



UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Mariam dos Passos Afonso da Conceição

**Implantação de um Controlador
Nebuloso para Navegação Autônoma
em um Robô Real Simples**

Rio de Janeiro, Brasil
Setembro de 2012

Mariam dos Passos Afonso da Conceição

Implantação de um Controlador Nebuloso para Navegação Autônoma em um Robô Real Simples

Monografia apresentada para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade Federal do Rio de Janeiro.

Orientador:
Adriano Joaquim de Oliveira Cruz
Co-orientador:
Bruno Bottino Ferreira

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Rio de Janeiro, Brasil
Setembro de 2012

**Implantação de um Controlador Nebuloso para Navegação Autônoma em um
Robô Real Simples**

Mariam dos Passos Afonso da Conceição

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Autor do Trabalho:

Mariam dos Passos Afonso da Conceição

Aprovado por:

Prof. PhD. Adriano Joaquim de Oliveira Cruz
Universidade Federal do Rio de Janeiro
Orientador

B.Sc. Bruno Bottino Ferreira
Universidade Federal do Rio de Janeiro
Co-orientador

Prof. D.Sc. Adriana Santarosa Vivacqua
Universidade Federal do Rio de Janeiro

Resumo

Implantação de um Controlador Nebuloso para Navegação Autônoma em um Robô Real Simples

Mariam dos Passos Afonso da Conceição

Robôs autônomos inteligentes são alvo de diversas pesquisas atualmente por sua aplicabilidade em diversos contextos, seja doméstico, industrial ou, ainda, em ambientes inóspitos. O objetivo deste trabalho foi implementar, testar e desenvolver algoritmos que permitissem a navegação autônoma de um robô real simples usando um controlador baseado em lógica nebulosa. Tanto as características necessárias do robô quanto o controlador em questão foram elaborados por Mota (2010). Seu intuito era utilizar sensores que pudessem ser implementados a partir de componentes comumente disponíveis. Indo além, o projeto aqui apresentado propõe a adoção de uma plataforma com baixo custo, de fácil construção e facilmente programável, baseada no Arduino Mega ADK. Também foi analisada a possibilidade de se tornar o modelo mais flexível, diminuindo as restrições impostas originalmente sobre as dimensões do ambiente. Além disso, neste texto são discutidas algumas características dos sensores empregados e as estratégias necessárias para que o robô se comportasse conforme esperado. Por fim são mostrados e analisados os resultados dos testes de navegação.

Abstract

Implementation of a Fuzzy Controller for Autonomous Navigation in a Simple Real Robot

Mariam dos Passos Afonso da Conceição

Nowadays, autonomous intelligent robots are the target of extensive research on many contexts, such as home and industrial ones, and on inhospitable environments. The goal of this work was to implement, test and develop algorithms which could allow the autonomous navigation of a simple real robot using a controller based upon fuzzy logic. The necessary features for the robot and the controller were elaborated by Mota (2010). His aim was the utilization of sensors that could be implemented with commonly available components. Beyond that, the project presented here proposes the adoption of a low cost, easily assembled and easily programmed platform, based on Arduino Mega ADK. The possibility of making the model mode flexible was analyzed, by reducing the restrictions originally imposed over the environment's dimensions. Furthermore, this text discusses some characteristics of the sensors used as well as the necessary strategies in order to make the robot behave as expected. Finally the navigation tests results are presented and analyzed.

Dedicatória

Em primeiro lugar, dedico este trabalho ao Supremo Deus e aos meus pais, Marcos Luiz e Maria Amélia, sem os quais eu não teria chegado até aqui e não seria quem sou. Meu pai me ensinou a ser forte, digna e madura desde cedo. Obrigada por ter sido o meu herói quando eu precisava de heróis. Minha mãe, a mulher mais firme e mais doce que já conheci. A pessoa que dedicou sua vida a edificar a minha, e que sempre confiou em Deus e em mim. Você foi minha primeira professora, me ensinando a ler e a escrever, e ainda hoje tem tanto a me ensinar... Obrigada por me dar forças e coragem quando eu mesma já não tinha esperança, por acreditar incondicionalmente em mim, por me dizer "você é capaz" e por estar sempre ao meu lado. Amo muito vocês!

No (quase) fim desse longo caminho, não poderia me esquecer de duas pessoas muito especiais: tia Amélia e sua filha, Nilma. Vocês me deram apoio ao longo de muitos anos, e um exemplo que levarei por toda a vida. Muito obrigada!

Também deixo minha dedicatória ao Professor Adriano Cruz. O senhor foi meu orientador não só neste projeto, mas sim em muitos momentos ao longo da graduação. Sou-lhe profundamente grata pelas oportunidades e pela confiança no meu trabalho.

Agradecimentos

Agradeço sinceramente a todos aqueles que contribuíram, ao longo dessa jornada acadêmica, para que eu chegasse até aqui. Tive a permissão de conhecer pessoas maravilhosas, e ganhei de presente amigos mais que especiais.

Meu muito obrigada aos meus grandes e divertidos parceiros de Pedro II e CEFET: Bianca, Maria, Andressa, Jorge, Diego Pupato; Leandro, Felipe, Rebeca, Fernando, Thiago, Aline, Diego Wanderley, Claudio, Gustavo e Natalia. Agradeço, também, aos queridos Professores dessa época que tanto marcaram minha vida: Inês Rocha, Wolney Malafaia, Gisela Viana, Sérgio Lima, Eliane Trigo, Osni, Ricardo e Laércio Brito. É uma honra ter sido aluna de pessoas tão dedicadas, que foram exemplos de competência e amor à profissão.

Obrigada aos queridos companheiros de Fundão. Vocês não só me ajudaram nos estudos e trabalhos, mas também tornaram meu caminho mais leve e me apoiaram durante a execução deste projeto: Pedro Cunha, Patrícia Zudio, Evelyn, Débora, Pedro Rocha, Thomaz, Andressa, Paula Ballard, Rogério, Renato, Bianca, Cinila, Denilson, Yanko, Mário, Ricardo, Julianne, Paula Silva, Sérgio, Eduardo, Leonardo, Cecília, Patrícia Borges.

Em especial, meu sincero agradecimento ao amigo e co-orientador Bruno Bottino, por toda a paciência que teve comigo. A Tiago Mota, por ter sido tão solícito ao esclarecer minhas dúvidas. Agradeço também à Professora Adriana Vivacqua pela concessão de material, e aos amigos Renan e Fabrício por terem se lembrado dele.

Por fim, minha gratidão a todos aqueles que não foram citados aqui, mas que torceram pelo meu sucesso.

Índice de Tabelas

Tabela 2.1: Parâmetros dos conjuntos nebulosos das variáveis de entrada de distância, d_{f1} , d_{f2} e d_{f3}	18
Tabela 2.2: Parâmetros dos conjuntos nebulosos da variável de entrada de ângulo do robô virtual, ϕ	18
Tabela 2.3: Parâmetros dos conjuntos nebulosos da variável de saída de ângulo da roda, θ	19
Tabela 2.4: Parâmetros dos conjuntos nebulosos da variável de saída de velocidade, Δ	20
Tabela 3.1: Tabela de interpolação usada nos testes de navegação.....	30
Tabela 3.2: Média da saída y filtrada em quatro direções.....	31
Tabela 3.3: Tabela de correspondência de giro.....	42
Tabela 3.4: Tabela de correspondência de passo.....	43

Índice de Figuras

Figura 2.1: Ambiente virtual com obstáculos.....	14
Figura 2.2: Determinação da posição do robô virtual no ambiente.....	14
Figura 2.3: Direções dos sensores de distância.	15
Figura 2.4: Módulo decisor.....	16
Figura 2.5: As regiões são delimitadas pelas linhas tracejadas.....	17
Figura 2.6: Funções de pertinência para d_{f1} , d_{f2} e d_{f3}	18
Figura 2.7: Funções de pertinência para o ângulo do robô virtual, ϕ	19
Figura 2.8: Funções de pertinência para o ângulo da roda, θ	20
Figura 2.9: Funções de pertinência para a velocidade, Δ	20
Figura 2.10: Vista superior do Arduino Uno.....	21
Figura 2.11: Vista superior do Arduino Mega ADK.....	22
Figura 2.12: Sensor de bússola HM55B.....	22
Figura 2.13: Esquema de operação do sensor PING)))	23
Figura 2.14: Alguns casos de diminuição da eficácia do PING)))	24
Figura 2.15: Sensor PING))) acoplado ao motor de passo, na frente do robô.....	24
Figura 2.16: O robô real.....	25
Figura 2.17: O robô real, vista superior.....	25
Figura 3.1: Diagrama das etapas de uma iteração típica de navegação.....	26
Figura 3.2: Medidas do raio do robô e da distância do pivô de rotação à sua traseira.....	27
Figura 3.3: Tamanho fictício do robô.....	28
Figura 3.4: Bússola segmentada para auxiliar a calibragem do sensor HM55B.....	29
Figura 3.5: Distância entre o PING))) e seu apoio sobre o motor de passo.....	34
Figura 3.6: Esquema gráfico para aplicação da Lei dos Cossenos.....	35
Figura 3.7: Robô e obstáculos no início da iteração i.....	36
Figura 3.8: Posição do robô no início da iteração i+1.....	37
Figura 3.9: O movimento é decomposto em giro.....	37

Figura 3.10: ... e deslocamento sobre o eixo x.....	38
Figura 3.11: Relação entre o argumento angulo e o sentido da rotação dos servos.....	41
Figura 3.12: Padronização do domínio da velocidade para ambas as rodas.....	41
Figura 4.1: Ambiente de testes do robô real.....	45
Figura 4.2: Ambiente de testes do robô real visto de cima.....	45
Figura 4.3: Ambiente de testes do robô real.....	46
Figura 4.4: Ambiente de testes, mostrando a saída do corredor (direção horizontal)	46
Figura 4.5: Trajetória aproximada do robô no primeiro teste ($\phi^0 = 0^\circ$).....	47
Figura 4.6: Configuração inicial do segundo teste ($\phi^0 = -90^\circ$) e trajetória aproximada do robô.....	48
Figura 4.7: Configuração inicial do terceiro teste ($\phi^0 = 90^\circ$) e trajetória aproximada do robô...48	
Figura 4.8: Configuração inicial do quarto teste ($\phi^0 = -45^\circ$) e trajetória aproximada do robô...49	
Figura 4.9: O robô virou para o sentido contrário ao da saída do corredor no quarto teste.....49	
Figura 4.10: Configuração inicial do quinto teste ($\phi^0 = 45^\circ$) e trajetória aproximada do robô...50	
Figura 4.11: Indicação das direções tomadas pelo robô nos testes 1 e 6.....51	
Figura 4.12: Indicação das direções tomadas pelo robô no sétimo teste.....51	
Figura 4.13: Trajetória aproximada do robô no teste 7.....52	

Sumário

1. Introdução	11
2. O Robô	13
2.1. Modelo Base de Navegação – Ambiente e Robô Virtuais.....	13
2.2. Controlador	17
2.3. Características Físicas do Robô Real.....	21
3. Implementação.....	26
3.1. Corpo Fictício do Robô	27
3.2. Localização: Determinação do Estado do Robô	28
3.3. Detecção de Obstáculos	33
3.4. Módulo Decisor.....	36
3.4.1. Pré-processador	36
3.4.2. Aplicação do Controlador	39
3.5. Movimentação	40
4. Testes e Resultados	44
4.1. Ambiente de Testes	44
4.2. Testes Realizados	46
5. Conclusões	54
5.1. Desafios e Dificuldades	54
5.2. Comentários, Contribuições e Sugestões de Trabalhos Futuros	55
Referências Bibliográficas.....	57

1. Introdução

A aplicação de estratégias da inteligência computacional à robótica tem se mostrado cada vez mais presente em diversos cenários, como a indústria e a pesquisa científica, seja acadêmica ou com fins comerciais. Esse crescente interesse se deve, em grande parte, à capacidade de robôs autônomos inteligentes executarem tarefas em diferentes ramos de conhecimento, inclusive substituindo seres humanos em locais inóspitos ou em atividades de alto risco.

O objetivo deste projeto foi implementar, testar e desenvolver os algoritmos necessários para possibilitar a navegação autônoma de um robô real simples dotado de um controlador baseado em lógica nebulosa. Para tanto, é indispensável que o robô seja capaz de gerar corretamente as entradas do controlador e empregar de forma apropriada as suas saídas, e são esses os papéis dos códigos desenvolvidos. A definição das características exigidas do robô e o controlador utilizado foram elaborados por Mota (2010), que, por sua vez, baseou-se no trabalho de Moratori (2006). O objetivo de ambos é, em linhas gerais, permitir que um robô navegue autonomamente, desviando de obstáculos e das paredes do corredor onde se encontra, até que atinja a saída. Mota (2010), particularmente, teve o intuito de utilizar sensores que pudessem ser implementados a partir de componentes comumente disponíveis no mundo real. Isso não só viabilizaria a aplicação do controlador num robô físico, mas também possibilitaria adotar uma estrutura de baixo custo. Dessa forma, o robô real utilizado neste trabalho necessitou apenas de uma estrutura móvel, com rodas diferenciais, um sensor ultrassônico de distância, um sensor de campo magnético e uma placa Arduino.

Em geral, modelos de navegação são primeiramente testados em ambientes virtuais a fim de se obter embasamento teórico para sua utilização em aplicações reais, mas simulações nem sempre são capazes de refletir as imprecisões do mundo real. Assim, é útil dispor de uma estrutura física para realizar essa segunda etapa, permitindo confrontar ambos os desempenhos e levantar possíveis melhorias para o modelo. Mota (2010) fez somente simulações, deixando justamente como sugestão de trabalho futuro a implantação do controlador num robô físico. No decorrer do projeto aqui apresentado, algumas das estratégias empregadas resultaram em adaptações interessantes do modelo.

Este texto apresenta as características do robô no Capítulo 2, com a descrição dos principais pontos do modelo lógico usado como base nas Seções 2.1 e 2.2, e a parte física na Seção 2.3. O Capítulo 3 trata da principal contribuição deste projeto, começando pela explicação da necessidade de se adotar medidas fictícias para o corpo do robô na Seção 3.1. A partir da Seção 3.2 são mostradas as etapas do desenvolvimento na mesma ordem do fluxo de execução de uma iteração. Esta nada mais é que o conjunto de operações realizadas pelo robô, o qual se repete até o final da navegação. Já os testes de navegação em ambiente são

explorados no Capítulo 4 e, por fim, alguns comentários, contribuições e sugestões de trabalhos futuros são discutidos no Capítulo 5.

2. O Robô

Este capítulo descreve as características do robô utilizado. Contudo, para que se compreendam algumas escolhas feitas, como os tipos de sensores empregados, primeiramente é necessário conhecer o modelo de navegação adotado. Entenda-se por “modelo de navegação”, ou simplesmente “modelo”, o conjunto que engloba o controlador e as configurações necessárias para o ambiente e o robô autônomo a fim de que o controlador opere da forma esperada.

O modelo em questão foi elaborado por Mota (2010). Seu objetivo, em linhas gerais, é permitir que um robô navegue autonomamente, desviando de obstáculos e das paredes do corredor onde se encontra até atingir a saída. Uma das contribuições de seu trabalho foi propor um modelo de robô virtual que fizesse uso de sensores acessíveis, facilmente implementáveis a partir de componentes físicos comuns. Dessa forma, seria possível reproduzi-lo num robô real de baixo custo.

Assim, este capítulo começa com a apresentação das características mais relevantes do modelo de Mota (2010) – ambiente e robô virtuais na Seção 2.1 e controlador na Seção 2.2. Em seguida, na Seção 2.3, são mostradas as características físicas do robô real utilizado, desde suas medidas até os sensores adotados.

