BRUNO HENRIQUE LOBO NETTO PEIXOTO
LUCAS LUDOVICO MARTINS DA SILVA

LOCALIZAÇÃO E NAVEGAÇÃO DE UM ROBÔ AUTÔNOMO

São Paulo
2015

BRUNO HENRIQUE LOBO NETTO PEIXOTO
LUCAS LUDOVICO MARTINS DA SILVA

Localização e navegação de um robô autônomo

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia Mecânica – ênfase em Automação e Sistemas

Orientador: Prof. Dr. Eduardo Lustosa Cabral

São Paulo
2015

BRUNO HENRIQUE LOBO NETTO PEIXOTO
LUCAS LUDOVICO MARTINS DA SILVA

Localização e navegação de um robô autônomo

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia

Área de Concentração: Robótica

Orientador: Prof. Dr. Eduardo Lustosa Cabral

São Paulo
2015

AGRADECIMENTOS

Os agradecimentos principais são direcionados ao Prof. Dr. Eduardo Lustosa Cabral pelas orientações à distância dadas durante a realização do projeto.

Também agradecemos ao Dr.-Ing. Eric Lenz pelos conselhos concedidos e tutoria durante o período na Alemanha. Da mesma forma, ao Dr. –Ing Andreas Haun por servir de intermédiário entre a Technische Universität Darmstadt e a Escola Politécnica da USP e possibilitar-nos a dupla-diplomação.

Não somente, mas também ao Prof. Dr. Thiago de Castro Martins e ao Prof. Dr. Marcos Ribeiro Pereira Barreto em razão dos conselhos concedidos relacionados a processamento de imagens e eletrônica durante a realização deste trabalho.

Além disso, agradecemos aos familiares pela compreensão e suporte durante o trabalho.

A todos estes e também àqueles que contribuíram grandiosamente para com o nosso aprendizado e formação, tenham o nosso mais sincero agradecimento.

“I can live with doubt and uncertainty. I think it's much more interesting to live not knowing than to have answers which might be wrong." - Richard Feymann.

RESUMO

Robôs móveis é um dos tópicos mais pesquisados na atualidade na área de robótica. Estes fazem-se presentes em contexto militar, na medicina, entretenimento, automobilística, entre outros. Para que o robô autônomo obtenha sua localização no espaço, este deve fazer uso de informações provenientes de sensores que possibilitam sua interação com o meio no qual ele está inserido. A forma mais comum de autolocalizar-se utiliza processamento de imagens. Outra forma mais simples faz uso de informações provenientes da odometria, apesar do acúmulo proveniente de escorregamento das rodas. Este trabalho objetiva conceber um sistema mecatrônico no qual um robô está inserido em um ambiente desconhecido. Assim, por meio de dados provenientes das rodas e de uma câmera acoplada ao teto do recinto, ele é capaz de localizar-se de forma mais precisa e atingir o seu objetivo sem colidir com nenhum obstáculo em seu percurso. Para tal, foi-se implementado uma estratégia de controle que leva em consideração diversos tipos de obstáculos, de forma que o robô não somente alcance seu objetivo, mas também não se perca no ambiente no qual está inserido. Além disto, devido a inconstante disponibilidade de dados provenientes da câmera, implementa-se também um algoritmo de fusão de dados advindos da câmera e da odometria, objetivando informações mais confiáveis.

Palavras-Chave: Mecatrônica, robótica, autônomo, processamento de imagem, navegação

ABSTRACT

Mobile Robots is one of the most researched subjects from nowadays in the robotic area. Among others, their applications may be seen on military context, entertainment, automobiles, etc. The autonomy of the robot depends of its ability to find itself in the space. In order to do that, it must rely on the data of internal sensors to allow it to interact with the environment in which it was placed. The most common way to do so is based on the image processing. Another easier way consists to exploit the data from odometry, even though it may not be reliable in reason of the error coming from slippage. This work aims to conceive a mechatronic system in which the robot is inserted in an unknown environment. With the data coming from the wheels as well the processed data of the camera coupled to the ceiling, the robot is able to localize itself on a safer way and reach his goal without collide with any obstacle on his path. This is possible thanks to the implemented control strategy which consider different types of obstacles, in such a way that the robot not just reach the goal, but also do not get lost. In addition, because of the inconstant disposal of the camera data, an algorithm must be implemented to merge the data coming from the camera and the odometry, to provide reliable information and enable a safer navigation.

Keywords: Robotics, autonomy, image processing, navigation

SUMÁRIO

1. Introdução

Há um século robôs atiçam a imaginação do ser humano. O termo, cravado pelo escritor tcheco K. Čapek no início da década de 20 do século XIX, origina-se da língua eslava e tem como significado “labuta” ou “trabalho voluntário” (Adamy, 2014) Desde então a área denominada “Robótica” desenvolveu-se exponencialmente. Isto propiciou não somente um aumento na produção industrial, mas também surtiu, e surte, efeito na vida cotidiana das pessoas.

Na indústria, robôs fazem-se mais presentes na montagem e soldagem automobilística além da manipulação de medicamentos no setor farmacêutico. Na medicina, robôs manipuladores auxiliam médicos na realização de cirurgias além de também serem concebidos como aparelhos ortopédicos que possibilitam a reabilitação de pacientes com restrições motoras. Dentre outros, um ramo que cada vez mais tem sido divulgado pela mídia em geral é o relacionado a robôs móveis.

Robôs móveis apresentam diversas aplicações. Em âmbito militar, por exemplo, podem ser utilizados para vasculhar uma região a qual sofreu um acidente nuclear, um prédio no qual há risco de explosão ou desmoronamento ou mesmo com intuito de carregar artilharia e mantimentos. Em geral, o robô encontra-se em um local desconhecido e tem de confiar nos dados obtidos pelos sensores locais para localizar-se no espaço. Entretanto, para aplicações onde o custo é um requisito crítico, sensores de baixo custo, assim como, aspectos físicos e construtivos de projeto comprometem a performance do robô, gerando a necessidade de desenvolvimento de algoritmos para compensação desta restrição de projeto (Chieh-Chih Wang, 2003).

A solução mais comum utilizada como sensor para navegação de robôs autônomos consiste no uso de uma câmera embarcada. Dentre outros, um tema bastante pesquisado na atualidade, por exemplo, consiste na implementação de algoritmos de auto localização e mapeamento de um automóvel (em inglês, *Simultaneous Localization and Mapping*) (Juan Fasola, 2005). Como o próprio nome diz, por meio de processamento de imagens e de um modelo probabilístico, o robô determina sua localização ao mesmo tempo que constrói um mapa do ambiente no qual se encontra, possibilitando uma navegação mais segura.

Restrito não somente a câmeras embarcadas, estas podem ser instaladas também em ambientes *indoor*. Apesar de menor campo de visão, elas também

fornecem informações necessárias para o posicionamento e orientação do robô. O desenvolvimento deste projeto faz uso deste princípio.

A ideia principal deste projeto consiste em desenvolver e implementar algoritmos que possibilitam um robô ir de um ponto inicial a um final estipulado pelo usuário em um ambiente. Esse ambiente pode ser totalmente desconhecido, parcialmente ou completamente conhecido. Desta forma, busca-se, por meio de sensores locais (câmera, sensores infravermelhos e encoder), estabelecer a interação entre robô e o ambiente. Apesar de clara, a tarefa não é de fácil execução, pois em princípio o robô não tem informações do local em que está inserido e podem haver obstáculos que obstruem o caminho.

2. Objetivos

O projeto em questão visa o desenvolvimento e teste de um algoritmo de navegação para robôs autônomos. O objetivo primário consiste em desenvolver uma estratégia de controle para o robô chegar a um local desejado no ambiente, estipulado pelo usuário e ao longo do percurso, desviar de obstáculos. Para tal, o robô necessita obter dados de sua localização e sua orientação. Em primeiro momento, dados de localização oriundos da odometria são extremamente úteis. Contudo, devido ao deslizamento das rodas ocorre no resultado da odometria um acúmulo de erros que afeta a localização do robô e por consequência sua trajetória. Em um caso ideal, as rodas não deslizam e, assim, por meio da odometria é possível saber a trajetória do robô e sua localização.

A fim de solucionar o problema de localização e, assim, obter uma estimativa melhor da posição do robô, é usado, em conjunto com a odometria, uma câmera e um sistema de processamento de imagens. Por meio de imagens adquiridas pela câmera instalada no topo de uma sala, pode-se obter a posição e a orientação do robô. Estes dados são transferidos por meio de comunicação wireless ao robô que, assim, consegue realizar a estratégia de controle desejada de uma maneira mais precisa.

Além disso, é necessário cconsiderar eventos tais como intervenções de agentes externos. Como por exemplo, pode-se citar a colisão com outros robôs assim como com pessoas. Valores oriundos da odometria, da mesma forma que no caso da derrapagem, são após tais eventos não mais confiáveis. Da mesma forma, o ambiente pode ter obstáculos que impeçam a locomoção em linha reta do robô.

A área de navegação de robôs autônomos tem apresentado um grande desenvolvimento nas últimas décadas. Assim, a principal motivação dos orientados para realizar este trabalho é o grande interesse em robótica e sua aplicação prática. Além disso, o projeto objetiva o aprimoramento de conhecimento em áreas como Controle e Automação, Modelagem, Robótica e Programação de Microprocessadores.

Acredita-se também que este seja um tema de grande relevância na área da Engenharia Mecatrônica, uma vez que mescla diversos assuntos abordados durante todo o curso de graduação. Dentre outros, estão a montagem de um robô, concepção de circuitos elétricos e eletrônicos, comunicação direta de software e hardware através de programação, além da implementação de técnicas de controle para navegação autônoma e processamento de imagens.

3. Revisão Bibliográfica

3.1 Navegação de robôs autônomos

A navegação de robôs autônomos até locais ou posições pré-estipuladas é um tema de pesquisa na área robótica desde o fim da década de 90. Tal problema é em geral classificado como global ou local, dependendo do ambiente investigado. No primeiro caso, o ambiente é conhecido e um determinado caminho é selecionado, por exemplo, por meio de uma rota traçada em um mapa gráfico, onde os obstáculos são visíveis. No segundo, este é desconhecido ou parcialmente conhecido. Em ambos os caso, tanto uma estratégia de controle quanto a utilização de sensores são imprescindíveis para possibilitar a interação do robô com o meio (Atsushi Fujimori, 1997).

As técnicas existentes para para o controle da navegação de robôs autônomos é divida em três vertentes: arquitetura de controle deliberativa, reativa e híbrida. Na arquitetura deliberativa, os robôs primeiramente recebem os dados do ambiente através de seus sensores e os usa para construir um modelo do ambiente. A partir daí, o robô gera uma estratégia que o permita chegar a seu objetivo e a executa. Suas desvatagens são que um grande tempo é gasto para sensoriar o ambiente e construir o modelo. Uma vez que o robô é incapaz de operar fora do ambiente modelado, ruídos e imprecisões comprometem sua navegação, tornando-o um sistema pouco ágil e pouco confiável. Na arquitetura reativa, os robôs seguem o princípio da reatividade. Estão em contato com o ambiente durante toda a navegação através de seus sensores e o controle é dividido em uma série de processos que concorrem entre si. Cada processo consiste de uma ligação entre um ou mais sensores e atuadores e tem a possibilidade de inibir outros processos advindos, dependendo das prioridades de cada um. A arquitetura híbrida combina o melhor dos sistemas deliberativos e reativos. Nesse tipo de arquitetura, pode-se usar componentes da arquitetura deliberativa para, por exemplo, planejamento de longo prazo (ir para um local), e componentes da arquitetura reativa para lidar com situações mais imediatas, como por exemplo, evitar obstáculos (Heinen, 2002).

Problemas de navegação local e arquitetura híbrida, levando em consideração sua dinâmica, são abordados em (Amy Briggs, 2006), (Juan Fasola, 2005), (Couto,

2009), (Amy Briggs, 2006) e navegação global e arquitetura deliberativa em (Giuseppina Gini, 2002).

(Juan Fasola, 2005), implementa em um robô AIBO da Sony um algoritmo de navegação que permita o robô chegar a um destino e também evitar obstáculos. Equipado de uma câmera monocular, o robô é capaz de detectar e localizar obstáculos. Para alcançar o seu objetivo, o algoritmo de navegação utiliza duas rotinas: (1) caso a área à sua frente esteja livre, navegar diretamente até onde deseja; (2) caso contrário, contornar o objeto até que o caminho ao destino esteja livre. Devido ao seu restrito campo de visão, o caminho ótimo, em geral, não é garantido.

(Amy Briggs, 2006) utiliza um robô com uma câmera que obtém subsequentes imagens durante a navegação e compara a imagem atual com as já armazenadas. A partir disso, através da comparação da localidade atual com as já conhecidas, a posição e a orientação do robô são determinadas.

Ao invés de embarcar a câmera, (Giuseppina Gini, 2002) afixa uma câmera em uma posição acima do robô, como ao teto de uma sala. Após sua calibração, imagens do chão são obtidas e a posição dos obstáculos é calculada. Em sequência, as novas imagens são comparadas, criando um mapa do ambiente. Em seguida, uma trajetória é transferida ao robô para ser executada.

Existem sistemas nos quais a localização do robô é feita por meio de um sistema híbrido usando odometria e visão por câmera (iGPS) (Couto, 2009). A informação obtida pela câmera é usada para corrigir o erro acumulado durante o percurso, proveniente da odometria. A fusão de ambas informações é feita através de um filtro de Kalman. A câmera utilizada, neste caso, objetiva apenas o cálculo da posição e não o reconhecimento gráfico do ambiente.

