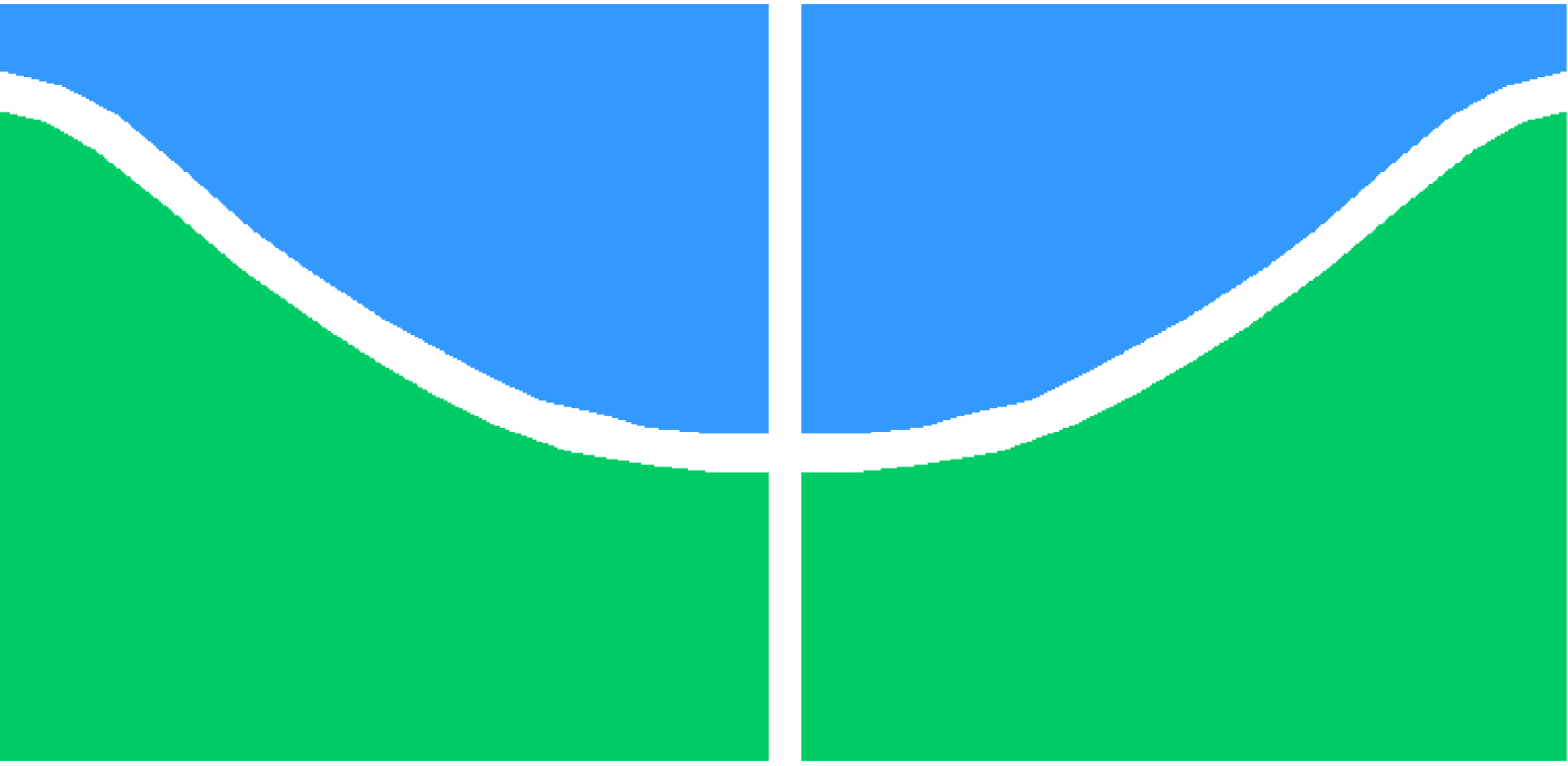

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Auto-Localização e Mapeamento de Ambientes: Uma Abordagem para Robôs Simples

Autor: Rafael Fazzolino P. Barbosa

Orientador: Prof. Dr. Maurício Serrano
Coorientadora: Profa. Dra. Milene Serrano

Brasília, DF

2017

Rafael Fazzolino P. Barbosa

Auto-Localização e Mapeamento de Ambientes: Uma Abordagem para Robôs Simples

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Maurício Serrano

Coorientador: Profa. Dra. Milene Serrano

Brasília, DF

2017

Rafael Fazzolino P. Barbosa
Auto-Localização e Mapeamento de Ambientes: Uma Abordagem para Robôs Simples/ Rafael Fazzolino P. Barbosa. – Brasília, DF, 2017-
177 p. : il. ; 30 cm.

Orientador: Prof. Dr. Maurício Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2017.

1. Robótica. 2. Auto-Localização. I. Prof. Dr. Maurício Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Auto-Localização e Mapeamento de Ambientes: Uma Abordagem para Robôs Simples

CDU 02:141:005.6

Rafael Fazzolino P. Barbosa

Auto-Localização e Mapeamento de Ambientes: Uma Abordagem para Robôs Simples

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 28 de junho de 2017:

Prof. Dr. Maurício Serrano
Orientador

Profa. Dra. Milene Serrano
Coorientadora

Profa. Dra. Carla Silva Rocha Aguiar
Convidado 1

Prof. Ms. Giovanni Almeida Santos
Convidado 2

Brasília, DF
2017

Agradecimentos

Agradeço, primeiramente, à minha mãe, Monica Fazzolino Pinto, por todo o apoio e compreensão durante minha caminhada até aqui. À minha familia como um todo e a todos os meus amigos da universidade, principalmente ao Eduardo Brasil, que me acompanha desde o começo desta caminhada. Agradeço, também, a todos os amigos que obtive durante a vida e à minha namorada, Leticia de Souza, por todo o apoio e paciência durante minha graduação.

Agradeço ao prof. Dr. Maurício Serrano por todo o apoio durante disciplinas, projeto e TCC. E a profa. Dra. Milene Serrano por toda a paciência e ajuda durante o curso como um todo. Agradeço ainda, aos dois professores, pela oportunidade e apoio durante a participação no projeto de pesquisa orientado diretamente por eles.

