Gabriel Cosme Barbosa, Matheus Rezende Pereira

Desenvolvimento de um robô de alta performance para a categoria Micro Mouse

Brasil

2024

Gabriel Cosme Barbosa, Matheus Rezende Pereira

Desenvolvimento de um robô de alta performance para a categoria Micro Mouse

Projeto de um robô de alto desempenho para a categoria MicroMouse. O projeto abrange áreas mecânicas, elétricas e computacionais, integradas para desenvolver um robô capaz de explorar labirintos de forma autônoma e eficiente.

Universidade de São Paulo - USP

Escola Politécnica

Programa de Graduação

Orientador: Bruno de Arantes Basseto

Brasil

2024

Agradecimentos

Os agradecimentos principais são direcionados aos nossos pais, pelo apoio incondicional durante toda a nossa jornada acadêmica.

Agradecemos também ao nosso amigo e colega Pedro de Santi, que iniciou o projeto conosco e, mesmo à distância durante seu intercâmbio, continuou contribuindo de forma significativa.

Ao professor Bruno Basseto, pela orientação dedicada ao longo deste trabalho.

Agradecemos também a nossos amigos Claudio Dardengo, Eduardo Barreto e João Retti pelo suporte no processo de pesquisa e desenvolvimento.

Somos gratos à equipe ThundeRatz de robótica, que nos forneceu patrocínios, equipamentos, estrutura, e onde nasceu nossa paixão pela robótica.

Agradecemos também à STMicroelectronics, pelo fornecimento de componentes fundamentais e à escola politécnica da Universidade de São Paulo, pela formação que recebemos ao longo da graduação.

Resumo

Esta monografia descreve o desenvolvimento de um robô autônomo de alto desempenho para a categoria MicroMouse. O projeto integra as áreas mecânica, elétrica e computacional para criar um robô capaz de navegar de maneira eficiente em labirintos complexos. Utilizando o microcontrolador STM32G474 e firmware desenvolvido em C++, foram incorporados algoritmos avançados de mapeamento, planejamento de trajetórias e navegação. A implementação final atende aos requisitos funcionais e não funcionais da competição, destacando sua viabilidade e competitividade em eventos internacionais de alto nível. Este trabalho visa contribuir para a comunidade de robótica, oferecendo um framework de código aberto e bem documentado, incentivando inovações no desenvolvimento de sistemas robóticos autônomos.

Palavras-chaves: Robótica. MicroMouse. Navegação Autônoma. Solução de Labirintos. STM32G474. Sistemas de Alto Desempenho.

Abstract

This thesis details the development of a high-performance autonomous robot for the MicroMouse category. The project integrates mechanical, electrical, and computational domains to create a robot capable of efficiently navigating complex mazes. Utilizing the STM32G474 microcontroller and firmware written in C++, the robot incorporates advanced algorithms for mapping, pathfinding, and navigation. The final implementation demonstrates compliance with the competition's functional and non-functional requirements, showcasing its viability and competitiveness in high-level international events. This work aims to contribute to the broader robotics community by providing an open-source, well-documented framework, encouraging further innovation in autonomous robotic systems.

Key-words: Robotics. MicroMouse. Autonomous Navigation. Maze Solving. STM32G474. High-Performance Systems.

Sumário

1 Introdução

1.1 Motivação

1.1.1 Contexto

A robótica competitiva tem se consolidado como uma área de grande importância tanto no Brasil quanto no cenário internacional. Eventos e competições de robótica atraem participantes de várias partes do mundo, promovendo a troca de conhecimento e inovação. No Brasil, ao longo dos últimos anos, a robótica competitiva tem se expandido, impulsionando o desenvolvimento de novas tecnologias e garantindo avanços significativos na formação de profissionais altamente capacitados. Internacionalmente, essas competições servem como um catalisador para a criação de soluções tecnológicas inovadoras, influenciando diversas indústrias e setores.

O Micro Mouse é uma categoria de robôs autônomos que se destaca por sua inovação e desafios únicos, exigindo alto nível de precisão e inteligência dos robôs competidores. Historicamente, o Micro Mouse possui uma forte tradição no Japão, reconhecido por sua intensa competitividade e pelo engajamento de universidades e institutos renomados na busca por soluções de navegação eficientes em labirintos complexos, resultando em robôs altamente otimizados e performáticos, que serviram de inspiração para este projeto.

1.1.2 Estado da Arte

Como a categoria de Micro Mouse ainda não é difundida no Brasil, os principais concorrentes a serem analisados são robôs japoneses, que tiveram a oportunidade de iterar seus projetos ao longo de diversos anos de competição e conseguiram atingir um alto nível de performance. A seguir, são analisados brevemente dois dos vencedores dos últimos anos na principal competição do mundo, a NTF All Japan.

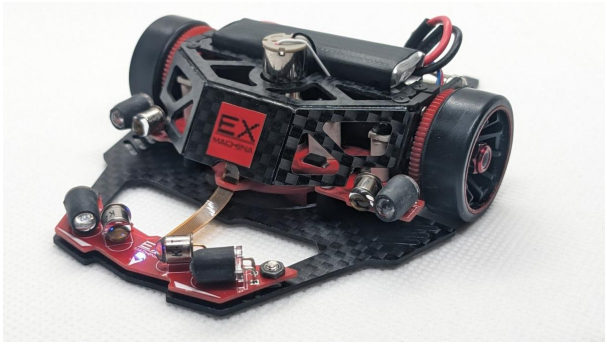


Figura 1 – Carmine

Vencedor da edição de 2023, o robô Carmine traz alguns conceitos inéditos ao pódio, como o fato de ser um Micro Mouse que foge do tradicional design de placa estrutural, que funciona também como base, e opta por desenvolver uma placa de fibra de carbono, sobre a qual duas placas menores são fixadas, uma com os sensores de parede, a outra com o resto do circuito do robô. Usa um STM32L4P5CGU6, com um ARM Cortex M4. Possui velocidade máxima de $6m/s$, aceleração máxima de $35m/s^2$ e velocidade máxima em curvas de $3m/s$. (FUNADA, 2023)

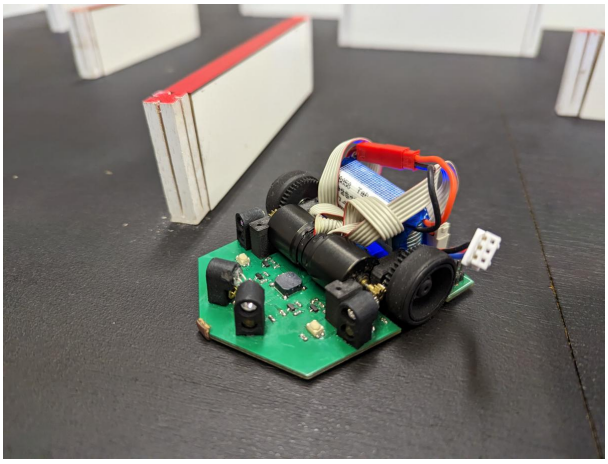


Figura 2 – Trident

Vencedor da edição de 2022, Trident apresenta um design mais simples e tradicional, com placa única e motores Faulhaber. O microcontrolador utilizado é um STM32F405RGT6 com Cortex M4 e ele se destaca pelas dimensões reduzidas e capacidade de executar diagonais de forma estável e consistente. (KEVIN, 2022)

1.2 Objetivo

Este trabalho tem como objetivo projetar e construir um robô da categoria Micro Mouse Full Size de acordo com os padrões de competição NTF - All Japan, e apto a atuar competitivamente na categoria. O projeto envolve o desenvolvimento de três frentes: projeto mecânico, elétrico e computacional, além da integração destes para o funcionamento do robô.

A categoria foi escolhida pela sua intrínseca demanda por um projeto de engenharia abrangente, englobando diversas áreas e necessitando de uma integração eficaz entre elas, ressaltando a importância da interdisciplinaridade no desenvolvimento bem-sucedido do projeto.

Dessa forma, o escopo abrangente deste trabalho almeja transcender a mera construção de um robô, buscando a criação de uma síntese engenhosa e eficiente entre a mecânica, eletrônica e computação. A intrincada rede de conexões entre esses domínios

será cuidadosamente tecida para assegurar não apenas a conformidade com os requisitos da competição, mas também para dotar o robô de uma performance excepcional e competitiva. A escolha da categoria Micro Mouse Full Size se reveste, assim, de um propósito estratégico e desafiador, culminando em um projeto de engenharia completo e integrado.

1.3 Justificativa

O desenvolvimento de um robô de alto desempenho para a categoria MicroMouse é fundamental por diversos motivos. Primeiramente, a categoria MicroMouse possui uma tradição rica em desafios técnicos, representando uma oportunidade valiosa para estudantes e entusiastas de engenharia desenvolverem habilidades práticas e integrarem conhecimentos de diferentes áreas, como eletrônica, mecânica e programação.

Embora a categoria seja reconhecida internacionalmente, ainda não é difundida no Brasil, o que limita o acesso dos brasileiros a essa experiência enriquecedora. Portanto, introduzir e promover a categoria MicroMouse no país não só amplia o horizonte da robótica nacional, mas também oferece novas oportunidades de aprendizado e colaboração.

Além disso, a maioria dos robôs vencedores na categoria MicroMouse não disponibiliza seus códigos-fonte, o que cria uma barreira no compartilhamento de conhecimento e na evolução da pesquisa. Mesmo quando disponíveis, os códigos muitas vezes carecem de implementações computacionais avançadas.

Nesse contexto, nosso trabalho visa preencher essa lacuna, desenvolvendo um robô com código-fonte acessível, bem documentado e implementando algoritmos modernos e complexos. Isso não apenas contribuirá para o avanço da comunidade de robótica, mas também inspirará novas gerações de engenheiros a explorar o campo da robótica e impulsionar a inovação em nossa sociedade.

1.4 Organização do Trabalho

Este trabalho está organizado em algumas seções principais. O capítulo 2 busca apresentar ao leitor os principais aspectos conceituais do trabalho, abordando de maneira superficial como o projeto foi dividido, de que forma e com que motivação, além disso, aborda também de que maneira trabalhou-se em cada aspecto ao longo do desenvolvimento. No capítulo seguinte, temos as especificações de requisitos do projeto, apresentando os requisitos funcionais, não funcionais e as especificações deliberadas para cada aspecto principal do projeto. Em seguida, temos o desenvolvimento, que detalha cada aspecto desenvolvido nas diferentes frentes do projeto, além dos testes realizados e por fim as considerações finais, que buscam apresentar os resultados obtidos no projeto.

2 Metodologia de trabalho e Conceitos empregados

Para realizar o desenvolvimento do projeto, alguns conceitos principais precisam ser estabelecidos. Neste projeto, as tarefas foram divididas em três frentes principais: a mecânica, que visa fabricar as peças e projetar a estrutura, a elétrica/eletrônica, responsável pelo projeto da placa e dimensionamento dos componentes e a computacional, encarregada pelo desenvolvimento de todo o código de controle e funcionamento do robô, tão como a arquitetura e definição dos aspectos da camada de abstração de hardware.

O desenvolvimento do projeto se deu por fases de maneira sequencial, de forma que este trabalho apresenta os eventos cronologicamente. Portanto, nas seções iniciais, os projetos apresentados não passam de protótipos e versões iniciais. Ao longo da elaboração do projeto, as decisões são consolidadas e o robô se aproxima cada vez mais de sua versão final.

Para o entendimento do projeto, é fundamental que o leitor esteja a par das regras da categoria, uma vez que são elas que pautam as decisões de projeto tomadas ao longo do desenrolar do trabalho; desta forma, esta seção busca detalhá-las.

A categoria tem 3 conjuntos de regras principais acerca de cada aspecto envolvido na competição: As regras para os robôs, as regras para o labirinto e as regras para o torneio. Abaixo, cada um dos três é abordado (NTF, 2024):

2.1 Regras

2.1.1 Regras para o robô

- O robô deve ser autônomo.
- Seu hardware ou software não deve ser modificado durante a competição, só são permitidas pequenos reparos se necesário
- O robô não deve deixar peças para trás no labirinto
- O robô não pode escalar, pular ou danificar as paredes.
- Sua projeção deve ser contida em um quadrado com lados de 25cm e altura infinita.

2.1.2 Regras para o labirinto

- As laterais das paredes são brancas, o topo das paredes é vermelho, e o piso do labirinto é preto, pintado com tinta preta fosca. O topo das paredes da zona de chegada e do ponto de partida pode ser vermelho, branco ou amarelo.
- O labirinto é composto por quadrados unitários de 18 x 18 cm, com no máximo 16 x 16 quadrados. As paredes têm 5 cm de altura e 1,2 cm de espessura.
- O ponto de partida estará localizado em um dos quatro cantos do labirinto, com o robô começando no sentido horário. A zona de chegada está no centro, ocupando quatro quadrados.
- Nas extremidades de cada quadrado unitário há pequenas zonas quadradas de 1,2 x 1,2 cm, chamadas de pontos de grade. O labirinto é construído de forma que haja pelo menos uma parede conectando cada ponto de grade, exceto no interior da zona de chegada. O labirinto é completamente cercado por paredes externas.

2.1.3 Regras para a competição

- O tempo oficial será registrado com base na volta mais rápida do ponto de partida até a zona de chegada. A competição avalia a velocidade e independência do robô, com critérios e prêmios definidos por cada torneio.
- Após a abertura do labirinto ao público, o operador não pode fornecer informações do labirinto ao robô. Durante a competição, o operador não pode alterar informações coletadas pelo robô na exploração inicial.
- O robô inicia a execução no ponto de partida e a finaliza ao retornar ao ponto inicial. A execução também termina se o robô parar por 2 segundos ou com aprovação do organizador.
- Caso o robô inicie automaticamente outra execução após retornar ao ponto de partida, ele deve parar por pelo menos 2 segundos antes de continuar.
- O operador não pode tocar no robô durante a execução, a menos que autorizado pelo organizador. O operador pode solicitar a interrupção caso detecte um mau funcionamento. Em outros casos, a memória do labirinto deve ser apagada para reiniciar.
- O tempo limite para cada competidor é de 7 minutos, podendo ser reduzido para 5 minutos dependendo do torneio. Cada robô pode realizar até 5 execuções.

- O robô conclui a execução quando toda a sua parte inferior (até 5 cm acima do chão) entra na zona de chegada. O tempo é medido do momento em que o sensor no ponto de partida detecta o robô até o momento em que o sensor na entrada da zona de chegada o detecta.
- Iluminação, temperatura e umidade serão as do ambiente. Não serão aceitos pedidos de ajuste na iluminação.
- O organizador pode questionar o operador sobre a adequação do robô, interromper execuções, declarar desclassificação ou solicitar orientações, conforme apropriado.
- Prêmios e critérios de avaliação são definidos individualmente para cada competição.

3 Especificação de Requisitos

3.1 Requisito Funcionais

Para possibilitar o início do desenvolvimento do projeto e maximizar suas chances de êxito, os requisitos funcionais foram levantados e analisados cuidadosamente. Abaixo encontram-se listados e descritos detalhadamente estes requisitos identificados como necessários para a concretização do projeto proposto:

1. **Identificar as paredes do labirinto:** Por se tratar de um robô que soluciona labirintos, o coração do projeto encontra-se na capacidade do robô de utilizar fototransistores em conjunto com LEDs infravermelhos para identificar e detectar paredes ao seu redor, desta forma enviando enviando ao microcontrolador sinais que permitem o mapeamento do labirinto e consequentemente sua resolução.

2. **Mapear o labirinto com as paredes identificadas:** Com as informações coletadas através dos sensores de parede, o projeto deve ser capaz de mapear internamente o labirinto, através de uma espécie de "gêmeo digital" simplificado, representando a projeção bidimensional do labirinto, a qual o robô será capaz de utilizar para se localizar e também calcular a rota mais rápida (não necessariamente a mais curta) até o ponto final de um labirinto qualquer.

3. **Calcular a rota mais rápida até o fim do labirinto:** Através do mapa gerado com as informações coletadas, o robô deve ser capaz de utilizar algoritmos de grafos e flood para determinar rotas possíveis até o final do labirinto e além disso, ser capaz de identificar dentre estas rotas a mais rápida, ou seja, aquela onde a maior aceleração pode ser desenvolvida de forma a finalizar o trajeto mais rapidamente.

4. **Percorrer uma rota calculada:** Dado uma rota calculada, o robô precisa conseguir seguir tal rota da forma mais precisa possível, ou seja, através de sensores de inércia, corrente e encoders angulares em seus eixos, o robô deve ser capaz de saber o quanto já foi percorrido e utilizar algoritmos de controle (como PID) para manter-se na rota sem bater em paredes ou perder velocidade.

5. **Estimar sua posição e orientação atual no labirinto:** Utilizando os mesmos sensores direcionados a percorrer a rota calculada, o projeto utilizará de técnicas de odometria para estimar sua posição no labirinto real através do gêmeo digital construído e armazenado no microcontrolador.

6. **Salvar as informações do labirinto na memória flash:** O gêmeo digital deve ser armazenado na memória flash e atualizado conforme a exploração do labirinto é realizada, uma vez que esta informação deve ser mantida entre voltas e mudanças de modo do robô, que deve ser capaz de operar em modo de "exploração" onde o labirinto é mapeado e "corrida" onde a rota mais rápida identificada é executada em função do conteúdo da memória.

7. **Identificar fim do labirinto:** Além de explorar, calcular uma rota e executá-la, o projeto, ao final do labirinto, deve identificá-lo, com uma pequena parada no ponto de chegada e uma emissão sonora através de um buzzer, assim sinalizando o encontro do objetivo e dando início ao retorno ao início.

8. **Calcular uma rota para o início do labirinto que mapeie outras partes do labirinto:** Ainda que o retorno para o ponto de início após a chegada ao final do labirinto não seja obrigatório, nota-se que os projetos no estado da arte em geral realizam o retorno até o início através de uma rota diferente daquela feita na volta de exploração. Isto é feito para que o labirinto seja mapeado de forma holística, portanto o projeto precisa ser capaz de, ao encontrar o final do labirinto, desenvolver uma rota de retorno que explore áreas não visitadas ainda e desconhecidas do labirinto, buscando encontrar a rota ótima para a solução.

9. **Parar no começo do labirinto:** Após o retorno, o projeto também deve ser capaz de se manobrar e finalizar a volta, com uma parada total no início, sinalizando o final da volta em questão e permitindo o prosseguimento das voltas subsequentes.

3.2 Requisitos não Funcionais

De forma análoga, características mínimas inerentes ao projeto necessárias para seu sucesso, representadas por seus requisitos não funcionais, foram levantadas. Tais características encontram-se listadas a seguir:

1. **Sua projeção no chão deve caber em um quadrado de 25x25cm:** O regulamento prevê como regra que nenhum robô apto a competir na categoria deve ter projeção maior que $25cm^2$, portanto se enquadra como requisito não funcional mínimo para o sucesso do projeto, uma vez que pretende-se criar um projeto apto a competir.

2. **Ter bateria suficiente para durar 5 minutos em execução:** O robô tem direito a 5 voltas no total, porém estas 5 devem somar, no máximo, 5 minutos de volta, desta forma a bateria do robô, que não pode ser trocada entre voltas, precisa ser capaz de sustentar o funcionamento do robô por, ao menos, 5 minutos.

3. **Ter uma capacidade de aceleração e freio de 30 m/s²:** Ao observar-se o estado da arte da tecnologia, nota-se que por tratar-se de uma competição de velocidade, porém em espaços pequenos uma vez que o labirinto trata-se de um quadrado de 2.88m de lado, a principal característica do robô é possuir uma capacidade alta de acelerar e desacelerar, desta forma desenvolvendo retas da forma mais veloz possível. Os campeões mundiais da categoria chegam a desenvolver acelerações teóricas de até 30m/s², o que se mostra uma aceleração alcançável e que mantém o projeto com um grau de competitividade satisfatório.

4. **Ter uma velocidade máxima de 6 m/s:** Como detalhado na seção relativa a aceleração mínima do projeto, por se tratar de um espaço limitado, não é prioridade que a velocidade máxima seja exorbitantemente alta, uma vez que seria praticamente impossível desenvolver e mantê-la em espaços pequenos, porém ao analisar a categoria, 6m/s é a velocidade máxima média dos principais competidores atualmente, permitindo desenvolver a maior reta possível no labirinto (2.8m) em menos de um segundo. Desta forma, é a velocidade máxima que busca-se desenvolver neste projeto. Esta velocidade será alcançada através da combinação de motores de baixo torque e alta rotação utilizados geralmente em aeromodelismo com uma redução mecânica que permite a produção de um torque na roda suficiente para a locomoção do robô.

5. **Ser capaz de fazer curvas de 90° com mais de 1 m/s:** Para manobrar no labirinto de forma eficiente, é necessário que o robô tenha capacidade de realizar curvas de forma rápida. Todas as curvas são de 90º ou 45º, desta forma, o projeto não deve desacelerar-se na curva então uma velocidade de curva de cerca de 1m/s é satisfatória.

