



Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Luís Felipe Monteiro da Fonte

**Desenvolvimento de um time de futebol de robôs para a
categoria IEEE VSSS**

Rio de Janeiro

2023

Luís Felipe Monteiro da Fonte

**Desenvolvimento de um time de futebol de robôs para a categoria IEEE
VSSS**



Projeto de Graduação apresentado, como requisito parcial para obtenção do grau de Engenheiro Eletricista com ênfase em Sistemas Eletrônicos, à Faculdade de Engenharia da Universidade do Estado do Rio de Janeiro.

Orientador:

Prof. Dr. Téo Cerqueira Revoredo

Rio de Janeiro

2023

Luís Felipe Monteiro da Fonte

**Desenvolvimento de um time de futebol de robôs para a categoria IEEE
VSSS**

Projeto de Graduação apresentado, como requisito parcial para obtenção do grau de Engenheiro Eletricista com ênfase em Sistemas Eletrônicos, à Faculdade de Engenharia da Universidade do Estado do Rio de Janeiro.

Aprovado em: 19 de Dezembro de 2023

Banca Examinadora:

Prof. Téo Cerqueira Revoredo, D.Sc. (Orientador)
Faculdade de Engenharia - UERJ

Prof. José Paulo Vilela Soares da Cunha, D.Sc.
Faculdade de Engenharia - UERJ

Prof. Lisandro Lovisolo, D.Sc.
Faculdade de Engenharia - UERJ

Rio de Janeiro

2023

AGRADECIMENTOS

[MUDAR AQUI]

Por Luís Felipe Monteiro da Fonte

RESUMO

Fonte, Luís. Projeto de Graduação (Graduação em Engenharia Elétrica com Ênfase em Eletrônica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, 2024.

O futebol de robôs é um desafio clássico, muito estudado na área de robótica móvel e controle. Um time de robôs com capacidade de jogar futebol, na prática, é um time de robôs que pode atender a muitas outras aplicações. Nesse contexto, este trabalho apresenta o desenvolvimento de um time completo de futebol de robôs da categoria IEEE VSSS 3x3. Inicialmente, são discutidas as regras da categoria e sua relevância acadêmica, mostrando a construção do campo e dos robôs, que utilizam impressão 3D e circuitos microcontrolados com conexão sem fio. Em seguida, é apresentado o desenvolvimento de sistema de visão computacional e interface gráfica, os mesmos são capazes de fornecer os dados de pose dos robôs e bola em tempo real. Por último, são apresentados o controlador PID de velocidade angular, embarcado nos robôs, e o controlador de trajetória usando campos vetoriais.

Robôs móveis diferenciais; Planejamento de trajetórias; Futebol de Robôs VSSS; Visão computacional; Controladores;;

ABSTRACT

Robot soccer is a classic challenge, widely studied in the field of mobile robotics and control. A team of robots capable of playing football, in practice, is a team of robots that can serve many other applications. In this context, this work presents the development of a complete robot soccer team in the IEEE VSSS 3x3 category. Initially, the rules of the category and their academic relevance are discussed, showing the construction of the field and robots, which use 3D printing and microcontrolled circuits with wireless connection. Next, the development of a computer vision system and graphical interface is presented, which are capable of providing robot and ball pose data in real time. Finally, the angular velocity PID controller, embedded in the robots, and the trajectory controller using vector fields are presented.

Keywords: Diferential robots; Trajectory planning; VSSS robot soccer; Computer vision; Controllers.

LISTA DE FIGURAS

Figura 1 campo da categoria VSSS 3x3. Fonte: regras da categoria [1].	16
Figura 2 Bola utilizada no jogo. Fonte: o autor.	17
Figura 3 Jogador e limite do encobrimento da bola. Fonte: regras da categoria [1] ..	18
Figura 4 Equipe UERJBotz. Fonte: UERJBotz.	19
Figura 5 Modalidade de locomoção diferencial. Fonte: o autor	21
Figura 6 Vistas da montagem do robô em CAD. Fonte: o autor.....	24
Figura 7 Peças do robô de futebol. Fonte: o autor.....	24
Figura 8 Montagem dos robôs. Fonte: o autor	25
Figura 9 Módulo ESP12 a esquerda e o mesmo módulo sem a proteção a direita. Fonte: o autor	27
Figura 10 Placa de controle. Fonte: o autor	28
Figura 11 Circuito de Alimentação. Fonte: o autor	29
Figura 12 Módulo com o circuito integrado DRV8833PWP. Fonte: o autor	30
Figura 13 Modulo com CI DRV8833PWP. Fonte: alterado da folha de dados do componente [2]	31
Figura 14 Placa vespa. Fonte: página do produto na Robocore.	32
Figura 15 Instalação da IMU MPU6050	33
Figura 16 Montagem Final do robô com a placa Vespa	33
Figura 17 Bateria LiPo 2S de 300mAh, Fonte: Hobbyking [3].	34
Figura 18 Placa ESP32 com cabo fazendo o papel de <i>Host</i>	35
Figura 19 Campo de futebol IEEE VSSS 3x3 construído.	36
Figura 20 <i>Webcam</i> Lenovo 300 FHD	37
Figura 21 Estrutura menor para testes	38
Figura 22 Padrão de etiquetas LARC/CBR VSSS 3x3 [1]	39
Figura 23 Etiquetas desenvolvidas para este trabalho.....	39
Figura 24 Logos das bibliotecas	40
Figura 25 Fluxograma do processamento dos frames. Fonte: o autor.....	42
Figura 26 Exemplo de captura com a visão. Fonte: o autor	42
Figura 27 Regiões das cores ao longo de H. Fonte: o autor	43

Figura 28 Exemplo de detecção de cores. Fonte: o autor	44
Figura 29 Representação gráfica dos centros das etiquetas e do robô. Fonte: o autor .	46
Figura 30 Representação gráfica do vetor distância e dos centros das etiquetas.	
Fonte: o autor	47
Figura 31 Dicionário de python gerado pelo sistema de visão. Fonte: o autor	47
Figura 32 Interface gráfica do projeto. Fonte: o autor	48
Figura 33 Painel de captura. Fonte: o autor.....	49
Figura 34 Monitores do sistema de visão. Fonte: o autor.....	50
Figura 35 Monitores dos jogadores. Fonte: o autor	50
Figura 36 Monitor da bola. Fonte: o autor	51
Figura 37 painel de conexão serial. Fonte: o autor	51
Figura 38 Painéis de ajuste de parâmetros. Fonte: o autor.....	52
Figura 39 Painel de controle do jogo e testes. Fonte: o autor	52
Figura 40 Modalidade de locomoção diferencial	53
Figura 41 Representação do centro de rotação e da trajetória de cara rodas em azul e vermelho	54
Figura 42 Diagrama em blocos da cinemática inversa do robô	55
Figura 43 Modelagem do motor representado em forma de circuito elétrico	56
Figura 44 Diagrama em blocos do motor.....	57
Figura 45 Representação gráfica das grandezas estudadas. Fonte: o autor.	57
Figura 46 Diagrama dos torques em função das velocidades angular e tangencial do robô	59
Figura 47 Diagrama do modelo completo do robô.....	59
Figura 48 Medição da velocidade angular com o giroscópio ao longo do tempo, me- dições a cada 2 segundos.	61
Figura 49 Campo vetorial gerado por duas cargas com polaridades opostas. Fonte: o autor.	65
Figura 50 Campo vetorial gerado, em azul os adversário e em vermelho a bola. Fonte: o autor.	67
Figura 51 Primeira medição com os robôs e bola parados.	68
Figura 52 Segunda medição com a bola em movimento.	69
Figura 53 Terceira medição com um robô em movimento.	70

Figura 54 Medições do sinal de controle e velocidade angular desejada e medida, com $PWM_{medio} = 0$.	73
Figura 55 Medições do sinal de controle e velocidade angular desejada e medida, com $PWM_{medio} = 1000$.	74
Figura 56 Medição de trajetória sem obstáculos.	75
Figura 57 Medição de trajetória com obstáculo, representado pelo jogador azul.	76

LISTA DE TABELAS

Tabela 1	Parâmetros dos motores	22
Tabela 2	Tabela verdade dos estados da ponte H	30
Tabela 3	Tabela de estados da ponte H em relação ao sinal PWM aplicado.....	30
Tabela 4	Lista dos principais comandos	36
Tabela 5	Características das ações de controle que compõe um controlador PID. $e(t)$ denota o erro de Regime Permanente, t_r o tempo de subida e t_s o tempo de acomodação.....	62
Tabela 6	Erros médios para valores variados.....	71

SUMÁRIO

1	INTRODUÇÃO	13
1.1	Visão geral da categoria IEEE VSSS	16
1.1.1	Campo	16
1.1.2	Bola	17
1.1.3	Jogadores	17
1.1.4	Goleiro	18
1.2	Objetivos	18
1.3	Equipe UERJBotz	19
1.4	Estrutura do texto	19
2	MONTAGEM DOS ROBÔS E ELEMENTOS DO JOGO	21
2.1	Os Robôs	21
2.1.1	Sistema de locomoção	21
2.1.2	Projeto Mecânico	23
2.1.3	Sistema eletrônico	25
2.1.3.1	Placa de controle, abordagem 1	26
2.1.3.2	Abordagem 2, usando a Placa Vespa	31
2.1.3.3	Bateria	33
2.2	Comunicação entre o computador e os robôs	34
2.2.1	O protocolo ESPNOW	35
2.2.2	Comandos da comunicação	35
2.3	Considerações finais sobre o sistema eletrônico	35
2.4	Construção do campo	36
3	VISÃO COMPUTACIONAL E INTEGRAÇÃO	37
3.1	Visão computacional	37
3.1.1	Material utilizado	37
3.1.2	Etiquetas dos robôs	39
3.1.3	Recursos computacionais	40
3.1.4	O algoritmo de visão	40

3.1.4.1	Captura	41
3.1.4.2	Detecção das cores	42
3.1.4.3	Segmentação e detecção de contornos	44
3.1.4.4	Posição e orientação	45
3.1.4.5	Encapsulamento	47
3.2	Interface gráfica	48
3.2.1	TKinter	48
3.2.2	Elementos da GUI	49
3.2.2.1	Painel de captura	49
3.2.2.2	Monitores	50
3.2.2.3	Monitores da bola e jogadores	50
3.2.2.4	Conexão serial	51
3.2.2.5	Painéis de ajuste de parâmetros	51
3.2.2.6	Painel de controle do jogo e testes	51
4	MODELAGEM MATEMÁTICA	53
4.1	Modelagem cinemática e locomoção diferencial	53
4.1.1	Cinemática direta	53
4.1.2	Cinemática inversa	54
4.2	Modelagem do motor	55
4.3	Modelo dinâmico do robô	57
5	CONTROLADOR DE VELOCIDADE ANGULAR	60
5.1	Calibração do giroscópio	60
5.2	Controle Proporcional Integral Derivativo - PID	60
5.2.1	Controlador PID discreto	63
5.3	Atuador	63
6	MOVIMENTAÇÃO AUTÔNOMA	64
6.1	Implementação usando controladores proporcionais	64
6.2	Planejamento de Trajetória	64
6.2.1	Campos vetoriais artificiais	64
6.2.2	Implementação	65

7	VALIDAÇÃO EXPERIMENTAL	68
7.1	Sistema de visão computacional	68
7.2	Controlador PID de velocidade angular	71
7.3	Controlador de trajetória com campos vetoriais	72
8	CONCLUSÃO	77
8.1	Trabalhos Futuros.....	78
	REFERÊNCIAS.....	79

1 INTRODUÇÃO

A robótica é a ciência que estuda a montagem e a programação de robôs, os quais são caracterizados como dispositivos autônomos reprogramáveis controlados por um programa de computador. As ações de construir e programar um robô exigem a combinação de conhecimentos de diversas áreas, o que confere à robótica um caráter multidisciplinar. Além disso, as atividades da robótica são normalmente mais produtivas quando realizadas por um grupo de pessoas trabalhando em conjunto, ao invés de um único indivíduo. Desse modo, a robótica tem potencial para ser uma ótima ferramenta de auxílio ao ensino. Convergindo teoria e prática, ela é capaz de desenvolver nos alunos alguns conceitos por vezes pouco abordados por outras disciplinas, tais como: Trabalho em equipe, autodesenvolvimento, capacidade de solucionar problemas, senso crítico, integração de disciplinas, exposição de pensamentos, criatividade, autonomia, postura empreendedora, etc. Além disso, a robótica estimula os alunos a buscarem soluções que integrem conceitos e aplicações de outras disciplinas envolvidas, tais como física, mecânica, eletrônica, design, computação, dentre outras.

No dia a dia de uma faculdade de Engenharia, nas suas disciplinas de laboratório, nota-se a constante repetição de experimentos tradicionais, frutos de conhecimentos já solidificados com o passar dos anos. A robótica educacional se insere como um agente de mudanças nesse modelo de repetições, pois demanda a participação de um grupo de alunos na concepção e na modelagem de um problema e da sua solução. O resultado é um projeto em forma de máquina (robô) que demonstra a aplicação dos conceitos discutidos e aprendidos em sala de aula e no cotidiano do grupo. Desse modo, o professor deixa de ser o único e exclusivo provedor de informações para tornar-se um parceiro no processo de aprendizagem.

Com todo esse potencial, é direto entender o surgimento e a sedimentação de diversos eventos de caráter educacional na área de robótica, que envolvem com frequência competição de robôs para os mais diversos objetivos. Como exemplo, cita-se a Competição Brasileira de Robótica (CBR), que é a maior competição de robótica e inteligência artificial do Brasil, realizada em parceria com o Instituto de Engenheiros Elétricos e Eletrônicos (IEEE), e que ocorre anualmente desde 2003.

A CBR comprehende as categorias da RoboCup Federation [4], uma iniciativa cien-

tífica internacional com o objetivo de promover o desenvolvimento de robôs inteligentes, inteligência artificial (IA) e automação. O público-alvo são os pesquisadores e estudantes universitários da área de robótica e inteligência artificial e estudantes do ensino técnico, médio e fundamental de todo o país. Em 2023, cerca de 200 equipes de mais de 130 instituições participaram da Competição Brasileira de Robótica, perfazendo mais de 2000 inscrições individuais.

Cabe ressaltar que a CBR ocorre tipicamente em conjunto com outros eventos da área, tais como a Olimpíada Brasileira de Robótica (OBR) e a Mostra Nacional de Robótica (MNR), além de simpósios científicos tais como o LARS (Simpósio Latino Americano de Robótica, do inglês *Latin American Robotics Symposium*), o SBR (Simpósio Brasileiro de Robótica), o WRE (*Workshop* de Robótica na Educação, do inglês *Workshop of Robotics in Education*) e o WTDR (*Workshop* de Teses e Dissertações em Robótica) e CTDR (Concurso de Teses e Dissertações em Robótica), além de eventos co-localizados.

A CBR é composta por 16 categorias que reproduzem problemas do cotidiano, nas quais robôs autônomos devem realizar tarefas corretamente, de acordo com regras e critérios bem definidos, dando aos concorrentes a oportunidade de desenvolver soluções completas e validá-las de forma prática em áreas que envolvem: Controle, Inteligência Artificial, Sistemas Multiagentes, Sensoriamento, Interface Homem-Robô, Reconhecimento de Objetos e Pessoas, etc.

Entre as categorias que compõem a CBR, seis são focadas em futebol de robôs e são resumidas a seguir:

Robocup small size soccer. Também conhecido como F-180, reproduz uma partida de futebol com robôs controlados remotamente ou à distância. Dois times com 06 robôs cada se enfrentam e quem fizer mais gols, em um intervalo de tempo definido, vence. Os robôs devem ter até 150mm de altura e 180mm de diâmetro.

Humanoid League. Nesta categoria, robôs autônomos com corpo e sentidos humanos jogam futebol um contra o outro. Os robôs são divididos em três classes de tamanho: *KidSize* (40-90 cm de altura), *TeenSize* (80-140 cm de altura) e *AdultSize* (130-180 cm de altura). O objetivo é andar, correr e chutar a bola dinamicamente, mantendo o equilíbrio, a percepção visual da bola, outros jogadores e o campo.

Robocup simulation 2D. Trata-se de uma liga de futebol simulada, onde duas equi-

pes de 11 agentes autônomos inteligentes jogam uma partida de futebol em duas dimensões. Cada agente, que representa um jogador, recebe informações limitadas sobre a situação do jogo e deve decidir sua ação pensando na equipe como um todo.

Robocup simulation 3D. Nesta liga também simulada, duas equipes de 11 robôs autônomos humanoides jogam futebol em 3 dimensões. O objetivo da competição de simulação em 3D é voltado para o design de comportamentos estratégicos do futebol para o controle de baixo nível de robôs humanoides, e para a criação de comportamentos básicos como caminhar, chutar, virar e levantar, entre outros.

Robocup junior soccer. Nesta categoria, os alunos precisam preparar dois robôs autônomos para um jogo de futebol. Cada equipe possui 2 robôs móveis autônomos, que jogam em um ambiente altamente dinâmico, rastreando uma bola emissora de luz especial em um campo fechado.

Very small size soccer (VSSS). Trata-se de uma categoria IEEE, na qual duas equipes de 3 robôs de até $7,5 \times 7,5 \times 7,5$ cm jogam uma partida de futebol. Os robôs são controlados remotamente por um computador, mas sem intervenção humana. O computador processa a imagem de uma câmera de vídeo colocada acima do campo e comanda os robôs.

A existência de tantas categorias baseada em futebol na Competição Brasileira de Robótica evidencia o interesse e a utilidade para a comunidade científica deste tipo de categoria como ferramenta não apenas de robótica educacional, mas também para a formação de profissionais e pesquisadores da área de robótica e para o desenvolvimento de ferramentas de controle, instrumentação, programação, etc. De forma geral, competições de futebol com robôs são formas lúdicas de incentivar a realização de PD&I na área de robótica autônoma multi-agente e permitir a implantação de sistemas experimentais de baixo custo no ambiente universitário (e escolar). Um time de robôs com capacidade de jogar futebol, na prática, é um time de robôs que pode atender a muitas outras aplicações.

Nesse contexto, este projeto propõe o desenvolvimento de um time de futebol de robôs passível de competição na categoria IEEE *Very small size soccer*. Apresenta-se desde o desenvolvimento conceitual, até testes de desempenho simples, passando pela construção e programação dos robôs e dos sistemas de visão e controle.

1.1 Visão geral da categoria IEEE VSSS

O futebol de robôs é uma disputa entre dois times com 3 ou 5 robôs em cada, podendo haver jogadores reservas, na qual o time vencedor é aquele que tiver o maior saldo de gols. Assim como no esporte com humanos, em caso de empate pode haver prorrogação e disputa de pênaltis. Em termos de tamanho, a VSSS é a menor entre as subcategorias de futebol. Os jogadores são pequenos, até 75x75x75 mm cada e jogam com uma bola de cor laranja, similar a uma de golf, em um campo com dimensões de 150 cm por 130 cm. Resumindo-se a seguir as principais características da categoria IEEE VSSS. Os detalhes podem ser encontrados no documento com as regras da categoria no link

1.1.1 Campo

A base e as laterais do campo são construídas em madeira de acordo com as medidas descritas na Figura 1. Ele possui diversas marcações que indicam onde deve ser posicionada a bola e delimitam as áreas do campo. As cores do campo são determinadas nas regras e não podem ser alteradas.

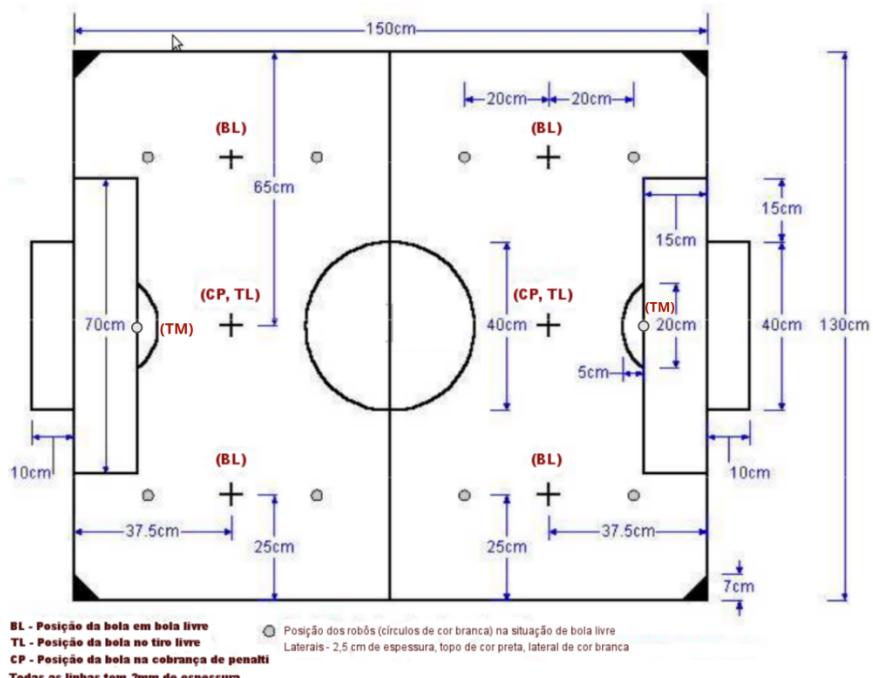


Figura 1: campo da categoria VSSS 3x3. Fonte: regras da categoria [1].

