Wálisson Gôbbo de Águas

APLICAÇÃO DE FPGA PARA CONTROLE DE SISTEMAS EMBARCADOS MULTIAGENTES: ROBÔS JOGADORES DE FUTEBOL DA CATEGORIA IEEE VERY SMALL SIZE SOCCER

Goiânia

Agosto de 2018

sistema de bibliotecas ufg

PRPG
PRÓ-REITORIA DE PÓS-GRADUAÇÃO

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: **[X] Dissertação** **[ ] Tese**

2. Identificação da Tese ou Dissertação:

Nome completo do autor: Walisson Gobbo de Aguas

Título do trabalho: Aplicação de FPGA para controle de sistemas embarcados multi-agentes: Robôs jogadores de futebol da categoria IEEE Very Small Size Soccer.

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Walisson Gobbo de Aguas.
Assinatura do(a) autor(a)[2]

Ciente e de acordo:

Karina Rocha G. da Silva
Assinatura do(a) orientador(a)[2] Data: 18/10/2018

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

[2] A assinatura deve ser escaneada.

Wálisson Gôbbo de Águas

APLICAÇÃO DE FPGA PARA CONTROLE DE SISTEMAS EMBARCADOS MULTIAGENTES: ROBÔS JOGADORES DE FUTEBOL DA CATEGORIA IEEE VERY SMALL SIZE SOCCER

Dissertação de mestrado apresentada à Escola de Engenharia Elétrica, Mecânica e de Computação da Universidade Federal de Goiás como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica e de Computação. Área de concentração: Sistemas Eletro-Eletrônicos Embarcados.

Universidade Federal de Goiás - UFG

Escola de Engenharia Elétrica, Mecânica e de Computação

Programa de Pós-Graduação

Orientador: Profª. Drª. Karina Rocha Gomes da Silva

Goiânia

Agosto de 2018

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Águas, Wálisson
Aplicação de FPGA para controle de sistemas embarcados multiagentes: Robôs jogadores de futebol da categoria IEEE Very Small Size Soccer [manuscrito] / Wálisson Águas. - 2018.
79 f.: il.

Orientador: Prof. Karina Silva.
Dissertação (Mestrado) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Goiânia, 2018.
Bibliografia.
Inclui abreviaturas, símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Multiagentes. 2. Robôs. 3. Protótipo. 4. Futebol. I. Silva, Karina, orient. II. Título.

CDU 621.3

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Ata de Dissertação de Mestrado

Ata da sessão de julgamento da Dissertação de Mestrado em Engenharia Elétrica e de Computação, área de concentração Engenharia de Computação, do candidato **Wálisson Gôbbo de Águas**, realizada em 14 de setembro de 2018.

Aos quatorze dias do mês de setembro de dois mil e dezoito, às 09:00 horas, na sala Caryocar brasiliensis, bloco "A" da Escola de Engenharia Elétrica e de Computação (EMC), Universidade Federal de Goiás (UFG), reuniram-se os seguintes membros da Comissão Examinadora designada pela Coordenadoria do Programa de Pós-graduação em Engenharia Elétrica e de Computação: Os Doutores Karina Rocha Gomes da Silva – Orientadora (EMC/UFG), Djones Vinícius Lettnin – (UFSC) e Marco Antônio Assfalk (EMC/UFG), para julgar a Dissertação de Mestrado de **Wálisson Gôbbo de Águas**, intitulada **"Aplicação de FPGA para controle de Sistemas Embarcados Multiagentes: Robôs jogadores de futebol da categoria IEEE Very Small Size Soccer"**, apresentada pelo Candidato como parte dos requisitos necessários à obtenção do grau de Mestre, em conformidade com a regulamentação em vigor. A Professora Doutora Karina Rocha Gomes da Silva, Presidente da Comissão, abriu a sessão e apresentou o candidato que discorreu sobre seu trabalho, após o que, foi argüido pelos membros da Comissão na seguinte ordem: Djones Vinícius Lettnin, Marco Antônio Assfalk. A parte pública da sessão foi então encerrada e a Comissão Examinadora reuniu-se em sessão reservada para deliberar. A Comissão julgou então que o candidato, tendo demonstrado conhecimento suficiente, capacidade de sistematização e argumentação sobre o tema de sua Dissertação, foi considerado **aprovado** e deve satisfazer as exigências listadas na Folha de Modificação de Dissertação de Mestrado, em anexo a esta Ata, no prazo máximo de 30 dias, ficando a professora orientadora responsável por atestar o cumprimento dessas exigências. Os membros da Comissão Examinadora descreveram as justificativas para tal avaliação em suas respectivas Folhas de Avaliação, anexas a esta Ata. Nada mais havendo a tratar, a presidente da Comissão declarou encerrada a sessão. Nos termos do Regulamento Geral dos Cursos de Pós-graduação desta Universidade, a presente Ata foi lavrada, lida e, julgada conforme, segue assinada pelos membros da Comissão supracitados e pelo candidato. Goiânia, 14 de setembro de 2018.

Comissão Examinadora Designada:

Karina Rocha G. da Silva
Karina Rocha Gomes da Silva – Orientadora (EMC/UFG) (Avaliação: APROVADO)

Djones Vinícius Lettnin – (UFSC) (Avaliação: Aprovado)

Marco Antonio Assfalk de Oliveira
Marco Antônio Assfalk – (EMC/UFG) (Avaliação: APROVADO)

Geyverson Texeira de Paula – (EMC/UFG) (Avaliação: _____)

Cássio Leonardo Rodrigues– (EMC/UFG) (Avaliação: _____)

Candidato:

Walisson Gobbo de Aguas.
Wálisson Gôbbo de Águas

Este trabalho é dedicado a todos aqueles que de alguma forma me ajudaram a trilhar meu caminho até aqui.

AGRADECIMENTOS

Faço os meus agradecimentos primeiramente a Deus por me dar a oportunidade de poder cursar uma pós-graduação (Mestrado). Agradeço também a minha mãe que desde sempre me incentivou a seguir o caminho dos estudos, ao meu pai, já falecido, que enquanto vivo ajudou em minha caminhada até aqui. Por fim agradeço os meus professores que me ensinaram o ofício da engenharia e aos meus amigos que sempre que precisei estavam comigo.

Agradeço também a fonte financiadora desse trabalho que em forma de bolsa apoia não somente eu mas muitos alunos de Mestrado pelo país inteiro, a CAPES meus sinceros agradecimentos.

"A mente que se abre a uma nova ideia,
jamais voltará ao seu tamanho original."
(Albert Einstein)

RESUMO

O desenvolvimento de novas tecnologias está ligado principalmente à construção de dispositivos que visam de alguma forma auxiliar o ser humano em suas tarefas, tornando-as mais rápidas, fáceis e precisas. Desde a descoberta do transistor, a mais de 70 anos por John Bardeen e Walter Brattain, ocorreu uma revolução enorme, fazendo com que novas áreas de pesquisas fossem criadas, principalmente em robótica cooperativa e multiagentes.

Com o aumento constante do número de robôs em um ambiente industrial, cientistas e tecnólogos frequentemente enfrentaram problemas de cooperação e coordenação entre diferentes robôs e seu auto-controle em um espaço de trabalho. Isso levou à evolução dos sistemas autônomos cooperativos multi-robóticos. Os desenvolvedores de sistemas autônomos multi-robóticos precisavam de um modelo para testar as teorias propostas em termos de eficácia e eficiência. Não é uma surpresa que eles começaram a se concentrar no futebol de robôs. Os robôs jogadores de futebol possuem grandes demandas em todas as áreas da tecnologia de robôs, mecânica, sensores e de inteligência artificial.

Dessa forma o objetivo desse trabalho é desenvolver um sistema multiagente de dimensões reduzidas (7,5x7,5x7,5cm) destinado a jogar futebol em um ambiente dinâmico pré determinado. Um protótipo será desenvolvido para fins de comparação com um robô já desenvolvido anteriormente tendo como foco melhorias no sistema controlador do multiagente e na estrutura física do hardware.

Palavras-chaves: Multiagentes, Robôs, IEEE-VSSS, Futebol.

ABSTRACT

The development of new technologies is mainly related to the construction of devices that aim in some way to assist the human being in his tasks, making them faster, easier and more precise. Since the discovery of the transistor, more than 70 years ago by John Bardeen and Walter Brattain, there has been a huge revolution, and new research areas have been created, especially in cooperative robotics and multi-agent.

With the steady increase in the number of robots in an industrial environment, scientists and technologists have often faced problems of cooperation and coordination between different robots and their self-control in a workspace. This led to the evolution of autonomous multi-robotic cooperative systems. Developers of multi-robotic autonomous systems needed a model to test the proposed theories in terms of effectiveness and efficiency. Not surprisingly, they began to focus on robot soccer. Robots soccer players have great demands in all areas of robot technology, mechanics, sensors and artificial intelligence.

Thus, the objective of this work is to develop a small multi-agent system (7.5x7.5x7.5cm) intended to play soccer in a predetermined dynamic environment. A prototype will be developed for comparison purposes with a previously developed robot focusing on improvements to the multiagent controller system and the physical structure of the hardware.

Key-words: Multi-agents, Robots, IEEE-VSSS, Soccer.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
AGc	Algoritmo Genético Compacto
CAD	*Computer Aided Design*
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CBR	Competição Brasileira de Robótica
CC	Corrente Contínua
DC	*Direct Current*
EMC	Escola de Engenharia Elétrica, Mecânica e de Computação
FIRA	*Federation of International Robot Soccer Association*
FPGA	*Field Programabe Gate Array*
HDL	*hardware description language*
I/O	*Input / Output*
IIDI	*Ivrea Interaction Design Institute*
IoT	*Internet of Things*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
LARC	*Latin American Robotcs Competition*
LIPO	Polímero de Lítio
LUT	*Look-up Table*
MDF	*Medium Density Fiberboard*
MSL	*Middle League*
OVI	*Open Verilog International*
PCI	Placa de circuito impresso
PID	Proporcional Integral Derivativo
PMEC	Núcleo de Robótica Pequi Mecânico

PROM	*Programmable Read-Only Memory*
PWM	*Pulse Width Modulation*
RPS	Rotações Por Segundo
SSL	*Small League*
VAG	Veículo Autônomo Guiado
UFG	Universidade Federal de Goiás
VHDL	*VHSIC Hardware Description Language*
VSSS	Very Small Size Soccer

LISTA DE SÍMBOLOS

$\int$	Integral
<=	Atribuição não-bloqueante à variáveis dos algoritmos
=	Atribuição bloqueante à variáveis dos algoritmos
Kp	Ganho proporcional
Ki	Ganho integral
Kd	Ganho derivativo
e(t)	Erro de estado estacionário do controlador PID
dt	Derivada do tempo
u(t)	Função de transferência do PID
G(s)	Função de transferência final do motor
m_j	Quantidade de bits necessários
a_j	Limite inferior do intervalo
b_j	Limite superior do intervalo
t_r	Tempo de subida
t_s	Tempo de estabilização
M_P	Sobressinal (overshoot) máximo
E_{ss}	Erro de estado estacionário
x_j	Ganho Kp, Ki ou Kd calculado
m_j	Quantidade de bits necessários
a_j	Limite inferior do intervalo
b_j	Limite superior do intervalo

SUMÁRIO

1 INTRODUÇÃO

Considerações Preliminares

O desenvolvimento de novas tecnologias está ligado principalmente à construção de dispositivos que visam de alguma forma auxiliar o ser humano em suas tarefas, tornando-as mais rápidas, fáceis e precisas. Desde a descoberta do transistor a mais de 70 anos por John Bardeen e Walter Brattain ocorreu uma revolução enorme, fazendo com que novas áreas de pesquisas fossem criadas (BRINKMAN; HAGGAN; TROUTMAN, 1997).

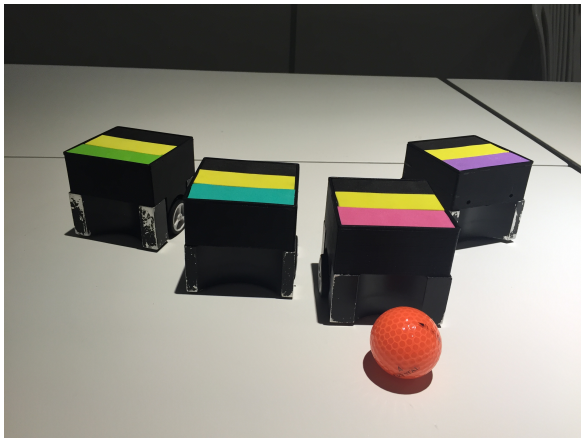
Atualmente no Brasil e no Mundo acontecem vários eventos destinados justamente à pesquisa e incentivo de desenvolvimento de novas tecnologias. Em diferentes tipos de formatos, os eventos científicos buscam mostrar o que há de melhor em pesquisas feitas por estudantes e professores do mundo. Um dos principais eventos reconhecidos mundialmente é organizado pela *Federation of Internacional Robot Soccer Association* (FIRA), a *Micro-Robot World Cup* (MiroSot).

Segundo FIRA (2017b) a iniciativa MiroSot é um bom ponto de partida para pesquisas de robôs multiagentes que lidam principalmente com protocolo de cooperação por controle distribuído e comunicação efetiva entre agentes, o que melhora a eficiência, cooperação, adaptação e robustez dos agentes robóticos.

Com o aumento constante do número de robôs em um ambiente industrial, cientistas e tecnólogos frequentemente enfrentaram problemas de cooperação, coordenação, e auto-controle dos robôs em um espaço de trabalho específico. Isso levou à evolução dos sistemas autônomos cooperativos multi-robôs. Os desenvolvedores de sistemas autônomos multi-robóticos precisavam de um modelo para testar as teorias propostas em termos de eficácia e eficiência. Uma forma de testar essas teorias é através do futebol de robôs. Os modelos de robôs jogadores de futebol possuem grandes demandas em todas as áreas da tecnologia de robôs, mecânica, sensores e de inteligência artificial (FIRA, 2017b).

A Figura 1 mostra um exemplo de time de futebol. Esse time é formado por robôs autônomos que possuem dimensões de 7,5cm x 7,5cm x 7,5cm. O projeto foi desenvolvido na Universidade Federal de Goiás (UFG) pelo Núcleo de Robótica Pequi Mecânico (PMEC). O projeto engloba algoritmo computacional e já possui resultados interessantes, mas precisa de algumas melhorias no sistema como um todo pois os robôs ainda não possuem um sistema PID eficiente de controle dos robôs (MARTINS et al., 2016). Os robôs possuem um sistema integrado micro-controlado responsável por todo o gerenciamento do robô que inclui recebimento de informação via rádio, controlador via malha fechada das velocidades dos motores através de um controlador PID e contagem de pulsos dos encoder dos motores.

Figura 1 – Robôs que jogam futebol que utilizam controlador PID.



Fonte: Próprio Autor.

Um ponto de partida para a melhoria do sistema multiagente mostrado na Figura 1 é a utilização de controlador com processamento paralelo como o *Field Programmable Gate Array* (FPGA). Esse dispositivo pode ser programado e configurado de acordo com as necessidades do usuário de modo a obter módulos que trabalham em paralelo. Cada módulo é responsável por uma ação no robô e assim são independentes entre si, compartilhando algumas informações necessárias para o funcionamento correto do robô.

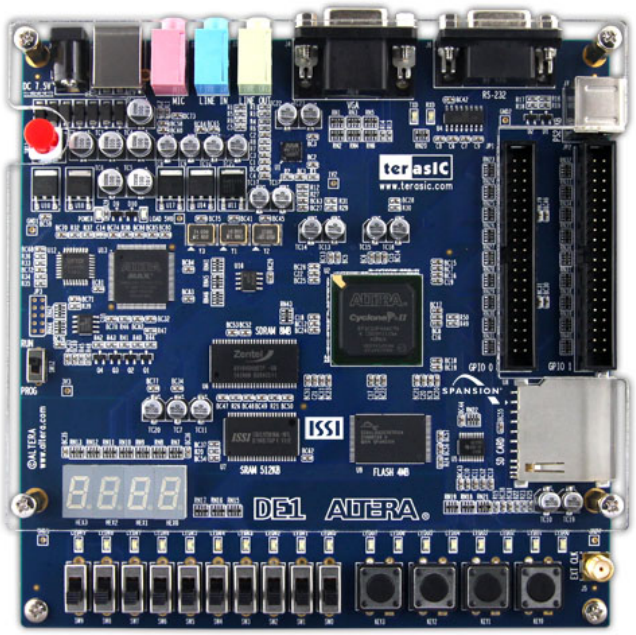
Os FPGAs são encontrados facilmente como kits diádicos onde são integrados com vários outros dispositivos que facilite seu aprendizado. Um bom ponto de partida é a utilização de um kit didático tal como o mostrado na Figura 2. Trata-se de um kit *Altera DE1 Cyclone® II 2C20 FPGA device* (ALTERA, 2012). Ele pode ser usado para prototipação mas não para ser integrado nos robôs, visto que seu tamanho é incompatível com as dimensões requeridas. Nesse caso faze-se necessário a utilização de um módulo reduzido ou até mesmo o projeto de um módulo específico para essa aplicação.

Objetivos

Existe a necessidade de melhorar alguns pontos no sistema dos robôs atuais desenvolvidos pelo PMEC. Tais problemas vão desde uma comunicação mais confiável até mesmo um controle mais preciso nas velocidades aplicadas aos motores. Um ponto importante é a velocidade de processamento do sistema embarcado (robô), de modo que quanto mais rápido o processamento, mais rápido será sua resposta. Outro ponto importante é a escolha adequada das peças do robô e dos protocolos de comunicação. Um protocolo muito extenso pode acarretar atraso de mensagens ou até mesmo o não recebimento das informações necessária naquele instante de tempo.

Esse trabalho de dissertação é destinado ao desenvolvimento de um protótipo robótico

Figura 2 – Kit *Altera DE1 Cyclone® II 2C20 FPGA device*.



Fonte: Próprio Autor.

funcional, projetado e desenvolvido computacionalmente, com o objetivo de implementar e obter resultados melhores do que os apresentados nos protótipos atuais. Esse objetivo pode ser decomposto nos seguintes objetivos secundários:

- Estudar e escolher um sistema novo de modo a melhorar o processamento do robô e todo o gerenciamento dos periféricos necessários para o bom funcionamento do robô;
- Desenvolver um sistema de calibração do sistema de controle dos motores em ambiente computacional utilizando Algoritmos Genéticos compactos (AGc);
- Desenvolvimento de um protocolo de comunicação que apresente comunicação eficiente entre indivíduos (robôs) e coordenador (computador);
- Projetar uma estrutura que comportará todos os componentes do robô em um cubo de 7,5cm x 7,5cm x 7,5cm;
- Comparar os resultados do sistema de controle dos robôs a serem projetados com com os já desenvolvidos pelo PMEC, tal como as velocidades aplicadas aos motores.

Contribuições

Além dos objetivos e objetivos secundários citados, no final do trabalho algumas contribuições são esperadas, são elas:

- Um robô que utilize um controlador FPGA em plataforma embarcada;
- Artigos nacionais na área de robótica móvel utilizando multiagentes;
- Projeto de hardware para estudos multiagentes;
- Algoritmo Genético compacto destinado a calibração de motores utilizando PID;

Organização do Trabalho

Esse trabalho é destinado ao estudo e implementação de um novo sistema embarcado multiagentes jogadores de futebol da categoria IEEE-VSSS. Nesse sentido, essa dissertação está organizada da seguinte forma:

No Capítulo 2 será apresentado o estudo bibliográfico realizado, bem como os trabalhos já existentes publicados e relacionados com o tema. Também serão apresentadas aplicações importantes desse tipo de agentes no mercado e o que já possui de implementações reais. Por fim uma abordagem do desenvolvimento de algumas equipes serão utilizadas como referência para o desenvolvimento do novo protótipo buscando aproveitar as principais e melhores característica dos dispositivos utilizados.

No Capítulo 3 será abordada a metodologia utilizada na execução do projeto proposto. Nesse capítulo será abordado a estrutura organizacional do robô bem suas funções primárias e secundárias.

No Capítulo 4 serão apresentadas as características necessárias e quais componentes serão utilizados para a confecção de um exemplar do projeto aqui proposto. Também é apresentado uma breve descrição dos requisitos esperados para a melhora do desempenho do robô comparado com a versão antiga do projeto.

No Capítulo 5 serão abordados como sistema de controle, estratégia e visão computacional trabalham em conjunto para enviar as informações necessárias de velocidade e aceleração para que o robô tenha um bom funcionamento no ambiente de multiagentes.

No Capítulo 6 serão mostrados os resultados obtidos. Todos os procedimentos de montagem e desmontagem do robô, os algoritmos implementados bem como os módulos criados serão abordados de maneira separada para melhor apresentação do conteúdo utilizado.

Por fim, no Capítulo 7, será apresentado a conclusão dos resultados obtidos e os trabalhos futuros de continuação do projeto.

2 REVISÃO BIBLIOGRÁFICA

Os projetistas e pesquisadores quando estudam e desenvolvem sistemas embarcados para robótica móvel estão sempre atentos em escolher e planejar corretamente os dispositivos necessários para que a performance do robô seja a melhor possível, dentro de um determinado orçamento e projeto. Dentre os mais diferentes tipos de robôs e sistemas embarcados existentes destacam-se os robôs multiagentes que podem interagir entre si ou até mesmo serem controlados por um único coordenador, podendo ainda possuir tamanho reduzido, facilitando a cooperação entre agentes em ambientes simulados.

Para testar essas plataformas multiagentes em um ambiente multi cooperativo os desenvolvedores procuram desafios que possam ser realizados em grupos de robôs. Tais desafios são importantes porque cada robô agente desempenha um papel individual que integra uma tarefa maior. Diversas pesquisas são realizadas justamente para estudar práticas de inteligência computacional e que já possuam resultados bastante interessantes atualmente. Um dos principais temas de pesquisas são os voltados para controle de agentes robóticos em um ambiente pré conhecido através de *feedback* de um sistema global de visão computacional. Alguns trabalhos podem ser citados e que exploram exatamente o ponto em questão: o controle de multiagentes por sistema global de visão computacional.

