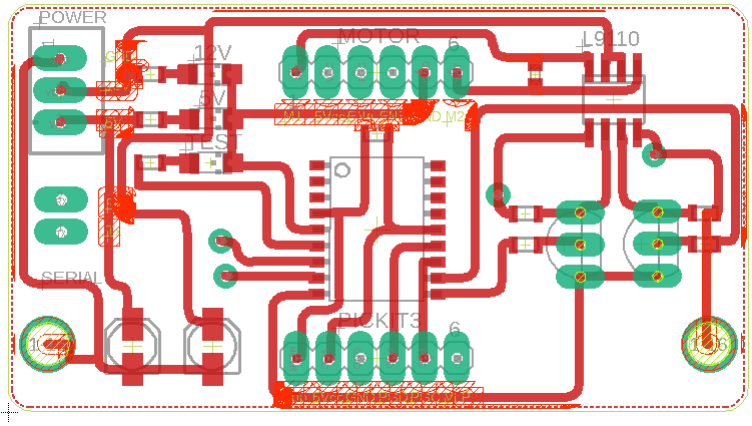

Figura 3.7: Placa projetada no software Eagle

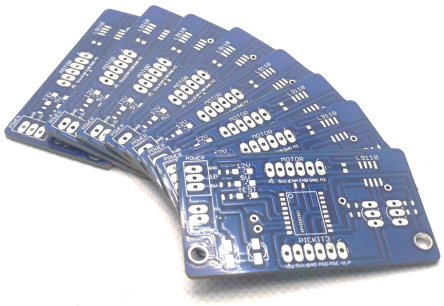

Figura 3.8: Placa do controlador manufaturada

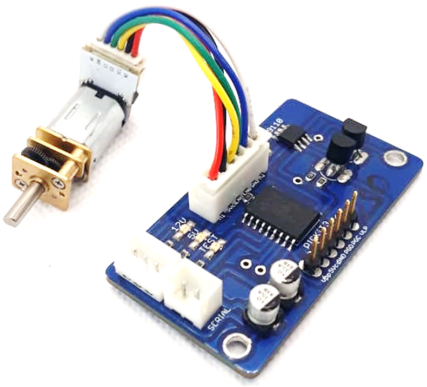

Figura 3.9: Placa do controlador de velocidade

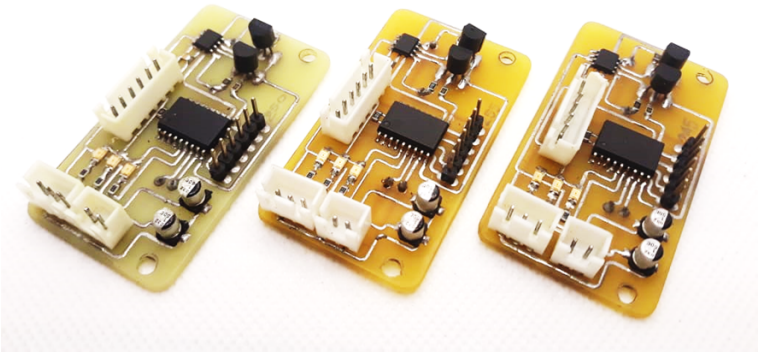

Figura 3.10: Placa do controlador de velocidade confecção manual

do controlador de velocidade, como pode ser visto na Figura 3.11.

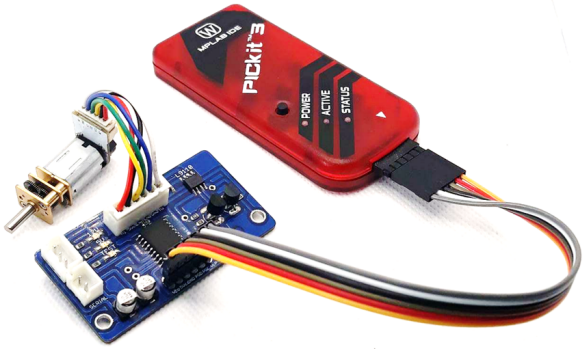

Figura 3.11: Gravador PicKit3

Com todo o projeto de *hardware* do controlador de velocidade do robô confeccionado, torna-se possível implementar o *software* de controle para os motores de corrente contínua já apresentado neste trabalho. Na Figura 3.12 pode-se verificar o fluxograma de funcionamento do controlador.

Os comandos UART utilizados para comandar a placa de controle PID são baseados no padrão ASCII. A placa deve receber 3 bytes respeitando as seguintes configurações:

Primeiro Byte: deve ser o endereço da placa, que pode ser qualquer valor entre 0 e 255;

Segundo Byte: indica o *set-point* de velocidade do motor, os valores aceitos devem estar entre 0 e 100;

Terceiro Byte: determina o sentido de rotação ou parada do motor, o caractere (f) faz o motor girar para a direita, o caractere (b) faz o motor girar para a esquerda e o

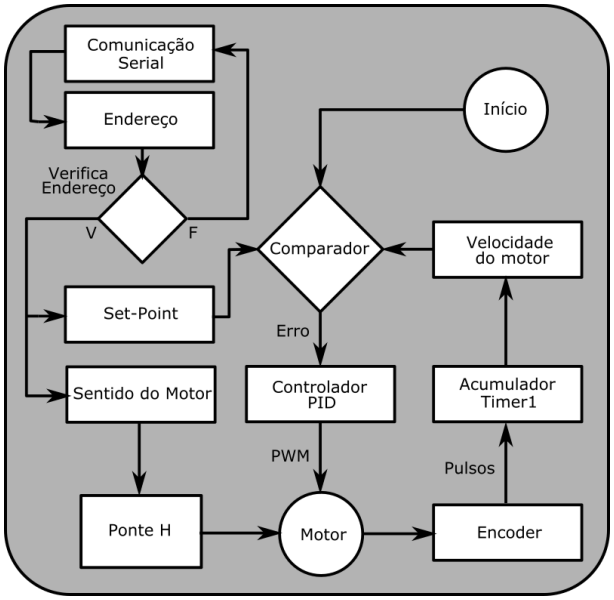

Figura 3.12: Fluxograma Controlador PID

caractere (s) é usado para parar o motor.

3.1.1 Ajuste do Controlador PID usando Ziegler/Nichols

A técnica utilizada para sintonizar o PID para o controlador de velocidade dos motores foi a de malha aberta de Ziegler-Nichols. Uma vez que a resposta do motor ao degrau de entrada se mostrou com uma curva sigmoidal (Figura 3.13), a técnica de Ziegler/Nichols pode ser implementada, dado que esta forma de onda de saída corresponde a um sistema de primeira ordem com atraso.

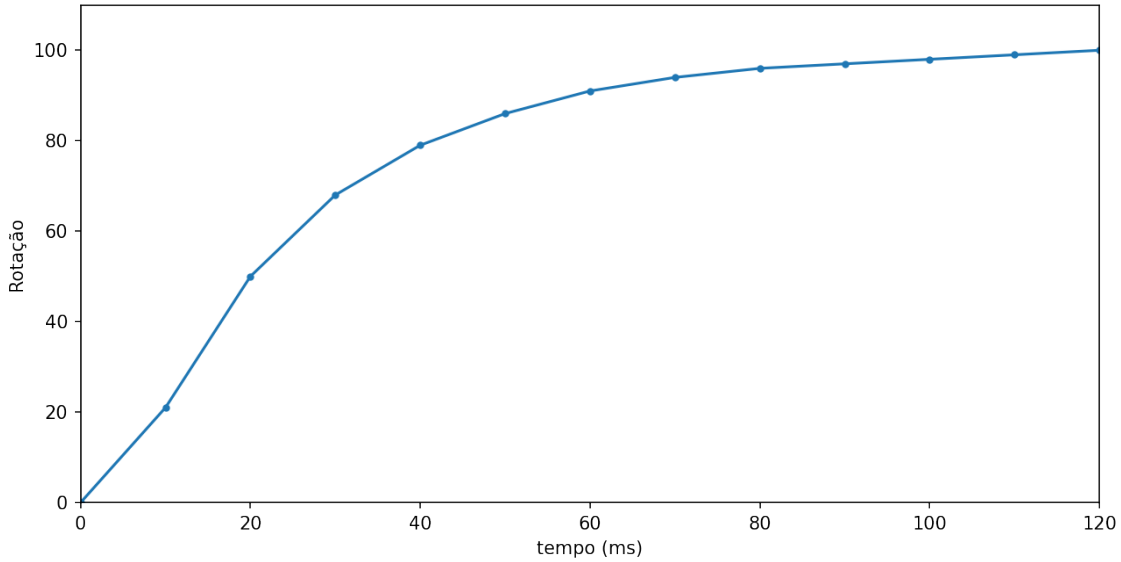

Figura 3.13: Resposta experimental do motor ao degrau

Na Figura 3.14 é possível verificar a aplicação da técnica de malha aberta no gráfico

de resposta ao degrau do sistema. Assim, foi possível coletar os valores das variáveis de sintonia, onde $K = 100$, $L = 0{,}13$ e $T = 0{,}35$, para que assim seja possível gerar os valores de sintonia das variáveis de controle K_p, K_i e K_d, para o controlador de velocidade dos motores do robô Omnino.

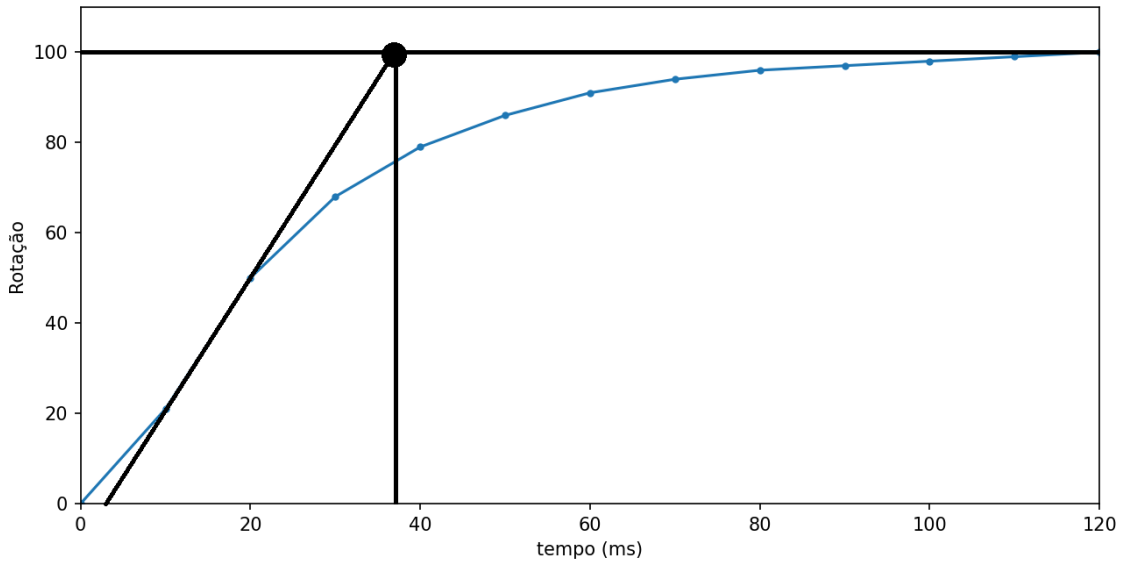

Figura 3.14: Aplicação da técnica de Malha Aberta de Ziegler e Nichols

Utilizando a Tabela 2.2 de Ziegler-Nichols de sintonia, é possível substituir os valores adquiridos de K, L e T para obter os dados de sintonia de K_p, K_i e K_d para o sistema de controle PID. A Equação 3.1 determina o valor de sintonização proporcional do controlador.