Agradeço a todos os professores do curso de Engenharia de Software, em especial, ao prof. Dr. Sergio Freitas, por ampliar minhas visões em relação a projetos de pesquisa e extensão na universidade. Entre outros, destaco a profa. Dra. Rejane Costa, por todo o apoio durante o curso.

Resumo

Grande parte das pesquisas relacionadas à robótica é focada, principalmente, na mobilidade do robô. Isto ocorre pela necessidade, na maioria das atividades, da navegação e auto-localização no ambiente. Com este objetivo, a técnica de SLAM (Auto- localização e mapeamento simultâneos de ambientes) vem sendo implementada em diversos contextos por toda a comunidade de robótica. Esta pesquisa buscou analisar técnicas renomadas de auto-localização no contexto da robótica mundial, a partir da execução de uma revisão sistemática sobre o tema, selecionando a técnica do Filtro de Partículas para adaptação e implementação no contexto limitado da Robótica Educacional. Durante as etapas de implementação e análise dos resultados, a pesquisa busca documentar de maneira clara e objetiva os procedimentos realizados, garantindo a possibilidade da execução dos procedimentos por interessados no assunto. Além da aplicação no contexto educacional, deve-se ressaltar que esta pesquisa faz referência a utilização de robôs simples no processo de auto-localização, o que abrange sua utilização também em contextos reais, porém com limitações de *hardware*.

Palavras-chaves: Auto-localização, Filtro de Partículas, SLAM, robôs simples, Mindstorms NXT.

Abstract

This research sought to analyse renowned techniques of auto localization in the context of the current world of robotics, starting from the execution of a sistematic revision about the theme, selecting the Particle Filter to the adaptation and implementation in the limited context of Educational Robotics. During the steps of the analysis of results, the research sought to document clearly and objectively the proceedings, ensuring the possibility of the execution of them by the interested researchers in the theme. Besides the application on the educational context, it must be emphasized that this search makes reference to the utilization of simple robots on the auto localization process, which also includes its utilization on real contexts, with hardware limitations, however.

Key-words: Auto-localization, Particle Filter (Monte Carlo Localization), SLAM problem, simple robots, educational robotic, Mindstorms NXT.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MIT	Massachusetts Institute of Technology
API	Application Program Interface
BPMN	Business Process Model and Notation
IDE	Integrated Development Environment
RAM	Random Access Memory
USB	Universal Serial Bus
SLAM	Simultaneous Localization and Mapping

Sumário

1 Introdução

A robótica é um termo que gera curiosidade para qualquer pessoa, desde uma criança aprendendo a ler, até idosos surpresos com as novidades. Por esse e outros motivos, a robótica vem sendo pesquisada e evoluída em diversos contextos, como a robótica educacional, por exemplo.

Entre as áreas de pesquisa que se enquadram na robótica, uma delas merece atenção especial: a robótica móvel. A mobilidade robótica é um dos grandes desafios para a comunidade devido à complexidade e à imprecisão dos dados obtidos, a qual será apresentada durante este trabalho de conclusão de curso. As próximas seções deste capítulo buscam apresentar a contextualização da pesquisa, a questão de pesquisa a ser investigada, a justificativa para realização da mesma, e os objetivos a serem alcançados.

1.1 Contextualização

O nascimento da robótica se deu no contexto industrial, no qual ferramentas autônomas foram desenvolvidas para executar atividades de forma repetitiva e incansável, maximizando a qualidade dos produtos e minimizando o custo e o tempo para produção dos mesmos [Romano 2002]. Segundo [Romano 2002], a palavra robô é derivada da palavra *robota*, de origem eslava, que significa *trabalho forçado*, ou seja, robôs podem ser considerados ferramentas incansáveis que apóiam o trabalho humano.

Segundo [Oliveira 2008], a autonomia de um robô é condicionada pela sua capacidade de perceber o ambiente de navegação, interagindo com o meio e realizando tarefas com o mínimo de precisão. Este mínimo, de acordo com [Oliveira 2008], seria a navegação sem colisão em obstáculos.

Para que robôs sejam capazes de navegar em um ambiente desconhecido sem que haja colisão em objetos e obstáculos, os mesmos necessitam de informações sobre este ambiente. Estas informações são adquiridas utilizando sensores. Como foi apresentado por [Costa e Okamoto Jr. 2002], no livro de Robótica Industrial, os sensores possuem o dever de fornecer informações ao sistema de controle do robô sobre distâncias de objetos, posição do robô, contato do robô com objetos, força exercida sobre objetos, cor e textura dos objetos, entre outras.

Além de obter dados sobre o ambiente, o robô precisa se auto-localizar para processar as informações obtidas e traçar rotas sem colisões até o ponto de destino. Para isso, foram desenvolvidas muitas formas de auto-localização, algumas delas são citadas por [Santos, Silva e Almeida 2002], como:

- **Utilização de Mapas**: O robô conhece o mapa onde realizará a navegação à priori, conhecendo os obstáculos e os caminhos possíveis. Possuindo essas informações, o robô irá traçar as rotas mais eficientes para chegar em seu objetivo.

- **Localização Relativa em Grupos**: Esta técnica utiliza a navegação simultânea de muitos robôs, cada robô sabe a posição relativa dos outros robôs, podendo calcular sua posição relativa.

- **Utilização de Pontos de Referência**: Conhecendo pontos de referência que estão distribuídos pelo mapa de navegação, o robô consegue calcular sua posição através da técnica de triangulação.

- **Localização Absoluta com GPS**: A partir desta técnica, é fácil obter a posição absoluta do robô em relação à terra. O grande problema desta técnica é a margem de erro presente no sistema de GPS, inviável para navegações internas.

- **Utilização de Bússolas**: É uma técnica interessante para reconhecimento da orientação do robô, o que facilita na navegação do mesmo. Entretando, as Bússolaslas são muito suscetíveis a interferências externas, como por exemplo, a proximidade de materiais ferro-magnéticos ou as fugas magnéticas dos motores presentes no próprio robô.

- **Odometria**: Consiste na medição da distância relativa percorrida pelo robô, utilizando sensores presentes nas rodas do mesmo. Necessita do conhecimento do ponto de origem. Apresenta um erro cumulativo que cresce com o decorrer da navegação, ou seja, é proporcional a distância percorrida e ao número de mudanças na orientação do robô.

