UNIVERSIDADE FEDERAL DO MARANHÃO

EDUARDO COSTA DE CARVALHO

PLANEJAMENTO DE TRAJETÓRIA DE UM ROBÔ MÓVEL USANDO O ALGORITMO A*

São Luís - MA

2025

EDUARDO COSTA DE CARVALHO

PLANEJAMENTO DE TRAJETÓRIA DE UM ROBÔ MÓVEL USANDO O ALGORITMO A*

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação na Universidade Federal do Maranhão (UFMA).

Orientador: Prof. Areolino de Almeida Neto

São Luís - MA
2025

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Carvalho, Eduardo Costa de.
Planejamento de trajetória de um robô móvel usando o algoritmo A* / Eduardo Costa de Carvalho. - 2025.
58 f.

Orientador(a): Areolino de Almeida Neto.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2025.

1. Robô de Baixo Custo. 2. Algoritmo A*. 3. Planejamento de Trajetória. I. Almeida Neto, Areolino de. II. Título.

EDUARDO COSTA DE CARVALHO

PLANEJAMENTO DE TRAJETÓRIA DE UM ROBÔ MÓVEL USANDO O ALGORITMO A*

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação na Universidade Federal do Maranhão (UFMA).

Orientador: Prof. Areolino de Almeida Neto

Aprovado em 08 de agosto de 2025

Banca Examinadora

_____

Prof. Dr. Areolino de Almeida Neto - UFMA
Orientador

_____

Prof. Dr. Alexandre César Muniz de Oliveira - UFMA
Examinador 1

_____

Prof. Dr. Paulo Rogério de Almeida Ribeiro - UFMA
Examinador 2

São Luís - MA
2025

AGRADECIMENTOS

Agradeço primeiramente aos meus familiares, de forma especial aos meus pais, Paulo e Catherine, por acreditarem no meu potencial.

Também agradeço ao meu orientador Areolino por todos os ensinamentos ao longo do desenvolvimento deste trabalho.

Além disso, sou grato a UFMA e a seus professores por fornecerem os conhecimentos necessários para a minha formação.

RESUMO

Este trabalho tem o objetivo de implementar o algoritmo A* em um robô, fazendo-o movimentar-se em um ambiente previamente conhecido sem o uso de sensores de percepção externa. O robô utilizado é de baixo custo, cuja estrutura é leve e feita de um chassi de acrílico retangular de 16 cm x 9 cm. Além disso, o robô possui três rodas, nas quais as dianteiras são motorizadas e a traseira não. Nas rodas motorizadas, foi aplicada a direção diferencial, ou seja, as rodas são controladas de forma independente. Para o planejamento da trajetória, foi utilizado o algoritmo A*, por combinar rapidez de processamento e qualidade da solução encontrada. Essa eficiência é devido ao uso de uma heurística. Neste trabalho, a heurística usada é relacionada com a distância, na qual foi escolhida a distância Manhattan. Foram realizados 15 testes para 10 ambientes diferentes. Esses ambientes foram formados em um espaço dividido em 10 x 10 células de tamanhos iguais. Nesses ambientes, os obstáculos foram aleatoriamente dispostos. Nos experimentos realizados, foi possível comprovar a eficiência do algoritmo A* para gerar caminhos ótimos. Assim, conclui-se que é possível que um robô de baixo custo navegue em um ambiente previamente conhecido.

Palavras-chaves: robô de baixo custo, algoritmo A*, planejamento de trajetória.

ABSTRACT

This work aims to implement the A* algorithm in a mobile robot, enabling it to navigate through a previously known environment without the use of external perception sensors. This robot is a low-cost model, built with a lightweight rectangular acrylic chassis measuring 16 cm x 9 cm. Furthermore, it contains three wheels, with the two front wheels motorized and independently controlled using differential drive, while the rear wheel is passive. For path planning, the A* algorithm was chosen due to its balance between processing speed and solution quality, which is achieved through the use of a heuristic. In this study, the heuristic is related with distance, in which the Manhattan distance was chosen. A total of 15 tests were conducted across 10 different environments, each modeled as a 10 x 10 grid with randomly placed obstacles. The results demonstrate the effectiveness of the A* algorithm in generating optimal paths. Therefore, it is concluded that a low-cost robot can successfully navigate in a predefined environment.

Keywords: low-cost robot, A* algorithm, path planning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D	Duas dimensões
CCET	Centro de Ciências Exatas e Tecnologias
cm	Centímetros
COCRC	Centro de Rastreio e Controle de Satélites
EVA	Etil vinil acetato
Fig.	Figura
GB	Gigabyte
GND	Tensão elétrica nula, em inglês
GPS	Sistema de posicionamento global, em inglês
IDE	Ambiente de desenvolvimento integrado, em inglês
INPE	Instituto Nacional de Pesquisas Espaciais
kg	Kilograma
Lab. INOVTEC	Laboratório de Inovação Tecnológica
LIDAR	Varredura e detecção de luz, em inglês
LIT	Laboratório de Integração e Testes
m	Metros
máx.	Máxima
NASA	Administração Nacional de Aeronáutica e Espaço, em inglês
PID	Proporcional integral derivativo
PWM	Modulação por largura de pulso, em inglês
RAM	Memória de acesso randômico, em inglês
ROS	Sistema operacional de robô, em inglês
RPM	Rotações por minuto
RX	Pino serial de recepção
TX	Pino serial de transmissão
UART	Receptor / transmissor assíncrono universal, em inglês
UFMA	Universidade Federal do Maranhão
USB	Barramento serial universal, em inglês
V	Volts

LISTA DE SÍMBOLOS

A	Arco de 90°
α	Ângulo de rotação da roda em radiano
c	Circunferência da roda
D	Distância entre as rodas dianteiras
d	Distância percorrida
d_e	Distânicia euclidiana entre dois pontos
d_m	Distânicia manhattan entre dois pontos
$f(.)$	Função do custo total
$g(.)$	Custo da posição inicial até a posição atual
$h(.)$	Estimativa de custo da posição atual até a posição final (heurística)
n	Posição que está sendo avaliada
R	Rotações necessárias da roda
r	Raio da roda
T	Tempo de um giro completo da roda
t	Tempo de acionamento dos motores
v	Velocidade do robô
x_1	Coordenada x da posição inicial da trajetória
x_2	Coordenada x da posição final da trajetória
y_1	Coordenada y da posição inicial da trajetória
y_2	Coordenada y da posição final da trajetória
(x, y)	Coordenada de uma posição no plano cartesiano
θ	Orientação
|.|	Operação módulo
π	Número pi

SUMÁRIO

1 INTRODUÇÃO

A robótica móvel é uma área muito promissora dentro da robótica e da ciência da computação. Essa área possui muitas aplicações que vão desde veículos autônomos até sistemas de exploração de ambientes inóspitos, como o espaço e o fundo do mar. O desenvolvimento dessa área é impulsionado pelo avanço de tecnologias mecânicas, eletrônicas e computacionais, como peças mecânicas mais resistentes e mais leves, sensores eletrônicos mais precisos, processamento digital mais rápidos e técnicas de inteligência artificial menos custosas.

A capacidade de um robô móvel de navegar em ambientes desconhecidos e dinâmicos é um dos fatores importantes para seu funcionamento, principalmente quando a tomada de decisão precisa ser feita em tempo real. Para que um robô seja capaz de se deslocar de forma segura e eficiente, ele deve ter algoritmos de planejamento de trajetória. Esses algoritmos são responsáveis por calcular uma rota entre um ponto de partida e um ponto de destino. Além disso, sistemas de navegação muitas vezes utilizam sensores para perceber o ambiente ao redor e detectar as mudanças.

Diversos algoritmos de planejamento de trajetórias foram desenvolvidos para resolver o problema da navegação. Entre esses algoritmos, um dos mais eficientes e adotados é o algoritmo A*. Proposto por Hart, Nilsson e Raphael em 1968 (ZANCHIN, 2018), o algoritmo A* destaca-se pelo equilíbrio entre eficiência computacional e qualidade da solução encontrada. Esse algoritmo é uma escolha popular em áreas que vão desde jogos, navegação de robôs móveis e até planejamento de trajetórias em aplicativos de mobilidade urbana.

Os sistemas de navegação autônomos geralmente possuem sensores como *Light Detection and Ranging* (LIDAR, do inglês, varredura e detecção de luz), câmeras ou sensores ultrassônicos para detectar obstáculos e mapear o ambiente em tempo real. Esses sensores permitem que o robô perceba o ambiente ao redor e corrija sua trajetória conforme as mudanças no ambiente (AUTOPILOT REVIEW, 2025). Entretanto, esses dispositivos apresentam algumas desvantagens, como alto custo e complexidade na integração.

Diante desse cenário, o presente trabalho propõe uma abordagem mais simples: a navegação baseada em um ambiente previamente mapeado, sem o uso de sensores de percepção externa. Portanto, assume-se que os obstáculos são fixos

e suas posições são conhecidas. Isso permite que o planejamento da trajetória seja realizado antes do deslocamento do robô. Este trabalho tem o intuito de avaliar o tempo de execução, a precisão do algoritmo de planejamento de trajetória e as variações na posição final do robô causadas por diferentes execuções.

O robô usado neste estudo foi programado utilizando a plataforma Arduino. Este componente foi escolhido devido à sua facilidade de programação, adoção em projetos de robótica e compatibilidade com diversos componentes eletrônicos.

O sistema robótico usado possui servomotores conectados nas rodas e sua movimentação foi realizada com base no tempo de ativação dos motores. Isso ocorreu para ajustar a posição do robô durante a execução da trajetória em controle do tipo malha aberta.

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo geral do trabalho é implementar e avaliar um sistema de planejamento de trajetória em um ambiente previamente mapeado, utilizando um robô físico de baixo custo, sem o uso de sensores.

1.1.2 Objetivos específicos

Para atingir o objetivo principal, os seguintes objetivos específicos foram definidos:

- Montar um robô de baixo custo;
- Desenvolver um sistema de comunicação com o robô via Bluetooth
- Implementar o algoritmo A* no Arduino por meio da linguagem C++;
- Desenvolver a programação de locomoção do robô sem usar sensores;
- Construir diversos cenários de teste do sistema de navegação.

1.2 Justificativa / Motivação

As motivações principais foram o desenvolvimento de um robô de baixo

custo, implementação prática do algoritmo A* e navegação em ambientes predominantemente estáticos. Assim, esses conceitos, que parecem ser complexos, podem ser implementados com recursos de *hardware* limitados.

A navegação autônoma de robôs móveis em ambientes previamente mapeados é um campo de estudo relevante dentro da robótica. Por exemplo, em ambientes agrícolas, o planejamento de trajetória é essencial para garantir deslocamentos eficientes. Entretanto, quando um robô móvel não possui sensores externos para percepção do ambiente, surgem desafios relacionados à precisão da navegação. Esses desafios são ocasionados principalmente pelo acúmulo de erros que podem comprometer a execução da trajetória.

Este trabalho contribuirá em fornecer uma análise sobre a implementação prática do algoritmo A* em um robô móvel de baixo custo. Para isso, foram identificadas as limitações da abordagem baseada no controle por tempo de ativação dos motores das rodas e do cálculo prévio da trajetória. Além disso, os resultados podem ser úteis para futuras pesquisas em robôs móveis de baixo custo.

1.3 Trabalhos relacionados

Nesta seção, são analisadas pesquisas em robótica móvel que utilizam o algoritmo de planejamento de trajetória abordado neste trabalho, o algoritmo A*. Esse algoritmo tem sido estudado na robótica móvel devido à sua eficiência na busca de caminhos ótimos. Diversos trabalhos exploram a implementação do algoritmo A* em diferentes contextos.

Um exemplo é o trabalho de William Santos Ferreira, intitulado "Controle de Trajetória de Robô Móvel com Desvio de Obstáculos" (FERREIRA, 2023). O objetivo desse trabalho foi desenvolver um sistema de controle de trajetória para robôs móveis capaz de desviar de obstáculos. Assim, garante-se uma navegação autônoma e eficiente em espaços complexos.

Ainda com relação ao trabalho de William Santos Ferreira, o autor apresenta a cinemática do robô móvel, abordando as equações do seu movimento. Para o planejamento de trajetória, foi escolhido o algoritmo A*. Além disso, o estudo fez o uso do controlador Klancar, devido à sua simplicidade e eficácia no seguimento de trajetórias planejadas. Esse controlador utiliza uma matriz de ganhos para ajustar o

erro de posição do robô, gerando comandos de velocidade linear e angular adequados.

Outro trabalho é intitulado "Planeamento de Trajetórias em Ambientes Agrícolas" de Luís Carlos Feliz Santos (SANTOS, 2017). O objetivo desse estudo é o desenvolvimento de estratégias para o planejamento de trajetórias de veículos autônomos em ambientes agrícolas. O foco do trabalho é na otimização de operações, como a colheita e a pulverização.

A solução proposta nessa dissertação consiste na adaptação do algoritmo A*, considerando as características do terreno. Essa adaptação leva em conta três variáveis: as coordenadas da localização do robô no mundo (x, y); a orientação (θ); a altitude do terreno. O estudo destaca a importância de adaptar os algoritmos de planejamento de trajetórias considerando as características dos terrenos agrícolas, que têm irregularidades e obstáculos naturais. Para isso, foram usadas técnicas que levam em conta a topografia do terreno e a presença de obstáculos.

Outro trabalho é intitulado "Comparação entre planejadores de caminhos globais de um robô móvel", de Pedro Medeiros de Assis Brasil (BRASIL, 2020). O objetivo do estudo é comparar algoritmos de busca de menor caminho utilizando um robô móvel com o sistema *Robot Operating System* (ROS, do inglês, sistema operacional de robô). Foram realizados experimentos em diferentes ambientes para analisar os algoritmos A* e Dijkstra, tanto em simulações quanto no robô real. Os testes foram feitos com o robô TurtleBot 3 Burger. Para a análise dos resultados, foi utilizado um sistema de captura de posição do robô, composto por uma câmera posicionada sobre os experimentos e um código de reconhecimento que converte pixels em metros.

1.4 Organização do trabalho

Além deste capítulo, o trabalho foi dividido da seguinte forma:

- Capítulo 2 – Fundamentação teórica: são explorados os conceitos de robótica móvel, planejamento de trajetória e algoritmo A*;
- Capítulo 3 – Metodologia: são detalhados os componentes do robô, os ambientes de testes, a implementação do algoritmo A* e a lógica de movimentação do robô;

- Capítulo 4 – Análise dos testes: é apresentada a análise dos experimentos realizados, comparando o desempenho dos cenários testados;
- Capítulo 5 – Conclusão: resume as principais descobertas da pesquisa e sugere melhorias para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Breve histórico da robótica móvel

A robótica móvel teve seu desenvolvimento acelerado a partir da segunda metade do século XX, impulsionada pelo avanço de tecnologias como a inteligência artificial. Desde então, tornou-se uma das principais áreas de pesquisa na robótica, com aplicações que vão desde a exploração espacial até a navegação autônoma de veículos terrestres.

