UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Otávio Almeida Leite

# Sistema de Navegação para um Robô Móvel Omnidirecional em Competições de Robótica

**Uberlândia, Brasil**

**2025**

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Otávio Almeida Leite

# Sistema de Navegação para um Robô Móvel Omnidirecional em Competições de Robótica

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Jefferson Rodrigo de Souza

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

Otávio Almeida Leite

# Sistema de Navegação para um Robô Móvel Omnidirecional em Competições de Robótica

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 05 de maio de 2025:

---

**Prof. Dr. Jefferson Rodrigo de Souza**
Orientador

---

**Prof. Dr. Leandro Nogueira Couto**

---

**Prof. Dr. Paulo Henrique Ribeiro Gabriel**

Uberlândia, Brasil
2025

*A todas as pessoas que me trouxeram a robótica,*
*e a todas aquelas que a robótica trouxe a mim.*

# Agradecimentos

Agradeço a meus pais, Délcio e Rita, a meu irmão Hugo e a toda minha família pelo apoio incondicional durante toda a minha jornada acadêmica.

Também a todos os amigos que me acompanharam nesse caminho, seja de perto ou de longe. Em especial aos colegas que pude conhecer e compartilhar grandes momentos dentro da Faculdade de Computação da UFU.

Sou imensamente grato também aos membros, ex-membros e amigos da Equipe Roboforge de Robótica, da qual tive o enorme privilégio de fazer parte durante toda a minha graduação. Da mesma forma, a todas as amizades que pude construir nas competições de robótica, sejam juízes, competidores ou colegas.

Não poderia deixar de mencionar os professores, aos quais agradeço o imenso suporte, dedicação e incentivo, que com certeza foram fundamentais para a concretização deste trabalho. Em especial ao Prof. Carlos Roberto Lopes e ao Prof. Diego Nunes Molinos, pelo apoio através da Equipe Roboforge, ao Prof. Jefferson Rodrigo de Souza pela orientação neste trabalho, e a todos os professores da Faculdade de Computação e da Universidade Federal de Uberlândia com os quais tive contato e pude construir tanto aprendizado.

Agradeço especialmente à Professora Suselaine da Fonseca Silva, responsável pelo meu primeiro contato com robótica e pelo apoio em todos os campeonatos e experiências anteriores à graduação que me fizeram me apaixonar por essa área.

Por fim, a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

*"We are participating in a competition.*
*We like to win. We want to learn.*
*And we also want to have fun.*
*(...)*

*Sometimes we fail and that is OK.*
*Original ideas come from failing.*
*Winning is nice but failing is part of our journey."*

- The World Robot Olympiad Ethics Code for Teams

# Resumo

Este trabalho apresenta o desenvolvimento, avaliação e documentação de alternativas de software para dotar um robô móvel omnidirecional, construído com peças do kit LEGO *Mindstorms* EV3, de autonomia e eficiência na realização das tarefas propostas pela RoboCup Brasil *Challenge* 2024, modalidade *OPEN*. Desta maneira, foi adotada uma arquitetura distribuída em dois controladores EV3 e implementados subsistemas de movimentação omnidirecional baseados em rodas *mecanum* e controle PID, algoritmos de localização odométrica com estratégias de correção e alinhamento, representação interna em grafos para planejamento de rotas via algoritmo de Dijkstra e um algoritmo para controle de caminho. A percepção do ambiente se deu com sensores de cor, ultrassônicos e infravermelhos, adicionando estratégias de processamento dos sinais, como a classificação de cores utilizando técnicas de aprendizado de máquina. Os resultados demonstraram que o sistema autônomo alcançou resultados satisfatórios e competitivos, obtendo o 4º lugar na modalidade OPEN, apesar de variações e instabilidades em cenários mais complexos.

**Palavras-chave**: robótica, localização, navegação, robô autônomo e competições.

# Lista de ilustrações

# Lista de abreviaturas e siglas

| | |
|---|---|
| ARM | *Advanced RISC Machine* |
| CBR | Competição Brasileira de Robótica |
| cm | Centímetros |
| IA | Inteligência Artificial |
| IEEE | *Institute of Electrical and Electronics Engineers* |
| kNN | *K-Nearest Neighbours* |
| LARC | *Latin American Robotics Competition* |
| LED | *Light Emitting Diode* |
| MB | Megabytes |
| MHz | Megahertz |
| Ncm | Newton-centímetro |
| PID | Proporcional, Integral e Derivativo |
| RAM | *Random Access Memory* |
| RCB | RoboCup Brasil |
| RGB | *Red, green and blue* |
| RGB-D | *Red, green, blue and depth* |
| RJ12 | *Registered Jack 12* |
| ROS | *Robot Operating System* |
| RPM, rpm | Rotações por minuto |
| SLAM | *Simultaneous Localization and Mapping* |
| STEM | *Science, Technology, Engineering and Math* |
| USB | *Universal Serial Bus* |
| Wi-Fi | *Wireless Fidelity* |

# Sumário

# 1 Introdução

## 1.1 Contextualização

### 1.1.1 Robótica Móvel

A robótica é o campo interdisciplinar de estudo que se dedica aos conceitos relacionados ao desenvolvimento e à construção de dispositivos conhecidos como robôs. Segundo Mataric (2007): "um robô é um sistema **autônomo** que existe no **mundo físico**, consegue **perceber** os seus arredores (seu ambiente), e consegue **agir** nele a fim de alcançar alguns **objetivos**". Para a autora, cada uma das partes dessa definição é fundamental:

- Com relação ao aspecto de autonomia, se trata de um sistema que toma as próprias decisões, e não é controlado ou operado por um humano;
- Existir no mundo físico é uma propriedade fundamental de robôs, afinal impõe um conjunto de limitações e desafios com os quais tais sistemas precisam lidar, numa complexidade irreplicável através de softwares (simulações, por exemplo);
- Perceber seu ambiente, diz sobre como se faz necessário que os robôs tenham as formas de conseguir informações sobre onde estão inseridos, seja sob quaisquer meios, mas de forma que estas não sejam dadas por terceiros, e sim por eles;
- Agir em seu ambiente no sentido que, para esta definição, uma máquina que não transforma o ambiente para além de sua própria constituição não é um robô;
- Por último, com relação aos objetivos, um robô age segundo um propósito, e toma decisões de forma inteligente. Esses objetivos podem ser tanto simples e triviais, quanto complexos e exigirem mecanismos de deliberação muito elaborados.

Com a evolução significativa das tecnologias envolvidas no processo de pesquisa e desenvolvimento de robôs nas últimas décadas (ROMERO et al., 2014), estes têm se tornado cada vez mais presentes em diversas áreas da sociedade. Entre os avanços que possibilitam essa evolução, pode-se citar a miniaturização de componentes de hardware, ao mesmo tempo que o aumento na capacidade de processamento dos mesmos, o desenvolvimento de atuadores mais robustos e precisos, melhorias na eficiência energética desses componentes em geral, sensores e câmeras com tecnologias mais robustas, mais baratas e que ocupam menos espaço, etc. Tudo isso reflete nos avanços de software, onde os algoritmos são constantemente desenvolvidos e otimizados para utilizar o máximo possível dos recursos disponíveis, e aportar as novas quantidades de dados.

Em diversas áreas da sociedade, robôs já se fazem presentes, ritmo que tende a ficar ainda mais acelerado nos próximos anos. De acordo com Murphy (2019), robôs são destinados a tarefas "sujas, maçantes ou perigosas" (do inglês "*dirty, dull and dangerous*"). Robôs trabalhando com humanos em atividades como inspeção de tubulações de esgoto, gasodutos, poços marítimos, serviços de limpeza, transporte de material, logística interna em armazéns e fábricas, plantações e colheitas automatizadas, monitoramento de áreas e/ou prédios, operações militares e resgate, cirurgias de precisão, e muitas outras, já são uma realidade. Além dessas, o autor destaca aplicações relacionadas a assistência humana: cuidados domésticos, médicos e dia a dia, como robôs aspiradores, outros para apoio e cuidado de idosos, e carros com direção autônoma. No entretenimento, por exemplo, a Disney USA, uma das 10 empresas mais lucrativas do ramo no mundo (REIFF, 2024), tem times de pesquisa e desenvolvimento trabalhando em robôs bípedes autônomos, altamente interativos e com visuais e comportamentos inspirados em franquias de ficção científica da empresa, integrar o elenco de personagens dos seus parques temáticos (GRANDIA et al., 2025).

### 1.1.2 Competições de Robótica

Segundo Siciliano e Khatib (2016), o campo da robótica é particularmente adequado para inovação através de competições, além de ser um daqueles em que estas têm o maior impacto, principalmente em educação e pesquisa. A presença de tais eventos nessa área atende ao objetivo de criar uma estrutura motivadora para o desenvolvimento de ciência e tecnologia, assim como de novas habilidades, ideias e soluções. Competições que envolvem o projeto e implementação de robôs cobrem todo o espectro da cadeia educacional, desde crianças no ensino básico até cientistas e pesquisadores profissionais, o que é um reflexo da sua relevância na perspectiva educacional e motivadora. Além disso, dada a natureza interdisciplinar da robótica, formar uma equipe vencedora envolve uma cooperação substancial entre seus membros, combinando diferentes habilidades e formações técnicas, o que se faz muito interessante pela troca de conhecimentos entre as áreas envolvidas.

Brancalião et al. (2022) pontuam que essas iniciativas têm sido boas ferramentas educacionais no ensino superior, por exemplo, por ajudarem universidades no ensino de uma variedade de tópicos de engenharia, como prototipagem, programação e mecatrônica. Murphy (2001) traz a perspectiva de como competições podem ser aliadas às disciplinas e cursos para maximizar a experiência de aprendizagem e o desenvolvimento intelectual.

Já pelo lado da pesquisa em robótica, Siciliano e Khatib (2016) argumentam que não apenas vantajosas, as competições promovem cenários valiosos por conta da presença de um ou mais agentes interagindo com o ambiente físico. Outras técnicas de compartilhamento e validação de pesquisas comuns em outras áreas, como testes padronizados,

testes com conjuntos de dados compartilhados, implementação de referência, ainda são inestimáveis à robótica, mas não cobrem sozinhas todo o escopo de um sistema robótico operando no mundo real. Como robôs são compostos por sistemas interagindo uns com os outros e o mundo real, os limites dessas técnicas se tornam evidentes, e as competições entram como uma possibilidade para agregar nesses pontos às pesquisas. Ainda segundo Siciliano e Khatib (2016), estas provêm uma grande oportunidade de projetar novos *benchmarks* (métricas, testes padronizados) e técnicas de experimentação científica: avaliação de performance é um problema aberto na robótica, e as competições têm dado contribuições significativas na direção do desenvolvimento de métricas e bancos de testes repetíveis. Os robôs, algoritmos e implementações diferentes serem colocados à prova em testes adversários também adiciona profundidade ao impacto dessas iniciativas à pesquisa.

Num aspecto econômico, campeonatos aparecem no radar de interesse de empresas de tecnologia tanto pela possibilidade de desenvolvimento de ferramentas e produtos de capacidade técnica que vão além do estado da arte, quanto por estratégias de recrutamento, apoio educacional e publicidade (SICILIANO; KHATIB, 2016). Em outra perspectiva, têm um impacto significativo na opinião e entendimento públicos de ciência e tecnologia: competições podem ter forte apelo ao entretenimento e tornar o progresso científico mais acessível, através da linguagem dos esportes. Brancalião et al. (2022) citam entre os grandes benefícios dessas iniciativas para a sociedade como um todo: a promoção de novas soluções e inovação, impulsionamento do estado da arte em diversos campos, *benchmarking*, motivação de estudantes a participar de atividades envolvendo os campos de ciência, tecnologia, engenharia e matemática (STEM), e encorajamento dos mesmos a seguirem carreira nessas áreas.

Além disso, Graffin, Sheffield e Koul (2022) ressaltam como competições de nível básico, especialmente quando aliadas a uma perspectiva educacional, trazem impactos positivos e consistentes na literatura em relação à inclusão de grupos sub-representados, como estudantes do sexo feminino, minorias étnicas e estudantes com necessidades especiais. Segundo os autores, a longo prazo, a participação nessas competições tem efeitos estatisticamente significativos sobre o interesse por áreas STEM, a aprendizagem, e as escolhas relacionadas ao ensino superior e à carreira, especialmente entre alunas.

Quanto às tecnologias envolvidas em artigos relevantes a competições de robótica, Brancalião et al. (2022) citam as principais sendo métodos como soluções baseadas em IA, aprendizado de máquina, visão computacional, localização e mapeamento simultâneos (SLAM), processamento de imagem, reconhecimento de fala (linguagem natural), reconhecimento de objetos, reconhecimento de gestos, filtros estendidos de Kalman, controle em tempo real e impressão 3D. Entre elas aparecem os softwares e simuladores como ROS, "*hardware-in-the-loop*", *SimTwo*, *Gazebo*, *TeamBots*, *USARSim*, *OpenCV*, *TensorFlow*, *LabView* e componentes como sensores, atuadores, câmeras, microcontroladores e

kits LEGO *Mindstorms*.

De acordo com Siciliano e Khatib (2016), as competições podem ser separadas em:

- Aquelas inspiradas em competições humanas, como de esportes, por exemplo.
- As que propõem um problema específico a ser resolvido pelas máquinas, na expectativa que ao tentar alcançar seus objetivos novas capacidades dos robôs sejam desenvolvidas e/ou amadureçam nos campos de pesquisa envolvidos.

Este trabalho de conclusão de curso foca em uma competição do segundo tipo, cujos detalhes serão introduzidos nos próximo capítulo: a RoboCup Brasil *Challenge OPEN* 2024.

## 1.2 Objetivos

Este trabalho tem como objetivo desenvolver, avaliar e documentar alternativas, com foco em software, que permitam a um robô omnidirecional atingir autonomia e eficiência no cenário proposto na competição RoboCup Brasil *Challenge OPEN* 2024, apesar das limitações impostas pelas regras e condições de hardware e desenvolvimento disponíveis. Como objetivos específicos:

- Desenvolver e documentar o software responsável pela resolução autônoma por parte do robô das tarefas propostas pela competição.
- Estudar e implementar no robô a coleta de informações do ambiente de forma eficiente com sensores disponíveis, para diminuir o tempo gasto para resolução das tarefas.
- Estudar e implementar estratégias de navegação autônoma eficiente, apesar das limitações do robô e do desafio.
- Estudar as limitações de hardware e software na plataforma utilizada e possíveis formas de superá-las.

Tudo isso trabalhando com foco na proposição de soluções alternativas viáveis ao contexto de competições de robótica como a RCB *Challenge*.

## 1.3 Organização do Trabalho

A organização deste trabalho se dá em 5 capítulos: Introdução, Referencial Teórico, Metodologia, Resultados e Conclusão. Os capítulos estão organizados da seguinte forma:

- Referencial Teórico: apresenta conceitos teóricos necessários para o desenvolvimento do trabalho. O foco será na competição e na perspectiva de técnicas de software aplicáveis nas diversas esferas que envolvem o desenvolvimento de um robô móvel autônomo com as rodas omnidirecionais, como localização, navegação, planejamento de rota, etc.

- Metodologia: apresenta o plano de desenvolvimento do trabalho de pesquisa, então descrevendo os aspectos técnicos do desenvolvimento do robô, quais estratégias foram implementadas e testadas.

- Resultados: descreve o desempenho obtido nos testes realizados a partir da implementação das estratégias no robô real, assim como os resultados atingidos.

- Conclusão: apresenta as considerações finais, retoma em discussão o que foi desenvolvido neste trabalho, e direciona os pontos de extensão para trabalhos futuros.

# 2 Referencial Teórico

## 2.1 Fundamentação Teórica

### 2.1.1 Competição CBR *Challenge*

A Competição Brasileira de Robótica (CBR) é um evento anual que reúne estudantes e pesquisadores de diversas idades para participar de desafios de robótica (CBR..., 2025), uma das tradicionais e antigas competições de robótica do Brasil (AROCA et al., 2019). São várias categorias, desde as mais introdutórias, que contam inclusive com a participação de alunos de Ensino Fundamental, Médio ou Técnico, até as mais complexas, que envolvem aplicações de robôs autônomos e se aproximam do estado da arte de robótica, com o envolvimento de pesquisadores, alunos e professores de universidades e institutos de pesquisa (CATEGORIAS..., 2025). Segundo seu site oficial, trata-se da maior competição de robótica e inteligência artificial do Brasil. A CBR é patrocinada por empresas - como a Petrobras - e organizada pela RoboCup Brasil, a frente nacional da *RoboCup Federation*, com o objetivo de promover o desenvolvimento de robôs inteligentes, IA e automação (ROBOCUP..., 2025).

A RoboCup se destaca como iniciativa com significativo impacto na educação e pesquisa em robótica mundialmente, com abordagens e soluções desenvolvidas em várias áreas, como a robótica humanoide, sistemas multiagentes e multirrobôs, e métricas de performance (SICILIANO; KHATIB, 2016). Um exemplo é o robô de busca e resgate *Quince*, que atuou nas operações após o acidente nuclear de 2011 em Fukushima, no Japão, e foi desenvolvido na categoria *RoboCup Rescue*, onde demonstrou eficiente mobilidade em comparação a seus concorrentes. Segundo Brancalião et al. (2022) trata-se da maior das competições, e uma das mais famosas. Nas categorias da competição são definidos diferentes problemas a serem resolvidos por robôs de forma autônoma. O foco é robôs que atuem em cenários físicos, sejam estes realistas ou inspirados em situações reais.

Uma das categorias presentes é a RCB *Challenge* (RoboCup Brasil *Challenge*), exclusiva da iniciativa brasileira e voltada para proporcionar desafios para estudantes universitários e do Ensino Médio (CATEGORIAS..., 2025). A *Challenge*, anteriormente parte da IEEE LARC e nomeada IEEE *Standard Educational Kit*, tem em seu histórico a proposta de desafios para serem resolvidos por robôs pequenos (em geral, de até 30 centímetros de altura), construídos com peças de kits educacionais de robótica. Até a edição de 2023, os robôs competidores precisavam ser construídos com peças de um único fabricante. Na edição de 2024, a categoria foi dividida em duas modalidades: a *KIT*, que seguia o formato tradicional, e a *OPEN*, que retirava as limitações de material e/ou

fabricante, desde que os robôs construídos não utilizassem câmeras ou nenhum sistema de visão computacional. A partir de 2025, as limitações de material não estarão presentes na categoria, e ela segue da forma que foi executada na modalidade *OPEN* em 2024.

Os desafios da categoria *Challenge* se dispõem em um mapa em que os robôs precisam atingir alguns objetivos de forma autônoma. Os objetivos rendem pontos para as equipes em tomadas de tempo fixo, chamadas de partidas. Dentro das partidas, falhas e reinícios diminuem o saldo de pontos das equipes. A cada dois anos o tema central do desafio muda, a fim de representar aplicações reais da robótica na sociedade. Em 2019 e 2022 (intervalo de tamanho excepcional por decorrência da pandemia de COVID-19), por exemplo, o mapa da competição simulava um gasoduto que passava no fundo do leito de um rio, representado por uma área baixa e coloração azul no mapa da competição (PARANAíBA, 2022). As equipes precisavam construir e programar seus robôs para inspecionar o gasoduto, encontrar pontos que precisassem de reparo e colocar as peças de reposição certas nos lugares necessários, conforme pode ser visto nas Figuras 1 e 2.

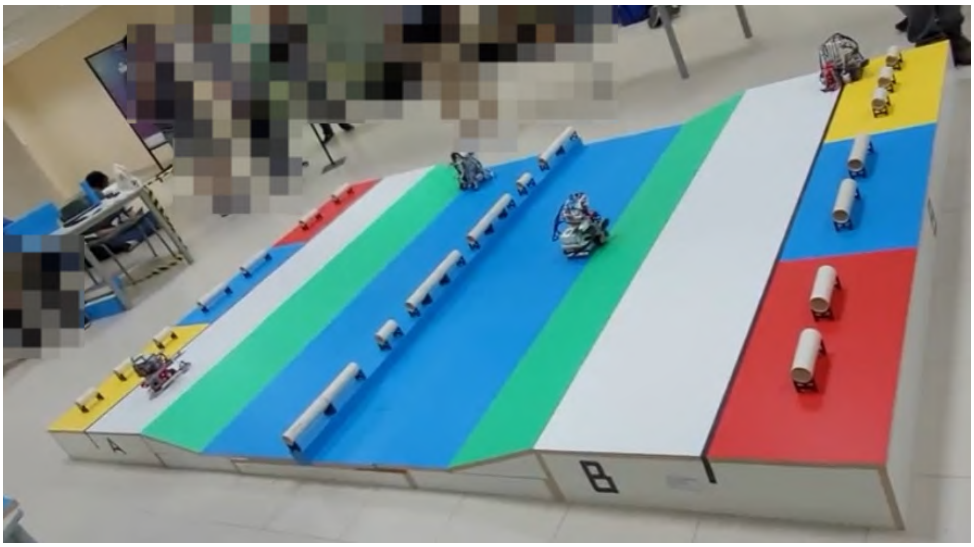


Figura 1 – Mapa da competição em 2022, durante uma partida classificatória. Fonte: Arquivo pessoal.

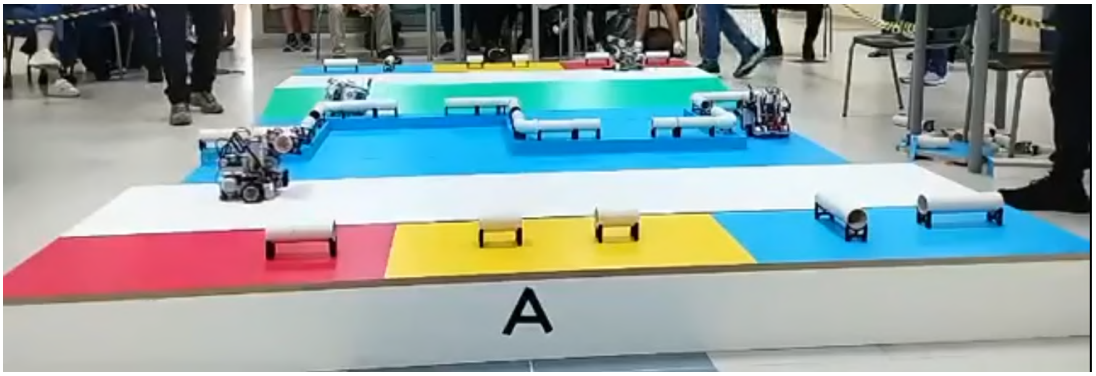


Figura 2 – Mapa da competição em 2022, durante a partida final. Fonte: Arquivo pessoal.

A categoria se organiza como competição de enfrentamento, onde em cada partida uma equipe joga contra outra. A equipe cujo robô fizer mais pontos dentro da partida é a vencedora. A depender do número de equipes inscritas, podem existir as fases classificatórias, de grupos e de enfrentamento em chaves. Nesta última, a cada enfrentamento uma equipe é eliminada da competição, e as equipes vencedoras seguem se enfrentando até uma partida final.

### 2.1.2 Regras da RCB Challenge 2024

Em 2023 e 2024, o desafio da categoria *Challenge* se inspira em cidades inteligentes para propor que as equipes desenvolvam seus veículos autônomos (AZEVEDO, 2024). O mapa simula uma cidade, com vários pontos de interesse até os quais os passageiros podem solicitar transporte aos robôs (Figura 3). A área branca do mapa delimita a região que os robôs podem se locomover e as áreas coloridas os estabelecimentos. O mapa é simétrico e, durante uma partida, que dura no máximo 12 min, cada equipe posiciona seu robô em um dos lados da cidade. A área central, marcada em azul e que contém uma plataforma elevada, trata-se da zona de embarque de passageiros. Ela é a única área compartilhada entre as equipes, ou seja, os robôs de cada lado do mapa competem pelos mesmos passageiros (Figura 4).

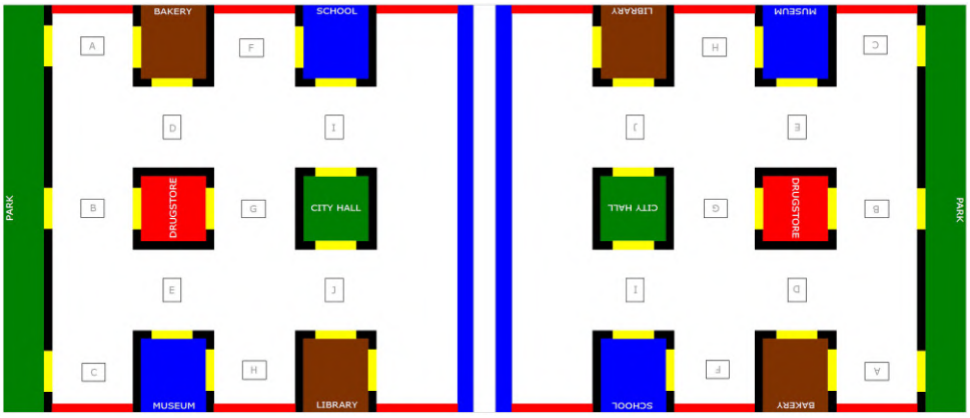


Figura 3 – Vista superior da arena da RCB *Challenge* 2024. Fonte: Extraído de (AZEVEDO, 2024).

Os passageiros são representados por tubos coloridos, dispostos em pé na zona de embarque. As cores e os tamanhos dos tubos representam suas solicitações de transporte. Por exemplo: o tubo verde de 15cm de altura representa um adulto que quer ir à prefeitura, enquanto o tubo verde de 10cm representa uma criança que quer ir ao parque. Há ainda um tubo branco, sem cor nenhuma, que representa uma pessoa que não fez nenhuma solicitação de transporte. Esse tubo não deve ser retirado da zona de embarque. No início da partida os robôs iniciam em qualquer posição da área branca, em qualquer orientação. As posições iniciais são equivalentes para as duas equipes jogando e são determinadas

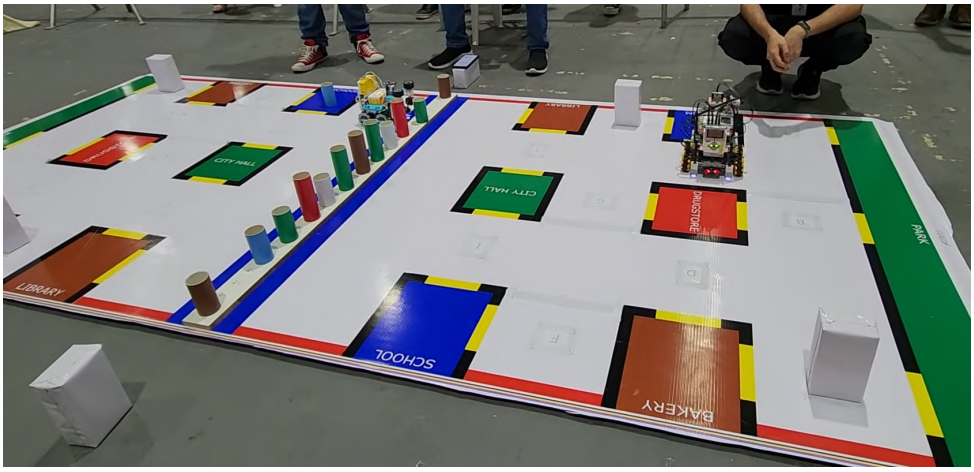


Figura 4 – Mapa da competição em 2024, durante uma partida. Fonte: Arquivo pessoal.

pelos juízes no momento da partida. A ideia é que os robôs não recebam nenhum tipo de informação sobre onde estão na cidade e consigam se localizar de forma autônoma.

