UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA ELETRÔNICA

WILLIAM SERRA NOVA FERNANDES

PLANEJAMENTO DE TRAJETÓRIA EM ROBÔ MÓVEL COM INTELIGÊNCIA ARTIFICIAL EMBARCADA

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2019

WILLIAM SERRA NOVA FERNANDES

PLANEJAMENTO DE TRAJETÓRIA EM ROBÔ MÓVEL COM INTELIGÊCIA ARTIFICIAL EMBARCADA

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso 2, do curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. Marcos Roberto Bombacini

Coorientador: Prof. Dr. José Luis Sousa Magalhães Lima

TOLEDO
2019

TERMO DE APROVAÇÃO

Título do Trabalho de Conclusão de Curso Nº 81

Planejamento de Trajetória em Robô Móvel com Inteligência Artificial Embarcada

por

William Serra Nova Fernandes

Esse Trabalho de Conclusão de Curso foi apresentado às 10h do dia 28 como requisito parcial para a obtenção do título Bacharel em Engenharia Eletrônica. Após deliberação da Banca Examinadora, composta pelos professores abaixo assinados, o trabalho foi considerado APROVADO.

Alberto Vinicius De Oliveira
UTFPR

Alberto Yoshihiro Nakano
UTFPR

Marcos Roberto Bombacini
COELE
Orientador (a)

Fabio Rizental Coutinho
Coordenador(a) da COELE

"O termo de aprovação assinado encontra-se na coordenação do curso"

Dedico este trabalho à minha família e aos meus amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter iluminado meu caminho me dando forças e saúde para concluir mais uma etapa de minha vida.

Aos meus pais Getulio e Sueli, pela paciência, carinho, compreensão e pelo apoio incondicional, incentivando para continuar sempre, mesmo com todas as adversidades e momentos difíceis durante estes anos passados na universidade.

Ao meu orientador Prof. Dr. Marcos Roberto Bombacini, pelo profissionalismo, comprometimento e dedicação para a realização deste trabalho.

À UTFPR, servidores e professores, especialmente ao Prof. Felipe Pfrimer, por me auxiliarem no aprendizado e na obtenção do conhecimento através da infraestrutura do *campus*.

Aos meus amigos que fizeram parte de minha graduação, principalmente Daniella Medeiros e Gabriel Cacilho com quem partilhei inúmeras madrugadas de estudo, preocupações, ansiedade, alegrias, conhecimento e amizade.

Por fim, agradeço a todas as pessoas de que fizeram parte desta etapa decisiva da minha vida.

RESUMO

Planejar rotas é fundamental à robótica móvel, posto que o presente trabalho visa a simulação e a construção de um robô móvel sendo empregando um acionamento diferencial. Existem diversas teorias que estabelecem procedimentos de planejamento de trajetória, portanto nesta atividade foi escolhido duas técnicas a serem desenvolvidas e comparadas. As estratégias de planejamento de trajetória, BUG 0 e BUG 1, fazem o robô tomar decisões do melhor caminho a percorrer. Com isso, foram empregadas simulações, para facilitar o desenvolvimento do robô móvel, através do software SimTwo, com as metodologias BUG 0 e BUG 1. Os materiais empregados para desenvolvimento do hardware do robô foram confeccionadas através do Arduino UNO, módulo ponte H, sensores de linha, e por fim uma estrutura acrílica para acoplamento dos componentes e rodas. Deste modo, foi obtido o resultado previsto pela teoria, onde o BUG 1 apresentou melhores estratégias de rotas do que o BUG 0.

Palavras-chave: Robô móvel. BUG 0. BUG 1. SimTwo. Arduino.

ABSTRACT

Planning routing is fundamental to mobile robotics, since the present work aims at the simulation and construction of a mobile robot employing a differential drive. There are several theories that establish trajectory planning procedures, so in this activity two techniques were chosen to be developed and compared. The trajectory planning strategies, BUG0 and BUG1, make the robot make decisions of the best way to go. With this, simulations were used to facilitate the development of the mobile robot through SimTwo software, using the methodologies BUG 0 and BUG 1. The technologies used to develop the robot hardware were made using the Arduino UNO, bridge module H, line sensors, and finally an acrylic structure for coupling of components and wheels. In this way, the result predicted by the theory was obtained, where BUG1 presented better route strategies than BUG0.

Keywords: Mobile robot. BUG 0. BUG 1. SimTwo. Arduino.

LISTA DE SIGLAS

AUVS	*Autonomous Underwater Vehicle* (Veículos subaquáticos autônomos)
IDE	*Integrated Development Environment* (Ambiente de desenvolvimento Integrado)
PWM	*Pulse Width Modulation* (Modulação por largura de pulso)
RMAS	Robôs Móveis Autônomos
VANTS	Veículos Aéreos Não Tripulados
XML	*Extensible Markup Language* (Linguagem de marcação extensivel)

LISTA DE SÍMBOLOS

$q(t)$	Caminho a ser percorrido
q_s	Ponto de partida
q_g	Ponto de chegada
C_{obs}	Caminho com obstáculo
C_{livre}	Caminho livre
$F(q)$	Função da trajetória
Q	Carga pontual

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

É fato que os robôs faziam parte apenas de filmes e livros de ficção científica, frutos da imaginação do homem. Hoje, os robôs e sistemas robotizados atuam em tarefas simples do dia a dia, como o portão da garagem que pode ser acionado ainda dentro do veículo, pesquisas submarinas, interplanetárias ou até mesmo na medicina, na qual muitas vidas são salvas devido ao avanço tecnológico da robótica. Dessa forma, de simples mecanismos a sofisticados robôs, tem-se uma abrangência muito grande da tecnologia. Em outras palavras, os robôs de hoje executam cada vez mais tarefas que o homem não pode ou não quer realizar.

Atualmente, a robótica é uma vasta área de pesquisa. Ela possui característica interdisciplinar abrangendo várias engenharias, tais como, mecânica, eletrônica e computação. Existe conceitos como a microeletrônica, inteligência artificial, entre outros necessários para o funcionamento de um robô.

A capacidade de um robô de agir e planejar seu próprio movimento é fundamental para sua autonomia. Por conta disso, o planejamento de seu movimento se tornou parte essencial na moderna robótica inteligente.

Uma necessidade fundamental na robótica é ter algoritmos que convertam especificações de tarefas do ser humano para movimentos robóticos. O termo planejamento de trajetória é frequentemente usado para problemas deste tipo. Uma versão clássica para o problema de planejamento de movimento é o problema da mudança do piano, que é ilustrado na Figura 1.

Considerando o movimento como uma função contínua no tempo no interior de um espaço delimitado, desenha-se uma curva ilustrando o caminho. O problema da “mudança do piano” corresponde ao problema de planejamento de trajetória, isto é, a instância geométrica do planejamento do movimento do robô conforme pode ser visto na Figura 1.

Figura 1 – Ilustração moderna do problema da "mudança do piano" onde dois personagens precisam mover um piano enquanto desviam de obstáculos.

Fonte: Laumond (2014)

A restrição para se evitar obstáculos é levada em consideração. Neste contexto, a tarefa de planejar a trajetória significa reduzir o comprimento do caminho sem considerar o tempo. O objetivo é encontrar o menor caminho entre dois pontos.

1.1 Breve Histórico

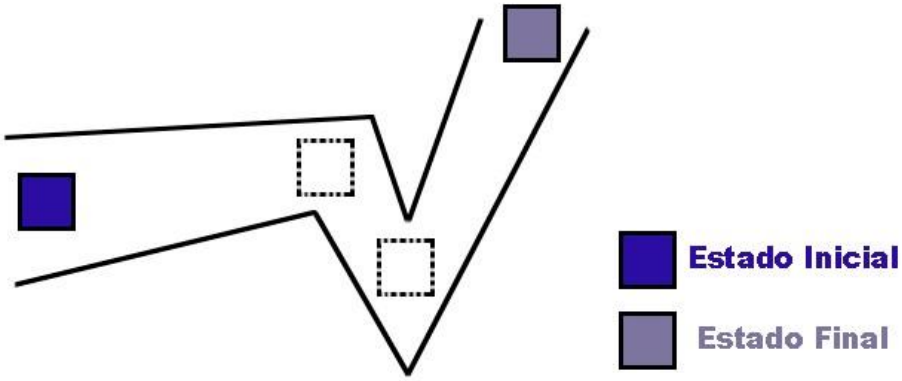
Existem publicações disponíveis na área do planejamento de trajetórias para robôs. A disciplina, "rotas para robôs", foi lançada em meados dos anos 60, mas não chamou a atenção da maioria da comunidade acadêmica até o artigo de Lozano e Wesley em 1979 sobre planejamento espacial. Os métodos desenvolvidos atualmente são variações de abordagens mais gerais, tais como, *Bug Algorithms*, *Roadmap*, *Cell Decomposition*, *Potential Fields*, *Sampling-based motion planning*, *Kalman filtering*, *Heuristic Approaches* e Programação Matemática. Esses métodos não são necessariamente mutuamente exclusivos e uma combinação é geralmente

usada para desenvolver um planejador de trajetórias (Masehin & Amin Naseri, 2004; Dongbin & Jianqiang, 2006).

1.2 Definição do problema

O problema da “mudança do piano” envolve o planejamento e o controle de movimento, de modo que um objeto se mova em um estreito caminho de uma determinada posição inicial para uma posição final desejada. O objeto deve evitar obstáculos e paredes para não colidir. Então, é necessário um controle de atitude altamente preciso. Schawartz e Sharir (1983) introduziram um algoritmo para resolver um problema bidimensional para um determinado corpo e em paredes delimitadas por uma coleção de regiões. O problema do piano pode ser visto na Figura 2.

Figura 2 - Ilustração de exemplo do problema do piano



Fonte: Adaptado de Yuko Ishiwaka (2004)

Uma desvantagem de utilizar este algoritmo é que tanto a forma do objeto, como a do ambiente, deve ser conhecida. No mundo real, o ambiente muda constantemente e a forma e o tamanho do robô não pode ser conhecido exatamente. Esse método matemático é difícil de adaptar para o mundo real.

1.3 Objetivos

O principal objetivo desse trabalho é desenvolver algoritmos, utilizando técnicas de planejamento de trajetória, para simulação e para um robô móvel com acionamento diferencial. Consequentemente, há outros objetivos que compõem este trabalho.

Objetivos específicos:

- Desenvolver algoritmos para planejar rotas.
- Implementar os algoritmos no simulador e testá-los no ambiente simulado.
- Implementar os algoritmos nos controladores para robô executar a trajetória.
- Construir e testar um robô com tração diferencial.

1.4 Justificativa

A navegação, fundamental à robótica móvel, é uma das tarefas mais complexas na problemática da locomoção de robôs móveis. Tal complexidade advém do fato de que a navegação deve integrar sensoriamento, atuação, planejamento, arquitetura, *hardware* e computação. Assim, a integração de todos esses pontos é inerente à obtenção de uma boa navegação robótica. Nesse sentido, o planejamento de trajetória em robótica móvel objetiva prover, aos robôs, a capacidade de planejar seus próprios movimentos, sem a necessidade de interferência humana. Além disso, a elaboração de um plano de movimentação é uma tarefa com elevado grau de dificuldade.

Existe um grande número de métodos para resolver problemas gerais de planejamento de movimentos, entretanto, nem todos resolvem todas as generalizações deste problema. Por exemplo, alguns métodos necessitam que o espaço de trabalho seja bidimensional e os obstáculos poligonais. Apesar de muitas diferenças externas, os métodos são baseados em algumas técnicas gerais: *roadmap*, *cell decomposition*, *potential fields* e *bugs algorithms* (Dongbin & Jianqiang, 2006).

Conforme Reges, Silva, Bezerra & Alexandria (2017) são necessários maiores aprimoramentos em seu trabalho, o qual utiliza a lógica *fuzzy* como meio de acionamento de um robô diferencial. Segundo eles, a técnica é inovadora, porém só funciona em linha reta. Portanto, é necessário um nível maior de complexibilidade, como curvas e obstáculos que requerem desvio.

Nesse projeto, será abordado um estudo de métodos para que um robô consiga identificar um caminho a percorrer para alcançar um objetivo. Embora exista muitos métodos para obter o planejamento de caminho livre de obstáculos, nesse projeto propõe-se utilizar a técnica *bug algorithms*, mais especificamente os

algoritmos BUG 0 e BUG 1, para a discretização do espaço e desenvolvimento de estratégia para o robô seguir.