$$\begin{aligned} K_p &= \frac{1{,}2T}{K}L \\ K_p &= \frac{1{,}2 \cdot 0{,}35}{100}0{,}13 \\ K_p &= 0{,}03231 \end{aligned} \tag{3.1}$$

A Equação 3.2 determina o valor de sintonização integral do controlador.

$$\begin{aligned} K_i &= 2L \\ K_i &= 2 \cdot 0{,}13 \\ K_i &= 0{,}26 \end{aligned} \tag{3.2}$$

A Equação 3.3 determina o valor de sintonização derivativo do controlador.

$$\begin{aligned} K_d &= 0{,}5L \\ K_d &= 0{,}5 \cdot 0{,}13 \\ K_d &= 0{,}065 \end{aligned} \tag{3.3}$$

Após a implementação dos valores de sintonia, o controlador obteve a resposta mostrada na Figura 3.15. A saída alcança o valor desejado de *set-point*, mas apresenta um *overshoot* de 40% e oscilação elevada até alcançar estabilidade em aproximadamente 300ms. Para que o controlador tenha um melhor desempenho é ideal que o sinal de saída se aproxime de uma forma de onda quadrada.

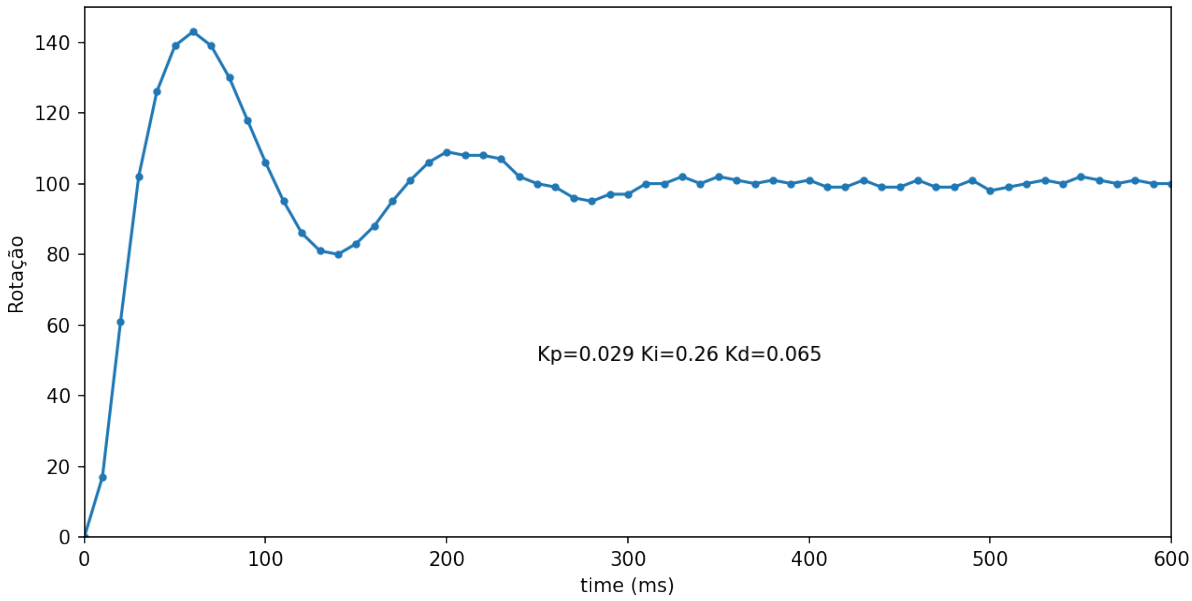

Figura 3.15: Sintonia Ziegler-Nichols

Para obter um controlador mais estável e rápido, o método de aproximações sucessivas foi empregado ao controlador até que uma resposta de saída do sistema se mostrasse o mais próximo de uma forma trapezoidal. Este método consiste em modificar as ações de controle e observar os efeitos na variável de saída do processo. A modificação das ações são realizadas continuamente até a obtenção de uma resposta do sistema ótima. Em função da sua simplicidade é um dos métodos mais utilizados, mas seu uso fica impraticável em processos com grande inércia.

Após a aplicação da técnica de aproximações sucessivas foi alcançada uma resposta de saída do sistema de controle de velocidade que pode ser vista na Figura 3.16, onde os critérios que avaliam o desempenho do controlador podem ser avaliados e comparados.

Na Tabela 3.1 a seguir, pode-se verificar que os critérios de desempenho do controlador foram bastante satisfatórios após a utilização da técnica de aproximações sucessivas. Foi possível avaliar que o controlador de velocidade da plataforma Omnino se aproximou de uma forma de onda trapezoidal em sua saída.

De acordo com os testes realizados e dados coletados, o método de sintonização de Ziegler-Nichols de malha aberta não pode ser a metodologia definitiva para a parametrização de um controlador PID, mas demonstra ser um referencial de partida para aplicação de outras técnicas de controle e otimização. Isto pode ser verificado quando é aplicada a téc-

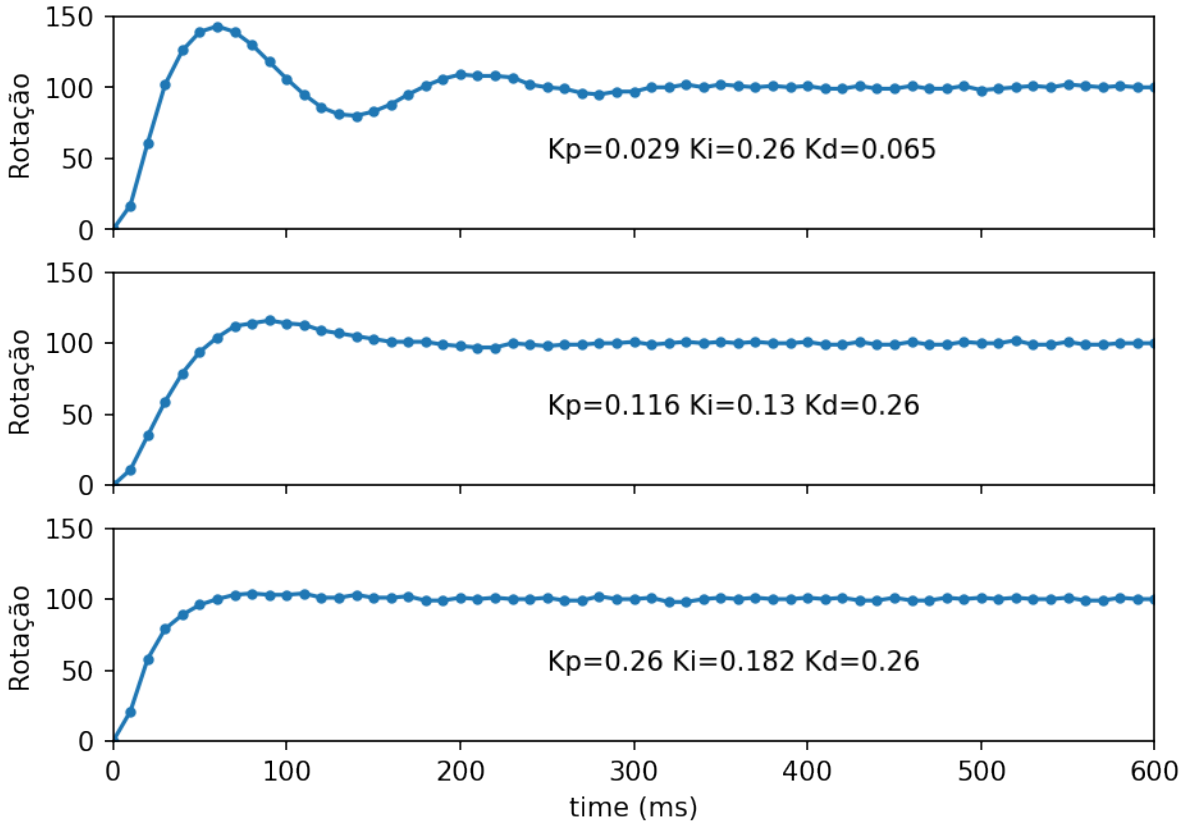

Figura 3.16: Sintonia Ziegler-Nichols e Aproximações Sucessivas

Tabela 3.1: Performance de Sintonias (ZN=Ziegler/Nichols, ZN-AS=Ziegler/Nichols e Aproximação Sucessiva

Método	Overshoot (%)	Tempo de Subida (ms)	Tempo de Estabilização (ms)
PID-ZN	43	30	300
PID-ZN-AS	4	30	60

nica de aproximações sucessivas ao controlador, gerando um sinal de saída muito próxima de valores ideais, com baixa oscilação, baixo overshoot e baixo tempo de estabilização.

3.2 Placa de Potência

A placa de regulação (Figura 3.17) foi desenvolvida com reguladores de tensão da família 78xx, dispositivo eletrônico popular entre projetistas e de fácil acesso. Os reguladores 7805 utilizados são capazes de suportar correntes de até 1A, oferecem proteção contra curto circuito, suportam um range de entrada de 7,5Vcc a 20Vcc e proteção térmica. Leds indicadores foram alocados para indicar o funcionamento de cada um dos *packs* de baterias que estão conectados à placa. Diodos foram adicionados para evitar a inversão de polaridade na fonte e para redução de FCEM, que podem afetar o bom funcionamento de circuitos digitais. Capacitores eletrolíticos estão acoplados a entrada e saída da fonte, a fim de reduzir oscilações e desligamentos da tensão de alimentação causadas pelo acimamento abrupto de cargas.

