Universidade de Brasília

Faculdade do Gama

Controle de Trajetória e Gerenciamento de Estratégias Baseado em Aprendizado por Demonstração aplicado a Futebol de Robôs Móveis

Matheus de Sousa Luiz

TRABALHO DE CONCLUSÃO DE CURSO

ENGENHARIA ELETRÔNICA

Brasília

2025

Universidade de Brasília

Faculdade do Gama

Controle de Trajetória e Gerenciamento de Estratégias Baseado em Aprendizado por Demonstração aplicado a Futebol de Robôs Móveis

Matheus de Sousa Luiz

Monografia submetida ao curso de Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do grau de Engenheiro Eletrônico.

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Coorientador: M.Sc. Mario Andrés Pastrana Triana

Brasília

2025

FICHA CATALOGRÁFICA

Sousa Luiz, Matheus de.

Controle de Trajetória e Gerenciamento de Estratégias Baseado em Aprendizado por Demonstração aplicado a Futebol de Robôs Móveis / Matheus de Sousa Luiz; orientador Daniel Mauricio Muñoz Arboleda; coorientador Mario Andrés Pastrana Triana. -- Brasília, 2025.

72 p.

Trabalho de Conclusão de Curso (Engenharia Eletrônica) -- Universidade de Brasília, 2025.

1. Aprendizado por Demonstração. 2. Controle de Trajetória. 3. Futebol de Robôs. 4. Robótica Móvel. 5. Very Small Size Soccer.I. Mauricio Muñoz Arboleda, Daniel, orient. II. Andrés Pastrana Triana, Mario, coorient. III. Título.

Universidade de Brasília
Faculdade do Gama

Controle de Trajetória e Gerenciamento de Estratégias Baseado em Aprendizado por Demonstração aplicado a Futebol de Robôs Móveis

Matheus de Sousa Luiz

Monografia submetida ao curso de Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do grau de Engenheiro Eletrônico.

Trabalho aprovado. Brasília, 24 de fevereiro de 2025:

Prof. Dr. Daniel Mauricio Muñoz Arboleda,
UnB/FGA
Orientador

Prof. Dr. Roberto de Souza Baptista,
UnB/FGA
Examinador interno

Prof. Dr. Guillermo Alvarez Bestard,
UnB/FGA
Examinador interno

Este trabalho é dedicado à minha família que
sempre me deu forças para continuar tentando.
Me voliv tumen.

Agradecimentos

Primeiramente, agradeço a minha família por tudo que fizeram por mim. Agradeço à minha irmã Milene pelos conselhos e constante incentivo e apoio, ao meu irmão Emiliano, que me presenteou com meu primeiro kit de eletrônica e me buscou todas as noites no começo do meu curso técnico e à minha irmã Melissa que sempre foi minha referência na vida acadêmica. Aos meus pais, Emiliano e Simone, sou imensamente grato por todas as batalhas travadas para que eu pudesse me tornar quem sou hoje. Aos meus avós, Emiliano, Lourdes, Hamilton e Sirlei e aos meus irmãos de coração, Leonardo e Marcella agradeço por sempre acreditarem e estarem comigo. Aos meus sobrinhos, Alice e Arthur, por me lembrarem da beleza de ver o mundo com os olhos de uma criança. Um agradecimento especial à minha noiva, Nadyr, que esteve ao meu lado a todos os momentos, me apoiou e cuidou de mim enquanto perdia os cabelos com este trabalho.

Aos irmãos que a vida me presenteou - Bruno, Caio, Levy, Lucas, Pedro e Thiago - e aos meus grandes amigos Alexandre, Camilo, Christian, Felipe, João, Rafael, Erick agradeço por terem tornado a vida acadêmica mais leve e suportável. Agradeço a todos os membros da Equipe de Robótica Titans, principalmente ao time de controle - Erick e Julia - por toda a ajuda no desenvolvimento deste projeto. Um agradecimento especial ao Raul e ao Luiz, que trouxeram um troféu diretamente do Peru e demonstraram até onde nossa dedicação pode nos levar. Espero ter contribuído para a equipe tanto quanto todos vocês contribuíram para minha jornada.

Sou profundamente grato ao meu coorientador, Mario Pastrana, por ter concebido a ideia deste projeto e por seu empenho incansável em torná-lo realidade; e ao meu orientador, professor Daniel, por acreditar nesta pesquisa e me guiar pelo caminho certo com sua orientação e apoio contínuo.

Por fim, expresso minha sincera gratidão aos membros da banca examinadora, professores Guillermo e Roberto, pelas valiosas sugestões e pelas correções que contribuíram para aprimorar este trabalho.

Eu não teria conseguido sem vocês.

"I have not failed.
I've just found 10,000 ways that won't work."
(Thomas A. Edison)

Resumo

O presente trabalho propõe um sistema de controle de trajetória e gerenciamento de estratégias baseado em Aprendizado por Demonstração (LfD) aplicado ao futebol de robôs móveis. A abordagem visa possibilitar que os robôs aprendam comportamentos a partir de demonstrações humanas, reduzindo a necessidade de desenvolvimento manual de algoritmos complexos. Para isso, utilizou-se redes neurais artificiais para o gerenciamento de estratégias e controladores PID para o controle de velocidade e trajetória.

Os resultados demonstraram que o gerenciador de estratégias foi capaz de abstrair corretamente os comportamentos ensinados, embora melhorias na coleta e tratamento de dados possam aumentar sua eficácia. O controlador de velocidade mostrou-se funcional, mas com limitações na estabilização do sistema. O controlador de trajetória, por sua vez, apresentou instabilidades que impedem sua aplicação prática no estado atual. A integração total do sistema não foi concluída devido a essas limitações.

Dentre os desafios enfrentados, destacam-se as dificuldades mecânicas do robô e a sintonização do controlador de trajetória sem a identificação do sistema. Como contribuição principal, o trabalho permitiu o desenvolvimento do robô Corobeu como um sistema autônomo, possibilitando futuras aplicações em controle de enxames, monitoramento de máquinas e logística automatizada.

Palavras-chave: Aprendizado por Demonstração. Controle de Trajetória. Futebol de Robôs. Robótica Móvel. Very Small Size Soccer.

Abstract

This work proposes a trajectory control and strategy management system based on Learning from Demonstration (LfD) applied to mobile robot soccer. The approach enables robots to learn behaviors from human demonstrations, reducing the need for manually developing complex rule-based control algorithms. To achieve this, artificial neural networks were used for strategy management, and PID controllers were employed for speed and trajectory control.

The results showed that the strategy manager was able to correctly abstract the taught behaviors, although improvements in data collection and processing could enhance its effectiveness. The speed controller proved functional but had limitations in system stabilization. The trajectory controller, in turn, exhibited instabilities that prevent its practical application in its current state. The full integration of the system was not completed due to these limitations.

Among the challenges faced, mechanical difficulties with the robot and tuning the trajectory controller without system identification stand out. As the main contribution, this work enabled the development of the Corobeu robot as an autonomous system, opening opportunities for future applications in swarm control, machine monitoring, and automated logistics.

Keywords: Learning from Demonstration. Trajectory Control. Robots Soccer. Mobile Robotics. Very Small Size Soccer.

Lista de figuras

Figura 1.1 Diagrama de blocos simplificado da Arquitetura proposta 17
Figura 2.1 Análise geométrica da cinemática dos robôs móveis de tração diferencial. O quadrado verde representa o corpo do robô, enquanto os retângulos azuis representam as rodas. 23
Figura 2.2 Três casos de movimento para os robôs com tração diferencial. É representado um robô com dois pontos de apoio (quadrados brancos) e duas rodas tracionadas (quadrados pretos) assim como as velocidades de cada roda (setas tracejadas) e a velocidade resultante do robô (seta completa). 24
Figura 2.3 Estágios da metodologia de aprendizado por demonstração 28
Figura 2.4 Representação genérica de um neurônio artificial. 29
Figura 2.5 Três funções de ativação mais comuns para os neurônios artificiais. . . . 29
Figura 2.6 Representação de uma rede neural MLP genérica com j entradas, k neurônios da camada de entrada, 1 camada escondida com m neurônios e n neurônios na camada de saída. 30
Figura 2.7 Exemplo de uma unidade de processamento básica de uma rede LSTM. Cada retângulo representa um neurônio com sua função de ativação sendo especificada pelo texto dentro, já os círculos representam operações ponto a ponto com os elementos podendo ser multiplicações, somas ou aplicações da tangente hiperbólica, por fim as entradas da rede são os vetores c_{t-1}, h_{t-1} e x_t enquanto as saídas são os vetores c_t e h_t. 31
Figura 3.1 Conexões eletrônicas do robô Corobeu 36
Figura 3.2 Foto do robô Corobeu desenvolvido pela Equipe de Robótica Titans. . . 37
Figura 3.3 Primeiro cenário para simulação contendo apenas um robô e uma bola. 39
Figura 3.4 Dimensões e marcações do campo de acordo com o manual de regras oficial da competição VSSS. 40
Figura 3.5 Foto do campo real desenvolvido pela Equipe de Robótica Titans. 40
Figura 3.6 Diagrama de blocos da arquitetura proposta 41
Figura 3.7 Arquitetura da rede neural de gerenciamento de estratégias 43
Figura 3.8 Bloco de controle de trajetórias isolado. Este bloco recebe um vetor de posições desejadas e as posições e orientações atuais do robô e da bola em campo e retorna um valor de velocidades desejadas para o motor esquerdo e direito. 45

Figura 3.9 Exemplificação do funcionamento do controlador de trajetórias. Os pontos verdes são as posições desejadas, as linhas azuis representam a orientação desejada e as linhas amarelas representam a orientação do robô. As linhas tracejadas são as orientações futuras do robô. 45
Figura 3.10 Arquitetura do controlador proposto para simulação utilizando o kit de desenvolvimento FPGA Zynq 7020 para acelerar o cálculo do controlador PID. 46
Figura 3.11 Laço de simulação para sintonização automática do controlador PID de controle de trajetória. 47
Figura 4.1 Lugares geométricos das raízes para os sistemas controlados dos motores direito e esquerdo. 49
Figura 4.2 Resposta ao degrau em simulação do motor direito com um valor desejado de 180 RPM. 49
Figura 4.3 Resposta ao degrau em simulação do motor esquerdo com um valor desejado de 180 RPM. 50
Figura 4.4 Resposta ao degrau do motor direito com um valor desejado de 180 RPM. 50
Figura 4.5 Resposta ao degrau do motor esquerdo com um valor desejado de 180 RPM. 51
Figura 4.6 Matriz de confusão do modelo treinado para definição de comportamento do robô. 52
Figura 4.7 Melhores trajetos implementados pelo robô em cada experimento durante a sintonização utilizando o algoritmo MFO. 53
Figura 4.8 Melhores trajetos implementados pelo robô em cada experimento durante a sintonização utilizando o algoritmo DE. 53
Figura 4.9 Resultados do controlador de trajetórias, primeira parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado. 55
Figura 4.10 Resultados do controlador de trajetórias, segunda parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado. 56
Figura 4.11 Resultados do controlador de trajetórias, terceira parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado. 56
Figura 4.12 Resultados do controlador de trajetórias, quarta parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado. 57
Figura 4.13 Resultados do controlador de trajetórias, quinta parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado. 57

Figura 4.14 Resultados do controlador de trajetórias, sexta parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado. 58
Figura 4.15 Resultados do controlador de trajetórias, sétima parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado. 58

Lista de tabelas

Tabela 2.1 Comparação dos trabalhos correlatos de desenvolvimento de técnicas de Gerenciamento de Estratégias e Seguimento de Trajetórias na categoria VSSS 34
Tabela 3.1 Parâmetros utilizados no PSO para sintonização do PID de controle de velocidade das rodas. 38
Tabela 3.2 Tabela de relação entre saída da rede neural e comportamento desejado para os robôs 43
Tabela 4.1 Métricas de avaliação do controlador PID de velocidade das rodas. 48
Tabela 4.2 Constantes do controlador PID para cada motor. 49
Tabela 4.3 Polos em malha fechada para cada motor. 49
Tabela 4.4 Relatório de classificação do modelo de redes neurais para gerenciamento de estratégias. 51
Tabela 4.5 Constantes do controlador PID obtidas por cada controlador. 53
Tabela 4.7 Constantes do controlador PID de seguimento de trajetória. 54
Tabela 4.6 Listas de reprodução com os vídeos de sintonização dos controladores. . 54

Lista de Abreviaturas e Siglas

Adam	*Adaptative Moment Estimation*
APF	*Artificial Potential Fields*
DE	*Diferential Evolution*
FSM	*Finite State Machine*
HiL	*Hardware in the Loop*
LfD	*Learning from Demonstraition*
LGR	Lugar Geométrico Das Raízes
LiPo	Polímero de Lítio
LSTM	*Long Short-Term Memory*
MFO	*Moth-Flame Optimization*
MLP	*Multi-layer Perceptron*
PID	Proporcional, Integral e Derivativo
PPO	*Proximal Policy Optimization*
PSO	*Particle Swarm Optimization*
RNA	Rede Neural Artificial
RNR	Redes Neurais Recorrentes
RRT	*Rapidly-exploring Random Tree*
SLP	*Single-layer Perceptron*
SoC	Sistem On Chip
TDP	*Team Description Paper*
VSSS	*Very Small Size Soccer*

Sumário

1 Introdução 16
1.1 Contextualização 16
1.2 Justificativa 17
1.3 Perguntas de Pesquisa 18
1.4 Objetivos 18
1.4.1 Objetivos Gerais 18
1.4.2 Objetivos Específicos 18
1.5 Organização do Trabalho 19

2 Referencial Teórico 20
2.1 Competição IEEE *Very Small Size Soccer* 20
2.1.1 Estrutura da Competição 20
2.1.2 Regras Básicas 20
2.1.3 Desafios Tecnológicos 21
2.2 Robótica Multi-Agente 21
2.3 Robôs Móveis de Tração Diferencial 22
2.4 Navegação de Robôs Móveis 23
2.4.1 Planejamento de Trajetória em Robótica móvel 23
2.4.2 Controle de Trajetória em Robótica Móvel 26
2.5 Aprendizagem por Demonstração 27
2.6 Rede Neurais Artificiais 28
2.7 Otimização por Enxame de Partículas 32
2.8 Trabalhos Correlatos 32

3 Metodologia 36
3.1 Robô Corobeu 36
3.1.1 Sintonização dos Controladores PID de Velocidade das Rodas 37
3.2 Cenários de Implementação 39
3.2.1 Ambiente de Simulação 39
3.2.2 Cenário Físico 39
3.3 Proposta de Controle 40
3.4 Gerenciador de Estratégias 42
3.4.1 Criação da Base de Dados de Treinamento 44
3.4.2 Treinamento da Rede Neural 44
3.5 Seguidor de Trajetórias 44
3.5.1 Sintonização do Seguidor de Trajetória em Simulação 46

4 Resultados **48**
4.1 Controlador de Velocidades das Rodas 48
4.2 Treino do Gerenciador Estratégias 51
4.2.1 Validação das Redes 51
4.3 Controlador de Trajetória 52
4.3.1 Simulação 52
4.3.2 Implementação Física 54

5 Conclusões **59**
5.1 Conclusões Gerias 59
5.2 Contribuições 60
5.3 Limitações 60
5.4 Trabalhos Futuros 60

Referências **62**

Apêndices **66**

Apêndice A Esquemático Eletrônico do Robô Corobeu **67**

Apêndice B Desenho Técnico Mecânico do Robô Corobeu **70**

1 Introdução

1.1 Contextualização

A robótica, especialmente a robótica móvel, é um campo de estudo de alta evolução e grande impacto no mundo moderno e traz consigo diversas aplicações, seja na indústria (Dzedzickis *et al.*, 2022), na saúde (Dupont *et al.*, 2021), na agricultura (Oliveira; Moreira; Silva, 2021) ou no entretenimento. Dentro dessas aplicações, destaca-se a necessidade do uso de técnicas de robótica multiagente, isto é, múltiplos robôs trabalhando de forma cooperativa para atingir um objetivo especificado (Drew, 2021).

Em 1993 é publicado o primeiro artigo que propõe a ideia de robôs trabalhando cooperativamente e competitivamente para jogar futebol (Mackworth, 1993). A partir deste ponto, diversos esforços se concentraram em aprimorar a capacidade dos robôs nessa modalidade esportiva, tanto para aproximar o comportamento dos robôs ao comportamento humano, quanto para desenvolver novas tecnologias. Um marco neste contexto foi a criação da *IEEE Very Small Size Soccer* (VSSS), categoria de competição onde equipes de três robôs autônomos controlados remotamente por um computador disputam uma partida de futebol (VSSS League Brasil, 2019).

Nesta categoria, os principais desafios encontrados são a visão computacional para reconhecimento dos elementos da partida, a tomada de decisão rápida em cenários dinâmicos, o planejamento de trajetórias de acordo com as estratégias definidas, a execução destas trajetórias por meio do controle dos robôs e, por fim, a comunicação entre o computador e os robôs.

Diferentes propostas de soluções para estes desafios têm sido apresentadas pela comunidade. Na visão computacional, por exemplo, o seguimento de bordas é utilizado para identificar objetos no campo (Oliveira *et al.*, 2020). Para o planejamento de rotas, métodos como *Rapidly-exploring Random Tree* (RRT) (Vázquez-Hurtado; Rodriguez-Padilla, 2022), campo de vetores unitários (Oliveira *et al.*, 2020) (Ferreira *et al.*, 2023), e algoritmos bioinspirados como DE-Spline e DE-NURBS (Freitas; Cohen; Guimarães, 2023) têm sido explorados. Na definição de estratégias, destacam-se algoritmos baseados em redes neurais (Pedroza, 2023), árvores de decisão (Baldão; Maximo; Cesar, 2020) (Oliveira *et al.*, 2020), e algoritmos de aprendizado profundo, como *Proximal Policy Optimization* (PPO) (Medeiros; Máximo; Yoneyama, 2020) (Brandão *et al.*, 2022).

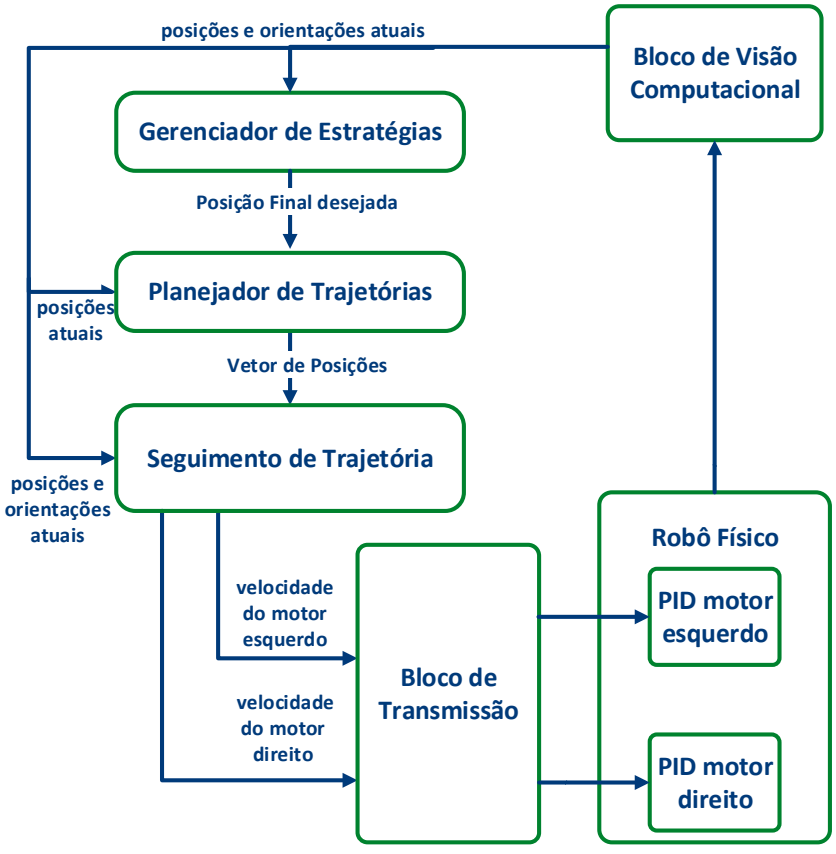
Estes algoritmos, além de possuírem grande complexidade computacional, exigem alto domínio do programador sobre as características do sistema físico. Uma alternativa é a técnica de Aprendizado por Demonstração (do inglês, *Learning from Demonstraition* (LfD)), uma técnica de aprendizado de máquina que permite que os robôs aprendam comporta-

mentos imitando demonstrações humanas (Sakr *et al.*, 2022) (Xu *et al.*, 2019), como pode ser observado no robô MARIA (Triana, 2022). Além disso, no trabalho de Triana (2022) a implementação desta metodologia é realizada através de Redes Neurais Artificiais treinadas utilizando algoritmos bioinspirados, como a Otimização por Enxame de Partículas (do inglês *Particle Swarm Optimization* (PSO)).