2.1. Modelo Base de Navegação – Ambiente e Robô Virtuais

O trabalho de Mota (2010) foi desenvolvido e testado no MATLAB¹, versão 7.0. Ele propôs um ambiente de navegação e um robô virtuais com configurações apropriadas para a aplicação do controlador elaborado.

O ambiente virtual é um corredor retangular, com uma saída à direita, onde são dispostos alguns obstáculos fixos. São consideradas apenas as componentes horizontal e vertical dos objetos no ambiente, como se ele estivesse sendo visto de cima. O corredor tem 200 unidades de medida de comprimento por 100 unidades de medida de largura. Foi adotado um sistema de coordenadas relativas ao ambiente, onde os pontos (0, 0) e (200, 100) estão, respectivamente, sobre seus cantos inferior esquerdo e superior direito. Os obstáculos são circulares, com 3 unidades de medida de raio. A priori, pode-se definir aleatoriamente sua quantidade e suas posições, mas ambas devem permanecer as mesmas ao longo da navegação. Um esquema gráfico do ambiente virtual é mostrado na Figura 2.1.

¹ <<http://www.mathworks.com>>. Acesso em: 11-08-2012.

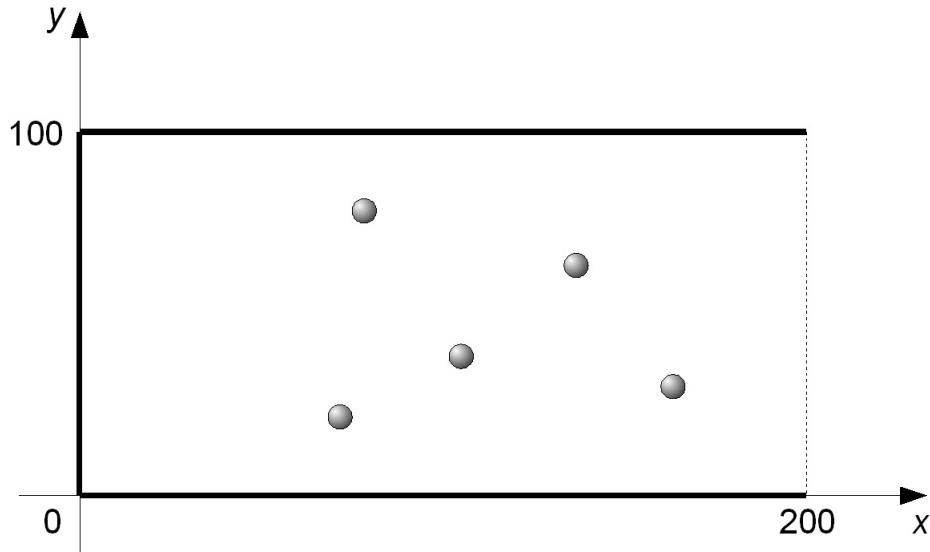


Figura 2.1: Ambiente virtual com obstáculos. Retirada de Mota (2010).

O robô virtual também é circular, com 6 unidades de medida de raio. O estado do robô virtual corresponde a sua posição no ambiente, que a cada instante de tempo t é dada pela tripla $(x_r^{(t)}, y_r^{(t)}, \phi^{(t)})$. O par $(x_r^{(t)}, y_r^{(t)})$ fornece as coordenadas relativas do centro do robô virtual, e $\phi^{(t)}$ é o ângulo entre a direção frontal do robô virtual e a linha paralela à horizontal do ambiente que passa pelo seu centro. A Figura 2.2 mostra um esquema do robô virtual no ambiente, destacando os componentes da sua tripla de posição. A seta indica sua direção frontal.

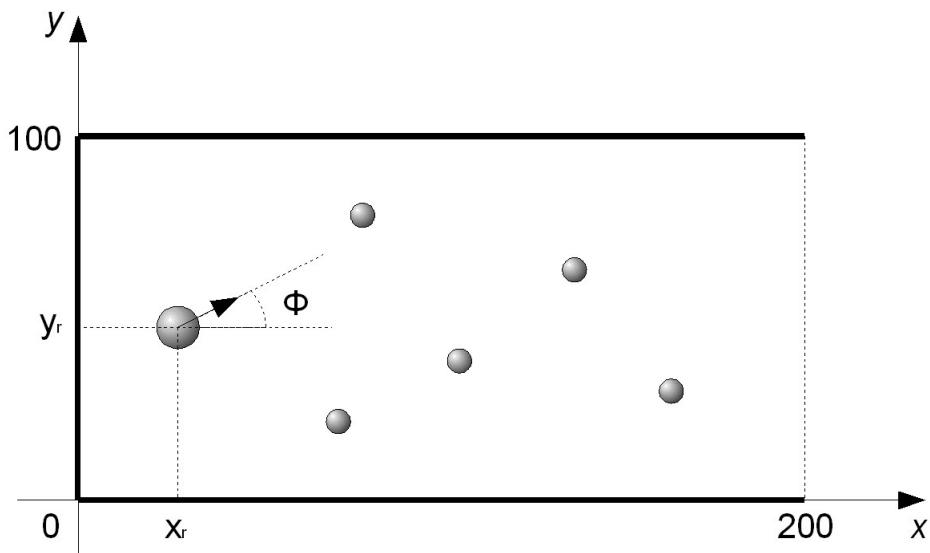


Figura 2.2: Determinação da posição do robô virtual no ambiente. Retirada de Mota (2010).

O domínio de $\phi^{(t)}$ é $[-180^\circ, 180^\circ]$. A convenção adotada foi a mesma do círculo trigonométrico: $\phi^{(t)}$ é positivo se o ângulo denotado estiver na região superior à horizontal, e negativo quando estiver na região inferior.

Nesse modelo, a navegação sempre tem início com o robô virtual encostado na parede esquerda, ou seja, $x_r^0 = 6$. O valor de y_r^0 é livre, podendo variar no intervalo $[6, 94]$, e ϕ^0 é restrito ao intervalo $[-90^\circ, 90^\circ]$. O objetivo do robô virtual é chegar à saída do corredor ($x_r \geq 194$) sem colidir com suas paredes ou com os obstáculos. Daqui por diante, como o robô interpreta uma parede ou um obstáculo propriamente dito simplesmente como um objeto do qual deve desviar, ambos serão chamados indistintamente de "obstáculo".

O robô virtual foi estruturado em três módulos: sensores, módulo decisão e módulo de movimentação. Ele não possui nenhum conhecimento a priori, ou seja, não sabe nada sobre o ambiente ou sua posição nele. São os sensores que, captando dados externos, permitem-lhe perceber seu entorno. Desta forma, o robô virtual utiliza sensores de distância e de bússola para se orientar no corredor.

O sensor de distância informa a distância até o obstáculo mais próximo na sua linha de detecção. Foram utilizados três dispositivos desse tipo, colocados no centro do robô virtual. Suas posições, dadas em relação à frente do robô, são $\alpha_1^* = 15^\circ$, $\alpha_2^* = 0^\circ$ e $\alpha_3^* = -15^\circ$. A convenção adotada para determinar os sinais dos ângulos com relação à frente do robô é a mesma de $\phi^{(t)}$ – a do círculo trigonométrico. A Figura 2.3 esquematiza de forma simples essa configuração, representando com as linhas tracejadas as direções para onde apontam os sensores.

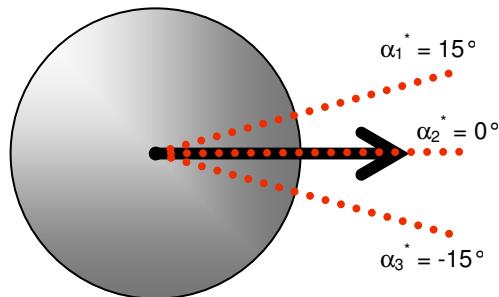


Figura 2.3: Direções dos sensores de distância (linhas pontilhadas). A seta indica a direção da frente do robô virtual.

Somente um sensor de bússola foi empregado, com o papel de informar o ângulo $\phi^{(t)}$ da frente do robô virtual com a horizontal do ambiente. Na verdade, isto é uma simplificação de um sensor real desse tipo. Mota (2010) inclusive pondera que, nesse caso, obter-se-ia a medição do campo magnético da Terra, ou a direção apontada por ele em relação ao norte magnético, sendo necessário tratar esse valor para transformá-lo em $\phi^{(t)}$ de fato.

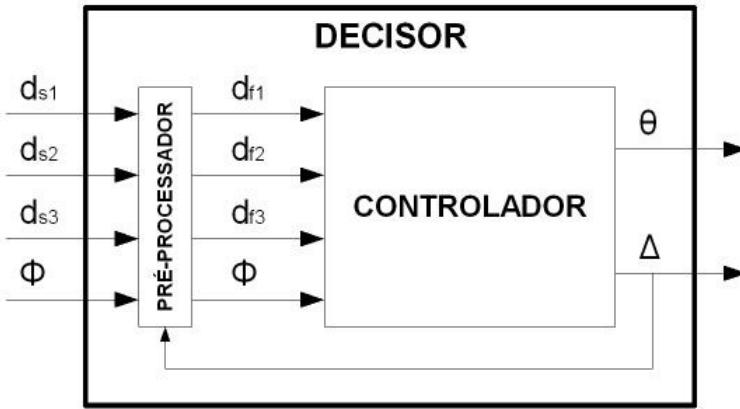


Figura 2.4: Módulo decisor. Retirada de Mota (2010).

O chamado módulo decisor, ilustrado na Figura 2.4, contém o controlador, que será abordado na Seção 2.2, e um módulo pré-processador. Este módulo interno é responsável por receber os dados captados pelos sensores, tratá-los, analisá-los e, por fim, enviar ao controlador informações mais relevantes. Ele possui uma memória onde a localização dos obstáculos detectados nas iterações mais recentes pode ou não ser incluída. Em caso positivo, esse dado é armazenado no formato de coordenadas polares, em relação ao centro do robô virtual. A cada iteração, antes que o robô se movimente, os pontos em memória são atualizados de acordo com sua última posição. Para determinar se um novo ponto detectado deve ou não ser inserido, lhe é aplicado um critério de relevância. Se sua distância a cada ponto já armazenado for maior que 3 unidades, considera-se que o ponto pode agregar informações e ele é acrescentado à memória. Para informações mais detalhadas sobre a atualização da memória, consulte Mota (2010).

Isto feito, o pré-processador seleciona os valores das três variáveis de distância que são entradas do controlador juntamente com $\phi^{(t)}$: $d_{f1}^{(t)}$, $d_{f2}^{(t)}$ e $d_{f3}^{(t)}$. O intuito dessas três variáveis é indicar a distância do obstáculo em memória que está mais próximo da esquerda, da frente e da direita do robô virtual, respectivamente. Para tanto, foram definidas três regiões em relação a sua frente, reproduzidas na Figura 2.5:

- região esquerda, de 15° a 105° ;
- região frontal, de -15° a 15° ;
- e região direita, de -105° a -15° .

O módulo de movimentação, como o próprio nome sugere, é responsável pela movimentação do robô. Há dois movimentos possíveis: giro (em torno do pivô de rotação) e passo (avanço em linha reta). A magnitude deles a cada iteração é determinada pelas saídas do controlador, $\theta^{(t)}$ (ângulo de giro) e $\Delta^{(t)}$ (tamanho do passo).

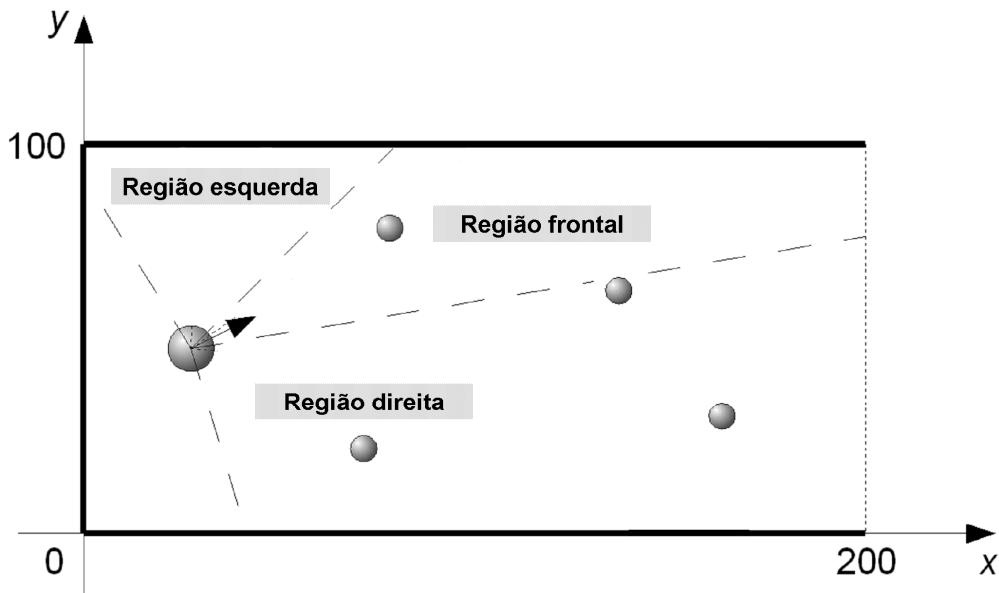


Figura 2.5: As regiões são delimitadas pelas linhas tracejadas. Adaptado de Mota (2010).

Nesta seção foram expostos os dados envolvidos na navegação e como eles são tratados. Resumidamente, uma iteração é a sequência de coleta de informações do ambiente, processamento, giro e deslocamento, nesta ordem. Isto se repete até que o robô virtual colida com algum obstáculo ou atinja o objetivo estipulado para a navegação.

2.2. Controlador

O controlador consiste em um sistema nebuloso do tipo Mamdani (Mamdani, 1977), com quatro variáveis de entrada, duas variáveis de saída e duzentas regras. Suas variáveis de entrada e saída já foram explicadas na Seção 2.1; aqui serão mostrados seus domínios e conjuntos nebulosos. Maiores detalhes sobre o controlador, como funções de pertinência dos conjuntos e regras, são encontrados em Mota (2010).

As três variáveis de entrada de distância, d_{f1} , d_{f2} e d_{f3} , possuem domínio [0, 500]. Há quatro conjuntos nebulosos, também iguais para ambas: "Muito Perto" (MP), "Perto" (Pe), "Médio" (Me) e "Longe" (Lo), cujos parâmetros e gráficos são explicitados na Tabela 2.1 e na Figura 2.6, respectivamente. A representação usada na Tabela 2.1 segue o padrão do toolbox de lógica nebulosa do MATLAB² para definição de conjuntos fuzzy.

² <<http://www.mathworks.com/help/toolbox/fuzzy/fp754.html>>. Acesso em: 23-08-2012.

Conjunto	Formato	Parâmetros
Muito Perto (MP)	Trapézio	[0, 0, 10, 15]
Perto (Pe)	Triângulo	[10, 20, 30]
Médio (Me)	Triângulo	[20, 50, 75]
Longe (Lo)	Trapézio	[50, 75, 500, 500]

Tabela 2.1: Parâmetros dos conjuntos nebulosos das variáveis de entrada de distância, d_{f1} , d_{f2} e d_{f3} . Adaptada de Mota (2010).