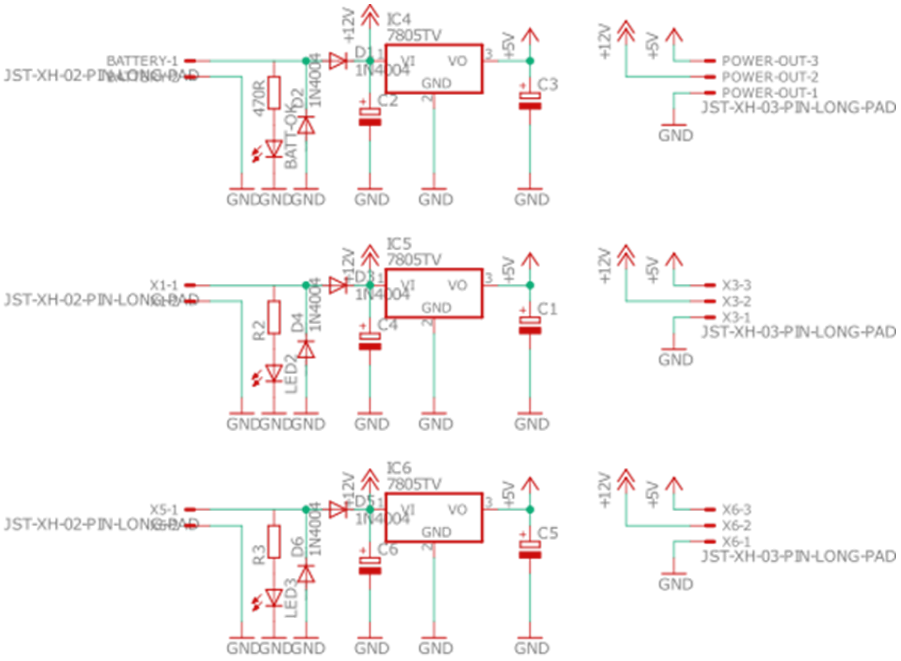

Figura 3.17: Diagrama esquemático da placa de potência

A placa de potência tem capacidade de fornecer uma tensão de 5Vcc com uma corrente nominal de até 3A.

3.3 Estrutura Omnino e Impressão 3D

O advento da impressão 3D revolucionou a prototipagem industrial. Isto teve um impacto fundamental na área de robótica. A utilização de ferramentas básicas de modelagem CAD

e o acesso a uma impressora 3D, propiciou a estudantes e pesquisadores a capacidade de construir rapidamente protótipos para atender aos requisitos de um projeto. Partes ou mecanismos podem ser testado e melhorados de forma contínua durante sua fase de prototipagem, tomando apenas algumas horas entre a detecção da falha e/ou melhoria, e a impressão de uma nova solução.

Um grande esforço foi dedicado ao desenvolvimento da estrutura responsável para acomodação das placas de controle, motores e baterias do Omnino. A tecnologia de impressão 3D foi essencial para a fase de projeto da plataforma, permitindo que o Omnino seja uma plataforma modular com encaixes precisos de componentes, que requer um número mínimo de recursos para a troca rápida de peças e materiais do robô.

O robô Omnino é construído basicamente por quatro plataformas descritas a seguir:

A plataforma 1 é responsável pela acomodação dos motores de corrente contínua, rodas suecas, placas de controle PID de velocidade e os *packs* das baterias (Figura 3.18).

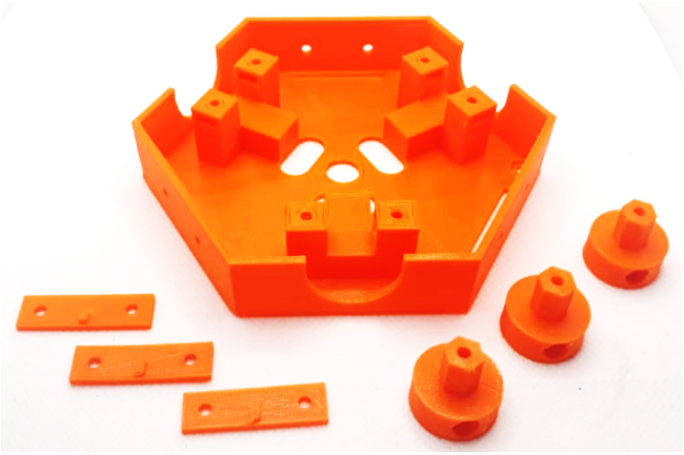

Figura 3.18: Base 1 Chassi Omnino

A plataforma 2 foi desenvolvida para isolar os terminais elétricos da placa de potência das placas de controle, evitando assim curto-circuitos acidentais (Figura 3.19).

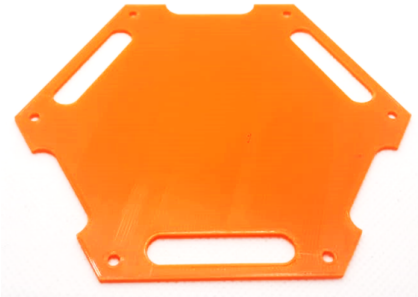

Figura 3.19: Base 2 Chassi Omnino

A plataforma 3 é responsável por dar suporte a placa Raspberry Pi 3 B+, possui uma série de pinos para o encaixe da placa ao robô sem necessidade de uso de parafusos e porcas (Figura 3.20).

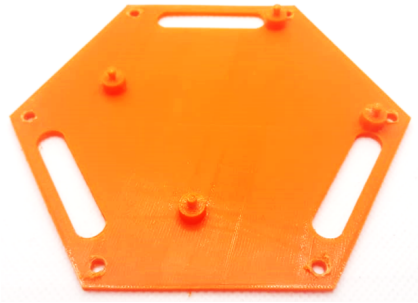

Figura 3.20: Base 3 Chassi Omnino

A plataforma 4 é responsável por dar suporte à câmera RaspCam, que fica apontada para cima a fim de realizar a localização do Omnino através da leitura das tags ArUco (Figura 3.21).

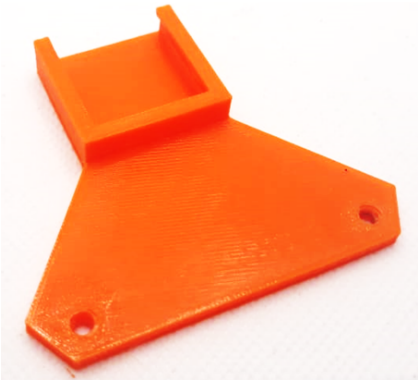

Figura 3.21: Base 4 Chassi Omnino

A impressão 3D de todas as peças da plataforma pode ser realizada em curto espaço de tempo, a depender da impressora utilizada. Os arquivos podem receber melhorias e modificações feitas pelos usuários, a plataforma está aberta para receber novos módulos e estão disponíveis para a impressão no repositório do projeto, que pode ser acessado no seguinte link: `https://github.com/HenrickRamalho/Holonomic-Robot-Swarm-Agent`. Os *softwares* gratuitos Sketchup e Cura, foram utilizados respectivamente para o desenvolvimento do projeto e fatiamento do arquivo de impressão. A impressora 3D utilizada foi uma Ender 3, que usa a técnica de Modelagem por Deposição Fundida (FDM). Basicamente, FDM é um processo de manufatura aditiva em que plásticos e outros materiais são derretidos por um extrusora e depositada camada por camada sobre uma superfície de impressão. A Ender 3 é uma impressora com um preço acessível e de fácil operação, com um volume de impressão de 22cm x 22cm x 25cm e estrutura robusta. O

filamento escolhido para impressão foi do tipo PLA, por ser um material biodegradável e que não sofre deformações que podem ser causada por rajadas de ar durante a impressão, comprometendo esteticamente e mecanicamente o produto final.

3.4 Custo de hardware

A Tabela 3.2 mostra os custos envolvidos para a construção do Omnino, os valores considerados foram feitos em relação a compra destes materiais na China. Além de ser uma plataforma de baixo custo, ter o Raspberry Pi 3 B+, fornece ao estudante e pesquisador a possibilidade de realizar a construção de algoritmos com alta capacidade de processamento e de forma totalmente descentralizada. O que destaca o Omnino da grande maioria das plataformas robóticas, que utilizam microcontroladores e sistemas de processamento centralizados para a execução de suas tarefas.

Tabela 3.2: Tabela de Custo do Omnino em USD

Item	Cost
3 Rodas Omnidirecionais	$13
3 Motores com Encoders	$17
3 Suportes de Baterias	$2
6 Baterias	$6.2
1 Raspberry Pi 3 Model B+	$35
1 Raspicam	$4
Partes impressas em 3D	$9
3 Controladores de velocidade	$10
1 Placa de potência	$3.5
Elementos de Fixação	$2
Total (incluindo Raspberry)	$ 101.70
Total (sem Raspberry)	$ 66.70

Capítulo 4

Integração com ROS

O Sistema operacional ROS é uma estrutura de código aberto baseada em Linux que permite a rápida implementação de hardware e software para sistemas robóticos autônomos e enxames. Em sua essência, consiste em um modelo de publisher/subscriber para comunicação, onde "Tópicos"compostos por estruturas de mensagens predefinidas são utilizados para a comunicação entre os vários 'nós' na rede. Esses tópicos, por exemplo, 'localização', podem ser acessado por qualquer 'nó' da rede, permitindo fácil escalabilidade dos publishers e subscribers. Hardware e software atuando como publishers e subscribers podem se comunicar facilmente com outros componentes do sistema em um ambiente de maneira bem definida.

Nos últimos anos, o ROS se tornou a principal plataforma operacional para o controle de robôs em ambientes de ensino e pesquisa, em parte graças à sua extensa funcionalidade e onipresença através de pacotes disponíveis gratuitamente, inclusive para simulação e controle.

4.1 Calibração Câmera

As câmeras digitais estão se difundindo em aplicações embarcadas devido ao fato de que seu custo de fabricação tem sido cada vez mais reduzido. Porém, esse fácil acesso e redução de custos não garantem qualidade do equipamento, suas lentes geram pequenas distorções às imagens capturadas. Isto acarreta problemas para aplicações com visão computacional, pois tais algoritmos necessitam de grande fidelidade entre as imagens capturadas e o mundo real. Por este motivo, se torna fundamental a realização de sua calibração para a correta utilização da câmera em visão computacional. O efeito olho de peixe, por mais que seja imperceptível aos olhos humanos, trazem erros que podem prejudicar o bom funcionamento de seus algoritmos. Em casos em que o software é responsável pelo cálculo de dimensões de um objeto ou de distâncias, este efeito irá gerar erros em seus resultados. A fim de reduzir este efeito gerador de erros, a calibração da câmera se faz necessária. O pacote Camera_Calibration auxilia o operador na realização

de tal tarefa. A calibração da câmera é feita utilizando um padrão conhecido em tamanho, dimensão e formato. Este padrão é um tabuleiro de xadrez de formato retangular com as dimensões de seus quadrados conhecidas, como mostrado na Figura 4.1.