1.2 Justificativa

Para solucionar o desafio de Gerenciamento de Estratégias e Execução de Trajetórias proposto na categoria VSSS, o presente trabalho propõe o desenvolvimento de um módulo de Criação/Gerenciamento de Estratégias ofensivas e defensivas (posteriormente chamado de **Coach**). Adicionalmente, propõe-se implementar um sistema de controle de trajetórias para os robôs Corobeu desenvolvidos pela Equipe de Robótica Titans, projeto de extensão vinculado ao curso de engenharia eletrônica da Universidade de Brasília. A arquitetura simplificada para o sistema proposto é apresentada na Figura 1.1. A fim de simplificar o processo de treinamento deste módulo, é proposta a abordagem utilizando a metodologia LfD.

Figura 1.1 – Diagrama de blocos simplificado da Arquitetura proposta



Fonte: Do autor

Neste contexto, serão utilizadas redes neurais recorrentes do tipo Memória de Longo Curto-Prazo para, a partir dos valores de posições dos robôs e posição, angulação e velocidade da bola, definir a estratégia que mais se adequa ao cenário atual da partida. Os dados de treinamento desta Rede Neural Artificial (RNA) serão obtidos pela metodologia LfD.

O bloco **Gerenciador de Estratégias** define qual estratégia o robô deverá implementar, e por tanto, qual sua posição final desejada para aquele instante. Esta, por sua vez, é alimentada ao bloco **Planejador de Trajetórias** que deve calcular os pontos intermediários por onde o robô deve passar para ir da sua posição atual até a posição desejada. Este vetor de posições é enviado ao bloco de **Seguimento de Trajetória** que implementa um controlador Proporcional, Integral e Derivativo (PID) para minimizar o erro entre a posição angular do robô e a posição angular desejada, para que o robô se movimente na direção correta.

1.3 Perguntas de Pesquisa

Três perguntas principais motivam a realização deste trabalho; são elas:

1. Usar estratégias pré-definidas, por meio do LfD, para ditar o comportamentos dos robôs é uma técnica suficiente e viável para a resolução do problema de tomada de decisão e planejamento de rotas na competição VSSS?
2. Um time de robôs desta categoria utilizando esta metodologia será capaz de abstrair corretamente as estratégias implementadas por um humano em cenários dinâmicos e complexos?
3. Um time de robôs desta categoria utilizando esta metodologia será capaz de compor dinamicamente novas estratégias/comportamentos durante as partidas?

1.4 Objetivos

1.4.1 Objetivos Gerais

O objetivo deste trabalho é desenvolver um módulo de controle autônomo para robôs na competição VSSS utilizando a metodologia de aprendizado por demonstração. Este módulo deve gerenciar dinamicamente o comportamento dos robôs nos diferentes cenários do jogo.

1.4.2 Objetivos Específicos

O objetivo geral deste trabalho será cumprido alcançando os seguintes marcos:

- Desenvolver um conjunto de estratégias para futebol de robôs com base nos exemplos fornecidos por humanos.

- Implementar um modelo de gerenciamento de estratégias a partir da base de dados de estratégias utilizando a metodologia LfD.
- Desenvolver o controle autônomo do robô com base em um controlador PID para ajuste da velocidade angular do robô.
- Integrar os diversos módulos da solução proposta no sistema físico desenvolvido pela Equipe Titans.

1.5 Organização do Trabalho

O presente trabalho contém 5 capítulos, organizados da seguinte maneira. No Capítulo 2 é apresentada a fundamentação teórica sobre aprendizado por demonstração, redes neurais artificiais recorrentes, algoritmos bioinspirados e algoritmos de seguimento de trajetória. O Capítulo 3 aprofunda a explicação do sistema proposto por meio da apresentação da plataforma utilizada para este trabalho, dos módulos aqui desenvolvidos, da metodologia de validação utilizada e do procedimento de treinamento do sistema. O Capítulo 4 apresenta e discute os resultados obtidos; por fim, no Capítulo 5 são apresentadas as conclusões e considerações finais.

2 Referencial Teórico

Neste capítulo será abordado todo o referencial teórico que sustenta este trabalho. Ao final do capítulo é apresentada uma pesquisa a respeito do estado da arte do planejamento de estratégias e seguimento de trajetórias no futebol de robôs móveis. Foram analisados artigos de pesquisa desenvolvidos por outras equipes, os artigos de descrição do time (do inglês *Team Description Paper* (TDP)) necessários para a inscrição nas competições e também monografias desenvolvidas com base na competição VSSS.

2.1 Competição IEEE *Very Small Size Soccer*

A competição IEEE Very Small Size Soccer (VSSS) é uma das categorias do RoboCup, um evento internacional voltado para a robótica e a inteligência artificial. O objetivo principal dessa categoria é promover a pesquisa e o desenvolvimento de tecnologias avançadas através de partidas de futebol entre robôs. (VSSS League Brasil, 2019).

2.1.1 Estrutura da Competição

A competição VSSS é realizada em um campo de jogo retangular, geralmente medindo 1,5 metros por 1,3 metros. Cada equipe é composta por três robôs, que devem obedecer a restrições específicas de tamanho, não ultrapassando 7,5 centímetros de diâmetro e 7,5 centímetros de altura. O campo de jogo possui marcações semelhantes às de um campo de futebol tradicional, incluindo áreas de gol e linhas de meio-campo (VSSS League Brasil, 2023).

2.1.2 Regras Básicas

As principais regras da competição são:

1. **Duração da Partida:** As partidas são divididas em dois tempos de cinco minutos cada.
2. **Controle e Comunicação:** Os robôs são controlados de forma autônoma durante as partidas, sem intervenção humana direta. A comunicação entre os robôs e o sistema de controle é realizada através de uma rede sem fio.
3. **Detecção e Localização:** O sistema de visão computacional é responsável por detectar a posição dos robôs e da bola em tempo real. Esse sistema utiliza câmeras posicionadas acima do campo de jogo.

4. **Infrações e Penalidades:** Existem penalidades para infrações como obstrução, bloqueio e comportamento antidesportivo. Dependendo da gravidade, penalidades podem resultar em advertências ou exclusão temporária do robô infrator (VSSS League Brasil, 2023).

2.1.3 Desafios Tecnológicos

Esta competição serve como uma plataforma para testar novas tecnologias, soluções e incentivar a colaboração entre pesquisadores e estudantes de diversas partes do mundo. Dentre os principais desafios que as equipes encontram nesta categoria podemos citar a visão computacional, o controle e navegação e a coordenação multi-agente.

2.2 Robótica Multi-Agente

Sistemas de robótica multi-agente fazem parte de um campo da robótica que estuda o desenvolvimento de sistemas compostos por múltiplos robôs autônomos que devem interagir de forma cooperativa. Esta cooperação pode se dar por necessidade de comunicação, para alcançar um objetivo comum ou até mesmo para criar uma melhora em um resultado específico do sistema (como, por exemplo, melhorando o tempo de execução do sistema).

No trabalho de Cao, Fukunaga e Kahng (1997) é apresentada a seguinte definição para robótica multi-agente cooperativa:

> *Dado alguma tarefa especificada por um projetista, um sistema de múltiplos robôs exibe comportamento cooperativo se, devido a algum mecanismo subjacente (ou seja, o "mecanismo de cooperação"), houver um aumento na utilidade total do sistema.(tradução livre de Cao, Fukunaga e Kahng (1997, seção 1)*

A utilidade total do sistema é definida em relação ao objetivo geral do sistema, trata-se de uma medida da capacidade do sistema de realizar a tarefa especificada de forma eficiente. Pode-se quantificá-la de diversas formas, como através do tempo de execução, do custo da solução, da qualidade da solução ou até mesmo da robustez do sistema.

Para aplicações de robótica multi-agente faz-se necessário a decomposição do objetivo geral do sistema em objetivos específicos para cada agente. Esta decomposição gera a necessidade de um controle organizado do sistema, que por sua vez possui duas abordagens possíveis: centralizado ou descentralizado (Farinelli; Iocchi; Nardi, 2004).

No primeiro caso, há apenas um agente central, podendo ser um computador dedicado ou um dos próprios robôs, que define a ação dos demais com base nas informações de todo o sistema. Ou seja, este agente central define quais tarefas especificas cada um dos robôs hierarquicamente inferiores devem executar. Para isso é necessária uma comunicação confiável entre o agente central e os demais agentes.

Já no caso descentralizado os agentes tomam decisões através das informações locais e da comunicação com seus vizinhos. Esta bordagem pode ser hierárquica, onde determinados agentes estão submetidos ao controle de agentes de maior hierarquia (pode ser considerado como centralizado localmente) ou controle distribuído, onde cada agente é igual ao outro em termos de controle. Sendo assim, é necessária a coordenação entre os agentes através da negociação das tarefas ou de algoritmos de consenso.

2.3 Robôs Móveis de Tração Diferencial

Robôs móveis com tração diferencial são caracterizados por possuírem duas rodas tracionadas, cada qual com um motor independente, e um ou mais pontos de apoio adicionais para garantir estabilidade (Corke, 2017). Este tipo de robô possuí grande simplicidade mecânica, já que não necessita de um mecanismo de direção complexo como os robôs com rodas direcionais. No entanto, o movimento desses robôs está sujeito a restrições não-holonômicas, o que significa que as equações que descrevem seu movimento envolvem suas velocidades e não podem ser expressas apenas em termos de suas coordenadas. Essas restrições tornam o controle da movimentação mais complexo, pois requer a coordenação dos dois motores para realizar manobras.

A Figura 2.2 exemplifica a complexidade para realizar os movimentos e controle de trajetória para este tipo de robô. São apresentados três casos de movimentação para um robô de tração diferencial: na Figura 2.2a é demonstrado como o robô pode assumir uma trajetória retilínea ao implementar velocidades de mesma direção, sentido e magnitude em ambas as rodas; uma trajetória levemente curvada à direita é demonstrada na Figura 2.2b onde as duas rodas ainda possuem velocidades na mesma direção e sentido, porém com magnitudes diferentes, sendo a velocidade esquerda maior do que a velocidade direita, por fim, a Figura 2.2c indica uma rotação em torno do eixo vertical do robô ao implementar velocidades de direções e magnitudes iguais, porem sentidos opostos.

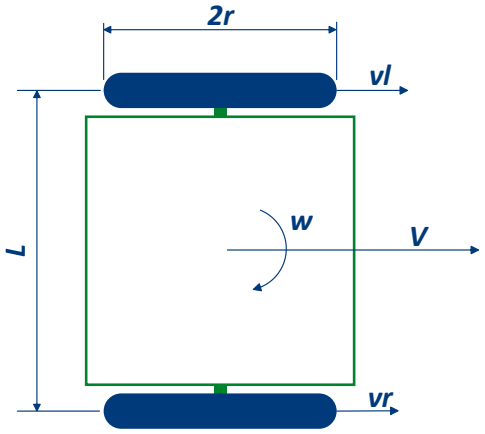
A partir de uma analise geométrica, apresentada na figura Figura 2.1, são obtidas as equações (2.1) e (2.2) que representam, respectivamente, a velocidade linear (v) e angular (ω) do robô em função das velocidades de cada roda (v_r e v_l), seus diâmetros (r) e a distância entre ambas (L) (Bräunl, 2006).

$$v = \frac{(v_r + v_l)\, r}{2} \tag{2.1}$$

$$\omega = \frac{(v_r - v_l)\, r}{L} \tag{2.2}$$

Uma simples combinação linear das equações (2.1) e (2.2) resultam nas equações que descrevem as velocidades lineares de cada roda do robô, conforme apresentado nas equações (2.3) e (2.4).

Figura 2.1 – Análise geométrica da cinemática dos robôs móveis de tração diferencial. O quadrado verde representa o corpo do robô, enquanto os retângulos azuis representam as rodas.



Fonte: Do autor

$$v_r = \frac{v}{r} + \frac{\omega L}{2r} \tag{2.3}$$

$$v_l = \frac{v}{r} - \frac{\omega L}{2r} \tag{2.4}$$

2.4 Navegação de Robôs Móveis

O processo de navegação consiste em direcionar um veículo a fim de atingir uma posição desejada (IEEE..., 1983). Desta forma, existem dois tipos de navegação para robôs móveis, reativa e baseada em mapas (Corke, 2017). Na navegação reativa, o robô não possui informações sobre sua posição e nem sobre o local onde ele está e apenas se movimenta reagindo à estímulos sensoriais.

Já a navegação baseada em mapas é o processo de guiar um robô com informações sobre sua posição e sobre o ambiente ao seu redor. Para isto, utiliza-se algoritmos de planejamento de trajetória e, então, algoritmos de seguimento de trajetória, para , respectivamente, criar um caminho livre por onde o robô possa se locomover e movimentar o robô de forma que ele permaneça neste caminho.

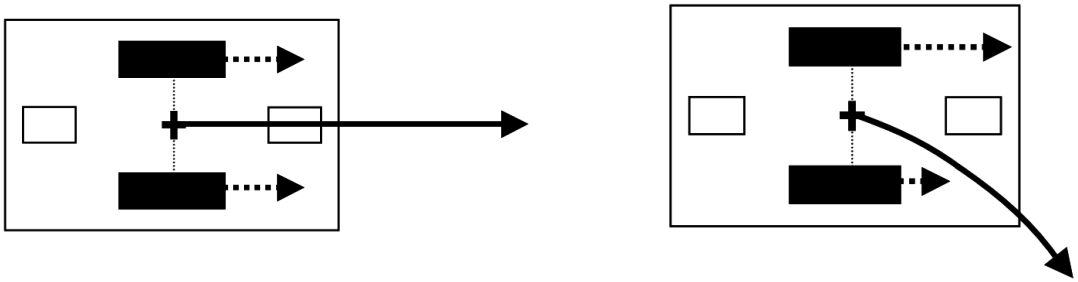
2.4.1 Planejamento de Trajetória em Robótica móvel

O planejamento de trajetórias em robótica móvel é o processo de encontrar um caminho livre de colisões para um robô se mover de um ponto inicial até um ponto final seguindo um conjunto de restrições especificadas. Estas restrições podem tanto ser obstáculos no trajeto, quanto um custo (de tempo, ou energia, por exemplo) relacionado a cada diferente caminho possível.

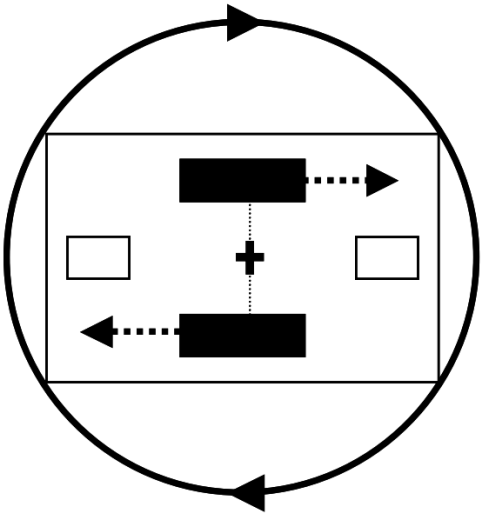
Figura 2.2 – Três casos de movimento para os robôs com tração diferencial. É representado um robô com dois pontos de apoio (quadrados brancos) e duas rodas tracionadas (quadrados pretos) assim como as velocidades de cada roda (setas tracejadas) e a velocidade resultante do robô (seta completa).

a Trajetória retilínea para frente. Velocidades com mesmas direções, sentidos e magnitudes em ambas as rodas.

b Trajetória curvilínea para frente. Velocidades com mesmas direções e sentidos em ambas as rodas, porém magnitude de velocidade da roda esquerda deve ser maior do que a magnitude da roda direita.



c Movimento rotacional no sentido horário. Velocidades das rodas com mesmas direções e magnitudes, porem sentidos opostos.



Fonte: Adaptado de Bräunl (2006)

Existem diversos algoritmos de planejamento de trajetórias já consolidados, dentre eles podemos citar o RRT (Corke, 2017), o A* e o algoritmo de Dijkstra (Bräunl, 2006).

2.4.1.1 Algoritmo de Dijkstra

O algoritmo de Dijkstra encontra o caminho mais curto entre um nó inicial e todos os outros nós em um grafo com pesos não negativos nas arestas (Bräunl, 2006). O algoritmo funciona expandindo o conjunto de nós visitados a partir do nó inicial, sempre escolhendo o nó com a menor distância acumulada até o momento. O algoritmo termina quando o nó final é visitado ou quando todos os nós foram visitados. Seu pseudo-código é apresentado no Algoritmo 2.1

Algoritmo 2.1 Pseudocódigo do Algoritmo de Dijkstra

```
1:  function DIJKSTRA(Graph, source)
2:      dist[source] ← 0                                  ▷ Distância do nó inicial
3:      for cada vértice v em Graph do
4:          if v ≠ source then
5:              dist[v] ← ∞                ▷ Distância desconhecida para todos os outros nós
6:              prev[v] ← indefinido                      ▷ Nó anterior desconhecido
7:          end if
8:          adicione v a Q                 ▷ Todos os nós na lista de prioridades
9:      end for
10:     while Q não está vazio do                         ▷ Loop principal
11:         u ← vértice em Q com dist[u] mínimo            ▷ Nó com a menor distância
12:         remova u de Q
13:         for cada vizinho v de u do          ▷ Onde v ainda não foi removido de Q
14:             alt ← dist[u]+ LENGTH(u, v)
15:             if alt < dist[v] then
16:                 dist[v] ← alt
17:                 prev[v] ← u
18:             end if
19:         end for
20:     end while
21:     return dist[], prev[]
22: end function
```

2.4.1.2 Algoritmo A*

O algoritmo A* é uma extensão do algoritmo de Dijkstra que utiliza uma função heurística para estimar a distância do nó atual até o nó final (Bräunl, 2006). A função heurística deve ser admissível, ou seja, ela nunca deve superestimar a distância real. O algoritmo A* funciona de forma semelhante ao algoritmo de Dijkstra, mas a cada passo ele escolhe o nó com a menor soma da distância acumulada e da estimativa heurística da distância até o nó final, conforme apresentado no Algoritmo 2.2.

2.4.1.3 Rapidly Exploring Random Tree

Diferente dos dois anteriores, o algoritmo RRT não se trata de um algoritmo com base em grafos, mas sim em amostragem. Trata-se de um algoritmo probabilístico para planejamento de movimento que é particularmente adequado para sistemas com alta dimensionalidade e restrições não holonômicas, ou seja restrições que não dependem apenas da posição atual do objeto (Corke, 2017). Ele constrói uma árvore de possíveis trajetórias, expandindo-a aleatoriamente em direção ao objetivo conforme apresentado no Algoritmo 2.3.

Algoritmo 2.2 Pseudocódigo do Algoritmo A*

```
1: function A*(start, goal)
2:     openSet ← {start}
3:     cameFrom ← mapa vazio
4:     gScore[start] ← 0
5:     fScore[start] ← HEURISTIC_COST_ESTIMATE(start, goal)
6:     while openSet não está vazio do
7:         current ← o nó em openSet com o menor valor de fScore[]
8:         if current = goal then
9:             return RECONSTRUCT_PATH(cameFrom, current)
10:        end if
11:        remova current de openSet
12:        adicione current a closedSet
13:        for cada neighbor de current do
14:            if neighbor está em closedSet then
15:                continue                ▷ Ignore o vizinho que já foi avaliado
16:            end if
17:            tentative_gScore ← gScore[current]+
    DIST_BETWEEN(current, neighbor)
18:            if neighbor não está em openSet then
19:                adicione neighbor a openSet
20:            else if tentative_gScore ≥ gScore[neighbor] then
21:                continue                ▷ Este não é um caminho melhor
22:            end if
23:            cameFrom[neighbor] ← current
24:            gScore[neighbor] ← tentative_gScore
25:            fScore[neighbor] ← gScore[neighbor]+ HEURIS-
    TIC_COST_ESTIMATE(neighbor, goal)
26:        end for
27:    end while
28:    return falha
29: end function
```

2.4.2 Controle de Trajetória em Robótica Móvel

O controle de trajetória pode ser resumido em mover o robô para um ponto na trajetória especificada e, então, movê-lo para o ponto seguinte e assim por diante até o fim da trajetória (Corke, 2017). Uma comum abordagem para isto é minimizar a diferença entre a posição angular do robô e o ângulo entre o robô e o ponto desejado, isto é, alinhar o robô com o ponto desejado no espaço cartesiano, e, então, mover o robô nesta direção.