Um dos primeiros marcos da robótica móvel foi o desenvolvimento do robô Shakey, criado pelo Instituto de Pesquisa de Stanford no final da década de 1960. Esse robô foi pioneiro na integração de percepção do ambiente, planejamento de trajetória e tomada de decisões autônomas. O Shakey utilizava sensores e câmeras para mapear o ambiente e tomar decisões com base no planejamento computacional (ZANCHIN, 2018).

Nas décadas seguintes, a robótica móvel evoluiu, com o desenvolvimento de algoritmos mais eficientes e sensores mais avançados. Durante os anos de 1980 e 1990, instituições como a National Aeronautics and Space Administration (NASA, do inglês, Administração Nacional de Aeronáutica e Espaço) passou a investir em pesquisas em robótica móvel. Isso resultou em robôs como o Sojourner, que fez parte da missão Mars Pathfinder em 1997. Esse foi o primeiro robô móvel a explorar a superfície de Marte, utilizando um sistema de controle remoto (NASA, 2021).

No Brasil, o Instituto Nacional de Pesquisas Espaciais (INPE) teve um papel fundamental no avanço da robótica no país, principalmente na área espacial. Durante as décadas de 1970 e 1980, o INPE avançou em sua infraestrutura. Em 1987 foi inaugurado o Laboratório de Integração e Testes (LIT), e em seguida, implementado o Centro de Rastreio e Controle de Satélites (COCRC), ambientes que exigem automação, precisão e sistemas de controle, componentes centrais da robótica aeroespacial (INPE, 2021).

Atualmente, avanços em aprendizado de máquina permitiram o surgimento de robôs mais avançados. Empresas como Google e Tesla passaram a investir no desenvolvimento de robôs autônomos e veículos autodirigidos. Tecnologias como o LIDAR, visão computacional e redes neurais possibilitaram que robôs móveis se locomovessem em ambientes dinâmicos de forma autônoma e eficiente.

2.2 Conceitos básicos de robôs

Um robô é um dispositivo eletromecânico capaz de realizar trabalhos de maneira autônoma ou pré-programada. Ele combina componentes mecânicos, eletrônicos e computacionais para executar tarefas específicas, com ou sem intervenção humana direta. A estrutura de um robô envolve seus atuadores, processamento e sensores. Os sensores são utilizados para perceber o ambiente ao redor, entre os mais utilizados estão os sensores ultrassônicos, LIDAR e câmeras. Esses dispositivos permitem ao robô detectar obstáculos e medir distâncias. Os atuadores são utilizados para interagir com o ambiente. Esses atuadores podem ser rodas, pernas ou garras. Já o processamento envolve realizar cálculos e enviar comandos para os atuadores.

Os robôs podem ser fixos ou móveis. Os robôs fixos possuem sua base fixa e não se locomovem. Esses robôs são geralmente utilizados na indústria, interagindo com o ambiente por meio de garras. Os robôs móveis são sistemas capazes de se locomover de forma autônoma ou controlada dentro de um ambiente. Esses robôs normalmente possuem rodas ou pernas para se locomover. Além disso, podem ter braços e garras para interagir com o ambiente.

2.2.1 Classificação dos robôs móveis

Os robôs móveis podem ser classificados com base em diferentes critérios, como o sistema de locomoção e o nível de autonomia. Os robôs com rodas são os mais comuns devido à sua simplicidade e facilidade de controle. Eles podem se deslocar em superfícies planas e apresentam menor complexidade em comparação com robôs que utilizam pernas. Os principais tipos de direção de robôs móveis com rodas incluem (ROS, 2025):

- Diferencial: cada roda é controlada de forma independente;
- Ackerman: tipo de direção utilizada em veículos automotores com mais de três rodas, na qual as rodas direcionais são alinhadas de modo a manter um único ponto de centro de curvatura;
- Omnidirecional: utiliza rodas especiais, permitindo movimento em qualquer direção sem a necessidade de girar o corpo do robô;

- Triciclo: utiliza uma configuração semelhante à de um triciclo, com uma roda frontal direcionável e duas rodas traseiras motorizadas;
- Com esteiras: utiliza esteiras no lugar de rodas, permitindo maior aderência em terrenos irregulares, comum em robôs militares e veículos de exploração.

Além dos robôs com rodas, existem também os robôs com pernas. Esses robôs são projetados para ambientes onde rodas não são eficazes, como degraus ou superfícies irregulares. No entanto, esses sistemas são mais complexos e exigem maior capacidade computacional para o controle da locomoção. Os principais exemplos incluem:

- Robôs bípedes: projetados para imitar a caminhada humana;
- Robôs quadrúpedes: inspirados em animais que possuem quatro patas;
- Robôs aracnídeos: possuem oito patas, inspirados em aranhas.

Além desses robôs terrestres, há sistemas móveis que operam em outros meios (RUSSO; CECCARELLI, 2020):

- Robôs aéreos: locomovem-se no ar. Utilizados, como exemplo, para monitoramento, mapeamento e diversas aplicações, como na agricultura e operações militares;
- Robôs aquáticos: locomovem-se dentro da água. Utilizados, como exemplo, para monitoramento e regaste submarino.

2.2.2 Níveis de autonomia em robôs móveis

Os robôs móveis podem ser classificados de acordo com o grau de autonomia, ou seja, o nível de controle e tomada de decisão que possuem. Os principais tipos são:

- Robôs teleoperados: são controlados remotamente por um operador humano. Esses robôs não tomam decisões de forma autônoma, sendo utilizados em aplicações como exploração espacial e operações militares;

- Robôs semiautônomos: possuem um grau intermediário de autonomia, realizando algumas tarefas de forma independente, mas ainda dependem da intervenção humana em alguns momentos;
- Robôs autônomos: operam de maneira independente, tomando decisões baseadas em informações do ambiente e em algoritmos.

2.3 Planejamento de trajetória

2.3.1 Tipos de planejamento de trajetórias

O planejamento de trajetória tem o objetivo de encontrar o caminho do ponto inicial até o ponto final, respeitando as restrições do robô e do ambiente. O planejamento pode ser:

- Deliberativo: o robô constrói toda a trajetória antes de iniciar o movimento. Esse método tem a vantagem de ser simples e a desvantagem de não saber lidar com mudanças no ambiente;
- Reativo: o robô planeja a trajetória completa se o ambiente for conhecido, ou parcialmente, caso navegue em um ambiente desconhecido. Entretanto, essa trajetória pode ser alterada considerando as mudanças captadas por sensores. A vantagem desse método é sua capacidade de responder rapidamente a mudanças no ambiente, sendo usado em robôs que se locomovem em ambientes dinâmicos. A desvantagem é que sua implementação é mais complexa.

2.3.2 Heurística

Na informática, a heurística é um conjunto de procedimentos e normas usados em pesquisa feita por meio da quantificação de proximidade a um determinado objetivo. (MICHAELIS, 2025). Portanto, a heurística é usada para chegar a respostas a questões complexas de forma rápida e satisfatória. Na programação, a heurística não encontra a melhor solução, mas sim, uma solução satisfatória. Essa solução é ideal para situações complexas em que testar todas as combinações seria inviável. Neste trabalho, o custo do caminho é relacionado com a

distância. A seguir, são apresentadas as principais distâncias usadas na heurística de navegação de robôs, onde (x_1, y_1) é a posição inicial da trajetória e (x_2, y_2) é a posição final:

- Manhattan: equivale a soma das distâncias na vertical e horizontal. Em um ambiente com duas dimensões (2D), a distância manhattan é calculada como abaixo:

$$d_m = |x_2 - x_1| + |y_2 - y_1| \quad (1)$$

- Euclidiana: equivale a distância em linha reta entre dois pontos. Em um ambiente 2D, a distância euclidiana é calculada como abaixo:

$$d_e = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2)$$

No presente trabalho, o robô desloca-se em um ambiente composto por uma matriz de células quadradas e não se locomove diretamente na diagonal. Essa limitação de movimentação se assemelha ao conceito de distância Manhattan. Portanto, essa distância foi escolhida devido a uma decisão de projeto, mas que a distância euclidiana também poderia ter sido usada.

2.3.3 Algoritmos de busca de caminhos

Os algoritmos de busca de caminho andam juntos à navegação autônoma. Eles são responsáveis por encontrar uma rota entre dois pontos. Além disso, os algoritmos de busca podem ter mais de um critério para escolher esse caminho, como a distância, o tempo e os obstáculos. Os algoritmos mais populares são o algoritmo de Dijkstra, a busca em largura e o algoritmo A*.

O algoritmo de Dijkstra foi desenvolvido por Edsger Dijkstra em 1956. Esse algoritmo é um dos mais conhecidos e utilizados para encontrar o caminho mais curto em grafos ponderados. Ele garante que, ao final da execução, sempre será encotrado o caminho mais curto. O algoritmo de Dijkstra é utilizado em muitos

sistemas de otimização de rotas, como *Global Positioning System* (GPS, do inglês, sistema de posicionamento global) (SINGAL; CHHILLAR, 2014).

Na busca em largura, as posições são exploradas por níveis, começando pelas mais próximos da origem e expandindo para as seguintes. Para avançar para o próximo nível, o algoritmo precisa ter encontrado o caminho mais curto do nível anterior. A principal vantagem desse algoritmo é a garantia de sempre encontrar o caminho mais curto. Já a desvantagem, é a necessidade de testar todas as possibilidades (APPLIED AI TEAM, 2025).

O algoritmo A* é um dos mais utilizados para o planejamento de trajetória em robótica devido à sua eficiência na busca de caminhos ótimos. Ele combina qualidade da solução encontrada com eficiência computacional. Essa combinação torna o algoritmo A* uma boa abordagem para o planejamento de trajetórias. Por ser o algoritmo escolhido para o presente trabalho, ele é aprofundado na seção abaixo.

2.3.4 Algoritmo A*

O algoritmo A* é um dos mais utilizados na busca de caminhos. Esse algoritmo faz uso de uma heurística. Uma heurística muito comum é baseada na distância, sendo a distância Manhattan uma muito utilizada. O algoritmo é baseado em uma função de avaliação $f(n)$, definida como:

$$f(n) = g(n) + h(n) \tag{3}$$

onde:

- $f(n)$: função do custo total;
- $g(n)$: custo da posição inicial até a posição atual;
- $h(n)$: uma estimativa de custo da posição atual até a posição final, obtida por uma heurística.

O custo mencionado acima é relacionado com a distância. Neste trabalho, o $g(n)$ representa a distância percorrida da posição inicial até a posição atual, ou seja, algo que já ocorreu. Já o $h(n)$ é algo que ainda vai ocorrer, uma estimativa de distância da posição atual até o destino.

O algoritmo A* inicia-se a partir de uma posição inicial que se expande para todos os vizinhos. Para cada posição explorada, a função $f(n)$ é calculada. As posições são armazenados em uma fila de prioridade, de modo que a posição com o menor valor de $f(n)$ seja expandida primeiro. Esse processo continua até que o destino seja alcançado. O caminho encontrado é considerado ótimo para a heurística adotada (EVOLVERS, 2025).

2.4 Odometria e controle por tempo de ativação

A navegação de um robô móvel exige um método para controlar sua movimentação e estimar sua posição no ambiente. Em muitos robôs móveis, essa estimativa é feita com base na odometria. Esse método utiliza sensores para medir o deslocamento das rodas e calcular a posição do robô ao longo do tempo. No entanto, em sistemas mais simples, pode-se adotar uma abordagem baseada no tempo de ativação dos motores, sem a necessidade de sensores adicionais. Cada uma dessas abordagens possui vantagens e limitações, dependendo do nível de precisão exigido e dos recursos disponíveis.

A odometria é uma técnica amplamente utilizada na robótica móvel para estimar a posição e a orientação de um robô com base no deslocamento de suas rodas (ADDISON, 2025). Esse método é baseado na contagem do número de rotações das rodas e no cálculo da distância percorrida a partir dessas informações. O sistema de odometria geralmente utiliza encoders nos motores, que registram as rotações das rodas e fornecem dados sobre o deslocamento linear e angular das rodas. Com esses dados, é possível estimar a posição do robô no ambiente a cada instante. A equação da odometria para um robô pode ser descrita como:

$$d = r \times \alpha \tag{4}$$

onde:

- d é a distância percorrida;
- r é o raio da roda;
- α é o ângulo de rotação da roda em radianos.

A odometria apresenta como vantagem a facilidade de estimar a posição do robô em tempo real. Isso permite um controle mais eficiente da trajetória do robô. No entanto, essa técnica também possui algumas desvantagens. Ela depende de sensores adicionais, como encoders, o que aumenta a complexidade do *hardware* e eleva o custo do sistema. Além disso, sua precisão pode ser muito reduzida em terrenos irregulares ou escorregadios. Nesses ambientes, o deslocamento real do robô pode não corresponder ao deslocamento calculado.

O controle por tempo de ativação é uma abordagem mais simples para movimentação de robôs móveis. Os motores são ativados por um tempo pré-determinado para realizar o deslocamento. Esse método não utiliza sensores para medir o deslocamento do robô e confia apenas no tempo de acionamento dos motores para estimar a posição. Para usar esse método, é necessário conhecer a frequência do motor. Se a velocidade dos motores for constante, o tempo de acionamento pode ser calculado por:

$$t = d / v \tag{5}$$

onde:

- t é o tempo de ativação dos motores;
- d é a distância percorrida;
- v é a velocidade do robô.

O controle por tempo de ativação apresenta como principal vantagem a sua implementação simples, o que contribui para a redução da complexidade do *hardware*. Contudo, esse método possui limitações. Entre as desvantagens, destaca-se a incapacidade de corrigir erros durante a navegação. Além disso, é necessário realizar uma verificação constante dos motores para garantir que estão sendo corretamente alimentados.

2.5 Principais desafios na navegação de robôs móveis

A navegação de robôs móveis é uma área desafiadora da robótica. A capacidade de um robô se locomover em diferentes ambientes depende de uma série de fatores. Entre esses fatores estão a precisão da locomoção, a percepção do

ambiente, o planejamento de trajetória e a capacidade de adaptação a mudanças inesperadas.

Mesmo em ambientes previamente mapeados e conhecidos, como no presente estudo, diversas questões precisam ser consideradas. A seguir, são discutidos os principais desafios enfrentados na navegação de robôs móveis.

2.5.1 Execução de trajetória

O planejamento de trajetória é essencial para garantir que o robô consiga alcançar seu destino. No presente trabalho, a trajetória é calculada utilizando o algoritmo A*, que encontra um caminho ótimo entre o ponto de partida e o ponto de destino. Entretanto, há desafios que podem comprometer a execução correta da trajetória planejada, tais como:

- Acúmulo de erros: pequenas imprecisões no deslocamento do robô podem se somar ao longo do trajeto. Isso resulta em desvios significativos em relação ao caminho esperado, dependendo da complexidade da rota;
- Deslizamento das rodas: o movimento do robô pode ser afetado por diferentes atritos da superfície;
- Variações na tensão da bateria: a movimentação é baseada no tempo de ativação dos motores e não em sensores de feedback. Variações na tensão da bateria podem alterar a velocidade dos motores;
- Desgaste dos componentes mecânicos: motores, rodas e engrenagens podem sofrer desgaste ao longo do tempo, alterando a movimentação do robô.