Durante a movimentação na cidade, os carros podem encontrar obstáculos. Eles podem ser tanto lombadas nas vias quanto obstruções, que impedem a passagem pela via em que se encontram. O carro deve ser capaz de recalcular a rota de transporte do passageiro e ainda levá-lo ao local correto. Estes obstáculos têm posições específicas no mapa em que podem ser colocados (e posteriormente retirados), o que pode acontecer em qualquer momento, antes ou durante a partida. Os robôs não podem adentrar as áreas dos estabelecimentos, delimitadas pelas linhas pretas, ou sair do mapa, delimitado pelas linhas vermelhas. Apenas os sensores usados para detecção das cores podem adentrar essas linhas. A exceção é durante a entrega do passageiro no estabelecimento correto. Nesse caso, desde que o tubo passe com sua projeção inteira por cima da faixa amarela da linha preta, o robô pode entrar no estabelecimento e realizar o desembarque do passageiro.

Caso o robô deixe o tubo cair, desembarque o passageiro em qualquer outro lugar que não seja o destino correto ou a zona de embarque, ou aconteça um posicionamento indevido, aquele passageiro é considerado como perdido. Assim como os passageiros começam posicionados em pé na zona de embarque, eles devem ser entregues em pé nos estabelecimentos. Nos casos de tubo perdido, o juiz da partida é responsável por recolocá-los de volta nas suas posições iniciais, na zona de embarque, e eles voltam a estar disponíveis para os carros.

Os robôs construídos precisam respeitar a limitação de 25cm máximos de altura, largura e comprimento, mesmo com seus mecanismos (garras, braços, atuadores, etc) completamente desenvolvidos. Não há limite de peças, sensores ou controladores. A única restrição é com relação a câmeras e similares: nesta categoria não é permitido nenhum tipo de processamento de imagem. Para mais detalhes, as regras da competição estão inclusas nos anexos deste trabalho.

### 2.1.3 Robôs móveis sobre rodas

Um dos aspectos fundamentais no desenvolvimento de robôs é entender como a ativação e uso dos seus atuadores afeta o movimento do robô como um todo (sua posição, orientação), na execução de uma tarefa no mundo real. No contexto mais particular de robôs móveis sobre rodas, significa entender como as velocidades aplicadas nos motores afetam a velocidade final do robô e como o torque aplicado afeta sua aceleração. Esses aspectos são a modelagem cinemática (focada na "geometria do movimento") e a modelagem dinâmica (focada nas "forças do movimento") do sistema robótico (LYNCH; PARK, 2017).

Segundo Lynch e Park (2017) robôs móveis sobre rodas podem ser classificados em duas categorias: omnidirecionais e não holonômicos. A diferença entre as duas pode ser explicitada através da análise cinemática dos mesmos, ou seja, da geometria dos seus movimentos. Os robôs não holonômicos são aqueles sujeitos a restrições de velocidade no espaço em que se encontram: robôs semelhantes a carros, por exemplo, não conseguem se mover lateralmente devido às suas configurações mecânicas. Portanto, esses robôs não conseguem se mover em qualquer direção. Os omnidirecionais não têm tais restrições, e isso se dá por conta do tipo de roda e da disposição dos atuadores utilizados neles.

Robôs não holonômicos utilizam rodas convencionais, que giram em torno de um eixo perpendicular ao plano da roda, no centro dela. Nesse caso, eles podem ser dirigidos para direções diferentes rotacionando as rodas em torno de um eixo perpendicular ao chão. Existem também os robôs que atingem esse direcionamento por conta de seu movimento ser baseado em duas rodas convencionais, cada uma com um motor dedicado. Esses são chamados robôs de acionamento diferencial ("*differential drive*"), e conseguem variar suas direções mudando a taxa de rotação relativa entre as rodas (SICILIANO; KHATIB, 2016).

Os robôs omnidirecionais usam rodas especiais, com estruturas mecânicas que contêm rolamentos para permitir o movimento em múltiplas direções, garantindo os três graus de liberdade para movimentação em um plano (SICILIANO; KHATIB, 2016). Esses rolamentos são de rolagem passiva livre e dispostos na circunferência externa da roda. A velocidade nas direções primárias (frente e ré) é controlada pelo atuador da roda, enquanto a velocidade lateral é determinada pela composição da atuação das outras rodas.

Existem as rodas cujos rolamentos giram na direção perpendicular com relação ao eixo de rotação da roda, e aquelas cujos rolamentos estão dispostos em um determinado ângulo diagonal, seus eixos não estando contidos ao plano das rodas. Essas últimas são comumente referidas como rodas "*mecanum*" (LYNCH; PARK, 2017), e exigem uma composição em 4 rodas, enquanto as citadas anteriormente exigem composição de apenas 3. A Figura 5 ilustra o funcionamento da movimentação de robôs seguindo essas duas ideias.

No caso do robô com rodas *mecanum*, as combinações de movimentos das quatro

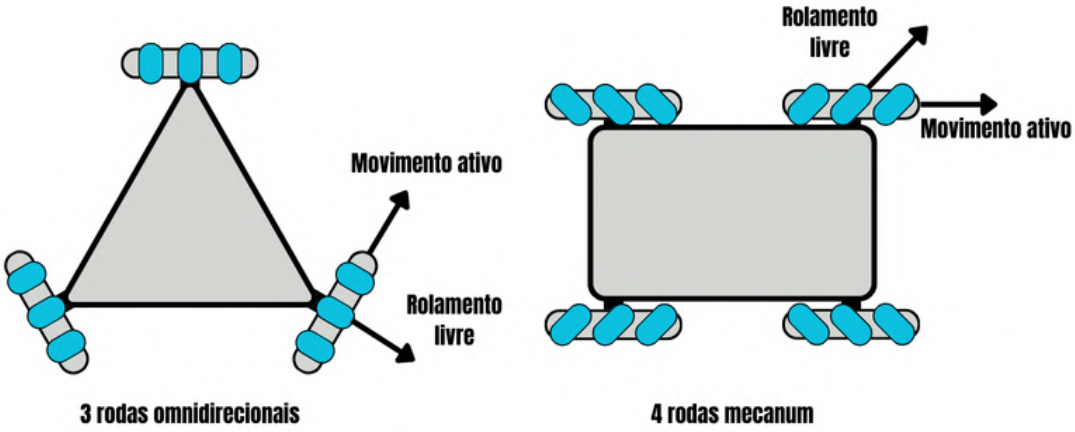


Figura 5 – Princípio de funcionamento mecânico dos tipos de rodas omnidirecionais. Fonte: Adaptado de (LYNCH; PARK, 2017).

rodas podem gerar movimentos mais complexos, representadas através de vetores, como na Figura 6.

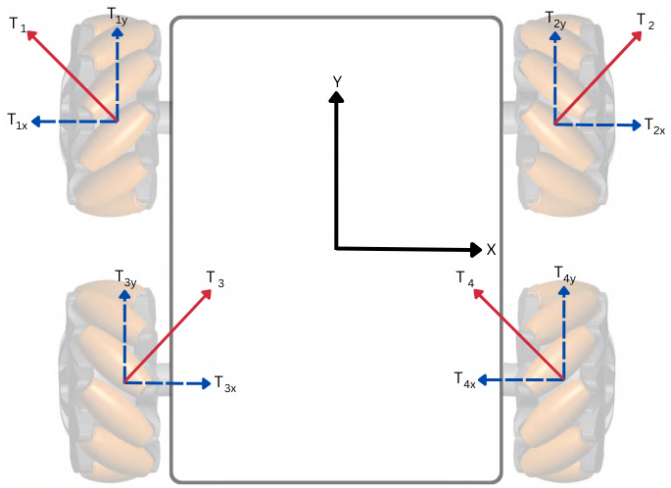


Figura 6 – Esquema de um robô com rodas omnidirecionais indicando os vetores de força em cada roda. Fonte: Adaptado de (SALIH; RIZON; YAACOB, 2006).

Nomeando os motores como 1, 2, 3 e 4 sendo, respectivamente, os motores da frente-esquerda, frente-direita, trás-esquerda e trás-direita, alguns movimentos podem ser programados da seguinte forma:

- Movimento frontal: é obtido com todas as rodas girando no mesmo sentido, com o mesmo módulo de velocidade. O sinal da velocidade determina se o robô se moverá para frente ou para trás.
- Movimento lateral: motores 1 e 4 no mesmo sentido, motores 2 e 3 no sentido oposto, todos com a mesma velocidade. Ou seja, motores diametralmente opostos giram com a mesma velocidade para o mesmo lado. O vetor resultante dos motores promove

o movimento lateral do robô. Dependendo do sinal da velocidade, o robô se move para a direita ou para a esquerda.

- Movimento diagonal: os motores de determinada diagonal se mantém parados, enquanto os motores da outra diagonal se movem no mesmo sentido. O sinal da velocidade aplicada determina em qual sentido sobre a diagonal determinada o robô se move. Por exemplo: quando os motores 1 e 4 permanecem imóveis enquanto 2 e 3 se movem para frente, o robô se move na diagonal frente-esquerda.

- Curva em torno do próprio eixo: de forma análoga a um robô com rodas convencionais, as rodas de um lado do robô se movimentam em um sentido, e as do outro lado no sentido oposto. A troca de sentidos provoca uma curva para o lado contrário.

Além desses movimentos citados, outros podem ser alcançados através de diferentes técnicas de controle e cinemática.

### 2.1.4 Navegação

Os principais desafios na locomoção de robôs envolvem a capacidade de atingir um ponto-alvo, e de seguir um caminho ou trajetória, problemas que podem aparecer isolados ou combinados em diferentes contextos (MATARIC, 2007). A subárea de planejamento de trajetória, ou de movimento, dedica-se à geração desses trajetos para robôs móveis ou manipuladores, com a complexidade crescendo nesses últimos dependendo do número de graus de liberdade. Já a tarefa de navegação refere-se à capacidade de conduzir o robô até um destino no espaço, podendo englobar ou não o planejamento detalhado do trajeto.

Mataric (2007) define "navegação" como o desafio de mover o corpo inteiro do robô a vários destinos, valendo-se de quaisquer mecanismos de locomoção disponíveis para isso, como os de robôs terrestres, voadores, aquáticos, bípedes e outros. A dificuldade desse problema reside na incerteza de suas noções principais: localização do próprio robô, do seu objetivo, como se aproximar dele e avaliação do progresso. Cada um desses problemas é definido de forma separada, e a integração de estratégias que lidam com localização, mapeamento, planejamento e controle é o que viabiliza a navegação autônoma. As próximas seções exploram os conceitos envolvidos em cada um deles.

### 2.1.5 Localização

O desafio da localização consiste em o robô conseguir entender onde ele se encontra no espaço. A estratégia mais básica é chamada de integração de caminhos, do inglês *path integration*, e se baseia nas leituras de odometria dos atuadores responsáveis pela locomoção do robô, tipicamente os encoders das rodas (MATARIC, 2007). A ideia é

que o robô consiga acompanhar sua localização a partir do ponto inicial, adicionando os deslocamentos realizados pelos atuadores.

Tanto atuadores quanto sensores de odometria apresentam imprecisões: a cada movimento do robô acumula-se uma diferença entre a posição que ele se encontra no ambiente e a posição estimada, fenômeno conhecido como "*drift*" de odometria (JAULIN, 2019). Estratégias como utilizar leituras de outros sensores (acelerômetros, giroscópios, etc.) para identificar pontos de referência no ambiente ajudam o robô a mitigar esses erros. Lynch e Park (2017) pontuam que estratégias puramente odométricas funcionam bem apenas em escalas de tempo curtas, sendo então necessário em sistemas mais robustos que as estimativas sejam periodicamente corrigidas por integrações com outras modalidades de percepção, como filtros de Kalman, filtros de partículas ou arquiteturas similares.

Pela perspectiva do modelo de mundo do robô, a complexidade da estimativa de posição pode variar (MATARIC, 2007). Um robô que tenha diversos sensores e um mapa do ambiente ainda pode ter dificuldades para identificar um local específico entre os múltiplos existentes, dado que o reconhecimento de particularidades dos pontos de referência pode não ser possível com a quantidade limitada de informação que o robô possui.

Abordagens de localização incluem o método de localização de Markov e o filtro de Kalman, por exemplo.

O método de localização de Markov é baseado em um conjunto discreto de posições possíveis assumíveis pelo robô no ambiente, dispostas em um grafo topológico ou em um grid geométrico (PANIGRAHI; BISOY, 2022). A posição atual do robô é estimada a partir das informações disponíveis de posições anteriores e dos dados de odometria. Esse método ainda funciona bem em casos em que o robô começa sua trajetória em um ponto desconhecido e precisa estimar a sua posição inicial enquanto trabalha.

O filtro de Kalman é um método para resolver o problema de localização que pode ser utilizado em modelos de mundo contínuos e no qual é necessário conhecer a posição inicial do robô (PANIGRAHI; BISOY, 2022). Pode ser visto como um caso especial da localização de Markov e possui uma etapa de tentativa de previsão das próximas medidas a serem observadas pelos sensores a partir da posição estimada. Em seguida, a diferença dessas previsões para as novas medidas reais é utilizada para refinar a estimativa da posição.

### 2.1.6 Mapeamento

Para ser possível que um robô considere realizar as tarefas de navegação, faz-se necessário que ele seja capaz de perceber o mundo físico (através de sensores) que tenha um entendimento do mesmo (uma representação interna com aspectos que sejam relevantes e

significativos) e tenha algum nível de capacidade de relacionar sua concepção interna de mundo com o que está sendo observado através dos sensores (MATARIC, 2007).

Na perspectiva de percepção, Everett (1995) estabelece que não apenas os sensores devem ser capazes de prover dados com resolução suficiente acerca dos arredores físicos do robô, isso também deve ser feito de forma prática e rápida, e considerando a energia e recursos computacionais limitados do veículo móvel. Portanto, no projeto e utilização de determinado robô para uma determinada tarefa, deve-se levar em consideração as capacidades e limitações daquele conjunto de sensores específico naquela aplicação.

Já quanto à representação interna, Mataric (2007) discute as diferentes formas de fazer um mapa, o chamado modelo de mundo. Robôs que realizam tarefas diferentes precisam de níveis de informação diferentes. São destacados quatro abordagens principais para construção desses modelos:

- Caminho odométrico, no qual o robô armazena as curvas e distâncias percorridas em um determinado caminho para entender a trajetória já feita;
- Caminho baseado em marcos, onde o robô se lembra das decisões e movimentações feitas de forma relativa a marcos no ambiente em que se encontra;
- Mapa topológico, segue ideia análoga aos marcos, mas estabelece relações deles entre si e com ações, de forma que a representação é significativa independente da ordem que o agente os encontra;
- Mapa métrico, no qual o robô constrói um mapeamento completo buscando utilizar as medidas exatas de distância entre os pontos encontrados.

Esses modelos podem ser implementados usando representações como imagens, grafos e listas, cada uma com suas vantagens e limitações. Como o robô é um dispositivo com recursos limitados (memória, energia, processamento, limitação de recursos de sensores, erros de odometria), a escolha dessa representação interna de acordo com o contexto específico se torna um aspecto fundamental.

### 2.1.7 Planejamento de Caminho

O problema do planejamento da rota de um robô parte de situações com os seguintes princípios: a posição atual é conhecida, assim como uma posição objetivo, e existe um sistema de referência comum em que o robô consegue relacionar as duas posições, seja algo explícito como um mapa ou implícito, como alguma heurística (MATARIC, 2007). Em diversos cenários, irão existir múltiplos caminhos entre origem e destino, e encontrá-los envolve uma busca no modelo de mundo, que, em geral, nesse passo, é adaptado para um grafo. A otimização dessa busca pode ser baseada em diversos fatores, mas o processo de

encontrar uma rota ótima pode vir a ser computacionalmente complexo. Como a maioria dos robôs é de sistemas de tempo real, são valorizados algoritmos que encontram rotas boas o suficiente em tempo hábil em vez de algoritmos que encontrariam a rota ideal, mas exigem uma computação custosa, que consome muito tempo.

O algoritmo de Dijkstra é um método clássico de busca não informada que encontra caminhos mínimos em grafos ponderados com pesos não-negativos (LAVALLE, 2006). Ele inicia atribuindo ao vértice inicial distância zero e a todos os outros vértices distância infinita, e então, iterativamente, seleciona o próximo vértice não visitado com menor distância acumulada para reavaliar suas arestas, atualizando as distâncias dos vizinhos quando encontradas rotas mais curtas. O algoritmo é completo e garante encontrar o caminho ótimo sempre que os pesos das arestas forem não-negativos. Porém, isso significa que ele considerará todas as possibilidades de caminho antes de retornar uma resposta.

Já o algoritmo A* é uma busca informada que, ao incorporar uma função heurística que representa a tendência de cada posição a aproximar-se do destino final, consegue priorizar a expansão de nós com maior probabilidade de conduzir ao objetivo, mantendo-se ótimo e completo sob heurísticas admissíveis (LAVALLE, 2006). Ele é uma extensão do algoritmo de Dijkstra que incorpora essa função heurística, e pode terminar antes de haver a necessidade de conferir todas as possibilidades de caminho. Caso um caminho seja possível, eventualmente se chega ao objetivo, e é possível reconstruir o caminho a ser retornado através de *backtracking*.

Independente do algoritmo, o caminho, ou seja, o conjunto subsequente de pontos na representação de mundo que o levam do ponto inicial até o seu objetivo, caso exista, será fornecido ao final da execução. Porém, dado o caminho planejado, um novo desafio surge: como transformá-lo em instruções para os motores do robô.

### 2.1.8 Controle

Uma vez definido o caminho, o robô o executa via controlador de movimento que converte a trajetória em comandos de velocidade. Em problemas em que tempo e perfis cinemáticos (velocidade, aceleração) são críticos, seja para garantir conforto, segurança ou estabilidade, fala-se em “rastreamento de trajetória”; caso contrário, em “seguimento de caminho” (LYNCH; PARK, 2017). Existem abordagens desacopladas (planejamento e seguimento independentes) e métodos diretos, que derivam leis de controle unificadas a partir de um modelo cinemático (CHOSET et al., 2005).

Em arquiteturas comportamentais ou reativas, usadas em sistemas baseados em navegação topológica, o robô não necessariamente utiliza uma estimativa de posição e orientação métrica contínua global. Nesses casos, ele pode se orientar por primitivas simples (seguir muro, linha ou vetor) e por gatilhos de reconhecimento de marcos para transitar

entre eles. Esse estilo de navegação usa sequências de comportamentos discretos, cada um responsável por levar o robô de um ponto de interesse a outro, sem a necessidade de computar trajetórias detalhadas (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

### 2.1.9 Plataforma e componentes

Um conjunto de aspectos importantes a serem considerados para o desenvolvimento deste trabalho são as especificações técnicas e princípios de funcionamento dos componentes da plataforma utilizada. Como descrito anteriormente, embora não haja nenhuma limitação de natureza de componentes na modalidade *OPEN* da categoria *Challenge*, por uma questão de disponibilidade de materiais e familiaridade prévia tanto da equipe envolvida quanto do autor, foram utilizadas principalmente peças do kit LEGO *Mindstorms* EV3. Trata-se da terceira geração de kits de robótica desenvolvidos pela LEGO, e a última da linha *Mindstorms*, utilizada em várias competições de robótica, em destaque as com caráter educacional mais forte, como a Olimpíada Brasileira de Robótica, a *FIRST LEGO League*, e a *World Robot Olympiad*, por exemplo (BRANCALIÃO et al., 2022).

O kit é composto por peças para montagem de estruturas e mecanismos, e componentes especiais modulares: controladores, motores e sensores (GUIA. . . , 2016). A Figura 7 ilustra os principais componentes conectados entre si.

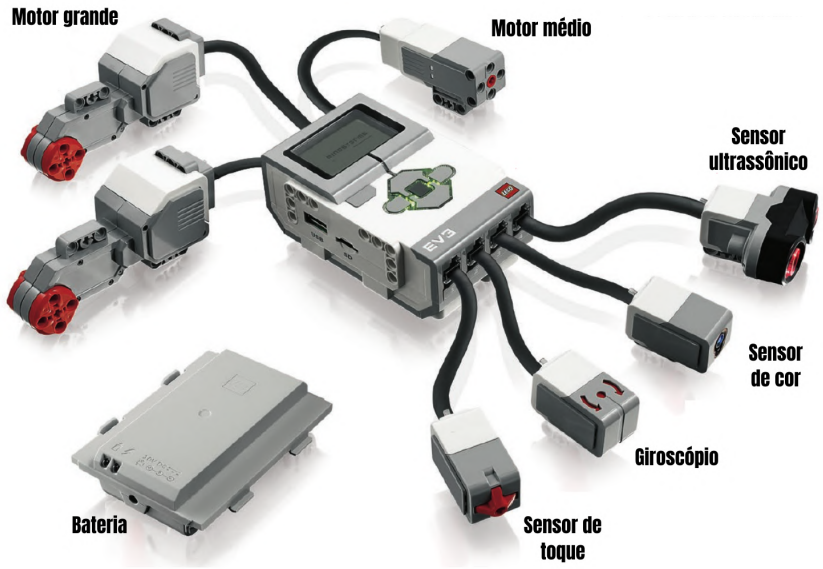


Figura 7 – Principais componentes do kit EV3, conectados ao *brick*. Fonte: Adaptado de (GUIA. . . , 2016).

O controlador, chamado de "*brick*", ou apenas "EV3", é o componente central de computação do sistema, fundamental para o controle e programação dos outros. Ele é semelhante a uma caixa, tem um monitor com resolução de 178x128 pixels monocromáticos para interface visual do sistema e dos programas, e um teclado de botões direcionais

para interação direta do usuário, que é iluminado por LEDs com capacidade de 3 cores diferentes (verde, laranja e vermelho). Estão presentes nele 8 portas RJ12 dedicadas para os dispositivos de entrada e saída compatíveis do kit: onde se conectam os motores e sensores. Nas 4 portas superiores, próximas ao monitor e nomeadas de "A" a "D", são conectados os motores. Já nas portas inferiores, próximas ao teclado e numeradas de 1 a 4, são conectados os sensores.

Além destas, existem ainda: uma porta para mini USB, utilizada para conectar o bloco programável a um computador; uma porta para cartão SD, disponível para expansão de memória em até 32 GB; uma porta *host* USB, que pode ser utilizada para adicionar adaptadores Wi-Fi e permitir conexões a redes sem fio e também para encadeamento *daisy-chain* de outros *bricks*. Uma bateria recarregável própria pode ser acoplada na parte traseira do controlador, onde também há suporte para pilhas convencionais. Por último, também está presente um alto-falante e há suporte nativo a conexões *Bluetooth*. A Figura 8 destaca as características do controlador.

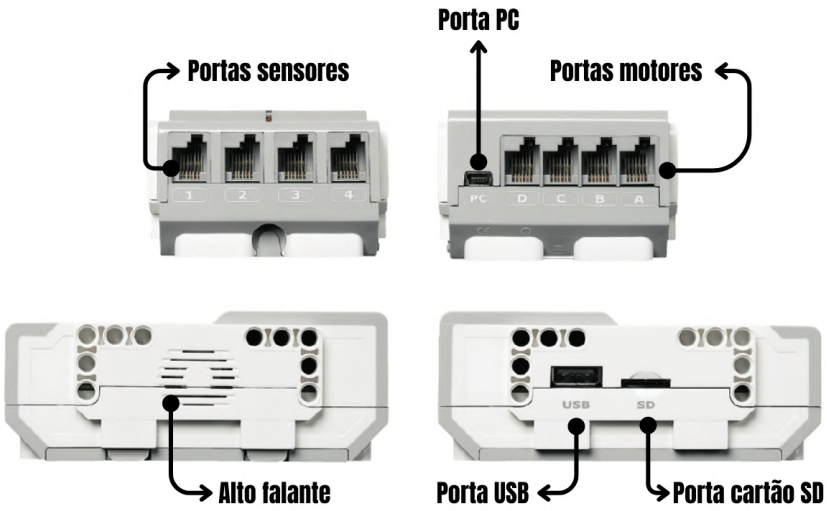


Figura 8 – *Brick* EV3 e suas portas. Fonte: Adaptado de (GUIA. . . , 2016).

O *brick* possui um controlador ARM 9 de 300 MHz e seu sistema operacional é baseado em Linux. Tem 64 MB de capacidade de memória RAM e 16 MB de armazenamento, baseado em memória *flash*. Seu sistema base é proprietário da LEGO, e tem seu princípio de funcionamento baseado exclusivamente nos softwares educacionais da empresa, com o uso de uma linguagem própria de programação em blocos, por exemplo. Mas existem alternativas *open-source* que permitem aos desenvolvedores de robôs maior liberdade para utilização dos recursos computacionais presentes, possibilitando lidar diretamente com o sistema operacional, sistema de arquivos, protocolos de comunicação, dispositivos de entrada e saída e outras linguagens de programação. O maior exemplo dessas iniciativas é o *ev3dev*, um sistema baseado em Debian Linux, inicialmente desenvolvido a partir de engenharia reversa, que pode ser utilizado no EV3 a partir da sua

instalação em um micro SD bootável (EV3DEV..., 2020).

Várias bibliotecas foram desenvolvidas pela comunidade para disponibilizar interfaces para os componentes do kit em diversas linguagens de programação. As principais suportam Python, MicroPython e Java, mas também pode-se citar Go, C++, C, Rust. A documentação do *ev3dev* também descreve drivers e interfaces disponíveis para que novas bibliotecas possam ser implementadas em qualquer linguagem. Dentre as mais relevantes para este trabalho, destaca-se a solução em MicroPython: uma variante enxuta da linguagem Python que oferece inicialização significativamente mais rápida, porém com compatibilidade parcial em relação ao Python padrão e suas bibliotecas. A biblioteca Pybricks é uma versão especializada de MicroPython para o LEGO *Mindstorms* que tem suporte da empresa, sendo sua solução oficial para linguagens de programação no EV3 (PROGRAMMING..., 2020).

Dentre os outros componentes disponíveis, listam-se os seguintes: motor grande, motor médio, sensor de cor, giroscópio, sensor de toque, sensor ultrassônico, sensor infravermelho, sensor de temperatura. Os princípios de funcionamento dos mais relevantes para este trabalho, assim como suas limitações, serão abordados a seguir.

#### 2.1.9.1 Motores

Existem duas variações de motores no kit EV3, identificados como motor grande e motor médio (GUIA..., 2016). Ambos possuem sensores de rotação com resolução de 1 grau, possibilitando estratégias de controle preciso. O motor grande é projetado para ser a base motriz dos robôs, impulsionando as rodas. Ele funciona com 160-170 rpm, com torque de funcionamento de 20 Ncm e torque neutro de 40 Ncm. Já o motor médio funciona com 240-250 rpm, com torque de funcionamento de 8 Ncm e torque neutro de 12 Ncm. No guia do usuário é resumido que o motor grande é mais lento, porém mais forte, enquanto o motor médio é mais rápido, porém menos potente. Na estrutura do robô utilizado como objeto de estudo, todos os motores das rodas são grandes, e a garra contém um motor grande para movimentação vertical ("sobe-desce") e um motor médio para movimentação horizontal ("abre-fecha"). A nível de software, os dois tipos de motores possuem as mesmas funcionalidades (GETTING..., 2020).

