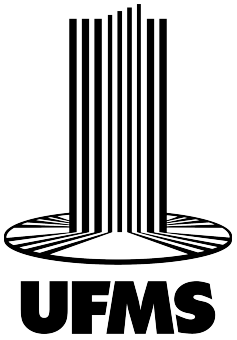


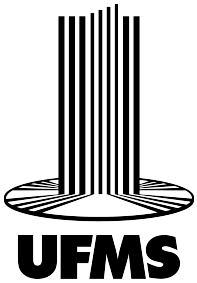
UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACULDADE DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Desenvolvimento de Robô para Categoria SSL-EL

Allan Menchik da Cunha - Amanda Ottoni Petini - Natália Silva Duarte

Campo Grande - MS

02 de Julho de 2025



UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACULDADE DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Desenvolvimento de Robô para Categoria SSL-EL

Allan Menchik da Cunha - Amanda Ottoni Petini - Natália Silva Duarte

Trabalho de Conclusão de Curso apresentado como exigência para obtenção do grau de Bacharel em Engenharia de Computação da Universidade Federal de Mato Grosso do Sul – UFMS.

Orientador: Prof. Dr. Fábio Iaione

Campo Grande - MS
02 de Julho de 2025

Desenvolvimento de Robô para Categoria SSL-EL

Trabalho de Conclusão de Curso apresentado como exigência para obtenção do grau de Bacharel em Engenharia de Computação da Universidade Federal de Mato Grosso do Sul – UFMS.

Banca Examinadora:

Prof. Dr. Fábio Iaione

Prof. Dr. Edson Takashi Matsubara

Prof. Dr. Renan Albuquerque Marks

Campo Grande - MS

02 de Julho de 2025

Resumo

Este trabalho descreve o processo de desenvolvimento de um robô autônomo para a categoria Small Size League – Entry Level (SSL-EL) da RoboCup 2024, a ser utilizado pela equipe AraraBots da Universidade Federal de Mato Grosso do Sul. A SSL-EL é uma categoria recente e voltada a equipes iniciantes, cujo objetivo é reduzir as barreiras técnicas e financeiras, promovendo a ampliação da participação na competição. O projeto contemplou todas as etapas do desenvolvimento do robô, incluindo a estrutura física, o sistema eletrônico, o *firmware* e a integração dos componentes, com ênfase na viabilidade técnica e na redução de custos. Foram explorados conceitos como cinemática inversa, controle orientado por campo e comunicação sem fio, juntamente à escolha estratégica de materiais e componentes de baixo custo, como motores *brushless* para *gimbal* e placas controladoras baseadas em controle orientado por campo. O robô desenvolvido foi submetido a testes tanto em ambiente competitivo — durante a RoboCup 2024, na qual apresentou desempenho satisfatório — quanto em ambientes controlados, com o objetivo de validar sua performance e demonstrar, em ambos os contextos, a eficácia das soluções adotadas. Além disso, o trabalho contribuiu para a consolidação de conhecimentos práticos e multidisciplinares na formação acadêmica dos integrantes da equipe, bem como para a capacitação da equipe AraraBots para futuras participações na categoria SSL-EL.

Palavras-chaves: Futebol de robôs, SSL-EL, sistemas embarcados, microcontrolador.

Abstract

This paper details the development of a low-cost autonomous robot for the Small Size League – Entry Level (SSL-EL) category of RoboCup 2024, created by the AraraBots team from the Federal University of Mato Grosso do Sul. The SSL-EL is a recent category designed to reduce technical and financial barriers for beginner teams, facilitating entry into competitions. It covers all stages of development: physical structure, electronic system, firmware, and component integration, with a focus on technical feasibility and cost containment. Key concepts implemented include inverse kinematics, field-oriented control (FOC), and wireless communication. The strategic selection of low-cost materials and components, such as brushless motors for gimbals and FOC-based controller boards, was critical. The robot was tested in competitive environments (RoboCup 2024, demonstrating satisfactory performance) and controlled environments, validating the effectiveness of the adopted solutions in both contexts. Furthermore, this project consolidated practical, multidisciplinary knowledge within the team and successfully enabled Ararabots' future participation in the SSL-EL category.

Keywords: Robot soccer, SSL-EL, embedded systems, microcontroller.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A *RoboCup Small Size League* (SSL) é uma das ligas de futebol mais antigas da RoboCup, sendo voltada à coordenação e ao controle inteligente de múltiplos agentes em um ambiente altamente dinâmico. Em 2024, foi introduzida uma nova categoria de entrada na RoboCup Brasil, a *Small Size League Entry Level* (SSL-EL), com o objetivo de facilitar a participação de novas equipes por meio da redução de barreiras técnicas e financeiras.

A categoria SSL regular é disputada em campos de 6 metros por 9 metros, e cada time pode colocar até 6 robôs em campo. Para diminuir a barreira de entrada na competição, a categoria EL é jogada com metade do tamanho do campo e da quantidade máxima de robôs da categoria regular, ou seja, 3 metros por 4,5 metros e até 3 robôs. Além disso, o uso do mecanismo de drible — que permite aos robôs da categoria regular carregarem a bola por um curto período de tempo — é proibido, o que simplifica os requisitos técnicos dos robôs participantes.

Embora a nova categoria imponha restrições aos robôs para simplificar sua construção e reduzir os custos de produção, adotar os mesmos componentes e métodos de fabricação utilizados pelas equipes da SSL regular, além de ser financeiramente inviável, pode ser extremamente complexo para muitas das novas equipes. Dessa forma, torna-se imprescindível a busca por alternativas de baixo custo para a produção dos robôs, de modo a viabilizar a participação nessa categoria.

A participação na competição é realizada por meio de equipes de robótica compostas por estudantes que atuam de forma colaborativa na construção dos robôs e dos demais sistemas necessários. Essas equipes são responsáveis por projetar e implementar hardware, firmware e software que atendam aos requisitos técnicos da categoria. Esse processo demanda dedicação, criatividade, capacidade de resolução de problemas, gestão de projetos e trabalho em equipe, promovendo uma significativa contribuição à formação acadêmica e profissional dos participantes, bem como um aumento expressivo no seu desenvolvimento intelectual.

Por conseguinte, este trabalho tem como objetivo descrever os diferentes métodos e materiais utilizados no desenvolvimento do robô da equipe de competição de robótica AraraBots, da Universidade Federal de Mato Grosso do Sul, visando a criação de um projeto de baixo custo e competitivo.

2 Revisão Bibliográfica

Nesta seção serão introduzidos os conceitos fundamentais para o entendimento deste trabalho: categoria SSL-EL, cinemática inversa e controle orientado por campo.

2.1 Categoria SSL-EL

A SSL-EL é uma categoria criada para auxiliar na transição das equipes da *Very Small Size Soccer* (VSSS) para a SSL, uma vez que, a partir de 2024, não haverá mais competições de VSSS na RoboCup Brasil. Além disso, a categoria visa facilitar a entrada de novas equipes na competição.

Os jogos ocorrem em campos retangulares com 4,5 metros de comprimento por 3 metros de largura, conforme demonstrado na Figura 1, sendo essas dimensões metade das utilizadas na categoria SSL regular. Cada partida é composta por dois tempos de cinco minutos, com uma pausa de cinco minutos entre eles. Em caso de empate, podem ser adicionados mais cinco minutos de jogo. A bola é a mesma em ambas categorias, uma bola de golfe na cor laranja, por questões de visibilidade (ROBOCUP, 2024).

As áreas em frente aos gols, como ilustrado na Figura 1, só podem ser ocupadas por um robô, previamente definido como goleiro de sua equipe. Nenhum outro robô pode adentrar essas áreas sem cometer uma infração. Além disso, quando a bola sai da área de jogo, a partida é pausada e a bola é reposicionada antes da retomada do jogo (ROBOCUP, 2024).

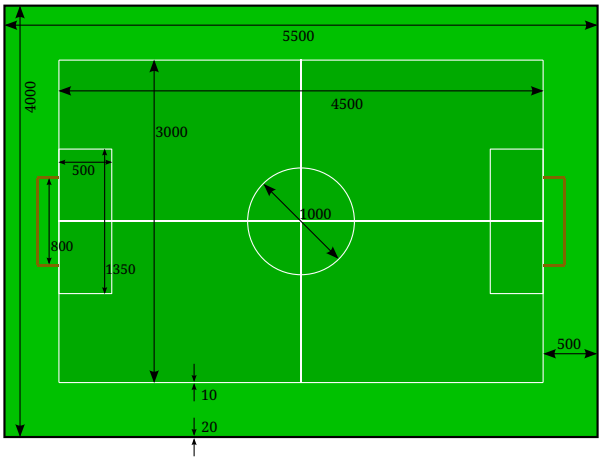


Figura 1 – Medidas do campo da categoria SSL-EL em milímetros. Retirado de (ROBOCUP, 2024).

Uma câmera posicionada acima do campo transmite imagens para um computador da organização, que utiliza técnicas de visão computacional para determinar, em tempo

real, as posições dos robôs e da bola no campo. Cada robô é equipado com uma identificação por cores única e padronizada em seu topo, permitindo seu reconhecimento por meio da imagem (ROBOCUP, 2024).

Os dados obtidos pelo computador da organização são transmitidos às equipes em tempo real por meio de uma rede local, utilizando o *Protocol Buffers*, da *Google*, para que as equipes possam tomar decisões e enviar comandos aos seus robôs (ROBOCUP, 2024).

Assim como o tamanho do campo, a quantidade máxima de robôs por equipe também é limitada à metade da SSL regular: na SSL-EL, são três; na SSL regular, são seis. Além disso, cada robô deve ter formato cilíndrico, com 0,18 m de diâmetro e 0,15 m de altura. As regras determinam que um robô não pode representar risco a si mesmo, a outros robôs ou a pessoas, sendo o juiz responsável por avaliar a situação e solicitar a remoção do robô, quando necessário (ROBOCUP, 2024).

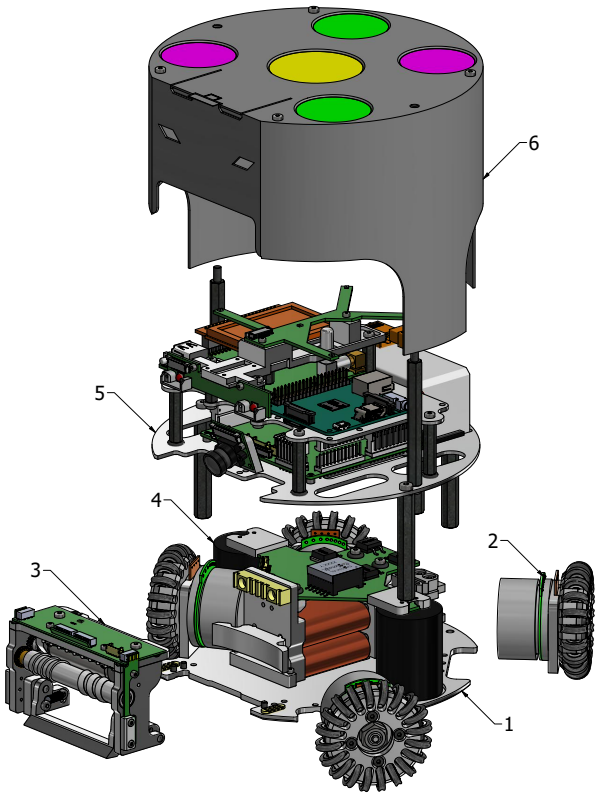


Figura 2 – Exemplo de robô de SSL regular. 1) Placa base. 2) Conjunto propulsor. 3) Mecanismo de drible. 4) Dispositivo de chute. 5) Eletrônicos. 6) Capa externa. Retirado de (RYLL; JUT, 2020).

O mecanismo de drible, mostrado na Figura 2, permite que o robô se mova sem perder o contato com a bola, ao girá-la em direção a si. Esse dispositivo é permitido apenas na categoria regular. Chutes são permitidos em ambas as categorias e podem ser realizados tanto na horizontal quanto em direção diagonal para cima. Normalmente, utilizam-se solenoides para a execução do chute (ROBOCUP, 2024).

Devido às limitações de tamanho e mobilidade do dispositivo de chute, a maioria

dos robôs de ambas as categorias utiliza rodas omnidirecionais e motores dispostos em ângulos não paralelos, permitindo que se desloquem em qualquer direção e rotacionem livremente. Isso facilita o alinhamento para o chute e reduz as restrições de controle.

2.2 Cinemática Inversa

A cinemática é a ciência que estuda a geometria do movimento, restringindo-se a uma descrição puramente geométrica, por meio da posição, orientação e de suas derivadas em relação ao tempo. Na robótica, as descrições cinemáticas dos atuadores e das tarefas a eles atribuídas são utilizadas para estabelecer as equações fundamentais da dinâmica e do controle (JAZAR, 2007).

A cinemática de um robô manipulador refere-se ao estudo da posição e da velocidade de seu efetuador e de seus elos. Podem ser distinguidos dois tipos de cinemática: direta e inversa. Na cinemática direta, deseja-se obter a posição e a velocidade do efetuador, a partir de uma dada posição das articulações. Já a cinemática inversa consiste no processo oposto, em que, conhecendo-se a posição e a velocidade do efetuador, buscam-se as posições e velocidades correspondentes das articulações (CABRAL, 2010).

