UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

RICARDO CARMINATI DE MELLO

NAVEGAÇÃO AUTÔNOMA DE ROBÔ UNICICLO: ESTUDOS DE CASOS EM UMA ABORDAGEM BASEADA EM COMPORTAMENTO

VITÓRIA – ES
DEZEMBRO/2015

RICARDO CARMINATI DE MELLO

NAVEGAÇÃO AUTÔNOMA DE ROBÔ UNICICLO: ESTUDOS DE CASOS EM UMA ABORDAGEM BASEADA EM COMPORTAMENTO

Parte manuscrita do Projeto de Graduação do aluno **Ricardo Carminati de Mello**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Ing. Edson De Paula Ferreira

VITÓRIA – ES
DEZEMBRO/2015

RICARDO CARMINATI DE MELLO

NAVEGAÇÃO AUTÔNOMA DE ROBÔ UNICICLO: ESTUDOS DE CASOS EM UMA ABORDAGEM BASEADA EM COMPORTAMENTO

Parte manuscrita do Projeto de Graduação do aluno **Ricardo Carminati de Mello**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovada em 15, de dezembro de 2015.

COMISSÃO EXAMINADORA:

_____

Prof. Dr. Ing. Edson De Paula Ferreira
Universidade Federal do Espírito Santo - UFES
Orientador

_____

Prof. Dr. Alessandro Mattedi
Universidade Federal do Espírito Santo - UFES
Examinador

_____

Eng. Leonardo de Assis Silva
Universidade Federal do Espírito Santo - UFES
Examinador

Aos meus pais e meu irmão.

Agradeço primeiramente aos meus pais e meu irmão por todo amor e carinho dedicados a mim. Agradeço pelo apoio incondicional, pelo exemplo e pela incomensurável paciência. Tudo que sou, devo a vocês.

Aos amigos que fiz no CT2, tanto àqueles que entraram no curso ao meu lado quanto aos que conheci depois, agradeço por todo o companheirismo. Agradeço pela ajuda nos estudos e pelos momentos de descontração, fundamentais para que eu seguisse em frente. Agradeço por se fazerem presentes sempre que preciso. Tenho vocês por irmãos.

Agradeço ao Prof. Dr. Ing. Edson de Paula Ferreira por toda a ajuda e orientação prestada, por sua disponibilidade e paciência, e pelo valioso estímulo ao pensamento crítico. Agradeço em geral aos professores do Departamento de Engenharia Elétrica pelas oportunidades de aprendizado.

Agradeço aos familiares, amigos e colegas de curso, por terem estado ao meu lado durante todo esse tempo.

Por fim, agradeço ao Pai por todas as oportunidades postas em meu caminho.

RESUMO

Um robô autônomo é aquele capaz de se movimentar sem supervisão humana direta, em ambientes sujeitos a mudanças detectáveis, tendo em vista alguma tarefa a ser realizada. A autonomia pode ser alcançada por meio de sensores que percebem o ambiente, dando ao robô a possibilidade de interpretá-lo, e maximizada pela mobilidade do robô, que permite que este navegue e atue com maior liberdade sobre seu entorno. O presente trabalho aborda aspectos do problema da autonomia na robótica sob o ponto de vista teórico e experimental. O teórico versa sobre o estudo da navegação autônoma de robôs móveis tendo por base os preceitos da robótica baseada em comportamentos. O experimental utiliza um robô tipo uniciclo, que é solução para problemas frequentes, que envolvem robôs de porte variável, nas mais diversas áreas de aplicação. Como estudos de caso, dois tipos de problemas, associados a tarefas específicas, foram propostos. A primeira tarefa consiste em navegar por uma representação de campo minado, buscando por representações de minas no chão. A segunda tarefa envolve a navegação por uma área delimitada, em busca de cilindros de metal, objetivando seu recolhimento e condução até uma região de base pré-estabelecida. Os dados experimentais para o presente projeto de graduação foram colhidos durante o intercâmbio do autor na Universidade Tecnológica Chalmers, na Suécia, a partir do uso do *hardware* disponibilizado pela universidade, que inclui o *kit* de robótica educacional *Boe-Bot* em conjunto com uma placa Arduino Uno. Neste trabalho são formalizados os conceitos supracitados e são explicitados e analisados os resultados alcançados com as experiências realizadas.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

FSM	*Finite-State Machine*
I/O	*Input/Output*
LED	*Light-Emitting Diode*
MIT	*Massachusetts Institute of Technology*
PWM	*Pulse Width Modulation*
SR	*Stimulus-Response*
UFES	Universidade Federal do Espírito Santo

SUMÁRIO

1 INTRODUÇÃO

1.1 Um olhar sobre robótica e autonomia

Ao longo do último século, os robôs saíram do imaginário coletivo e tornaram-se realidade. Apoderaram-se das mais variadas aparências para atuar em diversos ambientes, realizando uma enorme gama de tarefas. Fundamental para o setor industrial, a robótica teve seu grande sucesso quando do desenvolvimento e fabricação de braços robóticos, ou manipuladores: robôs capazes de realizar tarefas repetitivas com velocidade e precisão sobre-humanas (SIEGWART; NOURBAKHSH, 2004, p. 1).

Em contraste com os braços robóticos, robôs ditos não fixos são capazes de se locomover no ambiente no qual estão inseridos. Mobilidade pode também ser encarada como sinônimo de liberdade, permitindo ao robô, além de perceber e atuar sobre seu entorno, a possibilidade de navegar e explorar o ambiente. A categoria de robôs móveis engloba a utilização de formas diferentes de locomoção, como aquática e aérea (ISO, 2012). Neste trabalho considera-se apenas robôs que se locomovem sobre rodas, o que dá origem a uma vasta gama de problemas possíveis em função dos ambientes a serem explorados.

Murphy (2000, p. 16) estabelece que robôs podem ser bem aproveitados em três tipos de ocasiões: onde o ser humano corre risco significante, como aplicações nucleares, espaciais e militares; onde, por razões econômicas ou pela natureza do trabalho, o ser humano seja ineficiente, razão principal para a automação industrial; e para usos humanitários onde haja perigo, como, por exemplo, ao se retirar minas de um campo minado.

Em utilização ao redor do mundo, há robôs a rodas concebidos para atuar *off-road* e outros para caminhos mais regulares. Para o segundo caso, pode-se citar os utilizados em chão de fábrica e ambientes insalubres, e os utilizados em ambientes como escritórios e residências.

Neste trabalho é utilizado um robô tipo uniciclo: robô móvel que apresenta duas rodas de tração diferencial sobre um mesmo eixo e uma terceira roda, livre, para equilíbrio. O robô uniciclo, cujas características são explicadas em maiores detalhes no Apêndice A, apresenta vantagens como alta mobilidade e configuração simples de rodas e, devido a estas vantagens,

é o mais frequentemente utilizado tanto em robôs de pequeno porte como em aplicações industriais (SECCHI, 2012, p. 60).

Um robô dito inteligente se contrapõe àqueles classicamente utilizados na automação industrial por não atuar de forma irracional e repetitiva (MURPHY, 2000, p. 3). A relação entre inteligência e autonomia é definida de forma diferente por autores distintos e, por vezes, esses e outros conceitos se confundem.

Franklin e Graesser (1996), estabelecem que um agente autônomo é um sistema situado dentro de um ambiente, sendo parte deste, que sente o ambiente e age sobre ele, no tempo, em busca de seus próprios objetivos. Murphy (2000, p. 3) segue linha similar e define robô inteligente como uma criatura mecânica que pode funcionar de forma autônoma, o que indica que o robô pode operar de forma completamente independente sob todas condições razoáveis de operação e continuar para atingir seu objetivo.

Quanto à definição específica de autonomia, várias são as encontradas na literatura, cada uma mais adequada à uma área de estudo específica. Aqui, atendo-se à simplicidade, usa-se a definição dada por Wahde (2004, p. 2) de que um robô autônomo é aquele capaz de se movimentar livremente e sem supervisão humana direta, em ambientes sujeitos a mudanças repentinas, para atingir seus objetivos.

Atuar autonomamente permite que o robô não apenas substitua o indivíduo durante a execução da tarefa, mas também possibilita sua atuação em locais e situações onde a comunicação ou o controle remoto possam ser impossibilitados ou interrompidos. Também, o advento de máquinas capazes de agir de forma autônoma permite aplicações domésticas[1], industriais[2] e agrícolas[3], amenizando o trabalho do homem em determinadas áreas.

[1] Talvez o maior expoente desse tipo de robôs seja a Roomba, uma série de robôs autonomos usados na limpeza interna de casas e escritórios, produzida pela iRobot, empresa que tem Rodney Brooks como um de seus fundadores.

[2] O desenvolvimento de robôs autônomos para a indústria é área recente e tem por exemplos robôs como o *Baxter*, o *Little Helper* e o *KUKA youBot.*

[3] Um exemplo interessante desta aplicação pode ser conferido no trabalho de Van Henten et al. (2002) no desenvolvimento de um robô autônomo para colheita de pepinos.

1.2 Contexto e escopo

Disciplinas de robótica não fazem parte do currículo obrigatório do curso de graduação em Engenharia Elétrica da Universidade Federal do Espírito Santo, a UFES. Envolvendo robótica, autonomia e navegação, apenas disciplinas optativas são contempladas. Surgida a oportunidade de realizar um intercâmbio na Suécia, em um curso onde uma das disciplinas trata especificamente de agentes autônomos, o autor do presente trabalho vislumbrou a possibilidade de realizar seu projeto de graduação na área de navegação autônoma e assim o propôs.

O intercâmbio em questão foi realizado na *Chalmers University of Technology,* localizada na cidade de Gotemburgo, a segunda maior cidade da Suécia. Foram cursadas disciplinas do programa de mestrado *Complex Adaptive Systems*, de especial interesse aqui a disciplina *Autonomous Agents*, ministrada pelo Prof. Ph.D Mattias Wahde. A proposta desta é dar aos estudantes entendimento acerca de princípios de *design* de sistemas autônomos e proporcionar a oportunidade de aplicar o conhecimento na prática através da construção de um robô autônomo simples. Nessa disciplina, o semestre se divide em duas partes: a primeira teórica, indo desde o estudo da cinemática de movimento do robô móvel uniciclo até conceitos de navegação e exploração; e a segunda com enfoque prático, onde a partir de *kits* de robótica educacional fornecidos durante a disciplina, os alunos devem montar os robôs e realizar algumas tarefas avaliativas.

O presente projeto de graduação foi realizado em paralelo com o desenvolvimento da disciplina *Autonomous Agents*, cursada no primeiro semestre de 2015, aproveitando-se do *kit* de robótica educacional *Boe-Bot*, da *Parallax,* fornecido e dos conhecimentos adquiridos durante o curso. Os tipos de problemas propostos e as tarefas executadas, a serem descritas posteriormente, foram demonstradas em sala de aula e avaliadas. Neste trabalho são formalizados os conceitos e são explicitados e analisados os resultados alcançados com as experiências realizadas.

A ideia inicial da proposta era a montagem e programação de um robô capaz de navegar por um ambiente desconhecido, saindo de uma posição inicial até uma posição objetivo, envolvendo técnicas de sensoriamento, mapeamento, planejamento de trajetória e navegação.

Essa proposta foi elaborada cuidando-se para a manutenção de alguma flexibilidade de opções de realização.

Dentro das restrições encontradas, decidiu-se por não dar prioridade à ideia de mapeamento e planejamento de trajetória, e focar unicamente na parte de navegação autônoma, utilizando-se dos conceitos da robótica baseada em comportamento. Assim, foram executadas duas tarefas tipo, descritas em detalhes no Capítulo 4, trabalhando com problemas dentro das atividades possíveis no escopo da navegação autônoma.

1.3 Motivação

Na categoria de robôs móveis, a roda é de longe o mecanismo de locomoção mais utilizado. Nesta categoria, o robô mais popular é o tipo uniciclo, que pode ser encontrado em configurações que vão desde robôs de pequeno porte até robustos robôs industriais (SECCHI, 2012, p. 60). As pesquisas nessa área tendem a se concentrar em questões como manobrabilidade e controle (SIEGWART; NOURBAKHSH, 2004, p. 30). Estes dois aspectos caracterizariam o grau de mobilidade do robô. Embora este atributo, a mobilidade, garanta ao robô locomoção adequada pelo seu meio, é a autonomia que permite um funcionamento verdadeiramente cooperativo entre homem e máquina. Ao atuar autonomamente, sem supervisão humana, por dispensar um operador, o robô não só libera mão-de-obra humana para atuar em outras frentes, como também pode atuar ao lado do homem, compartilhando o espaço comum[4].

Encontram-se na literatura várias abordagens e arquiteturas para lidar com os mais variados problemas na área de autonomia, visando fornecer aos robôs pesquisados a capacidade de realizar com sucesso as tarefas para quais são programados. Estudar e entender as variadas abordagens utilizadas na robótica constitui um caminho para o crescimento do conhecimento na área. De acordo com Mataric (1992, tradução do autor):

> Com a robótica em sua infância, é prematuro limitarmo-nos à uma escolha particular de arquitetura. São necessários muitos experimentos e análises para produzir um corpo de algoritmos para diferentes problemas, e é necessário aplicar estes à

[4] Sobre esse tema, o autor recomenda a palestra de Rodney Brooks no TED2013, '*Why we will rely on robots'*. Disponível em: <https://www.ted.com/talks/rodney_brooks_why_we_will_rely_on_robots>. Acesso em: 30 nov. 2015

> diferentes tipos de arquiteturas para que se comece a obter um entendimento acerca dos problemas.

Dentre as abordagens existentes, aquela que deu origem ao que se convenciona chamar de robótica baseada em comportamentos busca inspiração em áreas da neurociência e biologia para o desenvolvimento e aplicação de novos métodos. Os resultados obtidos por essa abordagem foram observados em vários campos que se relacionam entre si, como vida artificial, computação evolutiva e sistemas multi-agentes (MATARIC, 1998).

Para Bishop (2002, tradução do autor), "abordagens baseadas em comportamento para controle de robôs são, em sua essência, uma forma de ver o mundo diferente daquela que estão acostumados engenheiros de sistemas de controle tradicionais". Ainda segundo ele, essas abordagens influenciaram a pesquisa em robótica principalmente porque funcionam em sistemas reais inseridos em ambientes reais e tendem a apresentar uma estrutura relativamente simples.

Mataric (1998) afirma que a robótica baseada em comportamento é capaz de demonstrar várias aplicações padrões de robôs móveis autônomos, tais quais desvio de obstáculos, navegação, mapeamento de terrenos e manipulação de objetos. Goldberg e Mataric (2001) relatam que esta abordagem já foi utilizada em aplicações que vão desde equipes de futebol de robôs até o controle de robôs aquáticos e robôs *ape-like*[5].

Muito embora se reconheça que atualmente problemas mais complexos tendam a ser abordados sob estratégias que fazem uso de arquiteturas que não sejam puramente baseadas em comportamento, sob fins educacionais faz-se interessante o estudo e a aplicação dos preceitos dessa área como ponto de início para um posterior aprofundamento no estudo da robótica.

Em suma, é a instigante possibilidade de se trabalhar com robótica, aliada à sua crescente popularização e presença na vida cotidiana, que motivam primariamente a realização deste

[5] Dois bons exemplos desse tipo de robô, *ape-like*, são o *RoboSimian*, da NASA, e *Charlie*, desenvolvido no *German Research Center for Artificial Intelligence.*

trabalho. O estudo da área e aplicação prática de uma abordagem consagrada na literatura se fazem propulsores desta motivação.

1.4 Análise da literatura especializada

Brooks (1986) deu largada àquilo que viria a se chamar de robótica baseadas em comportamentos com sua arquitetura de subsunção, que será melhor explanada no Capítulo 2. Em seu trabalho, Brooks define as bases do que seriam os comportamentos, descreve sua estrutura em camadas e mostra simulações de um robô com dois comportamentos básicos implementados. Estes dois comportamentos, “evitar colisões” e “vagar”, por serem premissas básicas de diversos sistemas, viriam a ser recorrentes em projetos futuros de vários pesquisadores.

Por apresentar uma abordagem alternativa às utilizadas tradicionalmente em robótica, Brooks (1987) argumentou desacreditando a importância de se realizar etapas de planejamento e modelagem do mundo real, etapas estas que constituem o núcleo das abordagens tradicionais, fortemente baseadas no sentir-planejar-atuar (do inglês *sense-plan-act*). Em conjunto com o grupo de robótica móvel do MIT, Brooks aplicou sua abordagem a uma gama de diferentes robôs móveis, aprofundando as técnicas de desenvolvimento de comportamentos e métodos para tomada de decisões (BROOKS; FLYNN, 1989; MAES; BROOKS, 1990; MATARIC, 1997). Abordagens similares também foram desenvolvidas e aplicadas com sucesso por outros pesquisadores, como Arkin (1987; 1998), Payton (1986) e Rosenblatt e Thorpe (1995), consolidando a chamada robótica baseadas em comportamentos.

O problema de autonomia em robótica é extensivamente abordado na literatura e este trabalho se apoia, em grande parte, nos livros de Murphy (2000) e Siegwart e Nourbakhsh (2004), *Introduction to AI Robotics* e *Introduction to Autonomous Mobile Robots*, respectivamente. As notas de aula de Wahde (2015) constituem a obra de referência principal utilizada durante a parte experimental deste trabalho, por configurar a base da disciplina *Autonomous Agents*, onde os experimentos foram realizados. Tratando especificamente sobre a abordagem baseada em comportamentos, o livro de Arkin (1998), *Behavior-Based Robotics*, também teve papel principal na pesquisa realizada. Ainda sobre esta abordagem, além dos artigos já citados, os

trabalhos de Mataric (1992; 1997; 1998), orientanda de Rodney Brooks em seu mestrado e Ph.D, são amplamente citados por serem altamente didáticos e ilustrativos.

O presente trabalho utiliza dessa literatura para realizar suas partes teórica e experimental. Durante a realização de ambas as partes, foram aprofundados e adotados apenas métodos e comportamentos mais básicos, similares aos fundamentados entre as décadas de 1980 e 90, inobservando conceitos fortalecidos posteriormente, como abordagens híbridas ou métodos mais sofisticados de aprendizado, integração e coordenação comportamental.

1.5 Caracterização do problema e dos objetivos gerais e específicos

Este trabalho tem por objetivo geral o estudo do problema da navegação autônoma na robótica, através dos preceitos da robótica baseada em comportamentos, com foco no robô tipo uniciclo. Tem também o objetivo básico de consolidar a formação do autor como engenheiro, agregando conceitos de diversas disciplinas do seu currículo.

Um projeto de graduação tem por objetivo básico o treinamento do aluno na aplicação de conhecimentos adquiridos em disciplinas cursadas durante a graduação em torno de um projeto. Ou seja, a obrigatoriedade do projeto advém da necessidade em se confirmar que o aluno é capaz de unir tais conhecimentos na realização de um projeto multidisciplinar. Embora a área de robótica não seja usualmente ministrada em disciplinas ofertadas para os alunos de graduação, esta integra por excelência os conhecimentos de várias áreas da engenharia elétrica. Assim, nesta área, são integrados conhecimentos da dinâmica do sistema físico em estudo, do seu sistema de controle, do sistema de programação embarcado e dos sistemas de acionamento e sensoriamento associados.

Como objetivos específicos tem-se a realização de dois estudos de caso, relativos à navegação autônoma, envolvendo dois problemas tipo propostos. O primeiro destes versa sobre o problema de detecção de minas terrestres, enquanto que o segundo trata do problema de forrageamento. A implementação destes objetivos específicos utiliza o *kit* de robótica educacional *Boe-Bot*, da *Parallax*, juntamente com uma placa Arduino Uno, para montar e programar um robô uniciclo capaz de navegar autonomamente.

Também são objetivos específicos deste trabalho:

- Propor uma tarefa experimental específica para cada um dos problemas tipo estudados;
- Realizar as tarefas experimentais, que englobam:
    - Estabelecer estratégias de atuação para cada uma das tarefas;
    - Integrar e adaptar adequadamente todo o aparato necessário de *hardware* do robô, interconectando *hardware* físico, de acionamento, de processamento e de sensoriamento;
    - Desenvolver, implementar e testar os programas a serem embarcados no *hardware* de processamento, para cada tarefa. Os programas são elaborados a partir dos preceitos da robótica baseada em comportamentos, tornando o robô capaz de receber e interpretar as informações advindas do sensoriamento e decidir autonomamente quais ações adotar para atingir seus objetivos;
- Verificar o desempenho do robô na realização das tarefas e realizar análise crítica quanto aos resultados obtidos, avaliando problemas, limitações e espaço para melhorias.