6. **Ter um tamanho máximo de 11.9 cm (Conseguir percorrer caminhos diagonais no labirinto):** O labirinto é formado por um grid de 16x16 de quadrados de 18x18cm. As paredes têm 1.2cm de espessura, de forma que sobram 16.8cm de espaço para o robô passar, sendo esse o primeiro requisito de tamanho do robô. Para ser capaz de passar na diagonal, o robô deve seguir um requisito mais forte de tamanho:

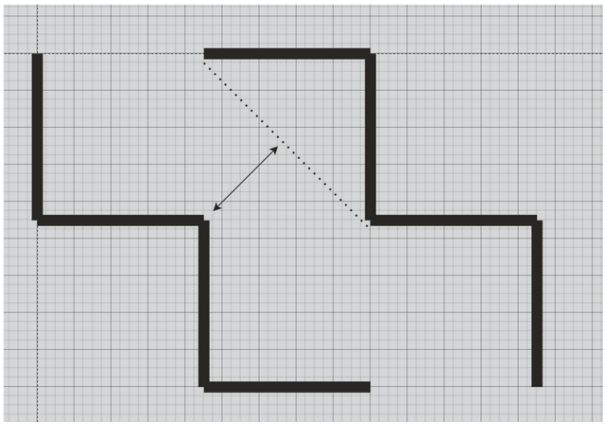


Figura 3 – Diagonal do Labirinto

O que resulta na seguinte situação:

$$d_{max} = \frac{16.8 \times \sqrt{2}}{2} = 11.879\ cm$$

Fornecendo o requisito de largura máxima do projeto.

3.3 Especificações Mecânicas

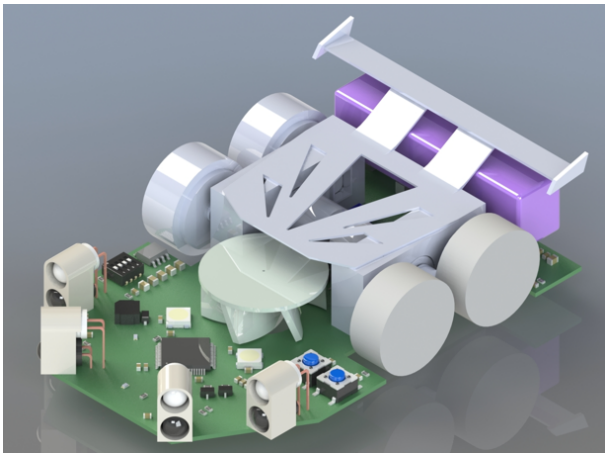


Figura 4 – Projeto mecânico do protótipo do robô

Acerca das especificações do projeto mecânico, para que o robô seja capaz de alcançar os requisitos funcionais e não funcionais, algumas limitações precisam ser respeitadas, sendo a mais básica delas a dimensão. Portanto, pode-se notar que no projeto atual tem-se um robô com 67mm de lado por 100 de comprimento. Desta forma o projeto é capaz de realizar as diagonais do labirinto com razoável folga, enquanto mantém uma área de placa boa para o posicionamento dos componentes.

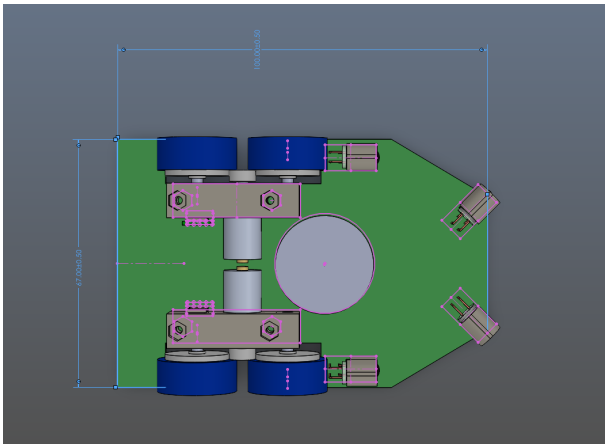


Figura 5 – Vista superior do protótipo mecânico

Além do tamanho reduzido, requer-se confiabilidade na fixação dos eixos das rodas, porém mantendo a capacidade de medição da rotação através de encoders magnéticos. Para alcançar este propósito, optou-se por um esquema de montagem com um eixo fixo e outro móvel, assim garantindo a simplicidade do sistema e a consistência das medições. Abaixo, é possível observar um detalhamento do mancal do sistema:

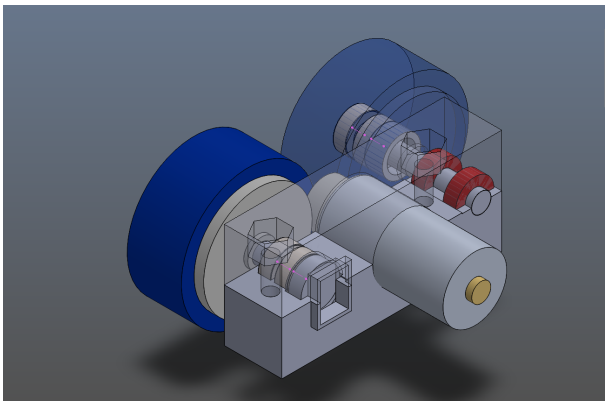
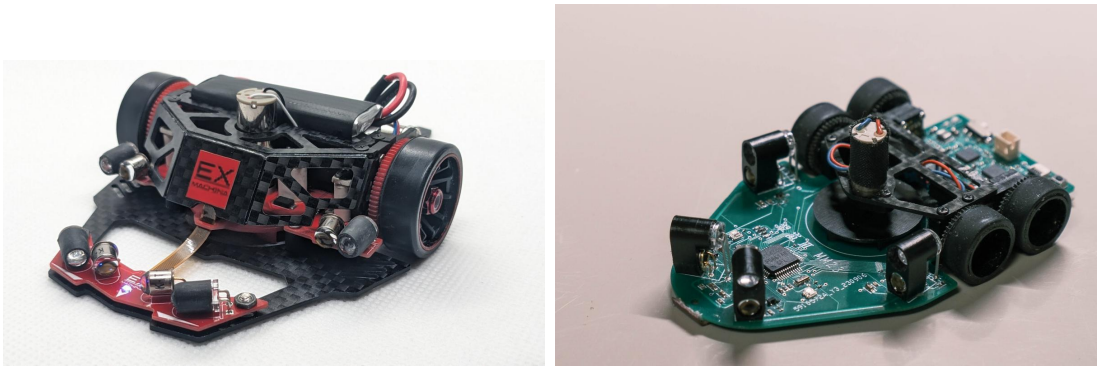


Figura 6 – Projeto mecânico do protótipo da locomoção do robô

Enfim, um ponto de discussão que antecedeu o desenvolvimento do projeto mecânico foi o número de rodas do robô a se desenvolver. Quando se observa os principais ganhadores da categoria Classic Mouse, pode-se notar que existe uma divisão entre robôs de duas e quatro rodas no pódio. (NTF, 2023)



(a) Carmine, vencedor 2023 (b) D-The-Star, terceiro lugar 2023

Figura 7 – Robôs competitivos com esquemas diferentes

Desta forma, não há um consenso claro de qual vertente apresenta superioridade de forma objetiva, porém na categoria (TONDRA, 2004; PETER, 2024) de forma geral, robôs com 2 rodas apresentam maior simplicidade de construção e manobrabilidade, porém têm desempenho pior em percursos com longas retas uma vez que conseguem desenvolver uma aceleração menor. Em contrapartida, Micro Mouses com 4 rodas têm uma estabilidade muito maior, e aceleração mais alta, uma vez que não têm pontos de apoio estáticos no chão que são "arrastados", porém têm maior dificuldade de desenvolver curvas e são mais complexos de modelar, uma vez que modelos de veículos sem diferencial, com curva por derrapagem (*skid-steering*) ainda são pouco estudados. (FERREIRA; FERREIRA; TAVARES, 2010)

Portanto, com estes fatores levantados, optou-se por desenvolver um robô com 4 rodas uma vez que a estabilidade extra e o potencial de aceleração maior neste modelo justificam as dificuldades.

3.4 Especificações Computacionais

Para atender aos requisitos funcionais e não funcionais do projeto, serão utilizados recursos computacionais estrategicamente escolhidos, visando a eficiência, desempenho e integração necessários para o sucesso do robô na categoria Micro Mouse Full Size.

O microcontrolador selecionado para o projeto é o STM32G474 da STMicroelectronics, baseado na arquitetura ARM Cortex-M4. Esta escolha foi feita levando em consideração a robustez, capacidade de processamento e recursos periféricos oferecidos pelo dispositivo, adequando-se às demandas computacionais do projeto.

O firmware do robô será desenvolvido utilizando a linguagem C++, utilizando-se como base a biblioteca de Hardware Abstraction Layer (HAL) desenvolvida pela ST, o que proporciona uma camada de abstração que facilita o acesso aos periféricos do

microcontrolador. Além disso, será utilizada a ferramenta STM32CubeMX para gerar a inicialização e configuração desses periféricos de forma eficiente e simplificada.

Para testar e validar a lógica computacional e os algoritmos implementados, faremos uso do Robot Operating System 2 (ROS2) em conjunto com o simulador Gazebo. Essa integração não apenas nos permitirá simular os sensores e atuadores do robô, mas também a interação deles com o ambiente físico. Dessa forma, poderemos validar o código embarcado, observar o comportamento do robô em um ambiente tridimensional e verificar sua eficácia e precisão na resolução de labirintos.

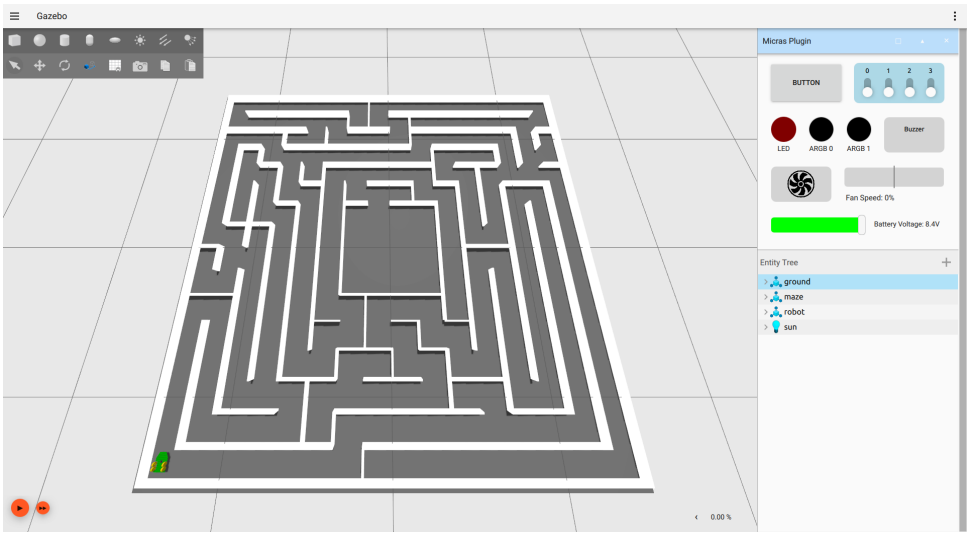


Figura 8 – Simulação do Robô utilizando Gazebo

A combinação dessas tecnologias computacionais proporcionará ao projeto a capacidade de identificar as paredes do labirinto, mapear sua estrutura, calcular rotas ótimas, seguir trajetórias definidas, estimar posição e orientação, armazenar informações do labirinto na memória flash, identificar o fim do labirinto e calcular rotas de retorno, atendendo assim a todos os requisitos funcionais estabelecidos.

Além disso, as especificações computacionais incluem a capacidade de comunicação eficiente entre os diferentes componentes do sistema, garantindo uma integração coesa e a troca de dados necessária para o funcionamento adequado do robô durante as competições.

3.4.1 Arquitetura de Firmware

O firmware do robô é estruturado em uma arquitetura de alto e baixo nível, com o uso do padrão de projeto Hardware Proxy para facilitar o acesso e controle de hardware. No nível mais baixo, temos o micras_hal, que é um wrapper em C++ para a biblioteca de Hardware Abstraction Layer (HAL) fornecida pela STMicroelectronics, que é escrito em C. Este wrapper fornece uma interface mais amigável e orientada a objetos para a biblioteca HAL, permitindo um controle mais fácil e direto do hardware subjacente.

Acima do micras_hal, temos o micras_proxy, que implementa o padrão de projeto Hardware Proxy. Este padrão de projeto envolve a criação de um elemento de software responsável pelo acesso a um pedaço de hardware e pela encapsulação da implementação de compressão e codificação de hardware. O proxy disponibiliza métodos públicos que permitem ler e escrever valores no dispositivo encapsulado, além de inicializar, configurar e desligar o dispositivo conforme apropriado. O proxy fornece uma interface independente de codificação e conexão para os clientes, promovendo fácil modificação caso a natureza da interface do dispositivo ou conexão mude.

No nível mais alto, o código utiliza as proxies para acessar as informações dos sensores e controlar os atuadores. Isso permite que o código de alto nível se concentre na lógica de controle e processamento de dados, sem se preocupar com os detalhes de baixo nível do acesso ao hardware.

Os códigos do Firmware estão disponíveis no respectivo repositório do GitHub. (BARBOSA; SANTI; PEREIRA, 2024a)

3.4.2 Simulação

Para facilitar os testes e desenvolvimento do projeto, foi desenvolvido um ambiente de simulação utilizando o framework ROS2 junto com o simulador de física Gazebo. Na simulação, a camada de proxy é simulada, conversando com a API da simulação ao invés do HAL, de forma que é possível rodar o próprio alto nível do Firmware em um ambiente simulado. Os códigos para o ambiente de simulação estão disponíveis no respectivo repositório do Github. (BARBOSA; SANTI; PEREIRA, 2024b)

3.5 Especificações Elétricas

O projeto elétrico do robô da categoria Micro Mouse Full Size desempenha um papel crucial na garantia do funcionamento eficiente e confiável do sistema como um todo. A PCB (Placa de Circuito Impresso) será desenvolvida considerando a interdependência entre os aspectos mecânicos e eletrônicos, garantindo uma integração eficaz entre ambas as áreas.

A concepção da PCB é orientada pela necessidade de acomodar todos os sensores, atuadores e demais componentes essenciais para as operações do robô, respeitando as restrições de espaço e tamanho impostas pelo regulamento da competição. Para isso, são utilizadas técnicas avançadas de layout e roteamento para minimizar interferências e garantir uma conectividade confiável entre os componentes.

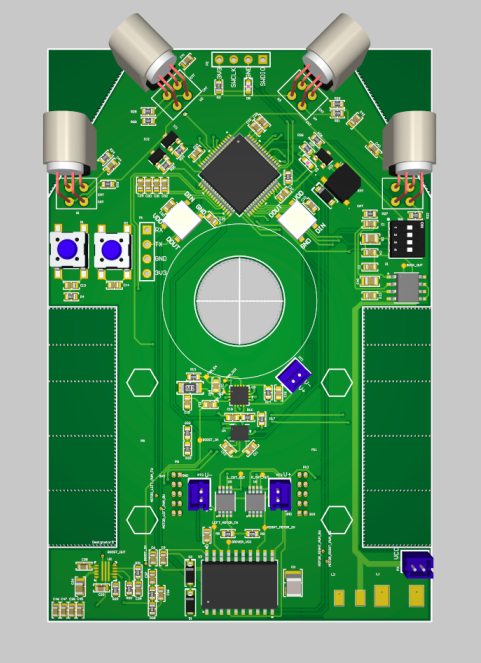


Figura 9 – Modelo 3D do protótipo da PCB do robô

O fornecimento de energia para o robô será assegurado por uma bateria de íon de lítio de alta capacidade, dimensionada para proporcionar autonomia suficiente para a realização de cinco voltas no labirinto, conforme exigido pelas regras da competição. A escolha da bateria levará em conta sua capacidade de fornecer corrente adequada e sua compatibilidade com os requisitos de espaço e peso do robô.

Para garantir o funcionamento correto dos componentes eletrônicos, serão utilizados reguladores de tensão para fornecer os níveis de tensão necessários aos diferentes dispositivos do sistema. Os sensores, como os sensores de parede, IMU (Unidade de Medição Inercial) e sensores de corrente, serão alimentados de forma precisa e estável para garantir a precisão das medições. Da mesma forma, os atuadores, como o motor da locomoção, receberão alimentação adequada para garantir um desempenho otimizado e confiável durante as operações do robô.

3.6 Fluxo de Informações

Abaixo, pode-se observar o fluxo de informações do projeto, detalhando como cada processo realizado pelo projeto é tratado pelo hardware ou software, e qual o endpoint físico deste processo. É possível notar que a partir das informações obtidas pelo sensor inercial, e pelos encoders, é calculada a posição atual do robô, que utilizada junto as medições dos sensores para mapear o labirinto, e construir uma trajetória ótima para resolver o labirinto, que é então seguida utilizando um algoritmo de controle que atua em cada um dos motores.

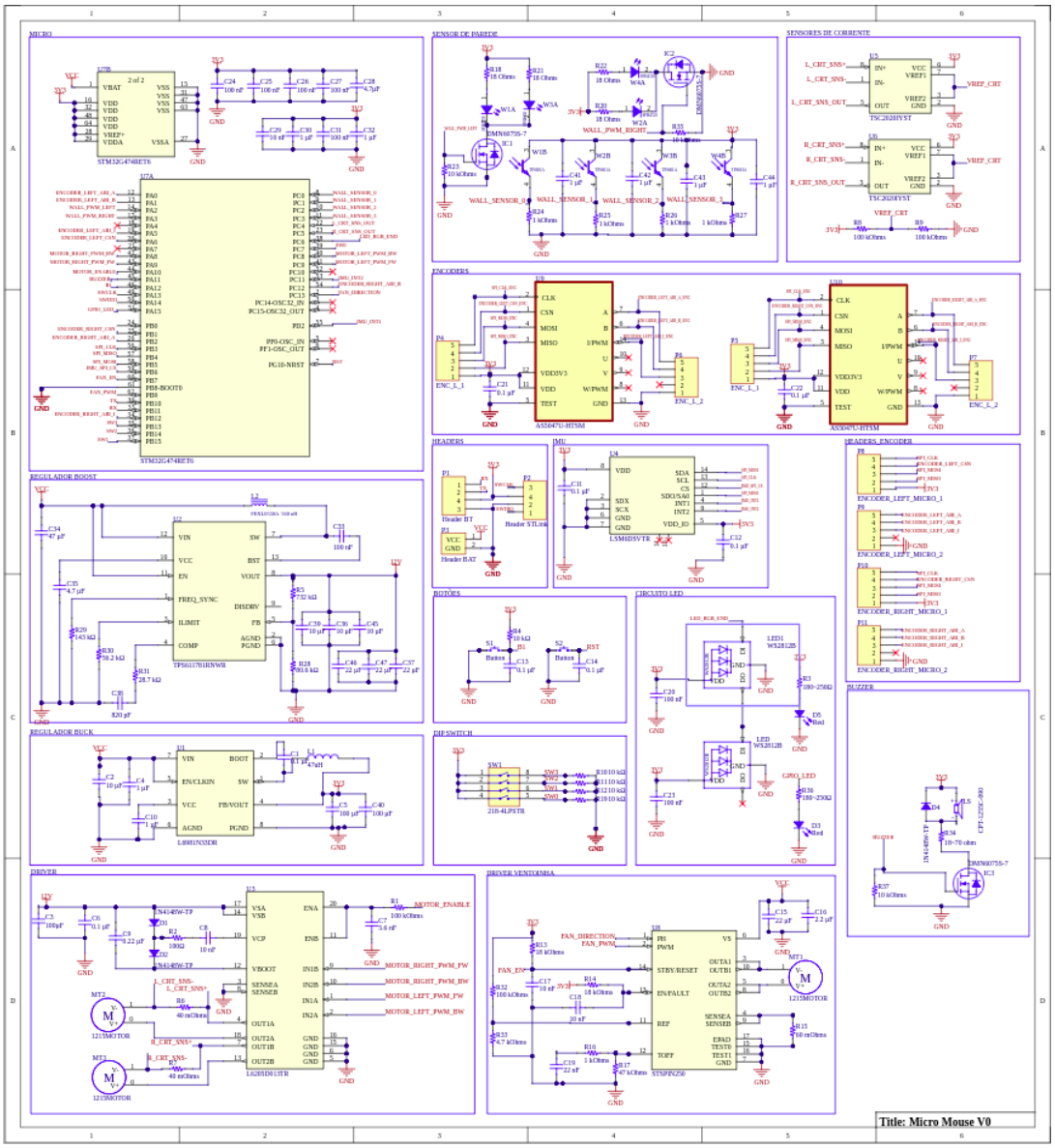


Figura 10 – Esquemático dos componentes utilizados na placa

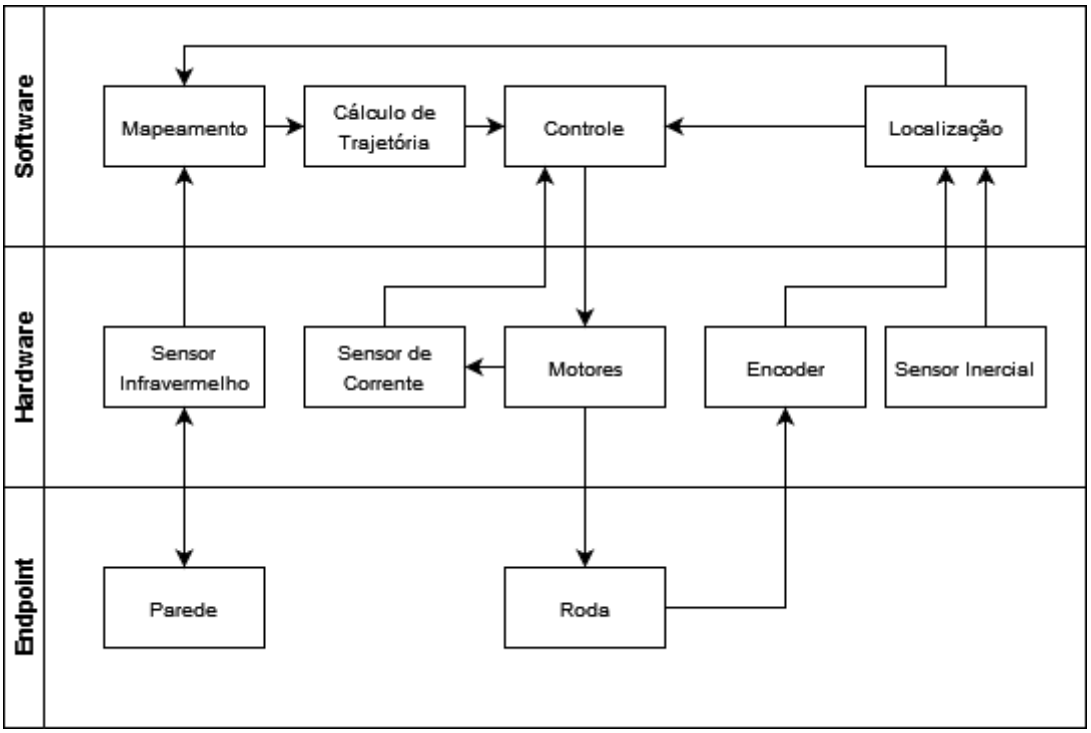


Figura 11 – Fluxo de informações do robô

4 Desenvolvimento

4.1 Tecnologias Utilizadas

4.1.1 Estrutura

Durante o desenvolvimento do projeto mecânico do robô, o Solidworks foi a principal ferramenta de modelagem tridimensional utilizada. Essa escolha se deu pela sua robustez no desenvolvimento de projetos mecânicos complexos, possibilitando a visualização detalhada de cada componente e simulação de montagens. O Solidworks permitiu projetar e ajustar as peças para garantir que todos os componentes do sistema elétrico fossem devidamente acomodados no espaço disponível, ao mesmo tempo em que se respeitavam as limitações dimensionais do robô. Além disso, essa ferramenta foi essencial na condução de análises estruturais e de movimento, assegurando que a transmissão de torque fosse adequada para a locomoção planejada.