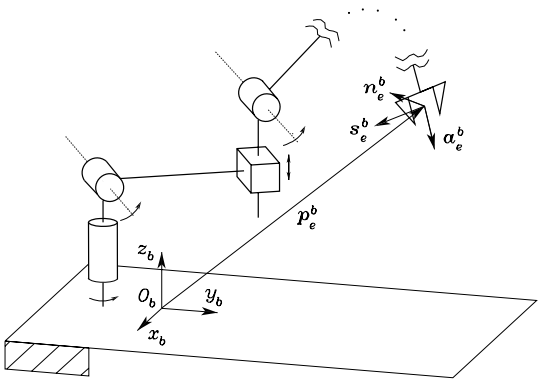


Figura 3 – Exemplo abstrato de braço robótico para estudo de cinemática. Retirado de (SICILIANO et al., 2010).

Em robôs móveis, as equações cinemáticas definem aspectos do movimento, como as velocidades linear e angular, convertendo-as em velocidades angulares individuais para cada roda (SOFWAN et al., 2019). No caso de robôs omnidirecionais, utiliza-se a cinemática inversa para determinar a velocidade necessária de cada roda, de modo que o robô se desloque com direção e velocidade específicas.

A seguir, serão deduzidas as equações necessárias para a aplicação da cinemática inversa em um robô omnidirecional genérico da categoria SSL.

Usando a Figura 4 como referência e as variáveis descritas na Tabela 1, é possível obter as Equações 2.1 e 2.2, que descrevem a velocidade translacional de cada roda.

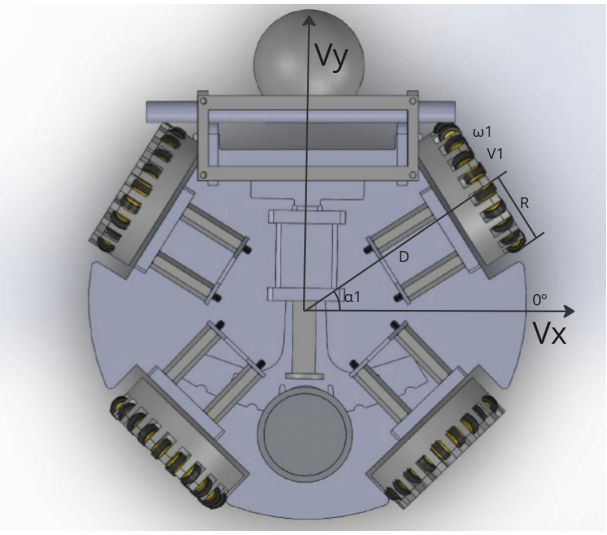


Figura 4 – Design de robô omnidirecional. Retirado de (DUMITRUF et al., 2020).

Tabela 1 – Variáveis utilizadas na cinemática inversa. Retirado de (SOFWAN et al., 2019).

Símbolo	**Unidade**	**Descrição**
V_x	m/s	Velocidade do robô no eixo x
V_y	m/s	Velocidade do robô no eixo y
V_θ	rad/s	Velocidade angular do robô
D	m	Distância das rodas até o centro do robô
R	m	Raio das rodas
α_i	rad	Angulação da roda i em relação ao eixo x
V_i	m/s	Velocidade translacional da roda i
ω_i	m/s	Velocidade angular da roda i

$$V_i = sen(\alpha_i)V_x + cos(\alpha_i)V_y + DV_\theta \tag{2.1}$$

$$V_i = \omega_i R \tag{2.2}$$

Substituindo a Equação 2.1 na Equação 2.2 e isolando a velocidade angular da roda, obtém-se a Equação 2.3.

$$\omega_i = \frac{1}{R}(sen(\alpha_i)V_x + cos(\alpha_i)V_y + DV_\theta) \tag{2.3}$$

Combinando as contribuições de cada uma das rodas, pode-se representar as velocidades angulares por meio de uma operação matricial, como demonstrado na Equação 2.4.

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{1}{R}\begin{bmatrix}sen(\alpha_1) & cos(\alpha_1) & D\\ sen(\alpha_2) & cos(\alpha_2) & D\\ sen(\alpha_3) & cos(\alpha_3) & D\\ sen(\alpha_4) & cos(\alpha_4) & D\end{bmatrix}\begin{bmatrix}V_x\\ V_y\\ V_\theta\end{bmatrix} \tag{2.4}$$

Assim, o firmware do robô transforma as velocidades desejadas (V_x, V_y e V_θ) nas velocidades angulares de cada roda (w_1, w_2, w_3 e w_4), permitindo o controle remoto do robô.

2.3 Controle Orientado por Campo

O Controle Orientado por Campo (FOC – *Field Oriented Control*) é uma estratégia altamente eficaz para o controle de motores comutados eletronicamente, como motores BLDC, motores de corrente alternada e outros, por meio da comutação de fases (BIDA; SAMOKHVALOV; AL-MAHTURI, 2018).

Diferentemente dos motores de corrente contínua com escovas, que dependem de comutação mecânica, os motores sem escovas (como motores BLDC, de passo e de corrente alternada) dependem de algoritmos de controle e circuitos eletrônicos para gerar os campos magnéticos adequados, garantindo, assim, o movimento desejado (BIDA; SAMOKHVALOV; AL-MAHTURI, 2018).

No caso de motores sem escovas trifásicos operando no modo senoidal, o FOC aplica uma comutação em que as três fases são energizadas continuamente com correntes senoidais defasadas em 120 graus entre si. Esse arranjo cria um campo magnético giratório (norte/sul) dentro do estator do motor (BIDA; SAMOKHVALOV; AL-MAHTURI, 2018).

Essa energização contínua das fases, combinada ao uso de correntes senoidais, proporciona transições mais suaves entre as fases quando comparada às correntes trapezoidais, que são mais comuns. Esse comportamento não só melhora a suavidade da operação do motor, como também assegura que o torque se mantenha constante durante toda a operação (BIDA; SAMOKHVALOV; AL-MAHTURI, 2018).

3 Metodologia

Este capítulo está dividido em três seções principais: estrutura, eletrônica e firmware.

A estrutura é responsável por manter todos os componentes em seus devidos lugares garantindo que sensores, motores, circuitos e demais elementos estejam corretamente posicionados para desempenhar suas funções. A eletrônica corresponde ao conjunto de componentes responsáveis por transmitir sinais elétricos, permitindo a comunicação entre os dispositivos e o microcontrolador. O firmware é responsável por processar os sinais recebidos pelos circuitos eletrônicos, tomar decisões e acionar os atuadores de forma adequada, garantindo o comportamento esperado do robô.

3.1 Estrutura

Esta seção tem por finalidade expor as decisões tomadas no projeto estrutural, destacando as motivações técnicas associadas a cada escolha. Durante o desenvolvimento da estrutura do robô, os seguintes objetivos foram considerados:

- Todos os componentes necessários devem estar corretamente posicionados para exercer suas funções.
- Os componentes não devem se deslocar de suas posições sem justificativa.
- A estrutura não deve quebrar quando são aplicadas forças comuns em partidas de SSL-EL.
- Deve ser possível conectar todos os componentes corretamente dentro da estrutura.
- O centro de massa do robô deve ser baixo e centralizado, a fim de evitar tombamentos e garantir boa mobilidade.
- O acesso e a troca de componentes deve ser possível, fácil e rápido, para que possa ser realizada nos curtos períodos de tempo de pausa durante as partidas.

Diante desses objetivos, as decisões e motivações do projeto desenvolvido serão descritas nas subseções a seguir: Material, Rodas e as Camadas Inferior, Intermediária e Superior.

Um modelo 3D da estrutura final pode ser observado na Figura 5, onde são apresentadas as camadas do robô, bem como seu encaixe junto às rodas.

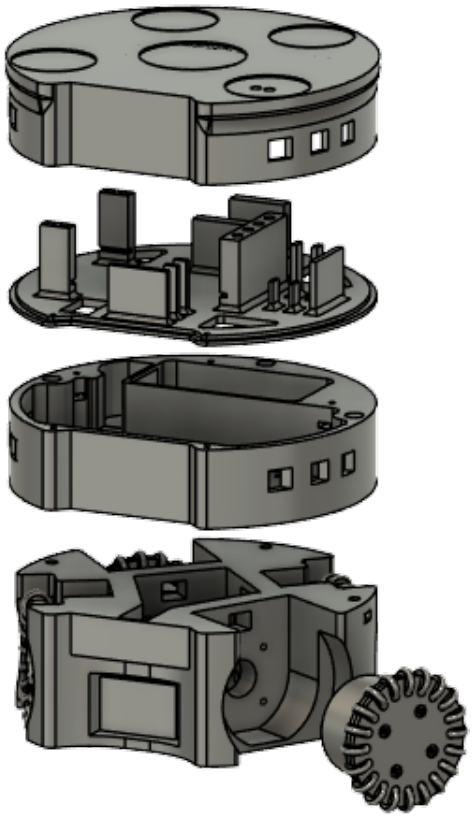


Figura 5 – Vista explodida da estrutura do robô.

3.1.1 Material

Para confecção da estrutura, optou-se por utilizar impressão 3D uma vez que é mais fácil e prático a prototipação e produção de peças. Dois materiais foram considerados, baseando-se na disponibilidade e acesso dos mesmos, PLA e ABS.

O PLA tem como ponto positivo a sua facilidade de impressão, todavia não suporta temperaturas muito altas. Já o ABS possui uma resistência maior à temperatura, mas apresenta maiores dificuldades na impressão, já que necessita de uma impressora de cabine fechada. Como exemplo do impacto disso, tem-se (RYLL; JUT, 2020), a equipe cita que durante as partidas da RoboCup 2019 suas rodas impressas em PLA deformaram e quebraram por conta da alta temperatura.

Após análise, optou-se por utilizar o ABS visto que, durante a movimentação do robô, a energia mecânica gera atrito e, consequentemente, calor que aquece a estrutura, podendo ocasionar danos estruturais caso o material usado não seja resistente. Além do atrito, outras fontes de calor que contribuem para esse processo são a bateria, que aquece ao fornecer correntes elevadas, os motores e os módulos eletrônicos, que também dissipam calor durante o funcionamento.

3.1.2 Rodas

As rodas (Figura 6) são compostas por um anel circular de aço que se encaixa entre duas peças circulares idênticas. Ao longo de todo o perímetro desse círculo, são encaixadas rodas menores com um anel de borracha em seu exterior. Essas rodas de borracha possuem um furo central por onde é fixado o anel de aço.

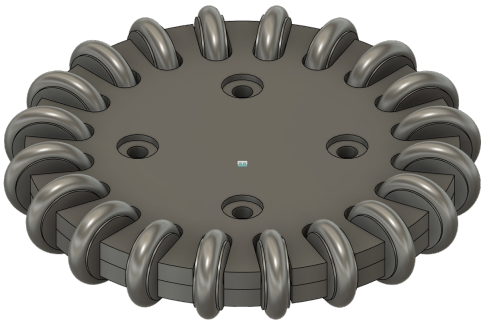


Figura 6 – Imagem 3D da estrutura da roda.

Além dela, foi projetada uma peça de encaixe específica para fixar as rodas aos motores (Figura 7). Sem essa peça, seria necessário parafusar as rodas diretamente nos motores, o que exigiria que o diâmetro da parte sólida das rodas fosse, no mínimo, igual ao diâmetro total do motor.

Isso ocorre porque os furos de fixação do motor selecionado estão posicionados próximos ao seu centro. Nessa configuração, as rodas precisariam ter um diâmetro maior para evitar a colisão entre o motor e as rodas menores de borracha. No entanto, aumentar o diâmetro das rodas comprometeria o projeto, pois reduziria a relação de torque do motor e ocuparia espaço essencial para outros componentes.

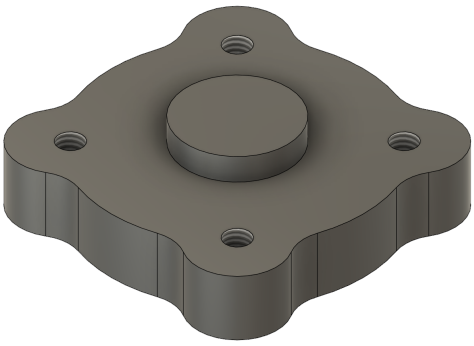


Figura 7 – Peça para encaixe da roda nos motores.

Na Figura 8 é possível ver a estrutura completa da roda com a peça de encaixe.