1.6 Descrição da metodologia do trabalho

Ao todo, o trabalho está organizado ao longo de seis capítulos. Neste primeiro capítulo o trabalho foi introduzido e contextualizado, tendo seu escopo definido. Foi também situado dentro da literatura existente e teve seus objetivos traçados.

O Capítulo 2 faz a conceituação teórica que fundamenta as técnicas utilizadas. Aborda os paradigmas deliberativo e reativo, e descreve os preceitos da robótica baseada em comportamentos.

O Capítulo 3 faz a descrição detalhada do *hardware* utilizado neste trabalho, sendo subdividido em seções que abordam, individualmente, o *hardware* mecânico, o de acionamento, de sensoriamento e de processamento.

No Capítulo 4 descrevem-se os estudos de casos acerca dos problemas tipos propostos e detalham-se as tarefas a serem realizadas.

O Capítulo 5 trata do desenvolvimento das tarefas, partindo das estratégias adotadas, passando pelas modificações e adaptações realizadas no *hardware*, pelo desenvolvimento da programação dos códigos embarcados no *hardware* de processamento, e finalizando com as descrições dos resultados obtidos e posteriores discussões acerca deles.

O Capítulo 6 faz o fechamento do trabalho, com suas conclusões e propostas de trabalhos futuros.

2 ROBÓTICA BASEADA EM COMPORTAMENTO

2.1 A abordagem deliberativa *versus* a reativa: uma introdução

Mataric (1992) estabelece que é a arquitetura do sistema robótico que provê os princípios de organização de um sistema de controle e que é a própria arquitetura de um robô autônomo que determina como este robô se comporta por lhe impor restrições na forma de sentir, processar e atuar.

Quanto às diferentes arquiteturas utilizadas em robótica, Sousa (2007, p. 11) afirma que, desde a formalização reativa apresentada por Brooks (1986), as diversas linhas de pesquisa podem ser agrupadas em duas abordagens distintas: a deliberativa e a reativa. A abordagem deliberativa também é tratada por tradicional ou clássica, por ser a mais antiga.

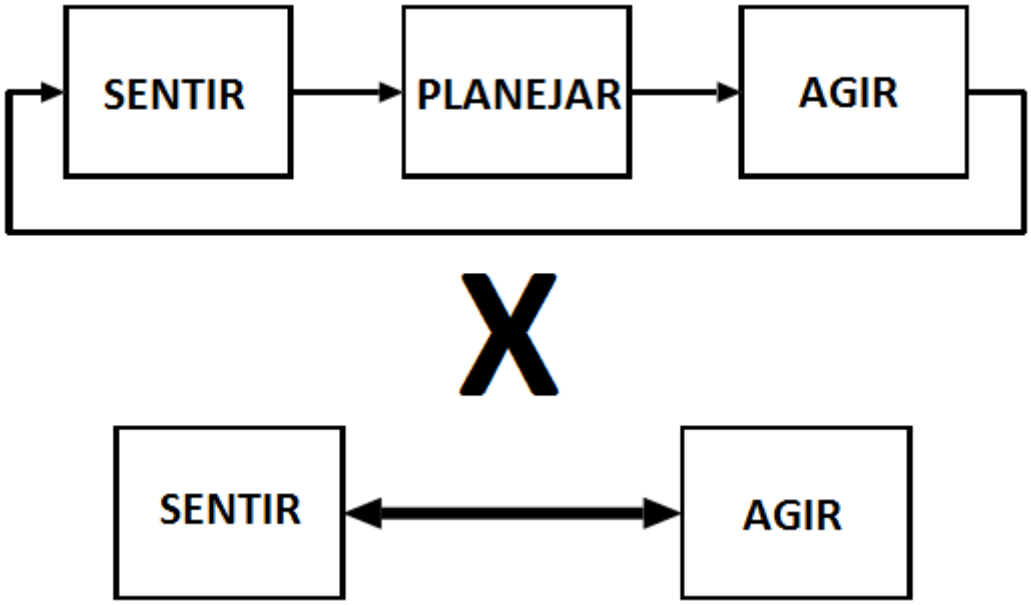
Sob a abordagem deliberativa, o robô opera de cima pra baixo (*top-down*), focando no planejamento para a execução de sua tarefa. A cada passo, o robô sente o mundo ao seu redor, planeja seu próximo movimento e age nesta direção. Toda informação proveniente do sensoriamento é reunida em um único modelo de mundo, que funciona como uma representação do mundo real que o robô utiliza para planejar e agir (MURPHY, 2000, p. 7).

Segundo Murphy (2000, p. 7), a forma como essa arquitetura se organiza é baseada em uma visão introspectiva de como o homem pensa. Brooks (1991) considera que a pesquisa em inteligência artificial teve, em seu início, o objetivo de replicar a inteligência humana em uma máquina, direção esta que viria a mudar com o passar dos anos graças às grandes dificuldades encontradas. Ainda de acordo com Brooks (1991), uma inteligência como a humana é por demais complexa e ainda pouco compreendida para ser corretamente decomposta e replicada.

Mataric (1992) afirma que, em contraposição à abordagem deliberativa, existe a abordagem puramente reativa, na qual a estratégia de controle é implementada como uma coleção de pares condição-ação. Este tipo de sistema consiste em uma coleção de regras puramente reativas e não possui modelos internos. O robô apenas seleciona, dentre as regras estabelecidas, a ação correta a ser tomada para cada conjunto de leituras dos sensores.

Enquanto que na abordagem deliberativa há uma noção de sequenciamento, onde primeiro é feito o sensoriamento, que fornece os dados para a constituição de um modelo do mundo real, e depois um planejamento, possibilitando que o robô atue na execução da tarefa, na abordagem reativa essa noção se perde e a informação advinda dos sensores está diretamente relacionada às ações executadas pelo robô. A Figura 1 ilustra ambas abordagens em diagramas.

Figura 1 – Abordagem deliberativa (acima) *versus* abordagem reativa (abaixo)



Fonte: Murphy (2000, modificado pelo autor).

Por acreditar que representações do mundo real são abstrações errôneas para construir sistemas inteligentes, Brooks (1991) propõe que o mundo real seja seu próprio modelo e apresenta em (BROOKS, 1986) uma abordagem alternativa que se baseia em comportamentos. Nesta abordagem – utilizando uma arquitetura que viria a ser conhecida como arquitetura de subsunção (do inglês *subsumption architecture*) - não há modelo interno e o robô age em resposta aos estímulos exteriores que recebe utilizado um sistema de controle em camadas paralelas hierarquizadas. Cada uma destas, um comportamento.

Arkin (1987) e Payton (1986) propuseram arquiteturas baseadas numa metodologia de campos potenciais que Mataric (1992) classifica como híbridas, ou seja, abordagens intermediárias entre as deliberativas e as puramente reativas. Ao mesmo tempo, Mataric (1992) considera uma arquitetura como a de subsunção diferente de uma puramente reativa,

porém não-híbrida, classificando-a como uma arquitetura baseada em comportamentos. Em literatura posterior, tanto Arkin (1998) quanto Murphy (2000) entendem que a arquitetura de subsunção é puramente reativa e equivalente àquelas de Arkin (1987) e Payton (1986), por todas poderem ser descritas em termos de comportamentos. Dessa forma, arquiteturas baseadas em comportamento podem ser consideradas puramente reativas, podendo também servir de base para arquiteturas híbridas, que misturam os conceitos reativos com os deliberativos. A discussão criteriosa de categorização foge do escopo deste trabalho, importando, aqui, apenas o fato de os trabalhos citados terem sido os principais na fundação daquilo que veio a ser chamado de robótica baseada em comportamentos (do inglês *behavior based robotics*), descrita em maiores detalhes na seção seguinte.

2.2 Fundamentos da robótica baseada em comportamentos

Tratando dos paradigmas deliberativo e reativo, Arkin (1998, p. 127) estabelece que sistemas baseados em comportamentos tem maior utilidade quando o mundo real não pode ser caracterizado ou modelado precisamente. Ele cita que quanto mais mutável for o ambiente durante o funcionamento do robô, o valor de qualquer plano gerado *a priori* se reduz e a abordagem por comportamentos passa a fazer cada vez mais sentido.

Arkin (1998, p. 67) considera que sistemas baseados em comportamentos fornecem ao robô meios de navegar em um mundo incerto e imprevisível sem planejamento, por dotá-lo de comportamentos que lidam com objetivos específicos de forma independente, coordenados por uma estrutura de controle. Ou seja, é possível que o robô atue autonomamente mesmo em ambientes com algum grau de dinamismo.

Mataric (1998) classifica a abordagem baseada em comportamentos como uma metodologia para projetar arquiteturas robóticas e controladores que dotem os robôs de comportamento inteligente. Nos sistemas que fazem uso dessa abordagem, o controlador do robô consiste de uma coleção de comportamentos, cada qual dedicado a um objetivo específico. Cada um desses comportamentos pode receber sinais advindos dos sensores do robô e de outros comportamentos, e enviar sinais para os acionadores do robô ou para outros comportamentos no sistema.

Brooks (1990) define comportamentos como um conjunto de regras independentes ativadas por diferentes motivos. De acordo com ele, presume-se que as regras que regem o sistema são observadas em paralelo e de forma assíncrona, e que podem existir relações de inibição e supressão entre os mecanismos que comandam o acionamento do robô. Segundo Mataric (1998), comportamentos devem ser relativamente simples, preparados para serem adicionados ao sistema de forma incremental. Ela estabelece que comportamentos são ativados em resposta a estímulos externos, condições internas ou outros comportamentos. O sistema como um todo ativa grupos inteiros de comportamentos de forma a aproveitar o paralelismo inerente à eles.

Murphy (2000, p. 111) cita comportamentos como blocos construtivos básicos do sistema e conclui que o comportamento geral demonstrado pelo robô é emergente. Ou seja, o comportamento que o robô apresenta é resultado da combinação dos comportamentos projetados em seu controlador. Brooks (1991) capitula que inteligência reside no olhar do observador e, em (BROOKS, 1986), afirma que comportamento complexo não precisa necessariamente ser produto de um sistema de controle complexo, naquilo que constitui um pensamento base da robótica baseada em comportamentos. Arkin (1998, p. 27) corrobora ao sintetizar que a inteligência surge da forma como o robô interage com seu entorno. Dessa forma, o comportamento emergente do robô, resultado da coordenação entre seus comportamentos individuais, é que o torna capaz de atingir seus objetivos autonomamente.

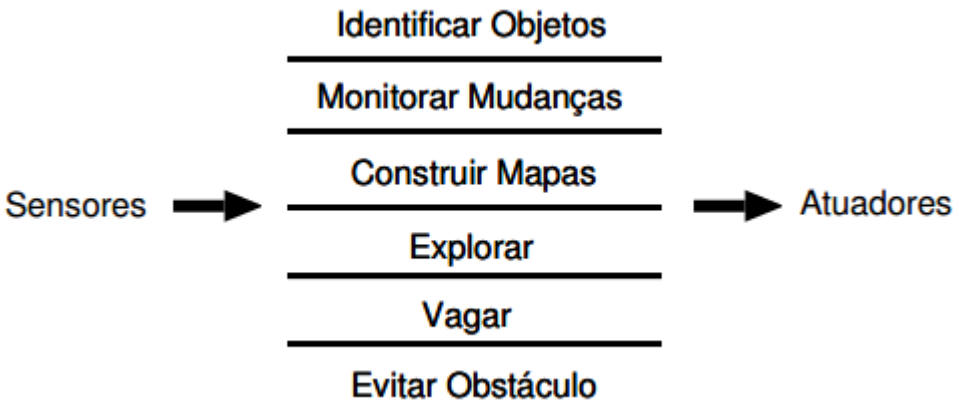
Embora, de acordo com Sousa (2007, p. 11), o termo comportamento esteja presente em muitos trabalhos de pesquisa em robótica móvel, inexistindo consenso quanto à sua definição exata, entende-se aqui que as definições citadas neste texto não são conflitantes e constituem suficiente caracterização para aquilo que é realizado neste trabalho. Na seção seguinte é feito um aprofundamento quanto ao projeto de comportamentos e coordenação destes.

2.3 Comportamentos: *design* e coordenação

Em seu trabalho seminal, Brooks (1986) compara a abordagem tradicional, onde o problema é decomposto em uma série de unidades funcionais, com sua própria abordagem, onde a decomposição do problema é feita em termos de comportamentos com objetivos próprios, tal qual ilustrado na Figura 2. Ele postula que a informação flui do ambiente do robô, pelo

sensoriamento, para o robô e, então, de volta ao ambiente, pela própria ação do robô. Os comportamentos são construídos de forma independente, porém incremental. Começa-se o projeto pelos comportamentos responsáveis pelos objetivos mais básicos, prioritários, verificando seu funcionamento, e, então, adicionando novos comportamentos, responsáveis por objetivos mais específicos.

Figura 2 – Decomposição do sistema de controle de um robô móvel em comportamentos



Fonte: Brooks (1986, modificado pelo autor).

Tal qual ilustrado na Figura 2, Brooks (1986) conceitua como comportamento mais básico o de evitar obstáculos, por ser o responsável pela manutenção da integridade física do robô. Construído este comportamento, após sua implementação, parte-se para a constituição do comportamento seguinte, vagar, que permite ao robô perambular pelo ambiente. Combinados, os dois comportamentos resultam em um robô que vaga sem colidir com obstáculos. O terceiro comportamento, explorar, possibilitaria ao robô detectar locais distantes que podem ser alcançados e seguir naquela direção, a fim de explorar o ambiente. Um a um, os comportamentos são construídos e incrementam o sistema, fornecendo ao robô maior capacidade de atuação.

Arkin (1998, p. 75), descreve um método experimental para projeto de comportamentos, onde a premissa básica é fornecer ao robô um conjunto limitado de habilidades, realizar experimentos no mundo real, observar o que funciona e o que não funciona, corrigir comportamentos imperfeitos e, só então, adicionar novos comportamentos, até que o sistema exiba performance satisfatória. Ou seja, uma variação do método utilizado por Brooks (1986), onde os comportamentos vão sendo incrementados em um sistema que já funciona adequadamente de maneira global dentro das premissas de maior prioridade adotadas.

Mataric (1992, tradução do autor) se aprofunda na descrição de projeto de comportamentos e escreve que:

> A decomposição e construção do sistema começam com a identificação do conjunto básico de reflexos que garantem uma "sobrevivência" segura em um ambiente dinâmico, não-estruturado. Por exemplo, no caso de um robô móvel, estes devem garantir uma navegação sem colisões. Os reflexos são uma coleção de regras simples que recebem sinais dos sensores, e enviam comandos diretamente aos atuadores. O projeto prossegue bottom-up, mas também incorpora restrições top-down.

Mataric (1998) também escreve que sistema baseados em comportamentos são tipicamente projetados para que o efeito dos comportamentos interaja com o ambiente, ao invés de interagir internamente no sistema, de forma a se aproveitar da interação dessas dinâmicas na composição do comportamento emergente do robô.

Murphy (2000, p. 77) ressalta que um comportamento pode servir de estímulo para ativar outro e que um sistema baseado em comportamento também pode apresentar memória, ou algum tipo de estado interno. Isso possibilita que comportamentos possam ser ativados em sequência e, ainda segundo Murphy (2000, p. 77), o sequenciamento de comportamentos básicos na construção de comportamentos complexos é um dos *insights* mais interessantes dessa abordagem.

Em geral, múltiplos comportamentos podem atuar sobre o mesmo grupo de acionadores, o que invariavelmente geraria conflitos caso estes comportamentos fossem ativados simultaneamente. Dessa forma, há de se haver técnicas que previnam estes conflitos, seja, por exemplo, através de priorização, com a ativação de um comportamento suprimindo a de outros, ou via fusão dos sinais de controle enviados aos atuadores pelos comportamentos ativados em conjunto.

De acordo com Mataric (1998), a coordenação de comportamentos é questão primordial, constituindo um desafio de controle que gera uma arbitração, ou seja, que decide qual comportamento é executado a cada momento de tempo. Segundo ela, em geral, ao prezar pela simplicidade, os sistemas utilizam ordenações fixas de prioridades entre os comportamentos, a

fim de garantir que apenas um dos comportamentos exerça controle sobre os atuadores de cada vez.

Arkin (1998, p. 111) descreve dois tipos de métodos de coordenação comportamental: métodos competitivos e cooperativos. O segundo prevê fusão comportamental, admitindo que múltiplos comportamentos se ativem simultaneamente e utilizando de algum mecanismo que pondere as saídas dos comportamentos e consiga combiná-las de forma coerente. Já o primeiro tipo de método, competitivo, resolve o problema de conflito entre os sinais provenientes dos comportamentos ao estabelecer uma hierarquia comportamental, seja através de uma priorização fixa, tal qual utilizada por Brooks (1986), ou por meio de algum outro mecanismo competitivo que selecione qual o comportamento mais conveniente a ser utilizado.

Na parte experimental deste trabalho, o mecanismo utilizado para a coordenação de comportamentos é, primordialmente, baseado nos métodos competitivos, admitindo uma hierarquia fixa para certos comportamentos e um método de seleção do comportamento mais conveniente. É de se ressaltar, também, que durante a seleção de comportamentos pode ser admitido algum grau de cooperação entre estes. Goldberg e Mataric (2001) utilizam método relativamente parecido em seu controlador homogêneo, e a própria representação em diagrama utilizada por eles inspira a forma como os comportamentos e sua interação com sensores e atuadores são representados no Capítulo 5.

Inserida em um sistema robótico onde cada comportamento é capaz de atingir seus objetivos, a correta coordenação de comportamentos permite que a atuação conjunta destes gere um comportamento global coerente. Assim, o comportamento emergente do robô atua no sentido de realizar as tarefas para o qual é designado, garantindo a este autonomia para realizá-las.

Na parte experimental deste trabalho, o robô é programado com base nos preceitos aqui explicados. As informações advindas do sensoriamento são, em geral, as únicas responsáveis pela ativação dos comportamentos. Estes comportamentos são projetados de forma independente e incremental, muito embora não sejam necessariamente implementados numa codificação que explicite essa independência. Comportamentos são criados com objetivos específicos e se deseja que sua interação com o ambiente resulte em um comportamento

emergente global coerente com a tarefa a ser realizada. A fim de se resolver conflitos causados por ativações simultâneas, há sempre uma forma de controle responsável por decidir qual comportamento pode deter controle do acionamento a cada momento. Essa estrutura de coordenação admite tanto supressão de um comportamento de ordem inferior por um de ordem superior, quanto seleção de qual comportamento é mais desejável em determinado momento, admitindo também, como explicado, algum grau de cooperação. Para cada uma das tarefas realizadas, o controlador e os comportamentos projetados são explicitados e explicados em maiores detalhes.

3 *HARDWARE* DO ROBÔ UTILIZADO

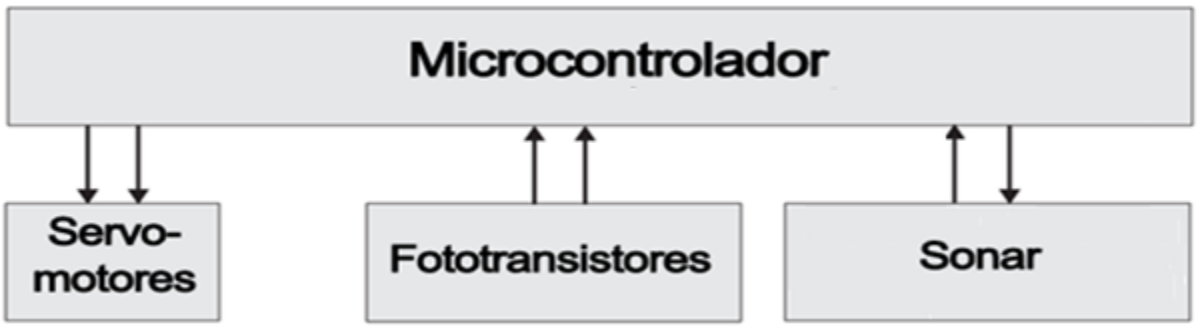
> Para conduzir pesquisas em robótica, robôs são necessários.
> Ronald C. Arkin

3.1 *Hardware*: visão geral

O *hardware* que compõe o robô pode ser subdivido em quatro: o *hardware* mecânico, o de acionamento, o de processamento e o de sensoriamento. Cada uma dessas quatro partes exerce uma função fundamental para que o robô possa funcionar adequadamente.

A Figura 3 ilustra a configuração básica de *hardware* de um robô tal qual o aqui utilizado. É este tipo de configuração que provém ao robô capacidade de processamento para atuar no ambiente onde as tarefas são caracterizadas (WAHDE, 2015, p. 19).