As principais tecnologias utilizadas durante o desenvolvimento incluem:

- **Solidworks**: Para modelagem CAD (Computer-Aided Design), simulação e verificação da montagem.
- **Impressora 3D FDM**: Utilizada para manufaturar todas as peças do robô e versioná-las, assegurando um resultado satisfatório sem comprometer as capacidades computacionais do projeto.
- **Cura Slicer**: Software utilizado para preparar os modelos 3D para impressão na impressora FDM.

4.1.2 Placa

O desenvolvimento do sistema elétrico do robô foi realizado com o objetivo de garantir um funcionamento eficiente e confiável, integrando de maneira eficaz os aspectos mecânicos e eletrônicos. Para o projeto da Placa de Circuito Impresso (PCB), utilizou-se o Altium Designer, uma ferramenta avançada que permite a criação de layouts complexos e a otimização do roteamento dos componentes eletrônicos.

A fabricação da PCB foi realizada por meio de serviços de manufatura da JLCPcb, que oferecem qualidade e precisão na produção de placas eletrônicas. Após a fabricação, os componentes foram soldados manualmente pelos integrantes do grupo, garantindo atenção a cada detalhe e a correção de eventuais problemas durante o processo.

As principais tecnologias utilizadas durante o desenvolvimento incluem:

- **Altium Designer**: Para o projeto da PCB e elaboração do esquema elétrico, assegurando uma integração eficaz entre componentes.
- **JLCPcb**: Serviço de manufatura utilizado para a produção da PCB, garantindo qualidade e conformidade com o projeto.
- **Testes Unitários em C++**: Desenvolvidos para observar o comportamento individual de cada subsistema elétrico, permitindo a identificação de falhas e a verificação da funcionalidade dos componentes.

4.1.3 Computação

O desenvolvimento computacional do projeto foi guiado pela necessidade de controlar com precisão o robô Micro Mouse Full Size, garantindo a navegação eficiente pelo labirinto e a interação correta com o ambiente. Diversas tecnologias foram escolhidas para facilitar o desenvolvimento e garantir que o robô pudesse operar em um cenário de competição.

Uma das ferramentas principais utilizadas foi o STM32CubeMX, um software da STMicroelectronics que permite a configuração dos periféricos do microcontrolador STM32G474. Este software possui uma interface gráfica que simplifica o processo de configuração de componentes críticos, como os pinos de entrada e saída (GPIOs), os temporizadores (timers) e os conversores analógicos-digitais (ADCs), que são usados para ler dados dos sensores de proximidade. Por exemplo, no caso dos sensores infravermelhos responsáveis por detectar as paredes do labirinto, o STM32CubeMX foi utilizado para configurar os ADCs que capturam os sinais analógicos desses sensores, permitindo a conversão desses sinais em informações digitais que podem ser processadas pelo microcontrolador.

O desenvolvimento do firmware foi feito no VSCode, uma plataforma integrada de desenvolvimento que oferece suporte total à linguagem C++ e permite a depuração direta no hardware. A integração do VSCode com o STM32CubeMX através de um script CMake de compilação customizado possibilitou a geração de código inicializado automaticamente, que serviu como base para a implementação de funcionalidades mais complexas. Por exemplo, após a configuração inicial do ADC para os sensores, foi possível desenvolver algoritmos de filtragem de sinais, como filtros móveis de média, para garantir que os dados dos sensores fossem precisos e não influenciados por ruídos temporários.

Além disso, utilizou-se o STM32CubeMonitor, uma ferramenta que permitiu a visualização em tempo real de variáveis de execução do firmware. Durante o processo de desenvolvimento, o monitoramento de variáveis, como a velocidade dos motores e a

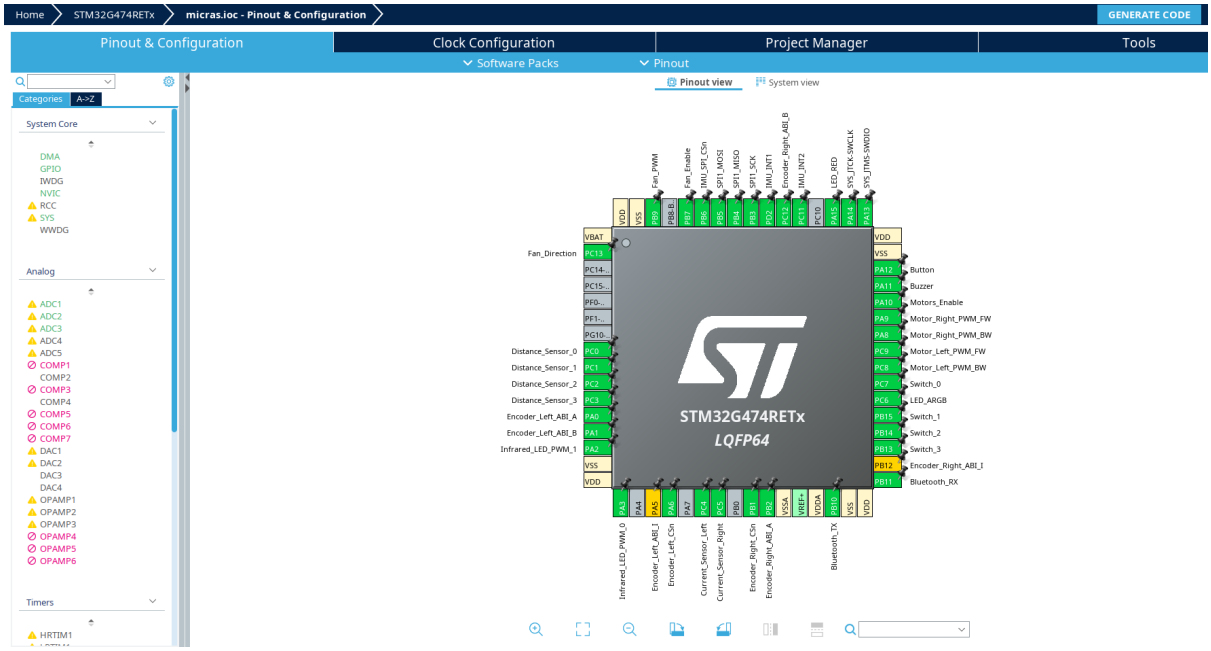


Figura 12 – Tela principal do STM32CubeMX

posição estimada do robô, foi crucial para validar o comportamento do robô em tempo real. Um exemplo prático de uso foi a verificação da precisão do sistema de odometria, onde a velocidade dos motores, calculada a partir de contadores de pulsos dos encoders, foi comparada com a posição estimada pelo robô ao longo do tempo. Essa ferramenta foi utilizada para ajustar a odometria, refinando a conversão de pulsos do encoder em distância percorrida.

Outro software importante foi o STM32CubeProgrammer, utilizado para gravar o firmware no microcontrolador e monitorar a memória flash. Este software simplificou o processo de gravação repetitiva do código à medida que ajustes e novas funcionalidades eram implementados, permitindo também a leitura e gravação direta de setores de memória flash. Isso foi útil no desenvolvimento de funcionalidades que exigiam o armazenamento de dados persistentes, como o mapeamento do labirinto, onde a memória flash foi utilizada para gravar a configuração do labirinto que o robô explorava.

A simulação foi parte fundamental do processo de desenvolvimento. O Robot Operating System 2 (ROS2), em conjunto com o simulador Gazebo, foi utilizado para validar o comportamento do robô em um ambiente virtual antes de transferir o código para o hardware real. O ROS2 permitiu a criação de nós que simulavam os sensores e atuadores do robô, facilitando o teste de algoritmos de controle de navegação e exploração do labirinto. No Gazebo, foram configurados diferentes cenários de labirintos, nos quais o robô pôde ser testado de forma a simular as condições da competição. A integração com o ROS2 possibilitou a simulação de sensores virtuais, como os infravermelhos, e o comportamento dos motores, sem a necessidade de hardware físico, permitindo ajustes

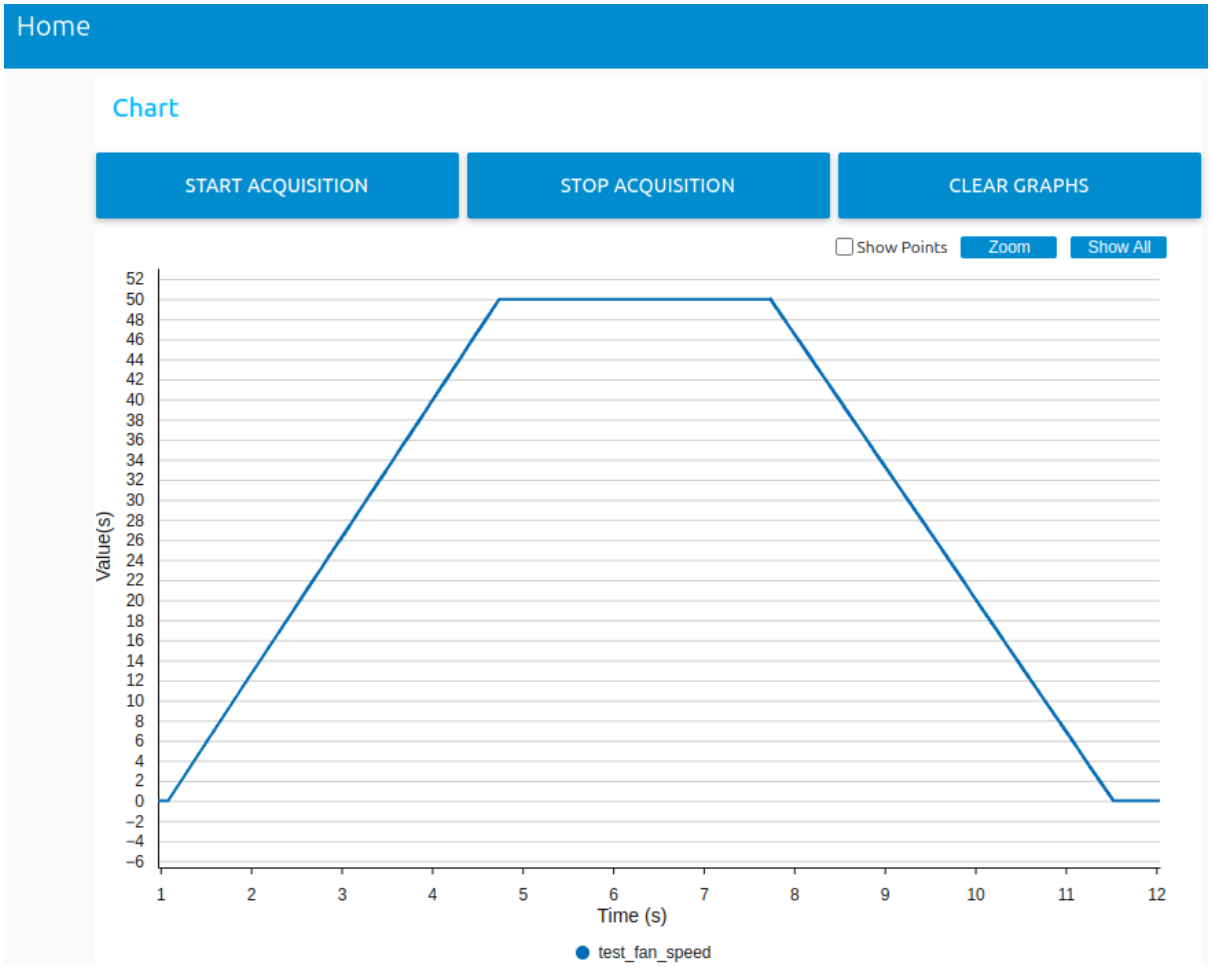


Figura 13 – Gráfico gerado em tempo real pelo STM32CubeMonitor

iterativos rápidos.

O uso dessas tecnologias não apenas acelerou o processo de desenvolvimento, mas também garantiu que os algoritmos e funcionalidades fossem validados e ajustados em um ambiente controlado, minimizando o risco de erros durante os testes no robô físico.

4.2 Estrutura do robô

O projeto da estrutura do Micro Mouse tem como principal objetivo desenvolver uma solução com complexidade mínima, alta confiabilidade e fabricação simplificada. Esses fatores são essenciais para que o robô atue de forma consistente no ambiente, atendendo aos requisitos funcionais do projeto. Simultaneamente, o projeto deve cumprir os requisitos não funcionais, como evitar o desperdício de peso (minimizando o peso desnecessário) e garantir que a distância entre as rodas permaneça inferior a 11,9 cm. Esse limite é crucial para viabilizar o desenvolvimento de trajetórias diagonais no labirinto.

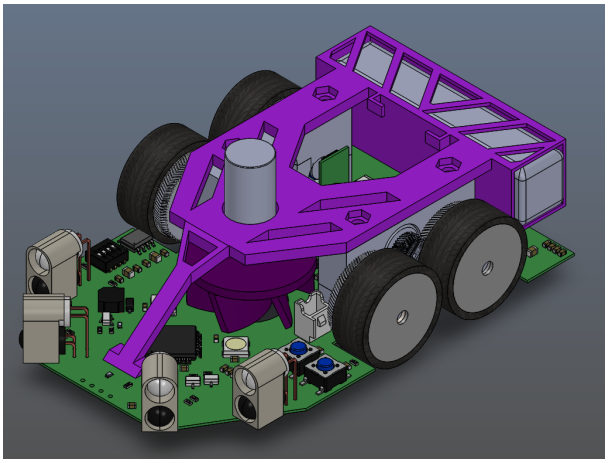


Figura 14 – Projeto Estrutural Completo

Com esses objetivos em mente, detalha-se a seguir os aspectos principais do projeto estrutural, suas características e as motivações que embasaram as decisões tomadas durante seu desenvolvimento.

4.2.1 Projeto

A estrutura do robô é dividida em duas porções principais: o mancal (componente responsável pela fixação dos eixos e motores responsáveis pela locomoção) e a tampa (componente responsável por fixar a bateria e o conjunto da ventoinha na estrutura do robô). Abaixo, detalham-se os aspectos centrais de cada um desses componentes:

4.2.1.1 Mancal

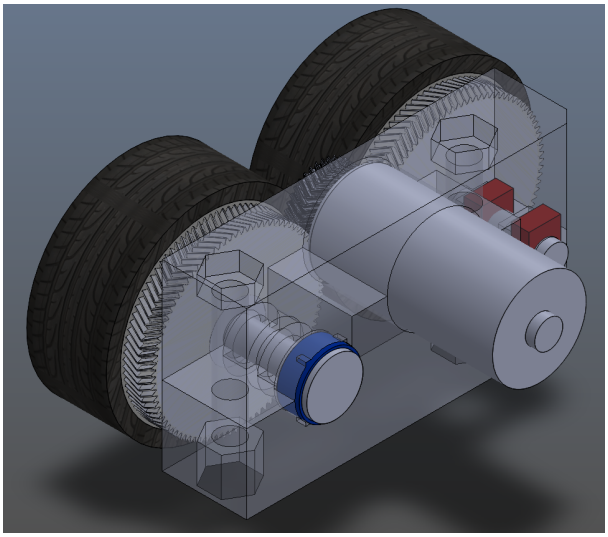


Figura 15 – Projeto do conjunto do mancal

O mancal é a peça mais complexa da estrutura do robô, responsável por fixar à placa todos os componentes necessários para a locomoção do robô, enquanto minimiza

a folga mecânica (*backlash*) entre as engrenagens das rodas e o pinhão do motor. Isto é alcançado desenvolvendo um mancal "Sanduíche", ou seja, duas peças abraçam os eixos das rodas e o motor da locomoção, garantindo que o sistema é travado linear e axialmente.

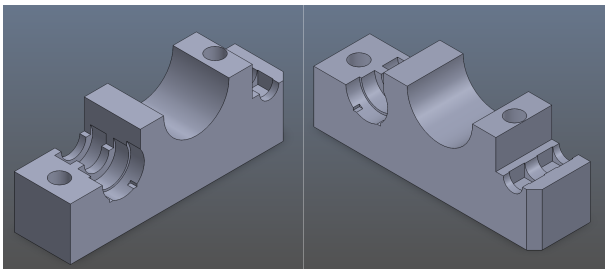


Figura 16 – Vista em detalhe do mancal inferior em posições diferentes

Ao analisar o mancal em detalhe, nota-se que ele é assimétrico. Esta escolha é deliberada e justifica-se pela necessidade de diminuir a folga mecânica do projeto, mas assegurando a maior resolução possível do encoder magnético. Para isso, desenvolveu-se um sistema de eixo misto, onde a roda dianteira tem um eixo fixo, com o rolamento dentro da própria roda, que gira ao redor do eixo, enquanto a roda traseira tem um eixo móvel, com a roda fixa no eixo, que gira livremente, com rolamentos fixos no eixo e apoiados no mancal, além de um ímã radial fixado na ponta do eixo, o que possibilita a medição da posição angular das rodas pelo encoder.

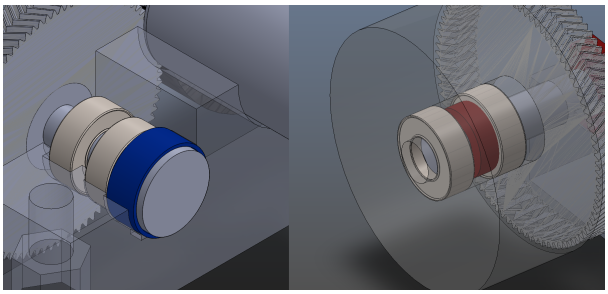


Figura 17 – Vista em detalhe do projeto do eixo móvel e do projeto do eixo fixo

Importante notar que ambos os eixos são biapoiados, ou seja, possuem dois pontos de apoio distintos no mancal, o que garante maior estabilidade do sistema e ajuda a minimizar a folga mecânica.