1.1.2 Bola

A bola deve ser Laranja com aproximadamente 42,7mm de diâmetro e 46g de massa (similar a uma bola de golf).



Figura 2: Bola utilizada no jogo. Fonte: o autor.

1.1.3 Jogadores

Cada partida conta com no máximo 3 jogadores. Os robôs são retangulares de cor preta, com etiquetas na parte superior, que devem identificar o time e é conveniente que ela permita distinguir os robôs e sua orientação. As dimensões máximas de cada robô são de 75x75x75mm, além disso podem vestir uniformes, desde que não ultrapassem 80x80x80mm. Cada time recebe uma cor que pode ser azul ou amarelo, os times devem ter etiquetas já preparadas para estas duas opções (é conveniente que o topo do robô possa ser removível para facilitar a alteração). Além da cor do time, as etiquetas também podem conter mais duas cores, desde que sejam diferentes da cor do time adversário e não sejam branco, laranja ou cinza. A etiqueta deve estar contida na região de um quadrado de 3,5cm de lateral ou na região de um círculo de 4cm de diâmetro. É permitido que cada jogador possua um mecanismo para manusear a bola, desde que 70% do diâmetro da bola ainda fique a mostra.

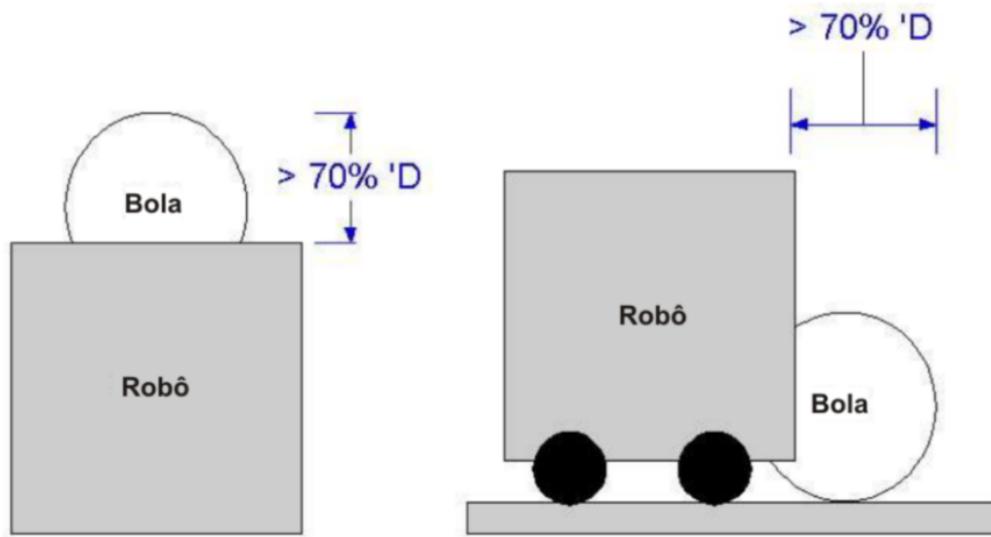


Figura 3: Jogador e limite do encobrimento da bola. Fonte: regras da categoria [1]

1.1.4 Goleiro

Cada time pode designar um jogador para a função de goleiro. Em uma partida o goleiro é o jogador que está a mais tempo na área de seu próprio gol, quando um goleiro sai da área de seu próprio gol, seu contador é zerado e o próximo a robô do seu time a entrar na área do gol é considerado o goleiro. É permitido apenas ao goleiro segurar a bola em sua própria área e encobrir mais do que 30% do diâmetro da bola.

1.2 Objetivos

Este trabalho tem por objetivo apresentar o desenvolvimento de um time de futebol de robôs aplicável a categoria IEEE Very small size soccer, desde a sua concepção até a validação experimental. Os objetivos específicos do trabalho podem ser resumidos como segue:

1. Fabricação dos jogadores (robôs) e do campo de futebol
 - (a) Fabricação da estrutura mecânica dos robôs;
 - (b) Construção do sistema de controle embarcado;
 - (c) Construção do campo de jogo e de uma estrutura menor para testes;
 - (d) Fabricação de estrutura auxiliar para posicionamento de câmeras.

2. Desenvolvimento do(a):

- (a) Canal de comunicação do computador com os robôs em campo;
- (b) Algoritmo de visão computacional capaz de identificar os jogadores, o campo e a bola;
- (c) Interface gráfica para monitoramento do jogo e comunicação com os jogadores;
- (d) Algoritmos de controle para movimentação autônoma dos robôs.

1.3 Equipe UERJBotz

O desenvolvimento deste trabalho teve o apoio da equipe UERJBotz, a qual o autor é membro desde 2018. A UERJBotz é uma equipe de robótica associada à Universidade do Estado do Rio de Janeiro (UERJ), comprometida desde 2013 com o progresso e realização de projetos em diversas categorias dentro do campo da robótica [5].



(a) Logo da Equipe UERJBotz.



(b) Foto da equipe em competição de robótica

Figura 4: Equipe UERJBotz. Fonte: UERJBotz.

1.4 Estrutura do texto

O texto deste trabalho é estruturado como segue: O Capítulo 2 discorre sobre o projeto, a fabricação e os sinais de comando dos robôs móveis que compõem o time de futebol. No Capítulo 3 é apresentado o sistema de visão computacional, sendo descritos desde os materiais utilizados até a integração e operação, incluindo a interface gráfica desenvolvida. No Capítulo 4 é apresentada a modelagem matemática dos robôs móveis que baliza o desenvolvimento dos algoritmos para movimentação autônoma apresentados

nos Capítulos 5 e 6. O Capítulo 7 apresenta a validação experimental dos sistemas e no Capítulo 8 são feitas considerações finais sobre o trabalho.

2 MONTAGEM DOS ROBÔS E ELEMENTOS DO JOGO

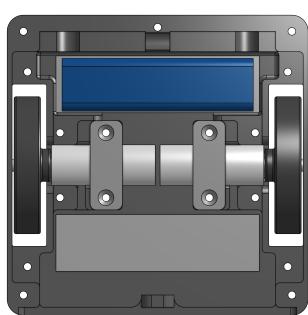
Neste capítulo é descrita a montagem dos robôs, do campo e da interface de comunicação entre o computador e os robôs. Destaca-se inicialmente a montagem dos robôs, seus projetos mecânico e eletrônico, apresentando-se em seguida, a placa que realiza a interface entre os robôs e o computador.

2.1 Os Robôs

Os robôs da categoria VSSS 3x3 são controlados sem fio e têm dimensões limites de 75 x 75 x 75mm, podendo chegar a 80 x 80 x 80mm com uniforme. Na parte superior de cada robô são posicionadas as etiquetas que indicam a cor do time e o numero do jogador.

2.1.1 Sistema de locomoção

O sistema de locomoção adotado é do tipo diferencial, por conta da facilidade de montagem e maior liberdade de movimentação, comparado com outras geometrias tais como a de Ackerman [6]. Os robôs utilizam dois motores com rodas de mesmo diâmetro acopladas ao eixo, que são posicionados em sentidos opostos de forma concêntrica, a Figura 41 exibe uma vista superior do robô e suas rodas direita e esquerda que compõem o sistema de locomoção. É comum neste sistema de locomoção o uso de rodas bobas de apoio na frente e na parte de trás do robô, no entanto, as opções disponíveis eram muito grandes. A solução adotada foi deixar a parte frontal e traseira o mais próximo possível do chão para reduzir o efeito de gangorra, apesar disso o atrito com o chão é um problema. Porém como o chão é e o atrito com o plástico é baixo não inviabiliza a locomoção.



(a) Vista superior



(b) Motores e rodas

Figura 5: Modalidade de locomoção diferencial. Fonte: o autor

Tabela 1: Parâmetros dos motores

Parâmetro	Equação	Unidade	Descrição
n	fornecido	RPM	Frequência de rotação nominal
$I_{\text{no_load}}$	fornecido	A	Corrente sem carga
I_{stall}	fornecido	A	Corrente de rotor bloqueado
V_n	fornecido	V	Tensão nominal
V_A	fornecido	V	Tensão aplicada aos motores
r	fornecido	m	Raio da roda do robô
F_{at}	medido	N	Força de atrito entre robô e chão
m	medido	kg	Massa do robô
k	$\frac{30(V_n - I_{\text{no_load}}R_A)}{\pi n}$	Vs/rad	Constante do motor
R_A	$\frac{V_n}{I_{\text{stall}}}$	Ω	Resistência de armadura
v_{max}	$\frac{r(V_A - I_{\text{loc}}R_A)}{k}$	m/s	Velocidade máxima do robô
I_{loc}	$\frac{2rF_{\text{at}}}{k}$	A	Corrente estimada de locomoção
a_{max}	$\frac{F_{\text{max}}}{m}$	m/s^2	Aceleração máxima do robô
F_{max}	$\frac{2kI_{\text{stall}}}{r} - F_{\text{at}}$	N	Força máxima do robô

Os motores foram escolhidos em função de seu tamanho, velocidade e torque. Foram escolhidos dentre as opções disponíveis, micromotores de corrente contínua escovados de ímãs permanentes chamados comercialmente de N20. Eles são leves com apenas 9,6g e possuem uma micro-redução na saída que reduz a velocidade e aumenta a capacidade de torque na saída. Esses motores possuem variações em função da tensão nominal e velocidade nominal. O modelo escolhido possui velocidade nominal de 500RPM, com tensão nominal de 6V. Dentre as opções disponíveis o modelo de 500RPM possui velocidade e torque adequado para movimentar o robô. A Tabela 1 apresenta os principais parâmetros do robô e fornece equações que aproximam os parâmetros dos motores em função dos dados fornecidos por vendedores. Elas foram utilizadas para balizar a escolha dos motores, em função da velocidade final desejada e da aceleração máxima.

2.1.2 Projeto Mecânico

As partes mecânicas dos robôs, desenvolvidas pelo autor, são impressas em uma impressora 3D do tipo *FDM* [7] (*fused deposition modeling* - método de fabricação aditiva) usando os materiais PLA (Biopolímero ácido poliláctico) e PETG (Polietileno Tereftalato Glicol). A modelagem da estrutura dos robôs foi realizada usando o software *CAD* (Computer Aided-Design) *Onshape* [8]. Foram levados em consideração durante a modelagem os seguintes requisitos:

- Desenvolver peças viáveis para impressão 3D, minimizando a dependência de suportes durante a impressão;
- Limitação de espaço de até 75 x 75 x 75mm;
- A parte frontal que encaminha a bola não deve encobrir mais que 30% do diâmetro da bola em vista superior;
- Tampa fácil e rápida de ser trocada;
- Facilidade de montagem e troca de peças;
- Menor número possível de peças;
- Locomoção diferencial;
- Espaço interno suficiente para acomodar a bateria, os componentes eletrônicos e lastro de chumbo¹;
- Preferência ao uso de encaixes ao invés de parafusos;
- Centro de massa no centro do robô em vista superior e o mais baixo possível em relação ao chão.

Ao final, todos os requisitos foram atendidos com sucesso. Cada jogador possui duas rodas com pneus de silicone e 4 peças impressas: uma carcaça lateral, duas presilhas, uma para cada motor, e uma tampa que pode ser trocada. A Figura 8 apresenta as vistas isométrica da carcaça, superior e uma vista livre exibindo a traseira do robô. Os elementos

¹O chumbo é um dos metais mais densos, os lastros servem para aumentar a tração nas rodas dos robôs. A tração é importante para garantir força em disputas de bolas e aceleração.

que compõem o robô são expostos em suas posições na Figura 7, com exceção da placa de controle que é posicionada no espaço que sobra acima das rodas.

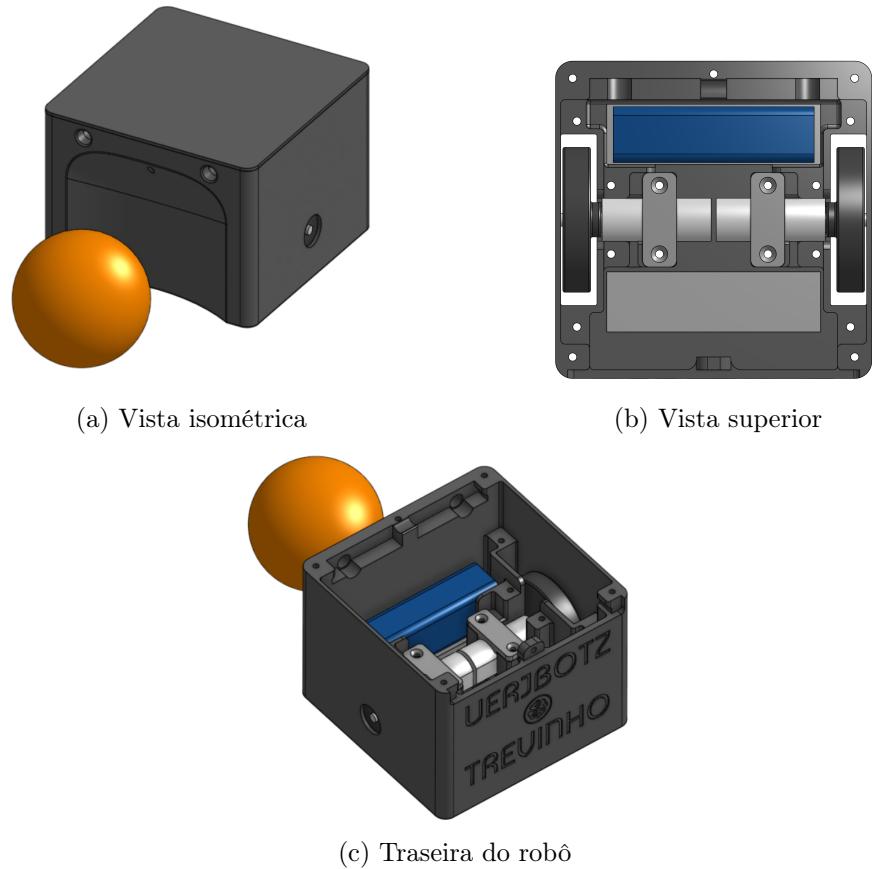


Figura 6: Vistas da montagem do robô em CAD. Fonte: o autor

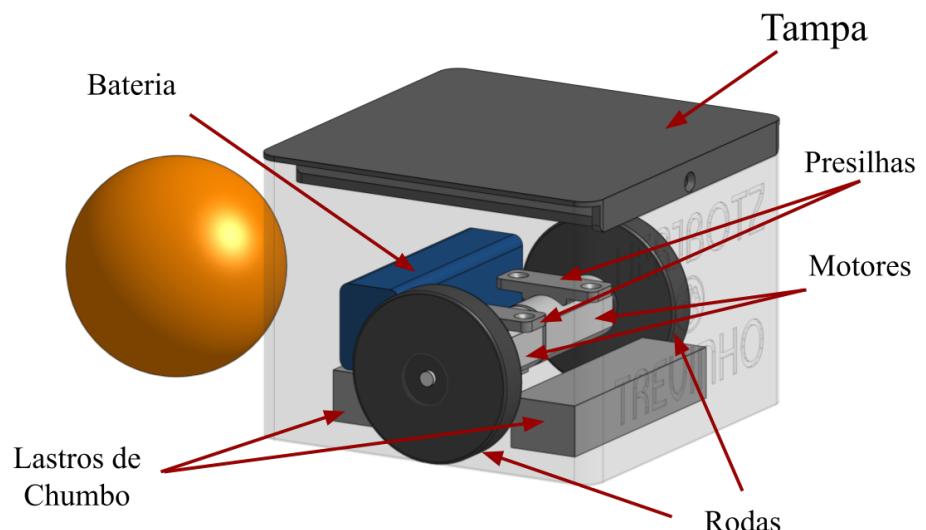


Figura 7: Peças do robô de futebol. Fonte: o autor

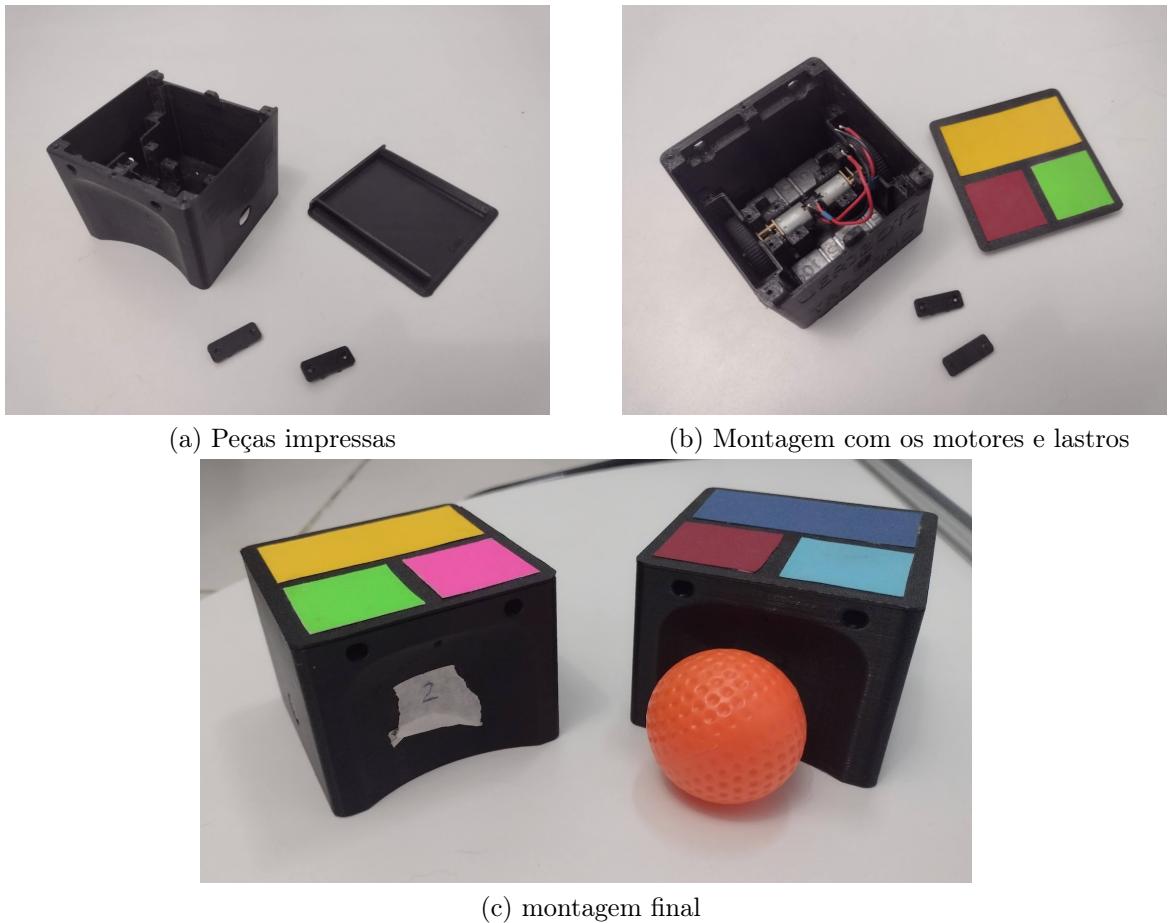


Figura 8: Montagem dos robôs. Fonte: o autor

2.1.3 Sistema eletrônico

O sistema eletrônico do robô é composta por um circuito receptor de radiofrequência, um microcontrolador que processa os sinais recebidos e aciona os motores usando um circuito de potência composto por duas pontes H (uma para cada motor) e um circuito de alimentação para suprir as demandas de tensão e corrente. Além de um sistema de sensoriamento usando uma *IMU* (*Inertial Measurement Unit*) que é utilizada na regulação de velocidade angular de cada robô.

No intuito de permitir a confecção de mais robôs, foram adotadas duas abordagens equivalentes na montagem dos sistemas eletrônicos. A abordagem 1 utiliza componentes discretos, módulos e circuitos integrados para construir uma placa de controle. Enquanto a abordagem 2 utiliza a placa Vespa, fabricada pela loja virtual Robocore, ela contém quase todos os circuitos necessários pra controlar os robôs. A seguir são listados os requisitos definidos para o projeto e em seguida apresentadas as duas abordagens.

1. Usar microcontroladores da Espressif que permitam o uso do protocolo ESPNOW.
2. Possuir circuito de alimentação capaz de receber baterias 2S (com duas células) ou seja, tensão máxima de entrada superior a 8,4V.
3. Possuir circuito de proteção contra polaridade reversa, para proteger em caso do conector da bateria ser conectado ao contrário.
4. Ser capaz de medir a tensão da bateria para sinalizar a necessidade de troca.
5. Possuir duas pontes H que possam ser alimentadas com tensão 2S e possuir corrente contínua de ao menos 600mA, para garantir força e aceleração adequadas dos motores.
6. Possuir unidade inercial para balizar a regulação de velocidade angular.