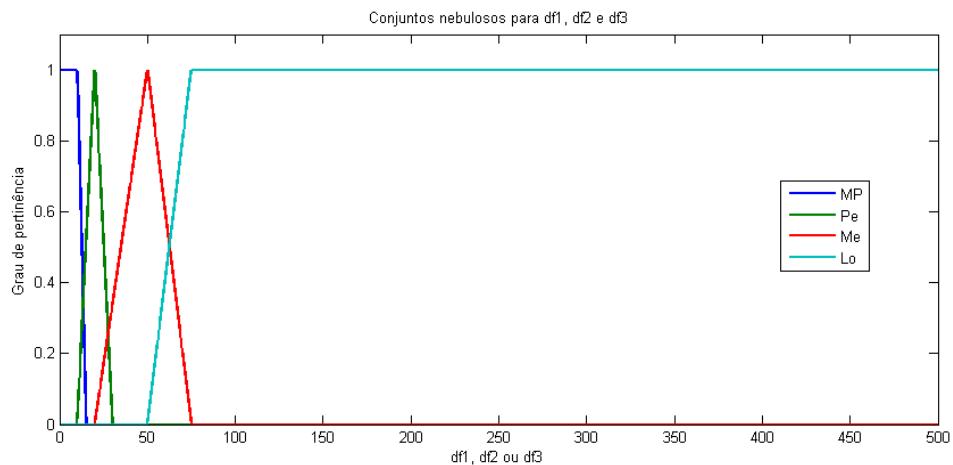


Figura 2.6: Funções de pertinência para d_{f1} , d_{f2} e d_{f3} . Retirada de Mota (2010).

A variável de entrada ϕ , ângulo do robô virtual, tem domínio $[-180, 180]$ e cinco conjuntos nebulosos: “Baixo-Trás” (BT), “Baixo” (Ba), “Frente” (Fr), “Cima” (Ci) e “Cima-Trás” (CT). Eles são mostrados na Tabela 2.2 e na Figura 2.7.

Conjunto	Formato	Parâmetros
Baixo-Trás (BT)	Triângulo	[-210, -180, -120]
Baixo (Ba)	Triângulo	[-180, -90, -0]
Frente (Fr)	Triângulo	[-60, 0, 60]
Cima (Ci)	Triângulo	[0, 90, 180]
Cima-Trás (CT)	Triângulo	[120, 180, 210]

Tabela 2.2: Parâmetros dos conjuntos nebulosos da variável de entrada de ângulo do robô virtual, ϕ . Adaptada de Mota (2010).

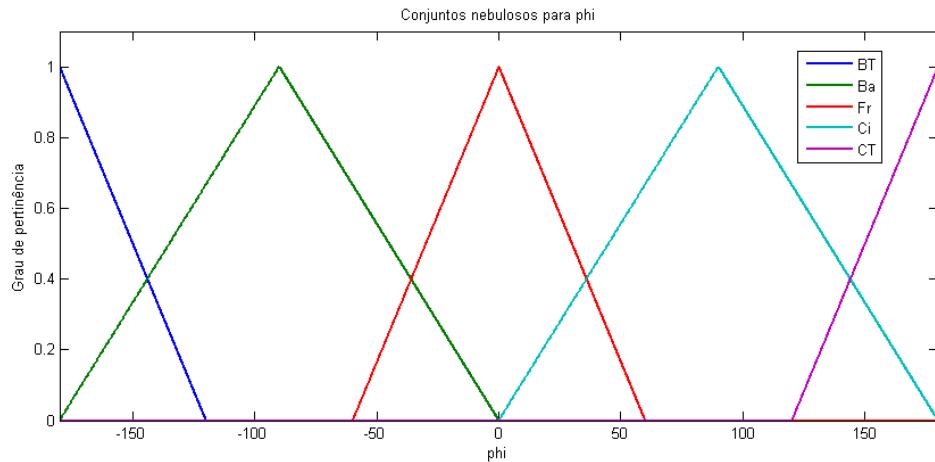


Figura 2.7: Funções de pertinência para o ângulo do robô virtual, ϕ . Retirada de Mota (2010).

A variável de saída de ângulo da roda θ (aqui chamada de “ângulo de giro”), com domínio $[-30, 30]$, possui sete conjuntos nebulosos: “Muito Direita” (MD), “Direita” (Di), “Pouco Direita” (PD), “Zero” (Ze), “Pouco Esquerda” (PE), “Esquerda” (Es) e “Muito Esquerda” (ME). Os conjuntos são detalhados na Tabela 2.3 e na Figura 2.8.

A segunda variável de saída, velocidade Δ (aqui chamada de “tamanho do passo”), possui domínio $[0, 10]$. Na Tabela 2.4 estão os parâmetros dos seus cinco conjuntos nebulosos: “Muito Pequena” (MP), “Pequena” (Pe), “Média” (Me), “Grande” (Gr) e “Muito Grande” (MG). A Figura 2.9 mostra suas funções de pertinência.

Conjunto	Formato	Parâmetros
Muito Direita (MD)	Trapézio	$[-30, -30, -20, -12.5]$
Direita (Di)	Triângulo	$[-17.5, -10, -5]$
Pouco Direita (PD)	Triângulo	$[-10, -5, 0]$
Zero (Ze)	Triângulo	$[-5, 0, 5]$
Pouco Esquerda (PE)	Triângulo	$[0, 5, 10]$
Esquerda (Es)	Triângulo	$[5, 10, 17.5]$
Muito Esquerda (ME)	Trapézio	$[12.5, 20, 30, 30]$

Tabela 2.3: Parâmetros dos conjuntos nebulosos da variável de saída de ângulo da roda, θ .

Adaptada de Mota (2010).

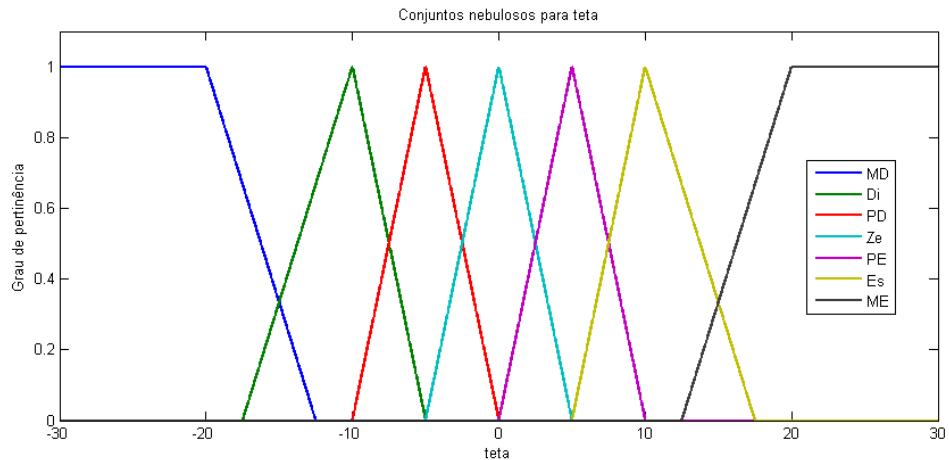


Figura 2.8: Funções de pertinência para o ângulo da roda, θ . Retirada de Mota (2010).

Conjunto	Formato	Parâmetros
Muito Pequena (MP)	Trapézio	[0, 0, 0.1, 0.5]
Pequena (Pe)	Triângulo	[0, 0.5, 1]
Média (Me)	Triângulo	[0.5, 1.3, 2]
Grande (Gr)	Triângulo	[1.3, 3, 5]
Muito Grande (MG)	Trapézio	[4, 5, 10, 10]

Tabela 2.4: Parâmetros dos conjuntos nebulosos da variável de saída de velocidade, Δ .

Adaptada de Mota (2010).

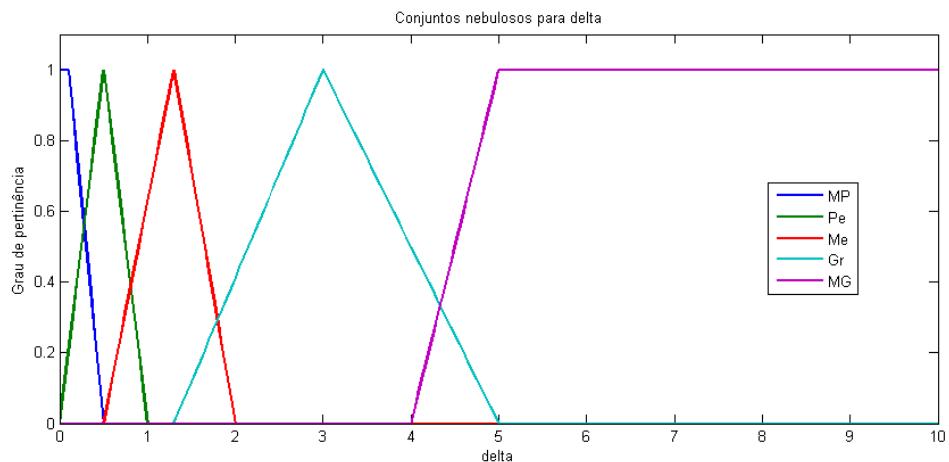


Figura 2.9: Funções de pertinência para a velocidade, Δ . Retirada de Mota (2010).

2.3. Características Físicas do Robô Real

A estrutura do robô³, feita em MDF (Medium-density fibreboard), é circular e seu raio mede 10.5cm. Ele possui duas rodas de tração dianteiras com servomotores independentes, permitindo que elas girem com velocidades diferentes, e um rodízio na parte traseira para estabilização, que não interfere no direcionamento do robô.

A plataforma escolhida foi o Arduino⁴, devido a seu baixo custo, sua flexibilidade e sua facilidade de programação. O ambiente de desenvolvimento é simples, multiplataforma, e sua linguagem é implementada em C/C++. Tanto hardware quanto software são livres e extensíveis, havendo uma quantidade considerável de algoritmos e bibliotecas disponíveis para uso gratuito. Em seu site é possível encontrar tais algoritmos, além de materiais para aprendizagem⁵ e uma boa referência de linguagem⁶.

Inicialmente foi adotado o Arduino Uno⁷ (Figura 2.10), que é baseado no microcontrolador ATmega328. Ele possui 32KB de memória flash, 14 pinos digitais de entrada e saída e tensão de operação de 5V. Contudo, essa quantidade de memória não foi suficiente para comportar o controlador e, por fim, optou-se pelo Arduino Mega ADK⁸ (Figura 2.11). Baseado no microcontrolador ATmega2560, ele conta com 256KB de memória flash, 54 pinos digitais de entrada e saída e também opera sob uma tensão de 5V.

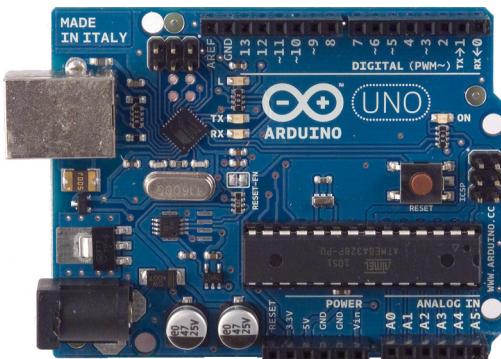


Figura 2.10: Vista superior do Arduino Uno.

³ <<http://www.labdegaragem.com.br/loja/robokit03.pdf>>. Acesso em: 11-08-2012.

⁴ <<http://arduino.cc/en>>. Acesso em: 11-08-2012.

⁵ <<http://arduino.cc/playground>>; <<http://arduino.cc/en/Tutorial>>. Acesso em: 11-08-2012.

⁶ <<http://arduino.cc/en/Reference>>. Acesso: em 11-08-2012.

⁷ <<http://arduino.cc/en/Main/ArduinoBoardUno>>. Acesso em: 11-08-2012.

⁸ <<http://arduino.cc/en/Main/ArduinoBoardADK>>. Acesso em: 11-08-2012.

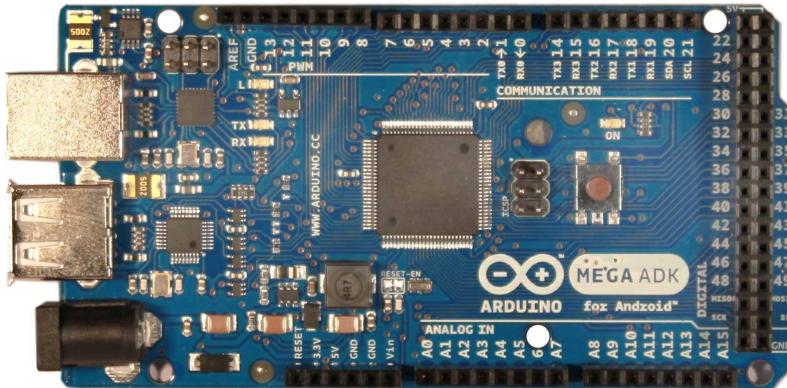


Figura 2.12, com a convenção de que seu valor é positivo no sentido horário a partir do Norte. Para obter-se γ , então, basta calcular $\arctan(-Y/X)$, onde X e Y simbolizam os respectivos produtos escalares explicitados anteriormente. Observe que o HM55B não indica a direção Norte, mas sim a intensidade do campo magnético ao redor de si. A partir do cálculo de γ , porém, é possível inferir para que direção o sensor está apontando.

O HM55B requer calibragem, feita via software. Seus objetivos são compensar a interferência de possíveis campos magnéticos no entorno dele e corrigir a posição dos eixos, caso estejam transladados e/ou rotacionados. Como o sensor foi desenhado originalmente para o microcontrolador BASIC Stamp, programável em PBASIC, há códigos prontos nessa linguagem para calibrá-lo. O código disponível no site do Arduino⁹, entretanto, é suficiente apenas para interagir com o sensor, que possui um protocolo de comunicação serial síncrona próprio. As funções desenvolvidas para calibragem e outras estratégias envolvendo o uso do HM55B são abordadas na Seção 3.2.

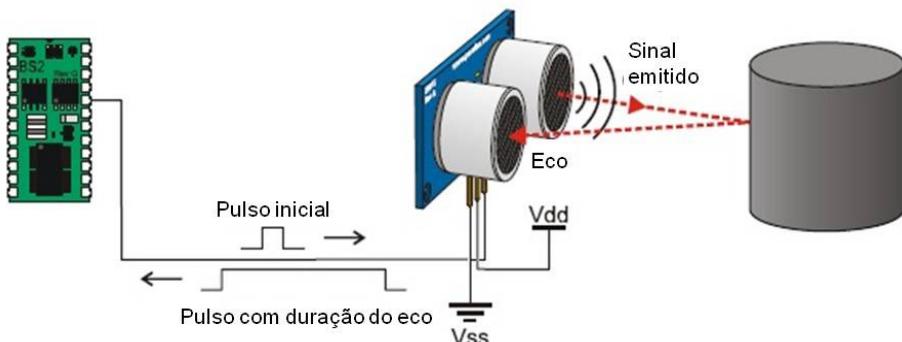


Figura 2.13: Esquema de operação do sensor PING)). Adaptação.

Para detectar obstáculos, adotou-se o sensor ultrassônico de distância Parallax PING))), que opera a partir da emissão de um pulso ultrassônico e da consequente recepção de seu eco (Parallax Inc., 2009). Como mostra a Figura 2.13, esse intervalo de tempo é o que determina a distância entre o sensor e o objeto detectado, e sua faixa de detecção nominal é de 2cm a 3m. A eficácia do PING))) diminui com superfícies localizadas de modo a não refletir o eco na direção dele (Figura 2.14), e com anteparos menos capazes de refletir uma quantidade suficiente de ondas, a exemplo de objetos muito pequenos (Figura 2.14), superfícies irregulares e materiais que absorvem o som. Deve-se, também, tomar cuidado com o posicionamento do sensor, pois é possível a detecção de objetos acima ou abaixo dele. No site do Arduino é

⁹ <<http://arduino.cc/playground/Main/HM55B>>. Acesso em: 11-08-2012.

possível encontrar um código para interagir com esse sensor¹⁰, o que facilita ainda mais seu uso.

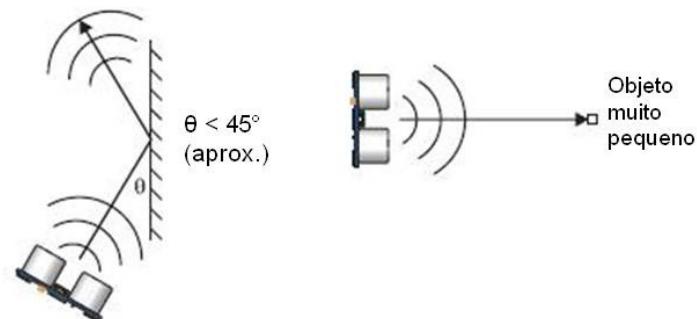


Figura 2.14: Alguns casos de diminuição da eficácia do PING))). Adaptação.

