UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA
CAMPUS UNIVERSITÁRIO DE VITÓRIA DA CONQUISTA - BAHIA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

CONSTRUÇÃO DE UM PROTÓTIPO DE BAIXO CUSTO PARA O DESVIO DE OBSTÁCULOS COM SENSOR DE ULTRASSOM

THÓMAS JÉFFERSON DA SILVA TEIXEIRA

Vitória da Conquista / BA
2015

THÓMAS JÉFFERSON DA SILVA TEIXEIRA

CONSTRUÇÃO DE UM PROTÓTIPO DE BAIXO CUSTO PARA O DESVIO DE OBSTÁCULOS COM SENSOR DE ULTRASSOM

Trabalho de conclusão de curso, apresentado ao curso de Ciências da Computação, da Universidade do Sudoeste da Bahia, em Vitória da Conquista - Bahia, como requisito parcial para a obtenção do título de Bacharel em Ciências da Computação.

Orientador: Prof. Dr. Roque Mendes Prado Trindade

Co-orientador: Prof. Ms. Marcos Gomes Prado

Vitória da Conquista - BA

2015

____, Teixeira, Thómas Jéfferson da Silva,
CONSTRUÇÃO DE UM PROTÓTIPO DE BAIXO CUSTO PARA O DESVIO DE OBSTÁCULOS COM SENSOR DE ULTRASSOM / Thómas Jéfferson da Silva Teixeira, Vitória da Conquista, Bahia.

54p.: il.; 30 cm

(Monografia apresentada ao Curso de Ciências da Computação, da Universidade Estadual do Sudoeste da Bahia - UESB, sob a orientação do prof. Dr. Roque Mendes Prado Trindade e da co-orientação do Prof. Ms. Marcos Gomes Prado.

1. Robótica móvel. 2. Desvio de obstáculos 3. Baixo custo. 4. Ultrassom. I. Trindade, Roque Mendes Prado. II. Prado, Marcos Gomes, III Título.

CDD – _____

(Nome da bibliotecária):_____, Bibliotecária CRB _____

UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA
CAMPUS UNIVERSITÁRIO DE VITÓRIA DA CONQUISTA / BAHIA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

MONOGRAFIA

CONSTRUÇÃO DE UM PROTÓTIPO DE BAIXO CUSTO PARA O DESVIO DE OBSTÁCULOS COM SENSOR DE ULTRASSOM

Autor: Thómas Jéfferson da Silva Teixeira
Orientador: Prof. Dr. Roque Mendes Prado Trindade
Co-orientador: Prof. Dr. Marcos Gomes Prado

Este exemplar corresponde à redação final da monografia defendida por **Thómas Jéfferson da Silva Teixeira** e aprovada pela Comissão Julgadora.

Data: 27.07.2015
Assinatura:.....
Prof. Dr. Roque Mendes Prado Trindade
Prof. Ms. Marcos Gomes Prado

COMISSÃO JULGADORA

_____

Prof.

_____

Prof.

_____

Prof. Dr. Roque Mendes Prado Trindade
Orientador
Vitória da Conquista / 2015

DEDICATÓRIA

AGRADECIMENTOS

Agradeço, primeiramente a Deus, que sempre me deu o discernimento de acreditar na força superior de nosso pai celestial.

A meus pais José Bonifácio Teixeira e Maria Soares da Silva Teixeira pelo apoio, amor, carinho e atenção em todas as horas que tive de lhe pedir.

Aos meus irmãos Lara Shelene e José Bonifácio Teixeira Filho pelo apoio e carinho.

A todos os meus familiares avó Lino, Vovós Zizi e Judite, tios, tias e primos pelo incentivo nessa caminha acadêmica.

A todos(as) as pessoas que acreditam na educação e na construção continua do conhecimento.

Aos meus orientador e co-orientador pela colaboração, trabalho em equipe desenvolvido durante as atividades acadêmicas. Meu respeito e minha gratidão.

A todos os professores do curso de Ciências da computação.

Aos colegas pelo companheirismo, amizade construída nessa caminhada acadêmica.

Aos colegas e professores do colegiado de Ciências da Computação.

Aos participantes da Banca de Defesa do curso de Ciências da Computação.

Agradeço a todos que direto ou indiretamente contribuíram para a conclusão desse curso.

Meu muito obrigado!

RESUMO

Para que um robô móvel possa navegar em segurança em ambientes dinâmicos é necessária à capacidade de detectar e evitar obstáculos em tempo real, sendo um importante requisito para qualquer aplicação prática de veículos autônomos. Esse trabalho propõe o uso de uma solução de baixo custo para a navegação local do robô em ambientes dinâmicos, onde o robô desconhece o ambiente e deve navegar do ponto de origem até o destino em segurança. Um dos requisitos desse projeto é que o algoritmo seja simples para executar em microcontroladores de baixo poder computacional e que tenha um resultado bom o suficiente para ser usado em aplicações reais em robôs móveis em ambientes indoor. É apresentada uma possível arquitetura do robô que se adéqua ao método de desvio de obstáculos. No trabalho é feito um levantamento dos principais algoritmos de desvio de obstáculos e é apresentado uma análise do melhor algoritmo para a solução do problema. Entre eles o algoritmo escolhido foi o Bubble Rebound desenvolvido por Susnea et. al. (2010).

Palavras-chave: Robótica móvel. Desvio de obstáculos. Baixo custo. Ultrassom.

ABSTRACT

For a mobile robot can navigate safely in dynamic environments is required the ability to detect and avoid obstacles in real time is an important requirement for any practical application of autonomous vehicles. This paper proposes the use of a low-cost solution to local navigation robot in dynamic environments where the robot ignores the environment and must navigate from point of origin to the destination safely. One of the requirements of this project is that the algorithm is simple to run on low computing power microcontrollers and has a good enough result for use in real applications in mobile robots in indoor environments. A possible architecture of the robot fits the obstacle avoidance method is displayed. In the work is done a survey of the main obstacle avoidance algorithms and presented an analysis of the best algorithm to solve the problem. Including the chosen algorithm was developed by Bubble Rebound Susnea et. al. (2010).

Keywords: Mobile robotics. Obstacle avoidance. Low cost. Ultrasound.

LISTA DE QUADROS E FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

DOF	Graus de liberdade
IMU	Unidade de Medição Inercial
INS	Sistema de Navegação Inercial
MEDIA LAB	Laboratório de mídia
MIT	Massachusetts Institute of Technology
RUR	Rossum’s Universal Robots
VANT	Veículo Aéreo Não Tripulado

SUMÁRIO

1 INTRODUÇÃO

Para facilitar a vida da humanidade, o homem sempre buscou meios de construir máquinas autônomas com capacidade para auxiliar em suas tarefas diárias. Isaac Asimov, em umas das suas obras, prevê que essa evolução chegaria a tal ponto, que essas máquinas seriam capazes de superar as habilidades humanas.

Essas máquinas são conhecidas como Robô, que tem sua origem na palavra tcheca robota, que significa "trabalho forçado". Elas foram usadas primeiramente pelo dramaturgo Karel Capek, na peça de teatro, em 1921, denominada de Rossum's Universal Robots (RUR). Nessa peça o inventor Rossum criou um servo para substituir os seres humanos, em qualquer tipo de trabalho, mas um dia eles se revoltariam e destruiriam a raça humana. (MURPHY, 2000)

Já o termo Robótica, foi popularizado pelo escritor de Ficção Científica, nascido na Rússia com nacionalidade norte-americana, Isaac Asimov, para representar a ciência voltada aos estudos dos robôs. Na sua obra "I, Robot", de 1950, criou "Leis da robótica" que, para ele, seria as leis fundamentais para criação dos robôs no futuro, sendo elas:

- "Um robô não pode fazer mal a um ser humano, e nem por omissão, permitir que algum mal lhe aconteça".

- "Um robô deve obedecer às ordens dos seres humanos, exceto quando estas contrariarem a Primeira lei.".

- "Um robô deve proteger a sua integridade física, desde que, com isto, não contrarie a Primeira e a Segunda lei.".

- E em obras posteriores, Asimov acrescentou a lei zero, que acima de todas as outras: "um robô não pode causar mal à humanidade ou, por omissão, permitir que a humanidade sofra algum mal". (DUDEK & JENKIN, 2010).

Entre as variadas definições de um robô, um que se encaixa nesse contexto, é citado por Romero (2014, p. 6) que:

> Um robô pode ser uma máquina capaz de executar tarefas repetitivas, sejam elas guiadas (teleoperadas) ou predefinidas (pré-programadas), mas também é capaz de realizar tarefas de modo inteligente (autônomo), sendo capaz de perceber o ambiente, tomar decisões e agir conforme a situação em que se encontra.

A partir desse conceito é possível definir a maioria dos robôs.

1.1 - Aplicações

Atualmente, os robôs fazem parte da nossa vida, tendo várias aplicações em diversos ambientes. De acordo com Kumar, Bekey & Zheng (2005), Murphy (2000) e a International Federation of Robotics as principais aplicações são:

• Na indústria: os manipuladores robóticos são largamente utilizados nas indústrias e grandes fábricas para automatizar o processo de construção do produto. Como, por exemplo, nas indústrias de automóveis os manipuladores robóticos realizam as tarefas de montagem, pintura e soldagem de automóveis, que são feitas por manipuladores, que exigem velocidade no processo e na alta precisão.

Figura 1 – Manipulador robótico

Fonte: www.abb.com

• Na medicina: a robótica auxilia a medicina desenvolvendo equipamentos cirúrgicos que são menos invasivos, como o robô Da Vinci, representado na figura 3, podendo utilizar os robôs para realizar as cirurgias à distância, e, sendo capazes de corrigir os pequenos tremores das mãos dos cirurgiões. Outra aplicação da robótica é o desenvolvimento de próteses para a reabilitação de deficientes físicos, como as próteses utilizadas por Hugh Herr, que lidera o grupo de pesquisa Biomecatrônica no MIT Media Lab. Ele projetou suas próprias pernas biônicas, que é o primeiro sistema de pé e panturrilha biônica, do mundo chamado de BIOM[1].

[1] BIOM. Descrição em: https://www.media.mit.edu/people/hherr.

Figura 2 – BIOM

Fonte: https://www.media.mit.edu/people/hherr

Figura 3 – Robó Da Vinci

Fonte: http://www.davincisurgery.com/

• Na exploração: Os robôs podem explorar ambientes impróprios ou hostis ao homem, como o robô móvel Sojourner da missão Mars Pathfinder da NASA, na exploração do solo de Marte, em 1996, e, os robôs Spirit e Opportunity do Programa Mars Exploration Rovers, em 2003. O objetivo é enviarem imagens, analisarem rochas e crateras, e, procurarem sinais de existência de água no planeta vermelho (BAJRACHARYA, 2008). Vários outros locais podem utilizar os robôs móveis para exploração como tubulações de óleo, realizando inspeções, nos túneis das pirâmides egípcias, procurando câmaras ainda não mapeadas e sobrevoando áreas para inspeções de oleodutos e linhas de transmissão (BUENO et al., 2002).