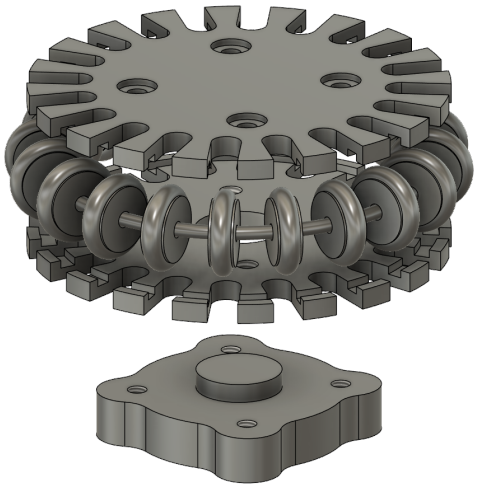
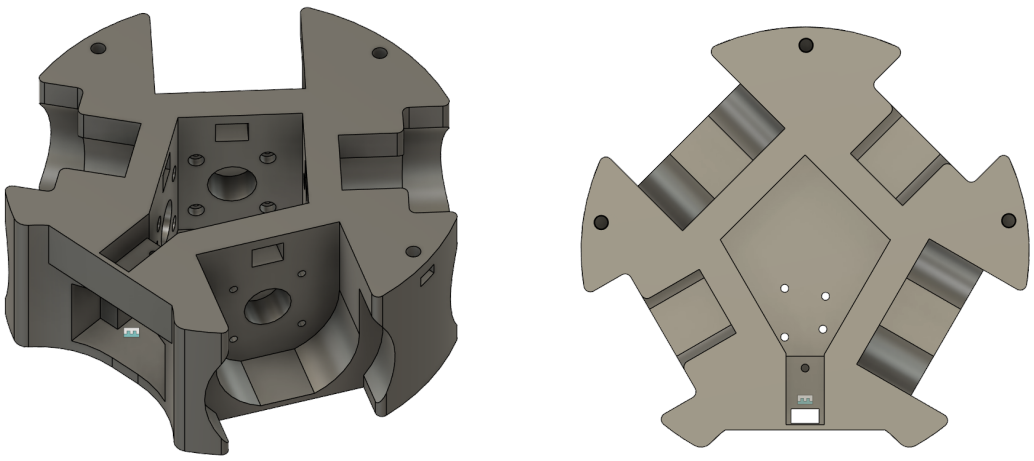


Figura 8 – Vista explodida da estrutura da roda completa.

3.1.3 Camada Inferior

A Camada Inferior é aquela que irá sustentar as demais camadas da estrutura, além de conter os quatro motores e toda a estrutura do chute. Ela foi projetada da forma mais sólida possível, para permitir a sustentação e proteção dos seus componentes internos e superiores.



(a) (b)

Figura 9 – Modelo 3D da Camada Inferior. (a) Vista isométrica do modelo 3D da da Camada Inferior. (b) Vista superior do modelo 3D da Camada Inferior.

Com o objetivo de maximizar o espaço disponível para os componentes na parte frontal do robô e aumentar a velocidade em relação ao eixo y, os motores dianteiros foram posicionados com um ângulo de 30 graus em relação ao eixo x do robô. Por outro lado, os motores traseiros foram inclinados em um ângulo de 45 graus em relação ao mesmo eixo, a fim de preservar a alta aceleração angular e garantir a omnidirecionalidade do robô. Os motores são fixados com parafusos à estrutura.

No que se refere ao solenoide do mecanismo de chute, o componente é parafusado no meio da estrutura e se conecta a uma peça retangular que terá contato direto com a bola, transmitindo a força da bobina para a mesma (Figura 10). Acima dessa pequena estrutura de chute, há um sensor infravermelho responsável por detectar a bola quando a mesma está encaixada na lacuna do chute.

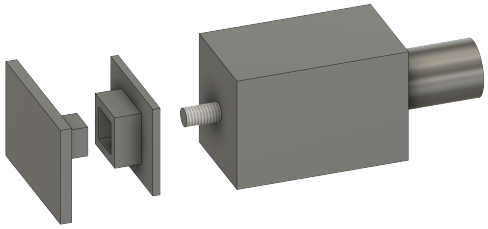


Figura 10 – Estrutura do chute e representação do solenoide.

3.1.4 Camada Intermediária

Essa camada é responsável por armazenar a bateria e um conversor de tensão, além de proporcionar e proteger a passagem de fios para a Camada Superior (seção 3.1.5), tanto de seus próprios componentes quanto dos componentes da Camada Inferior (seção 3.1.3). Ela também possui aberturas laterais para permitir a ventilação interna da estrutura, uma vez que os componentes ali presentes podem ser permanentemente danificados pelo calor gerado durante seu funcionamento.

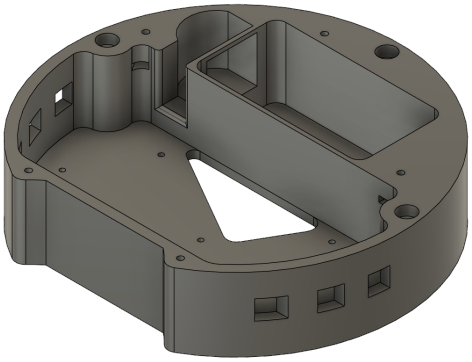


Figura 11 – Imagem 3D da Camada Intermediária.

3.1.5 Camada Superior

A última camada do robô, a Camada Superior, é dividida em duas peças: uma na qual são anexados os circuitos do robô, fixados em suportes projetados especificamente para cada um; e outra que funciona como uma tampa, onde são encaixadas as tags (identificação de cores única e padronizada) exigidas nos jogos da SSL-EL.

Para que ambas as peças fossem conectadas, além do próprio formato que permite um encaixe preciso, foram utilizados ímãs, garantindo que não se soltem durante a movimentação do robô.

Os fios dos componentes das camadas inferiores podem ser acessados por meio de aberturas feitas na base da camada. Também foi projetado um encaixe para uma chave que permite ligar e desligar o robô com facilidade sempre que necessário.

A Figura 12 mostra o modelo 3D da Camada Superior. No entanto, modificações foram realizadas após o período de impressão devido a alterações no circuito durante a competição. Os suportes para os ESCs (mais detalhes na seção 3.2.4) e um dos suportes para ímãs foram removidos e reconfigurados para acomodar os novos componentes.

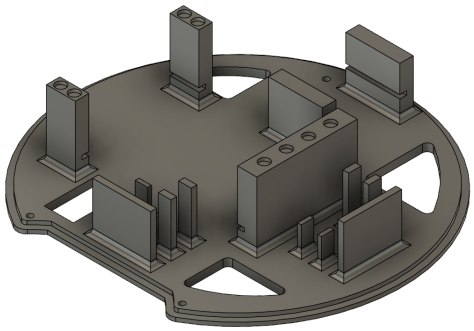


Figura 12 – Modelo 3D da Camada Superior.

A tampa (Figura 13), como citado anteriormente, possui suportes circulares para as tags em sua parte superior. Suas laterais contêm pequenos orifícios que permitem a entrada de ar na camada onde os componentes estão localizados, contribuindo para a redução da temperatura interna.

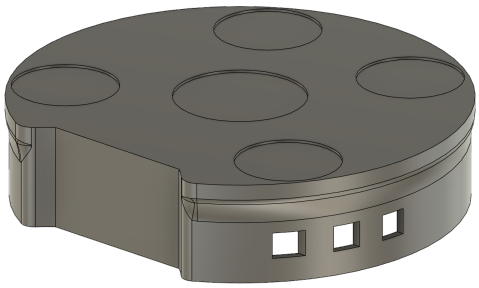


Figura 13 – Modelo 3D da tampa do robô.

3.2 Eletrônica

Esta seção apresenta o projeto eletrônico do robô, incluindo os componentes utilizados, suas conexões e a alimentação do sistema. Para facilitar a leitura, o conteúdo foi dividido em subseções: sistema de chute, microcontrolador, motores, interface de potência, giroscópio e acelerômetro, comunicação sem fio e circuito.

3.2.1 Sistema de Chute

O chute do robô é extremamente necessário durante as partidas de SSL e SSL-EL, já que é a única maneira consistente de marcar gols e ganhar partidas, pois somente os goleiros podem entrar na área do gol de seu respectivo time. Dentre as equipes, o componente mais comum para fazer o deslocamento linear necessário para o chute é o solenoide.

Em um canal de comunicação entre as equipes foi feita a recomendação de uma empresa brasileira (Soletec) que fabrica solenoides em miniatura (Figura 14) em poucas quantidades e por um preço relativamente baixo, R$ 64,00. Foram adquiridos 4 solenoides modelo 657 025 881, 12 Vcc, com um período de ligação relativo de 100%. O período de ligação relativo (PL) é a razão entre o tempo em que o solenoide passa ativado e o tempo total decorrido.

A bobina do solenoide reside dentro de uma caixa metálica de 43 mm de comprimento, 30 mm de largura e 24 mm de altura, enquanto seu eixo tem comprimento de 71 mm e pode se mover até 8 mm quando acionado. Informações sobre o posicionamento do solenoide podem ser encontradas na seção 3.1.3. O modelo escolhido também possui mola de retorno, para não ser necessário fazer a alimentação contrária para a retração do eixo, simplificando o projeto eletrônico.

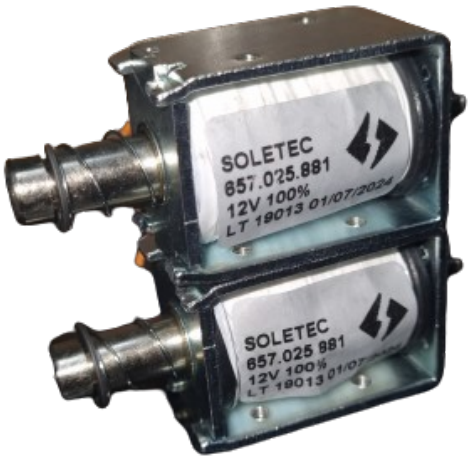


Figura 14 – Solenoide modelo 657 025 881, 12 Vcc e PL de 100%.

O modelo com PL de 100% é feito para acionamento contínuo, e por isso possui uma resistência interna maior, consequentemente, aciona com menos força comparado aos modelos com valores de PL menor. Ao ser consultado, o fabricante informou que, como o PL real é mais próximo de 25%, é possível utilizar uma tensão maior, de até 50 V, aumentando a potência de 6 W para 100 W. Para alimentar o solenoide com 50 V, dado que a bateria fornece entre 18 V e 22 V, foi usado um conversor *step up*.

A potência maior gera um risco de danificar o solenoide caso um PL maior do

que 25% seja usado. Para garantir a segurança e diminuir o consumo de energia com acionamentos em falso, caso em que a bola não está na posição para ser chutada, foi adicionado um sensor infravermelho de curta distância voltado para o local da bola no momento do chute. Esse sensor, quando detecta algum objeto em sua frente, emite um sinal elétrico que é lido por um dos microcontroladores e, caso o robô receba tanto o sinal de chute pelo rádio quanto o do sensor infravermelho, o chute é acionado.

Foi utilizado um circuito auxiliar que permite o acionamento do solenoide com uma voltagem alta de forma isolada do restante do circuito, permitindo maior segurança aos demais componentes.

Esse circuito auxiliar é conectado ao microcontrolador em uma porta configurada como *pull down*. Quando acionada com nível lógico 1, a porta atua como fonte 3,3V, fazendo a corrente circular do microcontrolador até o *Gate* de um transistor, acionando-o. Isso corrobora para que haja corrente circulando pelo Drain e Source e, consequentemente, pelo solenoide, ativando o mecanismo do chute.

No momento em que essa porta é colocada no nível lógico 0, ou seja, interrompe a alimentação no *Gate* do transistor levando-o para o estado desligado, não haverá passagem de corrente elétrica pelo circuito do solenoide, desativando-o.

O diagrama esquemático do circuito completo do robô, incluindo o circuito do chute, pode ser encontrado na seção 3.2.8.

3.2.2 Microcontrolador

A escolha do microcontrolador STM32F103C8T6, que pode ser visto na Figura 15, foi baseada em diversos fatores. A linha STM32 é amplamente utilizada por outras equipes devido à sua alta performance e facilidade de aprimoramento, permitindo a substituição por modelos mais avançados sem grandes alterações no projeto caso seja necessário. Além disso, seu custo é relativamente baixo, variando entre R$ 20,00 e R$ 30,00. O microcontrolador opera com uma frequência de *clock* de até 72 MHz, possui 128 KBytes de memória flash e 20 KBytes de memória SRAM. Conta com 37 pinos de entrada e saída, 10 canais conversores analógico-digitais de 12 bits, 4 *timers* independentes e configuráveis, além de duas conexões SPI, três seriais UART e duas I2C. O módulo utilizado mede 53 mm x 22 mm, consome no máximo 50 mA e pode ser alimentado com 5 V ou 3,3 V.

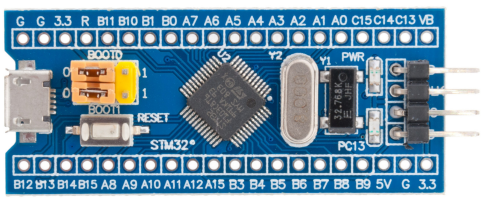


Figura 15 – Microcontrolador STM32F103C8T6.

Durante o desenvolvimento do projeto foi feita uma troca no método de acionamento dos motores (mais detalhes na seção 3.2.4). Consequentemente, somente um microcontrolador não era capaz de executar todo o firmware com desempenho necessário, principalmente por conta da biblioteca SimpleFOC, que exige um período de execução do loop principal do firmware de, no máximo, 2 ms para funcionar corretamente. A solução encontrada foi dividir o processamento em dois microcontroladores e fazer com que se comuniquem a fim de passar as informações necessárias. Mais detalhes estão nas seções 3.3.6 e 3.3.7.