O PING))) foi colocado sobre um motor de passo localizado na frente do robô, conforme mostrado na Figura 2.15. Esse motor gira com uma amplitude máxima de 180° entre as extremidades direita e esquerda da região frontal do robô. Na verdade, a extrema direita deveria ser equivalente a 0° , mas há uma diferença de aproximadamente 8° . Dessa forma, a posição central, por exemplo, que deveria ser alcançada pelo motor a 90° , é atingida satisfatoriamente apenas quando ele se encontra em 98° . Detalhes da aplicação deste sensor são dados na Seção 3.3.



Figura 2.15: Sensor PING))) acoplado ao motor de passo, na frente do robô.

¹⁰ <<http://arduino.cc/en/Tutorial/Ping>>. Acesso em: 11-08-2012.

Algumas medidas do robô são alteradas pelo posicionamento do PING))) e seu motor de passo. Embora a estrutura do robô tenha 21cm de diâmetro, é preciso levar em conta o espaço ocupado por esses dispositivos na sua região frontal – um adicional de 1.6cm nessa direção. Observe que não há alteração do raio do robô em si, apenas deve-se considerar o acréscimo dessa medida numa pequena região. A altura do robô também é influenciada, chegando a 18.5cm no total. As Figuras 2.16 e 2.17 exibem o robô real, sendo possível visualizar a maioria dos elementos citados nessa seção.

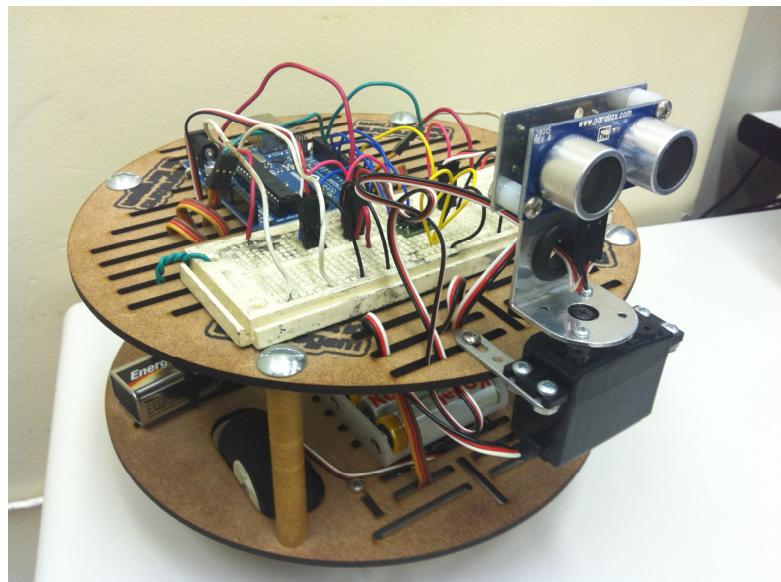


Figura 2.16: O robô real.

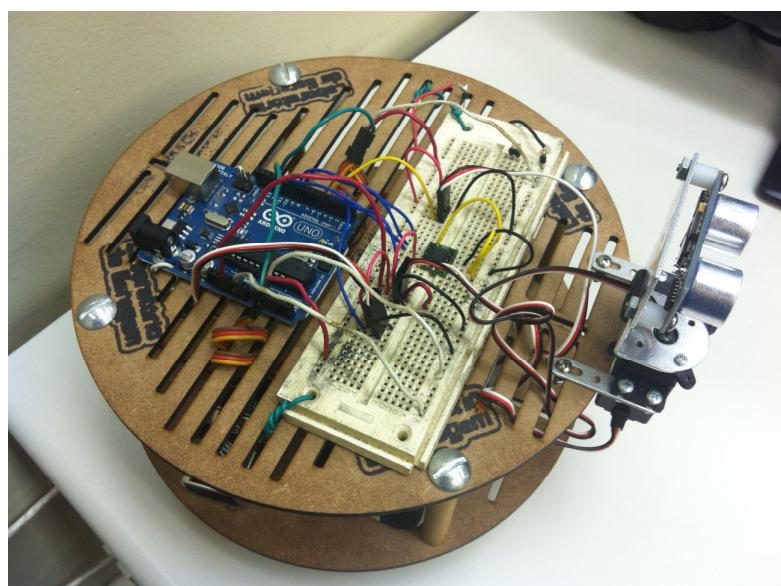


Figura 2.17: O robô real, vista superior.

3. Implementação

Aqui é tratada a principal contribuição deste trabalho: a lógica implementada para permitir que o robô navegue de acordo com o modelo adotado. São relatados os problemas enfrentados e as estratégias de adequação mais importantes, começando pela caracterização do corpo fictício do robô. A apresentação do conteúdo restante segue a ordem das etapas de uma iteração típica de navegação, composta por determinação do estado do robô; detecção de obstáculos; inclusão dos dados em memória; determinação das entradas do controlador e seu uso; e movimentação. Esta pareceu ser a ordem mais natural, pois, ao mesmo tempo, é possível visualizar o fluxo da informação ao longo da navegação, facilitando seu entendimento. Foram realizados testes “modulares”, ou seja, cada parte desenvolvida que trazia uma nova funcionalidade ao projeto era avaliada, de forma a assegurar que tudo operava conforme esperado. O código completo preparado para rodar na plataforma Arduino está disponível em <http://code.google.com/p/pf-mariam-afonso-dcc-ufrj-2012/> sob licença GNU GPL.

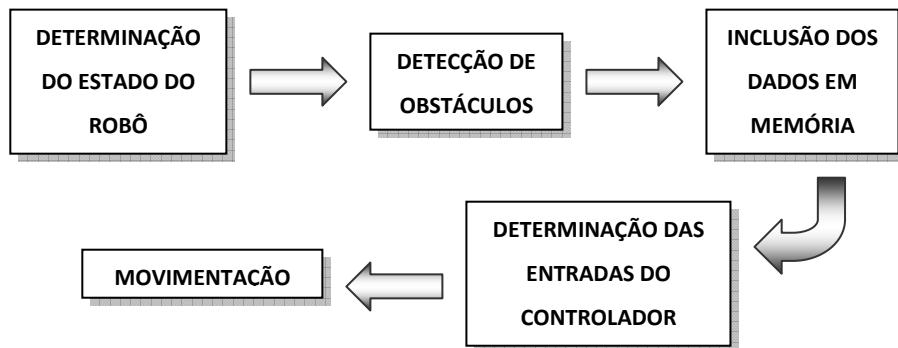


Figura 3.1: Diagrama das etapas de uma iteração típica de navegação.

Este capítulo, então, tem início com a abordagem do corpo fictício do robô na Seção 3.1. A determinação do estado do robô e a detecção de obstáculos são assuntos das Seções 3.2 e 3.3, respectivamente. A Seção 3.4 discorre sobre o módulo decisão, responsável pela atualização da memória e por fornecer as entradas ao controlador. Algumas medidas necessárias para a inclusão do controlador também são exploradas nessa seção. Por fim, a Seção 3.5 fala da movimentação.

3.1. Corpo Fictício do Robô

Conforme mostrou a Seção 2.1, as medidas de estado (tripla de posição) e de distância a obstáculos são todas determinadas com relação ao centro do robô virtual. A configuração do robô real, porém, não favorecia o uso do seu centro como referencial para o segundo tipo de medida, uma vez que a melhor localização para o PING))) era à sua frente. Se colocado sobre o centro do robô, além de aumentar significativamente sua altura, o sensor poderia detectar a estrutura do robô abaixo de si em vez dos obstáculos.

Assim sendo, a estratégia mais simples seria simular que o PING))) estivesse sobre o centro físico do robô. Isso seria possível com alguns cálculos para transladar a origem da detecção da frente para o centro do robô. Entretanto, havia mais uma questão: como as rodas de tração não ficam no meio do robô, esse ponto não coincide com seu pivô de rotação. Como mostra a Figura 3.2, enquanto o raio mede 10.5cm, o pivô de rotação está a 13.6cm da sua traseira. Isto é um problema porque, ao girar, o centro físico do robô muda de lugar, o que não condiz com o modelo.

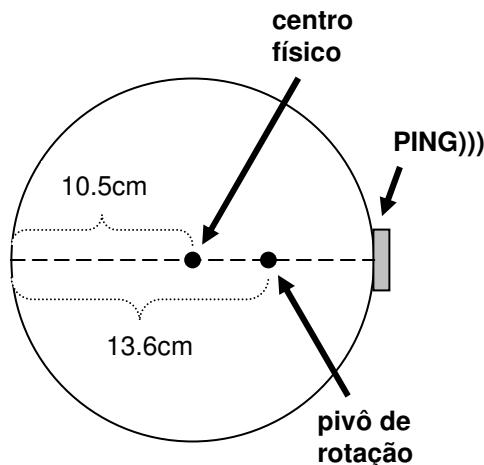


Figura 3.2: Medidas do raio do robô e da distância do pivô de rotação à sua traseira.

A solução, por fim, foi adotar um corpo fictício para o robô, cujo esquema é exibido na Figura 3.3. Ele continua circular e com a mesma altura, mas agora tem 13.6cm de raio e é centrado sobre seu pivô de rotação, o que resolve os dois problemas citados anteriormente nesta seção. Daqui por diante, então, referências ao “centro do robô” dirão respeito ao centro de seu corpo fictício, pois toda lógica elaborada neste trabalho considera seu novo tamanho. Além disso, permaneceu a ideia de transladar a origem das medidas de distância para o centro do robô, estratégia que será explicada mais à frente, na Seção 3.3.

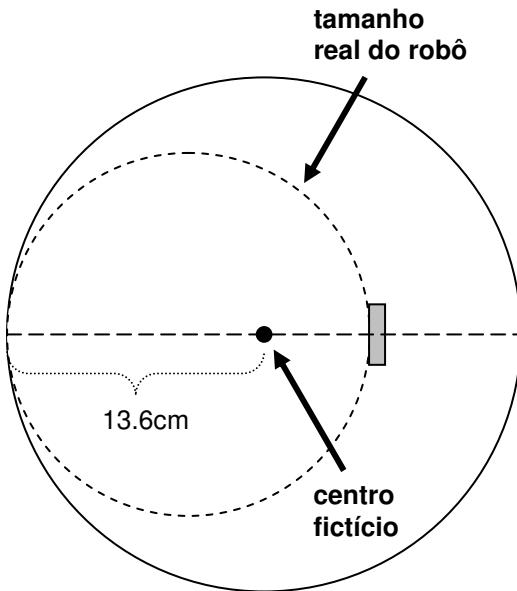


Figura 3.3: Tamanho fictício do robô.

3.2. Localização: Determinação do Estado do Robô

A primeira etapa de uma iteração típica da navegação é a localização do robô; isto é, a determinação do seu estado, ou ainda, da sua posição no ambiente. Na Seção 2.1 foi dito que o estado do robô virtual num instante de tempo t é dado pela tripla $(x_r^{(t)}, y_r^{(t)}, \phi^{(t)})$. O controlador, porém, utiliza somente o valor de $\phi^{(t)}$, ignorando as coordenadas relativas ao ambiente. Por esse motivo decidiu-se determinar o estado do robô real através de $\phi^{(t)}$ apenas, evitando a computação e o armazenamento de dados desnecessários à navegação.

O sensor de campo magnético HM55B é o dispositivo chave para essa tarefa. Como citado na Seção 2.3, os algoritmos necessários para captar a direção para onde ele aponta estão disponíveis no site do Arduino. No entanto, os primeiros testes com o sensor usando apenas esse código obtiveram medições ruins, bem diferentes das sugeridas por uma bússola. Assim, estudando a documentação do HM55B (Parallax Inc., 2005), viu-se que outros procedimentos eram necessários para que ele operasse corretamente. Para auxiliar em algumas dessas tarefas foi usada a Figura 3.4, que será chamada de “bússola segmentada”. Colocada sobre uma superfície plana, ela foi alinhada com as indicações fornecidas por uma bússola de bolso e então afixada naquela posição.

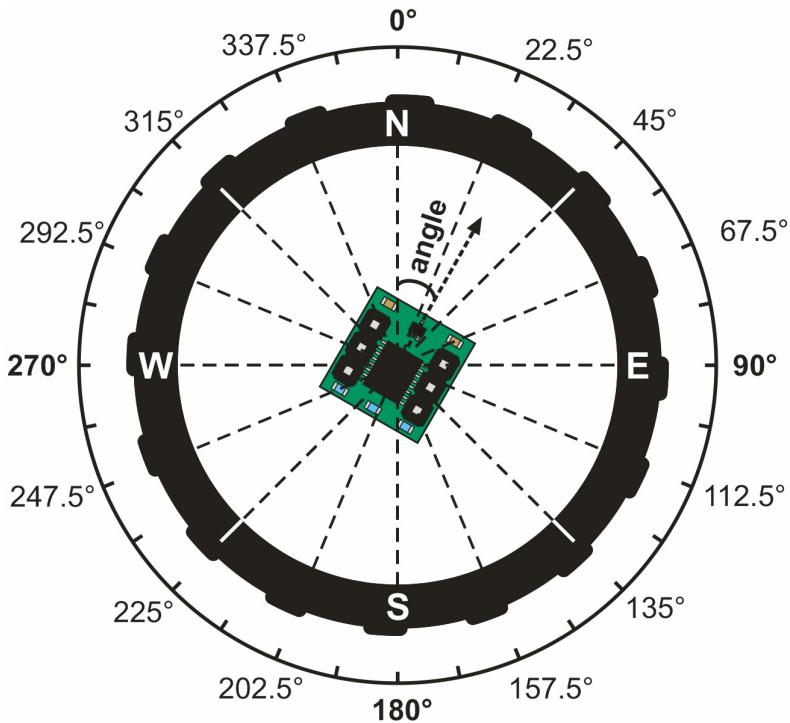


Figura 3.4: Bússola segmentada para auxiliar a calibragem do sensor HM55B.

De acordo com a teoria sobre forças e campos magnéticos, a força do campo magnético medida nas direções Norte e Sul deveria ser igual em módulo, o mesmo valendo para Leste e Oeste (Young, Freedman, 2009). Em outras palavras, considerando o sistema de coordenadas intrínseco ao sensor, seus eixos deveriam estar centrados em zero, o que não necessariamente ocorre. Por isso, antes de tudo é necessária a sua calibragem, cujo primeiro passo é a determinação de offsets para a translação dos seus eixos. O Algoritmo 3.1 mostra como obter esses valores.

Algoritmo 3.1: Calibragem do sensor HM55B – determinação de offsets.

Sejam as direções dos pólos geográficos **Norte**, **Leste**, **Sul** e **Oeste**, aferidos por uma bússola.

Sejam **N**, **E**, **S** e **W** as medições do sensor HM55B da força do campo magnético em cada uma dessas direções, respectivamente.

Sejam **X_offset** e **Y_offset** os valores de offset para os eixos x e y, respectivamente. Então

$$\begin{aligned} \text{X_offset} &= (\text{N} + \text{S}) / 2.0 \\ \text{Y_offset} &= (\text{E} + \text{W}) / 2.0 \end{aligned}$$

O robô foi colocado sobre a bússola segmentada com o sensor na direção central da figura para efetuar as medições em questão. Uma vez obtidos os offsets, para que a translação seja efetuada de fato eles devem ser subtraídos da medição no eixo correspondente sempre que o HM55B for usado.