2.5.2 Eficiência energética

A eficiência energética é um aspecto fundamental para a robótica móvel, principalmente em aplicações que exigem operações prolongadas sem possibilidade de recarga. O consumo de energia do robô depende de fatores como o tipo de motor utilizado. Servomotores consomem energia proporcionalmente ao esforço necessário para movimentar o robô. Além disso, é importante considerar o tamanho

e peso do robô. Um robô mais pesado exige maior potência para se deslocar, aumentando o consumo da bateria.

No presente trabalho, a eficiência energética não foi um problema. Foi utilizada uma bateria de 12 V (volts), suficiente para alimentar o robô e os motores operando em sua potência máxima. Em determinado momento, observou-se uma queda na potência dos motores no final do percurso, indicando o início da descarga da bateria. Durante os experimentos, que duraram aproximadamente 2 horas e 30 minutos, foi necessário carregar a bateria apenas uma vez.

2.5.3 Segurança na navegação

Outro desafio, é garantir a segurança do robô e das pessoas ao redor. O presente estudo é realizado em um ambiente conhecido e controlado. No entanto, sistemas reais de navegação autônoma devem ser projetados para evitar acidentes e garantir a segurança.

Em 1950, o escritor Isaac Asimov criou a obra “Eu, Robô”, em que imaginou um mundo onde robôs e seres humanos coexistem (GIZMODO UOL, 2025). Para regular essa convivência, o autor apresentou as três leis da robótica, com o objetivo de garantir a segurança dos seres humanos. Apesar de serem leis fictícias, elas possuem utilidade na robótica, sendo interessante que os robôs sigam essas regras. Essas leis são as seguintes:

- Um robô não pode ferir um ser humano ou permitir que um ser humano sofra algum dano;
- Um robô deve obedecer às ordens dadas por seres humanos, exceto quando essas ordens entrarem em conflito com a Primeira Lei;
- Um robô deve proteger sua própria existência, desde que isso não entre em conflito com a Primeira ou Segunda Leis.

3 METODOLOGIA

3.1 Visão geral

A abordagem baseia-se na implementação prática de um sistema de planejamento de trajetória para um robô móvel. Esse robô utilizou o algoritmo A* em um ambiente previamente mapeado. O foco principal foi na aplicação prática do algoritmo em um robô de baixo custo, avaliando sua eficácia na navegação sem o uso de sensores externos.

Na primeira etapa, a implementação física foi realizada. Foram utilizados um Arduino Mega, servomotores SM-S4306R acoplados nas rodas dianteiras e uma bateria de 12 V para alimentar o Arduino. Um conversor de tensão 12 V para 5 V foi usado para fornecer a tensão exigida pelos servomotores. A comunicação sem fio com o sistema foi feita via módulo Bluetooth HC-06.

Na segunda etapa, o ambiente foi representado por uma matriz de 10 x 10 posições. O ambiente foi previamente definido e representado no código do Arduino. O algoritmo A* foi implementado para calcular a melhor rota entre um ponto de início e destino, utilizando a distância Manhattan para fazer as estimativas. O objetivo dessa etapa foi verificar se o algoritmo calcula corretamente a trajetória.

Na terceira etapa, foram realizados testes experimentais para avaliar o desempenho do sistema. O robô executou múltiplas trajetórias. A precisão do deslocamento foi analisada comparando a trajetória planejada com a real.

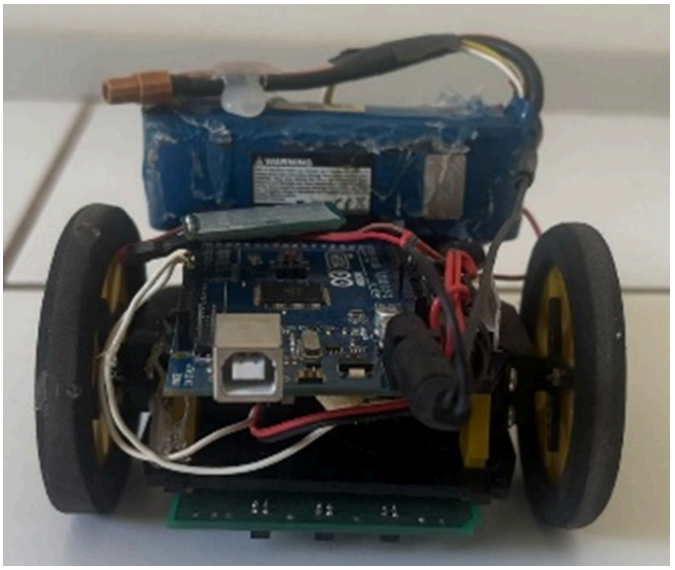
3.2 Componentes do robô

Nesta seção, são apresentados os principais componentes do robô usados neste trabalho, com destaque para os componentes eletrônicos e computacionais. Com relação a parte mecânica, vale mencionar que o chassi do robô é de material acrílico, tem formato retangular com dimensões de 16 cm x 9 cm. Com todos os componentes eletrônicos usados, o seu peso é um pouco menos que 1 kg.

Da parte mecânica, um item que merece atenção especial é a roda de tração. O robô possui duas rodas de tração dianteiras e independentes uma da outra. Essas rodas são construídas de material plástico com envolvimento de borracha de etil vinil acetato (EVA) na superfície de contato com o solo. Uma dimensão importante neste

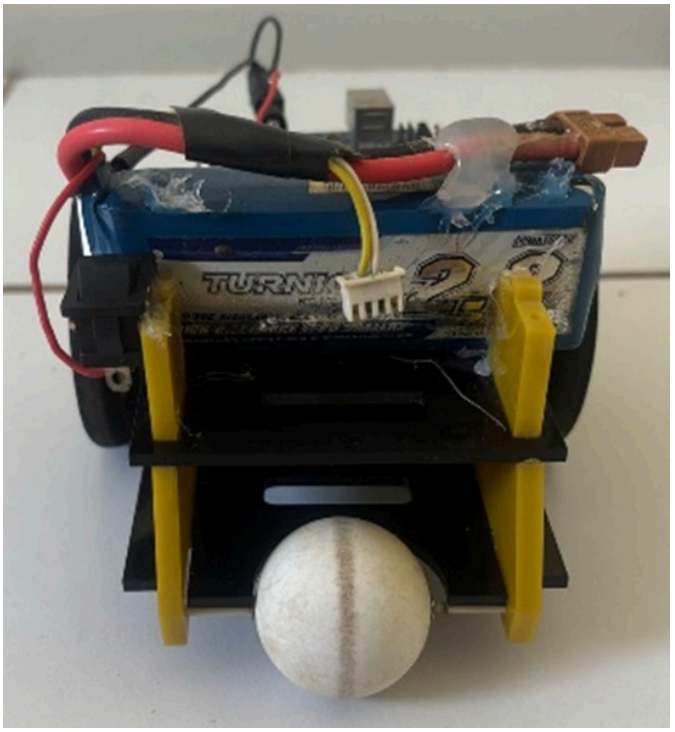
estudo é o tamanho do diâmetro da roda dianteira, neste caso se tem um valor de 7,4 cm. Além disso, o robô possui uma roda traseira esférica de teflon, que não é acoplada a um motor. Nas Fig. 1 a 7, são mostradas as fotos do robô e das rodas.