Figura 4 - Sojourner

Fonte: http://mars.nasa.gov/mer/home/

Figura 5 - Opportunity

Fonte: http://mars.nasa.gov/mer/home/

• Nos serviços: um robô de serviço opera semi ou totalmente autônoma para realizar serviços úteis para os seres humanos ou equipamentos. Exemplo de um robô para a agricultura, que agiliza o processo de colheita.

Figura 6 – Robô para a colheita de uvas

Fonte: www.theguardian.com/

• Na segurança: Diversas unidades de policias do mundo utilizam os robôs para auxiliarem no monitoramento de grandes áreas de fronteiras, ambiental e para o desarme de bombas. Como o CUTLASS que é equipado com braço manipulador que tem três dedos tipo pinça, e, possui nove graus de liberdade para maior movimentação e agilidade dentro de espaços limitados[2].

Figura 7 - CUTLASS

Fonte: http://www.army-guide.com/

• Nas Sociais e educação: São robôs que interagem com humanos e sua principal utilidade é para o entretenimento. Por exemplo, o Amazon Echo, que conectado a internet ele responderá dúvidas, atenderá comandos e até informará o usuário sobre notícias do dia ou receitas para o jantar. Outro exemplo é o Nao, desenvolvido pela Aldebaran Robotics, que é

[2] Disponível em: http://www.army-guide.com/eng/product.php?prodID=5022

um humanoide de 58 centímetros de altura que dança, joga futebol, reconhece vozes, conversa em 19 idiomas e detecta expressões. Ele é bastante usado para fins educacionais e para campeonatos de robótica[3].

Figura 8 - Echo

Fonte: http://www.amazon.com/oc/echo/

Figura 9 - NAO

Fonte: https://www.aldebaran.com/en/humanoid-robot/nao-robot

Um dos principais motivos da evolução da robótica foi devido a suas aplicações em manufatura, tendo como foco a automatização das linhas de montagem. Os robôs podem fazer os trabalhos manuais mais rápidos, mais precisos e mais baratos que os seres humanos. Assim, a utilização da robótica aumentava a produtividade e tornava mais flexível, uma vez que um braço de robô foi programado, ele deve ser capaz de operar durante semanas ou meses com pouca manutenção (KANNIAH et. al., 2013).

Apesar do seu sucesso em realizar atividades de manufatura, os robôs de base fixa sofrem uma desvantagem fundamental: a falta de mobilidade. Um manipulador fixo tem a sua limitação em relação ao seu espaço de trabalho pela sua falta de mobilidade. Ao contrário, um robô móvel seria capaz de viajar por toda a fábrica, com flexibilidade para desenvolver outras funções em uma variedades de ambientes. (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011)

[3] Disponível em: https://www.aldebaran.com/en/humanoid-robot/nao-robot e em: http://www.amazon.com/oc/echo.

1.2 Justificativa

Embora os robôs que vemos em filmes de ficção científica parecem navegar com precisão sem esforço, a tarefa da navegação em robôs móveis é um problema de difícil investigação, sendo que varias universidades e empresas no mundo tentam resolver esse problema da melhor forma, na busca, cada vez mais, de soluções de baixo custo.

Em um ambiente conhecido e estático é fácil planejar as ações do robô. Por outro lado, em um ambiente, parcialmente ou completamente desconhecido, é necessário um nível mais avançado de inteligência. (KHATIB, 1986).

Para manter o robô seguro durante a sua navegação, é importante o desenvolvimento de um algoritmo inteligente que evite danos ao robô. Tal algoritmo, para evitar obstáculos, tem como entrar nas informações sensoriais, assim, sendo capaz de tomar uma decisão rápida ao encontrar um obstáculo e gerar uma trajetória mais curta e suave, dando autonomia ao robô para que possa chegar ao seu destino sem maiores problemas.

1.3 Objetivos

1.3.1 Objetivo geral

- Construir um robô móvel de baixo custo para testes de navegação em ambientes internos, local, propondo um método de desvio de obstáculos que se encaixe nos requisitos de baixo custo.

1.3.2 Objetivos específicos

- Construir um robô móvel de baixo custo.
- Construir um cinturão de sonares para identificação do obstáculo.
- Desenvolver um algoritmo para solucionar os problemas dos sonares.
- Fazer a análise da viabilidade da utilização do algoritmo bubble rebound para o desvio de obstáculos.

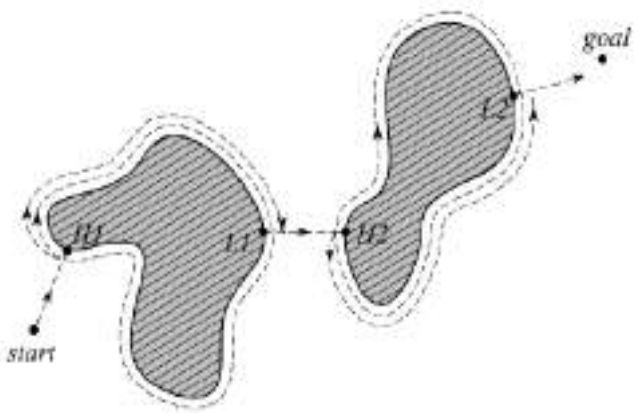
2 TRABALHOS RELACIONADOS

2.1 Algoritmos BUGs

2.1.1 BUG 1

O algoritmo Bug 1 é um dos o mais conhecidos algoritmos desvio de obstáculos. De acordo com Lumelsky & Skewis (1990a) quando um obstáculo é detectado, o robô circula completamente o objeto até atingir o ponto de partida. Durante esse movimento em torno do obstáculo, o robô calcula um ponto com a melhor saída, tendo uma distância mínima do contorno do objeto até o destino e, por fim, gera um novo caminho de ponto de partida para seu destino. Após a sua volta completa pelo obstáculo, ele reinicia o seu movimento até chegar ao ponto de saída com a menor distância do objetivo, afastando-se das proximidades do obstáculo.

Figura 10 – Bug 1



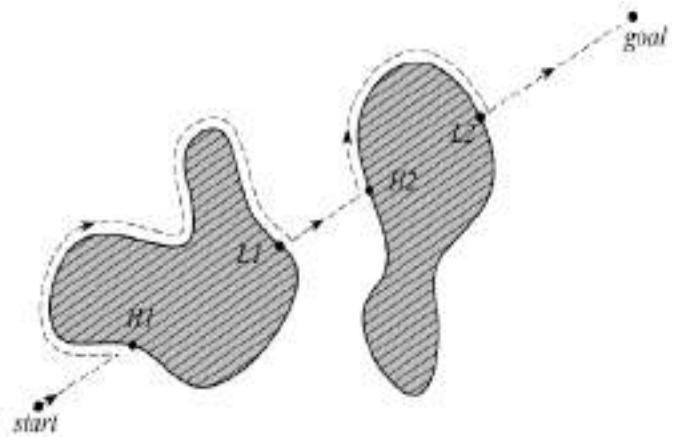
Fonte: Siegwart, Nourbakhsh & Scaramuzza (2011)

2.1.2 BUG 2

Em Lumelsky & Stepanov (1990b) é proposto uma melhoria do algoritmo bug 1. No bug 2, o robô traça uma caminho mínimo da origem até o destino, salvando essa inclinação, e começa a segui-lo até que ele seja interrompido por um obstáculo. Quando o robô encontrar um obstáculo ele faz o contorno do obstáculo, calculando a sua inclinação a cada nova posição até encontrar a sua inclinação inicial novamente. Quando ele encontrar a sua

inclinação inicial, o robô segue o seu caminho novamente até o destino, saindo das proximidades do obstáculo. (YUFKA & PARLAKTUNA, 2009).

Figura 11 – Bug 2



Fonte: Siegwart, Nourbakhsh & Scaramuzza (2011)

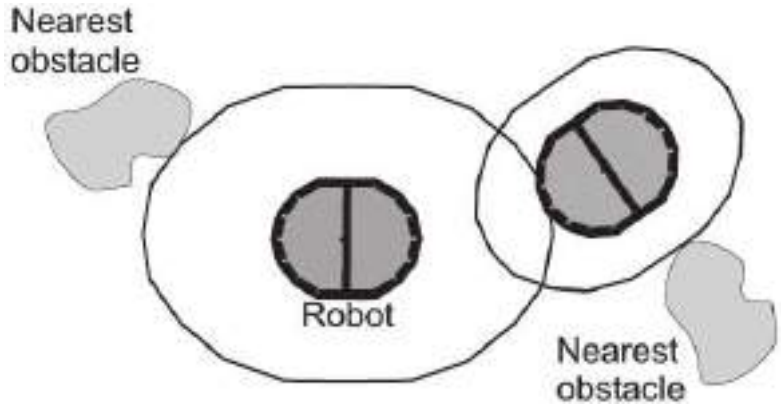
Apesar da simplicidade desses algoritmos, existem algumas desvantagens citadas por Susnea et. al. (2010) e Zhu et. al. (2010), como:

- O algoritmo considera apenas as leituras mais recentes dos sensores, com isso, o ruído afeta, seriamente, o desempenho geral do robô;
- Apesar de o robô chegar ao destino, os seus trajetos são muito lentos;
- Essas estratégias não consideram outros obstáculos durante o processo de detecção de bordas;
- Assume o robô como um ponto no mapa sem considerar suas dimensões.

2.2 Bubble band

Essa técnica foi proposta, inicialmente, por Quinlan & Khatib (1993). Depois foi estendida para veículos não-holonômicos (KHATIB et. al., 1997). Nele é defina uma “bolha” contendo o espaço máximo livre ao redor do robô, que pode ser percorrida em qualquer direção sem colisão. A forma e o tamanho da bolha são construídos a partir de um modelo simplificado da geometria do robô, baseado nas informações sensoriais, como na figura 12.

Figura 12 – Bubble band

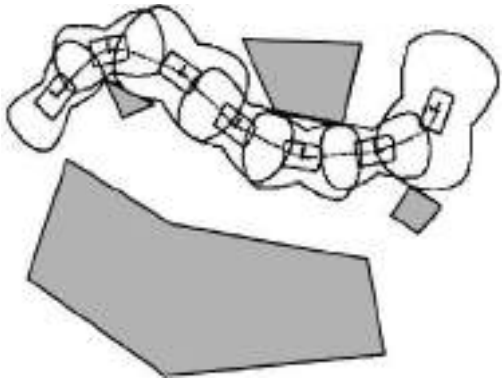


Fonte: Susnea et. al. (2010)

A técnica *bubble band* mescla o planejamento de trajetória e o desvio de obstáculos. O conjunto de bolha representa uma determinada posição com base na distância máxima, aferida pelos sensores entre o robô e os obstáculos que cercam em todas as direções. No planejamento de trajetória, o robô segue um caminho previamente calculado, sem permitir uma distensão excessiva entre bolhas das posições calculadas. Assim, o robô começa a desenhar trajetórias mais suaves. (KHATIB, 1996), (SUSNEA et. al. 2010), (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011).