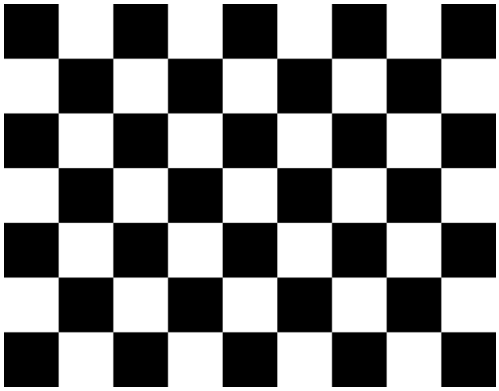

Figura 4.1: Tabuleiro de Calibração

O tabuleiro deve ser impresso em uma superfície plana, ou em uma folha e, logo após, ser colada em uma superfície reta e resistente, de modo que deformações não gerem erros no processo de calibração. Com o tabuleiro pronto, devem ser informados ao programa alguns parâmetros para que assim a calibração seja realizada, i.e.: o número de linhas verticais e horizontais do tabuleiro e a dimensão das arestas dos quadrados. Após a inserção isso abrirá a janela de calibração, que destacará o tabuleiro como na Figura 4.2. O software inicia uma série de captura de imagens e ao fim retorna um arquivo com a matriz de calibração da câmera.

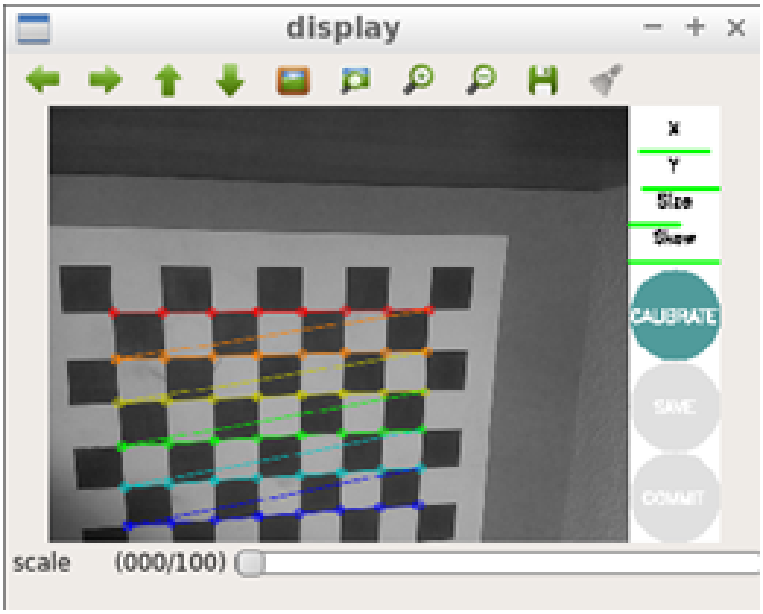

Figura 4.2: Tela de Calibração

Com este processo concluído a câmera embarcada ao Omnino será capaz de fornecer os dados de localização com maior confiabilidade.

4.2 SLAM com ArUco

A estrutura montada para a realização dos experimentos (Figura 4.3), foi idealizada para ter suporte para a distribuição das tags ArUco. A colocação das tags foi feita de tal maneira que fosse possível, para o robô Omnino, visualizar ao menos uma delas em qualquer ponto em que o robô estivesse localizado sobre a mesa, evitando problemas de oclusão durante a execução de tarefas.

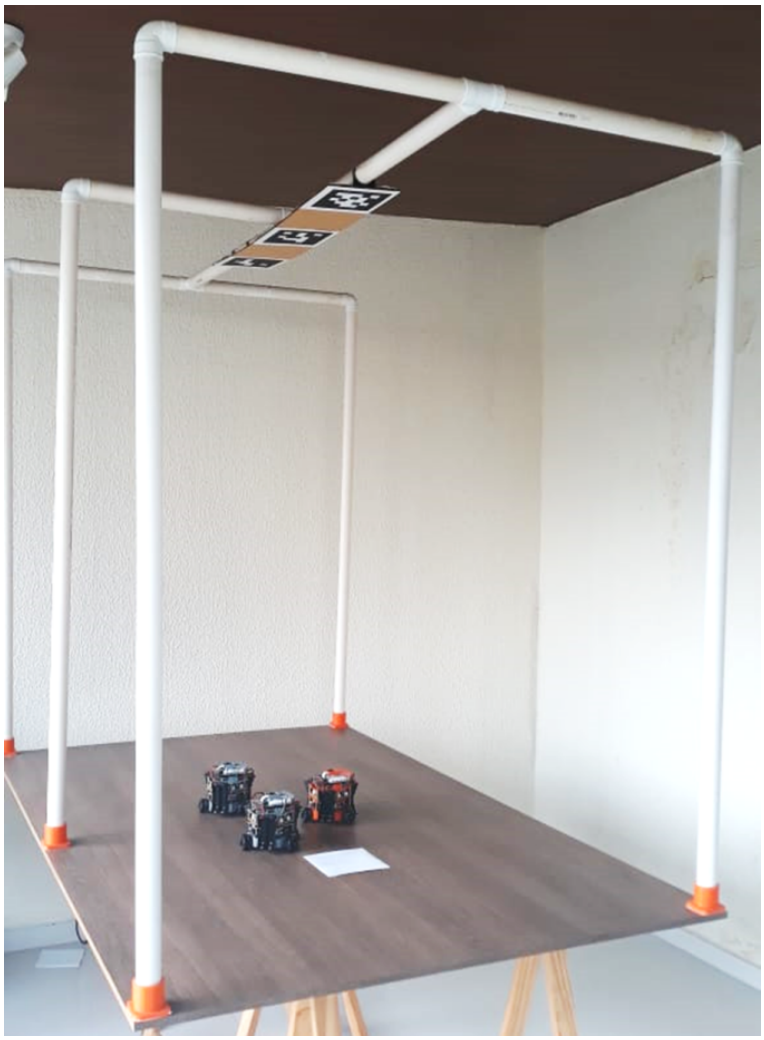

Figura 4.3: Estrutura para experimentos

A área de trabalho da mesa tem largura de 120cm e 200cm de comprimento. A altura entre a superfície da mesa e as tags ArUco é de 150cm.

Ao executar o pacote fiducials_slam no ROS, isso produzirá uma exibição como mostrado na Figura 4.4.

O painel inferior esquerdo mostra a visão atual da câmera. Isso é útil para determinar se a detecção do fiducial está sendo efetiva. O painel direito mostra o mapa de fiduciais enquanto o mesmo está sendo construído. Cubos vermelhos representam fiduciais que estão atualmente à vista da câmera. Cubos verdes representam fiduciais que estão no

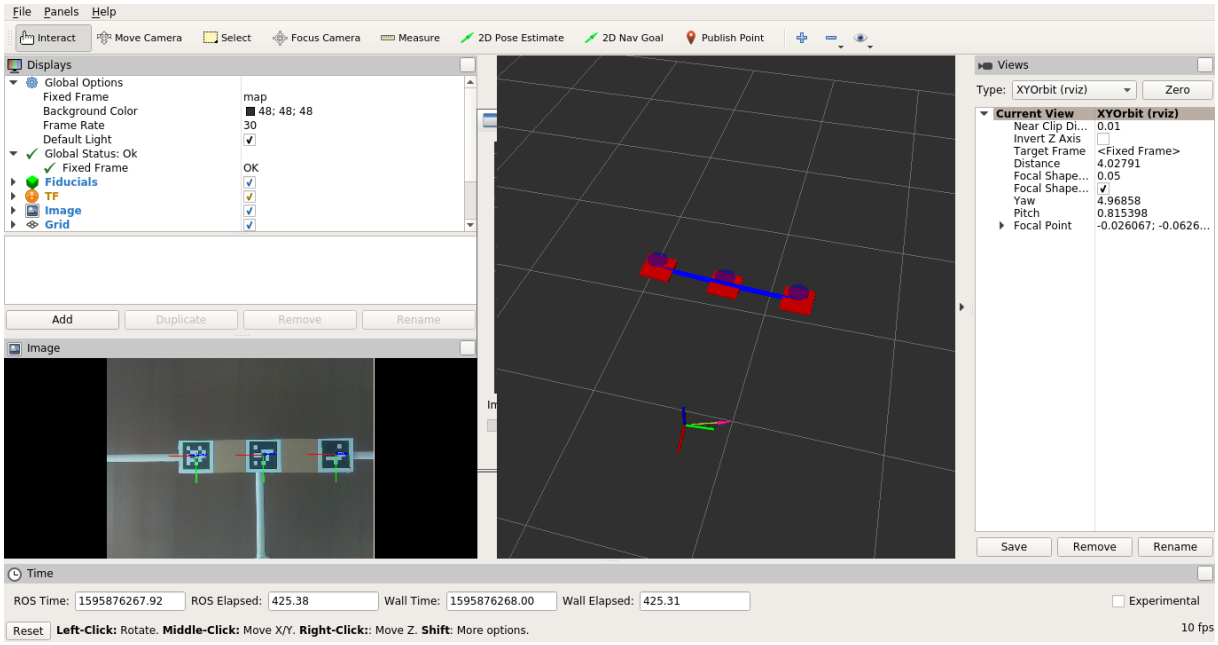

Figura 4.4: SLAM em execução no Rviz

mapa, mas não estão atualmente na exibição da câmera. As linhas azuis mostram pares de fiduciais conectados que foram observados na visualização da câmera ao mesmo tempo. A robustez do mapa é aumentada com um alto grau de conectividade entre os fiduciais. Se o mapeamento estiver funcionando, o layout espacial dos fiduciais no mapa corresponderá ao layout dos marcadores físicos.

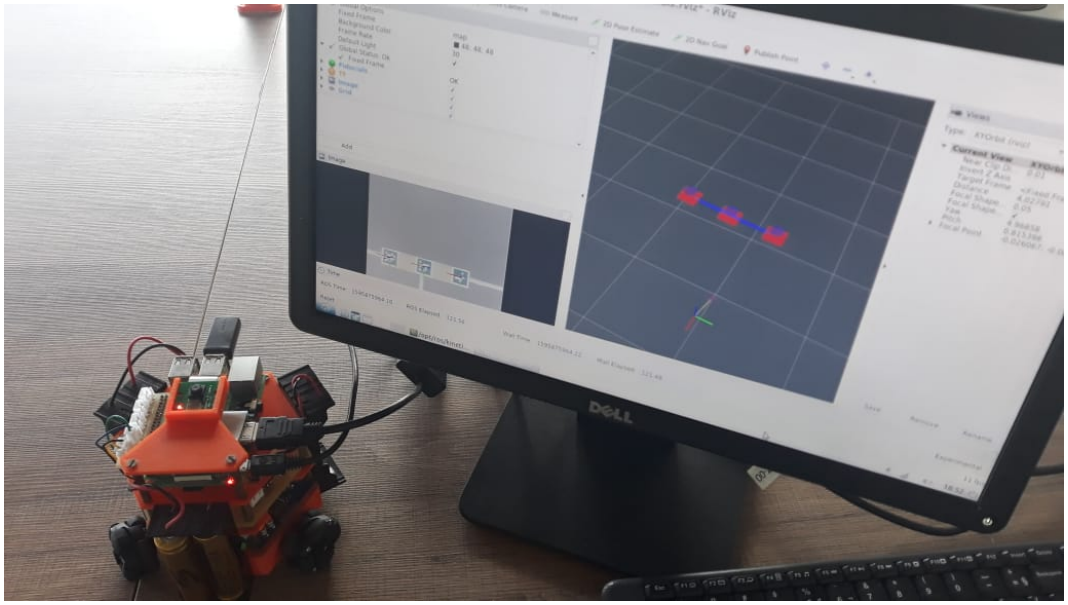

Figura 4.5: Omnino executando SLAM no Rviz

Na Figura 4.5 é possível verificar em tempo real a atualização do SLAM, embarcado a plataforma Omnino.