2.1 Multiagentes

2.1.1 Jogadores de Golf

Em seu trabalho sobre sistemas autônomos jogadores de golfe Jabson et al. (2008) descreve como robôs podem interagir entre si para jogarem golf, escolhendo melhores estratégias de jogo e principalmente fazendo identificação dos elementos característicos do jogo como jogadores, bolas, buracos e delimitações da quadra/campo. Nessa mesma linha de raciocínio, os jogadores devem possuir um sistema de navegação e localização bem como um sistema de locomoção. O sistema funciona basicamente em um ambiente simulado com dimensões pré determinadas e totalmente retilíneo, uma bola de golfe real, micro jogadores multiagentes e marcações representando os buracos alvo.

Para o adequado funcionamento do sistema multiagentes jogador de golfe Jabson et al. (2008) cita que um sistema inteligente é dividido em duas partes: Um sistema controlador em um computador e um sistema multiagente composto pelos robôs. Basicamente um sistema de visão computacional captura imagens do campo a uma determinada taxa de quadros por segundo que possuem elementos a serem identificados como os jogadores (identificados por tags coloridas), buracos e bola bem definidos nas imagens. As imagens então são processadas

por um software e de acordo com as estratégias definidas processa e envia comandos através de um rádio transmissor para os multiagentes se posicionarem e jogarem golfe.

2.1.2 FIRA WorldCup

Outros trabalhos bastante interessantes são desenvolvidos por equipes que disputam campeonatos mundiais de futebol de robôs utilizando multiagentes cooperativos. Assim como os robôs multiagentes jogadores de golfe, os robôs jogadores de futebol trabalham em conjunto para conseguirem jogar futebol. De acordo com a FIRA (2017b) mais de 5 tipos de categorias de futebol de robôs são disputadas em competições organizadas por essa federação anualmente. As categorias podem ser listadas:

- ***HuroCup***: A categoria HuroCup enfatiza o desenvolvimento de robôs flexíveis, robustos e versáteis que podem executar muitas tarefas diferentes em diferentes domínios. A HuroCup incentiva a pesquisa sobre as muitas áreas da robótica humanoide, especialmente o andar e o equilíbrio, o planejamento de movimentos complexos e a interação entre robôs.
- ***RoboSot***: A categoria RoboSot consiste em uma série de desafios dentro de uma competição tradicional de futebol de robôs. Os problemas e os desafios que desenvolvedores enfrentam nessas plataformas são variados e muitas vezes diferentes dos problemas enfrentados em ligas menores do mesmo estilo.
- ***SimuroSot***: A categoria SimuroSot tem como principal objetivo permitir que os pesquisadores desenvolvam algoritmos de controle e estratégias de equipe sem a necessidade de uma configuração de hardware complexa e dispendiosa. As equipes são encorajadas a usar a plataforma de simulação para a avaliação de seus algoritmos e, eventualmente a participar da liga MiroSot, que está exposta a condições do mundo real.
- ***AndroSot***: A categoria AndroSot visa promover as habilidades de ataque e defesa em robôs humanoides e também consiste em várias tarefas como prevenção de obstáculos, detecção de trajetória, guarda-redes, organização e controle de posicionamento.
- ***MiroSot***: A categoria MiroSot é uma boa arena para a pesquisa com multiagentes, lidando com micro robôs de dimensões específicas de jogadores de futebol em categorias de 5 *versus* 5 (*Middle League*) ou 11 *versus* 11 (*Large League*). O ambiente de jogo é bem dinâmico e os robôs podem ser semi ou completamente autônomos, mas sem intervenção humana. É dessa categoria que se originou a categoria disputada no Brasil atualmente, a *IEEE Very Small Size Soccer* (IEEE-VSSS) com disputas de 3 *versus* 3.

Para desenvolver um sistema multiagente o desenvolvedor utiliza implementações inteligentes compostas por agentes cooperativos com controle de movimento independentes entre si. Os multiagentes podem ser caracterizados com sistemas mais tolerantes a falhas do que

sistemas compostos por robôs únicos, fazendo com que os problemas para o qual esses multiagentes são destinados possam ser resolvidos mais rapidamente, com maior agilidade e eficiência. Por esse motivo são realizadas disputas anualmente em competições nacionais, como a (IEEE-VSSS), e em competições internacionais, como a MiroSot, justamente para incentivar os pesquisadores a melhorarem tais sistemas embarcados (HAN et al., 2002).

O desenvolvimento de multiagentes para a categoria MiroSot é um teste interessante para aplicação adaptativa com respostas em tempo real. Os robôs que jogam futebol, por exemplo, são diferentes de outros multiagentes visto que os robôs em uma equipe cooperam entre si para vencer a equipe adversária, tendo cada um seu papel dentro da estratégia da equipe. O algoritmo de controle multiagente inclui cinemática e dinâmica de baixo nível e por outro lado estratégias de alto nível para evitar obstáculos e competir com os robôs adversários. Em um ambiente tão dinâmico, um robô precisa de processamento rápido de algoritmos e uma estrutura de robô compacta e robusta (HAN et al., 2002).

2.1.2.1 MiroSot

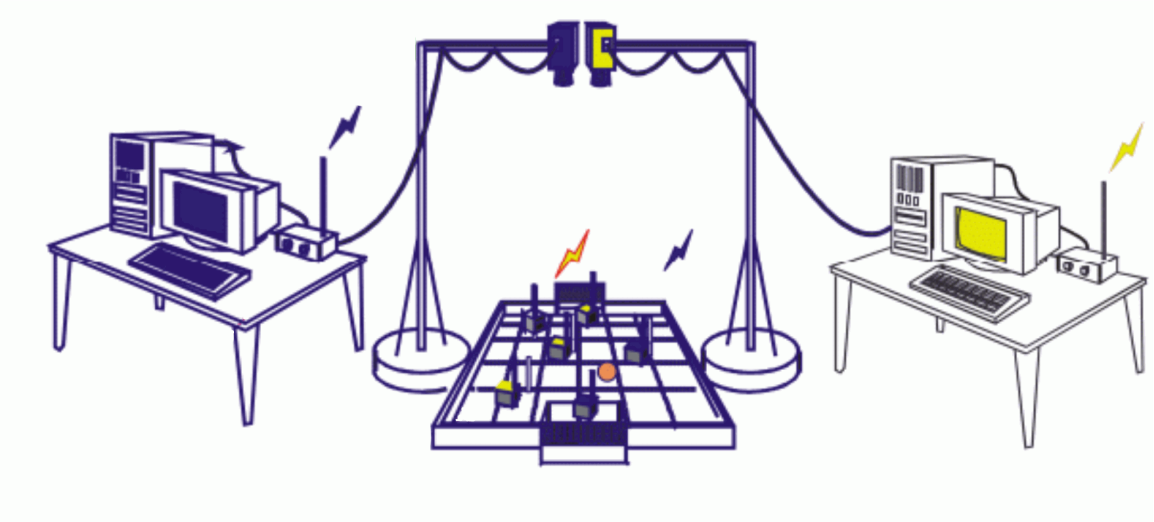
MiroSot é um torneiro de robótica voltado para a pesquisa interdisciplinar, onde cientistas, tecnólogos e estudantes de diversas áreas, como Robótica, Controle Inteligente, Comunicação, Tecnologia da Informação, Tecnologia de Sensores, Processamento de Imagens, Mecatrônica, Inteligência Artificial, etc., trabalham juntos para tornar os sistemas multi-robôs uma realidade. Os robôs usados na MiroSot são de tamanho pequeno (7,5cm x 7,5cm x 7,5cm), totalmente ou semi-autônomos e sem qualquer operador humano (FIRA, 2017b).

O ambiente de controle dos robôs é um espaço delimitado, similar a um campo de futebol, onde os robôs cooperam entre si para jogarem futebol. Como se trata de um jogo de futebol o ambiente é dinâmico forçando os agentes (robôs) a não somente procurar e levar a bola ao gol adversário (controlar) mas também fazer marcação do time adversário (rastrear). Dessa forma o futebol como ponto de partida para estudos de sistemas multiagentes é valido pois a cada momento o sistema (dinâmico) sofre mudanças proporcionando um sistema inteligente e complexo.

O futebol de robôs multiagentes consiste, então, em uma competição de robótica entre dois times onde o principal objetivo é, através de um computador, controlar remoto e autonomicamente os agentes de modo a se fazer gols no adversário. O esquema completo de um jogo é mostrado na Figura 3. Esse esquema mostra alguns dispositivos importantes que são utilizados durante os jogos tais como computadores, câmeras, rádios transmissores, pista, bola e robôs. Cada um dos dispositivos citados possuem suas especificações e importância como:

- **Computadores**: Os Computadores são considerados os técnicos dos times. Cada time possui um computador que é responsável por enviar as informações necessárias para que

Figura 3 – Montagem de um jogo de futebol de robôs.



Fonte: (FIRA, 2017b).

os robôs (agentes) consigam seguir a bola ou até mesmo executar algum tipo de jogada, definida pela estratégia implementada através de um algoritmo inteligente.

- **Câmeras**: As câmeras colocadas acima do campo com o auxílio de um suporte são utilizadas para fazer o reconhecimento por visão computacional. Trata-se de um retorno de informação do campo (ou simplesmente *feedback*) do posicionamento dos robôs. Cada robô devem possuir marcações coloridas, previstas no regulamento, para a identificação e distinção de cada jogador do próprio time ou até mesmo dos jogadores adversários.

- **Rádios transmissores**: Os robôs precisam receber informações de posicionamento da bola e deles mesmo no campo, e em alguns casos, até mesmo da estratégia dos jogadores adversários. Essa atualização é feita através de um rádio transmissor. Cada jogador possui o seu próprio rádio que se comunica com um rádio conectado ao computador.

- **Pista**: A pista é o campo propriamente dito. Ela é feita geralmente em madeira *Medium Density Fiberboard* (MDF) em formato de um campo de futebol, respeitando as especificações do regulamento.

- **Bola**: A bola possui tamanho, textura e peso específicos. Ela nada mais é que uma bola comum de jogar golf. Suas principais características são: a) Peso de 46g e b) Diâmetro de 42,7mm.

- **Robôs**: Dependendo da categoria, a quantidade de robôs pode mudar, mas basicamente os robôs devem ser autônomos e não devem ter dimensões maiores que 7,5cm x 7,5cm x 7,5cm. Cada robô deve possuir pelo menos uma tag colorida em sua superfície superior na cor amarela ou azul, que correspondem as cores dos times.

Segundo FIRA (2017a) ainda existe um sistema de normas referentes a faltas, pênaltis, posicionamento e etc.

2.1.3 RoboCup

Outra competição voltado para o desenvolvimento de multiagentes jogadores de futebol é a RoboCup. RoboCup é uma competição internacional com eventos de diferentes tamanhos e público alvo. Um dos principais objetivos da RoboCup é incentivar o desenvolvimento robótico e pesquisas em inteligência computacional fornecendo desafios reais (ROBOCUP, 2017). Dessa forma é possível obter desenvolvimentos importantes no que se trata de trabalho em equipe de desenvolvimento robótico. Por outro lado, o desenvolvimento de robôs multiagentes, destinados a jogar futebol, não gera um impacto social e econômico significativo. A realização como um todo traz para o campo de pesquisa contribuições valiosas que podem e já são utilizados no mercado e na industria.

A RoboCup, assim como *FIRA MiroSot WorldCup*, possui diversas áreas de pesquisas destinados justamente a área de futebol de robôs multiagentes. Pode-se enumerar-las da seguinte forma (ROBOCUP, 2017):

- ***Humanoid***: Na categoria ***Humanoid*** os robôs autônomos possuem forma e comportamentos semelhantes aos dos humanos, e tem por objetivo jogarem futebol. Cada robô possui sistema de visão próprio e não são controlados por nenhum agente externo, mas jogadores podem se comunicar entre si. Além do desafio de jogar futebol existem os desafios técnicos onde são analisados a caminhada dinâmica, corrida, chute da bola, equilíbrio, percepção visual da bola e de outros jogadores, campo, auto-localização e jogo em equipe.

- ***Standard Plataform***: Na categoria ***Standard Plataform*** todos os times possuem o mesmo modelo de robô, o robô NAO da SoftBank Robotics (SOFTBANK, 2017). Esses robôs jogam de forma totalmente autônoma e cada um toma decisões separadamente dos outros, mas eles ainda têm que jogar em equipe, característica de sistemas multiagentes.

- ***Middle League***: Na ***Middle League*** (MSL) times competem entre si (5 *versus* 5) com robôs totalmente autônomos. Nessa categoria as equipes são livres para projetar seus robôs e todos os sensores devem estar embarcados no robô, tendo limitação de tamanho e peso. Assim o foco dessa pesquisa é o desenvolvimento e design de protótipos, controle e cooperação entre sistemas multiagentes.

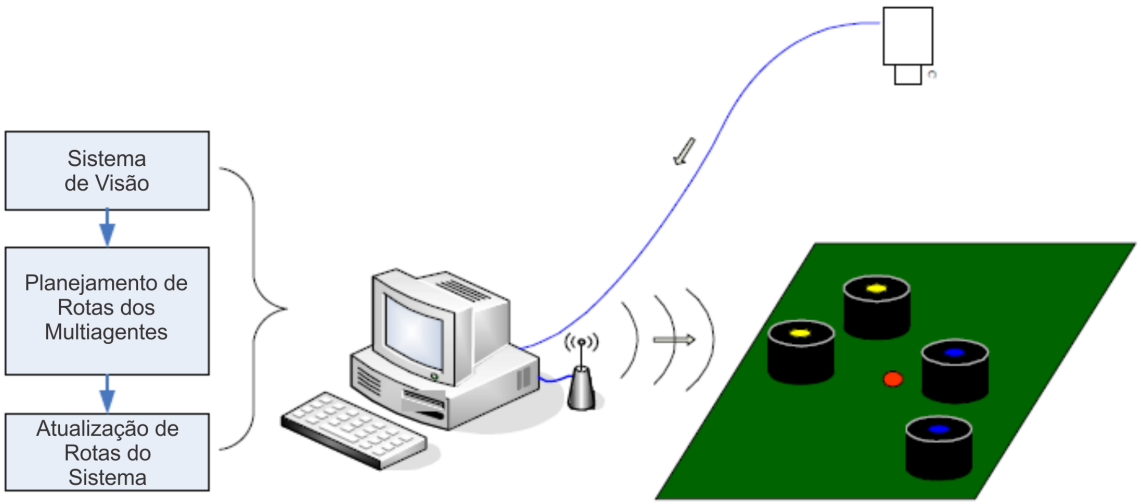
- ***Small League***: Na ***Small League*** (SSL), também conhecida como F180, é uma das ligas mais antigas da RoboCup. Essa liga se concentra no problema da cooperação e controle de multi-robôs/agentes inteligentes em um ambiente altamente dinâmico com um sistema híbrido centralizado/distribuído. Um jogo de futebol robô da SSL ocorre entre duas

equipes de seis robôs cada. Cada robô deve estar em conformidade com as dimensões conforme especificado nas regras F180: o robô deve caber dentro de um círculo de 180 mm de diâmetro e não deve ser superior a 15 cm. Os robôs jogam futebol com uma bola de golfe laranja em um campo de tapete verde de 9m de comprimento por 6m de largura.

- ***Simulation***: Na categoria ***Simulation*** se concentra na inteligência artificial e na estratégia dos times multiagentes. Os desenvolvedores programam suas estratégias para aplicação virtual de jogos de futebol, não se aplicando a robô físico. Essa liga possui duas sub categorias que são a 2D e a 3D, ambas em ambiente virtual.

Das categorias da RoboCup citadas anteriormente a SSL se destaca em abranger uma das maiores linhas de pesquisa, seja no desenvolvimento de protótipos robóticos, estratégias, inteligência artificial, visão computacional, modelagem dinâmica ou mecânica. A SSL tem como foco resolver problemas utilizando inteligencia artificial aplicada a ambientes com multiagentes onde agentes trabalham em ambientes altamente dinâmicos. A Figura 4 mostra uma visão geral de um sistema típico de um jogo da SSL (LI et al., 2010).

Figura 4 – Visão geral de um sistema de jogo da SSL.



Fonte: Modificado de (LI et al., 2010).

O sistema mostrado na Figura 4 é composto por camadas contendo subsistemas que processam em tempo real múltiplas tarefas. Um sistema digital de visão global (*Vision System*) é responsável por captar e enviar as imagens do campo para um software em um computador, que será responsável pro processar as imagens recebidas e identificar os robôs, delimitações do campo e a bola. Após esse processamento as informações de estados e posicionamento dos robôs são enviadas para um sistema de planejamento de rotas dos multiagentes (*Multi-Agent Planning System*). Após o planejamento, um sistema é responsável por calcular e aplicar as ações em cada robô do sistema multiagente através de envio de informação sem fio (*Path Pro-*

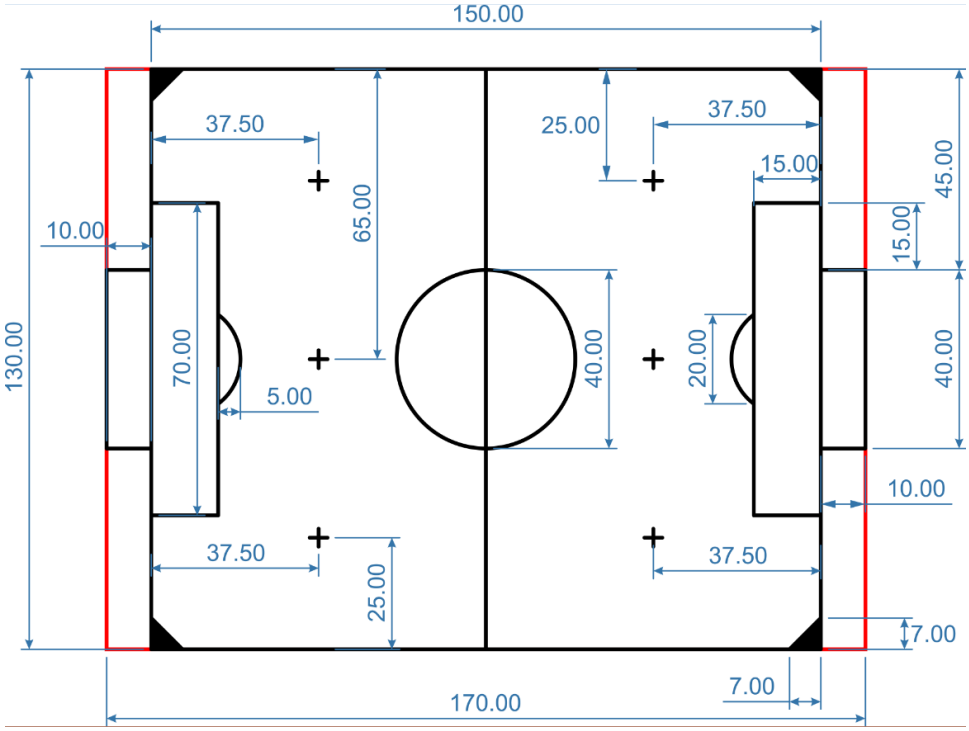
gram System). Dessa forma é possível aplicar conceitos de inteligência artificial e por à prova as estratégias dos times por meio de disputas de jogos (LI et al., 2010).

2.1.4 *IEEE Very Small Size Soccer*

No Brasil também são disputadas algumas das categorias da RoboCup em um evento que abrange não somente o Brasil mas também toda a América Latina. A Competição Brasileira de Robótica (CBR) acontece todos os anos juntamente com a Competição Latino-Americana de Robótica (*Latin American Robotics Competition* - LARC). Nesse evento além de algumas das categorias da RoboCup citadas anteriormente são disputadas outras categorias destinadas ao controle e implementação de sistemas multiagentes para resolução de alguma tarefa. Trata-se da *IEEE Very Small Size Soccer*.

A categoria *IEEE Very Small Size Soccer* (IEEE-VSSS) é uma categoria derivada da *FIRA MiroSot WorldCup* citada anteriormente. Os objetivos são desenvolver um sistema totalmente autônomo que consiga disputar uma partida de futebol. As dimensões do arena ou campo que são disputadas nessa categoria são mostradas na Figura 5

Figura 5 – Dimensões do campo onde são disputadas as partidas de futebol da categoria IEEE-VSSS.



Fonte: Próprio Autor

A categoria IEEE-VSSS possui as mesmas regras que a MiroSot, exceto em alguns pontos como quantidade de robôs e tamanho do campo. A Tabela 1 mostra as principais diferenças entre as duas categorias.

Tabela 1 – Principais diferenças das categorias FIRA MiroSot e IEEE-VSSS.

Características:	MiroSot	IEEE VSSS
Tamanho do campo (cm)	400x280	150x130
Quantidade de robôs por time (un)	5 ou 11	3

Fonte: Próprio Autor

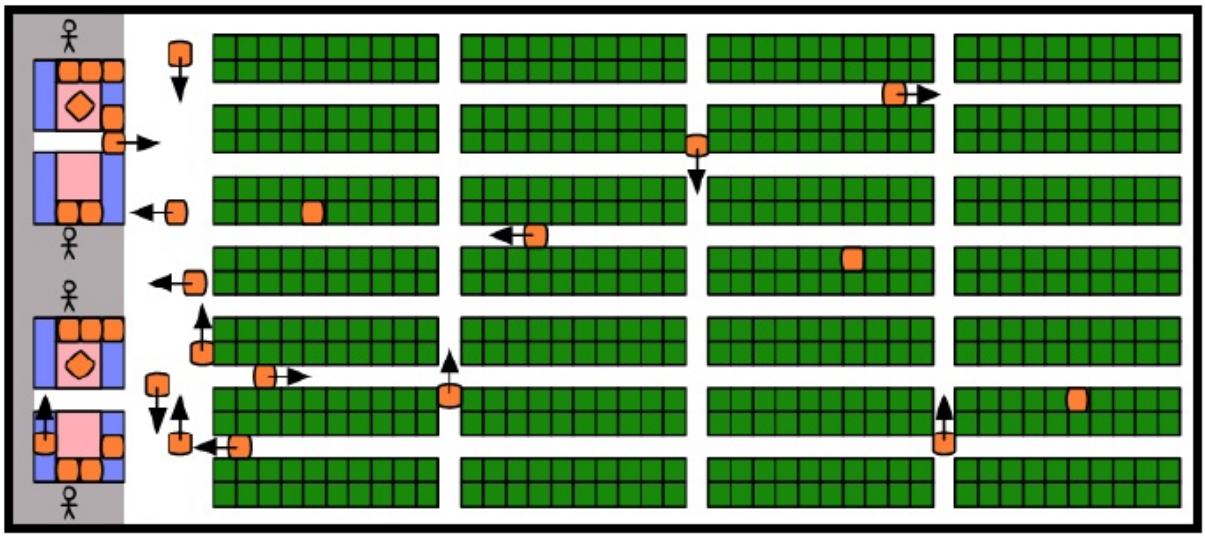
2.1.5 Multiagentes em Ambiente Industrial

Em um ambiente industrial os sistemas multiagentes podem ter aplicações interessantes no auxilio e agilidade nas atividades industriais. O principal foco é realizar uma tarefa complexa por meio de centenas de agentes menores a fim de se obter um bem comum, a realização da tarefa.