À medida que o robô encontrar os valores imprevistos dos sensores, o modelo de *bubble band* é usado para o desvio de obstáculos, que leva em consideração seu caminho, inicialmente, calculado de uma forma que minimiza a tensão do conjunto de bolhas. Logo, essa técnica requer um conhecimento prévio do ambiente, como mapa global e também planejador de caminho global. Com isso, torna a sua utilização mais complexa. A figura 13 mostra a trajetória gerada pelo método. (KHATIB, 1996), (SUSNEA, et. al. 2010), (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011).

Figura 13 – Trajeto do bubble band

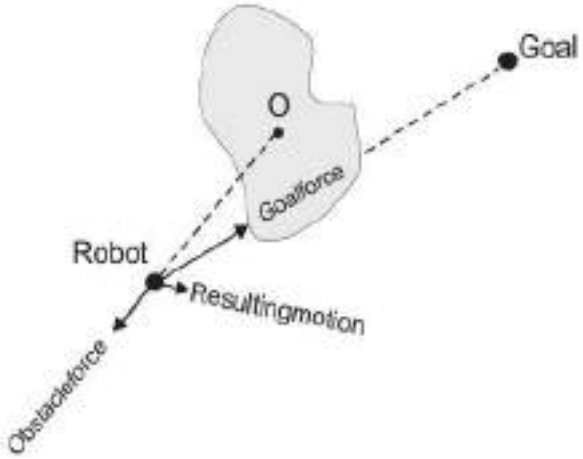


Fonte: Khatib et. al. (1997)

2.3 Campos potenciais

Esse algoritmo se baseia no princípio de que no campo potencial, proposto inicialmente por (KHATIB, 1986). Considerado uma das primeiras abordagens para o problema de planejamento de rota. De acordo com Oroko & Nyakoe (2014) a técnica pode ser aplicada na versão on-line, que tem o comportamento de desvio de obstáculos.

Figura 14 – Campos potenciais



Fonte: Susnea et. al. (2010)

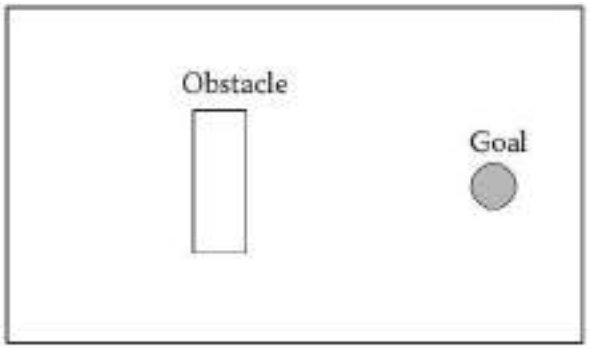
O robô e os obstáculos atuam como uma carga positiva, e, o objetivo atua no mapa como uma carga negativa. Assim, os obstáculos repelem o robô através da geração de força repulsiva e o objetivo atrai o robô por ser de força de atração. Com isso, a força resultante sobre o robô é a soma vetorial de toda força repulsiva e atraente.

A magnitude da força é calculada pela distância, ou seja, quando o robô está mais próximo do obstáculo, a força de repulsão será maior. Porém, quando o robô está longe do objetivo a sua força de atração é maior, logo a sua velocidade será alta. Quando estiver perto, sua força de atração e a sua velocidade será menor, dando suavidade ao planejador.

Apesar de suavizar o movimento, mantendo uma boa qualidade na rota, geralmente, passa a uma distância segura dos obstáculos. Essa técnica possui algumas deficiências, citado por Choset et. al. (2005). O algoritmo gera mínimo local, levando ao robô uma saída sem fim, como um obstáculo em forma de U. Com isso, o planejador pode não ser completo, tem dificuldades em passagem estreitas, e, é difícil de usar em tempo real.

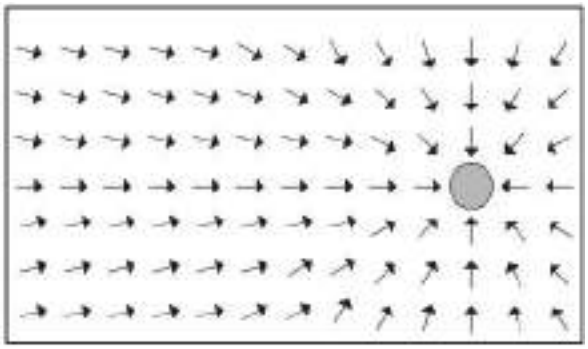
As figuras abaixo demonstram um possível cenário de atuação das forças. Na figura 15a mostra um obstáculo e o objetivo sem atuação das forças. A figura 15b demonstra a força de atração gerada pelo objetivo que afeta todo o mapa.

Figura 15a – Campos potenciais, sem força de atuação



Fonte: Kanniah Ercan & Calderon (2013)

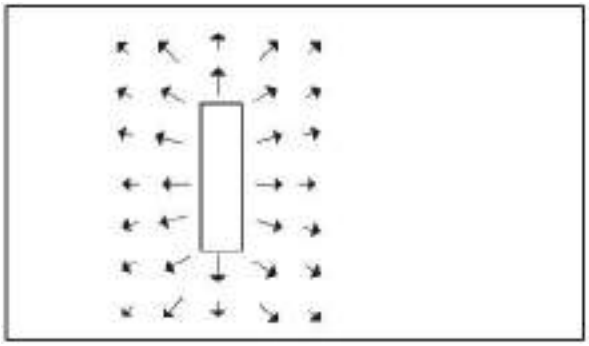
Figura 15b – Campos potenciais, objetivo



Fonte: Kanniah Ercan & Calderon (2013)

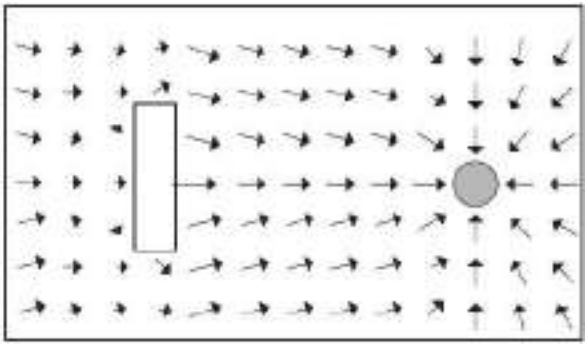
Na figura 15c o obstáculo exerce uma força de repulsão no mapa, diferente da força do objetivo, a força de obstáculo não afeta todo o mapa. E por fim na figura 15d mostra a força resultante do mapa.

Figura 15c - Campos potenciais, obstáculo



Fonte: Kanniah Ercan & Calderon (2013)

Figura 15d - Campos potenciais, força resultante



Fonte: Kanniah Ercan & Calderon (2013)

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Robótica Móvel

No contexto da contínua busca da humanidade para construir máquinas mais capazes, levando até mesmo a superação das capacidades humanas, para o desenvolvimento desses sistemas, a capacidade de navegação em ambiente é um obstáculo fundamental.

O que define robótica móvel, para além de outras áreas de investigação, tais como robótica convencional de manipulador, inteligência artificial e visão por computador, é a ênfase em problemas relacionados com a compreensão do espaço em grande escala, isto é, regiões do espaço, substancialmente maior, do que aqueles que podem ser observados, necessitando a navegação em ambientes desconhecidos. Para se comportar de forma inteligente em um ambiente de grande escala, o robô móvel precisa lidar com a aquisição incremental de conhecimento, estimar a sua localização, capacidade de reconhecer objetos, lugares importantes ou familiares, e, resposta em tempo real, mas também exige que todas estas funcionalidades sejam executadas em conjunto. (DUDEK & JENKIN, 2000), (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011).

Segundo Dudek & Jenkin (2000) O estudo de robôs móveis é uma área de pesquisa interdisciplinar que envolve intrinsecamente:

Mecânica: design do veículo e em especial os mecanismos de locomoção.

Ciência da computação: representações de sensoriamento e de desenvolvimento de algoritmos.

Engenharia elétrica: integração de sistemas, sensores e comunicações.

A psicologia cognitiva, percepção e neurociência: insights sobre como os organismos biológicos resolvem problemas semelhantes.

Mecatrônica: a combinação de engenharia mecânica com ciência da computação, engenharia da computação e / ou engenharia elétrica.

Com isso, robótica móvel é uma área de pesquisa que lida com o controle de veículos autônomos e semiautônomos. De acordo com Dudek & Jenkin (2000) os robôs moveis podem ser classificados em quatro tipos, levando em consideração o ambiente em que eles atuam, sendo estes:

Terrestres: robôs terrestres são aqueles que viajam ao longo do chão. A sua locomoção se dar pelo atrito do solo com o atuador. Os tipos mais comuns de robôs móveis terrestres são

o de rodas, mas existem robôs que podem caminhar, escalar, rolar, usar esteiras, ou deslizar sobre a superfície para se mover.

Aquáticos: robôs aquáticos operam na água, seja na superfície ou debaixo dela. A maioria dos veículos aquáticos utiliza jatos de água ou hélices para fazer a movimento para a locomoção. Os robôs aquáticos são muito utilizados para exploração, já que a maior parte da superfície da Terra é coberta pela água, mas grande parte do oceano não é facilmente acessível para os seres humanos.

Aéreos: os robôs aéreos operam no ar, podendo ser chamado de veículos aéreos não-tripulado (VANT). Atualmente os robôs aéreos tem se popularizado mais do que os terrestre pela sua diversidade de aplicações, tanto para os hobbistas quanto para o uso profissional. Exemplos: Drones, helicópteros robóticas, aeronaves de asa fixa, e dirigíveis têm sido desenvolvidos.

Espaciais: Alguns robôs são projetados para operar na microgravidade do espaço exterior, sendo que a sua principal aplicação é a manutenção da estação espacial.

A robótica móvel é um campo atuação muito extenso, por isso o foco desse trabalho são os robôs móveis terrestres para ambientes internos.

3.2 Componentes físicos do robô

Em Murphy (2000) é citado as três funções primitivas de um robô, que é sentir, planejar, agir. Para isso os robôs precisam de um mecanismo físico para interagir com o ambiente, sendo que os sensores são responsáveis pela captação das informações, que é o sentir. O controlador é o responsável pelo planejamento das ações e, como resposta, os atuadores agem no ambiente. Abaixo é descrito os três principais mecanismos físicos dos robôs.