4.3 Arquitetura de Controle Omnino

A Figura 4.6 ilustra a arquitetura geral de integração do ROS do sistema Omnino. O ROS foi configurado para operar localmente no robô de forma totalmente embarcada.

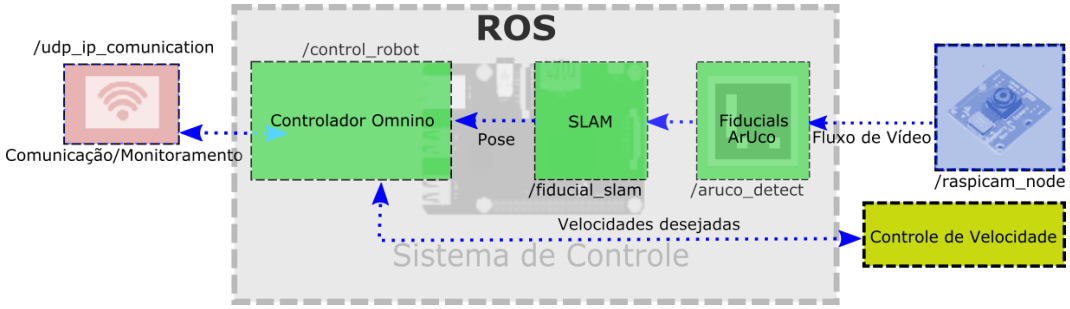

Figura 4.6: Arquitetura ROS

Fluxo de vídeo

Utilizando a RaspberryCam, o nó /raspicam_node é responsável pela captura e envio de imagens para processamento de dados realizado pelo nó /aruco_detect.

Para realizar a execução do nó da câmera do Raspberry Pi o seguinte comando deve ser executado:

raspicam_node camerav1_1280x720.launch

As configurações de captura da câmera podem ser modificados de forma dinâmica através da seguinte operação:

rqt_image_view

Onde, contraste, brilho e saturação são alguns dos parâmetros que podem ser ajustados para a geração das imagens capturadas pela câmera. Mais informações podem ser obtidas no repositório https://github.com/UbiquityRobotics/raspicam_node.

Fiducials Aruco

O nó /aruco_detect, realiza a detecção das tags com padrão ArUco nas imagens enviadas pela RaspberryCam. Usando a biblioteca ArUco, códigos binários são gerados a partir de um dicionário da biblioteca OpenCV. Os padrões de pixeis são identificados e sua posição de centro é encontrada. Se as dimensões da tag são conhecidas, a pose dos marcadores pode ser calculada. Para a criação do das Tag's Aruco o seguinte comando deve ser realizado:

rosrun aruco_detect create_markers.py 100 112 fiducials.pdf

Após as Tag's serem impressas, elas podem ser afixadas no ambiente. Não é necessário que as Tag's sejam colocadas em padrões específicos, mas a densidade deve ser tal que dois ou mais marcadores possam ser vistos pela câmera no robô, para que o mapa possa ser construído. Dois nós devem ser executados, aruco_detect, que lida com a detecção

dos fiducials e combina as estimativas de pose fiducial, e fiducial_slam, que constrói o mapa e faz uma estimativa da posição do robô.

SLAM

O nó /fiducial_slam, combina as posições de cada uma das poses enviadas pelo nó /aruco_detect e os combina para construir o mapa e dá a estimativa de posição e angulo em que o robô se encontra. O comando a seguir produzirá a representação gráfica da distribuição das Tag's Aruco:

roslaunch fiducial_slam fiducial_rviz.launch

Estando o mapeamento em funcionamento, o layout espacial dos fiduciais no mapa corresponderá ao layout dos marcadores reais. Mais informações podem ser obtidas no repositório http://wiki.ros.org/fiducials.

Controlador

O nó /control_robot, recebe a pose do robô e calcula as velocidades que devem ser empregadas em cada uma das rodas, utilizando a equação cinemática inversa vista anteriormente. O Controlador é responsável pelo envio dos comando seriais, via RS232, para os controladores PID de velocidade dos motores, determinado o endereço, velocidade e sentido de rotação de cada uma das rodas do Omnino.

Comunicação

O nó /udp_ip_comunication é responsável pela troca de informações, via wifi, entre o robô e outros agentes que estão envolvidos na mesma tarefa, cada agente tem um sistema de cliente/servidor, onde recebe e envia suas posições a cada solicitação gerada. Estes dados também podem ser acessados por outros dispositivos que estejam inseridos na mesma rede, como um PC para a geração de gráficos de deslocamento dos agentes e coleta e envio de dados para o sistema.

Nesta etapa de desenvolvimento do projeto, foi percebido que a frequência de atualização da pose do agente é de 2Hz, em estudos futuros, técnicas para aumentar a eficiência do sistema de captura e processamento de imagens podem melhorar o desempenho do sistema de localização.

4.4 Plataforma de simulação Gazebo

É disponibilizada uma plataforma para a simulação do robô Omnino a fim de poder usufruir da integração com o ROS em ambientes de sala de aula ainda no processo de desenvolvimento. Busca-se, então, que as técnicas de controle ou navegação possam ser facilmente transferíveis ao robô real após testes em ambiente virtual. Assim, mesmo em

ambientes fora de laboratório, os alunos/pesquisadores podem trabalhar no desenvolvimento exigindo baixo recurso.

A arquitetura de software baseada em ROS torna transparente a utilização do mesmo código de Controle de alto nível no ambiente de simulação. A Figura 4.7 mostra em verde o mesmo bloco/código utilizado na arquitetura de simulação. A Comunicação/Monitoramento pode ser realizada pelo próprio terminal do computador executando a simulação. As Velocidades desejadas são tratadas pelo pacote aberto ROS Control e a simulação é realizada na plataforma aberta Gazebo (Koenig and Howard, 2004), que fornece os dados de Posição/Orientação ao Controle de alto nível. O ambiente fornecido é de código aberto e disponível no repositório: https://github.com/EASY-SPARC/omnino_gazebo

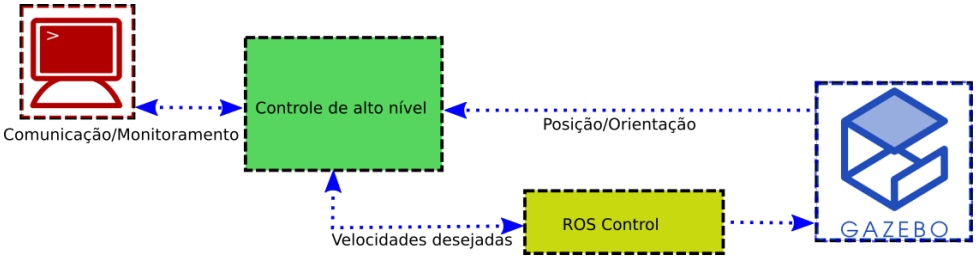

Figura 4.7: Simulação do Omnino em Gazebo

Possibilitar a simulação desempenha um papel crítico na aprendizagem e pesquisa como uma ferramenta de rápido desenvolvimento. O ambiente de simulação Gazebo permite recriar uma descrição de mundo num ambiente 3D, compondo múltiplos robôs e objetos. A Figura 4.8 mostra uma comparação entre os robôs num experimento real (Figura 4.8a) e no ambiente simulado (Figura 4.8b).

A transparência fornecida pelo ambiente de simulação permite fácil desenvolvimento dos algoritmos de controle e fácil porte do código para um ambiente real. Isto permite ampla utilização em sala de aula e de desenvolvimento de trabalhos por parte de discentes em projeto que podem, em determinados estágios, ser testados num ambiente real.

A simulação fornecida em Gazebo é compatível com os demais pacotes disponibilizados pelo ROS. Por exemplo, o módulo de navegação[1], que pode ser utilizado em disciplinas de navegação de robôs, pode ser aplicado como mostrado na Figura 4.9.

[1] http://wiki.ros.org/navigation

(a) Experimento real com três Omninos.

(b) Simulação com três Omninos.

Figura 4.8: Comparação entre experimento real e simulação

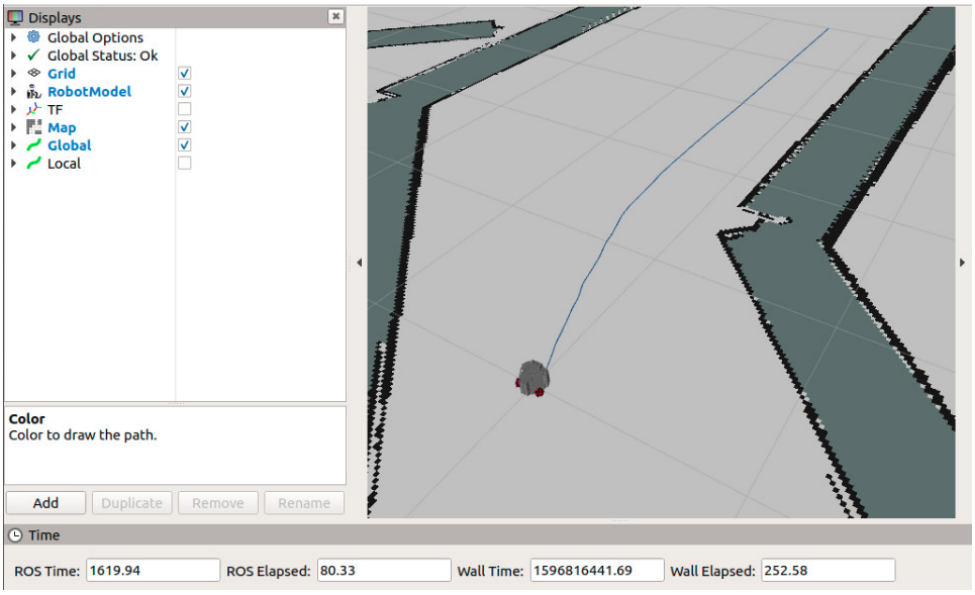

Figura 4.9: Navigation stack para Omnino

Capítulo 5

Resultados experimentais

Neste capítulo serão apresentados os resultados obtidos a partir da construção e implementação de todas as etapas discutidas anteriormente. Para cada etapa definida e descrita, são expostos os resultados obtidos a fim de demonstrar o funcionamento da plataforma de robótica móvel Omnino.

Para os resultados mostrados a seguir, a central de processamento de dados da plataforma Omnino foi o próprio Raspberry Pi 3 B+ embarcado. Todos os gráficos e dados foram gerados no sistema ROS, com o auxílio da biblioteca matplotlib do Python.