O processo pode ser implementado através de um controlador PID que recebe o erro de orientação do robô e retorna a velocidade angular (ω) do mesmo para ser aplicada às Equações 2.4 e 2.3 (Siciliano *et al.*, 2009). Neste caso, para garantir que o robô se mantenha em movimento, a sua velocidade linear deve ser diferente de zero, podendo ser mantida constante, ou assumir forma trapezoidal (possuindo uma rampa de aceleração e desaceleração) para

Algoritmo 2.3 Pseudocódigo do Algoritmo RRT

```
1:  function RRT(start, goal, numIterations)
2:      T.init(start)                          ▷ Inicializa a árvore com o nó inicial
3:      for i = 1 to numIterations do
4:          q_rand ← SAMPLEFREE                ▷ Amostre um ponto aleatório no espaço livre
5:          q_near ← NEAREST(T, q_rand)        ▷ Encontre o nó mais próximo na árvore
6:          q_new ← STEER(q_near, q_rand)      ▷ Estenda a árvore em direção a q_rand
7:          if OBSTACLEFREE(q_near, q_new) then
8:              T.add_vertex(q_new)
9:              T.add_edge(q_near, q_new)
10:             if q_new == goal then
11:                 return T                   ▷ Caminho encontrado
12:             end if
13:         end if
14:     end for
15:     return T                               ▷ Caminho não encontrado
16: end function
```

obter um controle mais suave.

2.5 Aprendizagem por Demonstração

A metodologia de Aprendizado por Demonstração (do inglês *Learning from Demonstraition*) é um paradigma de aprendizado de máquina que permite que robôs aprendam novas habilidades através da observação e da imitação de um professor, que pode ser tanto um humano quanto outro robô (Ambhore, 2020). A LfD é composta por três estágios (Bakker; Kuniyoshi, 1996):

a) Estágio de Demonstração, onde o professor apresenta a tarefa desejada para o robô e fornece exemplos de como realizar a mesma, os dados de entrada e saída do sistema que reprezentam a solução do professor são armazenados em memória;

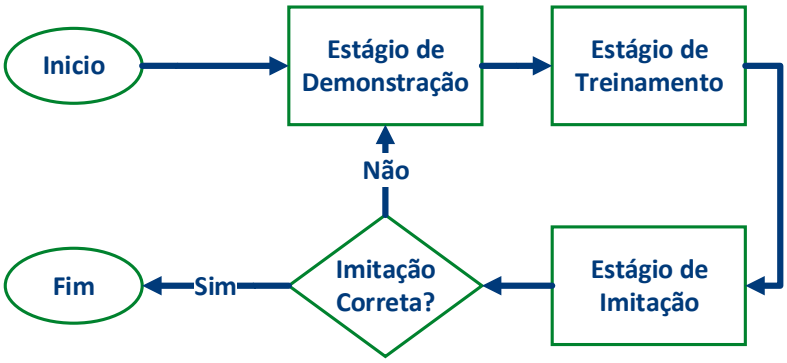
b) Estágio de Treinamento, o robô utiliza os dados coletados durante a demonstração para aprender um modelo da tarefa. Esse modelo pode ser representado de diversas formas, como um conjunto de regras, um modelo probabilístico ou uma rede neural artificial. O processo de treinamento geralmente envolve a otimização de parâmetros do modelo para minimizar o erro entre o comportamento do robô e o comportamento demonstrado pelo professor;

c) Estágio de Imitação, por fim, uma vez que o robô tenha aprendido um modelo da tarefa, ele pode tentar imitar o comportamento do professor. A imitação pode ser realizada em um ambiente simulado para testar e refinar o modelo antes de ser aplicada em um ambiente real.

O fluxograma representando os estágios da metodologia é apresentado na Figura 2.3,

percebe-se que neste caso, a LfD é também uma metodologia de aprendizado por reforço, pois o próprio operador pode julgar se a imitação foi correta e executar novamente o treinamento caso contrário.

Figura 2.3 – Estágios da metodologia de aprendizado por demonstração



Fonte: Do autor

Diversas são as formas de demonstração que o professor pode realizar, alguns exemplos incluem:

- **Cinestésica:** o professor guia fisicamente o robô para que ele efetue o comportamento desejado;
- **Teleoperação:** o robô é controlado remotamente pelo professor para que o comportamento desejado seja executado;
- **Sombreamento:** um sistema de visão computacional captura os movimentos do professor e os reproduz no robô;
- **Apresentação:** o professor realiza a tarefa em um ambiente simulado ou real, e o robô observa e registra os dados relevantes.

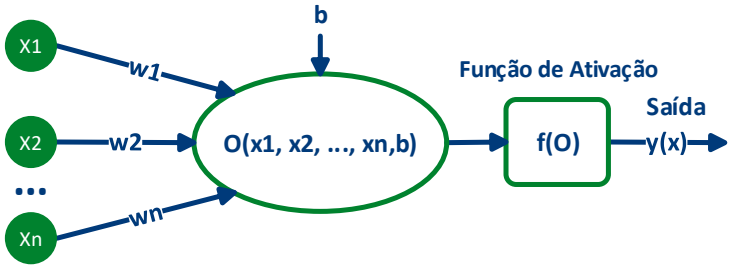
Uma forma de implementação desta metodologia é através do treino de uma Rede Neural Artificial que receba os dados de controle do professor e calcule os sinais de atuação do robô com base nos dados coletados durante a demonstração.

2.6 Rede Neurais Artificiais

As redes neurais artificiais são um grupo de algoritmos que buscam imitar o comportamento do cérebro humano para realizar uma tarefa específica (Haykin, 2001; Bräunl, 2006). Elas consistem em unidades básicas de processamento chamadas neurônios artificiais, que são organizados em camadas e conectados por sinapses artificiais. Cada neurônio recebe um conjunto de entradas, realiza uma operação matemática seguida por uma função de ativação e gera uma saída, conforme apresenta a Figura 2.4, onde $O(x_1, x_2, ..., x_n, b)$ é a

função matemática que relaciona as n entradas do neurônio com seus pesos e viés, $f(O)$ é a aplicação da função de ativação à esta relação, que resulta na saída $y(x)$ do neurônio.

Figura 2.4 – Representação genérica de um neurônio artificial.

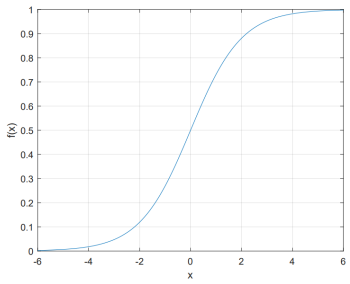


Fonte: Do autor

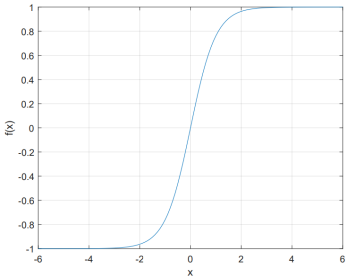
Diversas são as funções de ativação utilizadas, sendo as mais comuns a sigmoide, a tangente hiperbólica e a função linear saturada entre $[-1, 1]$. A representação gráfica de cada uma dessas funções é apresentada na Figura 2.5.

Figura 2.5 – Três funções de ativação mais comuns para os neurônios artificiais.

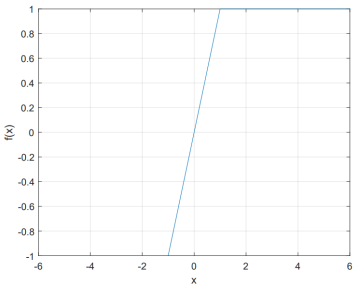
a Função sigmoide.
$f(x) = \frac{1}{1+e^{-x}}$



b Função tangente hiperbólica.
$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$



c Função linear saturada.
$f(x) = \begin{cases} -1, & x \leq -1 \\ x, & -1 < x < 1 \\ 1, & x \geq 1 \end{cases}$



Fonte: Adaptado de Triana (2022)

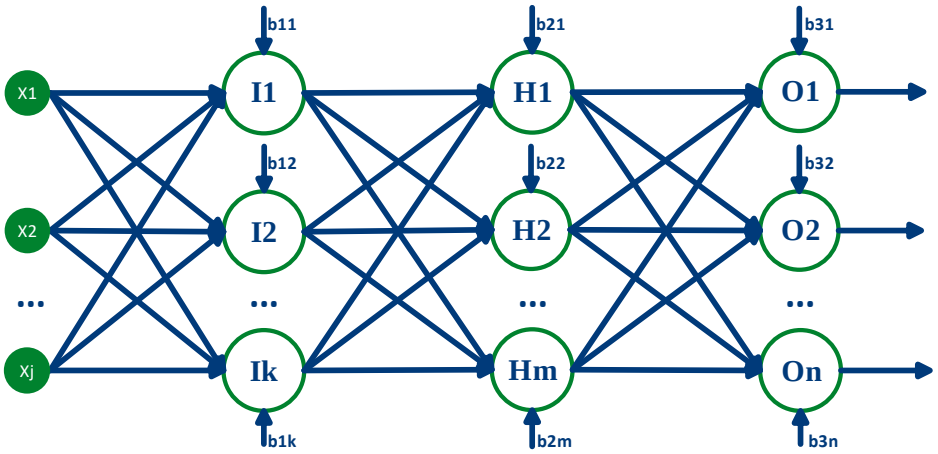
Redes neurais do tipo *perceptron* organizam os neurônios em camadas onde cada neurônio de uma camada esta conectado à todos os neurônios da camada seguinte através de ligações sinápticas. Dessa forma, a informação flui de maneira unidirecional, da entrada para a saída, caracterizando uma rede do tipo *feedfoward* (Haykin, 2001). Existem dois tipos principais de redes perceptron: o perceptron de única camada (do inglês *Single-layer Perceptron* (SLP), que possui apenas uma única camada, e o perceptron de multicamada (do inglês *Multi-layer Perceptron* (MLP)), composto por uma camada de entrada, uma camada de saída e uma ou mais camadas escondidas. As camadas escondidas se tratam de um conjunto de neurônios entre as camada de entrada e a camada de saída. Em ambos os casos, cada camada pode conter múltiplos neurônios.

Uma rede MLP genérica com 1 camada escondida é apresentada na Figura 2.6. Cada

neurônio implementa a Equação 2.5, onde y é a saída do neurônio, x_i são as entradas, w_i são os pesos sinápticos, b é o viés do neurônio e $f(x)$ é a função de ativação. Para os neurônios da camada de entrada os valores de x_i serão os valores de entrada da rede, enquanto que para os neurônios da camada escondida e da camada de saída os valores de x_i serão as saídas dos neurônios das camadas anteriores. A saída da rede são os valores y dos neurônios da última camada.

$$y = f\left(\sum_{i=1}^{N} x_i w_i + b\right) \tag{2.5}$$

Figura 2.6 – Representação de uma rede neural MLP genérica com j entradas, k neurônios da camada de entrada, 1 camada escondida com m neurônios e n neurônios na camada de saída.



Fonte: Do autor

Redes neurais *feedfoward* apresentam dificuldades para lidar com dados sequenciais e com dependências temporais. Para solucionar este problema, foram desenvolvidas as Redes Neurais Recorrentes (RNR), que possuem retroalimentação entre seus neurônios. Tal característica possibilita que informações de estados anteriores influenciem estados futuros. Contudo, RNRs tradicionais apresentam dificuldades em aprender longas sequências temporais, o que compromete seu desempenho em determinadas aplicações.

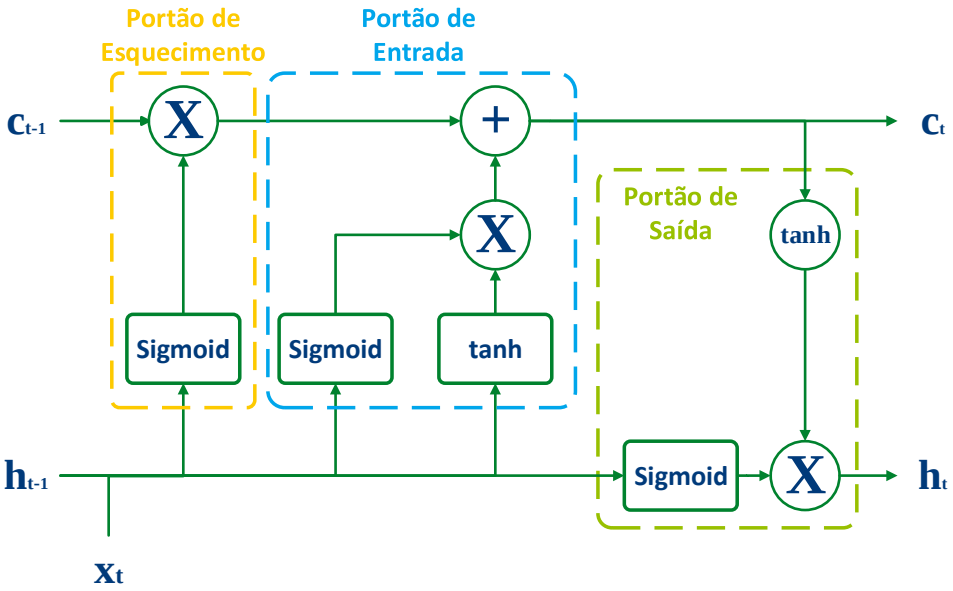
Para contornar estas limitações, Hochreiter e Schmidhuber (1997) propuseram as redes do tipo Memória de Longo Curto-Prazo (do inglês *Long Short-Term Memory* (LSTM)), uma variação das RNNs projetada especificamente para capturar dependências de longo prazo em dados sequenciais. Isso é possível com a introdução das células de memória e três "portões"distintos na arquitetura da rede: o portão de entrada, o portão de esquecimento e o portão de saída.

Cada portão contribui de maneira diferente para a atualização do estado da célula, o portão de esquecimento define o quanto a informação armazenada deve ser mantida para o novo estado, o portão de entrada define o quanto a informação nova deve ser adicionada

à célula de memória, por fim, o portão de saída define o quanto a informação da célula de memória deve ser utilizada como saída da rede.

A Figura 2.7 apresenta a arquitetura de uma unidade de processamento básica de uma rede LSTM. Nela, pode-se perceber que a entrada da rede (x_t) em conjunto com o estado oculto anterior da rede (h_{t-1}) definem o quanto a informação do estado da célula de memória (c_{t-1}) será modificado pelos diferentes portões, definindo o quanto a informação anterior e atual devem ser mantidas e prorrogadas para os demais níveis da rede.

Figura 2.7 – Exemplo de uma unidade de processamento básica de uma rede LSTM. Cada retângulo representa um neurônio com sua função de ativação sendo especificada pelo texto dentro, já os círculos representam operações ponto a ponto com os elementos podendo ser multiplicações, somas ou aplicações da tangente hiperbólica, por fim as entradas da rede são os vetores c_{t-1}, h_{t-1} e x_t enquanto as saídas são os vetores c_t e h_t.



Fonte: Do autor

Os processos de treinamento das redes neurais consistem em ajustar os valores de parâmetros das redes de modo que, dado um conjunto de valores de entrada x, a saída da rede neural y se aproxime dos valores de saída esperados $\hat{y}$.

Para isso, diversos algoritmos podem ser aplicados, sendo o mais comum o método de aprendizado supervisionado através de retropropagação de erro (do inglês *error backpropagation*) (Bräunl, 2006), que consiste em ajustar os pesos e viés da rede a partir da camada de saída até a camada de entrada seguindo uma regra de minimização de erro.

Outras técnicas podem ser empregadas para o treinamento das redes neurais, como os algoritmos bioinspirados de otimização, que têm o objetivo de minimizar uma função custo, neste caso o erro da rede neural, utilizando técnicas que visam imitar o comportamento de sistemas naturais (Mirjalili; Dong, 2020). Um dos algoritmos bioinspirados mais bem

consolidados é o algoritmo de Otimização por Enxame de Partículas, que será tratado na Seção 2.7.

2.7 Otimização por Enxame de Partículas

Um problema de otimização consiste em maximizar ou minimizar o valor de uma função objetivo $f(x)$. Em outras palavras, é encontrar os valores de um vetor x, pertencente ao espaço de busca X (valores aceitáveis de x), de modo que, ao serem aplicados na função custo, o resultado seja maximizado ou minimizado. Vale observar que a função objetivo pode estar sujeita a funções de restrição $G(x)$ (Weise, 2009).

A otimização por enxame de partículas é um tipo de algoritmo bioinspirado de otimização que implementa características do movimento de bando de pássaros, cardumes ou enxames no geral para encontrar soluções sub-ótimas de uma função objetivo (Kennedy; Eberhart, 1995).

O PSO consiste em inicializar diversos indivíduos (valores do vetor x) aleatoriamente, avaliar sua resposta na função objetivo e, caso a resposta não cumpra a condição de parada, modificar a posição do indivíduo no espaço de busca de acordo com as equações (2.6) e (2.7).

$$v_{ij}^{t+1} = \omega v_{ij}^{t} + c_1 U_{1j} \left(y_{ij}^{t} - x_{ij}^{t} \right) + c_2 U_{2j} \left(y_{sj}^{t} - x_{ij}^{t} \right) \tag{2.6}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \tag{2.7}$$

A equação (2.6) é responsável por atualizar a velocidade da partícula i na direção j, onde ω é o fator de inercia do algoritmo (tipicamente diminui com o tempo); c_1 é o coeficiente cognitivo e c_2 é o coeficiente social, U_1 e U_2 são números aleatórios distribuídos entre [0,1], Estes quatro fatores irão indicar se a velocidade da partícula irá depender mais de sua melhor posição individual (y_i) ou da posição da melhor partícula do enxame (y_s).

As posições das partículas são atualizadas utilizando a equação (2.7) a partir da posição atual da partícula e de sua velocidade no próximo instante de tempo. Como o fator de inércia (ω) costuma diminuir com o tempo, nas primeiras iterações os indivíduos realizam uma exploração mais abrangente no espaço de busca, já nas últimas iterações a velocidade das partículas diminui, e a busca passa a ser mais local, ao redor do ponto da melhor solução encontrada.

2.8 Trabalhos Correlatos

Diversos trabalhos na área de futebol de robôs abordam o problema de criação de estratégias para definir o comportamento dos robôs em campo, e controle de trajetória para guiar os robôs ao executarem este comportamento.

Algoritmo 2.4 Pseudocódigo do algoritmo PSO

function PSO($S, N, c_1, c_2, Max_{iter}, thres$)
  Inicialização do enxame
  $iter = 1$
  while $f(y_j) < thres$ **and** $iter \leq Max_{iter}$ **do**
    for $i = 1$ **to** S **do**
      if $f(x_i) \leq f(y_i)$ **then**
        $y_i \leftarrow x_i$ ▷ Avaliação e detecção da melhor posição individual
      end if
    end for
    Avaliação de y_j utilizando os S valores de aptidão $f(y_i)$ ▷ Melhor Global
    for $i = 1$ **to** S **do**
      for $j = 1$ **to** N **do**
        $v_{ij}^{(t+1)} = wv_{ij}^{(t)} + c_1U_1(y_{ij}^{(t)} - x_{ij}^{(t)}) + c_2U_2(y_j - x_{ij}^{(t)})$
        $x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}$ ▷ Atualização
      end for
    end for
    $iter = iter + 1$
  end while
  return Posição da melhor partícula x e sua aptidão $f(x)$
end function

Dentre as soluções presentes na comunidade para a criação de estratégias bons exemplos são os algoritmos baseados em Árvores de Decisão e Máquinas de Estados Finitos (do inglês *Finite State Machine* (FSM)), a abordagem com Aprendizado por Reforço e abordagem baseada em algoritmos genéticos (Baldão; Maximo; Cesar, 2020; Oliveira *et al.*, 2020) (Brandão *et al.*, 2022; CORREIA, 2023; Medeiros; Máximo; Yoneyama, 2020) (Pedroza, 2023).

Já para o controle de trajetória dos robôs algumas abordagens que se destacam são os Campos Potenciais Artificiais (do inglês *Artificial Potential Fields* (APF)), o *Univector Field Navigation* e o Controlador PID (Oliveira *et al.*, 2020) (Baldão; Maximo; Cesar, 2020) (CORREIA, 2023).

Adicionalmente, o estudo de Martins (2023) avalia diferentes paradigmas de aprendizado por reforço profundo para o aprendizado de políticas de controle para os robôs da categoria VSSS.

A Tabela 2.1 apresenta o quadro resumo de trabalhos correlacionados. Foram avaliados nove artigos com diferentes abordagens tanto para o gerenciamento de estratégias quanto para o seguimento de trajetórias. A revisão desta literatura foi realizada utilizando as palavras chave desta dissertação em base de dados e plataformas de buscas acadêmicas como *Scopus*, *IEEE Xplorer* e o *Google Schoolar*. O principal critério de filtragem de dados foi a implementação das técnicas na categoria IEEE VSSS.