Ainda assim, o sistema de coordenadas do sensor pode não estar satisfatoriamente alinhado. Nesse caso nota-se discrepância entre a indicação de uma bússola e o valor de γ (ângulo entre a direção Norte e o eixo x do HM55B; vide Seção 2.3) numa direção qualquer. Dessa forma, a fim de corrigir uma possível rotação ou inclinação é realizada a segunda e última fase da calibragem. Com o robô colocado novamente sobre a bússola segmentada, são computados os valores de γ nas dezesseis direções nela marcadas, que dividem a circunferência em setores com 22.5° de abertura. Cabe lembrar que aqui os offsets já foram aplicados para corrigir as medições.

Com esses dados é montada a chamada “tabela de interpolação”, que faz a correspondência entre as dezesseis direções de referência e os respectivos valores de γ . Para tornar a explicação mais clara é apresentada a Tabela 3.1 com as informações usadas durante os testes de navegação do robô (abordados posteriormente no Capítulo 4).

Direção de referência	Valor de γ
0	314.24
22.50	343.78
45	26.00
67.50	55.75
90	73.52
112.50	85.17
135	97.31
157.50	111.51
180	133.38
202.50	168.74
225	202.64
247.50	233.68
270	250.12
292.50	264.72
315	277.36
337.50	291.95

Tabela 3.1: Tabela de interpolação usada nos testes de navegação. Valores em graus.

Preenchida a tabela, ela deve ser sempre usada para corrigir γ através de interpolação linear. Sejam chamados de “valores reais” os dados tabelados oriundos do HM55B. Pode-se dizer que eles também compõem uma bússola segmentada, cujos setores não são igualmente distribuídos. Contudo, para que um γ qualquer esteja próximo do valor ideal, é preciso manter a proporção entre os setores de referência e os reais. Por exemplo, seja $\gamma = 106.96^\circ$. Como $97.31^\circ \leq \gamma \leq 111.51^\circ$, ele pertence ao 7º setor. Isso significa que, na verdade, o certo seria que ele estivesse no intervalo $[135^\circ, 157.50^\circ]$. É preciso, então, saber a quanto γ corresponde nesse setor de referência, valor chamado de γ' . Seja $D = 9.65^\circ$ a distância entre γ e o início do setor real, cuja abertura é $A = 111.51^\circ - 97.31^\circ = 14.20^\circ$. A proporção de D em A é $P \approx 0.68$. De forma similar, D' é a distância de γ' ao início do setor de referência, cuja abertura $A' = 22.5^\circ$ já é conhecida. Segue que a proporção entre D' e A' também precisa ser P , de onde se obtém $\gamma' \approx 150.30$.

A documentação do sensor (Parallax Inc., 2005) possuía códigos para tais procedimentos na linguagem PBASIC. Foram feitas pesquisas em busca desse material disponível para Arduino, mas não houve sucesso. Também não foram encontradas referências sobre interpolação de γ ou calibragem do HM55B em materiais sobre o seu uso nessa plataforma. Por isso foi necessário estudar os algoritmos em PBASIC para compreendê-los e traduzi-los para a linguagem do Arduino.

Antes da calibragem o sensor foi preliminarmente testado quanto à variabilidade de medições sucessivas numa mesma direção, e notou-se uma sensibilidade considerável a interferências. Para amenizar um pouco esse efeito, foi-lhe aplicado um filtro média móvel (Smith, 2003) e, na verdade, cada medição é resultado da média de dez medições sucessivas. Depois de calibrado, sua acurácia foi avaliada novamente pela comparação entre a saída γ em diferentes direções e as indicações de uma bússola de bolso. O sensor se mostrou razoavelmente preciso, atingindo resultados próximos dos desejados, como pode ser visto na Tabela 3.2. A média e o desvio padrão foram calculados sobre dez amostras filtradas da saída γ em cada direção citada.

Direção	Média das amostras de γ	Desvio padrão das amostras
Norte (0°)	-4.35	0.79
Leste (90°)	89.22	0.53
Sul (180° ou -180°)	179.96	0.39
Oeste (-90°)	-90.06	0.94

Tabela 3.2: Média da saída γ filtrada em quatro direções.

Uma vez conhecida a forma apropriada de se utilizar o sensor de campo magnético, retomou-se o foco da etapa de localização: a determinação do valor de $\phi^{(t)}$. Na Seção 2.1 foi dito que ele é o ângulo entre a direção frontal do robô e a linha paralela à horizontal do ambiente que passa pelo seu centro. Assim, a protoboard e o sensor foram posicionados de forma que ele ficasse sobre o centro do robô. Por outro lado, não havia como saber a direção da horizontal de forma imediata. A solução foi usar o próprio HM55B para computá-la, associando à horizontal o γ obtido com o robô apontado na sua direção. Esse dado é fixo para o ambiente de testes, se mantendo constante ao longo da navegação. Dessa forma, $\phi^{(t)}$ nada mais é que a diferença entre o γ medido no instante t e o valor referente à horizontal.

Perceba que essa estratégia permite que a horizontal seja alterada antes de qualquer navegação. Tal opção e o fato de que $x_r^{(t)}$ e $y_r^{(t)}$ não são úteis ao controlador trouxeram uma vantagem: a possibilidade de se abstrair do tamanho do corredor. Da mesma forma que o sistema fuzzy não toma conhecimento das coordenadas relativas ao ambiente, suas dimensões também não são relevantes. Ao mesmo tempo, o uso do sensor de campo magnético para atribuir um valor à horizontal tornou-a dependente de informações extrínsecas à configuração do ambiente, diferentemente do que é adotado no modelo. Com isso surgiu a dúvida sobre o que passaria, então, a ser o objetivo do robô. Originalmente ele deve atingir a saída do corredor, o que ocorre quando $x_r \geq 194$, desviando dos obstáculos. Já o objetivo do controlador é conduzir uma navegação sem colisões enquanto procura manter o ângulo ϕ o mais próximo possível da direção horizontal do ambiente. Veja que as duas metas são similares. Mais que isso, como não há nada além do sistema fuzzy comandando os movimentos do robô, foi possível atribuir-lhe o objetivo do controlador. Assim o robô real é capaz de navegar em qualquer ambiente minimamente controlado, com paredes à direita e à esquerda, mas com dimensões flexíveis.

Considerando todo o tempo dedicado ao desenvolvimento deste projeto, esta foi a etapa mais trabalhosa e mais longa. Os principais motivos foram a necessidade do estudo de códigos numa linguagem que não se dominava, os procedimentos exigidos para obter melhores resultados do sensor e atender aos requisitos do modelo, bem como os diversos testes de ajuste realizados ao longo dela. Na Seção 3.3 será explorada a detecção de obstáculos, onde também precisaram ser aplicadas estratégias de adequação ao modelo e artifícios para lidar com o sensor utilizado.

3.3. Detecção de Obstáculos

Depois de se situar no ambiente, o robô precisa saber se há obstáculos a sua volta. Conforme exposto no Capítulo 2, o modelo utiliza três sensores de distância posicionados em diferentes ângulos, mas o robô real dispõe de somente um sensor PING))). Contudo, como ele está acoplado a um motor de passo capaz de girar 180°, a cada iteração ele é posto nas direções desejadas. Com isto foi possível simular os três sensores a partir de um só, em vez de empregar mais dispositivos. Além do custo, o tamanho do PING))) foi outro grande incentivo para a adoção desse mecanismo, pois dificilmente caberiam três exemplares seus nos locais necessários.

O Arduino dispõe de uma biblioteca padrão capaz de controlar o motor de passo, chamada Servo¹¹. Ela provê a função `write(ângulo)`, cujo argumento `ângulo` determina a posição, de 0° a 180°, onde o motor deve ficar. Como ele está na frente do robô, foi possível apontar o sensor de distância para as direções necessárias sem maiores dificuldades, apenas observando alguns detalhes. A angulação do motor cresce da direta para a esquerda do robô, além de existir uma defasagem de 8° entre suas extremidades (Seção 2.3). Assim, para que o sensor estivesse orientado a 15°, 0° e -15° da frente (α_i^* , $i \in \{1, 2, 3\}$), o motor precisou ser posicionado em 113°, 98° e 83°, respectivamente.

O código para interação com o PING))) disponível no site do Arduino (Seção 2.3) foi usado como ponto de partida do desenvolvimento dessa etapa. Foram realizados alguns testes para assegurar se sua faixa de detecção nominal, de 2cm a 300cm, era realmente praticada. Descobriu-se que o sensor consegue identificar obstáculos distantes até 368cm. Aparentemente ele retorna leituras de 369 ou 370 quando considera que não há nada a sua frente, seja porque não há objetos ao alcance do sinal ultrassônico emitido ou porque ocorre falha na medição. Um dos problemas é que, abaixo do limite inferior da faixa (distância menor que 2cm), seu retorno também é 370. Além disso, notou-se que obstáculos posicionados entre 2cm e 5cm são detectados como se estivessem a 4cm ou 5cm de distância. Observando a localização do sensor, constatou-se que ele está aproximadamente 4cm longe da base metálica que o apoia no motor de passo, como indicado na Figura 3.5. Acredita-se, então, que o resultado aparentemente equivocado seja, na verdade, a detecção desse suporte.

¹¹ <<http://arduino.cc/en/Reference/Servo>>. Acesso em: 11-08-2012.

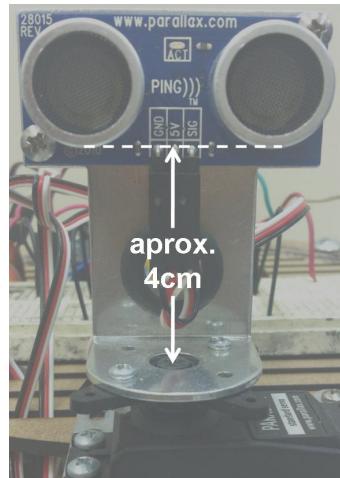


Figura 3.5: Distância entre o PING))) e seu apoio sobre o motor de passo.

Esses testes evidenciaram a existência de algumas leituras dúbias do PING))), que precisaram ser tratadas para não prejudicar a navegação. Convencionou-se que o menor valor possível de medição seria 5cm, e, em vez de descartar valores menores, eles foram trocados por esse piso. Isto se aplica porque há um espaço de 4.6cm entre as frentes do sensor e do robô de acordo com suas medidas fictícias. Ou seja, um obstáculo nessas condições estaria logicamente a 0.4cm do robô, e não é esperada uma proximidade maior que esta, pois o sistema fuzzy tende a conduzir manobras evasivas antecipadamente. Leituras de 370 foram ignoradas porque, de fato, não haveria como precisar se seu significado era a inexistência de objetos no alcance do sensor, ou uma falha de operação, ou uma colisão com ele. Ignorou-se também o valor 369 pelos dois primeiros motivos anteriores. O descarte desses valores é simbolizado por uma marcação que evita sua inclusão na memória do robô (mais detalhes no Capítulo 4).

Uma vez que as leituras do sensor de distância já haviam sido tratadas satisfatoriamente, o procedimento seguinte foi a mudança do seu referencial. A exposição dessa ideia teve início na Seção 3.1, onde destacou-se a necessidade de haver um sistema de coordenadas relativo ao centro do robô para as medidas de distância. Nesse estágio elas já poderiam ser representadas em coordenadas polares (Anton, Bivens, Davis, 2005) para posterior inclusão na memória. Por isso, a Lei dos Cossenos serviu perfeitamente a este propósito, permitindo o cálculo da distância entre o centro do robô e o obstáculo, e do ângulo que a linha que une ambos faz com a direção frontal do robô (Anton, Bivens, Davis, 2007). O esquema em que tais cálculos se basearam é mostrado na Figura 3.6, onde

- $r = 9\text{cm}$ = distância entre o centro do robô e a frente do sensor;
- d' = distância medida pelo PING));
- d = distância com relação ao centro, usada na representação polar;
- $\alpha = \alpha_i^*, i \in \{1, 2, 3\}$;
- β = ângulo para a representação polar.

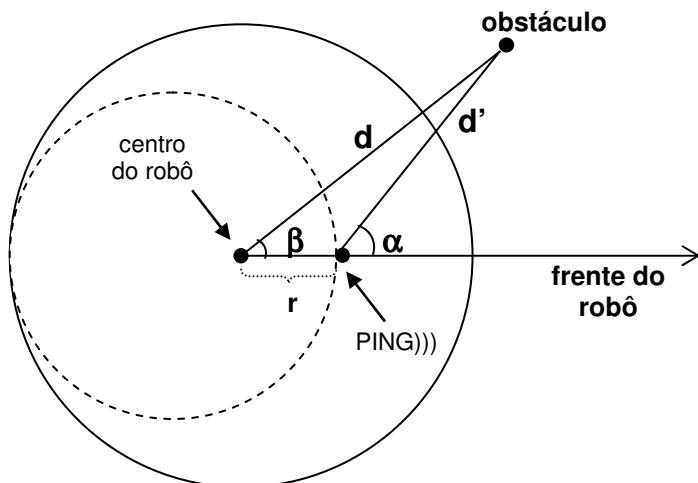


Figura 3.6: Esquema gráfico para aplicação da Lei dos Cossenos.

Como a Lei dos Cossenos também se aplica a triângulos degenerados, não foi necessário tratamento especial quando $\alpha = \alpha_2^* = 0^\circ$. No caso em que $\alpha = \alpha_3^* = -15^\circ$, as contas foram feitas considerando apenas o módulo de α e, no final, simplesmente multiplicou-se β por -1 .

Um sensor ultrassônico de distância é uma escolha geralmente mais barata que um sensor infravermelho, mas, por outro lado, costuma ser menos preciso. Os testes realizados ao longo desta etapa de implementação mostraram que a detecção do PING)) muitas vezes é falha, chegando até mesmo a não ocorrer. Por exemplo, garrafas plásticas de 500ml, cuja parte superior é mais fina que o restante, não são detectadas a pouca distância dele. Quando cobertas por um cilindro de papel da mesma altura, o sensor passou a encontrá-las. Acredita-se que isso ocorria por causa do formato das garrafas, apesar de seu corpo também ser cilíndrico.

A Seção 3.4 discorrerá sobre o módulo decisivo, introduzido na Seção 2.1. De posse dos resultados das medições já convertidos para coordenadas polares, o primeiro passo é cuidar da atualização da memória. Em seguida devem ser definidas as entradas do controlador, que é novamente abordado, mas com enfoque nas ferramentas necessárias para sua aplicação no robô.

3.4. Módulo Decisor

3.4.1. Pré-processador

O modelo define o módulo decisor como o conjunto formado pelo pré-processador e pelo controlador. O primeiro é um submódulo responsável por tratar os dados disponíveis antes de repassá-los ao controlador. Ele é dotado de memória, que foi implementada como um vetor de nove posições apenas, inicialmente vazia. Sabe-se da Seção 3.3 que, descontadas as detecções ignoradas, a cada iteração são identificados, no máximo, três obstáculos. Esses k objetos, $k \in \{0, 1, 2, 3\}$, são representados como pontos em coordenadas polares com relação ao centro do robô para serem possivelmente incluídos na memória. Como ela ainda está vazia na primeira iteração, os k pontos são armazenados diretamente nas suas k posições mais baixas. A partir da segunda iteração, é preciso atualizar o que já se encontra em memória antes do procedimento de inclusão. Isto é necessário porque, como o robô se movimenta, tais pontos devem acompanhar a mudança de posição do seu referencial. Os cálculos realizados são os mesmos indicados em Mota (2010), e serão brevemente detalhados a seguir com o auxílio das Figuras 3.6 a 3.9. As ilustrações são pequenos esquemas para facilitar a compreensão do passo-a-passo desse procedimento.

Suponha que o robô esteja na posição da Figura 3.7 no início da i -ésima iteração. O robô se movimenta e, no início da $(i+1)$ -ésima iteração, seja sua nova posição aquela mostrada na Figura 3.8. Suponha, também, que a localização dos três obstáculos desenhados já está em memória. Como eles são fixos, sua posição relativa ao centro do robô acaba sendo alterada.