3.2.1 Sensores

Os robôs móveis precisam de informação sobre o mundo para que eles possam relacionar-se com o meio ambiente, assim como os animais. Para isso, eles contam com dispositivos sensores que transformam os estímulos do mundo em sinal elétrico. Esses sinais são dados elétricos que representam estado do mundo lido pelo sensor, e deve ser interpretado pelo robô para atingir seus objetivos. Existe uma variedade de sensores utilizados para esse fim.

De acordo com Romero et. al. (2014) os sensores podem ser classificados por alguns recursos como o método de medição, aplicação, e de sinal. Sobre o método de medição, os sensores são classificados em sensores ativos e passivos. Os sensores ativos aplicam um sinal conhecido para o ambiente, como uma onda sonora, e verificam o efeito deste sinal no ambiente. Enquanto que os sensores passivos não emitem sinal para medir o mundo, mas eles são capazes de fazer leituras de fenômenos emitidos naturalmente pelo mundo. Por exemplo, uma câmera que captar a luz.

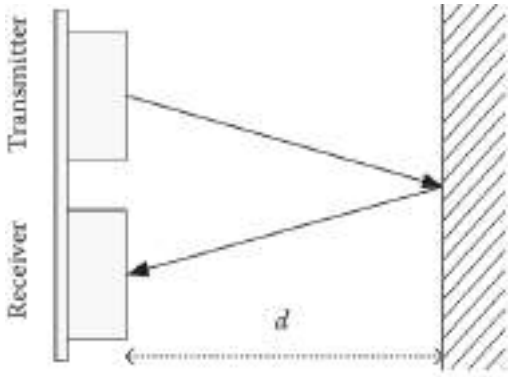
Dependendo da sua utilização, os sensores podem ser proprioceptivos ou exteroceptivos. Sensores proprioceptivos estão relacionados com os elementos internos do robô, para que eles monitorem o estado interno como mecanismos, dispositivos e motores. Já os sensores exteroceptivos recolhem informações, a partir do ambiente onde o robô se encontra, geralmente, estão relacionados com a navegação robô e aplicação.

Sensores também podem ser classificados de acordo com o sinal de saída elétrica, sendo divididos em sensores digitais e analógicos. Em geral, os dados sensoriais, geralmente, são imprecisos, o que aumenta a dificuldade de utilizar as informações fornecidas por eles.

3.2.1.1 Sensor de ultrassom ou sonar

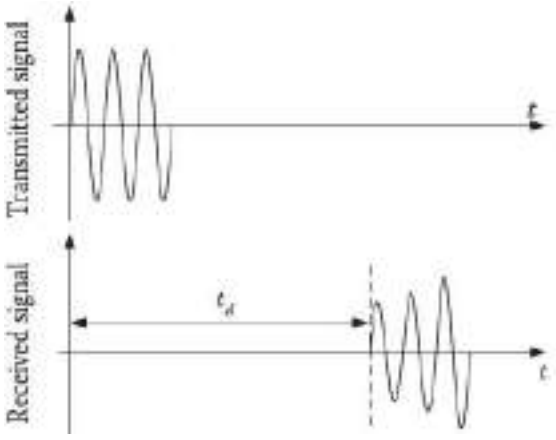
Sensores de ultrassom transmitem um sinal em forma de onda ultrassônica e recebem o sinal refletido. Ele é utilizado para medir distâncias, calculado pelo tempo necessário para que um sinal reflita no objeto e retorne para o sensor. As frequências utilizadas variam de 40 e 180 kHz. O ultrassom é composto de um par de alto-falantes e microfone, um para produzir o som e outro receber o eco (ROMERO et. al., 2014).

Figura 16 – Reflexão da onda



Fonte: Kanniah et. al. (2013).

Figura 17 – Ping do ultrassom



Fonte: Kanniah et. al. (2013).

Um sinal ultrassônico curto é gerado como se mostra na Figura 17, e o temporizador é ativado. O receptor capta o som ecoando e para o cronômetro. Este período de tempo, conhecido como o tempo de voo (TOF), é dada como Td. (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011).

Assim, a distância *d* é calculada pela formula:

Figura 18 – Fórmula para medir a distância do ultrassom

$$d = \frac{c_{\text{air}} \times t_d}{2}$$

Fonte: SIEGWART, NOURBAKHSH & SCARAMUZZA (2011).

Onde, *Cair* é a velocidade do som no ambiente em que atua. No ar é cerca de 330 m/s e na água é 1500 m/s.

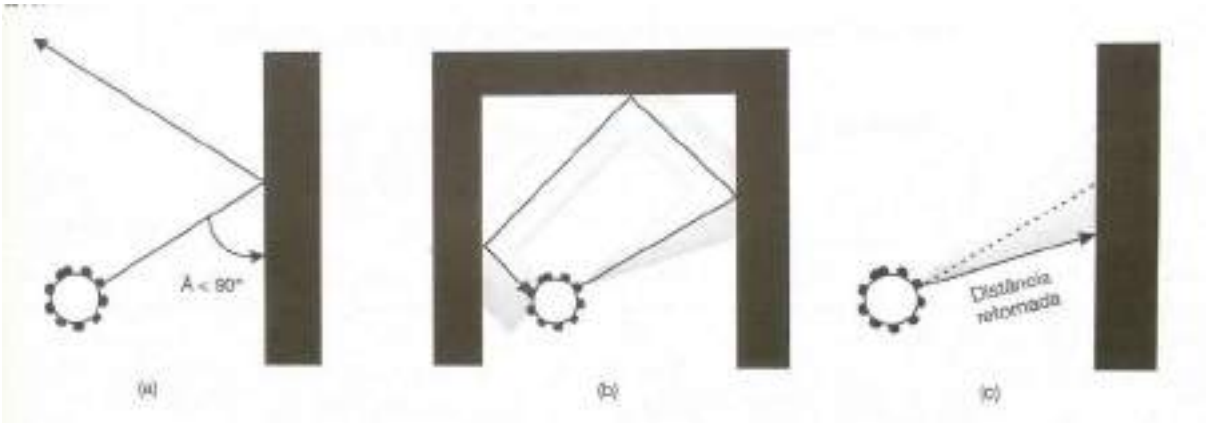
A vantagem do ultrassom é o seu baixo custo e baixa complexidade na sua utilização. Os sensores de ultrassom têm algumas limitações que devem ser levados em consideração durante a sua implementação. Segundo Siegwart, Nourbakhsh & Scaramuzza (2011) esses problemas são a reflexão especular, cross-talk e foreshortening.

Sensores de ultrassom dependem de reflexão, e, por isso, são suscetíveis a *reflexão especular*. A Reflexão especular é a forma de onda quando atinge uma superfície em ângulo agudo e a onda salta para longe do receptor do sonar. Logo, o sensor não consegue identificar o obstáculo dessa situação. A Reflexão especular pode levar a outro problema chamado de *cross-talk*. Esse problema ocorre quando o sensor recebe um sinal originado de um sensor vizinho. O ultrassom gera uma onda que normalmente se espalha em diversos objetos, podendo retornar ao outro sensor que não emitiu o sinal original. Como o ultrassom não consegue identificar a origem do sinal, o sensor identifica o sinal refletido do seu sensor vizinho, levando, assim, a medições erradas.

Outro problema dos sensores do ultrassom é o foreshortening, pois o sonar gera uma onda em forma de cone com abertura de 30º em relação ao eixo do sensor. Quando a superfície do obstáculo não é perpendicular ao sensor, um lado do cone irá atingir o ponto de menor distância, não levando em conta a referência central do sensor. Com isso, traz uma leitura de distância menor do que a real.

A figura 19 ilustra os três problemas:

Figura 19 - (a) reflexão especular, (b) cross-talk, foreshortening, (c) distancia retomada.



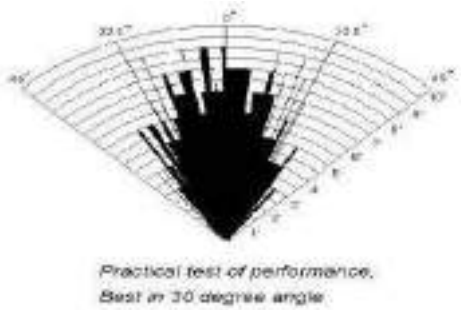
Fonte: (ROMERO et. al. 2014).

O sensor de ultrassom utilizado na implementação do robô HC-SR04 é de baixo custo para determinar a distância de um objeto, a partir de 2 centímetros a 400 centímetros. Tem as seguintes características:

- Tensão: 5V DC
- Corrente: 15mA
- Ângulo a curta distância: <15 °
- Ângulo a longa distância: +/- 30 graus
- Variando Distância: 2cm - 400 centímetros / 1 "- 13 pés
- Resolução: 0,3 centímetros
- Largura de pulso de entrada do Trigger: 10uS
- Dimensão: 45 milímetros x 20 milímetros x 15 milímetros

Figura 20 - Ultrassom

Figura 21 – Ângulo do ultrassom



Fonte: http://elecfreaks.com/estore/download/EF03085-HC SR04_Ultrasonic_Module_User_Guide.pdf

3.2.1.2 Unidade de medição inercial (IMU)

Uma unidade de medição inercial (IMU) é um dispositivo que utiliza giroscópios e acelerômetros para estimar a posição relativa, velocidade e aceleração de um veículo em movimento. Um IMU também é conhecido como um sistema de navegação inercial (INS), e tornou-se um componente de navegação comum de aeronaves e navios. Um IMU estima a pose de seis graus de liberdade (DOF) do veículo: posição (x, y, z) e orientação (roll, pitch, yaw). Alguns IMUs podem ter mais sensores integrados, aumentando, assim, a quantidade de graus de liberdade. (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011)

Os IMUs são extremamente sensíveis a erros de medição em ambos os sensores, precisando de alguma técnica para tornar esse erro aceitável. Grewal et. al. (1991) utilizam o filtro de Kalman para a calibração e alinhamento dos sistemas de navegação por inércia. Já Euston et. al. (2008) propõem a filtro não-linear Complementary para medições de unidade de medida inercial de baixo custo.

O IMU proposto nesse trabalho é o GY-80 que na mesma placa possui o acelerômetro, giroscópio, magnetômetro e barômetro.