#### 2.1.9.2 Sensores de Cor

O sensor de cor (Figura 9) é um sensor digital capaz de detectar cores de superfícies ou intensidades luminosas (GUIA..., 2016), a partir da quantidade de luz que entra pela pequena janela na sua face. Ele emite uma luz que permite ao sensor captar as diferentes intensidades das frequências de onda refletidas por superfícies. Ele tem três modos de funcionamento: modo de luz refletida, modo de luz ambiente e modo de cores. No modo de luz refletida, o sensor emite uma luz vermelha e detecta numa escala de 0 a 100

a intensidade refletida, sendo 0 pouca luz e 100 muita luz. Esse modo é o ideal para detecção de marcações monocromáticas em superfícies, como para perceber uma linha preta numa superfície branca, por exemplo. No modo de luz ambiente, o sensor mede a luz vinda do ambiente, como de lâmpadas ou do sol, também em uma escala de 0 a 100.

Já no modo de cores, o sensor emite uma luz colorida (em tom branco-azulado) e reconhece um conjunto de sete cores diferentes (preto, azul, verde, amarelo, vermelho, branco, marrom, além de leituras "sem cor"), segundo um padrão de classificação implícito e definido pela própria fabricante (GUIA. . . , 2016). A nível de software, é possível capturar as leituras "cruas" do sensor de cor, sendo estas tuplas de três dimensões com valores de 0 a 100 para cada uma das cores básicas detectadas (vermelho, verde e azul) (GETTING. . . , 2020). Cada um dos valores dessa tupla RGB indica a intensidade de luz refletida que se encaixa no espectro daquela cor. Isso é especialmente útil por conta da principal limitação desse tipo de sensor: o modo padrão de cores não é suficientemente preciso ou configurável.

No mundo real, cores não são variáveis discretas, e têm complexidade maior do que o representável em três valores de reflexão percentual. É possível observar como isso afeta a efetividade desses sensores: o modo de detecção de cores frequentemente responde com leituras diferentes das que um ser humano classificaria como corretas, confundindo cores relativamente similares (como amarelo e marrom, ou preto e azul escuro, ou preto e verde escuro, como exemplos comuns) e, às vezes, até fazendo leituras variadas numa mesma superfície de coloração aparentemente homogênea, inclusive sendo influenciado pelas condições de iluminação do ambiente. No contexto de competições, por exemplo, é muito comum que tonalidades de cores variem entre os cenários em que as equipes treinam e os das competições, além das condições de iluminação serem praticamente imprevisíveis. Além disso, a distância da superfície ao sensor impacta bastante a corretude das classificações, que geralmente são confiáveis apenas com a superfície entre 1 e 2cm de distância da face do sensor, o que limita consideravelmente as suas possibilidades de uso.

Para lidar com esses problemas é comum que equipes elaborem estratégias de "re-calibração" dos valores utilizados para categorização de cores, ou que considerem usar sensores melhores, dependendo da permissividade das regras de cada competição. Para o robô deste trabalho, essas duas estratégias foram abordadas, e os detalhes da implementação serão descritos no próximo capítulo, partindo de técnicas de classificação de dados abordadas nas próximas subseções. Na via de sensores melhores, foi utilizado "*HiTechnic color sensor V2*", produzido por outro fabricante, mas compatível com o sistema do EV3.

Diferente dos sensores de cor padrão da LEGO, o *HiTechnic* (Figura 9) pode emitir uma única luz branca, de intensidade mais forte e canalizada. Ele possui um princípio de funcionamento diferente, baseado em um chip sensível a luz, com partes cobertas por filtros vermelho, verde e azul (HURBAIN, 2009). Por conta disso, ele consegue detectar as cores de qualquer luz que o adentre, mesmo num modo passivo, sem necessariamente

ter que ligar seu próprio LED.

Além de ser possível pegar as leituras cruas, esse sensor também consegue fazer a própria discretização de cores, detectando 18 diferentes. O intervalo de distância de detecção também é maior, tendo resultados consistentes em até 5cm de distância das superfícies lidas, e erros menores e menos frequentes a uma distância de 8cm. (HURBAIN, 2009). Porém, a luz emitida pelo sensor, dependendo da reflexividade da superfície, pode acabar interferindo nas leituras. Para lidar com isso é recomendável usar o sensor inclinado a um leve ângulo, e não exatamente perpendicular à superfície lida.

Testes executados por Hurbain (2009) mostram há uma alta resistência a diferentes condições de iluminação ambiente, que praticamente não afetam os resultados. Porém, testes também mostram que o *HiTechnic* teve um desempenho inferior no quesito velocidade de detecção quando comparado com o sensor de cor do kit LEGO *Midstorms* NXT, geração anterior à do EV3. Assim, o *HiTechnic* não é a melhor opção para aplicações que exigem leituras rápidas, como detecção de linhas em um robô seguidor, por exemplo, mas é a escolha certa para aplicações que se beneficiem de uma distância de leitura maior, uma discriminação em mais tons de cor, ou imunidade a condições de iluminação externa.


Figura 9 – Sensores infravermelho, ultrassônico, de cor e *HiTechnic*.
Fonte: Adaptado de (GUIA..., 2016); renderização virtual do *HiTechnic* a partir do software *Bricklink Studio*.

#### 2.1.9.3 Giroscópio

O giroscópio é um sensor digital capaz de detectar movimento rotacional em torno de um eixo, sua taxa de rotação em graus por segundo, e de fazer o rastreio do ângulo total, em graus (GUIA..., 2016). Portanto, pode ter muito valor para estratégias de navegação e estimativa de estado para robôs móveis.

Porém, com relação a limitações desse sensor, empiricamente é perceptível que ele possui problemas frequentes de "*drift*" que tornam a sua utilização mais complexa, exigindo mais cuidados e tratativas para erros. Por conta disso, e pela maior necessidade de outros sensores que já ocupavam todas as portas disponíveis, ele acabou não entrando para os componentes utilizados no robô objeto de estudo deste trabalho.

#### 2.1.9.4 Sensor Ultrassônico

Sensores ultrassônicos compõem o primeiro tipo de sensor de distância utilizado: são capazes de medir a quantos centímetros algum objeto detectado se encontra (GUIA..., 2016). O princípio de funcionamento é baseado em ondas sonoras de alta frequência, que são emitidas e capturadas pelo sensor, que então é capaz de calcular distâncias baseando-se no tempo levado para a onda se propagar até o objeto e retornar. A distância detectável pelo sensor ultrassônico do kit EV3 é entre 3 e 250 cm, com precisão de 1 cm para mais ou para menos. O valor retornado quando o sensor não consegue detectar nenhum objeto é de 250 cm, o que pode ser um problema na interpretação de objetos detectados quando estão mais próximos do que a distância suportada pelo sensor (menos de 3 cm). Por conta da forma em que o som se propaga, esse sensor detecta objetos não em linha reta, mas num "raio" de direções de propagação.

#### 2.1.9.5 Sensor Infravermelho

O sensor infravermelho é outro tipo de sensor de distância, diferente do ultrassônico: seu princípio de funcionamento é baseado na propagação de luz infravermelha, e na detecção da porcentagem refletida em objetos (GUIA..., 2016). Portanto, o sensor infravermelho não é capaz de prover informações precisas sobre a distância em centímetros dos objetos detectados, mas retorna um valor entre 0 e 100 correspondente a essa proximidade, sendo 0 perto e 100 longe. Como diferentes materiais e cores refletem a luz de diferentes formas, isso pode acabar influenciando os valores retornados pelos sensores. Apesar disso, o sensor do EV3 é capaz de detectar objetos em até 70 cm de distância, dependendo do tamanho e formato. Por conta da forma com que a luz se propaga, esse sensor faz as detecções de objetos em linha reta.

### 2.1.10 Aprendizado de Máquina para Classificação de Dados

O problema da classificação de cores citado pode ser abordado dentro do escopo de técnicas de aprendizado de máquina. Os detalhes de implementação serão trabalhados no próximo capítulo, e esta seção contextualiza os conceitos teóricos envolvidos.

O aprendizado de máquina (ou *machine learning*) é uma área da inteligência artificial voltada para o desenvolvimento de algoritmos capazes de aprender padrões a partir de conjuntos de dados e realizar previsões ou tomadas de decisão com base neles. Um dos

problemas estudados é o de classificação, que envolve treinar um modelo computacional para que seja capaz de classificar corretamente mesmo entradas previamente desconhecidas. Os atributos são utilizados para que o modelo consiga inferir a classe (como um rótulo) das observações. Para isso, ele é treinado num conjunto de dados em que essas classificações sejam previamente conhecidas, caracterizando uma técnica de aprendizado supervisionado (MICHALSKI; CARBONELL; MITCHELL, 2013).

Algumas técnicas de classificação com aprendizado supervisionado são: k-vizinhos mais próximos (kNN), árvores de decisão, florestas aleatórias e redes neurais.

A técnica de k-vizinhos mais próximos classifica novas entradas com base na classificação das entradas conhecidas mais próximas no espaço de características (BISHOP; NASRABADI, 2006). A classe que prevalecer entre os k-vizinhos cujas medidas de distância são as menores à nova entrada será a resultante. Existem algumas métricas que podem ser aplicadas, um exemplo sendo a distância euclidiana. Nessa técnica não há uma fase de treinamento explícita: a classificação exige o processamento dos cálculos em tempo de execução da consulta.

Árvores de decisão particionam recursivamente o espaço de características em regiões homogêneas quanto à classe alvo (BISHOP; NASRABADI, 2006). Cada nó da árvore faz uma pergunta que separa os dados quanto a um atributo, e cada folha diz respeito à previsão de uma classe. O algoritmo de treinamento é que vai projetar quais perguntas proporcionam maior ganho de informação, para que a árvore gerada seja a mais significativa possível. As vantagens de árvores de decisão são que elas não requerem normalização de dados, lidam bem com variáveis categóricas e são facilmente interpretáveis. Porém, elas tendem a se sobreajustar ("*overfitting*") sobre o conjunto de dados de treinamento.

Partindo da ideia das árvores de decisão, o método das florestas aleatórias cria múltiplas árvores durante o treinamento, a partir de várias segmentações feitas aleatoriamente no conjunto de dados disponível (HASTIE et al., 2005). Na fase de classificação, todas as árvores geradas são consultadas, e a classe que prevalecer entre elas será a resultante. Esse processo ajuda a diminuir a tendência a *overfitting* da técnica anterior, possibilitando modelos com alta precisão, robustez a ruídos e capacidade de estimar a importância de atributos. Porém, perdem o fator de interpretabilidade das árvores mais simples.

Por último, redes neurais são modelos computacionais inspirados no cérebro humano, compostas por pequenas e numerosas unidades simples de processamento, os neurônios, que são amplamente conectados entre si em diversas camadas (BISHOP; NASRABADI, 2006). A ideia é que o conjunto de estímulos de entrada de um neurônio passa por sua função de ativação, e aquele neurônio propaga o sinal resultante para todos com os quais está conectado na próxima camada, e assim sucessivamente até a rede convergir para uma saída. Em problemas de classificação, essa saída é a classe resultante. O

aprendizado do modelo da rede acontece graças ao algoritmo de retropropagação de erros, que progressivamente calibra os pesos de cada uma das conexões entre os neurônios. As redes neurais são extremamente versáteis e conseguem capturar padrões complexos, porém pode ser necessário grandes volumes de dados rotulados e alto poder computacional para treinamento para que resultados precisos sejam gerados. Além disso, trata-se de uma técnica com difícil interpretação de suas decisões, já que a rede neural é como uma "caixa-preta", no sentido de que o modelo produzido é um grande conjunto de pesos de nós computacionais interrelacionados.

## 2.2 Trabalhos relacionados

Esta seção se dedica a apresentar trabalhos correlatos, também dentro da área de pesquisa em robôs autônomos móveis, que proponham, implementem ou comparem estratégias para solução de um ou múltiplos dos desafios de localização e navegação em ambientes estáticos ou dinâmicos, relacionados ou não a robôs omnidirecionais e competições de robótica.

Alves (2017) investiga o problema de um time de robôs em um ambiente interno, cada robô tendo que se localizar e acessar localidades para concluir a tarefa. O escalonamento das rotas dos múltiplos agentes foi modelado como uma instância do problema de múltiplos caixeiros viajantes (do inglês, multiple Traveling Salesmen Problem, mTSP), e a abordagem utilizada para encontrar possíveis soluções foi a utilização de algoritmos de aproximação. Para a localização dos robôs, foi utilizada a combinação de duas técnicas: um algoritmo de localização via WiFi é responsável por acompanhar a localização global de cada robô, enquanto um algoritmo baseado nas leituras do sensor Kinect do robô estimava a posição específica do robô em uma área. Além disso, também são utilizadas técnicas baseadas em filtro de partículas. Para lidar com o ambiente dinâmico, é introduzido um algoritmo de desvio de obstáculos e um planejador dinâmico de rotas, baseados em técnicas como lógica fuzzy e redes neurais. Para o planejamento de rotas foi utilizado o algoritmo D*Lite, assim como modificações para otimização do mesmo em ambientes que requerem recálculos de rotas em tempo real. Todos os algoritmos são validados em robôs reais e simulados. Bons resultados são obtidos, mas os direcionamentos para trabalhos futuros envolvem melhorias nas integrações dos subsistemas, a realização de mais testes em times de robôs reais e possíveis melhorias nos sinais recebidos pelos sensores, assim como nas tratativas que lidam com os mesmos.

Suryaprakash, Thirumoorthi e Viswanathan (2023) apresentam um sistema de navegação para grandes ambientes internos complexos, desenvolvido considerando um robô baseado em Arduino com sensores ultrassônicos, inerciais e magnetômetros. Algoritmos de fusão de sensores são utilizados para obter melhor acurácia na estimativa de posição e

orientação, que influenciam diretamente na eficácia dos algoritmos de localização e navegação. O planejamento de rota é feito utilizando o algoritmo "*Probabilistic Road Map*", e comparado com a eficiencia dos algoritmos A* e Dijkstra, obtendo tanto caminhos quanto tempos de execução menores. O mapeamento é feito em uma rodada de inspeção em que o robô é dirigido manualmente pelo cenário enquanto coletando dados dos sensores ultrassônicos, um na frente do robô e um em cada lateral. São realizados testes de hardware (performance, sensores, movimentação, curvas) e software (algoritmo de fusão de sensores, localização, planejamento de caminho). A validação do sistema é realizada pela comparação da rota efetivamente traçada pelo robô com a gerada pelo algoritmo de planejamento, e isso é feito com o auxílio de um sistema de captura de movimentos por câmeras.

Leandro (2025) trabalha com o objetivo de desenvolver (hardware e software) um robô com rodas omnidirecionais *mecanum* e seu sistema de movimentação, desenvolvendo bastante a análise da cinemática envolvida. O robô, baseado em Arduino e seguindo a filosofia de Desenvolvimento Baseado em Modelo, foi construído tanto em ambiente simulado, através da ferramenta Simulink, quanto físico. Foram realizados testes comparativos entre os movimentos realizados em ambiente simulado e real, e as divergências encontradas por conta de limitações do modelo, como a falta de sensores de velocidade nos motores, levantam melhorias para futuras implementações, a fim de aumentar a precisão e a fidelidade entre o comportamento simulado e o comportamento real do robô.

Júnior (2019) desenvolve um robô omnidirecional para participação em competições de futebol, em especial a categoria *Very Small Size*, da CBR e da IEEE LARC, através da equipe Rodetas Robô Clube, da Universidade Federal de Ouro Preto. São expostas as regras e limitações impostas aos robôs da categoria. O trabalho se concentra mais nos aspectos de engenharia e do projeto eletrônico e estrutural do robô, mas inclui a análise da mobilidade omnidirecional de um robô com rodas *mecanum*. O uso desse tipo de movimentação nessa categoria é apontado como inovador, e são levantadas vantagens e desvantagens da abordagem na competição: a principal vantagem sendo a versatilidade de movimento, mas as principais desvantagens sendo a maior suscetibilidade a escorregamentos e aumento na complexidade de estimativa de posição baseada em odometria. São sugeridos trabalhos futuros que lidem com essas estimativas a partir do processamento das imagens da câmera presente no sistema de partidas da categoria, e com a implementação de um sistema de controle de velocidade mais adaptado.

Chenchani et al. (2023) descreve a abordagem adotada pela equipe b-it-bots para vencer o campeonato mundial da *RoboCup@Work* 2023, categoria que envolve robôs móveis autônomos executando tarefas no ambiente industrial, como mapeamento, localização, navegação e manipulação de objetos. O artigo foca nas últimas melhorias desenvolvidas pela equipe, tanto em hardware quanto em software, e descreve como foi feita a abordagem dos problemas de percepção, manipulação e planejamento de tarefas envolvidas nas novas

regras e no aumento da complexidade dos objetos envolvidos. O robô utilizado é baseado na plataforma *KUKA youBot*, e contém uma base de movimentação omnidirecional com 4 rodas *mecanum*, um manipulador com 5 graus de liberdade, sensores baseados em laser e câmeras RGB-D para visão computacional. A arquitetura do software é baseada em ROS.

# 3 Metodologia

### 3.0.1 Estrutura do robô utilizado

O robô foi inicialmente desenvolvido pela Equipe Roboforge de Robótica, projeto extracurricular da Universidade Federal de Uberlândia, para a Competição Brasileira de Robótica sediada em Goiânia em novembro de 2024. O modelo e software apresentados pela equipe na categoria RCB *Challenge OPEN*, com foco neste último, serão documentados e estendidos nos trabalhos desenvolvidos nesta monografia. O autor deste texto teve envolvimento fundamental em todas as etapas de desenvolvimento do robô e do software, dentro e junto da equipe, para a competição. Os pontos de melhoria levantados e desenvolvidos neste trabalho trazem a perspectiva, inclusive, de outros membros da equipe sobre como a eficiência do robô no desafio poderia ser melhorada. No evento, o robô em questão ficou classificado em 4º lugar na modalidade *OPEN*. A maior parte da elaboração deste trabalho acontece após o encerramento da participação da equipe na competição.

A estrutura é composta de peças do kit LEGO *Mindstorms* EV3, com a adição de peças especiais extras: um conjunto de 4 rodas "*mecanum*", permite que o robô possa se mover de forma omnidirecional, um sensor de cor *HiTechnic*, e outras peças, como ligas de borracha para maior aderência na pinça da garra. A escolha das peças foi pensada pela equipe de acordo com as que estivessem disponíveis, de acordo com limitações materiais e financeiras.

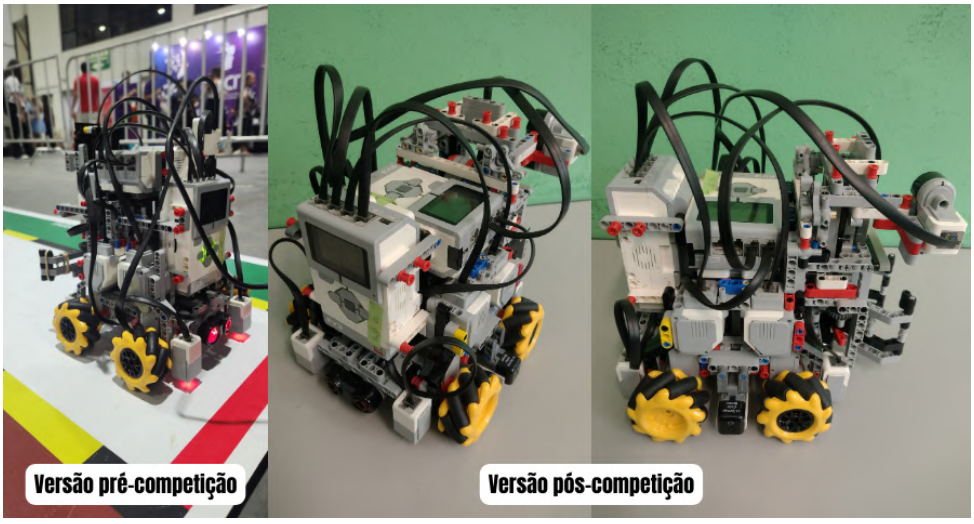


Figura 10 – Estrutura do robô e sua evolução pré e pós competição: algumas alterações incluem a disposição física dos controladores e o mecanismo da garra. Fonte: Arquivo pessoal.

O projeto da equipe sofreu alterações em vários momentos da preparação e parti-

cipação na competição (Figura 10). Os aspectos a seguir descrevem o conceito do robô, que cada versão diferente de base e estrutura montada sempre tentava respeitar. Como este trabalho foca mais nos aspectos de *software* do que *hardware* envolvidos, os detalhes da montagem e processo de desenvolvimento estrutural não serão abordados. Assim, o robô é composto por:

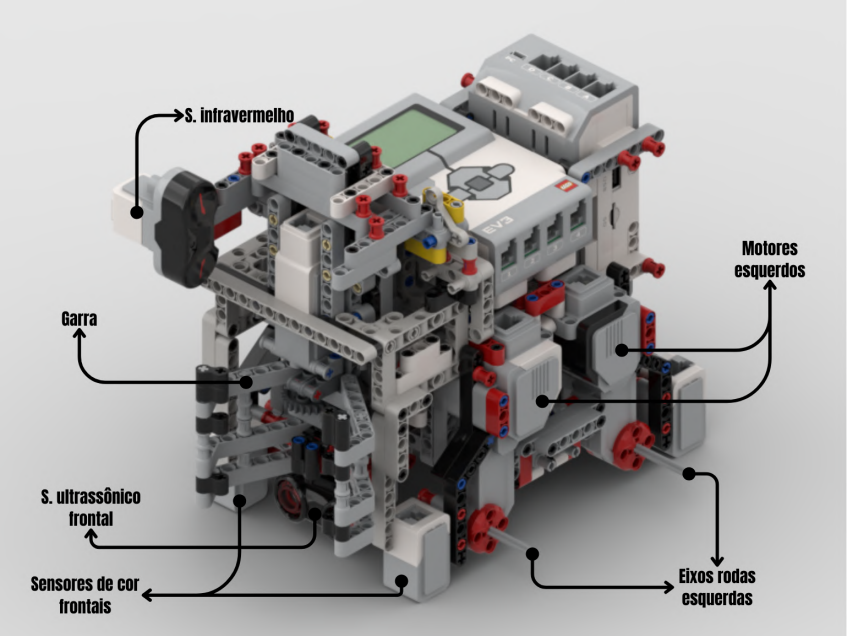


Figura 11 – Renderização digital do robô, visão frontal esquerda. Fonte: Autoral.

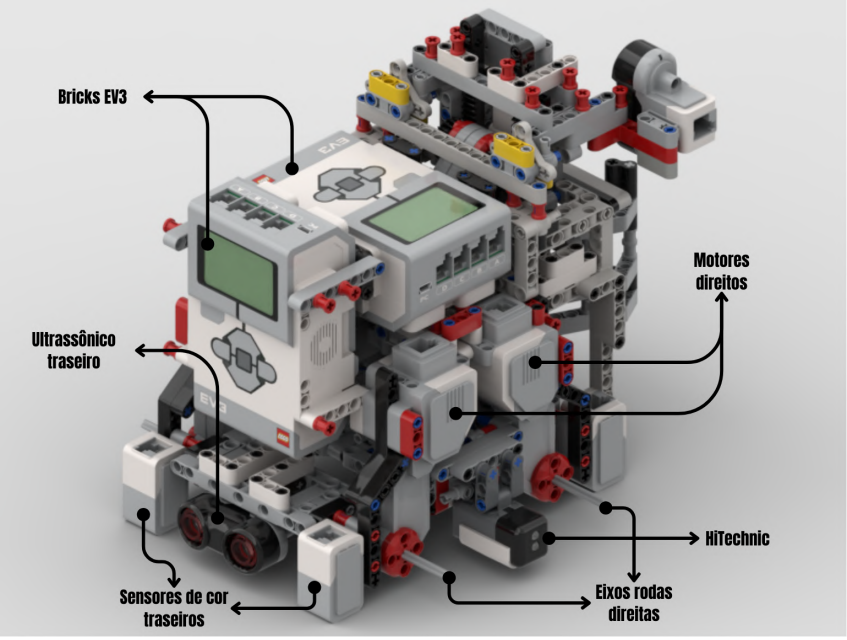


Figura 12 – Renderização digital do robô, visão traseira direita. Fonte: Autoral.

- 2 controladores EV3 ("*bricks*"), para suportar todos as conexões de sensores e motores (cada um tem 4 portas de motores e 4 portas de sensores, apenas);

- 4 motores grandes para movimentação das 4 rodas omnidirecionais;
- 4 sensores de cor apontados para o solo em cada extremidade do robô;
- 2 sensores ultrassônicos para detecção de obstáculos nas partes dianteira e traseira;
- uma garra com 2 graus de liberdade ("abre-fecha" e "sobe-desce") com um sensor de distância acoplado para detecção do passageiro a bordo;
- e um sensor de cor *HiTechnic* na lateral direita, apontado para fora, para varredura da área de embarque e detecção das cores associadas aos passageiros.

As Figuras 11 e 12 destacam os principais elementos do robô. O modelo digital foi feito utilizando o software *Bricklink Studio* (STUDIO. . . , 2025).

## 3.1 Arquitetura de Software

O código desenvolvido em Python (mais especificamente MicroPython, com a biblioteca Pybricks) se organiza em um repositório com funcionalidades do robô e de suporte ao desenvolvimento, com ferramentas para execução em um computador. A estrutura de arquivos se organiza como na Figura 13.

Os arquivos a serem transferidos para o robô, e posteriormente executados, são aqueles dentro da pasta nomeada "*src*". Nesta pasta está localizado um conjunto de arquivos Python disponíveis para execução no *brick*, assim como arquivos de constantes e variáveis que podem ser calibradas durante o desenvolvimento, como nos arquivos "*constants.py*" e "*pid_const.json*". O principal arquivo é o "*main.py*", que inicializa a rotina completa para resolução do desafio no robô. O arquivo "*color_calibration.py*" inicializa a rotina de coleta de dados para treinamento do modelo de classificação de cores, com o objetivo de calibrar os sensores de cor (mais detalhes sobre isso serão abordados nas próximas seções). Os arquivos "*pid_calibration.py*", "*test_boarding.py*", "*test_calibration.py*" e "*turn_calibration.py*" são executáveis que auxiliam os desenvolvedores a realizarem testes de funcionalidades isoladas (movimentação, curvas, embarque de passageiros) e na calibração de variáveis dos algoritmos, como constantes de controle PID de diversas funções. Portanto, são módulos não utilizados na rotina principal, na fase de execução, mas que têm sua relevância nas fases de desenvolvimento, treino e calibragem.

Ainda dentro da pasta "*src*" estão os subpacotes "*core*", "*decision_trees*" e "*domain*".