Outras abordagens semelhantes ao tema do projeto são encontradas em (Hansye S. Dulimarte, 1997), em que o robô é provido de um sistema de auto-localização (*self-localization*). Tal capacidade é muito útil caso haja falhas de navegação. Nesse caso, o robô parte de uma posição conhecida e a mesma é atualizada de acordo com suas ações durante o percurso. Porém, tal sistema está sujeito a ruídos de sensores, causando inconstância na estratégia de navegação.

3.2 Localização com câmeras

A identificação de objetos em uma cena pode ser realizada de diversas maneiras. Em primeiro momento, caso este tenha uma cor característica, esta pode ser identificada por meio da limiarização do espaço de representação de cores, limitando neste o intervalo no qual a cor do objeto está inserido. Fatores como saturação e intensidade podem ser utilizados a fim de aprimorar o algoritmo. Para tal, existem duas formas convencionais: a primeira necessita da conversão para o espaço HSI, i.e., *Hue* (do inglês, matiz), *saturation* (do inglês, saturação) e *intensity* (do inglês, intensidade), o qual desacopla em teoria fatores de cor, saturação e intensidade luminosa, respectivamente; a segunda faz uso ainda do espaço de representação RGB normalizado, visto que fatores de luminosidade tem grande influência sobre este (Cabral, 2015).

O uso de *features* identificáveis, como formas geométricas e/ou marcadores de referência, por sua vez, pode ser realizada de diferentes formas. Para o primeiro, utiliza-se o algoritmo de Hough, o qual identifica retas, círculos ou elipses na imagem. Estes podem ser acoplados ao robô e identificados. O uso de marcadores identificáveis é também uma forma eficaz de localizar um objeto em uma cena. Atualmente, existem bibliotecas bem estabelecidas que possibilitam uma maior rapidez para integração do algoritmo ao código de localização do robô (Cabral, 2015).

Após sua identificação, a obtenção no mundo real do robô deve ser realizada. Isto se dá por meio da calibração da câmera, ou seja, obtenção dos parâmetros que descrevem esta. O objetivo deste procedimento neste projeto consiste, por meio da posição em pixels de uma imagem, readquirir a posição do robô no mundo real. Em (Madhkour, Mancas, & Dutoit, 2015) é feita uma revisão dos métodos padrões para calibração. Até então, os mais utilizados pela comunidade científica são: Transformação Linear Direta (em inglês, *Direct Linear Transform*), método de Tsai e Haikkila (Borenstein, 1996) e, por fim, método de Zhang. Em essência, todos estes fazem uso da representação matemática de uma câmera *pinhole*.

O primeiro – dentre os demais o mais straight-forward – obtém os parâmetros desejados através da multiplicação do modelo da câmera pela seguinte matriz antissimétrica. A solução desse problema advém da minimização da equação que define o erro algébrico. O1 corpo de calibração é, em geral, o de um cubo ou padrão 3D com marcadores identificáveis e de medidas precisas no mundo real. Possíveis

melhorias podem ser obtidas por meio do cálculo de minimização do erro estabelecido entre a imagem salva e os valores do modelo encontrado.

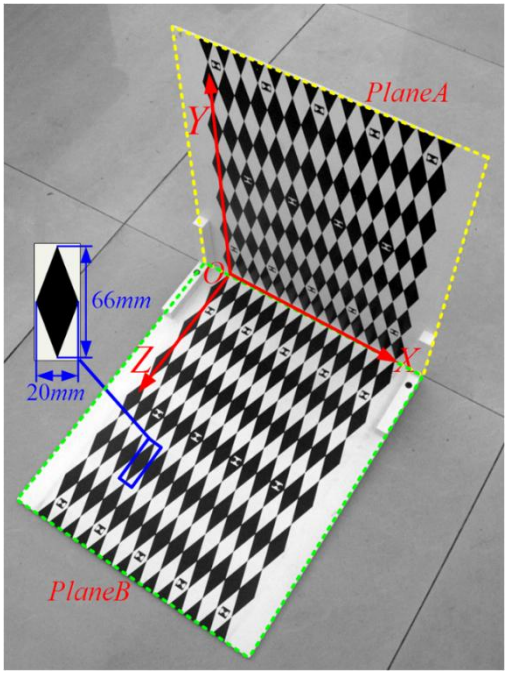

Figura 1 - Padrão tridimensional para calibração de uma câmera de acordo com o método de DLT (Fonte: http://opticalengineering.spiedigitallibrary.org/article.aspx?articleid=1389186)

O segundo, explicado de forma elucidativa em (Madhkour, Mancas, & Dutoit, 2015), aprimora o método acima por meio da incorporação do modelo de distorções radiais. Estas geralmente provêm de imperfeições de fabricação das lentes da câmera e são modeladas, dependendo do nível de deformação da imagem pelos dois primeiros componentes da série de Taylor. A ideia principal do método consiste primeiramente na obtenção dos parâmetros até então obtidos pelo método DLT. Desta forma, o cálculo de parâmetros de distorção assim como aprimoramento dos valores do foco f e posição t_z são obtidos por meio de um método não-linear de minimização da distância euclidiana.

O método mais utilizado em projetos que envolvem processamento de imagens é denominado método de Zhang (Coelho & R. S. T., 2003). Isto dá-se pelo fato deste ser o padrão implementado no Open CV, biblioteca de distribuição livre e que possui inúmeros algoritmos com este fim. O corpo de calibração utilizado por este podem ser um tabuleiro de xadrez, círculos dispostos em forma de grade ou alternadamente deslocados. No caso da escolha de um padrão xadrez de calibração, como demonstrado na figura 2, os pontos a serem utilizados são obtidos a partir da

intersecção de retas geradas a partir dos vértices das casas do tabuleiro. A partir disso, calcula-se da mesma forma que os métodos anteriores os parâmetros de calibração intrínsecos e extrínsecos assim como os relacionados à distorção da imagem.

Dentre outros, em (Madhkour, Mancas, & Dutoit, 2015) são citados ademais métodos para o processo de calibração. Estes diferenciam-se, em geral, por sua robustez, i.e., sensibilidade à fatores do ambiente, objeto utilizado para calibração da câmera assim como acurácia de cálculo da matriz de transformação.

No projeto em questão, pode-se simplificar sensivelmente este processo. Isto faz-se possível por meio da bidimensionalidade do problema. Como abordado em (Giuseppina Gini, 2002), o plano de projeção da imagem encontra-se paralelo ao de locomoção do robô. Com um padrão conhecido, faz-se uso desta simplificação e, assim, obtém-se a matriz de calibração para o problema. Tal abordagem, por exemplo, possibilita um menor custo computacional não somente na aquisição desta matriz, mas também durante a sua utilização no decorrer do programa.

4. Concepção do sistema

4.1 Aspectos construtivos mecânicos e eletrônicos

O chassi do robô deste projeto, comercializado no mercado e chamado de *magician chassis*, é constituído de duas plataformas nas quais os componentes eletrônicos, mecânicos e sensores são montados.

Na porção inferior são afixados dois motores DC de pequeno porte com tensão de alimentação de 6 V, velocidade de rotação máxima de 90 ± 10 rpm e de corrente sem carga de 190 mA. A estes são montadas duas rodas. Para suporte, um rodízio é utilizado na parte dianteira do chassi. Para realizar a interface entre a porção de controle e potência do circuito, é utilizado um driver de potência Ponte H que suporta até 1 A de corrente.

A aquisição de velocidade e sentido de rotação é realizada pelo “RedBot Sensor - Wheel Encoder”. Para tal, ele conta as transições entre lacunas geradas por um disco dentado acoplado ao eixo principal do motor por meio da sensibilização do fototransistor QRE1113GR. O sinal proveniente deste, porém, é primeiramente filtrado e amplificado.

Para cálculo da distância do robô aos obstáculos à sua volta, são instalados cinco sensores de distância infravermelhos de curto alcance localizados na laterais, diagonais e na dianteira do chassi. Estes detectam distâncias entre 4 e 30 cm em forma de uma tensão de 0.4 a 2.75 V. A relação entre tensão e distância é, para os sensores em questão, não-linear e é obtido experimentalmente. Após a amostragem de diversos pontos, é feita uma interpolação polinomial para melhor aproximacão. Esta tensão, por sua vez, é primeiramente convertida em um sinal digital compreendido entre 200 e 1375 por meio de um canal de 12-bits e 1,8 V. O posicionamento e disposição foram escolhidos a fim de cobrir a maior área possível e, assim, manter o robô alerta. Nenhum sensor foi escolhido para a porção anterior pois, do ponto de vista do projeto, ele não efetua nenhum movimento para trás.

Em sua porção superior encontram-se o microcontrolador BeagleBone Black (BBB) e o suporte de baterias. O sistema é alimentado por um conjunto de 8 baterias recarregáveis em série, totalizando 12 V. A decisão de comprá-las deu-se pelo seu alto consumo de energia. A placa de processamento BeagleBone Black, entretanto, tem como alimentação 5V, sendo necessário para isto um regulador de tensão para

adequar a tensão da fonte. Para que não haja retorno da corrente, foi-se instalado um diodo 1N4007 entre a tensão de alimentação e a entrada do regulador. Além disto, capacitores foram adicionados à entrada e saída do regulador de tensão de forma a manter as tensões livres de ruído.