2.1.3.1 Placa de controle, abordagem 1

Na abordagem 1, é projetada, fabricada e montada uma PCI (Placa de Circuito Impresso) que contém o circuito de controle do robô. A placa de controle é o circuito responsável por receber os comandos do computador e alterar a velocidade dos motores. Esta placa foi projetada usando o microcontrolador ESP8266 [9]. No intuito de deixar a placa compacta e leve, ela foi projetada usando componentes *SMD* (*Surface Mounted Device*). A fabricação da PCI foi realizada por uma empresa do ramo. Enquanto a montagem foi realizada manualmente usando uma estação de solda e retrabalho.

Microcontrolador. A escolha do microcontrolador levou em consideração a necessidade de conexão sem fio, baixo consumo, tamanho reduzido, baixo custo e boa capacidade de processamento. Avaliando as opções de microcontroladores que atendem a estes requisitos o microcontrolador ESP8266 foi o escolhido. De acordo com sua folha de dados [9], o microcontrolador adotado possui o processador *L106 Diamond* de 32bits, podendo operar de 80 a 160MHz, além de poder se comunicar sem fio na faixa de 2,4GHz, utilizando o protocolo IEEE 802.11 b/g/n ou o protocolo ESP-NOW. Além disso possui capacidade suficiente de processamento e memória para atender as necessidades do projeto, seu circuito de radiofrequência integrado dispensa a necessidade de módulos externos, como o NRF24L01, Bluetooth ou xBee,

simplificando o projeto. Para facilitar a confecção da placa e reduzir o número de componentes foi utilizado um modulo da empresa *AI-Thinker* que contém o ESP8266, chamado de ESP12 [10].

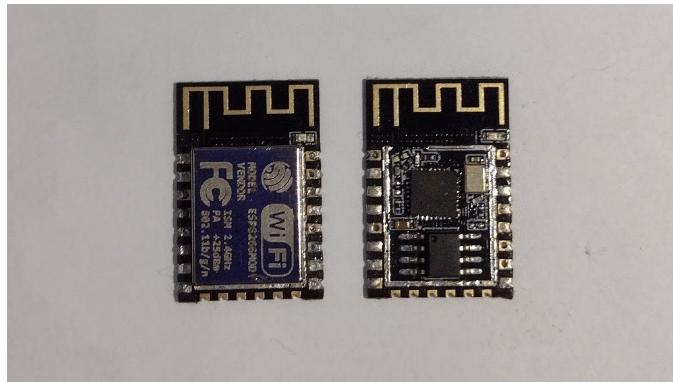


Figura 9: Módulo ESP12 a esquerda e o mesmo módulo sem a proteção a direita. Fonte: o autor

Circuito de alimentação. A alimentação do robô é realizada em duas etapas para aumentar suas eficiência e estabilidade. Foi utilizado um conversor buck de tensão CC-CC, o qual reduz a tensão para 5V que em seguida é regulada pelo regulador de tensão AMS1117 para 3,3V, que é utilizada para alimentar o microcontrolador ESP8266. O conversor buck faz parte de um módulo que utiliza o circuito integrado MP2307. O regulador de tensão utilizado possui encapsulamento sot-223 SMD (Surface Mount Device) e é soldado diretamente na placa. Capacitores de tântalo e cerâmicos complementam o circuito de alimentação, provendo maior estabilidade da tensão, filtrando ruídos. A alimentação também dispõem de um circuito de proteção contra tensão reversa, implementado com um MOSFET tipo P e um resistor. Este circuito é fundamental para evitar perdas recorrentes por conexões acidentais com polaridade invertida. A seguir são listados os recursos de alimentação da placa:

- LED que indica que a placa está sendo alimentada
- Capacitores para estabilizar a tensão, que atuam principalmente para impedir que o MCU reinicie durante picos de corrente causados pelos motores.
- Circuito de proteção contra tensão reversa na entrada de alimentação, impedindo que erros dos operadores queimem o circuito.

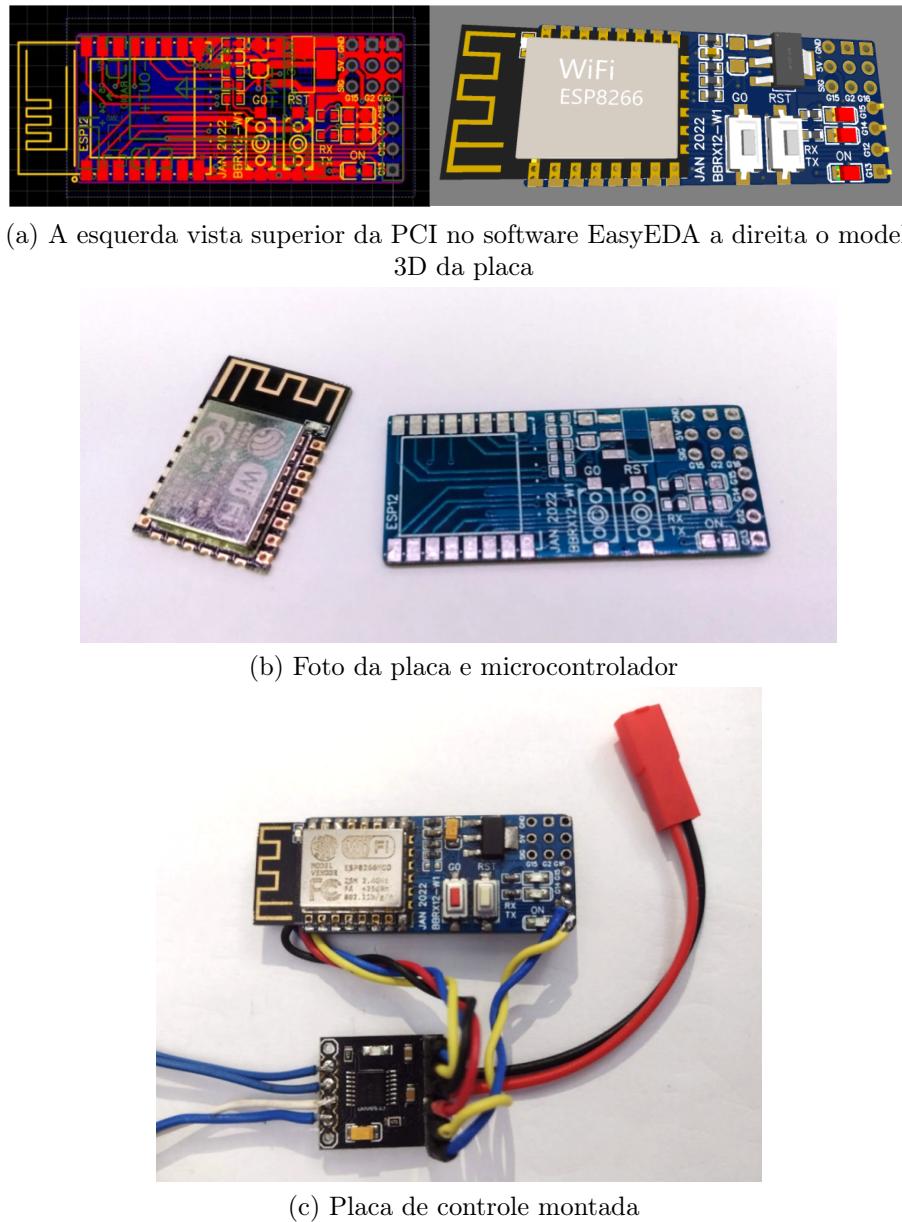


Figura 10: Placa de controle. Fonte: o autor

- Divisor resistivo conectado à entrada do ADC do microcontrolador para monitorar a tensão da bateria.

Ponte H. O acionamento dos motores é realizado aplicando tensão continua em seus terminais. A tensão aplicada é diretamente proporcional a velocidade angular na saída, onde a inversão de polaridade implica em inversão do sentido de rotação. O circuito mais completo para realizar o acionamento deste tipo de motor é uma ponte H. A estrutura deste circuito permite o controle de velocidade, sentido de rotação, freio e modo desligado. Em sua essência ele é composto por 4 chaves eletrônicas acionadas

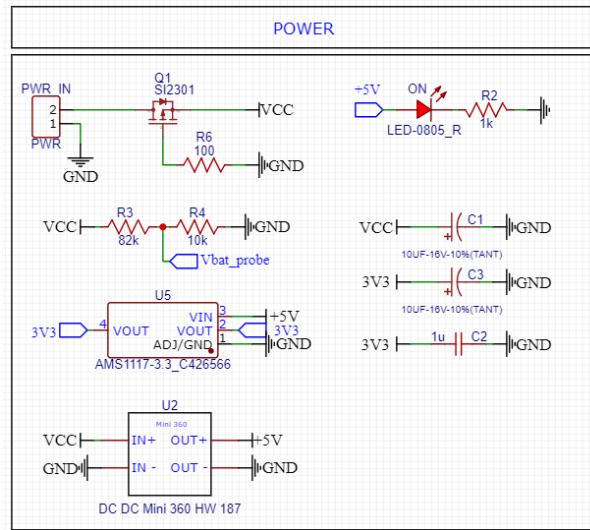


Figura 11: Circuito de Alimentação. Fonte: o autor

por um circuito específico que realiza o correto acionamento das chaves. O modo de operação é selecionado aplicando sinais eletrônicos na entrada do circuito. Foi escolhido um módulo de ponte H que utiliza o circuito integrado DRV8833PWP [2]. O circuito integrado DRV8833 possui duas pontes H completas em um único chip. A comutação é realizada usando MOSFETS canal N com baixo $R_{s(on)}$, $360m\Omega$ no total. Este módulo foi escolhido por ser pequeno, leve e atender as necessidades de tensão e corrente de operação do robô. Além disso possui proteção contra sobrecorrente e temperatura, tornando o projeto mais confiável e durável. Principais características do CI DRV8833:

- Tensão de alimentação 2,7V até 10,8V;
- Baixo $R_{s(on)}$, $360m\Omega$ no total (HS + LS);
- Corrente de pico de 2A por motor e 1,5A RMS;
- Proteção de contra sobre corrente e temperatura;
- Pinos de controle compatíveis com 3,3V ou 5V

A alimentação dos motores é realizada diretamente com a tensão da bateria, ou seja, 8,4V com a bateria carregada. Os comandos vindos da placa de controle são recebidos por 4 pinos, 2 para cada motor. Os comandos são programados de acordo com as tabelas verdade descritas na folha de dados do circuito integrado e apresentadas nas Tabelas 2 e 3. Para permitir o controle de direção e velocidade

um pino é mantido em 0V e no outro é aplicado um sinal *PWM* (*Pulse Width Modulation*), para inverter a rotação basta inverter os pinos, ou seja, o primeiro passa a receber *PWM* e o segundo 0V. No código isso é realizado pelo microcontrolador por uma função escrita em C++ que controla individualmente cada um dos motores.

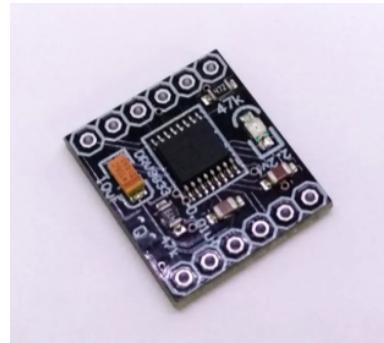


Figura 12: Módulo com o circuito integrado DRV8833PWP. Fonte: o autor

Tabela 2: Tabela verdade dos estados da ponte H

xIN1	xIN2	xOUT1	xOUT2	Função
0	0	Z	Z	desligado
0	1	L	H	para frente
1	0	H	L	para trás
1	1	L	L	Freio

Tabela 3: Tabela de estados da ponte H em relação ao sinal PWM aplicado.

xIN1	xIN2	Estado
0	0	desligado
0	PWM	para trás com velocidade controlada
PWM	0	para trás com velocidade controlada
1	1	Freio

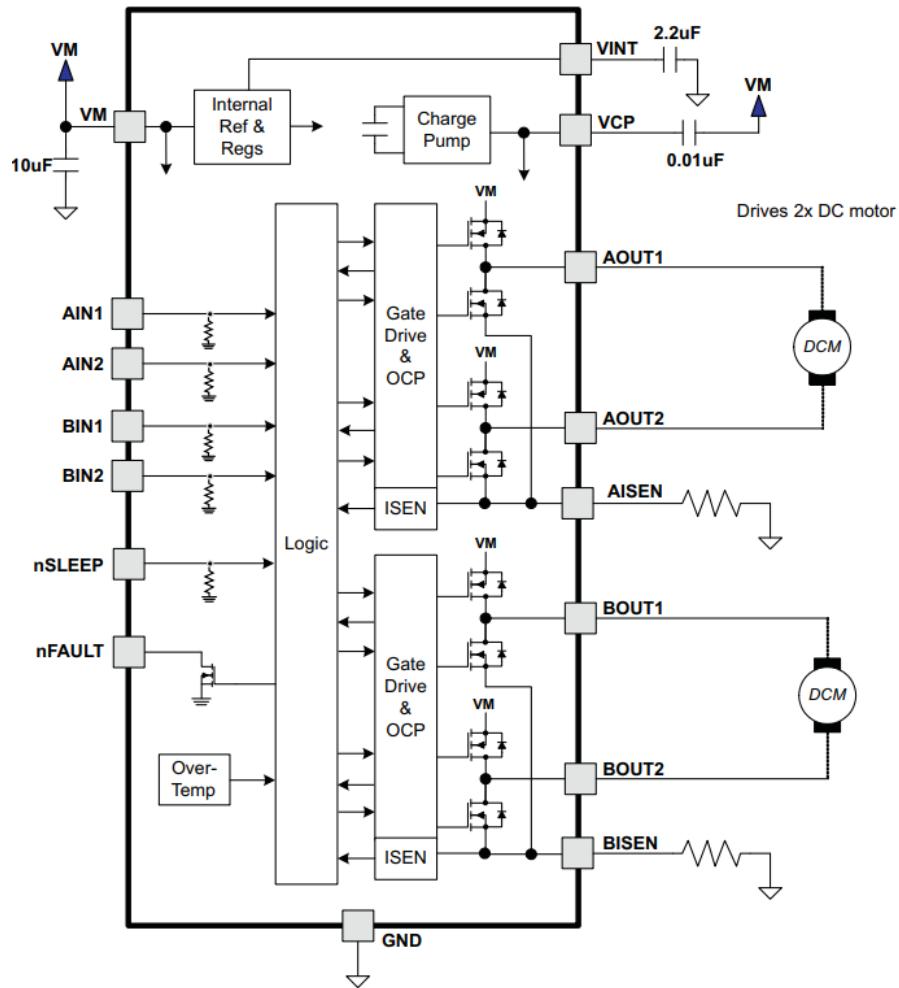


Figura 13: Modulo com CI DRV8833PWP. Fonte: alterado da folha de dados do componente [2]

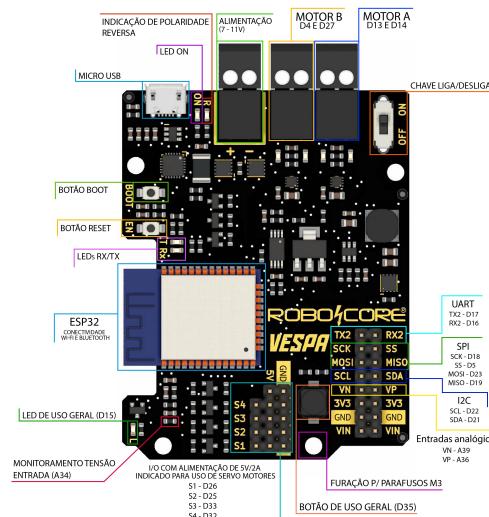
2.1.3.2 Abordagem 2, usando a Placa Vespa

A placa Vespa é fabricada pela Robocore [11], ela utiliza o microcontrolador ESP32 e possui formato compatível com a consagrada placa de desenvolvimento Arduino UNO Rev3 [12]. As principais características desta placa pertinentes ao futebol de robôs são: possui duas pontes H DRV8837 [13] (uma para cada motor) capaz de fornecer até 800mA contínuos, fonte DC-DC de alimentação capaz de fornecer até 2,5A e receber até 11V além de possuir proteção contra polaridade reversa e possuir pinos livres que permitem a conexão de uma *IMU*. A Figura 14, extraída do site do fabricante, apresenta a placa e sua pinagem com indicação dos componentes.

A Vespa não possui *IMU*, por isso foi fixada com fita dupla-face a *IMU* MPU6050, que foi conectada com pequenos fios soldados diretamente nos pinos.



(a) Foto da placa



(b) Pinagem

Figura 14: Placa vespa. Fonte: página do produto na Robocore.

A montagem final, apresentada na Figura 16, ficou compacta e com menos conexões, no entanto, foi preciso soldar os fios dos motores e do conector da bateria diretamente nela, uma vez que usando o conector que vem com ela não haveria espaço. Essas conexões foram realizadas com fios vermelhos e pretos que podem ser observada na Figura 15.

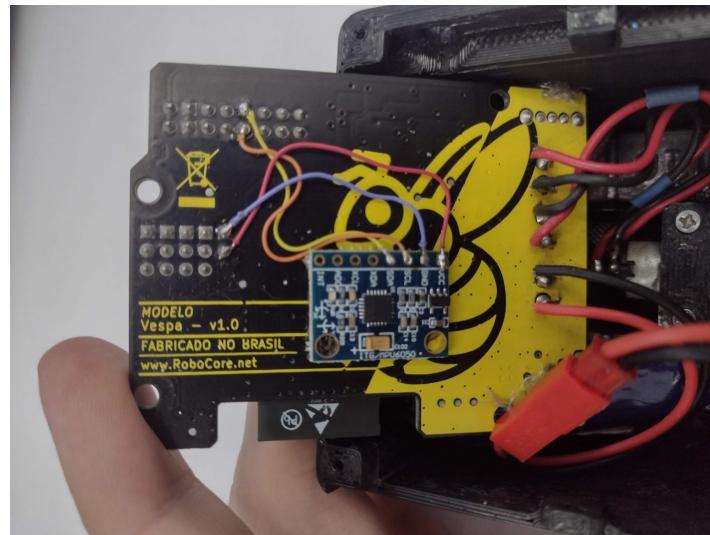


Figura 15: Instalação da IMU MPU6050



Figura 16: Montagem Final do robô com a placa Vespa

2.1.3.3 Bateria

A bateria é dimensionada para atender a velocidade máxima esperada e durar os dois tempos de jogo com folga. Foi escolhida uma bateria de LiPo (Polímero de Lítio) 2S de 300mAh. A simbologia 2s indica que a bateria possui duas células em série com tensão nominal de 3,7V. Na realidade cada célula pode variar entre 4,2V, quando carregada, até 3,3V, quando esta descarregada, abaixo deste valor não é considerado segura sua utilização. O projeto utiliza duas células para garantir tensão adequada para alimentar os motores e os circuitos. O cálculo da capacidade foi realizado através da Equação (1).

$$C = I_{medio}\Delta t \quad (1)$$

Na qual: C é a capacidade da bateria em mAh, I_{medio} é o consumo médio de corrente em mA (medido) e Δt o intervalo de tempo, em horas, no qual a bateria deve alimentar o robô.

Este tipo de bateria pode causar incêndios, em caso de perfuração ou curto-circuito. Por isso é importante armazena-las de forma adequada em bolsas conhecida como *LiPo safe bag*. O carregamento delas deve ser realizado por um circuito específico que monitora a tensão de cada célula para que não ocorra sobrecarga de nenhuma das células. São vendidos carregadores específicos para este tipo de baterias.



Figura 17: Bateria LiPo 2S de 300mAh, Fonte: Hobbyking [3].

2.2 Comunicação entre o computador e os robôs

O processamento do sistema de visão e dos algoritmos do jogo são executado em um computador externo ao campo. Desta forma é preciso estabelecer um canal de comunicação entre este e os robôs, de tal forma que possam receber comando de movimentação. A solução adotada neste projeto foi o uso de uma placa de desenvolvimento com o microcontrolador ESP32, apresentado na Figura 18. Ela recebe os comandos via conexão serial estabelecida entre a placa e o computador através da porta USB. Em seguida, os comandos recebidos são enviados para os robôs em conexão sem fio usando o protocolo ESPNOW.