O diagrama esquemático do circuito completo do robô, incluindo os dois microcontroladores, pode ser encontrado na seção 3.2.8.

3.2.3 Motores

Os motores sem escova (*brushless*), ou BLDC (*Brushless Direct Current*), são motores elétricos de corrente contínua que funcionam sem escovas, diferentemente dos motores convencionais. Essa ausência de escovas reduz atrito e o desgaste, proporcionando maior eficiência e durabilidade.

Ao invés de utilizar escovas, os motores *brushless* contam com um controlador eletrônico que alterna a corrente entre as bobinas do estator, assim gerando um campo magnético que ativa pares de bobinas, conferindo um movimento rotativo ao rotor. Deste modo, permite ajustes precisos de velocidade e torque, tornando esses motores ideais para aplicações que exigem alta precisão, como o projeto apresentado.

Em sua grande maioria, os motores *brushless* oferecem um design compacto, sendo apropriados para projetos de robôs da SSL-EL por conta da exigência de alto desempenho em espaços reduzidos.

Como um dos grandes objetivos era desenvolver um robô para SSL-EL com foco em baixo custo, surgiu a ideia de utilizar motores usados em estabilizadores de câmeras, mais conhecidos como *gimbals*, por oferecerem as mesmas características citadas acima, visto que, em sua maioria, são motores *brushless* e são de fácil aquisição com custo baixo, em torno de R$ 190,00, se comparado com outros motores geralmente utilizados na categoria.

Assim, os motores da fabricante *IFlight* para *gimbal* surgiram como fortes candidatos. A Tabela 2 apresenta as características essenciais para a sua escolha. Esses motores possuem um formato cilíndrico e um rotor externo, ou seja, a parte fixa está no centro, enquanto todo o acoplamento é responsável por transmitir o movimento.

Inicialmente, foi adquirido o motor GM3506 por conta da sua altura reduzida se comparado com outros da categoria, deixando mais espaço interno para outros componentes. Após uma série de testes, realizados com diversas interfaces de potência, ficou evidente que o torque que esse modelo fornecia não era suficiente para mover o robô.

Tabela 2 – Características técnicas dos motores Iflight GM3506 e GM4108-120T.

	GM3506	GM4108-120T
Corrente máxima com carga (A)	1	$1,5$
Alimentação máxima com carga (V)	12	20
Torque com carga ($g \cdot cm$)	$600 - 1000$	$1200 - 1800$
Diâmetro externo (mm)	$\phi 40 \pm 0,05$	$\phi 45 \pm 0,05$
Altura (mm)	$17,8 \pm 0.2$	$32,3 \pm 0.2$

Assim, ficou decidido utilizar o GM4108-120T (Figura 16), por ter um torque consideravelmente maior.

Figura 16 – Motor BLDC *gimbal* modelo GM4108-120T.

3.2.4 Interface de Potência dos Motores

O ESC (*Eletronic Speed Controller*), um controlador eletrônico de velocidade, foi a primeira opção utilizada para acionar os motores. Este dispositivo é amplamente utilizado em drones para o acionamento de motores *brushless*, e tem como função principal variar a rotação do motor elétrico através do controle de acionamento das bobinas.

O ESC utiliza um microcontrolador interno que gera uma sequência pré-definida de sinais em seu firmware para o acionamento do circuito de MOSFETs responsável por energizar as bobinas. O controle de velocidade é feito através de um sinal *Pulse Width Modulation* (PWM), que é recebido e interpretado pelo microcontrolador do ESC, convertendo-o em sinais para o chaveamento dos MOSFETs.

É possível observar na Figura 17 que o dispositivo possui três saídas, A, B e C destacadas, sendo que cada conexão dessa com o motor é responsável por acionar um par de bobinas. Caso as conexões A e C sejam invertidas, isso inverterá o sentido da rotação do motor.

Figura 17 – Controlador eletrônico de velocidade (ESC).

O modelo de ESC selecionado não apresentava a possibilidade de inversão do sentido da rotação do motor, fator indispensável para a movimentação do robô. Com o propósito de resolver esse problema, uma solução rápida foi implementada, invertendo eletronicamente, por meio de relés HJR1-2CL 12V da fabricante *Tianbo*, os sinais enviados para os terminais A e C, e, desta forma, invertendo o sentido de rotação quando necessário.

No início dos testes, o circuito inversor apresentou falhas devido à inversão ocorrer de forma lenta, comprometendo o dinamismo necessário para a aplicação.

Após realizar testes de movimentação, constatou-se que, embora os quatro motores possuíssem torque suficiente para deslocar o robô, o controle por meio dos ESCs resultava em um torque reduzido.

Isso ocorre porque os ESCs são amplamente empregados no aeromodelismo, uma aplicação que exige alta velocidade de rotação a maior parte do tempo. Além disso, o motor usado para acionar hélices trabalha com uma carga pequena (torque pequeno) em baixas rotações e, portanto, os ESCs para essas aplicações parecem não ser projetados para se obter torque elevado em baixas rotações. Na movimentação do robô, o cenário é bem diferente, pois é necessário torque elevado mesmo em baixas rotações para superar o atrito estático.

Assim, se tornou necessário buscar alternativas de controladores para o acionamento do BLDC. Uma das alternativas encontradas durante a CBR 2024, foi a utilização da placa SimpleFOCMini v1.1, por ser um driver compacto, de baixo custo e altamente eficiente desenvolvido para o controle de motores *brushless*. Esse *driver* faz parte do ecossistema SimpleFOC, uma plataforma de código aberto voltada à simplificação da implementação de controle orientado por campo (mais detalhes na seção 2.3).

A versão em miniatura da placa (Figura 18), além de manter a compatibilidade com a biblioteca SimpleFOC, foi projetada para ocupar pouco espaço físico, sendo ideal para projetos com restrições de tamanho. O SimpleFOCMini v1.1 utiliza um *driver* de três fases baseado no CI DR8313, permitindo o controle de motores trifásicos com tensão de alimentação entre 8 e 35V, com corrente máxima de 2,5A.

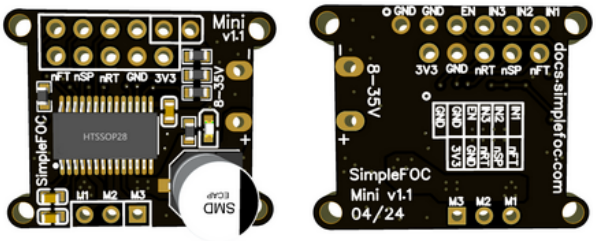


Figura 18 – Imagens do módulo SimpleFOCMini v1.1.

Ainda sobre o SimpleFOCMini, a placa é capaz de controlar um motor (porta M1, M2 e M3), por meio de 3 sinais *PWMs* gerados pelo microcontrolador, e também é responsável pela alimentação dos motores.

Os três sinais *PWMs*, recebidos pelas portas IN1, IN2 e IN3, são convertidos em uma forma de onda senoidal. Essa conversão permite uma ativação contínua dos motores. Uma das principais vantagens da técnica FOC (Field-Oriented Control) é a geração de um campo magnético intenso. Isso possibilita um controle preciso da intensidade do campo magnético e, consequentemente, do torque gerado pelo motor. As conexões dos quatro módulos SimpleFOCMini no robô podem ser vistas na Figura 21.

3.2.5 Giroscópio e acelerômetro

Foram feitos alguns testes com a unidade de processamento de movimento (MPU) com giroscópio e acelerômetro MPU6050 (Figura 19) produzido pela *InvenSense*. A escolha se deu principalmente pelo seu baixo custo e fácil utilização, mas o componente possui várias outras qualidades. O módulo não só faz medições da aceleração linear e da velocidade angular, mas também possui um conversor analógico-digital de 16 bits para digitalizar seus sinais e comunicar a outros componentes eletrônicos a partir da sua interface I2C.

Além das funções citadas anteriormente, também é possível ajustar o *full-scale range* do giroscópio de $\pm 250°/s$ até $\pm 2000°/s$ e do acelerômetro de $\pm 2g$ até $\pm 16g$ ($g = 9,81 m/s^2$). O módulo precisa ser alimentado com 3,3V e apresenta as seguintes dimensões: 16mm x 20mm.

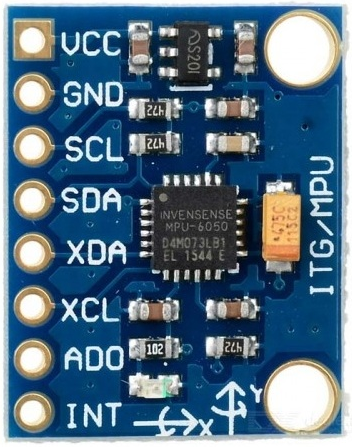


Figura 19 – Giroscópio e Acelerômetro MPU6050.

3.2.6 Comunicação sem fio

Para a comunicação sem fio foi usado o transceptor de rádio de chip único NRF24L01 (Figura 20) da *Nordic Semiconductors* por conta de seu baixo custo, cerca de R$ 20,00, fácil operação em razão de bibliotecas disponíveis, consumo máximo de 15 mA, amplo uso em projetos de robótica, além de ter pequenas dimensões (45,5 mm x 16,4 mm).

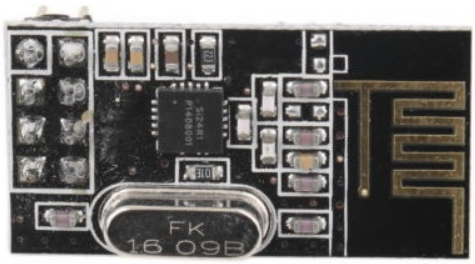


Figura 20 – Módulo de comunicação via rádio modelo NRF24L01.

O módulo opera entre as frequências de 2,400 GHz e 2,525 GHz, com a exata frequência sendo definida pelo canal escolhido. Cada canal ocupa uma largura de banda de 1 MHz e é necessário que ambos os dispositivos, transmissor e receptor, estejam no mesmo canal para que a transmissão tenha sucesso.

Todas as configurações do módulo podem ser feitas por um microcontrolador a partir da comunicação SPI, incluindo: potência de saída, canais de frequência e configuração de protocolo. Mais detalhes sobre a programação serão apresentados na Seção 3.3.3.

Na primeira versão da implementação do módulo, o mesmo foi alimentado diretamente pela saída de 3,3 V do microcontrolador. Por conta da baixa corrente máxima do regulador de tensão do microcontrolador, esse método fazia com que o NRF24L01 parasse de ser alimentado caso algum outro componente eletrônico exigisse um pouco a mais de

corrente. Quando o NRF24L01 é desligado e ligado novamente, é preciso refazer todas as suas configurações para que funcione corretamente de novo.

Foram feitos testes com capacitores conectados à entrada de energia do dispositivo, para filtrar ruídos e oscilações, mas o problema ainda persistia. A solução encontrada foi adicionar outro regulador de tensão de 3,3 V ao circuito, que alimenta exclusivamente o módulo de rádio, e assim não houve mais problemas de desconexão ou leitura de valores aleatórios.

Importante ressaltar que o mesmo cuidado com a alimentação deve ser observado para o rádio transmissor, sendo assim, o mesmo também precisa ser alimentado por uma fonte 3,3V que não limite sua corrente e prejudique o envio de mensagens. Durante a competição, foi utilizado uma fonte de bancada.

3.2.7 Fonte de Energia

A alimentação foi feita utilizando uma bateria Li-Po de 5 células, com tensão podendo variar entre 18V e 22V, dependendo da carga em suas células, e com capacidade de 2200mAh, suficiente para alimentar todos os elementos do robô durante uma partida, cerca de 10 minutos. Ela é ligada a dois componentes: um *step up* e um *step down*, além de estar conectada diretamente a todos os *drivers* de motor *SimpleFOCMini*.

Foi considerado usar um *step up down* para alimentar os motores quando o ESC era utilizado como interface de potência, visto que a tensão da bateria pode variar de acordo com sua carga e os motores podem ser alimentados com no máximo 20 V. Assim, um conversor capaz de aumentar e diminuir a tensão simultaneamente foi testado e encarregado de alimentar os motores.

Os *drivers* usados no final do projeto, *SimpleFOCMinis*, fazem a regulação da tensão que é aplicada nos motores. A tensão máxima que utilizam para o acionamento dos motores é de 10 V, já que o método de ativação FOC apresenta um funcionamento eficiente mesmo com tensões menores. Essa limitação de tensão é definida em firmware, como será explicado posteriormente na seção 3.3.5. Portanto, o uso do *step up down* foi descartado.

Cada componente alimenta uma parte do circuito diferente. O step up escolhido permite uma tensão de entrada entre 8,5V e 50V, e converte a mesma para uma tensão (ajustável) entre 10V e 60V, suportando uma corrente de até 15A. O dispositivo foi regulado para disponibilizar uma tensão de 50V utilizada na ativação do solenoide. O step down é responsável por fornecer uma tensão de 5V para o circuito de controle composto pelos microcontroladores e sensores, sua entrada pode variar entre 7V e 26V e sua corrente máxima é de 3A.

3.2.8 Diagrama Esquemático do Circuito Eletrônico

O circuito implementado pode ser dividido em 3 partes para facilitar seu entendimento. A primeira parte trata-se da alimentação, vista na seção 3.2.7, a segunda trata-se do circuito de controle e a terceira é o circuito auxiliar do chute (seção 3.2.1).