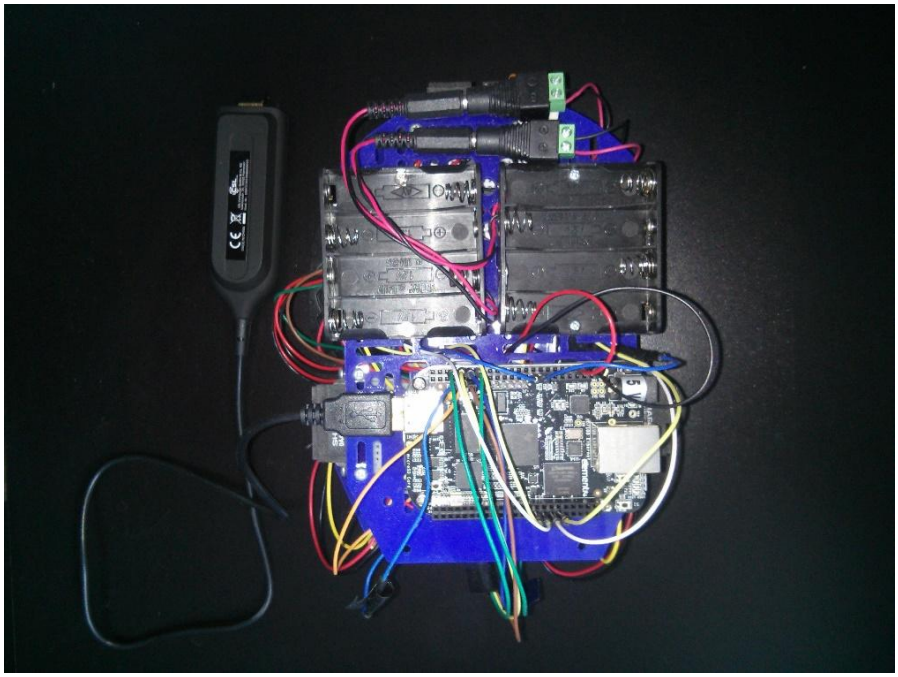

Figura 2 - Vista superior do robô QuickBot

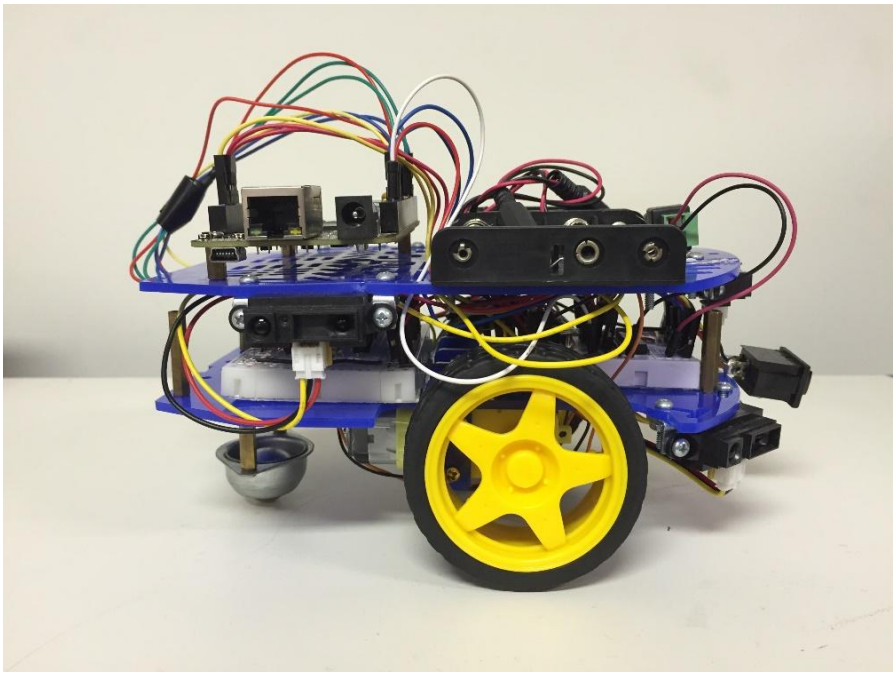

Figura 3 - Vista lateral do robô QuickBot

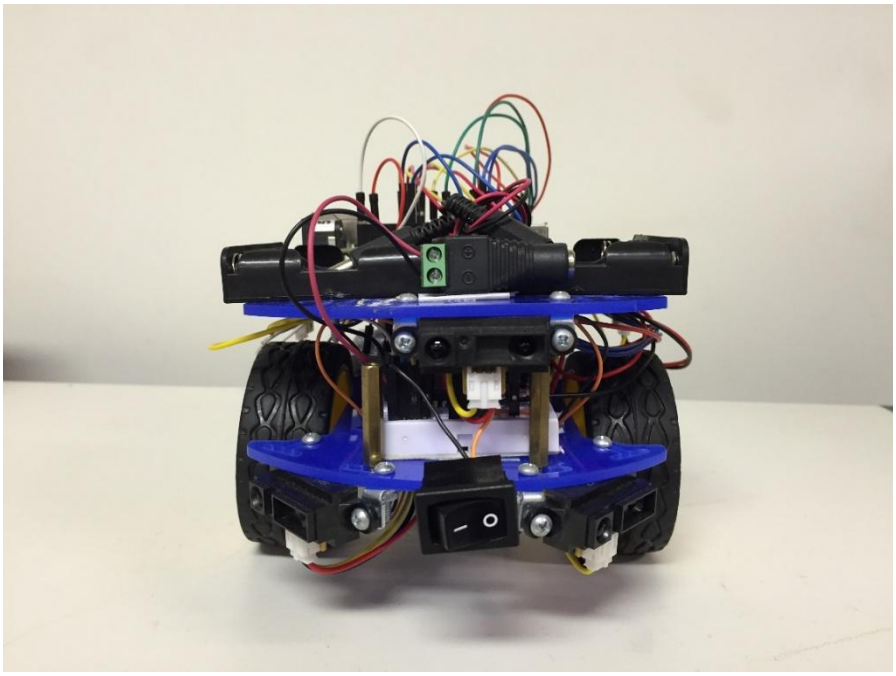

Figura 4 - Visão frontal do robô QuickBot

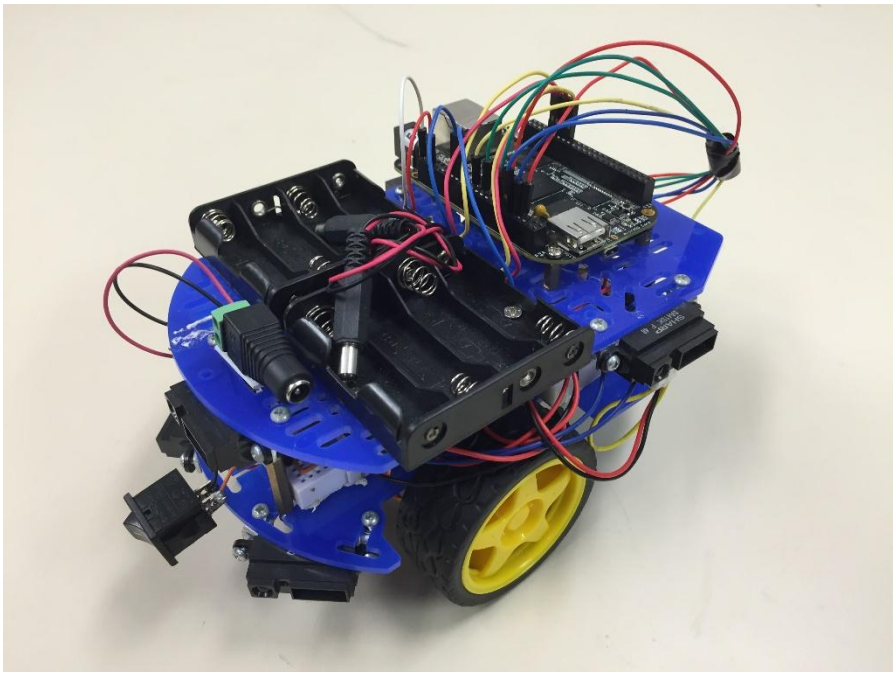

Figura 5 - Visão 3D do robô QuickBot

4.2 Localização do robô por câmera

Originalmente, o autômato tinha o design apresentado na Figura 5. Observou-se, no entanto, que esta configuração vai de encontro com alguns requisitos no que diz respeito ao processamento de imagem. Em sua parte superior, encontra-se um

soquete de pilhas alcalinas assim como o microprocessador, o que obstrui a visão e identificação do robô.

Frente a isso, a característica adotada para identificação do robô foi sua cor. Para isto, o robô deve distinguir-se sensivelmente de sua circunvizinhança assim como ter uma área facilmente identificável. A fim de solucionar este inconveniente, foi-se utilizado um cartão com apenas uma cor, posicionado sobre o robô, possibilitando sua identificação.

Para os cálculos necessários e identificação clara da orientação do robô, foram utilizados marcadores visivelmente diferenciáveis das demais cores à sua redondeza em sua parte superior.

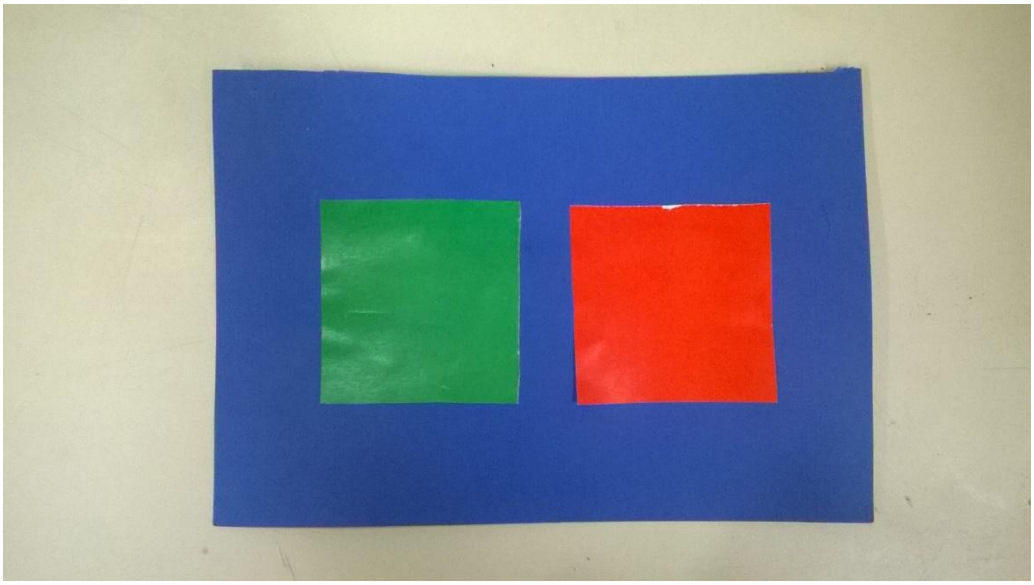

Figura 6 - Marcador do robô

A câmera para aquisição de imagens do ambiente é estática e afixada no teto. Esta tem como objetivo captar imagens de uma determinada região no espaço na qual o robô está inserido. Assim, é possível saber a posição e a orientação iniciais do robô (dados imprescindíveis para o ínicio de sua navegação). Não só isso, a câmera fornece, durante a navegação, dados da posição do robô que servirão de complemento para os dados advindos do encoder.

Para fins de simplificação, optou-se por trabalhar em um cenário ideal, no qual o robô é inserido em um ambiente com apenas uma cor. O robô da mesma forma, apresenta também apenas uma cor e sua orientação é obtida por meio de marcadores localizados em sua porção superior.

Por meio das simplificações adotadas, finalmente é possível elaborar uma estratégia de identificação do robô. Esta consiste em suma nos seguintes passos:

1. Aquisição de um frame pela câmera instalada no ambiente;
2. Conversão do espaço de cores RGB para HSV;
3. Thresholding dos pontos os quais são compreendidos no intervalo da cor do robô;
4. Cálculo dos contornos de cada um dos objetos encontrados na etapa anterior;
5. Dada a hipótese de simplificação, o robô corresponde ao objeto de maior área na imagem. Assim, realiza-se os cálculos necessários e seleciona-se aquele que satisfaz a hipótese adotada;
6. Cálculo da posição do centróide do contorno em questão;

4.3 Desenvolvimento do algoritmo de controle do robô

Uma vez que a posição e a orientação iniciais do robô são conhecidas, uma estratégia de controle pode ser iniciada. Como já citado anteriormente, o objetivo do robô é alcançar o seu destino e evitar os obstáculos no meio do caminho, tendo sua posição atualizada tanto pela câmera quanto pelo encoder. O diagrama a seguir ilustra o funcionamento do sistema de controle da navegação de forma generalizada.

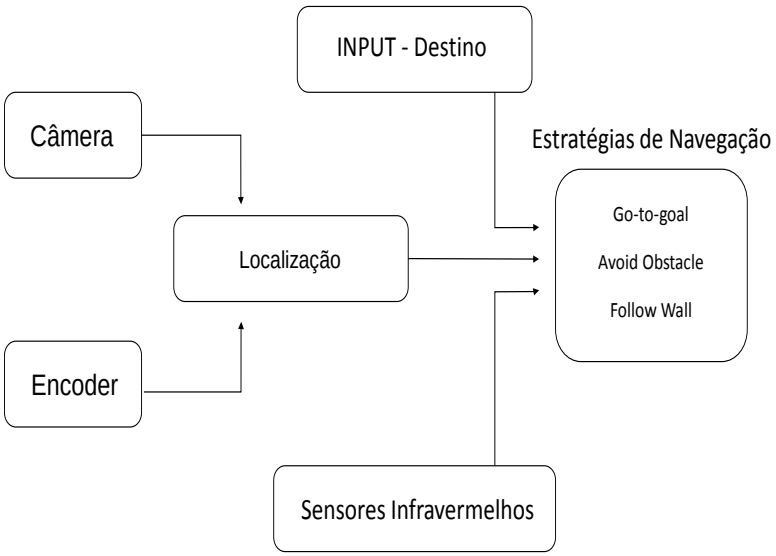

Figura 7 - Sistema de Controle do Robô

A estratégia de controle Go-to-Goal consiste em levar o robô ao destino desejado. No caso abordado, este faz menção à posição estipulada pelo usuário até a qual o robô deve navegar. A estratégia Avoid-Obstacle permite que o robô desvie de obstáculos que estejam em seu caminho, e é acionada quando dados oriundos dos sensores infravermelhos acusam que o robô se aproxima do obstáculo. A estratégia Follow Wall é desenvolvida para casos especiais onde os obstáculos são não convexos e permite que o robô siga os contornos do obstáculo até que sua ida ao destino final seja possível sem mais impedimentos locais.

Para implementar todas essas ações de forma ordenada e contínua, faz-se o uso de uma máquina de estados, onde os estados são as estratégias de navegação e os eventos de transição são condicionados aos valores das variáveis de sensoriamento (localização, infravermelhos).

4.4 Implementação do sistema de controle e localização

A implementação das estratégias de navegação e localização foram realizadas no software de simulação MATLAB. Através da interface “Sim.I.am”, toolbox implementada e desenvolvida pela Georgia Tech University (Egerstedt, 2015), é possível simular a navegação do robô. No entanto, o simulador só é capaz de localizar o robô através da odometria, não integrando a localização por processamento de imagem.

5. Modelo do robô

O robô utilizado nesse projeto é de grande aplicação na área da robótica e conhecido como robô de duas rodas diferenciais. Sua estrutura consiste basicamente de duas rodas motorizadas e uma terceira roda de apoio.

O modelo utilizado para equacionar o robô é obtido por meio de uma análise cinemática e dinâmica do sistema.

5.1 Modelo Cinemático:

As velocidades linear e angular do robô são funções das rotações das rodas e descritas pelas seguintes equações:

$$v_R = \frac{R}{2}(\omega_d + \omega_e) \tag{1}$$

$$\omega_R = \frac{R}{L}(\omega_d - \omega_e) \tag{2}$$

Sendo ω_d e ω_e a velocidades angulares respectivamente das rodas direita e esquerda, R é o raio das rodas e L é a distância entre elas. Para que a localização do robô no espaço seja controlada, é preciso conhecer tanto sua posição linear quanto angular. Introduzindo as relações entre velocidade linear e angular e posição linear e angular e isolando as velocidades angulares das rodas, obtém-se as seguintes relações:

$$\dot{x} = \frac{R}{2}(\omega_d + \omega_e)\, cos\, \emptyset \tag{3}$$

$$\dot{y} = \frac{R}{2}(\omega_d + \omega_e)\, sin\, \emptyset \tag{4}$$

$$\omega_d = \frac{2v_R + \omega_R L}{2R} \tag{5}$$

$$\omega_e = \frac{2v_R - \omega_R L}{2R} \tag{6}$$

$$\dot{\emptyset} = \frac{R}{L}(\omega_d - \omega_e) \tag{7}$$

Sendo as equações (5) e (6) oriundas do sistema de equações (1) e (2). Substituindo (5) e (6) em (3) e (4), obtém-se um modelo mais intuitivo em que as velocidades lineares nas direções x e y e a velocidade angular do robô são função respectivamente do módulo da velocidade linear e da velocidade angular.

$$\dot{x} = v_R \cos \emptyset \tag{8}$$

$$\dot{y} = v_R \sin \emptyset \tag{9}$$

$$\dot{\emptyset} = \omega_R \tag{10}$$

5.2 Modelo Dinâmico:

Ao aplicar-se a segunda lei de Newton e o teorema do momento angular no robô, assumindo que o mesmo é um corpo rígido e que as forças de atrito são desprezíveis, obtém-se as seguintes equações dinâmicas:

$$\left(m + \frac{2J_w}{R^2}\right)\dot{v}_R = F_R \tag{11}$$

$$\left(J_R + 2J_w \frac{L^2}{R^2}\right)\dot{\omega}_R = \tau_R \tag{12}$$

Onde m é a massa do robô, J_R é o momento de inércia no eixo perpendicular ao plano xy, J_w é o momento de inércia das rodas e das massas girantes acopladas aos eixos dos motores e F_R e τ_R são respectivamente a força e o torque gerados pelos motores.