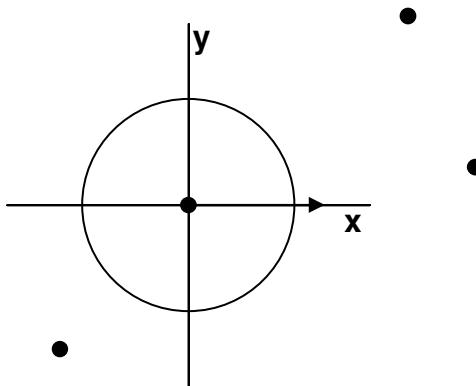


Figura 3.7: Robô, cuja frente é indicada pela seta, e obstáculos no início da iteração i .

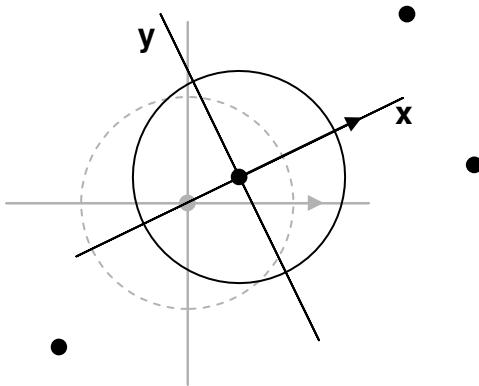


Figura 3.8: Posição do robô no início da iteração $i+1$, em preto. O sombreado mostra sua posição anterior.

Para refletir essa mudança, primeiro considera-se apenas o giro do robô, que resulta na rotação dos eixos (Figura 3.9). Assim o ângulo β da representação dos pontos é atualizado, enquanto sua distância d até o centro permanece a mesma.

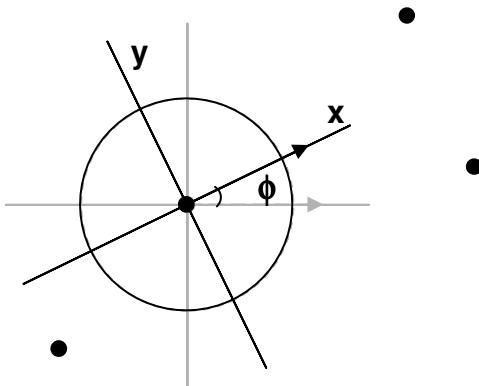


Figura 3.9: O movimento é decomposto em giro...

Antes de levar-se em conta o deslocamento do robô, os pontos são representados em coordenadas cartesianas. Como o robô avança na sua direção frontal, seu centro é transladado para a direita sobre o eixo x (Figura 3.10). Ou, sob outra perspectiva, os obstáculos se aproximam do centro, caminhando para a esquerda do plano. Então basta diminuir suas coordenadas x e, por fim, retornar à representação polar.

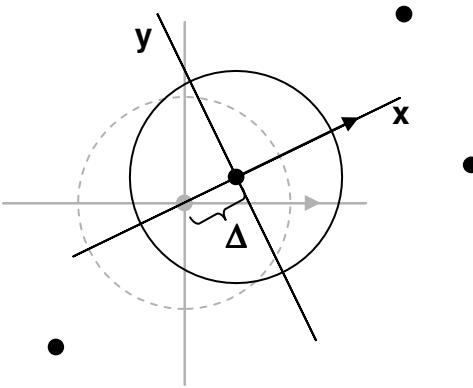


Figura 3.10: ... e deslocamento sobre o eixo x.

Feita a atualização, é avaliado se os k novos pontos devem ou não entrar na memória de acordo com o critério de relevância proposto no modelo. Para cada ponto são calculadas as distâncias euclidianas entre ele e cada elemento já armazenado. Se todos os valores forem maiores que 3cm, então o novo ponto é adicionado. Cabe lembrar que as detecções ignoradas são de fato descartadas e não são incluídas na memória.

O processo de atualização, avaliação e inclusão é efetuado em todas as iterações a partir da segunda. Dessa forma, em algum momento a memória fica cheia, sendo necessário substituir os dados mais antigos pelos mais novos. A ideia é que ela contenha informações de fato atuais – em média, das três últimas iterações – e o mais precisas possível. Se seu tamanho fosse maior, muito provavelmente as sucessivas operações sobre os dados armazenados fariam o erro acumulado crescer significativamente. A ocorrência de derrapagem das rodas (assunto abordado na Seção 3.5) também causaria propagação de erro, pois o robô não se moveria exatamente o esperado. Além disso, como o robô busca avançar, e não retroceder, a tendência é que os obstáculos outrora a sua frente sejam deixados para trás. Isto é, depois de um intervalo considerável de tempo, lembranças antigas do ambiente já não influenciam mais a navegação, pois aquelas regiões não serão revisitadas. Assim, o vetor de memória é preenchido como se fosse circular, com a primeira posição sendo contígua da última.

A tarefa seguinte do pré-processador é a determinação das entradas do controlador, assunto introduzido na Seção 2.1. Além de $\phi^{(t)}$, que não precisa de tratamentos adicionais, o módulo seleciona os valores que serão atribuídos às três variáveis de distância $d_{f1}^{(t)}$, $d_{f2}^{(t)}$ e $d_{f3}^{(t)}$. A ideia delas é informar ao sistema fuzzy onde estão os obstáculos mais próximos da esquerda, da frente e da direita do robô, respectivamente. Essas direções são representadas por regiões delimitadas com relação à frente do robô: de 15° a 105° (esquerda); de -15° a 15° (frente); e de -105° a -15° (direita).

Esses dados são extraídos da memória. Saber a que região pertence um elemento torna-se uma tarefa simples devido à representação polar: basta avaliar o valor de seu ângulo β . Terminada a varredura, é possível que uma ou mais regiões estejam vazias, isto é, sem

nenhum obstáculo identificado no seu interior. Nesse caso, a variável em questão recebe o valor padrão máximo de distância, 400cm. Embora seu domínio seja [0, 500] (vide Seção 2.2), já é sabido que o alcance do sensor de distância se limita a 368cm. Aplicada a mudança de referencial, esta medida corresponde a 376.7cm e, portanto, qualquer valor acima deste não seria confundido com uma leitura real. No entanto, avaliando o comportamento do controlador no MATLAB, notou-se que as saídas eram as mesmas quando o valor de uma das variáveis estava no intervalo [400, 500]. Em outras palavras, o controlador interpreta quaisquer entradas nesse intervalo da mesma maneira. Por isso optou-se por padronizar seu máximo em 400cm nessa implementação.

Antes de alimentar o sistema fuzzy, ainda foi necessário adequar as medidas do mundo real. Isto porque o robô real, que possui um raio fictício de 13.6cm, é 2.26 vezes maior que o robô virtual, com 6 unidades de medida de raio. Buscando garantir uma navegação próxima do esperado e evitar colisões, adotou-se a proporção 1:2.26 entre os mundos virtual e real. Assim, os valores de $d_{fi}^{(t)}$, $i = \{1, 2, 3\}$, são divididos por 2.26 e enviados ao controlador juntamente com $\phi^{(t)}$, enquanto sua saída Δ é multiplicada por este mesmo fator. Já o ângulo de giro θ não precisa ser manipulado.

3.4.2. Aplicação do Controlador

A configuração do sistema fuzzy já foi abordada na Seção 2.2. Nesta seção será tratado o aparato necessário para empregá-lo no robô. Primeiramente era preciso dispor de uma biblioteca de lógica nebulosa que lidasse com sistemas Mamdani e fosse compatível com o Arduino. Também seria necessário converter o arquivo com extensão .fis, construído no MATLAB, num formato que ela fosse capaz de manipular. Para tais finalidades foi utilizado o código elaborado por Mário Alberto Cecchi Raduan, integrante do LabIC¹² (Laboratório de Inteligência Computacional), no iNCE, UFRJ, assim como a autora desta monografia. Ele desenvolveu um interpretador para arquivos .fis serem usados no Arduino e uma biblioteca fuzzy aceita nessa plataforma.

Depois de converter o controlador, deve-se copiar alguns arquivos para o diretório do projeto do Arduino e incluir a biblioteca no código, com a diretiva `#include "libFuzzy.h"`. Além disso é requerida a criação de dois vetores do tipo float: `entrada[NUMINPUTS]` e `saída[NUMOUTPUTS]`. Como os próprios nomes sugerem, eles recebem as entradas e as saídas do sistema, respectivamente. O tamanho desses vetores é dado pelas constantes NUMINPUTS e NUMOUTPUTS, cujos valores, definidos pelo interpretador, são a quantidade de variáveis de entrada e de saída. Neste caso, tem-se que NUMINPUTS = 4 e NUMOUTPUTS = 2. Fornecidas as

¹² <<http://www.labic.nce.ufrj.br/>>. Acesso em: 23-08-2012.

entradas, basta chamar a função que representa o controlador e executa as devidas operações para calcular suas saídas: `rodarMeuFis(entrada, saida)`. Note-se que a ordem dos elementos nesses vetores é a mesma das variáveis no controlador, ou seja,

- $\text{entrada}[0] = d_{f1};$
- $\text{entrada}[1] = d_{f2};$
- $\text{entrada}[2] = d_{f3};$
- $\text{entrada}[3] = \phi;$
- $\text{saida}[0] = \theta;$
- $\text{saida}[1] = \Delta.$

Um fato interessante, mencionado na Seção 2.3, é que os 32KB de memória do Arduino Uno não foram capazes de comportar o controlador, composto de 200 regras. Diversas estratégias foram cogitadas, dentre elas, usar apenas duas variáveis de entrada (uma de distância e ϕ) ou reduzir a quantidade de regras. Antes que tais medidas fossem estudadas, porém, conseguiu-se um Arduino Mega ADK¹³, que possui 256KB de memória. O único trabalho adicional foi ligar os fios à nova placa e ajustar a numeração dos pinos, que não se manteve.

3.5. Movimentação

Uma vez disponíveis as saídas do sistema fuzzy, inicia-se a última etapa da iteração: a movimentação do robô. Assim como o motor de passo acoplado ao sensor PING)), os servomotores das duas rodas de tração são controlados pela biblioteca padrão Servo. Nesse caso, a função `write(angulo)` atua de forma diferente, com o argumento `angulo` indicando a velocidade de rotação do servo. Seu domínio continua sendo $[0^\circ, 180^\circ]$, e a velocidade mínima (que idealmente o faz ficar parado) é 90° . Os extremos do intervalo correspondem à velocidade máxima, mas em sentidos opostos. A forma como os servos das rodas são afixados no robô faz com que as rodas girem em direções contrárias com o mesmo valor de `angulo`. Essa relação fica mais clara com o esquema mostrado na Figura 3.11.

¹³ Agradecimentos à professora Adriana Vivacqua e aos colegas Renan Lobo e Fabrício Firmino – PPGI, UFRJ.

Roda direita:	--- avança ---	--- retrocede -	
Rotação:	max	min	max
Argumento angulo:	0	90	180
Roda esquerda:	-- retrocede -	--- avança ---	
Rotação:	max	min	max

Figura 3.11: Relação entre o argumento angulo e o sentido da rotação dos servos.

O movimento é composto por giro e deslocamento, e a quantidade deles é dada pelos valores do ângulo de giro $\theta^{(t)}$ e do tamanho do passo $\Delta^{(t)}$, num instante de tempo t. Note que os servomotores são independentes, ou seja, as rodas podem operar em velocidades diferentes, tornando viável o giro do robô em torno do seu eixo de rotação. Foram construídas pequenas funções com a sequência de comandos necessários para a execução de cada um dos componentes do seu movimento, e era desejável que elas servissem para as duas rodas. Por isso optou-se por padronizar o domínio da velocidade a partir da distância a 90°, que é o valor mínimo comum a ambas, resultando no intervalo [-90, 90]. A Figura 3.12 ilustra tal situação.

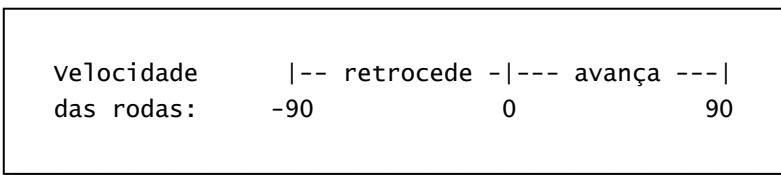


Figura 3.12: Padronização do domínio da velocidade para ambas as rodas.

É possível determinar a duração do movimento do servo com o uso da função `delay(tempo)` do Arduino, onde tempo deve estar em milissegundos, seguida de um comando de parada. Entretanto, as unidades de medida de $\theta^{(t)}$ e de $\Delta^{(t)}$ são, respectivamente, graus e centímetros. A solução adotada foi construir tabelas de correspondência relacionando a quantidade de tempo de operação dos servos para que o robô andasse $\Delta^{(t)}$ centímetros ou girasse $\theta^{(t)}$ graus. O mapeamento do giro foi dividido em dois sentidos, direita ($\theta^{(t)} < 0$) e esquerda ($\theta^{(t)} > 0$), com coleta empírica dos dados. Arbitrou-se o valor mínimo de 25ms, seguido por 50ms. O incremento para os demais foi de 50ms até se atingir 150ms, somando quatro durações averiguadas. Para cada uma delas houve cinco amostras sequenciais do ângulo percorrido, medido com auxílio de um transferidor centrado no pivô de rotação do robô. A

Tabela 3.3, chamada de “tabela de correspondência de giro”, foi montada com a média dos ângulos obtidos para cada tempo em cada sentido.

tempo (em ms)	Direita		Esquerda	
	Desvio padrão das amostras	Ângulo girado (em graus)	Desvio padrão das amostras	Ângulo girado (em graus)
25	3.9	-6.8	1.5	4.4
50	3.0	-17.4	3.2	10.2
100	2.2	-23.8	4.3	18.0
150	4.6	-29.8	3.9	23.0

Tabela 3.3: Tabela de correspondência de giro.

A Tabela 3.3 foi, então, usada para determinar a duração do movimento das rodas para que o robô pudesse girar $\theta^{(t)}$. Uma vez que o controlador tenha calculado seu valor, basta interpolá-lo com os pontos tabelados. A ideia é similar àquela empregada na interpolação do ângulo γ medido pelo sensor de campo magnético (Seção 3.2). Por exemplo, se $\theta^{(t)} = 7.3^\circ$, a distância ao valor tabelado imediatamente menor que ele é $D = 7.3^\circ - 4.4^\circ = 2.9^\circ$, e o tamanho do intervalo dos valores tabelados entre os quais ele está é $T = 10.2^\circ - 4.4^\circ = 5.8^\circ$. Assim, a proporção de D em T é $P = 0.5$. Similarmente, se D' é a distância entre as durações correspondentes a $\theta^{(t)}$ e a 4.4° , e $T' = 25\text{ms}$ é o tamanho do intervalo de tempo registrado, então a proporção de D' em T' também deve ser P . Disso obtém-se uma duração de 37.5ms , resultado que é truncado para 37ms pois o argumento tempo deve ser um valor inteiro.

Os procedimentos relativos ao deslocamento foram idênticos àqueles realizados para o giro, gerando a “tabela de correspondência de passo”. Para ela, a duração mínima foi 50ms em vez de 25ms , totalizando 8 valores investigados. As amostras foram as distâncias percorridas, medidas com uma trena, e suas médias são exibidas na Tabela 3.4.