No que se refere ao circuito de controle, o mesmo diz respeito aos microcontroladores e sensores do robô. Foram utilizados dois microcontroladores, conforme citado na seção 3.2.2, além de um sensor infravermelho e um módulo de comunicação sem fio (seção 3.2.6).

O diagrama esquemático do circuito do robô pode ser visto na Figura 21, bem como a pinagem dos componentes e a forma como as partes estão conectadas.

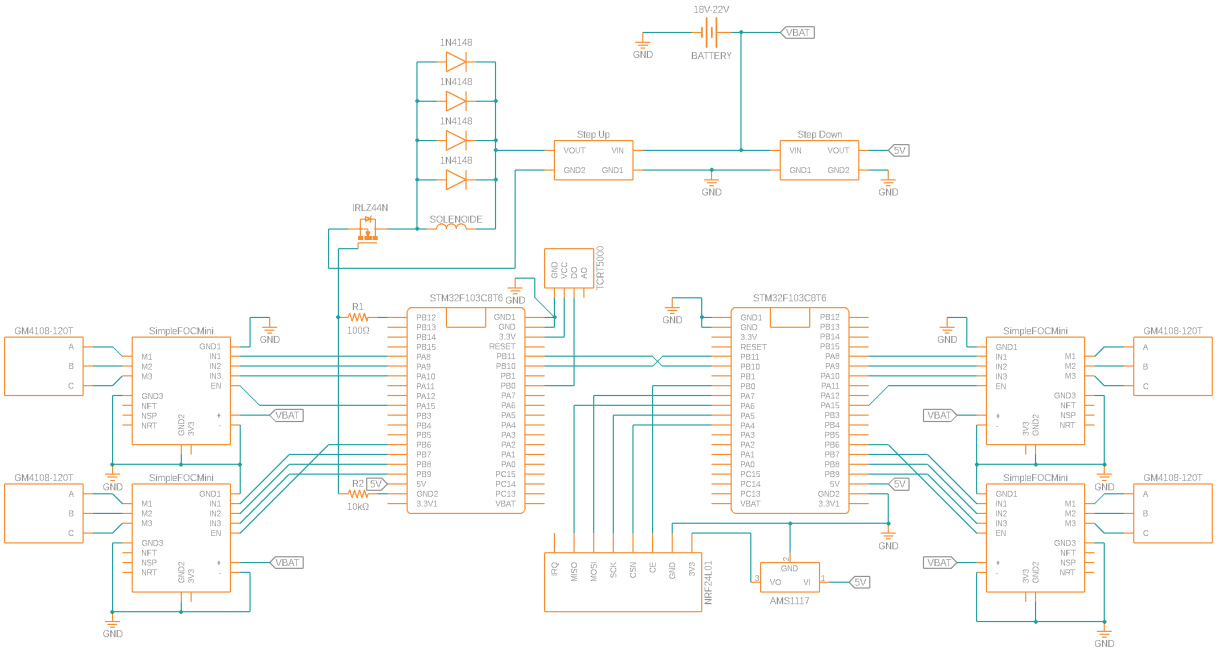


Figura 21 – Esquema Eletrônico do Robô.

O MOSFET utilizado no circuito auxiliar do chute é o IRLZ44N. Também foram usados dois resistores, R1 e R2, que têm como resistência 100Ω e 10$k\Omega$ respectivamente. No que diz respeito aos diodos, optou-se por usar o 1N4148, que é um diodo rápido e consegue absorver os transientes de alta tensão gerados pelo solenoide ao ser desligado.

Com relação à alimentação do NRF24L01, não se obteve bons resultados conectando o mesmo na saída de 3,3V fornecida pelo microcontrolador, pois o regulador do módulo microcontrolador acaba limitando a corrente, resultando em um mau funcionamento do rádio. Assim, foi utilizado um regulador AMS1117 que converte 5V para 3,3V.

3.3 Firmware do Robô

Nessa seção será abordada a escolha de ambiente de desenvolvimento e suas motivações, a implementação de um módulo de giroscópio e acelerômetro para controle embarcado, a comunicação sem fio para controle do robô à distância, a cinemática inversa para traduzir os comandos recebidos em velocidade das rodas, a ativação dos motores para movimentação, a comunicação serial entre os dois microcontroladores e a arquitetura do firmware. Durante o desenvolvimento do firmware, levou-se em conta os seguintes objetivos:

- Escalabilidade: Deve ser possível adicionar novas funcionalidades sem a necessidade de remodelar toda a arquitetura do firmware.
- Desempenho: O firmware deve apresentar um desempenho adequado para permitir a participação da categoria SSL-EL.
- Funcionalidade: Todas as funcionalidades do robô que precisam ser controladas pelo microcontrolador devem ser implementadas e devem funcionar de acordo com a necessidade de cada uma.
- Fácil leitura: O código deve ser facilmente entendido, para que futuros participantes da equipe possam fazer ajustes e melhorias no software.

3.3.1 Ambiente de desenvolvimento

Durante o desenvolvimento do projeto, foram utilizados três ambientes de desenvolvimento distintos: CubeIDE, Arduino IDE e Visual Studio Code com a extensão PlatformIO. Cada um desses ambientes apresenta vantagens, desafios e limitações, como descrito na Tabela 3.

O CubeIDE foi o primeiro ambiente adotado por ser desenvolvido pela mesma empresa fabricante dos microcontroladores utilizados, a STMicroelectronics. Entre as IDEs utilizadas, é a que oferece o controle mais avançado sobre o microcontrolador, possibilitando ajustes como a configuração da frequência de *clock*, habilitação e desabilitação de funcionalidades específicas, controle completo sobre os modos dos pinos, geração automatizada de código de configuração e diversas outras.

Entretanto, a ampla gama de funcionalidades também representa sua principal desvantagem, exigindo um nível elevado de conhecimento, tanto da IDE, quanto do microcontrolador, para o desenvolvimento de qualquer projeto. Além disso, a interface de desenvolvimento possui uma comunidade de usuários relativamente limitada, refletindo-se em recursos educacionais simplificados, o que pode dificultar o aprendizado e a resolução de problemas.

Apesar de sua alta complexidade, o CubeIDE foi funcional durante o desenvolvimento inicial, que usava controladores de motores de drones ativados por sinal *PWM*. As duas primeiras versões do firmware foram desenvolvidas nesse ambiente sem dificuldades significativas. No entanto, ao migrar para o uso da biblioteca SimpleFOC, projetada originalmente para a plataforma Arduino IDE, surgiram problemas de compatibilidade. A adaptação do código da biblioteca SimpleFOC para uso no CubeIDE mostrou-se inviável, considerando que a biblioteca depende de toda a base de código do Arduino IDE.

Essas limitações, somadas à necessidade de garantir a continuidade do projeto mesmo diante da rotatividade dos membros da equipe, evidenciaram a importância de buscar uma IDE alternativa que atendesse melhor às demandas do projeto.

Dada a necessidade de utilização da biblioteca SimpleFOC, desenvolvida originalmente para o Arduino IDE, esse foi o próximo ambiente de desenvolvimento explorado. A terceira versão do firmware foi desenvolvida nessa IDE e foi a versão utilizada durante a Competição Brasileira de Robótica de 2024.

Embora tenha sido possível implementar grande parte do projeto nesse ambiente, ele também se provou limitado em vários sentidos. Algumas funcionalidades que não são obrigatórias para o funcionamento do firmware mas contribuem para a sua qualidade e longevidade não estão presentes nessa IDE ou são limitados e demasiadamente complexos.

Alguns exemplos são: a dificuldade de configurar a frequência de clock de microcontroladores da linha STM32, a necessidade de selecionar bibliotecas e suas versões manualmente, o suporte para o desenvolvimento e execução de testes, a personalização de comandos de *upload* de *firmware* para o microcontrolador, entre outras funcionalidades. Processos complexos representam um alto risco de problemas futuros, diante da rotatividade de estudantes na equipe.

A solução final adotada foi a utilização do Visual Studio Code em conjunto com a extensão PlatformIO, que daqui em diante será referido como PlatformIO por conveniência. Essa abordagem permitiu a integração do código desenvolvido nas duas IDEs anteriores, oferecendo um equilíbrio entre controle e curva de aprendizado e garantindo o acesso a todas as funcionalidades essenciais de ambas as IDEs. Além disso, o PlatformIO possui todas as funcionalidades faltantes no ArduinoIDE atendendo as necessidades atuais e futuras do projeto. A Tabela 3 mostra uma comparação das três IDEs.

Adicionalmente, o Visual Studio Code é uma ferramenta amplamente utilizada por estudantes, o que contribui para facilitar o entendimento do projeto. Essa familiaridade prévia com a IDE diminuirá o tempo necessário para a preparação de novos membros da equipe AraraBots, facilitando sua integração e aumentando a agilidade na realização de alterações e melhorias no código.

Tabela 3 – Funções e características mais notáveis de cada IDE para a aplicação. Vale ressaltar que a curva de aprendizado está diretamente relacionada com a quantidade de controle que se tem sobre o microcontrolador em cada ambiente de desenvolvimento.

	CubeIDE	Arduino IDE	PlatformIO
Curva de aprendizado	Alta	Baixa	Média
Suporta SimpleFOC	Não	Sim	Sim
Frequência de clock do microcontrolador	Ajustável	Fixa	Ajustável
Suporte para testes unitários	Não	Não	Sim
Instalação automática de bibliotecas	Não	Não	Sim
Personalização de comando de *upload*	Não	Não	Sim

3.3.2 Sistema de controle

A proposta inicial do projeto consistia em incorporar um módulo de giroscópio e acelerômetro para atuar como sensores de velocidade angular e linear do robô, respectivamente, viabilizando a implementação de um sistema de controle diretamente em firmware. Informações sobre o sensor utilizado e sua implementação eletrônica foram detalhadas na seção 3.2.5.

Criou-se uma rotina de ajuste do sensor, onde este ficava totalmente estático e várias leituras eram feitas, com as quais calculava-se o erro médio do sensor, que era subtraído de qualquer leitura posterior para ajustá-la.

No entanto, após a implementação, constatou-se que o acelerômetro apresentava limitações significativas. Embora esse sensor meça a aceleração linear do robô em três eixos, sua aplicação prática revelou-se pouco útil por ser inconsistente e possuir muito ruído. Além disso, a velocidade linear, que permite a implementação de um sistema de controle de maneira simples, requer a discretização e somatória dos valores do sensor e esse processo acentua as inconsistências e os ruídos.

Em relação ao giroscópio, que mede a velocidade angular do robô em torno de três eixos, também foram observados problemas similares. Nessa caso, a integração dos dados para obter a posição angular introduzia um erro sistemático nas leituras, o que, com o tempo, resultava em desvios acumulados significativos.

Durante a análise de alternativas, considerou-se a possibilidade de complementar o giroscópio com outros sensores. Uma das opções avaliadas foi o magnetômetro, que mede a posição angular em relação ao campo magnético terrestre. Contudo, a presença de diversas bobinas nos motores e no mecanismo de chute do robô poderia gerar interferências magnéticas, comprometendo sua precisão.

Outra possibilidade analisada foi enviar a posição angular detectada pela câmera juntamente com os comandos enviados ao robô. No entanto, a alta latência de cerca de 100 ms, juntamente com a baixa capacidade computacional do microcontrolador, acarretou na decisão de implementar um sistema de controle mais robusto em software no computador,

antes do envio da mensagem para o robô.

A implementação do sistema de controle em software foge do escopo deste trabalho, caso o leitor queira mais informações, os *team description papers* (TDPs) (ITO; ANDO, 2022), (ITO MARINA SHIBATA; ANDO, 2023) e (HOFFMANN MARKUS LIERET; HAUCK, 2013) apresentam mais detalhes.

3.3.3 Comunicação sem fio

Para que a comunicação entre o transmissor e os robôs funcione, é preciso definir o formato das mensagens transmitidas. A decisão passou por várias iterações e no final foi decidido algo bem simples, principalmente pela falta de sensores de movimentação embarcados, que limitam as funcionalidades viáveis em firmware.

O primeiro byte da mensagem é um identificador do robô, utilizado pelo robô para verificar se ele é o destinatário da mensagem. Caso não seja o destinatário, o robô descarta a mensagem e continua a procurar pela próxima mensagem.

Como os robôs não possuem nenhuma informação do ambiente à sua volta, cada robô recebe somente informações da direção que ele deve seguir. O eixo x diz respeito à movimentação do robô para os lados, enquanto o eixo y refere-se aos movimentos para frente e para trás.

Tabela 4 – Formato da mensagem entre o transmissor e os robôs.

Sigla	Descrição	Tamanho	Tipo
ID	Identificador do robô	1 byte	char
VX	Velocidade linear desejada no eixo x	4 bytes	float
VY	Velocidade linear desejada no eixo y	4 bytes	float
VT	Velocidade angular desejada	4 bytes	float
KI	Sinal de chute	1 byte	integer

A mensagem definida na Tabela 4 requer 14 bytes, ou 112 bits, para cada robô. Para garantir o bom funcionamento dos robôs é preciso cerca de 60 mensagens por segundo e durante uma partida são usados três robôs simultaneamente. Portanto é necessária uma taxa de transferência de, no mínimo, 20.160 bits por segundo ($112 \times 60 \times 3$) para que os três robôs funcionem corretamente.