Os esforços F_R e τ_R possuem duas componentes: a da roda esquerda e da roda direita. Assumindo que os motores são iguais e sabendo que o torque de um motor elétrico é função da constante de torque K_T e da corrente elétrica, tem-se:

$$F_R = F_D + F_E = \frac{\tau_D}{R} + \frac{\tau_E}{R} = \frac{K_T}{R}(i_D + i_E) \tag{13}$$

$$\tau_R = \tau_D - \tau_E = K_T(i_D - i_E) \tag{14}$$

Onde F_D e τ_D são a força e o torque aplicados no eixo da roda direita, F_E e τ_E no eixo da roda esquerda, i_D e i_E são as correntes elétricas nos circuitos dos motores das rodas direita e esquerda respectivamente.

Desconsiderando a dinâmica dos circuitos elétricos dos motores e assumindo que os dois são iguais, suas correntes são dadas pelas seguintes expressões:

$$i_D = \frac{V_D}{R_m} - \frac{\omega_D}{R_m K_v} \tag{15}$$

$$i_E = \frac{V_E}{R_m} - \frac{\omega_E}{R_m K_v} \tag{16}$$

Onde V_D e V_E são as tensões elétricas aplicadas nos motores direito e esquerdo respectivamente, R_m é a resistência elétrica dos circuitos dos motores e K_v é a constante de velocidade dos motores.

Substituindo as equações (15) e (16) em (13) e (14), obtém-se as equações que descrevem a dinâmica do robô:

$$\left(m + \frac{2J_w}{R^2}\right)\dot{v}_R = -\frac{2K_T}{R^2 K_v R_m} v_R + \frac{K_T}{R R_m}(V_D + V_E) \tag{17}$$

$$\left(J_R + 2J_w \frac{L^2}{R^2}\right)\dot{\omega}_R = -\frac{2K_T L}{R K_v R_m}\omega_R + \frac{K_T}{R_m}(V_D - V_E) \tag{18}$$

Nota-se que as equações (8), (9), (10) juntamente com as equações (17) e (18) formam um modelo para o robô de quinta ordem. Tais equações dinâmicas, no

entanto, podem ser simplificadas ao verificar-se que os termos de inércias são consideravelmente menores em relação aos demais termos presentes na equação.

$$\left(m + \frac{2J_w}{R^2}\right)\dot{v}_R \ll \frac{2K_T}{R^2 K_v R_m} \; e \; \frac{K_T}{R R_m} \tag{19}$$

$$\left(J_R + 2J_w \frac{L^2}{R^2}\right) \ll \frac{2K_T L}{R^2 K_v R_m} \; e \; \frac{K_T}{R_m} \tag{20}$$

O significado prático das simplificações acima é de que os motores, e consequentemente as rodas, respondem instantaneamente a qualquer variação da tensão aplicada, já que o valor de sua inércia é relativamente pequeno. Logo, as equações dinâmicas do modelo resultam em:

$$v_R = \frac{RK_v}{2}(V_D + V_E) \tag{21}$$

$$\omega_R = \frac{RK_v}{2L}(V_D - V_E) \tag{22}$$

Substituindo as equações (21) e (22) em (8), (9) e (10), determina-se as equações que regem o movimento robô:

$$\dot{x} = \frac{RK_v}{2}(V_D + V_E)\cos\emptyset \tag{23}$$

$$\dot{y} = \frac{RK_v}{2}(V_D + V_E)\sin\emptyset \tag{24}$$

$$\dot{\emptyset} = \frac{RK_v}{2L}(V_D - V_E) \tag{25}$$

Nota-se desta vez que o modelo simplificado é de terceira ordem, facilitanto o controle de posição.

6. Circuito eletrônico

A implementação das estratégias de controle e navegação implementadas nesse projeto foram programadas no microprocessador BeagleBone Black. As conexões a este, por sua vez, e os elementos eletroeletrônicos e eletromecânicos do *QuickBot* – motores, sensores infravermelho, encoder e baterias. O circuito foi baseado em (Egerstedt, 2015).

Primeiramente, a ponte H SN754410 é ligada à protoboard juntamente com o switch e os DC Jacks.

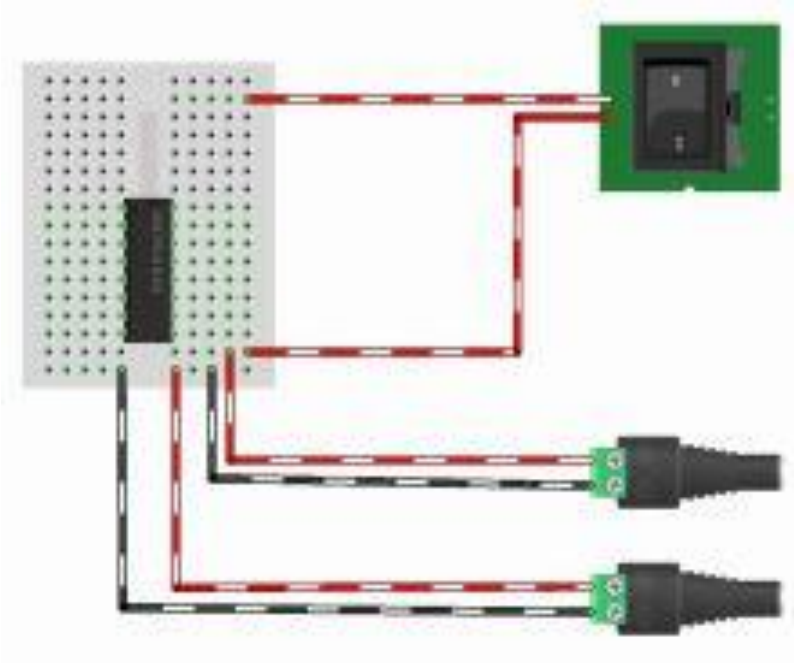

Figura 8 - Ponte H, Switch e DC Jacks (Fonte: (Egerstedt, 2015))

Subsequentemente, os polos positivos do motor da roda direita e esquerda são conectados, respectivamente, aos pinos 3 e 11 da Ponte H. Os polos negativos, por sua vez, aos pinos 6 e 14. Dessa forma, suas velocidades e sentido de rotação podem ser controlados.

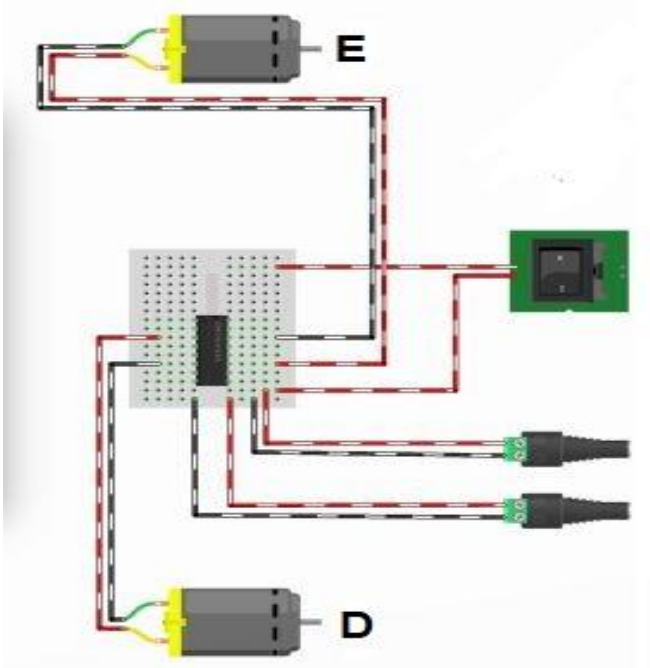

Figura 9 - Motores direito (D) e esquerdo (E) conectados à Ponte H (Fonte: (Egerstedt, 2015)))

Em seguida, o regulador de tensão L7805CV e o diodo 1N4007 são adicionados à protoboard. O regulador de tensão tem como função reduzir a tensão de 12V proveniente das baterias para 5V, o que é necessário para alimentar o Beaglebone Black. O diodo, por sua vez, liga o switch ao input do regulador, permitindo o fluxo de corrente das baterias para o regulador.

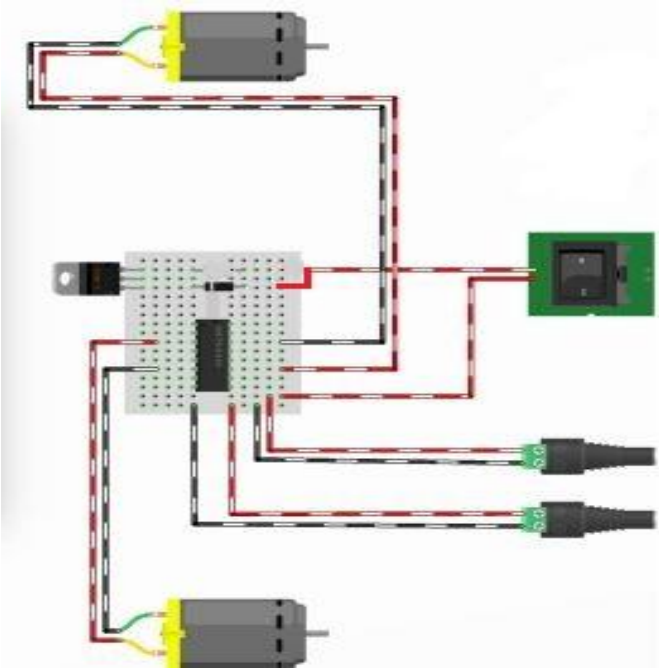

Figura 10 - Regulador de tensão e diodo (Fonte: (Egerstedt, 2015)))

Depois, a fim de diminuir possíveis ruídos no sistema, um capacitor de $100\mu F$ é adicionado entre a entrada e o terra do regulador de tensão, e outros dois de $0{,}1\mu F$ e $10\mu F$ entre a saída e o terra. O circuito resultante até o momento é o seguinte:

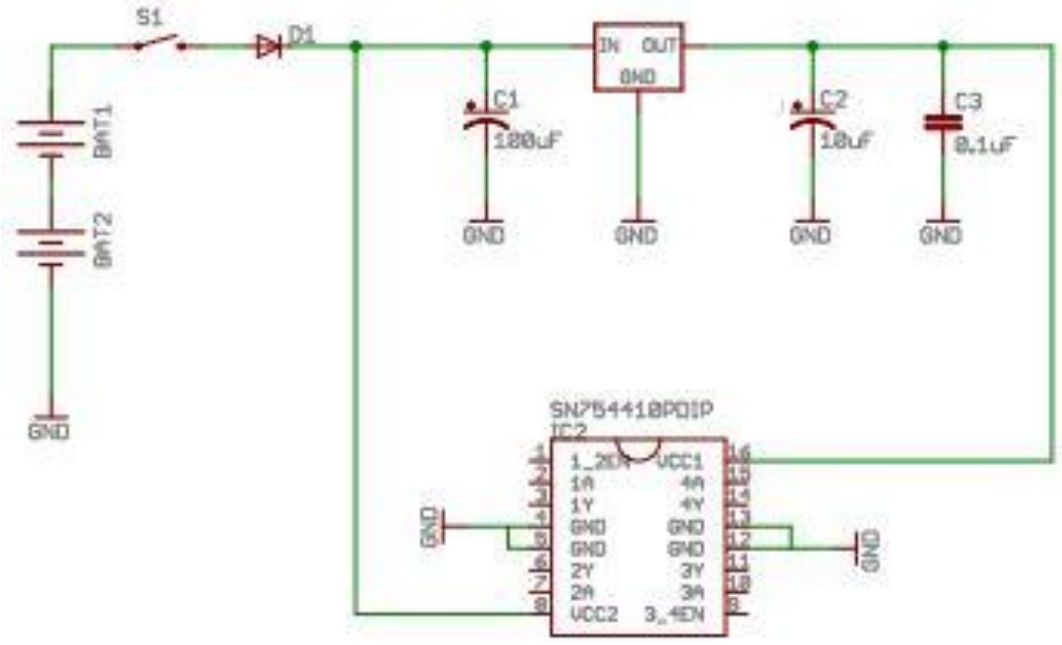

Figura 11 - Circuito intermediário (Fonte: (Egerstedt, 2015)))

Posteriormente, a Ponte H é alimentada tanta com a tensão de 12V como com a tensão de 5V através dos pinos 8 e 16 respectivamente. Além disso, o terra do regulador de tensão e da Ponte H são ligados ao terra da bateria.