As formas apresentadas anteriormente, para se trabalhar com auto-localização, possuem características únicas que as adequam para diferentes contextos de navegação. Por exemplo, segundo [Santos, Silva e Almeida 2002], a Utilização de Mapas é uma técnica bastante útil quando se está trabalhando com um ambiente conhecido e estático, porém, em ambientes mutáveis e não conhecidos, essa estratégia se torna um problema. A Localização Relativa em Grupos é a técnica adequada quando a navegação envolve muitos robôs, a qual não necessita de conhecimento prévio do mapa. A Utilização de Pontos de Referência é uma técnica comumente utilizada, sendo útil quando não se conhece o ambiente de navegação. Entretanto, nesse caso, o ambiente deve ser adaptado para instalação destes pontos de referencia.

Quando se têm ambientes abertos e amplos, a técnica de Localização Absoluta com GPS é a mais utilizada. Entretanto, sua margem de erro torna a navegação em ambientes pequenos ou fechados inviável, como apresenta [Santos, Silva e Almeida 2002].

A Navegação com utilização de Bússolas garante um apoio muito útil para orientação do robô. Contudo, essa técnica gera problemas relacionados a interferências externas, como materiais eletromagnéticos próximos à bússola [Santos, Silva e Almeida 2002].

A técnica de Odometria é muito utilizada em navegações curtas, em ambientes com o piso regular e plano. Entretanto, segundo [Santos, Silva e Almeida 2002], esta técnica se caracteriza pela adição de erros a cada centímetro percorrido, por meio de derrapagens e falhas no giro das rodas.

Desse modo, é fácil perceber que cada técnica possui características que se adaptam melhor para diferentes situações. Ao longo deste trabalho, o principal foco de interesse é a navegabilidade de robôs simples e baratos, como os Kit Lego Mindstorms[1]. Tais kits possuem poucas opções de sensores e características limitadas [Hámori, Lengyel e Reskó 2011]. Os sensores do kit que serão utilizados neste trabalho são: *Sensor de proximidade, sensor RGB, sensor de contato e sensor odométrico.*

1.2 Problema de Pesquisa

Todas as técnicas de auto-localização apresentadas na seção anterior já foram testadas, comparadas e refatoradas em diferentes contextos, desde a navegação marítima até questões relacionadas à tecnologia aeroespacial [Oliveira 2008]. Porém, sabe-se que, nestes contextos, o *hardware* utilizado para navegação é de alta tecnologia, possuindo processadores de alto desempenho, com a utilização de processamento paralelo [Paula et al. 2013], por exemplo, e sensores precisos.

Já em um contexto educacional, a infraestrutura disponível nem sempre engloba os critérios necessários para aplicação das técnicas de auto-localização. Um exemplo disso é a utilização, por [Oliveira 2008], de uma câmera omnidirecional para obter informações sobre o ambiente.

Desse modo, vê-se a necessidade da verificação de viabilidade da utilização de técnicas de auto-localização no contexto da Robótica Educacional, utilizando os kits de robótica *Mindstorms* da LEGO, neste caso. Nesses kits, a capacidade de processamento e a precisão dos sensores e dos atuadores são limitadas. A intenção é tomar como base a técnica de mapeamento do ambiente e auto-localização simultâneos (conhecida como problema de SLAM - *Simultaneous Location and Mapping*) [Dissanayake et al. 2001], utilizando o Filtro de Partículas como técnica principal da pesquisa.

A questão de pesquisa que será discutida durante este trabalho é *"Como tratar o problema de SLAM no contexto de robôs simples?"*.

[1] mindstorms.lego.com

1.3 Justificativa

A utilização da Robótica como uma forma de apoio ao aprendizado em escolas e faculdades, conhecida como Robótica Educacional [Maliuk 2009], traz alguns benefícios para o aluno. Conforme colocado pelos autores [Galvan et al. 2006], [Zhao et al. 2008], [Maliuk 2009] e [Wahab1, Azahari2 e Tajuddin1 2015], alguns desses benefícios são:

- maior interesse pelos conteúdos estudados em aula;
- capacidade de trabalhar em grupo;
- aplicação prática do conhecimento teórico, e
- multidisciplinaridade.

A Universidade de Brasília utiliza esta abordagem de ensino/aprendizagem durante a disciplina de Princípios de Robótica Educacional, ministrada desde 2013, pelo professor Dr. Maurício Serrano. A disciplina utiliza os Kits de robótica Mindstorms, da Lego, para desenvolvimento de soluções dos problemas presentes em um tapete de missões. A organização da disciplina se inspira nos campeonatos de robótica, como o *ciber-rato* [Lau et al. 2002] ou *micro-rato* [Santos, Silva e Almeida 2002]. Neste tipo de campeonato, a navegação é o quesito mais importante [Lau et al. 2002], a qual deve possuir a menor margem de erro para solucionar as missões.

As missões utilizadas durante a disciplina da UnB são referentes a problemas recorrentes no contexto da robótica mundial. A solução dos problemas é uma adaptação das técnicas existentes para o contexto limitado da disciplina, onde são utilizadas apenas ferramentas presentes no kit *Mindstorms* da Lego. Esta adaptação exige um conhecimento específico sobre a técnica, para que o estudante possa identificar características relevantes e adaptá-las de acordo com o *hardware* disponível.

A disciplina já influenciou dois estudantes de Engenharia de Software a se aprofundarem no contexto da Robótica e Robótica Educacional, gerando dois trabalhos de conclusão de curso, [Rincon 2015], que desenvolveu um *framework* de definição de trajetórias para robôs móveis, e [Ramalho 2015], que desenvolveu um algoritmo para tomada de decisões estratégicas em robótica educacional. O trabalho atual veio com o objetivo de complementar os trabalhos realizados pelos dois alunos.

1.4 Objetivos

1.4.1 Objetivo Geral

Analisar técnicas de resolução do problema de SLAM para o contexto de robôs simples, utilizando os kits de robótica *Mindstorms* da Lego, em um primeiro momento.

1.4.2 Objetivos Específicos

- Identificar soluções para o problema de SLAM em um contexto simplificado;
- Propor adaptações para o contexto de robôs simples, na Robótica Educacional;
- Implementar adaptação;
- Analisar viabilidade da resolução do problema de SLAM no contexto limitado da Robótica Educacional.