Enfim, o aspecto mais importante do mancal é garantir a transmissão adequada de movimento do eixo do motor para os eixos das rodas, enquanto sincroniza o movimento das rodas com o mínimo de *backlash* possível, assim minimizando o erro do encoder magnético. A transmissão do movimento se dá através de engrenagens manufaturadas pelo grupo. Optou-se por fabricar as engrenagens ao invés de adquiri-las, pois o tamanho reduzido dificultou encontrar um conjunto de engrenagens motoras e movidas que satisfizesse as

limitações de tamanho do robô; desta forma, utilizou-se o software OpenScad para projetar as engrenagens de forma paramétrica.

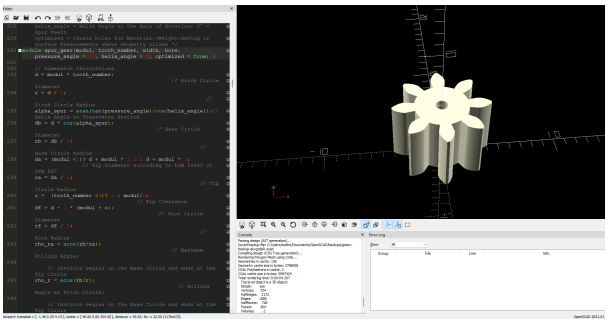


Figura 18 – OpenScad - Programa para desenvolvimento de peças com código

O OpenScad é um software Open Source de CAD paramétrico que permite utilizar lógica de programação para projetar peças que precisam de padrões repetitivos e facilitar a iteração de designs paramétricos, portanto ideal para o projeto de engrenagens e peças similares.

Através dele, é possível compartilhar scripts inteiros para construção de peças paramétricas, como o que foi utilizado para projetar as engrenagens do projeto. (SPEN, 2020)

Para a transmissão projetada, optou-se por um par de engrenagens de módulo 0.8, com 8 dentes para o pinhão e 20 dentes para as rodas, atingindo uma taxa de redução de 2.5.

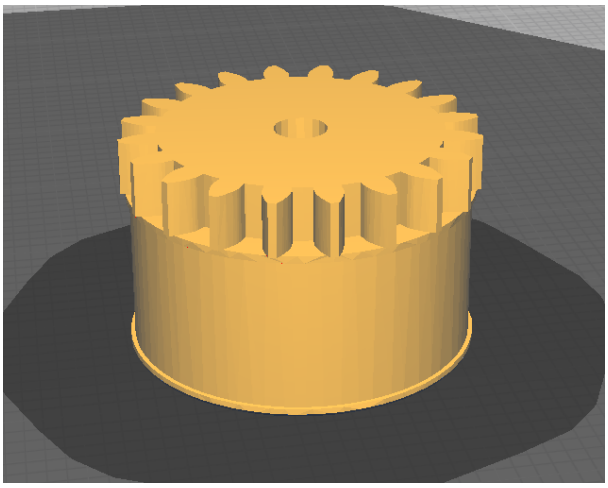


Figura 19 – Par roda/engrenagem pronto para ser fabricado

As engrenagens são impressas junto com as rodas, de forma que o movimento possa ser transmitido sem qualquer risco de escorregamento entre peças; portanto, as rodas do robô e as engrenagens são uma peça única.

4.2.1.2 Tampa

O conjunto da tampa é responsável por segurar o motor e servir de suporte para a bateria. Ele também é a peça responsável por exercer pressão na peça superior do mancal, por isso atua como uma "tampa"para o sistema como um todo.

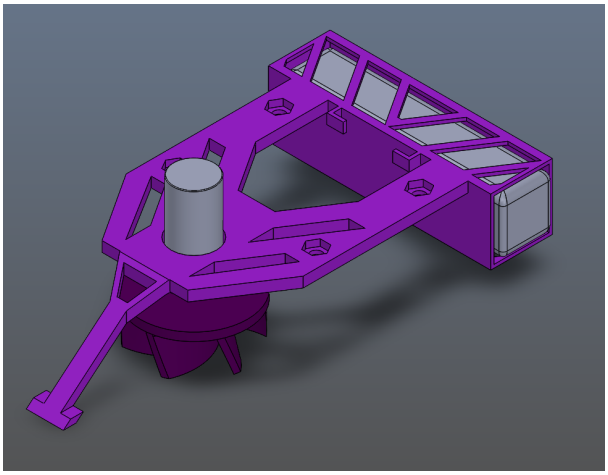


Figura 20 – Conjunto da tampa

Na parte de trás, a tampa possui um compartimento para encaixar a bateria, que fica presa com o atrito graças às baixas tolerâncias da peça, sendo assim o robô consegue andar livremente pelo labirinto apenas com a alimentação interna, sem depender de mais nada.

4.2.2 Implementação

Para realizar a fabricação dos componentes da estrutura do robô, foi utilizada uma impressora 3D FDM da marca Creality, modelo Ender 3 V3 SE. As peças foram impressas em PLA e a impressora foi modificada para que seu bico tivesse uma saída de 0.2mm; isso garantiu que peças com detalhes muito pequenos, como as engrenagens, fossem fabricadas com tecnologia FDM, uma vez que é mais barata e de mais fácil acesso.

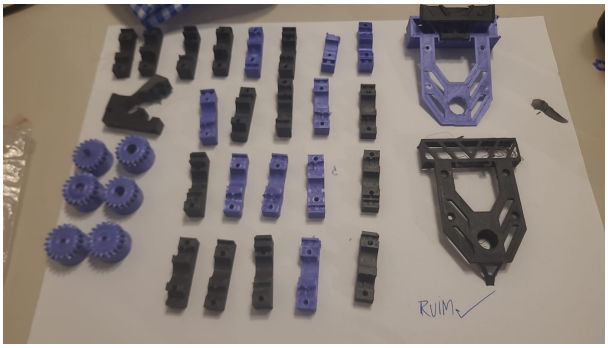


Figura 21 – Iterações dos componentes estruturais ao longo do desenvolvimento

Além disso, a opção por fabricar todos os componentes mecânicos internamente se provou eficiente, uma vez que diminuiu a dependência de disponibilidade de equipamentos de terceiros e possibilitou maior versatilidade no versionamento das peças.

4.3 Implementação Computacional

A implementação do sistema computacional do robô foi conduzida de forma incremental, começando pela abstração de baixo nível para o controle de hardware e evoluindo até a implementação de algoritmos complexos de navegação e resolução de labirintos.

4.3.1 Camada de Abstração do Hardware

Na camada de baixo nível, foi implementado um wrapper em C++ sobre a biblioteca HAL (Hardware Abstraction Layer), fornecida pela STMicroelectronics, que é desenvolvida em C, de forma a tornar possível a utilização de funcionalidades mais modernas da linguagem, como classes e namespaces, e tornando o código independente do hardware que está sendo utilizado.

Além da possibilidade de utilizar funções já implementadas para usufruir de determinados recursos do hardware, a utilização da biblioteca de HAL da ST permite utilizar o software STM32CubeMX para realizar a configuração do microcontrolador e de seus periféricos, salvando essas customizações através de defines. Dessa forma, o wrapper desenvolvido também encapsula essas configurações em structs, que são declaradas em cada uma das classes de HAL e definidas em um arquivo `target.h`.

A seguir, serão descritas cada uma das classes utilizadas, que representam respectivamente algum componente do Hardware, todas elas recebem em seu construtor uma struct `Config`,que se traduz diretamente em configurações utilizadas no HAL da ST, ou em alguns casos, outras particularidades dos componentes.

4.3.1.1 GPIO

A primeira classe do HAL a ser implementada foi o GPIO (General-Purpose Input/Output), que possui uma estrutura simples correspondente à representação apenas de um sinal binário, que pode ser de entrada ou de saída. Dessa forma, sua implementação se resume a uma função de `read`, que retorna o estado do sinal aplicado em um pino de entrada, e funções de `write` e `toggle`, que escrevem ou alternam o estado de um pino de saída, respectivamente.

Figura 22 – Diagrama de colaboração da classe Gpio

4.3.2 Timer

Na classe referente ao timer, foi utilizada uma abordagem diferente das demais, visto que o periférico correspondente tem diversas funcionalidades diferentes, que resultariam em um elemento muito extenso e complexo no HAL. Dessa forma, optou-se por separar cada uma das funcionalidades em diferentes classes, sendo que a responsabilidade de cronometrar intervalos de tempo ficou com essa classe principal.

Existem duas possibilidades de utilização da classe: A primeira visa cronometrar intervalos em milissegundos, utilizando como base o *SysTick*, um sinal interno do microcontrolador que possui um período de 1 ms. A segunda possibilita contagens de tempo em microssegundos, e para isso, é necessário passar ao construtor um handle de um periférico de timer que esteja configurado com um período de 1 ms.

Para possibilitar a interface com o cronômetro, foram implementadas funções de `reset`, que zeram a contagem, e funções `get_counter` que retornam o valor atual desta contagem. Além disso, também foram implementadas funções de `sleep`, que interrompem a execução do código de maneira preemptiva durante um intervalo definido de tempo.

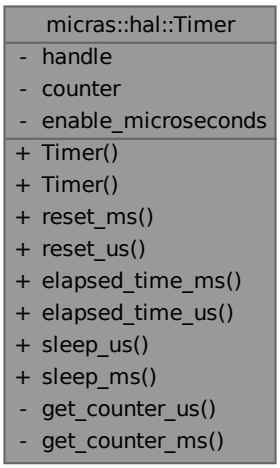


Figura 23 – Diagrama de colaboração da classe Timer

4.3.2.1 Encoder

A classe do Encoder representa uma das funcionalidades derivadas do periférico do timer, de forma que seu funcionamento interno também está relacionado com contagem, mas não da passagem do tempo e sim de alterações em um pino de entrada digital, de forma que, quando interligado a um encoder incremental, o valor relativo ao seu ponto inicial pode ser obtido através da função `get_counter`.

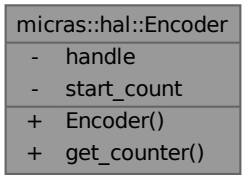


Figura 24 – Diagrama de colaboração da classe Encoder

4.3.2.2 PWM

O periférico do timer também pode ser utilizado para a geração de sinais de PWM (Pulse Width Modulation), e para isso, foi criada esta classe especializada, que é capaz de produzir uma saída digital que possua um duty cycle e frequência definidos; portanto, foram desenvolvidas as funções `set_duty_cycle` e `set_frequency` capazes de definir os respectivos parâmetros da PWM.

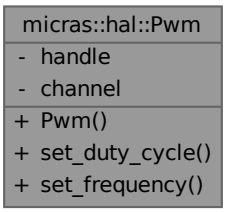


Figura 25 – Diagrama de colaboração da classe Pwm

4.3.2.3 PWM DMA

Uma outra forma de utilizar a PWM é através do DMA (Direct Memory Access), no qual ao invés de definir o duty cycle diretamente, este é passado através de um array, que é automaticamente percorrido. Ou seja, é enviado um sinal serial através da PWM, no qual o duty cycle de cada ciclo é a informação relevante. Para isso, foram definidas as funções `start_dma`, que inicializam o DMA, considerando dados de 16 e 32 bits, além de uma função `stop_dma` para interromper o envio de dados.

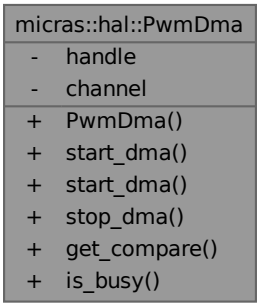


Figura 26 – Diagrama de colaboração da classe PwmDma

4.3.2.4 ADC DMA

Outro periférico que também se beneficia do DMA é o ADC (Analog-to-Digital Converter), que é capaz de realizar leituras analógicas em um conjunto de pinos, e sequencialmente converte-las em valores digitais, escrevendo os resultados ao longo de um array. Para isso, foram implementadas as funções `start_dma` e `stop_dma` que inicializam e interrompem as leituras, respectivamente.

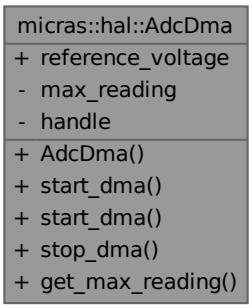


Figura 27 – Diagrama de colaboração da classe AdcDma

4.3.2.5 CRC

O CRC (Cyclic Redundancy Check) é um método de detecção de erros em mensagens que também possui um periférico especializado no microcontrolador, e para utilizá-lo, basta chamar a função `calculate` implementada, passando a mensagem na qual se deseja calcular o CRC.

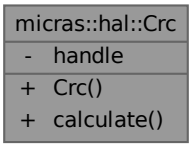


Figura 28 – Diagrama de colaboração da classe Crc

4.3.2.6 SPI

Outro periférico de grande relevância para o projeto é o SPI (Serial Peripheral Interface), que permite a comunicação do microcontrolador com outros dispositivos através de um mesmo barramento serial, sendo necessário apenas adicionar um GPIO de chip-select de forma exclusiva para cada conexão. Com isso, foram implementadas as funções de `select_device` e `unselect_device` que permitem ativar e desativar a comunicação com o dispositivo através do chip-select, e as funções de `transmit` e `receive`, que enviam e recebem um conjunto de dados, respectivamente.

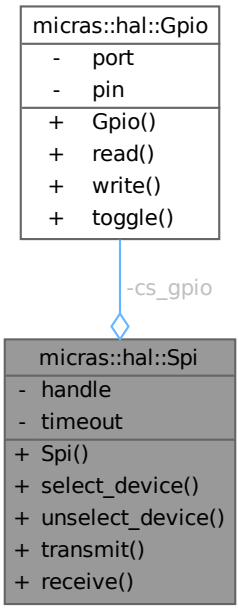


Figura 29 – Diagrama de colaboração da classe Spi

4.3.2.7 Flash

O microcontrolador escolhido para o projeto também permite que o usuário acesse diretamente a sua memória não volátil, de forma a possibilitar o armazenamento de informações que são mantidas mesmo com o desligamento do robô, como constantes de calibração e informações sobre o labirinto. Mas a utilização da memória flash não é tão simples quanto a memória convencional, visto que apresenta diversos perigos que podem causar o mau funcionamento do programa por usuários despreparados.

Dessa forma, a classe que utiliza a memória flash visa facilitar a sua utilização, e impedir que acessos sejam realizados em regiões não permitidas da memória, integrando diversas funcionalidades. O primeiro ponto de atenção é o fato de que o código sendo executado também se encontra na memória flash, mais especificamente, em seu início; portanto, a classe implementada armazena informações a partir do final da memória, evitando possíveis conflitos com o código fonte, que seriam extremamente difíceis de identificar e corrigir.

Outra complicação que a memória flash traz é na deleção de dados, que deve ser realizada em páginas de 2 kB. Não é possível simplesmente sobrescrever uma informação previamente escrita. Caso não seja desejável a perda de nenhum dado, toda a página deve ser lida para a memória RAM. As alterações devem ser realizadas para que a página possa ser apagada e os novos dados escritos.

Portanto, para a utilização da memória flash, foram desenvolvidas funções de `read` e `write` que podem receber endereços absolutos na memória, contando a partir do final, ou endereços relativos à página que se deseja ler ou escrever um dado, respectivamente. Também é necessária uma função de `erase`, que apaga todos os dados presentes em uma página.

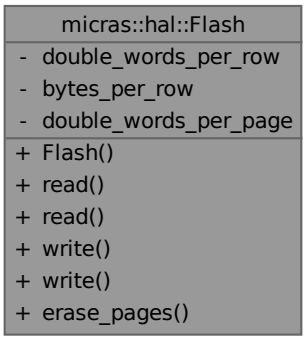


Figura 30 – Diagrama de colaboração da classe Flash

4.3.3 Hardware Proxies

Com a camada de mais baixo nível abstraída, os periféricos podem ser acessados de forma simplificada e seguindo os padrões de código modernos do C++. Mas ainda existem componentes de hardware que necessitam desses periféricos, e que poderiam ser utilizados em diversas partes do código; portanto, foi desenvolvida mais uma camada de abstração baseada no livro Design Patterns for Embedded Systems in C (DOUGLASS, 2011), os Hardware Proxies, que permitem que componentes de hardware, como botões, LEDs e outros sensores e atuadores, possam ter sua utilização simplificada pelo usuário.

Essas proxies também podem servir como uma interface para diferentes ambientes, ou seja, é possível implementar proxies que sejam executadas em um simulador, interfaceando com ROS 2 e Gazebo, por exemplo, e proxies que sejam executadas no dispositivo embarcado, se comunicando com os sensores e atuadores, possibilitando que o mesmo código de alto nível seja executado em diferentes cenários.

4.3.3.1 LED

O primeiro componente a ser representado foi o LED, que pode ser definido como uma simples implementação de um GPIO de saída, bastando então desenvolver as funções `turn_on`, `turn_off` e `toogle`, que são responsáveis por ligar, desligar e trocar o estado atual do LED, respectivamente.

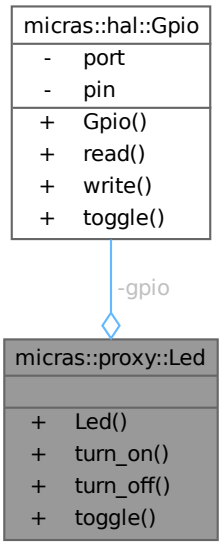


Figura 31 – Diagrama de colaboração da classe Led

4.3.3.2 Botão

Em seguida, foi implementada a classe do Botão, que além de implementar um GPIO de entrada, também utiliza 2 timers, de forma a lidar com o possível debounce que pode ocorrer no sinal ao ser pressionado e retornar diferentes status para tempos de atuação distintos. Com isso, a classe resultante possui as funções `is_pressed`, que verifica se o botão está pressionado no instante atual, já considerando o debounce, e a `get_status`, que retorna o último tipo de atuação que ocorreu no botão, que pode ser um `NO_PRESS`, `SHORT_PRESS`, `LONG_PRESS`, ou um `EXTRA_LONG_PRESS`.

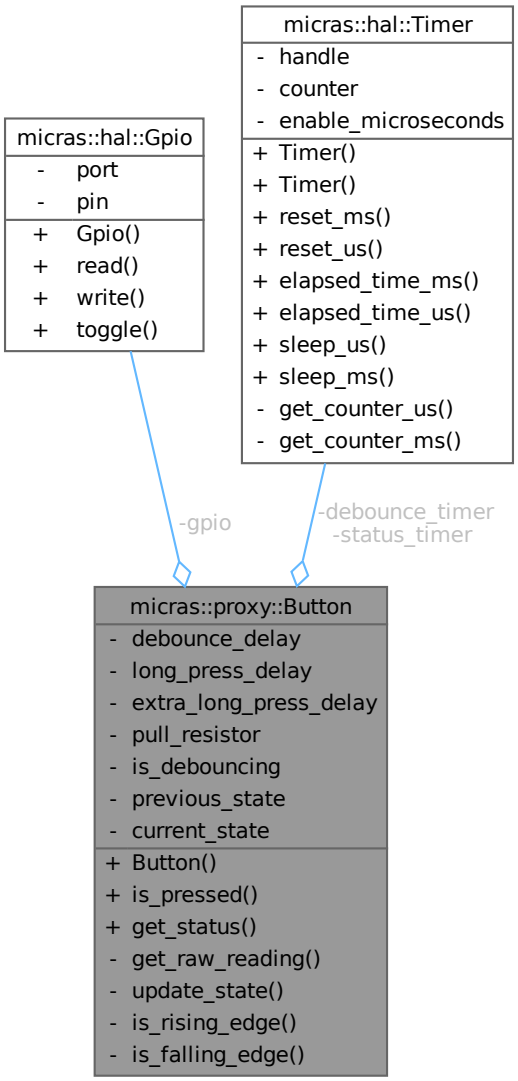


Figura 32 – Diagrama de colaboração da classe Button

4.3.3.3 DIP Switch

O interruptor DIP (Dual In-line Package) representa um array de GPIOs de entrada, mas com implementações individuais mais simples que a do Botão, visto que não é necessário lidar com o debounce ou tempo de atuação, já que a chave possui 2 estados estáveis. Dessa forma, a classe foi desenvolvida com o número de canais do switch sendo um template de C++, e possui as funções `get_switch_state`, para retornar o valor atual de um determinado switch e `get_switches_value`, que interpreta a posição dos switches como um número binário e retorna o número inteiro correspondente.

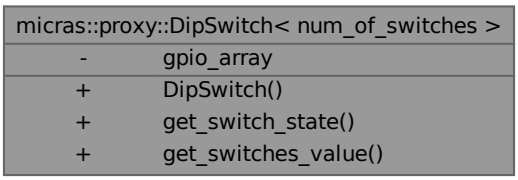


Figura 33 – Diagrama de colaboração da classe DipSwitch

4.3.3.4 Buzzer

Esta classe simplifica a utilização de um buzzer ativo, de forma a enviar uma PWM de frequência definida através da função `play`, que também utiliza um timer para garantir que cada nota seja enviada durante o período de tempo correto. Além disso, também é possível interromper o funcionamento do buzzer com a função `stop` ou definir um tempo de pausa entre as notas após o fim da nota atual, utilizando a função `wait`.