Figura 1 - Visão frontal do robô



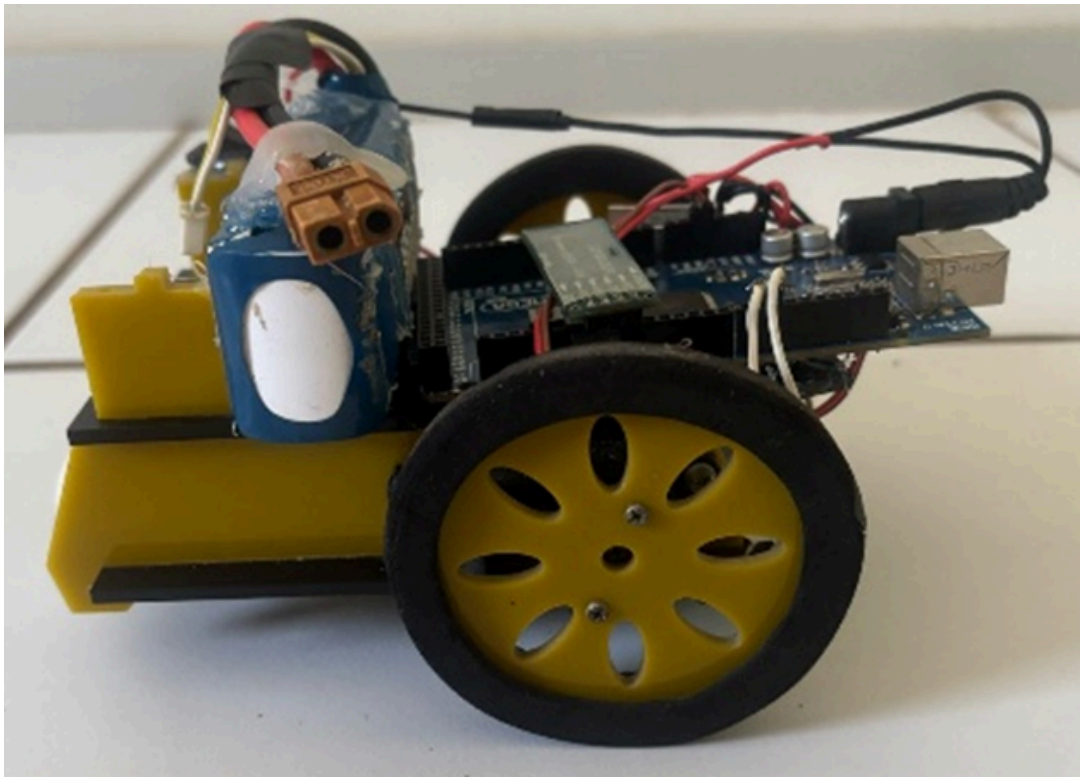
Fonte: Autor

Figura 2 - Visão traseira do robô



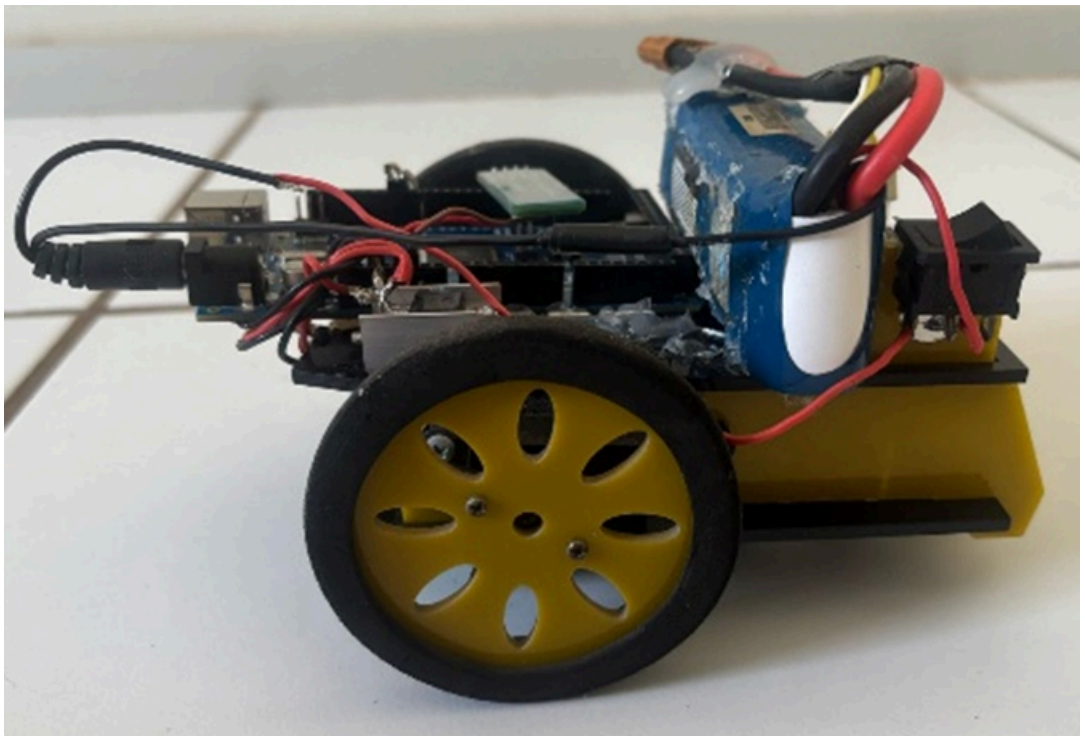
Fonte: Autor

Figura 3 - Lado direito do robô



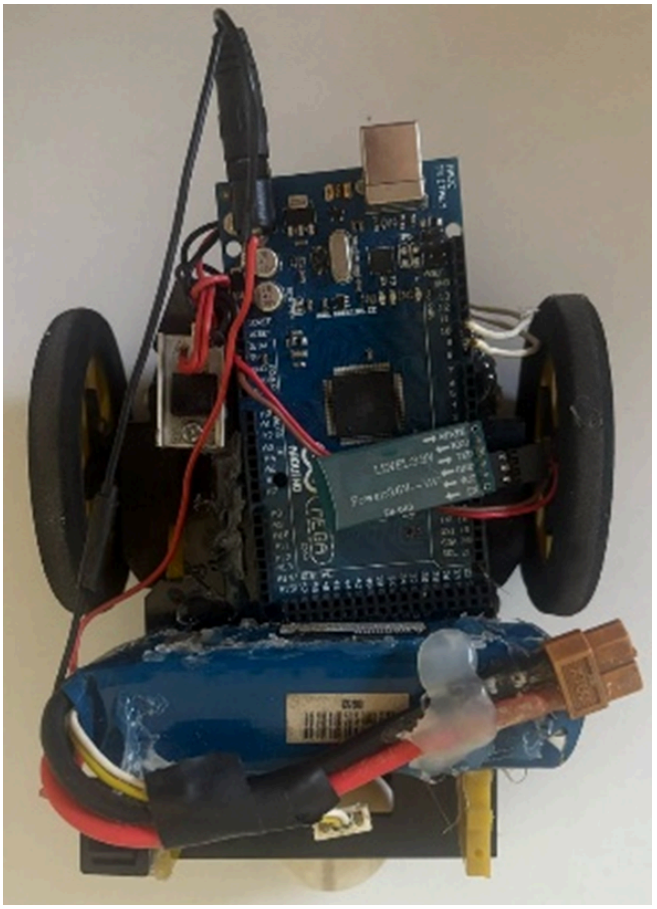
Fonte: Autor

Figura 4 - Lado esquerdo do robô



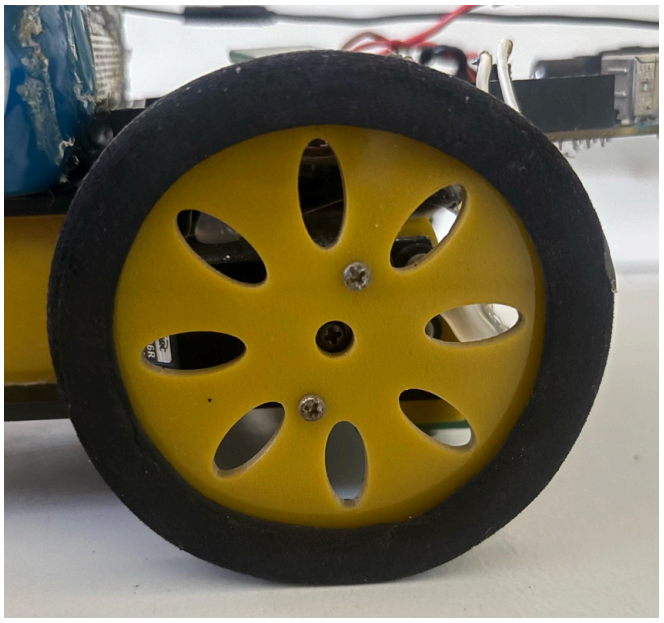
Fonte: Autor

Figura 5 - Visão superior do robô



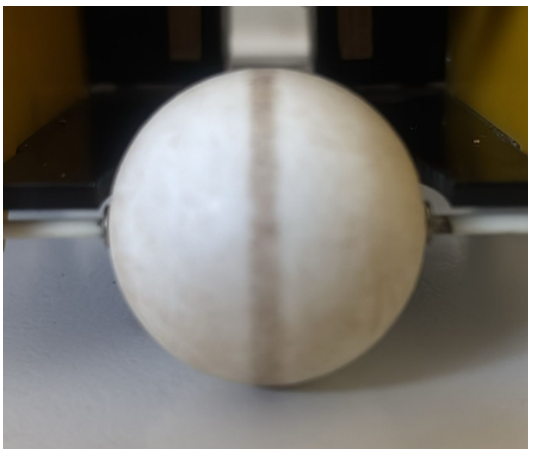
Fonte: Autor

Figura 6 - Roda dianteira



Fonte: Autor

Figura 7 - Roda traseira



Fonte: Autor

A implementação do planejamento de trajetória em um robô físico exige um *hardware* adequado para processar o algoritmo e controlar os componentes responsáveis pelo movimento. Nas subseções seguintes, são abordados todos os componentes utilizados no robô.

3.2.1 Arduino

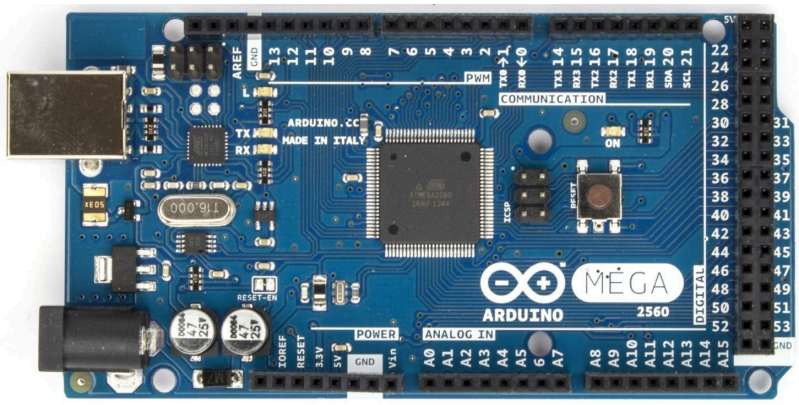
O Arduino é uma plataforma de prototipagem composta por uma placa com microcontrolador programável, que permite o desenvolvimento de diversas aplicações. Esse componente ajuda os programas desenvolvidos a interagirem com o mundo real. O baixo custo desse componente em comparação com outras plataformas de prototipagem faz com que seja ideal para estudantes. As principais vantagens do Arduino são:

- Baixo custo: comparado a outras plataformas de desenvolvimento, como Raspberry Pi;
- Facilidade de programação: a linguagem utilizada é baseada em C++, por meio do ambiente de desenvolvimento integrado (IDE) do Arduino, o que facilita o aprendizado;
- Compatibilidade com diversos componentes: o Arduino suporta a maioria dos componentes eletrônicos necessários para a implementação de projetos de robótica.

O Arduino possui pinos de entrada e saída digitais e analógicas. Os pinos digitais são utilizados para enviar sinais binários. Os pinos analógicos são usados para ler ou fornecer uma faixa de valores, sendo úteis em uma variedade de sensores. Para carregar os códigos, foi utilizado um cabo *Universal Serial Bus* (USB, do inglês, porta serial universal) tipo B com comprimento de 30 cm. Primeiramente, é realizada a compilação do código para a verificação de erros. Na sequência, é realizado o carregamento do código da IDE para o Arduino. Existem diversos modelos de Arduino. A quantidade de pinos varia de acordo com o modelo da placa. Os modelos mais populares para projetos de robótica são:

- Arduino Nano: mais compacto e ideal para sistemas muito leves;
- Arduino Uno: ideal para aplicações simples. Possui uma quantidade intermediária de pinos;
- Arduino Mega: possui maior quantidade de pinos e memória, adequado para projetos mais complexos que exigem múltiplas conexões.

Figura 8 - Arduino Mega



Fonte: http://Arduino.cc/en/uploads/Main/ArduinoMega2560_R3_Fronte.jpg

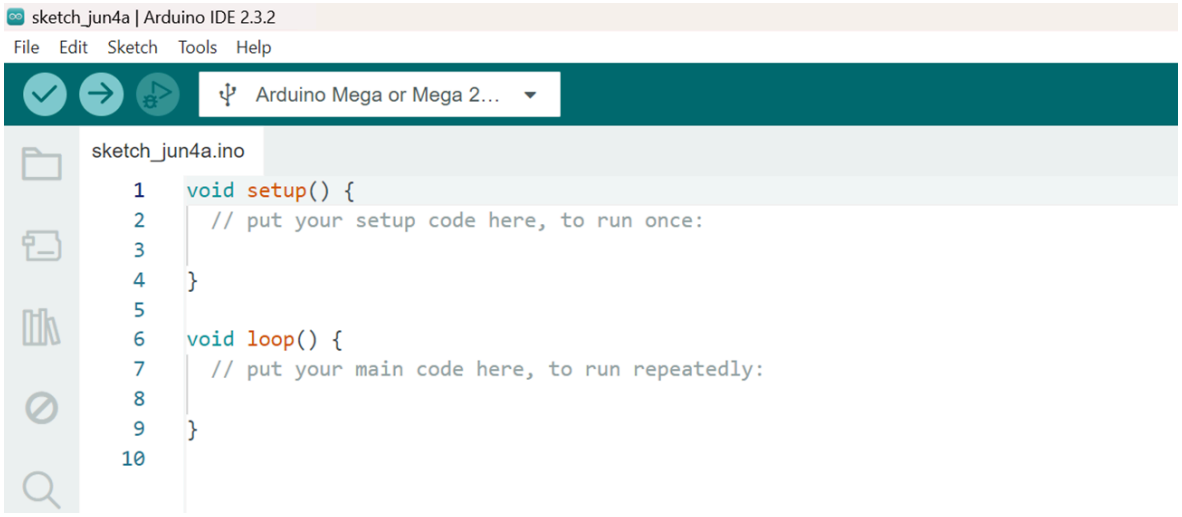
Para o presente trabalho, poderia ser utilizado o Arduino Uno. Este Arduino é mais do que suficiente para calcular o algoritmo A* e controlar os movimentos do robô. No entanto, no laboratório havia disponível apenas o Arduino Mega. Desta forma, foi utilizado o modelo mais robusto.

Neste trabalho, o Arduino é o componente principal, em que foi programada a lógica do algoritmo A* e também é responsável pelo controle das duas rodas motorizadas do robô. Além disso, no Arduino, são conectados os dispositivos

eletrônicos, nos quais são feitas a alimentação e a comunicação desses componentes.

Para desenvolver os programas, foi utilizado a linguagem de programação C++ na IDE do Arduino. Essa IDE possui obrigatoriamente duas rotinas (Fig. 9). A primeira rotina é chamada de *setup*, em que é colocado um código que será executado apenas uma vez, logo após iniciar o programa. Já a segunda rotina, é chamada de *loop*, que será chamada após o *setup*. A rotina *loop* executa o código repetidamente de forma infinita (ARDUINO, 2025).

Figura 9 - IDE do Arduino



Fonte: Autor

3.2.2 Servomotor SM-S4306R

O Servomotor SM-S4306R é um servo de rotação contínua que gira 360° em ambos os sentidos. Esse modelo controla a velocidade de rotação e a direção, destacando-se pela sua precisão. Ele permite um controle mais preciso dos movimentos, além de exigirem pouca corrente para se movimentarem. Ele opera com tensões entre 4,8 V a 6 V (SPRINGRC, 2025).

O SM-S4306R possui três fios (Fig. 10): o fio vermelho recebe a alimentação de 4,8 V a 6 V; o fio preto corresponde ao *ground* (GND, do inglês, tensão elétrica nula); o fio branco recebe os comandos do Arduino para movimentação. Esse controle é feito via *pulse width modulation* (PWM, do inglês, modulação por largura de pulso), uma técnica para controlar a quantidade de energia fornecida a um

dispositivo sem alterar a tensão. Essa técnina controla a velocidade dos motores por meio da diferença dos pulsos de tensão alta e baixa.

Figura 10 - Servomotor SM-S4306R

Fonte: https://img.alicdn.com/imgextra/i1/265677212/TB2pgF9a8YxQeBjSszeXXa0spXa_!!265677212.jpg

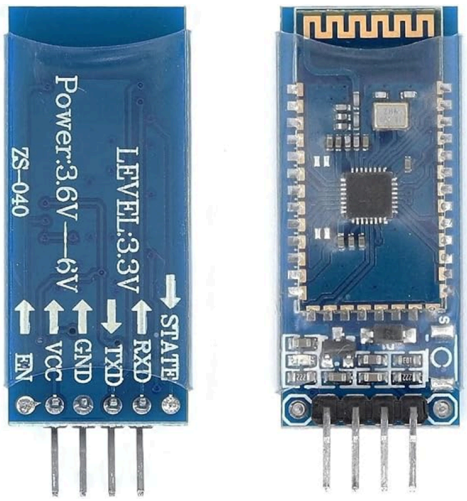
3.2.3 Módulo Bluetooth HC-06

Esse módulo é um dispositivo de comunicação sem fio muito usado em projetos de robótica. Ele permite a conexão do Arduino com outro dispositivo via Bluetooth. Por ser de baixo custo e simples, ele é uma das escolhas mais populares para controle remoto de robôs. De acordo com o manual do módulo e testes realizados, seu alcance máximo é de 20 m, mesmo em ambientes com obstáculos tipo paredes de alvenaria.

Neste projeto, a posição inicial e a posição final foram definidas manualmente em um aparelho celular e enviadas para o robô usando o canal do Bluetooth. Para enviar esses dados do celular para o robô, foi usado o módulo Bluetooth HC-06. Ele permite a conexão do Arduino com um dispositivo externo sem fio. Esse dispositivo opera por meio do protocolo de comunicação *Universal Asynchronous Receiver / Transmitte*r (UART, do inglês, receptor / transmissor assíncrono universal). O Arduino envia e recebe os dados via portas seriais e converte os sinais do módulo

Bluetooth em sinais seriais. O HC-06 possui um pino chamado TX (pino serial de transmissão) e outro chamado RX (pino serial de recepção). O módulo precisa ser alimentado na faixa de tensão de 3,6 V a 6 V, sendo essa alimentação normalmente proveniente de um dos pinos 5 V do Arduino. Além disso, o pino RX do módulo HC-06 precisa ser conectado ao pino TX do Arduino e o pino TX do módulo HC-06 conectado ao pino RX do Arduino (RUCKSIKAAR, 2025). Dessa forma, o módulo HC-06 consegue enviar e receber informações do Arduino.

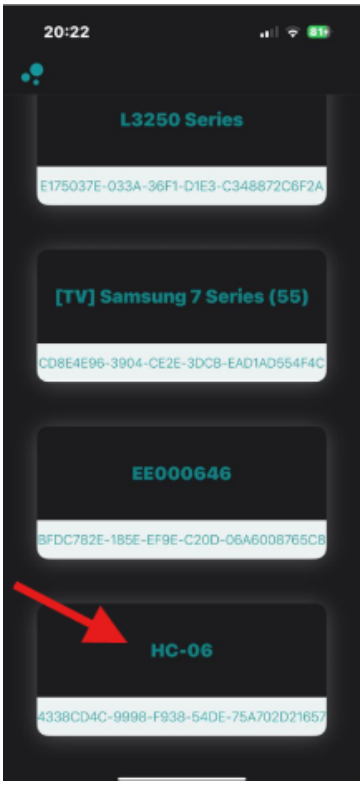
Figura 11 - Módulo HC-06



Fonte: https://m.media-amazon.com/images/I/61UbnzEx9TL._AC_SX569_.jpg

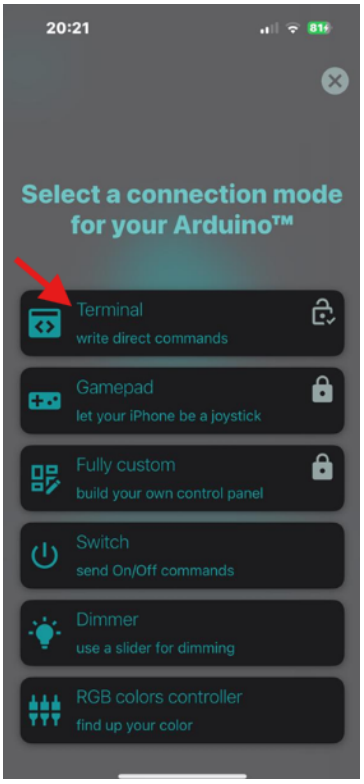
Para enviar os dados de posição, utilizou-se um aplicativo genérico baixado da App Store chamado “Bluetooth for Arduino”, que possui uma interface para conectar ao módulo HC-06 (Fig. 12). Para enviar comandos pelo celular, é necessário que este esteja com o Bluetooth ativado e o módulo HC-06 ligado. Ao ligar este módulo, uma luz vermelha irá piscar continuamente. Ao conectar ao módulo HC-06 na página inicial do aplicativo, a luz vermelha deixará de piscar e ficará sempre acessa, indicando que a conexão foi realizada com sucesso. Em seguida, é selecionado o modo de envio dos dados para o Arduino. A informação a ser enviada via canal do Bluetooth é realizada na opção desse aplicativo chamada de terminal (Fig. 13). Na sequência, são pedidas as coordenadas (x, y) das posições inicial e final (Fig. 14). A funcionalidade que pede a digitação das coordenadas foi desenvolvida neste trabalho e programada no Arduino como um módulo inicial do programa de execução do algoritmo A*.