1.5 Organização do Trabalho

Este trabalho de conclusão de curso está organizado nos capítulos:

- Introdução: Capítulo referente à contextualização, levantamento da questão de pesquisa, justificativa e definição dos objetivos do trabalho;
- Referencial teórico: O objetivo deste capítulo é fornecer ao leitor o conhecimento necessário para compreender a pesquisa realizada. O capítulo é sub-dividido nas seções *robótica e a auto-localização*, *o problema de SLAM* e *robótica educacional*;
- Suporte tecnológico: Apresenta as ferramentas e tecnologias utilizadas para auxiliar o desenvolvimento desta pesquisa, desde a pesquisa bibliográfica e documentação, até o desenvolvimento da prova de conceito e apresentação;
- Proposta: Capítulo destinado a apresentação da proposta do trabalho como um todo, definindo o ambiente de trabalho, como o mapa utilizado e a montagem do robô.
- Metodologia: Este capítulo busca apresentar as técnicas utilizadas para a realização da pesquisa, definindo as atividades a serem desempenhadas para conclusão do trabalho, e
- Resultados parciais: Neste capítulo, são apresentados os resultados obtidos durante o desenvolvimento da primeira etapa deste trabalho de conclusão de curso.

- Simple Slam: Neste capítulo é apresentado o produto utilizado para análise das técnicas, assim como as características de montagem e configuração do mesmo.
- Cenários de Teste: Neste capítulo são apresentados os cenários de teste para análise da viabilidade da utilização destas técnicas no contexto simplificado.
- Consiferações finais: Capítulo final do TCC_1 que tem como objetivo apresentar o status atual do trabalho, assim como o que se espera para a próxima etapa do trabalho.

2 Referencial Teórico

Parte da teoria relacionada à robótica móvel e à robótica educacional será descrita, brevemente, ao longo deste capítulo, visando facilitar o entendimento dos termos utilizados durante a realização desta pesquisa.

2.1 Robótica e a Auto-Localização

O nascimento da robótica se deu no contexto industrial, onde a automação de atividades repetitivas garantiu maior eficiência e, consequentemente, maior lucro [Romano 2002]. Porém, com o passar dos anos, a robótica vem se expandindo e fazendo parte da vida cotidiana de muitos [Galvan et al. 2006]. A robótica é uma importante ferramenta que apóia o trabalho humano em diversos contextos, seja para a limpeza de uma casa [Martins 2008] ou até para explorar novos planetas, por exemplo [Maimone, Cheng e Matthies 2007].

No contexto da robótica móvel, existem, por exemplo, os robôs de serviço. Esses robôs vêm sendo largamente evoluídos pela comunidade de robótica [Pinto 2008]. Ainda segundo [Pinto 2008], estes robôs, geralmente, desempenham ações que contemplam aplicações, como:

- Aplicações que envolvam risco de vida significativo para humanos [Maimone, Cheng e Matthies 2007];
- Funções economicamente desvantajosas no uso de trabalhadores humanos [Romano 2002];
- Uso humanitário (cadeiras de rodas autônomas, por exemplo);
- Uso educacional [FARROKHSIAR, KRYS e NAJJARAN 2010].

Robôs de serviço capazes de se deslocar livremente pelo ambiente, conhecendo-o, poderão realizar suas atividades de forma mais eficiente [Pinto 2008], o que garante sua autonomia.

Segundo [Romano 2002], a palavra *automação* traz à mente a noção de que a máquina será capaz de sentir e interagir com o ambiente, conseguindo se localizar e navegar por ele, executando suas atividades. Para que esta navegação seja possível, o robô precisa obter informações sobre o ambiente, as quais são obtidas a partir da utilização de sensores [Chew et al. 2009]. Segundo [Machado 2003], existem inúmeros tipos de sensores, desde

sensores de toque até sensores de visão ou de som. Os sensores mais utilizados em robôs móveis, segundo [Machado 2003], são:

- *Odômetro*:

  São sensores de implementação simples e de baixo custo. Este tipo de sensor conta a quantidade de rotações de cada roda do robô, o que permite calcular o trajeto percorrido pelo mesmo. Esta técnica é conhecida como *dead-reckoning*, como é apresentado por [Won et al. 2008]. Segundo [Machado 2003], técnicas como o *dead-reckoning* são bastante suscetíveis a erros, graças ao não alinhamento das rodas, derrapagens das mesmas e até erros no sinal dos sensores.

- *Câmera*:

  A utilização de câmeras pode ser bastante útil quando se deseja navegar em um ambiente fechado, construído pelo homem e com características bem definidas, como afirma [Machado 2003]. Porém, sua utilização possui uma exigência computacional [Oliveira 2008] que muitas vezes pode inviabilizá-la.

- *Sonar*:

  É um tipo de sensor de proximidade barato e de fácil utilização e, por esse motivo, é bastante utilizado em robôs móveis para ambientes fechados [Machado 2003]. Ainda segundo [Machado 2003], em ambientes abertos, este é um sensor falho, devido ao seu alcance limitado e por não ser direcionado.

- *Infravermelho*:

  É um tipo de sensor muito semelhante aos sonares, porém, estes são direcionados, ou seja, são capazes de identificar a direção do objeto [Pinto 2008].

- *Laser*:

  Também é um tipo de sensor semelhante aos sonares e sensores de infravermelho, entretanto, esses, além de serem direcionados, são mais precisos. Tal característica os tornam equipamentos mais caros, como apresenta [Machado 2003].

A utilização de cada tipo de sensor se dá de acordo com o contexto em que se deseja navegar, levando em consideração a maneira mais viável de entender o ambiente ao seu redor [Machado 2003]. Utilizando os sensores, o robô obterá informações sobre o ambiente, e precisa processá-las para *entender* o mesmo. A maneira de processar (analisar) essas informações também depende do contexto da navegação, como mostra [Santos, Silva e Almeida 2002].

Para que a navegação ocorra sem colisões em obstáculos, o robô precisa, além de informações sobre o ambiente, informações relacionadas a sua posição em relação a

este ambiente [Pinto 2008]. Para solucionar este problema, uma técnica bastante difundida é a utilização de mapas, como mostra [Santos, Silva e Almeida 2002]. Nesta técnica, o robô recebe o mapa do ambiente que se deseja navegar a priori e, a partir deste mapa, traça sua trajetória sem obstáculos. Utilizando este mapa e informações do ambiente, o robô é capaz de se auto-localizar no ambiente, navegando com maior precisão [Santos, Silva e Almeida 2002].