Ao lado da pasta "*src*", temos a pasta "*tools*", que contém códigos feitos para serem executados em um computador, a fim de auxiliar os desenvolvedores durante o desenvolvimento, calibração e testes. O módulo "*decision_trees.py*" contém o código que

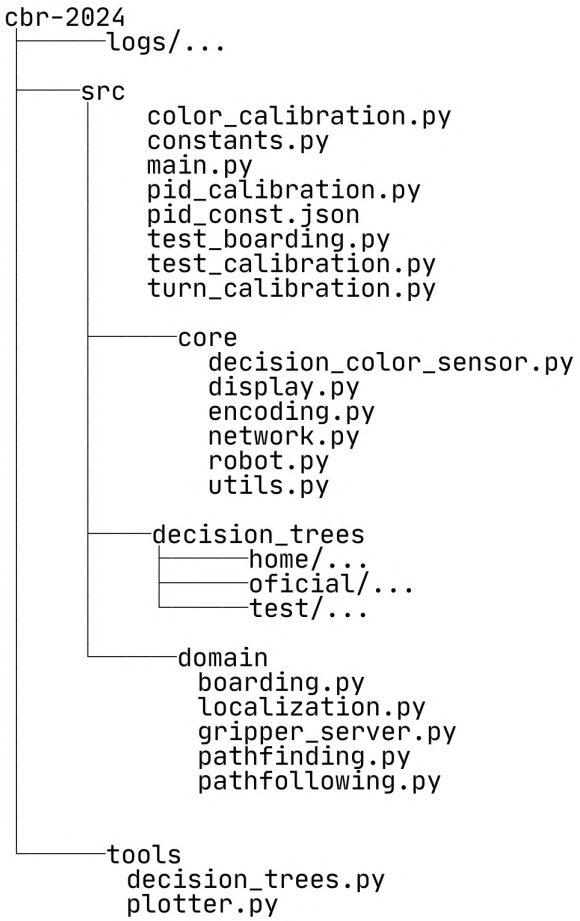


Figura 13 – Estrutura de arquivos do repositório do projeto. Fonte: Autoral.

treina e exporta os modelos de árvore de decisão, a partir de dados coletados pelos sensores de cor. O módulo "*plotter.py*" contém ferramentas para visualização gráfica de dados coletados por logs de execução. À medida que logs são gerados, eles vão sendo coletados e organizados na pasta "logs", para utilização e futuras consultas por parte dos desenvolvedores.

É dentro da pasta "*src*", no pacote "*decision_trees*" que as árvores de decisão geradas são colocadas, no formato de código Python pronto para utilização. Na Figura 13, por exemplo, os subpacotes "*home*", "*oficial*" e "*test*" contêm as árvores de decisão geradas pela calibração dos sensores nos mapas utilizados no desenvolvimento pela equipe Roboforge, nas partidas oficiais e na arena de testes da competição, respectivamente.

A organização de pacotes, módulos e a relação entre eles se organiza como na Figura 14.

A estrutura do repositório e dos pacotes foi inspirada em conceitos de *Domain Driven Design* e separação de responsabilidades, e segue uma ideia de camadas de abstração em diferentes níveis (Figura 15).

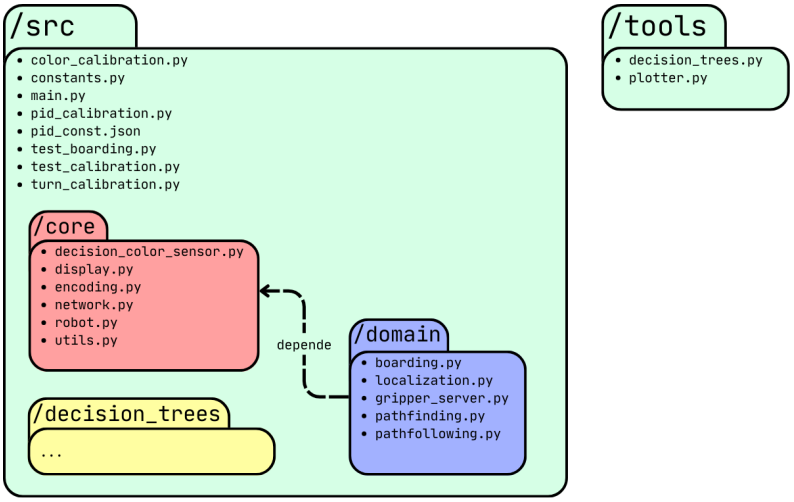


Figura 14 – Diagrama de pacotes do projeto. Fonte: Autoral.

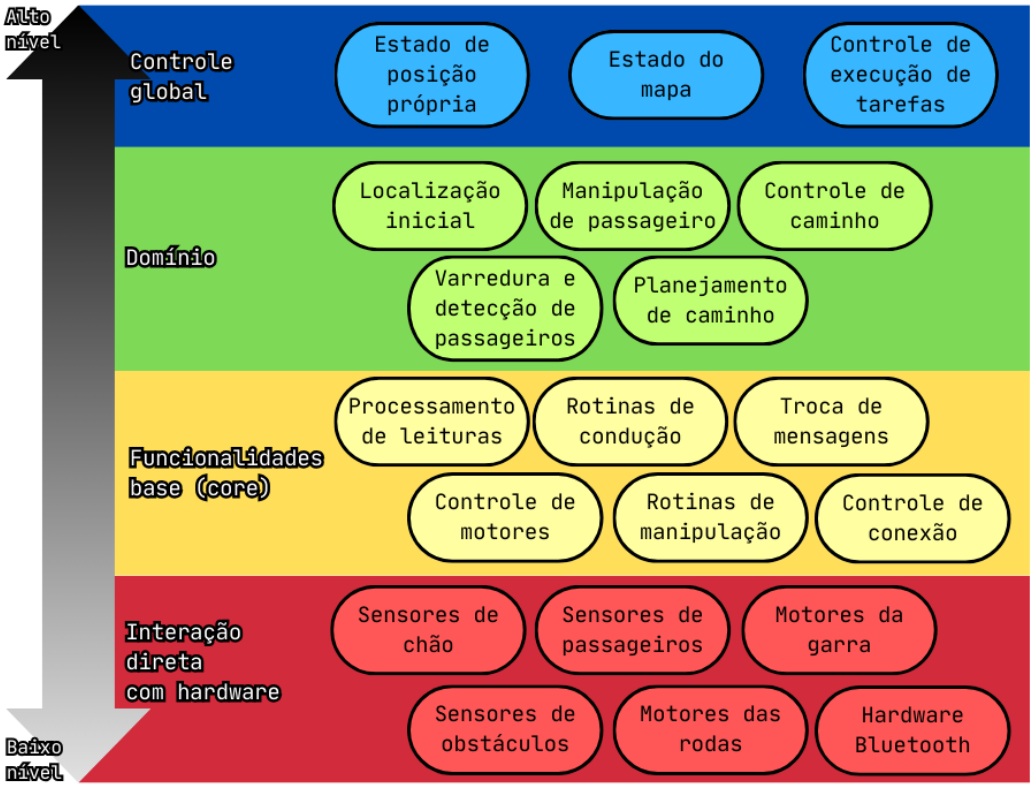


Figura 15 – Hierarquia das camadas de abstração do projeto. Fonte: Autoral.

Em mais baixo nível estão as funcionalidades que interagem diretamente com o hardware dos componentes, sejam sensores, motores ou até os componentes eletrônicos *Bluetooth*. Nessa camada está, por exemplo, a implementação da biblioteca PyBricks, que abstrai a interação direta com o hardware (junto aos drivers do sistema operacional) através de interfaces programáveis. Essas interfaces são utilizadas pelas camadas superiores, que não lidam com os detalhes da implementação.

Em um nível mais alto estão as funcionalidades básicas de um robô, que podem

ser reutilizadas em outros contextos e aplicações (outros desafios, por exemplo). Essa camada é de responsabilidade dos módulos no pacote "*core*" do projeto. Por exemplo, processamento de leituras é algo tratado pelo módulo "*decision_color_sensor.py*", que implementa a integração das árvores de decisão aos sensores de cor, entregando para as camadas superiores uma interface transparente sobre os detalhes de implementação. Controle de motores, rotinas de condução e rotinas de manipulação são implementadas no módulo "*robot.py*", enquanto a lógica de troca de mensagens e controle de conexão *Bluetooth* nos módulos "*encoding.py*" e "*network.py*". Além desses exemplos, os outros módulos desse pacote contêm funcionalidades que também se encaixam nessa camada de abstração.

Ainda mais acima estão as funcionalidades que já dizem respeito à implementação da solução para o desafio específico da categoria RCB *Challenge* 2024, sendo este o domínio de trabalho, e por isso o pacote "*domain*". As funcionalidades das camadas de abstração inferiores são os "blocos de construção" dos algoritmos desenvolvidos para atacar os problemas propostos pelo desafio.

Por último, as responsabilidades da camada de maior alto nível são dos módulos raiz do pacote "*src*". São eles que vão monitorar e controlar os estados de execução do robô, seja em casos de testes ou em partidas reais, assim como realizar a integração entre as partes abstraídas nas camadas inferiores em uma linha de execução que atinja os objetivos do robô de forma autônoma. A ideia é que não haja implementação de nenhuma solução específica, apenas deliberação de tarefas, controle e coordenação das partes.

Essa hierarquia não é necessariamente seguida com muita rigidez, mas serve como um bom direcionamento para uma melhor organização do código como um todo, auxiliando no desenvolvimento e manutenção de funcionalidades complexas.

## 3.2 Comunicação entre controladores

A motivação para uso de mais de um controlador no robô vem das limitações do número de portas para sensores e motores, e como elas impediriam a atuação satisfatória de um robô neste desafio. Apenas para controle das 4 rodas omnidirecionais, necessárias para que os movimentos em todas as direções do plano do mapa da competição sejam possíveis, já seriam utilizadas todas as portas de atuadores. Da mesma forma, apontados para o chão estão dispostos 4 sensores de cor, que já ocupam todas as portas para sensores de um controlador. Assim, esses componentes estão conectados em um primeiro *brick*, e os demais ao segundo controlador utilizado. São eles: sensores ultrassônicos frontal e traseiro, para detecção de obstáculos; sensores envolvidos na detecção e identificação de passageiros, sendo o infravermelho e o *HiTechnic*; dois motores para acionamento da garra.

Eles foram dispostos dessa forma pensando em agrupar em um mesmo controlador

os sensores e motores mais envolvidos nas mesmas tarefas, a fim de mitigar problemas de atrasos e temporização causados pela comunicação entre dois controladores. Portanto, no primeiro *brick* estão os componentes mais utilizados nas tarefas de movimentação no cenário e navegação, enquanto no segundo estão os sensores e atuadores da garra e os sensores para detecção de obstáculos. Esses últimos são utilizados nas tarefas de movimentação e navegação, mas não tanto quanto os outros sensores de cor.

Com relação à comunicação entre os dois *bricks*, foi estudada a possibilidade de integração em estilo *daisy-chain* via cabo USB, o que é possível no EV3 (DAISY..., 2018). Porém, não existe documentação de uma solução neste formato que seja compatível com o sistema operacional do "*ev3dev*", apenas com o sistema proprietário da LEGO. Assim, a via de comunicação entre os EV3 foi implementada via mensagens *Bluetooth* (Figura 16).

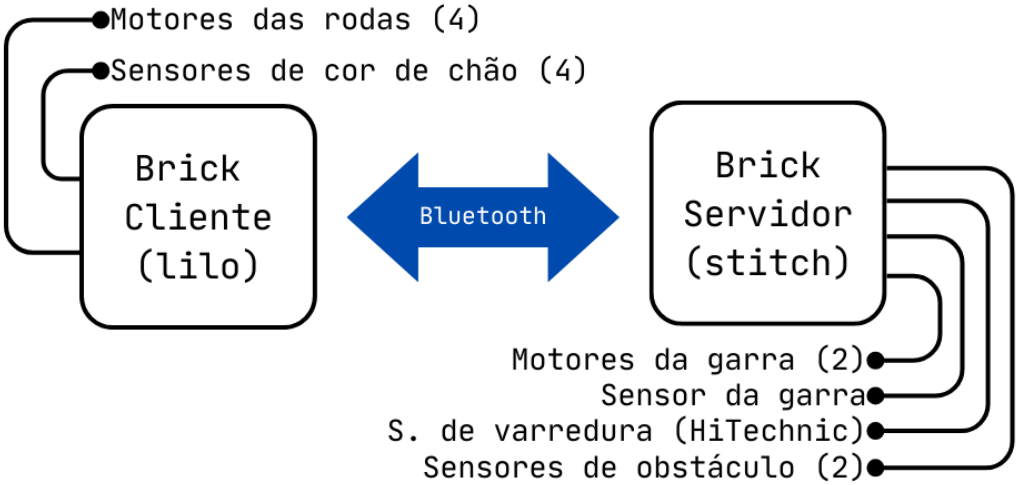


Figura 16 – Arquitetura de distribuição dos componentes, e da comunicação entre eles. Fonte: Autoral.

A biblioteca Pybricks fornece funcionalidades básicas de estabelecimento de conexão e envio de mensagens via *Bluetooth*, com um dos *bricks* atuando como cliente e o outro como servidor. Essas funcionalidades, assim como capacidades de codificação e decodificação de dados que promovessem maior versatilidade na troca de informações entre os controladores, foram aproveitadas e estendidas numa classe "*wrapper*" denominada "*Bluetooth*", dentro do módulo de *networking* do pacote *core*. Isso foi feito para abstrair os detalhes do controle da conexão *Bluetooth* do resto do código.

O projeto é organizado de forma que ambos os *bricks* rodem utilizando o mesmo código fonte, que é capaz de identificar em qual *host* está sendo executado, e direciona as tarefas de cada controlador. Isso é feito através do *hostname* do *ev3dev*, que foi alterado em cada um deles para promover essa identificação. O *brick* cliente é o responsável pela movimentação do robô no cenário, enquanto o *brick* servidor é o responsável pela garra e pelos sensores auxiliares. Essa designação foi pensada justamente porque o segundo EV3

foi programado para se comportar como um servidor, esperando requisições do cliente, processando-as e retornando possíveis respostas. Dessa forma, a deliberação sobre o uso dos recursos do servidor fica na responsabilidade do cliente.

Em alguns casos de uso, porém, faz-se necessário que haja uma transmissão contínua de dados entre os controladores, como quando alguns algoritmos executados em um deles exigem dados de sensores conectados ao outro. Para esses momentos é que foi estruturado um modelo de requisições no estilo "solicitação de transmissão". Em determinadas requisições, o *brick* servidor ficará emitindo os dados solicitados pelo cliente até que este envie outra mensagem, solicitando o encerramento da transmissão. Os dois modelos de requisição estão ilustrados na Figura 17.

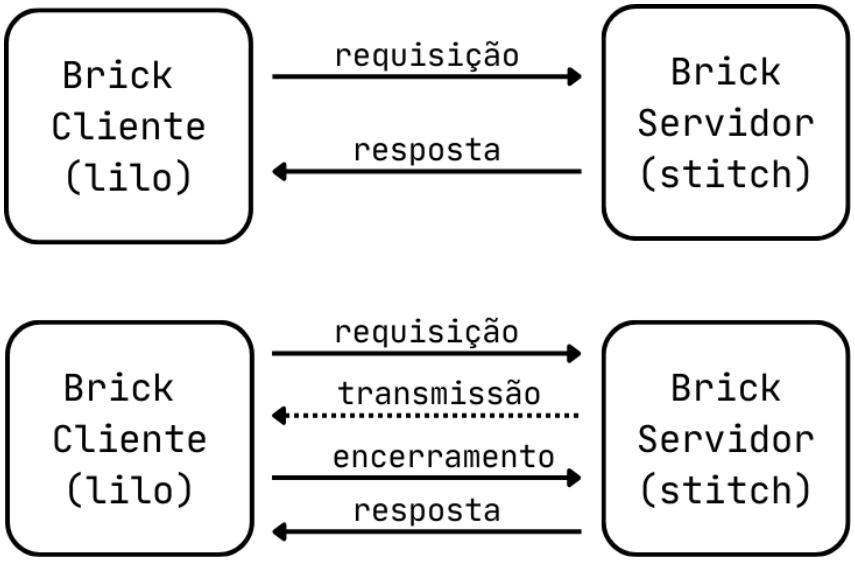


Figura 17 – Acima, o modelo de requisição-resposta; abaixo, o modelo de requisição para transmissão contínua de dados. Fonte: Autoral.

O processamento das requisições codificadas está implementado no módulo "*gripper_server*", do pacote *domain*. São elas:

- *CLAW_LOW*: requisição de movimentação da garra para baixo;
- *CLAW_HIGH*: requisição de movimentação da garra para cima;
- *CLAW_MID*: requisição de movimentação da garra para uma altura intermediária;
- *CLAW_OPEN*: requisição de abertura da garra;
- *CLAW_CLOSE*: requisição de fechamento da garra;
- *ULTRA_FRONT*: requisição de transmissão dos valores de distância lidos pelo sensor ultrassônico dianteiro;

- *ULTRA_BACK*: requisição de transmissão dos valores de distância lidos pelo sensor ultrassônico traseiro;
- *INFRA_CLAW*: requisição de transmissão dos valores de distância lidos pelo sensor infravermelho da garra;
- *COLOR_SIDE*: requisição de transmissão das leituras de cor do sensor *HiTechnic*, na lateral do robô.

Os *hostnames* utilizados são os codinomes utilizados pelos membros da equipe para se referir aos *bricks*: "lilo" para o cliente e "stitch" para o servidor (inspirados na dupla de personagens Lilo & Stitch, de filmes de animação infantis).

## 3.3 Funcionamento Geral

Definindo a arquitetura e organização do sistema, pode-se apresentar seu fluxo de funcionamento. Este está estruturado conforme ilustrado pelo fluxograma na Figura 18.

O primeiro passo é a identificação de acordo com o *hostname*, e em seguida acontece a designação dos fluxos de funcionamento individuais de cada controlador. Depois é estabelecida a conexão *Bluetooth*, com o servidor tendo aguardado a solicitação de conexão ativamente realizada pelo cliente. A partir daí, o *brick* stitch entra no ciclo de espera e tratamento de requisições, enquanto o *brick* lilo prossegue para o ciclo de atuação.

Como o robô pode iniciar em qualquer posição da área branca, apontado para qualquer um dos lados do mapa, o primeiro passo é a realização da rotina de localização. A rotina implementada inicialmente é inspirada na ideia de uma máquina de estados, cujos comportamentos guiam o robô pelos marcos reconhecidos no mapa, à medida que ele vai se movimentando, inclusive desviando de possíveis obstáculos. Uma escolha de projeto feita é que essa rotina deve sempre terminar posicionando o robô na mesma posição, na zona de embarque. Isso porque o próximo passo é a varredura e embarque do primeiro passageiro, assim como sua identificação (tamanho e cor).

A cor representativa do passageiro é identificada pelo *HiTechnic*, que é o responsável pela leitura lateral dos passageiros presentes enquanto o robô se move para frente, paralelamente à zona de embarque. A varredura é feita a fim de que o robô ignore os passageiros que não solicitaram transporte (tubos brancos) e pegue o primeiro tubo de cor válida reconhecido. Já o tamanho do passageiro, ou seja, se é um adulto ou uma criança, é deduzido a partir da leitura do sensor infravermelho da garra. O posicionamento desse sensor é tal que, dada a presença de um tubo na garra levantada, o sensor não identificará nenhum objeto próximo se for uma criança, mas, se for um adulto, parte do tubo ficará na frente da onda emitida pelo sensor, permitindo sua identificação.

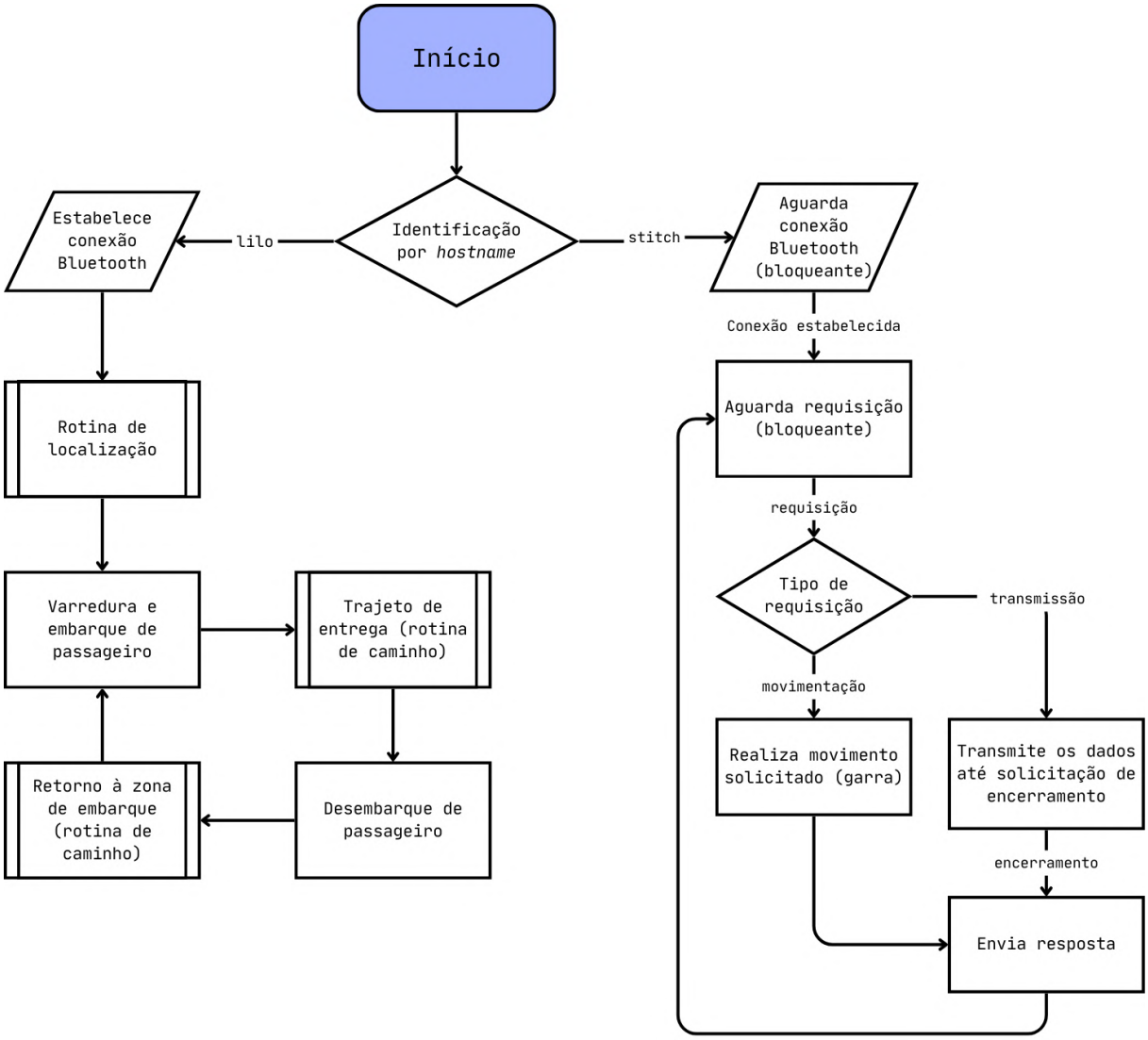


Figura 18 – Fluxograma de funcionamento geral do sistema. Fonte: Autoral.

Com a cor e tamanho do tubo identificados, o robô já consegue inferir o destino do passageiro. Aí então é chamada a rotina de caminho, que recebe a posição atual do robô (convencionalmente a extremidade sul da zona de embarque) e a posição alvo, ou seja, o local de destino do passageiro. Chegando no destino, enquanto possivelmente desvia dos obstáculos que obstruem as vias, o robô faz o desembarque e chama novamente a rotina de caminho, agora para retornar à zona de embarque e pegar o próximo passageiro. O ciclo continua indefinidamente, até que o robô seja manualmente encerrado, no final da partida ou por alguma interrupção.

## 3.4 Movimentação

A movimentação do robô foi implementada baseando-se na discretização das direções possíveis de condução omnidirecional. Isso foi feito visando uma menor complexidade

de modelagem cinemática, facilitando a implementação de funcionalidades de controle para a movimentação do robô. Porém, com esta abordagem, o potencial completo de eficiência levantado pela liberdade de movimentação omnidirecional não é alcançado. Pode ser interessante, em trabalhos futuros, projetar sistemas de controle que utilizem em maior profundidade a modelagem cinemática do sistema de 4 rodas omnidirecionais.

As direções discretas implementadas se dispõem como ilustrado na Figura 19.

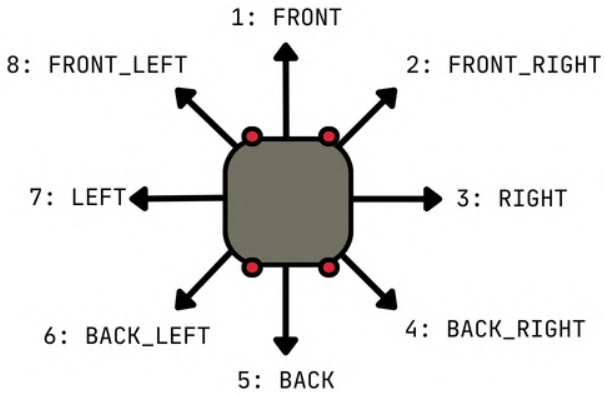


Figura 19 – Direções discretas implementadas para movimentação do robô, visto de cima. Fonte: Autoral.

Em código, as direções foram mapeadas para sinais de velocidade em cada um dos quatro motores, através de uma matriz. Por exemplo, por conta da disposição física dos motores, para o robô se mover para frente, os motores dianteiros devem se mover no seu sentido contrário, e os traseiros no seu sentido normal. Isso significa que, na ordem dianteiro esquerdo, dianteiro direito, traseiro esquerdo e traseiro direito, os sinais das velocidades passadas aos motores no controle de movimentação devem ser -1, -1, 1 e 1, respectivamente. As outras direções discretizadas são mapeadas seguindo a mesma lógica.

Várias funções básicas de movimentação são implementadas no módulo *robot*, dentro do pacote *core*, todas utilizando controle PID. Destacam-se funções de condução em linha reta (com minimização do erro de diferenças de velocidade entre os motores), curvas em torno do próprio eixo e seguimento de linha com sensor de cor. Essas funções são parametrizadas o suficiente para que possam ser reutilizadas em vários cenários de aplicação.

Uma função implementada que tem papel fundamental para contribuir com a correção de erros de navegação e odometria é a denominada "*align*", ou função de alinhamento. Ela usa o fato de que no mapa existem várias marcações de linhas retas no chão para propor uma rotina em que o robô utiliza essas linhas, junto com os sensores de cor devidamente posicionados, para corrigir sua orientação, assim como aproveita as marcações como "*ground-truth*" de posicionamento, não confiando apenas nas leituras odométricas. Essa funcionalidade, da forma como ela foi implementada, é possível através

da utilização de dois sensores de cor apontados para o chão, devidamente espaçados um em cada lado do eixo de rotação do robô, preferencialmente em posições simétricas e anteriores às rodas. O robô utilizado tem 4 sensores de cor, posicionados nas extremidades de suas laterais, justamente para poder se alinhar nas linhas do chão em qualquer direção. A Figura 20 ilustra o funcionamento do alinhamento dianteiro, por exemplo.