A comunicação sem fio foi realizada por meio do módulo NRF24L01, já abordado na seção 3.2.6. Existem diversas bibliotecas amplamente testadas que facilitam a utilização desse módulo, sendo utilizadas nesse projeto: a STM32RF24, desenvolvida pela equipe de robótica ThunderRatz, da Escola Politécnica da USP, nas versões implementadas na CubeIDE; e a RF24, disponível para o framework Arduino, aplicada nas versões desenvolvidas nas plataformas Arduino IDE e PlatformIO.

Antes de enviar e receber mensagens, o módulo precisa ser configurado pelo microcontrolador. A seguir, são apresentadas as configurações feitas e suas motivações:

- Potência: O aumento da potência aumenta o alcance do sinal do módulo sem fio, mas com a desvantagem de apresentar dados corrompidos com mais frequência. Considerando a desvantagem e que, na competição, o transmissor não fica a mais que 6 m de distância do robô no pior caso, utilizou-se o valor mínimo para a potência.
- Taxa de transmissão de dados: Como exposto anteriormente são necessários somente 20kbps para o bom funcionamento dos robôs, então foi selecionada a menor taxa de transmissão possível que atenda a demanda, 25kbps. Satisfazendo a necessidade e diminuindo as chances de possíveis falhas em taxas maiores.
- Endereço de comunicação e canal: Para que dois dispositivos se comuniquem eles precisam estar no mesmo canal e o transmissor precisa enviar a mensagem para o endereço do receptor. O canal foi escolhido junto da organização da competição de modo que não haja conflito com outras equipes e pode ser facilmente trocado no firmware. Foi escolhido um único endereço de forma aleatória e este é utilizado por todos os robôs, cada robô decide se é o destinatário da mensagem comparando seu ID com o primeiro byte da mensagem enviada, como pode ser visto na tabela 4.

3.3.4 Cinemática Inversa

As equações de cinemática inversa da seção 2.2 assumem que a movimentação no eixo x, no eixo y e a rotação em torno do próprio eixo pode ser feita de maneira independente. Como as rodas não são dispostas simetricamente em torno do eixo x (conforme a seção 3.1.3), a movimentação no sentido deste eixo também gera uma rotação, já que as componentes do eixo y dos motores não se cancelam.

Dado tal problema, uma solução testada foi, assim que receber um comando, o robô calcular a rotação gerada pelo futuro movimento no eixo x e compensar tal rotação no próprio comando recebido, adicionando ou subtraindo na rotação em torno do próprio eixo contida no comando.

A seguir estão os cálculos usados para implementar a solução. Usando as variáveis como definido na Tabela 1 e a equação 2.1, é possível descobrir a rotação gerada por um comando no eixo x, como dado na Equação 3.1:

$$V_{\theta x} = \sum_{i=1}^{4} \frac{\sin(\alpha_i) \cdot V_x}{D} \tag{3.1}$$

A intenção do teste era subtrair o valor da rotação gerado pelo movimento no eixo x ($V_{\theta x}$) do comando de rotação mandado para o robô (V_θ), e refazer o cálculo da velocidade angular das rodas (equação 2.3).

O resultado desse cálculo mostra que todas as rodas devem girar na mesma velocidade para que o robô não gire, com isso ele deixa de se deslocar no eixo x e sim em uma diagonal entre o eixo x e o eixo y.

Sendo assim, o movimento do robô no eixo x não pode ser corrigido com o cálculo da cinemática inversa e deverá ser feito pelo sistema de controle (seção 3.3.2).

3.3.5 Controle dos motores

Para controlar os motores, a utilização da biblioteca SimpleFOC compreende os seguintes passos:

1. Criação de objetos de motores e *drivers*
   Nesse passo, como é feito o controle de velocidade sem nenhum tipo de sensor, *velocity open-loop control*, é de extrema importância que as especificações dos motores sejam informadas. Por conta disso, a contagem de pares de polos, a resistência da fase e o valor de KV do motor, que dita a velocidade em rpm que o motor gira para cada volt da tensão de sua alimentação, são usados como parâmetros na criação do objeto do motor.

2. Configuração de opções
   Primeiro o *driver* precisa da tensão de entrada utilizada, depois é configurado o tipo de controle que será utilizado, *velocity open-loop control*, e por último informa-se qual motor está conectado no *driver* usando o objeto do motor citado anteriormente.

3. Movimentação
   Por fim é preciso especificar a velocidade desejada do motor sempre que haver uma atualização e ajustar o limitador de tensão proporcionalmente.

Utilizando o valor de 60 kV do motor (60 rpm/V) o limitador de tensão da biblioteca SimpleFOC é ajustado precisamente com a tensão necessária para o motor girar na velocidade obtida da cinemática inversa.

3.3.6 Comunicação serial

A biblioteca *SimpleFOC* tem um custo computacional relativamente alto para o microcontrolador STM32F103C8T6, com isso, somente um não é capaz de gerar os três sinais *PWM* necessários para cada motor com a precisão que o método exige. Assim, foram usados dois STM32F103C8T6 para distribuir o processamento do robô, cada um

controlando dois motores diferentes juntamente com uma parte das outras funcionalidades do robô.

O rádio foi conectado a somente um dos microcontroladores (*RadioSTM*) e foi adicionada uma comunicação serial usando o protocolo UART, para repassar dados necessários para o funcionamento do outro microcontrolador (*KickerSTM*). O microcontrolador conectado ao módulo de rádio recebe comandos do computador, aplica a cinemática inversa e envia ao outro microcontrolador as velocidades dos motores necessárias e o sinal do chute. O formato das mensagens pode ser visto na Tabela 5.

Tabela 5 – Padrão de mensagem serial entre os microcontroladores.

Sigla	Descrição	Tamanho	Tipo
W3	Velocidade angular desejada do motor 3	4 bytes	float
W4	Velocidade angular desejada do motor 4	4 bytes	float
KI	Sinal de chute	1 bytes	integer

A comunicação serial entre os microcontroladores é extremamente ruidosa e imprecisa, e alguns métodos foram usados para diminuir a quantidade de erros e seus impactos. O primeiro método utilizado foi limitar o *RadioSTM* para escrever somente quando o *KickerSTM* sinalizar que está pronto para ler, assim somente uma mensagem fica no *buffer* e diminui-se a chance de sobrescrever dados. A implementação foi feita por meio do mesmo serial, o *KickerSTM* escreve um único byte "R"sempre que está pronto para ler uma nova mensagem.

Além disso, o *KickerSTM* só começa a ler assim que a quantidade de *bytes* em seu *buffer* seja maior ou igual ao número de bytes que espera ler, dessa forma, menos bytes são perdidos. Tal método, no entando, apresentou erros nos valores enviados.

Outra tentativa de minimizar o impacto desses erros foi assegurar que todo valor de velocidade dos motores recebido deve estar dentro de uma faixa de valores para ser usado. Caso o valor recebido se encontre fora dessa faixa, o microcontrolador não modifica a velocidade e descarta o valor. Essa faixa foi definida baseada na velocidade máxima do motor, $60KV * 10V = 600rpm$.

Um outro método de detecção de erro considerado foi *Cyclic Redundancy Check* (CRC), com a biblioteca CRC8.h, mas seu cálculo é relativamente complexo e adiciona cerca de 1 ms ao tempo de execução de uma iteração dos microcontroladores. Esse aumento no período faz com que o acionamento dos motores tenha mais travamentos e por conta disso, foi decidido não utilizá-lo.

A velocidade de transmissão (*baudrate*) usada foi de 230400 bps, por ser a menor velocidade que mantém o tempo de execução de uma iteração dos microcontroladores abaixo de 2ms, recomendado para melhor funcionamento da biblioteca SimpleFOC.

3.3.7 Arquitetura do firmware

Como comentado na seção 3.3.1, o código foi feito para a plataforma Arduino usando a Extensão PlatformIO do Visual Studio Code na linguagem de programação C++. Para integrar o microcontrolador à plataforma Arduino foi utilizada a biblioteca STM32Duino. A implementação utiliza conceitos de orientação a objetos, como herança, métodos e atributos como sua principal forma de organização.

A Figura 22 mostra o fluxograma do firmware.

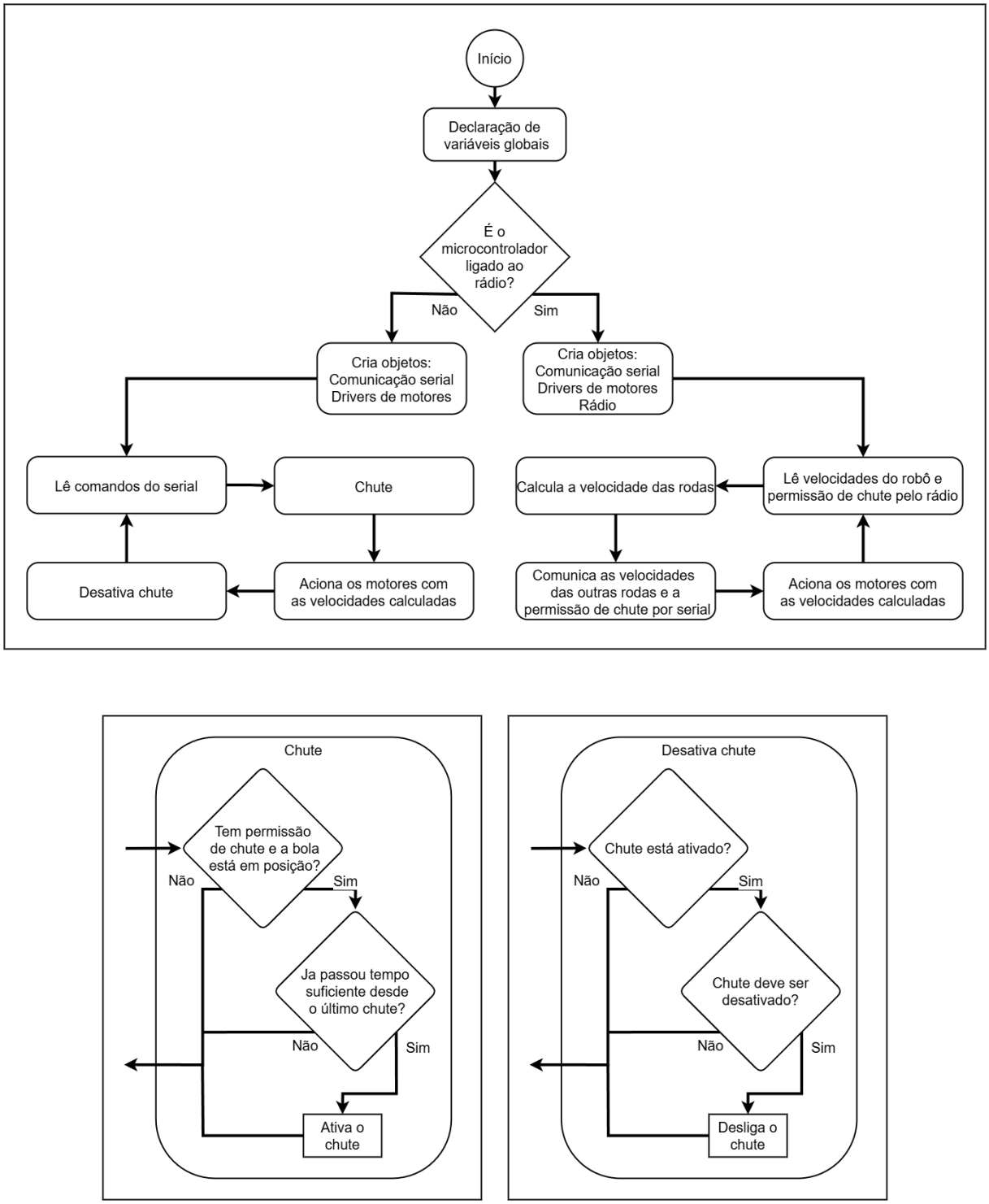


Figura 22 – Fluxograma do firmware.

Os firmwares de ambos os microcontroladores estão presentes no mesmo projeto para que seja simples fazer alterações. Cada um é uma classe que herda de uma mesma super classe, diminuindo a repetitividade do código.

4 Resultados

Nesta seção são apresentados os resultados obtidos a partir da aplicação da metodologia previamente descrita. As funcionalidades implementadas, o desempenho observado nos testes práticos e as principais limitações identificadas são expostas de forma organizada em subseções, de modo a permitir uma análise clara e objetiva da proposta desenvolvida.

(a) (b)

Figura 23 – Fotos do robô. (a) Foto isométrica frontal. (b) Foto isométrica traseira.

4.1 Sistema de Chute

A seguir, são descritos os resultados obtidos na avaliação do desempenho do mecanismo de chute implementado. Para isso, foram realizados testes práticos com o objetivo de medir a velocidade do chute e verificar a eficiência do circuito de acionamento e detecção da bola. Além disso, são discutidas as limitações observadas, bem como sugestões de melhorias para futuras versões do projeto, especialmente no contexto de competições como a RoboCup, nas categorias SSL e SSL-EL.