Por outro lado, esta técnica possui requisitos que, muitas vezes, são inviáveis, como o conhecimento prévio do mapa do ambiente. Caso o objetivo seja navegar em um ambiente desconhecido, onde não há mapa nem pontos de referência já conhecidos, deve-se buscar formas de se auto-localizar e navegar utilizando apenas as informações obtidas pelos sensores. Para isso, vem sendo discutida na comunidade de pesquisadores em robótica móvel, a resolução do problema de SLAM (Auto-Localização e Mapeamento de Ambientes Simultâneos). Segundo [Dissanayake et al. 2001], SLAM é considerado, pela comunidade, como o *Santo Graal* da robótica móvel.

2.2 O Problema de SLAM

Auto-localização e Mapeamento de Ambientes Simultâneos (SLAM) é uma técnica bastante utilizada para navegação em diferentes contextos, como na navegação marítima, por exemplo [Dissanayake et al. 2001]. Esta técnica garante a possibilidade do robô navegar em um ambiente desconhecido, construindo um mapa do ambiente e, simultaneamente, utilizar este mapa para se auto-localizar em relação ao ambiente [Dissanayake et al. 2001]. Ou seja, a técnica de SLAM, utilizando conceitos de inteligência artificial, garante autonomia móvel à maquina.

Segundo [Dissanayake et al. 2001], uma máquina capaz de partir de um ponto de origem desconhecido em um ambiente desconhecido e, utilizando seus sensores, mapear o ambiente, utilizando este mapa, simultaneamente, para se auto-localizar no ambiente faz juz à palavra *robô*. Desse modo, com a vontade de criar verdadeiros *robôs*, pesquisadores em robótica móvel vêm desenvolvendo e evoluindo diversas soluções para o problema de SLAM, nos mais diferentes contextos [Won et al. 2008]. Esta busca por uma solução elegante ao problema de SLAM possui o intuito de maximizar a efetividade da navegação autônoma [Pinto 2008].

A possibilidade de se auto-localizar em um ambiente garante façanhas importantes para a robótica, como a resolução do problema do *sequestro do robô*, por exemplo, como mostra [Bukhori, Ismail e Namerikawa 2015]. Este é um problema especial de localização global no campo da robótica móvel, onde o desafio deste problema, segundo [Majdik et al. 2010], é fazer com que o robô seja capaz de se localizar em um mapa após ser sequestrado.

A ocorrência do *sequestro* se dá quando, o robô, enquanto navega em um ambiente, é retirado do seu local e levado a outro totalmente desconhecido [Bukhori, Ismail e Namerikawa 2015]. Em uma situação normal, o robô não seria capaz de entender que não está mais no local onde estava, fazendo com que o mesmo se perdesse. Diversas técnicas para solucionar este problema já foram desenvolvidas, geralmente utilizando como apoio o filtro de partículas, que será apresentado mais à frente [Majdik et al. 2010].

De acordo com [Adams, Mullane e Vo 2013], devido a todas as peculiaridades encontradas na robótica móvel, a mesma se encontra refém da utilização de algoritmos probabilísticos. Desse modo, as diversas soluções desenvolvidas, geralmente, diferem umas das outras pela utilização dos algoritmos probabilísticos. Entre os algoritmos mais utilizados, destacam-se o *filtro de kalman* e o *filtro de partículas*, como mostram as seções 2.2.1 e 2.2.2.

2.2.1 Filtro de Kalman

De acordo com [Dissanayake et al. 2001], a abordagem probabilística mais utilizada para resolução do problema de SLAM é a utilização do filtro de Kalman. [Pinto 2008] descreve o filtro de Kalman como: *"Um algoritmo recursivo de processamento de informações, proporcionando a estimação ótima do estado de um sistema dinâmico com ruído linear"*. Sua utilização está representada de forma clara na Figura 1.

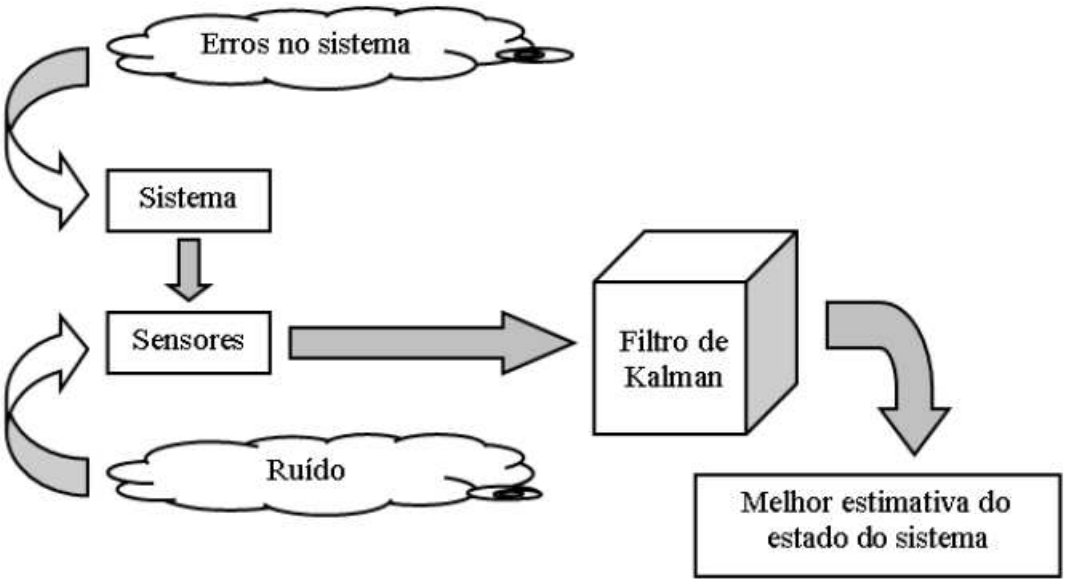

Figura 1 – Utilização do Filtro de Kalman. Fonte: [Machado 2003].

Como mostra [Pinto 2008], entre as disciplinas presentes na utilização deste filtro, encontram-se:

- Mínimos quadrados;
- Teoria das probabilidades;
- Sistemas dinâmicos;
- Sistemas estocásticos, e
- Álgebra.

Apesar das colocações, a utilização da abordagem do filtro de Kalman, geralmente, exige requisitos computacionais de alto custo, dificultando sua utilização em um contexto de robôs simples, onde baixo processamento e pouca memória fazem parte da realidade [Machado 2003].