5.1 Teste de Holonomia

Para demonstrar a operacionalidade do modelo cinemático do robô Omnino, foram realizados testes no qual uma variedade de trajetórias foram impostas ao modelo (Figura 5.1) para que assim fosse possível demonstrar sua holonomia translacional num plano. A pla-

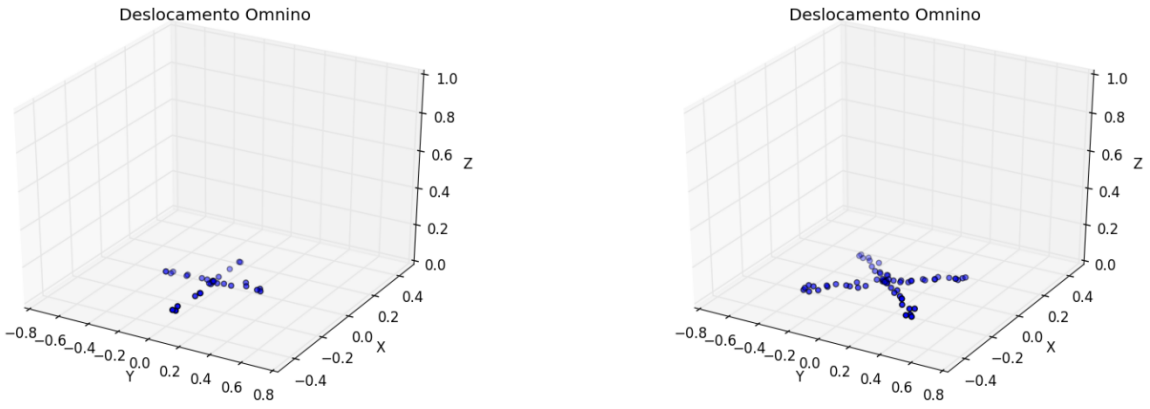

Figura 5.1: Omnino, teste de holonomia

taforma Omnino foi capaz de se deslocar sobre a superfície de trabalho sem a necessidade de alteração do seu ângulo, demonstrando assim sua capacidade holonômica.

5.2 Controle de Translação

Com o objetivo de avaliar a eficiência do algoritmo de controle de deslocamento da plataforma Omnino, esta seção mostra os resultados da aplicação do controlador ao agente autônomo. Todos os ensaios foram realizados em ambiente real e os dados coletados foram gerados pelo software de localização embarcado no Omnino.

Figura 5.2: Omnino controle de movimento

O controle de deslocamento do agente Omnino tem como objetivo fazer com que o agente se desloque de qualquer ponto da área de trabalho (*Start*) até um ponto desejado (*Goal*), controlando sua posição e ângulo final, a Figura 5.2 demonstra este deslocamento, onde o robô realiza o ajuste de sua trajetória com ângulo de chegada até o quadrado em branco (*Goal*).

O erro entre a posição atual do robô e a posição desejada é calculada como

$$\begin{aligned} e_x &= x^* - x, \\ e_y &= y^* - y, \\ e_\theta &= \theta^* - \theta. \end{aligned} \tag{5.1}$$

Em seguida a ação proporcional é calculada utilizando

$$\begin{aligned} u_x &= k_p * e_x, \\ u_y &= k_p * e_y, \\ u_\theta &= k_\theta * e_\theta. \end{aligned} \tag{5.2}$$

Durante os experimentos os valores de ganho proporcional para o controle de translação e de ângulo do robô foram: $k_p = 70$ e $k_\theta = 0.75$.

A ação proporcional é inserida no modelo cinemático do robô Omnino para assim gerar as velocidades necessárias em cada um dos motores para seu deslocamento de acordo com a Equação 5.3.

$$\begin{aligned} vel_{m1} &= -0.66u_x + 0.33u_\theta, \\ vel_{m2} &= 0.33u_x - 0.57u_y + 0.33u_\theta, \\ vel_{m3} &= 0.33u_x + 0.57u_y + 0.33u_\theta. \end{aligned} \tag{5.3}$$

Foram realizados uma sucessão de ensaios de controle de translação, colocando o robô em posições e ângulos diferentes com o objetivo de alcançar um ponto desejado sobre a área de trabalho, os erros de posicionamento final nos experimentos em cada uma das coordenadas do robô pode ser verificados Tabela 5.1.

Tabela 5.1: Erro nas coordenadas de posicionamento final do robô

	Erro Final X (mm)	Erro Final Y (mm)	Erro Final Ângulo (graus)
Experimento 1	4.221529365	19.102151394	4.672607422
Experimento 2	5.162507892	9.161417484	0.631164551
Experimento 3	2.79691875	0.265414715	1.641418457
Experimento 4	5.272254944	5.717818737	1.564788818
Experimento 5	9.516588748	7.366840839	-4.373565674
Experimento 6	1.028954387	10.483269691	-3.332977295
Experimento 7	16.460045576	3.667886257	4.205566406

Os gráficos a seguir demonstram uma série de experimentos que foram realizados com a plataforma. Em todos eles, é possível perceber que, independente do ponto inicial em que o robô é posicionado, o controlador sempre é capaz de levar o agente até o *Goal*. Também são demonstrados graficamente a redução dos erros no deslocamento e o controle do ângulo do robô até alcançar o objetivo final da tarefa.

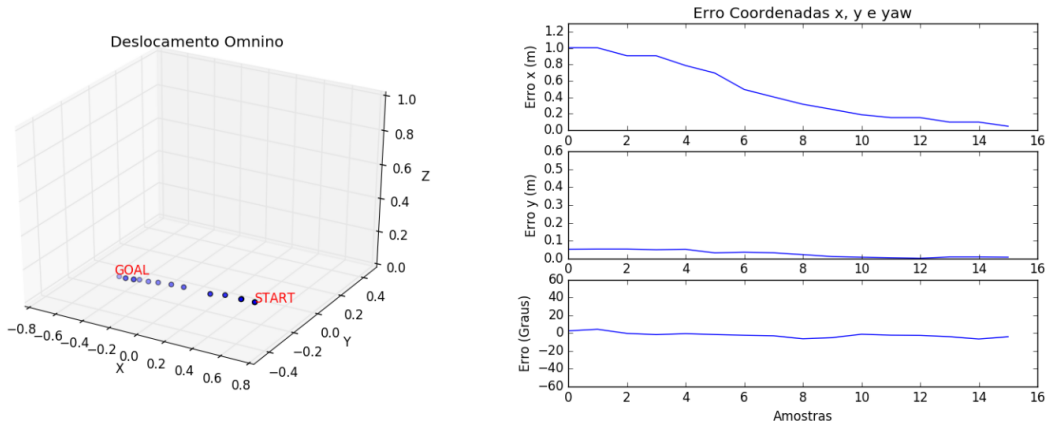

Figura 5.3: Controle de deslocamento experimento - 01

Os valores alcançados pelo controlador proporcional foram bastante satisfatórios, visto que o posicionamento do robô demonstrou um erro máximo de aproximadamente 2cm nas

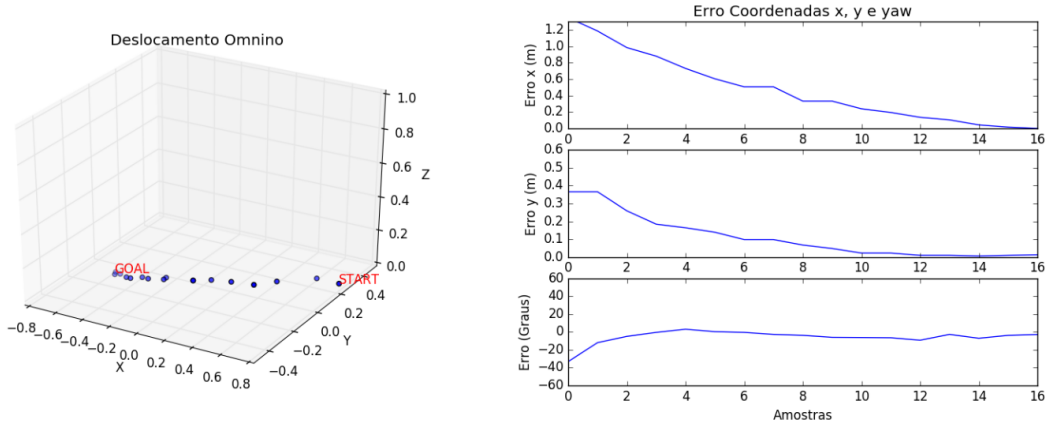

Figura 5.4: Controle de deslocamento experimento - 02

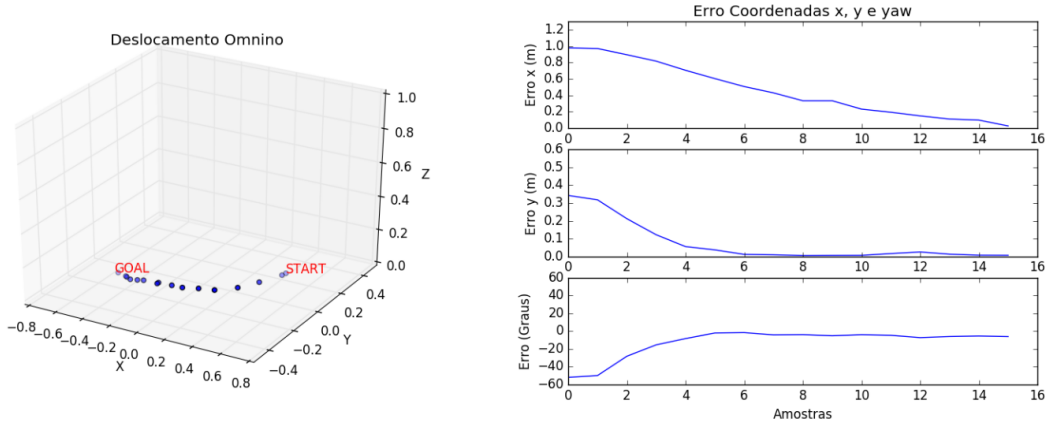

Figura 5.5: Controle de deslocamento experimento - 03

coordenadas de translação (X e Y) e um erro máximo de 5 graus em seu ângulo final. Com a implementação de controladores mais robustos é bastante provável que esses erros sejam ainda mais atenuados.

Um canal de vídeos com a execução das tarefas realizadas pelo Omnino foi criado com o intuito de demonstrar os experimentos realizados: https://www.youtube.com/channel/UC-jy1UgepSGjuAsUHcRQUBA.