A velocidade do chute obtida durante os testes foi de aproximadamente 1,5 m/s. Para essa medição, foi utilizado o software Tracker, que permitiu a observação do movimento da bola por meio da análise de vídeo.

Na categoria SSL, essa velocidade pode ser considerada baixa, uma vez que a regulamentação permite velocidades de até 6,5 m/s, enquanto que na SSL-EL a velocidade máxima é 3 m/s. Entretanto, durante a RoboCup 2024, na categoria SSL-EL, a presença desse mecanismo conferiu destaque ao robô em relação às demais equipes. Isso se deve ao fato de que, por se tratar de uma categoria iniciante, a maioria dos times não apresentou mecanismos de chute em seus projetos.

Todavia, este projeto ainda não é adequado para níveis mais avançados e deverá ser aprimorado conforme as exigências futuras. É imprescindível a utilização de uma tensão de acionamento do solenoide superior a 50V, além disso, o desempenho do solenoide utilizado mostrou-se insatisfatória, fornecendo uma força máxima de apenas 12 N, conforme especificado pelo fabricante. Assim, recomenda-se a utilização de um novo solenoide que apresente melhores características mecânicas e elétricas.

Adicionalmente, o uso de outras fontes de acionamento, como capacitores de alta capacidade, pode ser considerado em projetos futuros para melhorar o desempenho do mecanismo de chute.

Com relação ao circuito de acionamento, este se mostrou eficaz para o escopo do projeto. O sensor infravermelho foi capaz de identificar com precisão quando a bola estava posicionada para chute, e o componente IRLZ44N apresentou uma resposta rápida e eficiente como chave de acionamento do solenóide.

O teste foi feito com o microcontrolador usado no projeto, STM32F103C8T6, e com o mesmo firmware utilizado no robô.

4.2 Movimentação

Nesta seção, é descrita uma série de testes realizados com o projeto final do robô, utilizando o SimpleFOCMini v1.1 para o controle dos motores. Os testes foram feitos utilizando uma câmera superior para gravar a trajetória do robô. Para a obtenção das imagens, foi feito o traçado da rota percorrida e comparado com o caminho esperado do movimento.

A fim de testar apenas a movimentação do robô - sem a necessidade de alterar rotas ou considerar obstáculos - foi implementada a possibilidade de gerar, no computador, os valores de V_x, V_y e V_θ a partir das teclas no teclado.

Como já apontado anteriormente, o firmware do robô tem a responsabilidade de transformar as velocidades desejadas (V_x, V_y e V_θ), por meio da cinemática inversa, nas velocidades angulares de cada roda.

A trajetória do movimento pode ser observada na Figura 24. Mesmo sendo um sistema de malha aberta, sem sensores, o controle orientado por campo consegue fazer

com o que o acionamento dos motores seja extremamente preciso enquanto não existirem obstáculos para interferirem com o movimento do robô. O robô ser simétrico em relação ao eixo y também faz com que nenhuma rotação indesejada seja adicionada ao movimento.

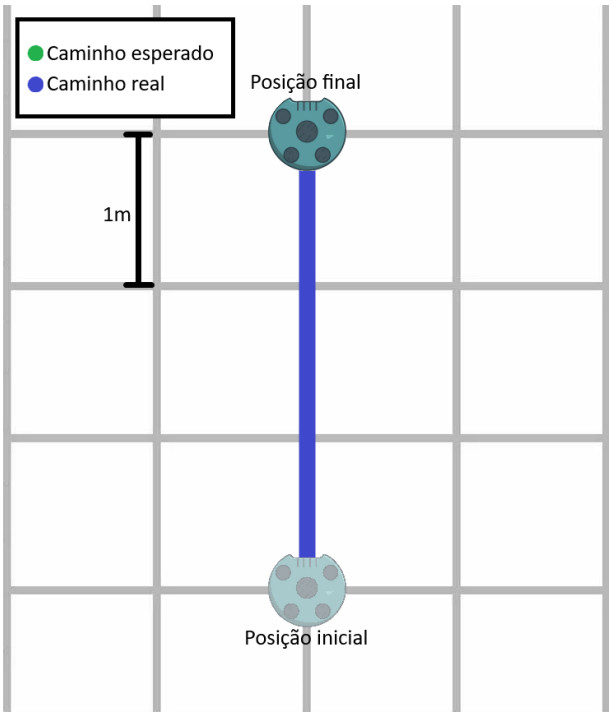


Figura 24 – Teste da movimentação para frente.

O deslocamento para trás apresenta os mesmos resultados do deslocamento para a frente por ser virtualmente idêntico, como pode ser observado na Figura 28.

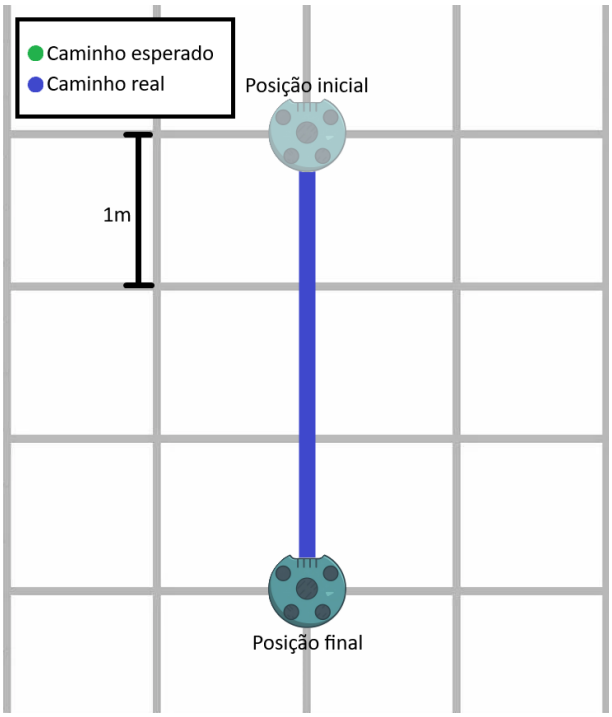


Figura 25 – Teste da movimentação para trás.

No teste de deslocamento lateral (Figura 26), como esperado, o robô exibiu certa rotação sobre seu próprio eixo, girando no sentido horário ou anti-horário, a depender do deslocamento esperado ser para a direita ou esquerda, respectivamente. Conforme descrito na seção 3.3.4, era previsto que a movimentação lateral não fosse efetiva sem um sistema de controle.

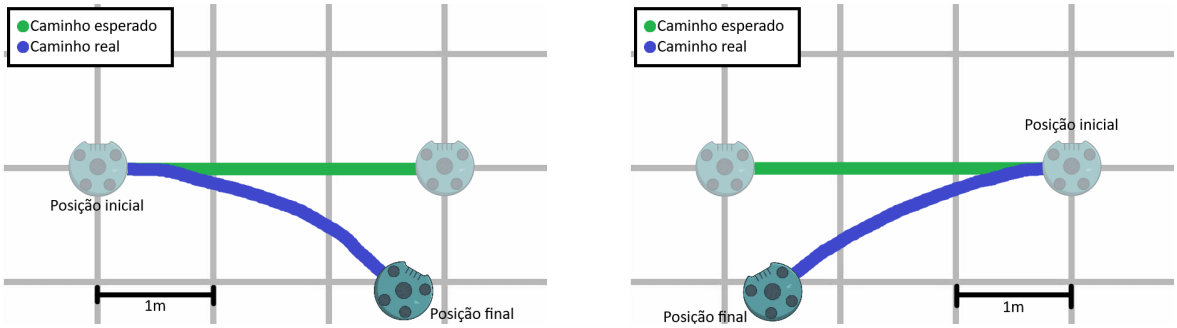


Figura 26 – Testes de movimentação lateral. (a) Movimentação para a direita. (b) Movimentação para a esquerda.

4.3 Comunicação

Esta seção mostrará os testes realizados sobre as comunicações e seus resultados. São inclusos aqui testes sobre a comunicação via rádio entre o computador e o robô, e via serial entre os dois microcontroladores do robô.

4.3.1 Comunicação via rádio

Para a execução deste teste foi conectado um microcontrolador a um computador via serial USB e a cada 16 ms esse microcontrolador recebia uma mensagem do computador e a enviava via rádio. O período de 16 ms permite com que o computador mande até 60 mensagens por segundo para o robô facilitando uma futura implementação de um sistema de controle no computador utilizando a imagem da câmera como sensor. Os valores enviados foram constantes, um comando assim como descrito na Tabela 4 com $VX = 1,01$, $VY = 1,01$, $VT = 1,01$ e $KI = 10$, escolhidos por serem relativamente complexos em binário, facilitando a detecção de erros.

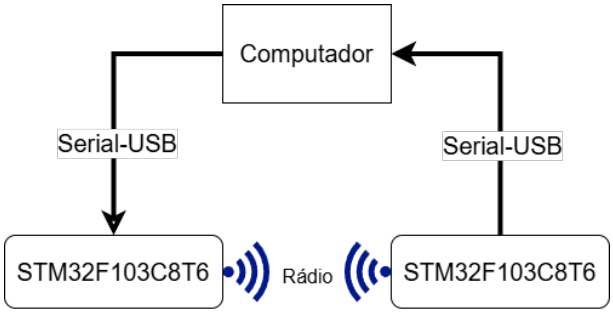


Figura 27 – Diagrama da conexão usada para o teste da comunicação via rádio.

Durante aproximadamente 5 minutos, o microcontrolador do robô recebeu as mensagens via rádio e as mandou via serial-USB de volta para o computador para que fosse checada a consistência da comunicação. Durante esse tempo, apenas 2% das mensagens recebidas mostraram algum tipo de erro.

4.3.2 Comunicação serial

Para esse teste, foi conectado a cada microcontrolador um novo serial-USB ligado a um computador. Sempre que um microcontrolador enviava ou recebia uma mensagem do outro, eles também enviavam o que foi enviado ou recebido para o computador via o serial extra.

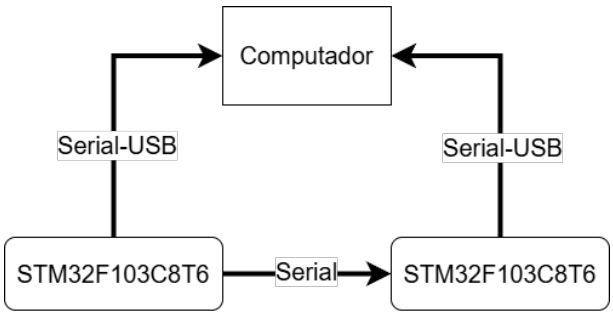


Figura 28 – Diagrama da conexão usada para o teste da comunicação via serial.

Para simular os comandos enviados pelo computador foi usado o seguinte código para mudar os valores de VY e KI (Tabela 4) constantemente:

```
if (vy > 10.0) {
    vy = -10.0;
    if (kik_sig == 1)
        kik_sig = 0;
    else
        kik_sig = 1;
}
else
    vy += 0.01;
```

Como essa comunicação é inconsistente, foram feitos dois testes de aproximadamente 10 minutos. Ao comparar o que foi enviado com o que foi recebido, houve diferenças em 4,5% das mensagens no primeiro teste e 8,9% no segundo.

4.4 Desempenho do Firmware

O desempenho do firmware é um dos grandes desafios desse projeto, impactando diretamente na eficiência de todo o robô. Por usar a biblioteca SimpleFOC, o desempenho se torna ainda mais importante, pois a qualidade do acionamento dos motores é inversamente proporcional ao período de iteração, sendo recomendado no máximo 2 ms.

Enquanto todas as funcionalidades de ambos os microcontroladores estavam ligadas e funcionando, foi adicionado no final do método *run*, tanto do microcontrolador ligado ao rádio (RadioSTM) quanto ao microcontrolador ligado ao sistema de chute (KickerSTM), o seguinte código para a medição do tempo de execução:

```
uart->println(String(millis() - start));
start = millis();
```

O trecho de código envia quanto tempo passou desde a última atualização da variável *start* em milissegundos e logo depois a atualiza. As Figuras 29 e 30 mostram os resultados deste teste.

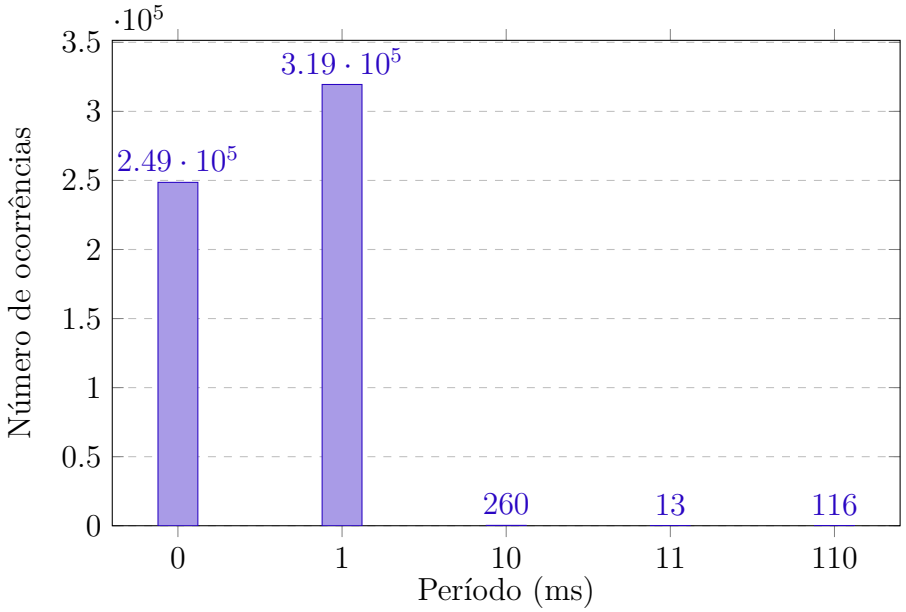


Figura 29 – Períodos de execução do método *run* do microcontrolador conectado ao sistema de chute, medidos durante 5 minutos.