Tabela 2.1 – Comparação dos trabalhos correlatos de desenvolvimento de técnicas de Gerenciamento de Estratégias e Seguimento de Trajetórias na categoria VSSS

Autor	**Técnica de Gerenciamento de Estratégias**	**Técnica de Seguimento de Trajetória**	**Observações Gerais**
Martins *et al.* (2020)	Aprendizado por Reforço Profundo (PPO)	Não especificado	Foco em um único agente e em políticas de controle contínuas.
Pedroza (2023)	Algoritmos Genéticos (NEAT)	Não especificado	Uso de NEAT para otimizar redes neurais no controle e estratégia.
Pena *et al.* (2020)	Aprendizado por Reforço Profundo (PPO)	Não especificado	O artigo discute a aplicação de técnicas de DRL, como DDPG e DQN, para o aprendizado de políticas de controle.
Brandão *et al.* (2022)	Aprendizado por Reforço Profundo (PPO)	Não especificado	Aplicação de MARL com self-play para o aprendizado de controle e tomada de decisões estratégicas.
CORREIA (2023)	Deep Deterministic Policy Gradient (DDPG)	Campos Potenciais Harmônicos (HPF)	Combinação de DDPG com um controlador PID para aprendizado assistido.
Baldão, Maximo e Cesar (2020)	Árvores de comportamento e Máquinas de Estados Finitos (FSM)	Univector Field Navigation	Desenvolvimento de estratégias dinâmicas para times VSSS, com foco na combinação de diferentes métodos de planejamento de trajetória e posicionamento.
Oliveira *et al.* (2020)	Árvore de decisão	Artificial Potential Field (APF)	Utilização de árvore de decisão para a tomada de decisão e APF para o planejamento de trajetórias em robôs VSSS.

Tabela 2.1 – Comparação dos trabalhos correlatos (continuação)

Autor	**Técnica de Gerenciamento de Estratégias**	**Técnica de Seguimento de Trajetória**	**Observações Gerais**
Medeiros, Máximo e Yoneyama (2020)	*Proximal Policy Optimization* (PPO)	Line Follower, Univector Field	Uso de PPO e Curriculum Learning para treinar agentes em um ambiente virtual para interceptar a bola e levá-la para o lado do adversário em VSSS.
Medeiros, Yoneyama e Máximo (2023)	*Proximal Policy Optimization* (PPO), Self-Play	Univector Field	Uso de PPO e Self-Play para treinar agentes em um ambiente virtual para cooperar em um ataque e marcar gols em VSSS.
Este trabalho (2024)	Aprendizado por Demonstração (LfD)	Controlador PID para velocidade angular	Arquitetura de controle centralizado baseada em LfD. LSTM define a estratégia e o controle de trajetória, com treinamento por PSO.

Dentre todos os trabalhos encontrados, a metodologia de Proximal Policy Optimization (PPO) é a mais presente, por sua capacidade de aprendizagem e adaptação. No contexto de seguimento de trajetórias, as abordagens baseadas em campos potenciais ou campos vetoriais são as mais observadas, por suas simplicidades e eficácias.

Vale ressaltar que na pesquisa realizada não foi encontrada nenhuma utilização das metodologias de Aprendizagem por Demonstração, portanto este trabalho se diferencia ao propor o uso destas abordagens.

3 Metodologia

3.1 Robô Corobeu

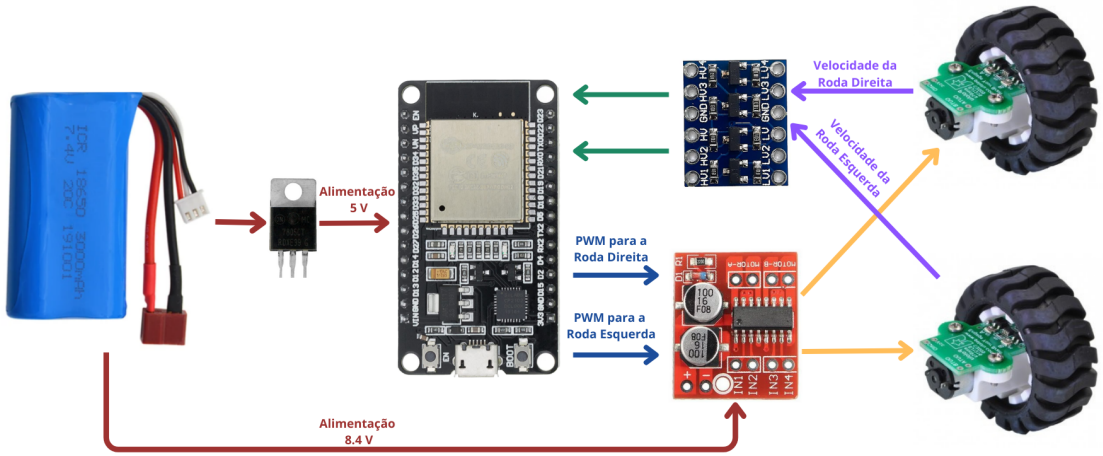
O robô Corobeu, Figura 3.2, é uma plataforma de robótica móvel diferencial, desenvolvida na Universidade de Brasília pela Equipe de Robótica Titans. Trata-se de um robô de dimensões 75x75x70 mm e com locomoção diferencial.

O esquemático do circuito eletrônico do robô está presente no Apêndice A. O robô conta com uma bateria de Polímero de Lítio (LiPo) de 7.4V e 1100mAh para alimentar dois motores N20 de 300 RPM e um regulador de tensão que fornece 5V para o circuito de controle.

Este, por sua vez, é composto por uma Sistem On Chip (SoC) ESP-32 que utiliza de sua comunicação Wi-Fi nativa para receber 2 bytes contendo os valores de velocidades esperadas das rodas direita e esquerda e implementar estes valores, por PWM, na ponte-H MX1616H. Adicionalmente, os valores de rotações atuais dos motores são lidos por dois sensores ópticos encoders de quadratura e fornecidos à ESP-32 através de um conversor de nível lógico. A Figura 3.1 apresenta as conexões entre os dispositivos eletrônicos do robô Corobeu.

O desenho técnico mecânico do robô está presente no Apêndice B. Toda a estrutura

Figura 3.1 – Conexões eletrônicas do robô Corobeu

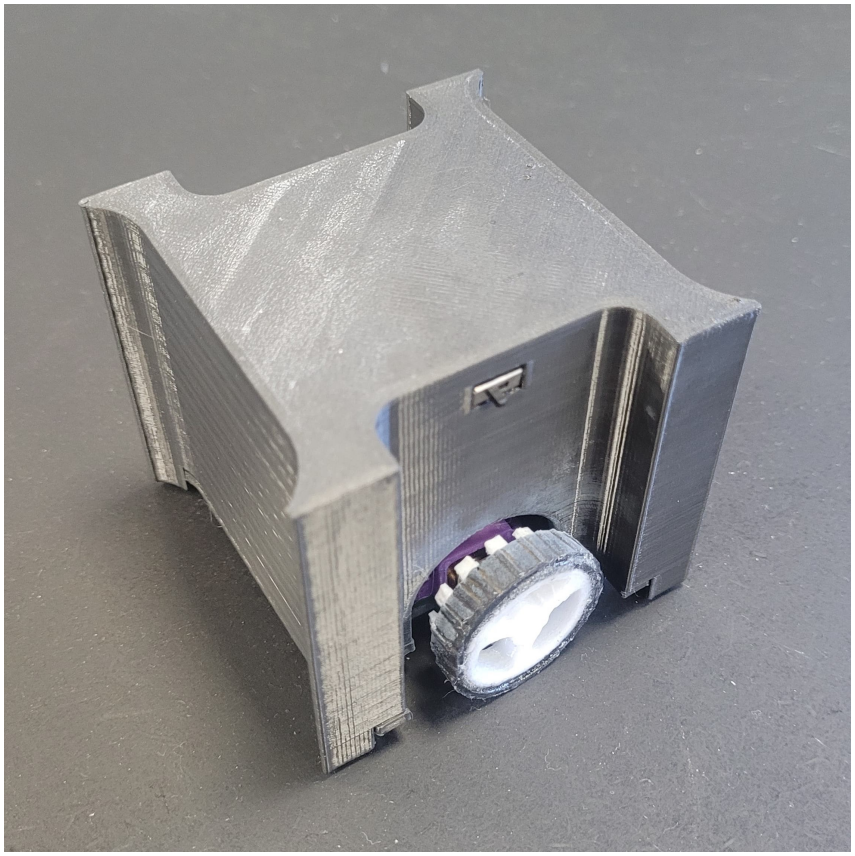


Fonte: Do Autor

do robô foi desenvolvida no *software* CAD *Fusion360* e, então, impressa em polímero PLA. As rodas do robô também foram desenvolvidas pela equipe e contam com 12 "dentes"para serem identificados pelos *encoders* de velocidade dos motores que atuam a uma taxa de amostragem de 700ms.

O robô impresso e montado pela equipe é apresentado na Figura 3.2.

Figura 3.2 – Foto do robô Corobeu desenvolvido pela Equipe de Robótica Titans.



Fonte: Do autor

3.1.1 Sintonização dos Controladores PID de Velocidade das Rodas

Para o correto controle de velocidade do motor, é necessário implementar um controlador PID. Este processo é realizado em três etapas: a identificação do sistema, a otimização dos parâmetros PID e a validação do controlador.

Para modelar a dinâmica dos motores N20, foi realizada uma identificação empírica dos sistemas. Para isso, os motores foram testados sem carga, aplicando-se uma entrada do tipo degrau com amplitude de 180 PWM em cada um. A velocidade de rotação das rodas foi medida por um encoder de quadratura, com uma taxa de amostragem de 700ms. A resposta do sistema foi gerada e visualizada por meio do *software* MATLAB®, permitindo a obtenção do ganho do sistema (K) e da constante de tempo (τ), os quais foram utilizados na Equação 3.1.

Tabela 3.1 – Parâmetros utilizados no PSO para sintonização do PID de controle de velocidade das rodas.

Parâmetro	Valor
Fator de Inercia Inicial	0.9
Fator de Inercia Final	0.1
Coeficiente Cognitivo	2.05
Coeficiente Social	2.05
Número de Individuos	20
Valor Mínimo do Espaço de Busca	0.01
Valor Máximo do Espaço de Busca	5
Número de Iterações	100
Número de Experimentos	16

$$G(s) = \frac{K}{\tau s + 1} \tag{3.1}$$

A constante de tempo do sistema é definida como o tempo que o sistema demora para atingir 63% do valor desejado, enquanto que o ganho do sistema pode ser calculado a partir da relação entre a variação do sinal de entrada e a variação do sinal de saída, conforme a Equação 3.2. Para o cálculo do K foram utilizados os valores do sistema após o regime transiente.

$$K = \frac{\Delta RPM}{\Delta PWM} \tag{3.2}$$

Com o modelo do sistema definido, o algoritmo de Otimização por Enxame de Partículas (PSO) foi empregado para sintonizar os ganhos K_p, K_i e K_d do controlador PID. O PSO foi configurado com os parâmetros apresentados na tabela 3.1.

A função custo utilizada (Equação 3.3) tem por objetivo a minimização do sobrepasso da resposta (OS), a minimização do erro total absoluto da resposta do sistema (e_a) e a minimização do erro em estado estacionário (e_{es}); esta última, porém, com uma intensidade maior do que as outras.

$$f = 0.3 * e_a + 0.4 * e_{es} + 0.3 * OS \tag{3.3}$$

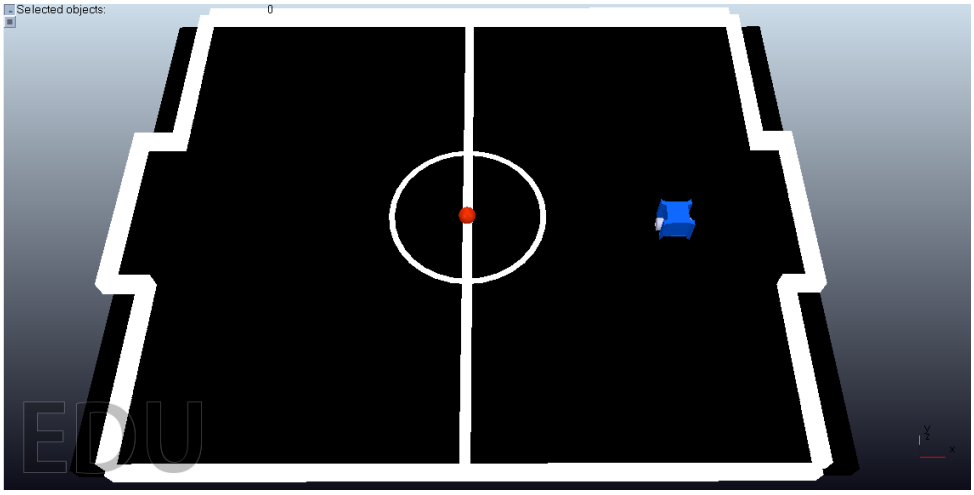
Os parâmetros otimizados foram implementados no robô físico, e o desempenho do controlador foi validado em testes de resposta ao degrau e seguimento de trajetórias. Pequenos ajustes manuais nos ganhos foram realizados para compensar discrepâncias entre o modelo teórico e as não linearidades do sistema real, como atrito mecânico e variações na carga da bateria.

3.2 Cenários de Implementação

3.2.1 Ambiente de Simulação

O ambiente de simulação foi configurado no *software CoppeliaSim*[1]. Para isso, foi desenvolvido um cenário inicial contendo apenas um robô e uma bola, conforme ilustrado na Figura 3.3.

Figura 3.3 – Primeiro cenário para simulação contendo apenas um robô e uma bola.



Fonte: Do autor

As dimensões do campo deste cenário condizem com as dimensões reais, sendo 1,7 metros de comprimento por 1,3 metros de largura, conforme especificado na Figura 3.4. A origem do eixo de coordenadas se encontra no centro do campo, sendo assim as coordenadas variam entre -0.85 e 0.85 para o eixo x e -0.65 e 0.65 para o eixo y.

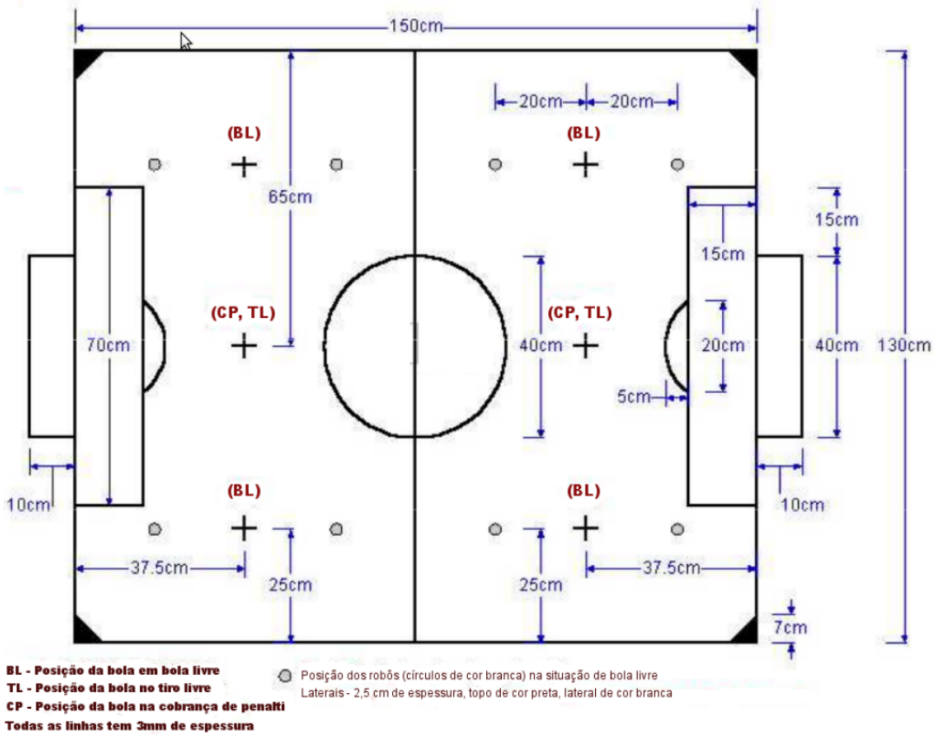
Além do cenário no *software CoppeliaSim* foram desenvolvidos *scripts* em *Python* para realizar a interface entre o modelo do robô Corobeu presente no simulador e o modelo de controle proposto.

3.2.2 Cenário Físico

Os testes do controle físico serão realizados no campo desenvolvido pela Equipe de Robótica Titans, apresentado na Figura 3.5.

[1] Disponível em www.coppeliarobotics.com

Figura 3.4 – Dimensões e marcações do campo de acordo com o manual de regras oficial da competição VSSS.



Fonte: Manual de regras da competição VSSS. (VSSS League Brasil, 2023)

Figura 3.5 – Foto do campo real desenvolvido pela Equipe de Robótica Titans.



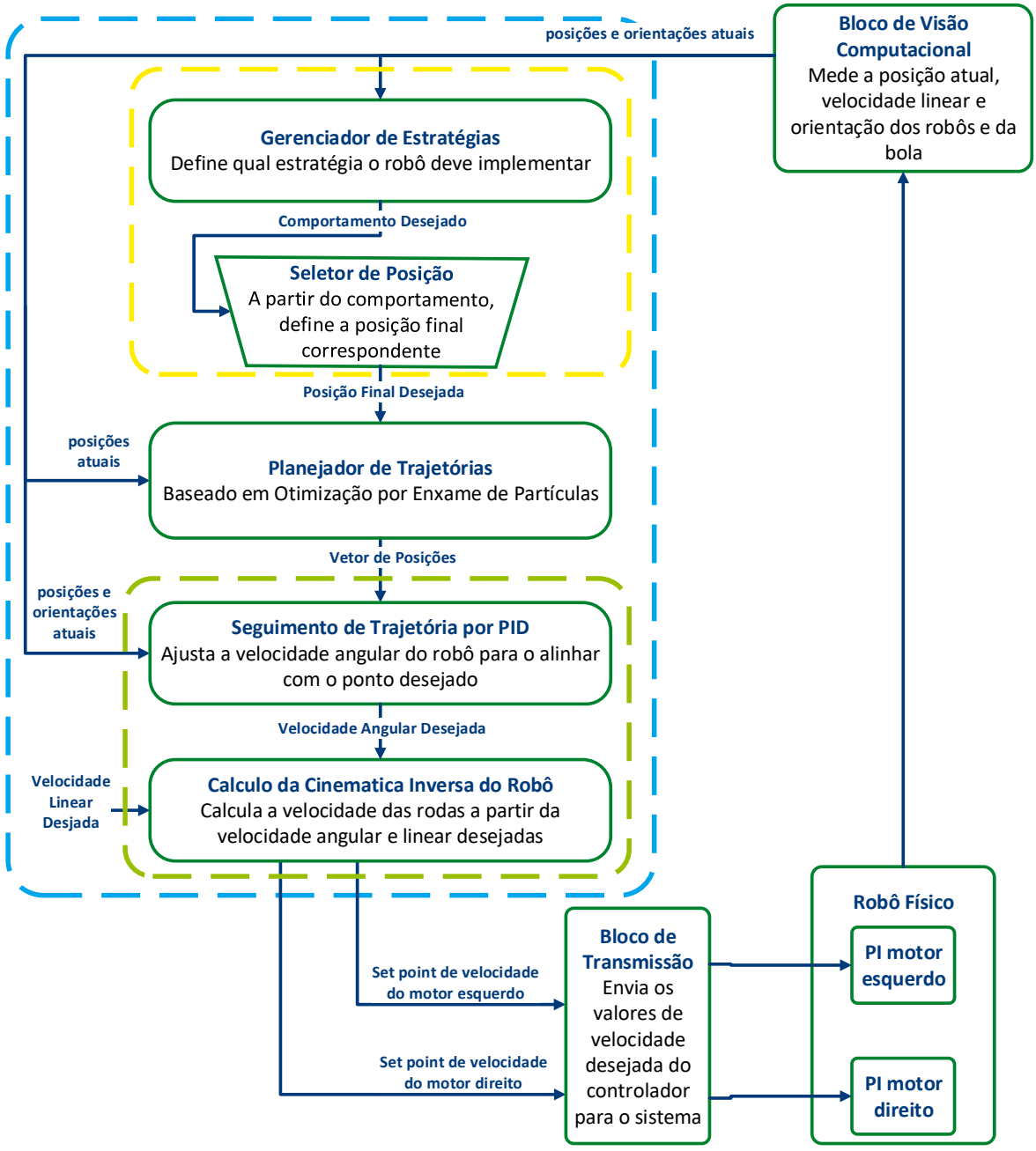
Fonte: Do autor.

3.3 Proposta de Controle

Este trabalho propõe a implementação de um controle centralizado baseado em aprendizado por demonstração para que os robôs executem automaticamente estratégias

criadas por operadores humanos. A arquitetura do sistema completo do robô está disposta na Figura 3.6.

Figura 3.6 – Diagrama de blocos da arquitetura proposta



Fonte: Do autor

Esta arquitetura proposta visa solucionar os desafios apresentados na Seção 1.1.

O bloco de Visão Computacional fornece, em tempo real, as posições, velocidades e orientações dos objetos no campo para os demais blocos do sistema. Essa tarefa ocorre por meio da captura do cenário da partida por uma câmera digital. O *software* utilizado, denominado VSS Vision, foi desenvolvido pela equipe Robocin e é compartilhado para

uso por todas as equipes que participam dessa categoria. Ele é amplamente consolidado nas competições do IEEE VSSS e fornece as posições dos objetos no campo com precisão e confiabilidade. O VSS Vision opera a uma taxa de 30 FPS, o que equivale a um frame válido a cada 33 ms. O código-fonte do sistema está disponível no repositório do GitHub: https://github.com/robocin/vss-vision.