Desde os anos de 1950 os Veículos Autônomos Guiados (VAG) são usados para mover materiais em estoques de industrias. Desde então as tecnologias vem se desenvolvendo consideravelmente ao ponto de baratear os VAGs e ao mesmo empo de se tornarem mais eficientes e autônomos. Os desenvolvimentos recentes, principalmente em engenharia de software, fizeram com que uma base para a construção de sistemas grandes e complexos de veículos autônomos fosse totalmente estruturada. Em particular, a programação de multiagentes mostrou ser uma maneira eficaz de construir e controlar sistemas complexos. Faz-se justo então que uma grande quantidade de pesquisa se concentre em agentes autônomos e sistemas multiagentes, pois tem-se que, em um futuro próximo, a maioria dos ambientes industriais sejam preenchidos com centenas ou milhares de agentes autônomos (WURMAN; D'ANDREA; MOUNTZ, 2008).

Para exemplificar veja o exemplo do sistema KIVA mostrado na Figura 6.

Figura 6 – Esquema de um layout do sistema KIVA. a) Em laranja: Robôs; b) Em verde: Prateleiras; c) Em azul: Estações.



Fonte: (WURMAN; D'ANDREA; MOUNTZ, 2008).

A principal ideia por trás desse sistema mostrado da Figura 6 é simples e ao mesmo tempo ambiciosa: Usar centenas de robôs móveis para trazer os produtos aos trabalhadores sem que estes se desloquem até as prateleiras do armazém e, assim, evitar que o funcionário se desloque toda vez para buscar os produtos, neste caso os robôs buscam as prateleiras para os funcionários que retiram o produto da prateleira. Atualmente esse sistema multiagente já é realidade em muitas empresas, principalmente nos centros de distribuição da Amazon, que adquiriu o sistema KIVA desde 2012 (ANDREA, 2012).

O sistema KIVA de multiagentes possui características importantes que facilitam a realização de tarefas. O sistema sabe qual o momento certo e qual agente deve se deslocar para trazer determinado produto em determinada prateleira, e isso tudo em tempo real e sem bater ou colidir com outros agentes.

2.2 Controladores e Microcontroladores

2.2.1 Plataforma Arduino

O Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software que são de fácil manuseio. As placas Arduino são capazes de ler entradas digitais e analógicas tais como luz em um sensor, um dedo em um botão ou uma mensagem no Twitter, e transformá-lo em uma saída tais como ativar um motor, ligar um LED ou publicar algo online. Assim, o Arduino pode ser controlado através de instruções que são enviadas para o microcontrolador na placa. Para fazer isso, uma linguagem de programação Arduino (baseada em *Wiring*) e o Software Arduino (IDE) (baseada em *Processing*) deve ser utilizada (ARDUINO, 2018a).

Ao longo dos anos, o Arduino tem sido o "cérebro"de milhares de projetos, desde objetos do cotidiano até instrumentos científicos complexos. Uma comunidade mundial de criadores - estudantes, amadores, artistas, programadores e profissionais - reuniu-se em torno dessa plataforma de código aberto (*open source*), e suas contribuições contribuíram para uma incrível quantidade de conhecimento acessível que pode ser de grande ajuda para novatos e especialistas (ARDUINO, 2018a).

O Arduino nasceu no *Ivrea Interaction Design Institute* (IIDI) como uma ferramenta fácil para prototipagem rápida, destinada a estudantes sem formação em eletrônica e programação. Assim que atingiu uma comunidade mais ampla, a placa Arduino começou a mudar para se adaptar às novas necessidades e desafios, diferenciando sua oferta de placas de 8 bits simples para aplicativos destinados a *Internet of Things* (IoT), impressão 3D e ambientes incorporados. Todas as placas do Arduino são completamente *open source*, capacitando os usuários a construí-las independentemente e, eventualmente, adaptá-las às suas necessidades específicas. O software também é de código aberto e está crescendo através das contribuições dos usuários em todo o mundo (ARDUINO, 2018a).

Graças à sua experiência de usuário simples e acessível, o Arduino tem sido usado em milhares de diferentes projetos e aplicações. O software Arduino é fácil de ser utilizado e flexível o suficiente para usuários avançados. Ele é executado no Mac, Windows e Linux. Professores e alunos o utilizam para construir instrumentos científicos de baixo custo, para provar princípios de química e física, ou para começar a programação e robótica. Designers e arquitetos constroem protótipos interativos, músicos e artistas para instalações e para experimentar novos instrumentos musicais.

O Arduino simplifica o processo de programação de microcontroladores, oferecendo vantagens para professores, alunos e amadores interessados. Pode-se listar algumas vantagens:

- **Custo**: As placas Arduino são relativamente baratas em comparação com outras plataformas de microcontroladores. A versão mais barata do módulo Arduino pode ser montada manualmente e até mesmo os módulos Arduino pré-montados custam menos de R$30,00 reais.
- **Multi-plataforma**: O software Arduino (IDE) é executado nos sistemas operacionais Windows, Mac e Linux. A maioria dos sistemas de microcontroladores é limitada ao Windows.
- **Ambiente de programação simples**: O Arduino Software (IDE) é fácil uso para iniciantes e flexível o suficiente para os usuários mais avançados.
- **Software *open source***: O software Arduino é publicado como ferramentas *open source*. A linguagem pode ser expandida através de bibliotecas C++, podendo ainda se entender a linguagem de programação AVR-C, a qual o Arduino se baseia. Da mesma forma, pode-se adicionar o código do AVR-C diretamente em programas do Arduino.
- **Hardware *open source***: Os projetos elétricos das placas Arduino são publicados sob uma licença *Creative Commons* o que possibilita a modificação por designers de circuitos experientes, estendendo-o e aprimorando-o. Até mesmo usuários relativamente inexperientes podem construir a versão de *breadboard* do módulo para entender como funciona e economizar dinheiro.

Porém o Arduino apresenta algumas desvantagens com relação ao desenvolvimento de projetos mais robustos e que precisam de um processamento mais eficiente e rápido. Dessa forma pode-se listar algumas desvantagens da plataforma Arduino:

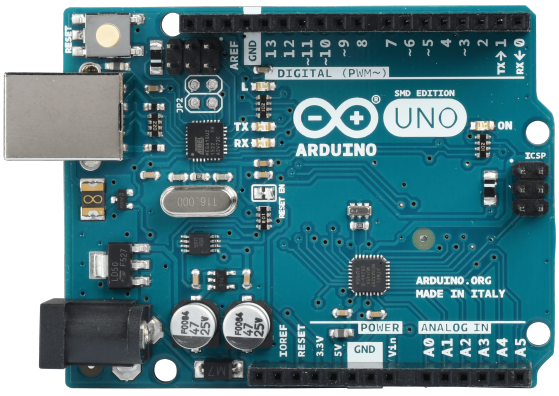
- **Processamento**: A maioria das placas Arduino possuem um clock de 16MHz. Dependendo dos requisitos do projeto essa velocidade de processamento deixa a desejar visto que o processamento não é paralelo, fazendo com que a leitura de alguns sensores ou até mesmo o recebimentos de informações via rádio transmissor fiquem prejudicados.

- **Plataforma de prototipagem**: Por mais que seja uma plataforma interessante para o desenvolvimento de projetos, dependendo da sua aplicação a plataforma fica sendo inviável para o uso em projetos comerciais, tendo então que ser remodelada de acordo com as necessidades.
- **Conhecimento Superficial**: Ao se programar um Arduino, o ponto de partida é usar uma plataforma própria que trabalha com um programa já pré-programado no microcontrolador utilizado na placa, o *bootloader*. Dessa forma a programação é facilitada porém o usuário perde o controle total de todas as configurações do microcontrolador, o que em alguns casos pode ser desvantajoso.
- **Portabilidade**: Os projetos desenvolvidos no Arduino são compatíveis somente com essa plataforma, o que dificulta a portabilidade com outros microcontroladores como os da família PIC da empresa Microchip.

Existem hoje no mercado diferentes tipos de placas Arduino. Cada placa possui características especificas, seja em tamanho, quantidade de pinos digitais e analógicos ou processamento. De todos os modelos conhecidos de Arduino um deles é largamente conhecido em todo o mundo: o Arduino Uno.

A Figura 7 mostra um Arduino Uno. O Arduino Uno é uma placa micro-controlada baseada no microcontrolador ATMEGA328P da empresa Atmel. Possui 14 pinos de entrada/saída digitais (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um cristal de quartzo de 16 MHz, uma conexão USB, um conector de energia, um conector In-Circuit Serial Programming (ICSP) e um botão de *reset*. Ele contém tudo o que é necessário para suportar o microcontrolador; Basta conectá-lo a um computador com um cabo USB ou ligá-lo com uma fonte de alimentação ou bateria apropriada (ARDUINO, 2018b).

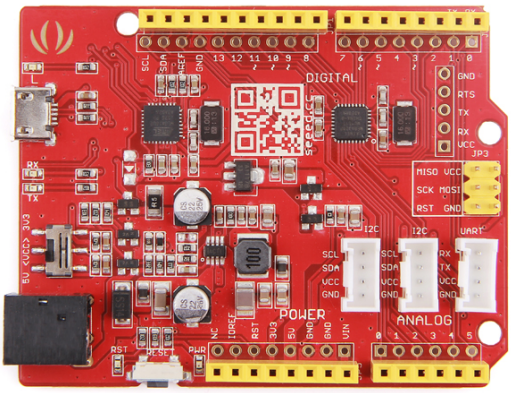
Figura 7 – Arduino Uno R3.



Fonte: (ARDUINO, 2018b).

Outra plataforma, baseada no Arduino, bem conhecida no mundo dos hobistas é a plataforma Seeeduino. A Figura 8 mostra uma placa Seeeduino V4.2.

Figura 8 – Seeeduino versão V4.2, baseada no Arduino Uno.



Fonte: (SEEDUINO, 2018).

O Seeeduino V4.2 é baseado no Arduino Uno. Possui um ATMEGA16U2 como um conversor UART-para-USB, o que significa que a placa pode basicamente funcionar como um chip FTDI. Pode-se programar a placa através de um cabo micro-USB. Além disso, pode-se alimentar a placa através de um Jack de entrada DC entre 7 e 15V. Há um interruptor para escolher a tensão de alimentação do sistema, 3.3V ou 5V, o que é muito útil para configurar o sistema para 3.3V, facilitando a compatibilidade com diversos sensores (SEEDUINO, 2018).

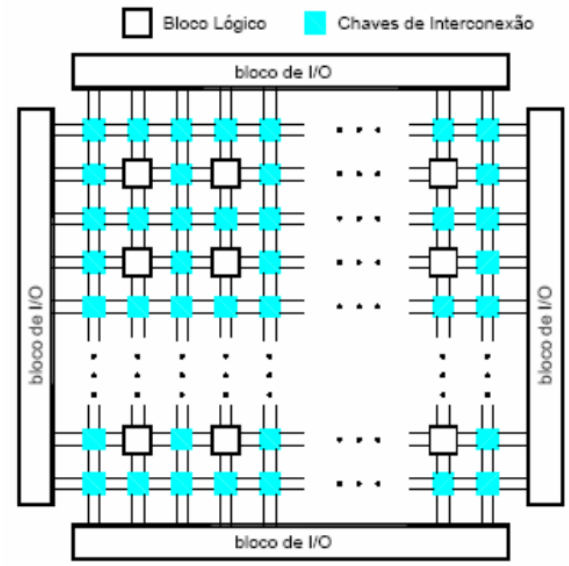
2.2.2 Field Programmable Gate Array - FPGA

Introduzido pela empresa Xilinx Inc. no ano de 1985, o *Field-Programmable Gate Array* (FPGA) é um dispositivo lógico programável que suporta a implementação de circuitos lógicos relativamente grandes. Os FPGAs não possuem planos de portas OR ou AND, em vez disso, estes componentes consistem de um grande arranjo de células configuráveis (ou blocos lógicos) que podem ser utilizadas para a implementação de funções lógicas. A Figura 9 ilustra como é a estrutura de um FPGA (NORONHA et al., 2015).

Segundo Noronha et al. (2015) um FPGA possui três tipos principais de recursos: blocos lógicos, blocos de entrada e saída (I/O), e chaves de interconexão. Os blocos lógicos formam um arranjo bidimensional enquanto as chaves de interconexão são organizadas como canais de roteamento horizontal e vertical entre as linhas e colunas de blocos lógicos. Estes canais de roteamento possuem chaves programáveis que permitem conectar os blocos lógicos de maneira conveniente, em função das necessidades de cada projeto.

Os FPGAs têm sido responsáveis pelas principais mudanças no modo em que os circuitos digitais são projetados. Os fabricantes destes dispositivos utilizam-se de diferentes tipos de blocos lógicos. O mais comumente utilizado é o *Look-Up Table* (LUT). Este tipo de bloco lógico contém células de armazenamento que são utilizadas para implementar pequenas funções

Figura 9 – Estrutura interna de um FPGA.



Fonte: (NORONHA et al., 2015).

lógicas (NORONHA et al., 2015).

Nos FPGAs disponíveis comercialmente, os LUTs possuem geralmente quatro ou cinco entradas, o que exige 16 e 32 células de armazenamento respectivamente.

Quando um circuito lógico é implementado em um FPGA, os blocos lógicos são programados para realizar as funções necessárias, e os canais de roteamento são estruturados de forma a realizar as interconexões necessárias entre os blocos lógicos. As células de armazenamento dos LUTs de um FPGA são voláteis, o que implica na perda do conteúdo armazenado no caso de falta de alimentação elétrica. Desta forma, o FPGA deve ser programado toda vez que for energizado. Geralmente um pequeno chip de memória do tipo *Programmable Read-Only Memory* (PROM) é incluído nas placas de circuito impresso que contêm FPGAs. As células de armazenamento são automaticamente carregadas a partir das PROMs toda vez que uma tensão elétrica é aplicada a estes chips (NORONHA et al., 2015).

Exitem vários tipos de linguagens de programação e que são destinadas a diversas aplicações. O tipo de linguagem de programação utilizado nos FPGAs são linguagens descritas como Linguagem de Descrição de Hardware, do inglês *Hardware Description Language* (HDLs).

2.2.2.1 Linguagem de Descrição de Hardware

As Linguagens de Descrição de Hardware, do inglês *Hardware Description Languages*, (HDLs) são tipos de linguagens que descrevem circuitos digitais de maneira rápida e prática. A principal diferença entre linguagens HDLs e linguagem de programação de softwares é justamente o paralelismo existente nas linguagens HDLs. Dessa forma os sistemas baseados

em HDLs possuem um alto poder de paralelismo aumentando o poder de processamento dos hardwares. Pode-se citar algumas dessas linguagens como Verilog e VHDL. A seguir a linguagem HDL Verilog é apresentada.

2.2.2.2 HDL Verilog

A linguagem Verilog foi introduzida em 1985 pela Gateway Design Automation. Em 1989, a Gateway foi comprada pela empresa Cadence Design Systems, que tornou a linguagem de domínio público em maio de 1990 com a formação da Open Verilog International (OVI). Hoje o Verilog é um padrão IEEE, já tendo duas extensões ou modificações: Verilog-95 (padrão IEEE 1364-1995), Verilog 2001 (IEEE 1364-2001) e Verilog 2005 (IEEE 1364-2005), mas novos trabalhos continuam em andamento com o desenvolvimento do Verilog-AMS com suporte a sinais analógicos e digitais. O Verilog tem uma grande semelhança com a linguagem de programação C (MIDORIKAWA, 2007).

O Verilog oferece ao projetista os meios para descrever um sistema digital em vários níveis de abstração, e também suporta ferramentas de projeto para síntese lógica. Uma representação abstrata pode ser usada para o projeto antes do projeto detalhado. Com o detalhamento do projeto, são criadas descrições com construções estruturais.

A construção Verilog básica é o *module*. Os sinais de entrada e saída de um módulo (a sua interface) são designados usando *ports*. Cada sinal de um *port* pode ser declarado como *input*, *output* ou *inout* (MIDORIKAWA, 2007).

Um procedimento Verilog é sempre um comando *always* ou *initial*, uma tarefa (*task*) ou função (*function*). Os comandos dentro de um bloco sequencial (comandos que aparecem entre um *begin* e um *end*) que é parte de um procedimento (*procedure*) são executados sequencialmente na ordem em que aparecem. Mas os procedimentos são executados concorrentemente com outros procedimentos.

Um bloco sequencial pode aparecer em um comando *always*, no caso em que o bloco deve ser executado repetidamente. Por outro lado, um comando *initial* especifica um bloco sequencial que é executado apenas uma vez, no início de uma simulação (MIDORIKAWA, 2007).

Uma *task* é um tipo de procedimento, chamado por outro procedimento, que tem entradas e saídas, mas não tem um valor de retorno. Uma tarefa pode chamar outras funções. Uma *function* é um procedimento usado em qualquer expressão, tem ao menos uma entrada, não tem saída e retorna um único valor. Uma função não pode chamar uma tarefa. O Verilog possui um tipo de controle de temporização que atrasa uma atribuição até que um evento específico ocorra. Um controle de evento é especificado por um @. Por exemplo, *@(posedge clk)* indica um evento referente a uma borda de subida do sinal *clk* (MIDORIKAWA, 2007).

2.3 Sistema de Controle

2.3.1 Controlador Proporcional derivativo - PID

O Controlador Proporcional Integral Derivativo, mais conhecido como controlador PID é muito utilizado em sistemas de controle em geral. O Controlador PID é usado em sistemas onde se deseja manter regulado uma certa grandeza, e em certos casos variar de um determinado estado para outro de modo rápido e preciso.

O controlador PID atua no sistema de três formas diferentes, cada uma com suas respectivas características. A primeira delas é a ação proporcional (P) que é responsável por promover uma resposta mais rápida do sistema sob uma variação no sinal de entrada, que é o sinal desejado. A segunda é a ação de integral (I) que tem o objetivo eliminar erros de estado estacionário e assim manter um estado desejável e estável. A terceira é a ação derivativa (D) que faz uma antecipação da correção do valor de saída do sistema, melhorando também a reação em caso de sobressinal. Combinando essas 3 ações um controlador PID é projetado de forma a se ter uma resposta do sistema mais rápida, estável e desejada em estado estacionário. Dessa forma, o controlador PID é representado pela Equação 2.1 (JOGALEKAR et al., 2013):

$$u(t) = K_p e(t) + K_i \int e(t) + K_d \frac{d}{dt} e(t) \tag{2.1}$$

A Equação 2.1 representada como equação de transferência é dada pela Equação 2.2:

$$G(s) = K_p + \frac{K_i}{s} + K_d s \tag{2.2}$$

onde:

K_p : Ganho Proporcional.
K_i : Ganho Integral.
K_d : Ganho Derivativo.

2.3.2 Algoritmos Genéticos

2.3.2.1 Visão Geral - Algoritmos Genéticos

Algoritmos Genéticos (AGs) são métodos de otimização e busca inspirados nos mecanismos de evolução de populações de seres vivos. Foram introduzidos por John Holland e popularizados por um dos seus alunos, David Goldberg. Estes algoritmos seguem o princípio da seleção natural e sobrevivência do mais apto, declarado pelo naturalista e fisiologista inglês Charles Darwin em seu livro "A Origem das Espécies". De acordo com Charles Darwin, "quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes". Otimização é a busca da melhor solução pra um dado problema. Consiste

em tentar várias soluções e utilizar a informação obtida neste processo de forma a encontrar soluções cada vez melhores. Um exemplo simples de otimização é a melhoria da imagem das televisões com antena acoplada no próprio aparelho. Através do ajuste manual da antena, várias soluções são testadas, guiadas pela qualidade da imagem obtida na TV, até a obtenção de uma resposta ótima, ou seja, uma boa imagem (PACHECO, 1999).

Para o emprego de técnicas de busca e otimização, geralmente deve-se ter: A) Um espaço de busca onde estão contidas todas as possíveis soluções para o problema proposto; e B) Um função objetivo (função *fitness*) utilizada para avaliar as soluções produzidas para cada uma das soluções produzidas, dando a ela um nota. Em outras palavras, otimizar é encontrar o ponto de máximo ou mínimo que satisfaça a função *fitness*.

Na busca por soluções que satisfaça uma função objetivo, seja otimizando para pontos máximo global ou mínimo global (pontos desejados), a função de *fitness* pode encontrar pontos de máximos ou mínimos falsos, os chamados pontos máximos locais ou pontos de mínimos locais. Para resolver problemas desse tipo são empregados técnicas apropriadas para que tais circunstancias não aconteçam, e se acontecerem que possam contornar de forma inteligente essas soluções locais rumo a solução global (PACHECO, 1999).

Para a implementação de algoritmos genéticos simples o principal ponto de partida é a escolha de uma população inicial de cromossomos (PACHECO, 1999), formada por um conjunto aleatório de soluções do problema proposto. Quando o processo evolutivo começa, ou seja, quando o cruzamento de indivíduos acontece surgem novos indivíduos do cruzamento. Cada indivíduo dessa nova população recebe uma nota de acordo com a função *fitness*. Os indivíduos mais aptos são selecionados e os menos aptos são descartados. Antes do próximo cruzamento pode haver mudanças nos cromossomos aptos em forma de *crossover* e mutação. Então, todo o processo se repete até se atingir o objetivo desejado ou até se obter uma resposta satisfatória.