Figura 3 – Exemplo de configuração de *hardware*



Fonte: Wahde (2015, modificado pelo autor).

O microcontrolador, que constitui o *hardware* de processamento, deve receber os sinais provenientes do sensoriamento, ser capaz de interpretá-los em seu *software* – a fim definir qual, ou quais, comportamentos adotar – e controlar os servomotores, que integram o *hardware* de acionamento do robô, para realizar a navegação da forma desejada. Na figura, o microcontrolador recebe os sinais provenientes dos sensores, que compõem o *hardware* de sensoriamento: os fototransistores, que enviam informação de forma contínua, e o sonar, que se comunica em via de mão dupla com o microcontrolador, pois é este que define quando aquele atua. Esta configuração específica de sensoriamento é colocada aqui de forma

ilustrativa.

Toda essa estrutura, microcontrolador, servomotores e sensores, deve estar montada sobre uma plataforma para que o robô possa existir de forma funcional, no chamado *hardware* mecânico, ou físico. Um chassi leve de alumínio, com suporte para a placa do microcontrolador, para os sensores e servomotores, em conjunto com um par de rodas conectadas aos eixos dos motores e a roda castor fornecendo a sustentação é uma solução usual para o robô uniciclo e é a adotada aqui na utilização do *kit* de robótica *Boe-Bot*.

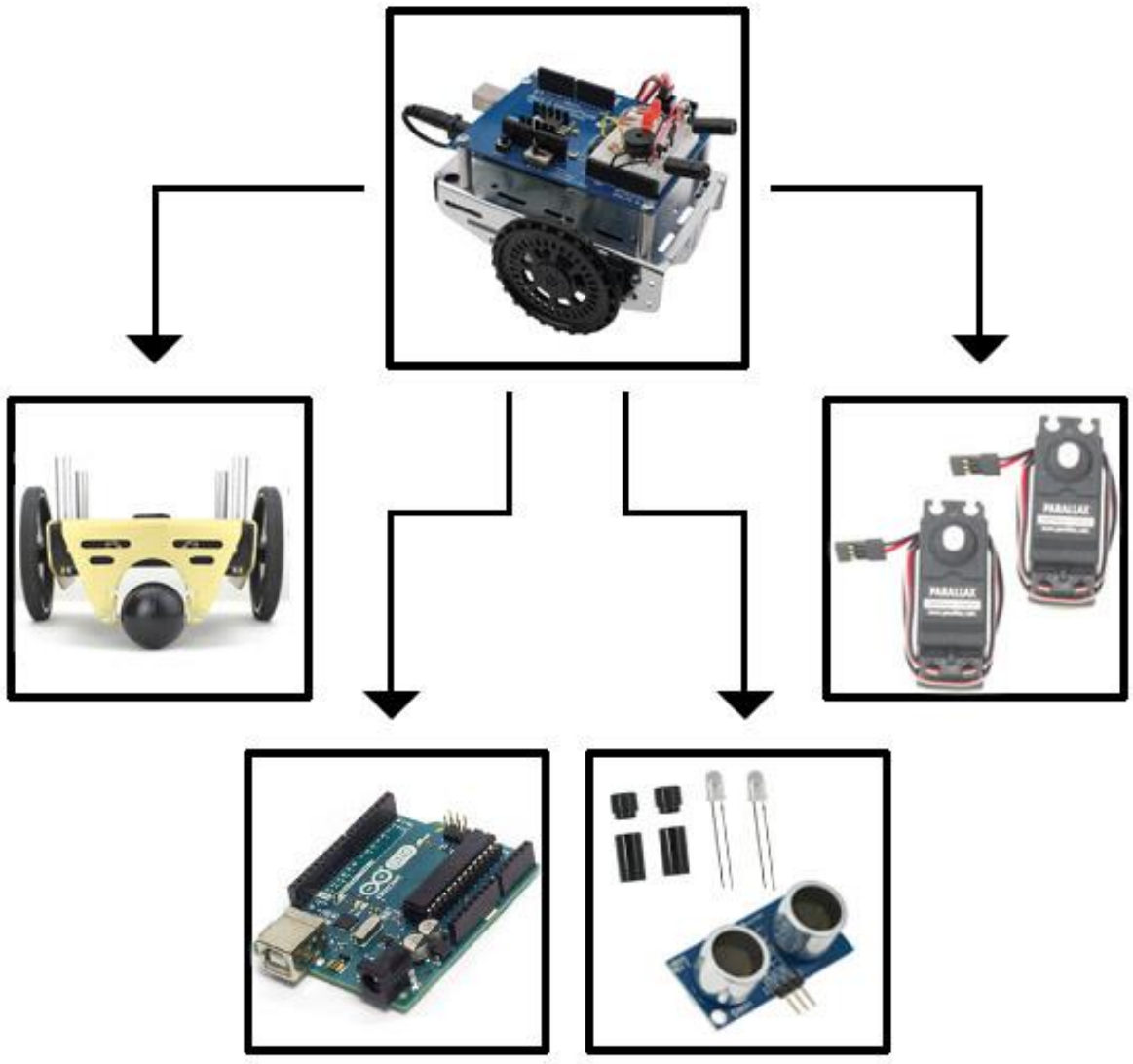
Segundo Miranda (2006, p. 19), vários são os *kits* de robótica educativa existentes, porém poucos são os produtos disponíveis no mercado brasileiro que possuem boa relação entre capacidade de recursos e custo. Embora auxiliem na popularização e disseminação do estudo da robótica, estes *kits*, em geral, tem custo proibitivo para a realidade brasileira (FILHO; GONÇALVES, 2008), e é usual que estes produtos não apresentem muita adaptabilidade quanto à interação com dispositivos de outros fabricantes (CÉSAR; BONILLA, 2007).

O *kit Boe-Bot* da *Parallax Inc.* custa em torno de US$160,00 e apresenta uma série de vantagens interessantes aos alunos sem experiência em robótica, dentre elas a facilidade para uso e a possibilidade de customização e adaptação de circuitos e sensores para diferentes tarefas (PARALLAX, 2015). O *kit* se constitui de um robô móvel uniciclo: corpo de metal, rodas do eixo principal e roda castor. Também se fazem presentes: a placa *Board of Education*, equipada com o microcontrolador BASIC Stamp 2, os servomotores – que constituem o *hardware* de acionamento – e alguns sensores e componentes eletrônicos, como resistores e LED’s.

A *Parallax* também vende, junto da mesma estrutura física clássica do *Boe-Bot*, o *kit* com um *shield* para Arduino, que substitui a *Board of Education* e é compatível com o popular Arduino Uno. O *shield*, além de ser responsável pela conexão com o Arduino, também apresenta uma matriz de contatos (ou *protoboard*) para a montagem de circuitos. Dessa forma, o robô pode ser adaptado para trabalhar com uma plataforma de processamento atualmente mais difundida no meio educacional. Esta é a configuração utilizada neste trabalho: o *kit Boe-Bot* com shield para Arduino em conjunto com o Arduino Uno.

A Figura 4 traz o *kit Boe-Bot* com *shield* para Arduino montado em sua configuração básica e sua decomposição nas quatro componentes de *hardware* descritas: mecânico, processamento, sensoriamento e acionamento.

Figura 4 – Decomposição ilustrativa do *kit Boe-Bot* com Arduino Uno em suas quatro componentes de *hardware*



Fonte: Arduino (2015, modificado pelo autor); Parallax (2010; 2013, modificado pelo autor).

Cada uma dessas partes é explicada em detalhes nas seções subsequentes.

3.2 *Hardware* mecânico

O *hardware* físico do *kit* engloba um chassi de alumínio, uma série de parafusos, conectores e suportes, as rodas de tração e a roda castor. Sobre o chassi do robô, quatro suportes são

utilizados para sustentar o *shield*, que, por sua vez, sustenta o Arduino. Abaixo do chassi são afixados os servomotores, que tem as rodas de tração presas aos seus eixos. Também abaixo do chassi são conectadas a roda castor, para fornecer sustentação à estrutura, e o suporte para as pilhas, que alimentam toda a eletrônica do robô.

Essa é uma configuração leve e de fácil montagem. Também é altamente customizável graças às várias entradas presentes no chassi, que permitem grande facilidade na adição de sensores e estruturas físicas.

3.3 *Hardware* de processamento

Arduino é uma plataforma *open source* de prototipagem pensada para ser de fácil utilização para iniciantes, embora também atenda profissionais mais experientes. O baixo custo desse tipo de plataforma é atraente para estudantes e auxiliou na popularização da mesma, utilizada em projetos que vão desde simples aquisição do sinal de uma botoeira até o controle de processos mais complexos. Por ter código aberto e contar com uma forte comunidade de usuários, uma vasta gama de bibliotecas estão disponíveis na Web, junto com inúmeros exemplos de projetos finalizados e testados. A linguagem de programação utilizada é baseada em C++/Java, bastante populares atualmente, configurando outro atrativo.

O Arduino Uno é uma das placas da família Arduino, baseada no microcontrolador ATmega328P. Possui 14 pinos *I/O*, dos quais 6 podem ser usados como saídas PWM, 6 entradas analógicas e conexão USB. Também é reprogramável e pode ser alimentado tanto via USB, cabo de força próprio, ou baterias.

Trabalhando em conjunto com o *Boe-Bot*, o Arduino deve ler os sinais dos sensores acoplados ao robô, processá-los dentro do *software* embarcado e enviar os sinais de controle dos servomotores via PWM.

3.4 *Hardware* de acionamento

Para atuar sobre o ambiente em que está inserido, o robô deve fazer uso de motores para realizar seus movimentos. Na navegação do robô uniciclo são usados motores acoplados às rodas de tração diferencial para fazer com que estas girem e, consequentemente, o robô se locomova.

Os motores presentes no *kit* utilizado são servomotores de rotação contínua. Este tipo de servomotor difere do servomotor usual pois, ao receber sinais *PWM* de comando, não os interpreta como uma posição física fixa, mas sim em termos de velocidade e sentido (PARALLAX, 2010). Ou seja, o sinal proveniente do *hardware* de processamento é interpretado no *hardware* de acionamento diretamente como um comando de velocidade em determinado sentido. Aplicados aos dois servomotores acoplados às rodas do robô, estes sinais comandam sua navegação.

3.5 *Hardware* de sensoriamento

De acordo com Secchi (2012, p. 24), "o sistema de percepção de um robô móvel permite que este seja capaz de fazer frente a situações de alteração do ambiente, assim como reagir mediante eventos imprevistos enquanto navega". Para isso, deve-se existir um sistema sensorial adequado que possa fornecer ao robô as informações necessárias a respeito do seu entorno.

Dentre os sensores disponibilizados aqui para uso no robô, observa-se a ausência de sensores proprioceptivos, aqueles que medem valores do próprio sistema, como velocidade dos motores ou carga da bateria. Apenas sensores ditos externos, aqueles que medem valores do ambiente, são utilizados. Listam-se:

- Fototransistor: sensor óptico capaz de detectar variações na luminosidade. Utilizado em um circuito tal qual o da Figura 5, a variação da incidência de luz na base do fototransistor ocasiona uma variação na corrente que flui do coletor para o emissor, regulando a tensão observada sobre o resistor.

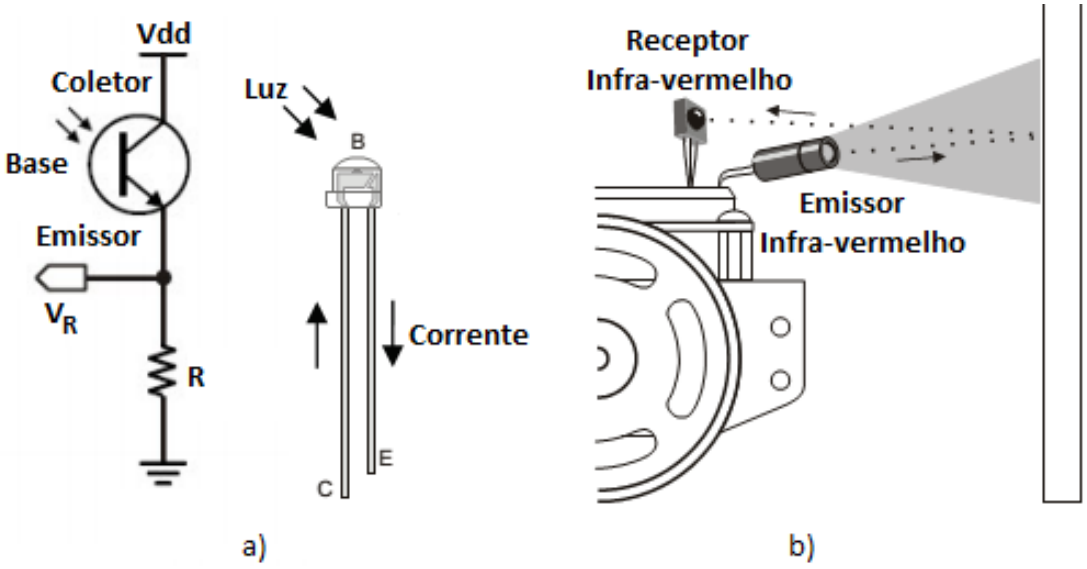
- Par emissor/receptor de IR: o par consiste em um LED emissor de radiação infravermelha (do inglês *infrared*, abreviado como IR) e um receptor de IR. O uso é tal qual descrito na Figura 5, o LED é encapsulado para direcionar a emissão de IR e o receptor detecta se há algum objeto à frente pela reflexão da radiação IR.

- Sensor ultrassônico: baseado em *time-of-flight*, este tipo de sensor recebe um comando do Arduino para emitir uma sequência de pulsos ultrassônicos e envia de volta ao

microcontrolador um pulso de duração proporcional ao tempo que o eco dos pulsos emitidos demora para retornar. Dessa forma, é possível calcular a distância do objeto que causou a reflexão da onda em relação ao sensor.

Observa-se na Figura 5 que a tensão V_R sobre o resistor conectado em série com o fototransistor pode ser lida por uma das entradas analógicas do Arduino, a fim de se detectar variações na luminosidade.

Figura 5 – Fototransistor e par emissor/receptor de IR



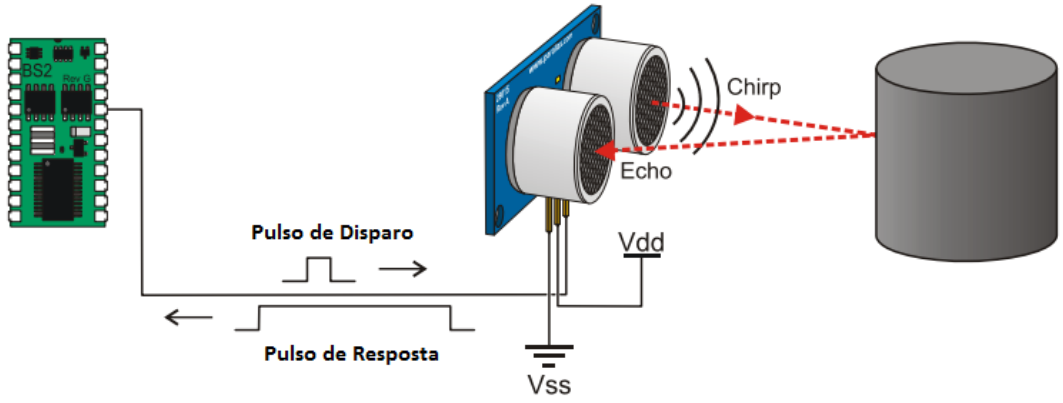
Legenda: a) Esquemático e representação do fototransistor
b) Ilustração do funcionamento do par emissor/receptor de IR
Fonte: Parallax (2010, modificado pelo autor).

O receptor IR é um dispositivo de três "pernas". Uma delas deve ser alimentada por uma tensão positiva, outra conectada ao terra do circuito, e a terceira deve ser ligada à uma entrada digital do Arduino[6], pelo fato de a própria saída do receptor ser digital. O receptor disponibilizado pela Parallax é projetado para detectar pulsos de IR em uma frequência próxima a 38kHz. Para a emissão de IR em pulsos pode-se utilizar o comando *tone* do Arduino. O ajuste da frequência interfere no *range* de detecção e frequências menores que 38kHz possibilitam a detecção de objetos mais distantes.

[6] É prudente que se use um pequeno resistor em série com a "perna" conectada ao Arduino, a fim de limitar a corrente de entrada no mesmo.

A Figura 6 trás a representação do princípio de funcionamento do sensor ultrassônico utilizado, chamado Ping))). Na figura, o sensor – devidamente alimentado – recebe um pulso de comando do microcontrolador a ele conectado e responde com um pulso de duração proporcional ao tempo-de-vôo do sinal ultrassônico, que nada mais é que o tempo que este sinal leva para deixar o sensor, ser refletido por um objeto, e retornar ao senor.

Figura 6 – Princípio de funcionamento do sensor ultrassônico utilizado



Fonte: Parallax (2013, modificado pelo autor).

O sensor Ping))), em condições ideais, é capaz de medir distâncias que vão desde um mínimo de 2cm até um máximo de 3m. Caso o tempo-de-voo do sinal seja superior ao equivalente à uma distância de 3m, o sensor retorna ao microcontrolador sua distância máxima. O sensor não é capaz de medir adequadamente distâncias para objetos que sejam demasiadamente pequenos para refletir eco suficiente e objetos cuja superfície reflexiva esteja em um ângulo fechado[7] em relação ao sensor. Ou seja, idealmente, o sensor mede acuradamente distâncias de objetos que estejam a sua frente, possuam tamanho considerável, e estejam a menos de 3m de distância.

Todos esses sensores são fornecidos pela própria Parallax para serem utilizados em conjunto com o *Boe-Bot*. O Arduino Uno lida com os dois primeiros tipos de sensores usando sua biblioteca padrão e com o sensor ultrassônico através da biblioteca *NewPing.h*[8].

[7] De acordo com Parallax (2013), o ângulo entre a superfície e a direção da onda ultrassônica deve ser maior do que 45º.

[8] Biblioteca criada por Tim Eckel. Disponível em <http://playground.arduino.cc/Code/NewPing>. Acesso em: 10 dez. 2015.

4 TIPOS DE PROBLEMAS PROPOSTOS

Para aplicação do conhecimento teórico adquirido, dois tipos de problemas de navegação foram propostos: o primeiro envolvendo o princípio de navegação na busca e detecção de minas terrestres; o segundo, problema do tipo de forrageamento (do inglês *foraging*), onde o robô deve navegar em busca de alvos, encontrá-los e carregá-los para local específico. De cada um desses problemas advém uma tarefa à qual foi submetido o robô. Essas tarefas foram realizadas de forma independente, com software específico desenvolvido para cada uma, adaptando-se a estrutura de hardware mecânico e de sensoriamento conforme conveniência, e seu desenvolvimento e execução são descritos no Capítulo 5. No capítulo atual, os problemas tipo são detalhados e as tarefas, definidas.

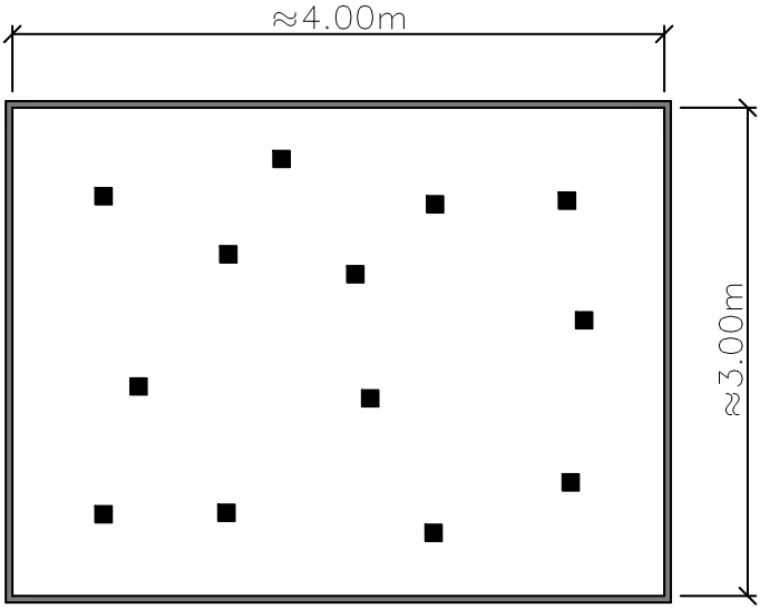
4.1 O Problema de detecção de minas e a tarefa 1

Minas terrestres foram usadas em praticamente todos os conflitos militares desde 1938 e apresentam sério risco à população civil (WALSH; WALSH, 2003). Este tipo de armamento tem, em geral, de 7 a 15cm de diâmetro e carregam entre 80 e 250g de explosivo, sendo normalmente detonadas por força aplicada sobre o detonador (TREVELYAN, 1997). Estimava-se, na década de 1990, que esse tipo de armamento matava e mutilava de 1.000 a 2.000 pessoas por mês, a maioria civis não combatentes (DOSWALD-BECK; HERBY; DORAIS-SLAKMON, 1995). Espalhadas aos milhões por diversas regiões do planeta, existe hoje um esforço coletivo e uma preocupação global com a limpeza de campos minados. Por ocuparem vastas regiões e apresentarem risco à quem trabalha com a limpeza destes campos, é interessante que se faça uso de grupos de robôs que sejam capazes de detectá-las e sinalizá-las. Esses robôs devem vagar pelo campo minado de forma ordenada e encontrar o maior número de minas possível. O sensoriamento pode ser realizado, por exemplo, via detector de metais, tal qual fazem as equipes compostas por humanos.