No Capítulo 2 deste encontra-se o referencial teórico.

No Capítulo 3, são descritos os materiais e os métodos que serão utilizados para o desenvolvimento do projeto.

O Capítulo 4 temos os resultados alcançados após a realização de simulações e montagem do robô.

No Capítulo 5 encontra-se as conclusões desse projeto e possíveis melhorias para trabalhos futuros envolvendo o mesmo tema.

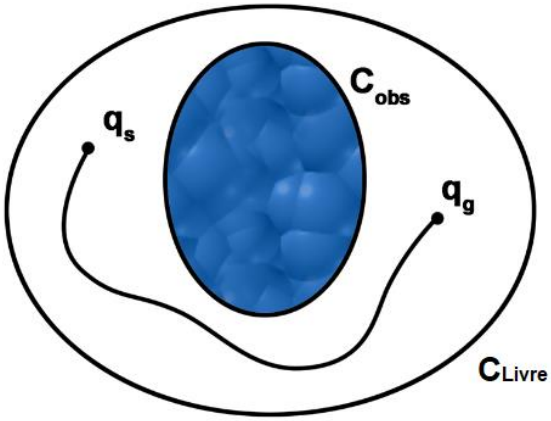
2 REFERENCIAL TEÓRICO

Nessa seção será apresentada as técnicas existentes para o planejamento da trajetória em robô móvel.

2.1 Problema do Piano

O objetivo básico do problema do piano é definir uma trajetória entre um ponto inicial e um ponto final desviando dos obstáculos no decorrer do percurso. Esta ideia abstrata pode ser mais rigorosamente definida como: estabelecer uma trajetória na forma de uma função q(x,y,t) definido como sendo o ponto de origem q(0,0,0) = q_s e como sendo o ponto de destino q(1,1,1) = q_g. Sendo que a função q(x,y,t) $\cup$ C_{Livre}. Na Figura 3 encontra-se o domínio do problema do piano.

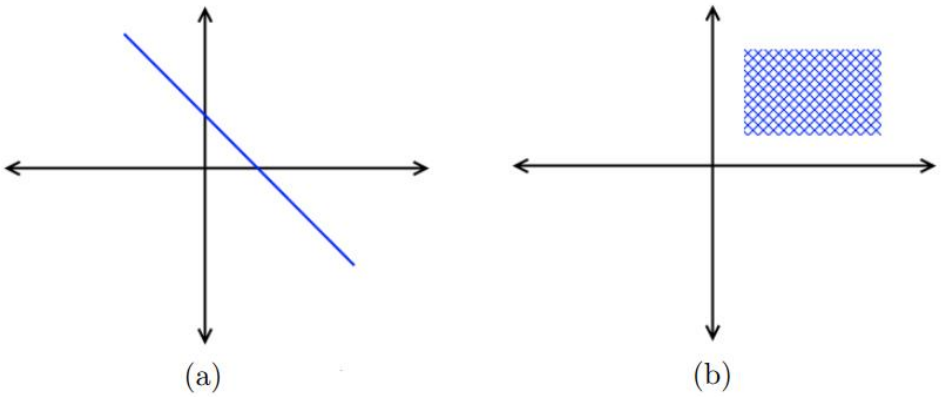
Figura 3 – Domínio do problema do piano



Fonte: Adaptado de Kenneth Carpenter e Evan Glasgow (2009)

Considerando-se que este problema possui restrições adicionais, às quais, a trajetória precisa satisfazer, tal como, uma restrição adicional para a trajetória pode ser q(x,y,t) = constante, um exemplo disso seria manter o robô em linha reta, sem qualquer mudança na direção ou sentido dele. Outro exemplo é para q(x,y,t) ≤ 0, que na prática representa a mudança de direção do robô. Esses dois exemplos destacam dois tipos diferentes de restrições: uma restrição de igualdade e uma restrição de desigualdade, respectivamente. Essas restrições podem ser vistas na Figura 4.

Figura 4 – Dois tipos de restrições: (a) restrição de igualdade; (b) restrição de desigualdade



Fonte: Adaptado de Kenneth Carpenter e Evan Glasgow (2009)

Com a necessidade de satisfazer as restrições, é fácil verificar que até mesmo no problema do piano, a trajetória a ser encontrada, em geral, possui um conjunto inerente de restrições.

$$q(x,y,t) = 1 \text{ se } q \in C_{obs}, \quad (1)$$

$$q(x,y,t) = -1 \text{ se } q \in C_{Livre}, \quad (2)$$

$$q(x,y,t) \leq 0. \quad (3)$$

Obs: C_{obs} é difícil de ser computado, devido ao não conhecimento do obstáculo em si, então embora isso seja útil para representar o domínio, é abstruso encontrar uma solução (Carpenter e Glasgow, 2009).

2.2 Robô Móvel

Os robôs móveis têm capacidade de se moverem ao redor de seus ambientes e em contraste com os robôs industriais que geralmente consistem de um braço articulado e um dispositivo de atuação em uma superfície fixa. Segundo Jung (2005), as atenções se voltaram para os robôs móveis, capazes de se deslocar no ambiente em que se encontram e, principalmente, nos RMAS (Robôs Moveis Autônomos) e Veículos autônomos Inteligentes. Robôs móveis são classificados de acordo com o ambiente em que se movem, podendo ser terrestres (normalmente apresentam rodas, mas alguns ainda possuem pernas), aéreos (VANTS) ou subaquáticos (AUVS). A robótica tem sido grande alvo de estudos e pesquisas nos

últimos anos, devido sua larga escala de utilização. Em Niku (2014), é mostrado que um robô sozinho é inútil, este deve vir acompanhado com acessórios que podem ser dispositivos periféricos, sensores, garras, câmeras, entre outros.

Para Wolf, Osório, Simões, & Trindade (2009), os RMAs possuem, como características fundamentais, as capacidades de operação de modo semi ou completamente autônomo. Também deve ser considerado que maiores níveis de autonomia serão alcançados somente à medida que o robô passe a integrar outros aspectos considerados da maior importância, como: capacidade de percepção (sensores que conseguem "ler" o ambiente onde ele atua), capacidade de agir (atuadores e motores capazes de produzir ações, tais como o deslocamento do robô ambiente), robustez e inteligência (capacidade de lidar com as mais diversas situações, de modo a resolver tarefas por mais complexas que sejam).

Pode-se citar aqui diversos exemplos de robôs móveis famosos (ver Figura 5), resultantes da pesquisa e do desenvolvimento que ocorre nessa área: o robô aspirador de piscina como o *Aquabot Xtreme* (aquaboat, 2012), Figura 5 (a), robôs domésticos para limpar a casa como *Roomba* (iRobot, 2009), Figura 5 (b), e para cortar a grama como o *AutoMower* (Huskvarna, 2009), Figura 5 (c), o robô cachorro *Aibo* (Sony, 2009), Figura 5 (d).

Figura 5 – Exemplos de robôs Móveis

Fonte: Aquaboat, iRobot, Huskvarna, Sony.

2.3 Planejamento de trajetória

Segundo Coelho e Valim (2001), muitos pesquisadores da área de robótica móvel concentram seus esforços no desenvolvimento de melhorias para a navegação e para o planejamento de caminhos para robôs.

O planejamento de um caminho é definido segundo Roland Siegwart (2004) como dado um mapa e uma meta, o planejamento de um caminho envolve a identificação de uma trajetória que fará com que um robô alcance o objetivo determinado. O planejamento de caminhos é uma solução estratégica para problemas de competência: como o robô deve decidir o que fazer a longo prazo para atingir seus objetivos. Basicamente, a solução para o problema do planejamento de trajetória é selecionar um caminho entre vários possíveis para o robô percorrer um espaço físico, partindo de uma posição inicial até a posição final, evitando colisões com obstáculos presentes no meio ambiente.

Nas subseções 2.3.1 a 2.3.4, serão apresentadas brevemente algumas das principais estratégias utilizadas para modelar o robô no ambiente e, em seguida executar o planejamento da trajetória.

2.3.1 *Roadmap*

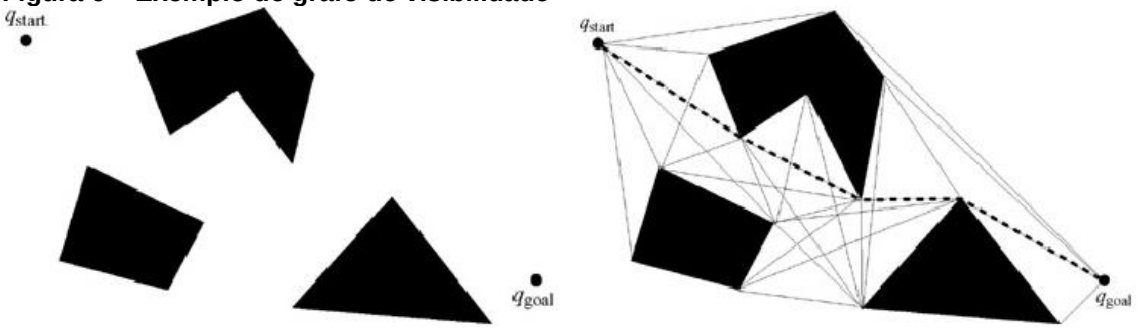
Segundo Polidoro (2010) a técnica de *roadmap* para planejamento de trajetórias consiste em capturar a conectividade do espaço livre do ambiente através de uma rede de curvas. Essa rede é vista como um conjunto padrão de caminhos. O planejamento da trajetória então se reduz a conectar os pontos inicial e final do robô no *roadmap* e buscar neste um caminho entre esses dois pontos. Se existir um caminho, este será dado pela junção de três subcaminhos: um subcaminho entre o ponto inicial até algum ponto do *roadmap*, um subcaminho do *roadmap* e um subcaminho do *roadmap* até o ponto final.

Existem vários métodos propostos que foram baseados nessa ideia, dentre eles: grafos de visibilidade, diagrama de Voronoi, rede de caminho livre e silhueta. A seguir serão apresentados, brevemente, alguns desses métodos:

Grafos de visibilidade: é um dos principais métodos de planejamento de trajetória, portanto, é popular na robótica móvel. Um grafo de visibilidade consiste na

representação de todas as conexões possíveis entre dois vértices quaisquer que se encontram no espaço livre do ambiente. Isso significa que para cada vértice, as conexões serão feitas para todos os outros vértices que podem ser vistos a partir dele. A Figura 6 mostra um exemplo de grafo de visibilidade onde o ponto inicial, *qstart*, e o ponto final, *qgoal*, são tratados como vértices. Conexões também podem ser feitas entre vértices vizinhos do mesmo polígono.

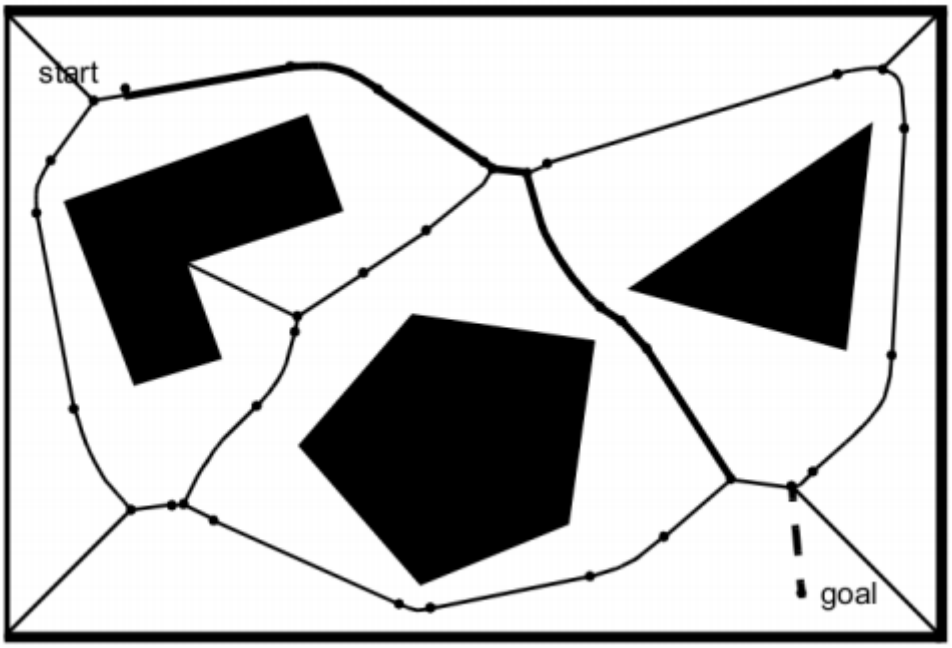
Figura 6 – Exemplo de grafo de visibilidade



Fonte: Rasoul Mojtahedzadeh (2011)

Diagrama de Voronoi: é um método completo de mapa de rotas que tende a otimizar a distância entre o robô e os obstáculos no mapa. Para cada ponto livre no mapa é calculada a distância para o obstáculo mais próximo. O diagrama de Voronoi consiste nos pontos que são equidistantes de um ou mais obstáculos. Um exemplo de diagrama de Voronoi pode ser visto na Figura 7.