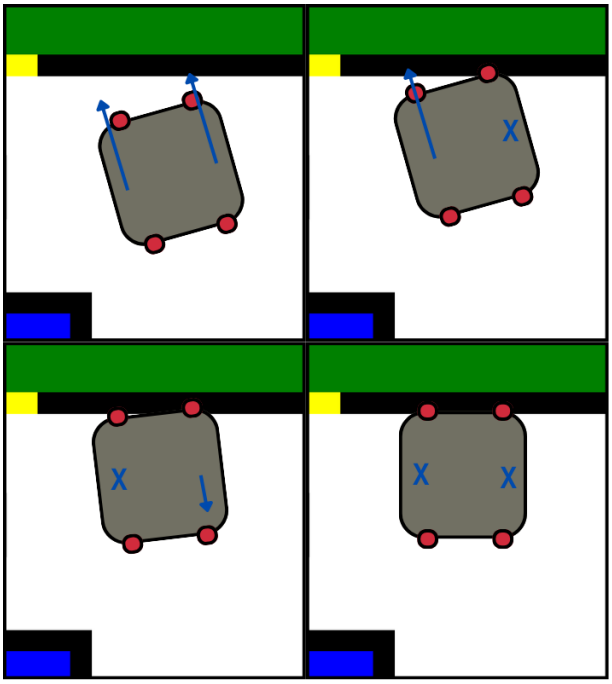

Figura 20 – Ilustração do princício de funcionamento da rotina de alinhamento. Fonte: Autoral.

No caso mostrado, o robô se move para frente, com orientação levemente inclinada devido a imprecisões acumuladas com o movimento dos motores. Na figura estão destacados os 4 sensores de cor, em vermelho, e os vetores representantes das resultantes de velocidade aplicadas nos motores direitos e esquerdos. Quando um sensor encontra uma mudança de cor, como a linha preta no exemplo, ele calcula um valor de reflexão alvo a ser almejado pelo sensor de cor. Por exemplo, se na área branca o sensor de cor lê 80% de reflexão e na linha preta ele lê 8%, ele definirá como alvo o valor de 44% (média aritmética) de reflexão. Esse valor é calculado para que as velocidades direita e esquerda do robô sejam controladas a fim de que ambos os sensores atinjam suas respectivas leituras alvo, sendo posicionados no limite entre a área branca e a linha preta.

Quando cada sensor identifica a troca de cor, a velocidade dos motores daquele lado passa a ser determinada pelo controle PID baseado na diferença entre o valor lido pelo sensor de cor e o valor alvo calculado. Isso resulta em que, inicialmente, o primeiro sensor a encontrar a linha cause uma aparente parada nos motores daquele lado, como ilustrado no segundo painel da figura. Com a movimentação do lado oposto continuando até que o segundo sensor atinja a mudança de cor, o primeiro sensor provavelmente sairá

da posição ideal. Assim, nesse caso, o controle PID poderá causar uma força contrária no primeiro lado, como é o caso do lado direito no terceiro painel da figura, posicionando assim ambos os sensores em posições que atinjam a leitura alvo. Como essas posições são no limite da linha preta, o robô terá uma orientação resultante mais alinhada com as direções do mapa do que inicialmente, o que é muito útil para cobrir erros de controle de velocidade, orientação e odometria, minimizando seus impactos no desempenho do robô.

Como no mapa da competição existem linhas amarelas, azuis, vermelhas e pretas, e experimentalmente foi observado que as leituras de reflexão são muito diferentes em cada uma delas, os alinhamentos foram feitos com base no componente azul das leituras RGB dos sensores. Essa abordagem apresentou resultados consistentes no alinhamento nas linhas, independentemente das cores das mesmas, o que não acontece utilizando o modo de reflexão padrão.

Outro aspecto a ser pontuado é que a possível presença de lombadas limita o uso indiscriminado da movimentação em qualquer direção relativa ao robô no mapa. Isso porque, por conta da geometria das rodas *mecanum* e do robô, ele não consegue atravessar as lombadas se movendo lateralmente, apenas de frente ou de ré. Na prática, isso significa que o robô deve se movimentar de forma a nunca andar lateralmente por uma posição que possa conter uma lombada, o que diminui a versatilidade omnidirecional. Ainda assim, a possibilidade de alinhamento lateral pode acrescentar uma boa eficiência e resiliência ao sistema de navegação do robô.

## 3.5 Localização

Na estratégia projetada junto à equipe para o mapa do desafio, a rotina de localização inicial do robô funciona como uma máquina de estados. Baseando-se na disposição das marcações de linhas no chão do mapa, à medida que o robô vai se movendo e reconhecendo marcos, seus comportamentos programados vão guiando-o até uma posição conhecida, inclusive desviando de obstáculos. Como escolha de projeto, a posição para isso é o extremo sul da zona de embarque, local ideal para início da varredura dos passageiros com o sensor de cor lateral apontado para o lado direito do robô. A máquina de estados desenvolvida segue o modelo ilustrado na Figura 21.

No primeiro estado (referido como “LOC” na figura), é realizado um procedimento arbitrário para direcionar o robô, independente de onde ele esteja inicialmente, para as extremidades laterais do mapa. Isso é feito a partir de leituras nas 4 direções cardeais primárias nas proximidades da posição inicial do robô. Da seguinte forma: o robô se move para frente até detectar uma parede de alguma cor ou entender que naquela direção não há nenhuma parede próxima, portanto a cor detectada é branca; se a cor detectada pelo robô for azul ou vermelha, ele sabe que já está em uma extremidade lateral (vermelhas são

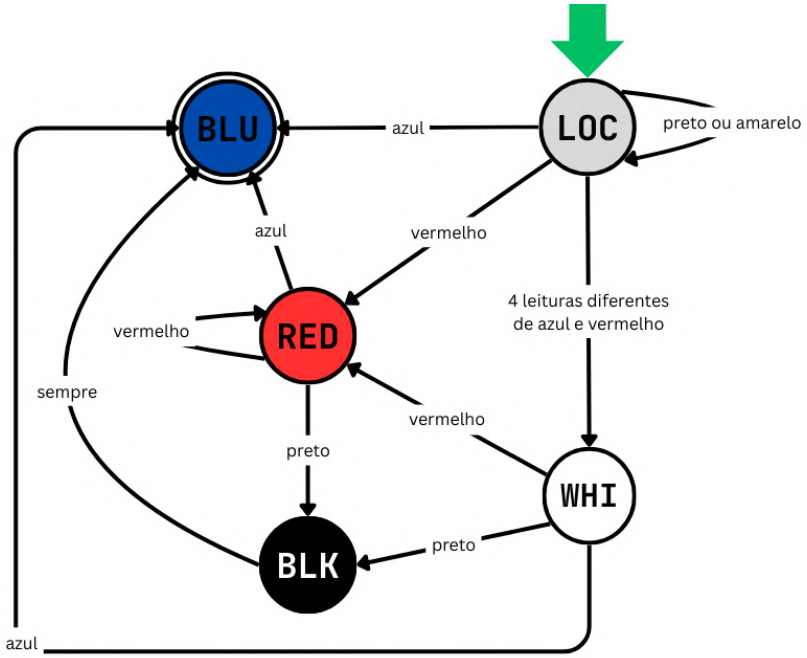


Figura 21 – Ideia de máquina de estados utilizada na localização inicial. Fonte: Autoral.

as extremidades laterais do mapa, azul é a extremidade ao centro, na zona de embarque), e já muda de estado de acordo com a cor encontrada (estado "BLU" para azul e "RED" para vermelho). Lida uma primeira cor no máximo 30 centímetros à frente do robô, ele vira à direita e repete o processo, guardando as cores lidas em cada direção numa lista. A Figura 22 exemplifica quais 4 cores seriam lidas nas proximidades do robô em uma posição do mapa, em ordem, dada uma orientação inicial. Caso o robô encontre nos quatro lados 4 leituras diferentes de azul ou vermelho, necessariamente haverá leituras de branco na lista criada. O robô então escolhe a primeira leitura de branco encontrada, se vira para a direção em que ela ocorreu e muda de estado, para "WHI".

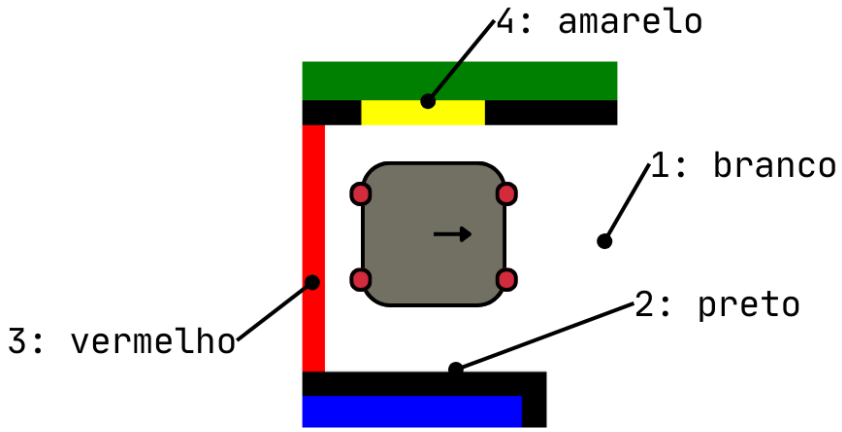


Figura 22 – Cores lidas nas proximidades do robô durante a rotina de localização em uma posição do mapa, em ordem, dada uma orientação inicial. Fonte: Autoral.

Se o robô estiver no estado "WHI", ele se move para frente até encontrar a próxima

marcação colorida ou um obstáculo. No caso de obstáculo, ele vira à direita e continua o processo. No caso de marcação colorida, as únicas possibilidades de cor são vermelho, preto e azul. Ao encontrar a linha, então, o robô muda para o estado correspondente à cor encontrada.

No estado "RED", saindo de uma posição de frente a uma linha vermelha, o robô primeiro dá uma ré a uma distância fixa para virar à direita no próximo cruzamento. E então segue reto, atento a possíveis obstáculos. Caso encontre, ele tenta realizar uma manobra para "trocar de rua", evitando o obstáculo e progredindo na mesma direção que estava. É possível encontrar um novo obstáculo, então o robô faz esse comportamento em loop, trocando o sentido das curvas a cada obstáculo detectado, para eventualmente chegar em uma linha azul, preta ou vermelha. No caso da linha vermelha, o mesmo estado é reiniciado, repetindo o comportamento até que se encontre outra cor. No caso das outras cores, é feita a mudança para o estado correspondente à cor encontrada.

No estado "BLK", as posições possíveis são próximas ao parque, na lateral do mapa oposta à zona de embarque. O robô então prossegue no sentido contrário, usando a tratativa de "troca de ruas" caso detecte obstáculos para eventualmente chegar à zona de embarque, transicionando para o estado "BLU".

O último estado, "BLU", é o estado em que a zona de embarque já foi encontrada. O robô então apenas manobra para uma posição fixa convencionada no extremo sul do mapa e corrige sua orientação para apontar para o norte. A rotina de localização termina, com o sistema tendo conhecimento da posição e orientação do robô.

## 3.6 Representação do Mapa e Planejamento de Caminho

Por escolha de projeto, o mapa da competição foi representado internamente no robô como um grafo direcionado, sendo os nós posições que representam a discretização das áreas do mesmo, e as arestas os caminhos que um robô poderia fazer (Figura 23). O grafo resultante é apresentado na Figura 24, com as cores dos nós representando os estabelecimentos correspondentes e a possível presença de obstáculos obstruindo a via naquela posição (nós cinza escuro).

Esse modelo foi construído seguindo a ideia das áreas delimitadas pelas paredes: os pontos agrupados que têm as "mesmas paredes próximas" nas quatro direções cardeais e a mesma cor de chão pertencem à mesma área. Essa ideia está ilustrada na Figura 25. O ponto representante daquela área do mapa é o localizado no centro da mesma.

O grafo foi feito com arestas direcionadas para que o algoritmo de busca de caminho não considere passar por dentro de estabelecimentos, o que não é permitido de acordo com as regras. Por exemplo, por mais que haja arestas entre os vértices 14 e 15 e os

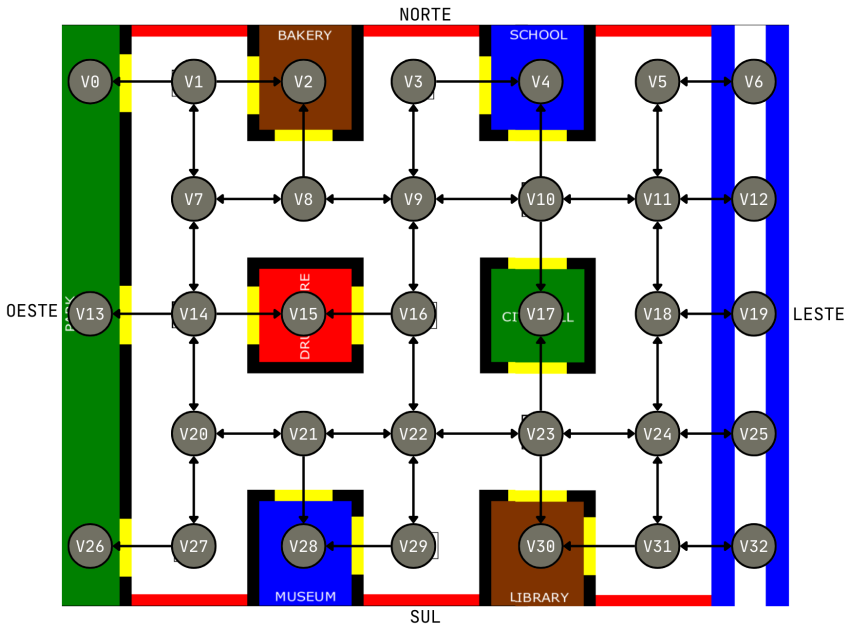


Figura 23 – Representação do mapa da competição como um grafo direcionado. Fonte: Autoral.

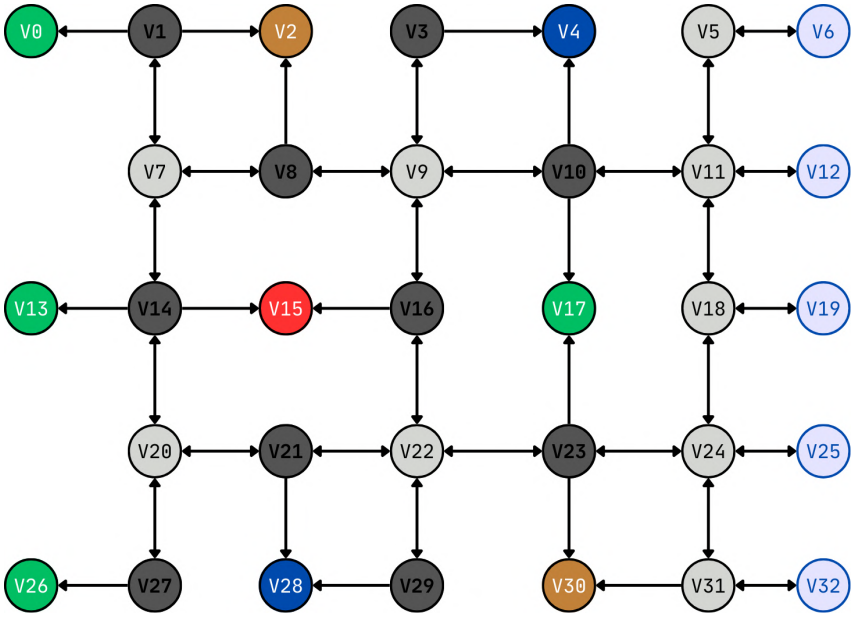


Figura 24 – Grafo resultante da representação do mapa, com cores correspondentes às áreas originais. Fonte: Autoral.

vértices 15 e 16 (ou seja, é possível chegar até o vértice 15, que representa a farmácia), não seria possível para o robô no vértice 14 chegar ao vértice 16 passando pelo vértice 15. Os vértices dos estabelecimentos são representados como sumidouros, ou seja, não têm arestas que partem deles. Isso porque existe a intenção de que o algoritmo de busca de caminho chegue nessas posições, mas não que trace rotas inválidas passando por elas. Em código, tratativas são feitas para que, após entregar corretamente um passageiro, o robô faça uma manobra fixa (ré) e se considere no vértice anterior ao do estabelecimento, na

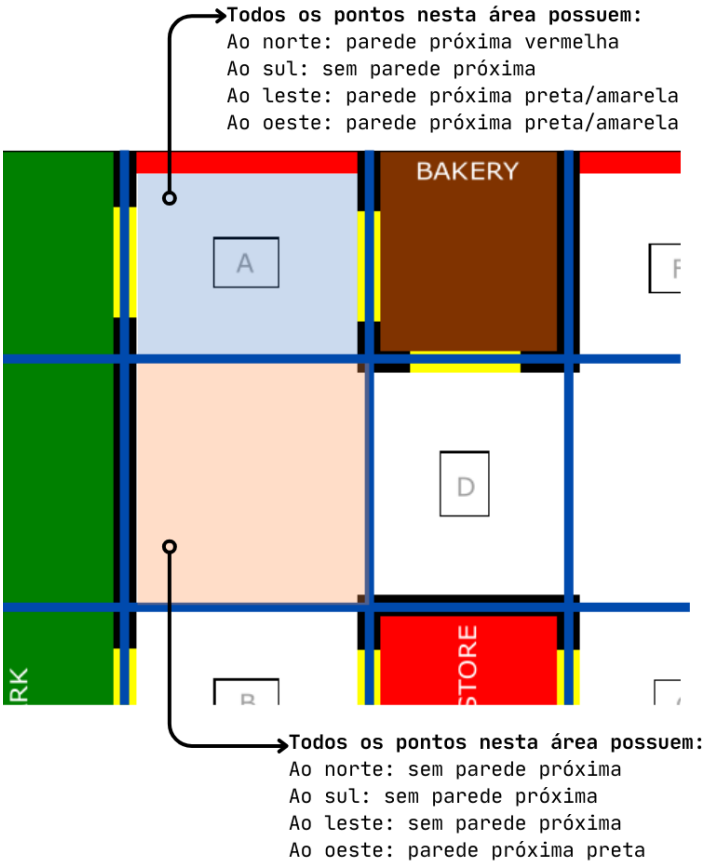


Figura 25 – Ideia de discretização das áreas na representação do mapa. Fonte: Autoral.

área branca do mapa.

A rotina de caminho implementada segue o fluxograma ilustrado na Figura 26. Esse fluxograma se relaciona ao de funcionamento geral do sistema (Figura 18) como sendo um detalhamento do que acontece dentro dos blocos “Trajeto de entrega” e “Retorno à zona de embarque”.

Seus parâmetros de entrada são a posição atual do robô e a posição alvo, à qual se deseja chegar. A rotina então submeterá essas variáveis para o planejador de caminho, que executará o algoritmo de Dijkstra no grafo para encontrar a melhor rota possível entre os pontos dados. As regras da competição garantem que, mesmo com a presença de obstáculos, sempre haverá pelo menos uma rota possível para a entrega dos passageiros.

O planejador de caminho retorna o caminho em si, uma lista dos vértices na ordem a serem visitados que compõe a rota da posição de origem até o destino desejado, e o conjunto de informações referentes às transições que devem ser aplicadas entre cada um dos pontos para que aconteça o seguimento de trajetória. Isso é possível por conta da escolha de implementação de atribuir às arestas uma tupla, representando qual a direção relativa entre os nós vizinhos e qual a distância entre as áreas que eles representam no mapa real, em centímetros. Um exemplo está ilustrado na Figura 27, que traz um subgrafo

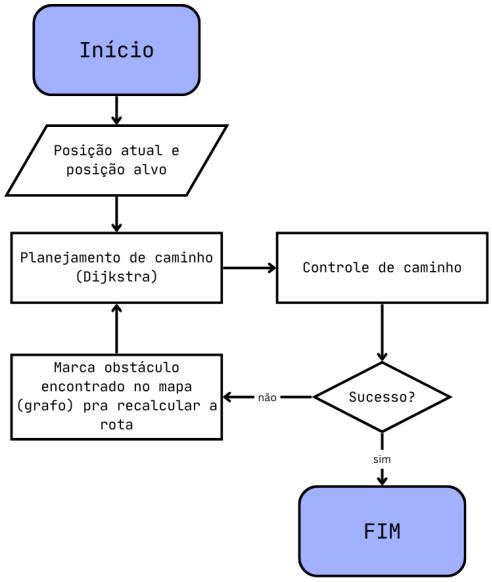


Figura 26 – Fluxograma da rotina de caminho. Fonte: Autoral.

representando uma seção do mapa.

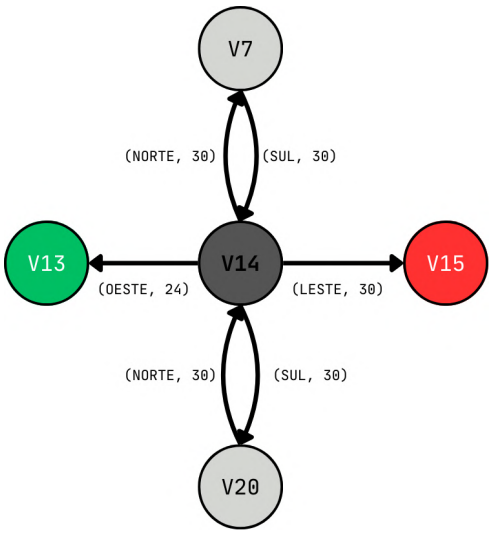


Figura 27 – Subgrafo do mapa destacando as informações associadas às arestas. Fonte: Autoral.

As informações retornadas pelo planejador são inseridas no algoritmo de controle de caminho, que a partir dela gera instruções a serem executadas pelos motores do robô a fim de percorrerem a rota traçada. Porém, no meio do trajeto, é possível que ocorra a detecção de um novo obstáculo. Nesse caso, o algoritmo de controle para e retorna falha. Isso é detectado na rotina de caminho, que marca o vértice onde o obstáculo foi encontrado como intransponível no grafo (suas arestas passam a ter peso "infinito"). O planejador é então chamado novamente para recalcular a rota, com o mesmo destino que a chamada anterior, mas agora a partir da posição em que o robô se encontra. Esse processo é repetido até que o robô chegue ao seu destino.

O algoritmo de controle de caminho implementado será detalhado na seção a seguir.

## 3.7 Seguidor de Caminho

O algoritmo implementado como seguidor de caminho segue o raciocínio ilustrado no fluxograma da Figura 28. Esse fluxograma se relaciona ao da rotina de caminho (Figura 26) como sendo um detalhamento do que acontece dentro do bloco "Controle de caminho".

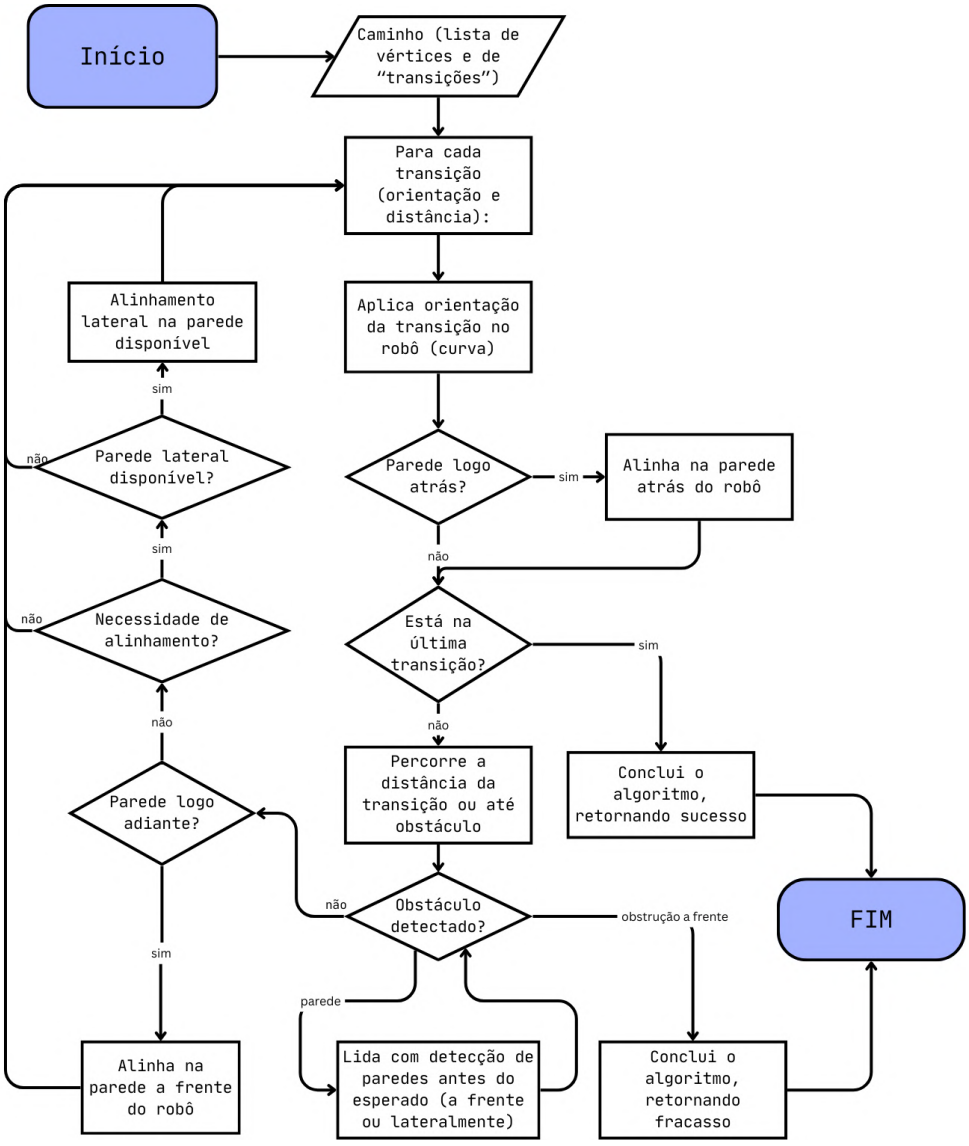


Figura 28 – Fluxograma do seguidor de caminho. Fonte: Autoral.

Como dito na seção anterior, ele recebe como parâmetros de entrada os vértices e as "transições" do caminho planejado na fase anterior, com o algoritmo de Dijkstra. O algoritmo do seguidor de caminho parte do princípio base de que a orientação do robô com relação ao mapa é conhecida, assim como sua posição. Portanto, para cada uma das transições, que são um par onde o primeiro item é uma orientação e o segundo

uma distância, o algoritmo corrigirá a orientação do robô para ser a mesma da transição, possivelmente executando uma curva, e moverá o robô para frente pela distância indicada.

A complexidade aumenta considerando a possível presença de obstáculos imprevistos. Caso o robô detecte algum obstáculo enquanto se move, ocorre uma parada. Da forma implementada, a noção de "obstáculo" inclui paredes de estabelecimentos (linhas pretas ou amarelas) que o robô encontrar em momentos inesperados, muito por conta de desvios de orientação e posição causados pelas imprecisões dos motores. Então é necessário identificar qual o "tipo" de obstáculo encontrado, e caso seja algum tipo de parede, lidar com isso antes de continuar o algoritmo. Caso realmente seja uma via obstruída, o algoritmo encerra com fracasso, sinalizando a necessidade de replanejamento de rota.