5.3 Controle de formação por consenso

Técnicas de consenso têm como objetivo promover a concordância dos robôs de um grupo com relação a determinada informação. O protocolo de consenso aplicado em cada robô é dado pela leis de controle vistas a seguir. Cada agente calcula o erro δ_{x_i}, δ_{y_i} e δ_{θ_i} referente ao padrão de formação definido pelas coordenadas ($x_1{}^*$,$y_1{}^*$,$\theta_1{}^*$) de acordo com

$$\begin{aligned} \delta_{x_i} &= x_1{}^* - x_1, \\ \delta_{y_i} &= y_1{}^* - y_1, \\ \delta_{\theta_i} &= \theta_i{}^* - \theta_i. \end{aligned} \tag{5.4}$$

Desta forma, o protocolo de consenso aplicado para garantir o alcance da formação é dado por

$$\begin{aligned} u_{x_i} &= k_p \sum_{j \in \mathcal{N}_i} (\delta_{x_j} - \delta_{x_i}), \\ u_{y_i} &= k_p \sum_{j \in \mathcal{N}_i} (\delta_{y_j} - \delta_{y_i}), \\ u_{\theta_i} &= k_\theta \sum_{j \in \mathcal{N}_i} (\delta_{\theta_j} - \delta_{\theta_i}) \end{aligned} \tag{5.5}$$

sendo $\mathcal{N}_i$ o conjunto de agentes vizinhos no grafo de topologia de rede, isto é, aqueles com arestas direcionadas ao agente i.

O grafo na Figura 5.6 indica a topologia de rede utilizada nos exemplos a seguir, considerando um agente líder que não possui informação de vizinhos, e os agentes seguidores, que a partir da posição dos agentes vizinhos e da aplicação dos protocolos dados especificamente por

$$\begin{aligned} u_{x_2} &= k_p \left(\delta_{x_1} - 2\delta_{x_2} + \delta_{x_3}\right), \\ u_{y_2} &= k_p \left(\delta_{y_1} - 2\delta_{y_2} + \delta_{y_3}\right), \\ u_{\delta_2} &= k_\theta \left(\delta_{\theta_1} - 2\delta_{\theta_2} + \delta_{\theta_3}\right), \end{aligned} \tag{5.6}$$

$$\begin{aligned} u_{x_3} &= k_p \left(\delta_{x_1} + \delta_{x_2} - 2\delta_{x_3}\right), \\ u_{y_3} &= k_p \left(\delta_{y_1} + \delta_{y_2} - 2\delta_{y_3}\right), \\ u_{\delta_3} &= k_\theta \left(\delta_{\theta_1} + \delta_{\theta_2} - 2\delta_{\theta_3}\right) \end{aligned} \tag{5.7}$$

permitem atingir a formação desejada. Os ganhos utilizados no protocolo de controle de consenso dos robôs seguidores foram: $k_p = 100$ para translação e $k_\theta = 0.75$ para o controle do ângulo do robô. O líder será controlado livremente sem influência do protocolo de consenso, pois não tem agentes vizinhos de acordo com o grafo na Figura 5.6, i.e. $\mathcal{N}_1 = \emptyset$.

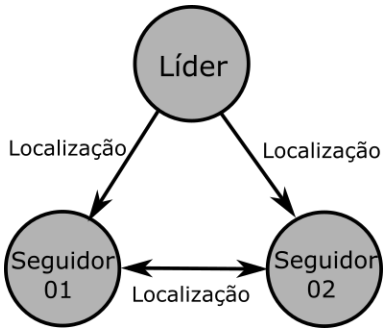

Figura 5.6: Grafo da topologia de rede utilizada no experimento de formação baseada em consenso.

Para a execução dos algoritmos de controle de formação dos robôs, foi utilizado a biblioteca de comunicação UDP provida pelo ROS. Neste projeto o objetivo da utilização do consenso foi a realizar uma formação triangular entre três plataformas Omnino.

Este experimento foi dividido em duas partes, a primeira destaca o controle da formação, onde o robô líder permanece parado enquanto os seguidores se colocam nas posições apropriadas para a criação da geometria desejada. A Figura 5.7 representa graficamente este experimento, onde é possível acompanhar o reposicionamento dos robôs seguidores durante a execução do algoritmo de consenso.

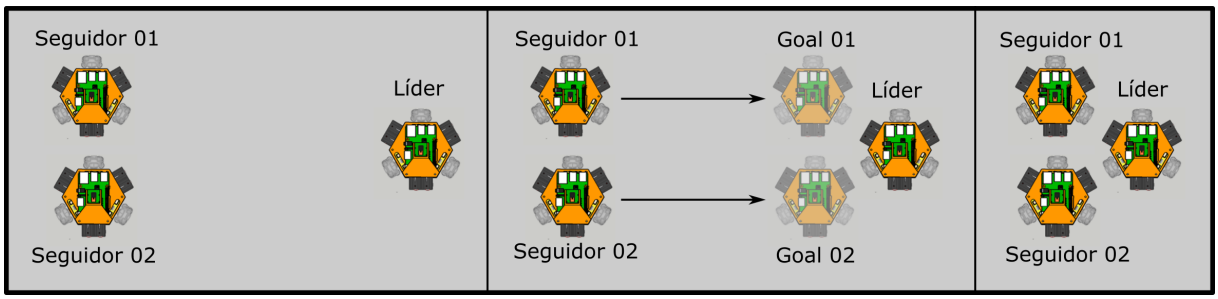

Figura 5.7: Formação por consenso: robôs fora de formação, robôs calculam o ponto de destino, robôs executam deslocamento.

A segunda etapa do experimento consiste em realizar o deslocamento de todo o enxame mantendo a formação, o robô líder realiza o deslocamento para o ponto de destino desejado, enquanto os robôs seguidores ajustam suas posições pelo algoritmo de consenso, afim de manter a formação do grupo até que o robô líder alcance o ponto final.

No gráfico da Figura 5.9, pode-se acompanhar o deslocamento de cada um dos robôs no primeiro experimento de formação, o robô líder esta representado em azul, o seguidor01 em verde e o seguidor02 em vermelho.

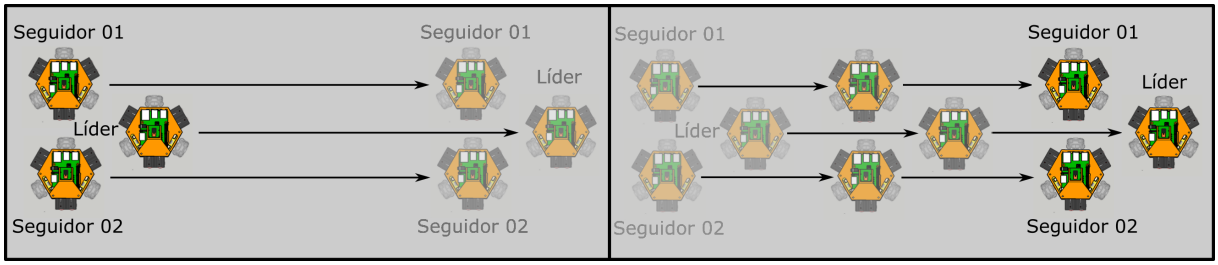

Figura 5.8: Deslocamento por Consenso: robôs em formação, líder e seguidores em movimento mantendo a formação.

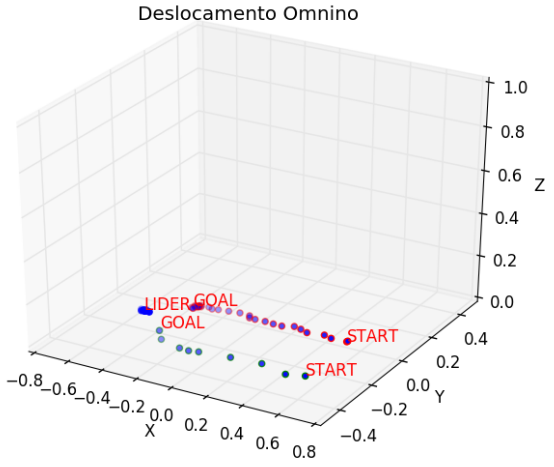

Figura 5.9: Controle de formação por consenso

No gráfico da Figura 5.10, verifica-se o deslocamento de cada um dos robôs no segundo experimento de deslocamento, o robô líder esta representado em azul, o seguidor01 em verde e o seguidor02 em vermelho.

Nesta etapa do projeto foi possível perceber que a plataforma Omnino foi capaz de realizar as tarefas de formação e deslocamento do enxame utilizando o algoritmo de consenso. Alguns atrasos de comunicação foram percebidos durante a execução das tarefas, mas isso ocorreu principalmente pelo fato do sistema de processamento do enxame ser totalmente descentralizado, a aplicação de protocolos de comunicação mais rápidos em trabalhos futuros podem contribuir para um melhor desempenho nesta tarefa.

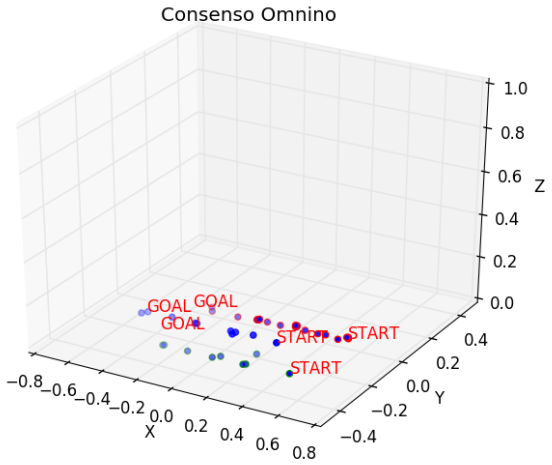

Figura 5.10: Controle de movimento por consenso

Conclusão

Neste trabalho foi apresentado o desenvolvimento da plataforma de robô móvel holonômico Omnino, que tem como objetivo principal ser uma ferramenta de apoio a pesquisa e educação na área de robótica e afins.

O projeto teve seu modelo cinemático implementado e testado em experimentos reais, demonstrando sua holonomia, a sua a capacidade de realização de tarefas de translação e localização autônomas de maneira descentralizada.

A técnica de Ziegler-Nichols foi utilizada para a realização da sintonia do sistema de controle PID de velocidade dos motores. Os controladores de velocidade pretendem ser versáteis para possibilitar sua utilização em outros projetos, desde modelos robóticos diferenciais e modelos holonômicos de quatro rodas, podendo inclusive ser utilizados em laboratório nas aulas de Controle, onde o aluno pode implementar técnicas e modelagem ao sistema e verificar sua resposta de maneira experimental.

A capacidade do robô se localizar de forma autônoma no ambiente, com o auxílio da visão computacional trouxe flexibilidade para o sistema, pois, em muitos casos, plataformas robóticas necessitam de agentes externos que determinem sua localização para que então o mesmo possa realizar suas tarefas.

Por ser uma plataforma completa, o aluno pode, através de interfaces de vídeo (HDMI) e com um teclado/mouse, ou por ssh, controlar e editar os códigos utilizados para o controle de alto nível, tudo considerando a compatibilidade com o sistema ROS.