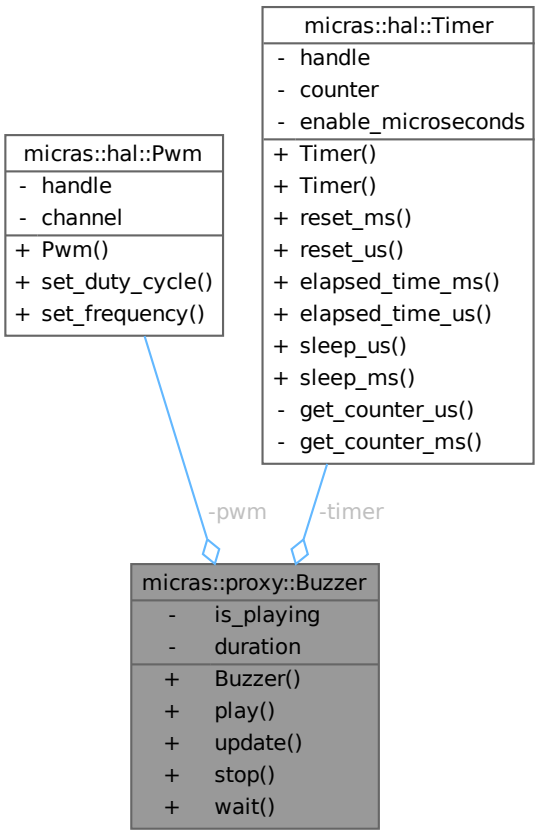


Figura 34 – Diagrama de colaboração da classe Buzzer

4.3.3.5 Motor

Esta classe facilita a utilização do driver escolhido para os motores no projeto, que de acordo com o datasheet (STMICROELECTRONICS, 2024a), pode ser controlado através do envio de 2 sinais PWM, que representam cada um dos sentidos de rotação do motor. Dessa forma, a função `set_command` recebe um valor que representa a porcentagem da velocidade máxima que o motor deve girar, e a classe produz a partir disto, os sinais PWM correspondentes.

Um detalhe que também foi considerado no desenvolvimento da classe é o deadzone, que representa a faixa inicial do comando de velocidade na qual o motor não possui torque suficiente para se locomover, o que acaba tornando a relação entre comando enviado e velocidade realizada menos linear. Para contornar isso, é realizado um mapeamento do comando de entrada da função, para valores acima do deadzone, linearizando a relação entrada/saída.

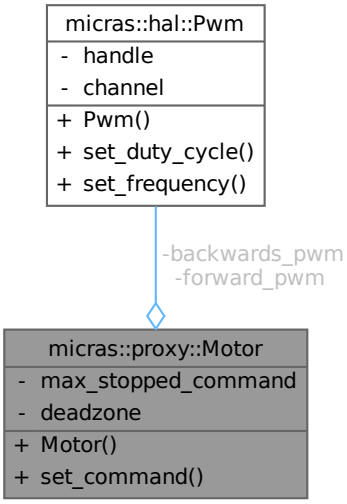


Figura 35 – Diagrama de colaboração da classe Motor

4.3.3.6 Ventoinha

Conforme detalhado em seu datasheet (STMICROELECTRONICS, 2024c), o driver escolhido para a ventoinha possui um mecanismo de controle diferente do escolhido para a locomoção, visto que ele é atuado através de uma PWM que controla a velocidade de rotação, um GPIO de enable, e um GPIO para controlar o sentido de rotação. Além disso, o controle da ventoinha utiliza um timer para limitar a variação do comando enviado em um determinado período de tempo, de forma a evitar que a aceleração da ventoinha fique alta o suficiente para gerar correntes acima das suportadas pelo driver. A utilização

da classe ocorre através das funções `set_speed`, que define o comando de velocidade da ventoinha, e `set_direction`, que define sua direção. Também é possível ativar e desativar o driver com as funções `enable` e `disable`, respectivamente.

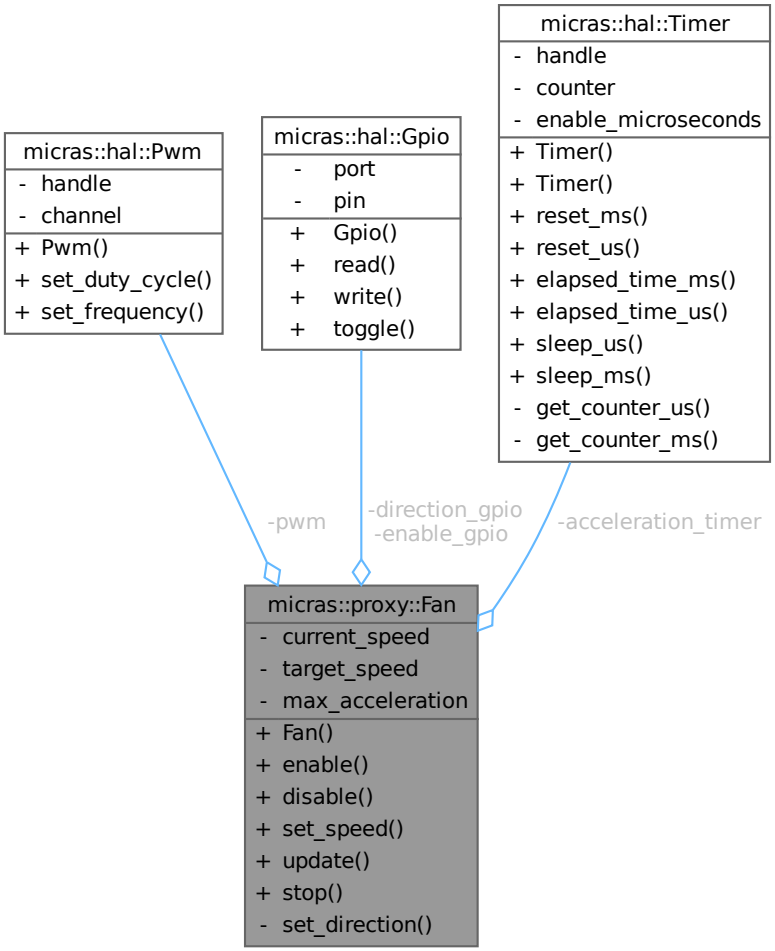


Figura 36 – Diagrama de colaboração da classe Fan

4.3.3.7 Locomoção

Visando simplificar ainda mais a interface com o hardware do projeto, foi desenvolvida a proxy da locomoção, que agrupa em uma classe as proxies dos dois motores que são utilizados para movimentar o robô, lidando com a conversão entre velocidade linear e angular, que a função `set command` recebe, para as velocidades de cada roda, que são enviadas para a função `set_wheel_command`. Além disso, a classe também integra um GPIO, que é utilizado nas funções `enable` e `disable` para ligar e desligar o driver escolhido.

A conversão entre as velocidades é realizada através das seguintes fórmulas:

$$D_{left} = D_{linear} - D_{angular}$$
$$D_{right} = D_{linear} + D_{angular} \tag{4.1}$$

Considerando que D é sempre um valor de 0 a 100, que representa uma porcentagem do comando máximo. Pode-se notar que é possível obter um valor maior que 100 em alguma das rodas, sendo necessário saturar este comando. Uma possibilidade seria simplesmente truncar o valor excedente, mas isso poderia gerar distorções no controle, visto que iria alterar a relação entre as velocidades de cada roda. A fim de evitar este cenário, os comandos que ultrapassarem 100 sempre são saturados em conjunto, mantendo a relação entre as velocidades, e portanto, o raio de curvatura da trajetória percorrida pelo robô.

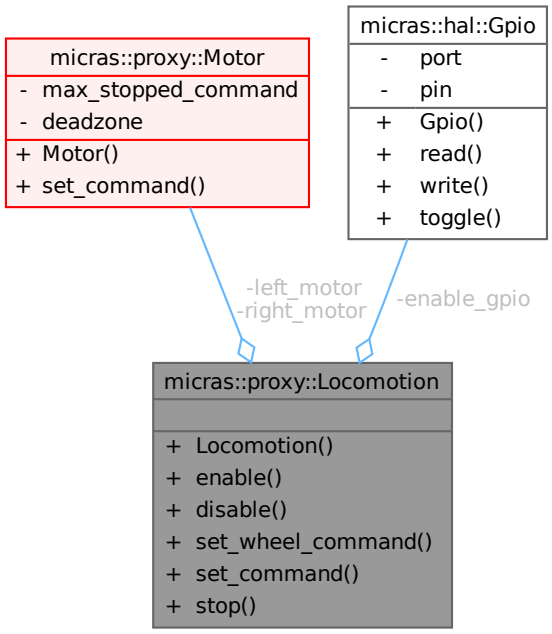


Figura 37 – Diagrama de colaboração da classe Locomotion

4.3.3.8 LED RGB Endereçável

Esta classe permite definir através de um sinal serial único, a cor de um conjunto de LEDs RGB encadeados, e para isso, é utilizado um protocolo que tem como base um sinal PWM como descrito no datasheet do componente (SEMI, 2020). Dessa forma, a proxy utiliza o PWM com DMA, percorrendo um buffer e enviando os valores correspondentes através do duty cycle do sinal. A proxy possibilita o controle dos LEDs de diferentes formas; definindo a cor de um LED específico, com a função `set_color`, ou de todos os LEDs ao mesmo tempo, com a função `set_colors`.

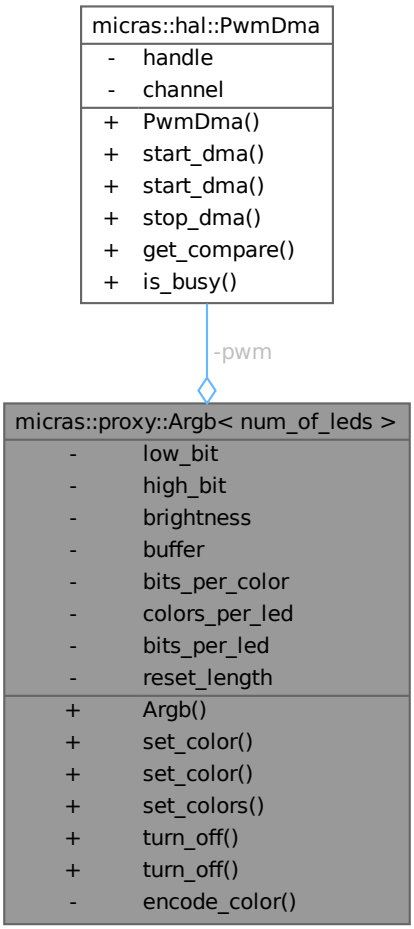


Figura 38 – Diagrama de colaboração da classe Argb

4.3.3.9 Bateria

O microcontrolador utilizado possui um pino V_{bat} dedicado à medição da bateria, que é conectado a um divisor de tensão de 3 para 1, e então interligado a um dos canais do ADC, de forma a possibilitar a medição de tensões de até $3 \cdot 3.3\ V = 9.9\ V$. Porém, os valores observados comumente se apresentavam ruidosos, de forma que foi necessário integrar um filtro passa-baixa no sinal, resultando em um valor mais suave que pode ser obtido através da função `get_voltage`, enquanto a versão original do sinal ainda está disponível, utilizando a função `get_voltage_raw`.

Visando tornar a implementação do filtro mais genérica, foi desenvolvida uma classe para um filtro de Butterworth de segunda ordem, de forma que este pudesse ser utilizado em outros locais do projeto com diferentes frequências de corte.

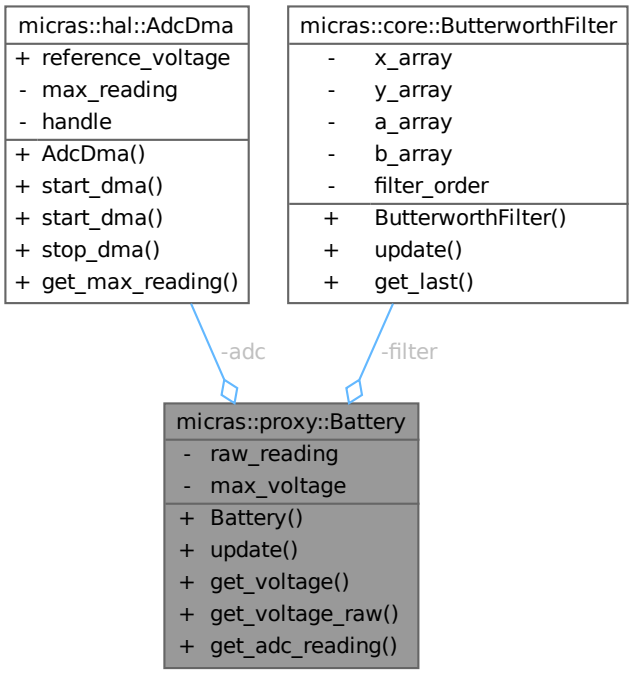


Figura 39 – Diagrama de colaboração da classe Battery

4.3.3.10 Sensor Inercial

O IMU (Inertial Measuring Unit) é um dos componentes mais importantes para o funcionamento do projeto, visto que fornece informações sobre a velocidade angular e aceleração linear do robô nos 3 eixos, através de uma interface SPI. Estes valores podem ser obtidos através das funções `get_angular_velocity` e `get_linear_acceleration`, que recebem o eixo da informação desejada.

Também foi desenvolvida uma função de calibrate, que visa amenizar o problema de *drift*, muito comum em sensores inerciais, nos quais uma pequena medição residual causa uma movimentação na posição angular calculada do robô, que se acumula ao longo do tempo. A função implementada utiliza um filtro de Butterworth para estimar esta medição residual, e a remove do valor medido antes de retorná-lo ao usuário.

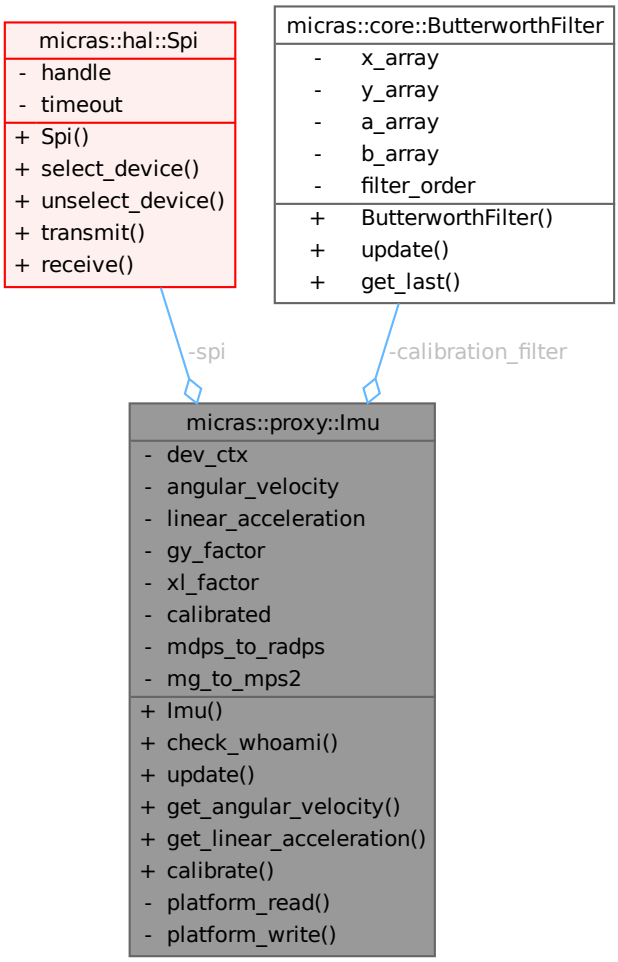


Figura 40 – Diagrama de colaboração da classe Imu

4.3.3.11 Sensor rotativo

Os sensores rotativos utilizam os timers no modo encoder para armazenar de forma paralela à execução do processador a posição angular atual das rodas do robô. Além disso, esses componentes possuem diversas configurações, que podem ser enviadas através da interface SPI, conforme descrito no datasheet do componente (OSRAM, 2024b).

Dessa forma, o valor da posição angular atual medido pelo sensor pode ser obtido de duas formas, com a função `get_position`, que utiliza a informação armazenada no timer, e com a função `read_register`, que se comunica com o sensor através do SPI, visando obter esta e outras informações.

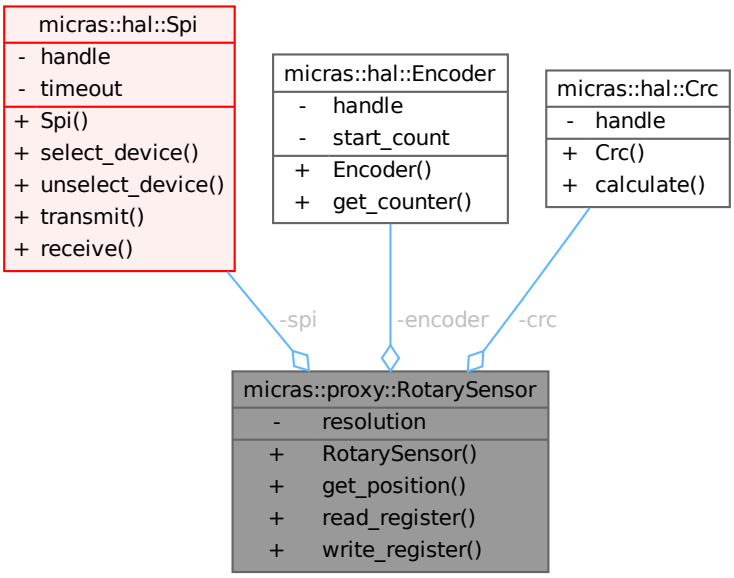


Figura 41 – Diagrama de colaboração da classe RotarySensor

4.3.3.12 Armazenamento

Como discutido anteriormente, a utilização da memória flash traz diversas complicações e riscos, portanto, visando sanar esses problemas, a classe de armazenamento torna extremamente fácil salvar valores e até classes inteiras na memória não-volátil do microcontrolador. Isto é feito utilizando mapas da biblioteca padrão do C++, que armazenam informações sobre a posição de cada variável na memória RAM e no buffer interno da classe, além disso, foi criada uma classe virtual `ISerializable`, que define funções de serialização e desserialização. Dessa forma, qualquer classe que implemente esses métodos virtuais herdados pode ser serializada e portanto, armazenada na memória flash.

Para utilizar a proxy, basta adicionar variáveis utilizando a função `create`, e quando desejar obter o valor da variável armazenado na memória, deve-se utilizar a função `sync`.

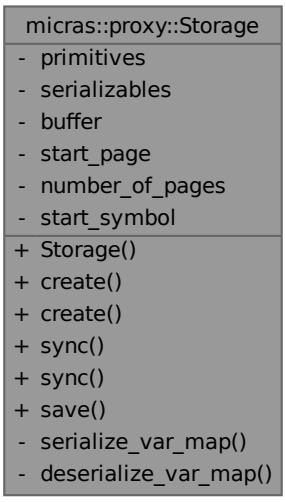


Figura 42 – Diagrama de colaboração da classe Storage

4.3.3.13 Sensores de Parede

Os sensores de parede são constituídos de dois componentes distintos, um LED infravermelho, controlado com uma PWM através das funções `turn_on` e `turn_off`, e um fototransistor, que resulta em um sinal analógico correspondente à intensidade da iluminação de uma determinada frequência que o atinge. Esse sinal é lido através de um ADC, e seu valor correspondente pode ser lido através da função `get_reading`.

A iluminação ambiente também possui grande efeito sobre os fototransistores utilizados para o sensoramento, de forma que para identificar corretamente paredes próximas ao robô, é necessário isolar a parte da medição correspondente á luz que é enviada pelo LED e refletida pelas paredes. Isso é feito piscando o LED milhares de vezes por segundo, e realizando medições com ele ligado e desligado, de forma que a diferença entre os valores corresponde à parcela emitida pelo LED.

Outro fator importante é a imprecisão proveniente do método de medição, que torna inviável obter a distância exata que a parede se encontra; portanto, os sensores devem ser calibrados para detectar a presença ou não de paredes em diversas situações previamente estabelecidas, tornando a utilização mais consistente.