Em seu trabalho, [Dissanayake et al. 2001] exemplifica este problema, quando quantifica a relação entre o crescimento do requisito computacional necessário para a quantidade de pontos de referência no ambiente, com a utilização do filtro de Kalman. Segundo ele, enquanto a quantidade de pontos de referência aumenta em N, os requisitos computacionais e de armazenamento necessários aumentam em N^2. [Dissanayake et al. 2001] mostra que este problema pode ser solucionado, em parte, utilizando técnicas de aproximação delimitada, por exemplo. Entretanto, estas técnicas minimizam o problema, mas não o solucionam completamente [Dissanayake et al. 2001].

2.2.2 Filtro de Partículas

Outra abordagem bastante utilizada para resolução do problema de SLAM envolve a utilização do filtro de Partículas, uma técnica para implementação de um *Filtro Bayesiano* de forma recursiva, utilizando o método de *Monte Carlo* [Won et al. 2008]. O método de Monte Carlo baseia-se em amostragens aleatórias em grande quantidade, com o objetivo de estabelecer o valor de uma grandeza que não é disponível através de uma expressão matemática [Mooney 1997] e [Neto 2015].

O funcionamento desta abordagem baseia-se em subdividir o ambiente em partículas espalhadas uniformemente, que representam o robô munido de seus sensores [Neto 2015]. Cada partícula é uma hipótese da posição atual do robô [Guizilini et al. 2004]. O robô e as partículas obtêm informações do ambiente; o robô utilizando seus sensores e as partículas utilizando equações matemáticas para geração destes dados [Neto 2015]. As informações obtidas pelo robô real são comparadas com as informações de cada partícula, excluindo as partículas que não oferecem informações semelhantes às do robô real [Neto 2015]. Com o passar dos ciclos, as partículas que ainda continuam no ambiente representam a posição atual do robô [Neto 2015].

A partir da comparação entre estas duas abordagens (filtro de Kalman e filtro de Partículas) [Neto 2015], observou-se que as duas possuem vantagens e desvantagens. O filtro de Kalman converge mesmo com um estado inicial impreciso, fornecendo informações sobre as incertezas presentes em cada estágio e permitindo, ainda, a incorporação de toda informação disponível (sensores) para minimizar a margem de erro da estimativa [Neto 2015]. Porém, ele só garante a efetividade da estimativa para sistemas lineares com ruídos gaussianos [Neto 2015]. O mesmo não resolve o problema do *sequestro do robô*, apresentado anteriormente.

Já o filtro de Partículas, permite sua utilização em sistemas sem ruído gaussiano, e soluciona o problema do sequestro do robô [Seifzadeh, Wu e Wang 2009]. Entre suas desvantagens, encontram-se o custo computacional e a dificuldade da definição da quantidade ideal de partículas a serem utilizadas [Neto 2015].

As duas abordagens apresentadas anteriormente são amplamente utilizadas em diversos contextos do mundo real, contemplando questões importantes a serem estudadas e refinadas por estudiosos de engenharia de todo o mundo, desde estudantes que buscam ingressar nesta área até profissionais da área [Chew et al. 2009]. Desse modo, vê-se a necessidade da adaptação de técnicas de resolução do problema de SLAM, seja a partir de uma abordagem baseada no filtro de Kalman, ou de uma abordagem baseada no filtro de Partículas, para um contexto Educacional. Nesse contexto, sensores, capacidades de processamento e memória são bastante limitados [Zhao et al. 2008].

2.3 Robótica Educacional

A utilização da robótica como uma ferramenta de apoio ao aprendizado vem sendo ampliada com o passar dos anos [Galvan et al. 2006]. Principalmente, graças ao reconhecimento, por muitos autores, dos benefícios da utilização destas tecnologias como abordagem de ensino, onde os alunos são inseridos no problema real e instigados a solucioná-lo [Galvan et al. 2006], [Nunes e Santos 2013], [Benitti et al. 2009].

Entre os benefícios apresentados por [Quartiero 2007] e [Costa 2010], pode-se destacar, o maior interesse dos alunos sobre o tema, a abordagem facilitadora para o relacionamento entre aluno, professor e conteúdo, a experiência com trabalho em grupo, a multidisciplinaridade e a construção do conhecimento por parte do aluno.

Referente ao aperfeiçoamento do interesse dos alunos sobre o tema, [Nunes e Santos 2013] e [Galvan et al. 2006] apresentam motivos deste aperfeiçoamento, como:

- Curiosidade sobre a tecnologia;
- Reconhecimento do problema real como um problema cotidiano do aluno, e

- Utilização do princípio *Hands-on* [Costa 2010]

Sobre o aperfeiçoamento da relação entre alunos, professor e conteúdo, [Nunes e Santos 2013] destaca a reformulação do padrão de ensino e aprendizagem durante a aula. De acordo com [Nunes e Santos 2013], no contexto do ensino tradicional, o aluno busca *clonar* o conhecimento do professor, decorando informações para, no futuro, utilizá-los no contexto real. [Benitti et al. 2009] acrescenta que, esta abordagem, além de limitar o aprendizado do aluno ao conhecimento do professor, minimiza a capacidade de aprendizado do aluno, já que o conhecimento não é construído, e sim repassado.

Já na abordagem da utilização de tecnologias no contexto educacional, a resolução prática do problema atacado faz com que alunos e professores trabalhem lado-a-lado, muitas vezes realizando troca de papéis aluno/professor, como apresentam [Nunes e Santos 2013] e [Quartiero 2007]. Desse modo, em muitas ocasiões, os alunos se encontram explicando uma possível solução do problema ao professor, o que acaba com o aprendizado limitado ao conhecimento do professor [Costa 2010]. [Nunes e Santos 2013] caracteriza o professor como um fio condutor do conhecimento, e não a fonte do mesmo.

Já, de acordo com a relação da experiência do aluno em trabalhos em grupo, a abordagem *Hands-on* garante que todos os envolvidos na solução devem interagir entre si, trocando conhecimento e ideias [Galvan et al. 2006]. Qualquer contexto em que se deseja trabalhar, atualmente, envolve inúmeras atividades de trabalho em grupo [Galvan et al. 2006]. Desse modo, o aperfeiçoamento da capacidade de trabalhar em grupo é uma atividade essencial para o futuro profissional do aluno [Costa 2010].