Nos experimentos reais o Omnino se mostrou um plataforma bastante adequada e flexível para aplicação em enxames, foram realizadas atividades no controle de formação utilizado a técnica de consenso.

As possibilidades de expansão e alteração da plataforma são grandes e, entre as possíveis modificações, estão sendo estudadas algumas no sentido de adicionar funcionalidades como a detecção de obstáculos, adição de ferramentas de manipulação, substituição do sistema de visão monocular pelo sistema de visão estéreo, melhorias em seu sistema de troca de dados, uso de outras técnicas de localização por visão etc. Enfim, o Omnino é uma plataforma com grande potencial para aplicação em sala de aula e laboratórios de pesquisa. Por ser totalmente aberta, seus usuários estão livres para realizar modificações e melhorias para que assim se tenha uma ferramenta eficiente, com acesso democrático e livre!

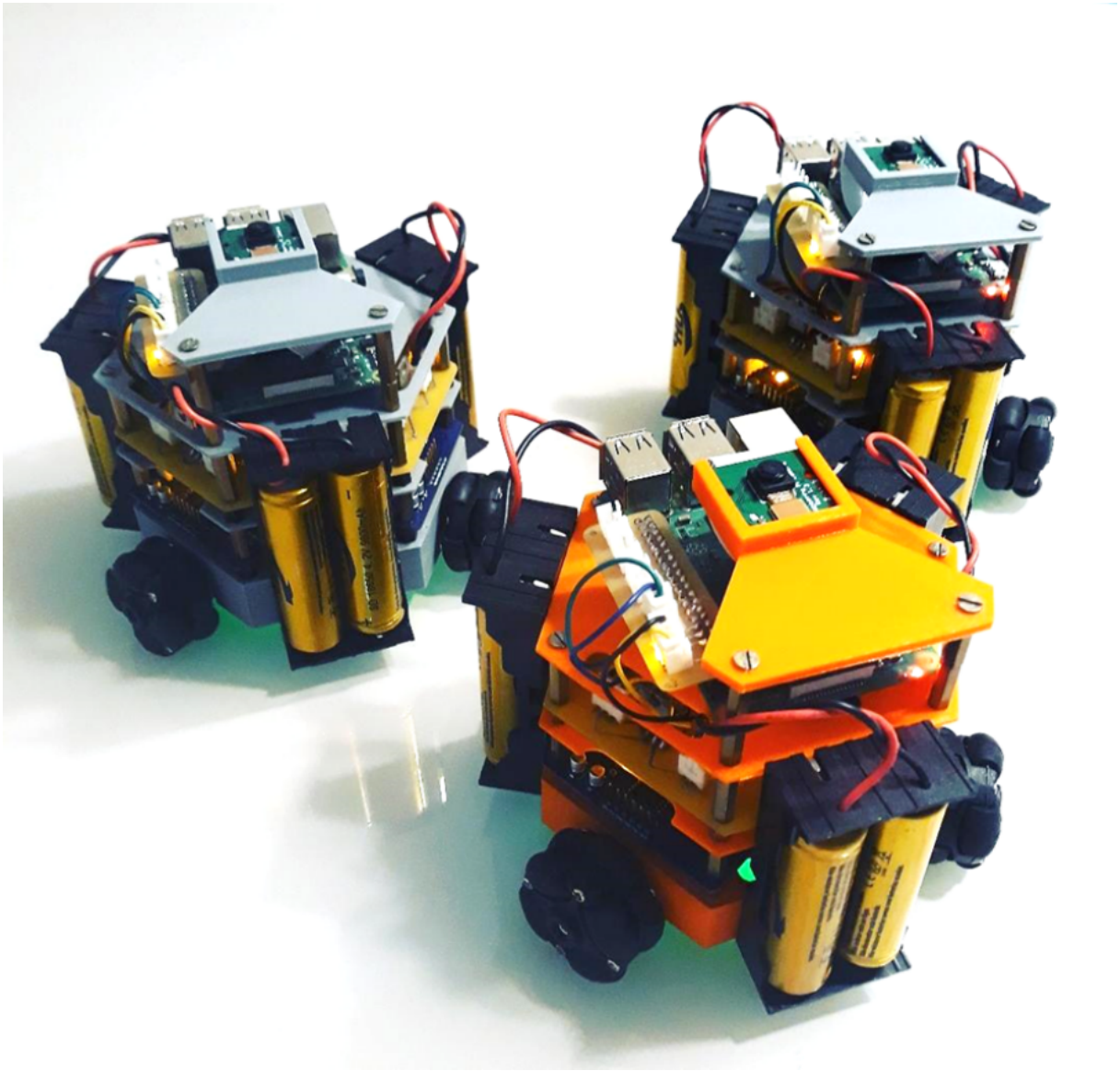

Figura 5.11: Plataforma de Pesquisa e Ensino OMNINO

5.4 Trabalho submetido durante o desenvolvimento do trabalho

- José Henrick Viana Ramalho, Erick Barboza, Heitor Savino. Omnino: Plataforma Aberta de Robô para Educação e Pesquisa. IX Congresso Brasileiro de Informática na Educação (CBIE 2020), Simpósio Brasileiro de Informática na Educação (SBIE 2020), 2020.

Bibliografia

Altshuler, Y., Yanovsky, V., Wagner, I. A., and Bruckstein, A. M. (2006). Swarm intelligence—searchers, cleaners and hunters. *Swarm Intelligent Systems*, pages 93–132.

Anderson, C., Theraulaz, G., and Deneubourg, J.-L. (2002). Self-assemblages in insect societies. *Insectes sociaux*, 49(2):99–110.

Ardakani, E. S., Ebel, H., and Eberhard, P. (2017). Transporting an elastic plate using a group of swarm mobile robots. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1393–1398. IEEE.

Arnold, R., Carey, K., Abruzzo, B., and Korpela, C. (2019). What is a robot swarm: A definition for swarming robotics. In *10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 74–81. IEEE.

Bacik, J. (2017). Autonomous flying with quadrocopter using fuzzy control and aruco markers. *Intelligent Service Robotics*, 3(10):185–194.

Bayındır, L. (2016). A review of swarm robotics tasks. *Neurocomputing*, 172:292–321.

Borenstein, J. and Feng, L. (1996). Gyrodometry: A new method for combining data from gyros and odometry in mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 423–428. IEEE.

Brooks, R. A. et al. (1999). *Cambrian intelligence: The early history of the new AI*. MIT press.

Cao, Y. U., Fukunaga, A. S., and Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7–27.

Caprari, G. and Siegwart, R. (2005). Mobile micro-robots ready to use: Alice. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3295–3300. IEEE/RSJ.

Chen, J., Gauci, M., Li, W., Kolling, A., and Groß, R. (2015). Occlusion-based cooperative transport with a swarm of miniature mobile robots. *IEEE Transactions on Robotics*, 31(2):307–321.

Dimeas, I., Petras, I., and Psychalinos, C. (2017). New analog implementation technique for fractional-order controller: a dc motor control. *International Journal of Electronics and Communications*, pages 192–200.

Doroftei, I., Grosu, V., and Spinu, V. (2007). *Omnidirectional mobile robot-design and implementation*. NTECH Open Access Publisher.

Dudek, G. and Jenkin, M. (2010). *Computational principles of mobile robotics*. Cambridge university press.

Groß, R., Mondada, F., and Dorigo, M. (2006). Transport of an object by six pre-attached robots interacting via physical links. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1317–1323. IEEE.

Han, V., J. M. J., and H., J. J. (2008). Comparative study on the educational use of home robots for children. *JIPS*, 4:159–168.

Kanojiya, R. G. and Meshram, P. M. (2012). Optimal tuning of pi controller for speed control of dc motor drive using particle swarm optimization. In *International conference on Advances in Power Conversion and Energy Technologies (APCET)*, pages 1–6. IEEE.

Kernbach, S., Thenius, R., Kernbach, O., and Schmickl, T. (2009). Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system. *Adaptive Behavior*, 17(3):237–259.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154. IEEE/RSJ.

Kube, C. R. and Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and autonomous systems*, 30(1-2):85–101.

Kurfess, T. R. (2018). *Robotics and automation handbook*. CRC press.

Le Goc, M., Kim, L. H., Parsaei, A., Fekete, J.-D., Dragicevic, P., and Follmer, S. (2016). Zooids: Building blocks for swarm user interfaces. In *29th Annual Symposium on User Interface Software and Technology*, pages 97–109. ACM.

Li, X. and Zell, A. (2009). Motion control of an omnidirectional mobile robot. In *Informatics in Control, Automation and Robotics*, pages 181–193. Springer.

Liu, Y., Wu, X., Zhu, J. J., and Lew, J. (2003). Omni-directional mobile robot controller design by trajectory linearization. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 4, pages 3423–3428. IEEE.

Marcolino, L. S. and Chaimowicz, L. (2009). Traffic control for a swarm of robots: Avoiding group conflicts. *nternational Conference on Intelligent Robots and Systems*, 61(9):1949–1954.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65. IEEE.

Neumann, P., Kohlhoff, H., Hüllmann, D., Lilienthal, A. J., and Kluge, M. (2017). Bringing mobile robot olfaction to the next dimension—uav-based remote sensing of gas clouds and source localization. *International Conference on Robotics and Automation (ICRA)*, pages 3910–3916.

Ogata, K. and Severo, B. (1998). *Engenharia de controle moderno*. Prentice Hall do Brasil.

Purcell, M., Gallo, D., Packard, G., Dennett, M., Rothenbeck, M., Sherrell, A., and Pascaud, S. (2011). Use of remus 6000 auvs in the search for the air france flight 447 in oceans'11 mts/ieee kona. *IEEE*, pages 1–7.

Rezeck, P. A., Azpurua, H., and Chaimowicz, L. (2017). Hero: An open platform for robotics research and education. In *Latin American Robotics Symposium (LARS) and Brazilian Symposium on Robotics (SBR)*, pages 1–6. IEEE.

Savino, H. J. (2016). *New methods for consensus in multiagent systems*. PhD thesis, UFMG.

Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Stonier, T. and Thornton, P. (1982). Forecasts of trends in the post-industrial society. In *Social forecasting for company planning*, pages 255–271. Palgrave Macmillan.

Tavakoli, M., Viegas, C., Marques, L., Pires, J. N., and De Almeida, A. T. (2013). Omniclimbers: Omni-directional magnetic wheeled climbing robots for inspection of ferromagnetic structures. *Robotics and Autonomous Systems*, 61(9):997–1007.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128:99–141.

Wei, H., Cai, Y., Li, H., Li, D., and Wang, T. (2010). Sambot: A self-assembly modular robot for swarm robot. In *IEEE International Conference on Robotics and Automation*, pages 66–71. IEEE.

Wolf, D. F. and Sukhatme, G. S. (2005). Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19:53–65.

Zou, J., Lin, Y., Ji, C., and Yang, H. (2018). A reconfigurable omnidirectional soft robot based on caterpillar locomotion. *Soft robotics*, 5(2):164–174.