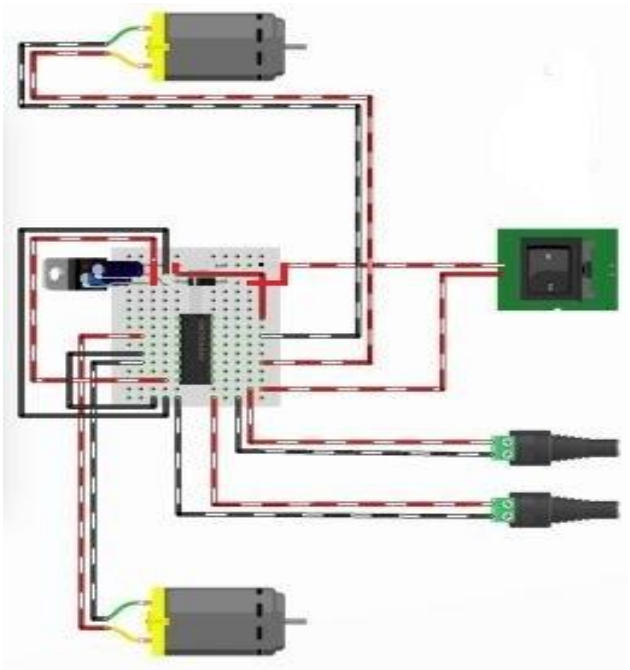

Figura 12 - Alimentação da Ponte H e aterramento (Fonte: (Egerstedt, 2015)))

Em seguida, as entradas de controle dos motores direito e esquerdo são conectadas entre a Ponte H e o BBB. Os pinos P8-10, P8-12 e P9-14 são ligados ao pino 2, pino 7 e pino 1 da Ponte H respectivamente e controlam o motor da roda direita. Os pinos P8-14, P8-16 e P9-16 são ligados ao pino 15, pino 10 e pino 9 da Ponte H respectivamente e controlam o motor da roda esquerda. Os 6 pinos do BBB acima mencionados são pinos de entrada e saída digitais GPIO (General Porpuse Input/Output). São basicamente portas programáveis que serão usadas para transmitir o sinal de PWM dos motores. Além disso, a alimentação do BBB é feita conectando-se o pino P9-05 do BBB ao pino 16 da Ponte H (VCC 5V) e o pino P9-01 do BBB ao terra da bateria.

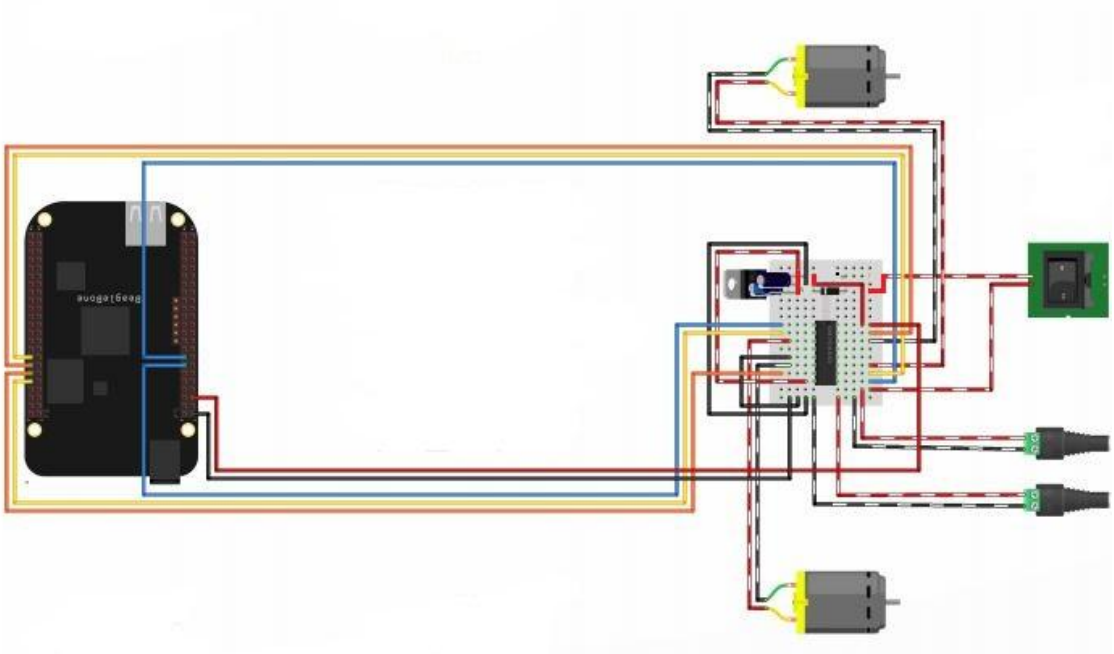

Figura 13 - Conexão BBB e Ponte H e alimentação (Fonte: (Egerstedt, 2015)))

Subsequentemente, o encoder e os 5 sensores infravermelhos são conectados ao microprocessador. Para isso, uma nova protoboard é usada para maior disponibilidade de conectores. O encoder possui 6 pinos, sendo 3 para cada roda – VCC, terra e saída de leitura de dados. Os sensores infravermelhos possuem 3 pinos cada – VCC, terra e saída de leitura de dados. Os fios conectados ao último pino citado, tanto do encoder quanto dos sensores infravermelhos, são ligados individualmente a um circuito de dois resistores de $10k\Omega$ e $20k\Omega$ em série. Esse circuito serve como um divisor de tensão, de forma a dividí-la por três. Tal procedimento é imprescindível, uma vez que as entradas analógicas do BBB possuem uma voltagem máxima de 1,8V.

Assim sendo, a saída de dados do encoder da roda direita e esquerda são conectados respectivamente às entradas P9-37, P9-39 do BBB. As dos sensores infravermelhos traseiro-direito, frontal-direito, frontal, frontal-esquerdo e traseiro-esquerdo são conectados respectivamente às entradas P9-33, P9-35, P9-36, P9-40 e P9-38 do BBB. Todas as entradas acima mencionadas são analógicas. Além disso, a nova protoboard também é alimentada com 5V proveniente do pino 16 da Ponte H e aterrada no terra da bateria.

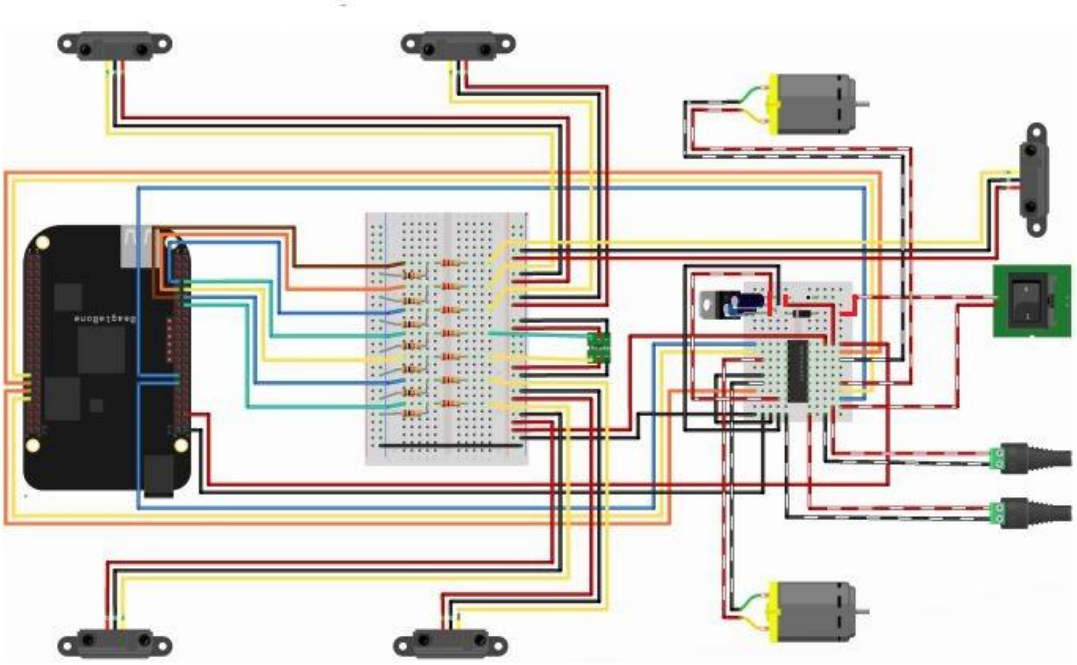

Figura 14 - Circuito eletrônico (Fonte: (Egerstedt, 2015)))

7. Sistema de localização

7.1 Processamento de imagem

O programa utilizado para desenvolvimento do software de identificação dos estados do robô foi o programa MatLab e faz uso de funções da "Computer Vision System Toolbox", bastante utilizada para projetos com este fim. A aquisição das imagens, por sua vez, foi feita a partir de uma câmera "Logitech HD WebCam C270", aparelho de aquisição de imagens de baixo custo, com resolução de 1280 x 720 pixels e fotos com resolução de até 3 MPixel, além de taxa de aquisição de 30 frames por segundo.

As imagens adquiridas a partir da câmera são obtidas em escala RGB (sigla inglesa para definir "vermelho, verde e azul"). Esta escala define a intensidade luminosa que um pixel em uma imagem é captado, neste caso, em três tonalidades a esta escala associada. O ambiente de testes para estes está representado pela figura 2. Foi-se escolhido para este projeto convenientemente a conversão deste sistema de representação para o HSV (expressão inglesa para definir "matiz, saturação e valor"). Em teoria, caso a imagem esteja representada em HSV, o primeiro componente, correspondente à matiz, pode ser escolhido de forma a representar puramente a cor. Esta, por sua vez, pode ser utilizada para cálculo de *thresholding* da imagem.

Devido à construção do robô utilizado neste projeto, o microprocessador assim como o suporte de baterias montados em sua porção superior obstruem a visão superior do mesmo. Assim, optou-se por confeccionar um cartão de mesmo tamanho a fim de cobri-lo. Esta decisão possibilitou uma melhor identificação. A ideia principal utilizada neste projeto consiste em utilizar marcadores assim como um cartão de apenas uma cor afixados à sua parte superior.

O software para reconhecimento da imagem, em linhas gerais, pode ser descrito da seguinte forma:

1. Inicialização do sensor externo, neste caso, da câmera acoplada ao teto do local para captação de imagens por meio do modo de captura de vídeo;

2. Configuração pelo usuário dos valores de "*thresholding"* correspondentes ao robô, marcadores anterior e posterior. Esta etapa consiste em uma conversão

para o espaço de cores HSV assim como ajuste de parâmetros a fim de limiarizar a imagem de acordo com o recinto em estudo;

3. Filtragem da imagem de acordo com as cores pré-calibradas;
4. Processamento da imagem e busca do robô dentre as regiões de interesse encontradas. Nesta etapa, toma-se como hipótese que o robô consiste na maior da imagem;
5. Caso ele for encontrado e estiver dentro do intervalo de segurança estipulado para a área de referência, busca-se os marcadores traseiros e frontais a partir do mesmo algoritmo de identificação do robô;
6. Caso não, e esta seja maior que o valor estipulado, o programa envia uma mensagem de erro ao usuário comunicando que o sistema, neste caso o robô, perdeu o sinal de *tracking*;
7. Por fim, caso esta seja menor que a área encontrada, o programa envia uma mensagem à tela e avisa o usuário que o robô deixou o campo de visão da câmera;

Devido ao baixo custo do sensor utilizado neste projeto, tem-se associado também um alto nível de ruído. Para tal, utilizou-se como hipótese o valor de maior blob na imagem.

Com mesmo objetivo, faz-se necessário a aplicação de um operador morfológico de fechamento, entretanto, para preencher “buracos” e lacunas vagas a fim de melhor identificação do objeto em questão. O operador morfológico de fechamento consiste na aplicação em sequência de dois operadores primários. Estes são o de dilatação e posteriormente de erosão. O primeiro expande o objeto em questão e o segundo desbasta-o. A partir de um elemento estruturador de tamanho a ser ajustado, estruturas como “buracos” e concavidades podem ser assim preenchidos. Como vantagem, tem-se que o objeto mantém seu tamanho original, o que também possibilita sua melhor visibilidade. Estruturas menores que o elemento estruturador como, por exemplo, ruído, são eliminados da imagem.

7.2 Odometria

Como já mencionado, o robô em questão possui duas rodas motorizadas responsáveis por sua propulsão. Através do encoder diretamente ligado a elas, é

possível obter a posição do robô ao longo da navegação a partir de sua posição e orientação iniciais. A ideia principal da odometria é a incrementação gradual da informação obtida pelos encoders ao longo do tempo.

Assumindo que o robô esteja percorrendo um arco com velocidade angular e linear constantes, a roda direita e esquerda percorrerão uma distância D_d e D_e respectivamente.