O bloco **Gerenciador de Estratégias** é responsável por, a partir das posições dos objetos no campo, definir qual a melhor estratégia a ser implementada no instante atual da partida. Depois de definida a estratégia, o bloco **Planejador de Trajetórias** calcula o caminho pelo qual o robô deve se movimentar para executar essa estratégia. Por fim, o bloco **Seguidor de Trajetória** deve definir as velocidades necessárias de cada roda para que o robô siga esse caminho.

O bloco de Transmissão é responsável por enviar os valores de velocidades desejadas para cada robô, que deve executá-las através de um controlador PID. Por limitações do sistema de transmissão implementado pela Equipe de Robótica Titans, só podem ser enviados valores de magnitude até 255. Ao executarem os movimentos, a câmera digital registra o novo estado da partida e, assim, é fechado o ciclo de funcionamento do sistema.

O presente trabalho aborda o desenvolvimento dos blocos Gerenciador de Estratégias e Seguimento de Trajetória, contornados pelos blocos tracejados em amarelo e verde da Figura 3.6.

As seguintes seções explicam cada bloco separadamente.

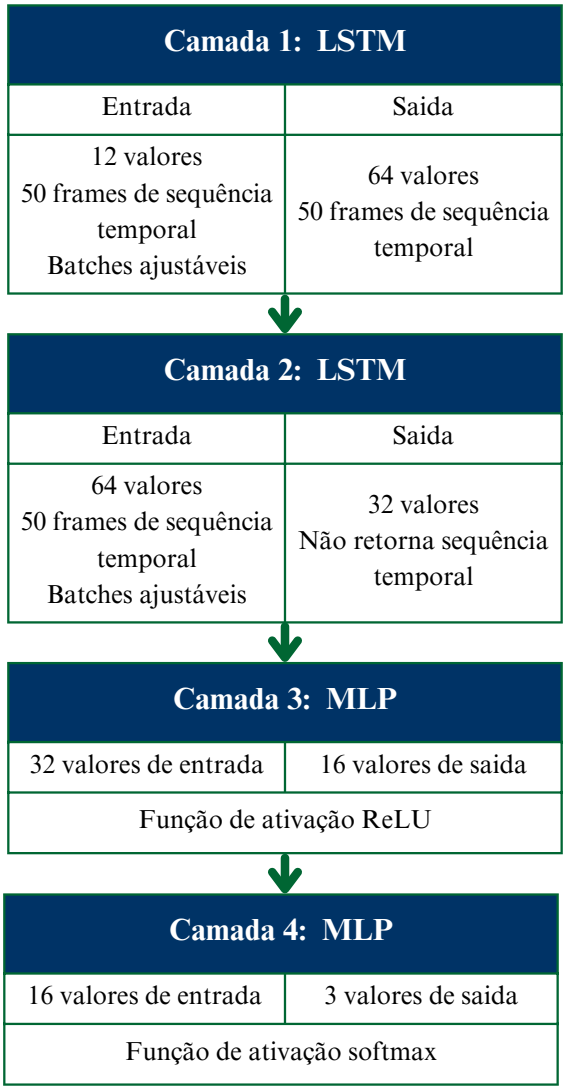
3.4 Gerenciador de Estratégias

Este módulo é responsável por iniciar o controle do robô, definindo a estratégia que ele deve seguir com base no estado atual da partida. Para isso, o bloco processa dados como as posições e orientações de todos os robôs, a posição da bola e sua velocidade, determinando o comportamento mais adequado para o robô adotar.

A implementação desse processo é realizada por meio de uma rede neural artificial composta por quatro camadas, conforme ilustrado na Figura 3.7. As duas primeiras camadas são redes LSTM, operando com uma sequência temporal de 50 frames. A primeira camada recebe 12 sinais de entrada, que incluem as posições (x e y) e as orientações (φ) dos dois robôs e da bola em campo, além das velocidades decompostas (v_x e v_y) e da velocidade linear (V) da bola. Essa camada gera 64 saídas, que são processadas pela segunda camada, responsável por reduzir a dimensionalidade do sinal. Nessa etapa, a segunda camada recebe 64 entradas e produz 32 saídas; entretanto, a estrutura sequencial do dado não é mais preservada.

A terceira camada é uma rede MLP com 32 sinais de entrada, função de ativação ReLU e 16 neurônios de saída, que alimentam a quarta camada. Esta última é composta por outra MLP com função de ativação *softmax* e 3 neurônios de saída, representando

Figura 3.7 – Arquitetura da rede neural de gerenciamento de estratégias



Fonte: Do autor.

os três comportamentos possíveis para os robôs, conforme detalhado na Tabela 3.2. Cada comportamento define também a posição final desejada para o robô, que será enviada aos blocos subsequentes para execução.

Tabela 3.2 – Tabela de relação entre saída da rede neural e comportamento desejado para os robôs

Saída da Rede	Comportamento	Posição Desejada
0	Goleiro	$\{x = -0.7, y = y_{bola}\}$
1	Zagueiro	$\{x = x_{bola}, y = y_{bola}\}$
2	Atacante	$\{x = 0.7, y = 0.7\}$

3.4.1 Criação da Base de Dados de Treinamento

O treinamento da rede foi realizado utilizando a metodologia de Aprendizagem por Demonstração. Para isso, foram empregados dois robôs Corobeu, um identificado com a cor azul e outro com a cor amarela, controlados remotamente por operadores humanos por meio de conectividade Bluetooth. Cada operador foi responsável por controlar um dos robôs, fazendo com que assumissem um dos comportamentos possíveis: atacante, zagueiro ou goleiro.

Durante a geração dos dados de treinamento, o sistema de visão computacional registrou as posições de ambos os robôs, bem como a posição e a velocidade da bola. Esses dados foram posteriormente pós-processados para incluir a informação do comportamento adotado por cada robô. O objetivo do modelo de Rede Neural é prever o comportamento do robô identificado como azul.

Para ampliar a base de dados de treinamento, os dados foram duplicados, mas, nessa segunda versão, o comportamento do robô identificado como amarelo foi definido como o alvo da modelagem. Essa abordagem visou enriquecer o conjunto de dados e melhorar a capacidade de generalização do modelo.

3.4.2 Treinamento da Rede Neural

O modelo foi treinado utilizando a Estimação Adaptativa de Momento (do inglês *Adaptative Moment Estimation* (Adam)) como otimizador e a Entropia Cruzada Categórica Esparsa (do inglês *Sparse Categorical Crossentropy*) como função custo. Foram usadas 100 épocas para o treinamento e 20% dos dados foram reservados para validação do treino.

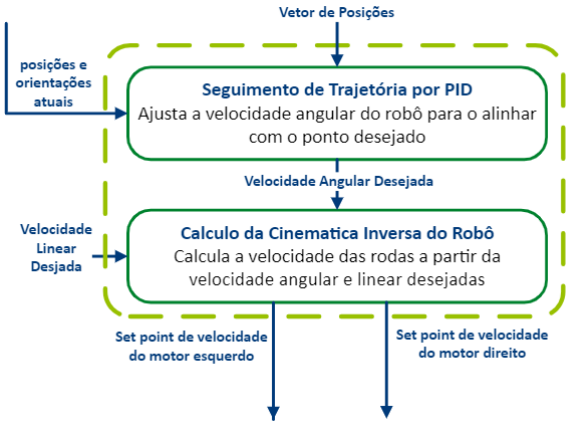
3.5 Seguidor de Trajetórias

O bloco de controle de trajetória, apresentado de forma isolada dos demais sistemas na Figura 3.8, utiliza um controlador PID para conduzir o robô até a posição final desejada. Com base na posição e orientação atuais do robô, bem como na posição-alvo, o erro de orientação (φ_{error}) é determinado por meio da Equação 3.4. Nessa equação, Y_d e X_d representam as coordenadas da posição desejada, enquanto Y_{robo}, X_{robo} e φ_{robo} correspondem à posição e orientação atuais do robô.

$$\varphi_{error} = \tan^{-1}\left(\frac{Y_d - Y_{robo}}{X_d - X_{robo}}\right) - \varphi_{robo} \tag{3.4}$$

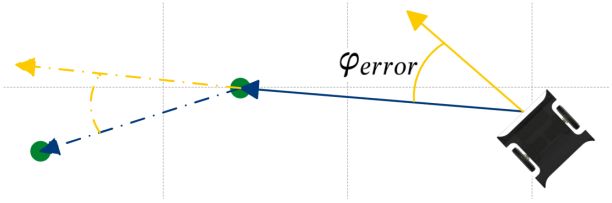
Desta forma, o controlador PID tem, como objetivo, minimizar o φ_{error} atuando sobre a velocidade angular do robô, conforme Equação 3.5, já a velocidade linear do robô é mantida constante. As duas velocidades são, então, aplicadas às equações de cinemática inversa do robô, apresentadas nas Equações 2.3 e 2.4. Assim, o controlador faz com que o robô se

Figura 3.8 – Bloco de controle de trajetórias isolado. Este bloco recebe um vetor de posições desejadas e as posições e orientações atuais do robô e da bola em campo e retorna um valor de velocidades desejadas para o motor esquerdo e direito.



Fonte: Do autor.

Figura 3.9 – Exemplificação do funcionamento do controlador de trajetórias. Os pontos verdes são as posições desejadas, as linhas azuis representam a orientação desejada e as linhas amarelas representam a orientação do robô. As linhas tracejadas são as orientações futuras do robô.



Fonte: Do autor.

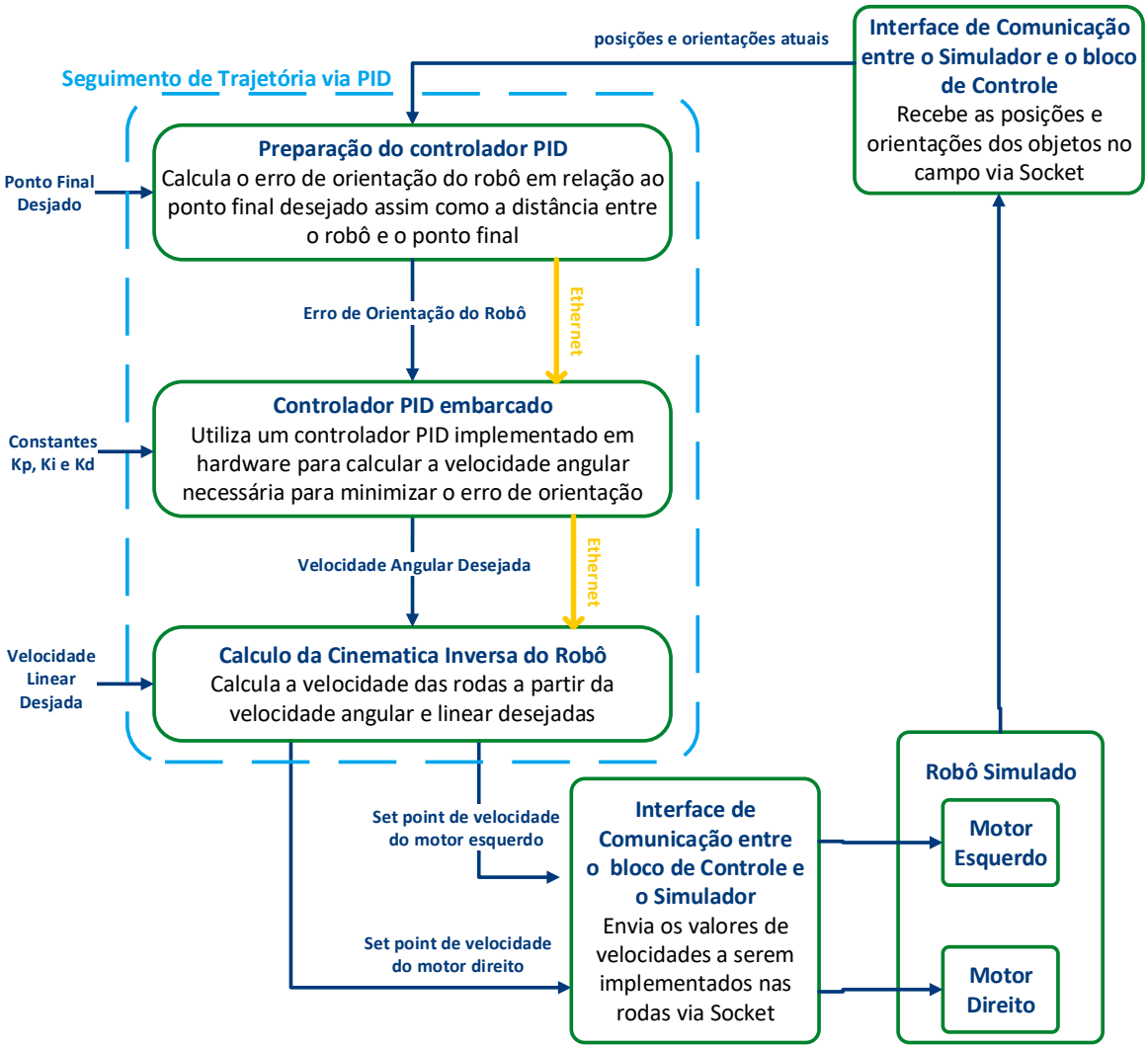
movimente enquanto ajusta sua orientação para se alinhar com o ponto desejado, conforme apresenta a Figura 3.9.

O controlador PID tem como objetivo minimizar o erro de orientação (φ_{error}) atuando sobre a velocidade angular do robô, conforme descrito na Equação 3.5. Enquanto isso, a velocidade linear do robô é mantida constante. Ambas as velocidades, angular e linear, são aplicadas às Equações 2.3 e 2.4, permitindo que o robô se movimente enquanto ajusta sua orientação para se alinhar ao ponto desejado, como ilustrado na Figura Figura 3.9.

$$\omega = PID\left(\varphi_{error}\right) \tag{3.5}$$

A sintonização deste controlador foi feita de maneira empírica, ajustando as constantes K_p, K_i e K_d de forma a fazer o robô chegar ao seu destino.

Figura 3.10 – Arquitetura do controlador proposto para simulação utilizando o kit de desenvolvimento FPGA Zynq 7020 para acelerar o cálculo do controlador PID.



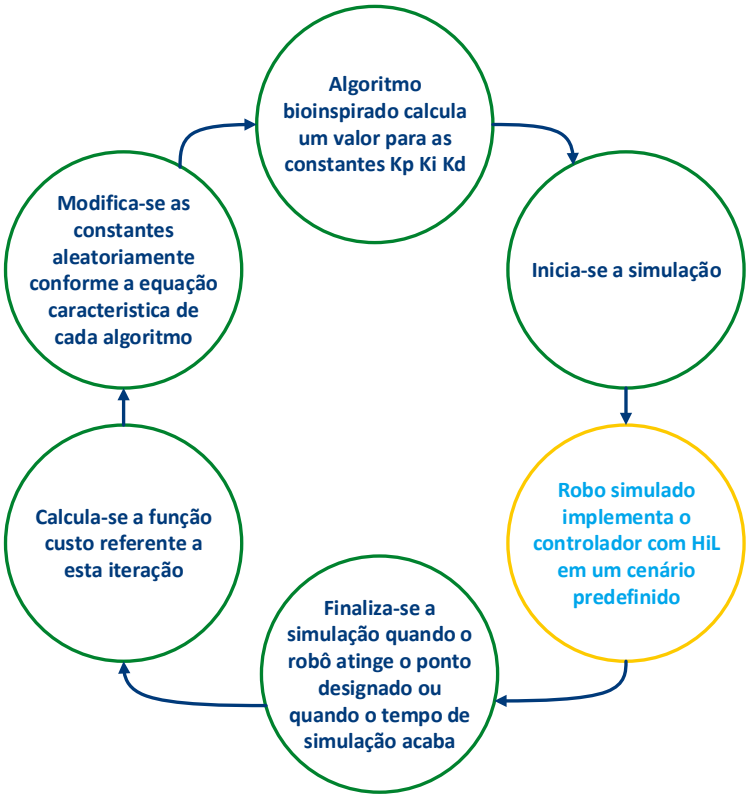
Fonte: Do autor.

3.5.1 Sintonização do Seguidor de Trajetória em Simulação

O controlador de trajetória foi testado em simulação para verificar sua aplicabilidade. Para isso, utilizou-se o cenário descrito na Seção 3.2.1, junto a um controlador PID embarcado em um AMD-Xilinx SoC FPGA Zynq 7020. Esse teste seguiu a metodologia de *Hardware in the Loop* (HiL), onde o *hardware* faz parte do laço de simulação. A arquitetura do sistema de simulação é apresentada na Figura 3.10.

Foram utilizados dois algoritmos de otimização bioinspirados para sintonizar automaticamente o controlador PID para seguimento de trajetória, são eles a Otimização Mariposa-Chama (do inglês *Moth-Flame Optimization* (MFO)) e a otimização por Evolução Diferencial (do inglês *Diferential Evolution* (DE)). Esses algoritmos geram possíveis soluções para as constantes do controlador, que são então enviadas e simuladas. Esse processo itera-

Figura 3.11 – Laço de simulação para sintonização automática do controlador PID de controle de trajetória.



Fonte: Do autor.

tivo continua até que o algoritmo bioinspirado convirja ou atinja uma condição de parada. O laço de simulação é apresentado na Figura 3.11, onde o círculo amarelo implementa a arquitetura de simulação apresentada na Figura 3.10.

A função custo foi definida de forma a ajustar o controlador PID, melhorando o desempenho do sistema, minimizando o erro e alcançando a estabilidade. Para isso, foram utilizados o erro quadrático médio (MSE) dos ângulos e a distância euclidiana, definindo a função custo conforme a Equação 3.6. Nessa equação, φ, x e y representam a orientação e as posições do robô, enquanto $\hat{\varphi}$, $\hat{x}$ e $\hat{y}$ correspondem à orientação e às posições do ponto final desejado.

$$f_{custo} = \sum_{i=1}^{n} \frac{(\varphi_i - \hat{\varphi}_i)}{n} + \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2} \tag{3.6}$$

O espaço de busca dos otimizadores foi limitado a um máximo de 0,8 e um mínimo de 0,0001. Cada algoritmo foi executado com um máximo de 20 iterações. Para garantir a confiabilidade da otimização, todo o processo foi repetido cinco vezes, o que resultou em dois controladores otimizados.

4 Resultados

4.1 Controlador de Velocidades das Rodas

O sistema de primeira ordem, descrito pela Equação 4.1, foi identificado por meio do método de Análise de Resposta ao Degrau, utilizando os dados de resposta do motor direito. O mesmo modelo foi aplicado aos dois motores, dada a similaridade de suas características dinâmicas.

$$F(s) = \frac{4{,}5517}{2{,}75s + 1} \tag{4.1}$$

Após a identificação dos sistemas, o algoritmo PSO foi empregado para determinar as constantes PID do controlador. Essas constantes foram aplicadas ao robô físico e ajustadas para otimizar o controle dos motores. Os resultados obtidos são apresentados na Tabela 4.2. A Figura 4.1 exibe o Lugar Geométrico Das Raízes (LGR) do sistema controlado com as constantes ajustadas, onde se observa que todos os polos estão localizados no semiplano esquerdo, confirmando a estabilidade do sistema. Os polos dos sistemas controlados são detalhados na Tabela 4.3.

As Figuras 4.2, 4.3, 4.4 e 4.5 apresentam as respostas ao degrau de $180RPM$ para os sistemas simulados dos motores direito e esquerdo e os sistemas reais direito e esquerdo, respectivamente.

Para os sistemas simulados, não há sobrepasso, e o erro em regime permanente é desprezível. No entanto, o tempo de acomodação é relativamente alto, próximo de 100 segundos. Por outro lado, no sistema real, observa-se a presença de sobrepasso e erro em regime permanente, mas o tempo de acomodação é significativamente menor, ficando abaixo de 5 segundos, conforme detalhado na Tabela 4.1.

Nas Figuras 4.4 e 4.5, é possível observar a robustez do sistema. Foram introduzidas duas perturbações para cada roda, uma por volta de 16 segundos e outra por volta de 44 segundos. Nota-se que o sistema se recupera rapidamente e retorna à estabilidade, o que evidencia uma sintonia adequada dos controladores PI.

Tabela 4.1 – Métricas de avaliação do controlador PID de velocidade das rodas.

Parâmetro	**Motor Direito**	**Motor Esquerdo**
Sobrepasso	28,96%	64,68%
Erro Máximo em Estado Estacionário	entre −3,17% e 3,17%	entre −8,72% e 9,12%
Tempo de Estabilização	2,8 segundos	4,2 segundos

Tabela 4.2 – Constantes do controlador PID para cada motor.