Figura 7 – Exemplo de Diagrama de Voronoi

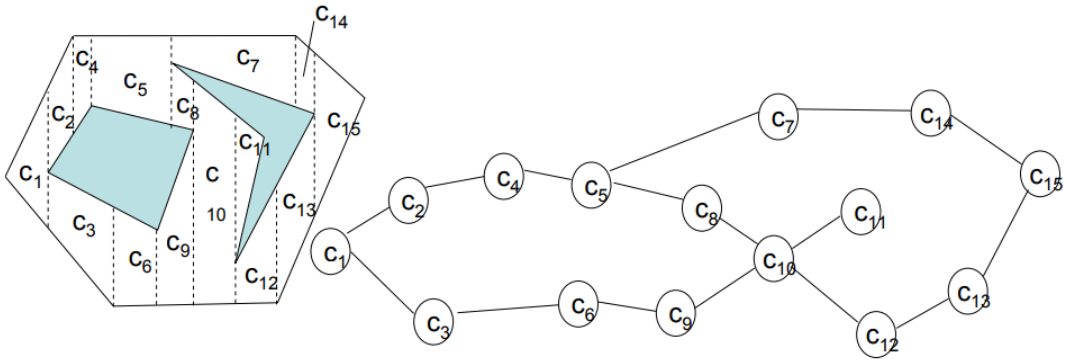


Fonte: Polidoro (2010)

2.3.2 *Cell Decomposition*

A ideia principal do "*Cell Decomposition*" é realizar uma divisão do ambiente em áreas geométricas finitas ou células. Para que essas regiões estejam conectadas em uma distribuição adjacente. Observe na Figura 8, que C_1 é o ponto inicial e C_{15} o objetivo.

Figura 8 – A decomposição celular à esquerda e o gráfico de adjacência à direita



Fonte: Fonte: Adapitado de Piardi (2018)

Deve haver uma discriminação de células entre áreas livres e áreas ocupadas por obstáculos. Assumindo que a decomposição é calculada, o

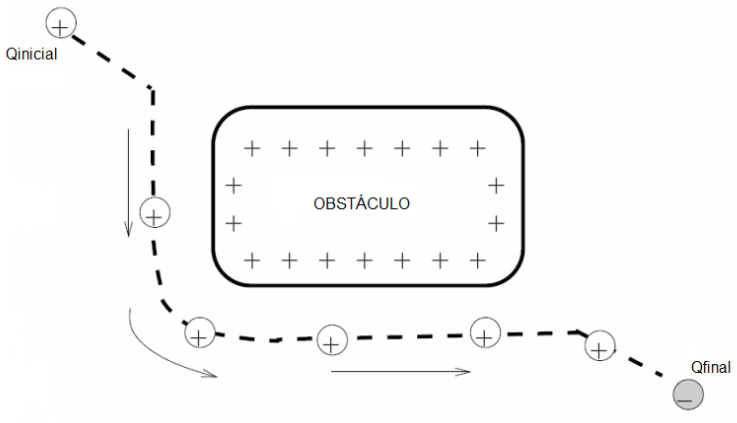
planejamento do caminho com uma decomposição celular geralmente é feito em duas etapas: primeiro, o planejador determina as células que contém o início e o objetivo, respectivamente e, em seguida, o planejador procura um caminho, gráfico de adjacência. O trabalho de Kloetzer, Mahulea e Gonzalez (2015) propõe um otimizador neste método, melhorando os pontos de passagem do robô na célula livre.

2.3.3 *Potential Fields*

As técnicas anteriores são baseadas em planejar o caminho a partir da classificação de espaço livre e espaço com objeto. A técnica "*Potential Fields*" usa uma metodologia diferente. Nessa abordagem cria-se o mapa em que o robô é inserido com base em campo elétrico. Este método trata o robô como uma carga pontual, sobre a influência de um campo potencial. As forças atuam no robô no sentido de levá-lo ao seu objetivo. A ideia por trás dessa abordagem é fazer com que o robô seja atraído para seu objetivo enquanto ele é repelido pelos obstáculos que são conhecidos.

Um exemplo de *Potential Fields* pode ser visto na Figura 9, imagine que o robô seja a carga Q, observe que ela será repelida pelo obstáculo e atraída para o seu objetivo 'Qfinal'. O caminho final é dado pela força resultante desses campos potenciais.

Figura 9 – Ilustração da ideia da técnica "*Potencial Fields*"



Fonte: Adaptado de Choset (2015)

2.3.4 *Bugs Algorithms*

Nessa estratégia de movimento lida-se com a incerteza do sensoriamento. De modo geral os algoritmos do Bug fazem as seguintes suposições:

- O robô é um ponto em um mundo 2D;
- Os obstáculos são desconhecidos;
- Uma posição inicial e uma meta são definidas;
- O robô é equipado com um sensor de curto alcance que pode detectar um limite de obstáculo a uma distância muito curto. Isso permite que o robô execute uma trajetória que siga o limite do obstáculo.

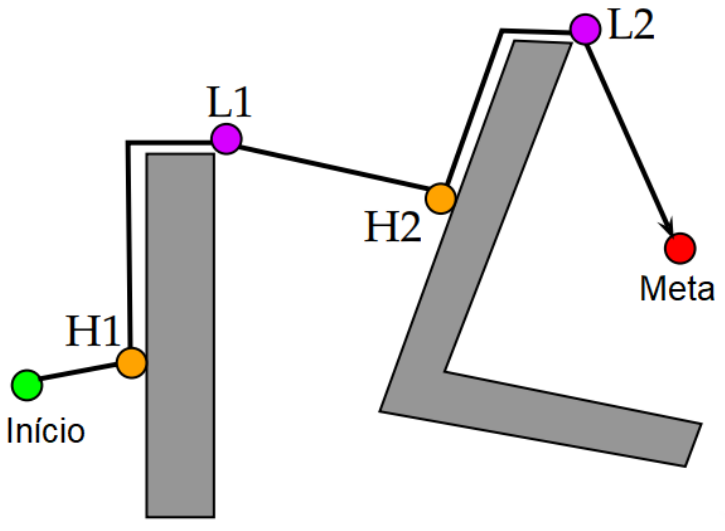
2.3.4.1 BUG 0

Segundo Erion Plaku (2009) a técnica do algoritmo BUG 0 pode ser descrita de forma resumida da seguinte maneira:

Repete até a meta ser encontrada{
Siga em direção ao objetivo;
Se (o sensor encontrar obstáculo) então{
Siga até o limite desse obstáculo }
}

Na Figura 10, temos o caminho traçado pelo robô com tal algoritmo.

Figura 10 – Exemplo de caminho segundo BUG 0.



Fonte: Adaptado Plaku (2009)

A técnica é considerada incompleta, pois o algoritmo pode não encontrar uma solução mesmo que haja alguma. Esse problema é identificado devido o método não possuir memória. Para resolver esse problema surgiu o algoritmo BUG 1.

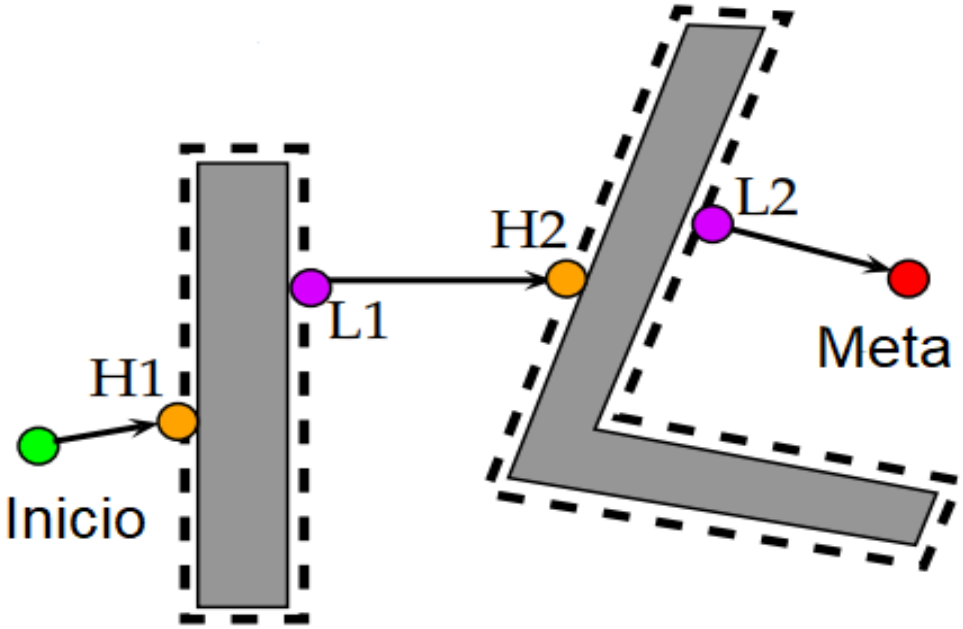
2.3.4.2 BUG 1

Propondo melhorar o BUG 0, o algoritmo BUG 1 implementa a memória em seu planejamento de trajetória.

Segundo Plaku (2009) o método funciona da seguinte maneira: o ponto se move em direção a meta até que ache um obstáculo, ao achar um obstáculo ele percorre em volta de todo o obstáculo até voltar no ponto de entrada deste. Ao chegar no ponto inicial verifica-se o melhor caminho até a meta e executa tal processo. Por fim, isso se repete até que a meta seja alcançada.

A Figura 11 ilustra o caminho feito pelo ponto até que a meta seja atingida segundo o algoritmo de Bug 1.

Figura 11 – Exemplo de caminho segundo BUG 1.



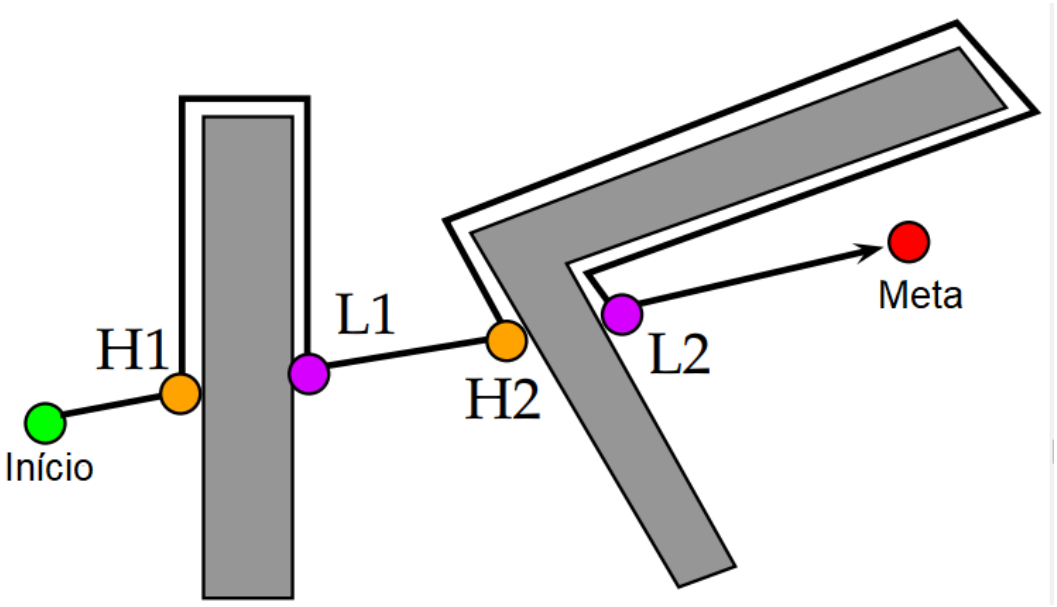
Fonte: Adaptado Plaku (2009)

2.3.4.3 BUG 2

Para Plaku (2009) no algoritmo BUG 2, o robô tenta se mover ao longo da linha de visão em direção ao objetivo. Se um obstáculo for encontrado, o ponto segue o obstáculo até que a linha de visão seja encontrada.