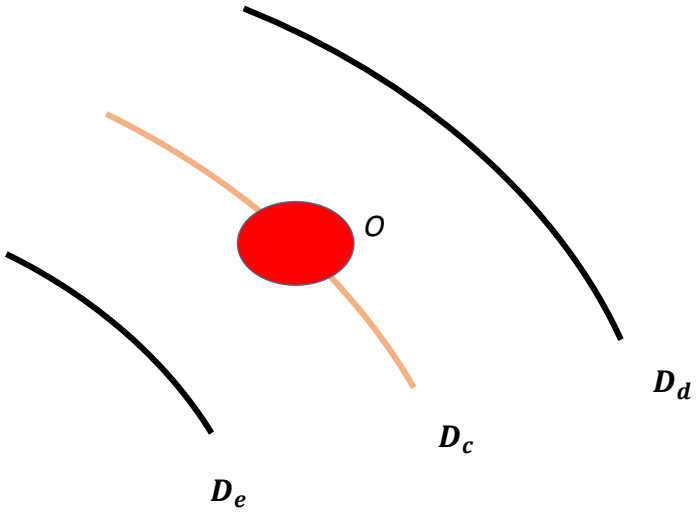

Figura 15 - Distância percorrida pelas rodas

Como pode-se observar na Figura 15, a roda direita percorre uma distância maior que a esquerda. D_c representa a distância percorrida pelo centro O do robô, ponto de referência de sua posição. Dessa forma tem-se que:

$$D_c = \frac{(D_e + D_d)}{2} \tag{26}$$

Para obter-se a posição do robô em coordenadas cartesianas e assim conseguir medir qual a distância percorrida, adota-se a seguinte relação:

$$x' = x + D_c * \cos \emptyset \quad (27)$$

$$y' = y + D_c * \cos \emptyset \quad (28)$$

$$\emptyset' = \emptyset + \frac{(D_d - D_e)}{L} \quad (29)$$

Onde x e y são as coordenadas cartesianas do robô e $\emptyset$ é o ângulo de rotação do robô em relação ao seu centro. x', y' e $\emptyset'$ são as posições e orientação incrementadas e L representa a distância entre eixos. Para se obter as distâncias percorridas pelas rodas, três variáveis são necessárias: (1) número de dentes da roda dentada acoplada às rodas dos motores (N); (2) raio das rodas (R); (3) o número de impulsos que o encoder recebeu referente ao número de dentes que passaram pelo sensor em determinado espaço de tempo (ΔA).

Assim, tem-se que:

$$D = 2\pi R * \frac{\Delta A}{N} \quad (30)$$

Tendo-se x', y' e $\emptyset'$ em cada instante de tempo discretizado, pode-se mensurar a localização do robô.

8. Sistema de Navegação

A navegação de um robô autônomo consiste em fazê-lo movimentar-se em um ambiente de maneira controlada e segura. Isso exige uma interação entre ele e o ambiente através de sensores.

Modos de navegação consistem, na prática, de controladores que implementam uma rotina desejada para que o robô realize a específica tarefa a que foi designado. Dois modos excludentes e suficientes para garantir um controle básico do robô são os seguintes (Egerstedt, 2015):

- *Go-to-goal* (GTG) – Ir para o destino;
- *Avoid-obstacle* (AO) – Evitar obstáculo;

Partindo das equações (3), (4), (5), (6) e (7) referentes ao modelo cinemático do robô, pode-se detalhar com mais precisão as estratégias de navegação acima citadas.

8.1 Goal-to-goal (GTG)

Primeiramente, a fim de fazer com que o robô ande em uma determinada direção, é preciso determinar as coordenadas de seu destino (x_g, y_g).

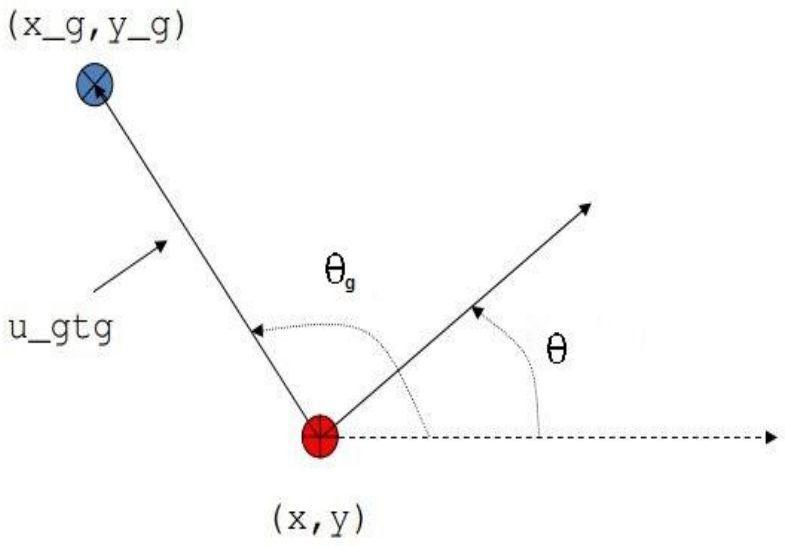

Figura 16 - Localização do robô e seu destino (Fonte: (Egerstedt, 2015)))

Esta, por sua vez, é expressa pelo ângulo θ_g e calculada da seguinte forma:

$$\theta_g = \tan^{-1}\left(\frac{y_g - y}{x_g - x}\right) \tag{31}$$

Seja a velocidade linear do robô constante, por meio de um controlador PID, é possível obter-se a velocidade angular com a qual o robô gira em torno de seu centro, garantindo-se, em $t \to \infty$, estabilidade estacionária:

$$\omega_R = PID(e) = K_P e + K_I \int_0^t e(\tau)d\tau + K_D \dot{e} \tag{32}$$

Para implementação, entretanto, é necessário discretizar a estratégia da equação 11. Assim sendo, os termos integrativo e derivativo são aproximados da seguinte forma:

$$\int_0^t e(\tau)d\tau = e + w.\Delta t \tag{33}$$

E

$$\dot{e} = \frac{e_k - e_{k-1}}{\Delta t} \tag{34}$$

A malha de controle resultante referente à estratégia “Go-to-Goal” é apresentada a seguir:

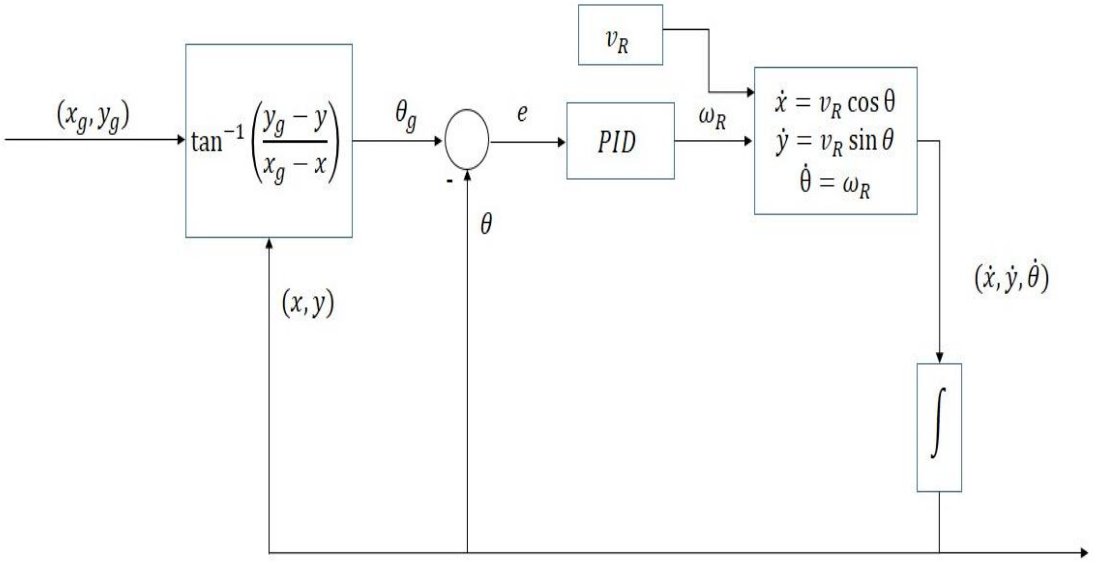

Figura 17 - Malha de controle (Fonte: (Egerstedt, 2015)))

8.2 Avoid-Obstacles (AO)

A fim de que o robô navegue sem entrar em contato com nenhum obstáculo que esteja em seu caminho, é preciso primeiramente que este interaja com o ambiente através de sensores. No caso do robô usado neste projeto, cinco sensores infravermelhos estão instalados separadamente no chassi, como mostra a seguinte figura:

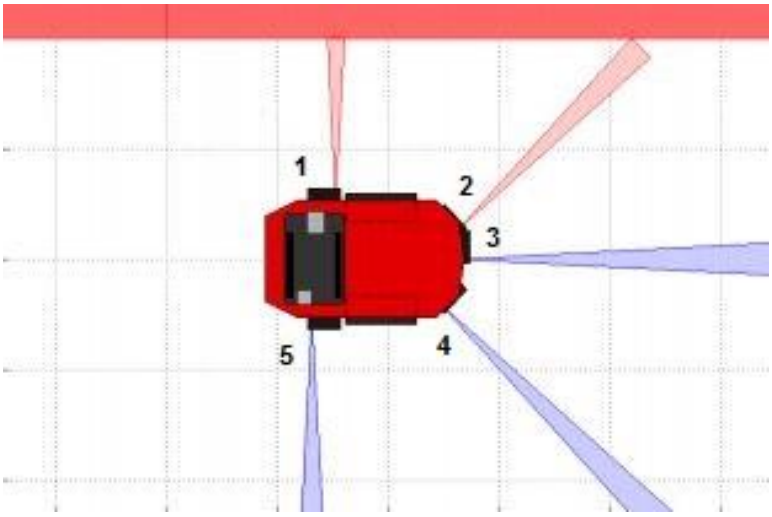

Figura 18 - Sensores infravermelho (Fonte: (Egerstedt, 2015)))

A implementação da rotina para evitar obstáculos é dividida em três passos (Egerstedt, 2015):

1 – Determinar a posição do obstáculo no ambiente. Para isso é preciso:

- Transformar o valor da distância detectada pelo sensor em um ponto no sistema de coordenadas do robô (em inglês, *robot frame*);
- Transformar o ponto do sistema de coordenadas do robô para o sistema de coordenadas absoluto do ambiente (em inglês, *world frame*);

2 – Calcular um vetor que aponte para o sentido contrário ao do obstáculo;

3 – Usar um controlador PID para rotacionar o robô na direção do vetor resultante obtido na etapa 2.

A distância detectada por um sensor é dada em um valor numérico absoluto. Esta, por sua vez, é dada a partir de um sistema de coordenadas acoplado ao respectivo i-ésimo sensor, mostrado no exemplo da figura abaixo.

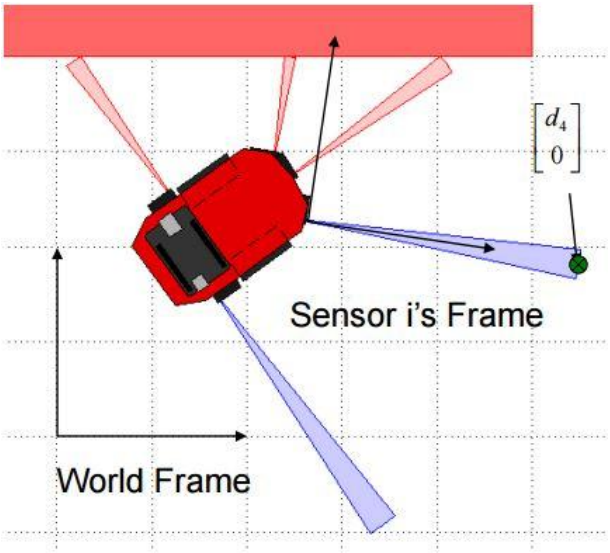

Figura 19 - Distância d_4 captada pelo sensor 4 (Fonte: (Egerstedt, 2015)))

As coordenadas do ponto identificado são, no respectivo sistema de coordenadas, igual a $(d_i, 0)$. Para que estas sejam transformadas no sistema de coordenadas do robô (RF), é necessário multiplicá-la por uma matriz de rotação e translação.

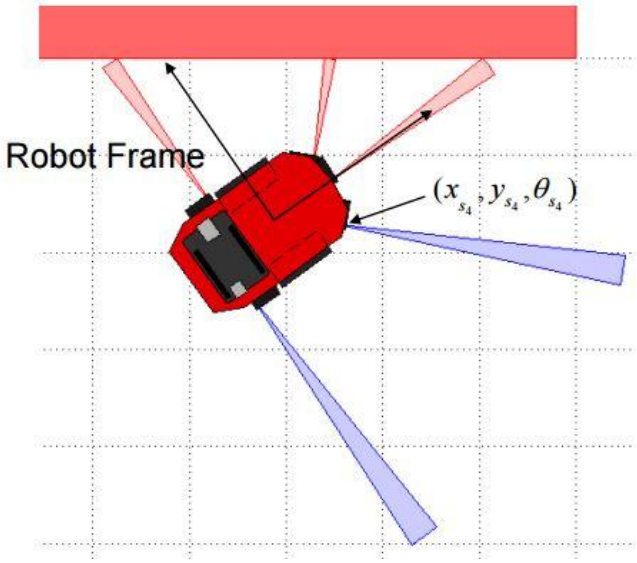

Figura 20 - Sistema de coordenadas do robô (robot frame) (Fonte: (Egerstedt, 2015)))

$$\begin{bmatrix} x_{d_i} \\ y_{d_i} \\ 1 \end{bmatrix}_{RF} = R\left(x_{s_i}, y_{s_i}, \theta_{s_i}\right)_{RF} \begin{bmatrix} d_i \\ 0 \\ 1 \end{bmatrix} \tag{35}$$

$$\begin{bmatrix} x_{d_i} \\ y_{d_i} \\ 1 \end{bmatrix}_{RF} = \begin{bmatrix} \cos\theta_{s_i} & -\sin\theta_{s_i} & x_{s_i} \\ \sin\theta_{s_i} & \cos\theta_{s_i} & y_{s_i} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_i \\ 0 \\ 1 \end{bmatrix} \tag{36}$$

Sendo $\left(x_{s_i}, y_{s_i}, \theta_{s_i}\right)_{RF}$ as coordenadas do sensor i em RF.