Embora os valores mínimos das tabelas sejam relativamente altos, não é feita interpolação abaixo deles. Nesse caso a duração do movimento é sempre definida como a menor conhecida. Essa decisão foi tomada porque quantidades de tempo menores que 50ms podem fazer as rodas deslizarem devido ao impulso dos motores, levando o robô a se mover além do desejado. Ainda assim, note que a tabela de correspondência de giro utiliza 25ms como dado inicial. Mesmo com a possibilidade de um movimento impreciso, isto foi necessário para que não se perdesse mais informação. Caso contrário, qualquer $\theta^{(t)} < -6.8^\circ$ ou $\theta^{(t)} < 4.4^\circ$ teria de ser interpretado como tais valores.

tempo (em ms)	Desvio padrão das amostras	Tamanho do passo (em cm)
50	0.07	1.50
100	0.18	2.75
150	0.19	3.75
200	0.19	4.25
250	0.19	5.25
300	0.23	5.75
350	0.23	6.25
400	0.23	7.25

Tabela 3.4: Tabela de correspondência de passo.

Apesar dos tempos máximos tabelados serem menores que os limites superiores dos domínios de $\theta^{(t)}$ e $\Delta^{(t)}$, o código desenvolvido é capaz de calcular extrapolações quando necessário. Além disso, os testes do controlador realizados no MATLAB indicaram que os intervalos praticados para suas saídas são, na verdade, $[-23.1^\circ, 23.1^\circ]$ e $[0, 7.27]$, respectivamente.

Mesmo com todos esses cuidados, nem sempre é possível que a movimentação ocorra livre de falhas. Além da derrapagem das rodas, depois de algum tempo de uso elas costumam afrouxar, sendo necessário firmá-las. Os servos não são muito precisos, motivo pelo qual o robô não andava em linha reta. A solução para diminuir esse desvio foi adotar velocidades diferentes para cada roda: 20 para a direita e 13 para a esquerda. Isto pareceu satisfatório para os giros de forma geral e distâncias curtas de deslocamento (até 7cm aproximadamente). Outro fato interessante é que as rodas deveriam ficar paradas com velocidade nula (equivalente a 90° para os servos), mas isso não acontece. Por isso foram criadas as funções `conectarRodas()` e `desconectarRodas()`, a última servindo para cessar o movimento.

4. Testes e Resultados

Após concluir a implementação dos algoritmos, procedeu-se aos testes de navegação do robô real utilizando o controlador escolhido. O ambiente utilizado e suas configurações, bem como os resultados obtidos nas avaliações são apresentados neste capítulo. Foram realizados alguns testes preliminares para verificar se a performance do robô correspondia ao esperado. Uma vez feitos os ajustes necessários, deu-se início a uma bateria de sete testes.

As características do ambiente são mostradas na Seção 4.1. A Seção 4.2 trata de todos os testes executados, trazendo uma discussão sobre os problemas encontrados.

4.1. Ambiente de Testes

O ambiente para os testes do robô real foi montado no Espaço Flex, uma sala multiuso do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais (iNCE)¹⁴. Esse local dispunha de área livre suficiente para a navegação do robô e foi possível manter o ambiente ali por alguns dias. Mesas viradas foram utilizadas como as paredes do corredor, aberto em um dos lados. A parede que fica inicialmente atrás do robô tem aproximadamente 2.30m de comprimento, enquanto as laterais medem em torno de 4.8m. Os obstáculos (cinco no total) são garrafas pet de 2L, com um pouco de água para não caírem no caso de colisão do robô. O robô sempre começa a navegação no mesmo lugar, mudando apenas o valor de ϕ^0 . As posições dos obstáculos e a posição inicial do robô foram marcadas no chão. As Figuras de 4.1 a 4.4 mostram o ambiente de diferentes ângulos. Conforme o modelo, a direção horizontal aponta para a saída do corredor, exibida na Figura 4.4.

¹⁴ <<http://portal.nce.ufrj.br/>>. Acesso em: 26-08-2012.



Figura 4.1: Ambiente de testes do robô real.

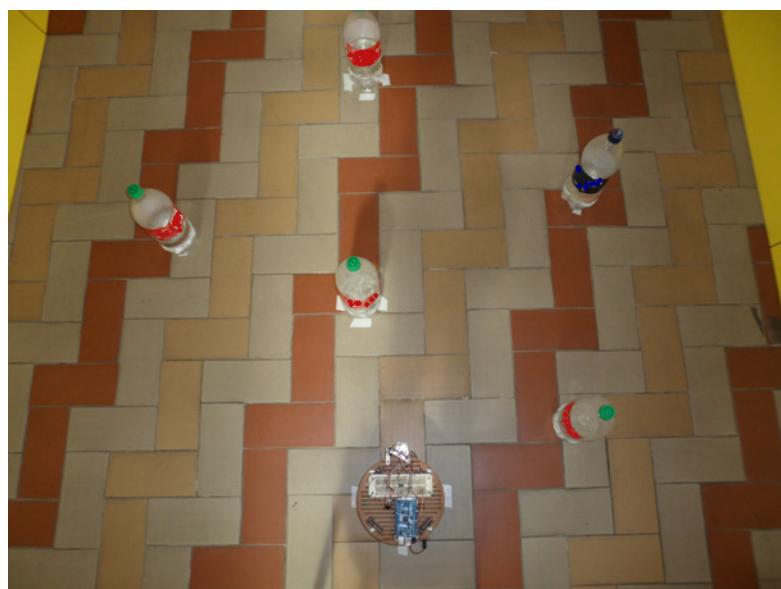


Figura 4.2: Ambiente de testes do robô real visto de cima.



Figura 4.3: Ambiente de testes do robô real.



Figura 4.4: Ambiente de testes, mostrando a saída do corredor (direção horizontal).

4.2. Testes Realizados

Foram realizados testes preliminares para averiguar a adequação do código implementado ao comportamento esperado da navegação. Foi escolhida a direção inicial do robô mostrada na Figura 4.4, ou seja, no mesmo sentido da saída do corredor ($\phi^0 = 0^\circ$). Dessa forma ele poderia começar andando em linha reta, pois os ajustes de ângulo do robô seriam

decorrentes apenas das manobras de desvio dos obstáculos. As posições dos obstáculos também são as mesmas vistas nas Figuras de 4.1 a 4.4.

Até então era usada a proporção 1:1, isto é, 1 unidade de medida no mundo virtual correspondia a 1cm no mundo real. A navegação, porém, não foi bem sucedida. Na maioria das vezes, ou o robô não conseguia desviar do obstáculo, ou o espaço entre eles durante a manobra era menor do que devia dado o tamanho do corpo fictício do robô. Cogitou-se que isso fosse em decorrência de valores de distância pequenos (conjuntos MP [0, 0, 10, 15] ou Pe [10, 20, 30]; Seção 2.2), que não garantiam espaço suficiente para movimentar-se em torno do objeto mais próximo. Isso se justifica porque o robô real tem 13.6cm de raio fictício embora o controlador tenha sido estruturado para operar sobre um robô com raio de 6 unidades. Assim, um obstáculo entre 6cm e 13.6cm está, na prática, dentro da área ocupada pelo corpo do robô. De fato, trocando a proporção para 1:2.26 (Seção 3.4.1) esse problema foi solucionado: o robô deixou de colidir, além de manter uma distância razoável dos objetos – uma melhora significativa na navegação. Ele também foi capaz de acertar sua direção para a saída do ambiente depois que os obstáculos foram deixados para trás. Diante destes resultados, o valor de referência da horizontal foi armazenado definitivamente e pôde-se dar início aos demais experimentos.

A bateria de testes foi realizada com os obstáculos e o robô nas mesmas posições anteriores, e o valor de referência da horizontal usado foi -71.43° . Novamente com $\phi^0 = 0^\circ$, o primeiro teste teve um bom resultado, similar ao obtido na averiguacão preliminar. A trajetória percorrida pelo robô está esquematizada na Figura 4.5. Em seguida utilizou-se $\phi^0 = -90^\circ$ (Figura 4.6). O robô desviou satisfatoriamente dos obstáculos, mas passou a se dirigir para a direita do ambiente (direita da foto) depois que os deixou para trás, inclusive colidindo com a parede desse lado.



Figura 4.5: Trajetória aproximada do robô no primeiro teste ($\phi^0 = 0^\circ$).

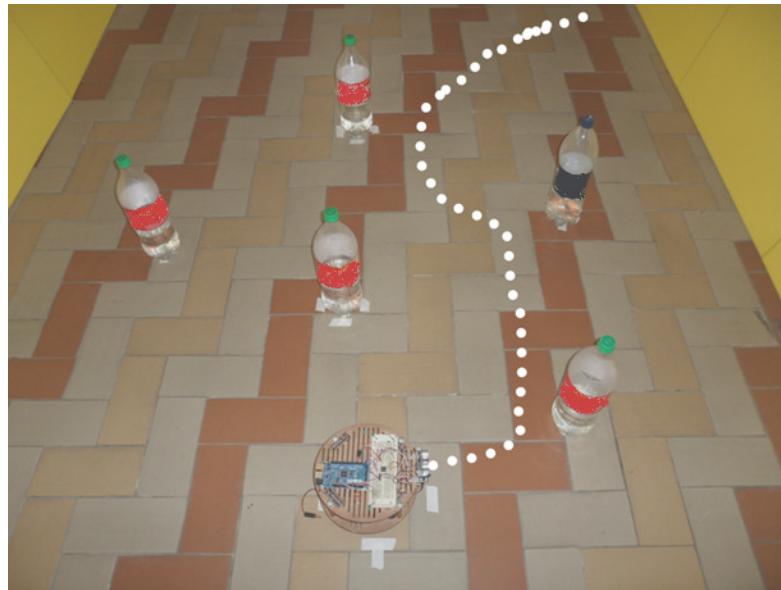


Figura 4.6: Configuração inicial do segundo teste ($\phi^0 = -90^\circ$) e trajetória aproximada do robô.

Com $\phi^0 = 90^\circ$ (Figura 4.7) o comportamento do robô se repetiu. No entanto, como ele saiu da zona de obstáculos pela esquerda, mesmo divergindo para a direita ele não chegou a colidir com a parede depois de um tempo considerável. Já o quarto teste, com $\phi^0 = -45^\circ$ (Figura 4.8), apresentou o resultado mais anômalo de todos. O robô não colidiu com nenhum obstáculo e chegou a ficar com o caminho livre na direção da saída do corredor. Entretanto, em vez de seguir em frente, ele passou a virar cada vez mais à direita até ficar na direção oposta à saída (Figura 4.9).

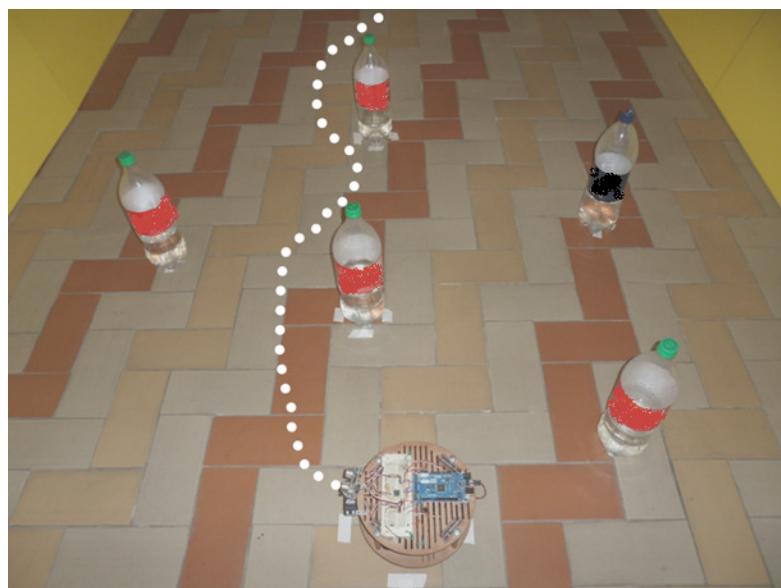


Figura 4.7: Configuração inicial do terceiro teste ($\phi^0 = 90^\circ$) e trajetória aproximada do robô.



Figura 4.8: Configuração inicial do quarto teste ($\phi^0 = -45^\circ$) e trajetória aproximada do robô.



Figura 4.9: O robô virou para o sentido contrário ao da saída do corredor no quarto teste.

Tais problemas trouxeram a suspeita de que alguma interferência eletromagnética tivesse alterado o valor da horizontal do ambiente, ou seja, o robô a considerava como -71.43° , mas isso não correspondia mais à realidade. Foi feita a medição desse valor de referência novamente, obtendo-se -67.70° . Os testes seguintes passaram, então, a utilizá-lo.

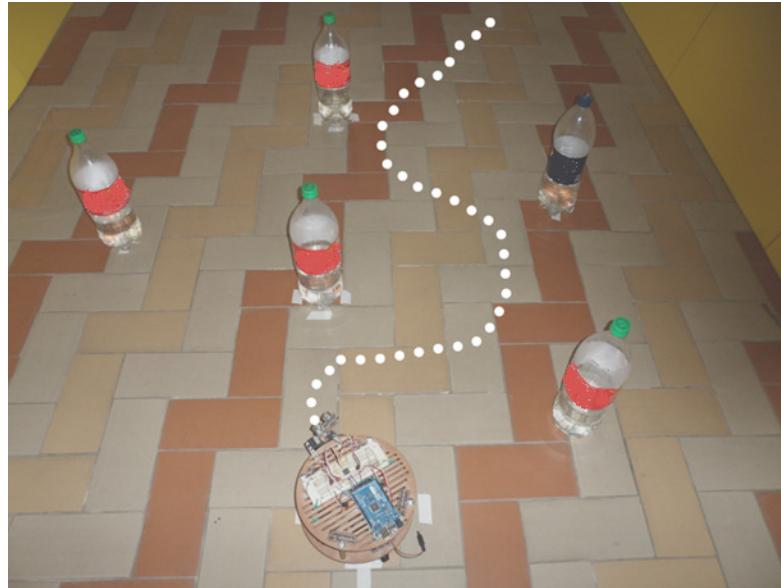


Figura 4.10: Configuração inicial do quinto teste ($\phi^0 = 45^\circ$) e trajetória aproximada do robô.

O quinto teste começou com $\phi^0 = 45^\circ$ (Figura 4.10) e mais uma vez não houve colisão com os obstáculos, mas o robô continuou tendendo para a direita. O sexto e o sétimo testes, os últimos realizados, tiveram $\phi^0 = 0^\circ$. Essa direção inicial foi repetida para que o desempenho do robô com o novo valor de referência da horizontal pudesse ser comparado com um resultado anterior. Em ambos os casos a tendência para a direita persistiu, mas houve um detalhe interessante. Note que há dois obstáculos na direção de saída do robô. Nos testes 1 e 6 ele desviou pela esquerda do primeiro obstáculo e pela direita do segundo (Figura 4.11). Já no sétimo, ele primeiro foi pela direita, e depois pela esquerda (Figura 4.12). Idealmente isto não deveria acontecer, pois sistemas fuzzy não produzem resultados aleatórios, e sim determinísticos. Ou seja, uma vez que suas entradas são as mesmas, a saída não deve mudar (Yen, Langari, 1999). Logo, nos casos em questão o controlador recebeu entradas distintas, conduzindo o robô por caminhos diferentes. A Figura 4.13 mostra a trajetória aproximada do robô no teste 7, enquanto o percurso realizado no sexto teste é idêntico ao do primeiro (Figura 4.5).