Na Figura 12 temos a trajetória segundo o algoritmo BUG 2.

Figura 12 – Exemplo de caminho segundo BUG 2



Fonte: Adaptado Plaku (2009)

3. MATERIAIS E MÉTODOS

A metodologia empregada neste trabalho consiste nas simulações e na construção de um robô móvel diferencial. Os algoritmos BUG 0 e BUG 1, descritos respectivamente nas seções 2.3.4.1 e 2.3.4.2, foram implementados na simulação e no robô físico. Neste capítulo, serão apresentados os materiais e a sequência de procedimentos desenvolvidas ao longo deste trabalho.

Primeiramente, a modelagem e a simulação do robô se deram através do *software* SimTwo, no qual diversos tipos de robôs podem ser emulados. Neste projeto, foi modelado um robô diferencial representando o protótipo do robô com maior fidelidade.

Em seguida, montou-se o protótipo, cuja estrutura contém os seguintes componentes: um chassi, um jogo de rodas, dois motores, três sensores de linha, um módulo de ponte H L298 e um Arduino Uno. Estes componentes serão descritos com maiores detalhes no decorrer deste capítulo.

3.1 Ferramentas de *Software*

Nesta seção serão apresentados os *softwares* utilizados para o desenvolvimento do projeto.

3.1.1 SimTwo

O SimTwo é um sistema de simulação realista onde é possível implementar vários tipos de robôs:

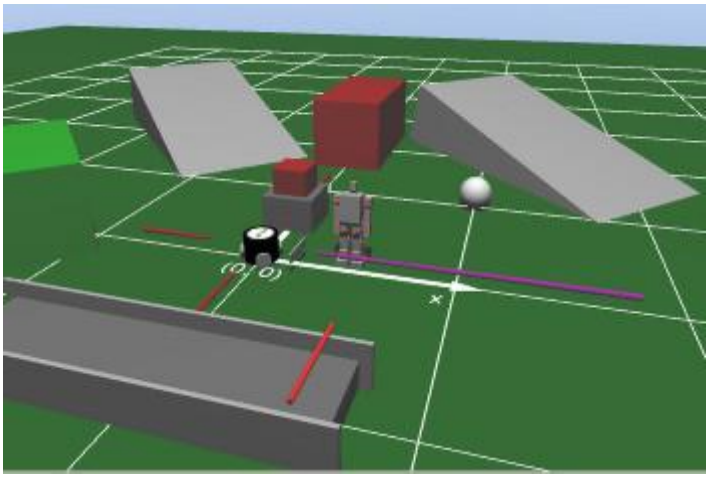
- Robôs móveis com diferentes configurações
    - Diferenciais
    - Com rodas omnidirecionais
- Manipuladores
- Quadrúpedes
- Humanóides
- Qualquer tipo de robô terrestre que possa ser descrito com uma mistura de articulações rotativas e rodas clássicas/omnidirecionais.

- Veículos mais leves que o ar com ou sem hélices para propulsão.

O realismo da dinâmica implementada no SimTwo é conseguido decompondo o robô num sistema de corpos rígidos, motores elétricos e hélices. A dinâmica associada aos corpos é simulada numericamente considerando as suas propriedades físicas: forma, massa e momentos de inércia, forças provenientes dos atritos entre corpos e elasticidade. Certas articulações típicas: eixo, cardan e calha são definidas explicitamente e podem ter associado um sistema de acionamento e sensores.

A Figura 13 mostra uma tela ilustrativa da Interface do *software*.

Figura 13 – Interface do software SimTwo



Fonte: SimTwo (2013).

O sistema de acionamento pode ser constituído por um motor DC, opcionalmente com caixa redutora e um controlador. O controlador pode ser de vários tipos: um PID aplicado ao sinal de posição ou de velocidade ou uma realimentação de estado.

O modelo do motor DC contém vários elementos não lineares tais como saturação da tensão aplicada, limite de corrente e atrito de Coulomb.

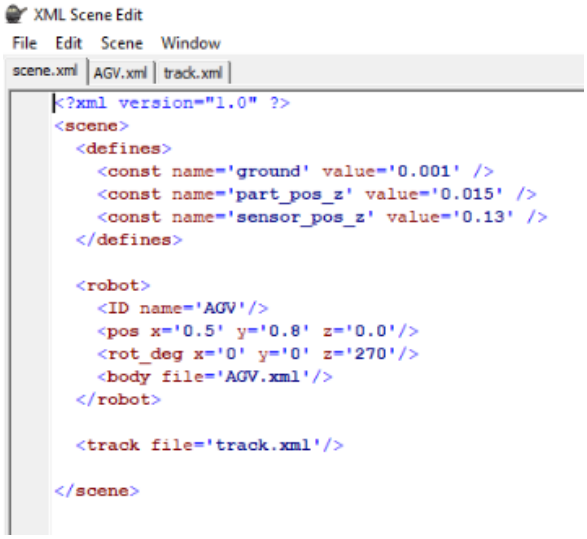
Além do sistema de controle de baixo nível, o SimTwo oferece a possibilidade de fornecer os sinais de referência para esses controladores através de um controlador de mais alto nível implementado pelo usuário. Este controlador pode ser implementado recorrendo:

- A uma linguagem de *script* editável e compilável no próprio SimTwo;
- A um programa remoto que se comunica via rede ou porta serial com o SimTwo.

Para realizar simulações com o SimTwo são necessários dois tipos de linguagem de programação: XML (*Extensible Markup Language*) e Free Pascal.

A principal característica do XML, é criar uma infraestrutura única para diversas linguagens, no caso do SimTwo utiliza-se esse tipo de linguagem para a criação do cenário e para a estrutura do robô (chassi, rodas e sensores). Na Figura 14 temos a interface do SimTwo para programar o XML.

Figura 14 – Tela para programação em XML.



Fonte: Autor.

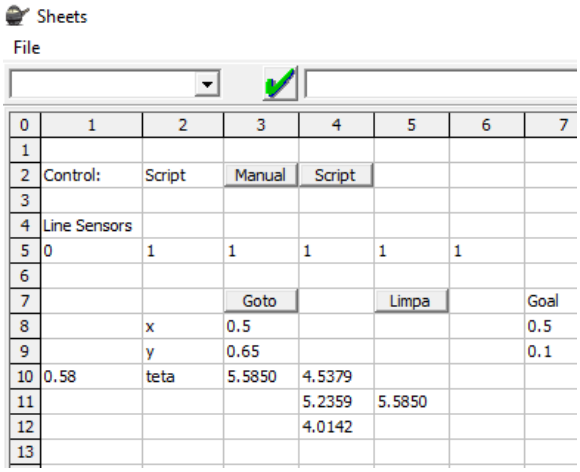
Após a montagem do cenário e do robô atraves do XML, é necessário fazer o código para o robô interagir com o cenário, e é através do código em Free Pascal que isso é feito. É aqui que se programa as condições dos sensores e atuação dos motores. Na Figura 15 temos a interface onde são escritos esses códigos no simulador.

Figura 15 – Interface para programar em Free Pascal

Fonte: Autor.

Além disso, o SimTwo ainda conta com uma interface no formato de planilha, na qual é possível inserir botões, *checkbox* ou valores. A Figura 16, mostra a tela dessa interface, com valores e botões desenvolvidos neste projeto.

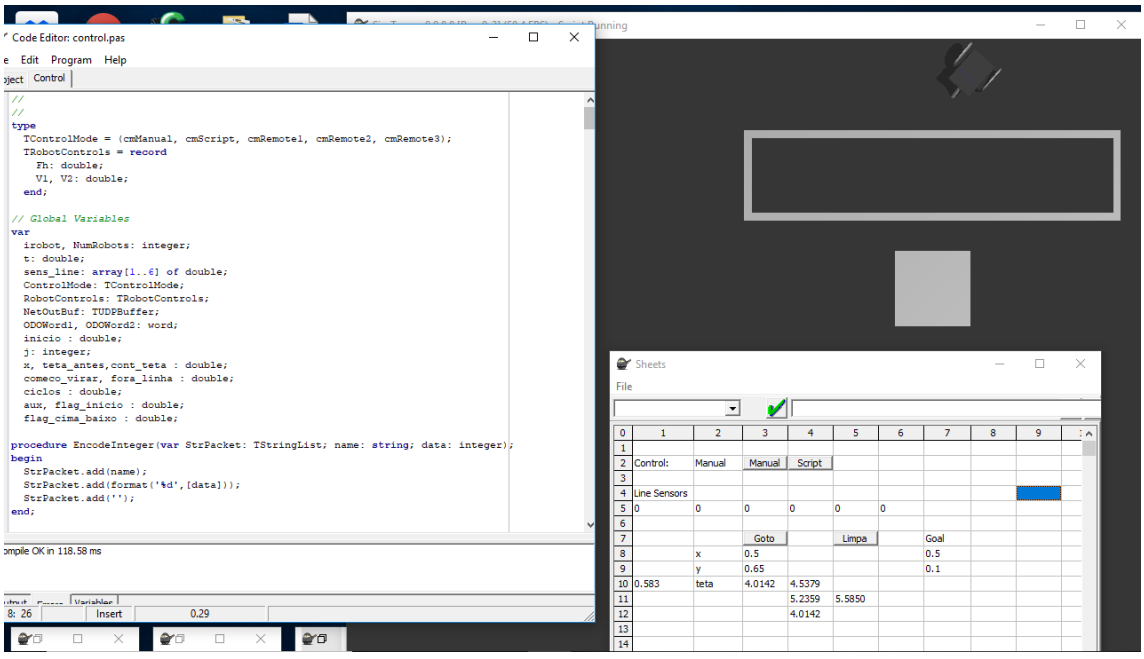
Figura 16 – Interface Sheets desenvolvida para o projeto



Fonte: Autor.

A Figura 17 ilustra a simulação pronta para aplicar os métodos de BUG 0 e BUG 1. No canto superior direito temos o robô com seu obstáculo, que é a linha retangular.

Figura 17 – Simulação do robô do projeto



Fonte: Autor.

3.1.2 Arduino IDE

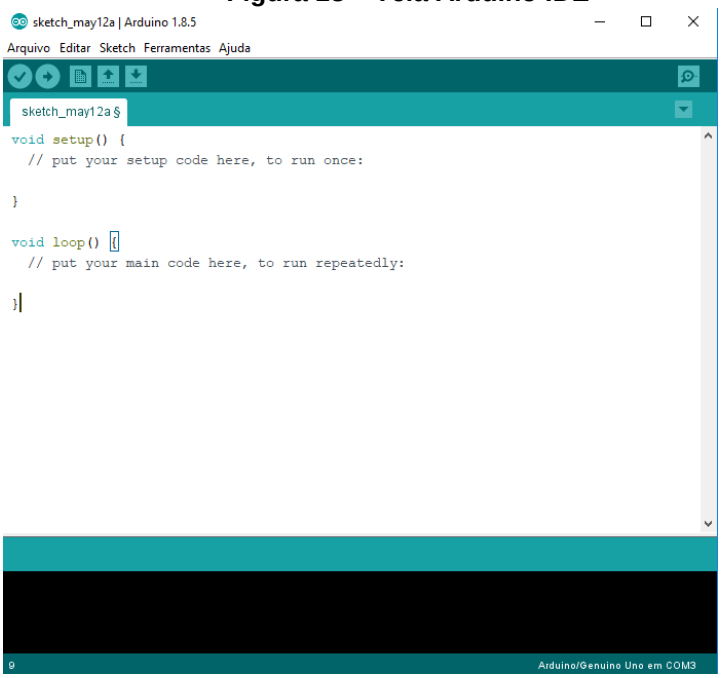
O Arduino IDE (*Integrated Development Environment*) é um aplicativo de plataforma cruzada (para Windowns, macOS, Linux) que é escrito na linguagem de programação Java. Ele é usado para escrever e carregar programas em placas compatíveis com Arduino, mas também, com a ajuda de outros gravadores, outras placas de desenvolvimento.