Em seguida, elas devem ser transformadas para o sistema de coordenadas do ambiente (WF).

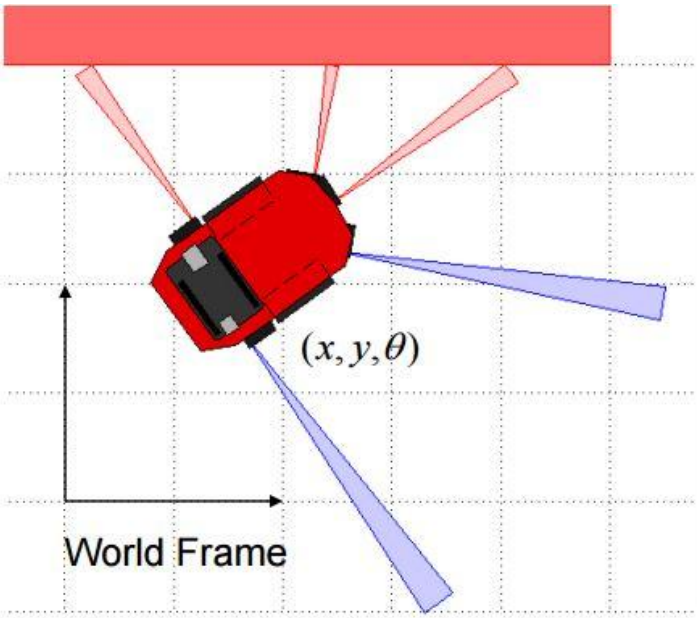

Figura 21 - Sistema de coordenadas do ambiente (Fonte: (Egerstedt, 2015)))

$$\begin{bmatrix} x_{d_i} \\ y_{d_i} \\ 1 \end{bmatrix}_{WF} = R(x, y, \theta)_{WF} \begin{bmatrix} x_{d_i} \\ y_{d_i} \\ 1 \end{bmatrix}_{RF} \tag{37}$$

$$\begin{bmatrix} x_{d_i} \\ y_{d_i} \\ 1 \end{bmatrix}_{WF} = R(x, y, \theta)_{WF} R\left(x_{s_i}, y_{s_i}, \theta_{s_i}\right)_{RF} \begin{bmatrix} d_i \\ 0 \\ 1 \end{bmatrix} \tag{38}$$

A partir deste ponto, calcula-se o vetor $\vec{u}_{AO} = \sum_{i=1}^{5}(d_{i_{WF}} - (x, y)_{WF})$, em que $(x, y)_{WF}$ define a posição do robô. Este vetor sempre apontará em uma direção contrária ao objeto tido como obstáculo. Finalmente, calcula-se a direção para qual $\vec{u}_{AO}$ aponta através do ângulo $\theta_{ao} = \tan^{-1}\left(\frac{y_{ao}-y}{x_{ao}-x}\right)$. O erro residual $e = \theta_{ao} - \theta$ e a velocidade angular necessária para rotacionar o robô são controlados, da mesma forma, através de um controlador PID:

$$\omega_R = PID(e) = K_P e + K_I \int_0^t e(\tau)d\tau + K_D \dot{e} \tag{39}$$

Uma solução mais aprimorada para o controle apresentado acima consiste na soma ponderada de $\vec{u}_{AO}$. O novo vetor resultante dá-se:

$$\overrightarrow{u_{AO}} = \sum_{i=1}^{5} k_i (d_{i_{WF}} - (x, y)_{WF}) \quad (40)$$

Com essa abordagem, atribui-se diferentes relevâncias a cada vetor, gerando assim, uma maior ou menor influência sobre o vetor resultante. Isso é interessante quando se trata de evitar colisões. O sensor 3 (frontal), por exemplo, deve apresentar uma importância maior que os laterais, uma vez que o robô somente realiza movimento translacional – causa de uma eventual colisão – na mesma direção do sensor frontal.

8.3 Arquitetura de navegação com GTG e AO

Existem duas maneiras de projetar a navegação do robô no meio usando os dois controladores apresentados. Uma delas é alternar entre as duas rotinas dependendo das condições que o robô encontrar (*Hard-Switching*). Por exemplo, enquanto o robô não detectar nenhum obstáculo, ele permanece na rotina *Go-to-Goal*. No momento em que os sensores perceberem a possibilidade de uma possível colisão, ocorre uma transição para a rotina *Avoid-obstacle*. Assim que a ameaça de colisão sumir, a rotina de navegação volta à estratégia original. Tal arquitetura apresenta algumas desvantagens, dentre elas:

- Navegação pouco eficiente: o caminho escolhido é, em geral, não otimizado;
- Navegação “truculenta”: a troca sucessiva entre as duas rotinas leva a um aspecto de percurso não contínuo.
- *Alternância* com alta frequência entre estratégias: este fenômeno pode danificar o de hardware do robô.
- *Instabilidade* – ambas as rotinas de navegação consistem em controladores estáveis. Porém, a alternância entre eles modos pode, em certos casos, gerar uma navegação instável.

A segunda maneira consiste em um controlador misto (em inglês, *blended*) por meio da implementação simultânea das rotinas *Go-to-Goal* e *Avoid-obstacle*. Ambos

os controladores são acionados ao mesmo tempo e a direção na qual o robô deve seguir, $u_{GTF,AO}$, é dado por:

$$\vec{u}_{GTF,AO} = \alpha \frac{\vec{u}_{AO}}{||u_{AO}||} + (1-\alpha)\frac{\vec{u}_{GTG}}{||u_{GTG}||} \quad (41)$$

Onde $\vec{u}_{GTG} = (x_g, y_g) - (x, y)$ e $0 \leq \alpha \leq 1$. Assim, da mesma forma que para a estratégia “Avoid Obstacle”, implementa-se um controlador PID a fim de controlar sua velocidade angular.

Os vetores $\vec{u}_{AO}$ e $\vec{u}_{GTG}$ foram normalizados de forma que seu tamanho real não tenha influência na ponderação feita. Por exemplo, se o objeto estiver muito longe, $\frac{\vec{u}_{GTG}}{||u_{GTG}||}$ terá uma participação desprocionalmente maior, independente do valor de α. Através do último, é possível arbitrar qual rotina deve ter mais influência no controlador misto. Este tem como vantagem uma navegação contínua e suave. Entretanto, uma desvantagem deste reside no fato de que a estabilidade do sistema não é garantida, dependendo do valor de α.

8.4 Follow Wall

Por meio de ambos controladores apresentados (*Go to Goal* e *Avoid Obstacle*), é possível conceber uma arquitetura de navegação que guie o robô de um ponto a outro e que desvie de obstáculos. A estratégia é garantida somente no caso de objetos chamado convexos. Define-se um objeto não-convexo como todo aquele cujo, para todo e qualquer segmento de reta definido por dois pontos em seu interior, este pode estar ou não contido em seu interior. Exemplo deste tipo de obstáculo é:

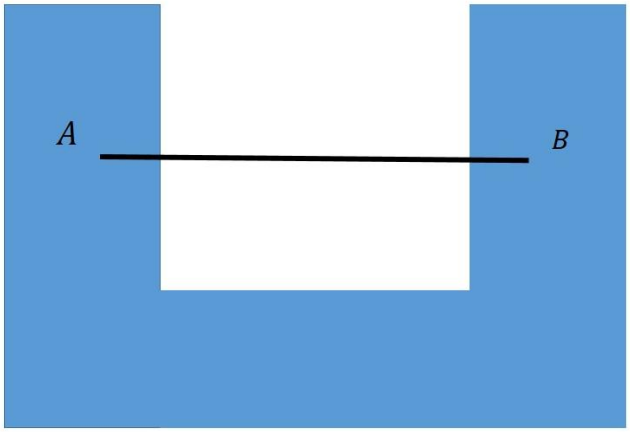

Figura 22 - Objeto não-convexo

Devido ao fato de não ter sido implementada uma estratégia que leve o robô a uma direção oposta ao do seu destino, neste caso, na perpendicular, este permanece preso no interior do obstáculo. Frente a isso, a rotina *Follow Wall* (segue parede) pode solucionar este impecilho.

Essa rotina consiste em três passos:

1 – Estimar uma seção da parede (obstáculo) usando os sensores infravermelhos;

2 – Calcular um vetor tangencial e perpendicular à ela;

3 – Somar estes e rotacionar o robô em sua direção.

Primeiramente, o robô deve utilizar os dois sensores laterais 1 e 2. A partir destes dois pontos, obtém-se o vetor $\vec{u}_{fw,t} = (p_2 - p_1)$ tangencial à parede, os quais p_1 e p_2 são os pontos captados respectivamente pelos sensores 1 e 2.

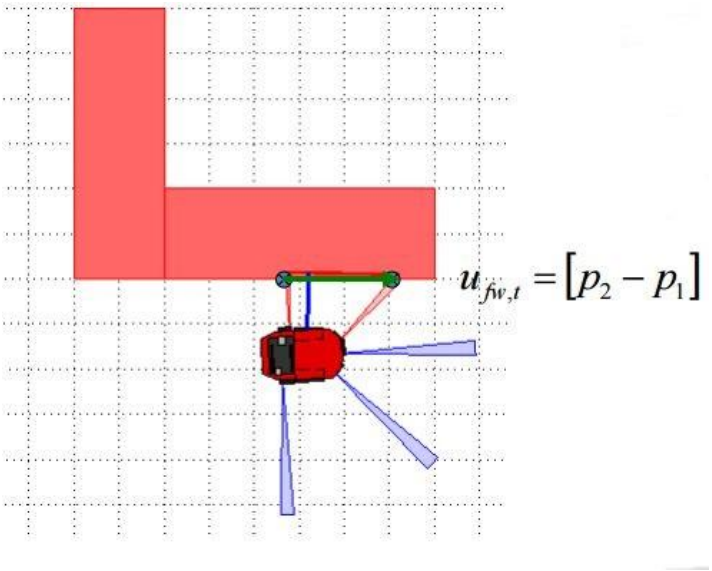

Figura 23 - Vetor tangencial (Fonte: (Egerstedt, 2015))

Em seguida, deve-se calcular o vetor perpendicular $\vec{u}_{fw,p}$, entre o ponto mais próximo da parede e o robô, dado pela seguinte expressão:

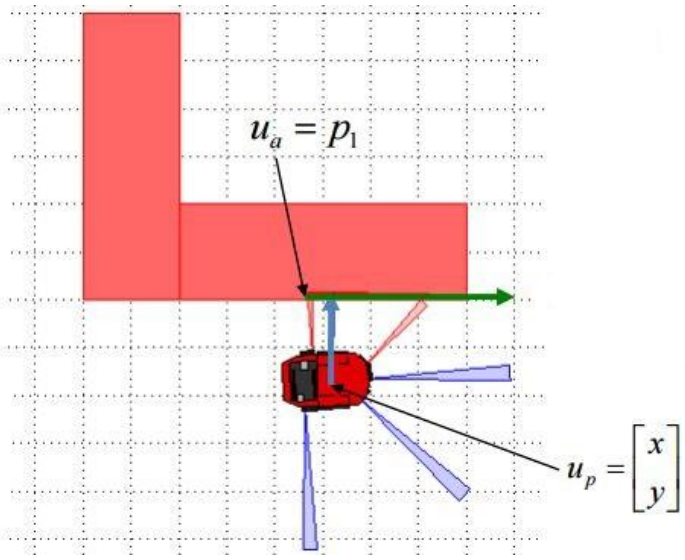

Figura 24 - Vetor perpendicular (Fonte: (Egerstedt, 2015))

Onde u_a é o ponto captado pelo sensor 1 e u_p é a posição do robô.

Subsequentemente é preciso gerar um vetor também perpendicular à parede, mas que aponte para a direção contrária. Além disso, tal vetor deve ser normalizado e multiplicado pela distância d_{fw} desejada que o robô deve se manter da parede durante a rotina.

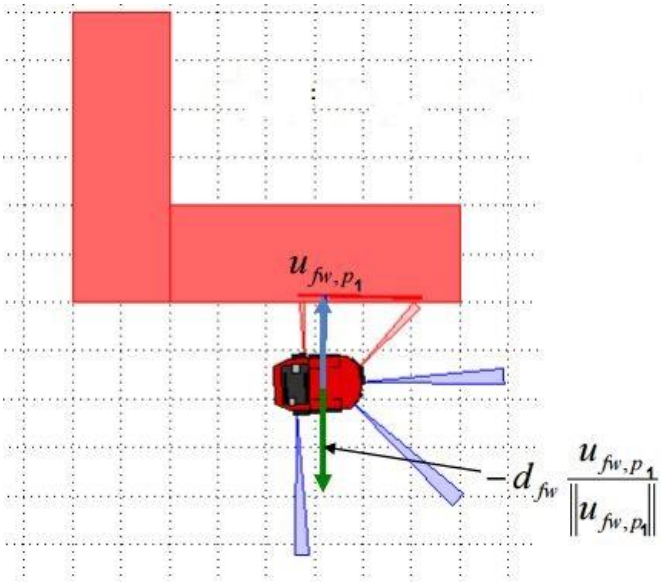

Figura 25 - Vetor perpendicular na direção contrária (Fonte: (Egerstedt, 2015))

O vetor perpendicular resultante $\vec{u}_{fw,p}$ é dado pela seguinte expressão:

$$\vec{u}_{fw,p} = \vec{u}_{fw,p1} - d_{fw}\frac{\vec{u}_{fw,p1}}{||\vec{u}_{fw,p1}||} \tag{42}$$

Tendo tanto o vetor tangencial ao objeto quanto o perpendicular, é possível obter o vetor $\vec{u}_{fw}$ que guiará a direção do robô através de uma simples soma:

$$\vec{u}_{fw} = \alpha\frac{\vec{u}_{fw,t}}{||\vec{u}_{fw,t}||} + \beta\frac{\vec{u}_{fw,p}}{||\vec{u}_{fw,p}||} \tag{43}$$

Onde α e β são constantes. Finalmente, basta usar um controlador PID para rotacionar o robô na direção de $\vec{u}_{fw}$. Uma problemática apresentada por tal rotina é mo caso de o objeto apresentar quinas. A estimativa da parede, nessa situação, não condiz com a realidade como mostrado na figura 16. Por isso o valor da distância d_{fw} deve ser cuidadosamente selecionada – não tão próxima à parede – a fim de evitar uma colisão ao contornar quinas.