Assim, o algoritmo segue a ideia de executar as transições entre os pontos e lidar com possíveis erros que possam ser detectados, devido a imprecisões dos motores. Porém, como algumas rotas podem envolver múltiplas curvas e múltiplos obstáculos encontrados, se faz necessário um processo mais elaborado de cobertura de possíveis erros, utilizando como referência as marcações do mapa para que o robô possa se reajustar. Uma alternativa é o algoritmo de alinhamento em linhas já apresentado.

Aqui, ele pode ser integrado ao fluxo do seguidor de caminho para promover alinhamentos que reforcem posicionamentos e correções de orientação no robô, cobrindo erros e imprecisões odométricos. Assim, existem no algoritmo apresentado verificações de presença de parede: logo atrás do robô, depois dele fazer uma curva e antes de percorrer a distância da transição, e também logo adiante, após o término da movimentação da transição com não detecção de obstáculo. Porém, existem cenários em que essa tratativa ainda não é o suficiente, como por exemplo caminhos que exigem várias curvas e encontro com obstáculos em posições onde não estão existem nem paredes adiante nem paredes logo atrás. A possibilidade explorada nesses casos, graças à mobilidade omnidirecional do robô, é um alinhamento em paredes laterais. O algoritmo irá manter um registro da necessidade de alinhamentos. Toda vez que fizer uma curva ou esbarrar com uma parede essa necessidade aumenta, e na próxima vez que uma parede lateral estiver disponível o robô se alinhará nela. As tratativas de parede adiante e logo atrás continuam valendo, e também resetam o registro de necessidade de alinhamentos.

## 3.8 Percepção

A estratégia relacionada à percepção foi o treinamento de modelos classificadores para identificação de cores precisa e calibrável, dada a presença de variáveis como tonalidades variadas de cor e diversas condições de iluminação possíveis. A abordagem implementada foi a de árvores de decisão, escolhida por gerar um modelo de fácil integração ao código do robô, fácil interpretação e pela execução eficiente da classificação em

tempo de execução. Foi desenvolvido um programa para coleta de dados, a ser executado no robô, e um *script* para construção, treinamento e exportação dos modelos. Isso foi desenvolvido de forma que cada sensor tem um modelo próprio de classificação de cores, treinado no conjunto de dados gerado com suas leituras. Os modelos foram exportados como código Python gerado de forma procedimental, facilmente incorporáveis ao projeto. Além disso, foi escrita uma classe para encapsular um sensor de cor com um classificador personalizado, denominada "*DecisionColorSensor*" e incorporada no pacote *core*. Essa classe respeita as interfaces definidas pela classe "*ColorSensor*", da biblioteca Pybricks, de forma que as duas podem ser utilizadas de forma intercambiável por um programa usuário.

O programa para captura de dados é o módulo "*color_calibration.py*". Ele pergunta ao usuário, pela interface do *brick* EV3, quais sensores e cores serão calibradas, uma combinação por vez. A cada opção escolhida pelo usuário serão feitas 100 leituras de tuplas RGB, espaçadas por um intervalo de 100 milissegundos. O usuário fazendo a calibragem deve apontar o sensor de cor correspondente para a superfície da cor a ser usada como referência. Para calibragens mais robustas, faz sentido variar a posição do sensor com relação à superfície, ou as condições de iluminação durante a coleta de dados. Porém, é importante que essas variações não gerem impactos significativos o suficiente para desfavorecer o processo de classificação, gerando dados não confiáveis ou de má qualidade para treinamento. As leituras são armazenadas em arquivos na memória interna do controlador, organizados de acordo com o sensor e a cor calibradas. Esses arquivos podem posteriormente ser transferidos para um computador, onde será executado o *script* de treinamento com base nesses dados, e exportado um modelo de árvore específico para cada sensor calibrado.

O *script* utiliza as bibliotecas *scikit-learn*, *pandas* e *numpy* para carregar, processar os dados e treinar as árvores de decisão. Os atributos, os componentes das tuplas RGB, são utilizados para gerar uma classificação de cores discreta. São geradas árvores de decisão específicas para cada sensor, que classificam até 8 classes (7 cores e uma classe "*no color*", para casos em que o sensor não detecta nenhuma superfície). O fato de os sensores de cor do chão e o sensor lateral precisarem categorizar conjuntos de cores diferentes pode ser levado em consideração na calibragem. Por exemplo: o sensor responsável pela leitura dos tubos nunca precisará classificar as cores amarelo ou preto, porque não existem tubos dessas cores no desafio, diferente das marcações no chão lidas pelos outros sensores (existem linhas amarelas e pretas). Para que as árvores geradas não sejam muito profundas, o limitante do treinamento do modelo é até que o número de folhas seja igual ao número de cores que se deseja classificar.

Para validação das árvores geradas, foi adicionada ao *script*, com a biblioteca *graphviz*, a funcionalidade de geração de uma representação visual dos modelos gerados,

incluindo indicadores sobre eles. A Figura 29 mostra a visualização gerada para uma árvore de decisão de um dos sensores de cor do EV3, calibrados na posição de leitura do chão.

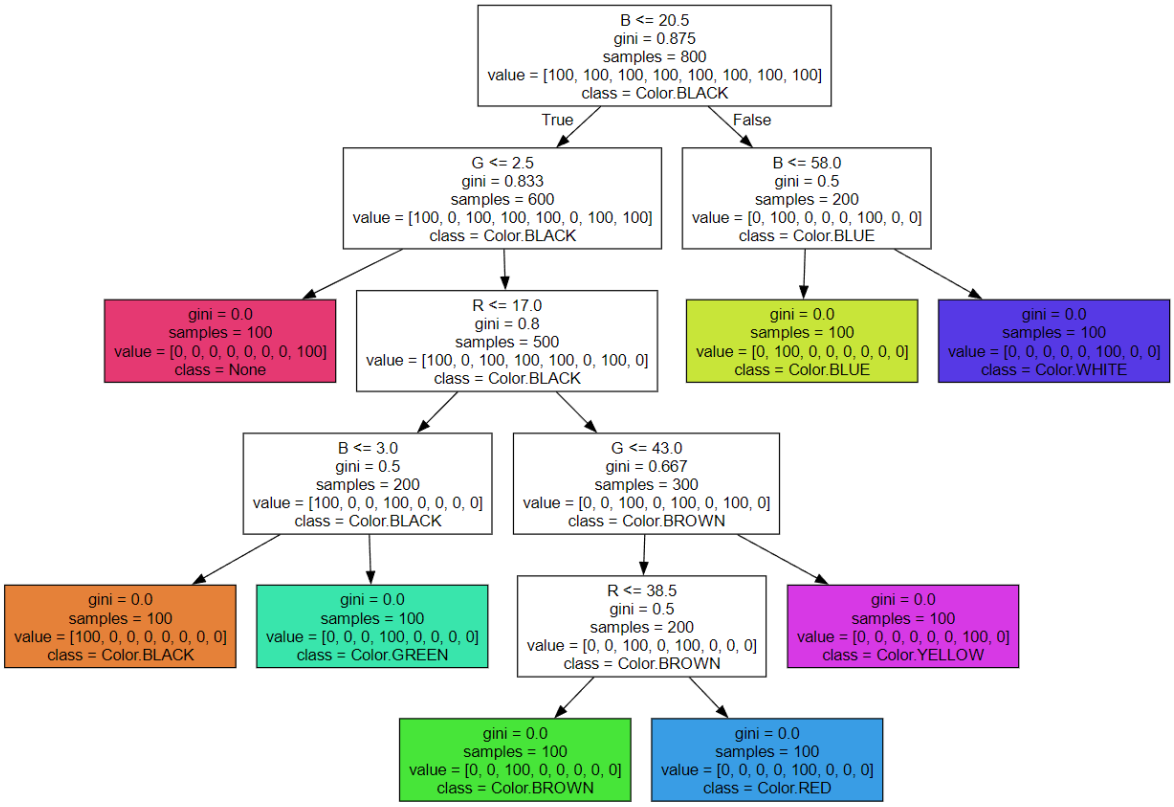


Figura 29 – Árvore de decisão para classificação de cores a partir de leituras RGB, gerada através de treinamento com dados coletados pelos sensores do EV3. Fonte: Autoral.

# 4 Resultados

Dado os trabalhos desenvolvidos neste projeto, este capítulo se dedica a discutir os resultados da implementação do software de navegação no robô para a competição. Essa seção traz aspectos da participação da equipe na competição em si, mas não se limita a eles e inclui fatores resultantes de trabalhos realizados pelo autor após a mesma.

## 4.1 Movimentação

Um experimento simples foi realizado para levantar dados sobre o resultado das implementações de controle de movimentação no robô. Ele foi programado para, apenas utilizando controle dos atuadores (PID) e odometria, realizar uma trajetória em linha reta de 150 cm (no mapa da competição, essa é a distância entre a zona de embarque e a delimitação do parque), repetidamente, indo até uma extremidade e retornando à posição inicial. O processo foi repetido por 10 iterações, e foram medidos os valores de distância de pontos fixos do robô às linhas de referência no mapa e as leituras odométricas dos atuadores, a fim de obter uma estimativa do erro de odometria (*drift*) presente no sistema. As medições foram feitas após os movimentos para frente (perto do parque) e após os movimentos para trás (retorno à posição inicial). A Figura 30 ilustra o experimento, destacando as posições onde os valores foram medidos, na primeira e na décima iterações.

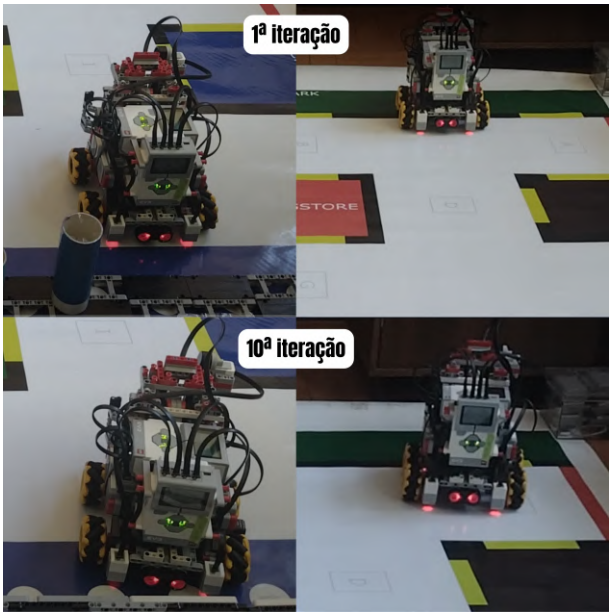


Figura 30 – Posições do robô na primeira e décima iteração do experimento realizado. Fonte: Arquivo Pessoal.

Já é possível perceber visualmente que existe erro: o robô não retorna precisamente

à mesma posição em nenhum dos extremos da trajetória após todas as iterações, ocorrendo desvios. Porém, esses desvios são de apenas alguns centímetros, mesmo após 10 iterações. Isso mostra que esses erros na movimentação, apesar de presentes, não são expressivos a ponto de prejudicarem a movimentação durante a resolução do desafio, situação onde nunca haverá a necessidade de tantas movimentações sucessivas a uma distância tão longa sem que o robô possa se alinhar ou corrigir sua posição usando outros pontos de referência, aliados ao controle de movimentação.

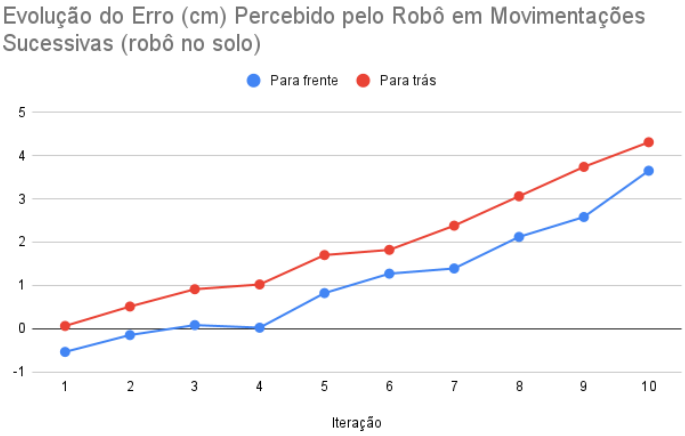


Figura 31 – Evolução do erro percebido pelo robô em movimentações sucessivas (robô no solo). Fonte: Autoral.

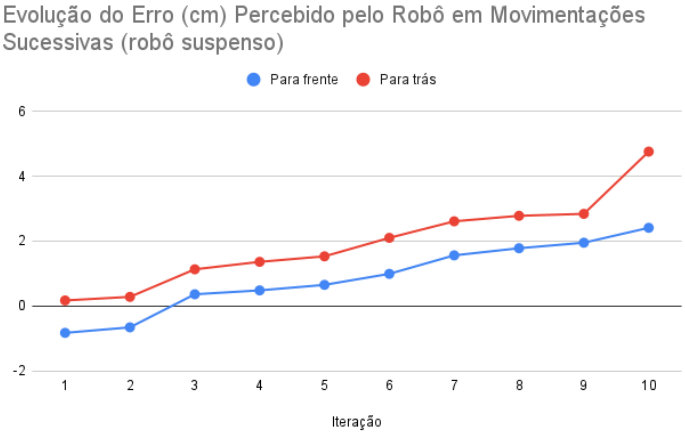


Figura 32 – Evolução do erro percebido pelo robô em movimentações sucessivas (robô suspenso). Fonte: Autoral.

Considerando apenas os valores de leituras odométricas do robô, os gráficos das Figuras 31 e 32 mostram a evolução do erro percebido, comparando as leituras feitas após a movimentação para frente e após a movimentação para trás, a cada iteração. Foram capturados tanto os valores da execução do robô no solo quanto suspenso (rodas sem contato com o chão), a fim de entender como a interação física com o ambiente impactaria

na percepção de erros por parte dos atuadores. Como os resultados foram muito parecidos em ambos os casos, não há uma diferença considerável.

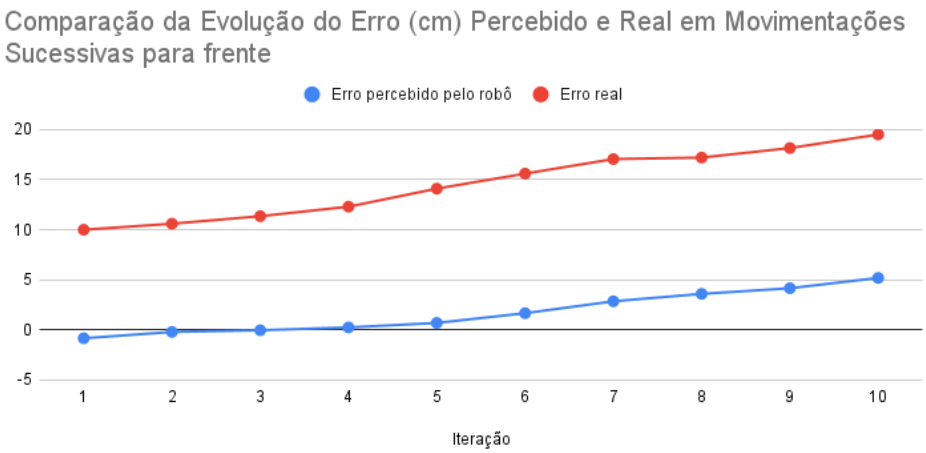


Figura 33 – Comparação da evolução do erro percebido e real em movimentações sucessivas para frente. Fonte: Autoral.

O gráfico da Figura 33 compara a evolução do erro em centímetros percebido pelo robô com o erro real, medido no mapa, isolando as medições após os movimentos para frente, ou seja, feitas perto do parque. É possível perceber que o erro real é bem maior que o erro percebido pelo robô através dos sensores odométricos, mas os dois crescem em ritmo semelhante a cada iteração. As curvas do gráfico aparecem com uma distância aproximada de 10 cm entre si, após o movimento de 150 cm, refletindo uma razão de aproximadamente 0.066 cm de diferença entre o erro percebido e o real (*drift*) para cada centímetro deslocado. Esse valor, em trabalhos futuros, poderia ser incorporado no algoritmo de controle a fim de ser compensado na movimentação dos motores, ou até possibilitar o uso de técnicas de estimativa de posição métricas.

O gráfico da Figura 34 desloca ambas as curvas verticalmente para que as primeiras leituras coincidam com a origem, a fim de comparar o crescimento dos dois erros (percebido e real). Fica evidente que, na verdade, a cada iteração o erro real cresce um pouco mais que o erro percebido pelos sensores odométricos do robô, mas ambos crescem de forma proporcional. Dessa forma, estratégias que objetivem a minimização do erro percebido também terão efeito sobre o erro real, proporcionando melhorias no controle de movimentação como um todo.

Portanto, o controle de movimentação do robô proporciona efetividade satisfatória para utilização na implementação de funcionalidades de mais alto nível. Os alinhamentos cobrem bem os erros de odometria, não houve problemas consideráveis relacionados a controle omnidirecional, e em geral o robô se move de forma sólida. Porém, instabilidades aparecem durante o desenvolvimento de algoritmos mais elaborados, como o caso do seguidor de caminho em cenários com várias curvas e/ou obstáculos. Os erros de posici-

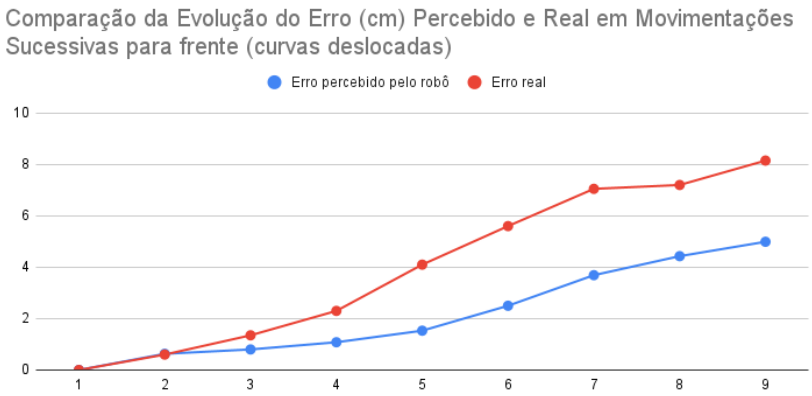


Figura 34 – Comparação da evolução do erro percebido e real em movimentações sucessivas para frente, com curvas deslocadas. Fonte: Autoral.

onamento para alinhamento lateral, por exemplo, detalhados na seção a seguir sobre o seguidor de caminho, poderiam ser minimizados com estratégias mais robustas de estimativa de posição que minimizem o *drift* observado. Talvez a implementação de um sistema métrico possibilitaria o uso de estratégias mais elaboradas de controle cinemático, localização e de caminho. Uma via possível é a de um sistema híbrido, com a localização baseada em marcos, por exemplo, e o controle de caminho com estimativa de pose métrica aliada a outras técnicas.

## 4.2 Localização

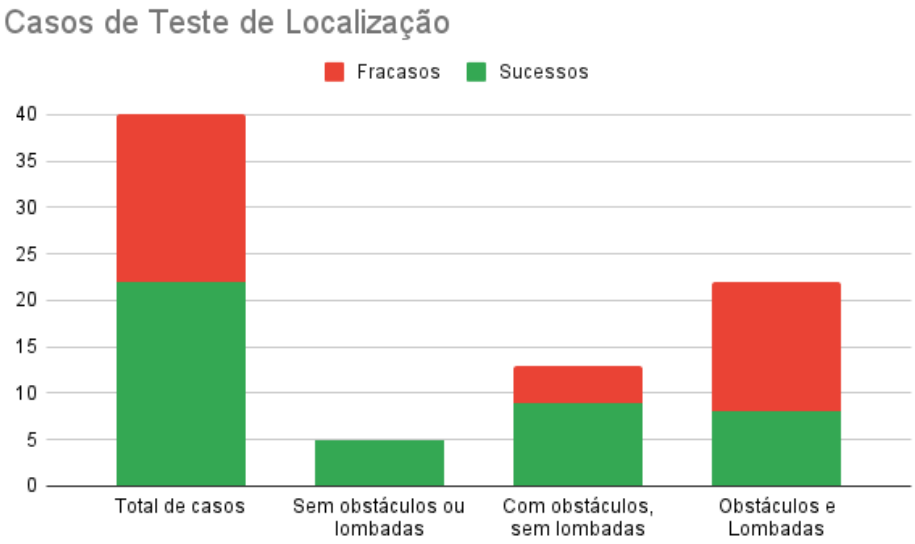


Figura 35 – Casos de teste de localização, com a relação entre a presença de obstáculos ou lombadas, e fracasos e sucessos. Fonte: Autoral.

Os testes e partidas documentados, que contêm diversos casos aleatórios de disposição do mapa, foram analisados com relação à rotina de localização. O gráfico da Figura 35 compila os resultados encontrados, mostrando a relação de sucessos e fracassos nos casos totais, e com relação à presença de obstáculos e lombadas.

Dessa forma, é possível concluir que a rotina de localização funciona de forma sólida para casos intermediários, em que o robô encontra poucos obstáculos e muda pouco entre os estados de controle desta etapa durante a execução. Casos mais complexos, principalmente cenários em que o robô encontra dois ou três obstáculos antes de chegar na zona de embarque, talvez até passando por lombadas, são mais instáveis e é frequente que a rotina de localização não atinja sucesso.

Mais testes em cenários diferentes podem ser executados, a fim de entender quais cenários estão mais prejudicados, e tentar traçar soluções que tornem a rotina mais robusta. Além disso, pode-se estudar como conceitos de filtros de partículas podem ser aplicados a fim de enriquecer a estratégia de localização. Talvez seja possível obter uma estimativa da posição do robô antes de ele precisar chegar na zona de embarque, o que poderia ser mais eficiente.

## 4.3 Seguidor de caminho

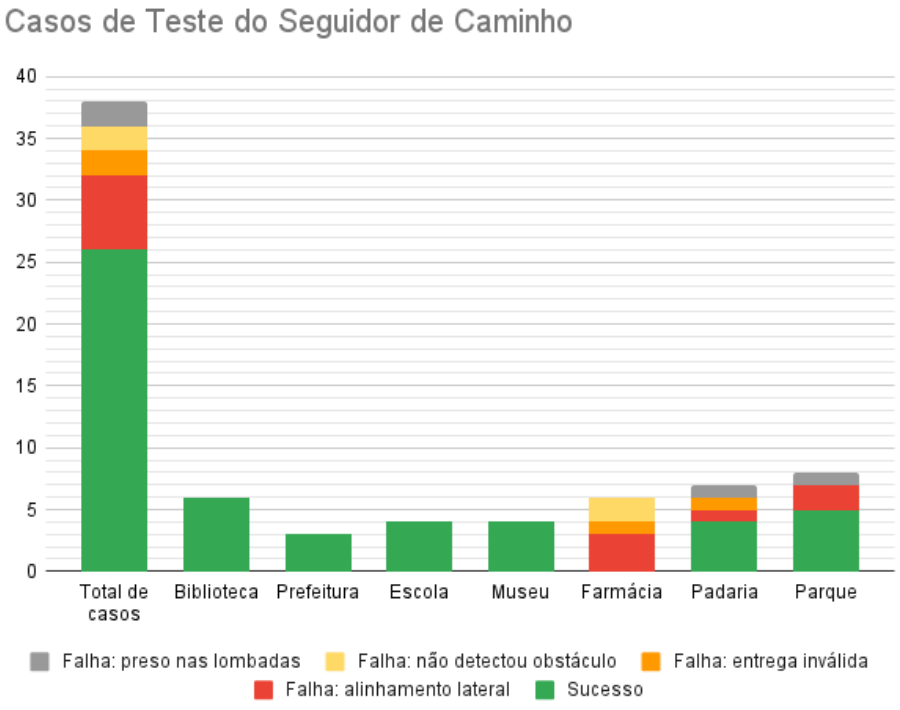


Figura 36 – Casos de teste do seguidor de caminho, com a relação de fracasos e sucessos por estabelecimento e motivos de falhas. Fonte: Autoral.

O seguidor de caminho funciona bem para caminhos curtos e médios, mas começa a apresentar instabilidade quando o caminho envolve grandes distâncias ou uma grande quantidade de curvas (farmácia, padaria, parque). Como é nessa rotina que o robô passa a maior parte da execução, esse é um dos maiores pontos a ser melhorado na implementação. O gráfico da Figura 36 compila os resultados do seguidor de caminho nos casos aleatórios documentados, segmentando por casos de entrega (estabelecimentos do mapa).

As partes problemáticas sendo: a implementação do alinhamento lateral, as tratativas para detecção de paredes antes do esperado e lombadas. É muito frequente que o robô comece o alinhamento lateral numa posição em que não conseguiria se alinhar, se prendendo nas bordas dos estabelecimentos e, a partir daí, saindo da linha de execução. As Figuras 37 e 38 mostram exemplos de alinhamentos laterais executados em posições indevidas, ocasionando esse erro. Na farmácia, por exemplo, esse tipo de erro aparece na maioria dos testes documentados, por conta da localização da mesma no centro do mapa, exigindo várias curvas e deslocamentos, o que, aliado à presença de apenas paredes curtas para alinhamento no trajeto (ao invés de longas, como nas extremidades do mapa), dificulta o trajeto.

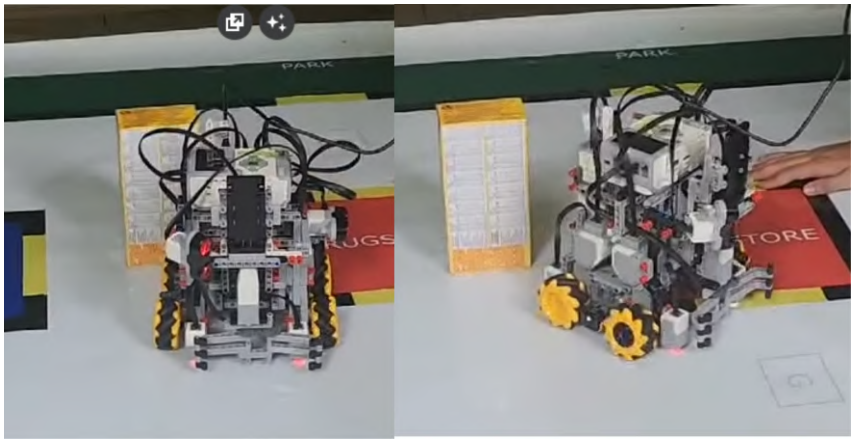


Figura 37 – Exemplo de erro no alinhamento lateral do seguidor de caminho nos treinamentos antes da competição. Fonte: Arquivo Pessoal.

Figura 38 – Exemplo de erro no alinhamento lateral do seguidor de caminho em uma partida na competição. Fonte: Arquivo Pessoal.

Com relação à capacidade de detecção de obstáculos e recálculo de rotas, o seguidor de caminho se mostra robusto nos testes realizados. A Figura 39 mostra um exemplo de detecção de obstáculo e recálculo de rota funcionando nos primeiros testes feitos com o robô.


Figura 39 – Exemplo de detecção de obstáculo e recalculo de rota com sucesso, já no início do desenvolvimento do robô. Fonte: Arquivo Pessoal.

## 4.4 Percepção

A abordagem utilizando árvores de decisão criadas a partir do treinamento nos dados de leituras de cores dos próprios sensores se mostrou muito sólida para lidar com o problema da calibragem de cores durante a competição (Figura 40). A visualização gráfica possibilitou a rápida identificação de configurações erradas feitas por engano, além de ajudar a evidenciar quais sensores ou cores poderiam ter classificações mais instáveis.