A média de tempo de execução de uma iteração do método *run* do RadioSTM foi de 0,43 ms (desvio padrão de 1,21 ms) e do Kicker STM foi de 0,59 ms (desvio padrão de 1,65 ms).

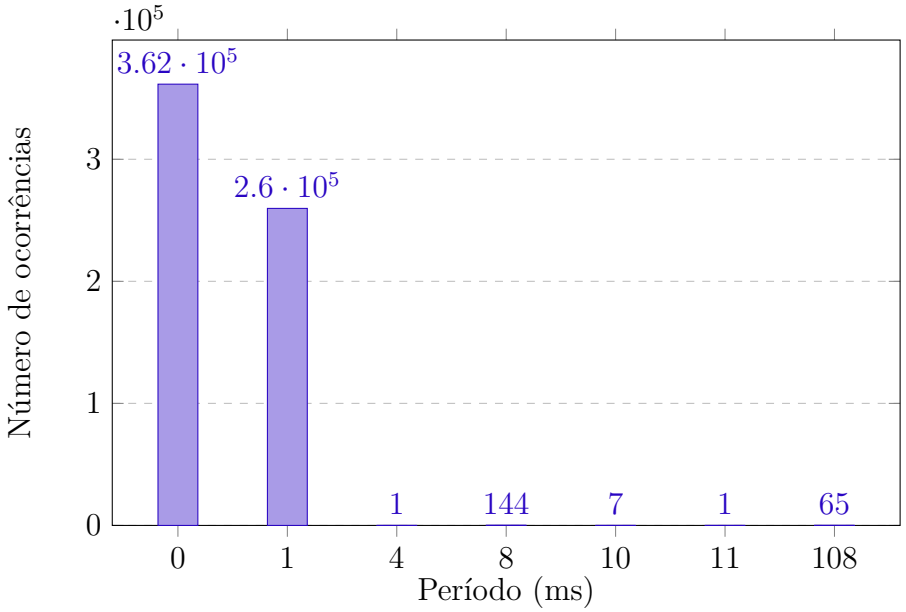


Figura 30 – Períodos de execução do método *run* do microcontrolador conectado ao rádio, medidos durante 5 minutos.

4.5 Cálculo de Custos

Um dos grandes objetivos do projeto foi desenvolver um robô de baixo custo, possibilitando que, além da equipe Ararabots, outras equipes de robótica pequenas tivessem a percepção de que é possível competir na categoria SSL-EL.

Dessa forma, nessa seção serão apresentados os valores gastos no robô (Tabela 6). Vale ressaltar que os valores foram calculados em Setembro de 2024, época na qual foram adquiridos todos os equipamentos necessários, e que muitos dos equipamentos usados não foram encontrados em território brasileiro, tornando necessária sua importação.

Tabela 6 – Custos dos componentes do robô. Outros se refere às peças que tem custo baixo demais para serem citadas individualmente (resistores, diodos, parafusos, fios, botões, entre outros). Os valores apresentados não contêm taxas, impostos e fretes.

Componente	Quantidade	Valor Total (US$ FOB)
STM32F103C8T6	2	10,94
NRF24L01 + antena	1	3,65
Soletec Série C PL 100%	1	11,67
IFlight GM4108	4	143,60
SimpleFOCMini v1.1	4	55,28
Bateria Leão 5S	1	45,60
Step Up (50v)	1	10,94
Step Down (5v)	1	2,19
Filamento ABS	624,21g	6,22
Placa de fenolite ilhada	1	4,56
Mosfet IRLZ44N	1	3,65
Arcos de rodas	4	18,24
Sensor infravermelho	1	1,37
Outros	-	7,3
Valor Total	**-**	**325,21**

O cálculo da quantidade de filamento gasto foi feito por meio de uma simulação de impressão na impressora 3D *K1 MAX* no *software Slicer*, no qual foi concluído que foram gastos aproximadamente 624,21g de filamento ABS. Como um carretel do filamento GTMax ABS premium custou R$ 76,44 e possuía 1,4 kg, a quantidade utilizada totalizou R$ 34,08.

Um fator relevante a ser considerado é que os valores apresentados são o total gasto para a montagem de um robô, não contabilizando os diversos materiais usados em todas as fases de projeto do mesmo. O custo de produção do robô também depende da quantidade de ferramentas necessárias, que precisam ser adquiridas, como impressora 3D, estação de solda, fresadora PCB, entre outras. Como estas ferramentas variam por uma infinidade de preços e também podem ser emprestadas, alugadas ou até substituídas por métodos mais simples, seu preço não foi considerado no orçamento.

Ao somar todos os valores apresentados na Tabela 6, chega-se ao valor de R$ 1783,10. Esse valor normalmente é gasto somente em motores para movimentação de equipes da categoria regular de SSL, como (OMMER ANDRé RYLL, 2024) e (FRANCA BEATRIZ BARROS; BARROS, 2024).

4.6 Resultados na Competição

A equipe Ararabots levou o robô deste projeto para competir na categoria SSL-EL na competição de robótica RoboCup de 2024, em Goiânia. Durante a competição, a

equipe avançou da fase de grupos para a fase eliminatória e alcançou o quinto lugar dentre as dez equipes competidoras.

Por ser a primeira competição que inclui a categoria SSL-EL e nenhuma equipe ter experiência prévia com a categoria, as equipes que tinham robôs funcionando tiveram bastante dificuldade em controlá-los. Sendo assim, a organização do evento decidiu que jogos sem gols teriam novos critérios de desempate, como quantidade de robôs que conseguiam jogar e capacidade de controlar o robô e movimentar a bola. Com esses critérios de desempate, a equipe Ararabots conseguiu ganhar 4 partidas, apesar de não marcar nenhum gol durante a competição.

Conclusão

Este trabalho teve como objetivo desenvolver um robô de baixo custo para competir na categoria SSL-EL da RoboCup Brasil. Assim, o robô desenvolvido é capaz de receber informações sem fio, se movimentar e chutar a bola. Durante todo o projeto, buscou-se utilizar componentes e soluções mais acessíveis e baratas, sem comprometer seu desempenho na categoria.

Os resultados obtidos para o sistema de chute, tanto em termos de hardware quanto de firmware, demonstraram desempenho satisfatório e compatível com os requisitos da categoria SSL-EL em competições nacionais. No entanto, para categorias superiores da RoboCup, torna-se necessário o desenvolvimento de soluções mais robustas e com maior nível de desempenho, a fim de atender às exigências técnicas e competitivas desses níveis.

A movimentação mostrou-se eficiente e eficaz, apesar de ser um sistema sem sensores. Ainda assim, a implementação de sensores de rotação do motor tornaria o projeto mais robusto e consistente, diminuindo a chance de falhas durante momentos cruciais de partidas, como disputas de bola, onde dois ou mais robôs se chocam. Também é importante ressaltar que é possível implementar um sistema de controle para a movimentação no computador que envia as mensagens para os robôs, usando a câmera como sensor. Apesar de não ser ideal por conta do atraso da imagem em relação ao movimento, ainda é uma opção viável e de baixo custo.

Quanto à comunicação via rádio, os resultados do teste mostraram que é extremamente confiável e uma ótima solução para o problema. Estes dados corroboram com a experiência que a equipe teve durante a competição RoboCup Brasil 2024, em que a comunicação não apresentou problemas após o módulo ser alimentado corretamente.

O mesmo não pode ser dito em relação à comunicação serial entre os microcontroladores do robô, pois os dois testes realizados mostram inconsistência na comunicação e isso acarreta problemas de movimentação e tempo de reação do chute. Assim, é necessário realizar uma investigação mais aprofundada para encontrar a causa desses erros de comunicação e considerar alternativas que apresentem mais confiança.

Os custos totais do projeto também se destacam por estarem significativamente abaixo dos valores normalmente observados em equipes da categoria SSL. Apesar disso, percebeu-se que pequenos aumentos no orçamento podem trazer melhorias significativas em diversas áreas como será descrito a seguir.

Como continuação deste trabalho, sugere-se aperfeiçoar o sistema de chute com o objetivo de atingir velocidades mais elevadas, compatíveis com níveis de competição mais

exigentes. Além disso, é indicado considerar a substituição dos microcontroladores atuais por um único modelo que consiga operar todos os componentes e funcionalidades do sistema satisfatoriamente, reduzindo a complexidade do sistema e eliminando a necessidade de comunicação serial entre múltiplos microcontroladores.

Algumas possíveis melhorias que podem contribuir para o controle preciso da movimentação e navegação do robô são: a utilização de *encoders* como sensores da movimentação real dos motores, e estudar a possibilidade de diminuição das rodas, aumentando torque, facilitando o controle e diminuindo o espaço ocupado por tais componentes.

Dado o exposto, conclui-se que esse trabalho deu uma contribuição importante para que a equipe Ararabots da UFMS consiga continuar a competir na categoria SSL-EL, visando colocações melhores e, talvez, a promoção para a categoria SSL regular no futuro.

Referências

BIDA, V. M.; SAMOKHVALOV, D. V.; AL-MAHTURI, F. S. Pmsm vector control techniques — a survey. In: *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. [S.l.: s.n.], 2018. p. 577–581.

CABRAL, E. Robôs industriais. 2010. Disponível em: <http:-//sites.poli.usp.br/p/eduardo.cabral/Cinem%C3%A1tica%20Direta-.pdf>http://sites.poli.usp.br/p/eduardo.cabral/Cinem%C3%A1tica%20Direta.pdf.

DUMITRUF, P. et al. 2020 team description paper: Ubc thunderbots. 2020.

FRANCA BEATRIZ BARROS, C. G. C. S. C. M. A. D. C. B. D. X. E. A. F. P. H. A. B. L. H. C. M. A. M. A. M. P. M. V. M. V. J. G. M. J. R. S. J. V. C. J. L. P. H. S. P. P. O. R. R. R. M. T. T. V. D. V. A. A.; BARROS, E. Robôcin small size league extended team description paper for robocup 2024. *RoboCup*, 2024.

HIJIKATA, M.; MIYAGUSUKU, R.; OZAKI, K. Wheel arrangement of four omni wheel mobile robot for compactness. *Applied Sciences*, v. 12, n. 12, 2022. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/12/12-/5798>https://www.mdpi.com/2076-3417/12/12/5798.

HOFFMANN MARKUS LIERET, S. K. M. E. P. N. M.; HAUCK, A. Er-forceteam description paper for robocup 2013. *RoboCup*, 2013.

ITO, M.; ANDO, Y. Robodragons 2022 extended team description. *RoboCup*, 2022.

ITO MARINA SHIBATA, S. A. M.; ANDO, Y. Robodragons 2023 extended team description. *RoboCup*, 2023.

JAZAR, R. N. *Theory of Applied Robotics: Kinematics, Dynamics, and Control*. [S.l.]: Springer Publishing Company, Incorporated, 2007. ISBN 0387324755.

MARIAPPAN, M. et al. Simultaneous rotation and translation movement for four omnidirectional wheels holonomic mobile robot. In: . [S.l.: s.n.], 2014. p. 69–73.

OMMER ANDRé RYLL, M. R. M. G. N. Tigers mannheim extended team description for robocup 2024. *RoboCup*, 2024.

PHUNOPAS, A.; INOUE, S. Motion improvement of four-wheeled omnidirectional mobile robots for indoor terrain. *Journal of Robotics, Networking and Artificial Life*, v. 4, p. 275, 03 2018.

ROBOCUP. *Rules of the RoboCup - Small Size League - Entry Level*. 2024. Disponível em: <https://cbr.robocup.org.br/wp-content/uploads/2024/08/sslrules-.pdf>https://cbr.robocup.org.br/wp-content/uploads/2024/08/sslrules.pdf.

RYLL, A.; JUT, S. Tigers mannheim extended team description for robocup 2020. *RoboCup*, 2020. Disponível em: <https://ssl.robocup.org/wp-content/uploads/2020/03/2020_ETDP_TIGERS.pdf>https://ssl.robocup.org/wp-content/uploads/2020/03/2020_ETDP_TIGERS.pdf.

SICILIANO, B. et al. *Robotics: Modelling, Planning and Control*. Springer London, 2010. (Advanced Textbooks in Control and Signal Processing). ISBN 9781849966344. Disponível em: <https://books.google.com.br-/books?id=vNpdewAACAAJ>https://books.google.com.br/books?id=vNpdewAACAAJ.

SOFWAN, A. et al. Development of omni-wheeled mobile robot based-on inverse kinematics and odometry. In: *2019 6th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE)*. [S.l.: s.n.], 2019. p. 1–6.