Figura 12 - Tela inicial do aplicativo “Bluetooth for Arduino”



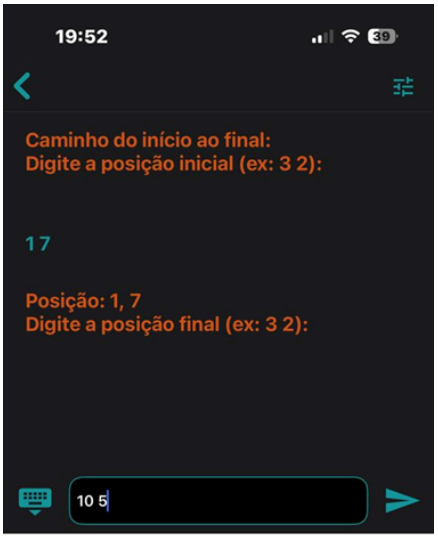
Fonte: Autor

Figura 13 - Segunda tela do aplicativo “Bluetooth for Arduino”



Fonte: Autor

Figura 14 - Terceira tela do aplicativo “Bluetooth for Arduino”



Fonte: Autor

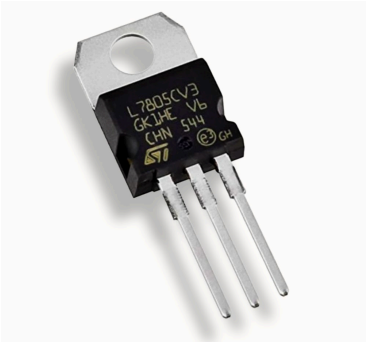
3.2.4 Alimentação

Foi usada uma bateria de 12 V (Fig. 15), sendo capaz de alimentar todo o robô trabalhando na sua potência máxima. Essa bateria foi conectada diretamente na entrada de alimentação do Arduino, fornecendo a alimentação necessária para o funcionamento do componente. Para alimentar os motores, foi necessário converter a tensão de 12 V em 5 V, por meio de um conversor (Fig. 16). Isso ocorreu, pois os motores trabalham na faixa de tensão de 5 V. Esse conversor possui três pinos: o pino do meio é conectado ao GND; o pino da esquerda (olhando de frente) é conectado à alimentação de 12 V; o terceiro pino permite sair uma tensão de 5 V.

Figura 15 - Bateria 12 V

Fonte: https://novatronicec.com/wp-content/uploads/2022/06/Turnigy-2200mAh-3S-30C-Lipo-Pack.jpg

Figura 16 - Conversor 12 V para 5 V



Fonte: https://www.electricsmart.in/public/uploads/all/kKjq9nZe9Z5tMypuFYSDFVOqFglyB1N11OY7OzKA.jpg

Para evitar o descarregamento da bateria, foi usado um interruptor que possui três pinos. O pino do meio corresponde ao GND, no qual é conectado o fio de 0 V. O fio de 12 V é conectado no pino localizado abaixo do botão de ligar. Isso faz com que a conexão seja ligada ou desligada conforme a posição do interruptor.

Figura 17 - Interruptor

Fonte: https://images.nexusapp.co/assets/a9/37/d3/246083506.jpg

3.3 Implementação do sistema de navegação do robô

Foi utilizada a IDE do Arduino para realizar toda a programação. Essa IDE foi instalada em um computador do modelo Lenovo Ideapad 1. Esse computador possui sistema operacional Windows 11, *Random Access Memory* (RAM, do inglês, Memória de Acesso Aleatório) de 16 Gigabytes (GB) e processador AMD Ryzen 7. A seguir, são detalhados aspectos dessa implementação, incluindo o ambiente de testes, a implementação do algoritmo A* e a configuração da movimentação do robô.

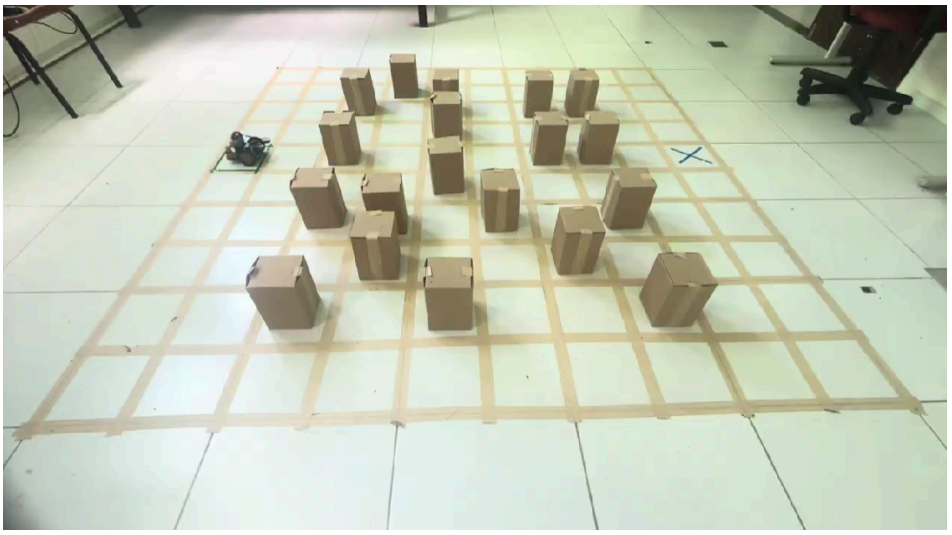
3.3.1 Ambiente de testes

Os testes foram realizados no Laboratório de Inovação Tecnológica (Lab. INOVTEC) na Universidade Federal do Maranhão (UFMA), localizado no prédio de Centro de Ciências Exatas e Tecnologias (CCET). Dentro desse laboratório, tem uma sala, em que foram montados os cenários presentes neste trabalho.

Os cenários foram montados no chão desse laboratório, cujo piso era quase totalmente liso e regular, contando com pequenas rugosidades. Os cenários foram formados por um quadrado com 10 x 10 células, em que cada célula tinha um tamanho de 22,85 cm x 22,85 cm. Foram aplicadas marcações visuais, feitas com fita crepe de cor marrom a fim de separar as células. Essas marcações garantiram que as delimitações das células fossem visíveis, facilitando a avaliação da precisão do robô. Também foram adicionados obstáculos fixos representados por 19 caixas vazias com 20 cm de largura, 20 cm de comprimento e 40 cm de altura. Cada caixa ocupava uma única célula. Esses obstáculos foram dispostos de forma aleatória para simular diferentes cenários de navegação.

Os cenários possuiam uma posição inicial e uma posição final. O robô começava na posição inicial e tinha o objetivo de alcançar a posição final desviando dos obstáculos. Para indicar a posição inicial, foi adicionado, com fita isolante verde, um quadrado dentro de uma célula. Também foi adicionado um ‘X’ com fita isolante azul dentro de uma célula para indicar a posição final. Segue abaixo um dos cenários testados:

Figura 18 - Exemplo de cenário



Fonte: Autor

O mesmo ambiente utilizado no mundo real foi replicado no código do Arduino. O ambiente de testes foi representado neste dispositivo por uma matriz 2D de 10 x 10 posições, garantindo que o ambiente seja conhecido pelo robô desde o início. As células com obstáculos foram representadas pelo número um e células livres representadas pelo número zero. Essas células correspondem ao deslocamento de um único passo no algoritmo A*. No código, foram definidas as mesmas posições inicial e final do ambiente físico por meio de coordenadas (x, y). Essas posições não são indicadas na matriz, mas sim, dentro do código por meio de variáveis.

Figura 19 - Matriz do cenário

```
int grid[GRID_WIDTH][GRID_HEIGHT] = {
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
  {0, 0, 1, 0, 1, 0, 1, 1, 0, 0},
  {0, 0, 0, 1, 1, 0, 0, 0, 0, 0},
  {0, 0, 1, 0, 0, 0, 1, 1, 0, 0},
  {0, 0, 0, 0, 1, 0, 0, 0, 0, 0},
  {0, 0, 1, 1, 0, 1, 0, 1, 0, 0},
  {0, 0, 0, 1, 0, 0, 1, 0, 0, 0},
  {0, 0, 1, 0, 1, 0, 0, 1, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
};
```

Fonte: Autor

3.3.2 Implementação do código

Primeiro, são definidas várias variáveis que indicam as características do robô, como o diâmetro da roda, a distância entre as rodas e o tempo para uma rotação completa dos motores. Também é definida a matriz do ambiente, explicada no item anterior. Em seguida, é declarada uma estrutura de dados chamada Node. Essa estrutura armazena, para cada célula visitada, a posição (x, y), os custos “g”, “h” e “f” (presentes na fórmula do algoritmo A*), além de ponteiros para o nó pai (utilizado na reconstrução do caminho) e para o próximo nó.

Ao executar o código, o usuário precisa informar as posições inicial e final (conforme explicado na Seção 3.2.3). Essas posições são verificadas para impedir

que o usuário digite uma posição com obstáculo ou fora dos limites da matriz. A partir das coordenadas informadas, é executada a rotina do algoritmo A*, que realiza a busca pelo caminho ótimo. Quando o caminho for encontrado, é chamada outra rotina que converte o caminho retornado pela função anterior em movimentos que devem ser executados sequencialmente pelo robô. Cada tipo de movimento (frente, trás, horário e anti-horário) possui sua própria rotina, onde é definida a direção das rodas e é feito o cálculo do tempo de ativação dos motores. Os detalhes dos movimentos do robô são explicados a seguir.

3.3.3 Configuração da movimentação do robô

O robô pode realizar movimentos de translação e rotação, assim ele pode mover-se nos sentidos: frente, trás, horário e anti-horário. Cada servomotor é associado a uma roda, sendo controlado diretamente pelo Arduino. A movimentação foi baseada na ativação dos servos por um tempo fixo. Nos movimentos para frente e para trás, cada ativação corresponde a um deslocamento equivalente ao tamanho de uma célula da matriz do ambiente. Caso o robô precise mover-se para uma célula ao lado, ele precisará primeiro fazer um movimento de 90° no sentido horário ou anti-horário e mover para frente em seguida. Contudo, entre esses movimentos de rotação e translação, os motores são desativados momentaneamente, causando uma pequena e rápida parada no movimento do robô. Essa pequena parada também ocorre quando o robô chega em uma célula, mesmo que o movimento seja de translação para outra célula a frente. Esses intervalos entre os deslocamentos ocorrem para evitar deslizamentos e sobreposição de movimentos, reduzindo o acúmulo de erros.

A rotina do algoritmo A* calcula o caminho do ponto inicial até o ponto final, retornando uma sequência de coordenadas. Em seguida, é feita a conversão dessas coordenadas em movimentos a serem executados sequencialmente pelos servos motores. As características do controle baseado no tempo de ativação dos motores são explicadas a seguir:

- Movimentação para frente e para trás: ambos os servos giram na mesma direção e velocidade. Nos experimentos práticos, verificou-se que os motores não realizavam o mesmo movimento, ainda que os comandos fossem iguais,

provocando um movimento do robô diferente de uma linha reta. Por isso, houve a necessidade de ajustar experimentalmente as velocidades das rodas no código, a fim de garantir a correção desse erro do *hardware* e o robô poder deslocar-se em linha reta;

- Movimentação para direita e para esquerda: é composto por dois movimentos: um movimento de rotação de 90°, na qual as rodas giram em direções contrárias, mas com mesma velocidade; e um deslocamento para frente. Também houve a necessidade de corrigir experimentalmente as velocidades das rodas para garantir que o robô rotacione exatamente 90°, pelo mesmo motivo explicado no movimento de translação.

Para calcular o tempo de acionamento dos motores para percorrer uma certa distância nos movimentos para frente ou para trás, é necessário conhecer a circunferência da roda dianteira, a distância que vai ser percorrida e o tempo de um giro completo da roda. Esse período da roda pode ser obtido pelo manual do fabricante do servomotor SM-S4306R, que informa a frequência máxima do motor, em rotações por minutos (RPM), quando submetido a uma determinada tensão (SPRINGRC, 2025). O cálculo do tempo de acionamento é feito em duas etapas, conforme as Equações 6 e 7 mostradas abaixo:

$$R = d / c \tag{6}$$

onde:

- R = rotações necessárias da roda;
- d = distância para percorrer;
- c = circunferência da roda.

$$t = T \times R \tag{7}$$

onde:

- t = tempo de acionamento do motor;
- T = tempo de um giro completo da roda.

Utilizando os dados do robô deste trabalho, a seguir são realizados cálculos para encontrar o tempo de acionamento das rodas dianteiras a fim de que o robô realize o movimento para frente ou para trás equivalente ao tamanho de uma célula dos cenários de testes adotados neste trabalho:

- Circunferência da roda: c ≈ 23,24 cm;
- Tempo de um giro completo da roda: T ≈ 1,4 s (tensão fornecida de 4,8 V);
- Distância para percorrer: d = 22,85 cm (tamanho de uma célula);
- Rotações necessárias:
  R = 22,85 / 23,24 ≈ 0,98 rotações;
- Tempo de acionamento:
  t = 1,4 * 0,98 ≈ 1,37 s.

Já para calcular o tempo de acionamento dos motores para o robô fazer um giro de 90° nas movimentações horária e anti-horária, além de ter conhecimento das informações anteriores do robô, também é necessário saber a distância entre as rodas dianteiras (diâmetro da curvatura). O cálculo do tempo de acionamento é feito em três etapas, conforme as Equações 8, 9 e 10 mostradas abaixo:

$$A = \pi \times D / 4 \quad (8)$$

$$R = a / c \quad (9)$$

$$t = T \times R \quad (10)$$

onde:

- A = arco de 90°;
- D = distância entre as rodas dianteiras.