Para atuar neste tipo de problema, uma situação análoga a do campo minado foi trazida para o laboratório. É proposto um campo, também chamado de arena, que consiste em uma região retangular, de aproximadamente 12m² (≈4x3m), delimitada por paredes de 30cm de altura. O piso é liso, de coloração clara, próxima do branco, e dentro do campo são espalhados treze quadrados de papel preto com dez centímetros de lado, representando as minas. Ou seja, não se busca por objetos metálicos enterrados, mas pelo contraste entre a coloração do chão e das

minas, o que afeta o tipo de sensor a ser utilizado e traz considerável praticidade ao desenvolvimento da tarefa. Apenas um robô é utilizado. Este robô realiza a busca sozinho, de forma autônoma e independente. A iluminação do campo não é ideal, existindo sombras e áreas mais claras, onde a luz do sol perpassa por frestas nas persianas das janelas do laboratório, sendo esta uma dificuldade adicional para o sensoriamento. A Figura 7 ilustra a arena descrita.

Figura 7 – Representação ilustrativa da arena utilizada na Tarefa 1



Legenda: A área sombreada em cinza representa as paredes que delimitam a arena; os quadrados pretos são as minas por ela espalhadas.
Fonte: Produção do próprio autor.

Neste cenário, se propõe a Tarefa 1, cujo objetivo primário é adaptar o *hardware* do robô para a realização da tarefa e programá-lo, equipando-o com comportamentos que o permitam navegar pela representação do campo minado, buscar e identificar minas, e não colidir com obstáculos. Nesta tarefa, apenas as paredes que delimitam a arena são obstáculos à sua navegação. O robô deve buscar minas pela região da representação de campo minado proposta, sinalizando claramente ao encontrar uma mina, sem nunca passar por cima delas com nenhuma de suas rodas e sem colidir com as paredes que delimitam o campo. Ao todo, treze minas são espalhadas pela arena. O objetivo é que se encontre tantas quanto possível a cada rodada de execução da tarefa. Três rodadas serão conduzidas. Cada uma dessas rodadas tem por limite o tempo de dois minutos, clamando por velocidade e eficiência no projeto do robô.

4.2 O Problema de forrageamento e a tarefa 2

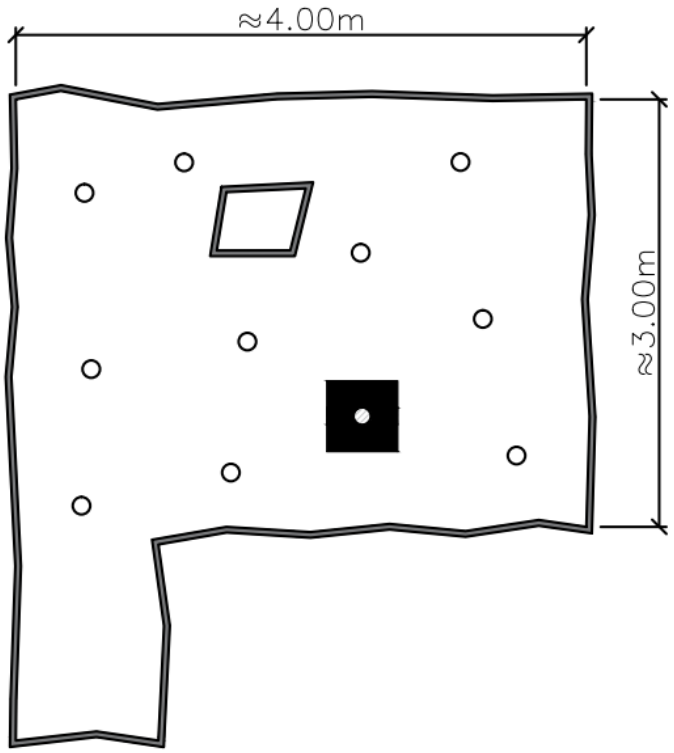
Muita pesquisa já foi realizada sobre a performance e atuação de robôs em tarefas de coleta e forrageamento (GOLDBERG; MATARIC, 2001). Arkin (1998, p. 130) descreve este tipo de tarefa como um robô que se move desde uma região de base procurando recolher objetos específicos. Aplicações típicas consistem em encontrar estes objetos e conduzir os exemplares coletados até a base.

Arkin (1998, p. 130) descreve, dentro dos princípios da robótica comportamental, três comportamentos básicos que devem existir no robô para o forrageamento: ***wander***, vagar pelo mundo em busca do objeto desejado; **adquirir**, mover-se em direção ao objeto quando detectado; e **recolher**, conduzir o objeto para base quando adquirido.

Goldberg e Mataric (2001) atuaram no problema de forrageamento subdividindo o ambiente em três regiões, de base, vazia (onde sabia-se de antemão não haver objetos) e de busca, e adotaram, por estratégia, conduzir os objetos encontrados um a um para a base.

Para atuar neste tipo de problema, aqui, é proposta uma arena, que consiste em uma região irregular, com piso liso de coloração clara, delimitada por paredes de 30cm de altura, onde o robô será posicionado. Dentro da arena estão espalhados, ao todo, dez cilindros de metal, que são os objetos buscados. Os cilindros são de alumínio, ocos, com aproximadamente 6cm de altura e 7cm de diâmetro. Além disso, está posicionada dentro da arena a *safe zone* (área segura, em tradução livre, a região de base), uma área quadrada, de 50cm de lado, demarcada por papel preto colado ao chão. No centro da *safe zone* existe um *beacon* infravermelho, posicionado a 10cm de altura, sobre um cilindro negro. O *beacon* é constituído por um anel de emissores de *IR*, uniformemente espalhados pelo anel, apontando para o exterior da *safe zone*, e pode ser usado pelo robô para localização desta. Contudo, irregularidades nos limites da arena e o obstáculo posicionado nesta geram áreas que impossibilitam a percepção do *beacon*. A Figura 8 traz uma ilustração da região delimitada pelas paredes da arena, com a *safe zone* e o *beacon* representados e os cilindros espalhados. Nesta ilustração, que objetiva apenas facilitar a visualização da configuração da arena, as dimensões das paredes e obstáculo são aproximadas, assim como e as dimensões da *safe zone* e dos cilindros, posicionados de forma meramente ilustrativas.

Figura 8 – Representação ilustrativa da arena utilizada na Tarefa 2



Legenda: A arena se apresenta como uma região retangular com uma espécie de corredor em anexo. As paredes e o obstáculo são representados pela área sombreada em cinza. A *safe zone* é constituída pela área preta, com o *beacon*, um círculo hachurado, em seu centro. Os cilindros são espalhados uniformemente pela parte ampla da arena.

Fonte: Produção do próprio autor.

Assim, é proposta a Tarefa 2. Dentro do cenário estabelecido, partindo de uma posição e orientação qualquer na arena, o robô deve encontrar os cilindros, alcançá-los, recolhê-los e conduzi-los até a *safe zone*. São realizadas três rodadas de execução e cada uma destas tem um tempo limite de cinco minutos. O objetivo é encontrar e depositar na *safe zone* tantos cilindros quanto possível.

5 DESENVOLVIMENTO DAS TAREFAS

> *Hands-on experience is the best way to learn about all the interdisciplinary aspects of robotics.*
> Rodney Brooks

No capítulo anterior foram expostos os tipos de problemas e as tarefas propostas. Neste, será dissertado acerca da realização das tarefas. Isso se dá, para cada uma destas, da seguinte maneira: primeiro, é discutida a estratégia proposta para a realização da tarefa em termos gerais de *hardware* e *software*; em seguida, são descritas as alterações necessárias no *hardware*, seja no físico ou no de sensoriamento, observando-se as limitações existentes; então é descrito o programa desenvolvido e embarcado no *hardware* de processamento, que rege o robô durante a execução da tarefa, com foco nos comportamentos construídos e sua coordenação; por fim, os resultados obtidos em cada rodada de execução da tarefa são relatados, analisados e discutidos.

5.1 Tarefa 1

A Tarefa 1 é uma representação do problema de detecção de minas e foi definida na Seção 4.1, onde constam todas as especificações pertinentes. Ela consiste, basicamente, na busca por quadrados de papel preto colados num chão branco, dentro de uma área delimitada por uma parede.

5.1.1 Estratégia proposta

Como a tarefa tem por objetivo que se encontre o máximo de “minas” – e esta é a última utilização deste termo entre aspas, sendo entendido daqui em diante não literalmente, mas como sua representação, quadrados de papel preto colados no chão – dentro de um tempo limitado, o projeto do robô deve focar em dois pontos específicos: velocidade de busca e eficiência na localização das minas.

A velocidade de busca deve ser maximizada para que se possa vasculhar a maior área possível dentro do tempo disponível. Para tal, combinam-se duas ideias: o robô deve navegar o quão rápido puder, e o sensor detector de minas deve ter a maior abrangência de área de busca que se possa alcançar. Como a arena em que o robô está inserido é por ele desconhecida, e por não

haver odometria[9], não é possível que se saiba onde o robô está em determinado momento e nem por onde ele já passou.

A estratégia de navegação adotada faz com que o robô ande em uma linha – quase – reta até que encontre uma mina ou um obstáculo. Um pequeno grau de aleatoriedade deve ser adicionado ao sinal que controla os servomotores para que a velocidade de ambos não seja sempre a mesma, fazendo com que o robô descreva uma trajetória levemente irregular. Ao encontrar uma mina, ou obstáculo, o robô deve rotacionar em sentido adequado e, só então, seguir novamente à frente.

A eficiência na localização das minas se faz necessária por dois motivos: o primeiro advém do próprio objetivo da tarefa, onde se deseja encontrar o maior número de minas, sendo inconveniente a não detecção de uma mina que deveria ter sido detectada; o segundo é, talvez, menos óbvio, porém talvez mais importante, o robô deve salvaguardar sua integridade física e, em uma situação real, não pode, sob hipótese alguma, passar por cima de uma mina com suas rodas. Toda a navegação do robô se baseia na presunção que a área à frente do robô é completamente varrida pelos sensores que buscam as minas, sabendo-se que uma falha na detecção muito provavelmente resultará no toque de uma das rodas na mina não percebida. Portanto, é necessário que os pré-requisitos estabelecidos sejam devidamente atendidos.

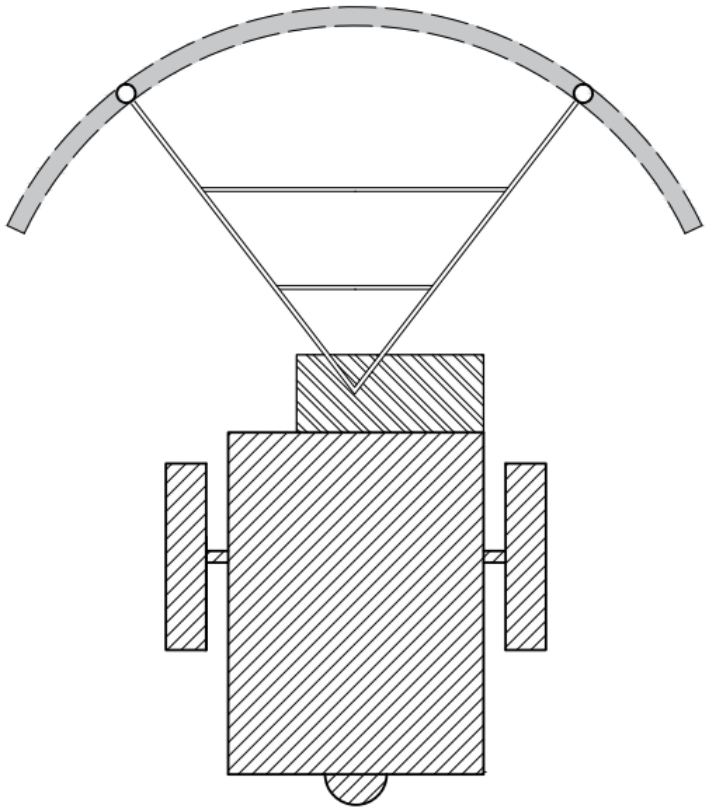
5.1.2 Adaptação do hardware

A primeira adaptação feita no *hardware* do *Boe-Bot* foi na parte sensorial. Dois pares de emissor/receptor de infravermelho foram adicionados. Os pares são montados na matriz de contatos do *shield*, e direcionados para a frente do robô, sendo um levemente virado para a esquerda e outro, para a direita. Dessa forma, é possível que se detecte paredes e obstáculos à frente. O circuito elétrico que envolve esses sensores, que são conectados à entradas e saídas digitais do Arduino, é tal qual descrito na Seção 3.5. Também conectado à uma das saídas digitais está um circuito com um LED, ligado em série com um resistor, cuja finalidade é sinalizar quando uma mina é encontrada. Ou seja, sempre que se encontrar uma mina, o LED de sinalização deve se acender.

[9] A odometria é um método utilizado para estimar a posição de um robô. Para mais informações, ver Apêndice A.

Tendo em mente a estratégia descrita na seção anterior, sugeriu-se a montagem de uma estrutura em forma triangular, com um vértice desta estrutura acoplado a um servomotor tradicional, e com uma estrutura de sensoriamento para detecção de minas montada em cada um dos outros dois vértices. Fixado na parte frontal do robô, o motor adicional tem por finalidade movimentar a estrutura para que se realize um varrimento, com os sensores, da área à frente deste. Não se deve confundir o funcionamento desse servomotor, tradicional, que interpreta sinais de comando como uma posição do eixo, com os servomotores utilizados nas rodas de tração, que tem rotação contínua. O projeto inicial dessa estrutura pode ser conferido na Figura 9.

Figura 9 – Representação da estrutura de sensoriamento proposta para a Tarefa 1



Legenda: O robô, suas rodas e o servomotor adicional são representados pelas áreas hachuradas. A estrutura triangular que suporta os sensores é acoplada ao servo adicional e sua movimentação ocasiona na varredura da área sombreada em cinza.

Fonte: Produção do próprio autor.

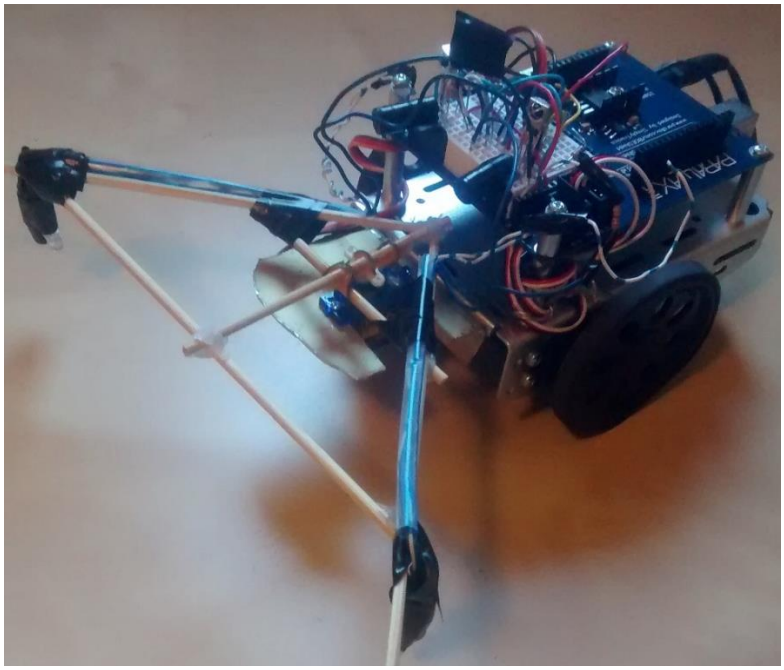
A estrutura de sensoriamento sugerida compreende um par LED/fototransistor direcionados para o chão, pouco acima deste. O fototransistor tem por finalidade diferenciar quando está posicionado sobre o chão branco e quando está sobre uma mina preta através do seu princípio

de funcionamento, já que a intensidade da luz refletida por uma superfície clara difere de uma escura. O LED é utilizado para manter a iluminação da região percebida pelo fototransistor relativamente constante, aliviando as consequências de uma iluminação não-uniforme do ambiente. O par é devidamente encapsulado e posicionado de forma a emitir luz para o chão e recebê-la majoritariamente deste. Os fototransistores são conectados às entradas analógicas do Arduino, que fornecem leituras em valores numéricos na faixa de 0 a 1023, e essas leituras é que serão usadas pelo *software* para se diferenciar minas e chão.

Esta estrutura de *hardware* adicional tem por grande virtude possibilitar o varrimento total da área à frente do robô utilizando poucos elementos sensores. Os dois pares LED/fototransistor representam oito conexões com o *shield* do *Boe-Bot*. Além disso, quanto maiores os lados do triângulo, maior a amplitude da área vasculhada, embora ao custo de uma estrutura mais robusta e pesada.

A estrutura de *hardware* proposta já montada pode ser conferida na Figura 10. Para sustentar o servomotor adicional uma plataforma metálica foi construída e acoplada ao corpo do robô. A estrutura triangular foi montada utilizando palitos de madeira, que são suficientemente leves e resistentes. Os fios que alimentam os sensores são conduzidos dentro de canudos presos à estrutura, para limitar seus movimentos durante a varredura.

Figura 10 – Estrutura de *hardware* proposta para a realização da Tarefa 1

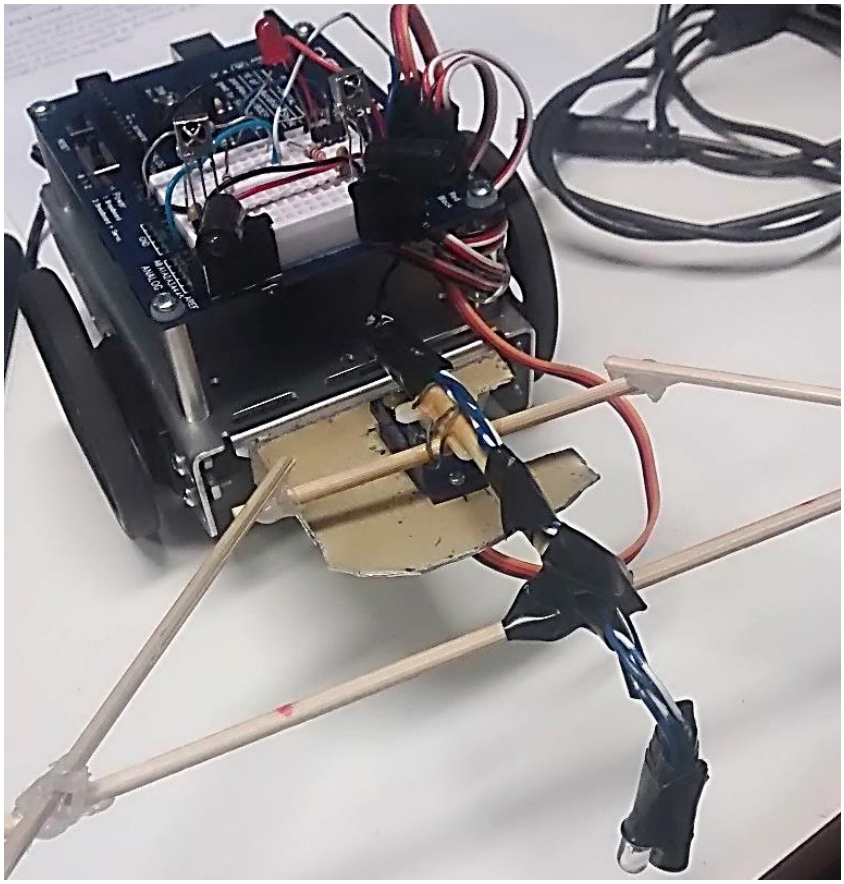


Fonte: Produção do próprio autor.