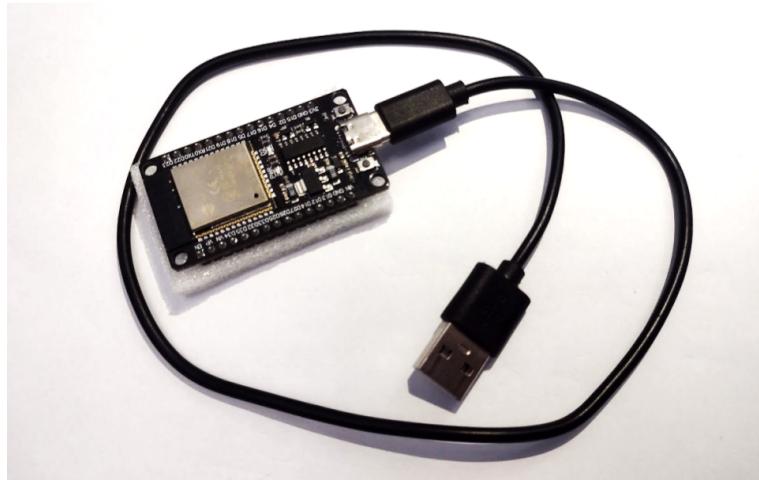


Figura 18: Placa ESP32 com cabo fazendo o papel de *Host*

2.2.1 O protocolo ESPNOW

O protocolo ESPNOW foi desenvolvido pela empresa Espressif Systems para seus microcontroladores. Ele permite o pareamento direto entre diversos ESP8266 e ou ESP32, permitindo enviar e receber dados [14]. A motivação para a escolha deste protocolo foi a facilidade de sua implementação, baixíssima latência e rápido reestabelecimento de conexão em casos de desligamento do transmissor ou receptor, em comparação com protocolos como *bluetooth* e *WiFi*, onde a reconexão é demorada. Poderia ser utilizado o protocolo IP, no entanto, não foi possível no tempo disponível.

2.2.2 Comandos da comunicação

A placa *Host* recebe comandos em formato de texto através da comunicação serial e as envia para os robôs via ESPNOW. A Tabela 4 apresenta os principais comandos.

2.3 Considerações finais sobre o sistema eletrônico

Ao final da montagem as duas abordagens para o sistema eletrônico funcionaram como esperado. A abordagem com a placa Vespa foi contraída mais rapidamente, porém teve custo mais elevado e ocupa mais espaço, a abordagem com a PCI montada foi mais barata porém levou mais tempo para ser construída. Além disso esta PCI poderia conter a ponte H e a *IMU* soldados na mesma, para reduzir a quantidade de módulos e conexões usando fios.

Tabela 4: Lista dos principais comandos

comandos	funções
reset	reinicia o robô
bot.stop	freia o robô
bot.move <vel. esq.> <vel. dir.>	aplica as velocidade em cada roda
pid.on <estado>	liga ou desliga o controlador PID
pid.linear_speed <vel.>	altera a velocidade tangencial, valor de PWM
pid.angular_speed <vel.>	altera o valor da velocidade angular em rad/s
pid.kp <valor>	altera ou retorna o valor de kp
pid.ki <valor>	altera ou retorna o valor de ki
pid.kd <valor>	altera ou retorna o valor de kd
pid.plot <state>	habilita os desabilita o retorno de dados do PID
bot.bip	o robô faz um alerta sonoro
bot.voltage	retorna a tensão do robô

2.4 Construção do campo

A construção do campo foi realizada na oficina da UERJ, seguindo as regras oficiais da categoria IEEE VSSS 3x3 a Figura 19 apresenta o campo construído.

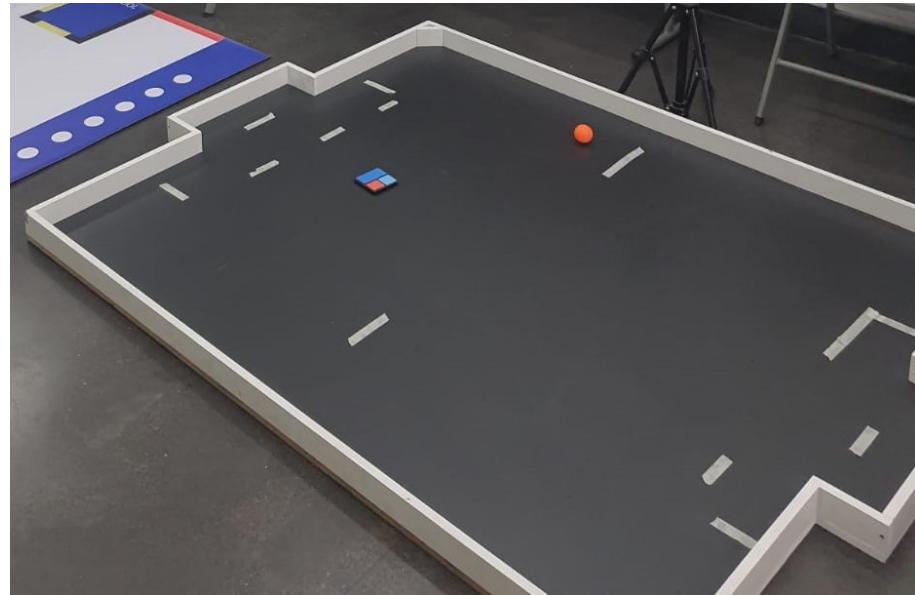


Figura 19: Campo de futebol IEEE VSSS 3x3 construído.

3 VISÃO COMPUTACIONAL E INTEGRAÇÃO

Neste trabalho foram desenvolvidos um algoritmo de visão computacional e uma aplicação multiplataforma com interface gráfica que controla e monitorar o time durante a partida.

3.1 Visão computacional

A visão computacional é o ramo da computação que busca extrair informações de imagens. No futebol de robôs ela é utilizada desde o surgimento da categoria como sistema que informa ao algorítimo principal, a posição e orientação de cada jogador, a posição da bola e as marcações do campo.

3.1.1 Material utilizado



Figura 20: *Webcam* Lenovo 300 FHD

Neste projeto é utilizada uma *webcam* USB de 30 FPS (Frames Por Segundo) [15], dentre as opções disponíveis ela foi a que obteve melhor qualidade de imagem e frequência de aquisição. Para obter melhores resultados, seria desejável o uso de uma câmera com maior frequência de aquisição, no entanto, por falta de recursos e tempo não foi possível. Ela é fixada em uma estrutura lateral que garante distância adequada entre sua lente e o campo, que de acordo com as regras da categoria deve ser de 2 metros. Foi construída uma estrutura principal junto ao campo, seguido todas as regras da categoria, e uma outra estrutura menor para realizar testes. A distância entre a câmera e o campo influencia

na razão de milímetros por pixel, quanto maior a distância maior a razão em mm/pixel. Esta razão deve ser ajustada na interface gráfica antes de toda partida, isso é realizado tomando como referência as dimensões do campo, que são fixas.

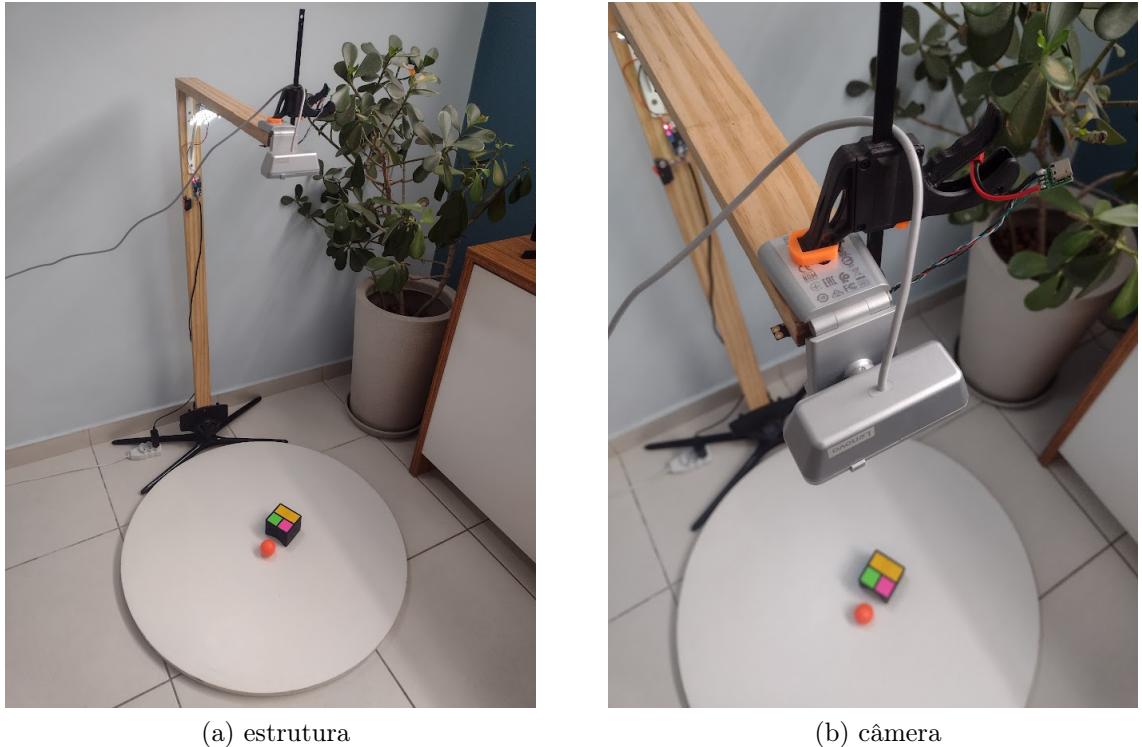


Figura 21: Estrutura menor para testes

3.1.2 Etiquetas dos robôs

A etiqueta é o elemento visual posicionado no topo do robô que indica o seu time e sua numeração. Ela pode ser construída de diversas formas, no entanto, recentemente na LARC e na CBR [1] as etiquetas foram padronizadas, definindo o formato, dimensões e combinação de cores. Na Figura 22a esta o desenho técnico da geometria padrão e na Figura 22b o esquema de cores das possíveis etiquetas permitidas. Os retângulos maiores indicam o time do robô, e só podem ser azul ou amarelo, enquanto os quadrados menores indicam a numeração do mesmo, podendo ser azul claro, verde, vermelho ou rosa. Além disso a tampa e o fundo da etiqueta devem ser pretos. A Figura 23 apresenta as etiquetas fabricadas neste trabalho com base nas regras oficiais da categoria VSSS 3x3.

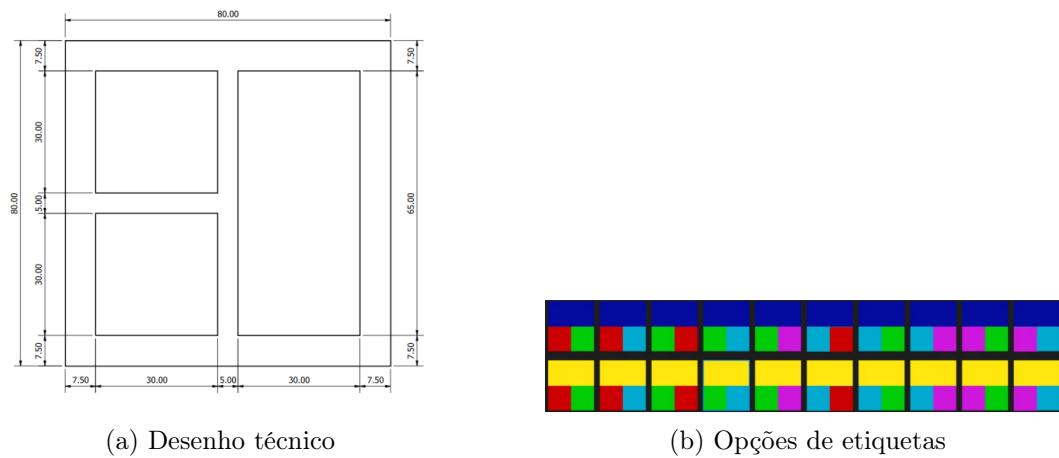


Figura 22: Padrão de etiquetas LARC/CBR VSSS 3x3 [1]



Figura 23: Etiquetas desenvolvidas para este trabalho

3.1.3 Recursos computacionais

A escolha da linguagem para o sistema de visão levou em consideração a disponibilidade de informações e documentação, o tempo médio de codificação (parâmetro humano) e o tempo de execução do código. Foram avaliados inicialmente duas linguagens: C++ e Python. Ao final, foi escolhido Python por ser mais fácil e rápido de codificar, possuir muita informação e documentação disponível e principalmente porque o tempo de execução foi similar a um programa equivalente em C++. Foram utilizadas as seguintes bibliotecas para o desenvolvimento deste trabalho:

- **OpenCV (*Open Source Computer Vision Library*):** Biblioteca multi-linguagem amplamente utilizada na área de visão computacional [16]. Ela possui as principais ferramentas para realizar o processamento de imagens;
- **Numpy:** Biblioteca de computação numérica para python [17]. Ela utilizada pacotes compilados em C e C++, fazendo com que seu tempo de processamento seja comparável a um programa escrito e compilado em C ou C++. Portanto, como o tempo de processamento é importante para o funcionamento dos algoritmos do futebol, o algoritmo foi escrito de tal forma que o máximo de operações numéricas fossem realizadas por funções da biblioteca numpy. Em testes iniciais os resultados obtidos com C++ usando OpenCV e Python usando OpenCV e numpy foram praticamente os mesmos;



(a) Numpy [17]



(b) OpenCV [16]

Figura 24: Logos das bibliotecas

3.1.4 O algoritmo de visão

Todo o algoritmo de visão foi desenvolvido em python, a seguir são apresentadas as etapas de execução e em seguida sua implementação.

1. **Captura:** captura dos quadros da câmera com maior frequência possível;
2. **Detecção das cores:** gera uma máscara para cada cor;
3. **Detecção de contornos:** detecção dos contornos das entidades;
4. **Segmentação:** analise das informações dos contornos e das máscaras para identificar as etiquetas dos robôs e a bola;
5. **Encapsulamento:** ao final as informações são encapsuladas em um dicionário de python.

3.1.4.1 Captura

A captura é o processo de aquisição de imagens. Do ponto de vista do software é desejável que o tempo de aquisição de um novo quadro seja o menor possível. Este tempo varia em função do protocolo de comunicação entre a câmera e o computador e também em função do tempo que a tecnologia da câmera permite obter novos quadros.

A aquisição de um quadro pela câmera não ocorre de forma imediata. Esse processo leva alguns milissegundos. Intervalo no qual o programa poderia processar outras informações. Para solucionar este problema foi desenvolvido um algoritmo *multithread*.

Uma *thread* é um pedaço de código que pode ser executado em paralelo com outras *threads*. Na prática este processamento não é de fato paralelo, pois um mesmo processador executa varias *threads*, mas uma de cada vez, porém, como ele realiza constantemente a comutação entre *threads*, aparentemente elas estão sendo executadas paralelamente ao longo do tempo. Em python o programa em execução é uma *thread*, sendo possível criar outras. A solução desenvolvida para o intervalo de aquisição da câmera utiliza uma classe de captura que executa esta tarefa em uma *thread* separada da principal. A integração entre as *threads* é realizada por intermédio de uma *flag* que sinaliza com valor booleano "verdadeiro" se uma nova imagem está disponível. Quando a *thread* principal detecta o valor "verdadeiro" na *flag*, ela realiza uma cópia da imagem captada e altera a *flag* para o valor booleano "falso". Ao final, enquanto o processamento de um quadro é realizado na *thread* principal, a aquisição do próximo quadro está sendo realizado na *thread* de captura, poupando tempo de processamento. A Figura 25 ilustra este processo em um fluxograma.

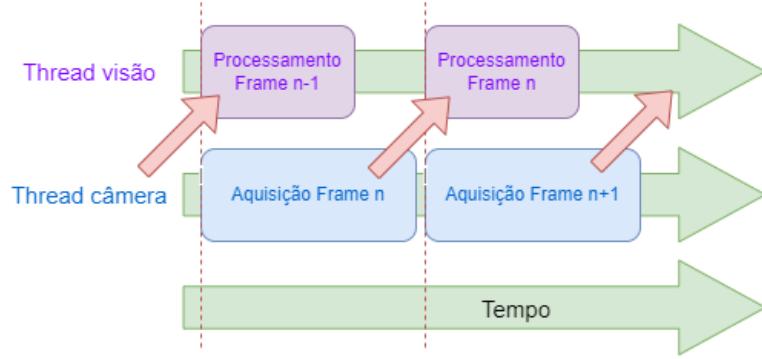


Figura 25: Fluxograma do processamento dos frames. Fonte: o autor

A Figura 26 exibe a tela com a captura da câmera, e as indicações da bola e dos jogadores. Em verde pode-se observar a frequência de aquisição em FPS, a qual varia constantemente pois depende do processamento de outras *threads*, mas tipicamente oscila em torno de 30 FPS, podendo variar entre 21 e 37 FPS.

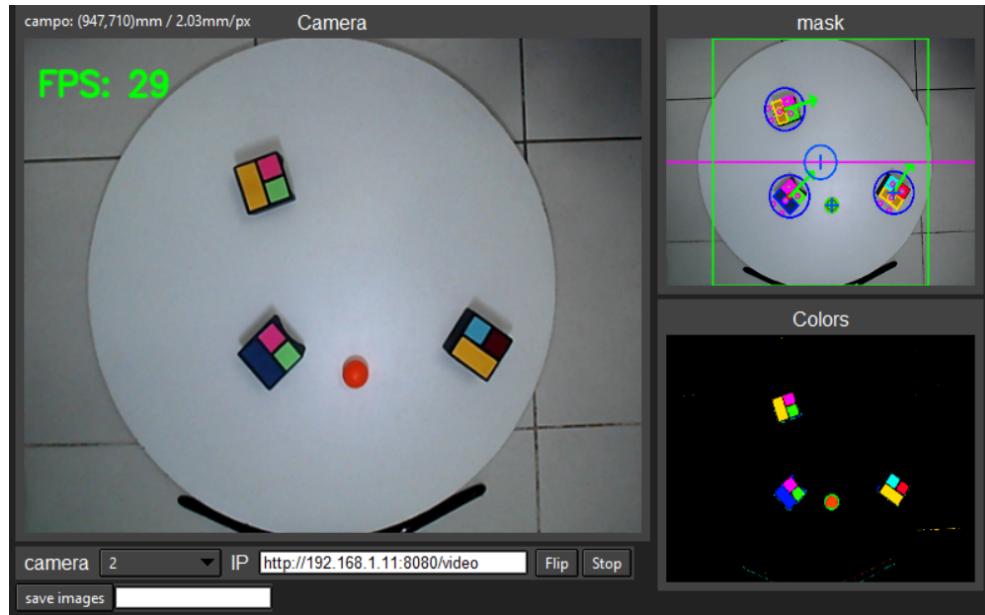


Figura 26: Exemplo de captura com a visão. Fonte: o autor

3.1.4.2 Detecção das cores

No sistema de visão a identificação dos jogadores e da bola é realizada através da detecção das regiões em que ocorrem cada cor. Apesar de poderem ocorrer até 8 cores diferentes no futebol de robôs nem todas precisam ser computadas. O preto e o branco não precisam ser detectados pois são usados em todo campo. A cor laranja é usado apenas

na bola portanto, sua detecção permite localizar a bola. As cores azul e amarelo permitem localizar e identificar de qual time é cada robô além de servirem como referência para a segmentação das cores adjacentes que podem ser: vermelho, azul claro, verde ou rosa. Estas ultimas servem pra distinguir os jogadores e ajudam a na medição da orientação dos robôs.

A função de captura retorna um arranjo de dimensão w por h , com 3 camadas de cores, uma para cada cor do sistema RGB, ou seja, vermelho, verde e azul. Pode-se definir uma determinada cor como uma região no espaço de cores RGB, sendo assim, é possível definir uma função matemática que recebe como entrada os níveis R, G e B e retorna verdadeiro ou falso quanto a pertinência do pixel a cor avaliada.

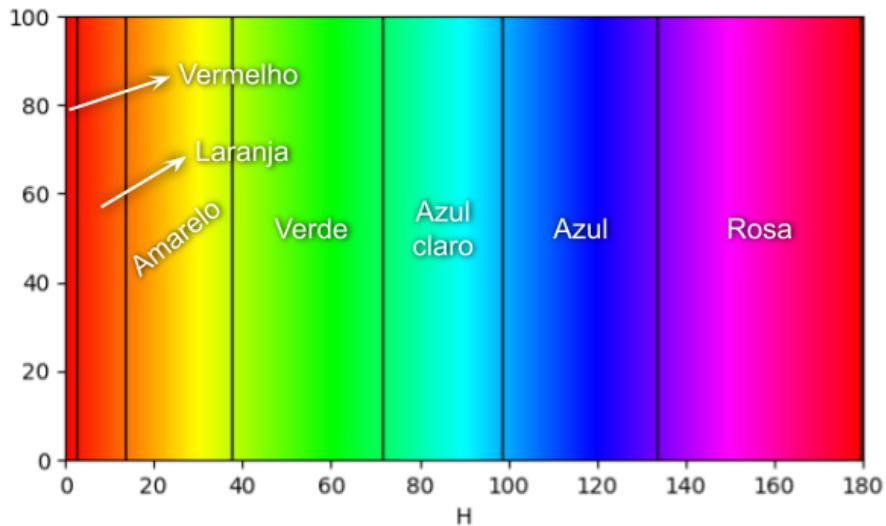


Figura 27: Regiões das cores ao longo de H. Fonte: o autor

O mesmo procedimento pode ser realizado no sistema de cores HSV (*Hue Saturation Value*), que foi o adotado neste projeto durante a detecção das cores. Este sistema é composto por 3 canais, dos quais o H (Matiz) esta diretamente relacionado a informação do que o sistema visual humano comprehende como cor ou tonalidade, o canal S (Saturação) relaciona o quanto de tonalidade cinza a cor contém e o V (Valor) esta relacionado ao brilho.