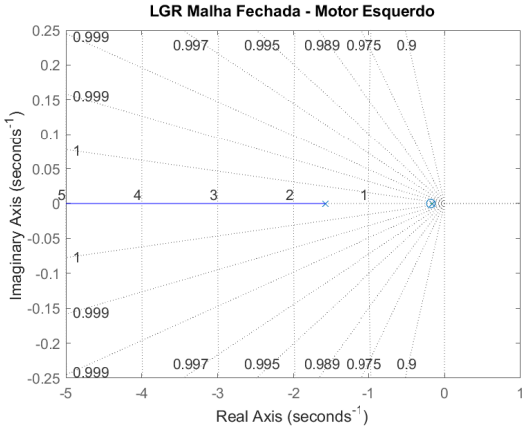
Constante	Motor Direito	Motor Esquerdo
K_p	0,8394	0,8302
K_i	0,1528	0,1508
K_d	0.0	0.0

Tabela 4.3 – Polos em malha fechada para cada motor.

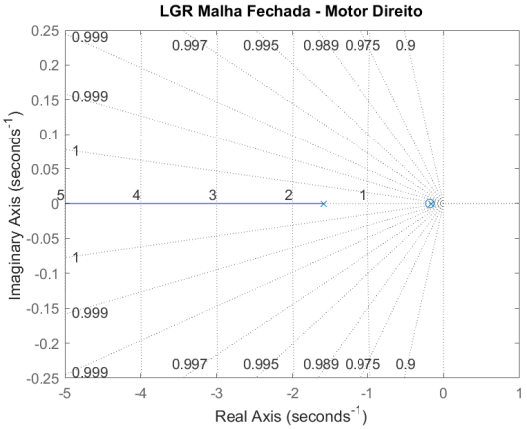
Polo	Motor Direito	Motor Esquerdo
p_1	-1.5798	-1.5944
p_2	-0,1580	-0.1586

Figura 4.1 – Lugares geométricos das raízes para os sistemas controlados dos motores direito e esquerdo.

a LGR do sistema controlado do motor esquerdo.



b LGR do sistema controlado do motor direito.



Fonte: Do autor.

Figura 4.2 – Resposta ao degrau em simulação do motor direito com um valor desejado de 180 RPM.

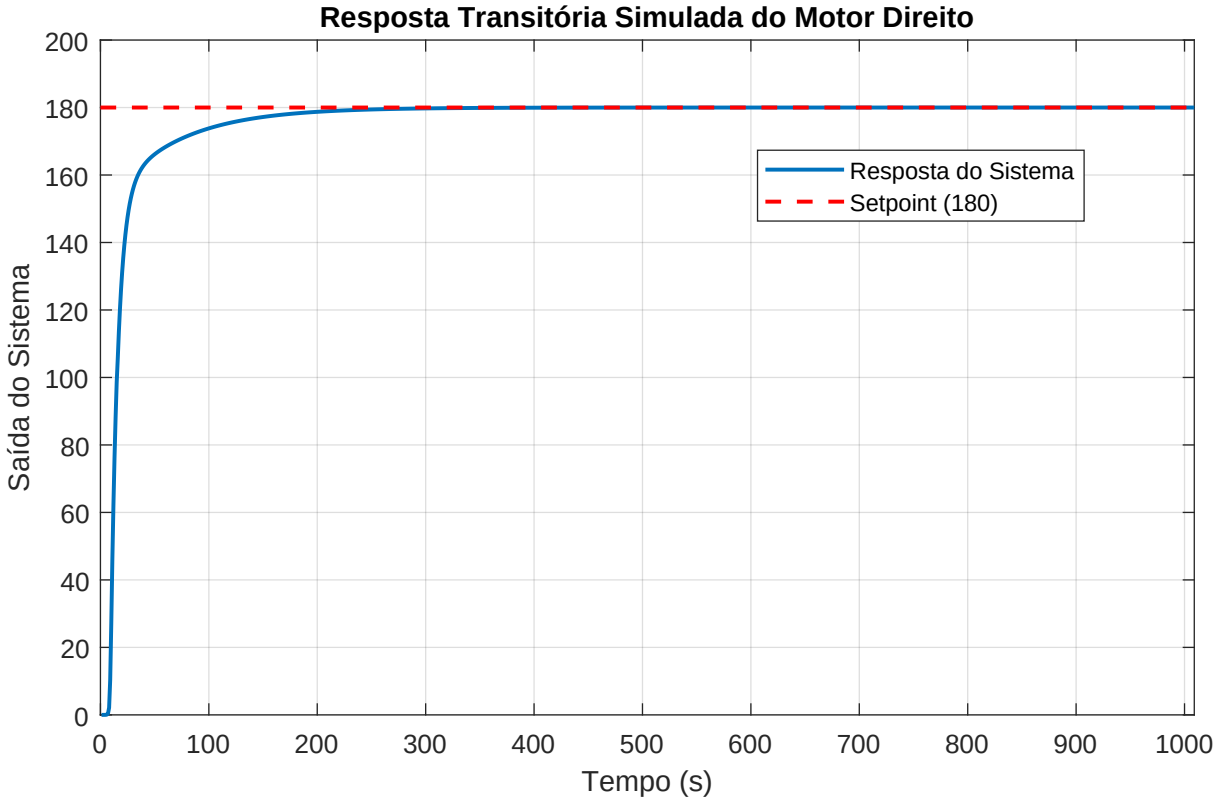
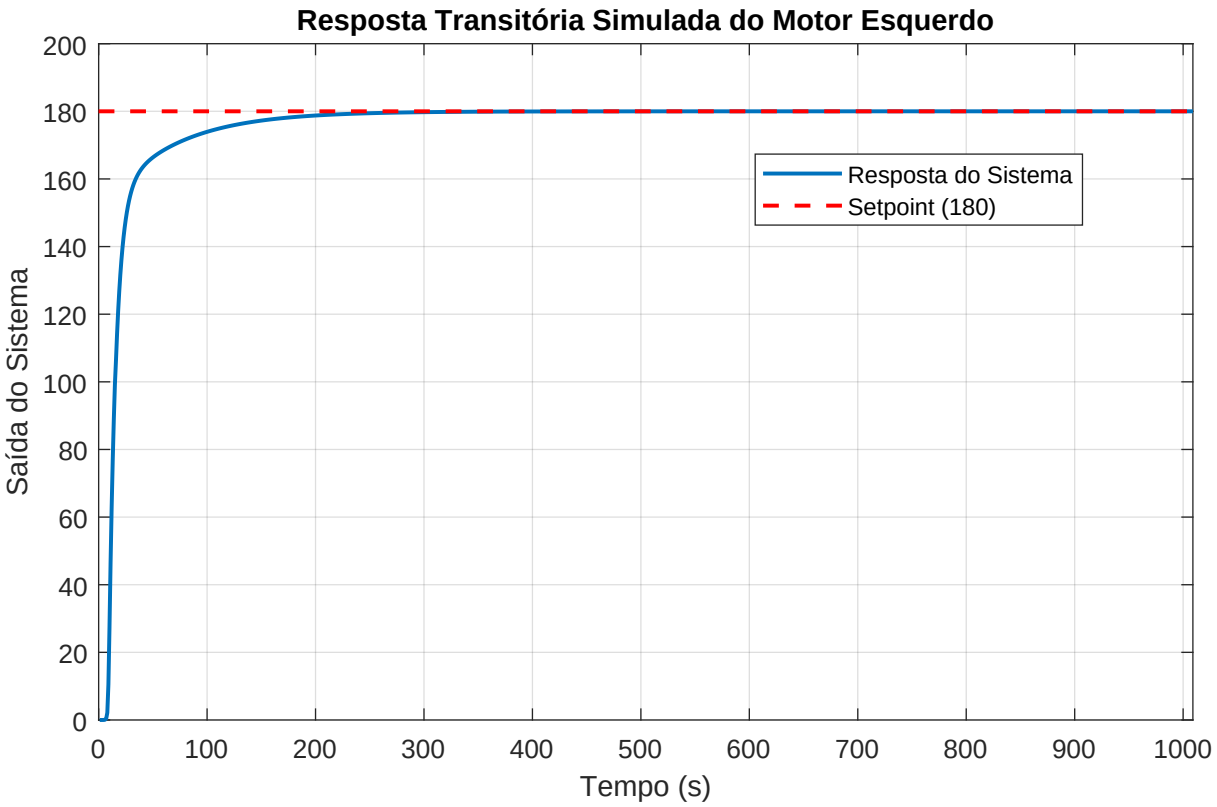
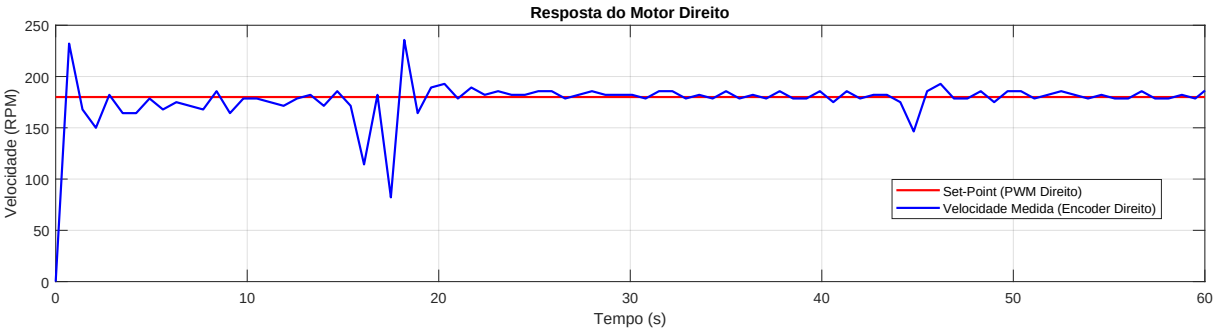


Figura 4.3 – Resposta ao degrau em simulação do motor esquerdo com um valor desejado de 180 RPM.



Fonte: Do autor.

Figura 4.4 – Resposta ao degrau do motor direito com um valor desejado de 180 RPM.

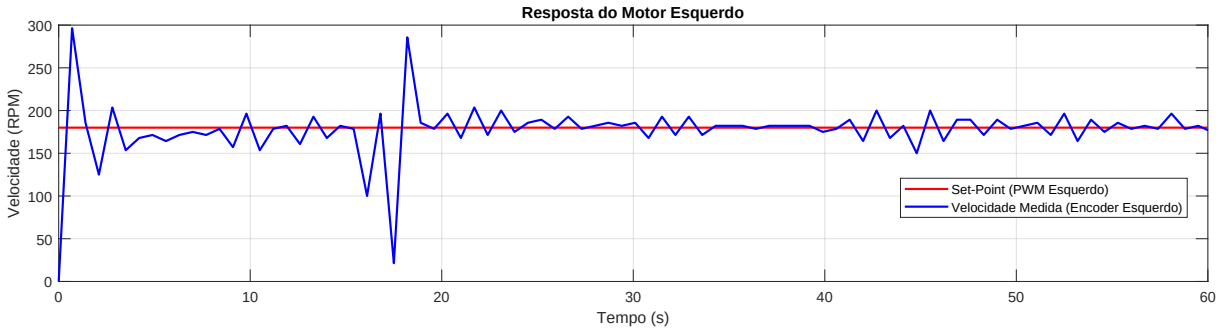


Fonte: Do autor.

Tabela 4.4 – Relatório de classificação do modelo de redes neurais para gerenciamento de estratégias.

Categoria	Precisão	Sensibilidade	Amostras
0	1.00	1.00	661
1	0.99	0.99	176
2	0.99	0.99	188
Acurácia			100%
Média entre as Métricas			99%
Média Ponderada entre as Métricas			99%

Figura 4.5 – Resposta ao degrau do motor esquerdo com um valor desejado de 180 RPM.



Fonte: Do autor.

4.2 Treino do Gerenciador Estratégias

O relatório de classificação do modelo é apresentado na Tabela 4.4, nele podemos ver que a precisão para cada um dos comportamentos está excelente, com 99% de acerto para os comportamentos de zagueiro e atacante (1 e 2 respectivamente) e 100% de precisão para os comportamentos de goleiro (etiqueta 0).

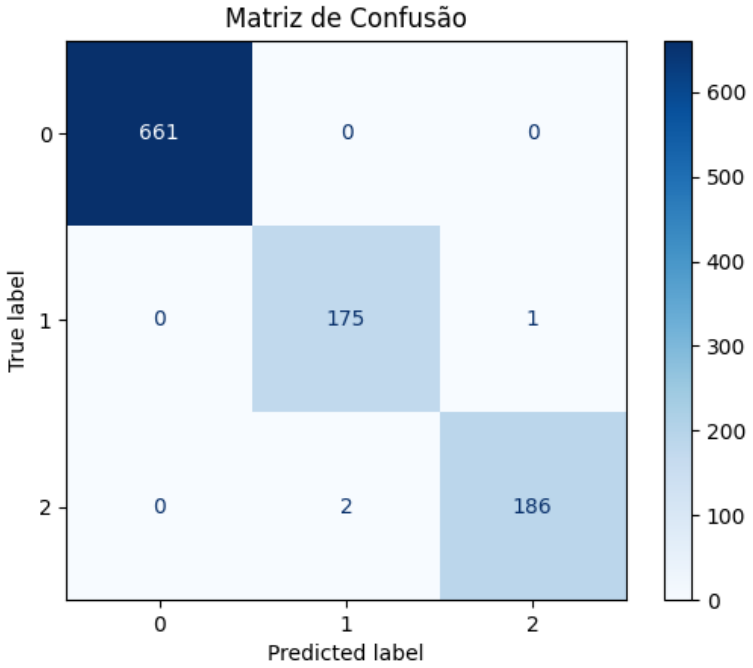
A matriz de confusão, apresentada na Figura 4.6 corrobora com este resultado, é possível notar que todos os comportamentos de goleiro foram devidamente categorizados como tal e apenas um comportamento de zagueiro foi categorizado como atacante e apenas dois de atacante foram categorizados como zagueiros.

Por fim, o tempo médio de inferência da rede foi medido em 0.016554s, ou aproximadamente 16ms, um valor promissor para a aplicação desejada, tendo em vista que a taxa de atualização do sistema de visão computacional é de 30ms.

4.2.1 Validação das Redes

Este modelo de rede foi aplicado a outros cenários de partidas adquiridos da mesma maneira que os cenários utilizados para o treinamento. O resultado desta aplicação está presente nos *gif*s presentes na seguinte URL: https://github.com/team-titans-unb/VSSS/tree/robot-identification/Controle/05-Modelagem/resultados/gif. Nota-se que a rede obteve

Figura 4.6 – Matriz de confusão do modelo treinado para definição de comportamento do robô.



Fonte: Do autor.

sucesso ao identificar qual comportamento o robô azul estava implementando nas ocasiões apresentadas.

4.3 Controlador de Trajetória

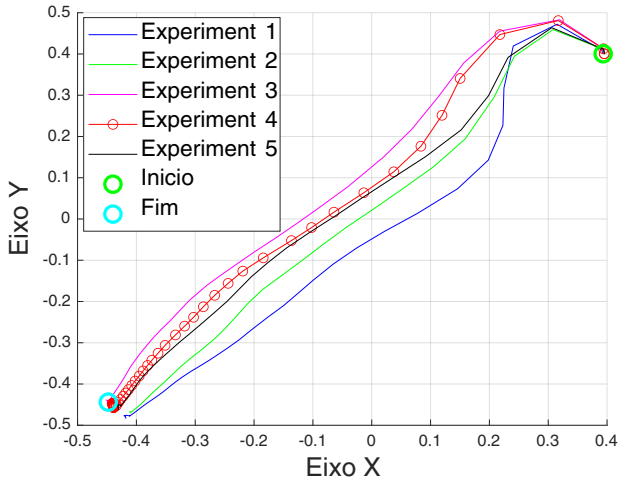
4.3.1 Simulação

As constantes obtidas pelos algoritmos de otimização são apresentadas na tabela 4.5, assim como o menor valor da função custo obtido, o tempo de simulação necessário para concluir os 5 experimentos e o número do melhor experimento para cada algoritmo. Percebe-se que o controlador sintonizado pelo algoritmo MFO obteve melhor resultado, fato demonstrado, também, pela comparação entre as Figuras 4.7 e 4.8 que apresentam os melhores trajetos implementados pelo robô em cada experimento durante a sintonização utilizando o MFO e o DE, respectivamente.

Tabela 4.5 – Constantes do controlador PID obtidas por cada controlador.

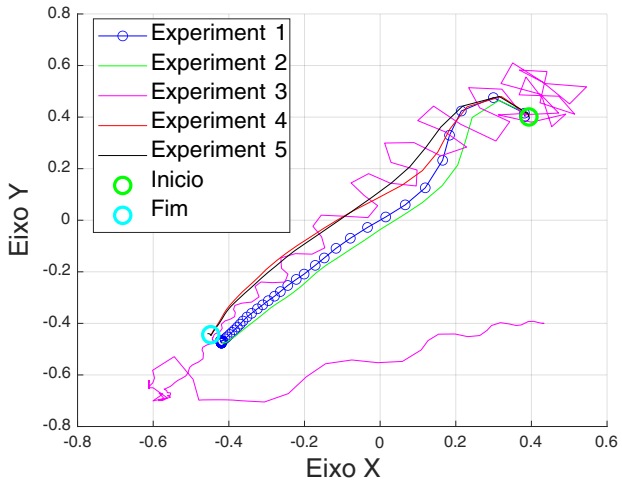
Parâmetro	MFO	DE
K_p	0,3278	0,3629
K_i	0,0774	0,1891
K_d	0,0003	0,0001
Custo	0,3736	0,4174
Tempo de Simulação	80 min	165 min
Melhor Experimento	4	1

Figura 4.7 – Melhores trajetos implementados pelo robô em cada experimento durante a sintonização utilizando o algoritmo MFO.



Fonte: Do autor.

Figura 4.8 – Melhores trajetos implementados pelo robô em cada experimento durante a sintonização utilizando o algoritmo DE.



Fonte: Do autor.

Tabela 4.7 – Constantes do controlador PID de seguimento de trajetória.

Constante	Valor
K_p	30
K_i	15
K_d	7
e_f	5,477

Os resultados da implementação com HiL são apresentados em duas listas de reprodução do *YouTube* apresentadas na Tabela 4.6. Nos vídeos de treinamento, observa-se que, nas iterações iniciais, os algoritmos bioinspirados apresentam um desempenho abaixo do esperado, impedindo o robô de alcançar o ponto final desejado. Entretanto, com o avanço das iterações, os algoritmos convergem para soluções melhores, resultando, nas iterações finais, em um controlador capaz de direcionar o robô até a posição final desejada.

Tabela 4.6 – Listas de reprodução com os vídeos de sintonização dos controladores.

MFO
https://www.youtube.com/playlist?list=PLPnSV069ZnAJDGYxc-8qK_a1ffl0dz1zL
DE
https://www.youtube.com/playlist?list=PLPnSV069ZnALDO2oHKTn4n6Jq6vndYD9i

4.3.2 Implementação Física

Para o Controlador de Trajetória, a velocidade linear foi ajustada para $340\frac{cm}{s}$ e a velocidade angular foi saturada entre $-68RPM$ e $68RPM$. A parte integrativa foi saturada entre -5 e 5, enquanto que à parte derivativa foi aplicado um filtro para diminuir o ruído presente na derivação de acordo com as Equações 4.2, 4.3 e 4.4, onde e_f é definido como a maior das raízes das constantes do controlador. Desta forma, as constantes do controlador PID encontradas são apresentadas na Tabela 4.7.

$$(1-\alpha) = e^{\frac{-\Delta t}{e_f}} \tag{4.2}$$

$$f_t = (1-\alpha) * f_{t-1} + \alpha * erro \tag{4.3}$$

$$\delta erro = \frac{f_t - f_{t-1}}{\Delta t} \tag{4.4}$$

Os resultados deste controlador estão apresentados nas Figuras 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 e 4.15. Foram realizados diversos experimentos, variando a posição inicial do robô no campo, sua orientação e o ponto final desejado. Para cada experimento, são apresentadas as figuras correspondentes ao trajeto percorrido pelo robô, além de um vídeo que registra a execução desse trajeto.