Características do IMU:

- Protocolo de comunicação: I2C
- Chip Acelerômetro: ADXL345
- Endereço I2C: 0x53
- Faixa do Acelerômetro: ±2, ±4, ±8, ±16g
- Chip Giroscópio: L3G4200D
- Endereço I2C: 0x69
- Faixa do Giroscópio: ±250, 500, 2000°/s
- Chip Magnetômetro: HMC5883L
- Endereço I2C: 0x1E
- Chip Barômetro: BMP085
- Endereço I2C: 0x77
- Tensão de operação: 3,3-5V
- Peso: 5g
- Dimensões: 25,8 x 16,8mm

Figura 22 - IMU

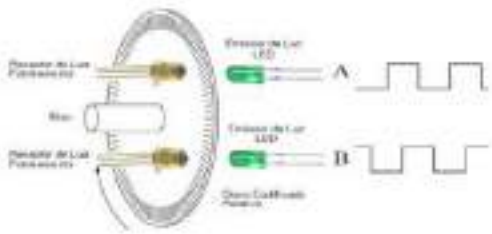
Fonte:http://www.dx.com/p/gy-80-bmp085-9-axis-magnetic-acceleration-gyroscope-module-for-arduino-145912

3.2.1.3 Encoder óptico

Um encoder óptico é um interruptor mecânico luz que produz certo número de pulsos de seno ou de onda quadrada para cada revolução do eixo. É formado por um fotoemissor, fotorreceptor e um disco vazado, anexado ao eixo do motor para gerar as interrupções. Quando o motor move, o disco ligado ao eixo gira, criando várias interrupções, que concebe uma onda senoidal, que é transformada em uma onda quadrada discreta de 0s e 1s, e, é representado por estados de luz e escuro, gerado pela interrupção do disco. (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011).

Os encoders mais utilizados na robótica são o que gera a onda quadrática. Neste caso, é usado um par de fotorreceptores, que são posicionados em um ângulo de 90º entre si. O sinal recebido por cada fotoreceptor é chamado de canal A e B, com a variação desses canais pode estimar a velocidade de rotação do motor e a direção. Com isso por meio da odometria o robô é capaz de prever a sua localização. (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011).

Figura 23 – Esquema do encoder



Fonte: www.ebah.com.br

Os encoders são utilizados para controlar a posição e a velocidade das rodas dos robôs, fornece uma estimativa da distância relativa navegada. Essa técnica de estimar a distância navegada para determinar sua posição no ambiente é chamada de odometria. Apesar de ela ser bastante utilizada na robótica, ela, por si só não é precisa, pois em longos espaços de deslocamento vários erros podem atrapalhar a medição, como à imprecisão dos codificadores, a derrapagem, irregularidade da superfície, pequena diferença entre os tamanhos de rodas, etc.

O encoder proposto nesse trabalho é encontrado no motor de scanner de impressora HP, ele pode ser facilmente encontrado em sucatas de multifuncionais velhas. A escolha desse encoder é pelo fato da alta precisão, do motor ser do mesmo modelo do Kit robótico, utilizado nesse trabalho e de fácil substituição. Na figura 24 mostra uma foto do encoder.

Figura 24 – Encoder da impressora HP com resolução de 32 pulsos

3.2.2 Atuadores

Os robôs móveis, além de perceberem o meio em que estão inseridos, devem atuar com esse ambiente, como descolar, interagir e manipular, de algum modo, com os elementos nesse ambiente. Para executar essa tarefa os robôs precisam de algum mecanismo de atuação, como os motores, para serem capazes de produzir as ações. Existem vários tipos de atuadores utilizados em robótica, normalmente eles são classificados em elétricos, hidráulicos e pneumáticos. (MURPHY, 2000).

O atuador utilizado na implementação é o motor que encontrar no Kit Magician, que tem a seguinte descrição: O Magician chassis[4] é uma plataforma robô móvel com cinemática diferencial. Ele possui dois motores com caixa de redução, rodas de 65 milímetros e uma roda boba traseira. As placas de chassis são feitas de acrílico com uma grande variedade de furos de montagem para sensores, controladores, alimentação, etc.

A escolha do kit robótico (Figura 22a e 22b) foi pela facilidade de montagem e baixo custo comparado com os outros kits no mercado.

Características do kit robótico:

- Dimensões: 110 x 174 milímetros
- Máxima tensão do motor: 6VDC
- Sem atrito: 90 ± 10rpm
- Corrente sem atrito: 190mA (max.250mA)
- Torque: 800gf.cm
- Redução: 48:1
- Pico de tensão: ~ 1A
- Diâmetro da roda: 65 milímetros e 30 mm de largura

Figura 25a – Kit robótico

Figura 25b – Motor elétrico do Kit robótico

Fonte:https://www.sparkfun.com/products/retired/10825

[4] Descrição do Magician chassis https://www.sparkfun.com/products/retired/10825.

3.2.3 Controlador

Para que o robô seja inteligente ele precisa de um cérebro que coordene as ações dos sensores e atuadores, e, assim, realizar uma tarefa. O controlador do robô é responsável pela tomada de decisão do robô, controlando do nível mais baixo, como ligar um motor, até o sistema de navegação completo do robô. O controlador utilizado no projeto é o microcontrolador arduino mega.

O Arduino é uma plataforma open-source de computação física, baseado no microcontrolador ATmega2560, com um ambiente de desenvolvimento que implementa a linguagem Processing/Wiring. O Arduino pode ser usado para desenvolver objetos interativos autônomos[5].

Características do arduino:

- Microcontrolador: ATmega2560 da ATmel [6]
- Clock: 16Mhz
- Tensão de entrada: 5-12V
- Pinos digitais: 54 Digital I/O. Dentre eles, 14 de saídas PWM
- Pinos analógicos: 16
- Memória flash: 256k
- Interrupções externas: 6

Figura 26 – Arduino Mega

Fonte: www.arduino.cc

[5] Disponível em: www.arduino.cc.
[6] Disponível em: http://www.atmel.com/pt/br/devices/ATMEGA2560.aspx

3.2.3.1 Ponte h

Para controlar o motor e isolar o microcontrolador de descargas elétricas, que o motor pode gerar, foi usado o L293D, que é um circuito integrado, projetado para o controle de motor, que conduz cargas indutivas, tais como solenóides, relés, motores DC, motores de passo bipolares. O CI é composto de drivers push-pull capazes de transportar correntes de saída para 600mA por canal. Cada canal é controlado por uma entrada lógica TTL-compatível e cada par de condutores (uma ponte completa) está equipada com uma entrada de inibição, que desliga todos os quatro transistores. Uma entrada de alimentação separada é fornecida para a lógica, de modo que ele pode ser executado fora de uma tensão mais baixa, protegendo o microcontrolador de danos que possa ocorrer com a geração de tensão dos motores.

Características da Ponte h:

- Tensão: 4,5 a 36V
- Corrente: 600mA por canal
- Pico: 1,2A por canal
- Desligamento automático com o super-aquecimento.

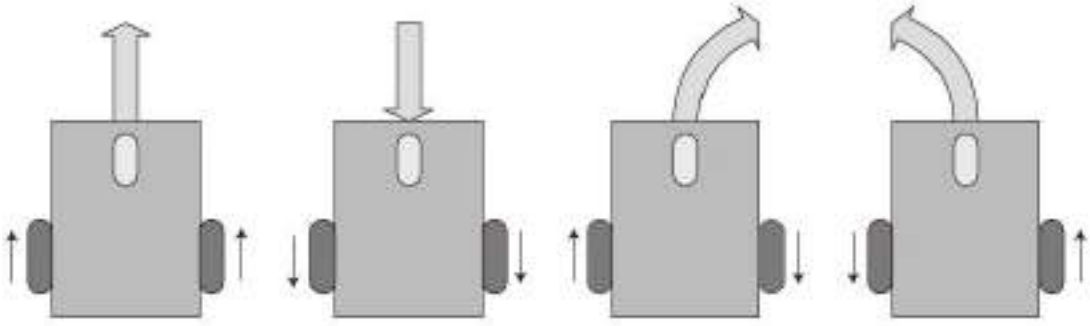
Figura 27 – Ponte H L293D

Fonte: http://www.ti.com/lit/ds/symlink/l293.pdf

3.3 Modelo cinemático diferencial

Esse tipo de sistema de tração é composto de duas rodas com motor independente. Ele pode ter uma ou duas rodas bobas para suportar o robô. Ao controlar a velocidade de cada roda, os movimentos do robô podem ser controlados. A Figura X ilustra o movimento do robô com diferentes velocidades das rodas. Um grande desafio em sistemas de acionamento diferenciais é que a velocidade da roda tem de ser controlada adequadamente e com precisão para atingir movimentos do robô desejados (SIEGWART, NOURBAKHSH & SCARAMUZZA, 2011).

Figura 28 – Dinâmica do veículo diferencial



Fonte: Kanniah et. al. (2013).

Para o robô se movimentar para frente ou para trás, ambas as rodas tem que ser acionadas na mesma velocidade. Se uma roda é ligeiramente mais rápida do que a outra, a roda mais rápida vai ultrapassar a mais lenta e fazer um movimento de giro. Quando ambas as rodas são movidas em direções opostas na mesma velocidade o robô faz um volta em seu próprio eixo, esse eixo se localiza no meio das duas rodas, o que é conhecido também como o centro de rotação. (KANNIAH et. al., 2013).

São apresentadas, abaixo, as equações da cinemática da posição do robô móvel descrito por Kanniah et. al. (2013). A velocidade do robô V é definida como a média da velocidade das rodas esquerda e direita.

Figura 29: Equações cinemáticas

$$V = \frac{V_l + V_r}{2}$$

E a velocidade em relação ao eixo X e Y será:

$$\dot{x} = V\cos\mathbf{q} \qquad \dot{y} = V\sin\mathbf{q}$$

O q velocidade angular pode ser dada por:

$$\dot{\mathbf{q}} = \frac{V_l - V_r}{d}$$

Em que d é a distância entre as rodas do robô.

Fonte:

No Modelo cinemático diferencial o movimento é uma função de duas variáveis, a velocidade da esquerda e direita. Ao controlar essas duas velocidades é possível controlar o movimento de qualquer robô, usando esse sistema de acionamento.

4 ANÁLISE E DISCUSSÃO DOS ALGORITMOS DE DESVIO DE OBSTÁCULO

Esse trabalho propõe o uso de uma solução de baixo custo para a navegação local do robô em ambientes dinâmicos, onde o robô desconhece o ambiente e deve sair do ponto de origem até o destino em segurança. Um dos requisitos desse projeto é que o algoritmo seja simples para executar em microcontroladores de baixo poder computacional, que tenha um resultado bom o suficiente para ser usado em aplicações reais em robôs móveis em ambientes indoor.

Na literatura existem vários métodos de desvio de obstáculos. Siegwart, Nourbakhsh & Scaramuzza (2011) fazem uma análise dos principais métodos e uma comparação entre eles. A tabela de comparação se encontra no Anexo 1.