2.3.2.2 O Algoritmo Genético Compacto - AGc

Os Algoritmos Genéticos Compactos (AGc) representam a população como uma distribuição de probabilidade sobre o espaço de soluções e são equivalentes ao algoritmo genético simples com crossover uniforme (CASSEB, 2012). O AGc foi proposto para sua utilização com parâmetros binários.

Uma dificuldade encontrada na utilização do Algoritmo Genético é a demanda por memória, exigida pela grande quantidade de indivíduos nas populações, impossibilitando assim a implementação em sistemas embarcados. Visando solucionar este problema e fornecer uma outra perspectiva das características dos AGs, foi proposto o Algoritmo Genético Compacto (HARIK; LOBO; GOLDBERG, 1999). O desenvolvimento do AGc revelou que não é necessário gerar uma população inteira com a finalidade de escolher apenas alguns indivíduos. Um desempenho similar ocorre por gerar apenas os indivíduos a serem utilizados através de um

vetor de probabilidade e depois atualizar tal vetor baseado nos indivíduos gerados. Cada valor do vetor de probabilidade contém a chance de um gene do indivíduo ser igual à 1 (CASSEB, 2012).

Existem muitos problemas de engenharia que podem ser solucionados de maneira mais simples aplicando AGs. Um problema muito comum inerente a área de robótica móvel é a má sintonização de controladores Proporcional Integral Derivativo (PID). De modo geral o controlador PID precisa ser sintonizado de modo a operar especificamente a uma planta de maneira rápida e precisa. Não é trivial a escolha dos ganhos do controlador levando a um mau funcionamento da planta.

Para uma melhor análise considere o Algoritmo 1.

Usualmente, o vetor de probabilidade é inicializado contendo apenas um número binário que representa a chance de cada bit do indivíduo ser 1, que inicialmente é de 50%. Dentre os indivíduos geradores, escolhe-se o melhor deles e o vetor de probabilidade é atualizado, fazendo as probabilidades tenderem a este melhor indivíduo em um fator de 1/n.

O fitness dos dois indivíduos gerados serão comparados, e o vetor de probabilidade será modificado pendendo ao melhor indivíduo. Para simular uma população de n indivíduos, o vetor probabilidade será incrementado ou decrementado em 1/n unidades. Este processo se repete até que todos os valores do vetor probabilidade sejam 0 ou 1.

Este processo é repetido até que o vetor de probabilidade atinja seu critério de parada, que é similar ao de um algoritmo genético comum, tal como o tempo de execução, número de gerações, ou, neste caso, quando o vetor de probabilidade possuir apenas chances nulas (0%) ou máximas (100%).

Este método simula a seleção e o crossover assim como no AG, porém não o faz com a mutação.

A alocação de memória do AGc é menor do que o AG simples, sendo este o principal motivo de sua utilização e assim podendo substituir muito bem o AG (HARIK; LOBO; GOLDBERG, 1999). Apesar de não substituí-lo em sua integridade, os resultados obtidos mostram a similaridade com os resultados obtidos através do AG.

Para fazer a correspondência dos genes em números reais com os genes representados de forma binária no AGc, utilizou-se neste trabalho uma relação entre o espaço real e o binário. Um indivíduo é composto por genes que serão convertidos depois para o correspondente valor de Kp, Ki e Kd, valor real. Assim uma parte dos genes do indivíduo representa Kp, outra parte Ki e outra Kd.

A implementação em hardware é de grande valia para o AG, pois é possível se beneficiar do paralelismo intrínseco à estes algoritmos, ou seja, através do paralelismo dos cálculos de todos os operadores do AG, a velocidade de processamento é suficiente para uma estimação em

Algoritmo 1: Algoritmo de um AGc padrão.

```
1  //Parâmetros:
2  n: Tamanho da população;
3  l: Comprimento do cromossomo em bits;

4  //1) Inicialização: Iniciar o vetor de probabilidade
5  for i = 1; i <= l; i++ do
6      p[i] ← 0.5;

7  //2) Geração: Gerar dois indivíduos através de PV
8  a ← gerar(p);
9  b ← gerar(p);

10 //3) Competição: Os indivíduos são avaliados e competem segundo um valor de fitness
11 a_fitness ← avaliar(a);
12 b_fitness ← avaliar(b);
13 if a_fitness > b_fitness then
14     vencedor ← a;
15     perdedor ← b;
16 else
17     vencedor ← b;
18     perdedor ← a;

19 //4) Atualização: Atualizar o vetor de probabilidade na direção do vencedor
20 for i = 1; i ≤ l; i++ do
21     if vencedor[i] ≠ perdedor[i] then
22         if vencedor[i] = 1 then
23             p[i] ← p[i] + 1/n;
24         else
25             p[i] ← p[i] − 1/n;

26 //5) Verificação: Verificar se o vetor convergiu
27 for i = 1; i ≤ l; i++ do
28     if p[i] > 0 & p[i] < 1 then
29         Vá para o passo 2;
30     return ← p;

31 //6) Finalização: p representa a solução final
32 return ← p;
```

tempo real.

AGcs podem resolver problemas de maneira semelhante a AGs, mesmo sem a utilização de crossover (HARIK; LOBO; GOLDBERG, 1999). O problema proposto aqui é ligado diretamente ao controle de velocidade e posicionamento de motores DC. No próximo capítulos

será apresentada a metodologia utilizada nesse trabalho.

2.3.2.3 Estado da arte e comparações

Depois de todos os conceitos apresentados anteriormente fica evidente algumas técnicas que podem ter um resultado mais expressivo com relação ao projeto atual dos robôs multiagentes. Esses problemas vão deste técnicas de melhoria do sistema de controle através de AGcs até o desenvolvimento de novas estruturas. A Tabela 2 mostra as principais características dos pontos esperados de melhoria para o projeto dos robôs multiagentes.

Tabela 2 – Principais Características dos Sistemas abordados e a serem projetados.

Item de Desenvolvimento	Robô Atual	Robô a ser Desenvolvido
Sistema de Controle	Sistema de controle PID.	Sistema de controle PID.
Método de obtenção de constantes kp, ki e kd	Obtenção de constantes de modo empírico.	Obtenção através de Algoritmos Genéticos (AGc) por meio de simulação usando a respectiva função de transferência do motor.
Peso do agente	450 gramas.	280 gramas.
Controlador/Microcontrolador	Controlado de 16MHz, um Seeeduino V2.1, com programação idêntica a um Arduino Uno R3.	FPGA com processamento paralelo e clock de 50MHz.
Função de Transferência Empregada	Função obtida através de referência bibliográfica com ajuste mínimos e imprecisos.	Função obtida através de referência bibliográfica com ajustes consistentes de taxa de transmissão.
Linguagem de programação	Linguagem C++ juntamente com bibliotecas decomponentes utilizados	Linguagem de descrição de Hardware Verilog com programação paralela.
Consumo do controlador/Microcontrolador	Controlador em modo de espera de mensagem consome em média 130mA, com o sistema todo ligado e sem movimentação dos motores. Valores medidos empiricamente.	Controlador em modo de espera de mensagem consome em média 80mA, com o sistema todo ligado e sem movimentação dos motores. Valores medidos empiricamente.
Características Físicas, Elétricas e Mecânicas	O robô possui um chassi de alumínio, ponte H, rádio transmissor, motores e uma placa conectora.	O robô possui um chassi de alumínio, ponte H, rádio transmissor, motores e uma placa conectora.

3 METODOLOGIA UTILIZADA

A metodologia utilizada nesse trabalho consistiu em desenvolver o projeto de um único agente para fins de testes e de aprimoramento de algoritmos de controle no que diz respeito a calibração de controladores PID. Dessa forma foi possível explorar os recursos necessários para que o robô fosse construído e testado durante a execução desse trabalho. Nesse sentido, os métodos e pesquisas desenvolvidas para a construção e programação do robô seguiram os seguintes passos:

1. Foi realizada uma pesquisa bibliográfica de projetos utilizando multiagentes existentes no meio científico bem como possíveis aplicações no mercado e na indústria. Foram adquiridas informações para utilização no projeto.

2. Foi Projetado, modelado (em ambiente CAD 3D) e escolhido os componentes necessários para a montagem do robô físico, respeitando todas as requisições impostas pelas regras pertinentes. Foram utilizados métodos de modelagem 3D e de projeto eletrônico de placas de circuito impresso (PCI).

3. Foi programado o robô utilizando linguagem de programação e software apropriado para FPGAs. Essa programação foi feita utilizando-se métodos de programação de módulos paralelos facilitando a implementação e o entendimento geral do código do robô.

4. Foi utilizado um sistema de calibração de controladores baseados em AGs para sintonizar o controlador do robô em ambiente computacional, desejando-se a melhoria da performance do robô comparado a versões anteriores do projeto. Nesse caso foram utilizados algoritmos genéticos compactos para essa tarefa.

5. Foram feitos testes reais com o controlador sintonizado de acordo com resultados obtidos em simulações no robô projetado. Essa parte precisou ser realizada utilizando-se de métodos experimentais com o sistema de controle e visão computacional previamente desenvolvido PMEC-UFG.

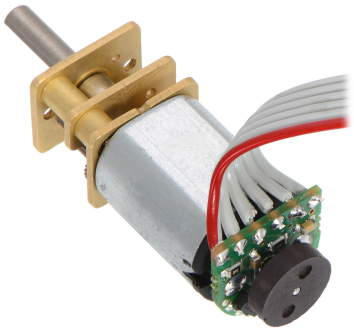
Os métodos enumerados possuem uma ordem cronológica e foram aplicados no decorrer do desenvolvimento do robô proposto. Nas próximas secções serão discutidas os assuntos que foram necessários para a implementação do projeto e assim cumprir os métodos descritos anteriormente.

3.1 Equação de Transferência do Motor

O controle de motores DC está ligado diretamente a sua equação de transferência, que como um modelo matemático representa como uma tensão elétrica aplicada ao motor se comporta, fornecendo rotação e velocidade angular como saída do sistema. A Figura 10 mostra um micromotor metálico com redução de 75,81:1, ideal para aplicações de robôs com tamanho reduzido. Segundo (OKUYAMA; MAXIMO; PINTO, 2015) a função de transferência desse tipo de motor e dada pela aproximação apresentada na Equação 3.1.

$$G_{motor} = \frac{3694,6}{s + 36,73} \frac{rpm}{V} \tag{3.1}$$

Figura 10 – Micromotor metálico com redução 75,81:1 e encoder magnético de efeito Hall (12 CPR).



Fonte: (POLOLU, 2017a).

A Equação 3.1 foi desenvolvida para um micromotor metálico com redução de 51,45:1 (OKUYAMA; MAXIMO; PINTO, 2015). O ajuste deve ser feito bastando multiplicar a Equação 3.2 por 51,45 e dividir por 75,81, que é o *gear ratio* adotado aqui. Assim a função de transferência será aquela mostrada na Equação 3.2.

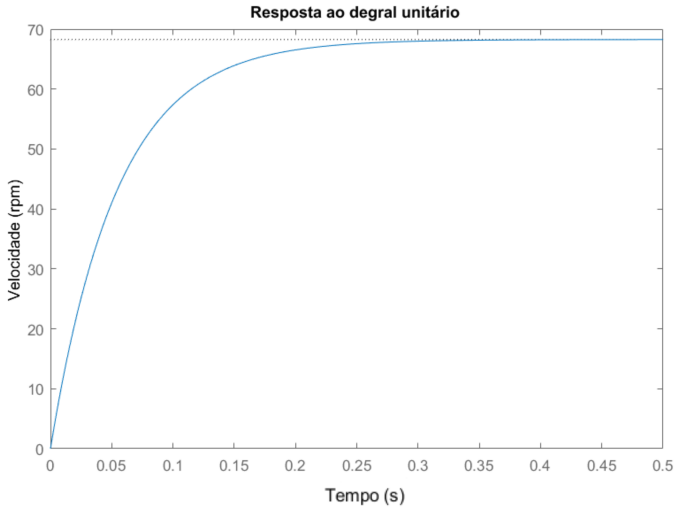
$$G_{motor} = \frac{2507,41}{s + 36,73} \frac{rpm}{V} \tag{3.2}$$

Fazendo-se uma plotagem simples da responta a um degrau unitário obtém-se a resposta da Figura 11. Essa figura mostra que para uma entrada de 1V (Degrau Unitário) o motor demora cerca de 300ms para estabilizar em uma velocidade próxima de 70rpm.

Assim o objetivo é usar um AGc para calcular as constantes Kp, Ki e Kd de um controlador PID. Os requisitos desejados para a resposta final com o controlador PID são os seguintes:

- Tempo de estabilização < 100ms;
- Overshoot < 1%;

Figura 11 – Resposta ao Degrau Unitário da Equação 3.2.



Fonte: Próprio Autor.

- Erro de estado estacionário <= 2%.

3.2 Controlador PID

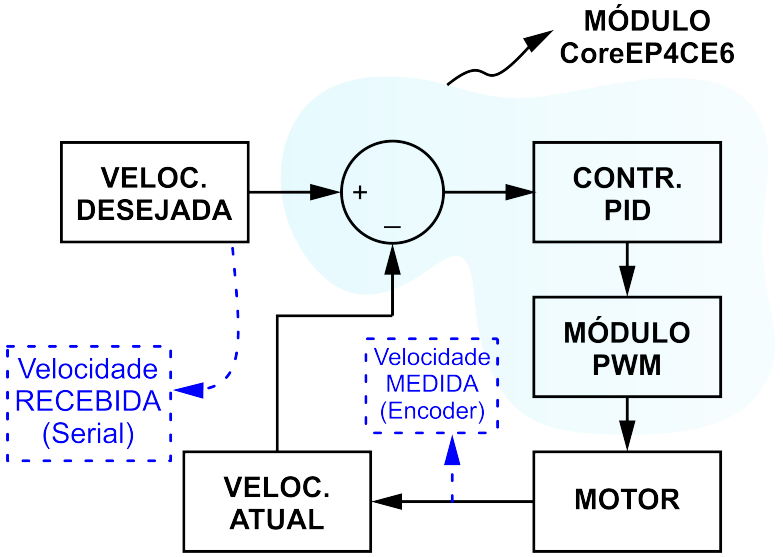
O Controlador Proporcional Integral Derivativo, mais conhecido como controlador PID é muito utilizado em sistemas de controle em geral. O Controlador PID é usado aqui para controlar a velocidade de motores DC através da modulação de largura de pulso (PWM).

Para a implementação de um controle PID em sistemas embarcados é necessário escolher um microcontrolador ou o chip que irá fazer todo o processamento e cálculos necessários. Utilizar FPGAs ao invés de microcontroladores traz mais flexibilidade e melhor eficiência energética além de diminuir custos.

A Figura 12 mostra um diagrama de malha fechada de um sistema de controle PID muito característico implementado com FPGA com o módulo CoreEP4CE6 da Wave Share (WAVESHARE, 2017). Analisando a malha fechada mostrada na Figura 12 nota-se que existe um erro que é corrigido entre entrada e saída do sistema a cada ciclo de processamento do FPGA pois essa é a característica de um sistema de controle de malha fechada. Após as correções necessárias feitas pelo FPGA, este é encarregado de atuar nos motores de modo a corrigir a velocidade aumentando ou diminuindo o *duty cycle* do PWM. A realimentação da malha fechada é feita pelos encoders dos motores que fazem uma contagem de quanto o motor gira a cada intervalo de tempo pré-estabelecido.

O Sistema de controle PID mostrado na Figura 12 é ajustado por três constantes: Kp,

Figura 12 – Sistema de controle desenvolvido.



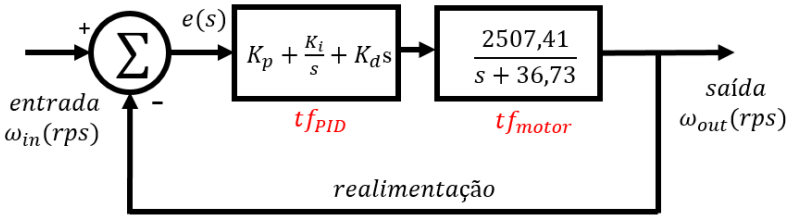
Fonte: Próprio Autor.

Ki e Kd. A Equação do controlador PID é dada pela Equação 3.3 (JOGALEKAR et al., 2013).

$$u(t) = K_p e(t) + K_i \int e(t) + K_d \frac{d}{dt} e(t) \tag{3.3}$$

Utilizando as Equações 3.2 e 2.2 pode-se montar a malha fechada desejada, mostrada na Figura 13.

Figura 13 – Sistema em malha fechada do motor e do controlador PID a ser projetado.



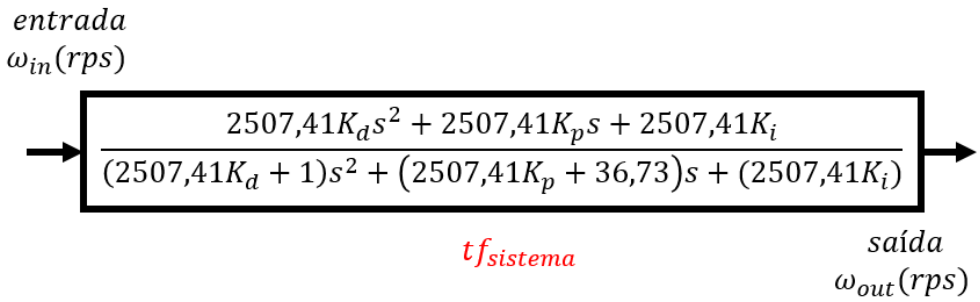
Fonte: Próprio Autor.

A partir do sistema apresentado na Figura 14 será possível testar os valores das constantes Kp, Ki e Kd geradas pelo AGc projetado e assim alimentar a função fitness para convergência do algoritmo.

A Figura 15 mostra o diagrama do PID que deve ser implementado no FPGA com somadores, multiplicadores e registradores bem como outros recursos.

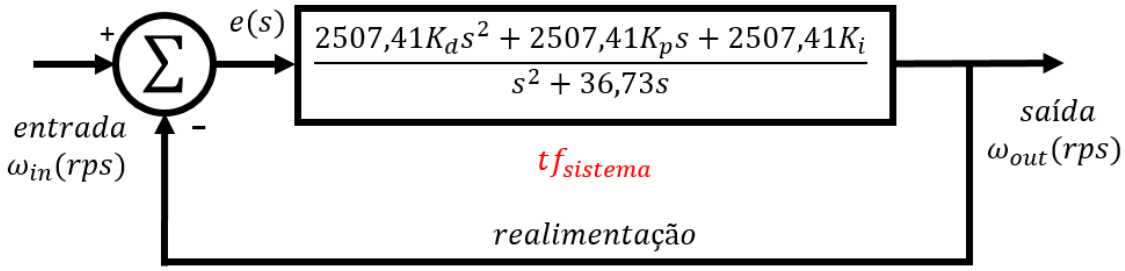
Com o controlador PID pode-se fazer o controle das velocidades requeridas pelo motor. A cada interação o erro é calculado e é utilizado para o cálculo de cada parcela do controlador. O controlador proporcional tende a chegar perto do seu objetivo proporcionalmente ao erro que

Figura 14 – Função de transferência entre entrada e saída do sistema com controlador e malha direta sem realimentação.



Fonte: Próprio Autor.

Figura 15 – Controlador PID utilizado.



Fonte: Próprio Autor.

ele está do mesmo. O controlador Integrativo diminui os erros a regime permanente de modo a evitar pequenas variação, e por fim o controlador Derivativo ajuda na estabilização mais rápida diminuindo o tempo de subida, evitando variações. O algoritmo 2 apresenta o código que foi implementado do controle de motores DC.

3.3 Implementação do AGc

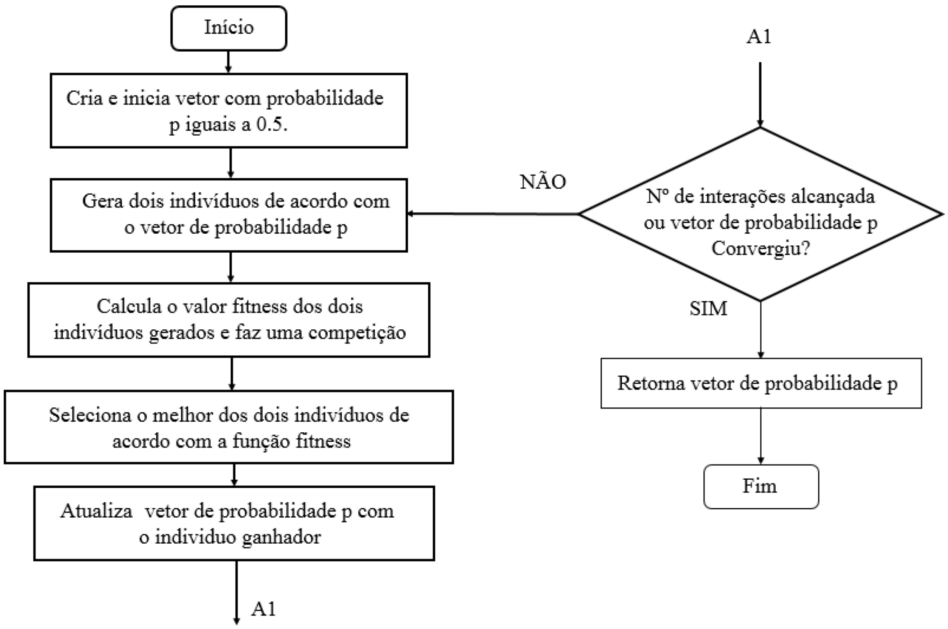
Seguindo os passos do Algoritmo 1 implementou-se o AGc, esquematizado na Figura 16.