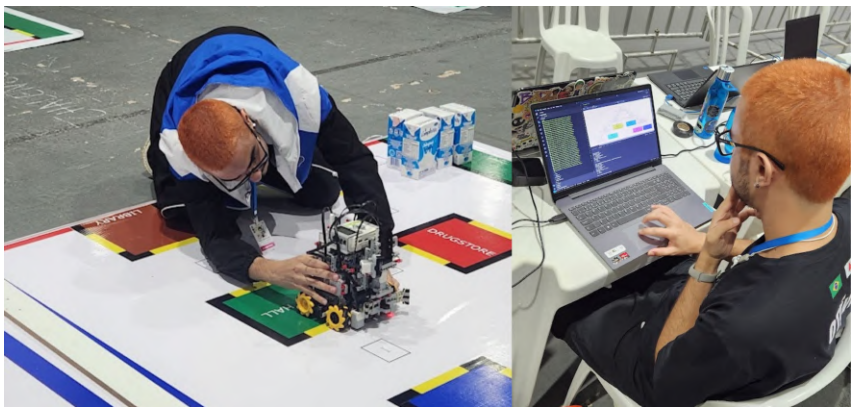

Figura 40 – Coleta de dados e treinamento de modelo de árvore de decisão no mapa da RCB *Challenge* 2024. Fonte: Arquivo Pessoal.

Durante as partidas, a leitura de cores foi constantemente monitorada através da interface do EV3, e erros de leituras em alguns casos foram resolvidos com recalibragem de um único sensor em uma ou duas cores mais problemáticas.

A integração automática do código exportado das árvores de decisão para o código fonte do projeto foi algo que promoveu grande satisfação aos desenvolvedores, diminuindo

o atrito na realização de novas calibrações. Além disso, o uso de variáveis em arquivos dedicados a constantes para controlar qual das calibrações implementadas utilizar em execução também se mostrou vantajoso, permitindo que a equipe trabalhasse no próprio mapa, no mapa de testes da competição e no mapa das partidas oficiais de forma contínua, sem necessidade de constante manutenção.

## 4.5 Outras Partidas e Testes

Nas imagens de uma das partidas da RCB *Challenge* 2024 é possível perceber uma versão anterior do robô tendo bons resultados e realizando feitos, apesar de falhas superadas ainda estarem presentes, e durante a partida ocorrerem vários reinícios. Os pontos de destaque estão trazidos na Figura 41, na ordem que aconteceram na partida, mas não necessariamente em uma única execução.

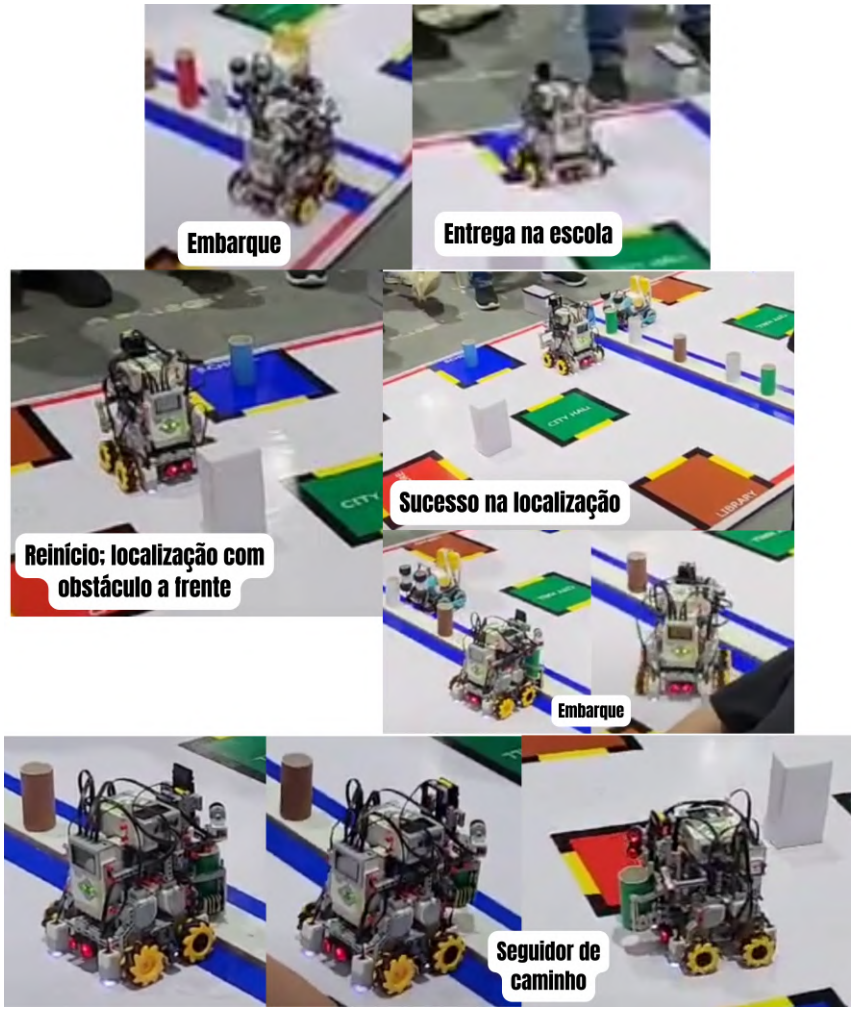


Figura 41 – Momentos de destaque de uma das partidas da RCB *Challenge* 2024. Fonte: Arquivo Pessoal.

Nas partidas mais avançadas da competição os cenários foram ficando mais elabo-

rados, colocando os robôs em testes de performance. A maior presença de lombadas, por exemplo, levantou alguns questionamentos importantes. O robô consegue atravessá-las, mas teve dificuldade em momentos durante a rotina de localização em que não estava com a orientação tão alinhada, ou acabou preso em cima de uma lombada por conta da rotina de alinhamento lateral ter sido ativada em cima dela (Figura 42).

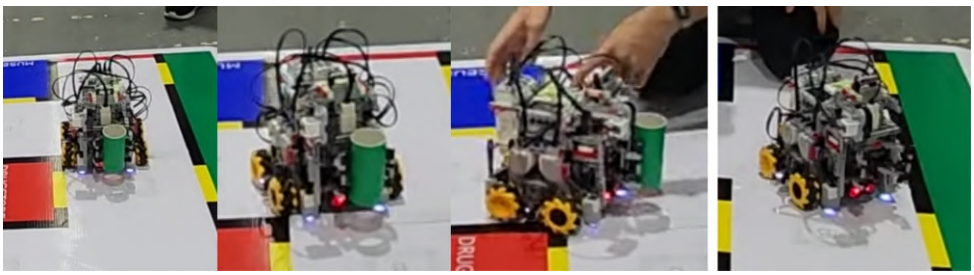

Figura 42 – Falha do alinhamento lateral em cima de uma lombada, e robô preso. Fonte: Arquivo Pessoal.

Em outros testes, com cenário simples de localização e as entregas sendo na ordem dos mais próximos à zona de embarque, os resultados foram satisfatórios (Figura 43). O robô conseguiu fazer entregas consecutivas na escola, prefeitura, biblioteca e padaria. Como o caminho até a padaria é longo, aparecem os problemas de alinhamento lateral no seguidor de caminho.

Foi desenvolvido um vídeo compilado de testes e partidas realizados com o robô, destacando tanto suas funcionalidades satisfatórias quanto casos de erro comuns. Ele pode ser acessado através do endereço: <https://youtu.be/zkDXA5L1D-c> .

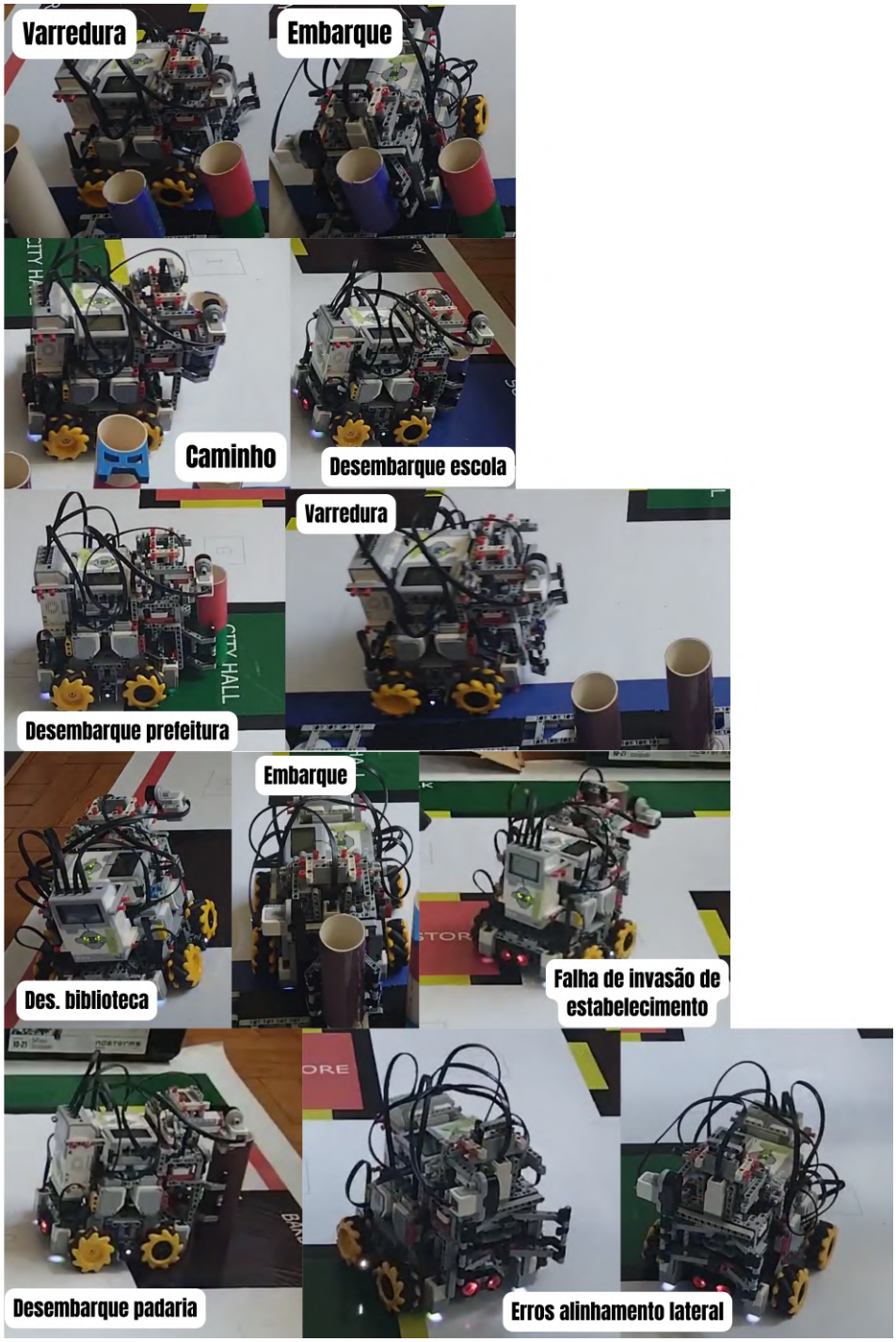


Figura 43 – Pontos relevantes de uma rodada de testes. Fonte: Arquivo Pessoal.

# 5 Conclusão

Este trabalho apresentou o projeto e a implementação de um sistema de localização e navegação autônoma para um robô autônomo omnidirecional construído com peças do kit educacional LEGO EV3, com o objetivo de resolver de forma eficiente o desafio proposto pela RoboCup Brasil *Challenge* 2024, modalidade OPEN. Ao longo do processo, foi possível aplicar conceitos fundamentais da robótica, como localização e navegação, ao mesmo tempo em que se consolidou o aprendizado de técnicas para lidar com tais desafios e a construção de conhecimento prático e teórico. A natureza multidisciplinar dessa área também se fez presente, aproximando áreas que se relacionam com a robótica da mesma forma que a computação, como engenharia, física, mecatrônica, etc., o que ampliou a visão do autor sobre os desafios e possibilidades da área. A participação na competição representou uma experiência satisfatória e motivadora, funcionando como um catalisador para o desenvolvimento do projeto do robô, graças à validação prática em cenários definitivos de testes e aos confrontos com outras equipes, além de estimular trocas valiosas com estudantes de diversas instituições do país, inclusive pela exposição a diferentes soluções desenvolvidas para o mesmo problema. Mesmo com limitações e pontos fracos, o sistema final demonstrou robustez para uso em contextos educacionais, de pesquisa e nas competições, alcançando um equilíbrio entre custo, esforço e desempenho. Além disso, os trabalhos desenvolvidos após a competição contribuíram para o amadurecimento técnico das estratégias inicialmente implementadas e do autor, agregando conhecimento à equipe e à comunidade científica e educacional envolvida.

Em geral, os direcionamentos para trabalhos futuros envolvem tanto os problemas e falhas existentes na implementação atual quanto possíveis propostas de melhorias ou estudos que agregariam no contexto de competições semelhantes.

Na perspectiva de problemas e falhas, são os principais pontos: polir a rotina de localização por conta de ainda não ser tão consistente para casos mais complexos (em que o robô encontra vários obstáculos antes de chegar na zona de embarque, por exemplo), pensar em estratégias para diminuir os erros relacionados ao seguidor de caminho para rotas mais distantes e complexas (ida para estabelecimentos mais distantes da zona de embarque, caminhos que exigem muitas curvas, ou casos com presença de lombadas). Entre esses fatores vale adicionar que a resposta do robô aos valores lidos pelos sensores de cor pode ser otimizada e a precisão das classificações de cor em tempo real ser melhorada. Erros de leitura de cor são bem menos frequentes do que sem as estratégias de calibração implementadas, mas ainda acontecem, principalmente em momentos em que a velocidade do robô dificulta ter uma maior precisão no posicionamento dos sensores para leituras.

Apesar de proibido pelas regras da competição, a utilização de tecnologias de visão computacional para resolver este desafio de forma ainda mais eficiente pode ser um bom motivador de pesquisas futuras. Os conceitos envolvidos com certeza seriam de aplicabilidade útil para competições como a categoria *RoboCup@Work*, por exemplo, que propõe problemas de localização, mapeamento, navegação e manipulação de objetos por robôs autônomos em um cenário simulado inspirado em ambientes industriais, como fábricas e depósitos (ROBOCUP@WORK. . . , 2025).

Dentro de pesquisas que poderiam ser exploradas para além das regras da competição, um novo problema poderia ser incorporado ao desafio a fim de aumentar sua complexidade: a implementação de um mapa completamente dinâmico, inclusive com as localizações dos pontos de interesse (estabelecimentos, zona de embarque) sendo aleatórias. Isso faria com que, além da localização e navegação, o robô também precisasse realizar tarefas de mapeamento de nível mais complexo, aumentando seu grau de autonomia para lidar inclusive com ambientes desconhecidos.

Por outro lado, enquanto dentro das regras da competição, a possibilidade de desenvolver um robô de hardware livre poderia ser mais explorada em trabalhos futuros com foco na mecânica, eletrônica e engenharia envolvidas (diferente deste trabalho, que teve foco específico em software). Pesquisas poderiam ser feitas para tentar replicar os feitos alcançados por este robô feito com peças do kit LEGO em um robô projetado apenas a partir de componentes eletrônicos e peças estruturais, por exemplo. Ou estudar a possibilidade de uma abordagem intermediária, integrando componentes mais eficientes além dos compatíveis com o kit LEGO. Considerar outras opções de controladores, como Arduino, ESP32 ou Raspberry Pi, permitiria explorar outras arquiteturas integradas, para além das limitações dos controladores do EV3.

Da mesma forma que outras pesquisas poderiam ser conduzidas pela motivação da utilização de outros materiais e componentes, caberiam trabalhos considerando o kit LEGO *Mindstorms* EV3. Estudos poderiam ser feitos sobre como diminuir os tempos de carregamento do sistema operacional utilizado para programar o EV3 em *Python*, assim como dos programas, considerando otimizações relacionadas a recursos do sistema operacional, dos processos, à presença de bibliotecas não utilizadas, ou até realizar estudos comparativos sobre a aplicabilidade de outras linguagens de programação nesse contexto. Seguindo essa linha, a implementação de um sistema "*fail-safe*" configurável pelos programadores poderia vir a ser útil para evitar que a cada reinício em uma partida o processo precise ser completamente derrubado e reinicializado, o que consome tempo considerável.

Na perspectiva da arquitetura do software, a comunicação entre os dois controladores principais poderia ser otimizada, explorando possibilidades de paralelização através de bibliotecas que envolvam *multithreading* ou multiprocessos, ainda que virtuais, ou paradigmas de programação assíncrona.

Considerando o aspecto da presença de adversários na competição e de experimentação, poderia ser implementado um sistema de testes envolvendo oponentes simulados. Uma varredura elaborada dos passageiros na zona de embarque poderia possibilitar a elaboração de estratégias no robô que levassem em conta os movimentos do robô adversário, inclusive com uso de Inteligência Artificial, a fim de maximizar seus pontos e minimizar os do concorrente. Nesse sentido, poderiam ser estudadas estratégias de *self-play*, técnicas envolvendo algoritmos como minimax, árvores de decisão, ou até aprendizagem por reforço.

# Referências

ALVES, R. M. F. Coordenação, localização e navegação para robôs de serviço em ambientes internos. **Tese (Doutorado em Ciência da Computação)**, Universidade Federal de Uberlândia, 2017. Citado na página 34.

AROCA, R. V.; BONÍCIO, D. O. H.; AIHARA, C. K.; SÁ, S. T. de L. Robótica educacional e as "competições". **Robótica e processos formativos: da epistemologia aos kits. Porto Alegre-RS: Editora Fi**, p. 245–269, 2019. Citado na página 17.

AZEVEDO, M. **Regras Challenge 2024**. 2024. Competição Brasileira de Robótica. Disponível em: <https://cbr.robocup.org.br/wp-content/uploads/2024/06/Regras_Challenge_2024_Portugues-V1.1.docx>. Acesso em: 10 mar. 2025. Citado na página 19.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4. Citado na página 33.

BRANCALIÃO, L.; GONÇALVES, J.; CONDE, M. Á.; COSTA, P. Systematic mapping literature review of mobile robotics competitions. **Sensors**, MDPI, v. 22, n. 6, p. 2160, 2022. Citado 4 vezes nas páginas 13, 14, 17 e 27.

CATEGORIAS - Competição Brasileira de Robótica. 2025. Disponível em: <https://cbr.robocup.org.br/index.php/categorias/>. Acesso em: 10 mar. 2025. Citado na página 17.

CBR - Competição Brasileira de Robótica. 2025. Disponível em: <https://cbr.robocup.org.br/index.php/o-evento/>. Acesso em: 10 mar. 2025. Citado na página 17.

CHENCHANI, G.; PATEL, K.; SELVARAJU, R.; SHINDE, S.; KALAGATURU, V.; MANNAVA, V.; NAIR, D.; AWAAD, I.; WASIL, M.; THODUKA, S. et al. b-it-bots: Winners of RoboCup@Work 2023. In: **Robot World Cup**. [S.l.]: Springer, 2023. p. 350–361. Citado na página 35.

CHOSET, H.; LYNCH, K. M.; HUTCHINSON, S.; KANTOR, G. A.; BURGARD, W. **Principles of robot motion: theory, algorithms, and implementations**. [S.l.]: MIT press, 2005. Citado na página 26.

DAISY Chaining. 2018. Disponível em: <https://ev3-help-online.api.education.lego.com/Retail/en-us/page.html?Path=editor%2FDaisyChaining.html>. Acesso em: 19 abr. 2025. Citado na página 43.

EV3DEV Home. 2020. Disponível em: <https://www.ev3dev.org>. Acesso em: 18 abr. 2025. Citado na página 29.

EVERETT, H. **Sensors for mobile robots**. [S.l.]: CRC Press, 1995. Citado na página 25.

GETTING started with LEGO® MINDSTORMS Education EV3 MicroPython. 2020. Disponível em: <https://pybricks.com/ev3-micropython/>. Acesso em: 18 abr. 2025. Citado 2 vezes nas páginas 29 e 30.

GRAFFIN, M.; SHEFFIELD, R.; KOUL, R. 'More than robots': Reviewing the impact of the FIRST® LEGO® league challenge robotics competition on school students' STEM attitudes, learning, and twenty-first century skill development. **Journal for STEM Education Research**, Springer, v. 5, n. 3, p. 322–343, 2022. Citado na página 14.

GRANDIA, R.; KNOOP, E.; HOPKINS, M. A.; WIEDEBACH, G.; BISHOP, J.; PICKLES, S.; MÜLLER, D.; BÄCHER, M. Design and control of a bipedal robotic character. **arXiv preprint arXiv:2501.05204**, 2025. Citado na página 13.

GUIA do Usuário: LEGO Mindstorms EV3. 2016. Disponível em: <https://le-www-live-s.legocdn.com/sc/media/files/user-guides/ev3/ev3_user_guide_ptbr-239a9c0ea7115a07ad83d3ce7dff6773.pdf>. Acesso em: 18 abr. 2025. Citado 6 vezes nas páginas 27, 28, 29, 30, 31 e 32.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J.; FRANKLIN, J. The elements of statistical learning: data mining, inference and prediction. **The Mathematical Intelligencer**, Springer, v. 27, n. 2, p. 83–85, 2005. Citado na página 33.

HURBAIN, P. **Color Sensors Showdown**. 2009. Disponível em: <https://www.philohome.com/colcomp/cc.htm>. Acesso em: 18 abr. 2025. Citado 2 vezes nas páginas 30 e 31.

JAULIN, L. **Mobile robotics**. [S.l.]: John Wiley & Sons, 2019. Citado na página 24.

JúNIOR, S. N. Desenvolvimento de um robô omnidirecional para participação em competições de futebol - categoria IEEE Very Small Size. **Trabalhos de Conclusão de Curso de Graduação - Engenharia de Controle e Automação**, Universidade Federal de Ouro Preto, 2019. Citado na página 35.

LAVALLE, S. M. **Planning algorithms**. [S.l.]: Cambridge university press, 2006. Citado na página 26.

LEANDRO, L. d. L. Implementação de movimentos em um robô móvel físico com quatro rodas mecanum usando arduino e o simulink. **Ciência da Computação - Russas - Monografias**, Universidade Federal do Ceará, 2025. Citado na página 35.

LYNCH, K. M.; PARK, F. C. **Modern robotics**. [S.l.]: Cambridge University Press, 2017. Citado 4 vezes nas páginas 21, 22, 24 e 26.

MATARIC, M. J. **The robotics primer**. [S.l.]: MIT press, 2007. Citado 4 vezes nas páginas 12, 23, 24 e 25.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. **Machine learning: An artificial intelligence approach**. [S.l.]: Springer Science & Business Media, 2013. Citado na página 33.

MURPHY, R. R. Competing for a robotics education. **IEEE Robotics & Automation Magazine**, IEEE, v. 8, n. 2, p. 44–55, 2001. Citado na página 13.

_____. **Introduction to AI robotics**. [S.l.]: MIT press, 2019. Citado na página 13.

PANIGRAHI, P. K.; BISOY, S. K. Localization strategies for autonomous mobile robots: A review. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, v. 34, n. 8, p. 6019–6039, 2022. Citado na página 24.

PARANAíBA, T. **Alunos da UFU conquistaram 1º lugar em Competição Brasileira de Robótica | Jornal Paranaíba**. 2022. Youtube. Disponível em: <https://youtu.be/EYFuh3XiJbg>. Acesso em: 10 mar. 2025. Citado na página 18.

PROGRAMMING Languages - ev3dev. 2020. Disponível em: <https://www.ev3dev.org/docs/programming-languages/>. Acesso em: 18 abr. 2025. Citado na página 29.

REIFF, N. **10 Biggest Entertainment Companies**. 2024. Investopedia. Disponível em: <https://www.investopedia.com/articles/investing/020316/worlds-top-10-entertainment-companies-cmcsa-cbs.asp>. Acesso em: 24 fev. 2025. Citado na página 13.

ROBOCUP Federation official website. 2025. Disponível em: <https://www.robocup.org>. Acesso em: 14 abr. 2025. Citado na página 17.

ROBOCUP@WORK League. 2025. Disponível em: <https://atwork.robocup.org>. Acesso em: 17 abr. 2025. Citado na página 70.

ROMERO, R. A. F.; JUNIOR, E. P. S.; OSÓRIO, F. S.; WOLF, D. F. **Robótica móvel**. [S.l.]: LTC, 2014. Citado na página 12.

SALIH, J. E. M.; RIZON, M.; YAACOB, S. Designing omni-directional mobile robot with mecanum wheel. **American Journal of Applied Sciences**, Science Publications, v. 3, n. 5, p. 1831–1835, 2006. Citado na página 22.

SICILIANO, B.; KHATIB, O. **Springer Handbook of Robotics**. [S.l.]: Springer International Publishing, 2016. Citado 5 vezes nas páginas 13, 14, 15, 17 e 21.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to autonomous mobile robots**. [S.l.]: MIT press, 2011. Citado na página 27.

STUDIO Bricklink. 2025. Disponível em: <https://www.bricklink.com/v3/studio/main.page>. Acesso em: 10 mar. 2025. Citado na página 39.

SURYAPRAKASH, S.; THIRUMOORTHI, P.; VISWANATHAN, T. Development of an indoor navigation system for mobile robots using sensor fusion and path planning algorithms. In: IEEE. **2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)**. [S.l.], 2023. p. 1–6. Citado na página 34.

# Anexos

# ANEXO A – Regras da RCB Challenge 2024


CATEGORIA CHALLENGE KIT E OPEN

## 1 Introdução

Para muitas pessoas, os carros autônomos podem parecer uma impossibilidade, uma visão de um futuro distante com ares de ficção científica. Contudo, muitas empresas de tecnologia e montadoras de veículos estão reunindo esforços para tornar a condução autônoma uma realidade cada vez mais próxima de nós.

Seguindo essa tendência tecnológica, os aplicativos de transporte também estão investindo e apostando nesse futuro promissor. Assim, algumas empresas estão promovendo uma competição para ver qual veículo seria capaz de transportar o maior número de passageiros no menor tempo possível.

## 2 Objetivo Geral

Os participantes deverão construir um robô capaz de se localizar, andar pelas ruas da cidade, não colidir com obstáculos, ultrapassar as lombadas, localizar os passageiros, além de transportá-los até os locais pré-estabelecidos com eficiência e agilidade.

## 3 Os Robôs

As equipes poderão construir apenas um robô que seja capaz de realizar todos os desafios propostos na competição.

O tamanho máximo permitido do robô, nas duas categorias, é de *25cm de altura x 25cm de largura x 25cm de comprimento* com eles **totalmente desenvolvidos** e não possuem número máximo de peças, sensores e controladores para suas construções.

O robô deve ser totalmente autônomo, não sendo permitido qualquer tipo de interferência externa, a não ser que o árbitro autorize.

Contudo, vale ressaltar que teremos este ano (2024), duas categorias rodando simultaneamente. As categorias foram divididas entre kits educacionais e hardware livre. Portanto, durante a descrição das regras, fiquem atentos na distinção entre as duas categorias. Já na inscrição das equipes para o evento, será disponibilizado a opção para que cada robô participe em apenas uma das categorias.