Utilizando os dados do robô deste trabalho, a seguir são realizados cálculos para encontrar o tempo de acionamento das rodas dianteiras a fim de que o robô realize um giro de 90° nos sentidos horário ou anti-horário:

- Circunferência da roda: c ≈ 23,24 cm;
- Tempo de um giro completo da roda: T ≈ 1,4 s (tensão fornecida de 4,8 V);

- Distância entre as rodas dianteiras: D = 11,8 cm;
- Arco de 90°:

  A = π * 11,8 / 4 ≈ 9,26 cm;
- Rotações necessárias:

  R = 9,26 / 23,24 ≈ 0,4 rotações;
- Tempo de acionamento do motor:

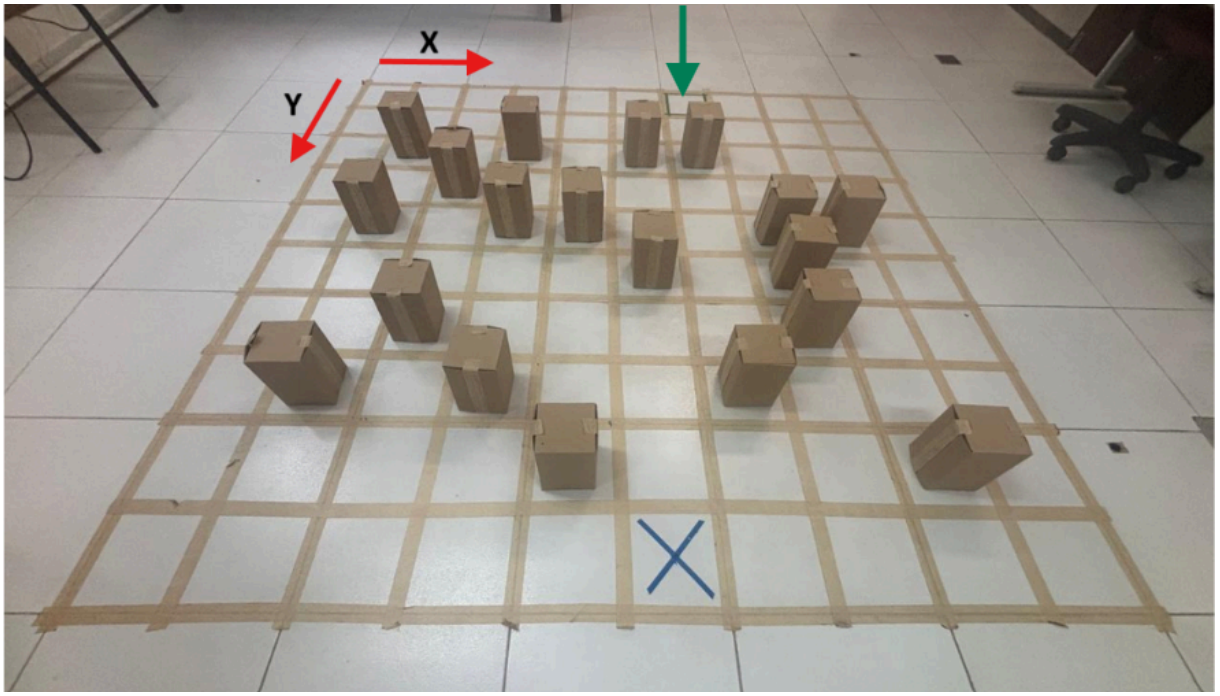
  t = 1,4 * 0,4 ≈ 0,56 s.

4 ANÁLISE DOS TESTES

Neste capítulo, são apresentados os testes práticos realizados com o robô móvel construído. O objetivo foi verificar a eficácia do robô em um ambiente físico previamente mapeado, sem o uso de sensores de percepção. Os testes buscaram validar a capacidade do sistema em planejar e executar trajetórias precisas em um cenário com obstáculos fixos.

Foram definidos 10 cenários distintos e, para cada um deles, foram feitas 15 execuções. Nos diferentes cenários, variaram-se os pontos de origem (marcação com um quadrado verde, apontada por uma seta verde) e destino (marcação com um ‘X’ azul), bem como os obstáculos intermediários dispostos de forma aleatória. Nas Figuras 20 a 29, são mostrados todos esses cenários.

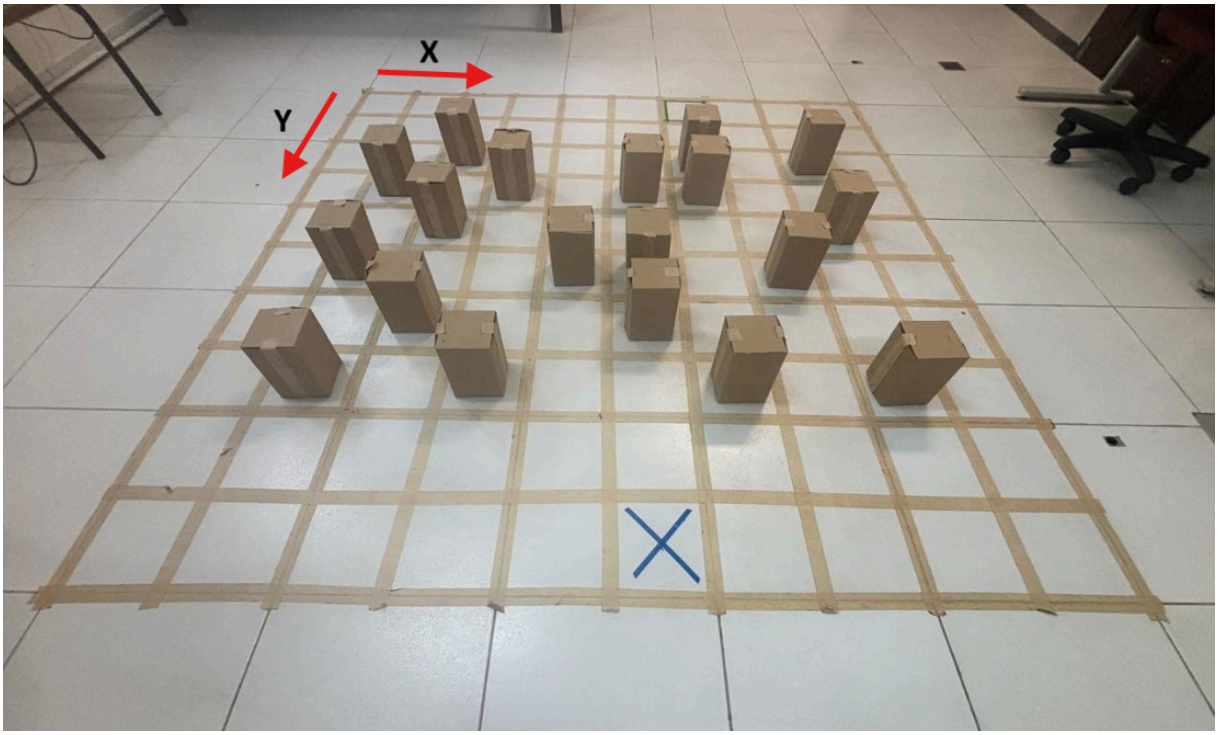
Figura 20 - Cenário 1



* O ponto de partida do robô está na coordenada **(7, 1)**. O ponto final está na coordenada **(6, 10)**

Fonte: Autor

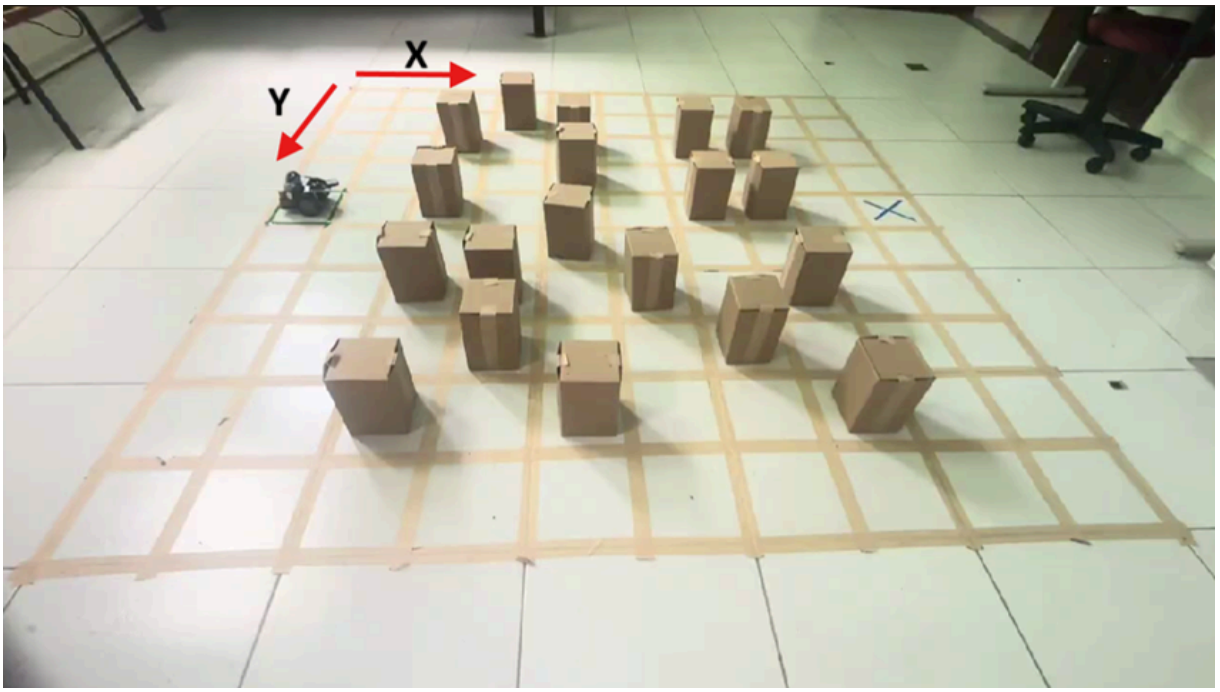
Figura 21 - Cenário 2



* O ponto de partida do robô está na coordenada **(7, 1)**. O ponto final está na coordenada **(6, 10)**

Fonte: Autor

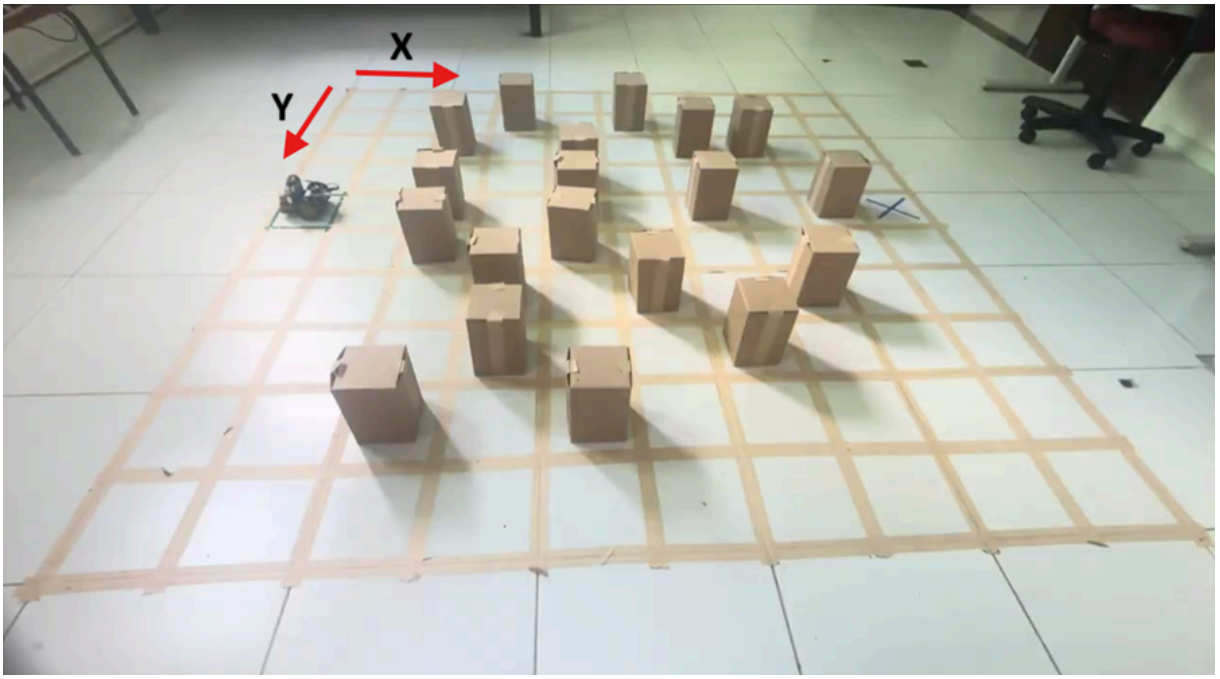
Figura 22 - Cenário 3



* O ponto de partida do robô está na coordenada **(1, 5)**. O ponto final está na coordenada **(10, 5)**

Fonte: Autor

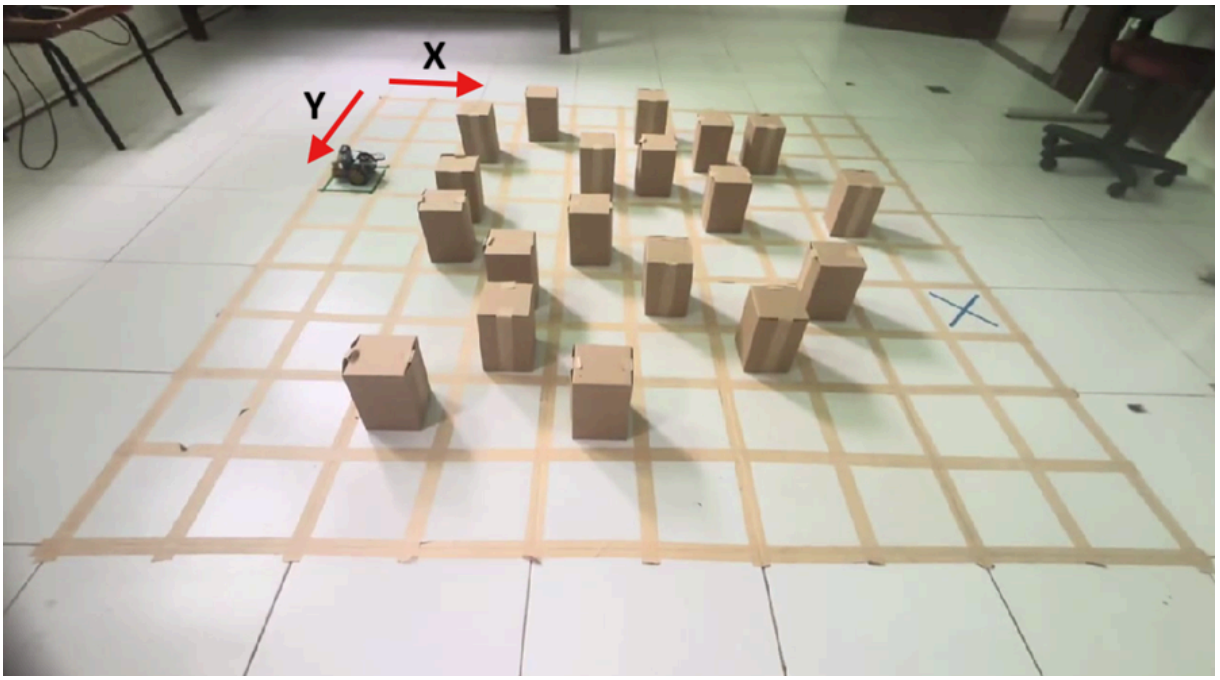
Figura 23 - Cenário 4



* O ponto de partida do robô está na coordenada **(1, 5)**. O ponto final está na coordenada **(10, 5)**