Antes de iniciar o algoritmo é preciso atribuir valores a algumas constantes de modo a ter o controle das interações. A primeira delas é vetor com os limites que as constantes que Kp, Ki e Kd podem alcançar. A população inicial é definida pela codificação do parâmetro PID em *Strings* binárias conhecidas como cromossomo. O comprimento das *Strings* depende da precisão necessária, que no caso é de 4 casas decimais, e pode ser definida diferentemente para cada uma das três constantes do controlador PID. A Equação 3.4 é utilizada para calcular a quantidade de bits necessárias de acordo com os limites das contantes (SAAD; JAMALUDDIN;

Algoritmo 2: Algoritmo do controlador PID.

```
1  if posedgeclk then
2      //1) Cálculo do ganho Proporcional (P)
3      actualError <= desiredVelocity - actualVelocity;
4      gP <= actualError * Kp;
5      //2) Cálculo do ganho Integral (I)
6      accumulator <= acumulator + actualError;
7      gI <= accumulator * Ki;
8      //3) Cálculo do ganho Derivativo (D)
9      gD <= (actualError - lastError) * Kd;
10     lastError = actualError;
11     //4) Soma dos ganhos P, I e D
12     gPID <= gP + gI + gD;
13     //5) Aplicação da velocidade ao motor
14     pwmOut <= gPID;
```

Figura 16 – Diagrama de funcionamento do AGc.



Fonte: Próprio Autor.

DARUS, 2012).

$$2^{m_j-1} < (b_j - a_j) \times 10^4 \leq 2^{m_j} - 1 \tag{3.4}$$

onde:

m_j : Quantidade de bits necessários.

a_j : Limite inferior do intervalo.
b_j : Limite superior do intervalo.

Utilizou-se os seguintes limites para as contantes: $K_p \in [0\ 8.0]$, $K_i \in [0\ 3.0]$ e $K_d \in [0\ 2.0]$. Esses limites fora escolhidos de acordo com as constantes apresentadas por Okuyama, Maximo e Pinto (2015).

O função fitness escolhida para avaliar os indivíduos do experimento foi aquela mostrada na Equação 3.5 (JALILVAND et al., 2011).

$$F_{obj} = \left\{ \left(100E_{ss}^{0.5} + 5M_P^2\right) \left(10t_s + t_r\right) \right\} \tag{3.5}$$

onde:

t_r : Tempo de subida.
t_s : Tempo de estabilização.
M_P : Sobressinal (overshoot) máximo.
E_{ss} : Erro de estado estacionário.

Toda vez que se avalia um indivíduo é preciso decodificar os bits obtendo as constantes Kp, Ki e Kd para serem aplicadas na função de transferência da Figura 14 e assim fazer a avaliação de acordo com a função fitness da Equação 3.5. Essa decodificação é feita de acordo com a Equação 3.6 (SAAD; JAMALUDDIN; DARUS, 2012).

$$x_j = a_j + decimal(substring_j) \times \frac{(b_j - a_j)}{2^{m_j} - 1} \tag{3.6}$$

onde:

x_j : Ganho Kp, Ki ou Kd calculado.
m_j : Quantidade de bits necessários.
a_j : Limite inferior do intervalo.
b_j : Limite superior do intervalo.

A parte da Equação 3.6 que cita $decimal(substring_j)$ é a parte do gene de cada indivíduo que corresponde a cada ganho, definido anteriormente. Como um indivíduo é composto por vários bits essa convergência faz-se necessária.

3.4 Comunicação Serial

A velocidade aplicada ao robô é proveniente de uma recepção serial que é feita través de um rádio receptor. Essa comunicação usa um padrão elétrico RS-232 que é usado para fazer uma comunicação serial com o computador coordenador (técnico). Como todos os algoritmos trabalhando juntos, utilizou-se de uma biblioteca serial para FPGA de modo a fazer a comunicação entre o computador e o FPGA. Foram feitos algumas modificações de modo a obter-se uma

melhor compatibilidade com o sistema de controle e protocolos de comunicação utilizados. O Algoritmo 3 mostra como é feito a interpretação pela biblioteca ***async.v*** (NICOLLE, 2006).

Algoritmo 3: Algoritmo para recebimento e transmissão de dados via Serial.

```
1 if posedge clk then
2     //1) Confirma se existe mensagem a ser recebida if RxTdataReady then
3         GPout <= TxDdata;
4     //2) Confirma se existe mensagem a ser enviada if RxDdataReady then
5         GPin <= RxDdata;
```

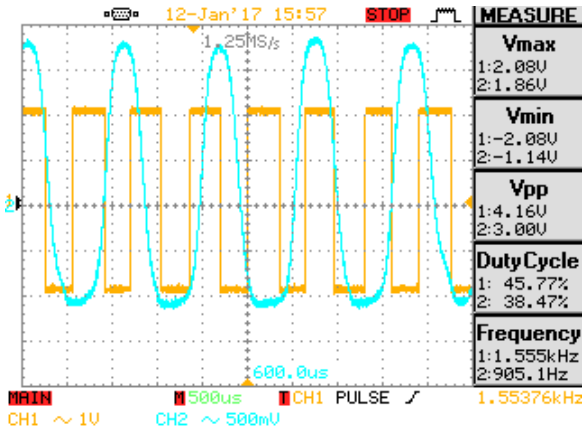
3.5 Leitura dos Encoders e Odometria

Os encoders são empregados para fazer a contagem da quantidade de voltas dadas pelo motor, para que assim, possa-se calcular a velocidade de rotação do motor. Segundo o *datasheet* do motor, para cada volta do eixo das engrenagens, o motor gira 75,81 voltas, gerando uma taxa de transmissão de 75,81:1 (POLOLU, 2017a). Além disso o encoder de quadratura acoplado ao eixo estendido do motor gera uma onda pulsante de 12 CPR (*Counts Per Revolution*) com 6 polos magnéticos em 2 canais de leitura. O papel do encoder é gerar pulsos com frequência proporcional a rotação do motor (LAMáR; KOCSIS, 2013). Cada canal do encoder de quadratura faz uma leitura de 6 pulsos dos quais 3 são de borda de subida da onda e 3 são de borda de descida.

Existe várias maneiras de se calcular a velocidade de rotação de um motor. Uma delas é contar quantos pulsos são gerados em um intervalo de tempo definido. Dessa forma sabe-se exatamente quantos pulsos por segundo são gerados e consequentemente a velocidade do motor ao qual o encoder está acoplado. Outra maneira é ao invés de fixar o tempo de contagem dos pulsos é saber o tempo de um único pulso. Ambos exemplos citados são empregados de maneira abundante no que diz respeito a leitura de pulsos de encoders. O primeiro é mais utilizado em controladores onde o *loop* de controle é extenso, de modo que, dependendo da velocidade de giro do motor esse não consegue fazer as contagens corretamente caso fosse pulso a pulso. O segundo possui maior precisão mas requer um processamento mais rápido devido a quantidade exagerada de pulsos por segundo que deve-se ler.

Os encoders mais conhecidos são os ópticos e magnéticos. A Figura 17 mostra duas formas de onda diferente. A onda com formato senoidal é aquela fornecida por um encoder óptico enquanto que a onda quadrada é aquela fornecida por um encoder magnético. Em se tratando de sistemas digitais é trivial que a onda do encoder magnético leva mais vantagens com relação onda do encoder óptico. Uma desvantagem característica é que os encoders ópticos podem fornecer mais de uma transição entre níveis lógicos para um mesmo pulso, enquanto que os encoder magnéticos são mais difíceis de acontecer devido a sua forma de onda.

Figura 17 – Forma de onda dos encoder magnético e óptico.



Fonte: Próprio Autor.

Como o processamento do FPGA é de 50MHz, esse valor é mais do que o suficiente para contar o tempo de um único pulso. Levando-se em consideração a velocidade máxima do motor, que é de 430 RPM (Rotações por Minuto), tem-se a relação mostrada na Equação 3.7.

$$Freq_{max} = \frac{pulsos_{enc} \times taxa_{trans} \times vel_{motor}}{60} Hz \tag{3.7}$$

onde:

$Freq_{max}$: Frequência máxima que a onda gerada pelo encoder pode alcançar (em Hz).
$pulsos_{enc}$: Quantidade de pulsos por revolução do motor.
$taxa_{trans}$: Taxa de transmissão entre eixo do motor e eixo final.
vel_{motor} : Velocidade máxima que o motor pode alcançar (em RPM).

De acordo com a equação 3.7 e com os dados apresentados anteriormente tem-se uma $Freq_{max}$ = 1629,91 Hz. Por outro lado deve-se obter uma equação de modo a envolver o tempo de um pulso com a frequência do clock do FPGA. Dessa forma, tem-se a Equação 3.8 que representa a velocidade do motor em relação ao valor do contador do FPGA, e que está sincronizado com o clock do mesmo.

$$Vel_{atual} = \frac{FPGA_{freq}}{pulse_{timer} \times pulsos_{enc} \times taxa_{trans}} rps \tag{3.8}$$

onde:

Vel_{atual} : Velocidade Atual calculada (em rps).
$FPGA_{freq}$: Frequência de operação do FPGA.
$pulsos_{enc}$: Quantidade de pulsos por revolução do motor.
$pulse_{timer}$: Tempo de um único pulso contado pelo FPGA.
$taxa_{trans}$: Taxa de transmissão entre eixo do motor e eixo final.

Todas as variáveis da Equação 3.8 são constantes, exceto $pulse_{timer}$ que é o tempo contado para um único pulso do encoder. Assim a Equação 3.8 pode ser reescrita de modo a utilizar uma constantes geral, utilizando os valores das constantes mencionadas. O valor da contante encontrado foi de K_{timer} = 219848. A equação reescrita é aquela mostrada na Equação 3.9.

$$Vel_{atual} = \frac{219848}{pulse_{timer}} rps \tag{3.9}$$

onde:

Vel_{atual} : Velocidade Atual calculada (em rps).
$pulse_{timer}$: Tempo de um único pulso contado pelo FPGA.

Utilizando a Equação 3.9, o algoritmo 4 foi desenvolvido para o cálculo da velocidade do motor levando em consideração o método de contagem do tempo de um único pulso do encoder por vez.

Algoritmo 4: Algoritmo para calculo da velocidade do motor.

```
1  //1) Clock gerado pelo encoder
2  if negedge pinENC then
3      clk1 = ∼clk1;
4  //2) Calcula a duração de 1 pulso de clock
5  if posedge clk then
6      if clk1 == clk2 then
7          clk2 = ∼clk2;
8          pulseTimer <= 0;
9          velocidadeAtual <= K_timer / pulseTimer;
10     else
11         if pulseTimer >= K_timer then
12             velocidadeAtual <= 0;
13             pulseTimer <= 0;
14         else
15             pulseTimer = pulseTimer + 1;
```

3.6 Geração dos pulsos PWM

A Modulação por Largura de Pulso, do inglês *Pulse Width Modulation* (PWM), é um tipo de modulação onde se deseja controlar o valor médio de um sinal na saída através de pulsos digitais com larguras controladas. Através dessa variação da largura do pulso é possível obter uma tensão média de saída entre os valores máximos e mínimos da entrada.

Tendo-se uma onda quadrada como referência, para gerar um PWM correto é necessário levar em consideração alguns parâmetros para fins de cálculos que são o período da onda quadrada e o comprimento do pulso propriamente dito, que nesse caso é chamado de duty cycle. O valor do duty cycle pode ser calculado pela Equação 3.10.

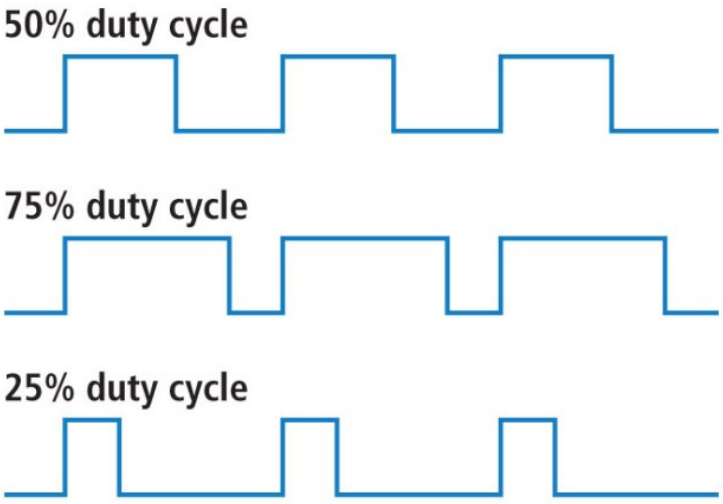
$$Dc = 100 \times \frac{T_{on}}{T} \tag{3.10}$$

onde:

Dc: Duty cycle da onda.
T_{on}: Largura do Pulso em nível alto.
T: Período da largura do pulso completo (1/f).

De acordo com a Equação 3.10 pode-se dizer que para uma largura de pulso com a metade do período nos dá um duty cycle de 50% o que significa que a tensão de saída será a metade daquela máxima do valor do pulso PWM. A Figura 18 mostra alguns exemplos de valores de duty cycle.

Figura 18 – Exemplos de duty cycle.



Fonte: (JORDAN, 2017).

De acordo com a Figura 18 pode-se dizer que para uma largura de pulso com a metade do período nos dá um *duty cycle* de 50% o que significa que a tensão de saída será a metade daquela máxima do valor do pulso PWM.

Outro fator importante é o valor da frequência da onda quadrada que será utilizada para gerar os pulsos PWM. Essa frequência deve ser fixa e não pode ser de frequência muito baixa porque pode interferir na transferência de potência ou até na qualidade do sinal final. Para a o projeto proposto necessita-se de uma frequência na ordem de kHz para controlar motores DC de pequena potência.

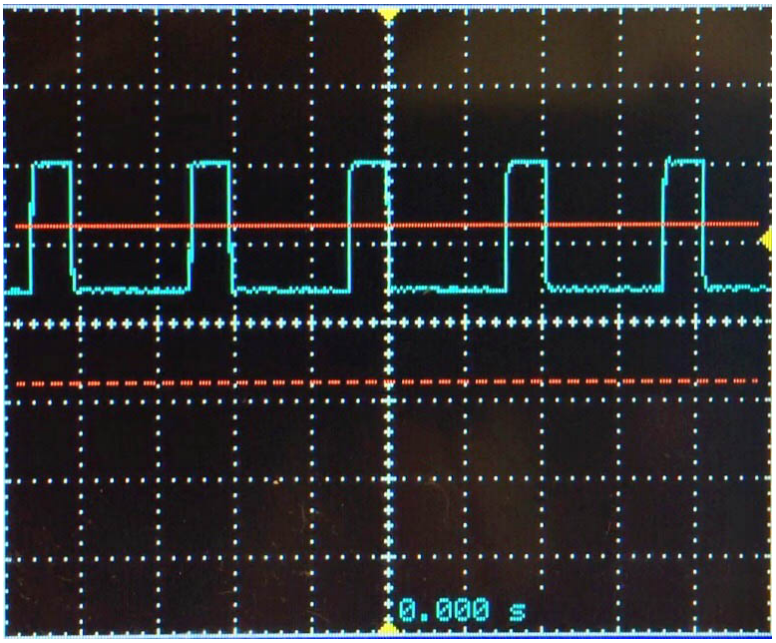
Para geral o PWM deve-se escolher uma frequência adequada para que não comprometa a transferência de potência. Uma frequência adequada para tal aplicação é 50kHz (BENNY, 2017). O algoritmo que gera os pulsos PWM necessita do clock do FPGA para funcionar que nesse caso é de 50MHz. Assim a cada 1000 ciclos do clock do FPGA o PWM completa um ciclo. Dessa forma o algoritmo divide esse tempo em forma de porcentagem de acordo com o valor recebido e o valor máximo disponível. Como escolheu-se uma modulação de 10 bits tem-se 2^{10} (Um total de 1024) valores diferentes de PWM, chagando-se a um valor total de 1024000 contagens por ciclo do PWM. O código é mostrado no Algoritmo 5. O valor de *pwmOut* no Algoritmo 5 é o valor que se quer modular no motor.

Algoritmo 5: Algoritmo gerador de PWM.

```
1 if posedge clk then
2     countPWM = countPWM + 1;
3     if countPWM <= pwmOut * 1000 then
4         pwm = 1;
5     else
6         pwm = 0;
7     if countPWM >= 1024000 then
8         countPWM = 0;
```

Ao se testar esse algoritmo obteve-se uma forma de onda característica do PWM. A Figura 19 mostra a forma de onda gerada pelo FPGA com um duty cycle de 25%.

Figura 19 – Onda PWM gerada pelo FPGA para um duty cycle menor que 50%.



Fonte: Próprio Autor.

No próximo capítulo será feito uma abordagem completa do Hardware atual e dos componentes necessários para a construção do novo robô objetivo desse trabalho.

4 DESENVOLVIMENTO DO HARDWARE

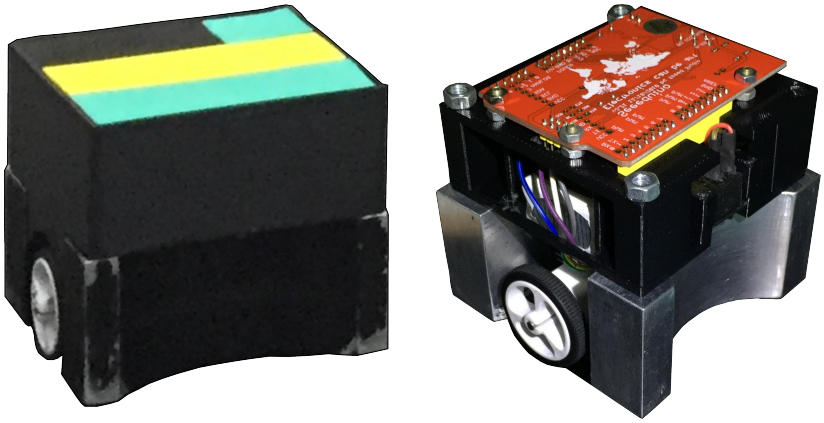
Esse capítulo é destinado à descrição das escolhas para a montagem de um robô a ser integrado em um sistema multiagente. Serão apresentadas as suas características, especificações bem como a descrição e embasamento teórico de cada dispositivos utilizado no robô. Mas antes uma breve apresentação do projeto atual será apresentada.

4.1 Hardware Atual: Características e Componentes

O Hardware atual da equipe VSSS 2016 do PMEC é caracterizado por um robô robusto com chassi de alumínio. Esse hardware é mostrado nas Figuras 20a e 20b.

Figura 20 – Robô atual do PMEC, categoria IEEE.

(a) Robô atual completo. (b) Estrutura interna do robô atual.



Fonte: Próprio Autor.

As principais características desse hardware mostrado nas Figuras 20a e 20b são:

- **Estrutura**: A estrutura foi projetada para ser a parte mais pesada do robô. É ela que comparta todas as outras partes do robô e as mantêm juntas. A estrutura tem formato arredondado em duas de suas faces (frente e trás) de modo a facilitar a posse de bola durante a partida de futebol.
- **Motores**: Os motores são motores da marca Pololu do tipo gearmotor. Esses gearmotors podem alcançar uma velocidade de até 400rpm com uma redução de 75:1 devido a caixa de engrenagem acoplada no eixo do motor.
- **Encoders**: Os encoders utilizados são encoder ópticos com uma resolução de 5 pulsos por volta do motor. O encoder é um encoder de quadratura possuindo assim dois sensores.

Esses sensores são necessários para saber quantas voltas o motor está girando por segundo e ao mesmo tempo a direção de giro do motor. Todo o processo de cálculo de velocidade e de direção é feito pelo controlador.

- **Rodas**: As rodas são de 30mm de diâmetro e são de borracha. São rodas específica e da mesma marca dos motores utilizados, o que facilita o acoplamento entre roda e motores.
- **Bateria**: A bateria é a fonte de energia do robô. Uma bateria de polímero de lítio de duas células, equivalendo a 7,4V, é a utilizada no projeto.
- **Rádio de comunicação**: O rádio de comunicação é um rádio Xbee S1 utilizado para fazer a comunicação entre robô e coordenador.
- **Ponte H**: A ponte H utilizada é o módulo HG7881 de dois canais, necessário para fazer a entrega de potência aos motores.
- **Controlador**: O controlador utilizado é um módulo derivado do Arduino Uno, a Seeeduino V4.2 da empresa Seeedstudio. Esse Arduino é destinado ao controle de todo o robô, bem como a leitura dos encoders, cálculo das velocidades dos motores, atualização do controlador PID e recebimento de mensagens via rádio comunicação.
- **Partes impressas em 3D**: Algumas partes e peças de junção foram impressas em 3D para facilitar a montagem do robô e respeita as dimensões definidas por regra da competição. Ela também, juntamente com o chassi de alumínio, ajuda a fixar todas as peças secundárias do robô.

Mais informações detalhadas podem ser encontradas do relatório técnico destina exclusivamente para esse fim (AGUAS, 2018).

4.2 Especificações dos Dispositivos do Robô a ser Projetado

Para se adequar às normas de dimensionamento dos robô multiagentes e também a algumas especificações requeridas, o novo robô deve possuir dispositivos e componentes eletrônicos que consigam atender às seguintes necessidades:

- **Dimensões**: Os robôs não podem exceder as dimensões estipuladas pelas regras que, como já foram citadas, são de 7,5cm x 7,5cm x 7,5cm. Os robôs podem ainda vestir um uniforme que pode chegar até 8,0cm x 8,0cm x 8,0cm.
- **Velocidade**: A velocidade aqui estipulada deve ser tal que os robôs possam ser controlados em um curto espaço (pista) e ao mesmo tempo dar agilidade e rapidez ao robô. Dessa

forma de acordo com as dimensões da pista, chegou-se a uma velocidade de 1,2m/s como velocidade máxima. Com essa velocidade o robô seria capaz de atravessar o campo em pouco mais de 1s.