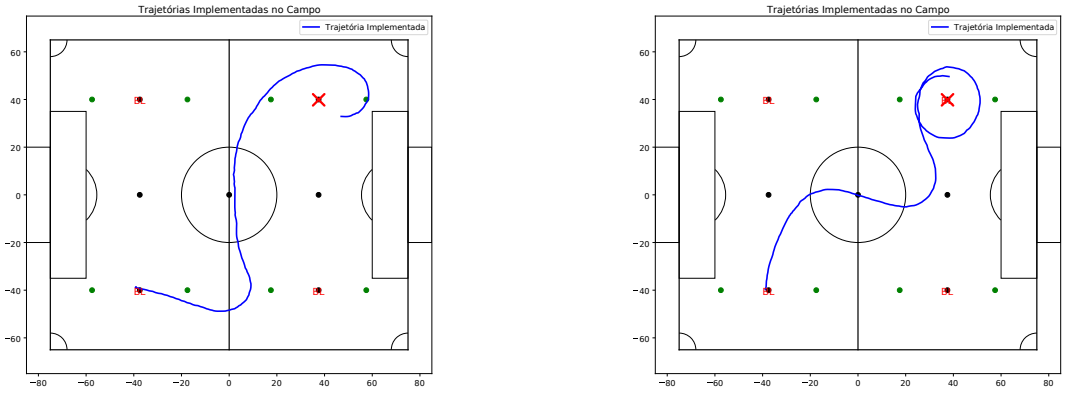
As Figuras 4.9b e 4.10a mostram que, mesmo com o mesmo ponto inicial e a mesma orientação, a trajetória executada varia de acordo com a implementação. Observa-se, também, uma tendência do robô a rotacionar no sentido horário. As Figuras 4.12a e 4.13a indicam que, quando o ponto final está à esquerda do robô, a trajetória se torna mais suave, com menos curvas. Nos demais casos, o trajeto apresenta mais desvios e oscilações.

As Figuras 4.14a, 4.14b, 4.15a e 4.15b ilustram uma trajetória do semi-plano direito para o semi-plano esquerdo do campo. O controlador não se mostrou eficiente nesse cenário, pois, conforme observado nas Figuras 4.14a e 4.14b, o robô oscilou durante o percurso e não alcançou o objetivo final. Para mitigar essa falha, no experimento das Figuras 4.15a e 4.15b, foi utilizada uma funcionalidade do Sistema de Visão Computacional que inverte as posições dos objetos no campo, permitindo uma melhora no desempenho do controlador, ainda que não tenha alcançado a posição final desejada.

Figura 4.9 – Resultados do controlador de trajetórias, primeira parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado.

a Clique aqui para ver o vídeo desta implementação: https://youtu.be/Avzj463amLI?si=udloj5ca0VPLQ0aK

b Clique aqui para ver o vídeo desta implementação: https://youtu.be/HIq0NM3epwk?si=f_czoDkqwKELgHxR

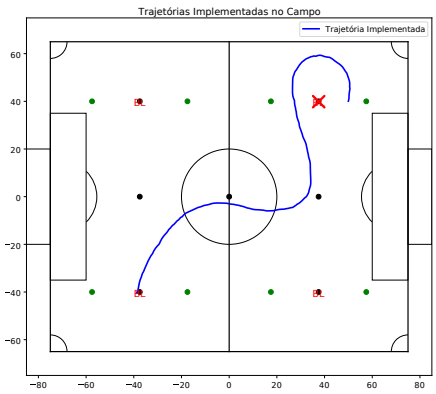


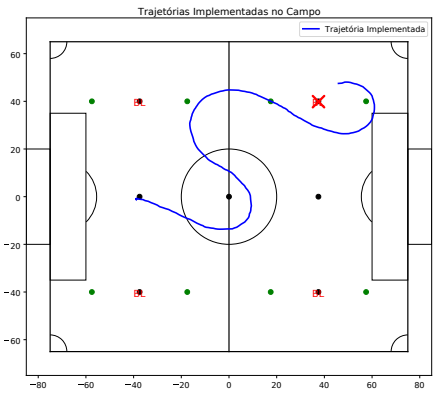
Fonte: Do autor.

Figura 4.10 – Resultados do controlador de trajetórias, segunda parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado.

a Clique aqui para ver o vídeo desta implementação: https://youtu.be/ofJA2iF_AUc?si=PMuwcYy8g2rJnpIx

b Clique aqui para ver o vídeo desta implementação: https://youtu.be/riek95m-RRA?si=96TMTFwPKGD-aQSx



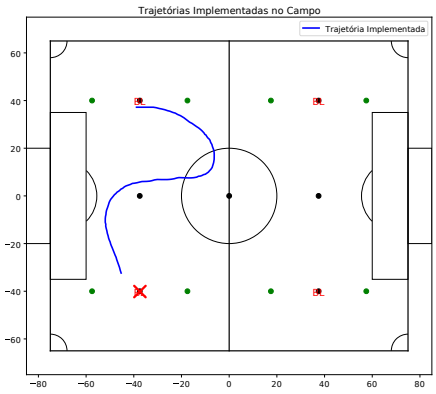


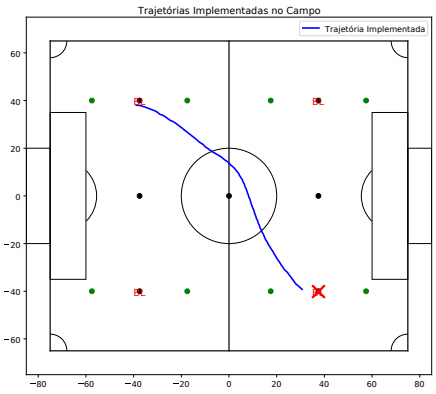
Fonte: Do autor.

Figura 4.11 – Resultados do controlador de trajetórias, terceira parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado.

a Clique aqui para ver o vídeo desta implementação: https://youtu.be/kTfDeWKLe_E?si=g-7ZPBM67ltGiWG0

b Clique aqui para ver o vídeo desta implementação: https://youtu.be/PdY20WMsAIk?si=vZVPZ-RMD6i3Vdot



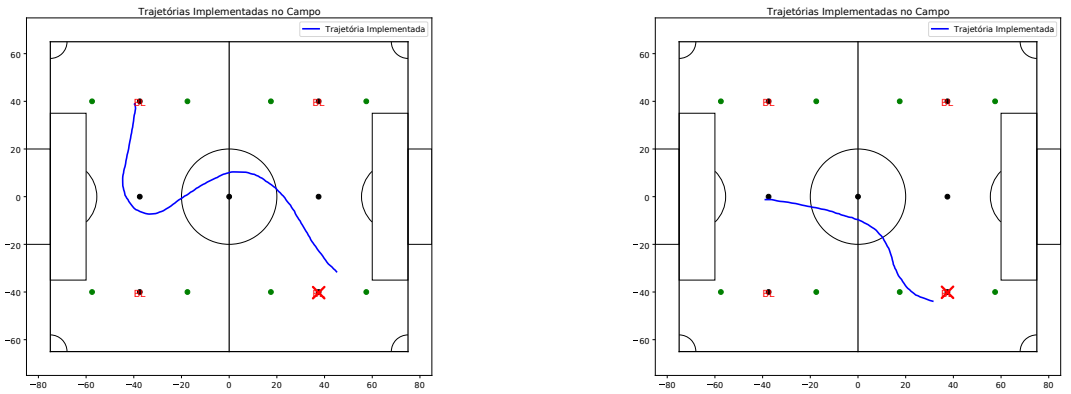


Fonte: Do autor.

Figura 4.12 – Resultados do controlador de trajetórias, quarta parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado.

a Clique aqui para ver o vídeo desta implementação: https://youtu.be/wHNQRvIkdLM?si=SCoerZ9FC7faNa6t

b Clique aqui para ver o vídeo desta implementação: https://youtu.be/2w2LTOnvJVo?si=E8fx72khhqiDsvWm

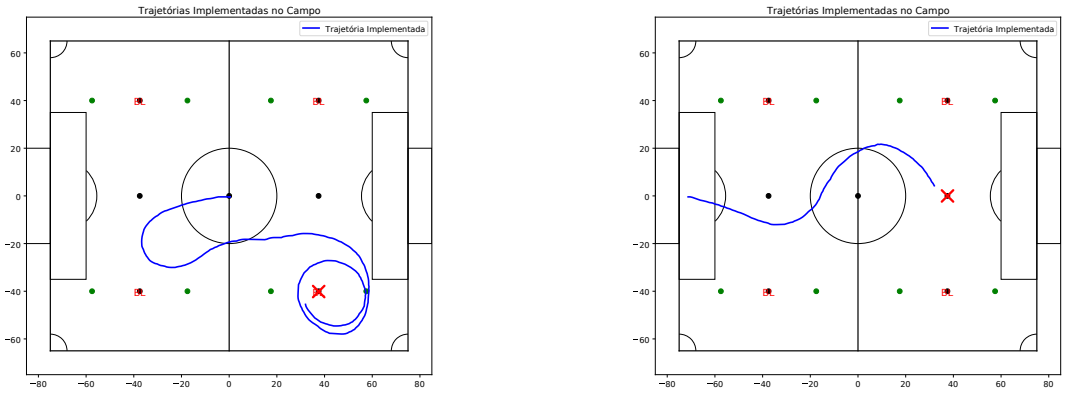


Fonte: Do autor.

Figura 4.13 – Resultados do controlador de trajetórias, quinta parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado.

a Clique aqui para ver o vídeo desta implementação: https://youtu.be/Pif_OhZLqdo?si=_LJwDsngtJ5yU8Lu

b Clique aqui para ver o vídeo desta implementação: https://youtu.be/QYB55pRFVzc?si=uCmNnWCaMnhQ8tRj

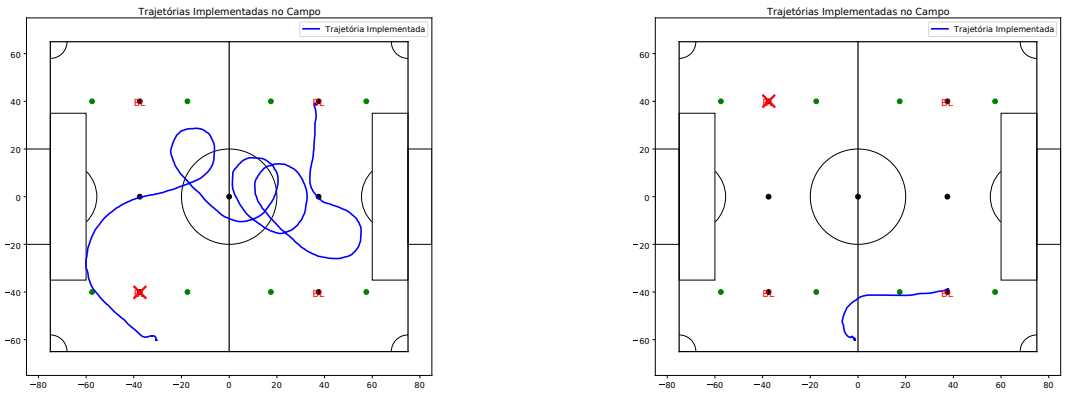


Fonte: Do autor.

Figura 4.14 – Resultados do controlador de trajetórias, sexta parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado.

a Clique aqui para ver o vídeo desta implementação: https://youtu.be/9SMHUKL2WaM?si=axh8aDNQqnEsAKrC

b Clique aqui para ver o vídeo desta implementação: https://youtu.be/dmssZkDULsM?si=IeGshpOdEyghvdiV

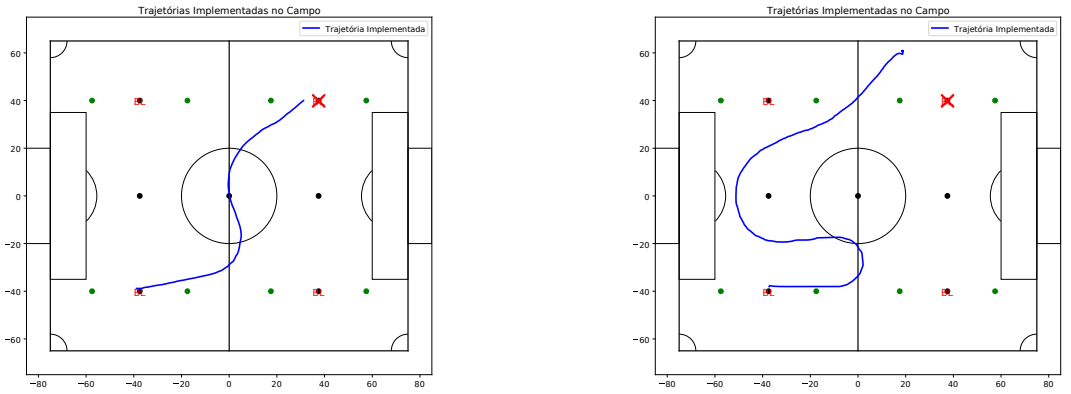


Fonte: Do autor.

Figura 4.15 – Resultados do controlador de trajetórias, sétima parte. Em azul, tem-se a trajetória executada pelo robô e o 'x' vermelho representa o ponto final desejado.

a Clique aqui para ver o vídeo desta implementação: https://youtu.be/B9lAxeZoY04?si=QBeKVhew3BBdJ5Ty

b Clique aqui para ver o vídeo desta implementação: https://youtu.be/AJEvQyqNUdQ?si=hJjMmz6DkR66DJb0



Fonte: Do autor.

5 Conclusões

5.1 Conclusões Gerias

O presente trabalho propôs um sistema de controle de trajetória e gerenciamento de estratégias baseado em Aprendizado por Demonstração aplicado ao futebol de robôs móveis, utilizando redes neurais artificiais e algoritmos de controle bioinspirados. Ao longo do desenvolvimento, diversos desafios foram enfrentados, e os resultados obtidos demonstraram tanto as vantagens da abordagem adotada quanto os pontos que ainda necessitam de melhorias.

Os controladores de velocidade dos motores do robô são adequados para este cenário, embora o motor esquerdo utilize um controlador menos eficiente, que apresenta um tempo de assentamento maior e um sobrepasso mais elevado. Essa diferença de desempenho entre os motores pode estar causando a tendência do robô de rotacionar no sentido horário, como observado nos resultados da Seção 4.3. Apesar dessa limitação, o robô ainda mantém um controle adequado de sua movimentação.

O gerenciador de estratégias demonstrou um excelente desempenho, sendo capaz de abstrair corretamente os comportamentos ensinados. Entretanto, ainda existem melhorias possíveis, especialmente na coleta e tratamento de dados de treinamento, na criação de novos comportamentos e na obtenção de mais dados de partidas reais para enriquecer o aprendizado do modelo.

O controlador de trajetória apresentou resultados razoáveis, mas ainda é instável para esta aplicação. Embora ele consiga atingir o objetivo na maioria das vezes, a instabilidade compromete a confiabilidade do sistema, tornando necessário um ajuste maior nas constantes do controlador para garantir um desempenho mais consistente e previsível.

A metodologia Aprendizado por Demonstração se mostrou vantajosa para esta aplicação. Com essa abordagem, foi possível evitar o desenvolvimento de algoritmos de controle complexos e baseados em regras fixas, o que permitiu que o sistema aprendesse comportamentos desejados apenas a partir da demonstração de exemplos humanos.

Infelizmente, a integração entre os módulos não foi concluída, devido às limitações do controlador de trajetória. Como o robô ainda não consegue executar corretamente os comportamentos planejados, a comunicação entre os diferentes sistemas permanece pendente.

5.2 Contribuições

A principal contribuição deste trabalho foi o desenvolvimento do robô Corobeu como um sistema autônomo, capaz de ser controlado remotamente por algoritmos executados em um computador, como a rede de gerenciamento de estratégias. Isso abre caminho para diversas aplicações futuras, como controle de enxames, monitoramento de máquinas em fábricas e transporte de mercadorias em armazéns.

Em resumo, a proposta atendeu parcialmente aos objetivos iniciais. Embora o sistema ainda não esteja 100% finalizado, ele constitui uma base sólida para futuras melhorias e integrações, permitindo avanços na implementação de futebol de robôs autônomos e aplicações correlatas.

Todos os arquivos desenvolvidos neste trabalho estão disponíveis no repositório remoto do GitHub em: https://github.com/team-titans-unb/VSSS.

5.3 Limitações

O desenvolvimento do sistema envolveu desafios significativos, tais como:

- Problemas mecânicos e eletrônicos do robô, que impossibilitaram a implementação do controlador antes de serem corrigidos.
- Processo de identificação de sistemas, que exigiu um firmware estável antes de produzir bons resultados.
- Criação e tratamento de dados de treinamento, um processo trabalhoso que exigiu a inserção manual dos comportamentos desejados.
- Sintonização do controlador de trajetórias sem identificação de sistema, o que exigiu restrições na velocidade angular do robô e ajuste nos tempos de comunicação do sistema, devido à diferença entre a taxa de atualização da visão computacional ($30ms$) e a frequência de recepção de comandos pelo motor ($700ms$).

5.4 Trabalhos Futuros

Diversas melhorias podem ser implementadas para avançar o projeto, incluindo:

- Reformulação estrutural do robô para aumentar a estabilidade e facilitar o controle e a montagem.
- Desenvolvimento de novos comportamentos para o gerenciador de estratégias.
- Obtenção e utilização de mais dados de partidas reais para enriquecer o aprendizado do modelo do gerenciador de estratégias.

- Investigação de novas topologias de redes neurais, incluindo o uso de filtro de Kalman para estimação do comportamento desejado.
- Aprimoramento da sintonização do controlador de trajetória, explorando algoritmos bioinspirados para ajustar parâmetros como a saturação da integral.
- Ajuste fino do controlador PI de velocidade do motor esquerdo e análise do impacto na tendência do robô de rotacionar no sentido horário.
- Implementação de algoritmos bioinspirados embarcados para sintonização *online* dos controladores PID de velocidade dos motores.
- Melhoria na estimação dos encoders, substituindo a contagem de pulsos em 700 ms pela medição da distância entre pulsos, reduzindo o tempo de operação do sistema. Para isso, será necessário refazer a identificação dos motores.
- Integração do planejador de estratégias com o controlador de trajetórias.

Referências

AMBHORE, S. A comprehensive study on robot learning from demonstration. *In*: **2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)**. [*S.l.: s.n.*], 2020. p. 291–299. Citado na p. 27.

BAKKER, P.; KUNIYOSHI, Y. Robot see, robot do : An overview of robot imitation. **AISB96 Workshop on Learning in Robots and Animals**, 05 1996. Citado na p. 27.

BALDãO, T. P.; MAXIMO, M. R. O. A.; CESAR, C. de A. C. Decision-making for 5x5 very small size soccer teams. *In*: **2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)**. [*S.l.: s.n.*], 2020. p. 1–6. Citado nas pp. 16, 33 e 34.

BRANDãO, B.; LIMA, T. W. D.; SOARES, A.; MELO, L.; MAXIMO, M. R. O. A. Multiagent reinforcement learning for strategic decision making and control in robotic soccer through self-play. **IEEE Access**, v. 10, p. 72628–72642, 2022. Citado nas pp. 16, 33 e 34.

BRäUNL, T. **Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems**. 2. ed. [*S.l.*]: Springer Berlin, Heidelberg, 2006. XIV, 458 p. (Computer Science, Computer Science (R0)). Copyright Information: Springer-Verlag Berlin Heidelberg 2006. ISBN 978-3-540-34319-6. Citado nas pp. 22, 24, 25, 28 e 31.

CAO, Y. U.; FUKUNAGA, A. S.; KAHNG, A. Cooperative mobile robotics: Antecedents and directions. **Autonomous Robots**, v. 4, n. 1, p. 7–27, March 1997. ISSN 1573-7527. Disponível em: https://doi.org/10.1023/A:1008855018923. Citado na p. 21.

CORKE, P. **Robotics, Vision and Control**: Fundamental algorithms in matlab® second, completely revised, extended and updated edition. 2. ed. Springer Cham, 2017. XXIX, 693 p. (Springer Tracts in Advanced Robotics). ISBN 978-3-319-54413-7. Disponível em: https://doi.org/10.1007/978-3-319-54413-7. Citado nas pp. 22, 23, 24, 25 e 26.

CORREIA, J. V. M. **Aprendizado por reforço assistido aplicado ao controle de robôs de acionamento diferencial**. 39 f. Monografia (Monografia (Graduação em Ciência da Computação)) — Faculdade de Ciências, Universidade Estadual Paulista "JÚLIO DE MESQUITA FILHO", Bauru, 2023. Disponível em: http://hdl.handle.net/11449/239124. Citado nas pp. 33 e 34.

DREW, D. S. Multi-agent systems for search and rescue applications. **Current Robotics Reports**, Springer, v. 2, n. 2, p. 189–200, 2021. ISSN 2662-4087. Disponível em: https://doi.org/10.1007/s43154-021-00048-3. Citado na p. 16.

DUPONT, P. E.; NELSON, B. J.; GOLDFARB, M.; HANNAFORD, B.; MENCIASSI, A.; O'MALLEY, M. K.; SIMAAN, N.; VALDASTRI, P.; YANG, G.-Z. A decade retrospective

of medical robotics research from 2010 to 2020. **Science Robotics**, v. 6, n. 60, p. eabi8017, 2021. Disponível em: https://www.science.org/doi/abs/10.1126/scirobotics.abi8017. Citado na p. 16.

DZEDZICKIS, A.; SUBAČIūTė-ŽEMAITIENė, J.; ŠUTINYS, E.; SAMUKAITė-BUBNIENė, U.; BUČINSKAS, V. Advanced applications of industrial robotics: New trends and possibilities. **Applied Sciences**, v. 12, n. 1, 2022. ISSN 2076-3417. Disponível em: https://www.mdpi.com/2076-3417/12/1/135. Citado na p. 16.