Durante a fase de testes da estrutura, alguns problemas foram observados. Apesar de realizar a varredura de uma grande área a frente do robô de forma muito rápida, a movimentação da estrutura se mostrou um fator complicador. O acoplamento entre o servomotor e sua engrenagem, que se prende ao resto da estrutura, foi problemático pois, após certo período com a varredura em funcionamento, a força que os fios, rígidos, conectados à matriz de contatos do *shield* exercia sobre este acoplamento acabava por desfazê-lo, liberando a estrutura do servomotor e comprometendo o desempenho global do robô. Não se conseguiu aumentar a resistência deste acoplamento, tornando necessária a modificação da estrutura.

Uma adaptação da estrutura foi então proposta. Mais leve e com menos fios em movimento, esperava-se que o problema de desacoplamento da engrenagem fosse resolvido, como de fato ocorreu. A forma triangular foi mantida para balanço, porém partes da estrutura foram retiradas, deixando-a mais leve e flexível. Um elemento sensor foi abandonado, utilizando-se apenas um par LED/fototransistor para detecção de minas, posicionado na estrutura no centro da aresta oposta ao servomotor, utilizando a própria rigidez dos fios que a conectam ao robô para estender seu alcance e garantir a varredura de toda a área à frente do robô. A estrutura adaptada, finalizada e montada no robô, pode ser conferida na Figura 11. Esta é a configuração final de *hardware*, aquela utilizada para a realização da tarefa.

Figura 11 – Estrutura de *hardware* utilizada para a realização da Tarefa 1



Fonte: Produção do próprio autor.

Agora, apenas metade dos fios que antes conectavam o sensoriamento da estrutura ao *shield* foram mantidos. Posicionados de forma a minimizar a percepção do movimento dos fios no entorno das conexões, garantindo suficiente flexibilidade ao conjunto, a força aplicada sobre o acoplamento diminuiu sensivelmente e a estrutura não mais se soltava após diminuto tempo de varredura. Ao contrário, resistia por mais tempo do que necessário, embora ainda requeresse certo cuidado eventual com sua manutenção.

5.1.3 Software

Dada a estratégia pensada e toda a estrutura de *hardware* determinada, deve-se definir o conjunto de comportamentos necessários a serem implementados no robô para o cumprimento da tarefa. Listam-se os três comportamentos desejados:

- **Buscar Minas**: satisfaz a necessidade do robô de buscar minas e compreende a navegação deste, em conjunto com o devido funcionamento da estrutura de detecção.

- **Detectar Minas**: ativado quando da detecção de uma mina, engloba a sinalização da detecção e o desvio da direção da mina, para que o robô não passe por cima dela.

- **Desviar de Obstáculos**: detectado um obstáculo, o robô não pode colidir com este, devendo, para tal, ajustar sua navegação de forma conveniente.

O *software* deve codificar de forma adequada os comportamentos propostos. O Arduino é programado em linguagem própria, baseada em C++/Java, e é nele que deve ser embarcado o código escrito. Apenas duas bibliotecas do Arduino se fazem necessárias, são elas *stdlib.h* e *Servo.h*, esta última utilizada para controlar os três servomotores utilizados (dois das rodas e um da estrutura de detecção).

O comportamento **Buscar Minas** é implementado tal qual descrito na estratégia proposta. O robô anda em linha quase reta, realizando a varredura da área à sua frente, até que encontre uma mina ou um obstáculo, estímulos estes que ativam os comportamentos **Detectar Minas** e **Desviar de Obstáculos**, respectivamente. Durante a busca, um pequeno grau de aleatoriedade é adicionado ao sinal que comanda os servomotores das rodas, fazendo com que o robô percorra um caminho levemente irregular.

Enquanto busca minas, a varredura da área à frente do robô é constante. O código determina os ângulos mínimo e máximo para o posicionamento do servomotor que sustenta a estrutura e mantém controle do ângulo atual. Quando da detecção de uma mina, o comportamento de detecção é ativado e o LED de sinalização é aceso. Como se tem registro do ângulo de posição da estrutura, sabe-se a direção da mina encontrada, e o robô pode decidir adequadamente em qual sentido deve rotacionar para não passar sobre a mina.

A fim de se evitar falsas detecções de minas, um limiar é selecionado para confirmar a detecção. Durante a inicialização do robô, parte-se do pressuposto de que este está sobre o chão branco e toma-se uma medida do valor lido pelo fototransistor. O programa entende como uma mina qualquer valor fornecido pelo sensor durante a varredura que esteja abaixo de trinta por cento deste valor inicialmente lido.

Os sensores infravermelhos, utilizados para a detecção de obstáculos, informam se há ou não algo à frente. Define-se via *software* a frequência de trabalho dos emissores infravermelhos a fim de se limitar a distância na qual os objetos passam a ser percebidos pelos sensores. Ao mesmo tempo, deve-se cuidar para que o robô não inicie o desvio a uma grande distância dos obstáculos, para evitar que se deixe de encontrar minas que estejam perto destes. Também não deve o robô se aproximar em demasia dos obstáculos, evitando colisões, em especial aquelas que envolvam a parte da estrutura de sensoriamento para detecção de minas.

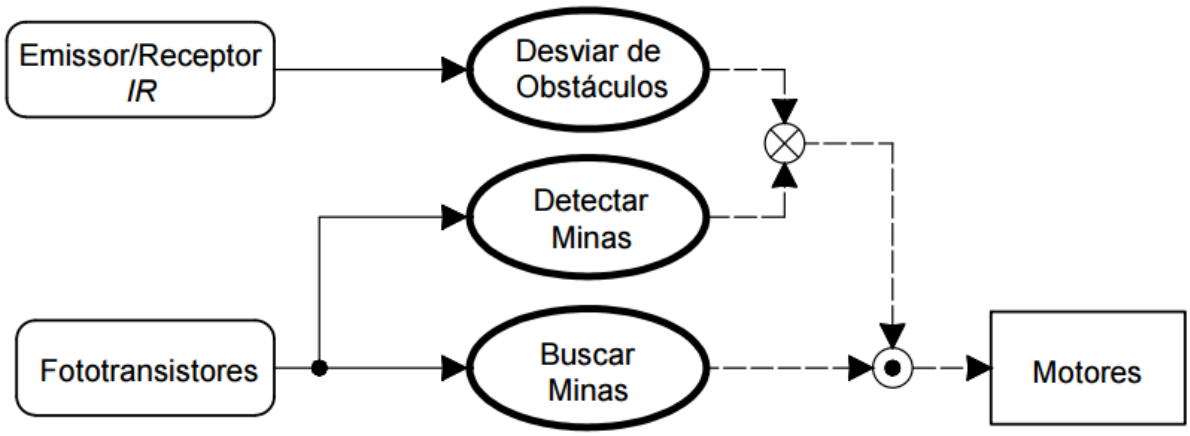
Para deflagrar a detecção de algo à frente, é feito um registrador de detecções para cada um dos sensores. Esse registrador é iniciado em zero e, para cada leitura do sensor que indique uma detecção, incrementa-se um. Da mesma forma, decrementa-se um a cada leitura realizada que não indique nada à frente do robô. Sucessivas leituras indicando algo à frente incrementam o registro até que este seja maior que um limiar estabelecido, o que ativa o comportamento de desviar do obstáculo encontrado. O comportamento permanece ativado até que o registrador seja decrementado – por leituras que não encontrem obstáculos – para abaixo do valor do limiar. O robô opta por girar na direção oposta àquela do registrador de maior valor. Ou seja, se as recentes leituras no sensor esquerdo forem mais frequentes que a do direito, o robô entende que o obstáculo está mais a sua esquerda e há de girar para a direita. Uma certa aleatoriedade é adicionada a essa comparação dos valores dos registros entre si, a fim de se diminuir possíveis vícios nas leituras.

Se um obstáculo for detectado no momento em que o robô encontra uma mina, ambos comportamentos trabalham em cooperação e a posição do obstáculo é levada em consideração a fim de que o robô seja posicionado em direção conveniente. Neste caso, ignora-se os registradores anteriormente citados e leva-se em consideração apenas a leitura atual dos sensores.

A interação entre os comportamentos descritos está representada no diagrama da Figura 12. Neste diagrama, os sensores são representados em retângulos de borda arredondada, os comportamentos, em elipses e os atuadores, em retângulos. As linhas sólidas levam as leituras dos sensores aos comportamentos e as linhas tracejadas, o comando dos comportamentos aos atuadores. O símbolo $\otimes$ indica seleção de comportamentos, realizada pelo controlador, e admite alguma cooperação quando os comportamentos estiverem ativos ao mesmo tempo,

enquanto que o símbolo ⊙ indica supressão do sinal proveniente do comportamento lateral pelo comportamento superior.

Figura 12 – Diagrama dos comportamentos utilizados na Tarefa 1



Fonte: Produção do próprio autor.

O código final, tal qual implementado no robô, pode ser conferido no Apêndice B.

5.1.4 Análise dos resultados e discussões gerais

Com relação ao objetivo da tarefa, resultados semelhantes foram obtidos em cada uma das três rodadas de execução pelo robô. Apesar da proximidade dos resultados, diferentes particularidades puderam ser observadas em cada execução. As rodadas foram realizadas intercaladas com intervalos de aproximadamente meia hora, no qual pequenas alterações e correções foram realizadas no código.

A primeira rodada durou aproximadamente um minuto e trinta segundos. O robô detectou corretamente quatro minas e fez sete desvios de obstáculos, incluindo dois desvios de quinas (onde as paredes da arena, que é retangular, se encontram). Não andou por sobre nenhuma mina e não colidiu com as paredes. Por vezes passou bem perto de algumas minas, não as encontrando ora por falta de alcance da estrutura que carrega o sensor, ora pela irregularidade de sua navegação. A rodada enfim terminou quando o robô ficou encurralado em uma quina, sem conseguir reagir efetivamente para sair daquela situação e retornar à arena.

A segunda rodada foi similar à primeira em vários aspectos. O robô detectou novamente quatro minas, sem tocá-las e sem falsas detecções, e desviou adequadamente das paredes, sem nunca colidir. Desta vez, a rodada teve fim quando o robô se viu preso entre uma mina e uma parede, que estavam próximas. O robô detectava a mina e partia em direção à parede, e, então, ao desviar da parede, detectava novamente a mina. Esse comportamento se repetiu até o esgotamento do tempo de execução da tarefa.

O robô logrou maior êxito na terceira rodada. Encontrou cinco minas e, além de não colidir com as paredes, não se encontrou preso em nenhuma das situações anteriores. Houve, porém, pela primeira vez uma falsa detecção de mina.

Pouco antes da execução da primeira rodada, durante manutenção rotineira na estrutura de sensoriamento, o LED do par LED/fototransistor parou de funcionar. Não havendo tempo hábil para se realizar a troca, as três rodadas foram executadas sem a utilização deste componente, que objetivava aumentar a regularidade da iluminação incidente sobre o chão. A falsa detecção de mina ocorreu na rodada na qual o robô fora inicializado em uma região mais clara da arena, o que eleva o valor absoluto do limiar estabelecido para ativação do comportamento Detectar Minas e facilita a ocorrência de falsas detecções em regiões mais escuras. A definição desse limiar parte de um valor determinado na leitura inicial do fototransistor e permanece constante durante toda a execução da tarefa, até que se inicialize novamente o robô. O limiar estabelecido, estático, fora assim definido como consequência da utilização do par LED/fototransistor. A fim de se contornar o problema do limiar estático, pode-se ter o valor absoluto do limiar estabelecido através de um percentual sobre a média móvel dos valores lidos pelo fototransistor – e interpretados como chão branco – durante a execução do robô. Dessa forma, uma possível distorção causada por uma inicialização em uma região mais clara da arena há de ser corrigida durante o tempo que o robô passar por regiões mais escuras, diminuindo a probabilidade de que uma sombra seja falsamente interpretada como uma mina. Esta abordagem não foi utilizada e é posta aqui apenas como algo que pode ser testado, sem que possíveis problemas advindos desta ideia tenham sido previamente avaliados.

As duas primeiras rodadas tiveram fim com o robô preso a ações repetitivas, sem conseguir escapar das situações em que se encontrava. Na primeira rodada, ao encontrar a quina da

arena à sua frente – e bem próxima – o robô permaneceu aparentemente estático, provavelmente oscilando, dentro do comportamento responsável pelo desvio de obstáculos, entre as ações de virar para um lado e para o outro. Dada a forma como o comportamento define para qual lado o robô deve rotacionar, se utilizando de uma série de leituras dos sensores IR, e dada a ausência de uma rotina de escape, talvez uma que fizesse o robô girar 180º ao se encontrar diante de tal situação, o comportamento emergente observado nesta situação foi ineficiente. Na segunda rodada, o robô se viu preso entre uma parede e uma mina, se desviando da parede em direção à mina e, então, se desviando da mina em direção à parede. Os graus de aleatoriedade inseridos nos ângulos que o robô gira ao manobrar e ao navegar fazem crer que eventualmente aquela situação seria superada. Porém, dada a limitação de tempo, o comportamento emergente foi novamente insuficiente. Isto poderia ser contornado com a inserção de uma rotina que percebesse o movimento repetitivo do robô, que realizava vários desvios em um curto espaço de tempo, e modificasse radicalmente os ângulos de manobra, a fim de tentar livrá-lo daquela situação.

Como diferentes posições – e orientações – iniciais foram escolhidas a cada rodada, a regularidade observada no número de minas encontradas indica a consistência da estratégia adotada. Ademais, o comportamento emergente observado no robô sempre atuou no sentido de salvaguardar sua integridade física, evitando colisões e atropelamentos de minas de forma eficiente. Também se entende que a velocidade de navegação, durante a busca, e velocidade de manobra, tanto no desvio de obstáculos quanto no de minas, foram suficientes para atender ao requisito de rapidez na atuação, a fim de aumentar a área vasculhada pelo robô e a consequente probabilidade de se encontrar minas. Dessa forma, julga-se que os preceitos que fundamentam o projeto foram atendidos.

5.2 Tarefa 2

A Tarefa 2 é uma representação do problema de forrageamento e foi descrita na Seção 4.2, onde constam todas as especificações pertinentes. Essa tarefa consiste, basicamente, na busca por cilindros de metal, seu recolhimento e condução à uma área demarcada.

5.2.1 Estratégia proposta

A limitação de tempo para a condução da tarefa, somada ao objetivo de se conduzir o maior número de cilindros para a *safe zone*, fazem da velocidade de busca, localização e condução dos objetos um requisito essencial. Deve o robô, portanto, buscar os objetos pela arena da forma mais rápida e eficiente possível.

Novamente, a inexistência de odometria não permite que o robô saiba onde está ou por onde passou. O relativo grande número de cilindros espalhados homogeneamente pela arena é um fator auxiliar para a navegação: percebendo os objetos a uma distância adequada, o robô pode ser preciso em sua direção de navegação, seguindo na direção do objeto mais próximo.

A velocidade de navegação do robô deve ser maximizada, sem que se comprometa a detecção e a condução dos cilindros. Por serem ocos, estes não possuem grande massa e não interferem significativamente na dinâmica do robô, ou seja, a adição de inércia ao sistema é considerada nula quando do recolhimento de um cilindro, permitindo que o robô conduza simultaneamente tantos quantos sua estrutura física possibilitar.

Após o recolhimento de um primeiro cilindro, considera-se que pode haver um segundo (ou terceiro, ou quarto, etc.) entre o robô e a *safe zone*, ou mesmo dentro de um entorno razoável. Dessa forma, é conveniente que o robô não prossiga cegamente a depositar o cilindro apanhado, mas que se permita continuar a buscar o ambiente por novos cilindros.

Pela existência de pontos cegos e possível dificuldade em se localizar a *safe zone*, é interessante que o robô possa carregar elevado número de cilindros em sua estrutura, maximizando seu potencial de entrega dentro do tempo limitado.

Dessa forma, enunciam-se os três preceitos estratégicos seguidos: velocidade de navegação, eficiência na busca, e maximização do número de cilindros a serem apanhados.

5.2.2 Adaptação do *hardware*

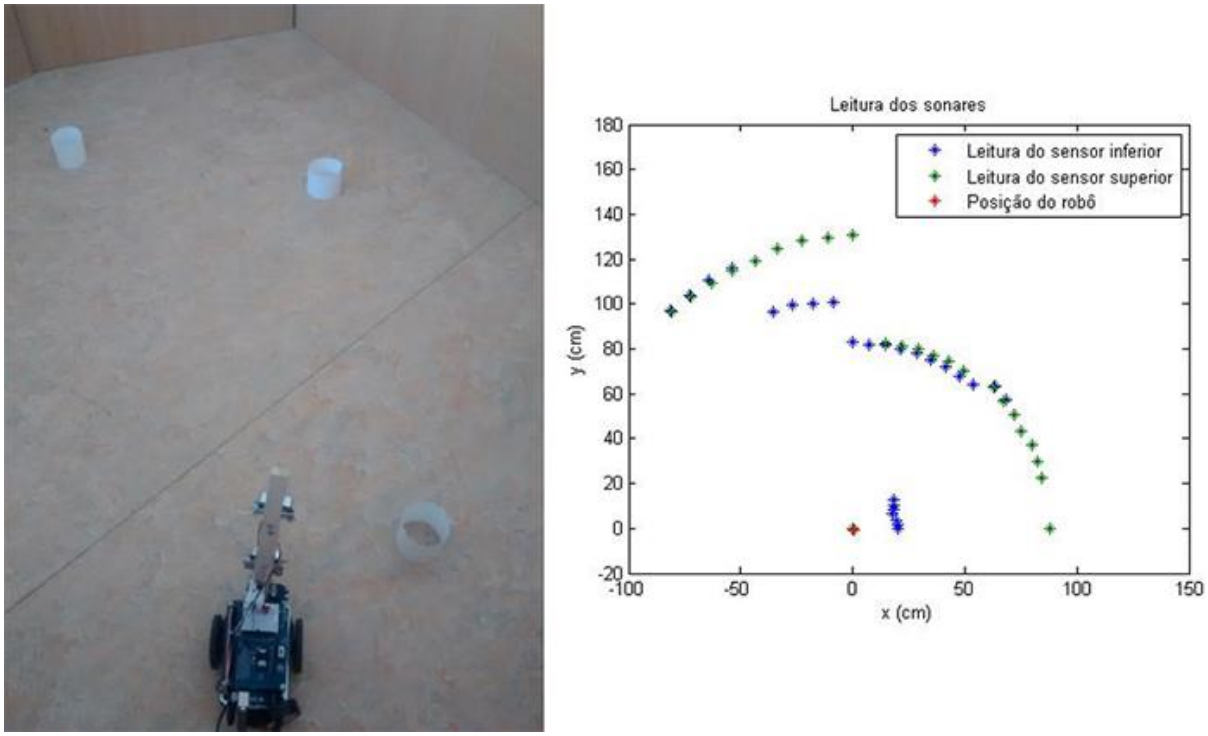
A estrutura responsável pela detecção dos cilindros buscados é acréscimo fundamental ao *hardware* do robô. Para a realização da Tarefa 2, além dos sensores anteriormente disponibilizados e utilizados na Tarefa 1, também há a opção de se utilizar sensores ultrassom. Há a preferência pela utilização destes, pois podem ser usados mais facilmente tanto na detecção de cilindros quanto na detecção de obstáculos.

A estrutura de sensoriamento proposta consiste em dois sensores ultrassom mirando na direção de movimento do robô. Pertencentes ao mesmo plano, perpendicular ao plano do chão, os sensores são alocados em diferentes níveis de altura. Quando em frente a um cilindro, o sensor mais baixo, posicionado perto do chão, produzirá medidas de distância menores do que as fornecidas pelo mais alto, posicionado em uma altura que excede a dos cilindros. Em situação distinta, fornecerão medidas similares, seja por haver uma parede ou obstáculo à frente ou pela inexistência destes.

O servomotor adicional, utilizado na Tarefa 1, é mantido e tem a estrutura que contém os sensores conectada a seu eixo. Esta configuração permite que se faça uma varredura mais completa daquilo que está à frente do robô, ou seja, dentro de um *range* de ângulos mínimo e máximo, estabelece-se um passo para o motor e, a cada passo, realiza-se a leitura dos sensores ultrassom.

Posteriormente, durante a fase de testes, para melhor entendimento do funcionamento dessa estrutura e das leituras advindas dos sensores, um código em Matlab foi feito para receber do robô os valores de distância lidos pelos sonares e traçar em um gráfico correspondente, na tela do *notebook*, em tempo real, aquilo que o robô estaria percebendo. A Figura 13 traz, do lado esquerdo, uma foto de uma situação de teste, com três cilindros e paredes à frente do robô, e do lado direito os dados lidos já plotados.