Fonte: Autor

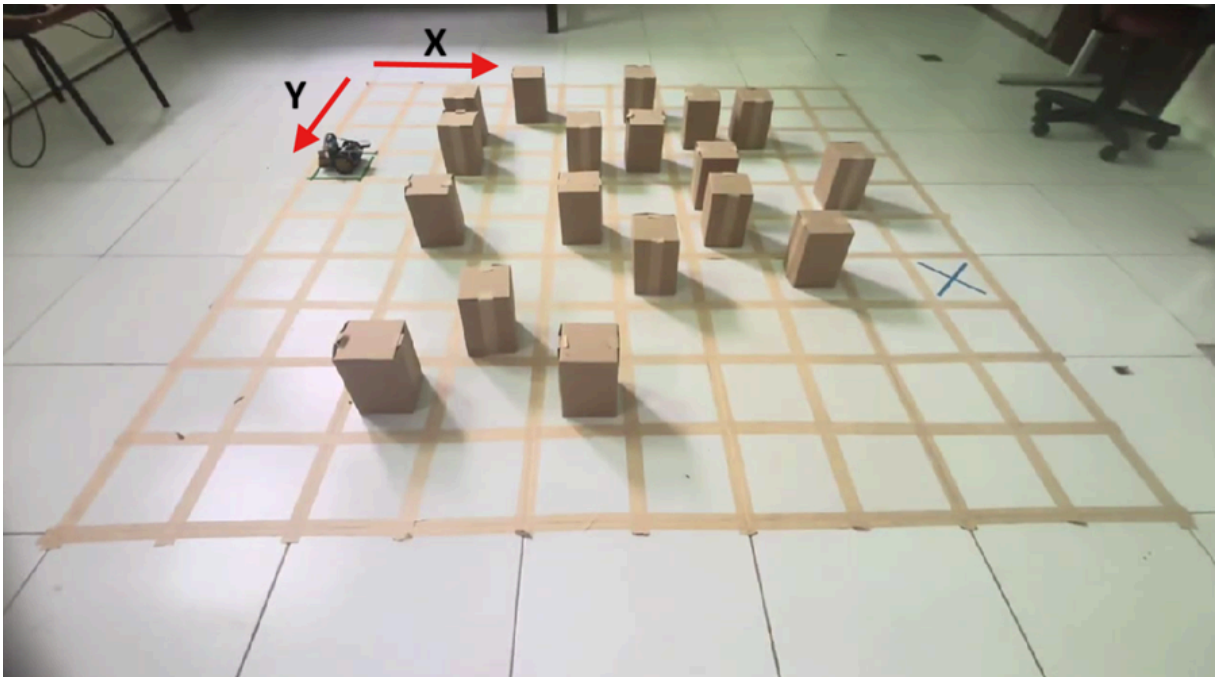
Figura 24 - Cenário 5



* O ponto de partida do robô está na coordenada **(1, 4)**. O ponto final está na coordenada **(10, 7)**

Fonte: Autor

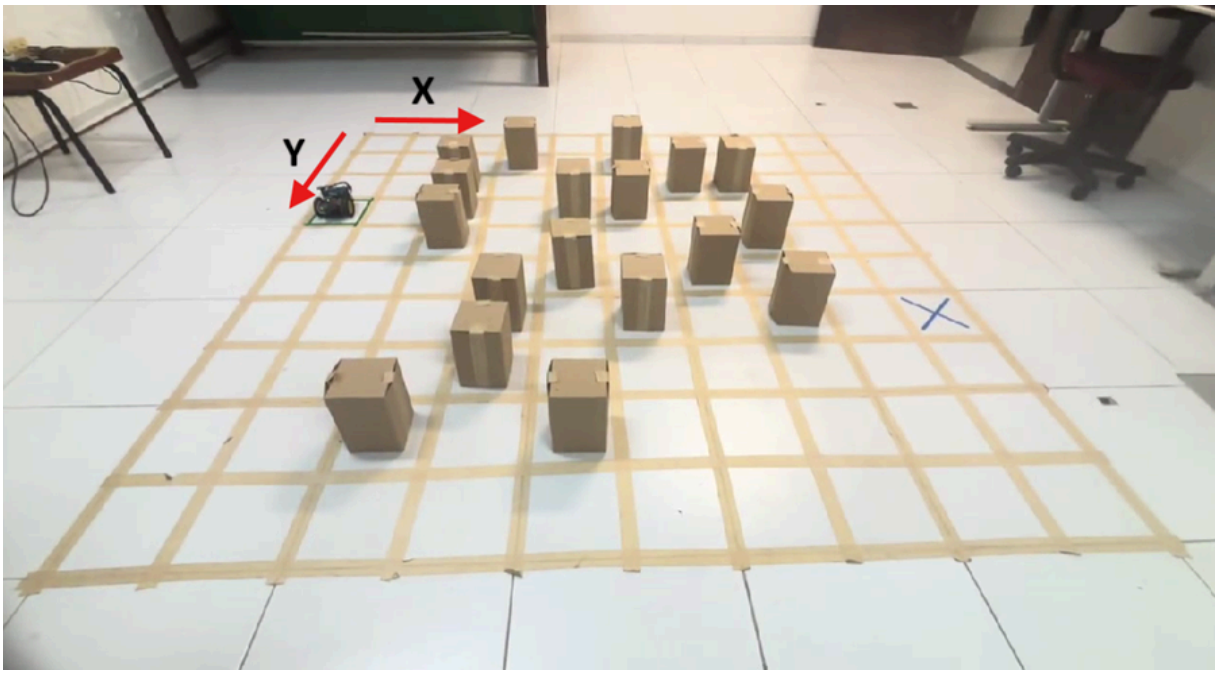
Figura 25 - Cenário 6



* O ponto de partida do robô está na coordenada **(1, 4)**. O ponto final está na coordenada **(10, 7)**

Fonte: Autor

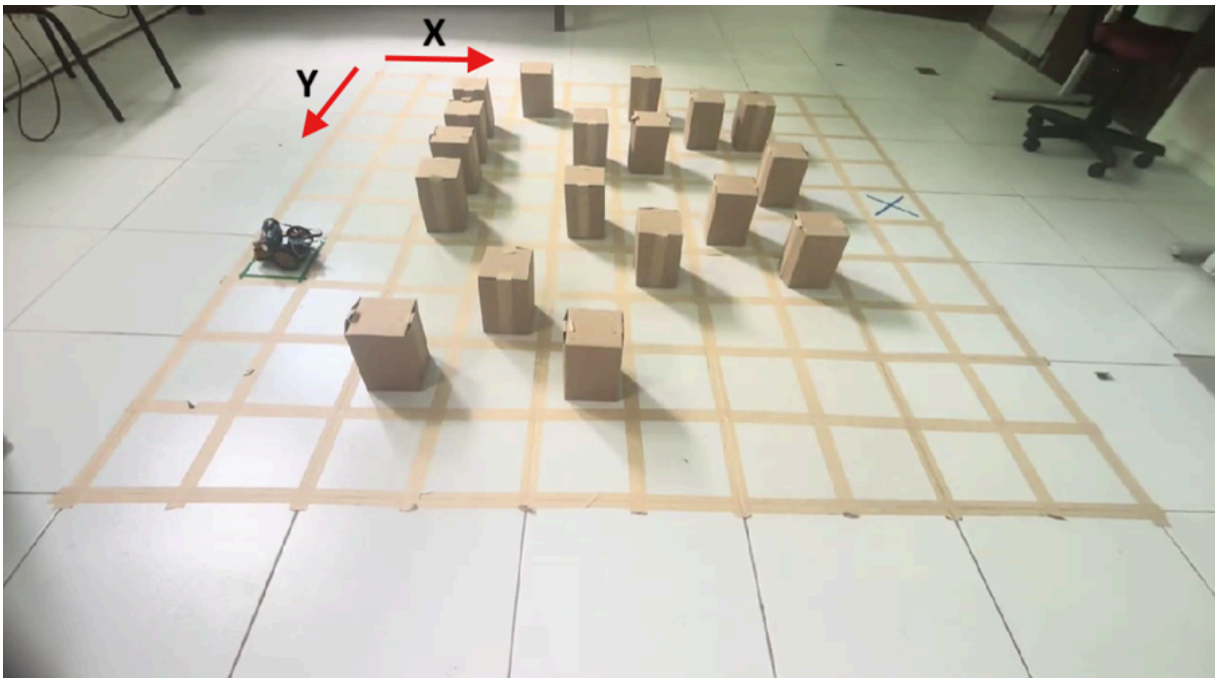
Figura 26 - Cenário 7



* O ponto de partida do robô está na coordenada **(1, 4)**. O ponto final está na coordenada **(10, 7)**

Fonte: Autor

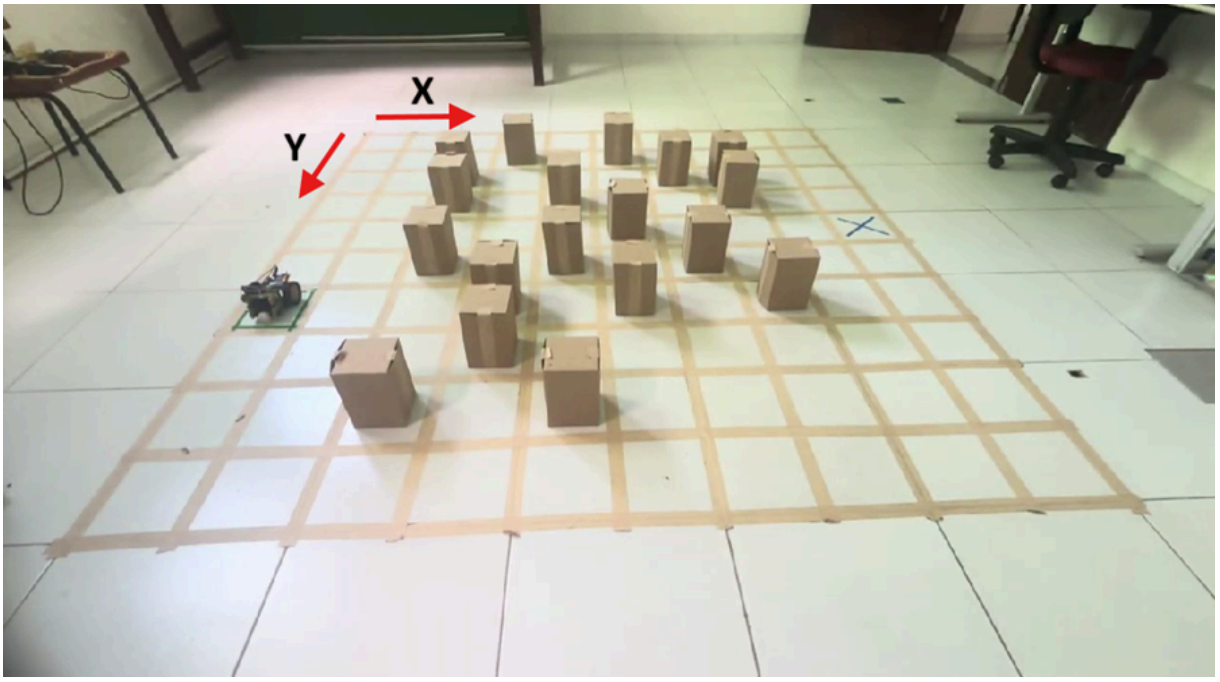
Figura 27 - Cenário 8



* O ponto de partida do robô está na coordenada **(1, 7)**. O ponto final está na coordenada **(10, 5)**

Fonte: Autor

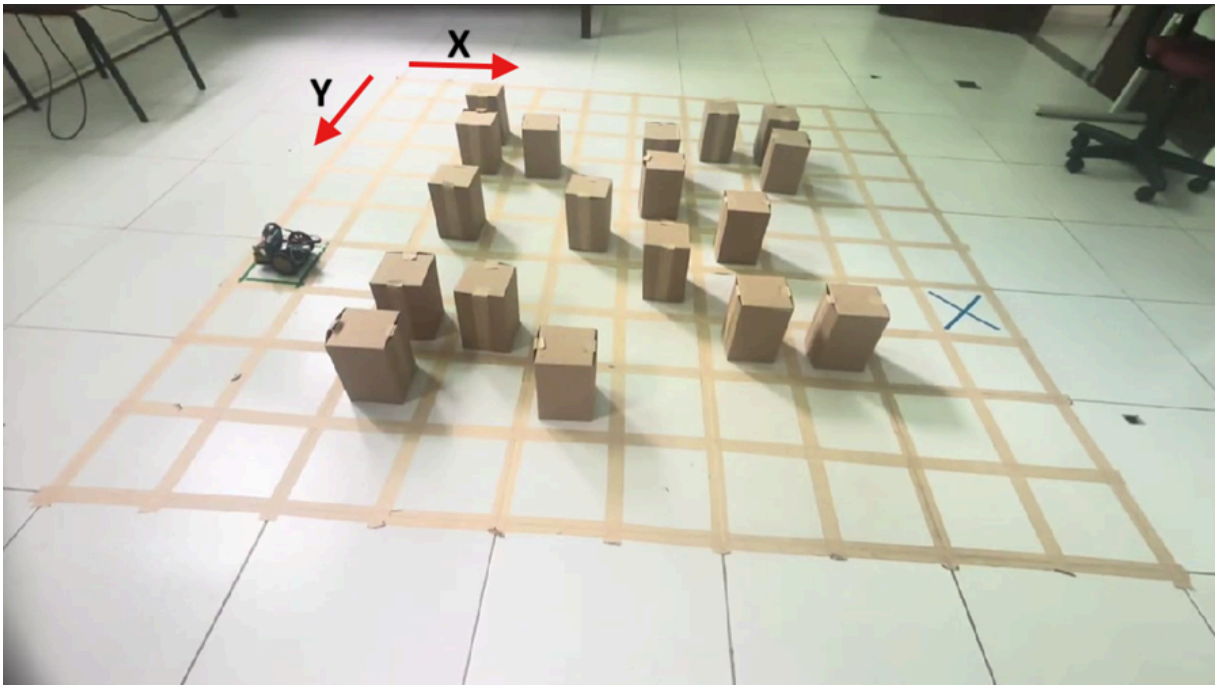
Figura 28 - Cenário 9



* O ponto de partida do robô está na coordenada **(1, 7)**. O ponto final está na coordenada **(10, 5)**

Fonte: Autor

Figura 29 - Cenário 10



* O ponto de partida do robô está na coordenada **(1, 7)**. O ponto final está na coordenada **(10, 5)**

Fonte: Autor

Durante os testes, observou-se que o algoritmo A* encontrou o caminho ótimo em todos os cenários propostos. Contudo, comparando a trajetória executada com a planejada, foram identificados pequenos desvios que se acumularam ao longo do percurso. Na maioria dos testes, o robô parou exatamente no destino. Entretanto, na minoria das execuções, o robô colidiu com algum obstáculo ou parou na célula final com um desvio. Isso ocorreu principalmente devido aos seguintes fatores:

- Diferentes atritos: observou-se diferentes atritos ao longo da superfície. Esses diferentes atritos impactaram na movimentação do robô, pois os motores são ativados por tempo de ativação;
- Tempo de ativação sensível a variações de tensão: os motores recebem uma tensão de 5 V. Porém, podem ocorrer variações para mais ou menos, ocasionando diferentes velocidades nos motores.