FARINELLI, A.; IOCCHI, L.; NARDI, D. Multirobot systems: a classification focused on coordination. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 34, n. 5, p. 2015–2028, 2004. Citado na p. 21.

FERREIRA, R. H. X.; SILVA, P. L. da; OLIVEIRA, V. C. R. de; JUNIOR, C. D. da S.; FREITAS, E. J. de R. Initial development of IEEE very small size soccer robots: Electromechanics, computer vision system and navigation strategy. **The Journal of Engineering and Exact Sciences**, v. 9, n. 10, p. 18148–18148, 2023. Citado na p. 16.

FREITAS, E. J. d. R.; COHEN, M. W.; GUIMARãES, F. G. An autonomous mobile robot path planner using spline curves and differential evolution. *In*: **2023 IEEE Latin American Conference on Computational Intelligence (LA-CCI)**. [*S.l.: s.n.*], 2023. p. 1–6. Citado na p. 16.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. Bookman Editora, 2001. ISBN 9788577800865. Disponível em: https://books.google.com.br/books?id=bhMwDwAAQBAJ. Citado nas pp. 28 e 29.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, p. 1735–1780, 11 1997. Citado na p. 30.

IEEE Standard Definitions of Navigation Aid Terms. **IEEE Std 172-1983**, p. 1–34, 1983. Citado na p. 23.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. *In*: **Proceedings of ICNN'95—International Conference on Neural Networks**. Perth, Australia: [*s.n.*], 1995. p. 1942–1948. Disponível em: https://doi.org/10.1109/ICNN.1995.488968. Citado na p. 32.

MACKWORTH, A. K. On seeing robots. *In*: BASU, A.; LI, X. (Ed.). **Computer Vision: Systems, Theory and Applications**. Singapore: World Scientific Press, 1993. p. 1–13. Reprinted in P. Thagard (ed.), Mind Readings, MIT Press, 1998. Citado na p. 16.

MARTINS, F. B. **Exploring multi-agent deep reinforcement learning in IEEE very small size soccer**. Dissertação (Dissertação de Mestrado) — Universidade Federal de Pernambuco, 2023. Citado na p. 33.

MARTINS, F. B.; MEDEIROS, H. R. de; SANTOS, L. H. C.; MACHADO, M. G.; BRAGA, P. H. M.; DELGADO, R. d. A.; BARROS, T. d. S.; BASSANI, H. d. F.; BARROS, E. N.

d. S. RoboCIn IA description paper. *In*: . [*s.n.*], 2020. p. 1–5. Disponível em: https://drive.google.com/file/d/1eXhvWIobnj6UIcc4j9Oz_hVGRf111mmQ/view. Acesso em: 20 de maio de 2024. Citado na p. 34.

MEDEIROS, T. F. d.; YONEYAMA, T.; MáXIMO, M. R. O. A. Deep reinforcement learning applied to IEEE's very small size soccer in cooperative attack strategy. *In*: **2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)**. [*S.l.: s.n.*], 2023. p. 349–354. Citado na p. 35.

MEDEIROS, T. F. de; MáXIMO, M. R. O. de A.; YONEYAMA, T. Deep reinforcement learning applied to IEEE very small size soccer strategy. *In*: **2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)**. [*S.l.: s.n.*], 2020. p. 1–6. Citado nas pp. 16, 33 e 35.

MIRJALILI, S.; DONG, J. S. Introduction to nature-inspired algorithms. *In*: ____. **Nature-Inspired Optimizers: Theories, Literature Reviews and Applications**. Cham: Springer International Publishing, 2020. p. 1–5. ISBN 978-3-030-12127-3. Disponível em: https://doi.org/10.1007/978-3-030-12127-3_1. Citado na p. 31.

OLIVEIRA, A.; SILVA, Y.; SILVA, L.; BRIOSO, V.; BRAUN, C.; MOURA, J.; CASTRO, A. Robotech-UFPA IEEE very small size soccer (VSSS) team description. *In*: . [*S.l.: s.n.*], 2020. Citado nas pp. 16, 33 e 34.

OLIVEIRA, L. F. P.; MOREIRA, A. P.; SILVA, M. F. Advances in agriculture robotics: A state-of-the-art review and challenges ahead. **Robotics**, v. 10, n. 2, 2021. ISSN 2218-6581. Disponível em: https://www.mdpi.com/2218-6581/10/2/52. Citado na p. 16.

PEDROZA, S. de M. **Utilização de uma rede neural artificial no controle e tomada de decisões de futebol de robôs da categoria VSSS**. 50 f. Monografia (Monografia (Graduação em Engenharia de Controle e Automação)) — Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2023. Disponível em: http://www.monografias.ufop.br/handle/35400000/5486. Citado nas pp. 16, 33 e 34.

PENA, C. H. C.; MACHADO, M. G.; BARROS, M. S.; SILVA, J. D.; MACIEL, L. D.; REN, T. I.; BARROS, E. N.; BRAGA, P. H.; BASSANI, H. F. An analysis of reinforcement learning applied to coach task in ieee very small size soccer. *In*: **2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)**. [*S.l.: s.n.*], 2020. p. 1–6. Citado na p. 34.

SAKR, M.; LI, Z. J.; LOOS, H. F. M. Van der; KULIć, D.; CROFT, E. A. Quantifying demonstration quality for robot learning and generalization. **IEEE Robotics and Automation Letters**, v. 7, n. 4, p. 9659–9666, 2022. Citado na p. 17.

SICILIANO, B.; SCIAVICCO, L.; VILLANI, L.; ORIOLO, G. **Robotics: Modelling, Planning and Control**. 1. ed. [*S.l.*]: Springer London, 2009. XXIV, 632 p. (Advanced Textbooks in Control and Signal Processing). Copyright Information: Springer-Verlag London 2009. Available as eBook in Engineering, Engineering (R0). Softcover ISBN: 978-1-84996-634-4 (Published: 21 October 2010). ISSN 1439-2232. ISBN 978-1-84628-641-4. Citado na p. 26.

TRIANA, M. A. P. **Controle baseado em comportamento aplicado a robótica móvel usando hardware reconfigurável**. 88 p. Dissertação (Dissertação) — Universidade de Brasília, Brasília, 2022. Mestrado em Ciências Mecânicas. Citado nas pp. 17 e 29.

VSSS League Brasil. **Liga Very Small Size**. 2019. Disponível em: https://ieeevss.github.io/vss/index_ptbr.html. Acesso em: 10 maio 2024. Citado nas pp. 16 e 20.

VSSS League Brasil. **Regras IEEE Very Small Size Soccer (VSSS) - Série A**. 2023. Disponível em: https://lars-larc2024.ucsp.edu.pe/rules/regrasVSS23.pdf. Acesso em: 09 junho 2024. Citado nas pp. 20, 21 e 40.

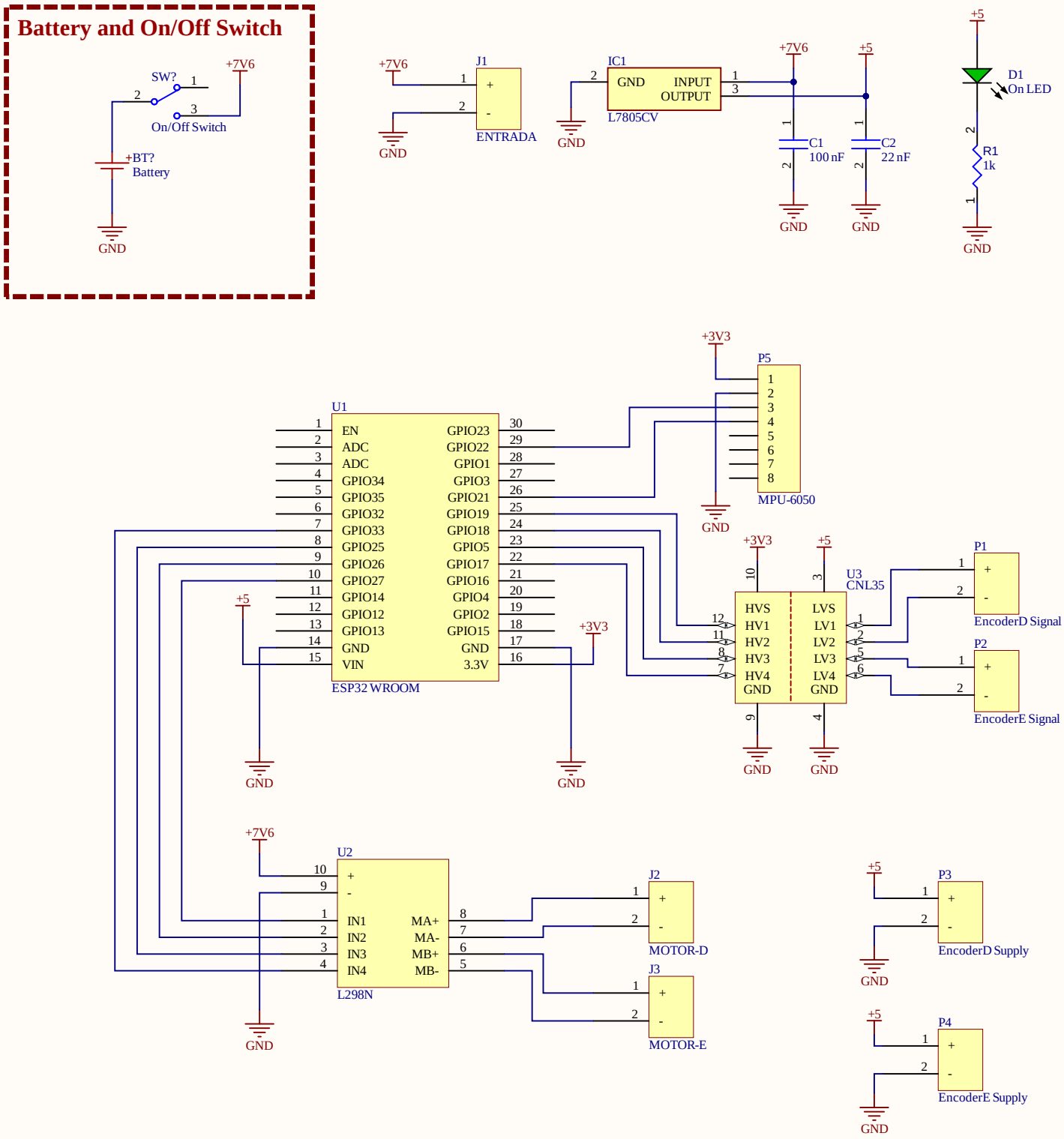
VáZQUEZ-HURTADO, C.; RODRIGUEZ-PADILLA, C. Adapting a very small size soccer (VSSS) competition for learning robotics in virtual teaching. *In*: **2022 IEEE Global Engineering Education Conference (EDUCON)**. [*S.l.: s.n.*], 2022. p. 553–558. Citado na p. 16.

WEISE, T. **Global Optimization Algorithms - Theory and Application**. Second. Self-Published, 2009. Online as e-book. Disponível em: http://www.it-weise.de/. Citado na p. 32.

XU, S.; OU, Y.; DUAN, J.; WU, X.; FENG, W.; LIU, M. Robot trajectory tracking control using learning from demonstration method. **Neurocomputing**, v. 338, p. 249–261, 2019. ISSN 0925-2312. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925231219300785. Citado na p. 17.

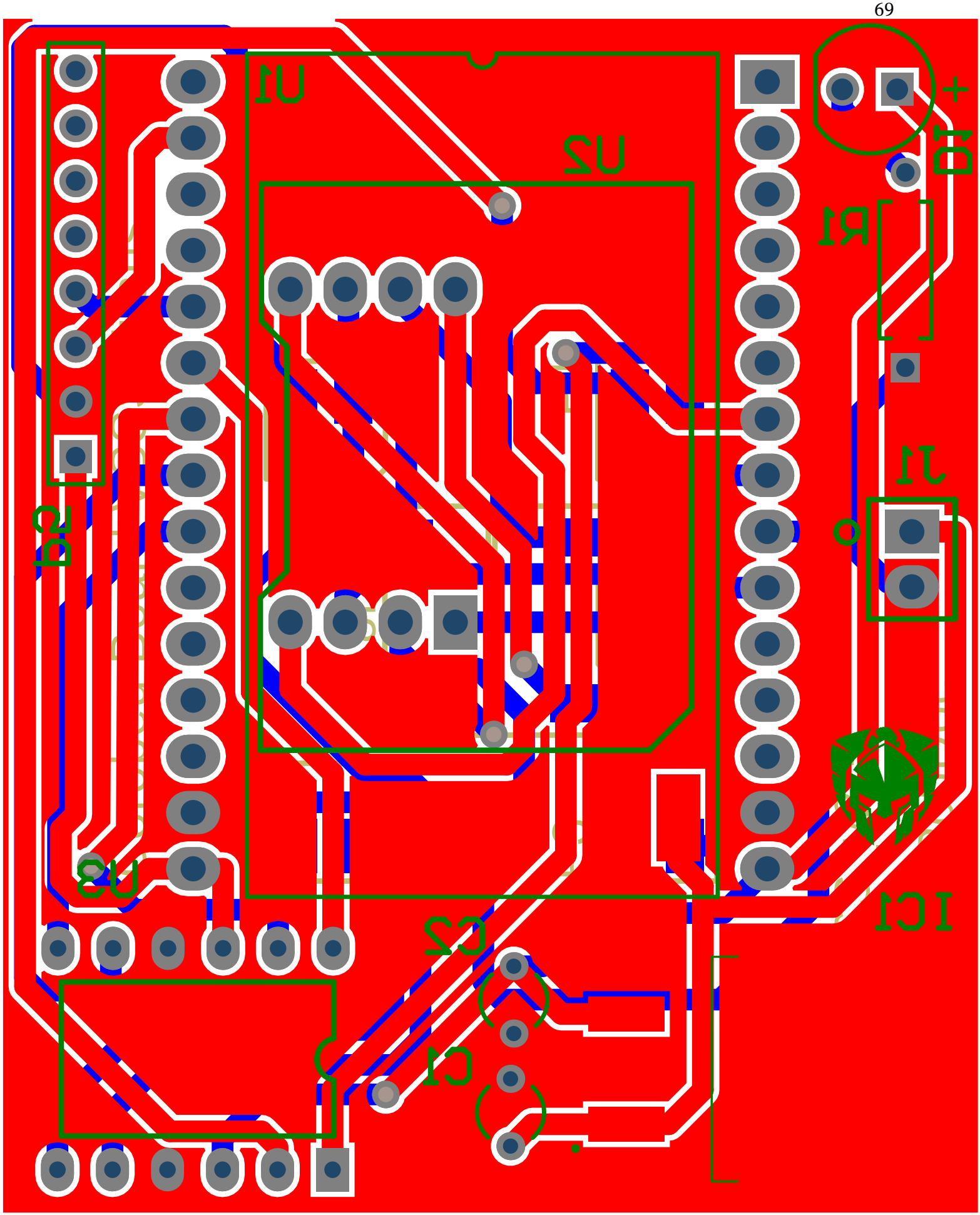
Apêndices

Apêndice A – Esquemático Eletrônico do Robô Corobeu



Título: Circuito principal VSSS

Equipe de Robótica Titans	Autor: Felipe Sidrone
Projeto: VSSS	Número: TT-CE-VSSS
Aprovado por: Matheus Matchuano	Revisão: V2.3
Data: 31/07/2024 — Última Modificação: 08/08/2024	Página 1 De 1

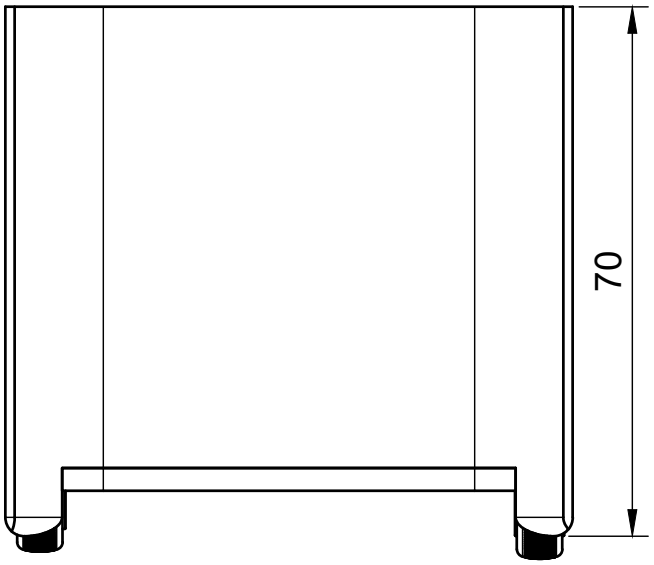

U1
U2
R1
J1
U3
C2
C1

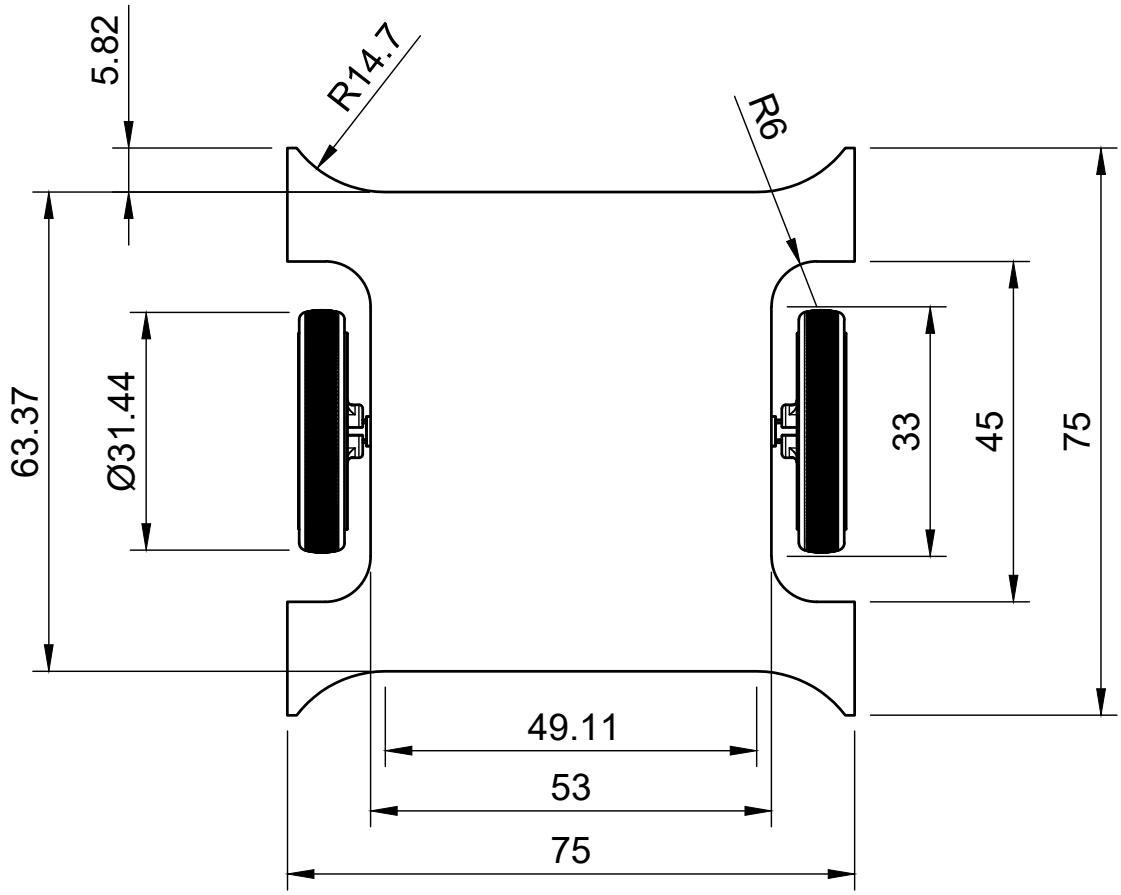
Apêndice B – Desenho Técnico Mecânico do Robô Corobeu

Projeto desenvolvido pela equipe



Dept.	Technical reference	Created by	Approved by
Mecânica	TT-ME-VSSS	Matheus de S. Luiz 24/11/2023	Raul Braga 24/11/2023
		Document type: Referência	Document status: Approved
		Title: VSSS Vista Isométrica Robô Completo	DWG No.: 1
			Rev.: v1 / Date of issue: 24/11/2023 / Sheet: 1/2





Dept.	Technical reference	Created by	Approved by
Mecanica	TT-ME-VSSS	Matheus de S. Luiz 24/11/2023	Raul Braga 24/11/2023
		Document type: Referência	Document status: Approved
		Title: VSSS Vistas Ortogonais Medidas Gerais	DWG No.: 1
			Rev.: v1 / Date of issue: 24/11/2023 / Sheet: 2/2