Levando em consideração a tabela, poucos métodos se adéquam ao principal requisito desse trabalho, o baixo custo. Nesse primeiro requisito só os algoritmos da família Bug e dos Bubble banda se adéquam. Todos os outros necessitam de um alto poder de processamento, comparado aos 16Mhz disponível no Arduino ou sensores de alto custo como o LIDAR.

Em relação ao segundo requisito, a eficiência, os algoritmos da família Bug só têm um bom desempenho em alguns casos, tendo inconsistência em sua execução. Os da família Bubble band requer um conhecimento prévio do ambiente, como um mapa global e também um planejador de caminho global, para que seja calculada, inicialmente, uma trajetória, para depois ser executada pelo robô. Com isso, torna-se inviável a sua implementação para esse projeto. O de método de campos potenciais gera mínimo local, tem dificuldades em passagem estreitas e é difícil de usar em tempo real.

4.1 Bubble Rebound

Os métodos, anteriormente descritos, não cumprem os requisitos desse projeto, sendo que o mais próximo é o trabalho proposto por Susnea et. al. (2010), que apresentam um novo algoritmo reativo para o desvio de obstáculos, contando com sonares de baixo custo ou sensores de infravermelho. Nas próximas seções é descrito o algoritmo Bubble Rebound.

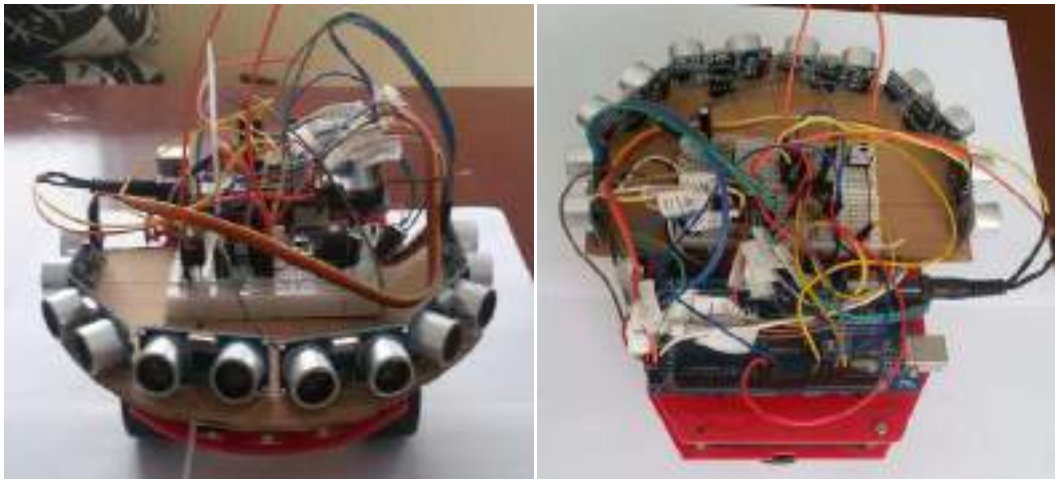
4.1.1 Detecção de obstáculos

Primeiramente, para a utilização do algoritmo o robô precisa está na configuração correta, de acordo com a solução proposta por Susnea et. al. Na frente do robô é feito um anel

de sonares equidistantes, cobrindo um ângulo de 180º, tal como representado nas figuras 30a e 30b.

Figura 30a – Frente do robô

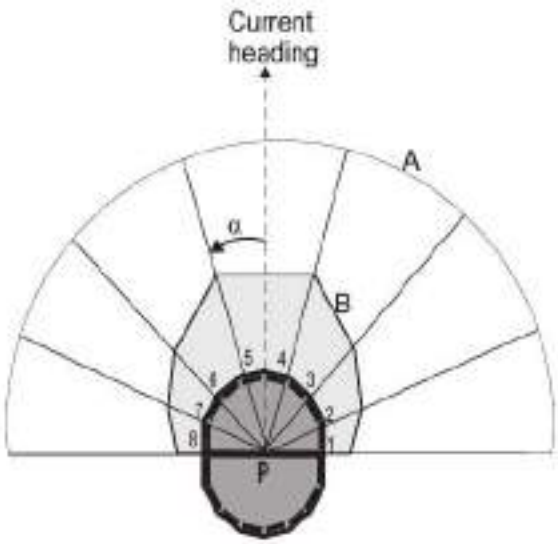
Figura 30b – Fundo do robô



Fonte: Fotos da construção do robô e os seus módulos integrados.

Após configuração dos sensores é criado e definido a bolha representada na figura 31 como o Ponto B. Essa bolha é o limite de segurança do robô. Se o obstáculo estiver dentro dessa bolha o robô executa o desvio e se afasta do obstáculo para evitar a colisão.

Figura 31 – Bolha de sensibilidade



Fonte: Susnea et. al (2010)

O pseudocódigo para a definição da sensibilidade da bolha é:

unsigned int sonar_readings[N];

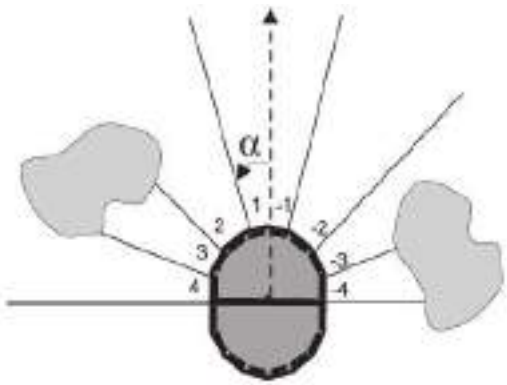
unsigned int bubble_boundary[N];

```
bubble_boundary[i]=Ki*V*delta_t;
int check_for_obstacles(void){
for (i = 0; i< N; i++){
        if ( sonar_readings[i] <= bubble_boundary[i]){
                return(1);
        }else{
                return(0);
        }
}
```

Onde V é a velocidade de translação do robô, Delta_T é o intervalo de tempo entre as avaliações sucessivas de dados do sensor, e, Ki são constantes de escala, utilizados para ajuste. A bolha não pode ultrapassar os limites dos sensores. O formato da bolha de sensibilidade é próxima a forma geométrica do robô, pois as leituras dos sensores representam a distância entre a posição real do sensor, que está no limite do veículo, e do obstáculo.

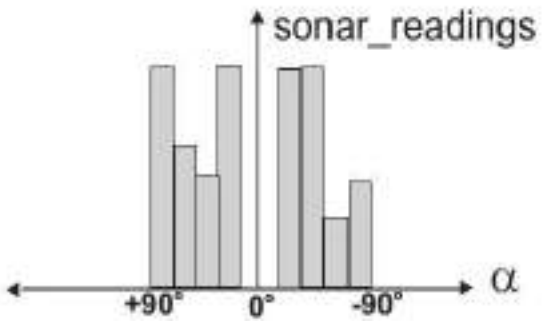
A figura 32a mostra a representação do índice de cada sonar junto com os obstáculos e na figura 32b mostra o diagrama polar do obstáculo nessa posição.

Figura 32a – Representação dos obstáculos



Fonte: Susnea et. al (2010)

Figura 32b – Diagrama polar



Fonte: Susnea et. al (2010)

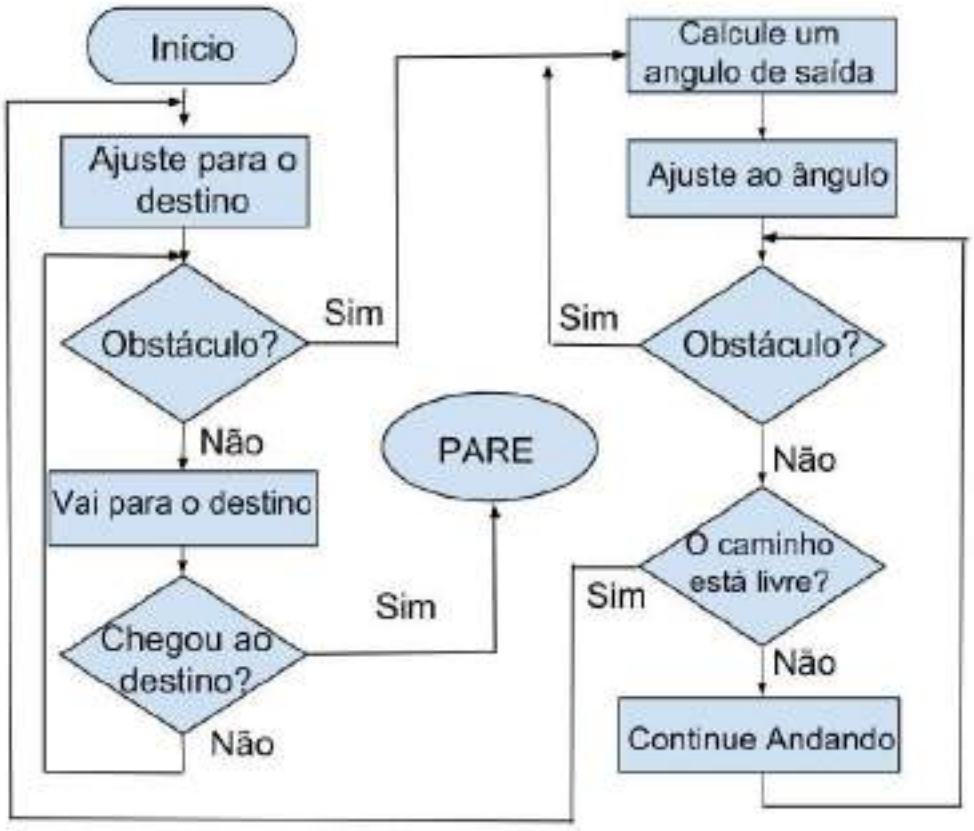
A partir do diagrama polar, percebesse que as zonas 1 e -1 estão livres, possibilitando a passagem do robô com uma margem de segurança.

4.1.2 Descrição do algoritmo

Inicialmente, o robô se ajusta e move em linha reta em direção à meta. Se for detectado um obstáculo dentro da bolha de sensibilidade o robô calcula um ângulo de recuo

para se afastar do obstáculo, (o ângulo de saída tende ser a mais baixa densidade polar dos obstáculos) e continua o seu movimento neste novo sentido, até que o objetivo torna-se visível (sem obstáculo dentro do intervalo de visibilidade do sonar, nesse sentido), ou até que um novo obstáculo é encontrado. O fluxograma na figura 33 representa o algoritmo.