Figura 13 – Varredura com os sonares: situação real e representação no Matlab

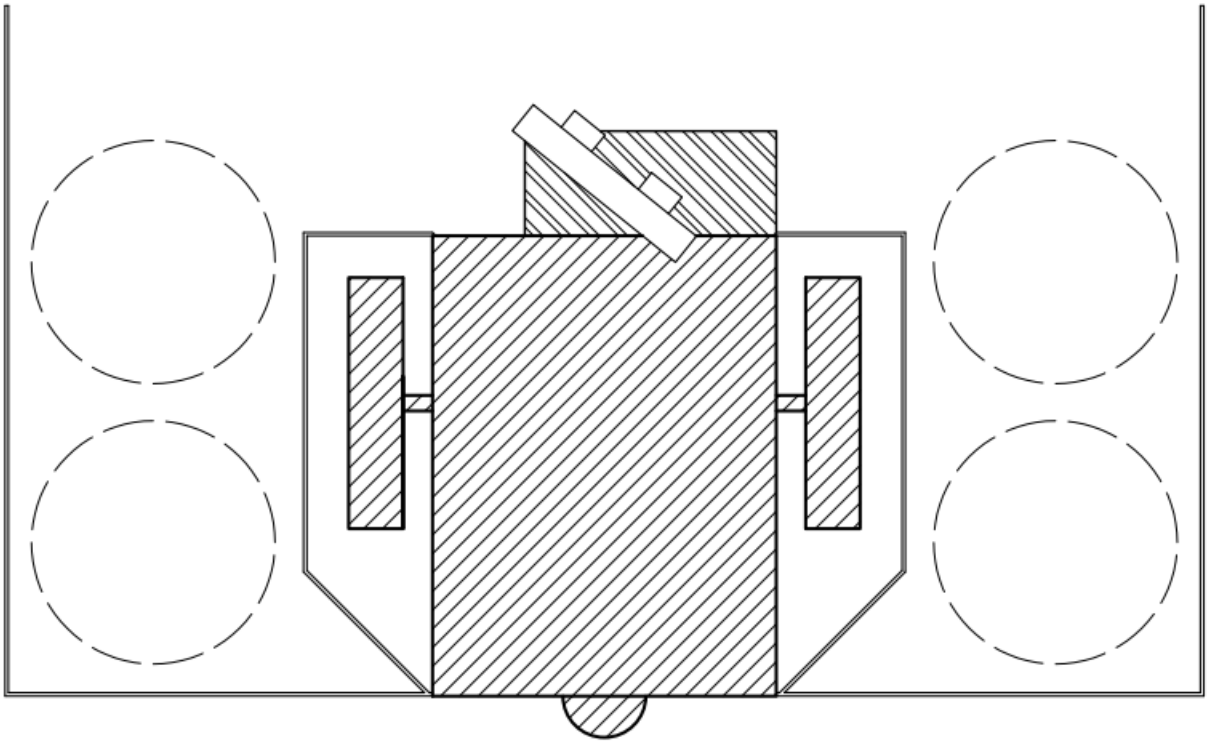


Fonte: Produção do próprio autor.

Percebe-se da Figura 13 que o robô encontrou corretamente dois dos três cilindros, havendo algumas distorções na detecção do segundo. Testes posteriores obtiveram melhores resultados depois de ajustes na velocidade de varredura e na quantidade de leituras realizadas. Embora satisfatória para detecção de cilindros e obstáculos, a utilização de sonares deu origem a algumas dificuldades, que serão discutidas em detalhes posteriormente.

Além de encontrar e se mover em direção aos cilindros, o robô deve ser capaz de recolhê-los. Para tal, é proposta uma estrutura a ser adicionada ao robô. Essa estrutura consiste em dois "braços" que saem da parte de trás do robô, acompanham suas laterais e se estendem o suficiente para que cada um deles possa suportar confortavelmente dois cilindros. As rodas do robô são protegidas pela estrutura para que os cilindros não as toquem. O projeto dessa estrutura está representado na Figura 14.

Figura 14 – Estrutura de recolhimento de cilindros proposta para a Tarefa 2



Legenda: O robô, suas rodas e o servomotor adicional são representados pelas áreas hachuradas. A estrutura de recolhimento dos cilindros se estende pelas laterais do robô, o suficiente para englobar os cilindros. O sonar está posicionado sobre o servo de forma ilustrativa.

Fonte: Produção do próprio autor.

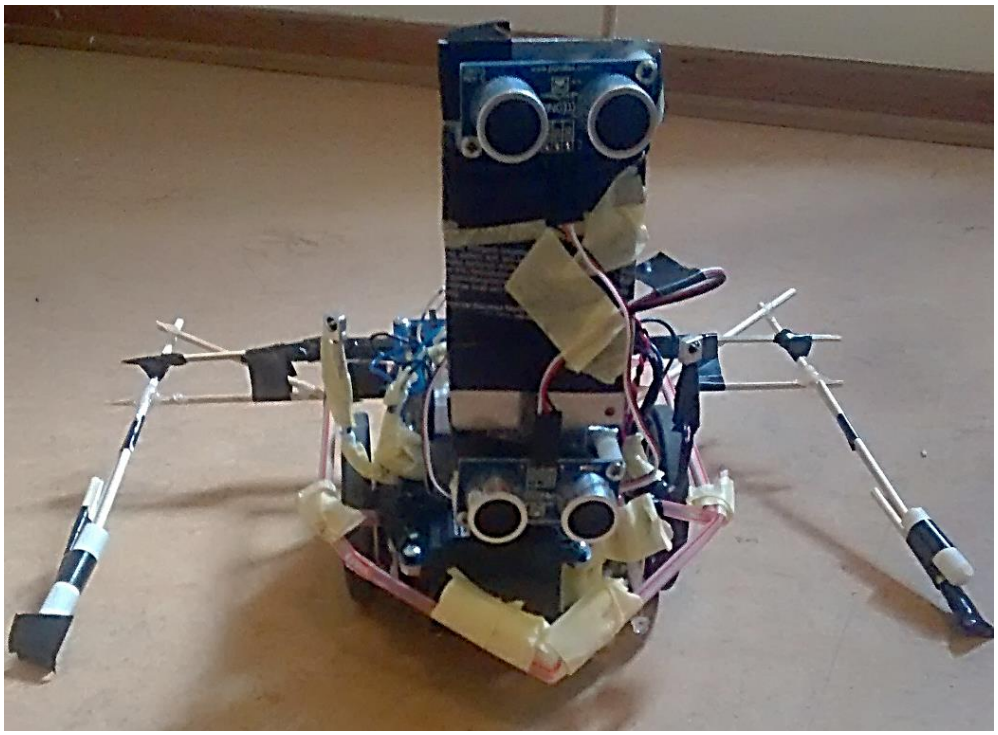
Uma posterior modificação da estrutura de recolhimento dos cilindros foi proposta quando da integração desta estrutura em conjunto da estrutura de sensoriamento anteriormente comentada e mostrou-se prudente a adição de uma nova parte, posta à frente do servomotor frontal, para direcionar lateralmente os cilindros conforme o robô avança sobre eles e evitar que estes pudessem ficar presos ao sensor ultrassom mais baixo. É uma simples estrutura em formato de "V", leve e feita de canudo de plástico. Esta modificação pode ser conferida, já inserida no robô, na Figura 15.

Por fim, além de buscar e recolher os cilindros, o robô deve ser capaz de conduzí-los e depositá-los na *safe zone*. Para tal, fazer uso do *beacon* infravermelho é a solução mais eficaz e os mesmos receptores infravermelhos anteriormente utilizados podem ser aproveitados. Dois desses receptores são colocados na frente do robô, levemente direcionados para lados opostos, e são responsáveis por detectar o sinal emitido pelo *beacon*. Um par LED/fototransistor é posicionado ao lado do servomotor adicional e é utilizado para detectar o revestimento escuro da *safe zone*, para que o robô saiba que esta finalmente foi atingida e

possa retroceder, deixando os cilindros dentro área adequada. Novamente utiliza-se um LED na tentativa de diminuir a influência de sombras e irregularidades na luminosidade do ambiente.

A anteriormente citada Figura 15 traz a configuração final da estrutura de *hardware* utilizada na execução da Tarefa 2.

Figura 15 – Estrutura de *hardware* utilizada na Tarefa 2



Fonte: Produção do próprio autor.

5.2.3 Software

Adotada a estratégia e determinada a estrutura de *hardware*, deve-se definir o conjunto de comportamentos necessários a serem implementados no robô para o cumprimento da tarefa. Listam-se seis comportamentos:

- **Desviar de Obstáculos**: comportamento responsável por evitar colisões. Faz uso da varredura dos sonares para inferir quando um obstáculo está por demasiado perto para que se efetue o devido desvio na trajetória do robô.

- **Buscar por Cilindros**: satisfaz a necessidade que o robô tem de encontrar cilindros. Utilizando-se da varredura dos sonares para detectar os cilindros, a busca consiste em seguir em uma direção adequada até que se encontre um (ou, eventualmente, mais de um) cilindro.

- **Recolher Cilindro**: uma vez encontrado o cilindro, este deve ser recolhido. Para tal, o robô se posiciona em direção ao cilindro desejado e segue até que o apanhe, fazendo ocasionais usos da varredura dos sonares para inferir a distância até os cilindros e a correta direção a ser seguida.

- **Buscar pela *Safe Zone***: comportamento responsável por buscar a *safe zone*, faz uso das leituras dos sensores infravermelhos para encontrar e direcionar o robô para ela.

- **Navegar para a *Safe Zone***: uma vez ativado, faz com que o robô se ajuste de acordo com as leituras dos sensores infravermelhos e navegue em direção à *safe zone*.

- **Depositar Cilindros**: comportamento ativado quando a *safe zone* é enfim atingida, o robô retrocede para depositar os cilindros dentro dela.

Os comportamentos foram codificados no *software* embarcado no Arduino através de uma máquina de estados. Da forma como o programa foi escrito, um mesmo comportamento pode se manifestar em diferentes estados da FSM e – partes de – diferentes comportamentos podem estar contidos em um mesmo estado. Esta forma de representação dos comportamentos se faz conveniente do ponto de vista da simplicidade, tanto de escrita quanto de entendimento do código. Embora possa parecer que diferentes comportamentos se confundem, em verdade, eles permanecem independentes.

Inicializado, o robô realiza uma varredura com os sonares para definir a melhor direção a seguir. O comportamento de **Desviar de Obstáculos** é ativado se houver algum obstáculo próximo à frente do robô e o comportamento de **Buscar por Cilindros** verifica se o sensoriamento encontrou algum cilindro. Caso nenhum cilindro seja encontrado, o comportamento de busca escolhe a melhor direção para seguir e avança por certo tempo nesta direção. Esta direção é tida como a direção em que a leitura da varredura dos sonares foi

maior, ou seja, na direção em que não há obstáculos, ou naquela em que estes estão mais afastados, e o tempo de avanço é tão maior quanto maior for a distância observada. Caso um ou mais cilindros sejam encontrados, o comportamento de **Recolher Cilindro** é ativado. Este identifica qual o cilindro que se encontra mais próximo, caso haja mais de um, e se move em direção a ele por certo tempo. A varredura é novamente feita e o robô prossegue em direção ao cilindro pelo mesmo período de tempo. Esse processo se repete até que o cilindro buscado desapareça da varredura, quando se assume que o cilindro foi devidamente recolhido. As varreduras com os sonares são realizadas constantemente, mas não ininterruptamente. O robô deve estar parado para que se faça a varredura, e esta é realizada após cada comando de navegação por parte dos comportamentos. Ou seja, durante a busca, por exemplo, o robô navega por um tempo, cessa seu movimento, e novamente varre o ambiente. Dessa varredura pode advir a ativação, ou não, de qualquer um destes três comportamentos.

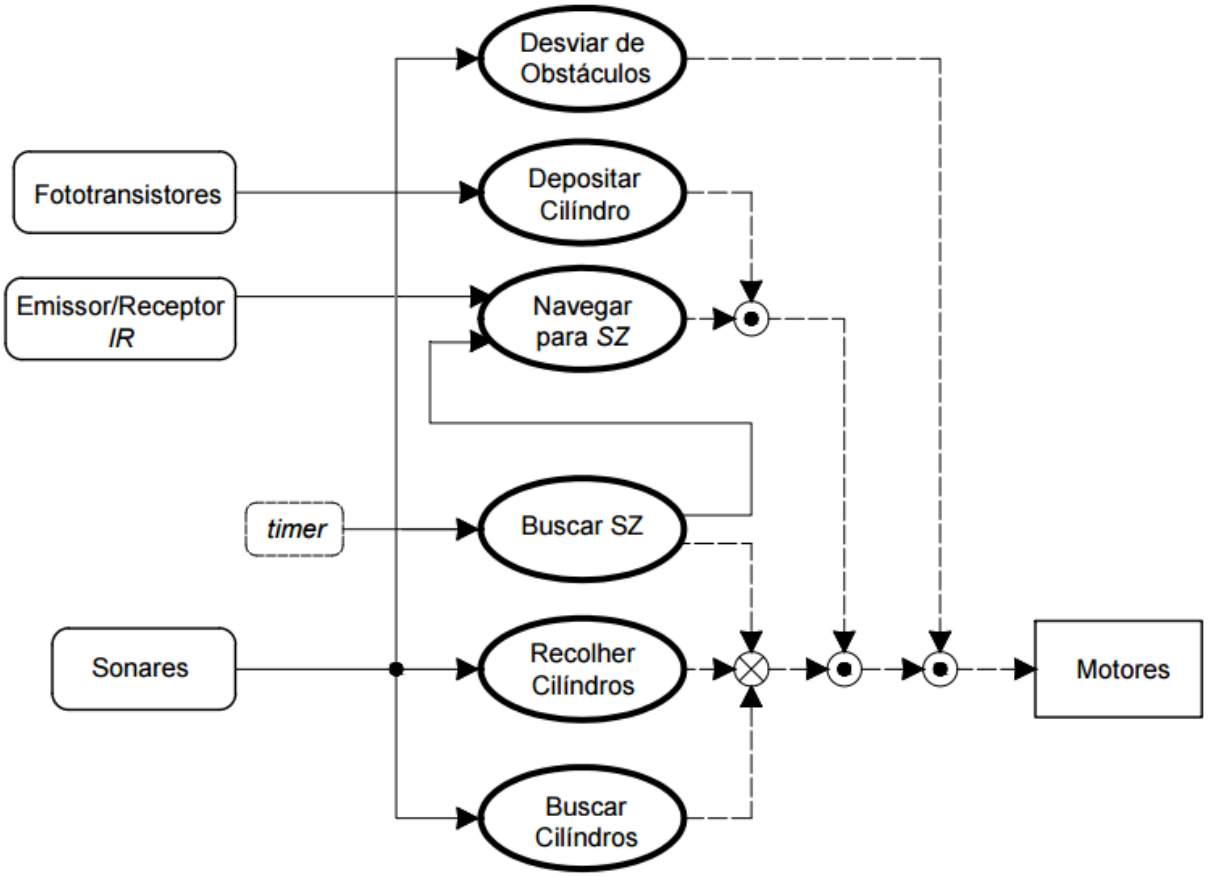
É escolhido um tempo limite para que o robô busque e recolha cilindros e, excedido este tempo, o comportamento de **Buscar pela *Safe Zone*** é ativado. Há ao menos duas vantagens nessa abordagem: caso o robô fique preso em alguma região vazia da arena, ou não consiga encontrar nada por muito tempo, ele deve buscar a *safe zone* e dela partir como ponto inicial da nova busca; também não há a necessidade de se manter registro de quantos cilindros foram recolhidos, o que evita erros provenientes de falsos recolhimentos. Pressupõe-se que o tempo estabelecido é suficiente para que o robô encontre e recolha cilindros, mas não grande o bastante para que o número de cilindros recolhidos exceda a capacidade de carregamento da estrutura. Deve-se salientar que as características da arena e de quantidade e distribuição dos cilindros favorecem essa abordagem, podendo esta ser desaconselhável em situações diferentes. Enquanto busca pela *safe zone*, o robô não ignora cilindros detectados, ou seja, os comportamentos de busca e recolhimento de cilindros podem ser ativados em paralelo.

Em **Buscar pela *Safe Zone***, o robô vaga pela arena até que se receba o sinal proveniente do *beacon*. Quando o comportamento de **Navegar para a *Safe Zone*** é ativado, o robô passa a seguir em direção ao *beacon*, ajustando sempre sua trajetória para tentar atingir a *safe zone*. Deve-se notar que, além do recebimento do sinal do *beacon*, para se ativar, o comportamento **Navegar para a *Safe Zone*** necessita da ativação anterior do comportamento **Buscar pela *Safe Zone***. Ou seja, para que o robô possa seguir em direção ao *beacon*, é necessário que o *timer* tenha, antes, ativado o comportamento de busca pela *safe zone*. Antes disso, os sinais IR

recebidos pelos sensores são ignorados. Uma vez atingida a *safe zone*, o fototransistor denuncia a mudança na cor do piso e o comportamento de **Depositar Cilindros** é ativado. Este comportamento faz com que o robô avance um pouco sobre a *safe zone*, a fim de garantir que os cilindros estejam dentro da área demarcada, devendo o robô, então, parar e retroceder, deixando ali os cilindros. O robô então se vira para a arena e segue sua busca por outros cilindros.

O diagrama da Figura 16 ilustra a interação entre os comportamentos. O diagrama é tal qual o utilizado na Tarefa 1, onde os sinais do sensoriamento ativam os comportamentos, que comandam os atuadores. A coordenação dos comportamentos é similar, admitindo seleção e supressão. Uma das principais diferenças é que, neste diagrama, um dos comportamentos, **Buscar pela *Safe Zone***, não é ativado por um sensor, mas por um estado interno do robô, seu *timer*, representado em um retângulo tracejado de bordas arredondadas. Além disso, o comportamento **Navegar para a *Safe Zone*** é ativado por um sensor e por outro comportamento, **Buscar pela *Safe Zone***.

Figura 16 – Diagrama dos comportamentos utilizados na Tarefa 2



Fonte: Produção do próprio autor.

O código final, tal qual implementado no robô na última rodada de execução, pode ser conferido no Apêndice C.

5.2.4 Análise dos resultados e discussões gerais

Diferentemente do ocorrido na execução da Tarefa 1, os resultados obtidos na execução da Tarefa 2 variaram significativamente entre as rodadas. Estas foram realizadas intercaladas com intervalos de aproximadamente meia hora, no qual pequenas alterações e correções foram realizadas no código.

Na primeira rodada, o robô não conseguiu deixar nenhum cilindro na *safe zone*. Ademais, também não conseguiu realizar nenhum recolhimento. Sendo inicializado ao fim de um corredor, de onde as paredes se abriam para uma parte mais ampla da arena, o robô não conseguiu sair dessa região durante toda a duração da rodada, navegando erroneamente em direção às paredes, colidindo ocasionalmente sua estrutura de recolhimento de cilindros.

Melhores resultados foram obtidos na segunda rodada. O robô apanhou três cilindros e entregou-os à *safe zone*. Após o recolhimento dos dois primeiros, uma falsa detecção de *safe zone* ocorreu e o robô, pelo comportamento de Depositar Cilindros, abandonou os cilindros apanhados, que viriam a ser novamente recolhidos em seguida. A rodada não foi realizada por completo, sendo abortada quando o robô, após depositar corretamente os cilindros, realizou uma volta completa em torno de si e avançou novamente sobre a *safe zone*, pondo em risco a integridade do *beacon*.

Na terceira rodada houve maior número de cilindros recolhidos. Ao todo, o robô apanhou cinco cilindros, porém não depositou nenhum deles na *safe zone*. Perto do fim da rodada, o robô parecia se direcionar corretamente em relação ao *beacon*, porém a ponta de um dos lados da estrutura de recolhimento colidiu com a quina de um obstáculo, deslocando o robô. De face para o obstáculo, uma falsa detecção de *safe zone* ocorreu e o robô abandonou os cilindros recolhidos. Voltava-se para apanhá-los novamente quando o tempo limite esgotou e a rodada foi encerrada.