Na página seguinte, são listados, para cada um dos 10 cenários, os tempos das 15 execuções. Com base nisso, são calculados o tempo de execução médio e o desvio padrão.

Tabela 1 - Tempos de execução dos cenários 1 a 5

Execução	Cenário 1 (s)	Cenário 2 (s)	Cenário 3 (s)	Cenário 4 (s)	Cenário 5 (s)
1	51	61	62	60	50
2	52	60	63	61	52
3	52	61	64	59	52
4	52	60	62	57	52
5	52	60	63	58	52
6	51	61	64	60	51
7	52	60	62	58	52
8	51	61	63	60	52
9	52	60	63	59	52
10	52	60	63	58	53
11	52	61	63	58	53
12	52	60	62	58	52
13	52	61	63	58	53
14	52	60	63	59	52
15	52	60	63	59	52
Média	**51,87**	**60,27**	**62,13**	**58,33**	**51,93**
Desvio Padrão	**0,35**	**0,458**	**0,915**	**0,724**	**0,593**

Fonte: Autor

Tabela 2 - Tempos de execução dos cenários 6 a 10

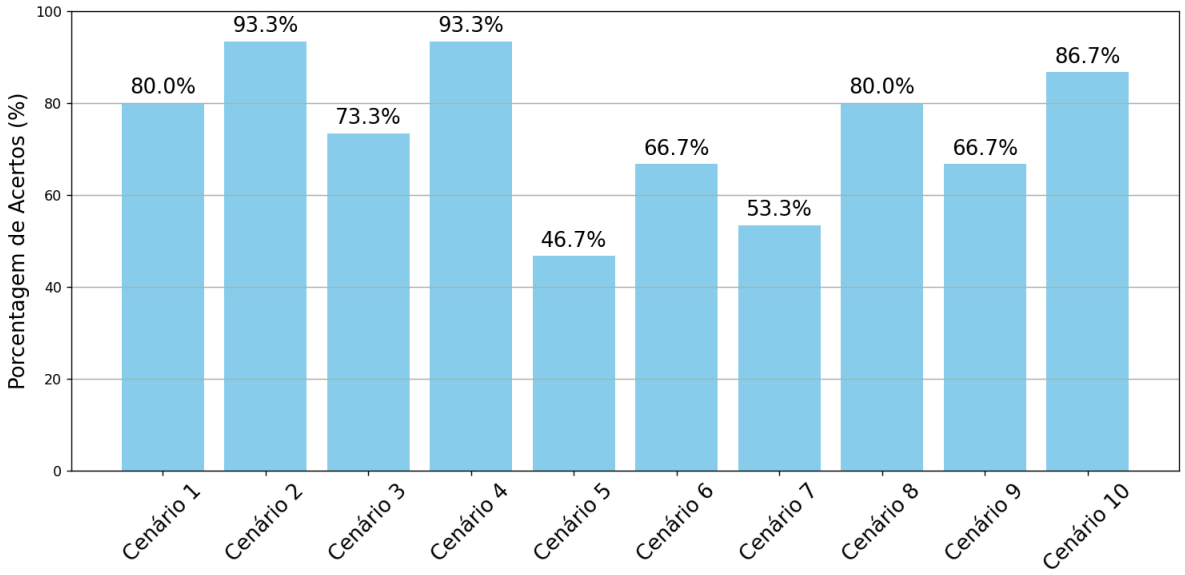
Execução	Cenário 6 (s)	Cenário 7 (s)	Cenário 8 (s)	Cenário 9 (s)	Cenário 10 (s)
1	56	65	46	63	55
2	57	66	47	63	54
3	57	65	46	63	54
4	57	65	46	62	54
5	56	66	47	63	55
6	56	65	46	63	54
7	56	66	47	63	55
8	56	66	46	62	54
9	57	66	47	63	55
10	56	65	46	63	55
11	56	65	46	62	54
12	57	66	47	63	55
13	56	66	46	62	54
14	56	68	47	64	55
15	57	66	47	63	54
Média	**56,53**	**65,47**	**46,33**	**62,67**	**54,53**
Desvio Padrão	**0,516**	**0,834**	**0,488**	**0,617**	**0,743**

Fonte: Autor

A análise dos tempos de execução junto ao desvio padrão demonstra que o sistema apresenta uma baixa variação entre os tempos de execução para o mesmo cenário. Essas variações ocorreram muito mais por causa de erros na cronometragem do que variações do *hardware*. Esses erros na cronometragem ocorreram ao considerar os tempos dos vídeos de cada execução. Essa baixa variação evidencia que a utilização do controle por tempo de ativação torna o sistema previsível, interessante para aplicações de planejamento de trajetórias. Essa estabilidade também pode ser atribuída ao Arduino. Esse componente calculou corretamente o tempo de execução para cada movimento, bem como as pausas entre os deslocamentos.

O critério para avaliar se uma execução foi bem-sucedida consistiu em verificar se o robô conseguiu atingir o destino sem colidir com os obstáculos e se mais da metade do corpo do robô chegou na célula-alvo. A seguir, é apresentado o gráfico de barras de acertos para cada cenário:

Gráfico 1 - Porcentagem de acertos por cenário



Fonte: Autor

Vale destacar que não houve nenhuma colisão frontal ou outra colisão que impedisse o robô de andar. As colisões que ocorreram foram laterais e de raspão, não impedindo a continuação do movimento pelo robô.

A partir do gráfico acima, conclui-se que os cenários, cujos trajetos eram mais curtos e com menos curvas, apresentaram um maior índice de acerto. Nos cenários

2 e 4, por apresentarem trajetos mais diretos e com menos curvas, favoreceram a precisão da trajetória. Entretanto, nos cenários 5 e 7, que eram trajetos mais complexos, observou-se um baixo índice de acerto, porém, como já mencionado, as colisões foram leves.

O gráfico evidenciou a sensibilidade do robô com relação à complexidade do trajeto. A calibração dos motores, verificação constante da voltagem da bateria e o uso de estratégias de correção de erros em tempo real podem aumentar o índice de acertos, principalmente em trajetos mais complexos.

Todos os experimentos realizados foram filmados e alguns podem ser vistos no canal do Youtube no endereço:https://www.youtube.com/@EduardoCarvalhoUfma

5 CONCLUSÃO

Foi possível observar que o algoritmo A* foi eficiente em gerar trajetórias ótimas em todos os cenários testados. A utilização da distância Manhattan para fazer as estimativas até o destino foi útil para encontrar rapidamente os caminhos ótimos. O uso da distância Manhattan foi devido a uma decisão de projeto, mas que a distância euclidiana também poderia ter sido usada.

Na parte prática, o Algoritmo A* calculou corretamente as trajetórias planejadas em todos os testes realizados. Porém, a movimentação do robô apresentou pequenos desvios, principalmente em trajetórias mais longas. Esses desvios aconteceram devido aos erros acumulados ao longo do percurso, por conta da direção do robô ser do tipo diferencial.

A ausência de sensores impossibilitou ajustes dinâmicos durante o percurso, dificultando uma correção mais precisa. Apesar dessas imprecisões, o sistema demonstrou ser eficiente para ambientes estáticos. Assim, conclui-se que é possível navegar de forma satisfatória sem percepção sensorial usando um robô de baixo custo. Mas para isso, deve ser feita a correção do giro dos motores, correta verificação da voltagem da bateria e o terreno deve ser regular e limpo. Sendo assim, este trabalho demonstrou que conceitos avançados, como planejamento de trajetória, podem ser implementados de maneira simples em um robô de baixo custo.

Portanto, a implementação real do algoritmo A* em um robô físico permitiu identificar as diferenças entre o comportamento simulado e a execução no mundo real. O planejamento permitiu que o robô seguisse as rotas calculadas, mesmo que com um pequeno erro. Contudo, esses erros não impediram que o robô alcançasse seu objetivo, mesmo que em alguns casos, o ponto final fosse atingido com pequeno desvio em relação ao centro da célula de destino. Portanto, embora a precisão absoluta não foi alcançada em todos os casos, a precisão obtida foi satisfatória desde que pequenas margens de erro sejam aceitáveis.

Os resultados deste trabalho mostram que um sistema de baixo de custo pode ser usado como um sistema auxiliar de navegação autônoma em ambientes predominantemente estáticos, como exemplo, residências com um só morador cadeirante e sem animais e ambientes de produção agrícola.

5.1 Trabalhos futuros

Durante o desenvolvimento e a realização dos testes, foram identificados alguns tópicos que podem melhorar a qualidade deste trabalho. Essas melhorias têm o objetivo de não apenas corrigir as limitações deste estudo, mas também aumentar a autonomia do robô e deixá-lo mais próximo de cenários reais. Algumas sugestões para melhorias futuras são:

- Integrar o sensor ultrassônico HC-SR04: fazer um controle simples em que o robô pare se algum obstáculo aparecer à frente, evitando a colisão;
- Navegação em ambientes que sofrem pequenas mudanças: expandir o sistema para permitir que o robô tenha um pouco mais de autonomia. Isso pode ser alcançado usando um sensor de baixo custo, como o ultrassônico HC-SR04. Assim, faz-se necessário criar uma rotina que recalcula o algoritmo A* considerando o novo obstáculo identificado;
- Aplicação de controle proporcional integral derivativo (PID): integrar o PID para ajustar dinamicamente o movimento das rodas, melhorando os deslocamentos em linha reta e nas rotações. Para isso, podem ser adicionados outros sensores, permitindo medir o deslocamento real do robô durante a navegação e corrigir dinamicamente os erros acumulados.

REFERÊNCIAS

ADDISON, A. **Calculating wheel odometry for a differential drive robot**. Disponível em: https://automaticaddison.com/calculating-wheel-odometry-for-a-differential-drive-robot/. Acesso em: 17 mai. 2025.

APPLIED AI TEAM. **Breadth-First Search (BFS) in AI**. Disponível em: https://www.appliedaicourse.com/blog/bfs-in-ai/. Acesso em: 13 ago. 2025.

ARDUINO. **Arduino Official Website**. Disponível em: https://docs.arduino.cc/learn/starting-guide/getting-started-arduino/. Acesso em: 15 mar. 2025.

BRASIL, P. M. de A. **Comparação entre planejadores de caminhos globais de um robô móvel**. 2020. Trabalho de Conclusão de Curso (Engenharia de Controle e Automação) – Universidade Federal de Santa Maria, Santa Maria, 2020. Disponível em: https://repositorio.ufsm.br/bitstream/handle/1/26567/Assis_Brasil_Pedro_Medeiros_de_2020_TCC.pdf?sequence=1. Acesso em: 28 abr. 2025.

AUTOPILOT REVIEW. **Lidar vs. cameras for self-driving cars**. Disponível em: https://www.autopilotreview.com/lidar-vs-cameras-self-driving-cars/. Acesso em: 8 mar. 2025.

EVOLVERS. **Desvendando o algoritmo A***. Disponível em: https://evolvers.com.br/desvendando-algoritmo-a/#:~:text=Heur%C3%ADstica%20no%20Algoritmo%20A*,busca%20de%20maneira%20mais%20eficiente. Acesso em: 15 jun. 2025.

FERREIRA, W. S. **Controle de Trajetória de Robô Móvel com Desvio de Obstáculos**. 2023. Trabalho de Conclusão de Curso (Engenharia Elétrica) – Universidade Federal do Ceará, Fortaleza, 2023. Disponível em:

https://repositorio.ufc.br/bitstream/riufc/73867/1/2023_tcc_wsferreira.pdf. Acesso em: 28 abr. 2025.

GIZMODO UOL. **Quais são as três "Leis da Robótica", criadas por Isaac Asimov em Eu, Robô?** Disponível em: gizmodo.uol.com.br/quais-tres-leis-robotica-criadas-por-isaac-asimov-em-eu-robo/. Acesso em: 21 jul. 2025.

INPE. **História**. Brasília, DF: Governo Federal, 2021. Disponível em: https://www.gov.br/inpe/pt-br/acesso-a-informacao/institucional/historia. Acesso em: 13 ago. 2025.

NASA. **25 Years Ago: Mars Pathfinder Launches to Mars to Deploy Sojourner, the First Planetary Rover**. 2021. Disponível em: https://www.nasa.gov/history/25-years-ago-mars-pathfinder-launches-to-mars-to-deploy-sojourner-the-first-planetary-rover/. Acesso em: 13 ago. 2025.

MICHAELIS. **Heurística**. Disponível em: https://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=heur%C3%ADstica. Acesso em: 13 ago. 2025.

ROS. **Mobile Robot Kinematics**: ros2_control documentation. Disponível em: https://control.ros.org/rolling/doc/ros2_controllers/doc/mobile_robot_kinematics.html. Acesso em: 13 ago. 2025.

RUCKSIKAAR. **Interfacing the HC-06 Bluetooth module with Arduino. Arduino Project Hub**. Disponível em: https://projecthub.arduino.cc/RucksikaaR/interfacing-the-hc-06-bluetooth-module-with-arduino-94aabd. Acesso em: 12 ago. 2025.

RUSSO, M.; CECCARELLI, M. A Survey on Mechanical Solutions for Hybrid Mobile Robots. **Robotics**, [*s. l.*], v. 9, n. 2, p. 32, 2020. DOI 10.3390/robotics9020032. Disponível em: https://www.mdpi.com/2218-6581/9/2/32. Acesso em: 15 ago. 2025.

SANTOS, L. C. F. **Planeamento de Trajetórias em Ambientes Agrícolas**. 2017. Dissertação (Engenharia Eletrotécnica e de Computadores) - Universade do Porto, Porto, 2017. Disponível em: https://repositorio-aberto.up.pt/bitstream/10216/105420/2/200809.pdf

SINGAL, P.; R.S.CHHILLAR, R. S. C. Dijkstra Shortest Path Algorithm using Global Position System. **International Journal of Computer Applications**, [*s. l.*], v. 101, n. 6, p. 12–18, 18 set. 2014. DOI 10.5120/17690-8654. Disponível em: http://research.ijcaonline.org/volume101/number6/pxc3898654.pdf. Acesso em: 10 ago. 2025.

SPRINGRC. **SM-S4306R Servo Motor**. Disponível em: http://www.springrc.com/en/pd.jsp?id=91#_jcp=3_50. Acesso em: 12 ago. 2025.

ZANCHIN, B. C. **Análise do algoritmo A* (a estrela) no planejamento de rotas de veículos autônomos**. 2018. Trabalho de Conclusão de Curso (Engenharia Eletrônica) – Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2018. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/16226/1/PG_COELE_2018_1_03.pdf Acesso em: 15 jul. 2025.