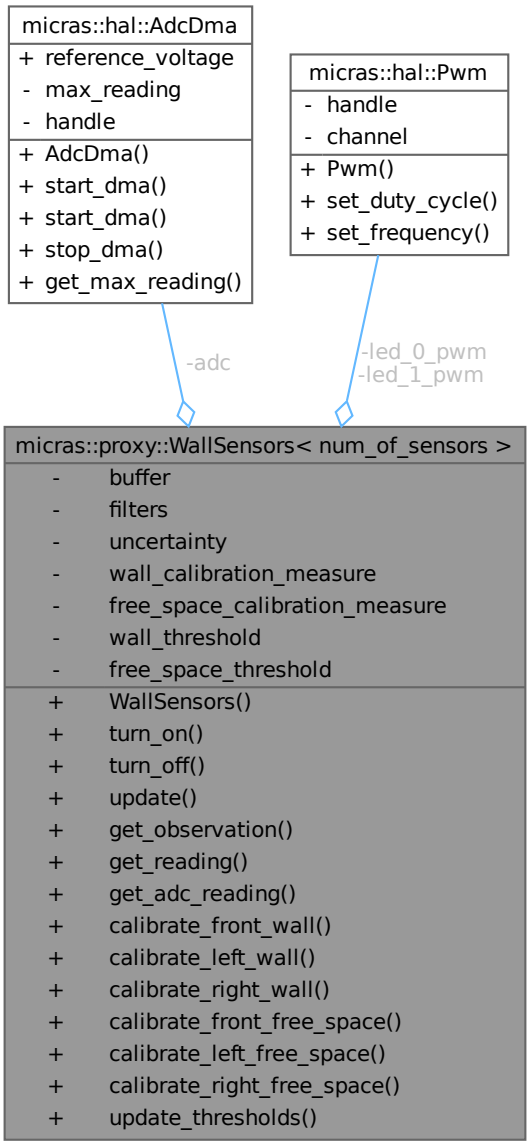


Figura 43 – Diagrama de colaboração da classe WallSensors

4.3.4 Alto Nível

4.3.4.1 Odometria

A odometria visa estimar o estado atual do robô em relação ao ponto inicial, isto é, sua posição x e y no plano, orientação no eixo z perpendicular a este plano, e suas velocidades linear e angular. Para isso, a primeira etapa é utilizar as proxies dos sensores rotativos para obter a posição angular de cada roda Θ_{left} e Θ_{right}, que podem ser subtraídas das posições armazenadas do último ciclo, obtendo as distâncias angulares Δ_Θ de cada roda.

Em sequência, a distância angular pode ser multiplicada pelo raio da roda, de forma a obter a distância linear percorrida por cada roda desde o último ciclo:

$$d_{wheel} = r_{wheel} \cdot \Delta_{\Theta}$$

As distâncias percorridas por cada roda podem ser utilizadas para calcular a distância linear percorrida pelo robô:

$$d_{lin} = \frac{d_{left} + d_{right}}{2}$$

Esta distância linear pode ser dividida pelo tempo decorrido desde o último ciclo, de modo a obter a velocidade linear instantânea do robô:

$$v_{lin} = \frac{d_{lin}}{\Delta_t}$$

Já a velocidade angular do robô pode ser adquirida através da proxy do sensor inercial, e ao ser multiplicada pela velocidade, tem-se a distância angular percorrida pelo robô desde o último ciclo.

$$d_{ang} = \omega \cdot \Delta_t$$

Por fim, é possível estimar as alterações na posição do robô através das seguintes fórmulas:

$$\Delta_x = d_{lin} \cdot cos\left(\Theta + \frac{\Delta_{\Theta}}{2}\right)$$
$$\Delta_y = d_{lin} \cdot sen\left(\Theta + \frac{\Delta_{\Theta}}{2}\right)$$

Um desafio nesse processo foi a acumulação de erros ao longo do tempo devido ao deslizamento das rodas ou a pequenas imperfeições no chão do labirinto. Para mitigar esse problema, o valor do estado atual do robô pode ser resetado com base na célula em que ele se encontra; isso ocorre em situações pré-definidas, como quando o robô se alinha com a parede de trás ou quando os sensores de parede realizam uma medição pré-calibrada.

4.3.4.2 Mapeamento

Outro componente essencial do sistema foi o algoritmo de reconhecimento de paredes, que utilizou os dados dos sensores infravermelhos para detectar a presença de paredes ao redor do robô. Para processar os dados desses sensores, foi implementado um

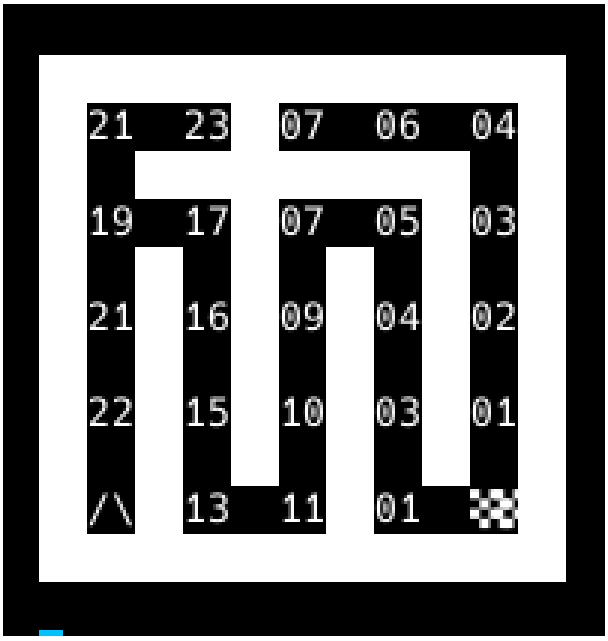


Figura 44 – Mapeamento do labirinto na memória do robô

sistema de verificação que compara os valores de distância lidos com um limiar previamente calibrado. Quando o valor de leitura ultrapassava esse limiar, o sistema identificava que havia uma parede próxima. Esses dados eram então utilizados para construir um mapa interno do labirinto, que foi armazenado na memória do robô.

O processo de mapeamento do labirinto foi realizado utilizando uma estrutura de dados de grade (grid), onde cada célula representava uma possível posição de parede ou espaço livre. À medida que o robô se movia, os dados dos sensores eram utilizados para atualizar o estado dessas células, permitindo que o robô construísse uma representação digital precisa do labirinto. O algoritmo de mapeamento utilizado foi uma adaptação do SLAM (Simultaneous Localization and Mapping), onde o robô simultaneamente atualiza sua posição no labirinto e constrói o mapa ao seu redor.

4.3.4.3 Planejamento de Trajetória

Após o mapeamento, foi implementado o algoritmo de resolução do labirinto, responsável por encontrar o caminho mais curto até a saída. O algoritmo escolhido para essa tarefa foi uma versão otimizada do A* (A-estrela), que utiliza uma função de heurística para estimar a distância até a saída e selecionar o caminho mais eficiente. Esse algoritmo foi executado após o robô explorar o labirinto completamente, utilizando o mapa gerado para calcular a rota mais curta.

Por fim, o algoritmo de exploração foi responsável por guiar o robô através do labirinto durante a fase de descoberta. O robô seguia um padrão de movimentação baseado em algoritmos de cobertura, que garantiam que ele explorasse todas as áreas possíveis do

labirinto. Ao identificar bifurcações ou áreas desconhecidas, o algoritmo decidia o próximo movimento com base nas informações já conhecidas, garantindo uma exploração eficiente.

Essa combinação de algoritmos e abstrações permitiu que o robô navegasse com eficiência pelo labirinto, identificando e mapeando as paredes, calculando rotas otimizadas e ajustando suas decisões de movimento em tempo real.

4.3.4.4 Controle

O principal algoritmo de controle utilizado foi o PID (Proporcional-Integrativo-Derivativo), que possibilita rápidos tempos de resposta, impede oscilações e evita erros estacionários, sendo essencial tanto para o controle angular quanto linear do robô.

Para o controle angular, o PID foi aplicado de duas formas distintas, a primeira e mais precisa, ocorre com a presença de paredes nas laterais do robô, de forma a possibilitar a utilização das medições dos sensores de distância para garantir que o robô permaneça alinhado e centralizado entre as duas paredes. No segundo caso, o controle é realizado com base na odometria, tendo um ponto final como objetivo, uma reta é traçada desde o ponto inicial, e o desvio da trajetória do robô em relação a esta reta define o erro angular que deve ser corrigido pelo controlador.

Já no caso do controle linear, o PID foi usado com base na odometria para regular a velocidade do robô, garantindo que ele acelerasse em retas longas e desacelerasse para evitar derrapagens nas curvas, aproveitando o máximo do percurso calculado.

Ambos os controles, linear e angular, foram então combinados para gerar um comando completo para o robô, com a resposta angular sendo ajustada com base na velocidade do robô, de forma a possibilitar uma rápida reação em altas velocidades. Este processo assegurava que o robô seguisse a trajetória planejada com precisão, ajustando automaticamente seus movimentos à medida que se deslocava no ambiente.

4.3.5 Infraestrutura de Comunicação

Abaixo, consta a infraestrutura de comunicação dos blocos dos componentes do projeto. Através deles, fica claro o que são as entradas e saídas do projeto e como a MCU opera sobre elas.

4.4 Placa do robô

O projeto da placa do Micro Mouse mira em desenvolver uma solução robusta capaz de satisfazer todas as necessidades do projeto de forma eficiente e confiável. Para isso, optou-se por desenvolver uma placa original com 4 camadas e que foi fabricada através de serviços de terceirização de manufatura de PCBs (*JLCPCB*).

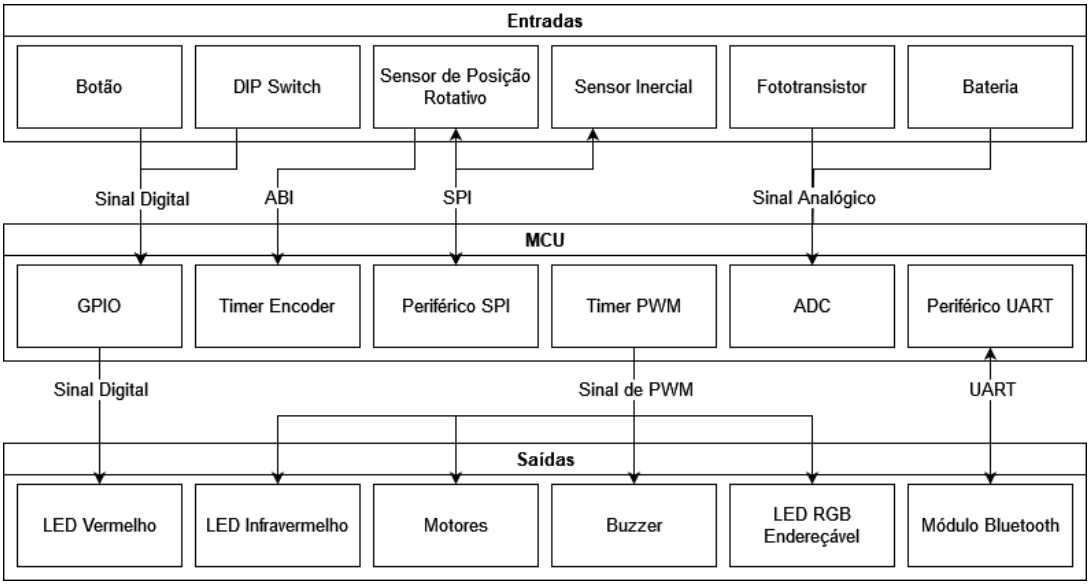


Figura 45 – Diagrama de comunicação do robô

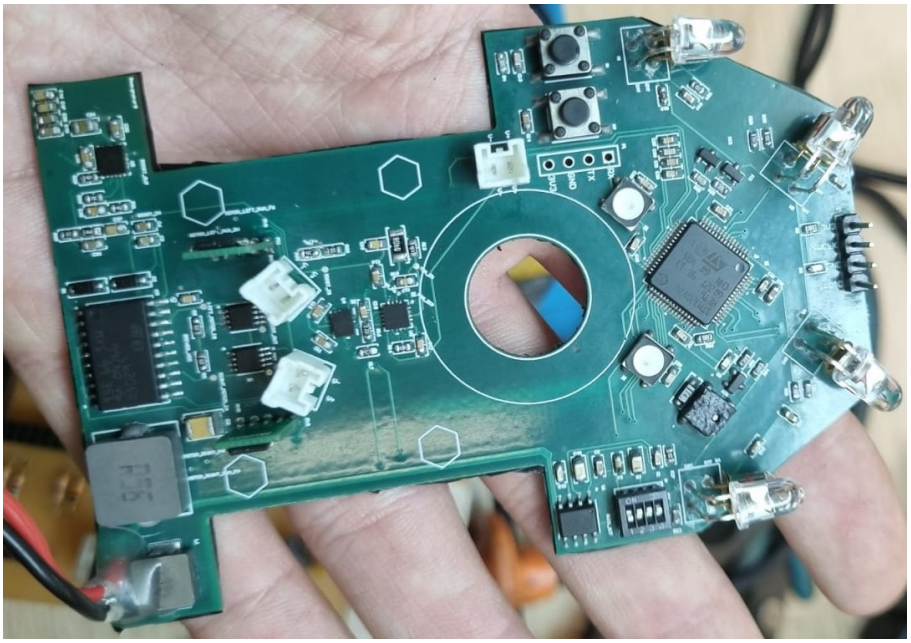
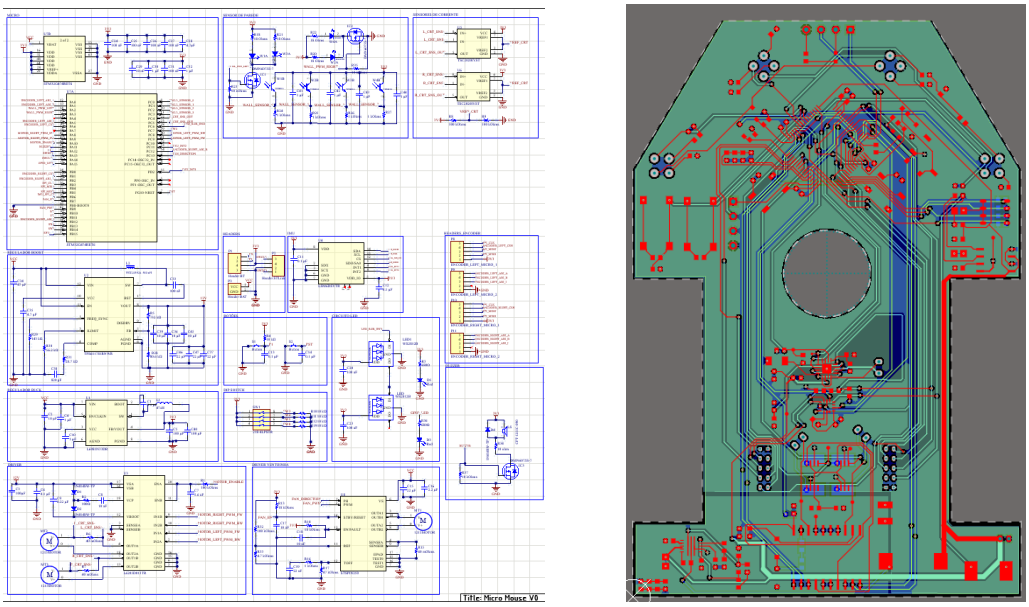


Figura 46 – Placa com todos os componentes soldados

A seguir, detalham-se os principais aspectos de cada circuito do Micro Mouse, além de uma breve explicação sobre os motores escolhidos para o robô.

4.4.1 Projeto e Implementação

A placa é alimentada por uma bateria de lítio de duas células (7.4V) que, por sua vez, alimenta dois circuitos independentes: o de potência (12V), responsável pela alimentação dos motores do projeto, e o lógico (3.3V), encarregado pela alimentação do microcontrolador e de toda a parte lógica do robô.



(a) Esquemático da placa (b) PCB com 4 camadas sobrepostas

Figura 47 – Projeto elétrico da placa

4.4.1.1 Circuito de potência

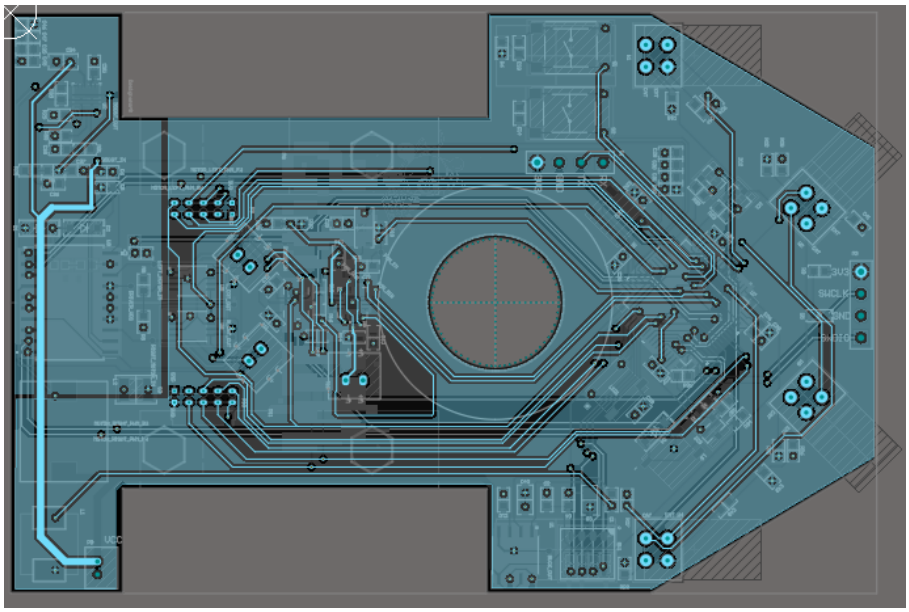


Figura 48 – Design da camada de potência da PCB

O circuito de potência do projeto parte de um regulador boost da Texas Instruments, o TPS61178 (TI, 2024). Dentre as opções analisadas, ele foi escolhido por satisfazer todas as necessidades dimensionadas para o robô, ou seja, Vmin = 7.2V, Vmax = 8.4V, Vout = 12V e Iout = 4.5A, além de possuir *Load Disconnect*, ou seja, o circuito do regulador permite um threshold de corrente que desconecta a carga caso a corrente passe do limite estabelecido pelo projeto.

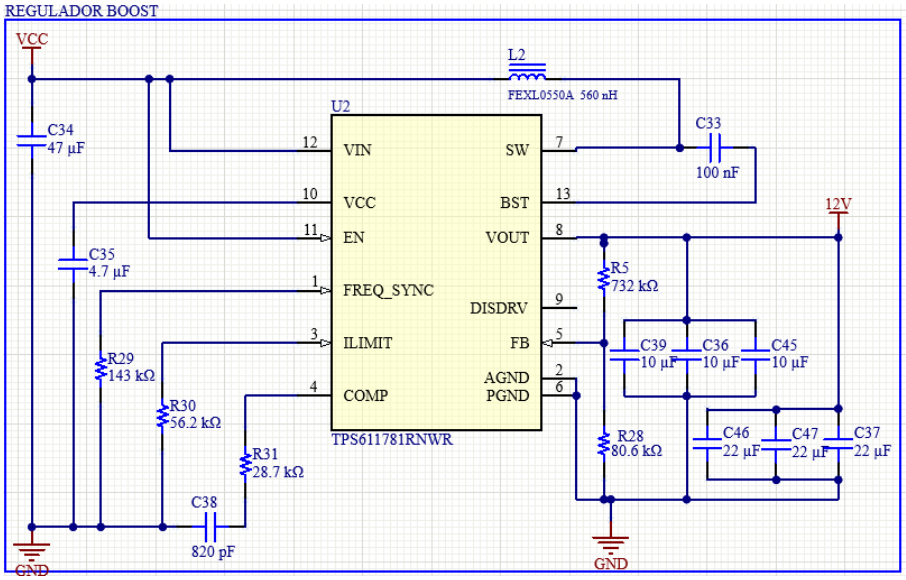


Figura 49 – Esquemático do circuito do regulador boost

O circuito do boost foi projetado com auxílio do Datasheet do componente e da calculadora de componentes da própria fabricante do regulador boost. A saída do circuito de boost vai diretamente para os outros dois componentes centrais do circuito de potência — o driver da ventoinha e o driver da locomoção.

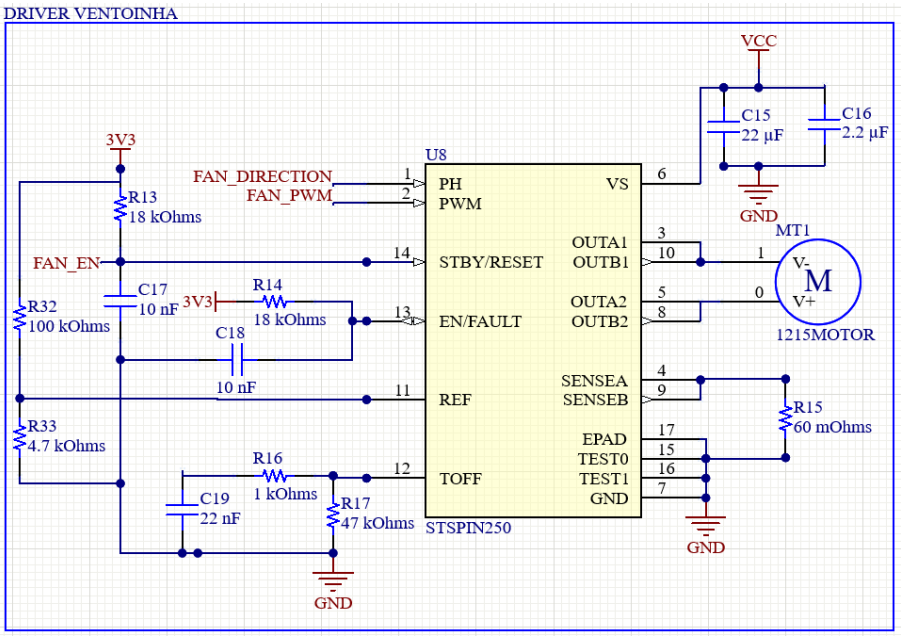


Figura 50 – Esquemático do circuito do driver da ventoinha

Para a ventoinha, utilizou-se um driver da STMicroelectronics, o STSPIN250 (STMICROELECTRONICS, 2024c). Ele trata-se de uma solução simples e eficiente para o controle da ventoinha, que é simples uma vez que não requer controle muito avançado como a locomoção. Ele possui controle de velocidade através de PWM, é não dual, portanto,

tem um encapsulamento menor do que seu equivalente dual e conta com proteção de sobrecorrente. Também vale notar que o driver da ventoinha tem sua alimentação ligada diretamente na bateria.