O Arduino IDE suporta as linguagens C e C++ usando regras especiais de estruturação de código. O Arduino IDE fornece uma biblioteca de *software* a qual provê muitos procedimentos comuns de entrada e saída. O código escrito pelo usuário requer apenas duas funções básicas *setup*(), onde é configurado os pinos de entrada e saída do arduino, e *loop*(), onde é feita a lógica do programa.

O *software* emprega o programa para converter o código de programação executável em um arquivo de texto em uma codificação hexadecimal que é carregada na placa do Arduino por um programa carregador de *firmware* da placa.

Na Figura 18 temos a tela principal do Arduino IDE onde é escrito o código que vai ser embarcado no microcontrolador.

Figura 18 – Tela Arduino IDE



Fonte : Autor.

Todos os códigos desenvolvidos, tanto do simulador quanto do arduino, podem ser vistos atraves do link do Google Drive: https://drive.google.com/drive/u/0/folders/1sjL_imH376S_92ZU63PCUlpBjIWsSCqj

3.2 Arduino

O Arduino é uma plataforma de prototipação eletrônica de código aberto e flexível que utiliza na versão UNO um microcontrolador ATMega328 da Atmel, que será o modelo utilizado neste projeto e pode ser visto na Figura 19. O Arduino pode receber sinais elétricos e lidar com essas informações para controlar luzes, relés, motores, servos ou quaisquer outros atuadores.

Figura 19 – Arduino UNO

Fonte: Fabio Souza (2014)

Para desenvolvimento do *software*, que controlará os pinos e ações do Arduino, será utilizado o ambiente de programação Arduino IDE, disponibilizado para *download* no site oficial do Arduino, que conta com uma interface gráfica simples e autoriza a ser desenvolvidos nas linguagens em C e C++.

O Arduino UNO possui 14 pinos de entradas\saídas digitais, dos quais 6 oferecem PWM, um oscilador a cristal de 16 MHz, uma conexão USB, uma entrada de alimentação e um botão *reset*.

O Quadro 1 mostra melhor definidas as características do Arduino UNO.

Quadro 1 – Características do Arduino UNO

Microcontrolador	ATmega328
Tensão de operação	5 V
Tensão de alimentação (recomendada)	7 a 12 V
Tensão de alimentação (limite)	6-20 V
Entradas e saídas digitais	14 das quais 6 podem ser PWM
Entradas analógicas	6
Corrente contínua por pino de I/O	40 mA
Corrente contínua para o pino 3,3V	50 mA
Memória Flash	32 kB (ATmega328) dos quais 0,5 kB são usados pelo bootloader
Memória SRAM	2 kB (ATmega328)
EEPROM	1 kB (ATmega328)
Frequência do Clock	16 MHz
Dimensões	68,58 mm x 53,34 mm
Peso	150 g

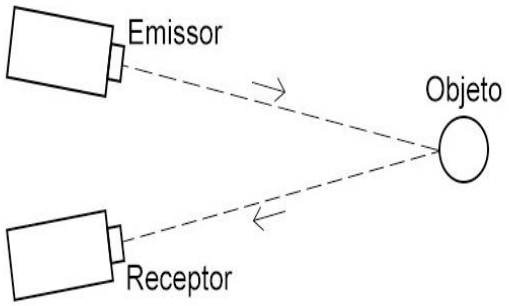
Fonte: Adaptado de Arduino.

3.3 Sensores

Em Niku (2014) foi mostrado que um robô é ineficaz sem ter seus periféricos e sensores. Segundo Romero, Preste, Osório, & Wolf (2014), em sistemas robóticos autônomos são empregadas combinações de sensores para melhor desempenho da máquina. Logo, na maioria dos sistemas robóticos são usados sensores múltiplos do mesmo tipo ou de tipos diferentes para proporcionar uma maior cobertura do ambiente. No trabalho de Reges, Silva, Bezerra & Alexandria (2017) foi desenvolvido um robô com três sensores ultrassônicos para detectar possíveis obstáculos.

Para o projeto do robô foi utilizado o módulo sensor de obstáculo infravermelho IR. Ele emite uma luz infravermelha por um LED (Light Emitting Diode) negro e capta o reflexo com um LED receptor (LED claro). Admite-se que, a luz reflete em superfícies claras e é absorvida em superfícies negras, como a fita isolante. Sendo assim o LED receptor irá detectar a luz infravermelha no branco e não detectar no preto. Na prática, este sensor é formado por um emissor e um receptor, tanto fixados em um mesmo conjunto como separados, dependendo do posicionamento relativo desejado, conforme mostra a Figura 20.

Figura 20 – Modelo de funcionamento do sensor

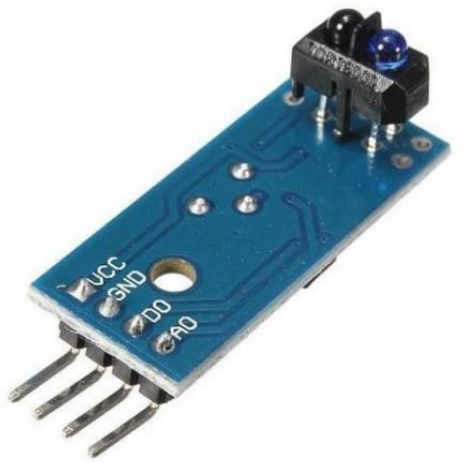


Fonte: Autor

De acordo com Wolf, Osório, Simões & Trindade (2009) os sensores individualmente fornecem apenas uma “visão de mundo”, parcial, incompleta e sujeita a erros, sendo papel do sistema de controle adquirir, unificar (realizar a soma dos sensores) e tratar estas informações de forma robusta e inteligente. Além disto, os comandos de atuação também não são precisos, sendo que muitas vezes uma instrução de avançar em linha reta ou de girar em certa direção também pode não ser executada de forma correta e precisa, pois está sujeita a erros de posicionamento do robô, de acionamento dos motores, bem como está sujeita também a forças e componentes externos (e.g. fricção, gravidade, aceleração, desaceleração, inércia, colisão com obstáculos e derrapagem das rodas). Cabe ao sistema de controle prover técnicas que permitam compensar e corrigir estes erros de modo que as tarefas do robô possam ser executadas corretamente e em segurança.

Neste projeto, utilizou-se três módulos seguidores de linha modelo TCR5000, que pode ser visto na Figura 21.

Figura 21 – Módulo TCR5000



Fonte: Eletrogate

Este módulo possui acoplado um infravermelho (emissor) e um fototransistor (receptor). Ele foi projetado para bloquear outras faixas de luz que não seja a do transistor que no caso mais comum se aplica as iluminações normais, deixando o projeto mais preciso. As características deste módulo podem ser vistas de acordo com o quadro 2.

Quadro 2 – Características do Módulo TCR5000

Tensão de Operação	3,3 a 5 V
Comparador	LM393
Tipo do Detector	Fototransistor
Dimensões	10,2 mm x 5,8 mm x 7 mm
Comprimento de Onda Emissor	950 nm
Máxima Detecção	25 mm

Fonte: Adaptado de Eletrogate

3.4 Motores DC com Ponte H L298

Para a movimentação do robô móvel utilizou-se dois motores DC (Direct Current) (3 a 6 V) com módulo drive ponte H L298. Esse módulo é projetado para controlar cargas indutivas como relés, solenoides, motores DC e motores de passo, permitindo o controle não só do sentido de rotação do motor, como também da sua velocidade, utilizando os pinos PWM do Arduino.

No Quadro 3 temos as especificações da ponte H L298.

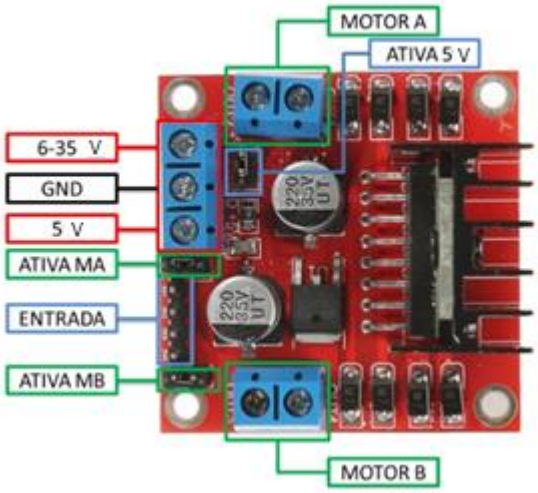
Quadro 3 – Especificações da ponte H L298

Tensão de Operação	4 a 35 V
Chip	ST L298N
Corrente de Operação máxima	2 A por canal ou 4 A
Tensão lógica	5 V
Corrente lógica	0 a 36 mA
Limites de Temperatura	-20 a +135°C
Potência Máxima	25 W
Dimensões	43 mm x 43 mm x 27mm
Peso	30 g

Fonte: Adaptado de Eletrogate.

Na Figura 22 temos o módulo ponte H L298 com seus pinos e jumpers representando.

Figura 22 – Módulo ponte H L298



Fonte: Adaptado Thomsen (2010)

Na Figura 22 temos:

- Motor A, Motor B - Se referem aos conectores para ligação dos motores;
- Ativa MA, Ativa MB - São os pinos responsáveis pelo controle PWM dos motores A e B;
- Ativa 5 V, 5 V - Esse drive possui um regulador de tensão integrado. Quando o drive está operando entre 6 a 35 V, este regulador disponibiliza uma saída regulada de +5 V no pino (5 V) para um uso externo (com jumper), podendo alimentar por exemplo outro componente eletrônico;

- 6 a 35 V, GND (Graduated Neutral Density Filter) – Aqui será conectado a fonte de alimentação externa quando o drive estiver controlando um motor que opere entre 6 a 35 V;
- Entrada – Esse barramento é composto por IN1, IN2, IN3, IN4. Sendo estes pinos responsáveis pela rotação do motor A (IN1, IN2) e motor B (IN3, IN4).

No quadro 4 temos a ordem de ativação do motor A através de IN1 e IN2. O mesmo esquema pode ser aplicado aos pinos IN3 e IN4, que controlam o motor B.

Quadro 4 – Ativação dos motores

MOTOR	**IN1 ou IN3**	**IN2 ou IN4**
Horário	5 V	GND
Anti-Horário	GND	5 V
Ponto Morto	GND	GND
Freio	5 V	5 V

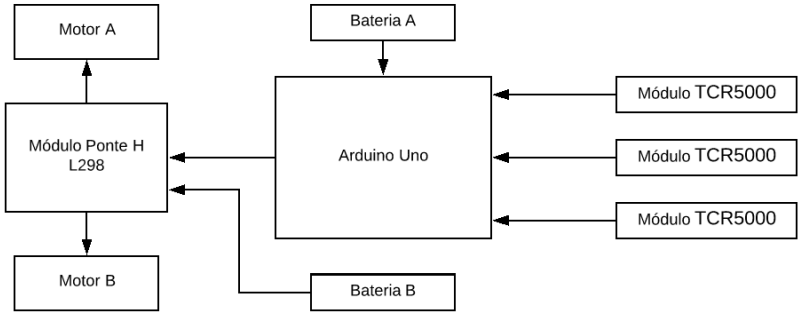
Fonte: Adaptado de Filipeflop.

3.5 Estrutura do Robô e *Hardware*

A estrutura de suporte do robô é feita de acrílico com dimensões de 22 cm x 14,7 cm, duas rodas de borracha com dimensões de 7 cm x 7 cm x 2,6 cm, uma roda boba (universal) para ajudar na sustentação do chassi e dois motores DC (3 a 6 V) que possuem caixa de redução (1:48) que são conectadas independentemente em cada roda.

Todos os componentes descritos até então, são acoplados no chassi do robô. A Figura 23 ilustra um diagrama de blocos simplificado das ligações do *hardware* do robô. Note que são necessárias duas baterias, uma para alimentação do Arduino e outra para o módulo ponte H, que controla os motores do robô.