A detecção de cores em uma imagem usando OpenCV é realizada usando a função *inRange*, que recebe como argumentos uma imagem e os intervalos de mínimo e de máximo de cada canal (neste caso: $(H_{min}, S_{min}, V_{min})$ e $(H_{max}, S_{max}, V_{max})$). Esta função retorna uma máscara, ou seja, um novo arranjo booleano com as mesmas dimensões da imagem

original onde cada elemento deste novo arranjo indica se o pixel equivalente está contido no intervalo ou não, ou seja, se é ou não da cor avaliada.

A determinação dos intervalos para cada cor são fundamentais para o bom funcionamento da detecção. Neste trabalho eles são determinados empiricamente, e devem ser ajustados em função das condições de luminosidade do local, tonalidade das cores das etiquetas e da bola, além do tipo de câmera utilizada (câmeras diferentes detectam cores de forma diferente, afetando os intervalos). No algoritmo de visão desenvolvido, o nível S_{max} e V_{max} são os máximos possíveis: 255 no OpenCV. Por simplicidade S_{min} e V_{min} são os mesmos para todas as cores, e podem ser ajustados pelo usuário em função das condições de luminosidade do local. Os valores de H_{min} e H_{max} são personalizados em função da tonalidade dos elementos do campo e do tipo de câmera. A Figura 27 exibe o intervalo de cada cor indo do mínimo para o máximo, da esquerda pra direita. Os intervalos foram determinado de tal forma que o fim de um é o inicio do seguinte, esta abordagem reduz o número de parâmetros. A Figura 30 exibe um exemplo da detecção de cores. A Figura a esquerda exibe a imagem original, enquanto a Figura a direita foi gerada a partir da sobreposição das máscaras de cada cor com suas respectivas tonalidades.

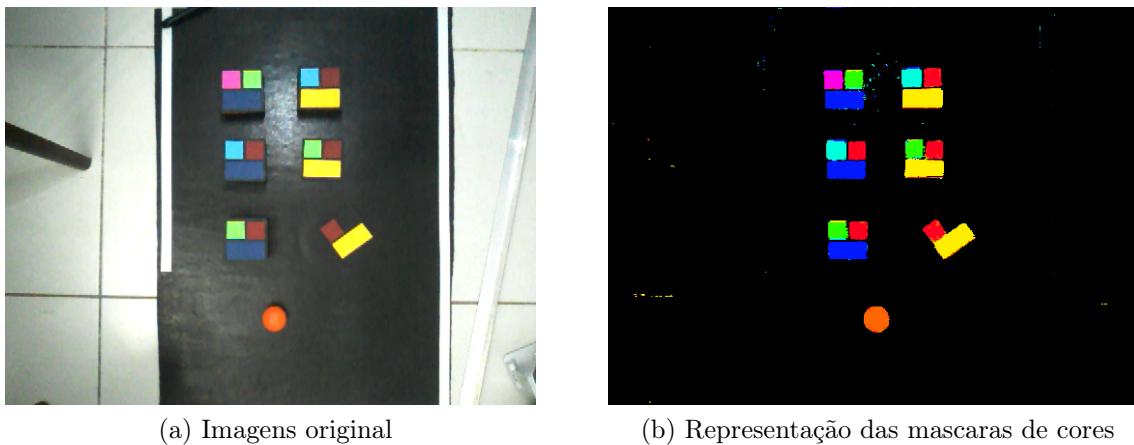


Figura 28: Exemplo de detecção de cores. Fonte: o autor

3.1.4.3 Segmentação e detecção de contornos

Em visão computacional, a segmentação é o processo de identificar regiões de *pixels* com propriedades em comum. Neste esse processo é fundamental para identificar os elementos do jogo (a bola e os robôs).

O método de segmentação utilizado neste trabalho é a detecção de contornos. Os

contornos podem ser descritos, neste caso, como a fronteira de uma região que possui uma mesma cor. Em *OpenCV* esse processo é realizado com a função *findContours* que recebe uma máscara e alguns parâmetros referentes ao método utilizado, e ao final retorna uma lista com os contornos. Em seguida, os contornos são ordenados em função da área em uma lista, sendo descartados aqueles com área inferior a um valor mínimo, pois se tratam provavelmente de ruídos. O valor mínimo de área para cada cor, pode ser ajustado pelo usuário na interface gráfica.

3.1.4.4 Posição e orientação

Uma vez identificados os contornos é preciso obter as informações relevantes para o projeto. Isso foi realizado aproximando a forma de cada contorno por uma figura conhecida. A bola é aproximada por uma circunferência usando a função de *OpenCV* *minEnclosingCircle*, que retorna o centro e o raio da circunferência que melhor se encaixa. As etiquetas são aproximadas por retângulos usando a função *minAreaRect* que retorna os vértices do retângulo. O centro dos retângulos são obtidos pela média das coordenadas dos vértices, como forme a Equação (2).

$$\vec{P}_{centro} = \sum_i \frac{(x_i, y_i)}{4} \quad (2)$$

A posição da bola é dada pelo centro da circunferência que aproxima seu contorno. A posição e orientação dos robôs é mais complicada, pois depende de 3 etiquetas, a seguir são listados os passos para identificá-los.

1. São analisadas as etiquetas que identificam os times primeiro, no caso, as 3 figuras com maior área de cada cor, azul e amarelo.
2. Com base nas dimensões da etiqueta do time, o algoritmo estima as posições onde as outras etiquetas menores devem estar.
3. Em seguida, ao redor de cada etiqueta de um time, o algoritmo busca outras etiquetas menores na cor azul claro, vermelho, verde ou rosa, que estejam próximas das posição estimada. O conjunto final de etiquetas é aquele que obtiver o menor erro total entre a posição real e a estimada. Além disso o erro precisa ser menor que um limiar, se não o em caso de falha na aquisição de uma etiqueta o algorítimo poderia

usar uma etiqueta muito distante como sendo pertencente a um determinado robô, aumentando muito o erro.

4. **Cálculo do centro de um robô:** O centro de um robô é calculado pela Equação (3). Onde o T_1 é a coordenada do centro etiqueta do time e E_1 e E_2 são os centros das outras etiquetas do robô. Neste projeto as etiquetas menores são colocadas do lado da frente do robô. Então a orientação é o angulo θ_{robo} dado pelo vetor diretor \hat{v}_{robo} da distância entre o centro da etiqueta do time e o ponto médio entre o centro das etiquetas menores, de acordo com a Equação (4), enquanto a Equação (5) descreve o calculo da orientação do robô.

$$\vec{P}_{robo} = \frac{2\vec{T}_1 + \vec{E}_1 + \vec{E}_2}{4} \quad (3)$$

$$\hat{v}_{robo} = \frac{\vec{E}_1 + \vec{E}_2 - 2\vec{T}_1}{|\vec{E}_1 + \vec{E}_2 - 2\vec{T}_1|} \quad (4)$$

$$\theta_{robo} = \arccos(\hat{v}_{robo} \cdot \mathbf{i}) \quad (5)$$

5. No caso de nenhum conjunto de etiquetas atender ao limiar de erro para um determinado robô seu centro é considerado como sendo o da etiqueta do time e sua orientação, é calculada através da posição dos vértices do retângulo que aproxima a etiqueta do time.

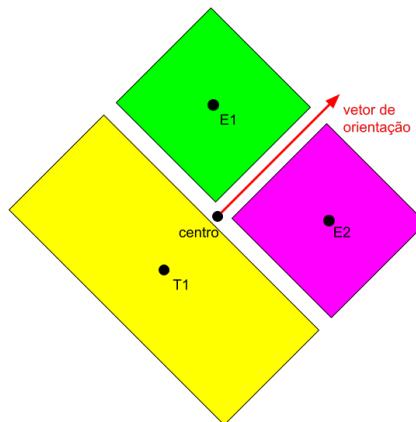


Figura 29: Representação gráfica dos centros das etiquetas e do robô. Fonte: o autor

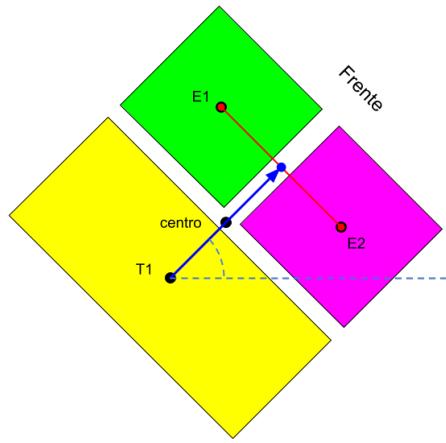


Figura 30: Representação gráfica do vetor distância e dos centros das etiquetas. Fonte: o autor

3.1.4.5 Encapsulamento

Todo o algoritmo de visão é realizado em um módulo de python, como se fosse uma biblioteca. O processamento de um quadro é realizado passando para a função *vision* do módulo desenvolvido o quadro e os parâmetros a serem utilizados no processamento, como: área mínima, erro mínimo, intervalo das cores etc. Ao final, esta função retorna os dados obtidos em formato de um dicionário. A estrutura de organização dos dados foi inspirado em outros programas de visão computacional para futebol de robôs [18].

```
ball_detection = {
    'ok': False,
    'pos': 0,
    'dimension': 0
}

bot_detection = {
    'id': 0,
    'pos': 0,
    'orientation': 0,
    'dimension': 0,
    'vector': 0,
    'colors': ['orange', 'orange']
}

game = {
    'ball': ball_detection,
    'team_blue': {},
    'team_yellow': {}
}
```

Figura 31: Dicionário de python gerado pelo sistema de visão. Fonte: o autor

3.2 Interface gráfica

A função da interface gráfica é exibir as informações do jogo, tais como: a imagem captada pela câmera, orientação e posição dos robôs e da bola, bem como permitir ajustes de parâmetros dos sistemas de visão e controle, receber os comandos de controle do jogo e comandos de teste. A interface desenvolvida é apresentada na Figura 32.

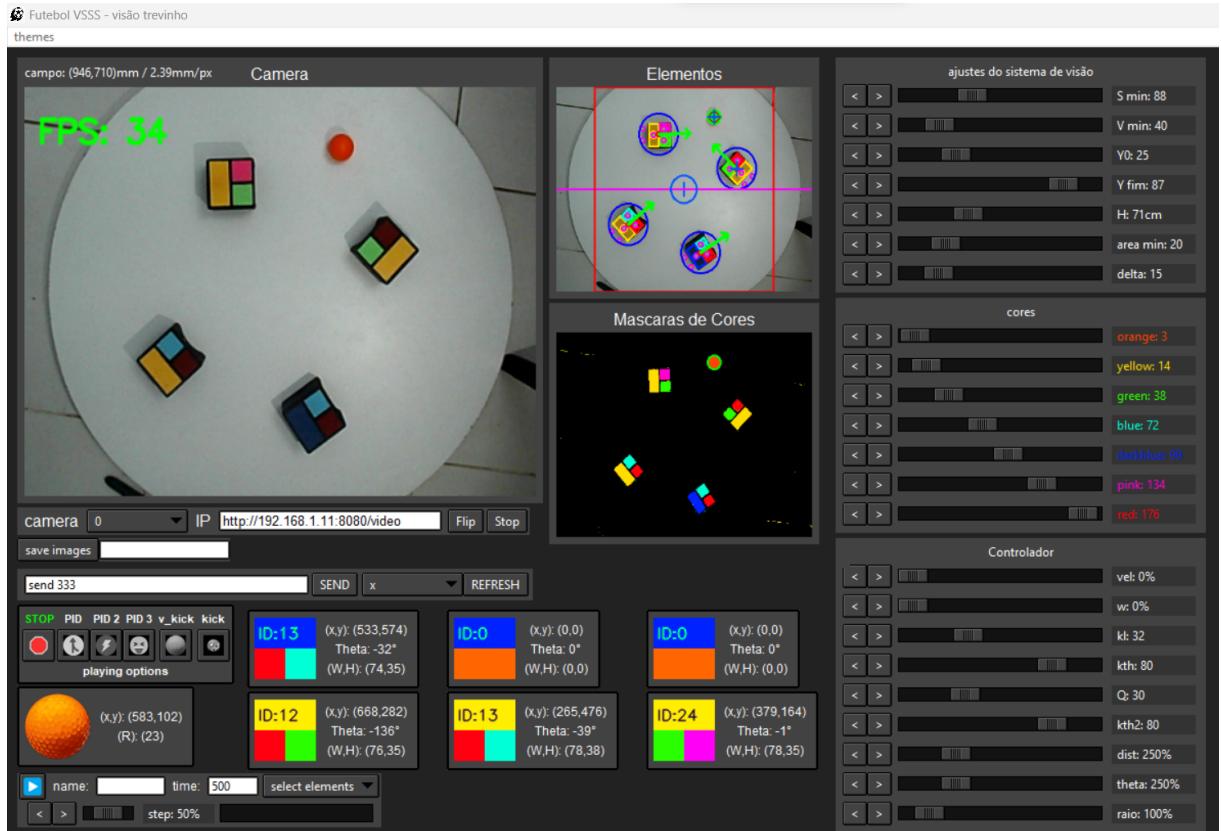


Figura 32: Interface gráfica do projeto. Fonte: o autor

3.2.1 TKinter

Neste projeto é utilizado a biblioteca para python TKinter [19] que facilita o desenvolvimento de *GUI* (*Graphical User Interface* - Interface gráfica do usuário). Esta biblioteca contém funções para criação de janelas com temas diversos e possui varias opções de *widgets* personalizáveis para montar uma interface.

3.2.2 Elementos da GUI

No intuito de facilitar o desenvolvimento de interfaces gráficas foram desenvolvidas algumas classes de elementos gráficos com base no TKinter. Todas elas foram armazenadas em um módulo chamado *widgets* permitindo usa-las como classes de uma biblioteca nos outros códigos. A seguir são descritos os elementos da interface gráfica principal, que esta contida no arquivo "main_{ui}.py".

3.2.2.1 Painel de captura

O painel de captura exibe em tempo real os quadros captados pela câmera e permite escolher qual câmera que é utilizada, sendo possível até mesmo usar uma câmera IP. Também é possível salvar o quadro atual, espelhar a imagem usando o botão *flip* e pausar a captura usando o botão *stop*. Na imagem é adicionada a frequência de captura.



Figura 33: Painel de captura. Fonte: o autor

3.2.2.2 Monitores

Os monitores exibem de forma gráfica os dados obtidos pelo sistema de visão, como por exemplo, as máscaras de cada cor, circunferência marcando os locais onde estão os robôs e as bolas e setas que indicam a orientação de cada robô.

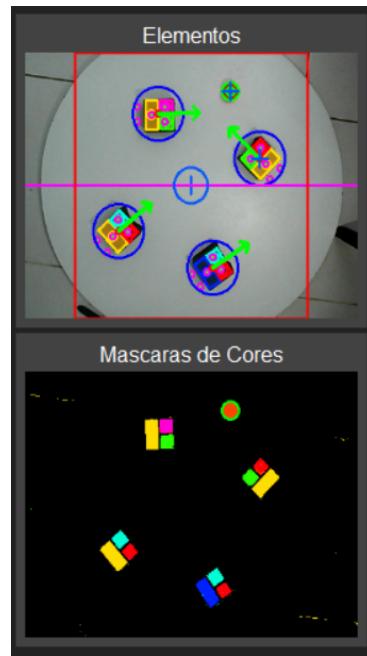


Figura 34: Monitores do sistema de visão. Fonte: o autor

3.2.2.3 Monitores da bola e jogadores

Os monitores dos jogadores exibem a posição de cada jogador em milímetros em relação ao topo esquerdo da imagem, além da orientação em graus e as dimensões de cada jogador. Os monitor da bola exibe a posição dela em milímetros em relação ao topo esquerdo da imagem, assim como os jogadores, e seu raio em milímetros.

ID:13	(x,y): (533,574)	Theta: -32°	(W,H): (74,35)
ID:0	(x,y): (0,0)	Theta: 0°	(W,H): (0,0)
ID:0	(x,y): (0,0)	Theta: 0°	(W,H): (0,0)
ID:12	(x,y): (668,282)	Theta: -136°	(W,H): (76,35)
ID:13	(x,y): (265,476)	Theta: -39°	(W,H): (78,38)
ID:24	(x,y): (379,164)	Theta: -1°	(W,H): (78,35)

Figura 35: Monitores dos jogadores. Fonte: o autor



Figura 36: Monitor da bola. Fonte: o autor

3.2.2.4 Conexão serial

O painel de conexão serial permite escolher em qual porta COM o dispositivo *Host* está conectado, também é possível enviar comandos inseridos pelo usuário. O *baudrate* é fixo e definido pelo *host*, neste projeto 115200.



Figura 37: painel de conexão serial. Fonte: o autor

3.2.2.5 Painéis de ajuste de parâmetros

Estes painéis permitem ajustar em tempo real os parâmetros do sistema de visão e parâmetros de controle de trajetória dos robôs. São fundamentais para realizar os ajustes de forma rápida antes de cada partida. É muito comum precisar alterar os parâmetros relacionados a interpretação das cores, uma vez que a iluminação do local pode alterá-los.

3.2.2.6 Painel de controle do jogo e testes

O painel de controle do jogo e testes, como o nome já diz, permite alterar o modo de jogo e também é possui modos usados para realizar testes, como por exemplo o modo *kick* que testa o chute.



Figura 38: Painéis de ajuste de parâmetros. Fonte: o autor



Figura 39: Painel de controle do jogo e testes. Fonte: o autor

4 MODELAGEM MATEMÁTICA

A modelagem matemática é a técnica que busca utilizar modelos matemáticos para descrever o comportamento de sistemas. Neste trabalho são estudados modelos para representar o sistema de locomoção dos robôs tendo como entradas as variáveis de controle que são as tensões nos motores. Neste capítulo, é apresentado inicialmente a modelagem cinemática do robô e em seguida o modelo dinâmico.

4.1 Modelagem cinemática e locomoção diferencial

A modalidade de locomoção estudada é a diferencial, ela é composta por dois motores dispostos em paralelo e concêntricos em relação aos eixos. A roda da direita possui velocidade v_r e a roda da esquerda possui velocidade v_l . De forma intuitiva, é possível observar que quando ambas as velocidades são iguais e positivas o robô de desloca para frente em linha reta, no entanto, quando as velocidades são diferentes ainda pode ocorrer deslocamento, porém com trajetórias circulares. A diferença de velocidade entre as rodas proporciona rotação.