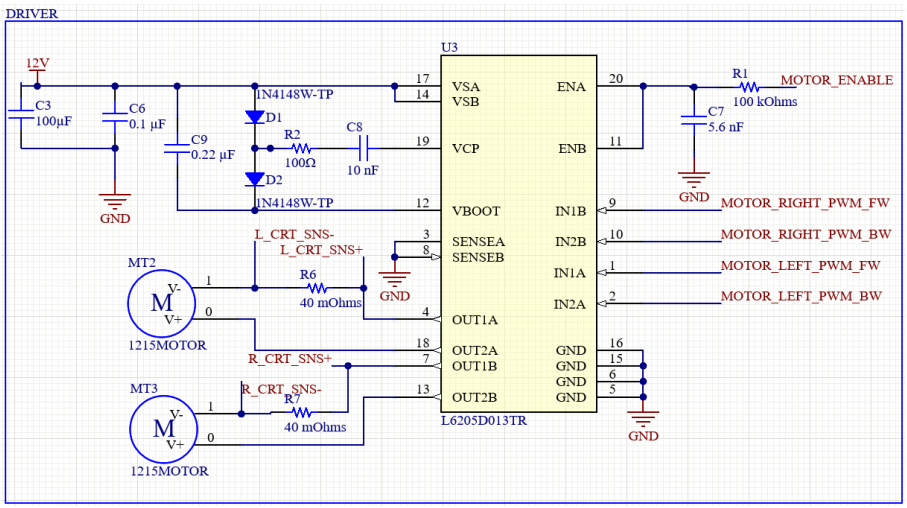


Figura 51 – Esquemático do circuito do driver da locomoção

Já para a locomoção, optou-se por um driver dual que permite o controle de ambos os motores responsáveis pelo movimento do Micro Mouse e o componente também é um driver da STMicroelectronics, o L6205D013TR(STMICROELECTRONICS, 2024a). Além de ser dual, também consta suportar uma entrada de 8V a 52V, 5.6A de corrente de saída de pico, frequência de operação de até 100kHz e shutdown térmico, que busca mitigar qualquer problema oriundo de um curto ou travamento dos motores que possa causar sobrecorrente, danificando a placa.

4.4.1.2 Motores

Conhecidos os aspectos centrais do circuito de potência, é importante entender os motores que foram escolhidos para o projeto. Por se tratar de uma categoria que tem requisitos fortes de tamanho e peso, nota-se que os motores devem ser pequenos e leves, porém também precisam ser capazes de desenvolver uma rotação alta para conseguir movimentar o robô em velocidades competitivas. Ao analisar outros competidores da categoria, nota-se que os motores mais utilizados não são motores brushless nem os escovados tradicionais, mas sim os ***motores coreless***.

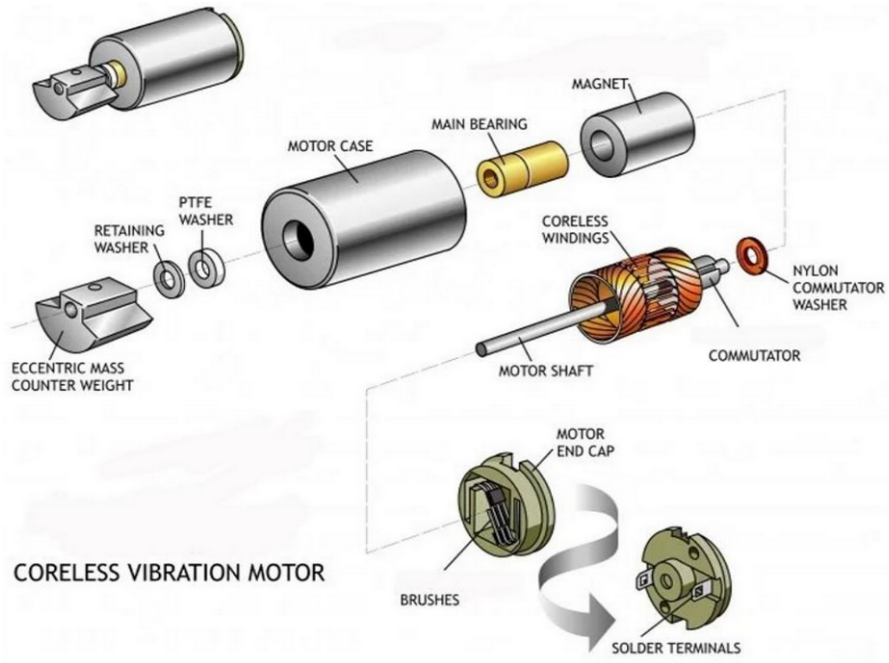


Figura 52 – Esquemático do circuito do driver da locomoção

Os motores coreless são uma evolução dos motores escovados tradicionais, projetados para serem mais leves, eficientes e rápidos. Eles não possuem um núcleo de ferro sólido no rotor, mas sim um enrolamento em forma de cilindro oco, o que reduz significativamente sua inércia e permite acelerações mais rápidas e maior eficiência energética. Contudo, essa ausência de núcleo de ferro também reduz a capacidade de dissipação de calor e aumenta a vulnerabilidade a danos em altas cargas ou condições extremas. Esses *tradeoffs* são aceitáveis em aplicações que demandam alta performance em tamanhos compactos, como micro-robôs, onde leveza e agilidade são priorizadas em detrimento da robustez térmica (IOI, 1999).

Para dimensionar os motores, algumas contas foram feitas visando deliberar velocidade e aceleração máxima. Ainda que os valores sejam puramente teóricos, o método foi importante para conseguir comparar os motores disponíveis online, ainda que as informações fossem limitadas. Abaixo encontram-se as equações utilizadas:

$$v_{max} = \omega_{motor} \cdot \frac{d}{2} \cdot \frac{1}{n} \cdot \frac{V_{in}}{V_{nom}}$$

Onde d é uma constante que equivale ao diâmetro da roda, de $22 \cdot 10^{-3}$ m e n a taxa de redução, que neste caso é 2.5, uma vez que o pinhão tem 8 dentes e as engrenagens das rodas 20.

$$F \cdot v = 2 \cdot \omega \cdot \tau \implies a = \frac{2 \cdot \omega \cdot \tau}{m \cdot v}$$

$$a_{max} = \frac{4 \cdot \tau}{m \cdot d} \cdot n \cdot \frac{V_{in}}{V_{nom}}$$

$$a_{slip} = \frac{m + m_{vent}}{m} \cdot g \cdot \mu$$

$$\omega_{max} = \frac{2 \cdot v_{max}}{l}$$

$$m \cdot a \cdot v = I \cdot \alpha \cdot \omega \mid I = \frac{m \cdot r^2}{2}$$

$$\alpha_{max} = \frac{4 \cdot a_{max}}{l}$$

A partir destas equações, pode-se utilizar os dados dos fabricantes para entender melhor como os motores vão operar dentro das condições fornecidas pelo robô, e tomar a melhor decisão.

Desta forma, optou-se por utilizar dois modelos diferentes de motores coreless: O HZ1020 para a locomoção e o HZ1022 para a ventoinha. Abaixo, encontram-se as tabelas com as dimensionadas para o .

HZ1020 - Entradas		
Dado	**Valor**	**Unidade**
Velocidade Máxima (v_{rot})	17000	rpm
Torque Stall (t)	80	g·cm
Tensão Nominal (V_{nom})	12	V
Redução (n)	2.5	-
Massa Motor (m)	7.5	g

Tabela 1 – Tabela de características fornecidas pelo fabricante acerca do HZ1020

HZ1020 - Saídas		
Dado	**Valor**	**Unidade**
Velocidade Máxima (v_{max})	7.833	m/s
Aceleração Máxima (a_{max})	44.576	m/s^2
Aceleração máxima de Deslizamento (a_{slip})	55.898	m/s^2
Velocidade Angular Máxima (ω_{max})	223.801	rad/s
Aceleração Angular Máxima (ω_{max})	2547.182	rad/s^2

Tabela 2 – Tabela de características equacionadas do HZ1020

Para a ventoinha, os mesmos dados não precisaram ser levantados uma vez que a única característica era a rotação máxima, que deve ser suficientemente alta, então optou-

se por adquirir um 1022 utilizado em drones de brinquedo, que é capaz de desenvolver até 54 kRPM em 7.4 V.

4.4.1.3 Circuito lógico

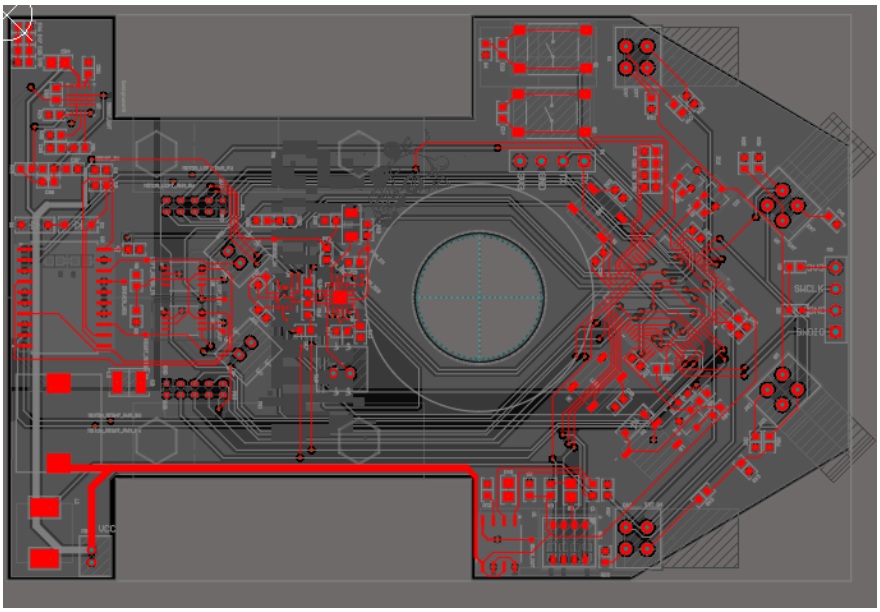


Figura 53 – Design da camada lógica da PCB

A base do circuito lógico é um regulador buck que garante que a parte lógica do robô vai ser alimentada com 3.3V de maneira constante. Para isso, implementou-se um circuito com o L6981N33DR, da STMicroelectronics.

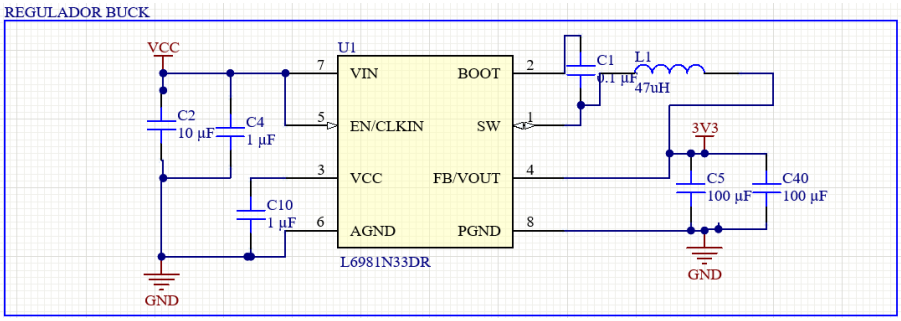


Figura 54 – Esquemático do circuito buck

O circuito foi dimensionado com auxílio da calculadora da STM e garante que a porção lógica do projeto mantém-se com uma tensão de saída (V_{out}) de 3.3V, corrente de saída (I_{out}) de 0.5A e um ripple de 0.01% o que garante a estabilidade necessária para o microcontrolador e os circuitos periféricos.

O componente central do projeto é um microcontrolador da STM, o STM32G474RE, que possui um Cortex M4 de 32 bits com FPU, 170MHz de clock, 128KB de SRAM e

512KB de memória flash. O micro possui 64 pinos, 2 timers de 32 bits, 2 timers de 16 bits, 3 timers com 8 canais de PWM cada, totalizando 24 canais de PWM.

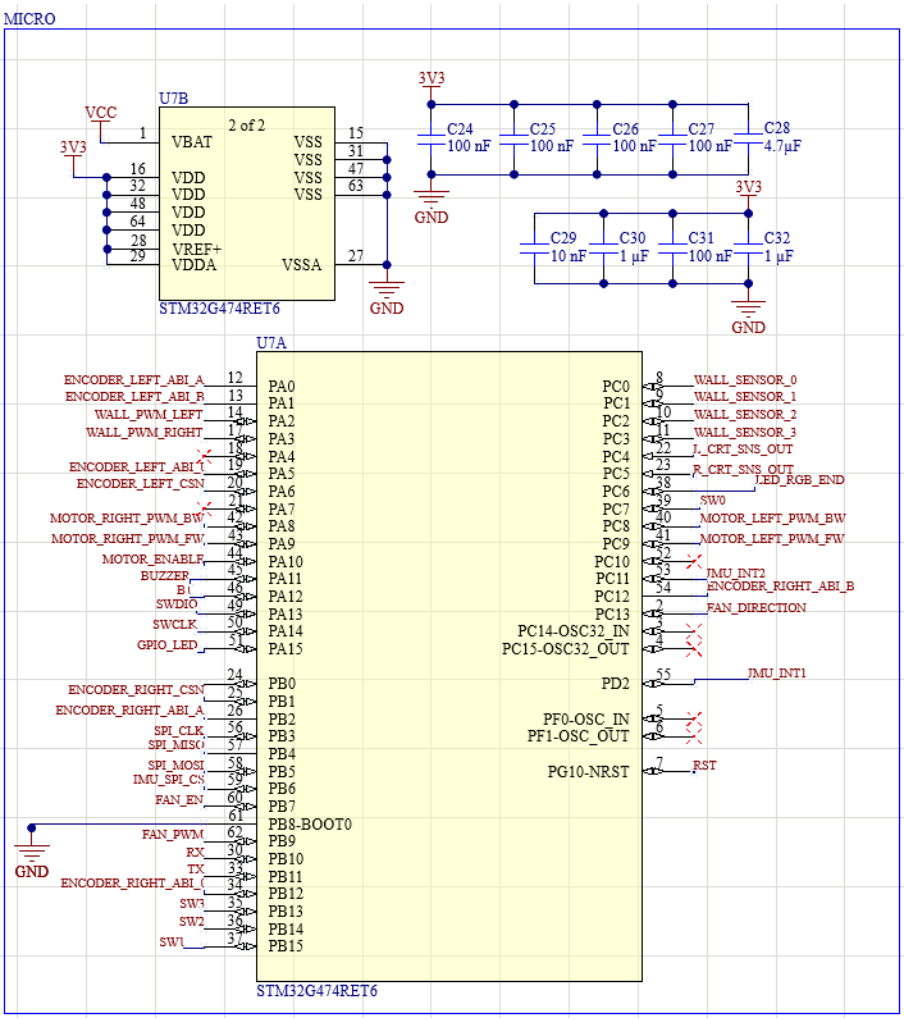
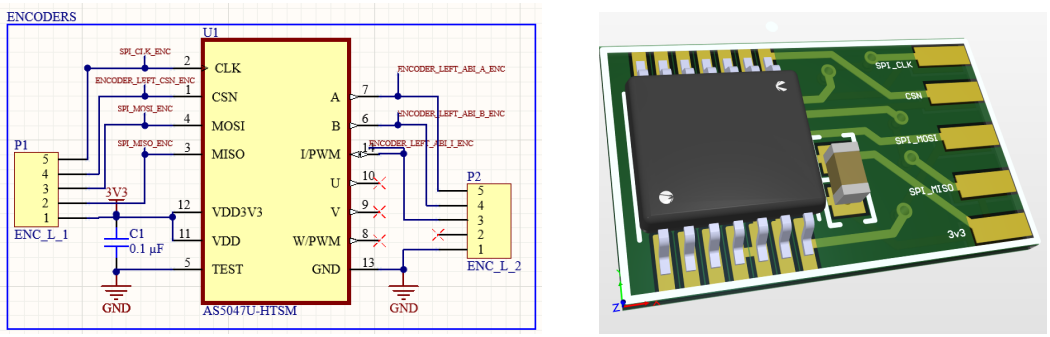


Figura 55 – Esquemático do circuito do microcontrolador

Estas foram as principais características levadas em conta no momento da escolha do chip responsável pelo controle do Micro Mouse. A escolha foi pautada através dos requisitos funcionais estabelecidos para o projeto e também por comparação com projetos ganhadores do mundial no Japão em 2023 (NTF, 2023), ainda que nenhum projeto use este micro, buscou-se um micro da STM equivalente ou melhor em capacidade de processamento do que os microcontroladores utilizados pelos ganhadores, uma vez que a STMicroelectronics forneceu componentes gratuitamente para a execução do projeto.

Ligados ao microcontrolador, existem vários dispositivos utilizados para o sensoriamento do labirinto, eles viabilizam a odometria e garantem que o robô consiga explorar e mapear todo o percurso percorrido. A odometria começa com os encoders magnéticos. Para o projeto, utilizou-se um encoder AS5047U, que possui uma resolução de 16989 passos. A vantagem dos encoders magnéticos sobre os encoders ópticos convencionais

encontra-se na sua resolução muito maior, e em menos espaço, uma vez que operam a partir do efeito Hall ao invés de necessitarem de um disco óptico ou qualquer tipo de estrutura especial para identificar o movimento angular. Neste caso, apenas um pequeno ímã radial é necessário. (RLS, 2019)



(a) Esquemático da placa do encoder (b) PCB independente do encoder

Figura 56 – Projeto elétrico do encoder

Vale notar que, uma vez que o encapsulamento do encoder precisa ser posicionado paralelamente ao eixo que se deseja medir o movimento angular, foi necessário desenvolver uma subplaca que foi soldada perpendicularmente ao plano do chão, desta forma, permitindo que o encoder identificasse corretamente a variação do campo magnético provocada pelo ímã fixado no eixo das rodas.

A odometria não se baseia apenas nas medições do encoder, para adicionar uma camada de redundância e confiabilidade, também implementou-se uma unidade de medição inercial (IMU), que permite identificar o posicionamento do robô no labirinto, tão como sua posição angular.

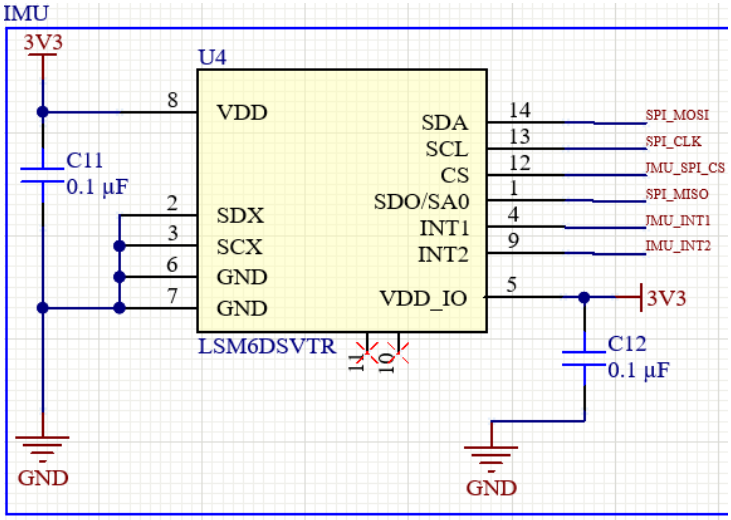


Figura 57 – Esquemático do circuito da IMU

O IMU escolhido foi o LSM6DSV(STMICROELECTRONICS, 2024b) da STM,

que possui encapsulamento pequeno, acelerômetro digital de 3 eixos e giroscópio digital de 3 eixos, permitindo aferição rápida e com baixo ruído da posição do robô.