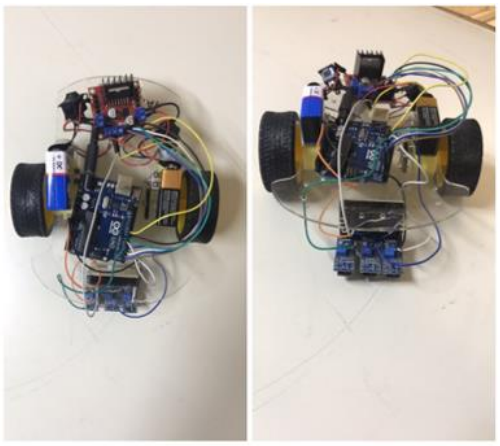
Figura 23 – Diagrama de Blocos do Hardware Simplificado.



Fonte: Autor

A Figura 24 (a) mostra a vista superior do protótipo montado e a Figura 24 (b) a vista frontal do protótipo.

Figura 24 – Protótipo montado



(a) **(b)**

Fonte: Autor.

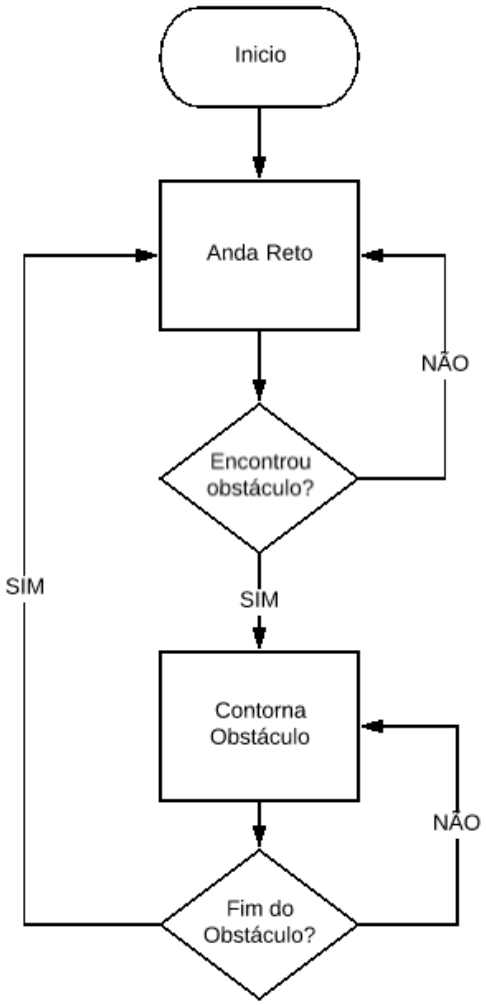
3.6 *Software*

Para desenvolvimento das técnicas BUG 0 e BUG 1, primeiro foi feita a simulação dos metodos. Sendo feita a programação em XML do cenário e em seguida o código de programação do robô em Free Pascal.

Para o robô real, foi feita uma tradução e adaptação de Free Pascal para C, que é o código aceito pela placa Arduino Uno.

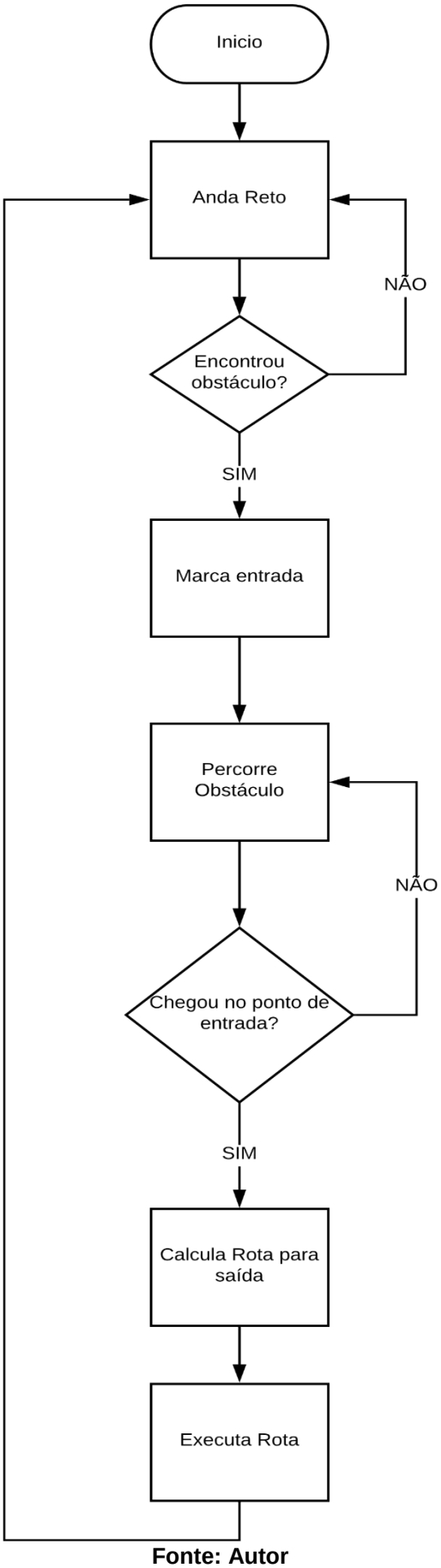
As Figuras 25 e 26 ilustram os fluxogramas dos processos feitos para o robô em ambas as linguagens para os métodos BUG 0 e BUG 1, respectivamente.

Figura 25 – Fluxograma BUG 0



Fonte: Autor.

Figura 26 – Fluxograma BUG 1



Fonte: Autor

3.7 Custo do projeto

O quadro 5 mostra os custos dos insumos para a construção deste projeto. Todos os *softwares* utilizados para desenvolvimento deste projeto são *open source*.

Quadro 5 – Preço dos insumos do projeto

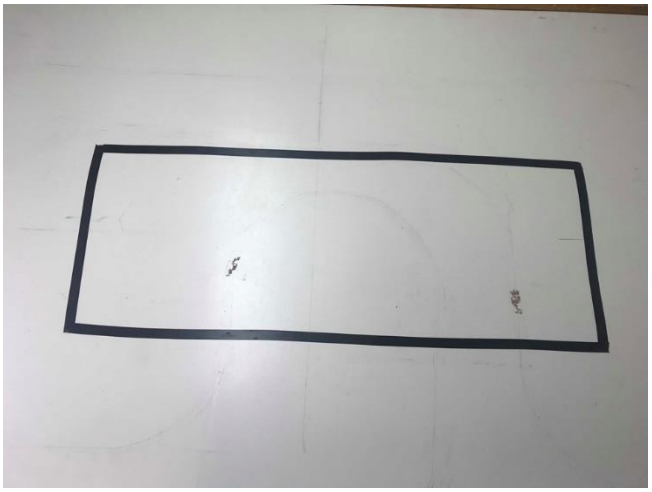
Descrição de Itens	Quantidade	Valor Unitário	Valor Total
Placa Arduino UNO	1	R$ 54,90	R$ 54,90
Chassi em Acrílico	1	R$ 12,90	R$ 12,90
Motores DC (3 a 6 V)	2	R$ 20,30	R$ 40,60
Rodas de Borracha	2	R$ 6,9	R$ 13,80
Roda Boba (Universal)	1	R$ 6,90	R$ 6,90
Módulo TCR5000	3	R$ 9,90	R$ 29,70
Modulo drive ponte H L298	1	R$ 19,90	R$ 19,90
Bateria (9 V)	2	R$ 22,50	R$ 45,00
Chave Gangorra	2	R$ 1,50	R$ 3,00
Jogo de parafusos	1	R$ 3,00	R$ 3,00
Jogo de Cabos (*Jumpers*)	1	R$ 9,90	R$ 9,90
Fita Isolante (10 m)	1	R$ 3,90	R$ 3,90
		Valor Total	R$ 243,50

Fonte: Autor.

4. RESULTADOS E DISCUSSÕES

Os experimentos realizados neste projeto foram direcionados a medir o tempo de execução em cada um dos métodos em ambiente simulado e real, para uma pista de forma retangular. A validação do protótipo ocorreu sobre uma plataforma plana branca de dimensões 1,0 m x 0,3 m. A rota possui perímetro de 2,6 m e foi desenhada com fita isolante preta, conforme mostrado na Figura 27.

Figura 27 – Pista do robô

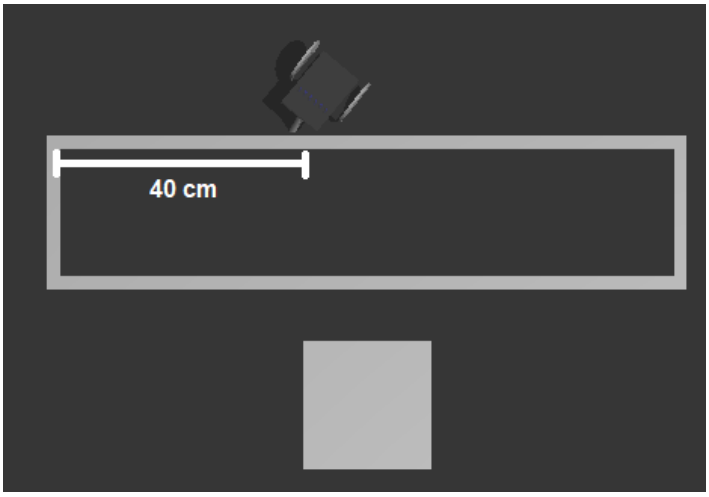


Fonte: Autor.

4.1 Simulação *versus* Real

A validação dos algoritmos foi realizada através da simulação do robô com o SimTwo, e obtidas amostras do tempo de execução de cada um. A ratificação foi feita através de uma posição inicial fixa do robô, posicionado a 40 cm de distância em relação a borda da direita. A Figura 28 ilustra a posição do robô simulado, a linha que tem a forma retangular representa o obstáculo para o robô e o quadrado cinza uma representação do seu objetivo.

Figura 28 – Posição de testes

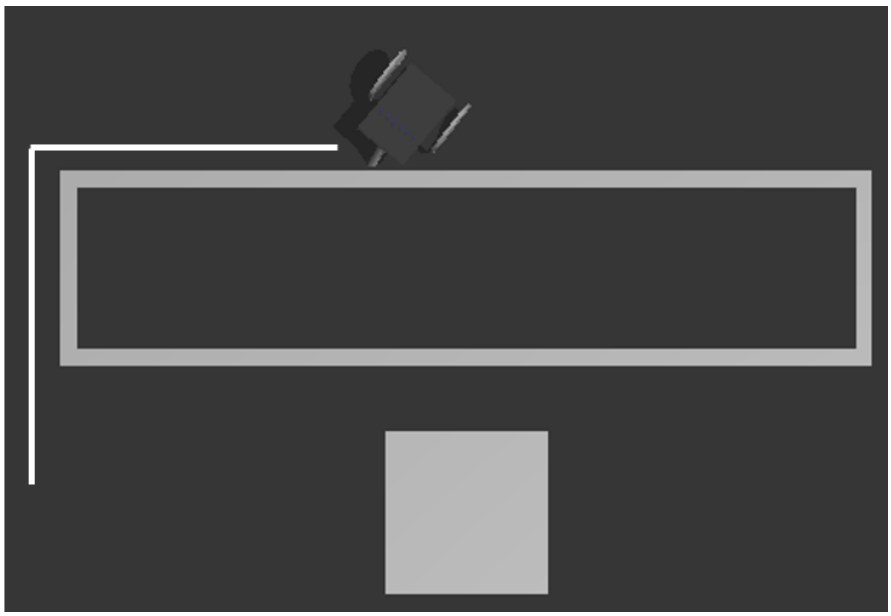


Fonte: Autor.

O percurso realizado pelo robô no método BUG 0 pode ser visto na Figura 29 em sentido anti-horário, note que a linha branca representa a rota, e o percurso do método BUG 1 é encontrado na Figura 30, também em sentido anti-horário, a linha pontilhada representa o mapeamento no método e a linha continua a rota executada pelo robô.