Figura 33 – Fluxograma do Bubble rebound



Fonte: O próprio autor

4.1.3 Calculo do ângulo de recuo

Considerando que as células do sonar são distribuídas uniformemente a um ângulo que proteja toda a frente do robô:

Figura 34: Fórmula para divisão das células

$$\alpha_0 = \frac{\pi}{N}$$

Em seguida, o índice do sonar, i, contém informações angular:

$$\alpha_i = i\alpha_0$$

Onde N é o número total de células de sonar.

$$\alpha_i \in \left[-\frac{N}{2}, \frac{N}{2}\right]$$

A partir dessas informações, é possível calcular o ângulo de recuo é:

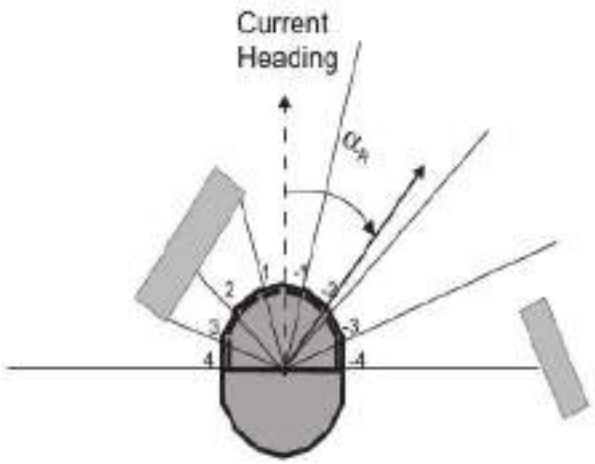
$$\alpha_R = \frac{\sum_{i=-\frac{N}{2}}^{\frac{N}{2}} \alpha_i D_i}{\sum_{i=-\frac{N}{2}}^{\frac{N}{2}} D_i}$$

Onde Di é o valor aferido pela célula do sonar i.

Fonte:

A figura 35 mostra o calculo do ângulo de recuo para um robô com 8 sonares.

Figura 35 – Cálculo do ângulo de saída

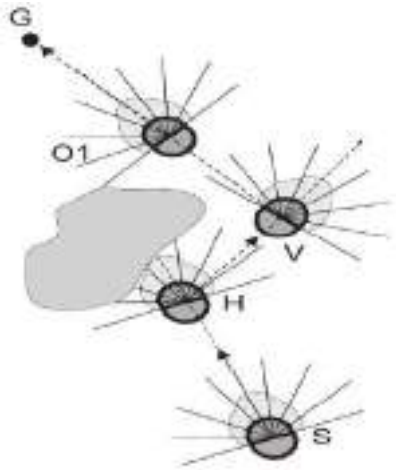


Fonte: Susnea et. al (2010)

Após o cálculo o robô se posiciona para o ângulo de recuo, deixando a sua frente alinhada, com isso o robô vai está livre para seguir em linha reta até que desvie do obstáculo e tenha a sua visão livre para o objetivo.

Por fim, uma ilustração da execução do algoritmo mostrado na figura 36.

Figura 36 – Execução do algoritmo



Fonte: Susnea et. al (2010)

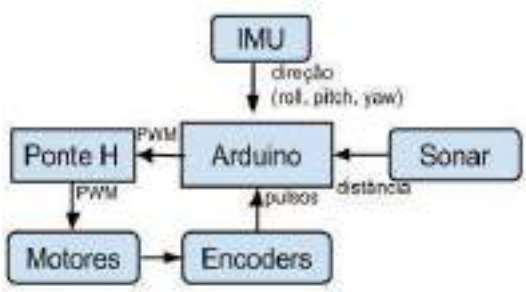
No ponto S o robô inicia a sua rota. Primeiramente ele se posiciona em direção ao objetivo e segue em linha reta. No ponto H ele identifica um obstáculo e calcula um ângulo de recuo para desviar do obstáculo. Rotaciona para alinhar ao ângulo e se desloca em linha reta. Em V o robô não identifica mais nenhum obstáculo próximo, e, volta para a seu estado inicial em direção ao objetivo. Chegando nele, finaliza a sua execução.

5 RESULTADOS

5.1 Arquitetura do Robô

O Arduino executa toda a comunicação e o processamento dos sensores, sendo que cada sensor é responsável pela informação necessária para o processamento do algoritmo. O fluxograma abaixo apresenta a arquitetura proposta do robô.

Figura 37 – Diagrama do robô

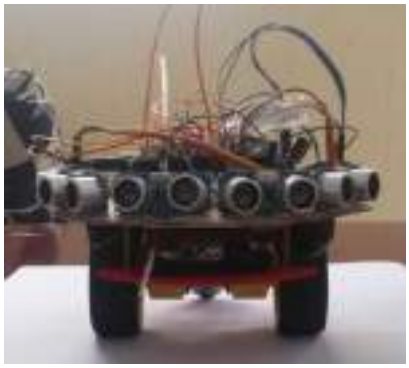


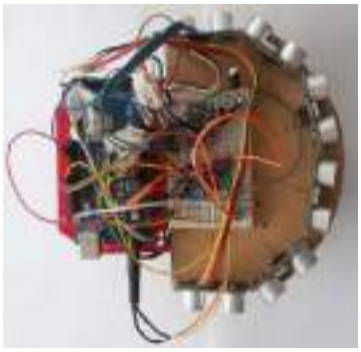
Fonte: O próprio autor

Para a execução do algoritmo, o robô precisa defini uma meta para se locomover. Para isso, a proposta é que o robô utilizaria a odometria, junto com a fusão de sensores com IMU e utilizaria um filtro para melhorar a estimativa de localização.

O controle da velocidade dos motores é feito pelo Proporcional-Integrativo-Derivativo (PID), que é um sistema de controle de malha fechada, sendo alimentado pelos encoders. Ele é bastante utilizado na robótica para um controle preciso dos motores. (DUDEK & JENKIN, 2010).

Figura 38a e 38b – A construção do robô e os seus módulos integrados.





O encoder não foi colocado no carro, pois não consegui fazer uma soldagem de precisão nos fios por falta de equipamentos para executar uma micro-soldagem, e, pela falta de técnica. Na tentativa de soldagem da placa ocorreu a danificação dos componentes. Apesar da falta do encoder o robô conseguiu se locomover pelo ambiente sem maiores problemas.

5.2 Detecção dos obstáculos e melhorias para o problemas dos sonares

O anexo 2 apresenta o código implementado em Arduino do teste para calcular as distâncias do 6 sonares com a bolha de sensibilidade. O programa imprime as distâncias aferidas pelos sensores, caso o obstáculo esteja no limite da bolha, imprime a existência do obstáculo dentro da bolha. Esse programa testa o cálculo da bolha e propõe melhorias para ruídos e para problemas dos sonares citados por Siegwart, Nourbakhsh & Scaramuzza (2011).

Para a diminuição do problema reflexão especular foi utilizado um anel de sonares no robô, cobrindo uma área angular maior. Aumentando, assim, a possibilidade de um sensor está perpendicular ao obstáculo, como mostrado na figura 30a. Já o problema do cross-talk foi totalmente resolvido com a divisão do tempo do ping dos sonares.

No programa foi utilizado a biblioteca *NewPing* desenvolvida por Tim Eckel[7] que é responsável por fazer a comunicação e abstração da parte física de vários modelos de sonares. Nessa biblioteca é definido o tempo de execução de cada sensor HC-SR04 com uma margem de segurança. No programa é implementado uma função concorrente onde é feita as leituras dos sensores melhorando o desempenho dos robôs, pois o programa não fica ocioso esperando o tempo de retorno da onda sonora. Além dessa melhoria no desempenho do robô, é definido um espaçamento entre as leituras dos sensores, eliminando o problema do cross-talk, pois a onda sonora não estaria vagando pelo ambiente quando o próximo sensor estiver executando. Na biblioteca é definida também, a leitura de cada sensor. Com isso, pode-se fazer o cálculo de quanto tempo demora cada ciclo. Então, com o tempo para cada leitura de 33ms x 6 sonares, é encontrado o tempo de cada ciclo, que é 198ms.

[7] Descrito em: //playground.arduino.cc/Code/NewPing

6 CONCLUSÃO

Durante a execução de um caminho em ambientes dinâmicos, o robô móvel deve identificar os obstáculos em tempo real e mudar a sua trajetória, baseado nas informações aferidas dos sensores para evitar a colisão. Esta é a competência do desvio de obstáculos.

Este trabalho faz uma análise das principais técnicas de desvio de obstáculos, aplicado a um robô de baixo custo. Dentre as técnicas citadas a de Susnea et al. (2010) é a que mais se adéqua ao problema. Assim, no trabalho é feita a análise do algoritmo.

A construção do robô foi planejada para a utilização do método de desvio de obstáculo Bubble Rebound, com um anel de sonares na frente do robô. Os problemas sofridos pelos sonares foram parcialmente resolvidos. Na detecção de obstáculos obteve-se uma melhoria na reflexão especular em relação a utilização de três sonares. Já no cross-talk o problema foi completamente resolvido com a divisão de tempo entre a execução dos sonares. Entretanto, o problema de foreshortening não foi possível implementar uma solução, por falta de tempo.

Para trabalhos futuros é sugerido à implementação do método Bubble Rebound para testes em ambientes físicos. Para isso, é necessário o acoplamento correto do encoder, implementar a odometria com fusão de sensorial do IMU e a utilização do filtro de Kalman proposto por Grewal et al. (1991) para a melhoria da estimativa da localização. Em relação ao problema do foreshortening do sonar, uma possível melhoria é a utilização da análise intervalar proposta por Trindade (2009).

REFERÊNCIAS

ASIMOV, I., I, ROBOT, B. B., (reprint). 1994

BAJRACHARYA, Max; MAIMONE, Mark W.; HELMICK, Daniel. **Autonomy for mars rovers: Past, present, and future**. Computer, v. 41, n. 12, p. 44-50, 2008.

CHOSET, H., LYNCH, K. M., HUTCHINSON, S., KANTOR, G., BURGARD, W., KAVRAKI, L. E., & Thrun, S. **Principles of robot motion: theory, algorithms, and implementations**. 2005. MITPress, Boston.

Da Vinci Surgery - **Minimally Invasive Robotic Surgery with** ... Disponível em: <www.davincisurgery.com/>. Acesso em 20.07.2015.

DUDEK, G., & JENKIN, M. **Computational principles of mobile robotics**. Cambridge university press. 2010.

ELFES, Alberto et al. **Robotic airships for exploration of planetary bodies with an atmosphere: autonomy challenges**. Autonomous Robots, v. 14, n. 2-3, p. 147-164, 2003. Disponível em: <https://www.media.mit.edu/people/hherr.> Acesso em 20.07.2015.

KANNIAH, J., ERCAN, M. F., & CALDERON, C. A. A. **Practical Robot Design: Game Playing Robots**. CRC Press. 2013.

KHATIB, M. **Sensor-based motion control for mobile robots**. Maher Khatib Sensor-based motion control for mobile robots PHD thesis, LAAS-CNRS, December, 1996.