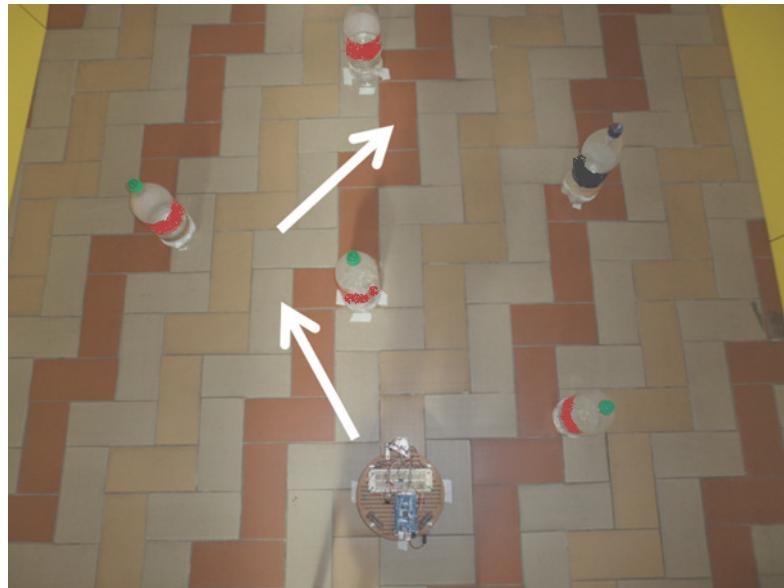


Figura 4.11: Indicação das direções tomadas pelo robô nos testes 1 e 6.

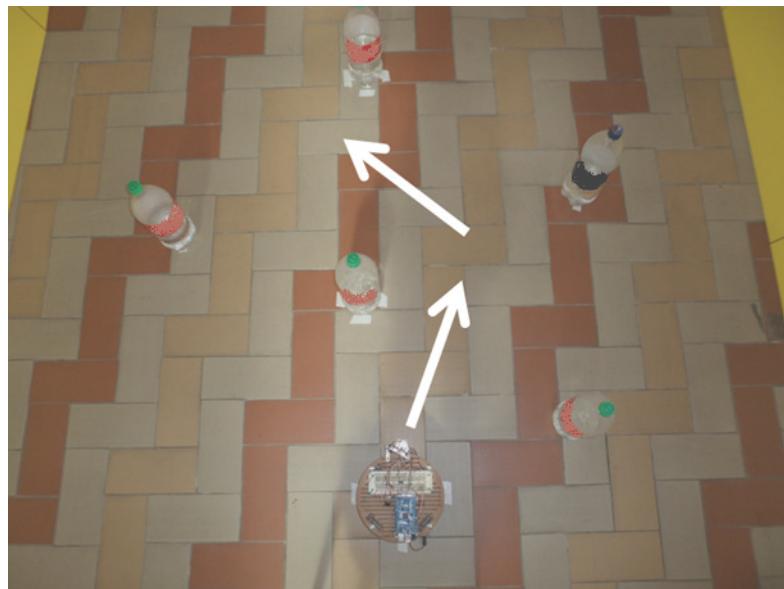


Figura 4.12: Indicação das direções tomadas pelo robô no sétimo teste.

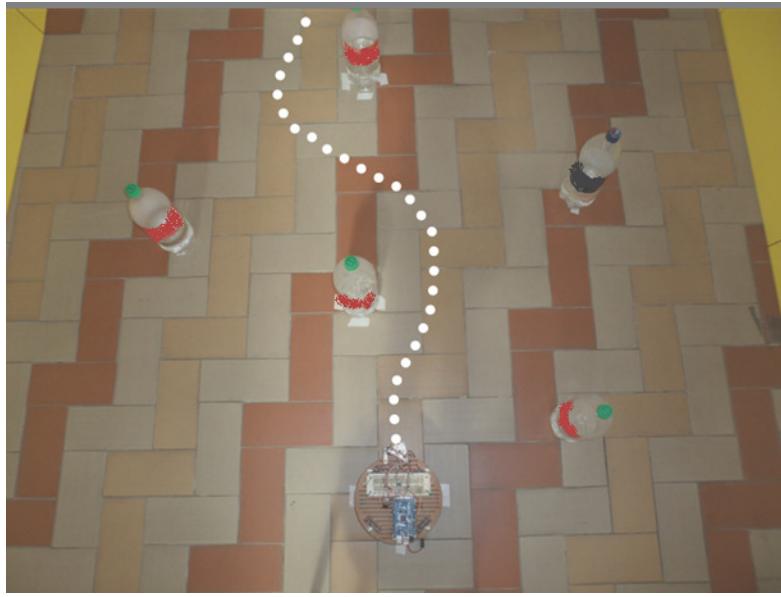


Figura 4.13: Trajetória aproximada do robô no teste 7.

Acredita-se que os problemas e anomalias identificados ao longo dos testes tenham origem na propagação de erro nos dados em memória, na imprecisão de alguns equipamentos e em interferências no ambiente. Os algoritmos desenvolvidos e implementados foram todos avaliados durante a execução deste trabalho, e funcionaram conforme desejado. Por isso, a chance de que sejam eles a fonte dessas anormalidades é baixa. O controlador utilizado obteve bons resultados no ambiente simulado no MATLAB (Mota, 2010), portanto ele também foi descartado como causador de tais problemas. Por outro lado, como foi dito na Seção 3.4.1, as frequentes operações sobre os dados contidos na memória do robô os tornam suscetíveis ao acúmulo de erro, o que poderia prejudicar a navegação. O fato de o robô tender a se movimentar para a direita após sair da zona com obstáculos muito provavelmente se deve a interferências eletromagnéticas captadas pelo sensor de campo magnético. Como foi dito na Seção 3.2, ele é bastante consistente com as leituras de uma bússola de bolso, que também seria afetada nesse caso. Além disso, não necessariamente essas forças atuariam da mesma forma em todo o ambiente, distorcendo ainda mais os resultados esperados do sensor. Outro ponto que conduz a tal conclusão é que nos testes preliminares e no primeiro dos testes subsequentes o robô não desviou para a direita, seguindo satisfatoriamente na direção da saída do corredor. Quanto aos caminhos distintos tomados pelo robô ao partir com $\phi^0 = 0^\circ$, é possível que a principal responsável seja a derrapagem das rodas, levando-o a se mover mais do que deveria (Seção 3.5). Ainda pode-se mencionar a imprecisão do sensor ultrassônico de distância (Seção 3.3), fornecendo ao controlador entradas inconsistentes com a realidade. É provável que este também tenha sido o motivo da colisão com a parede no segundo teste. Como o robô foi capaz de desviar dos cinco obstáculos em todos os casos, mesmo quando estavam bem à sua frente, dificilmente ele optaria por colidir ainda que estivesse na direção interpretada como

certa. Logo, crê-se que os dados sobre distância coletados (e armazenados em memória) estivessem equivocados naquela ocasião.

Para realizar os testes era necessário dispor de uma sala ampla, que permitisse a construção de um ambiente com espaço suficiente para a movimentação do robô. Também era preciso que o ambiente permanecesse montado e intacto por alguns dias, e inicialmente não havia um lugar que atendesse a tais requisitos. A busca por um espaço adequado com datas e horários disponíveis se mostrou uma grande dificuldade, juntamente com o esforço necessário para reunir o material para a montagem do ambiente de testes e a montagem em si. Ao mesmo tempo, a única opção de sala encontrada e utilizada esteve disponível apenas por alguns dias. Sendo assim, estes motivos impediram fortemente a realização de testes mais numerosos e detalhados.

5. Conclusões

Uma vez apresentados o modelo de navegação, as etapas de desenvolvimento deste trabalho e os resultados dos testes, este capítulo resume as dificuldades e os problemas encontrados e propõe algumas melhorias. A Seção 5.1 lista os principais desafios e dificuldades enfrentados durante a realização do trabalho. Já na Seção 5.2 são tecidos comentários sobre alguns pontos de destaque. Também são mostradas contribuições e listadas sugestões de trabalhos futuros.

5.1. Desafios e Dificuldades

Nenhum dos integrantes do LabIC¹⁵ (Laboratório de Inteligência Computacional) detinha conhecimento sobre os sensores usados. Por isso foi necessário dedicar muito tempo à pesquisa sobre o funcionamento desses dispositivos, em especial o HM55B. Não foram encontradas referências sobre suas etapas de calibragem relacionadas ao Arduino, sendo necessário traduzir códigos em PBASIC para a linguagem desta plataforma – o que também demandou bastante empenho e pesquisa.

Para ambos os sensores foram necessários muitos testes de precisão, tanto para aferir seus limites de funcionamento quanto a acurácia de suas medições, e pode-se dizer que eles foram avaliados exaustivamente. O PING))) se mostrou bem menos preciso do que se esperava, tendo sido analisado seu desempenho na detecção de diferentes tipos, tamanhos e formatos de materiais. Já o HM55B foi testado em diferentes locais, revelando-se muito sensível a interferências eletromagnéticas, mesmo tentando aumentar a precisão de sua calibragem.

A ideia de se adotar um corpo fictício para o robô não existia inicialmente. Ela só veio com a necessidade de fazer coincidir seu pivô de giro com seu centro de referência para as medidas de distância a obstáculos. Foi necessário estudar qual seria a configuração mais adequada, e que não influenciasse tanto a movimentação do robô.

A troca do Arduino Uno pelo Arduino Mega ADK não foi imediata. A princípio não se dispunha de outra plataforma além do Uno, e por isso começou-se a pensar nas possíveis estratégias que o permitissem comportar o controlador dentro do seu limite de memória. Passada uma semana é que surgiu a possibilidade de se utilizar o Mega ADK no projeto.

Os servomotores das rodas foram outro ponto problemático deste trabalho. Primeiramente, a ideia original era adotar a mesma velocidade para ambos, mas, ao serem testados, o robô estava desviando para a sua direita. Somente depois de diversos ajustes e tentativas chegou-se aos valores utilizados, exigindo bastante tempo. Não bastasse isso, ainda havia a diferença entre a unidade de medida usada pelo Arduino para controlar os

¹⁵ <<http://www.labic.nce.ufrj.br/>>. Acesso em: 23-08-2012.

servomotores e as unidades usadas pelo controlador. A construção das tabelas de correspondência de giro e de passo demandou tempo e trabalho manual para efetuar todas as medições.

Além disso, houve problemas quanto ao ambiente de testes. Demorou-se para encontrar um local apropriado e para reunir os materiais necessários para sua montagem, e o tempo que a sala esteve disponível foi curto. Também houve dificuldade para montar e desmontar o ambiente, sendo necessária ajuda para tombar as mesas que serviram como paredes e reorganizar o lugar ao final de tudo. Tais fatores contribuíram fortemente para que não fossem realizados mais testes de navegação.

5.2. Comentários, Contribuições e Sugestões de Trabalhos Futuros

De tudo o que se realizou, o maior desafio do projeto foi a determinação de $\phi^{(t)}$ e da horizontal do ambiente. Além de ter sido necessário buscar estratégias para utilizar o sensor de campo magnético nessas tarefas, este dispositivo ainda precisou de diversos ajustes até funcionar corretamente.

A ideia de usar um motor de passo para girar o sensor ultrassônico de distância foi interessante na época, pois, de forma simples e precisa, pôde-se simular os três sensores propostos no modelo. Por outro lado, a precisão do PING))) às vezes é falha, o que pode diminuir a qualidade da navegação. No entanto, talvez a derrapagem das rodas seja o fator com maior potencial de prejudicá-la. Outra questão relevante é a impossibilidade de se controlar o quanto as rodas andam ou giram, sendo necessário obter dados de referência empiricamente e utilizá-los para aplicar interpolação.

Embora tenha havido falhas na navegação, relatadas no Capítulo 4, pode-se dizer que ela foi bem-sucedida num âmbito geral. Levando em consideração todos os testes a partir do uso da proporção 1:2.26 (preliminares ou não), o robô sempre foi capaz de evitar colisões na zona com obstáculos, e somente em um caso não conseguiu se guiar satisfatoriamente por entre eles. Apesar da pouca quantidade de testes, acredita-se que o objetivo deste trabalho foi cumprido: implementar, testar e desenvolver os algoritmos necessários para possibilitar a navegação autônoma do robô real de acordo com o modelo escolhido. Como cada funcionalidade codificada foi avaliada antes e depois de ser integrada às demais, tem-se segurança para crer que os algoritmos de fato funcionam corretamente. Essas mesmas avaliações mostraram que os sensores e os servomotores usados possuem “pontos fracos” que possivelmente atrapalhariam a navegação. Entretanto, eles são características ou limitações físicas desses dispositivos que não poderiam ser atenuadas com o que havia disponível. Exemplo disso é a tendência do robô em ir para a direita do ambiente, observada durante a maior parte da bateria de testes. Conforme foi cogitado na Seção 4.2, tal comportamento talvez

seja fruto de interferências eletromagnéticas no ambiente que estariam alterando o funcionamento do sensor HM55B, e um local livre desse tipo de ruído estava além das possibilidades naquele momento.

Ainda sobre o sensor de campo magnético, na Seção 2.3 foi dito que o material disponível no site do Arduino não contemplava sua calibragem. Por isso foi necessário estudar o código existente em PBASIC para codificá-lo na devida linguagem, o que produziu uma contribuição inesperada. Os algoritmos gerados (relatados na Seção 3.2) poderão, em breve, ser submetidos ao site do Arduino para colaborar com sua comunidade de usuários. Outros pontos interessantes foram a determinação do estado do robô apenas através de $\phi^{(t)}$ e a flexibilização das dimensões do ambiente. Estas possibilidades certamente são vantajosas, facilitando a aplicação do modelo de Mota (2010) no mundo real.

Com relação a trabalhos futuros, a primeira sugestão é o uso de uma conexão bluetooth, possibilitando a comunicação sem fio entre o robô e o computador. Isso facilitaria muitíssimo a visualização do log da navegação em curso, que posteriormente poderia ser usado para avaliar o desempenho mais detalhadamente. Também seria útil realizar mais testes variando a disposição e a quantidade dos obstáculos, bem como adotando outras direções iniciais para o robô. Substituir o sensor de distância ultrassônico por um infravermelho é uma estratégia que possivelmente aumentaria a acurácia da detecção. Outra ideia levantada foi a utilização de um regulador de tensão para a alimentação das rodas. Seriam trazidas mais melhorias para a locomoção do robô com o uso de servomotores mais precisos e de um tacômetro (ou conta-giros) para se ter controle sobre a distância percorrida. Além disso, talvez fosse interessante possibilitar que robô identificasse o final do ambiente. Desta forma, uma vez que ele saísse do corredor, seria parado automaticamente.

Referências Bibliográficas

ANTON, H.; BIVENS, I.; DAVIS, S. **Cálculo: Volume I.** 8^a ed. São Paulo: Bookman, 2007. 680p.

ANTON, H.; BIVENS, I.; DAVIS, S. **Cálculo: Volume II.** 8^a ed. São Paulo: Bookman, 2005. 680p.

MAMDANI, E.H. Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis. **IEEE Transactions on Computers**, v.C-26, n.12, p.1182-1191, Dec. 1977.

MORATORI, P.B. *et al.* Analysis of Stability of a Fuzzy Control System Developed to control a simulated robot. **The 2005 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2005)**, Nevada, p.726-730, May 2005.

MOTA, T.C. **Análise e Proposta de Controladores para Navegação Autônoma de um Robô Inteligente.** 2010. 131p. Dissertação (Mestrado em Informática) — Programa de Pós-Graduação em Informática, UFRJ, Rio de Janeiro, RJ, Brasil. 2010.

PARALLAX, INC.. **Hitachi HM55B Compass Module.** Disponível em: <<http://parallax.com/Portals/0/Downloads/docs/prod/compshop/HM55BModDocs.pdf>>. Acesso em: 11-08-2012.

PARALLAX, INC.. **PING))) Ultrasonic Distance Sensor.** Disponível em: <<http://www.parallax.com/Portals/0/Downloads/docs/prod/acc/28015-PING-v1.6.pdf>>. Acesso em: 11-08-2012.

SMITH, S.W. **Digital Signal Processing: A Practical Guide for Engineers and Scientists.** Massachusetts: Elsevier Science, 2003. 650p.

YEN, J.; LANGARI, R. **Fuzzy Logic: Intelligence, Control, and Information.** New Jersey: Prentice-Hall, 1999. 548p.

YOUNG, H.D.; FREEDMAN, R.A. **Física III: eletromagnetismo.** 12^a ed. São Paulo: Addison Wesley, 2009. 448p.