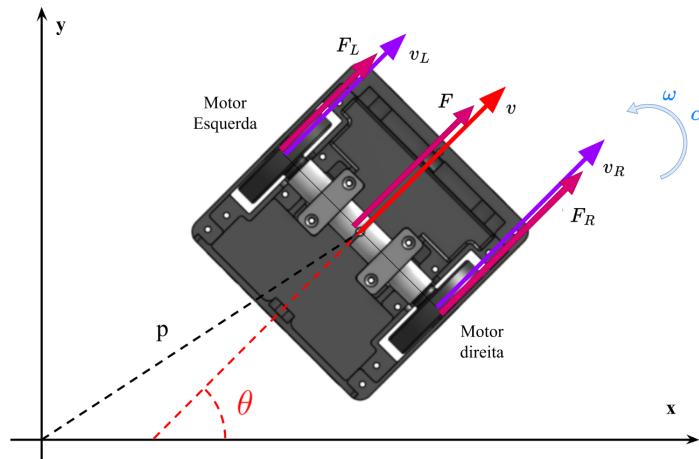


Figura 40: Modalidade de locomoção diferencial

4.1.1 Cinemática direta

No caso do robô diferencial a cinemática direta informa a velocidade angular e tangencial em função da velocidades de cada roda [20] [21]. Também é possível obter por

integração no tempo a pose final do robô (posição (x,y) e orientação θ). Na locomoção diferencial, quando as rodas têm velocidades iguais a velocidade tangencial é a mesma de cada roda, porém quando as velocidades são diferentes, a diferença obriga a roda mais rápida a alterar a trajetória do robô para o lado da roda mais lenta, descrevendo um arco de circunferência. A velocidade tangencial, v , é obtida através da média entre as velocidades de cada roda, v_{dir} e v_{esq} . A velocidade angular surge por conta da diferença de velocidade entre as rodas, como se a roda mais rápida girasse em torno da roda mais lenta, desta forma ela é calculada como a diferença entre a velocidade direita e esquerda das rodas, dividido pela distância entre as rodas, l .

$$v(t) = \frac{v_{dir}(t) + v_{esq}(t)}{2} \quad (6)$$

$$\omega(t) = \frac{v_{dir}(t) - v_{esq}(t)}{l} \quad (7)$$

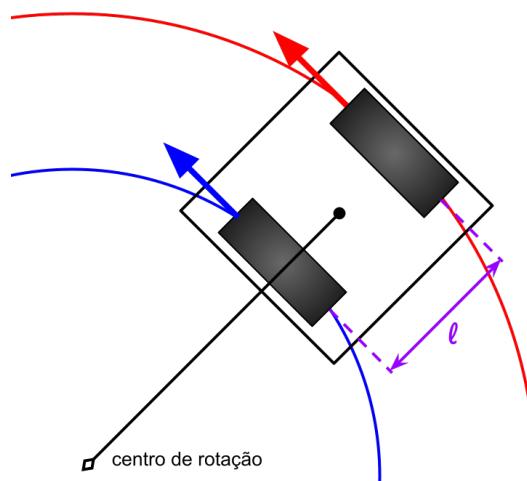


Figura 41: Representação do centro de rotação e da trajetória de cara roda em azul e vermelho

4.1.2 Cinemática inversa

Na cinemática inversa busca-se obter os sinais que devem ser aplicados nos motores em função das velocidades angular e tangencial desejadas. Isso é realizado manipulando as Equações (6) e (7) para isolar as velocidades de cada roda, resultando nas Equações (9) e (8). O modelo do motor depende da velocidade angular das rodas e é estudado a

diante no domínio de Laplace, por isso essas transformações foram realizadas obtendo as Equações (10) e (11). A Figura 42 representa o modelo cinemático inverso em diagrama em blocos.

$$v_{dir}(t) = v(t) + \frac{l}{2}\omega(t) \quad (8)$$

$$v_{esq}(t) = v(t) - \frac{l}{2}\omega(t) \quad (9)$$

$$\Omega_{dir}(s) = \frac{1}{r}V(s) + \frac{l}{2r}\Omega(s) \quad (10)$$

$$\Omega_{esq}(s) = \frac{1}{r}V(s) - \frac{l}{2r}\Omega(s) \quad (11)$$

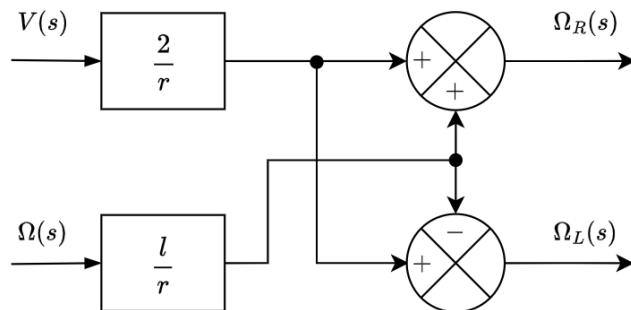


Figura 42: Diagrama em blocos da cinemática inversa do robô

4.2 Modelagem do motor

Os motores aplicados neste projeto pequenos motores de corrente contínua escovados de ímãs permanentes com uma micro-redução na saída. Motores CC podem ser descritos em forma de circuito como é exposto na figura 48. Aplicando as leis de Kirchhoff é obtida a equação (12). Os parâmetros R_A e L_A representam respectivamente, as perdas resistivas e a indutância natural gerada pelo enrolamento de armadura do motor. O terceiro e ultimo elemento do circuito representa a tensão induzida na armadura devido a rotação do rotor.

$$V_A(t) = R_A i(t) + L_A \frac{di(t)}{dt} + \omega(t)k \quad (12)$$

$$T(t) = k i(t) \quad (13)$$

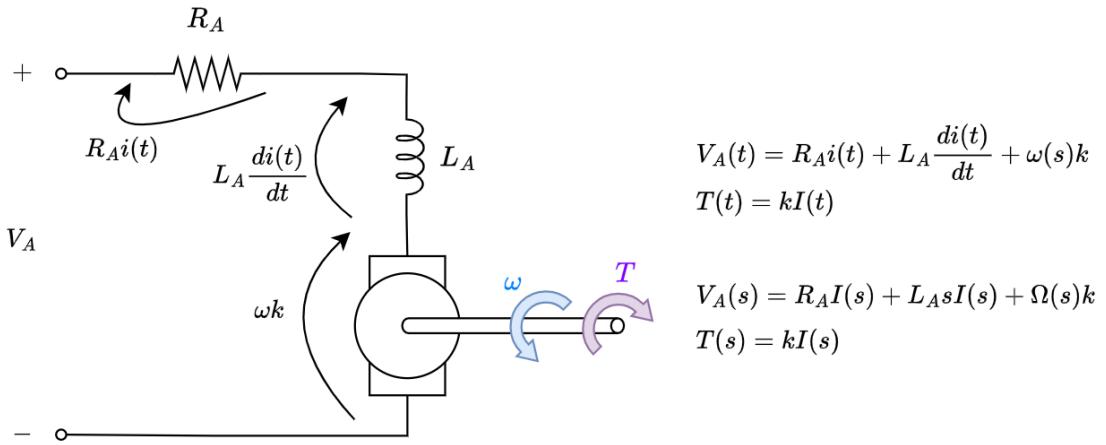


Figura 43: Modelagem do motor representado em forma de circuito elétrico

Sabe-se que em condições normais de operação, a tensão induzida é diretamente proporcional a velocidade angular do rotor ω , onde a razão é representada pela constante k do motor [22]. A constante k depende de parâmetros de construção dos motores, como: o material do núcleo da armadura, dimensões do motor, intensidade de campo do ímã e número de espiras da armadura. Não por acaso, a constante k também representa outra relação importante do motor, a razão entre o torque e a corrente consumida. Tal fato pode ser demonstrado analisando a construção do motor ou aplicando a lei da conservação de potência no terceiro elemento do circuito. A potência mecânica consumida na saída é dada por ωT , enquanto a potência elétrica recebida é $i e_{ind} = i \omega k$, desta forma $i \omega k = \omega T$, organizando a equação resulta na segunda equação para o motor apresentada em (13).

Aplicando transformada de Laplace nas Equações (12) e (13) são obtidas as Equações (14) e (15). Em seguida, as equações foram organizadas na forma de diagramas em blocos, exposto na Figura 44

$$V_A(s) = R_A I(s) + L_A s I(s) + \Omega(s)k \quad (14)$$

$$T(s) = kI(s) \quad (15)$$

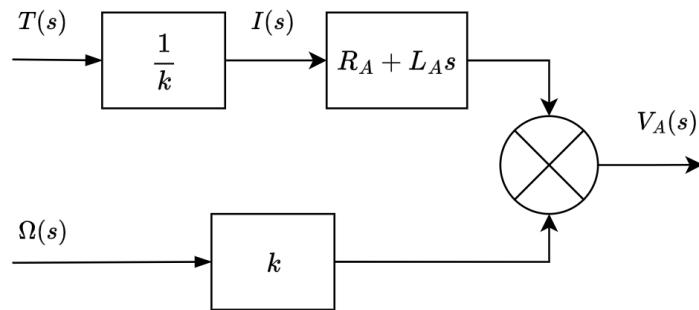


Figura 44: Diagrama em blocos do motor

4.3 Modelo dinâmico do robô

Nesta seção é desenvolvido um modelo matemático que relaciona a velocidade de deslocamento v e a velocidade angular ω , com os respectivos torques e velocidades angulares de cada motor. Inicialmente é aplicado o somatório de forças e momentos no robô, obtendo as Equações (16) e (17)

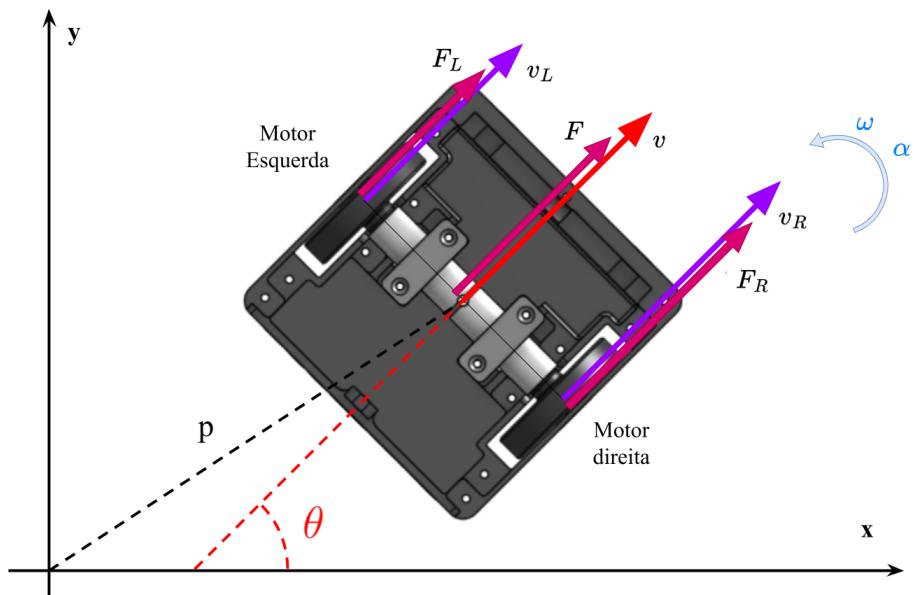


Figura 45: Representação gráfica das grandezas estudadas. Fonte: o autor.

$$ma = F_{dir} + F_{esq} \quad (16)$$

$$J\alpha = 2l(F_{dir} - F_{esq}) \quad (17)$$

Aplicando a transformada de Laplace:

$$msV(s) = F_{dir}(s) + F_{esq}(s) \quad (18)$$

$$Js\Omega(s) = l(F_{dir}(s) - F_{esq}(s)) \quad (19)$$

Em seguida, manipulando o sistema se obtém $F_{esq}(s)$ e $F_{dir}(s)$ em função de $\Omega(s)$ e $V(s)$:

$$F_{dir}(s) = \frac{s(mV(s) + \frac{J}{l}\Omega(s))}{2} \quad (20)$$

$$F_{esq}(s) = \frac{s(mV(s) - \frac{J}{l}\Omega(s))}{2} \quad (21)$$

Por ultimo, são obtidos os torques nos motores, os quais resultam da multiplicação da força na roda pelo seu raio r . As Equações (22) e (23) apresentam a formulação final para o torque em cada motor, enquanto a Figura 46 representa esta relação na forma de diagrama de blocos.

$$T_{dir}(s) = \frac{sr(mV(s) + \frac{J}{l}\Omega(s))}{2} \quad (22)$$

$$T_{esq}(s) = \frac{sr(mV(s) - \frac{J}{l}\Omega(s))}{2} \quad (23)$$

O diagrama completo é obtido pela união dos diagramas das Figuras 42, 44 e 46, que representam respectivamente o modelo cinemático inverso do robô, o modelo do motor e a os torques nas rodas em função de $V(s)$ e $\Omega(s)$. O digrama completo é demonstrado na Figura 47.

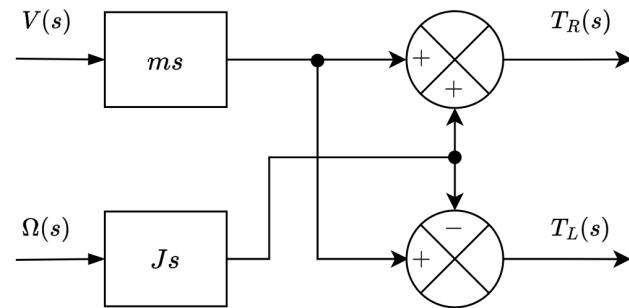


Figura 46: Diagrama dos torques em função das velocidades angular e tangencial do robô

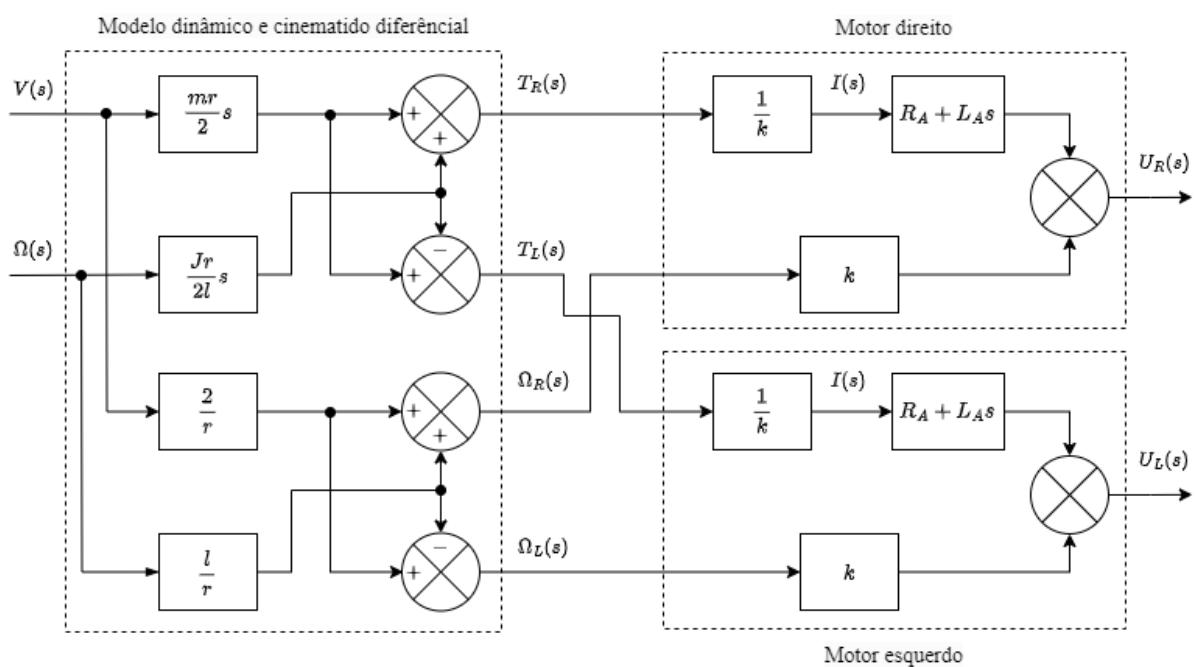


Figura 47: Diagrama do modelo completo do robô

5 CONTROLADOR DE VELOCIDADE ANGULAR

Como pode-se concluir do capítulo de modelagem, vários fatores podem acarretar discrepâncias entre as velocidades estimadas pelo modelo cinâmético e o obtido na prática, dentre ela a inércia do robô, desalinhamento do centro de massa em relação ao centro de rotação ou pequenas diferenças de parâmetros entre os motores, as rodas ou até mesmo na superfície. Esses efeitos podem comprometer totalmente o controlador de trajetória, ou piorar seu desempenho, uma vez que ele funciona com frequência limitada através do sistema de visão. Para melhorar o desempenho dos robôs é recomendável o uso de um controlador em cada robô que regule a velocidade angular.

Neste trabalho é utilizado um controlador PID que recebe como *setpoint* o valor desejado de velocidade angular em rad/s. A medição do erro é realizada pela diferença entre o *setpoint* e a velocidade angular medida pelo giroscópio da *IMU* de cada robô. O controlador é processado localmente no microcontrolador de cada robô.

5.1 Calibração do giroscópio

O sinal de velocidade angular fornecido pelo giroscópio precisa ser filtrado pois é muito suscetível a ruídos. Essa filtragem é realizada pela Equação (24). O coeficiente ω_0 foram obtidos através de experimentação, fazendo leituras consecutivas com o robô parado e obtendo o valor médio. Além da Equação (24), caso o módulo de $\omega_{filtrado}$ seja inferior a um limiar ele é considerado nulo, esse valor foi definido empiricamente como 0,02 rad/s.

$$\omega_{filtrado} = \omega_g - \omega_0 \quad (24)$$

5.2 Controle Proporcional Integral Derivativo - PID

O controle proporcional-integral-derivativo é uma das técnicas mais utilizadas no setor industrial. Trata-se de uma técnica de fácil implementação e baixo custo com capacidade de compensar grande parte dos processos industriais práticos [23], eliminando o erro em regime permanente com respostas transitórias satisfatórias [24].

O controle PID é composto pela soma de três ações de controles independentes,

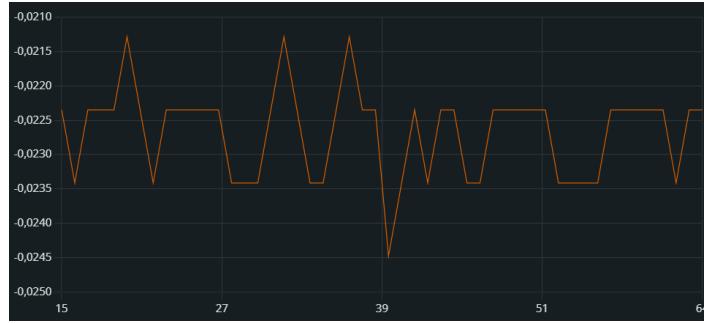


Figura 48: Medição da velocidade angular com o giroscópio ao longo do tempo, medições a cada 2 segundos.

denominadas Proporcional (P), Integral (I) e Derivativa (D), sendo os parâmetros de projeto os ganhos associados: K_p (proporcional), K_i (integral) e K_d (derivativo). A definição matemática do controlador PID é apresentada na Equação (25).

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (25)$$

Na qual: $u(t)$ é o sinal de controle, $e(t)$ é o erro entre o sinal de referência (do inglês, *setpoint*) e o sinal de saída do sistema (medido).

A ação proporcional (**P**) pode ser representada da seguinte forma:

$$u_p(t) = K_p e(t) \quad (26)$$

O principal objetivo da ação proporcional é estabilizar a planta, atuando para compensar o erro existente entre o valor de referência e o valor medido [25].

Quando se controla um sistema pela ação proporcional obtém-se os seguintes resultados: diminuição do erro ($e(t)$) e do tempo de subida (t_r), variação do tempo de acomodação (t_s) e aumento do sobrevalor percentual (do inglês, *overshoot*) [26].

A ação integral (**I**) age de maneira proporcional a integral do erro do sistema e pode ser representada da seguinte forma:

$$u(t) = K_i \int_0^t e(t) dt \quad (27)$$

Trata-se de uma ação complementar à proporcional, que almeja eliminar o erro de regime permanente que a ação proporcional é incapaz de diminuir [25].

Quando atua-se sobre um sistema com ação integral é possível eliminar o erro em

regime permanente, diminuir o tempo de subida, aumentar o tempo de acomodação e o *overshoot* [26].

A ação derivativa (**D**) é proporcional a derivada do erro e pode ser representada da seguinte forma:

$$u(t) = K_d \frac{de(t)}{dt} \quad (28)$$

A ação derivativa tem como objetivo melhorar o desempenho transitório do sistema em malha fechada. Ao se aplicar um sinal proporcional a derivada do erro, obtém-se a tendência de evolução do mesmo, o que faz a ação derivativa ser também conhecida como antecipativa ou preditiva e implica que o sistema tenha respostas mais rápidas [25].

Quando se aplica a ação derivativa em um sistema, tem-se pequena variação no tempo de subida, redução do *overshoot*, diminuição no tempo de estabilização e pequena variação do erro de regime permanente [26].

A Tabela 5 apresenta um resumo dos efeitos das ações de proporcional, integral e derivativa que compõem o controle PID.

Tabela 5: Características das ações de controle que compõe um controlador PID. $e(t)$ denota o erro de Regime Permanente, t_r o tempo de subida e t_s o tempo de acomodação.

Ação	$e(t)$	<i>Overshoot</i>	t_r	t_s
Proporcional	Diminui	Aumenta	Diminui	Muda pouco
Integral	Elimina	Aumenta	Diminui	Aumenta
Derivativa	Muda pouco	Diminui	Muda pouco	Diminui

Cabe observar que a ação derivativa aumenta o ruído na variável do processo, especialmente em altas frequências, que podem resultar em danos para o sistema [25]. Para reduzir esse problema, é projetado um filtro passa-baixas de primeira ordem a ser aplicado em conjunto à componente derivativa do controle PID. Assim, o controle PID toma uma nova forma apresentada na Equação (29).

$$u(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{\frac{K_d}{N} s + 1} \quad (29)$$

O parâmetro do filtro derivativo (N) é ajustável e assume tipicamente valores entre 1 e 33 [25].

5.2.1 Controlador PID discreto

O controlador PID utiliza a operação de derivada e integral, essas operações podem ser realizadas por circuito com amplificadores operacionais, no entanto, elas devem ser realizadas via programação em um microcontrolador. Pra isso essas funções devem ser aproximadas por integração e derivação numérica. A Equação (30) descreve a integração numérica e a Equação (31) descreve a derivação. Todos os valores com índice $n-1$ são armazenados em variáveis do tipo *float* que guardam o valor anterior para serem usadas no próximo ciclo. A Formulação final do controlador é descrita pela Equação (32).

$$I[n] = \Delta t e[n] + I[n - 1] \quad (30)$$

$$D[n] = \frac{e[n] + e[n - 1]}{\Delta t} \quad (31)$$

$$u[n] = k_p e[n] + k_i \Delta t e[n] + I[n - 1] + k_d \frac{e[n] + e[n - 1]}{\Delta t} \quad (32)$$

5.3 Atuador

O sinal obtido pelo controlador informa qual deve ser a diferença de velocidade aplicada nos motores, onde o objetivo final é definir o sinal *PWM* que deve ser aplicado em cada motor. Além da velocidade angular o controlador recebe do *Host* o sinal médio de *PWM* (valor que define a velocidade média de deslocamento do robô) com isso o valor final de cada motor é calculado de acordo com as Equações (33) e (34).

$$PWM_{dir} = PWM_{medio} + 0,5PWM_{pid} \quad (33)$$

$$PWM_{esq} = PWM_{medio} - 0,5PWM_{pid} \quad (34)$$

Os valores de PWM_{dir} e PWM_{esq} em módulo não podem ser superiores ao sinal máximo de *PWM*, 1023 neste caso, caso isso ocorra o controlador irá limitar o valor entre -1023 e 1023.