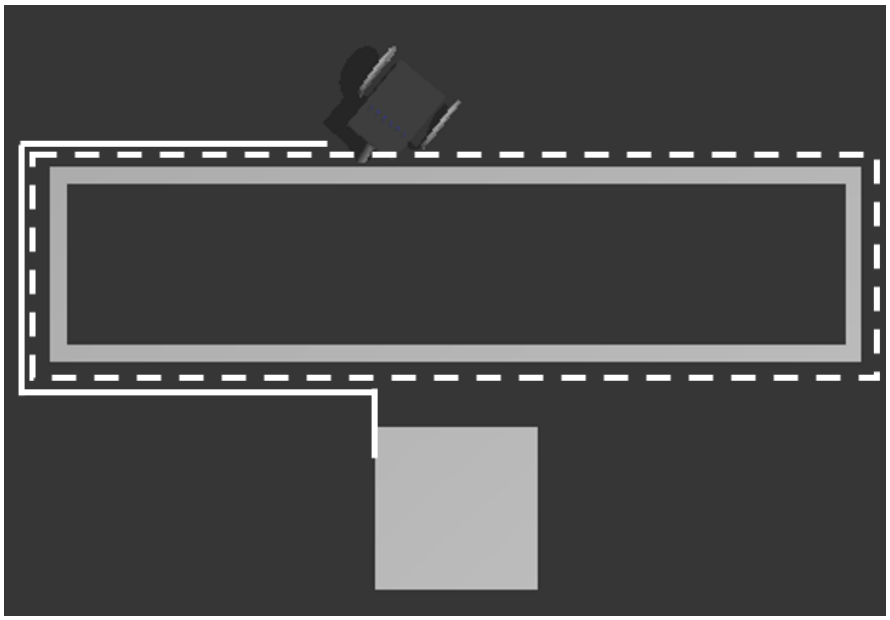
No caso no algoritmo BUG 1, deve-se ter um objetivo bem definido para a execução da rota, para isso foi definido o objetivo do robô, como seguir em linha reta após a execução do percurso.

Figura 29 – Caminho pelo método BUG 0.



Fonte: Autor.

Figura 30– Caminho pelo método BUG 1



Fonte: Autor.

Os algoritmos, BUG 0 e BUG 1, foram testados na pista real e o tempo de execução destes podem ser vistos no quadro 6, juntamente com o tempo simulado.

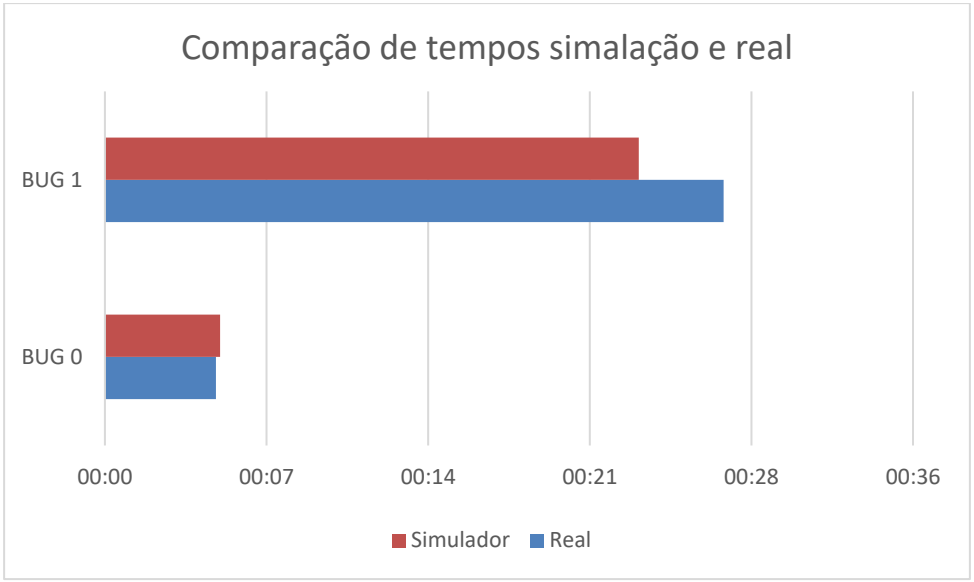
Quadro 6 – Tempos de execução da rota

	Tempo (s)		
Método	Real	Simulado	Diferença
BUG0	04:57	05:08	00:11
BUG1	27:34	23:47	03:47

Fonte: Autor.

O gráfico 1 mostra os valores reais e simulados em relação ao tempo de execução da rota tomada pelo robô. Observe que, no método BUG 0 não houve a diferença é de milésimos de segundo no tempo de execução, o que já não ocorre no BUG 1 tendo uma diferença de 3:47 segundos.

Gráfico 1 – Tempos simulados e reais dos métodos BUG 0 e BUG 1.



Fonte: Autor.

Para os valores reais, foram executadas diversas vezes o trajeto com o robô, e feita uma média nos tempos para melhor comparação, isso devido a variações no nível da bateria que está ligada no módulo ponte H, que influencia diretamente na força dos motores. O tempo de rota, para comparação, foi medido a partir do instante que o robô entra na linha até o instante que ele a deixa.

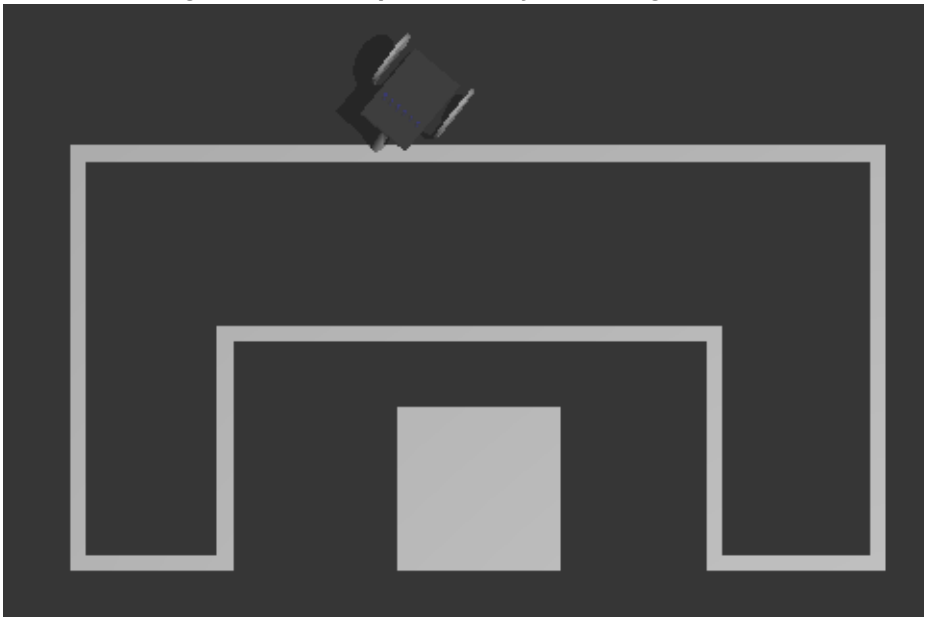
No caso do algoritmo BUG 0 nota-se que não houve grande variação do tempo simulado em relação ao real, o que já era esperado devido a simplicidade do código, que conta apenas com leitura de sensores e condições mais básicas.

No BUG 1 houve uma diferença maior, mas ainda assim está dentro dos limites aceitáveis. Essa diferença se deu pelo nível carga da bateria, que interfere diretamente na performance do protótipo. Para um melhor desempenho deve-se utilizar baterias totalmente carregadas, caso contrário os motores apresentam certa dificuldade de tirar o carro da inércia. Outro ponto a ser analisado são as fitas da pista real que não estão totalmente retas como as da simulação.

4.2 Comparação de Performance dos Algoritmos

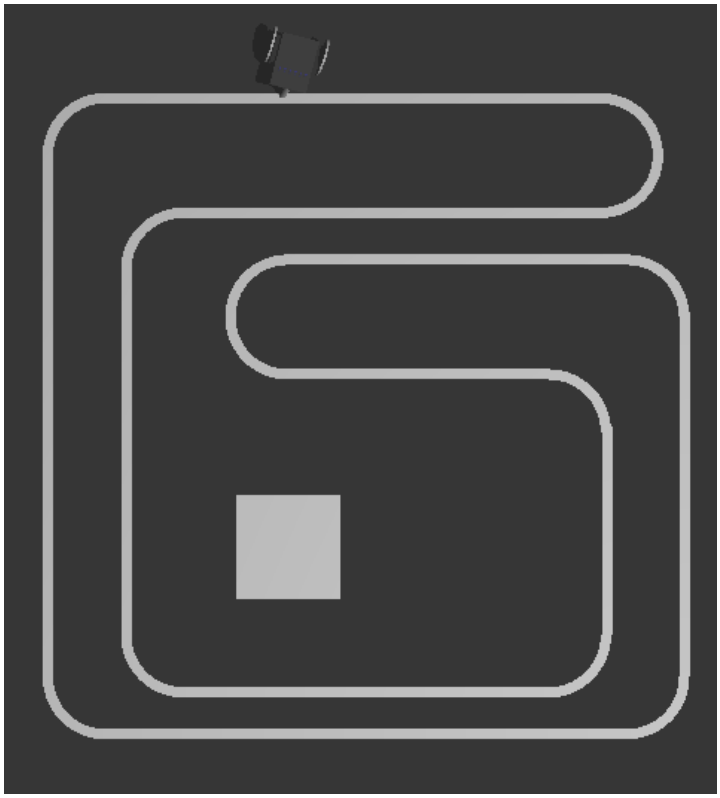
Para validação dos algoritmos desenvolvidos foram aplicadas novas simulações, dos algoritmos BUG 0 e BUG 1, em ambientes simulados diferentes. As Figuras 31 e 32 ilustram os novos cenários desenvolvidos para confirmação da funcionalidade desses algoritmos.

Figura 31 – Pista para validação dos algoritmos.



Fonte: Autor.

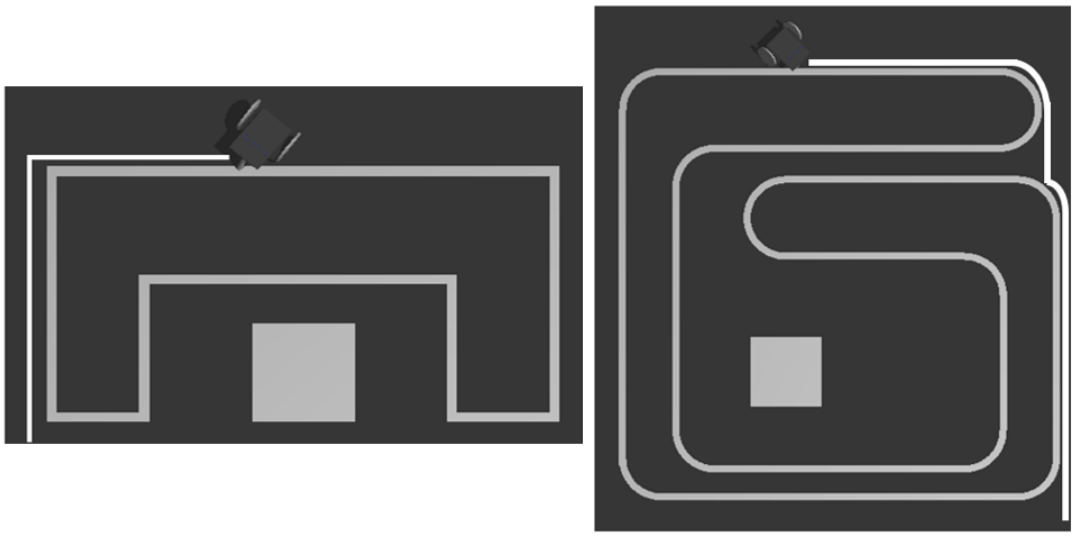
Figura 32 – Pista para validação dos algoritmos.



Fonte: Autor.

O caminho para os novos ambientes segundo o algoritmo BUG 0 pode ser visto na Figura 33. Observe que a teoria se confirma nesses ambientes, o robô não consegue chegar a uma solução da pista mesmo tendo uma solução.