A leitura dos sonares é problemática com quinas. Por dependerem da reflexão da onda, caso esta atinja a superfície em um ângulo agudo, a onda pode ser refletida para longe do sensor, podendo, inclusive, vir a refletir em um segundo objeto e atingir o sonar, causando uma leitura errônea. O robô, tal qual foi programado, utiliza como distância a média de três leituras dos sensores, média esta que pode ser distorcida por eventuais leituras muito altas ou baixas. A onda sonora emitida não tem o formato de um feixe, mas de um cone, o que também pode causar problemas na resolução das leituras. Caso o objeto seja pequeno, como os cilindros buscados, e esteja a uma distância grande, há o risco de que não se reflita uma porção significativa da onda. Citando Murphy (2000, p. 215, tradução do autor), "quanto mais longe do robô, maior o objeto tem de ser". Esta última característica reflete diretamente aquilo que fora observado durante a fase de testes, onde o robô tinha dificuldade de identificar corretamente cilindros mais afastados. Outra consequência do formato cônico da onda é a possibilidade de que o sonar superior também detecte o cilindro, o que levaria o robô a interpretar o cilindro como parede.

Os problemas observados na primeira rodada, com o robô incapaz de sair do corredor, foram provavelmente causados pelo fato de o corredor ser estreito e com várias quinas entre as placas de madeira que formavam as paredes. Outros robôs, testados por outros estudantes, também tiveram problemas naquela parte da arena, sendo esta posteriormente fechada para a realização das outras rodadas.

O obstáculo com o qual o robô colidiu durante a realização da terceira rodada, além de pequeno em termos de dimensões laterais, por ter a forma de um cubo, apresentava também quinas entre as placas de madeira, que estavam mal ajustadas. A ponta da estrutura apenas tocou a ponta do obstáculo, toque suficiente para desviar a direção de navegação do robô e direcioná-la para o próprio obstáculo, e acredita-se que, justamente pela angulação das peças constituintes do obstáculo, o robô foi incapaz de perceber o quão perto estava deste.

Colisões com as paredes foram observadas na primeira e terceira rodada. Na primeira rodada, em especial, o comportamento emergente do robô foi confuso e tinha-se a impressão que este navegava em direção às paredes propositalmente. Acredita-se que o robô, imerso em um corredor estreito, colidiu com as paredes por navegar na direção escolhida durante tempo demais. Embora este tempo fosse o tempo mínimo de navegação em determinada direção,

antes de uma nova varredura dos sonares, este permitiu aproximação demasiada do robô com extremos da arena. A decisão de fazer com que o robô navegasse por um tempo pré-fixado durante a escolha da direção a ser seguida se mostrou adequada apenas na parte mais ampla da arena, para qual fora pensada.

A colisão com a parede observada durante a terceira rodada, diferentemente da colisão com o obstáculo, foi causada devido à proximidade de um dos cilindros com aquela, porém em uma situação especial. Por vezes, no ato de recolher o cilindro, este está parcialmente inserido na estrutura do robô, porém, por não estar suficiente dentro desta, os sonares ainda conseguem enxergá-lo, fazendo com que o robô se vire e siga na direção deste cilindro para efetuar o recolhimento total. O mesmo acontece quando um cilindro escapa da estrutura (geralmente por uma combinação entre recolhimento parcial e manobra brusca) ou quando se apanha mais cilindros do que se pode comportar (então um dos cilindros permanece parcialmente recolhido, sem nunca adentrar a estrutura por completo). Uma combinação dessas duas situações foi observada durante a terceira rodada, com o robô completamente carregado de cilindros. Acredita-se que, quando o robô cessou seu movimento para realizar a varredura com os sonares, um dos cilindros se deslocou por alguns centímetros, ficando ao alcance dos sensores. Por estar perto da parede, o robô veio a colidir com esta, conseguindo, em seguida, desviar-se e continuar a conduzir todos os cilindros apanhados.

Após o fim da primeira rodada, dado o desempenho insatisfatório observado, o robô foi submetido a uma rápida bateria de testes para confirmar o correto funcionamento de seu *hardware* como um todo. Após essa confirmação, pequenas alterações foram realizadas no código, a fim de aumentar a rejeição do robô às paredes da arena. Acredita-se, porém, que durante a realização da segunda rodada, aquela em que o robô conseguiu apanhar e entregar cilindros, a estrutura de sensoriamento responsável por detectar o sinal IR do *beacon* da *safe zone* parou de funcionar. Essa falha viria a ser confirmada apenas nos testes que antecederam a última rodada.

O *beacon* IR ficava posicionado sobre uma estrutura de formato cilíndrico, de altura semelhante àquela dos cilindros de metal a serem recolhidos. Durante a segunda rodada, o robô apanhou cilindros que se localizavam nas redondezas da *safe zone* e, provavelmente, encontrou o cilindro que suportava o *beacon* e tentou recolhê-lo. Ao entrar na *safe zone*, o

comportamento de Depositar Cilindros foi ativado e o robô recuou. Porém, por erro de código, o robô se permitiu rotacionar 360º, voltando-se novamente em direção ao centro da *safe zone* e partindo, novamente, para "apanhar" o *beacon*. A fim de se proteger a integridade do *beacon*, a rodada foi abortada, com o robô impedido de terminar sua ação ao ser capturado por mão humanas, que o retiraram da arena.

Durante o tempo disponível entre a segunda e terceira rodada não foi possível corrigir o problema dos sensores IR. O robô iniciou a rodada final com essa estrutura comprometida, confiando na possibilidade de, novamente, encontrar o cilindro que suportava o *beacon* e entregar corretamente os cilindros na *safe zone*. Dessa vez, o erro de código que permitiu o robô realizar uma volta completa sobre seu eixo fora corrigido e esperava-se que o robô pudesse performar a entrega de forma satisfatória. Durante a rodada, o robô não vislumbrou o cilindro do centro da *safe zone* em momento algum, partindo para o recolhimento de cilindros que se encontravam em áreas longínquas.

Apesar da inconsistência dos resultados e dos problemas observados, julga-se que os preceitos do projeto foram atendidos. O robô mostrou-se capaz de encontrar e recolher grande número de cilindros, de evitar colisões com as paredes em uma arena ampla, e de entregar os cilindros à *safe zone* (quando a encontrou). O comportamento emergente observado nas duas últimas rodadas era tal qual esperado, agindo de forma eficiente no recolhimento, embora, por falha em componentes de *hardware*, fosse incapaz de localizar a *safe zone* da maneira na qual fora projetado.

6 CONCLUSÕES E TRABALHOS FUTUROS

A reunião e aplicação conjunta de diferentes conhecimentos adquiridos durante a graduação é o objetivo básico de um projeto de graduação. A multidisciplinaridade inerente à robótica garantiu, neste trabalho, a integração de várias áreas da engenharia elétrica. Assim, foram aprimorados e utilizados em conjunção conhecimentos acerca da dinâmica do sistema físico em estudo, do seu sistema de controle, do sistema de programação embarcado e dos sistemas de acionamento e sensoriamento associados. Dessa forma, julga-se que o projeto realizado atende seu objetivo precípuo.

Fruto de considerável estudo da literatura, o Capítulo 2 sintetizou os principais conhecimentos teóricos adquiridos acerca do problema da navegação autônoma na robótica, tendo por base os preceitos da robótica baseada em comportamentos. Estes conhecimentos fundamentam a realização da parte experimental do projeto e consolidam este trabalho como um todo.

Os estudos de caso realizados são pertinentes e condizem com problemas enfrentados por pesquisadores da área de robótica. Ademais, os dois problemas tipo estudados dão sentido e embasam as tarefas propostas. Pode-se, também, observar um encadeamento entre as tarefas, sendo que a primeira exigiu a formulação de ideias e comportamentos que, aproveitados, serviram de base para o desenvolvimento da tarefa seguinte.

Nas fases iniciais de preparação para a realização da Tarefa 1, em especial durante o planejamento, a aplicação na prática da teoria estudada era completa novidade para o autor deste trabalho. O projeto foi pensado objetivando obter resultados tão bons quanto possíveis com os recursos disponíveis. Embora fosse proposto um mecanismo relativamente sofisticado para realizar o varrimento da arena, houve uma preocupação em manter o *software* simples. A busca por velocidade e eficiência norteou o projeto e, dados os resultados alcançados durante a execução da tarefa, julga-se que estas metas foram atingidas. Entende-se que as adversidades observadas poderiam ter sido superadas por um código mais minucioso, que se utilizasse de rotinas de escape que não foram inicialmente pensadas. A consistência dos resultados observados nas diversas execuções, indica claramente a boa qualidade do projeto.

A Tarefa 2 se mostrou mais complexa do que a primeira e exigiu um sensoriamento mais sofisticado. O projeto propôs a varredura do ambiente à frente do robô através de um conjunto de sonares direcionados por um servomotor, ideia esta que se mostrou, por vezes, problemática em alguns aspectos. Observaram-se problemas inerentes à própria utilização do sensor ultrassônico, que resultaram em ocasionais interpretações incorretas do ambiente no qual o robô estava inserido. Ainda assim, a eficiência na localização e recolhimento de cilindros se mostrou razoável. Este fator, em conjunto com a velocidade de navegação apresentada e o número de cilindros que o robô conseguiu comportar, validam a estratégia adotada. Lamenta-se a quantidade de problemas observados, que comprometeram parcialmente os resultados finais de duas das três rodadas de execução da tarefa, porém acredita-se que, uma vez observados, a solução destes problemas estava a pleno alcance. Por fim, julga-se que os preceitos do projeto foram atendidos e que, mesmo com as dificuldades encontradas, os resultados observados foram positivos.

Este trabalho aponta para várias linhas distintas a serem seguidas. Pode-se dar como exemplo, dentre essas linhas, o estudo e aplicação de diferentes tipos de coordenação de comportamentos, e a implementação de uma arquitetura híbrida entre a abordagem baseada em comportamentos e a deliberativa. Outra alternativa viável é a realização de estudos de casos utilizando robôs que apresentem cinemática e/ou dinâmica diferente – robôs com quatro patas ou humanoides, por exemplo – mas ainda tendo por base arquiteturas similares com as adotadas neste trabalho. Enfim, partindo deste projeto, há uma vasta gama de possibilidades envolvendo o problema de navegação autônoma a serem exploradas. Algumas já foram extensamente documentadas na literatura. Muitas ainda não foram vislumbradas por ninguém.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ARDUINO. **Arduino Uno**. Disponível em:<http://arduino.cc/en/Main/ArduinoBoardUno> Acesso em 11 dez. 2015.

ARKIN, R. C. **Behavior-based robotics**. 1. ed. Cambridge, MA: The MIT Press. 1998.

ARKIN, R.C. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In: 1987 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 3, 1987, Raleigh. **Proceedings 1987 IEEE International Conference on Robotics and Automation**. Washington, D.C: Computer Society Press of the IEEE, 1987. p. 264 - 271.

BISHOP, B. E. Book reviews: Behavior-Based Robotics. **Journal Automatica**, v. 38, n. 12, p. 2193-2196, dez. 2002.

BROOKS, R. A. A robust layered control system for a mobile robot**. IEEE Journal of Robotics and Automation**, Cambridge, MA, v. 2, n. 1, p. 14-23, 1986.

BROOKS, R. A. Intelligence without representation. **Artificial Intelligence**, v. 47, p. 139-159, 1991.

BROOKS, R. A. Planning is just a way of avoiding figuring out what to do next. **A.I. Memo 1227**, MIT AI Laboratory, 1990. Disponível em: <http://people.csail.mit.edu/brooks/papers/AIM-1227.pdf>. Acesso em: 11 dez. 2015.

BROOKS, R. A. The behavior language: user's guide. **Working Paper 303**, MIT AI Laboratory, 1997. Disponível em: <https://dspace.mit.edu/bitstream/handle/1721.1/41202/AI_WP_303.pdf?sequence=4>. Acesso em: 11 dez. 2015.

BROOKS, R. A.; FLYNN, A. M. Fast, cheap and out of control: a robot invasion of the solar system. **Journal of the British Interplanetary Society**, v. 42, 10, p. 478 - 485, 1989.

CÉSAR, D. R; BONILLA, M. H. S. Robótica livre: implementação de um ambiente dinâmico de robótica pedagógica com soluções tecnológicas livres no CET CEFET em Itabirito - Minas Gerais – Brasil. In: XIII WIE – WORKSHOP SOBRE INFORMÁTICA NA ESCOLA, 2007, Rio de Janeiro. **Anais do XIII WIE – Workshop sobre Informática na Escola**. Porto Alegre, Rio Grande do Sul: Editora SBC, 2007.

DOSWALD-BECK, L.; HERBY, P.; DORAIS-SLAKMON, J. Basic Facts: the human cost of landmines. **Landmines in Africa: fact sheet**. International Committee of the Red Cross, 01 jan. 1995. Disponível em: <https://www.icrc.org/eng/resources/documents/misc/57jmcy.htm>. Acesso em: 11 dez. 2015.

FILHO, D. M.; GONÇALVES, P. Robótica educacional de baixo custo: uma realidade para as escolas brasileiras. In: XIV WORKSHOP DE INFORMÁTICA NA ESCOLA, 2008, Belém. **Anais do XIV WIE - Workshop de Informática na Escola**. Porto Alegre, Rio Grande do Sul: Editora SBC, 2008.

FRANKLIN, S; GRAESSER, A. Is it an agent, or just a program?: a taxonomy for autonomous agents. In: THIRD INTERNATIONAL WORKSHOP ON AGENT THEORIES, ARCHITECTURES, AND LANGUAGES, 3, 1996, Budapeste. **Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages** Budapeste: Springer Berlin Heidelberg, 1996. p. 21-35.

GOLDBERG, D.; MATARIC, M. In: BALCH, T.; PARKER, L. E. **Robot teams**: from diversity to polymorphism. 1. ed. Natick, MA: A. K. Peters. 2002. cap. 11, p. 315 - 344.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 8373:2012**: Robots and robotic devices — Vocabulary. Genebra, 2012.

MAES, P.; BROOKS, R. A. Learning to coordinate behaviors. In: EIGHTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-90), 8, 1990, Boston, MA. **AAAI'90 Proceedings of the eighth National conference on Artificial intelligence**. Menlo Park, CA: The AAAI Press, 1990. p. 796 - 802.

MATARIC, M. Behaviour-based control: examples from navigation, learning, and group behaviour. **Journal of Experimental & Theoretical Artificial Intelligence**, v. 9, 2, p. 323 - 336, 1997.

MATARIC, M. Behavior-based control: main properties and implications. **IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems**, Nice, França, p. 46-54, 1992.

MATARIC, M. Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. **Trends in Cognitive Science**, Cambridge, MA, v. 2, n. 3, p. 82 - 87, mar. 1998.

MIRANDA, L. C. de. **RoboFácil**: Especificação e Implementação de Artefatos de Hardware e Software de Baixo Custo para um Kit De Robótica Educacional. 2006. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Federal do Rio de Janeiro, Rio de Janeiro e 2006.

MURPHY, R. R. **Introduction to AI robotics**. 1. ed. Cambridge, MA: MIT Press, 2000.

PARALLAX. **Boe-Bot Robot Kit - USB**. Disponível em: <https://www.parallax.com/product/28832> Acesso em: 1 dez. 2015.

PARALLAX. **Robotics with the Boe-Bot**: Student Guide. 3. ed. Rocklin, CA: 2010. 310 f. Disponível em: <https://www.parallax.com/downloads/robotics-boe-bot-text> Acesso em: 11 dez. 2015.

PARALLAX. **PING))) ultrasonic distance sensor (#28015)**. 2. ed. Rocklin, CA: 2013. 9 f. Disponível em: <https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf> Acesso em: 11 dez. 2015.

PAYTON, D. An architecture for reflexive autonomous vehicle control. In: 1986 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 3, 1986, São Francisco, CA. **Proceedings 1986 IEEE International Conference on Robotics and Automation**. Washington, D.C: Computer Society Press of the IEEE, 1986. p. 1838 - 1845.

ROSENBLATT, J. K.; THORPE, C. E. Combining multiple goals in a behavior-based architecture. In: 1995 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEM, 1995, Pittsburgh, PA. **Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots**. Los Alamitos, CA: IEEE Computer Society Press, 1995. p. 136 - 141.

SECCHI, H. **Uma introdução aos robôs móveis.** e. abr. 2012. Espírito Santo: IFES, 2012. Disponível em: <http://www.iwatufes.com/RobMov/RobosMoveis.pdf>. Acesso em: 11 dez. 2015.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to autonomous mobile robots**. 1. ed. Cambridge, MA: MIT Press, 2004.

SOUSA, R. V. **Robô agrícola móvel (RAM)**: uma arquitetura baseada em comportamentos hierárquicos e difusos para sistemas autônomos de guiagem e navegação. 2007. Tese

(Doutorado em Engenharia Mecânica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2007.

TREVELYAN, J. Robots and landmines. **The Industrial Robot**, v. 24, n. 2, p. 114-125, jan. 1997.

VAN HENTEN, E. J. et al. An autonomous robot for harvesting cucumbers in greenhouses. **Autonomous Robots**, v. 13, n. 3, p. 241–258, 2002.

WAHDE, M. Evolutionary robotics: the use of artificial evolution in robotics. 2004. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 17, 2004, Sendai, Japan. **A tutorial presented at IROS 2004**. Gotemburgo: 2004. Disponível em: <http://www.me.chalmers.se/~mwahde/robotics/TechReports/TR-BBR-2004-001.pdf>. Acesso em: 11 dez. 2015.

WAHDE, M. **Introduction to autonomous robots**. Gotemburgo: 2015. 122 f. Notas de aula. Disponível em: <http://www.me.chalmers.se/~mwahde/courses/aa/2015/FFR125_LectureNotes.pdf >. Acesso em: 11 dez. 2015.

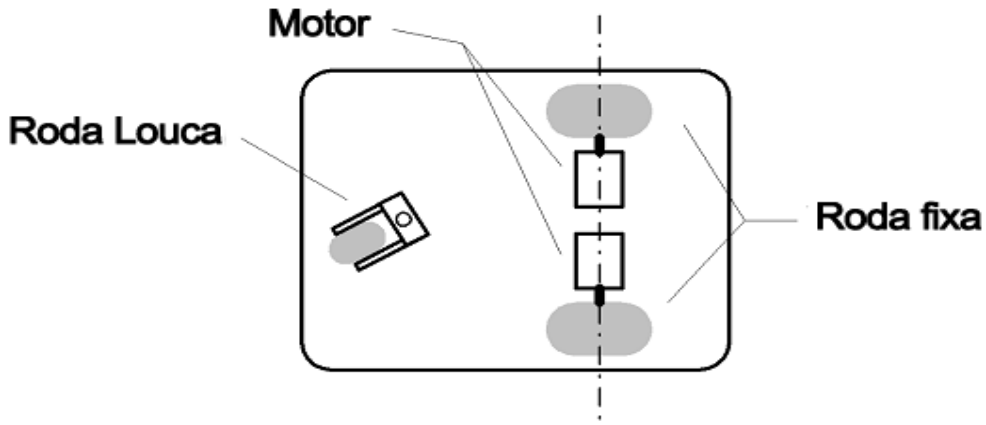
WALSH, N. E.; WALSH, W. S. Rehabilitation of landmine victims: the ultimate challenge. **Bulletin of the World Health Organization**, v. 81, n. 9, p. 665-670, set. 2003.

APÊNDICE A – O robô uniciclo

Esta seção tem por objetivo fornecer ao leitor leigo uma visão geral acerca do robô do tipo uniciclo, tal como o utilizado neste trabalho, para que possa se orientar melhor quanto a alguns conceitos considerados, porém não explicados, no corpo do texto. Aqui é feita uma breve explicação acerca da constituição física, modelo cinemático e odometria deste tipo de robô. Não há pretensões maiores. Ao leitor que desejar se aprofundar no tema, Secchi (2012) realiza uma introdução ao tema robótica móvel de maneira extremamente didática e apresenta diferentes modelos cinemático e dinâmico para estes robôs. Siegwart e Nourbakhsh (2004) também discorrem sobre diferentes tipos de robôs móveis e descrevem a cinemática e funcionamento do robô uniciclo em detalhes.

Um robô móvel uniciclo corresponde a um robô que possui duas rodas equidistantes ao eixo longitudinal, sem movimento lateral instantâneo. A estrutura é balanceada por uma roda orientável não centralizada, livre, também conhecida por roda castor (do inglês *castor wheel*), ou roda louca, que, segundo Secchi (2012, p. 19), tem por principal finalidade dar estabilidade à estrutura mecânica como roda de direção. O esquema básico da estrutura física de tal robô é representado na Figura 17.