6 MOVIMENTAÇÃO AUTÔNOMA

Existem dois controladores atuando na movimentação dos robôs, o primeiro é realizado internamente de cada robô, fechando a malha de controle com a leitura de velocidade angular dos giroscópios, demonstrado no Capítulo 5. Este primeiro controlador reduz o erro de velocidade angular do robô e atua mais rapidamente que o segundo controlador. O segundo controlador atua no controle de trajetória de cada robô, pra isso ele utiliza o sistema de visão como sensor que indica a pose atual de cada robô e da bola.

6.1 Implementação usando controladores proporcionais

É esperado que o primeiro controlador reduza os erros causados pela dinâmica do robô, desta forma o segundo controlador foi projetado usando apenas controladores proporcionais. Eles são definidos pelas Equações (35) e (36). No qual, os ganhos k_v e k_a e os erros são definidos em função da técnica de controle de trajetória.

$$v = k_v error_d \quad (35)$$

$$\omega = k_{ang} error_{ang} \quad (36)$$

6.2 Planejamento de Trajetória

O planejamento de trajetória serve para definir o melhor caminho para um robô sair de um ponto e chegar em outro sem colidir com outros obstáculos. Existem varias formas de realizar este planejamento, neste trabalho é realizado através de campos vetoriais.

6.2.1 Campos vetoriais artificiais

A técnica de simular o comportamento de campos elétricos permite definir trajetórias que desviam dos obstáculos com baixo custo computacional comparado com outras técnicas como o *RRT* (*Rapidly exploring random tree*) [27]. Ela funciona gerando campos repulsivos em torno dos obstáculos e campo atrativos em torno do destino. Matematicamente esses campos são descritos por campos vetoriais, ou seja, para cada posição (x,y)

no espaço existe um vetor associado a este ponto. A Figura 49 ilustra um campo vetorial gerado pela interação de duas cargas de polaridades opostas.

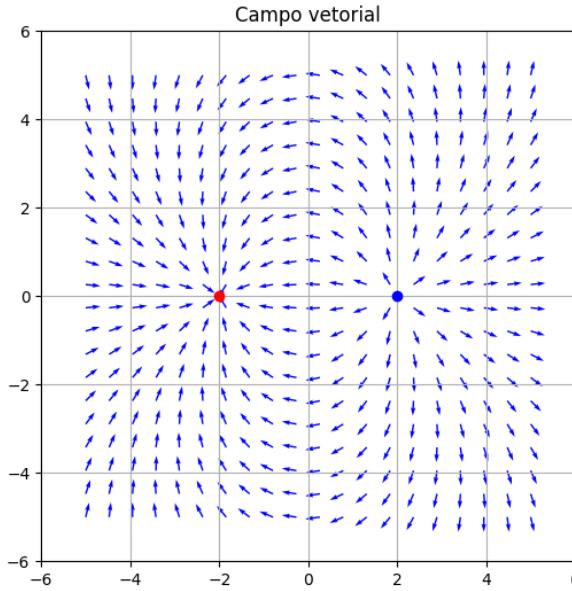


Figura 49: Campo vetorial gerado por duas cargas com polaridades opostas. Fonte: o autor.

A grande vantagem de um campo vetorial é que ele pode ser calculado de forma independente da pose anterior, só importando a posição atual, a posição dos outros obstáculos e a posição do destino. Representando os obstáculos como cargas negativas e o destino como carga positiva, se obtém a Equação (37) que descreve a força resultante em um ponto (x,y) , nesse caso a trajetória é definida pelo ângulo que o robô deve seguir em cada ponto (x,y) , indicado pela Equação (38).

$$\vec{F}(x, y) = \sum_{cargas} \frac{Q_i}{(x_i - x)^2 + (y_i - y)^2} \quad (37)$$

$$\theta = \arctg \left(\frac{F_x(x, y)}{F_y(x, y)} \right) \quad (38)$$

6.2.2 Implementação

O campo vetorial utilizado é descrito pela Equação (37), no qual o ângulo desejado para cada posição no espaço é calculado pela Equação (38). Na implementação final, o ângulo θ do campo vetorial é utilizado como referência do controlador proporcional de velocidade angular, descrito pela Equação (36). Enquanto a velocidade tangencial é defi-

nida pelo controlador proporcional descrito na Equação (35), no qual o erro é a distância do robô ao alvo. Os ganhos dos controladores k_{pl} e k_{pth} foram obtidos empiricamente.

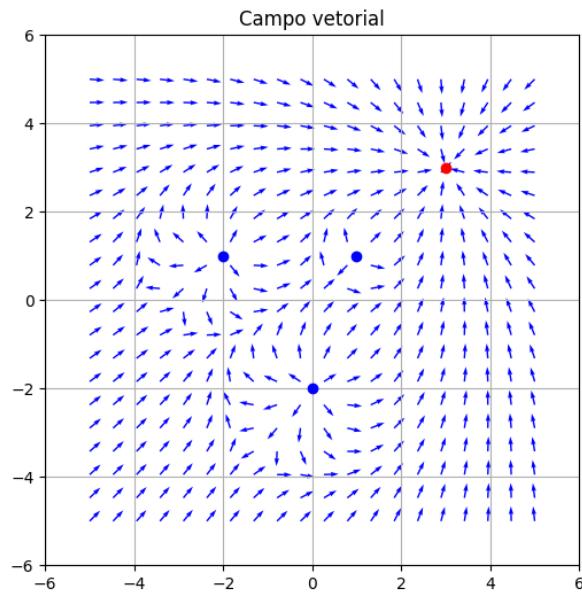
A implementação em código é descrita pelo Algoritmo 1. Ele contém uma descontinuidade no controlador de velocidade, quando o módulo do erro angular seja maior que 1 radiano, o robô apenas gira, sem se deslocar. Esta descontinuidade impede que o robô faça manobras muito bruscas [28]. A Figura 50b apresenta simulação do campo vetorial e das linhas de campo, sendo a bola representada em vermelho e os obstáculos (outros jogadores) em azul.

Algoritmo 1 Implementação do controlador de trajetória

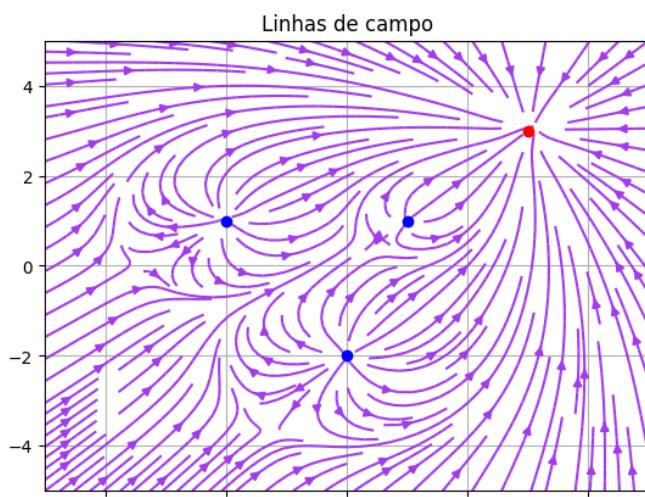
```

Distancia = 1000 // valor alto
enquanto Distancia < 50 faça
    robo_vetor = vetor_robo_amarelo() // vetor diretor
    robo = pos_robo_amarelo() // posição (x,y)
    bola = pos_ball() // posição (x,y)
    azuis = lista_pos_robo_azul() // lista todas as posições (x,y)
    vetor = vetor_campo( robo, bola, azuis ) // calcula o vetor resultante
    erro_a = defasagem( robo_vetor, vetor ) // defasagem angular entre os vetores
    w = erro_a*kp_ang
    Distancia = dist(robo,bola)
    vel = Distancia*kp_vel + 300 // 300 é o valor mínimo para movimentar
    se mod(erro_angulo) > 1 então
        vel = 0
    fim se
    envia_robo( vel, w ) // envia velocidade tangencial e angular
fim enquanto

```



(a) Campo vetorial



(b) Linhas de campo

Figura 50: Campo vetorial gerado, em azul os adversário e em vermelho a bola. Fonte: o autor.

7 VALIDAÇÃO EXPERIMENTAL

7.1 Sistema de visão computacional

Para validar o funcionamento do sistema de visão computacional foram realizadas 3 aferições em situações diferentes. A primeira com 3 robôs de cada time e a bola parados, a segunda com a bola em movimento e a terceira com um robô em movimento.

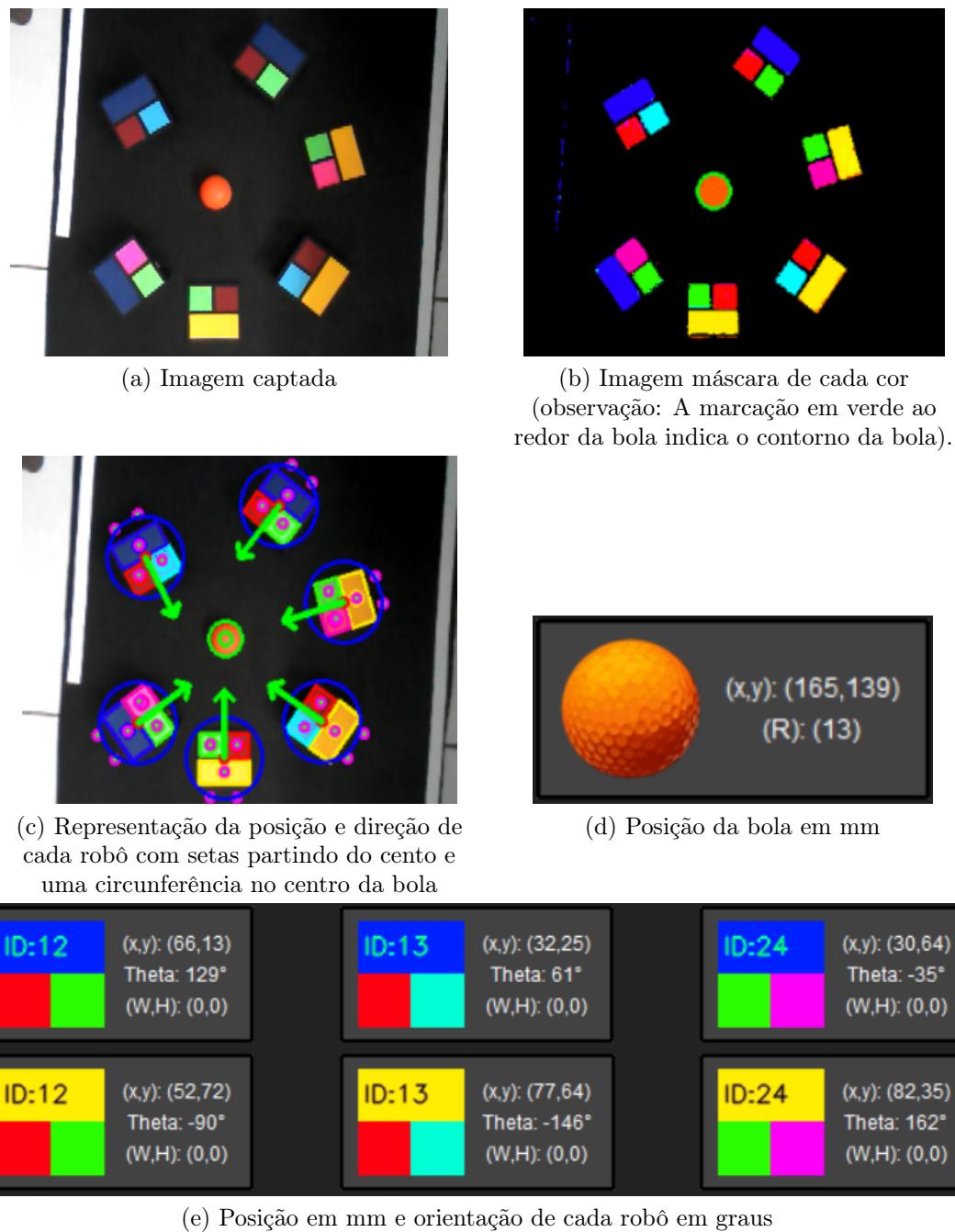


Figura 51: Primeira medição com os robôs e bola parados.

A primeira medição, apresentada na Figura 51 demonstra que o sistema de visão é capaz de identificar e segmentar os robôs e a bola. Enquanto, a segunda e a terceira medições, apresentadas nas Figuras 52 e 53, demonstram que o algoritmo funciona mesmo com os objetos em movimento, característica essencial para o projeto, uma vez que os robôs e a bola estão constantemente em movimento.

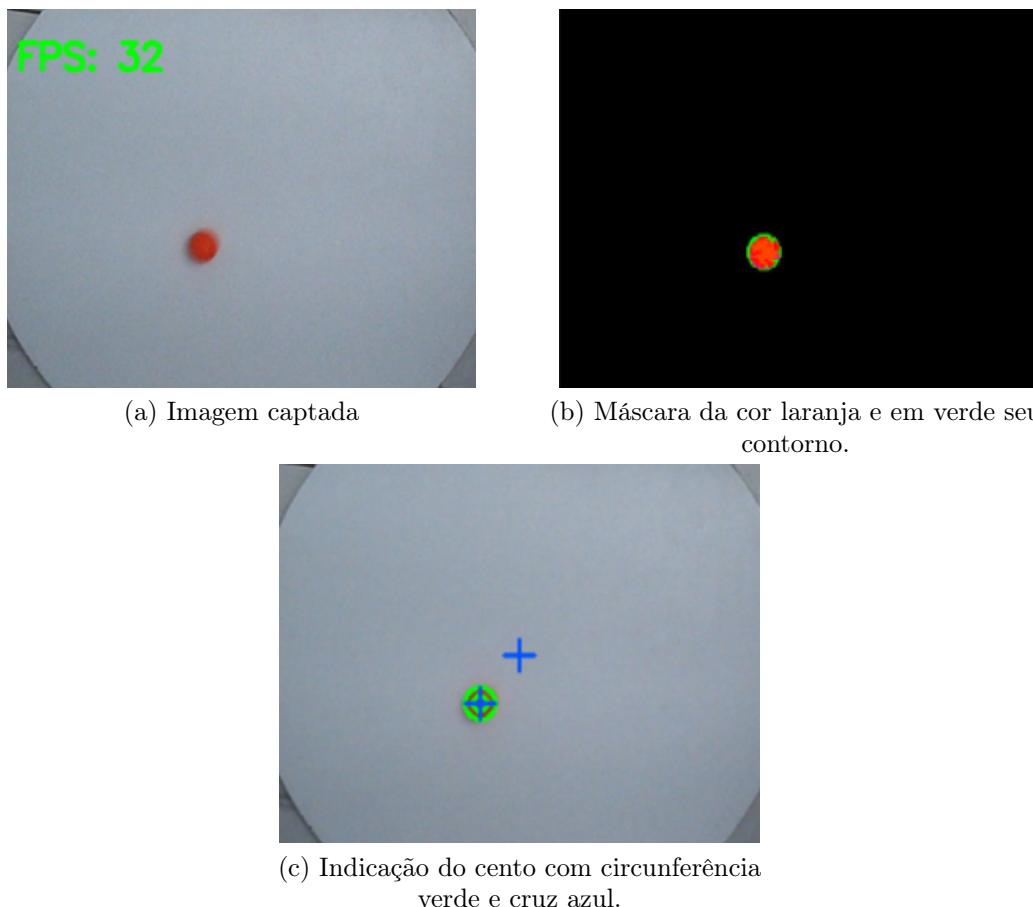


Figura 52: Segunda medição com a bola em movimento.

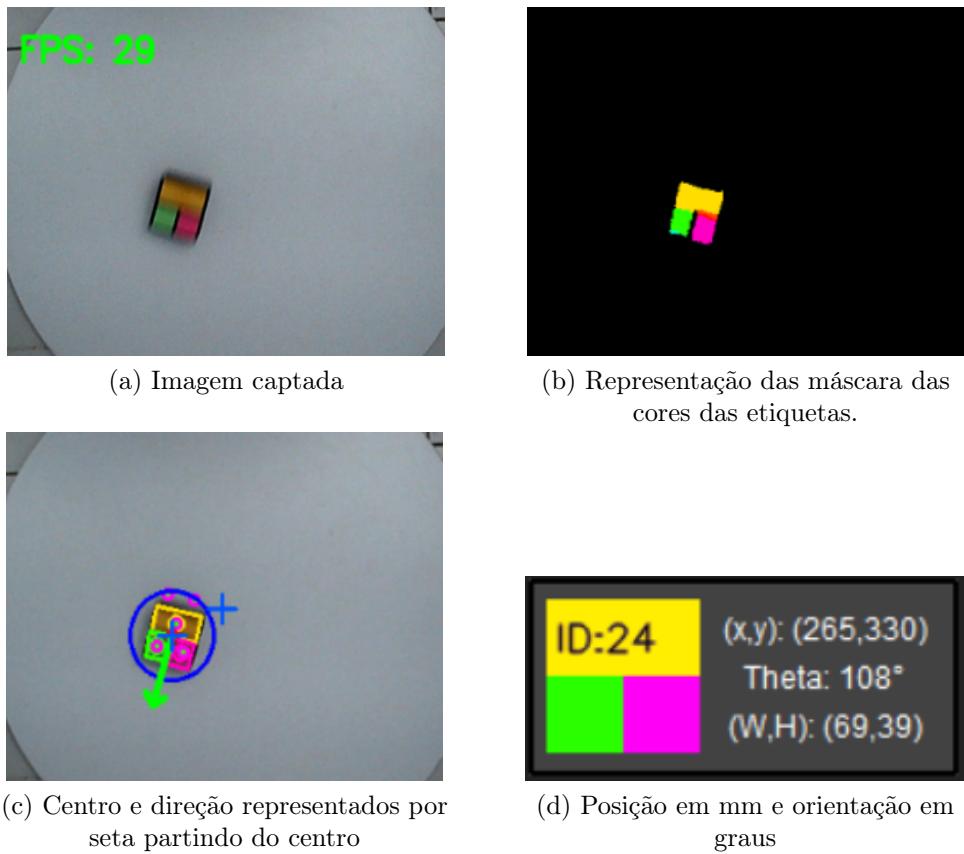


Figura 53: Terceira medição com um robô em movimento.

7.2 Controlador PID de velocidade angular

Foram realizadas medições para valores distintos de velocidade angular e valor médio de PWM. Foram medidos e registrados os valores desejados e medidos de velocidade angular e sinal de controle. As Figuras 54 e 55 apresenta os resultados. A Tabela 6 apresenta os erros médios para valores distintos. Os resultados demonstram que o controlador funciona em uma ampla gama de valores de velocidade angular e PWM_{medio} , apresentando erros médios inferiores ou iguais a 1,86 rad/s e desviam entre 10 e 15% do valor desejado. Os resultados gráficos demonstram que o controlador em resposta à entrada em degrau é capaz de regular a velocidade, tipicamente em menos de 500ms. Também é possível observar que os erros médios são menores quando o valor de PWM_{medio} é maior.

Tabela 6: Erros médios para valores variados.

PWM_{medio}	$\omega_{desejado} [rad/s]$	$Erro_{medio} [rad/s]$	$Erro_{medio} [\%]$
0	3,0	0,36	11,9%
0	5,0	0,75	15,04%
0	10,0	1,26	12,57%
0	20,0	1,86	9,3%
500	15,0	1,86	12,43%
600	3,0	0,41	13,56%
1000	0,0	0,35	inf %
1000	-4,0	0,6	14,9%
1000	4,0	0,66	16,53%
1000	10,0	0,67	6,68%

7.3 Controlador de trajetória com campos vetoriais

O controlador de trajetória desenvolvido é testado em duas situações nas quais o objetivo do jogador amarelo é alcançar a bola sem atingir o jogador azul. No primeiro experimento são posicionados em campo apenas a bola e o jogador amarelo, a Figura 56 apresenta imagem da posição inicial e ao lado, o gráfico da trajetória do robô. Enquanto isso, no segundo experimento foi posicionado um jogador do time azul (um obstáculo para o jogador amarelo), a Figura 57 apresenta imagem da posição inicial e ao lado, o gráfico da trajetória do robô.

Analizando os resultados fica claro que o robô do time amarelo é capaz de alcançar o alvo e desviar do obstáculo, no entanto, a trajetória resultante realiza muitos desvios e oscilações desnecessárias, principalmente na presença de um obstáculo.