KHATIB, M., JAOUNI, H., CHATILA, R., & LAUMOND, J. P. Dynamic path modification for car-like nonholonomic mobile robots. In: **Robotics and Automation**, 1997. Proceedings., 1997 IEEE International Conference on (Vol. 4, pp. 2920-2925). IEEE, April 1997.

KHATIB, O. **Real-time obstacle avoidance for manipulators and mobile robots**. The international journal of robotics research, *5*(1), 90-98, 1986.

KUMAR, V., BEKEY, G., & ZHENG, Y. (2005). **Industrial, personal, and service robots**. DRAFT REPORT, 41.

LUMELSKY, V. J., & STEPANOV, A. A. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. In: **Autonomous robot vehicles** (pp. 363-390). Springer New York, 1990.

LUMELSKY, V., & SKEWIS, T. **Incorporating range sensing in the robot navigation function**. Systems, Man and Cybernetics, IEEE Transactions on, *20*(5), 1058-1069, 1990.
MURPHY, Robin. **Introduction to AI robotics**. MIT press, 2000.

Niaja Farve | MIT Media Lab. Disponível em: www.media.mit.edu/people/nfarve. Acesso em 20.07.2015.

OROKO, J., & NYAKOE, G. N. Obstacle Avoidance and Path Planning Schemes for Autonomous Navigation of a Mobile Robot: A Review. In: **Proceedings of Sustainable Research and Innovation Conference** (pp. 314-318), December, 2014.

QUINLAN, S., & KHATIB, O. Elastic bands: Connecting path planning and control. In: **Robotics and Automation,** 1993. Proceedings., 1993 IEEE International Conference on (pp. 802-807). IEEE, May, 1993.

ROMERO, R. A. F.; SILVA JÚNIOR, Edson Prestes e; OSÓRIO, F. S.; WOLF, D.F. **Robótica Móvel**. 1. ed. Rio de Janeiro: LTC, 2014. v. 01. 350 p.

SIEGWART, R., NOURBAKHSH, I. R., & SCARAMUZZA, D. (2011). **Introduction to autonomous mobile robots.** MIT press.

SUSNEA, I., FILIPESCU, A., VASILIU, G., COMAN, G., & RADASCHIN, A. The bubble rebound obstacle avoidance algorithm for mobile robots. In: **Control and Automation (ICCA)**, 2010 8th IEEE International Conference on (pp. 540-545). IEEE, 2010.

YUFKA, A., & PARLAKTUNA, O. Performance Comparison of the BUG's Algorithms for Mobile Robots. In: **International Symposium on INnovations in Intelligent Systems and Applications** *(INISTA'09)* (pp. 416-421), 2009.

ZHU, Y., ZHANG, T., SONG, J., & LI, X. A new bug-type navigation algorithm considering practical implementation issues for mobile robots. In: **Robotics and Biomimetics (ROBIO**), 2010 IEEE International Conference on (pp. 531-536). IEEE, 2010.

Sites da internet:

http://elecfreaks.com/estore/download/EF03085-HC SR04_Ultrasonic_Module_User_Guide.pdf

http://mars.nasa.gov/mer/home/

http://www.amazon.com/oc/echo/

http://www.arduino.org/products/arduino-mega-2560

http://www.army-guide.com/eng/product.php?prodID=5022

http://www.army-guide.com/eng/product.php?prodID=5022

http://www.ti.com/lit/ds/symlink/l293.pdf

https://www.aldebaran.com/en/humanoid-robot/nao-robot

https://www.media.mit.edu/people/hher

https://www.sparkfun.com/products/retired/10825. **kit magician**.

ANEXOS

Anexo 1- Tabela de comparação dos algoritmos de desvio de obstáculos. (SIEGWART, R., NOURBAKHSH, I. R., & SCARAMUZZA, D., 2011).

Tabela 1 – Tabela de comparação dos algoritmos de desvio de obstáculos

Vector Field Histogram (VFH)			Bug			method	
VFH* [322]	VFH+ [176, 323]	VFH [77]	Tangent Bug [161]	Bug2 [198, 199]	Bug1 [198, 199]		
circle	circle	simplistic	point	point	point	shape	model fidelity
basic	basic					kinematics	
simplistic	simplistic					dynamics	
essentially local	local	local	local	local	local	view	
histogram grid	histogram grid	histogram grid	local tangent graph			local map	other requisites
						global map	
						path planner	
sonars	sonars	range	range	tactile	tactile	sensors	
nonholonomic (GuideCane)	nonholonomic (GuideCane)	synchro-drive (hexagonal)				tested robots	
6 ... 242 ms	5 ms	27 ms				cycle time	performance
66 MHz, 486 PC	66 MHz, 486 PC	20 MHz, 386 AT				architecture	
fewer local minima	local minima	local minima, oscillating trajectories	efficient in many cases, robust	inefficient, robust	very inefficient, robust	remarks	

Dynamic window		Curvature velocity		Bubble band		method	
Global dynamic window [81]	Dynamic window approach [130]	Lane curvature method [168]	Curvature velocity method [291]	Bubble band [165]	Elastic band [166]		
circle	circle	circle	circle	C-space	C-space	shape	model fidelity
(holonomic)	exact	exact	exact	exact		kinematics	
basic	basic	basic	basic			dynamics	
global	local	local	local	local	global	view	
	obstacle line field	histogram grid	histogram grid			local map	other requisites
C-space grid				polygonal	polygonal	global map	
NF1				required	required	path planner	
180° FOV SICK laser scanner	24 sonars ring, 56 infrared ring, stereo camera	24 sonars ring, 30° FOV laser	24 sonars ring, 30° FOV laser			sensors	
holonomic (circular)	synchro-drive (circular)	synchro-drive (circular)	synchro-drive (circular)	various	various	tested robots	
6.7 ms	250 ms	125 ms	125 ms			cycle time	performance
450 MHz, PC	486 PC	200 MHz, Pentium	66 MHz, 486 PC			architecture	
turning into corridors	local minima	local minima	local minima, turning into corridors			remarks	

Other				method	
Gradient method [171]	Global nearness diagram [225]	Nearness diagram [222, 223]	Schlegel [280]		
circle	circle (but general formulation)	circle (but general formulation)	polygon	shape	model fidelity
exact	(holonomic)	(holonomic)	exact	kinematics	
basic			basic	dynamics	
global	global	local	global	view	
	grid			local map	other requisites
local perceptual space	NF1		grid	global map	
fused			wavefront	path planner	
180° FOV distance sensor	180° FOV SICK laser scanner	180° FOV SICK laser scanner	360° FOV laser scanner	sensors	
nonholonomic (approx. circle)	holonomic (circular)	holonomic (circular)	synchrodrive (circular), tricycle (forklift)	tested robots	
100 ms (core algorithm: 10 ms)				cycle time	performance
266 MHz, Pentium				architecture	
		local minima	allows shape change	remarks	

Anexo 2

Código do programa implementado no Arduino

```
#include <NewPing.h>

#define SONAR_NUM 6 //número dos sensores
#define MAX_DISTANCE 200 // distância máxima de leitura
#define PING_INTERVAL 33 // tempo do ping entre os sensores

int bolha[SONAR_NUM] = {10,10,14,14,10,10}; // limite em cm da bolha já calculado, sendo que os sensores do meio tem o tamanho maio
unsigned long pingTimer[SONAR_NUM]; //array do tempo de cada execução do sonar
int sonarDistância[SONAR_NUM]; // array de distancias do sonar
uint8_t sonarAtivo = 0; //controle para saber qual sonar esta ativo


NewPing sonar[SONAR_NUM] = {     // Criando o objeto com vários sonar
 NewPing(23, 22, MAX_DISTANCE), // trigger pin, echo pin, max distance to ping.
 NewPing(25, 24, MAX_DISTANCE),
 NewPing(27, 26, MAX_DISTANCE),
 NewPing(29, 28, MAX_DISTANCE),
 NewPing(31, 30, MAX_DISTANCE),
 NewPing(33, 32, MAX_DISTANCE),
};

void setup(){

 Serial.begin(115200);// iniciando a Serial

    pingTimer[0] = millis() + 75;           // Primeiro de ping começa em 75ms, para dar tempo do arduino inciar
    for (uint8_t i = 1; i < SONAR_NUM; i++) // Define o tempo de inicio para cada sonar
     pingTimer[i] = pingTimer[i - 1] + PING_INTERVAL;
```

```
}
void loop(){
  for (uint8_t i = 0; i < SONAR_NUM; i++) { // Loop para percorrer todos os sonares
    if (millis() >= pingTimer[i]) {         // É hora deste sensor para executar ping?
     pingTimer[i] += PING_INTERVAL * SONAR_NUM;  // Definir a próxima vez este sensor vai ser pingado
     if (i == 0 && sonarAtivo == SONAR_NUM - 1) cicloCompletoSonares(); // Verifica se o ciclo de sonares foi completo.
     sonar[sonarAtivo].timer_stop();          // Certifique-se de temporizador anterior é cancelada antes de iniciar um novo ping.
     sonarAtivo = i;                          // Sensor i é acessado.
     sonarDistância[sonarAtivo] = MAX_DISTANCE; // Na ausência de obstáculos o valor do sonar dever ser igual ao valor máximo.
     sonar[sonarAtivo].ping_timer(echoCheck); // Faça o ping em paralelo ao processamento
    }
 }
 // Resto do programa que não utiliza o ping
}

void echoCheck() { // Se recebeu o ping, setar no array de sonar
 if (sonar[sonarAtivo].check_timer())
   sonarDistância[sonarAtivo] = sonar[sonarAtivo].ping_result / US_ROUNDTRIP_CM;
}

void cicloCompletoSonares() { // Função para executar alguma tarefa depois que todos os sonares forem executados
 // imprime as distâncias
 for (uint8_t i = 0; i < SONAR_NUM; i++) {
   Serial.print("      ");
   Serial.print(i+1);
   Serial.print(" = ");
   Serial.print(sonarDistancia[i]);
   Serial.print("cm ");
```

```
 }
 Serial.println();
 //Verifica se existe algum obstáculo na bolha
 for(int i=0;i < SONAR_NUM ;i++){
       if(sonarDistancia[i] <= bolha[i]){
          Serial.print(" Existe obstáculo na bolha: ");
          Serial.print(i+1);
          Serial.println();
          break;
        }
 }
}
```

Nesse teste foram feitas as leituras do sonar, sendo que o robô se encontra no meio dos obstáculos em forma de U. Para analisar a existência de uma possível interferência entre os sensores vizinhos, obstrui cada sensor e verifiquei a sua leitura. Nos resultados dos testes as leituras dos sensores vizinhos continuaram normais, comprovando a eliminação do problema de cross-talk.

Figura 39 – Teste do ultrassom