- **Processamento**: O processamento do robô deve ser melhor que o robô estudado para fins de comparação. Dessa forma seu sistema de controle deverá sofrer melhorias significativas devido às novas abordagens.
- **Centro de massa**: O centro de massa do robô deve ser o mais rente possível ao chão e ao mesmo tempo estar centrado no eixo das rodas e centro do robô.
- **Facilidade de manutenção**: O robô deve ser de fácil montagem e desmontagem de modo a facilitar sua manutenção e troca de eventuais peças ou dispositivos danificados devido a esforços físicos do robô como choques mecânicos bem como desgastes das escovas de motores, etc.
- **Comunicação sem fio**: A comunicação entre o computador e os robôs deve ser feita através de uma comunicação estável e sem fio. Deve-se usar algum tipo de rádio transmissor e receptor.

A seguir são discutidos os dispositivos e componentes propostos para atender às especificações citadas.

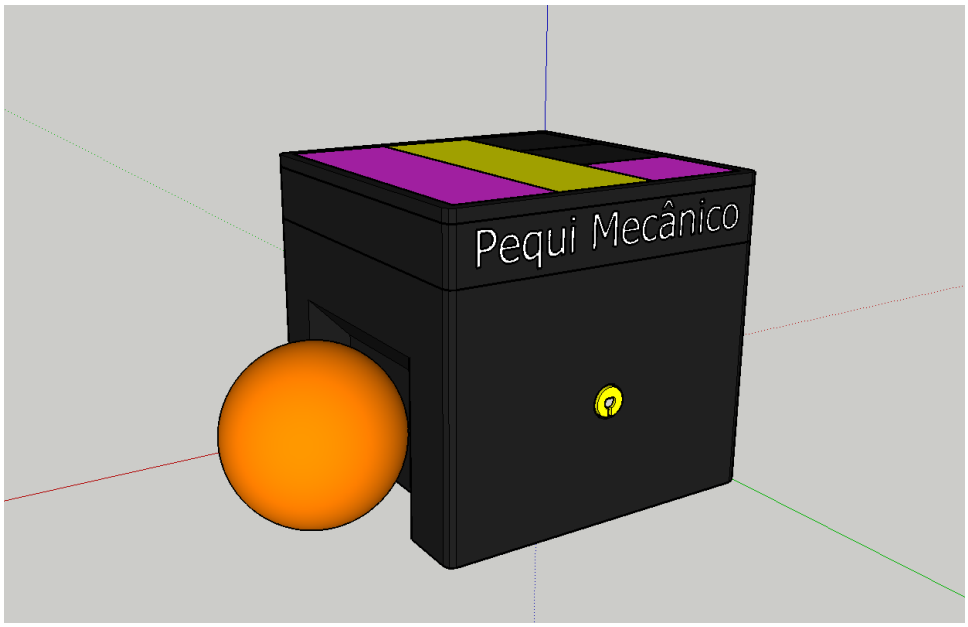
4.3 Estrutura do Robô

A estrutura do robô é destinada a suportar os contatos entre os robôs durante o jogo, e foi projetada para suportar os impactos que são inevitáveis quando se procura um mesmo objeto em comum, a bola. A estrutura do robô foi confeccionada em impressora 3D principalmente devido a facilidade de se produzir qualquer formato de peça que se queira. A estrutura também é responsável por manter todas as partes do robô fixas, tais como placa, motores e rodas, baterias e *case* das tags.

O projeto 3D do robô foi feito em ambiente CAD e é mostrado na Figura 21. Nota-se que as curvas na frente e atrás do robô são curvas destinadas a auxiliar o robô a conduzir a bola até seu objetivo, o gol do adversário. Dependendo da aceleração o robô pode conduzir a bola sem que esta escorra para as laterais, evitando assim a perda da posse de bola.

Outra característica básica é a altura da estrutura até o piso do campo. Pensando nesse quesito a estrutura não pode ser muito alta para evitar tombamento, e nem muito baixa evitando que o robô arraste no chão ao andar.

Figura 21 – Estrutura projetada em ambiente CAD do novo robô para competir na categoria IEEE Very Small Size Soccer.



Fonte: Próprio Autor.

4.4 Rodas, Motores e Encoders

As rodas e motores escolhidos para serem usados no projeto foram de tal forma que o robô pudesse ter um bom torque associado com uma boa velocidade. Motores de corrente contínua (CC) são bons, mas não possuem um torque adequadamente bom para o tamanho do motor que se deseja. Assim foram escolhidos motores com caixa de redução (os chamados gearmotors). Esses motores possuem uma velocidade relativamente alta, cerca de 400 rpm. O papel das engrenagens acopladas no eixo do motor é diminuir a velocidade de rotação do eixo acoplado às rodas e em compensação ganhar mais torque. A redução adequada para o peso e para a estrutura do robô desenvolvido é de 75,81:1, o que significa que a cada 75,81 voltas do eixo do motor a roda gira 1 volta completa. A Figura 22a mostra o motor escolhido.

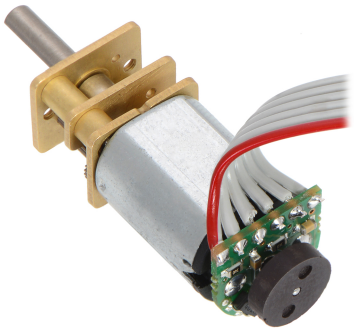
O consumo médio dos motores chega em torno de 1A de pico sendo alimentado por uma tensão de aproximadamente 6V. Como se trata de um motor bem pequeno o consumo médio é relativamente pequeno, cerca de 3-6W (POLOLU, 2017a).

As rodas são de plástico e possuem um diâmetro de 60mm. Para uma melhor aderência usa-se pneus de borracha evitando assim o deslizamento em ocasiões onde se exige um grande torque de partida. A Figura 22 mostra o kit roda-motor utilizado no projeto.

Uma outra mudança realizada foi a troca do tipo de encoder das rodas. Passou-se de um encoder óptico para um encoder magnético. A Figura 23 mostra o detalhe dos dois tipos de encoders, respectivamente. O encoder óptico possui uma resolução de 20 pulsos por volta e o encoder magnético de efeito hall possui 12 pulsos por volta de resolução. Apesar dos pulsos

Figura 22 – Conjunto roda-motor utilizado no robô

(a) Gearmotor 75,81:1 com encoder magnético.



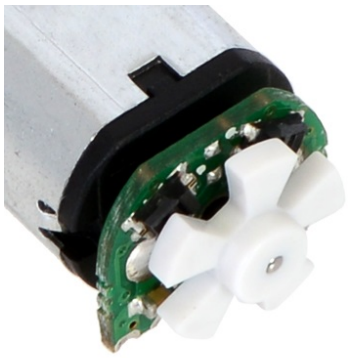
(b) Roda de 60mm de diâmetro.

Fonte: (POLOLU, 2017a) e (POLOLU, 2017e).

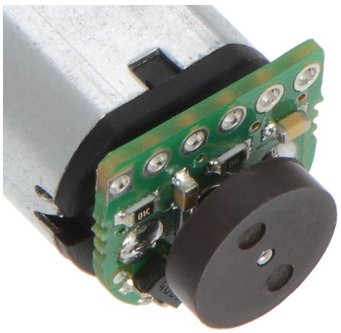
terem diminuído, a qualidade da forma de onda do encoder magnético é melhor comparado a forma de onda gerada pelo encoder óptico.

Figura 23 – Tipos de encoders.

(a) Encoder óptico.



(b) Encoder magnético.



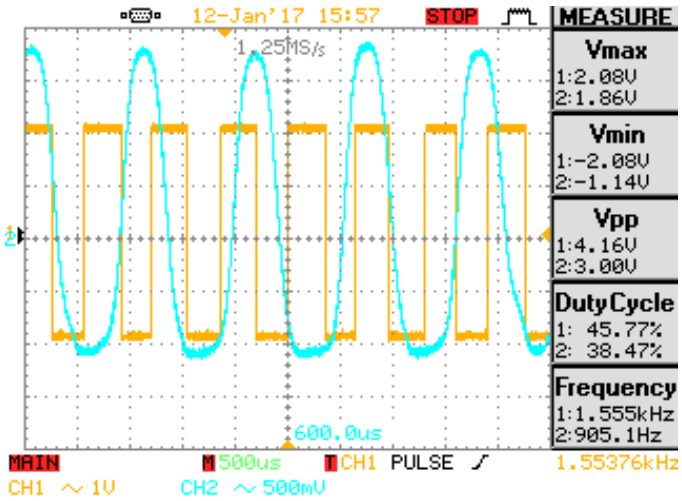
Fonte: (POLOLU, 2017d) e (POLOLU, 2017c)

Apesar de não ter melhorado a resolução do sinal, melhorou-se a forma de onda do sinal que antes era parecida com uma onda senoidal e agora é uma onda quadrada perfeita. A Figura 24 mostra a forma de onda de ambos encoders.

4.5 Controlador de Motores - Ponte H

Os controladores de motores, ou simplesmente Ponte H são circuitos integrados destinados a controlar motores, seja sua velocidade através de modulação ou até mesmo o sentido de giro do motor. Existem no mercado diversos tipos de pontes H. As principais características que as diferem são as tensões para qual foram fabricadas e as correntes de saída que alimentarão

Figura 24 – Ondas fornecidas pelos encoders magnéticos e ópticos.

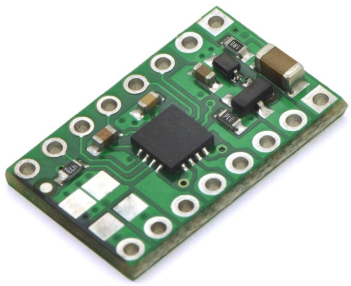


Fonte: Próprio Autor.

a carga. Para os motores aqui utilizados foi escolhida uma ponte H DRV8833 da Pololu que possui dois canais de saída (POLOLU, 2017b).

A Figura 25 mostra a ponte H utilizada. Essa ponte H consegue entregar 1,2A contínuos (com 2A pulsados) em cada um dos seus 2 canais de saída e trabalha com tensões de 2,7V a 10,8V. Outra característica importante é sua robustez. Possui dimensões de 12,7x20,32mm (POLOLU, 2017b).

Figura 25 – Ponte H DRV8833 utilizado na construção do robô.



Fonte: (POLOLU, 2017b).

4.6 Comunicação sem Fio - Xbee

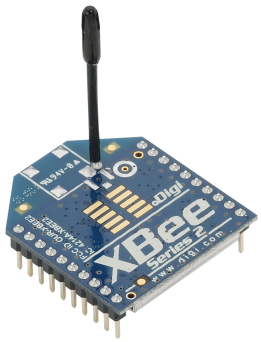
A comunicação entre robô e computador é feita utilizando módulos Xbee Série 1. Esses módulos possuem vários canais de comunicação que podem ser criptografados. A comunicação entre módulo e robô é feita por comunicação serial.

O envio de informação é bem simples e é atualizada várias vezes por segundo. Para enviar comandos ao robô um pacote contendo as informações de cada robô é criado e enviado

aos respectivos robôs de modo que somente o robô correto recebe a informação. Como o pacote de dados é criptografado, somente os módulos do respectivo time conseguem decifrar os pacotes e vice-versa. Esse tipo de comunicação evita interferência no recebimento da mensagem o que garante que a mensagem recebida será a mensagem enviada.

O emissor de pacotes ligados ao computador também é um módulo *Xbee* que funciona como módulo mestre, e os outros três robôs, cada um com um ID diferente, funcionam como escravos. Assim o fluxo de dados é sempre entre computador e robôs. O módulo Xbee é mostrado na Figura 26.

Figura 26 – Módulo *Xbee* Série 1 utilizado no projeto.



Fonte: (DIGI, 2017).

Mais informações detalhadas podem ser encontradas do relatório técnico destina exclusivamente para esse fim (AGUAS, 2018).

4.7 Controlador - FPGA

Um FPGA foi utilizado para fazer o controle do robô. Como a maioria dos FPGAs possuem muitos pinos, a construção de uma placa de circuito impresso (PCI) manualmente fica inviável. Por isso utilizou-se um pequeno módulo já pronto, mas que atende em todos os quesitos do projeto, principalmente em dimensões.

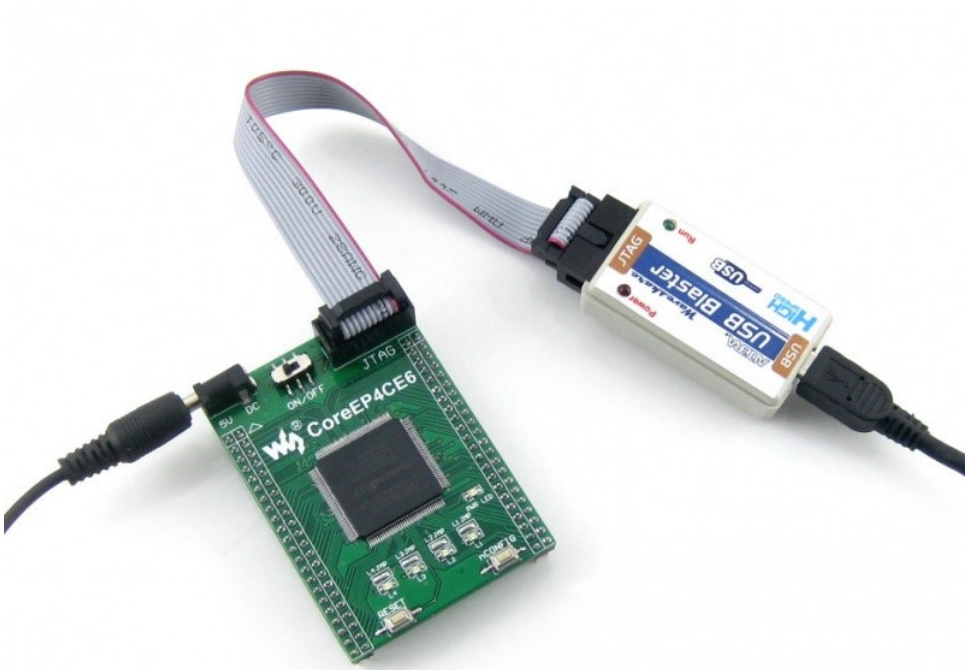
O módulo FPGA escolhido foi o CoreEP4CE6 da Wave Share (WAVESHARE, 2017). Esse módulo possui um chip FPGA da Altera EP4CE6E22C8N, 80 pinos I/Os de 2,54mm, possui dimensões iguais a 48,46x66,19mm e opera com um clock de 50MHz (BENNY, 2017).

Para programar o módulo FPGA pode-se utilizou-se de uma linguagem de programação DHL, a linguagem Verilog, descrito no Capítulo 1.

A Figura 27 mostra o módulo FPGA CoreEP4CE6. Para programa-lo é necessário um programador e uma IDE específica.

O programador utilizado é o mais convencional utilizado com FPGA, o USB-Blaster. Através do software Quartus II Web Edition (FPGA, 2018) é possível desenvolver, compilar e carregar os algoritmos no módulo FPGA.

Figura 27 – Módulo CoreEP4CE6.



Fonte: (WAVESHARE, 2017).

4.8 Fonte de Energia - Bateria

A fonte de energia mais viável atualmente são as baterias de Polímero de Lítio (LIPO). Como o robô possui uma estrutura reduzida, a bateria não pode ser muito grande e deve caber no interior da estrutura do robô. Uma bateria ideal para o tamanho do robô é a de duas células do tipo LIPO. Cada uma das células da bateria possui cerca de 3,7 Volts, totalizando 7,4 Volts. Essa fonte de energia é utilizada tanto pelos motores quanto pela parte controladora do robô.

Estudos estão sendo feitos a respeito de utilizar baterias mais leves com um ciclo de recarga menor. Baterias do tipo Li-Íon são conhecidas por possuírem tamanhos menores, serem mais leves e armazenarem uma quantidade de energia maior comparada as baterias de LIPO. A bateria utilizada nos robôs é aquela mostrada na Figura 28.

4.9 Robô montado

A estrutura do robô impressa em 3D, é mostrado na Figura 29. O robô já está com bateria, rodas, motores, encoders, sensores e placa eletrônica montada.

Figura 28 – Bateria de LIPO 2S 1000mAh 7.4V utilizada nos robôs.

Fonte: Próprio Autor.

Figura 29 – Robô montado com motores, encoders, rodas, bateria sistema de tags.

Fonte: Próprio Autor.

No próximo capítulo será abordado sobre o sistema de visão computação, necessário para o controle e planejamento de estratégias do sistema geral.

Mais informações detalhadas podem ser encontradas do relatório técnico destina exclusivamente para esse fim (AGUAS, 2018).

5 SISTEMA GLOBAL E VISÃO COMPUTACIONAL

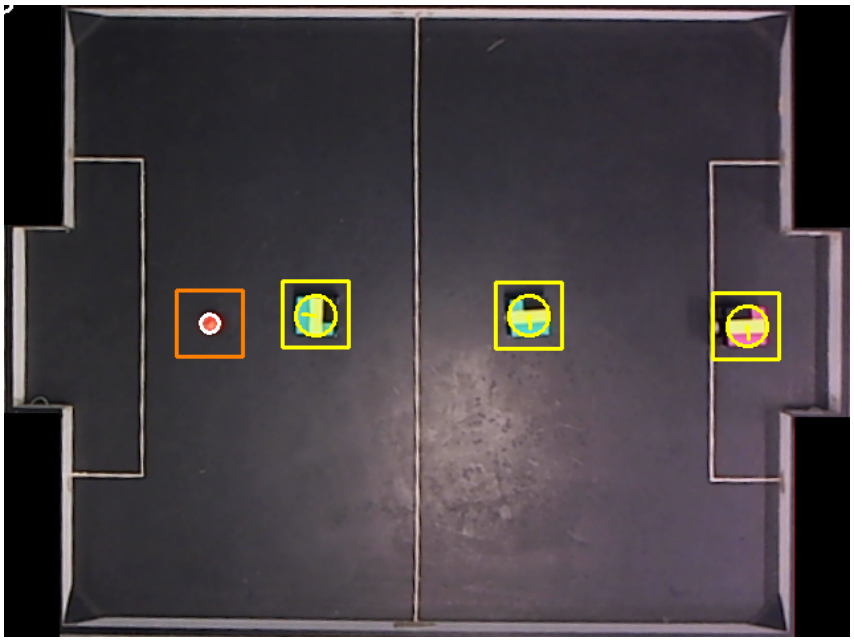
5.1 Sistema de Visão Computacional

A visão computacional é responsável por coletar os dados sobre o jogo em cada instante de tempo e passá-las de forma mais simplificada possível para o algoritmo responsável pela estratégia. Este processo é dividido nas seguintes etapas: inicialmente a câmera, colocada perpendicularmente a aproximadamente 2 metros de distância do campo, faz uma série de captura de frames e os envia para o computador. Um frame é uma imagem individual enviada pela câmera. A partir daí, é utilizado um algoritmo baseado na biblioteca OpenCV (OPENCV, 2017) para o processamento dessas imagens. A imagem é tratada por filtros que retiram partes da imagem que não são necessárias, deixando apenas o que é realmente importante: as tags de cores, que identificam os robôs, e a bola. Após o processamento da imagem para a detecção das tags de cores e da bola, um algoritmo de rastreamento é utilizado para encontrar, através de histogramas de cor, a posição absoluta de cada robô e da bola no campo.

Após todos os objetos serem encontrados, o algoritmo determina janelas de 50x50 pixels centralizadas na posição de cada objeto para que, nos próximos frames, a detecção de cores ocorra apenas dentro das janelas, ao invés de percorrer a imagem inteira. Isso faz com que o custo computacional seja reduzido drasticamente. Processar a imagem completa cria um descompasso entre o processamento da visão e o processamento das funções do robô. Processar apenas as janelas locais de cada objeto ajuda a diminuir esse descompasso de processamento. As janelas de cada objeto são deslocadas com o auxílio de um algoritmo de Filtro de Kalman (KALMAN, 1960). O filtro é responsável por estimar de forma eficiente qual será a posição da bola e dos robôs nos próximos frames para que cada janela tenha sua posição ajustada previamente, o que aumenta a probabilidade dos objetos rastreados estarem dentro da janela de processamento no próximo frame. Se algum robô ou a bola não for encontrada dentro de sua janela de processamento, o algoritmo terá que processar a imagem inteira para reencontrá-lo. Assim, a cada 33 milissegundos, é possível identificar a posição dos robôs e da bola, que são guardadas em memória e enviadas ao algoritmo de estratégia, que determina a ação de cada robô naquele instante de tempo. A Figura 30 mostra uma frame processado com janelas de processamento ao redor dos 3 robôs que compõe um time.

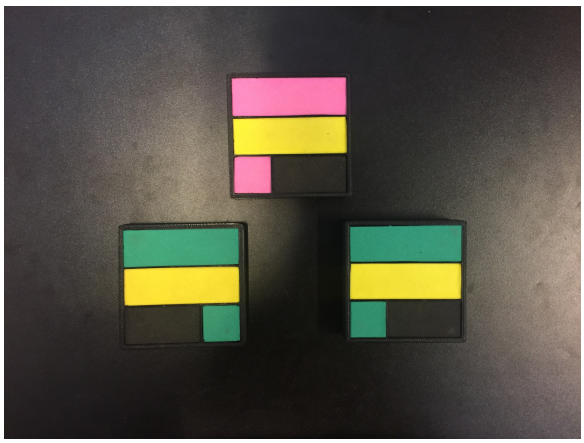
O sistema de tags de cores, utilizado para a detecção dos robôs no campo e mostrado na Figura 31. A posição do robô é identificada pelo centro da tag principal, pois ambos são concêntricos. Cada robô possui duas tags secundárias, de mesma cor, sendo que a tag secundária de maior tamanho indica a frente do robô. Como pode ser visto na Figura 31, dois robôs compartilham as mesmas cores. A posição da tag secundária de menor tamanho é responsável por diferenciar esses robôs de mesmas cores de tags.

Figura 30 – Imagem capturada pela câmera. Os quadrados indicam as janelas de processamento de cada objeto no campo.



Fonte: Próprio Autor.

Figura 31 – Novo sistema de tags dos robôs.



Fonte: Próprio Autor.