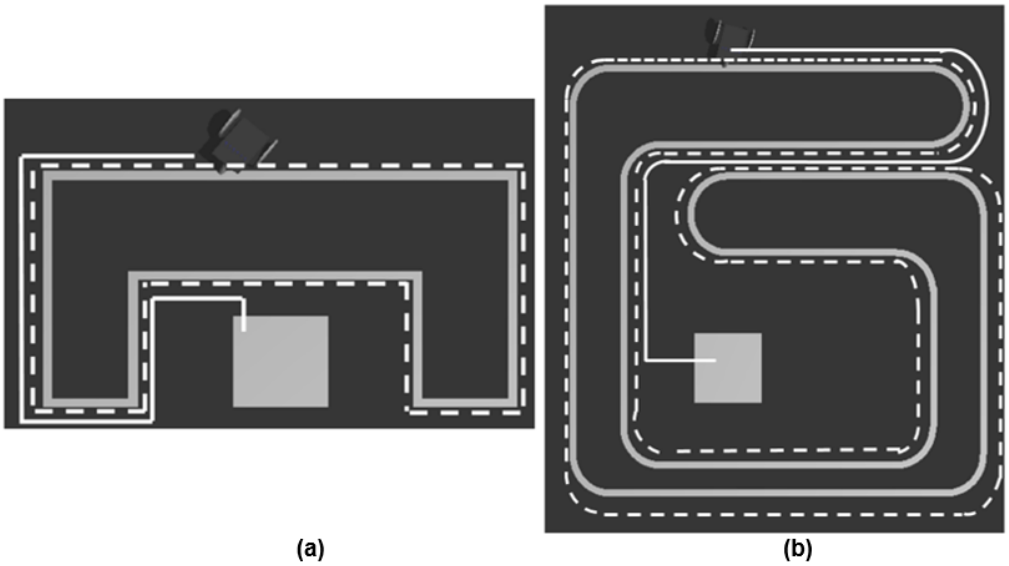
Figura 33 – Caminho pelo método BUG 0.



Fonte: Autor.

Para o método BUG 1 sempre é encontrado uma solução independente da pista, porém o objetivo deve estar bem definido para o robô. Nesse caso o objetivo é chegar até o quadrado cinza. A Figura 34 ilustra o caminho para as novas pistas, observe que o caminho em pontilhado é a rota toda do robô pelo obstáculo, e a linha continua a rota executada ate chegar no ponto desejado.

Figura 34 – Caminho pelo método BUG 1.

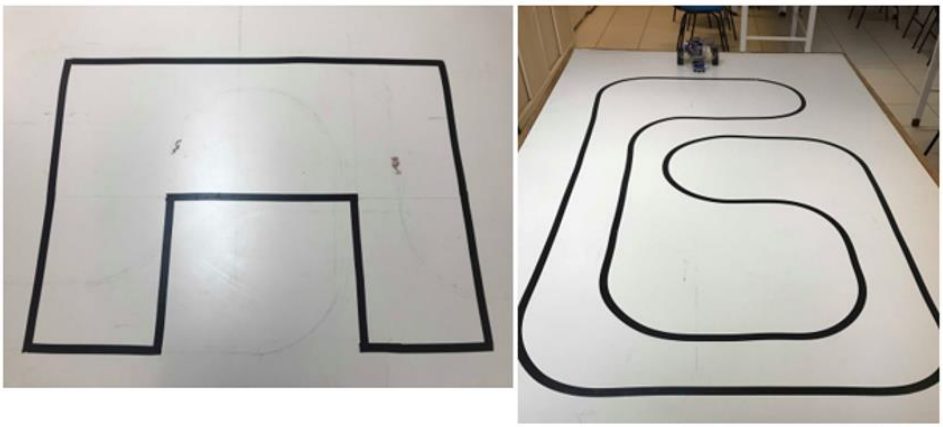


(a) **(b)**

Fonte: Autor.

Para validação do robô real também foram feitas pistas com as mesmas formas das simulações. A figura 35 ilustra as pistas reais utilizadas para testes da execução do robô.

Figura 35 – Pistas reais para o robô.



(a) (b)

Fonte: Autor.

Para o algoritmo BUG 0, o código se mostrou satisfatório na execução real para ambas as estruturas.

O algoritmo BUG 1, para a pista de esquerda, segundo a Figura 35, teve um ótimo desempenho. Porém para a pista da direita não consegue uma solução de rota. Isso se deu devido ao elevado número de pontos necessários para reconhecimento da pista. Observou-se que, por apresenta curvas convexas, são necessárias mais amostras para mapeamento desta pista, assim necessitando de mais memória. Como a placa do Arduino Uno apresenta uma memória RAM de apenas 2 KB há uma sobrecarga da memória, como solução seria utilizar uma placa de desenvolvimento com uma memória RAM maior.

5 CONCLUSÕES

Diante dos resultados obtidos verificou-se que o algoritmo BUG 1, por se tratar de um algoritmo com baixa complexidade e de fácil implementação, mostrou-se ser uma alternativa para a utilização em locomoção autônoma, com relação ao método BUG 0. Com relação a este, apesar de ser robusto, ele apresenta uma certa limitação, para algumas aplicações. Ainda, vale ressaltar como ponto positivo que ambos não necessitam de um conhecimento prévio do mapa para executarem.

Quanto ao SimTwo, é uma excelente opção para o planejamento de trajetórias, pois apesar de apresentar uma maior complexidade de código e um gasto computacional considerável com execução, realiza a exploração do mapa para criação da trajetória de forma inteligente. Entretanto, para que possa ser utilizado, há a necessidade de se conhecer o mapa previamente, uma vez que, este planeja a partir de um mapa estático, e caso haja alguma alteração no ambiente, ele não recebe a atualização até que o comando de redesenho seja executado.

Desta forma, temos o BUG 1 excelente por sua simplicidade de código, baixo custo de execução e bom desempenho para mapas simples e o SimTwo como um excepcional simulador de robôs e trajetórias, porém está atrelado com a necessidade de mapeamento prévio.

Para trabalhos futuros, fica a proposta de utilizar o SimTwo e o robô real para desenvolvimento do algoritmo BUG 2. Além disso utilizar outra placa de desenvolvimento com maior capacidade de memória RAM, para implementação do robô real para testes em pistas maiores.

6 REFERÊNCIAS

AQUABOAT. **Aquaboat Extreme**. Disponível em < http://www.aquaboat.com> Acesso em 12 de abril de 2018.

ARDUINO. 2013. Disponível em < http://www.arduino.cc> Acesso em 20 de abril de 2018.

CAPELLI, B. M., **Desenvolvimento de uma estufa controlada e monitorada remotamente**. EECS-USP, São Carlos. Novembro, 2014

CARPENTER, K.; GLASGOW, E., **Constraint Planning**. 2013. Disponível em <https://personalrobotics.ri.cmu.edu/files/courses/16662/notes/constraints/16662Lecture15.pdf > Acesso em: 10 de maio de 2018.

CHOSET, H., **Robotic Motion Planning: Potential Functions**. 2014. Disponível em <https://www.cs.cmu.edu/~motionplanning/lecture/Chap4-Potential-Field_howie.pdf > Acesso em: 12 de maio de 2018

COSTA, P.; GONÇALES, L.; LIMA, J.; MALHEIROS, P., **SimTwo realistic simulator: a tool for the development and validation of robot software**. Universidade de Porto, Portugual, 2011.

DONGBIN, Z.; JIANQIANG, Y., **Robot Planning with Artificial Potential Field Guided Ant Colony Optimization Algorithm**. ICNC 2006, Part II. Springer-Verlag Berlin Heidelberg, 222 – 231. 2006.

ELETROGATE – Loja de Arduinos. Disponível em <www.eletrogate.com> Acesso em: 17 de outubro de 2018.

FERREIRA, W. R. B.; PEREIRA, L. R.; CARVALHO J. C. M., **Planejamento de trajetórias com B-spline cúbica para robôs móveis**. Universidade Federal de Uberlândia. 2011.

FILIPEFLOP – Componentes eletrônicos. Disponível em <www.filipeflop.com> Acesso em: 17 de outubro de 2018

GARCIA, T. R.; VALENTE, R. L.; JESUS.; J. C.; SCHIRMER, R. D.; GAMARRA, D. F. T., **Construção de um robô móvel para ensino das disciplinas dos cursos de engenharia**. Universidade Regional do Noroeste do Estado do Rio Grande do Sul. 2017.

HUSKVARNA. **Automower (Robotic lawn mower)**. Desponível em: <http://www.automower.com/> Acesso em 12 de abril de 2018.

IROBOT. **Cleaning Robots (Roomba, Scooba, Dirt Dog, Verro)**. Disponível em <http://www.irobot.com/> Acesso em 12 de abril de 2018.

ISHIWAKA, Y.; YOSHIDA, T.; YOKIO, H.; KAKAZU, Y., **An approach to the piano mover's problem using hierarchic reinforcement learning**. 2004. Disponível em <https://www.researchgate.net/publication/220238901_An_Approach_to_the_Piano_Mover's_Problem_Using_Hierarchic_Reinforcement_Learning > Acesso em: 07 de maio de 2018.

JUNG, C.R.; HEINEN, F.; KELBER, C.; OSÓRIO, F.S., **Navegação de Veículos de Carga Autônomos utilizando Visão Computacional com Algoritmo de Segmentação por Cores**. In: IV Congresso Internacional de Automação, Sistemas e Instrumentação, 2004, São Paulo: ISA South America, 2004. v. 1. p. 1-10. 2004.

KLOETZER, M.; MAHULEA, C.; GONZALES, R**., Optimizing cell decomposition path planning for mobile robots using diferente metrics**. Internation Conference on System Theory, Control and Computing. Outubro 2015.

LAUMOND, J. P., **Robot motion planning and control**. LAAS-CNRS, Toulouse, France, 2014.

LOZANO-PEREZ, T.; WESLEY, M.A., **An Algorithm for Planning Collision Free Paths among Polyhedral Obstacles**. Communications of the ACM, p. 560-570. 1979.

MASEHIAN, E.; AMIN NASERI, M. R**., A Tabu Search based Approach for Online Motion Planning**. Journal of Robotic Systems. 2004.

MASEHIAN, E; AMIN NASERI, M. R., **A Voronoi diagram-visibility graph-potential fields compound algorithm for robot motion**. Journal of Robotic Systems. 2004.

MOJTAHEDZADEH, R., **Robot obstacle avoidance using the kinect**. Royal Institute of Technology. Estocolmo, 2011.

NIKU, S. B., **Introductions to robotics: analysis, control, applications**. New Jersey: Prentice Hall. 2014

OTTONI, G. L., **Planejamento de trajetórias para robôs móveis**. Universidade Federal do Rio Grande, 2000.

PEDROSA. D. P. F.; MEDEIROS, A. A. D.; ALSINA, P. J., **Um método de geração de trajetória para robôs não-holonômicos com acionamento diferencial**. Universidade Federal do Rio de Janeiro, 2003.

PIARDI, L. F., **Application of a mobile robot to spatial mapping of radiactive substances in indoor environment**. Instituto Politécnico de Bragança, Bragança, 2018.

PLAKU. E., **Introduction to robotics – Bug Algorithms**. Departamente of Eletrical Engineering and Computer Science Catholic University of America, 2009.

POLIDORO, H. L., **Planejamento de Trajetórias em ambientes com propriedades dinâmicas**. ICMC-USP, São Carlos. Julho, 2010.

REGES, J. P.; SILVA J. L. N.; BEZERRA, L. C. S.; ALEXANDRIA, A. R**., Controle fuzzy de robô diferencial**. Instituto Federal do Ceará, 2017.

SMIDT, A. C. G., **Implementação de uma plataforma robótica controlada remotamente utilizando Arduino**. EECS-USP, São Carlos. Junho, 2013.

RICON, R. L., **Framework de definições de trajetória para robôs móveis**. Universidade de Brasília. 2014.

ROLAND SIEGWART, I. R. N., **Introduction to autonomous mobile robots**, 1st. The MIT Press. 2004. isbn: 9781417561810.

SONY. **AIBO Entertainment Robots**. Disponível em <http://support.sonyeurope.com/aibo/> Acesso em 12de abril de 2018.

SOUZA, S. C. B., **Planejamento de trajetória para um robô móvel com duas rodas utilizando um algoritmo A-estrela modificado**. Rio de Janeiro: UFRJ/COPPE, 2008.

SCHWARTZ, J.T.; SHARIR, M., **On the 'piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds**. Adv. Applied Mathematics, vol.4, p.298–351,1983.

TANG, S. W.; KHAKSAR, N. B.; ISMAIL, N. B.; ARIFFIN, M. K. A., **A reviw on robot motion planning approaches**. Universiti Putra Malasya Press, 2012.

WOLF, D. F.; OSÓRIO, F. S.; SIMÕES, E.; TRINDADE, O. J., **Robótica Móvel Inteligente: Da Simulação às Aplicações no Mundo Real**. ICMC, Universidade de São Paulo – USP, São Carlos. 2009.