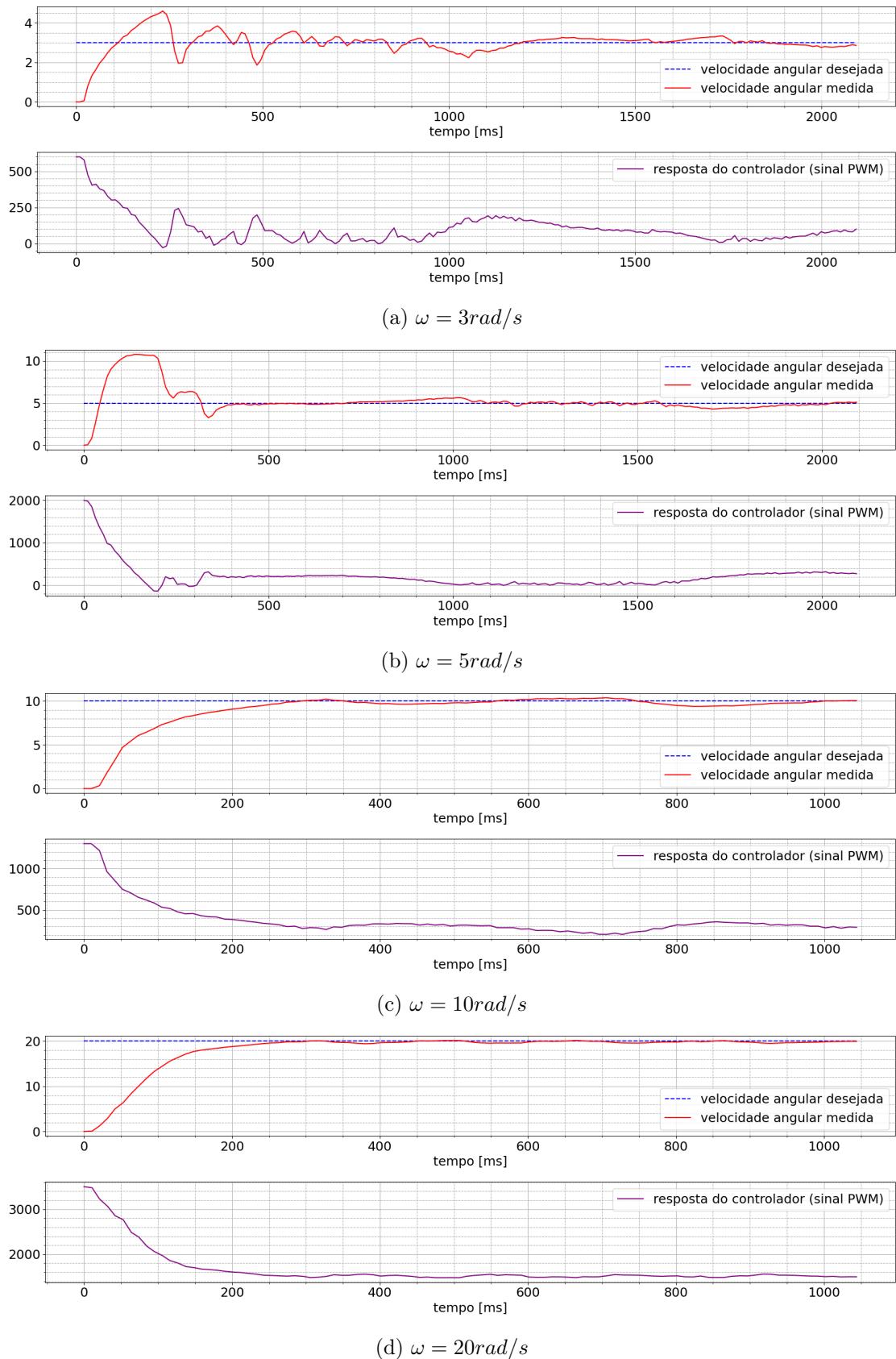


Figura 54: Medições do sinal de controle e velocidade angular desejada e medida, com $PWM_{medio} = 0$.

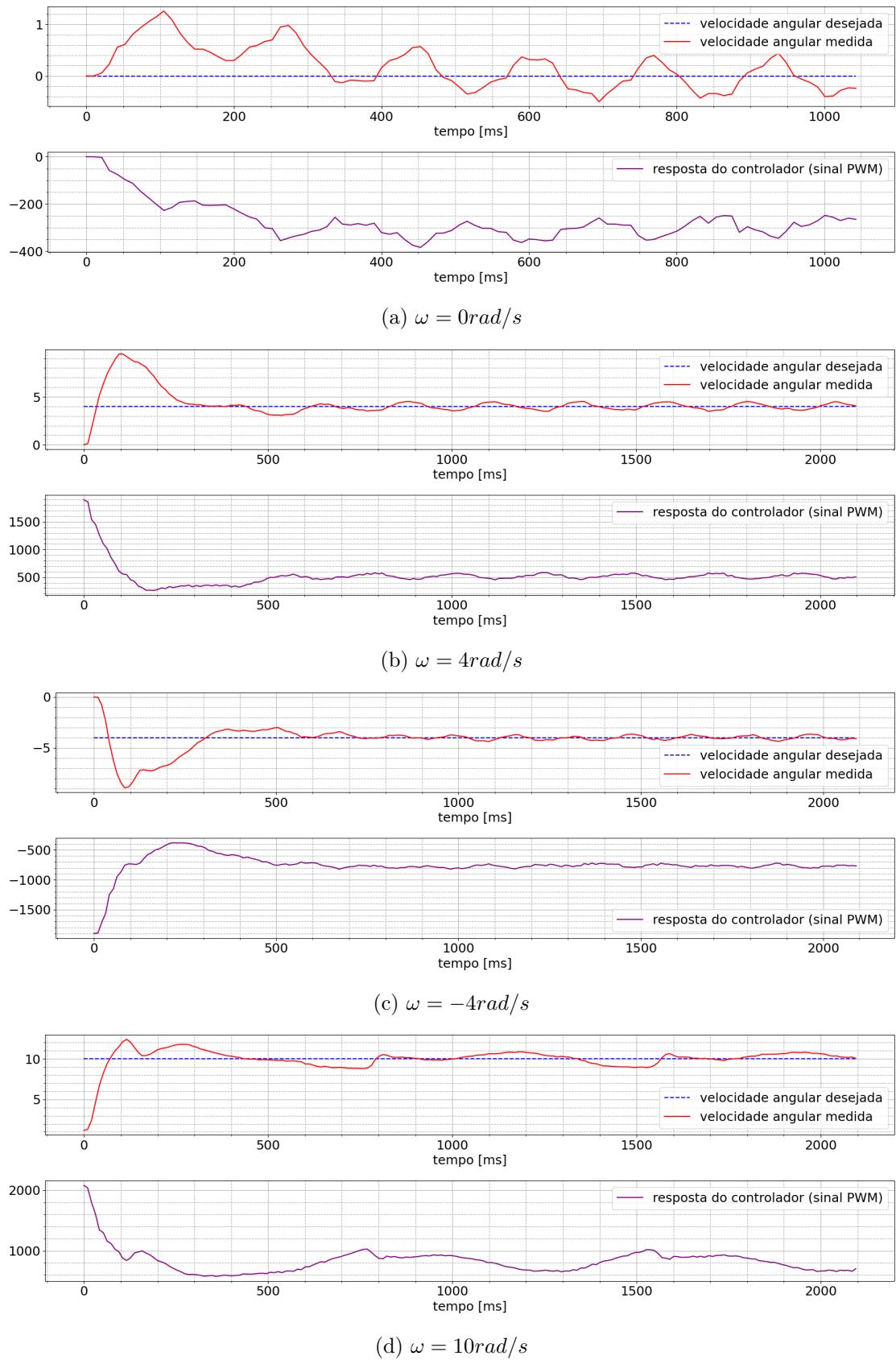
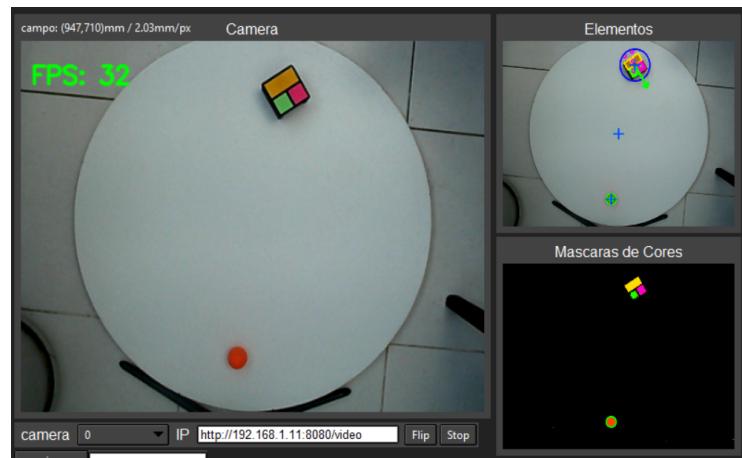
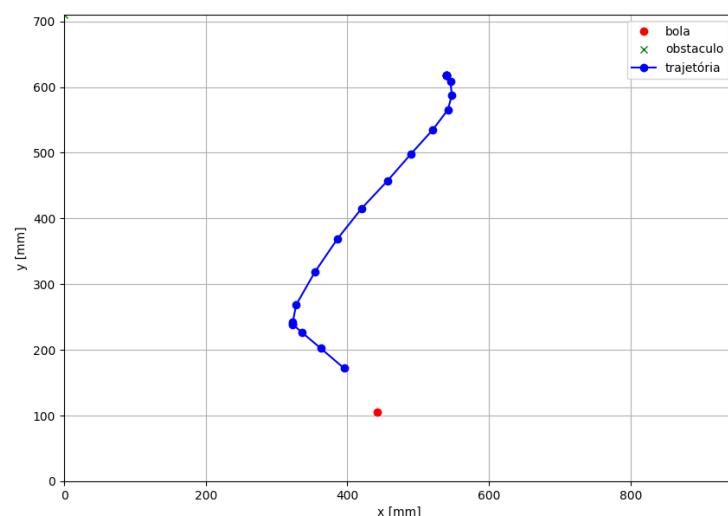


Figura 55: Medições do sinal de controle e velocidade angular desejada e medida, com $PWM_{medio} = 1000$.

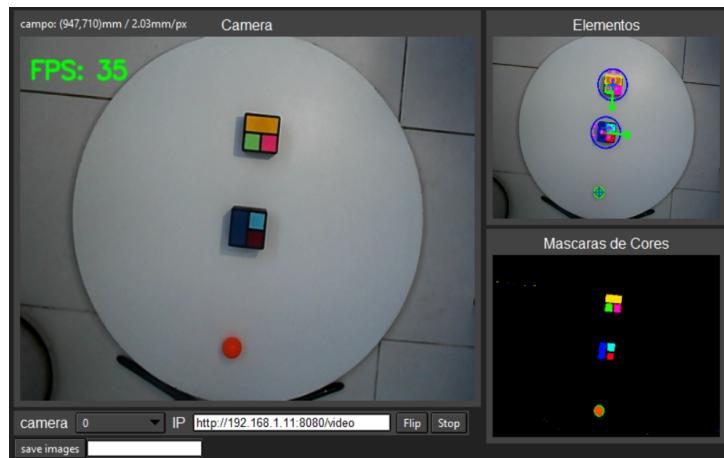


(a) Imagem da posição inicial

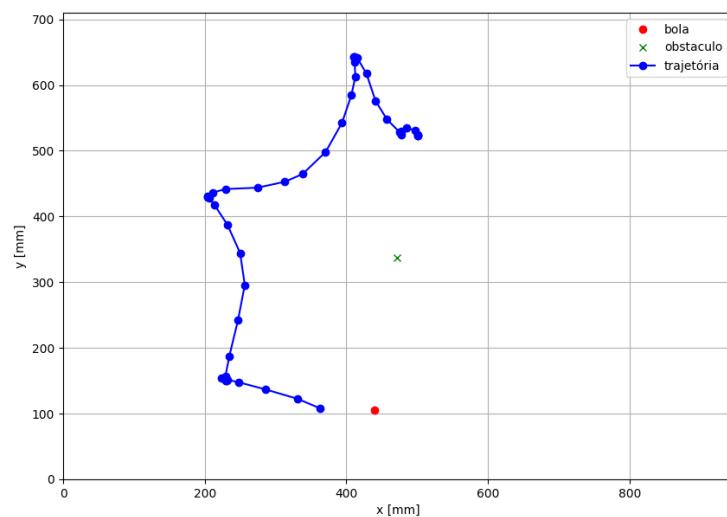


(b) Trajetória resultante

Figura 56: Medição de trajetória sem obstáculos.



(a) Imagem da posição inicial



(b) Trajetória resultante

Figura 57: Medição de trajetória com obstáculo, representado pelo jogador azul.

8 CONCLUSÃO

Ao final deste trabalho, pode-se concluir, que ainda não é possível realizar de forma completa uma partida de futebol VSSS 3x3, no entanto, ele representa o primeiro passo no desenvolvimento de um time competitivo junto a equipe UERJBotz. Além disso, ele promoveu a construção da infraestrutura para futuros trabalhos, alcançando os seguintes resultados:

1. Construção do campo principal com as medidas oficiais e suporte menor para câmera usado para testes.
2. Construção dos robôs: modelagem e impressão 3D das estruturas assim como
3. Desenvolvimento e fabricação do sistema eletrônico: projeto da placa de controle e modificações na placa vespa.
4. Desenvolvimento do algorítimo embarcado nos robôs, com capacidade de receber comandos do dispositivo *Host* e realizar a movimentação dos motores com controle em malha fechada de velocidade angular.
5. Desenvolvimento do sistema de visão computacional multiplataforma.
6. Desenvolvimento de interface gráfica multiplataforma, capaz de controlar os robôs durante jogos ou testes e monitorar e ajustar parâmetros do sistema de visão.
7. Desenvolvimento de algorítimo de controle de trajetória baseado em campos vetoriais.

Desta forma, dentre os objetivos destacados na Introdução, todos foram cumpridos com sucesso, com exceção do último objetivo ("Desenvolvimento de algoritmos de controle para movimentação autônoma dos robôs"). Apesar do controlador de velocidade angular funcionar de forma satisfatória, o controlador de trajetória ainda pode ser melhorado, uma vez que a movimentação apresenta muitos desvios de trajetória e oscilações até alcançar o alvo.

8.1 Trabalhos Futuros

Ao final deste projeto foram levantadas varias questões que poderão ser estudadas em trabalhos futuros, apresentadas a seguir.

1. Implementar arquivos de configuração no sistema de visão computacional, permitindo salvar as configuração sem a necessidade de alterar via linhas de código.
2. Estudar a viabilidade do uso de redes neurais para automatizar o processo de detecção das máscaras de cores.
3. Tornar o sistema de visão aberto para a comunidade e com boa documentação.
4. Gerar uma versão executável do sistema de visão.
5. O controlador embarcado no robô poderia funcionar melhor com *encoders* em vez de uma *IMU*? Ou talvez com uma solução híbrida?
6. Melhorar o algoritmo de planejamento de trajetória.
7. Desenvolver estratégias de controle mais robustas, e com foco especial na movimentação concorrente dos robôs.

REFERÊNCIAS

- [1] CBR. *Regras IEEE VSSS 3x3*. 2021. Acesso em: 01/02/2024. Disponível em: <<https://www.cbrobotica.org/wp-content/uploads/2021/05/vssRules3x321.pdf>>.
- [2] INSTRUMENTS, T. *DRV8833 Dual H-Bridge Motor Driver*. 2023. Acesso em: 16/02/2024. Disponível em: <<https://www.ti.com/lit/ds/symlink/drv8833.pdf>>.
- [3] HOBBYKING. *Pagina do vendedor bateria LiPo 2S 300mAh*. [S.l.], 2023. Acesso em: 16/02/2024. Disponível em: <https://hobbyking.com/pt_pt/turnigy-nano-tech-300mah-2s-35-70c-lipo-pack.html>.
- [4] FEDERATION, R. *A Brief History of RoboCup*. https://www.robocup.org/a_brief_history_of_robocup.
- [5] FONTE, L. F. M. da et al. *UERJBotz VSSS 2023 Team Description Paper*. 2023. Acesso em: 16/02/2024. Disponível em: <https://github.com/UerjBotz/Trabalhos/blob/main/VSSS/TDP_Futebol_VSSS_LARC2023_>.
- [6] ZHAO, J.-S.; LIU, Z.-J.; DAI, J. Design of an ackermann type steering mechanism. *Journal of Mechanical Engineering Science*, v. 227, 11 2013.
- [7] KRISTIAWAN, R. B. et al. A review on the fused deposition modeling (fdm) 3d printing: Filament processing, materials, and printing parameters. *Open Engineering*, v. 11, n. 1, p. 639–649, 2021. Disponível em: <<https://doi.org/10.1515/eng-2021-0063>>.
- [8] Onshape Inc. *Onshape*. 2023. Acesso em: 16/02/2024. Disponível em: <<https://www.onshape.com/>>.
- [9] ESPRESSIF. *Folha de dados do microcontrolador ESP8266*. [S.l.], 2023. Acesso em: 16/02/2024. Disponível em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>.
- [10] AI-THINKER. *Folha de dados do módulo ESP12*. [S.l.], 2023. Acesso em: 16/02/2024. Disponível em: <https://docs.ai-thinker.com/_media/esp8266/docs/esp-12f_product_specification_en.pdf>.

- [11] ROBOCORE. *Placa Vespa*. [S.l.], 2023. Acesso em: 16/02/2024. Disponível em: <<https://www.robocore.net/placa-robocore-vespa>>.
- [12] Arduino. *Arduino Uno R3*. [S.l.], Ano de Fabricação. Acesso em: 16/02/2024. Disponível em: <<https://docs.arduino.cc/hardware/uno-rev3/>>.
- [13] INSTRUMENTS, T. *DRV883x Low-Voltage H-Bridge Driver*. 2023. Acesso em: 16/02/2024. Disponível em: <<https://www.ti.com/lit/ds/symlink/drv8838.pdf>>.
- [14] SYSTEMS, E. *ESP-NOW api-reference*. 2023. Acesso em: 16/02/2024. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_now.html>.
- [15] LENOVO. *Webcam*. 2023. Acesso em: 16/02/2024. Disponível em: <<https://www.lenovo.com/br/pt/accessories-and-monitors/webcams-and-video/webcams/NET-BO-300-FHD-Webcam-Retail/p/GXC1E71383>>.
- [16] OPENCV.ORG. *OpenCV Realeases*. 2023. Acesso em: 16/02/2024. Disponível em: <<https://opencv.org/releases/>>.
- [17] NUMPY.ORG. *Numpy*. 2023. Acesso em: 16/02/2024. Disponível em: <<https://numpy.org/>>.
- [18] ROBOCIN. *VSS vision software*. 2023. Acesso em: 16/02/2024. Disponível em: <<https://github.com/robocin/vss-vision>>.
- [19] PYTHON. *Documentação TKinter*. [S.l.], 2023. Acesso em: 16/02/2024. Disponível em: <<https://docs.python.org/pt-br/3/library/tkinter.html>>.
- [20] T., A. *Introduction to Robotics: Inverse Kinematics*. 2017. Acesso em: 16/02/2024. Disponível em: <<https://www.cs.columbia.edu/allen/F17/NOTES/icckinematics.pdf>>.
- [21] SICILIANO, B.; KHATIB, O.; GROEN, F. (Ed.). *Springer Tracts in Advanced Robotics*. [S.l.]: Springer.
- [22] CHAPMAN, S. J. *Fundamentos de Máquinas Elétricas*. 5^a edição. ed. [S.l.]: Techbooks, 2013.

- [23] O'DWYER, A. *Handbook of PI and PID Controller Tuning Rules.* [S.l.]: Imperial College Press, 2009.
- [24] OGATA, K. *Engenharia de Controle Moderno. 5a ed.* [S.l.]: São Paulo: Pearson, 2010.
- [25] LIMA, G. V. *Modelagem Dinâmica e Controle para Navegação de um Veículo Aéreo não Tripulado do tipo Quadricóptero.* 2015. 102 f. Dissertação (mestrado), Pós-graduação em Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia-MG.
- [26] BENIGNO, T. C. P. *Modelagem Matemática e Controle de Atitude e Posição do Quadrotor.* 2015. 72 f. Dissertação (mestrado) - Mestrado em Sistemas de Comunicação e Automação, Universidade Federal Rural do Semi-Árido, Mossoró-RN.
- [27] LAVALLE, S. M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning.* 1998. Acesso em: 16/02/2024. Disponível em: <<https://msl.cs.illinois.edu/lavalle/papers/Lav98c.pdf>>.
- [28] JOHNATHAN. *Construção de um time de futebol de robôs para a categoria IEEE Very Small Size Soccer.* 2015. Acesso em: 16/02/2024. Disponível em: <<https://sirlab.github.io/assets/pdfs/tcc-johnathan.pdf>>.