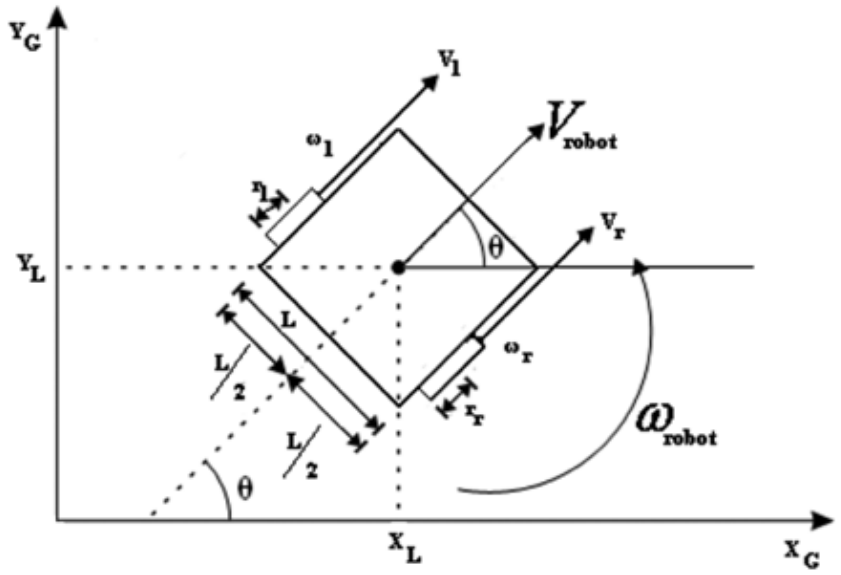
Até 2016, eram usados três cores secundárias diferentes para diferenciar os três robôs. Com o uso de apenas duas cores, ou seja, dois robôs terão tags de cores repetidas, diminui-se o custo computacional do algoritmo de visão, já que se elimina a segmentação de uma cor. Com menos cores para serem identificadas, possibilita o uso de cores secundárias que são mais distantes no espectro visível de cor. Isso diminui consideravelmente a chance de cores escolhidas sejam confundidas ao serem filtradas pelo algoritmo de visão.

5.2 Estratégia de Jogo e Técnicas de Controle

A estratégia, assim como em um jogo de futebol real, deve avaliar a situação do jogo e tomar decisões que possam otimizar o comportamento dos jogadores (agentes). O papel do módulo de estratégia é analisar os dados coletados pela visão computacional e decidir quais serão as ações a serem tomadas por cada robô no próximo instante de tempo. As estratégias normalmente utilizadas no futebol de robôs seguem uma estrutura fixa, compostas normalmente de um sistema supervisor que recebe informações do algoritmo de visão computacional, decide qual ação os robôs devem fazer e, por fim, repassa essa ação para um sistema de controle (SILAS et al., 2009) (MARTINOVIC et al., 2010). Esse sistema é confiável, mas apresenta uma performance não tão fluida, transformando as ações dos jogadores em jogadas pré-moldadas pelos criadores da estratégia.

Para fundir a camada de decisão com a camada de controle são empregadas técnicas computacionais com o mínimo custo computacional e com moderada flexibilidade de aplicação. A solução encontrada para relacionar o processo de decisão tática, com o controle do robô, foi a introdução do modelo diferencial que exprime a reação do robô ao aplicar determinadas velocidades em suas rodas. Este modelo pode ser definido obtendo a velocidade angular de orientação do robô e a velocidade linear de translação do robô, conforme representado na Figura 32.

Figura 32 – Modelo diferencial aplicado ao robô.



Fonte: (SILAS et al., 2009).

5.2.1 Estratégia: Robô Defensor

O robô defensor é responsável por evitar que o adversário marque um gol. O simples comportamento de reagir aos movimentos da bola garante uma relativa segurança na defesa,

mas é recomendado atribuir algumas tarefas mais complexas ao goleiro, tais como antecipar a posição futura da bola e reagir aos movimentos do jogador adversário. Estratégias como seguir a coordenada ordenada da bola fornecem uma certa garantia de defesa, e esta estratégia é vastamente utilizada em vários outros trabalhos. Porém, em alguns casos, movimentos mais complexos desempenhariam melhor a função de defesa e possuem grande potencial de desencadear um contra ataque. Para ponderar tais possibilidades, a função de avaliação das soluções necessita analisar o comportamento da bola durante a partida. Isto é, são necessários parâmetros de velocidade e posição a fim de obter a trajetória da bola. Além disso, parâmetros como posição dos adversários podem fornecer informações para que o robô possa antecipar uma bola que esteja em direção ao gol. Com a trajetória da bola o robô poderá realizar movimentos para acompanhar o movimento dela e, por fim, interceptá-la em um ponto externo ao gol. Nesta abordagem, são evoluídas as velocidades das rodas do robô defensor a fim de obter a melhor combinação que obedeça a função de avaliação descrita. Inicialmente as velocidades são evoluídas seguindo o modelo diferencial ideal do robô. Durante a execução de cada solução, é realizada a realimentação do sistema e o modelo diferencial é corrigido. Dessa forma, a função de avaliação torna-se mais precisa ao longo do tempo mesmo que ocorra mudanças no sistema físico do robô, como queda na tensão de alimentação ou imprevistos mecânicos.

5.2.2 Estratégia: Robô Atacante

Os robôs atacantes devem trabalhar juntos a fim de manter a posse de bola e, eventualmente, marcar o gol. Para executar esta tarefa os atacantes devem se posicionar de forma inteligente e, quando com a posse de bola, devem evitar investidas do adversário para chegar ao gol. Para simplificar a execução do desafio, usa-se uma solução que minimiza o custo computacional e que, ao mesmo tempo, da flexibilidade às ações possíveis dos jogadores. A estrutura do algoritmo dos atacantes se assemelha à usada no defensor, se diferenciando somente na função de seleção. Para a concepção da função seletiva dos atacantes existe uma área limitada nas proximidades do jogador que gera uma base para as ações e reações com o ambiente. Nesse raio de visão, o jogador reage aos objetos dependendo da sua natureza. A bola possui um campo espiral que converge para uma posição atrás da mesma, este campo tem o intuito de melhorar o controle e a posse de bola dos atacantes. Já os robôs adversários produzem um campo de repulsão circular, planejado para facilitar situações de drible (ALVES et al., 2011).

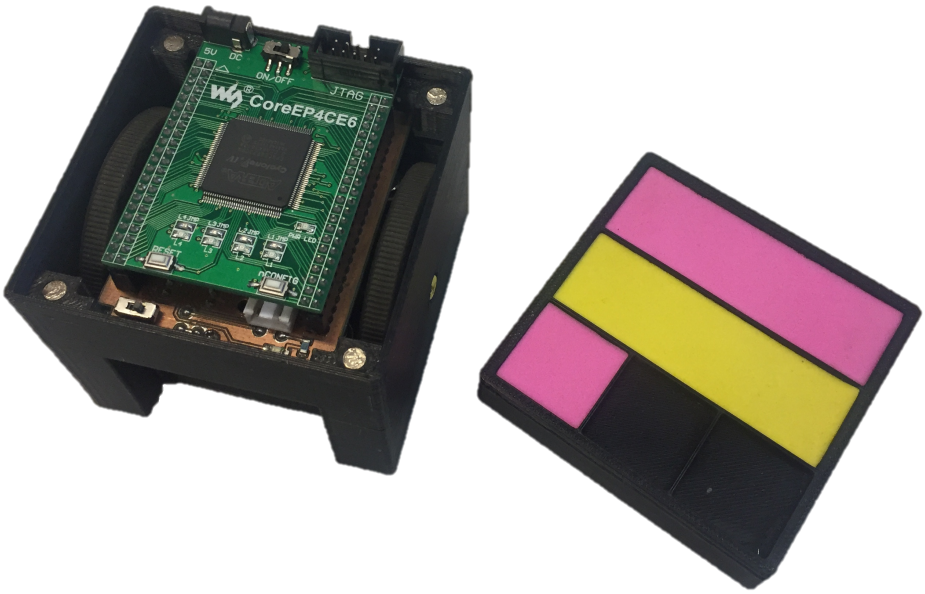
6 RESULTADOS FINAIS

Esse Capítulo é destinado a mostrar os resultados alcançados. Os resultados foram organizados de maneira sucinta para melhor apresentação dos dados.

6.1 Montagem do Robô

O robô foi montado de acordo com o projeto proposto e com os componentes descritos anteriormente. A Figura 33 Mostra o projeto final do robô a ser utilizado durante os testes de controle. O robô foi totalmente desenvolvido para esse projeto.

Figura 33 – Robô desenvolvido para o projeto proposto.



Fonte: Próprio Autor.

6.2 Sintonia do controlador PID com AGc

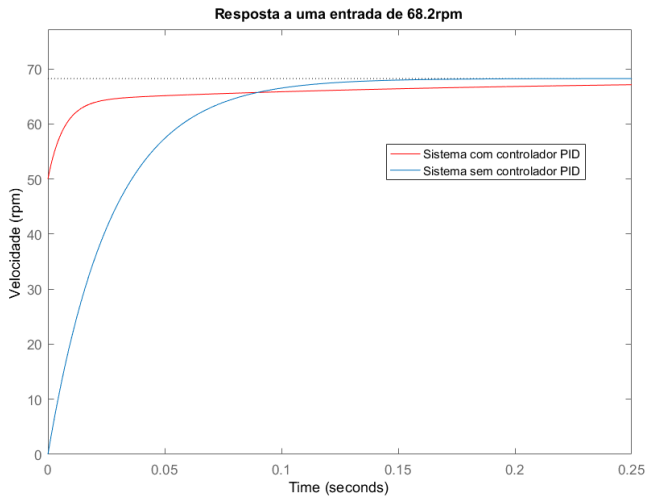
A análise dos resultados foi feita da seguinte forma: Primeiro escolheu-se o tamanho da população "$pop\_size = 50;$" e a quantidade de interação "$max\_interations = 100;$" para se obter um resultado final com resposta a uma entrada de 70rpm para que assim seja comparada a resposta da Figura 11. Nesse caso deseja-se obter uma resposta mais rápida e com um erro de estado estacionário menor. Depois será mostrado como a função converge de acordo com o tamanho da população e da quantidade de interações em alguns casos. Em seguida foi analisada a reação da função de transferência para um pulso de ondas quadradas de diferentes amplitudes.

Mais informações detalhadas podem ser encontradas do relatório técnico destina exclusivamente para esse fim (AGUAS, 2018).

6.2.1 Analise da resposta para o tamanho da população 50 e interações igual a 100

Durante a fase de execução do Algoritmo 1 obteve-se a resposta da Figura 34 já plotada com o valor não compensado. O valor da resposta a uma entrada de 70rpm foi escolhido para fins de comparação com a resposta a degrau unitário ao sistema do motor não compensado na Figura 11. Na Figura 34 tem-se que o valor desejado para a controlador foi alcançado. Note que o tempo de estabilização foi menor do que o requerido. As constantes geradas pelo AGc foi: $k_p = 0.2172,\ k_i = 1.1853\ e\ k_d = 0.0011$.

Figura 34 – Resposta a uma velocidade de entrada de 68.2rpm.



Fonte: Próprio Autor.

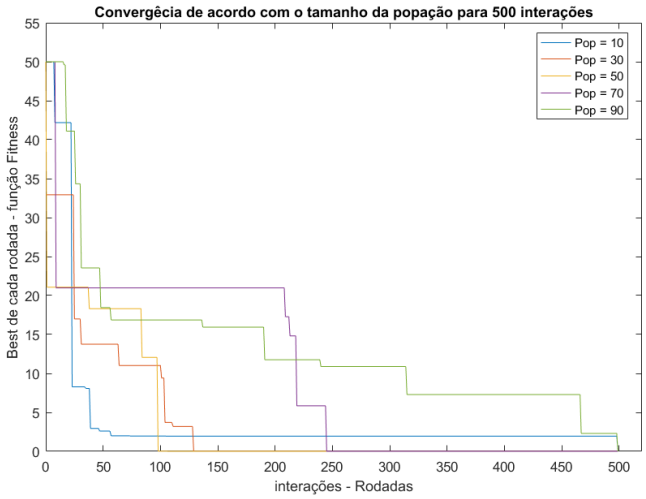
Nota-se ainda na Figura 34 que a resposta é bem aceitável e que o emprego do AGc atingiu seu objetivo em gerar as constantes para o controlador PID.

6.2.2 Análise da convergência para 500 interações

Variando o tamanho da população é possível ter mais seletividade em questão de atualização da probabilidade do vetor de probabilidade. Assim espera-se que quanto maior a população menos sensível a mudanças ou mais suave será a mudança e consequentemente uma convergência mais demorada porém mais precisa. Em uma análise oposta a apresentada, quanto menor a população mais rápido o vetor de probabilidade converge mas com menos precisão. A Figura 35 mostra como o vetor vai convergindo de acordo com o tamanho da população. A quantidade de interações foi mantida em 500 como forma de parada do algoritmo de modo que

se o algoritmo não convergir em até 500 interações o algoritmo retorna as constantes parciais do teste.

Figura 35 – Variação da convergência de acordo com a variação da população para 500 interações.



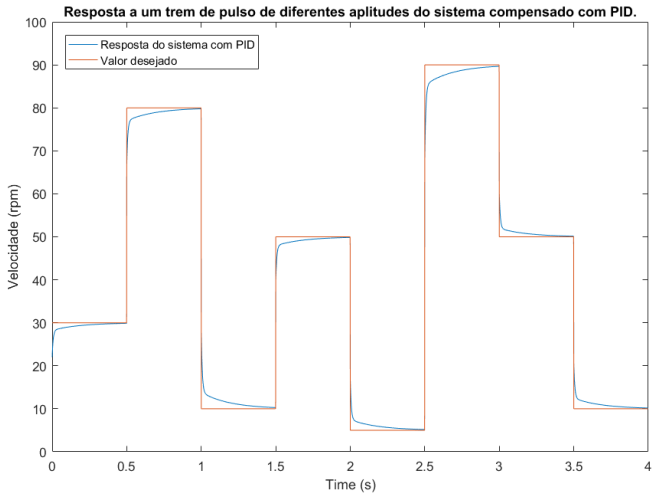
Fonte: Próprio Autor.

Nota-se pela análise da Figura 35 que para uma população de 50 indivíduos a convergência aconteceu mais rápida. Como se trata de uma análise probabilística isso pode acontecer pois existe variações no percorrer das interações. Dessa forma ainda analisando a Figura 35 pode-se notar que para uma população de 10 indivíduos a convergência aconteceu muito rápida pois a variação do vetor de probabilidade de acordo com a fase 4 mostrada no Algoritmo 1 muda de acordo com o tamanho da população. Por outro lado quando a população é maior, no caso de 90 indivíduos a mudança e convergência é mais sutil e mais precisa.

6.2.3 Resposta do sistema compensado pelo PID a um trem de pulso aleatório

A última análise aqui proposta é a resposta a um trem de pulso de velocidades, que é o que acontece na prática. Em sistemas embarcados principalmente robótica móvel, onde se emprega o uso de motores DC para sua locomoção, o controlador principal envia informações aos controlador dos motores que aplica tensões e correntes desejadas nos motores. Como o controlador envia diferentes valores de velocidades ao longo do tempo o sistema tem que responder rápido a esse trem de pulsos. Um trem de pulso aleatório foi aplicado ao sistema de modo a obter a resposta do mesmo. A Figura 36 mostra o valor desejado do trem de pulsos e a resposta do sistema. Nota-se que em menos de 200ms o sistema já se recupera totalmente para o novo valor.

Figura 36 – Trem de pulso aleatório aplicado ao sistema compensado com PID.



Fonte: Próprio Autor.

6.3 Sistema de Controle: Respostas Obtidas

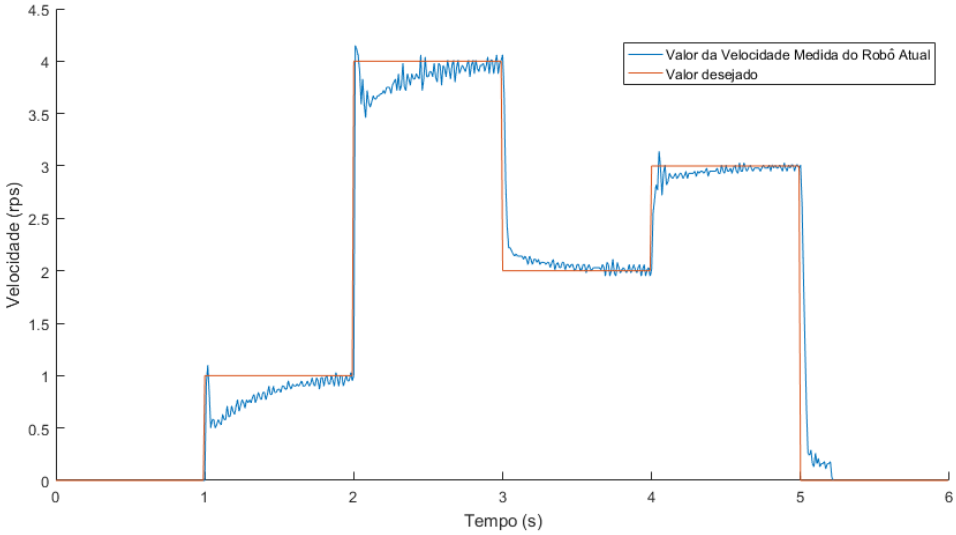
Os resultados que serão apresentados a seguir são resultados experimentais obtidos dos robôs a serem comparados: a) o Robô Atual (apresentado na Secção 4.1) e b) Robô Novo (robô projetado nesse trabalho e apresentado na Secção 4.2). Assim será adotado para melhor apresentação dos resultados como 'Robô Atual' sendo o robô a ser melhorado e 'Robô Novo' ou 'Robô Projetado' sendo o robô projetado no âmbito desse trabalho.

6.4 Resposta do sistema de controle: Robô Atual

O Robô Atual apresentado na Secção 4.1 foi submetido a uma sequência de pulsos de diferentes valores. O objetivo desse teste é obter a resposta do sistema de controle do robô (Controlador PID) com relação a velocidade atual do robô. A função de transferência desse robô é igual a mostrada e deduzida na Sessão 3.1. As constantes do controlador PID foram obtidas empiricamente e foram: $k_p = 1.2600$ (Constante Proporcional responsável por um tempo de subida menor), $k_i = 0.0481$ (Constante Integral responsável por amenizar os pequenos erros de estado estacionário) e $k_d = 0.0000$ (Constante Derivativa responsável por uma estabilização mais rápida e diminuição de overshoot). Nesse sentido aplicou-se uma sequência de pulsos específica a qual pode ser observada na Figura 37. Nessa mesma Figura pode-se observar ainda a resposta do sistema obtida em tempo real através do envio do trem de pulsos via cominuição sem fio para o robô.

Nota-se que de acordo com a Figura 37 o sistema apresenta dificuldades em atingir a velocidade desejada. Isso se deve principalmente a mau calibração do controlador PID com relação a escolha das constantes corretas. O sistema consegue estabilizar mas o tempo de esta-

Figura 37 – Trem de pulso aplicado ao sistema de controle PID do Robô Atual.



Fonte: Próprio Autor.

bilização desejado (apresentado na Secção 3.3) não é atingido, chegando próximo de 1s. Esse comportamento do sistema influencia diretamente na movimentação do robô visto que o sistema de atualização das velocidade das rodas do robô é mais rápido do que a própria estabilização do sistema. Isso significa que o robô não consegue a curto prazo atender a demanda imposta pelo técnico (computador geral). O resultado desse atraso de estabilização faz com que a velocidade do robô oscile, tendo como consequência um movimento de tranco nos motores do robô.

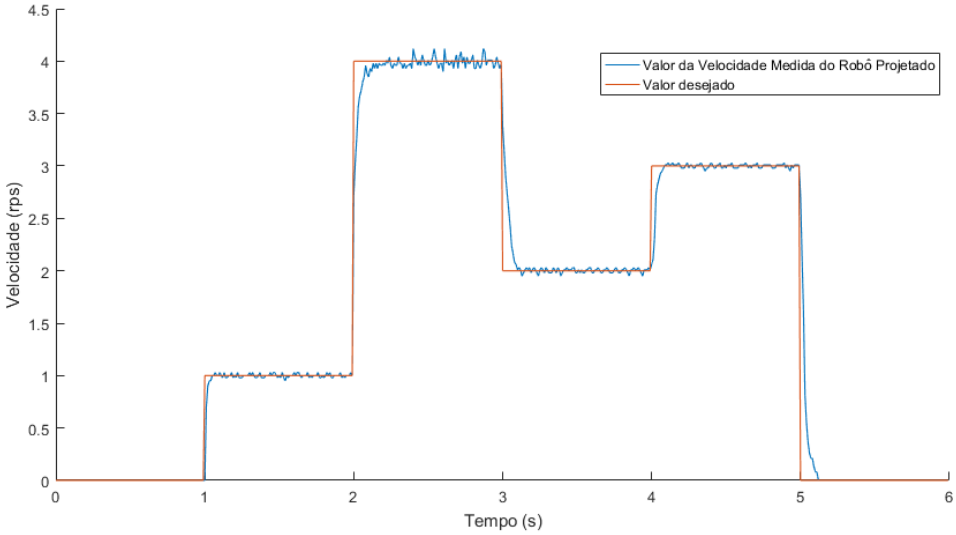
Para fazer então a melhoria desse sistema utilizou-se das metodologias aqui apresentadas para encontrar as constantes ideais do controlador PID que faço com que os requisitos desejados do projeto sejam atendidos.

6.5 Resposta do sistema de controle: Robô Projetado

O sistema de controle do Robô Projetado, apresentado na Secção 4.2, foi desenvolvido de acordo com a metodologia da Secção 3.2. Utilizando-se então dos algorítimos e das constantes $k_p = 0.2172$ (Constante Proporcional responsável por um tempo de subida menor), $k_i = 1.1853$ (Constante Integral responsável por amenizar os pequenos erros de estado estacionário) e $k_d = 0.0011$ (Constante Derivativa responsável por uma estabilização mais rápida e diminuição de overshoot) apresentadas na Secção 6.2.1 pode-se obter a resposta do sistema PID do Robô Novo. A resposta a um trem de pulsos específico pode ser observado na Figura 38.