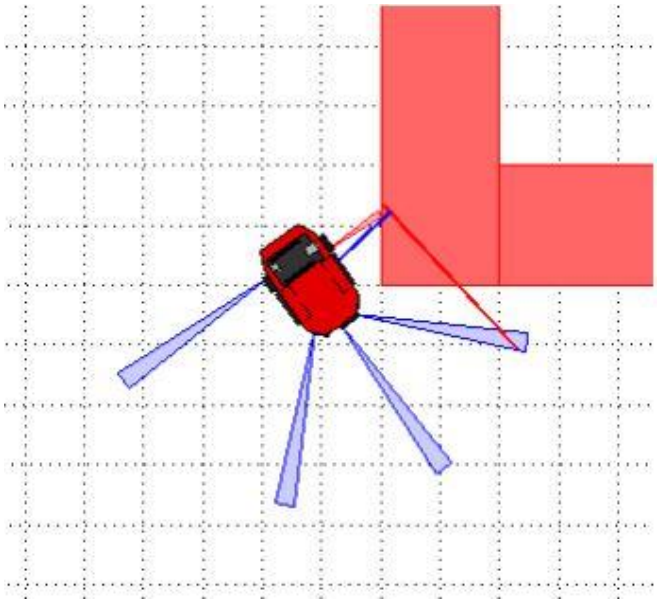

Figura 26 – Quinas (Fonte: (Egerstedt, 2015))

A fim de seguir a parede, entretanto, não foi-se determinado o sentido a se percorrer, i.e., horário ou anti-horário, em relação ao obstáculo em questão. Para isto, faz-se uso dos vetores $\vec{u}_{fw,l}$ e $\vec{u}_{fw,r}$, respectivos aos lados direito e esquerdo do robô. Estes são visíveis na figura 11.

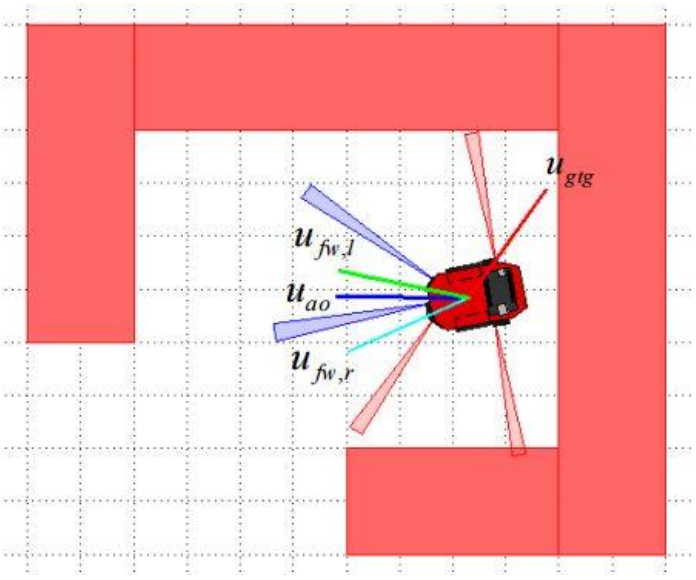

Figura 27 - Vetores dos controladores (Fonte: (Egerstedt, 2015))

A escolha do sentido dá-se por meio da solução do seguinte sistema linear:

$$\begin{bmatrix}\vec{u}_{gtg} & \vec{u}_{ao}\end{bmatrix}\begin{bmatrix}\sigma_1\\ \sigma_2\end{bmatrix} = \vec{u}_{fw,x} \quad (44)$$

O qual x para qual o sentido o robô deve navegar. indica qual o lado do vetor Esse sistema deve ser resolvido para os dois vetores $\vec{u}_{fw,l}$ e $\vec{u}_{fw,r}$. O vetor $\vec{u}_{fw}$ a ser escolhido corresponde àquele em que as soluções do sistema, σ_1 e σ_2, são ambos positivos. A expressão oriunda da álgebra linear,representa que, seja um espaço vetorial definido pelos vetores $\vec{u}_{gtg}$ e $\vec{u}_{ao}$, o vetor $\vec{u}_{fw}$ encontra-se entre se, e somente se, este for composto por uma combinação linear destes, cujos parâmetros são positivos. No caso da figura 18, o vetor escolhido seria $\vec{u}_{fw,l}$, o que faria com que o robô contorne a parede no sentido anti-horário.

8.5 Arquitetura de navegação com GTG, AO e FW

Definidas as três estratégias de navegação do robô, faz-se necessárias, por fim, a concepção de um algoritmo de forma a uní-las. Neste projeto, faz-se uso de uma máquina de estados para conectar todas as estratégias propostas e que o robô navegue apropriadamente

Tendo as três rotinas de navegação apresentadas, o robô é perfeitamente capaz de atingir seu objetivo – chegar a um destino – sem colidir com nenhum tipo de obstáculo. O último passo do projeto de navegação é arquitetar uma máquina de estados que conecte essas rotinas e permita que o robô navegue apropriadamente.

Para isso, algumas variáveis discretas (*flags*) precisam ser criadas a fim de sinalizar o instante em que a mudança de estados, ou seja, de controladores deve ser feita. Dentre elas, tem-se:

Flag	***true***	***false***
at_goal	Se o robô estiver a uma distância $d < d_{stop}$ do objeto	Se o robô estiver a uma distância $d > d_{stop}$ do objeto
at_obstacle	Se o robô estiver a uma distância $d < d_{at\_obs}$ do objeto	Se o robô estiver a uma distância $d > d_{at\_obs}$ do objeto
unsafe	Se o robô estiver a uma distância $d < d_{unsafe}$ do objeto	Se o robô estiver a uma distância $d > d_{unsafe}$ do objeto
progress_made	Se o robô está a uma distância $d_k < d_{k-1}$ do objeto, onde k denota o instante de tempo atual	Se o robô está a uma distância $d_k > d_{k-1}$ do objeto.
sliding_left, sliding_right	Se o vetor $\vec{u}_{fw}$ estiver entre $\vec{u}_{ao}$ e $\vec{u}_{gtg}$	Se o vetor $\vec{u}_{fw}$ não estiver entre $\vec{u}_{ao}$ e $\vec{u}_{gtg}$

Tabela 1 - Flags da máquina de estados

A arquitetura de navegação é descrita pela seguinte máquina de estados:

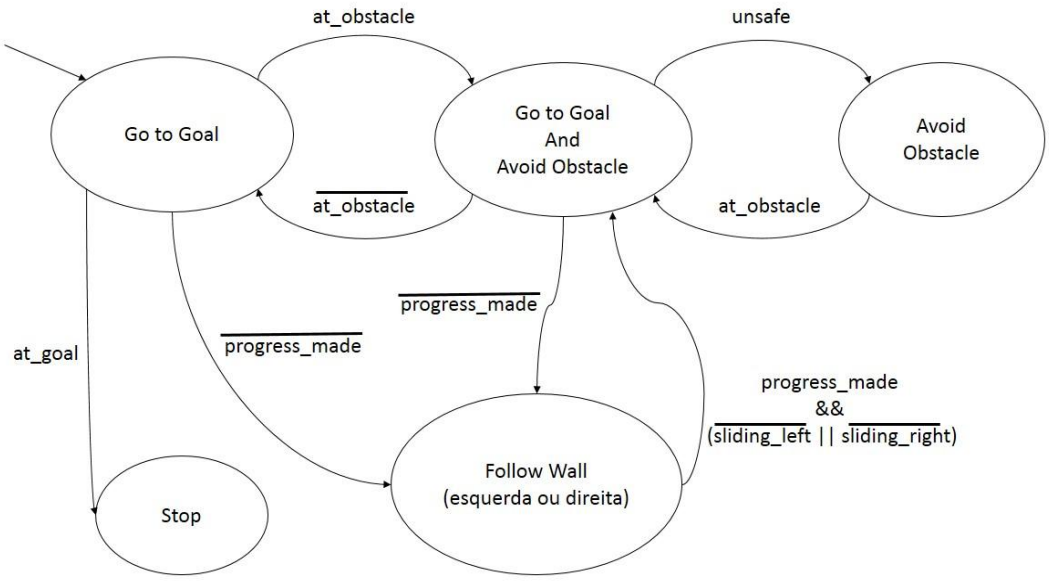

Figura 28 - Arquitetura de navegação

Analisando a estruturação da máquina de estados e suas condições de transição, observa-se os seguintes pontos:

- A rotina *Follow Wall* é somente acionada quando o robô não apresenta nenhum progresso, ou seja, a distância entre ele e o destino não está diminuido.
- A condição de saída da rotina *Follow Wall* se dá somente quando o vetor $\vec{u}_{fw}$ não está mais entre $\vec{u}_{ao}$ e $\vec{u}_{gtg}$ e o robô está progredindo em direção ao obstáculo, o que indica não ser mais necessário seguir a parede.

9. Resultados e Conclusões

Infelizmente, não obtivemos resultados concretos e empíricos do projeto proposto. Houve suscessivos problemas de hardware e software que dificultaram o andamento do trabalho. Os problemas de harware foram mais acentuados no protótipo do robô, não sendo possível obter os sinais do encoder. Já os problemas de software, se deram no processamento de imagem, no qual a posição e a orientação do robô não puderam ser identificadas de maneira consistente. Subsequentemente, não foi possível fazer uma fusão dos dados de localização (encoder e câmera).

Quanto às estratégias de navegação, as mesmas puderam ser simuladas na toolbox “Sim.I.Am” no Matlab e apresentaram resultados satisfatórios. O robô é capaz de navegar até um destino pré-determinado e ao mesmo tempo desviar de obstáculos seguindo o modelo aqui proposto.

Definitivamente, o desenvolvimento deste projeto não foi uma tarefa simples. Impecilhos advindos de componentes eletrônicos, como o microprocessador Beaglebone Black, ponte-H e encoder acabaram atrasando consideravelmente as atividades. Resultados ainda estão sendo buscado afim de demonstrar a teoria aqui apresentada.

References

Adamy, J. (2014). "Robotics and computational Inteligence" script. Darmstadt, Hessen, Germany: Shaker Verlag.

Amy Briggs, Y. L. (2006, May). Robot Navigation Using 1D Panoramic Images. *Conference on Robotics and Automation (ICRA 2006)*.

Atsushi Fujimori, P. N. (1997, AUGUST). Adaptive Navigation of Mobile Robots. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, pp. 596 - 602.

Borenstein, J. (1996, December). Measurement and Correction of Systematic Odometry Errors in Mobile R.obots. *ROBOTICS AND AUTOMATION,*.

Chieh-Chih Wang, C. T. (2003, September). Online Simultaneous Localization And Mapping with Detection And Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas. *International Coofcrentr 00 Robotics &Automation*, pp. 14 - 19.

Cornell University Courses. (2015, Jun 01). Retrieved from Kalman Filter Applications: http://www.cs.cornell.edu/courses/cs4758/2012sp/materials/mi63slides.pdf

Couto, M. S. (2009). *Localization and Navigation of Mobile Robots.* Lisbon: Department of Mechanical Engineering.

Egerstedt, M. (n.d.). Georgia Tech University: School of Electrical and Computer Engineering.

Giuseppina Gini, A. M. (2002). *Indoor Robot Navigation with Single Camera Vision.* Milano, Italy: Politecnico di Milano, piazza L. da Vinci 32.

Hansye S. Dulimarte, A. K. (1997). Mobile Robot Localization in Indoor Environment. *Pattern Recognition*, pp. 99-111.

Hartana, J. S. (2000). Sensor Data Fusion Using Kalman Filter. *Information Fusion*, (pp. 19 - 25).

Heinen, F. J. (2002, Maio). Sistema de Controle Híbrido para Robôs Autônomos. São Leopoldo, Rio Grande do Sul, Brasil.

Juan Fasola, P. E. (2005). Fast goal navigation with obstacle avoidance using a dynamic local visual model.

Manuela Veloso, J. B. (2010, May). Wifi localization and navigation for autonomous indoor mobile robots. *IEEE International Conference*, pp. 4379-4384.

Maria Cândida F. S. P. Coelho, J. M. (2003, Semptember). *Método de Calibração de Câmaras Proposto por Zhang.* Laboratorio de Optica e Mecanica Experimental.

Okutomi, S. A. (2009). A User-Friendly Method to Geometrically Calibrate Projector-Camera Systems. pp. 47 - 54.

Radhwan Ben Madhkour, M. M. (2015). A taxonomy of camera calibration and video projection correction methods. *Creative Technologies*, pp. 1 - 9.