Figura 17 – Esquema da estrutura física de robô uniciclo



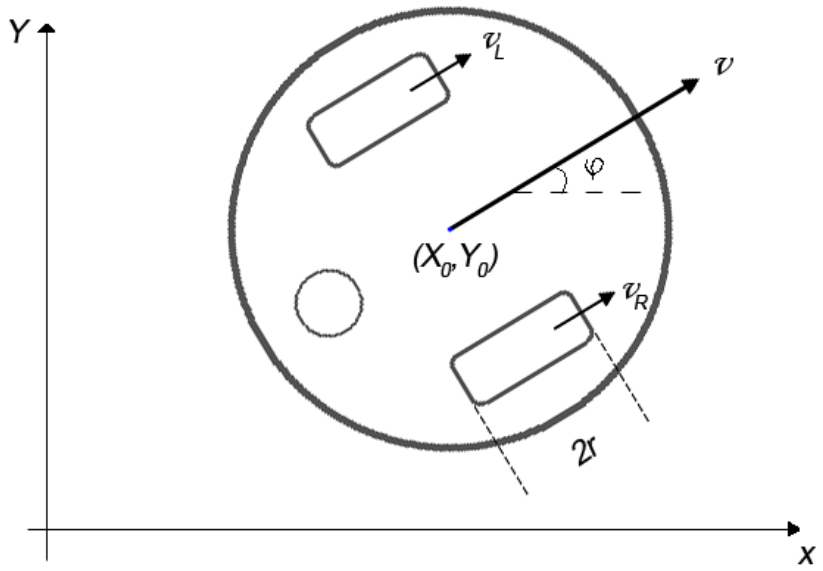
Fonte: SECCHI, 2012. Adaptado.

Este tipo de robô utiliza um sistema de tração diferencial para se locomover, ou seja, as duas rodas sobre o eixo latitudinal são controladas independentemente, cada uma por um motor separado (SECCHI, 2012, p. 22). Segundo Secchi (2012, p. 60), o robô uniciclo apresenta vantagens como alta mobilidade e simples configuração de rodas e, devido a estas vantagens,

é o mais frequentemente utilizado tanto em robôs de pequeno porte como em aplicações industriais.

Cinemática é o processo de determinar o conjunto de movimentos possíveis para o robô sem levar em consideração as forças que agem sobre o mesmo, mas observando-se as restrições impostas ao movimento (WAHDE, 2012, p.21). As equações cinemáticas dependem da estrutura do robô, sendo influenciadas pela quantidade de rodas e eixos, por exemplo. No caso do robô uniciclo, por girar livremente, a influência da roda castor nas equações cinemáticas é desconsiderada, podendo-se, assim, representar o robô tal como esquematizado na Figura 18, este inserido num plano cartesiano de coordenadas *x* e *y*, onde o ângulo φ indica a direção do movimento em relação à horizontal (WAHDE, 2012, p. 22).

Figura 18 – Robô uniciclo: velocidades e orientação



Fonte: Produção do próprio autor.

Assumindo que as rodas apenas se movem em direção perpendicular ao seu próprio eixo e que não há deslizamento, pode-se descrever sua velocidade como:

$$v_i = \omega r \tag{1}$$

Onde:

v_i é a velocidade da roda;

ω é a velocidade angular;

r é o raio da roda.

A cinemática do robô depende das velocidades de cada roda e pode ser descrita utilizando as restrições de movimento impostas ao se definir o robô como um corpo rígido (WAHDE, 2012, p. 23). Para quaisquer valores de velocidade das rodas, o movimento do robô pode ser interpretado como rotação pura, com velocidade angular $\omega = \dot{\varphi}$ em torno do centro instantâneo de rotação. Dessa forma, as velocidades individuais das rodas podem ser descritas por:

$$v_L = \omega(L - R) \tag{2}$$

$$v_R = \omega(L + R) \tag{3}$$

Onde

v_L e v_R são as velocidades da roda esquerda e direita, respectivamente;

L é a distância entre o centro instantâneo de rotação e o centro do robô;

R é o raio do corpo do robô, considerando-o circular e com distribuição de massa simétrica.

Assim, a velocidade v do centro de massa do robô é descrita por:

$$v = \omega L \tag{4}$$

Substituindo as equações 2 e 3 em 4, o parâmetro L pode ser eliminado, e v e φ podem ser descritos apenas em termos das velocidades das rodas, como pode ser observado nas equações a seguir:

$$v = \frac{v_r + v_l}{2} \tag{5}$$

$$\omega = -\frac{v_r - v_l}{2R} \tag{6}$$

O robô uniciclo é capaz de girar sobre seu próprio eixo sem se deslocar, porém não é capaz de se mover no sentido do eixo que une suas rodas de tração. É possível decompor a velocidade do centro de massa do robô e integrá-la no tempo, assim sua posição no plano cartesiano é descrita por:

$$x - x_0 = \int_{t_0}^{t} V_{x(t)} dt = \int_{t_0}^{t} \frac{v_r + v_l}{2} cos\varphi dt \quad (7)$$

$$y - y_0 = \int_{t_0}^{t} V_{y(t)} dt = \int_{t_0}^{t} \frac{v_r + v_l}{2} sen\varphi dt \quad (8)$$

$$\varphi - \varphi_0 = \int_{t_0}^{t} \omega_{(t)} dt = -\int_{t_0}^{t} \frac{v_r - v_l}{2R} dt \quad (9)$$

Onde:

t é o tempo e t_0 é o instante inicial;

(x_0, y_0) é a posição inicial do robô;

(x, y) é a posição inicial do no tempo t;

$V_{x(t)}$ e $V_{y(t)}$ representam a velocidade do centro de massa do robô decomposta nos eixos x e y, respectivamente, em função do tempo;

$\omega_{(t)}$ é a velocidade angular em função do tempo;

e φ_0 é a direção inicial de movimento.

Ao se estimar as velocidades das rodas do robô, sua posição também pode ser estimada a partir do modelo cinemático descrito pelas equações 7, 8 e 9. Ao processo de se estimar a posição e direção do robô dá-se o nome de odometria. Como esta depende da leitura da velocidade, ela está sujeita a erros que se acumulam com o tempo.

APÊNDICE B – O código utilizado na Tarefa 1

O código a seguir é descrito na Seção 5.1.3 e foi embarcado no robô para a realização da última rodada da Tarefa 1. Deve-se observar o pouco tempo existente para se realizar modificações no código entre as rodadas de execução da tarefa, o que pode resultar em linhas de código residuais comentadas ou nunca acessadas. Também é possível que hajam eventuais erros de sintaxe (espaços extra ou em falta, ou a ausência de algum ponto e vírgula, por exemplo) ocasionados durante a transcrição do código para este documento. O leitor deve ficar atento a estes fatores caso queira utilizar este código.

```
#include<Servo.h>
#include<stdlib.h>

Servo servoLeft;
Servo servoRight;
Servo servoSweep;

int stayThreshold = 500;
int irLeftOverTime = 0,irRightOverTime = 0;
int turnT = 30;
int maxOverTime = 50;
int minAngle = 0;
int maxAngle = 180;
int sweepStep = 15;
int led = 6;
int sweepPos,sweepDir;
float lLightThreshold;
float rLightThreshold;
int irFreq = 43000;

void setup()
{pinMode(5, OUTPUT) ; pinMode(9, OUTPUT) ;
pinMode(4, INPUT) ; pinMode(2, INPUT) ; pinMode(led,OUTPUT);
```

```
servoLeft.attach(10);
servoRight.attach(11);
servoLeft.writeMicroseconds(1500);
servoRight.writeMicroseconds(1500);
servoSweep.attach(13);
servoSweep.write(90);
sweepPos = 90;
sweepDir = 1;
lLightThreshold = volts(A3) * 0.30;
Serial. begin(9600);
randomSeed(analogRead(0));
delay(100);
}
void loop() {
//Sweeping the stickers
sweepPos += sweepStep * sweepDir;
servoSweep.write(sweepPos);
if (sweepPos >= maxAngle || sweepPos <= minAngle){
sweepDir *= -1;
}

int irLeft = irReading(String("l"));
int irRight = irReading(String("r"));
int counter = 0;
if (irRight == 0) {
irRightOverTime = min(irRightOverTime + 1, maxOverTime);
} else {
irRightOverTime = max(irRightOverTime -1, 0);
}
if (irLeft == 0) {
irLeftOverTime = min(irLeftOverTime + 1, maxOverTime);
} else {
irLeftOverTime = max(irLeftOverTime - 1, 0);
```

```
}
// read the analog value for left phototransistor
float leftPhoto = volts(A3);
float rightPhoto;
rLightThreshold = lLightThreshold;
float photoReading = leftPhoto;
if (sweepPos <= 90)
{
leftPhoto = lLightThreshold + 1;
rightPhoto = photoReading;
}
else
{
rightPhoto = lLightThreshold + 1;
leftPhoto = photoReading;
}
if (irLeft == 0 && leftPhoto < lLightThreshold)
{
turn90degreesRight();
delay(1000);
}
else if (irRight == 0 && rightPhoto < rLightThreshold)
{
turn90degreesLeft();
delay(1000);
}
else
{
if(irRightOverTime > turnT && irRightOverTime + 1.0/random(1000) > irLeftOverTime) {
turnRight();
irRightOverTime = max(irRightOverTime + random(-4,5),turnT+1);
irLeftOverTime = 0;
} else if (irLeftOverTime > turnT) {
```

```
turnLeft();
irLeftOverTime = max(irLeftOverTime + random(-4,5),turnT+1);
irRightOverTime = 0;
} else {
goForward();
}
if (leftPhoto < lLightThreshold && rightPhoto >= rLightThreshold){
foundMine(String("l"));
}
else if (leftPhoto >= lLightThreshold && rightPhoto < rLightThreshold){
foundMine(String("r"));
}
else if (leftPhoto < lLightThreshold && rightPhoto < rLightThreshold){
// means both found mines
foundMine(String("b"));
}
}
}
void goForward()
{
servoRight.writeMicroseconds(1650-random(100));
servoLeft.writeMicroseconds(1350+random(100));
}
void turnRight()
{
servoRight.writeMicroseconds(1300);
servoLeft.writeMicroseconds(1300);
}
void turnLeft()
{
servoRight.writeMicroseconds(1700);
servoLeft.writeMicroseconds(1700);
}
```

```
void turn90degreesRight()
{
servoRight.writeMicroseconds(1500);
servoLeft.writeMicroseconds(1300);
}
void turn90degreesLeft()
{
servoRight.writeMicroseconds(1700);
servoLeft.writeMicroseconds(1500);
}
void stopMoving(){
servoRight.writeMicroseconds(1500);
servoLeft.writeMicroseconds(1500);
}
int irDetect(int irLedPin, int irReceiverPin, long frequency)
{
tone(irLedPin, frequency, 8) ; // IRLED 38 kHz for at least 1 ms
delay(1) ; // Wait 1 ms
int ir = digitalRead(irReceiverPin) ; // IR receiver -> ir variable
delay(10) ; // Down time before recheck
return ir; // Return 1 no detect, 0 detect
}
int irReading(String leftOrRight){
int irRead = 0;
if (leftOrRight=="l")
{
irRead = irDetect(9, 4, irFreq) ;
}
else
{
irRead = irDetect(5, 2, irFreq) ;
}
return irRead;
```

```
}
void foundMine(String leftOrRight){
stopMoving();
digitalWrite(led, HIGH);
delay(50);
digitalWrite(led, LOW);
if (leftOrRight=="l")
{
turnLeft();
Serial.println("left Detected");
}
else if (leftOrRight=="r"){
turnRight();
Serial.println("right Detected");
}
else{
// means both photo transistor find mines
turn90degreesRight();
Serial.println("both Detected");
}
delay(500);
goForward();
}
float volts(int adPin) // Measures volts at adPin
{ // Returns floating point voltage
return float(analogRead(adPin));// * 5.0 / 1024.0;
}
```

APÊNDICE C – O código utilizado na Tarefa 2

O código a seguir é descrito na Seção 5.2.3 e foi embarcado no robô para a realização da última rodada da Tarefa 2. Deve-se observar o pouco tempo existente para se realizar modificações no código entre as rodadas de execução da tarefa, o que pode resultar em linhas de código residuais comentadas ou nunca acessadas. Também é possível que hajam eventuais erros de sintaxe (espaços extra ou em falta, ou a ausência de algum ponto e vírgula, por exemplo) ocasionados durante a transcrição do código para este documento. O leitor deve ficar atento a estes fatores caso queira utilizar este código.

```
#include <NewPing.h>
#include <Servo.h>
#include <stdlib.h>

Servo servoLeft;
Servo servoRight ;
Servo servoSweep ;

int irPinL = 6 ; // Front IR
int irPinR = 5 ; // Back IR
int photoPin = A4 ;
int ledPin = 3 ;

int minAngle = 10;
int maxAngle = 150;
int sweepStep = 2 ;
int pingPin1 = 9 ;
int pingPin2 = 7 ;
int minDistance = 10;
long turnAccuracy = 3.0;
float detectionThreashold = 15;
```

```
int sweepPos, sweepDir;
unsigned long timer ;

int scanDistance = 100;
int goPastDistance = 30;
int maxDistance = 200;
int minWallDistance = 30;

int robotState = 0 ;
int lastScan [3] ;

float returnTime = 1000000 * 90;
int irAngle ;
float lLightThreshold = 0 ;

void setup( )
{
Serial.begin(9600) ;

pinMode(irPinL, INPUT) ;
pinMode(irPinR, INPUT) ;
pinMode(ledPin, OUTPUT) ;
servoLeft.attach(10);
servoRight.attach(11);
servoLeft.writeMicroseconds(1500);
servoRight.writeMicroseconds(1500);
servoSweep.attach(13);
servoSweep.write(0);
sweepPos = 0;
sweepDir = 1;
randomSeed(analogRead(0));
delay(100);
```

```
timer = micros();
for (int i = 0; i < 5 ; i++){
lLightThreshold += volts(photoPin)∗0.55;
}
lLightThreshold /= 5;
}
void loop() {

// State 0
// Scan and turn into best angle
if (robotState  ==  0){
getBestAngle(lastScan);
turn(lastScan[0]);
robotState = 2;
}

// State 1
// Check if stuck at wall
elseif(robotState == 1){
if(checkWall()){
turn(270);
}
robotState = 0;
}

// State 2
// Move forward
else if (robotState == 2){

boolean photo = false;

// No wall in front of robot
if(lastScan[2]>maxDistance){
```

```
int forwardTime = 4∗1000000;
float time = micros();
forward();
while(minWallDistance&&micros()−time<forwardTime&&!photo){
photo = readPhototrans(photoPin);
}
stopRobot();
}
// Something in front of robot
else{
photo = goForward(minWallDistance);
}
if (photo){
robotState = 5;
}else if (micros()−timer>returnTime){
robotState = 3;
}else {
robotState = 1;
}
}

// State 3
// Search for IR becon
else if (robotState == 3){
digitalWrite(ledPin,LOW);
irAngle=scanForIR();
// Serial.println(irAngle);
if (irAngle == −1){
robotState = 1;
}
else{
turn(irAngle);
robotState = 4;
```

```
digitalWrite(ledPin,HIGH);
}
}

// State 4
// Move towards IR
else if (robotState == 4){
boolean atSafeZone = moveToIR();
if(atSafeZone){
robotState = 5;
}else{
robotState = 3;
}
}

// State 5
else if (robotState == 5){
drop();
robotState = 0;
timer = micros();
}
}


void getBestAngle(int*result){
sweepPos = maxAngle;

int averageOver = 3;

int steps = (maxAngle−minAngle)/sweepStep;
float sensor1[steps],sensor2[steps];
int bestAngle;
int bestDistance=−1;
```

```
int maxObjects=5;
int objects[maxObjects][2];
int currentObject=0;

servoSweep.write(sweepPos);
delay(100);

for(int i = 0; i < steps; i++){
sensor1[i] = readSonarWithAverage(pingPin1,averageOver);
sensor2[i] = readSonarWithAverage(pingPin2,averageOver);

sweepPos −= sweepStep;
servoSweep.write(sweepPos);
}
servoSweep.write((maxAngle+minAngle)/2);
for(int i = 0; i < steps; i++){
if(sensor2[i] > bestDistance){
bestDistance = sensor2[i];
bestAngle = maxAngle−i∗sweepStep;
}
int n = 0;
while(i+n<steps&&sensor2[i+n]−sensor1[i+n]>detectionThreashold){
n++;
}
if(n>1&&currentObject<maxObjects){
objects[currentObject][0]=maxAngle−sweepStep∗(i+n/2);
objects[currentObject][1]=int(sensor1[int(i+n/2)]);
currentObject++;
}
i += n;
}

for(int i = 0; i < currentObject; i++){
```

```
if(objects[i][1]<bestDistance){
bestDistance=objects[i][1];
bestAngle=objects[i][0];
}
}
∗result++ = bestAngle;
∗result++ = bestDistance;
∗result = currentObject > 0;
}
int scanForIR(){

float leftRead = irDetect(irPinL);
float rightRead = irDetect(irPinR);

if (leftRead+rightRead < 0.3){
return −1;
}

int angle = int(90+90∗leftRead−90∗rightRead);
return angle;
}

float irDetect(int irReceiverPin)
{

int readings = 100;
int sum = 0;
for (int i =0; i < 100; i++){
sum += digitalRead(irReceiverPin); // IR receiver −> ir variable
}

return (readings−sum) / float(readings);
}
```

```
boolean moveToIR(){

float forwardTime = 4∗1000000;
float time = micros();
boolean photo = false;

forward();
while(!photo&&micros() - time < forwardTime){
photo = readPhototrans(photoPin);
}

stopRobot();
return photo; // return if back phototransister was positive
}

boolean checkWall(){

int timesToCheck = 2;
float maxDiff = 100;
float reading = readSonarWithAverage(pingPin2,1);
sweepPos = 90;
servoSweep.write(sweepPos);
delay(100);
for (int i = 0; i < timesToCheck; i++){
reading = readSonarWithAverage(pingPin2,1);
if(reading < minWallDistance){
return true;
}
}

sweepPos = 80;
servoSweep.write(sweepPos);
```

```
delay(100);
for(int i = 0; i < timesToCheck; i++){
reading = readSonarWithAverage(pingPin2,1);
if(reading < minWallDistance){
return true;
}
}

sweepPos = 100;
servoSweep.write(sweepPos);
for(int i = 0;i < timesToCheck; i++){
reading = readSonarWithAverage(pingPin2,1);
if(reading < minWallDistance){
return true;
}
}
return false;
}

void drop(){
forward();
delay(2000);
backward();
delay(2000);
turn(random(180,360));
}

void turn(int angle){

float revTime = 2750;
float angleTime = revTime/360;

if(angle < 90){
```

```
turnRight();
delay((90−angle)∗angleTime);
}else if (angle<=270){
turnLeft();
delay((angle−90)∗angleTime);
}else {
turnRight();
delay((angle−90)∗angleTime);
}
stopRobot();
}

boolean goForward(float minWallDistance){
int averageOver = 3;
float dist1 = 0;
float dist2 = 10000;
boolean photo = false;
float forwardTime = 3∗1000000;

servoSweep.write(90);
forward();
while(dist2 > minWallDistance&&
dist2−dist1>detectionThreashold&&!photo){
dist1 = readSonarWithAverage(pingPin1,averageOver);
dist2 = readSonarWithAverage(pingPin2,averageOver);
photo = readPhototrans(photoPin);
}

float time = micros();
while(dist2 > minWallDistance&&
micros()−time < forwardTime&&!photo){
dist2 = readSonarWithAverage(pingPin2,averageOver);
```

```
photo = readPhototrans(photoPin);
}
stopRobot();
return photo;
}

void forward()
{
servoRight.writeMicroseconds(1700);
servoLeft.writeMicroseconds(1300);
}
void backward()
{
servoRight.writeMicroseconds(1300);
servoLeft.writeMicroseconds(1700);
}
void stopRobot(){
servoRight.writeMicroseconds(1500);
servoLeft.writeMicroseconds(1500);
}
void turnLeft()
{
servoRight.writeMicroseconds(1300);
servoLeft.writeMicroseconds(1300);
}
void turnRight()
{
servoRight.writeMicroseconds(1700);
servoLeft.writeMicroseconds(1700);
}
float readSonarWithAverage(int pin,int averageOver){
int maxReads = 5;
```

```
int reads, tmp;
float sum = 0;

for(int i = 0;i < averageOver; i++){
tmp = 0;
reads = 0;
while(tmp==0&&reads<maxReads){
tmp = readSonarSensor(pin);
delay(15);
reads++;
}
sum += tmp;
tmp = 0;
}
return sum/averageOver;
}
boolean readPhototrans(int pin){
float v = volts(photoPin);
Serial.println(v);
if(v < lLightThreshold){
return true;
}
else{
return false;
}
}
long readSonarSensor(int pingPin){
NewPing sonar(pingPin, pingPin, 350);
return sonar.pingcm();
}
float volts(intadPin) // Measures volts at adPin
{// Returns floating point voltage
```

```
return float(analogRead(adPin));//∗5.0/1024.0;
}
```