Outra característica importante da utilização da tecnologia como ferramenta de aprendizado é a multidisciplinariedade [Silva et al. 2014], já que conteúdos referentes a diversas disciplinas são trabalhados e adaptados para buscar a solução do problema em investigação. [Galvan et al. 2006] sugere a utilização de atividades multidisciplinares para aprendizado de conteúdos base, como matemática e física, minimizando o problema do conhecimento parcial de certos conteúdos.

Além de todos os benefícios apresentados acima, uma forte qualidade da abordagem *Hands-on* é referente à construção do conhecimento por parte do aluno [Costa 2010]. Esta construção é gerada utilizando conhecimentos básicos de diferentes disciplinas para solucionar um problema que exige a integração de diversos conteúdos, seguindo uma filosofia construcionista, como é apresentado por [Nunes e Santos 2013].

De acordo com [Nunes e Santos 2013], quando o aluno se sente imerso no problema trabalhado, a maneira com que o mesmo aprende é aperfeiçoada, maximizando as relações entre aluno, professor e conteúdo. Este pensamento segue uma filosofia construcionista, a qual, de acordo com [Nunes e Santos 2013], implica no objetivo de ensinar, de forma a produzir o máximo de aprendizagem a partir do mínimo de ensino.

A utilização de aplicações rotineiras do mundo real é uma forma bastante eficaz de obter máximo interesse dos alunos nos conteúdos apresentados [Nunes e Santos 2013]. Desse modo, inserir alunos na resolução de problemas presentes no contexto da robótica móvel, por exemplo, fará com que os mesmos aprendam, com eficiência, diversos temas recorrentes no contexto mundial da robótica móvel [Chew et al. 2009], além de conteúdos presentes em diversas disciplinas, como matemática e física [Maliuk 2009].

O construcionismo é baseado em uma filosofia construtivista [Nunes e Santos 2013], que afirma que o conhecimento não deve ser uma cópia da realidade, e sim uma construção, realizada pelo aluno a partir da sua interação com o contexto do problema [Becker 2009].

Segundo [Chew et al. 2009], a utilização da robótica no meio da educação traz inúmeros benefícios que vão além dos objetivos diretos da melhoria da aprendizagem. De acordo com ele, sua utilização garante a introdução dos estudantes nos problemas recorrentes do contexto de Engenharia mundial. Esta introdução possui grande importância para a evolução do país, já que a demanda por profissionais qualificados na área de Engenharia, no mundo atual, é ampla [Chew et al. 2009]. De acordo com [Silva et al. 2014], em 2014 no Brasil, haviam cerca de 78 mil vagas relacionadas à área de Tecnologia da Informação e apenas 33 mil foram preenchidas, o que comprova a falta de profissionais na área. [Silva et al. 2014] afirma, ainda, que o motivo para esta falta de profissionais é resultado advindo do baixo interesse por parte dos estudantes brasileiros por ciências exatas.

Grande parte deste baixo interesse, segundo [Becker 2009], é devido à metodologia tradicional de ensino, onde os alunos buscam copiar o conhecimento do professor, sem atividades com abordagem construcionista de aprendizado, por exemplo. Uma forma de aumentar o interesse dos alunos, segundo [Maliuk 2009], é a utilização da robótica como uma ferramenta de apoio ao aprendizado, seguindo uma abordagem construcionista.

Atualmente, de acordo com [Silva et al. 2014], no Brasil, o ensino de computação e robótica se restringe aos níveis superiores e técnicos, exceto alguns projetos realizados por universidades em escolas de nível fundamental e médio, como [Serrano e Serrano 2014]. Esta introdução tem como objetivo introduzir noções lógicas e computacionais na Educação Básica, despertando o interesse dos alunos nas ciências exatas e estimulando-os a ingressar na carreira de Engenharia [Silva et al. 2014].

Diversas ferramentas já foram desenvolvidas com o objetivo de apoiar a inserção de alunos da Educação Básica no contexto da computação e robótica. Na seção 3.2, são apresentadas algumas utilizadas por [Silva et al. 2014], [Serrano e Serrano 2014], [Chew et al. 2009]. Ferramentas como essas, são utilizadas em diversos contextos, desde a utilização como apoio ao aprendizado, como mostra [Galvan et al. 2006], até a utilização em campeonatos e jogos, como mostra [Lau et al. 2002].

De acordo com [Silva et al. 2014], algumas abordagens utilizam ferramentas virtuais para facilitar o aprendizado. Por outro lado, segundo [Chew et al. 2009], a utilização de ferramentas reais, equipadas com sensores e atuadores, garante maior interesse e introdução do aluno no contexto da engenharia do mundo real. Desse modo, a ferramenta utilizada neste trabalho será o kit de robótica educacional Mindstorm, da Lego, que tem suas características especificadas na seção 2.3.1.

2.3.1 Lego Mindstorms NXT

Com o passar dos anos, os componentes de *hardware* vêm sendo evoluídos continuamente [Galvan et al. 2006]. Esta evolução faz com que o preço de componentes simples tenda a diminuir, como é o caso de pequenos processadores poderosos e praticamente descartáveis [Galvan et al. 2006]. Desse modo, esta evolução dos componentes de *hardware* vem permitindo a utilização dos mesmos em diversos contextos, como educação e entretenimento.

Com o crescimento de demandas relacionadas a estes componentes, nascem os kits de robótica que integram componentes simples, porém, poderosos para utilização no contexto educacional e de entretenimento [Galvan et al. 2006]. Entre os kits existentes atualmente no mercado, destaca-se o kit de robótica Mindstorms, da Lego, o qual foi desenvolvido na década de 80 por pesquisadores do MIT. O kit utiliza peças de montagem no padrão Lego para construir a estrutura do robô, diversos tipos de sensores e alguns atuadores, para garantir a mobilidade e interação com o ambiente [Galvan et al. 2006].

Desde o lançamento do kit, na década de 90, o mesmo é utilizado por instituições de ensino como ferramenta de apoio ao aprendizado. O sucesso do kit garantiu premiações devivo a sua reusabilidade, modularidade e simplicidade, associado ao seu baixo custo [Galvan et al. 2006].

O cérebro do kit pode ser observado na Figura **??**, onde se encontra um microprocessador disposto de diversos acessos para sensores e atuadores. Em alguns modelos do kit, o cérebro possui uma tela LCD e alguns botões para interação do usuário. Além do cérebro, o kit possui alguns sensores, como o apresentado na Figura 3, e atuadores, como na Figura 4

Figura 2 – Microprocessador Central

Figura 3 – Sensor RGB - Lego

Figura 4 – Atuador - Lego

Além da obtenção do kit completo, como mostra na Figura 5, é possível adquirir componentes separados, como sensores e atuadores específicos, o que garante a possibilidade de adquirir apenas os componentes necessários para o contexto de aplicação, minimizando os custos da sua utilização. A integração com sensores e atuadores externos ao kit é viável a partir da utilização da ferramenta leJOS NXT, que será apresentada ao longo do trabalho.