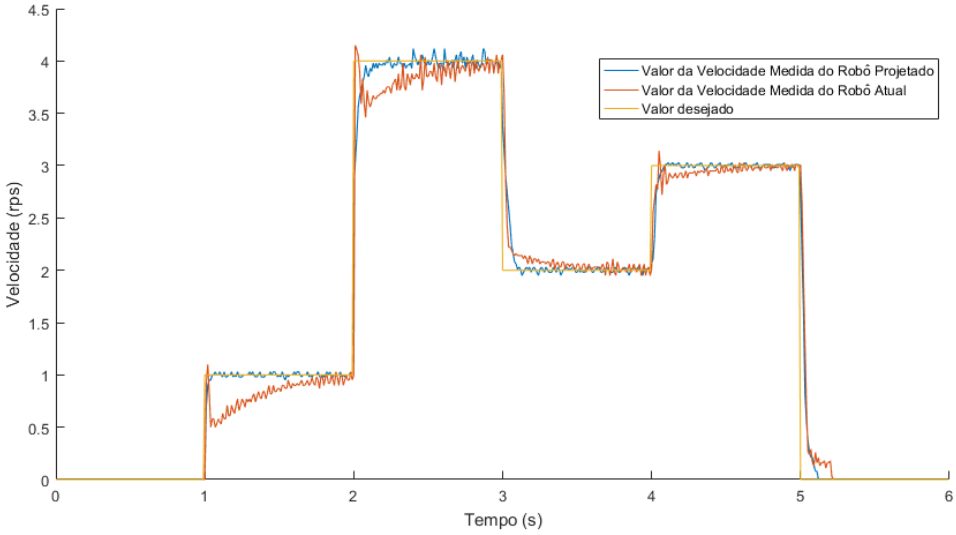
Nota-se claramente que o sistema se comportou de maneira diferente daquele obtido com o Robô Atual (Figura 37). Para uma melhor análise plotou-se a resposta dos dois sistemas, Atual e Projetado, junto com o valor desejado. O resultado é mostrado na Figura 39.

Figura 38 – Trem de pulso aplicado ao sistema de controle PID do Robô Projetado.



Fonte: Próprio Autor.

Figura 39 – Trem de pulso aplicado ao sistema de controle PID do Robô Projetado e Robô Atual.

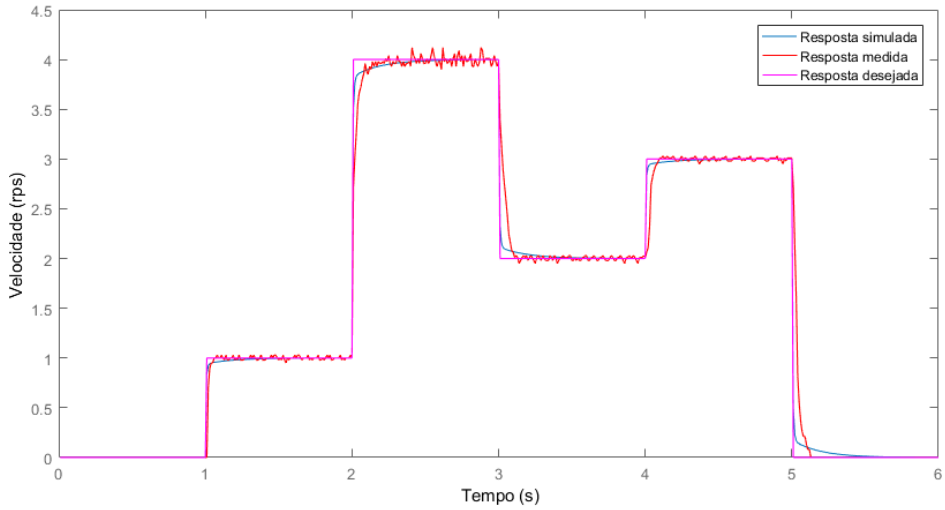


Fonte: Próprio Autor.

A melhora no sistema de estabilização foi a esperada no que diz respeito as especificações desejadas. A melhoria se enquadra não em valor quantitativo mas sim em valor qualitativo. A curva da resposta do sistema de controle do Robô Novo se comportou de maneira esperada a aquela obtida na simulação da Figura 36. Para uma melhor comparação tem-se a Figura 40.

A Figura 40 mostra a curva obtida com o Robô Novo bem próxima a curva de projeto obtida via simulação da equação de transferência do robô. Por outro lado a resposta do Robô

Figura 40 – Trem de pulso aplicado ao sistema de controle PID do Robô Projetado, Robô Atual e Simulado.

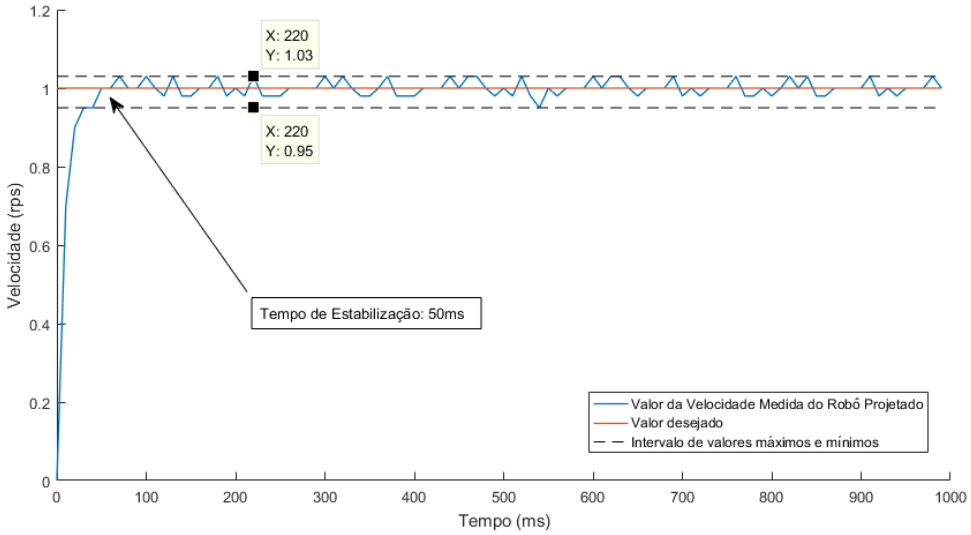


Fonte: Próprio Autor.

Atual fica aquém ao se compara com os resultados do novo sistema.

Para uma melhor análise dos requisitos do projeto o Robô novo foi submetido a uma entrada degrau unitário, que nesse contexto equivale a 1rps. O resultado é mostrado na Figura 41.

Figura 41 – Resposta do sistema do Robô Novo a um Degrau unitário (1rps)



Fonte: Próprio Autor

Fazendo uma análise da Figura 41 nota-se que não ocorreu *overshoot*, que é quando o sinal ultrapassa o valor desejado. Como o valor desejado no projeto era menor que 1%, esse re-

quisito foi atendido. O segundo requisito foi o tempo de estabilização menor que 100ms. Como mostrado na Figura 41 o tempo de estabilização foi aproximadamente 50ms, o que também está de acordo com os requisitos do projeto. Por último esperava-se que os sistema tivesse um erro de estado estacionário menor que 2%. Nesse caso os respectivos valores máximos e mínimos esperados para o estado estacionário é de 0.98rps e 1.02rps. Entretanto os valores mostrados na Figura 41 indicam que os valores máximos e mínimos foram atendidos em alguns pontos. Como se trata de um sistema dinâmico isso pode acontecer devido às próprias limitações elétricas do motor. Por outro lado destaca-se que o valor máximo encontrado de 1.03rps e o valor mínimo encontrado de 0.95rps aparecem somente em alguns pontos, tornando aceitável o resultado obtido.

7 CONCLUSÕES

Esse trabalho de dissertação foi destinado ao desenvolvimento de um protótipo robótico funcional projetado e desenvolvido computacionalmente com o objetivo de obter resultados melhores do que os apresentados nos protótipos atuais. Esses resultados incluíram também todos os passos necessários para que o controlador PID, mais especificamente as constantes PID, fosse calibrado de maneira eficiente através de técnicas computacionais modernas.

O emprego de AGc para modelagem de um sistema capaz de encontrar as constantes de um controlador para que o comportamento do motor seja aceitável foi de grande valia visto que esse processo é tortuoso e demorado se não utilizado ferramentas corretas. Dessa forma o emprego do AGc teve uma importância crucial e rápida para a sintonia correta do controlador PID.

A projeto do novo hardware, inspirado na versão atual dos robôs da equipe PMEC, foi desenvolvido utilizando conhecimentos prévios de modelagem 3D e montagem mecânica. Essas experiências foram importantes para o sucesso da construção da nova estrutura e principalmente conseguir embarcar um módulo FPGA em um robô com dimensões tão reduzidas. Dessa forma o projeto pôde ser finalizado de acordo com o planejamento inicial.

O desempenho do robô novo comparado com os robôs atuais é superior, como foi mostrado durante a apresentação dos resultados nessa dissertação. Tanto a estrutura quando o sistema de controle ficaram mais robustos.

E o projeto ainda possui muitas outras linhas de desenvolvimento que podem ser melhoradas. Essas melhorias não irão parar somente nessas apresentadas nesse trabalho. Outras melhorias podem ser realizadas, as quais podem ser enumeradas:

- Desenvolvimento do time completo (3 robôs) para obter testes do sistema de controle com mais precisão;
- Utilização de sensores inerciais para melhoria da navegação do robô;
- Fusão sensorial da parte ondométrica com a parte de sensores inerciais;
- Confecção de uma placa própria e específica para o projeto com mais recursos facilitando o processo de testes e aferição de resultados;
- Utilização de motores mais robustos e que tenham melhores resposta com relação ao controlador adotado, tais como motores de empresas renomadas como Faulhaber e Maxon;
- Utilização de sistemas híbridos de controlador, utilizando FPGA e microcontroladores, aproveitando as vantagens de cada um desses dispositivos.

O futebol de robôs multiagentes é um projeto desenvolvido na EMC-UFG pelo PMEC desde 2013 e de lá pra cá todo o sistema passa por melhorias constantes. Dessa forma o projeto não vai parar por ai e sempre irá ser aprimorado ano após ano.

REFERÊNCIAS

AGUAS, W. G. *Descrição e desenvolvimento de Multiagentes Jogadores de Futebol da categoria IEEE Very Small Size Soccer*. [S.l.], 2018. Disponível em: <https://sophia.bc.ufg.br/>. Acesso em: 07 out. 2018. Citado 4 vezes nas páginas 55, 60, 62 e 68.

ALTERA, P. *DE1 - Development and Education Board, User Manual*. [S.l.], 2012. Disponível em: <https://www.altera.com>. Acesso em: 16 jul. 2018. Citado na página 19.

ALVES, S. F. et al. Conceptual bases of robot navigation modeling, control and applications. In: *Advances in Robot Navigation*. InTech, 2011. Disponível em: <http://cdn.intechopen.com/pdfs/16161/InTech-Conceptual_bases_of_robot_navigation_modeling_control_and_applications.pdf>. Acesso em: 16 jul. 2018. Citado na página 66.

ANDREA, R. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, v. 9, n. 4, p. 638–639, 2012. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6295687>. Acesso em: 16 jul. 2018. Citado na página 30.

ARDUINO, P. *What is Arduino?* [S.l.], 2018. Disponível em: <https://www.arduino.cc/en/Guide/Introduction#>. Acesso em: 16 jul. 2018. Citado na página 30.

ARDUINO, S. *Arduino Uno REV3*. [S.l.], 2018. Disponível em: <https://store.arduino.cc/arduino-uno-rev3>. Acesso em: 16 jul. 2018. Citado na página 32.

BENNY, P. J. *Choosing the right PWM frequency*. [S.l.], 2017. Disponível em: <http://www.seattlerobotics.org/encoder/200011/pwm.html>. Acesso em: 16 jul. 2018. Citado 2 vezes nas páginas 52 e 60.

BRINKMAN, W. F.; HAGGAN, D. E.; TROUTMAN, W. W. A history of the invention of the transistor and where it will lead us. *IEEE Journal of Solid-State Circuits*, IEEE, v. 32, n. 12, p. 1858–1865, 1997. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=643644>. Acesso em: 16 jul. 2018. Citado na página 18.

CASSEB, M. V. G. *Algoritmos Genéticos Compactos Aplicados à Estimação Fasorial em Tempo Real*. 85 f. Monografia (Monografia) — Universidade de São Paulo, Escola de Engenharia de São Carlos, São Carlos, 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18154/tde-05102012-091637/en.php>. Acesso em: 16 jul. 2018. Citado 2 vezes nas páginas 37 e 38.

DIGI, P. *Digi Xbee S1 Series 802.15.4*. [S.l.], 2017. Disponível em: <https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-802-15-4>. Acesso em: 16 jul. 2018. Citado na página 60.

FIRA, P. *FIRA MiroSot Game Rules*. [S.l.], 2017. Disponível em: <http://www.inovamicro.com/support/MiroSot_Rules.pdf>. Acesso em: 16 jul. 2018. Citado na página 26.

FIRA, P. *Objectives*. [S.l.], 2017. Disponível em: <http://www.firaworldcup.org/VisitorPages/show.aspx?IsDetailList=true&ItemID=889,1>. Acesso em: 16 jul. 2018. Citado 4 vezes nas páginas 18, 23, 24 e 25.

FPGA, P. I. *Quartus II Web Edition*. [S.l.], 2018. Disponível em: <https://www.altera.com/downloads/software/quartus-ii-we/91sp2.html>. Acesso em: 16 jul. 2018. Citado na página 61.

HAN, K.-H. et al. Robot soccer system of soty 5 for middle league mirosot. In: *Proceeding of FIRA Robots World Congress*. [s.n.], 2002. p. 251–254. Disponível em: <http://www.khhan.com/pdf/Soty5FIRA.pdf>. Acesso em: 16 jul. 2018. Citado na página 24.

HARIK, G. R.; LOBO, F. G.; GOLDBERG, D. E. The compact genetic algorithm. *Transaction on Evolutionary Computation*, p. vol. 3, 1999. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=797971>. Acesso em: 16 jul. 2018. Citado 3 vezes nas páginas 37, 38 e 39.

JABSON, N. G. et al. The autonomous golf playing micro robot: with global vision and fuzzy logic controller. *International Journal on Smart Sensing and Intelligent Systems*, v. 8, p. 12, 1 2008. Disponível em: <http://www-ist.massey.ac.nz/s2is/Issues/v1/n4/papers/paper1.pdf>. Acesso em: 16 jul. 2018. Citado na página 22.

JALILVAND, A. et al. Optimal tuning of pid controller parameters on a dc motor based on advanced particle swarm optimization algorithm. *Internacional Journal on Technical and Physical Problems of Engineering (IJTPE)*, v. 11, p. 10–17, 2011. Disponível em: <https://pdfs.semanticscholar.org/b428/abb53436375e77339f54a3bfdc81a55a00ee.pdf>. Acesso em: 16 jul. 2018. Citado na página 47.

JOGALEKAR, K. et al. Implementation of pid architecture in fpga for dc motor speed control. *Circuits, Controls and Communications (CCUBE), 2013 International conference on*, p. 1–5, 2013. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6718557>. Acesso em: 16 jul. 2018. Citado 2 vezes nas páginas 36 e 44.

JORDAN, D. *Pulse-width Modulation*. [S.l.], 2017. Disponível em: <https://learn.sparkfun.com/tutorials/pulse-width-modulation>. Acesso em: 16 jul. 2018. Citado na página 51.

KALMAN, R. E. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, American Society of Mechanical Engineers, v. 82, n. 1, p. 35–45, 1960. Disponível em: <https://www.cs.unc.edu/~welch/kalman/media/pdf/Kalman1960.pdf>. Acesso em: 16 jul. 2018. Citado na página 63.

LAMáR, K.; KOCSIS, A. G. Implementation of speed measurement for electrical drives equipped with quadrature encoder in labview fpga. *Acta Technica Corviniensis-Bulletin of Engineering*, Faculty of Engineering Hunedoara, v. 6, n. 4, p. 123, 2013. Disponível em: <http://acta.fih.upt.ro/pdf/2013-4/ACTA-2013-4-22.pdf>. Acesso em: 16 jul. 2018. Citado na página 48.

LI, C. et al. The real-time and embedded soccer robot control system. In: *Robot Soccer*. In-Tech, 2010. Disponível em: <https://www.softbankrobotics.com/emea/en/robots/nao>. Acesso em: 16 jul. 2018. Citado 2 vezes nas páginas 27 e 28.

MARTINOVIC, J. et al. Robot soccer-strategy description and game analysis. In: *ECMS*. [s.n.], 2010. p. 265–270. Disponível em: <https://www.researchgate.net/publication/265105421_Robot_Soccer_-Strategy_Description_and_Game_Analysis>. Acesso em: 16 jul. 2018. Citado na página 65.

MARTINS, B. B. S. et al. Tdp equipe pequi mecânico vsss - futebol de robôs - categoria verysmall. *Competição Brasileira de Robótica, Recife*, 2016. Disponível em: <http://www.cbrobotica.org/mostravirtual/interna.php?id=14184>. Acesso em: 16 jul. 2018. Citado na página 18.

MIDORIKAWA, E. T. Uma introdução às linguagens de descrição de hardware. n. 2001, p. 1–14, 2007. Disponível em: <https://www2.pcs.usp.br/~labdig/material/intro-hdl.pdf>. Acesso em: 16 jul. 2018. Citado na página 35.

NICOLLE, J. P. *Serial interface (RS-232)*. [S.l.], 2006. Disponível em: <https://fpga4fun.com/SerialInterface.html>. Acesso em: 16 jul. 2018. Citado na página 48.

NORONHA, D. B. et al. Uma visão geral sobre dispositivos lógicos reconfiguráveis (fpga) e suas aplicações. *XIII Conferência de Estudos em Engenharia Elétrica*, 2015. Disponível em: <http://www.ceel.eletrica.ufu.br/artigos2005/ceel2005_059.pdf>. Acesso em: 16 jul. 2018. Citado 2 vezes nas páginas 33 e 34.

OKUYAMA, I. F.; MAXIMO, M. R. O. A.; PINTO, S. C. System identification of a hobby dc motor using a low cost acquisition setup. *Simpósio de automação inteligente*, 2015. Disponível em: <http://swge.inf.br/SBAI2015/anais/291.pdf>. Acesso em: 16 jul. 2018. Citado 2 vezes nas páginas 42 e 47.

OPENCV, P. *What is OPENCV?* [S.l.], 2017. Disponível em: <http://opencv.org/>. Acesso em: 16 jul. 2018. Citado na página 63.

PACHECO, M. A. C. Algoritmos genéticos: princípios e aplicações. *ICA: Laboratório de Inteligência Computacional Aplicada*, p. 28, 1999. Disponível em: <http://www2.ica.ele.puc-rio.br/Downloads/38/CE-Apostila-Comp-Evol.pdf>. Acesso em: 16 jul. 2018. Citado na página 37.

POLOLU, P. R. E. *75:1 Micro Metal Gearmotor HP 6V with Extended Motor Shaft*. [S.l.], 2017. Disponível em: <https://www.pololu.com/product/2215>. Acesso em: 16 jul. 2018. Citado 4 vezes nas páginas 42, 48, 57 e 58.

POLOLU, P. R. E. *DRV8833 Dual Motor Driver Carrier*. [S.l.], 2017. Disponível em: <https://www.pololu.com/product/2130>. Acesso em: 16 jul. 2018. Citado na página 59.

POLOLU, P. R. E. *Magnetic Encoder Pair Kit for Micro Metal Gearmotors, 12 CPR*. [S.l.], 2017. Disponível em: <https://www.pololu.com/product/2598>. Acesso em: 16 jul. 2018. Citado na página 58.

POLOLU, P. R. E. *Optical Encoder Pair Kit for Micro Metal Gearmotors, 5V*. [S.l.], 2017. Disponível em: <https://www.pololu.com/product/2590>. Acesso em: 16 jul. 2018. Citado na página 58.

POLOLU, P. R. E. *Pololu Wheel 60x8mm Pair - Yellow*. [S.l.], 2017. Disponível em: <https://www.pololu.com/product/1422>. Acesso em: 16 jul. 2018. Citado na página 58.

ROBOCUP, P. *Objectives*. [S.l.], 2017. Disponível em: <http://www.robocup.org/objective>. Acesso em: 16 jul. 2018. Citado na página 26.

SAAD, M. S.; JAMALUDDIN, H.; DARUS, I. Z. M. Implementation of pid controller tuning using differential evolution and genetic algorithms. *International Journal of Innovative Computing, Information and Control*, v. 8, n. 11, p. 7761–7779, 2012. Disponível em: <http://www.ijicic.org/ijicic-11-07073.pdf>. Acesso em: 16 jul. 2018. Citado 2 vezes nas páginas 46 e 47.

SEEDUINO, S. *Seeeduino V4.2*. [S.l.], 2018. Disponível em: <https://www.seeedstudio.com/Seeeduino-V4.2-p-2517.html>. Acesso em: 16 jul. 2018. Citado na página 33.

SILAS, F. R. et al. Tdp carrossel caipira - o time de futebol de robôs da unesp de bauru. *Competição Brasileira de Robótica - CBR*, 2009. Disponível em: <http://www.cbr09.fei.edu.br/inscricoes/TDP/caipirinha.pdf>. Acesso em: 16 jul. 2018. Citado na página 65.

SOFTBANK, P. *Robots: Who is the NAO Robot?* [S.l.], 2017. Disponível em: <https://www.softbankrobotics.com/emea/en/robots/nao>. Acesso em: 16 jul. 2018. Citado na página 26.

WAVESHARE, P. *CoreEP4CE6, ALTERA Core Board*. [S.l.], 2017. Disponível em: <http://www.waveshare.com/coreep4ce6.htm>. Acesso em: 16 jul. 2018. Citado 3 vezes nas páginas 43, 60 e 61.

WURMAN, P. R.; D'ANDREA, R.; MOUNTZ, M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, v. 29, n. 1, p. 9, 2008. Disponível em: <https://www.aaai.org/ojs/index.php/aimagazine/article/view/2082/1981>. Acesso em: 16 jul. 2018. Citado na página 29.