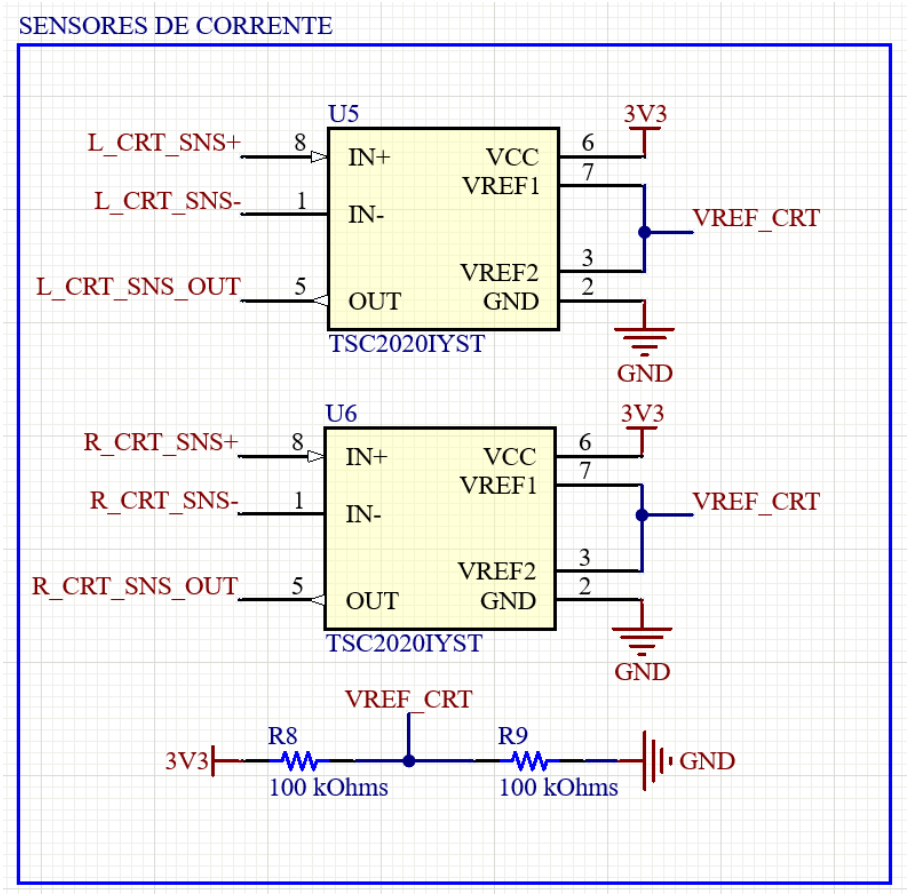
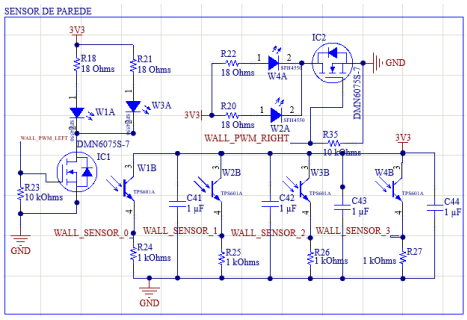


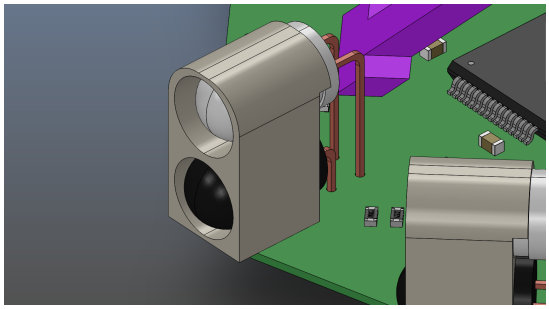
Figura 58 – Esquemático do circuito dos sensores de corrente

Junto com os encoders e IMU, projetou-se um circuito de sensor de corrente para os motores, o que possibilitaria identificar a corrente consumida por cada motor de forma independente, e consequentemente, extrapolar o torque exercido, adicionando ainda mais uma camada de controle e monitoramento dos atuadores do robô.

Por fim, foram desenvolvidos sensores de parede buscando imitar o estado da arte na categoria (RT, 2020), desta forma foram adquiridos pares de fototransistores TPS601A (TOSHIBA, 2024) e LEDs infravermelhos SFH4550 (OSRAM, 2024a) e foi possível construir um sensor de parede idêntico àqueles utilizados nas principais competições da modalidade.



(a) Esquemático do circuito do sensor de parede



(b) Modelo do sensor de parede

Figura 59 – Projeto do sensor de parede

O sensor de parede funciona enviando um sinal de tensão para o micro proporcional à intensidade da detecção de infravermelho captado. O sistema é muito simples eletronicamente, necessitando apenas de um filtro por amostragem implementado a nível de código para operar bem em diferentes condições de iluminação.

4.5 Testes e Avaliação

Nesta seção, detalhamos os testes realizados para garantir o correto funcionamento dos componentes do micro mouse desenvolvido, abrangendo desde os testes unitários até os de integração. Os testes foram planejados com o objetivo de validar o desempenho de cada componente isoladamente, garantindo que funcionalidades críticas como sensores, atuadores, armazenamento de dados e comunicação interna estejam operando conforme esperado.

Para isso, foram desenvolvidos testes unitários específicos que avaliam o comportamento individual de cada parte do sistema, com foco na detecção e correção de possíveis falhas em um estágio inicial do desenvolvimento. Esses testes ajudam a assegurar a confiabilidade dos componentes e a precisão dos dados coletados, essenciais para o desempenho eficiente do robô.

A seguir, listamos os testes unitários realizados, juntamente com uma breve descrição de cada um.

- **test_argb**: Teste para avaliar o funcionamento adequado do RGB endereçável.
- **test_battery**: Teste para monitoramento e desempenho da bateria.
- **test_button**: Teste para validar a funcionalidade do botão.
- **test_buzzer**: Teste para garantir o funcionamento correto do buzzer e seu desligamento ao final.

- **test_dip_switch**: Teste para verificar a operação dos interruptores DIP.
- **test_distance_sensors**: Teste para aferir a precisão dos sensores de distância.
- **test_fan**: Teste para validar o funcionamento do ventilador e seu desligamento ao final.
- **test_imu**: Teste para avaliar o desempenho da Unidade de Medição Inercial (IMU).
- **test_led**: Teste para verificação do funcionamento dos LEDs.
- **test_locomotion**: Teste para verificar a funcionalidade do sistema de locomoção com controle do status dos botões.
- **test_rotary_sensors**: Teste para validar a precisão dos sensores rotativos.
- **test_storage**: Teste para assegurar o armazenamento adequado de dados.
- **test_torque_sensors**: Teste para medir a precisão dos sensores de torque.

Após a validação dos componentes individuais, foram desenvolvidos testes de integração para avaliar o funcionamento conjunto dos sistemas no micro mouse. Esses testes têm como objetivo verificar a interação entre os diferentes módulos — como sensores, atuadores e algoritmos de controle — assegurando que o robô funcione de forma coordenada e precisa em situações semelhantes às de uma competição real.

Figura 60 – Configuração do labirinto de testes para testar algoritmo de controle

Os testes de integração foram executados em um labirinto reduzido, com 16 células, especialmente projetado para simular os desafios encontrados em competições. Nessa fase,

testou-se a aplicação prática das estratégias de controle, com destaque para o uso de controladores PID, fundamentais para o ajuste fino de velocidade e posicionamento do robô.

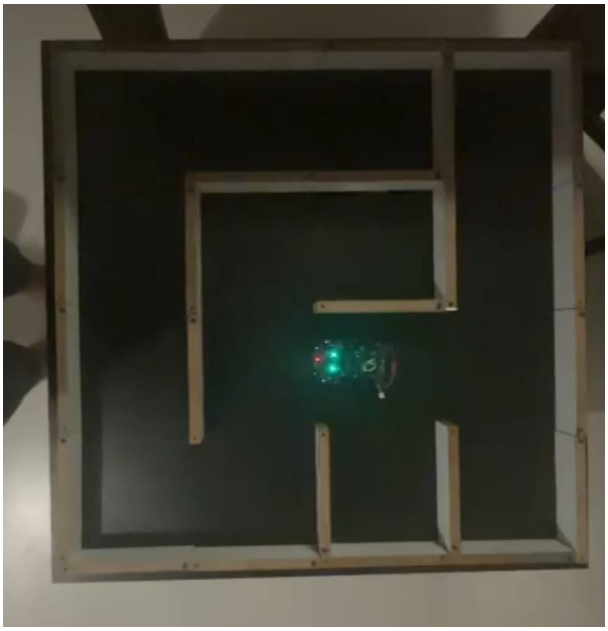


Figura 61 – Configuração do labirinto para testes de integração completos

Além disso, avaliou-se o desempenho do algoritmo de resolução de labirinto conhecido como *flood fill*, o qual permite que o robô explore e mapeie o ambiente de maneira eficiente, identificando o caminho ideal até o centro do labirinto. Esses testes de integração são cruciais para garantir que todos os subsistemas operem harmoniosamente e que o micro mouse esteja apto a executar trajetórias complexas com precisão e agilidade.

Durante o processo de testes, foram identificadas algumas inconsistências mecânicas que afetaram diretamente o desempenho do controle PID. Um dos principais problemas enfrentados foi a presença de folgas nas engrenagens do sistema de tração, que resultaram em pequenas imprecisões nos movimentos do robô. Essas folgas geravam atrasos e variações inesperadas na resposta dos motores, prejudicando a precisão dos ajustes de posição e velocidade esperados pelo controlador PID.

Outro fator que impactou o controle foi a escolha de uma baixa taxa de redução no início do projeto. Com uma taxa de redução menor, a transmissão de torque para as rodas foi limitada, o que dificultou a realização de correções finas de velocidade e posição, especialmente em situações que exigiam mudanças rápidas na direção ou desaceleração precisa em curvas. Esse aspecto reduziu a capacidade do robô de se ajustar rapidamente às mudanças no trajeto, o que foi particularmente desafiador ao realizar manobras em alta velocidade no labirinto.

Apesar desses problemas, os ajustes e calibrações adicionais permitiram que o

projeto mantivesse funcionalidade aceitável. Embora o desempenho ideal não tenha sido alcançado, especialmente em relação à precisão dos movimentos e resposta em tempo real, o micro-mouse foi capaz de executar suas tarefas principais.

5 Considerações finais

5.1 Conclusões

Dado que o desenvolvimento do micro mouse transcorreu conforme planejado, o grupo decidiu levar o projeto para uma competição, visando validar seu desempenho em um ambiente prático. A participação na RoboChallenge 2024, realizada no Instituto Mauá de Tecnologia, ofereceu uma oportunidade única de avaliar o robô em condições reais, com as exigências e imprevistos típicos de uma competição.

Figura 62 – Labirinto oficial da competição

A competição, que durou quatro dias, marcou a primeira edição da categoria de micro mouse no Brasil, e o grupo obteve o terceiro lugar. Durante o evento, alguns desafios foram enfrentados, especialmente devido à iluminação solar no local, que causava interferência nos sensores de infravermelho do robô. Esse problema foi contornado por meio de ajustes na amostragem dos sensores, permitindo que o micro mouse operasse propriamente.

Além disso, as limitações do sistema mecânico, como a folga nas engrenagens e a baixa taxa de redução, afetaram o desempenho do robô e limitaram sua capacidade de alcançar o primeiro lugar. Esses fatores influenciaram a precisão nas manobras e a resposta em alta velocidade, especialmente nas curvas e mudanças rápidas de direção no labirinto.

Apesar desses obstáculos, o desempenho geral foi considerado satisfatório, atendendo aos requisitos funcionais e não funcionais estabelecidos para o projeto. A experiência adquirida durante a competição gerou insights importantes para melhorias futuras e confirmou a eficácia das soluções implementadas ao longo do desenvolvimento, validando o projeto em um contexto prático e real.

5.2 Perspectivas de continuidade

O projeto apresenta um vasto potencial de continuidade, com diversas oportunidades de aprimoramento nos âmbitos de computação, elétrica e mecânica. As perspectivas descritas a seguir visam explorar essas possibilidades, com o objetivo de elevar ainda mais o desempenho, a eficiência e a praticidade do robô.

5.2.1 Computação

No âmbito da computação, há diversas melhorias planejadas para reduzir o tempo de execução do robô e otimizar seu desempenho. Entre as propostas está a revisão dos algoritmos responsáveis pelo cálculo do costmap e da rota de retorno, que podem ser refinados para acelerar o processamento. A adoção de periféricos como FMAC, voltado para a execução de filtros, e CORDIC, para operações matemáticas, promete descarregar tarefas do microcontrolador principal, deixando-o livre para operações mais críticas. Além disso, a substituição do microcontrolador atual pelo STM32H725, equipado com o núcleo Cortex-M7 e um clock significativamente maior, trará uma capacidade de processamento mais robusta, essencial para cálculos em tempo real e execução simultânea de tarefas mais complexas.

Em termos de performance, o robô poderá mapear continuamente o ambiente, eliminando a necessidade de paradas em cada célula, o que deve melhorar a fluidez de sua movimentação. Além disso, o robô será capaz de seguir paredes ao andar em trajetórias diagonais, aproveitando essas rotas para reduzir o tempo de exploração. Outra melhoria está na inclusão dos movimentos diagonais no cálculo dos custos no costmap, permitindo trajetórias mais otimizadas e realistas durante a navegação. O processo de desenvolvimento também será beneficiado com um monitor de código integrado no computador, onde gráficos, variáveis e a execução do programa poderão ser acompanhados em tempo real. Essa funcionalidade, junto com a implementação de comunicação via Bluetooth, facilitará os testes e o debugging do sistema.

5.2.2 Elétrica

Na parte elétrica, algumas correções importantes serão realizadas para garantir maior confiabilidade. Uma delas é a substituição do driver da ventoinha por um modelo que suporte correntes maiores, eliminando o risco de sobrecarga. O circuito do sensor de corrente também será ajustado para utilizar uma ligação de Kelvin, proporcionando medições mais precisas e consistentes. Melhorias no design elétrico facilitarão o uso do robô, como a inclusão de um switch para desligamento, que evitará a necessidade de remover a bateria manualmente. O conector da bateria será soldado diretamente na placa, melhorando a robustez e o aspecto visual do conjunto. Além disso, a substituição

do conector de gravação atual por um USB-C trará mais praticidade e confiabilidade, e a migração para um módulo BLE permitirá interfaces mais modernas, alinhadas às tecnologias de comunicação atuais.

5.2.3 Mecânica

No campo da mecânica, o foco será na consistência estrutural e na otimização do design. A troca das engrenagens impressas por modelos comerciais, que oferecem maior precisão, contribuirá para um sistema de movimentação mais confiável. Para aumentar o torque e possibilitar deslocamentos em baixas velocidades, será aplicada uma maior redução na transmissão do motor. A separação entre a estrutura mecânica e a placa de circuito permitirá maior flexibilidade no desenvolvimento, possibilitando mudanças independentes em ambas as partes. Com isso, a placa poderá ter componentes montados em ambos os lados e não precisará mais atuar como elemento estrutural, o que permitirá a fabricação de versões mais finas e leves. Mancais passantes e outros sistemas de fixação mais robustos também serão adotados, aumentando a estabilidade do robô. Por fim, o redesenho da distribuição de massa, visando aproximar os componentes do centro de rotação, tornará o movimento mais equilibrado e ágil, especialmente em curvas e deslocamentos rápidos.

Referências

BARBOSA, G. C.; SANTI, P. H. M.; PEREIRA, M. R. *Micras Simulation - NTF Classic Micromouse project with a STM32 microcontroller*. 2024. GitHub Repository. Disponível em: <https://github.com/Team-Micras/MicrasFirmware>. Acesso em: 3 Dec 2024. Citado na página 28.

BARBOSA, G. C.; SANTI, P. H. M.; PEREIRA, M. R. *Micras Simulation - NTF Classic Micromouse simulation environment*. 2024. GitHub Repository. Disponível em: <https://github.com/Team-Micras/micras_simulation>. Acesso em: 3 Dec 2024. Citado na página 28.

DOUGLASS, B. *Design Patterns for Embedded Systems in C: An Embedded Software Engineering Toolkit*. Newnes/Elsevier, 2011. ISBN 9781856177078. Disponível em: <https://books.google.com.br/books?id=8cgNlQEACAAJ>. Citado na página 47.

FERREIRA, J. A. M. L.; FERREIRA, J. A. P.; TAVARES, J. M. R. S. *Dynamic Modeling and Power Modeling of Robotic Skid-Steered Wheeled Vehicles*. 2010. ResearchGate. Disponível em: <https://www.researchgate.net/publication/221918595_Dynamic_Modeling_and_Power_Modeling_of_Robotic_Skid-Steered_Wheeled_Vehicles>. Acesso em: 10 Dec 2024. Citado na página 26.

FUNADA, K. *All Japan Micromouse Competition 2023 - Carmine Tech Specs*. 2023. Website. Disponível em: <https://www.ntf.or.jp/mouse/micromouse2023/Robot/AllJapan/TechCA34.html>. Acesso em: 3 dec 2024. Citado na página 14.

IOI, K. A mobile micro-robot using centrifugal forces. In: *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (Cat. No.99TH8399)*. [S.l.: s.n.], 1999. p. 736–741. Citado na página 67.

KEVIN, R. *All Japan Micromouse Competition 2022 - Trident Tech Specs*. 2022. Website. Disponível em: <https://www.ntf.or.jp/mouse/micromouse2022/AllJapan/robots/TechCA01.html>. Acesso em: 3 dec 2024. Citado na página 14.

NTF. *All Japan Micromouse Competition 2023*. 2023. Website. Disponível em: <https://www.ntf.or.jp/mouse/micromouse2023/Robot/AllJapan/index.html>. Acesso em: 20 nov 2024. Citado 2 vezes nas páginas 25 e 70.

NTF. *Micromouse Contest Rules*. 2024. Nihon Technology Foundation. Disponível em: <https://www.ntf.or.jp/?page_id=933>. Acesso em: 10 Dec 2024. Citado na página 17.

OSRAM. *SFH4550 - Infrared Emitting Diode*. [S.l.], 2024. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/view/218330/OSRAM/SFH4550.html>. Acesso em: 3 Dec 2024. Citado na página 72.

OSRAM, A. *AS5047U Magnetic Position Sensor Datasheet*. [S.l.], 2024. Disponível em: <https://look.ams-osram.com/m/48d90c982e0879e8/original/AS5047U-DS000637.pdf>. Acesso em: 3 Dec 2024. Citado na página 56.

PETER. *Four Wheel Micromouse: Difficult to Turn*. 2024. Micromouse Online. Disponível em: <https://micromouseonline.com/2013/02/24/four-wheel-micromouse-difficult-to-turn/>. Acesso em: 10 Dec 2024. Citado na página 26.

RLS. *MicroMouse Case Study*. 2019. PDF document. Disponível em: <https://www.rls.si/media/wysiwyg/case-studies/MicroMouse.pdf>. Acesso em: 2 Dec 2024. Citado na página 71.

RT, C. *Advancements in Robotics and Autonomous Vehicles*. [S.l.], 2020. Disponível em: <https://rt-net.jp/mobility/archives/14559>. Acesso em: 3 Dec 2024. Citado na página 72.

SEMI, W. *WS2812B - Intelligent Control LED Integrated Light Source*. 2020. Mouser Electronics. Disponível em: <https://www.mouser.com/pdfDocs/WS2812B-2020_V10_EN_181106150240761.pdf>. Acesso em: 10 Dec 2024. Citado na página 53.

SPEN, C. *Gears - A Collection of Useful Tools for Game Development*. 2020. GitHub Repository. Disponível em: <https://github.com/chrisspen/gears/tree/master>. Acesso em: 10 Dec 2024. Citado na página 39.

STMICROELECTRONICS. *L6205 - DMOS Dual Full Bridge for Motor Control Applications*. [S.l.], 2024. Disponível em: <https://www.st.com/en/motor-drivers/l6205.html>. Acesso em: 3 Dec 2024. Citado 2 vezes nas páginas 51 e 66.

STMICROELECTRONICS. *LSM6DSV - 6-axis Inertial Measurement Unit (IMU) Datasheet*. [S.l.], 2024. Disponível em: <https://www.st.com/resource/en/datasheet/lsm6dsv.pdf>. Acesso em: 3 Dec 2024. Citado na página 71.

STMICROELECTRONICS. *STSPIN250 - Single Brush DC Motor Driver*. [S.l.], 2024. Disponível em: <https://www.st.com/en/motor-drivers/stspin250.html>. Acesso em: 3 Dec 2024. Citado 2 vezes nas páginas 51 e 65.

TI. *TPS61178 - Boost Converter*. [S.l.], 2024. Disponível em: <https://www.ti.com/product/TPS61178>. Acesso em: 10 Dec 2024. Citado na página 64.

TONDRA, D. *A Detailed Design and Analysis of Micromouse Tondra*. 2004. University of Nevada, Las Vegas. Disponível em: <https://www.physics.unlv.edu/~bill/ecg497/Drew_Tondra_report.pdf>. Acesso em: 10 Dec 2024. Citado na página 26.

TOSHIBA. *TPS601A - High Efficiency Step-Up DC-DC Converter Datasheet*. [S.l.], 2024. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/view/115616/TOSHIBA/TPS601A.html>. Acesso em: 3 Dec 2024. Citado na página 72.