Os nomes das categorias ficaram como RoboCup Brasil Challenge KIT e RoboCup Brasil Challenge OPEN.

COMPETIÇÃO BRASILEIRA de ROBÓTICA 2024 RoboCup Brasil

# CATEGORIA CHALLENGE KIT E OPEN

## 1 Introdução

Para muitas pessoas, os carros autônomos podem parecer uma impossibilidade, uma visão de um futuro distante com ares de ficção científica. Contudo, muitas empresas de tecnologia e montadoras de veículos estão reunindo esforços para tornar a condução autônoma uma realidade cada vez mais próxima de nós.

Seguindo essa tendência tecnológica, os aplicativos de transporte também estão investindo e apostando nesse futuro promissor. Assim, algumas empresas estão promovendo uma competição para ver qual veículo seria capaz de transportar o maior número de passageiros no menor tempo possível.

## 2 Objetivo Geral

Os participantes deverão construir um robô capaz de se localizar, andar pelas ruas da cidade, não colidir com obstáculos, ultrapassar as lombadas, localizar os passageiros, além de transportá-los até os locais pré-estabelecidos com eficiência e agilidade.

## 3 Os Robôs

As equipes poderão construir apenas um robô que seja capaz de realizar todos os desafios propostos na competição.

O tamanho máximo permitido do robô, nas duas categorias, é de *25cm de altura x 25cm de largura x 25cm de comprimento* com eles **totalmente desenvolvidos** e não possuem número máximo de peças, sensores e controladores para suas construções.

O robô deve ser totalmente autônomo, não sendo permitido qualquer tipo de interferência externa, a não ser que o árbitro autorize.

Contudo, vale ressaltar que teremos este ano (2024), duas categorias rodando simultaneamente. As categorias foram divididas entre kits educacionais e hardware livre. Portanto, durante a descrição das regras, fiquem atentos na distinção entre as duas categorias. Já na inscrição das equipes para o evento, será disponibilizado a opção para que cada robô participe em apenas uma das categorias.

Os nomes das categorias ficaram como RoboCup Brasil Challenge KIT e RoboCup Brasil Challenge OPEN.

### 3.1 RoboCup Brasil Challenge KIT

A construção dos robôs deverá seguir as regras da antiga categoria SEK (Standard Educational Kit). Para isso, eles deverão possuir apenas peças de um único kit robótico. Por exemplo, se o kit escolhido for o Kit LEGO R, nenhuma peça ou acessório de algum outro fabricante (Vex R, pETe R ou FischerTechnik R) podem ser utilizados.

### 3.2 RoboCup Brasil Challenge OPEN

A construção dos robôs será com Hardware Free. Não terá restrições com relação a microcontroladores ou microprocessadores, nem mesmo o tipo de sensor, atuadores e as suas quantidades. A única restrição dar-se-á na utilização de câmeras ou similares. Portanto, não será permitido nenhum tipo de processamento de imagens.

Uma observação importante é que, por essa categoria se tratar de Hardware Free, será permitido que as equipes possam usar também kits educacionais, suas peças ou até mesmo sensores e atuadores, sem nenhum tipo de restrição.

## 4 A Arena

A arena será confeccionada com lona de banner e impressa utilizando um arquivo padrão disponibilizado pela organização do evento. As cores impressas deverão ser mais próximas do padrão RGB, para facilitar a detecção dos sensores.

A arena será idêntica dos dois lados, de uma forma espelhada, fazendo com que não haja qualquer tipo de vantagem entre um lado e o outro da arena.

Na Figura 1, podemos detalhar algumas áreas e os objetivos propostos nesse desafio.

- PARQUE – Área representada na cor VERDE, com dimensões de 150x15cm;
- PADARIA – Área representada na cor MARROM, com dimensões de 27x24cm;
- ESCOLA – Área representada na cor AZUL, com dimensões de 27x24cm;
- FARMÁCIA – Área representada na cor VERMELHO, com dimensões de 24x24cm;
- PREFEITURA – Área representada na cor VERDE, com dimensões de 24x24cm;
- MUSEU – Área representada na cor AZUL, com dimensões de 27x24cm;

COMPETIÇÃO BRASILEIRA de ROBÓTICA 2024 RoboCup Brasil

# CATEGORIA CHALLENGE KIT E OPEN

- BIBLIOTECA – Área representada na cor MARROM, com dimensões de 27x24cm;
- RUA - Área representada na cor BRANCO, tem como largura máxima de 30cm entre os estabelecimentos;
- LINHAS PRETAS – Representa os limites entre as ruas e os estabelecimentos. Possui uma largura de 3cm;
- LINHAS AMARELAS - Representa as entradas dos estabelecimentos. Possui uma largura de 3cm e comprimento de 15cm;
- LINHAS VERMELHAS – Representa o final das ruas, como se fossem ruas sem saídas. Possui uma largura de 3cm e comprimento de 30cm;
- RETÂNGULOS CINZA COM LETRAS – Representa os possíveis locais onde o árbitro da partida poderá colocar um obstáculo (caixa de leite);
- FAIXA BRANCA CENTRAL – Representa a área de embarque dos passageiros, mas que neste ano será colocado uma plataforma elevada com as dimensões de 80x1500mm (LxC) e uma altura entre 15 e 18mm. Portanto, todos passageiros estarão em cima dessa plataforma de MDF Branca, de uma forma aleatória, mas igualitária entre as duas metades da pista. Assim, para ficar bem claro, não teremos marcações dos possíveis locais onde estarão os tubos. O posicionamento deles ficará a critério do árbitro da partida, desde que a dimensão máxima do tubo (5cm de diâmetro) esteja totalmente em cima da plataforma.

  Os passageiros serão representados por tubos de PVC detalhados a seguir.

# 2024

# CATEGORIA CHALLENGE KIT E OPEN

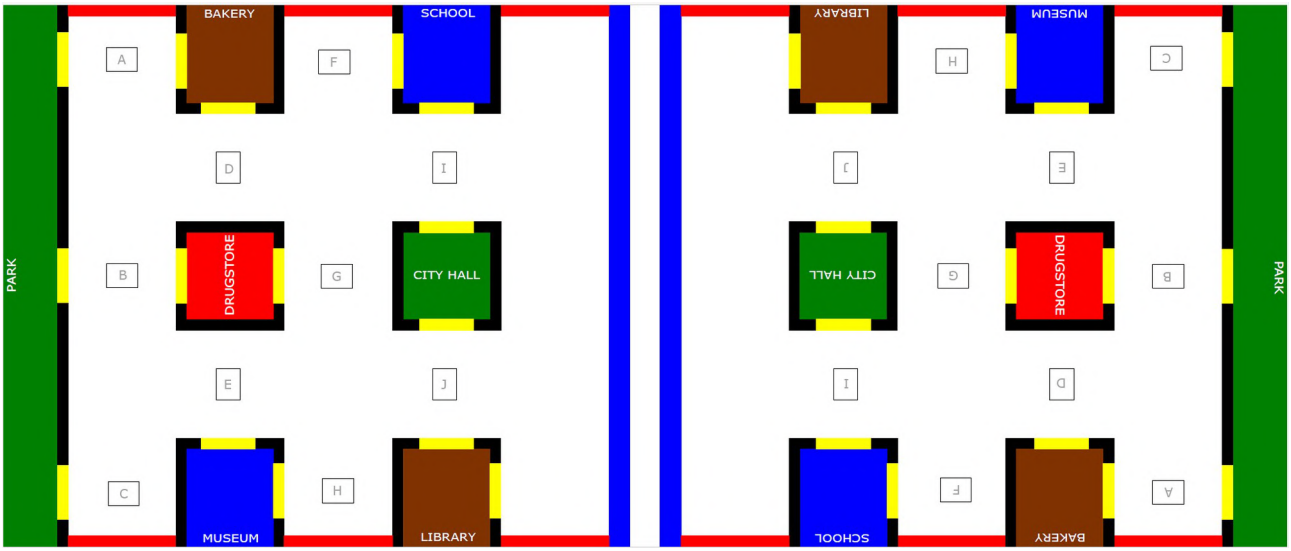


*Figura 1 - Vista superior da arena*

Como a arena é simétrica e representada de forma espelhada, iremos detalhar apenas uma das metades, com as posições e dimensões de uma forma mais completa.

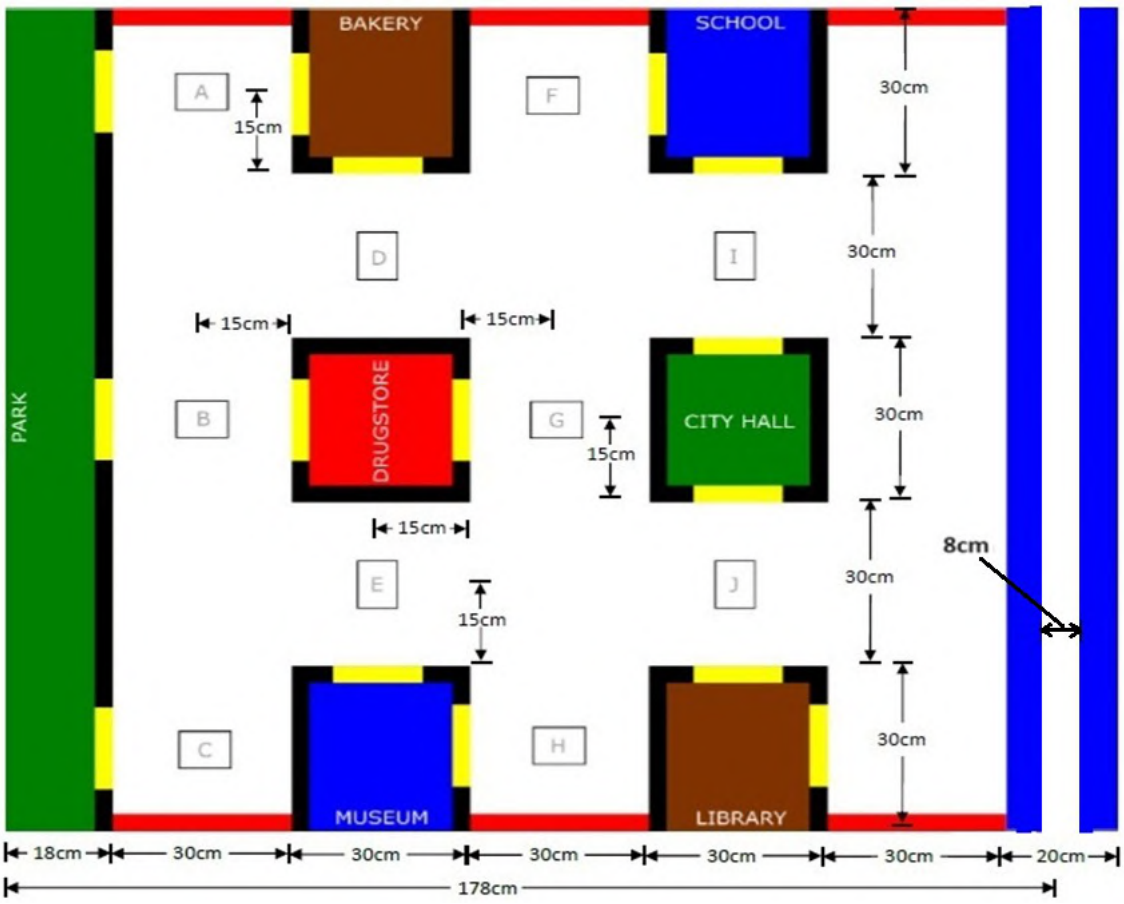


*Figura 2 - Vista superior da metade da arena com dimensões*

## 5 Os passageiros (tubos)

Os tubos são de PVC de padrão para tubulações de 5cm (aproximadamente 2 polegadas) de circunferência e com dois diferentes comprimentos. Estes tubos representarão pessoas que esperam o transporte até seu local específico que será detalhado a seguir. Os comprimentos dos tubos serão de **10cm** (para representar crianças) e de **15cm** (para representar pessoas adultas). Todos os tubos serão envelopados com papel contact com as cores AZUL, VERDE, VERMELHO e MARROM, apenas na superfície externa dos tubos.

Serão disponibilizadas 3 (três) unidades de cada tamanho de tubo e de cada cor. Portanto, teremos em cada partida **até** 3 tubos de cada tipo. Assim, o árbitro terá à disposição 3 tubos de 10cm e 3 tubos de 15cm na cor azul, 3 tubos de 10cm e 3 tubos de 15cm na cor verde, 3 tubos de 10cm e 3 tubos de 15cm na cor marrom e apenas 3 tubos de 15cm na cor vermelha.

Esse ano teremos uma novidade, tubos brancos (cor original do tubo ou papel adesivo branco) de 10 e 15cm, que representam pessoas que NÃO solicitaram carona pelo aplicativo de transporte. A regra das quantidades será a mesma dos demais

tubos, podendo ter em cada partida, até três tubos de 10cm de comprimento e/ou três tubos de 15cm de comprimento.

## 6 Lombadas

Será possibilitado a utilização de lombadas durante as partidas. As regras de utilização dessas lombadas serão similares as utilizadas com os obstáculos. Portanto, essas lombadas poderão ser colocadas antes ou durante a partida. Também será possibilitado de serem utilizado até três lombadas em cada lado da pista.

Essas lombadas terão 300mm de comprimento e 10mm de altura, com uma inclinação de 30°, conforme a figura abaixo (*Será disponibilizado o arquivo para que as equipes possam imprimir em 3D*).

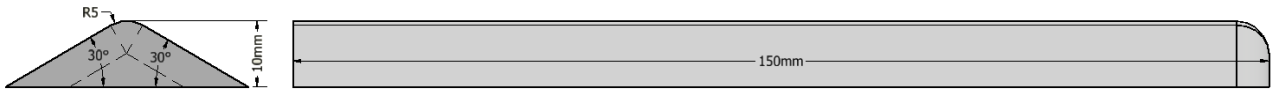


*Figura 3 - Vista frontal e vista lateral da metade da lombada*

Com relação aos possíveis posicionamentos dessas lombadas, para não atrapalhar as entradas dos estabelecimentos, será definido que as lombadas estarão centralizadas em relação as paredes (linha preta). Portanto, teremos 20 possíveis posições conforme a figura abaixo.

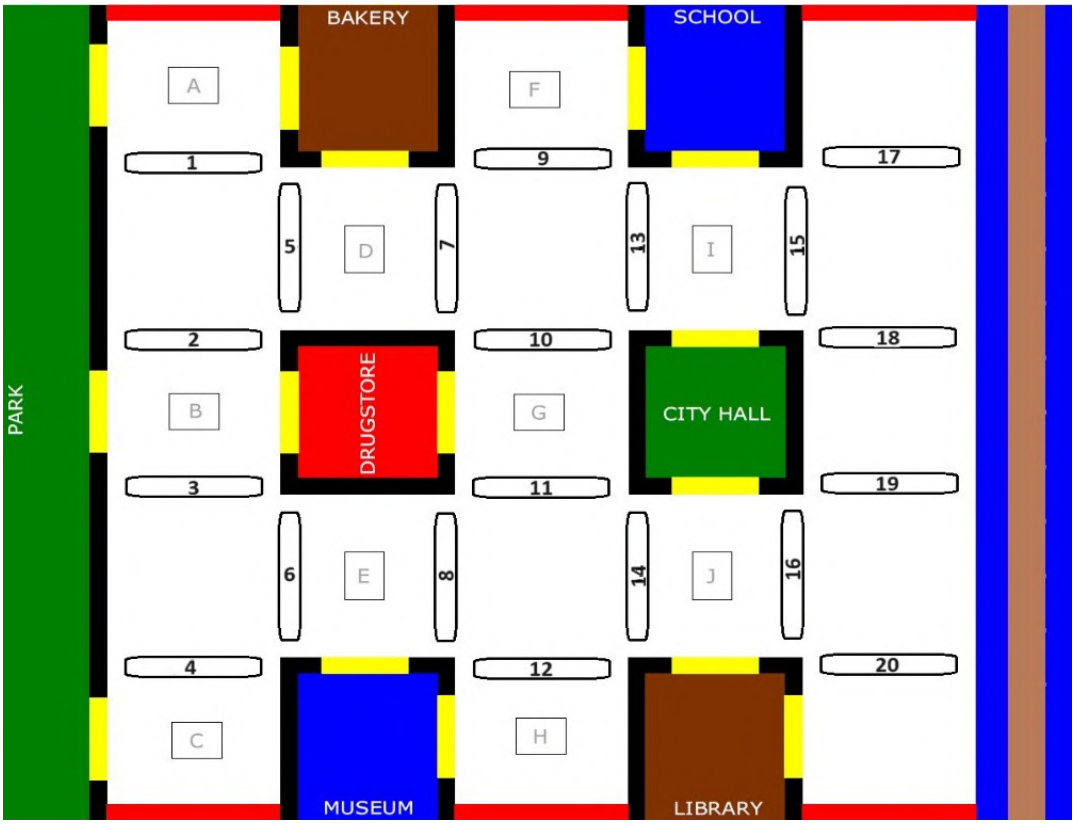


*Figura 4 - Pista com os possíveis locais onde poderão ter lombadas*

## 7 As Partidas

Cada uma das equipes competirá de um lado da cidade, tendo como objetivo conduzir o máximo de passageiros (tubos) para seus locais pré-definidos.

A equipe que obtiver a maior pontuação (pontuações disponíveis na Secção 8) será determinada como vencedora.

Cada partida contará com pelo menos um árbitro, e caberá ao mesmo a anotação das pontuações e gerenciamento dos passageiros no decorrer da disputa.

Todos os robôs iniciam em posição aleatória, em alguma rua (área branca) e em qualquer lugar da cidade, colocados pelo árbitro, sempre visando simetria entre as equipes.

Não será permitido que os robôs, de um lado da cidade, possam adentrar nos limites do outro lado. Portanto, as áreas que representam as ruas (área branca), só poderão ser ocupadas por um robô de cada lado.

Como teremos uma plataforma elevada no centro da pista, NÃO será permitido que os robôs subam e se desloquem por cima da plataforma.

# CATEGORIA CHALLENGE KIT E OPEN

Caso um robô adversário ultrapasse a zona de embarque e entre nas ruas (área branca) do outro lado da cidade, será considerado como robô perdido. O mesmo deverá ser reconduzido ao seu local de partida.

Também será considerado robô perdido quando o mesmo ultrapassar os limites da cidade (356x150cm), representado pelas as linhas vermelhas ou os limites das áreas coloridas dos estabelecimentos.

A faixa central, representada na cor azul, com a plataforma de MDF colocada de uma forma centralizada, será a zona de embarque dos passageiros. Para as equipes que possuem a lona impressa com o arquivo antigo, os 11 círculos com diâmetro de 5cm, ficarão embaixo da plataforma, portanto não serão utilizados como possíveis locais onde poderão ter passageiros (tubos) à espera dos veículos. Entretanto, o arquivo novo, disponibilizado no site oficial do evento, não possui estes círculos, apenas o local de embarque na cor azul com 200mm de largura e uma faixa branca de 80x150mm centralizada.

Para uma melhor aproximação com a realidade (como os tubos representam pessoas), todos tubos serão posicionados **em pé** (verticalmente) na zona de embarque.

A quantidade e disposição dos passageiros na zona de embarque serão definidos pelo árbitro antes ou durante a partida.

Será considerado um passageiro entregue, com sucesso, quando o tubo estiver completamente dentro da área colorida e de pé, não podendo estar com qualquer parte encostando nas linhas pretas e/ou amarelas.

Caso o tubo não fique em pé, em qualquer parte da arena, será considerado tubo perdido. O árbitro deverá posicioná-lo onde estava no início da partida.

Caso o robô venha a derrubar os tubos que já estavam posicionados antes de forma correta ou, deslocá-lo até uma posição proibida, a pontuação anterior será afetada (pontuações e descrição estão disponíveis na Secção 8) e o tubo afetado será imediatamente recolocado para a zona de embarque pelo árbitro.

Durante a partida, na zona de embarque, caso o robô tente pegar o passageiro sem sucesso ou desloque os tubos acidentalmente, será considerado tubo perdido. Assim, a equipe sofrerá penalidade descrita na pontuação (pontuações disponíveis na Secção 8) e o tubo será imediatamente recolocado pelo árbitro.

Não será permitido que os robôs saiam das ruas (área branca) e invadam qualquer estabelecimento (limitados pelos sensores de cor). Exemplo: O robô não pode passar por dentro da área da Padaria (marrom), mesmo que parcialmente, para chegar até o Parque. Portanto, os robôs terão que se deslocar pela cidade utilizando as ruas (área branca) e como referência as linhas pretas, vermelhas e amarelas que são os limites. Exceto quando for pegar ou soltar os passageiros nos locais definidos.

# CATEGORIA CHALLENGE KIT E OPEN

As linhas amarelas representam a entrada de cada estabelecimento. Sendo assim, só será considerado válido o passageiro que for solto dentro da área colorida, desde que o passageiro tenha adentrado no local passando, integralmente, entre os limites da linha amarela.

Para os estabelecimentos que possuem duas entradas, não haverá diferença na pontuação entre elas, portanto os robôs podem utilizar qualquer uma das entradas assinaladas com a linha amarela.

Com relação aos estabelecimentos que possuem duas entradas, não será permitido que os robôs atravessem os locais entrando em um dos lados e saindo do outro. Nesse caso será considerado robô perdido e deverá retornar ao local de início da partida.

Os retângulos identificados com as letras de A até J, serão os locais onde o árbitro **poderá** colocar um obstáculo. Este obstáculo será representado com uma caixa de leite, sem cor definida, que possuem dimensões aproximadas de 16x9x7cm. Fica previsto no máximo de até três obstáculos em cada lado da arena. Estes obstáculos poderão ser colocados antes ou depois do início da partida.

Caso o robô não identifique o obstáculo e venha a deslocá-lo ou até mesmo derrubá-lo, a equipe sofrerá a penalidade de obstáculo não identificado. Nesse caso, o obstáculo será reposicionado e o robô deverá voltar ao local inicial da partida.

A proposta da colocação destes obstáculos, antes ou durante a partida, é justamente simular um possível acidente ou algo que impossibilite que os veículos trafeguem pela rua. Portanto, quando o árbitro colocar um obstáculo em um determinado local, fica previsto que não será permitido que o robô tente passar pela rua ou tente colocar um passageiro pela entrada bloqueada. Com essa possibilidade, as equipes deverão estar preparadas para “recalcular a rota” e procurar a(s) outra(s) entrada(s) dos estabelecimentos.

Fica previsto também que não será permitido que o árbitro coloque os dois obstáculos fechando as duas entradas de qualquer estabelecimento.

Toda vez que uma equipe tiver problemas com o robô e precisar tocar nele, reposicioná-lo, reiniciar o programa, remontar alguma peça solta ou apenas deu um "bug", será considerado como um reinício.

Quando uma das equipes terminar de posicionar, com sucesso, o último tubo no local correto, automaticamente a partida será considerada encerrada.

Será permitido que o árbitro coloque, retire ou apenas troque tubos no local de embarque, mesmo depois do início da partida, posicionando-os nos espaços destinados aos passageiros de forma simétrica.

O tempo máximo de cada partida é de **12 minutos**.

Não será contabilizado pontuações negativas, portanto nenhuma equipe terá pontuação abaixo de zero.

# CATEGORIA CHALLENGE KIT E OPEN

Quando estiver chegando perto do término da partida e o árbitro identificar que está havendo um empate na pontuação geral (pontuações positivas e diferentes de zero), o mesmo deverá dar um acréscimo de até 3 minutos (overtime). Dentro desse tempo, será considerada vencedora a primeira equipe que realizar alguma tarefa que acrescente pontuação POSITIVA. Caso ocorra esse acréscimo, após os 15 minutos (12 minutes + 3 de acréscimo), se nenhuma equipe conseguir fazer pontuação positiva, a partida será decidida pelos critérios de desempate da pontuação de 0x0 (zero a zero).

Caso a partida termine com as pontuações gerais de 0x0 (zero a zero), será considerado os seguintes critérios de desempate que definirão a equipe vencedora, em ordem de importância:

- 1°- Será considerada vencedora a equipe que conseguir a maior pontuação positiva;
- 2°- Caso a pontuação positiva seja igual, será considerada vencedora a equipe que obteve menos pontuações negativas;
- 3°- Caso nenhuma das equipes consiga realizar alguma tarefa que some pontuação positiva, a partida será considerada empatada (fase de classificação);
- 4°- Fica previsto que (como última opção), caso a partida esteja em uma fase da competição onde se torna indispensável a definição de uma equipe vencedora (mata-mata), ficará de responsabilidade do árbitro a análise e interpretação do desempenho dos robôs durante a partida. Assim, será considerada vencedora a equipe que chegar mais perto de alcançar os objetivos propostos. Essa análise será realizada junto com os capitães das equipes participantes.

A cor, tamanho e destino dos passageiros, ficam definidos da seguinte maneira:

- Tubo AZUL com 10cm: Representa as crianças que irão para a ESCOLA;
- Tubo AZUL com 15cm: Representa os adultos que irão para a MUSEU;
- Tubo VERDE com 10cm: Representa as crianças que irão para o PARQUE;
- Tubo VERDE com 15cm: Representa os adultos que irão para a PREFEITURA;
- Tubo MARROM com 10cm: Representa as crianças que irão para a BIBLIOTECA;
- Tubo MARROM com 15cm: Representa os adultos que irão para a PADARIA;
- Tubo VERMELHO com 15cm: Representa os adultos que irão para a FARMÁCIA;

- Tubo BRANCO com 10 ou 15cm: Representa crianças ou adultos que não solicitaram o serviço de transporte e não deverão serem retirados da plataforma pelos robôs;

## 7.1 Fim de uma Partida

Existem quatro formas de ser decretada o fim de uma partida:

- **Time-out:** o tempo de 12 minutos acaba. Sendo assim, a equipe com mais pontos ao fim do tempo será declarada como vencedora.
- **Give-up:** uma das equipes desiste da partida. Sendo assim, a equipe adversária é imediatamente decretada como vencedora.
- **Overtime:** Caso a partida esteja terminando empatada e com pontuações positivas, o árbitro dará até 3 minutos de prorrogação. Dentro desse tempo, ganhará a equipe que realizar qualquer objetivo que some pontuação positiva.
- **Match completed:** quando todos passageiros são entregues nos destinos de forma correta. Sendo assim, a equipe com mais pontos ao final do desafio será declarada como vencedora.

## 8 As Pontuações

- Cada tubo VERMELHO de 15cm colocado com sucesso: **50 pontos**.
- Cada tubo MARROM de 15cm colocado com sucesso: **45 pontos**.
- Cada tubo AZUL de 15cm colocado com sucesso: **45 pontos**.
- Cada tubo VERDE de 15cm colocado com sucesso: **42 pontos**.
- Cada tubo VERDE de 10cm colocado com sucesso: **40 pontos**.
- Cada tubo MARROM de 10cm colocado com sucesso: **37 pontos**.
- Cada tubo AZUL de 10cm colocado com sucesso: **37 pontos**.
- Cada tubo BRANCO, não identificado pelos sensores do robô e, retirado totalmente de cima da plataforma de embarque, será considerado como passageiro errado: **- 8 pontos.**
- Cada tubo perdido (especificado nas Observações): **-3 pontos**.