Figura 5 – Kit Mindsotm Completo - Lego

Devido as vantagens de sua utilização, este kit foi selecionado para ser utilizado durante a realização deste trabalho de conclusão de curso. Possibilitando uma implementação prática da solução proposta ao longo do trabalho.

2.4 Considerações Parciais

A utilização da tecnologia como ferramenta de apoio ao aprendizado, como foi apresentado durante o capítulo, garante diversos benefícios ao aluno e, até mesmo, ao professor. Principalmente, quando o contexto trabalhado envolve problemas reais da robótica, como o problema de SLAM, por exemplo. A partir desta linha de pensamento, este capítulo buscou apresentar informações importantes para viabilizar a compreensão da pesquisa realizada, como a apresentação do problema de SLAM, os filtros probabilísticos mais utilizados e a abordagem educacional da Robótica.

A partir do conteúdo apresentado durante este capítulo, este trabalho busca aplicar técnicas de auto-localização no contexto limitado da robótica educacional, principalmente a técnica do Filtro de Partículas, aproximando os estudos de sala de aula aos contextos da robótica mundial.

3 Suporte Tecnológico

Nesta seção, serão apresentadas ferramentas e tecnologias utilizadas para auxiliar o desenvolvimento deste projeto, desde a organização e definição da metodologia de pesquisa, até o desenvolvimento dos projetos pilotos durante o trabalho. Esta seção está dividida em *Engenharia de Software* e *Robótica Educacional*.

3.1 Engenharia de Software

Neste tópico, serão apresentadas ferramentas e tecnologias voltadas ao contexto da Engenharia de Software que são utilizadas durante este trabalho, como, por exemplo, ferramentas para gerência de configuração e versionamento dos artefatos gerados.

3.1.1 GIT

A ferramenta GIT[1] foi desenvolvida por Linus Torvalds, mesmo criador do Linux, sendo uma ferramenta *open-source*. Disponibiliza uma eficiente forma de versionamento e gerenciamento de projetos.

3.1.2 Github

O Github[2] é uma ferramenta utilizada para hospedagem remota de projetos GIT. A ferramenta contempla uma *Wiki* para documentação do projeto e sistemas de *Issues* e *Milestones*[3] para gerenciamento de atividades.

3.1.3 Bonita BPMN

Ferramenta para modelagem de processos *BPMN*[4], o Bonita[5] 7 foi escolhido graças a sua facilidade de utilização e portabilidade para o sistema operacional Linux.

3.1.4 Linux Mint

O sistema operacional utilizado durante este trabalho é o Linux Mint[6], sistema livre e utilizado, neste projeto, principalmente para documentação do trabalho, utilizando a tecnologia LaTeX.

[1] https://git-scm.com/
[2] https://github.com
[3] https://guides.github.com/features/issues/
[4] http://www.bpmn.org/
[5] http://www.bonitasoft.com/
[6] https://www.linuxmint.com/

3.1.5 MAC OS Sierra

Sistema operacional da Apple, utilizado durante este trabalho para documentação e implementação das provas de conceito.

3.1.6 Windows 7

Sistema operacional da Microsoft[7], utilizado para implementação do código NXJ por possibilitar configuração simplificada da comunicação entre PC/robô.

3.1.7 LaTeX

O LaTeX[8] 3.14 é um sistema para criação de documentos utilizando textos *tex*, foi incialmente desenvolvido por Leslie Lamport, na década de 80. O LaTeX oferece diversos comandos avançados para organização de alto nível de documentos, incluindo facilitadores para citações, bibliografias, fórmulas matemáticas, figuras e tabelas.

3.1.8 Sublime Text 3

O Sublime Text 3[9] é um editor de texto bastante utilizado por programadores, por possuir apoio para diversas linguagens de programação, incluindo textos em LaTeX.

3.2 Robótica Educacional

3.2.1 RoboMind 6.0

É um ambiente de desenvolvimento proposto por Arvid Halma[10], da Universidade de Amsterdam, que tem como objetivo facilitar o ensino de programação para estudantes do ensino básico. Utiliza um robô virtual para exercitar os conceitos de inteligência artificial e lógica de programação. A linguagem utilizada é chamada Robo/Roo, e permite implementação da movimentação do robô em um ambiente 2D (duas dimensões). As funções básicas da ferramenta, são *ver*, *andar*, *pegar*, *pintar*, como mosta a Figura 6.

[7] https://www.microsoft.com/pt-br/
[8] https://www.latex-project.org/
[9] https://www.sublimetext.com/3
[10] http://www.robomind.net/pt/index.html

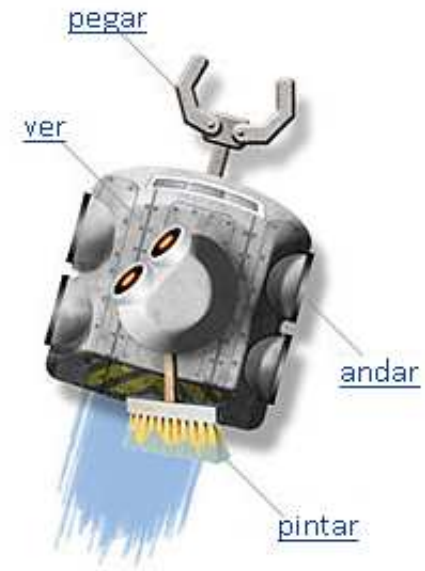

Figura 6 – Funções Básicas RoboMind

3.2.2 Scratch

É um ambiente de desenvolvimento criado por Lofelong Kindergarten Group(LLK)[11], que é um grupo de pesquisa do MIT. Também tem como objetivo introduzir conceitos de programação a alunos da educação básica. Com a utilização desta ferramenta, é possível desenvolver jogos, animações e histórias interativas, de modo visual.

A utilização da ferramenta pode ser visualizada na Figura 7, onde são apresentadas funções básicas no modo visual de desenvolvimento do Scratch.

[11] https://llk.media.mit.edu/projects/

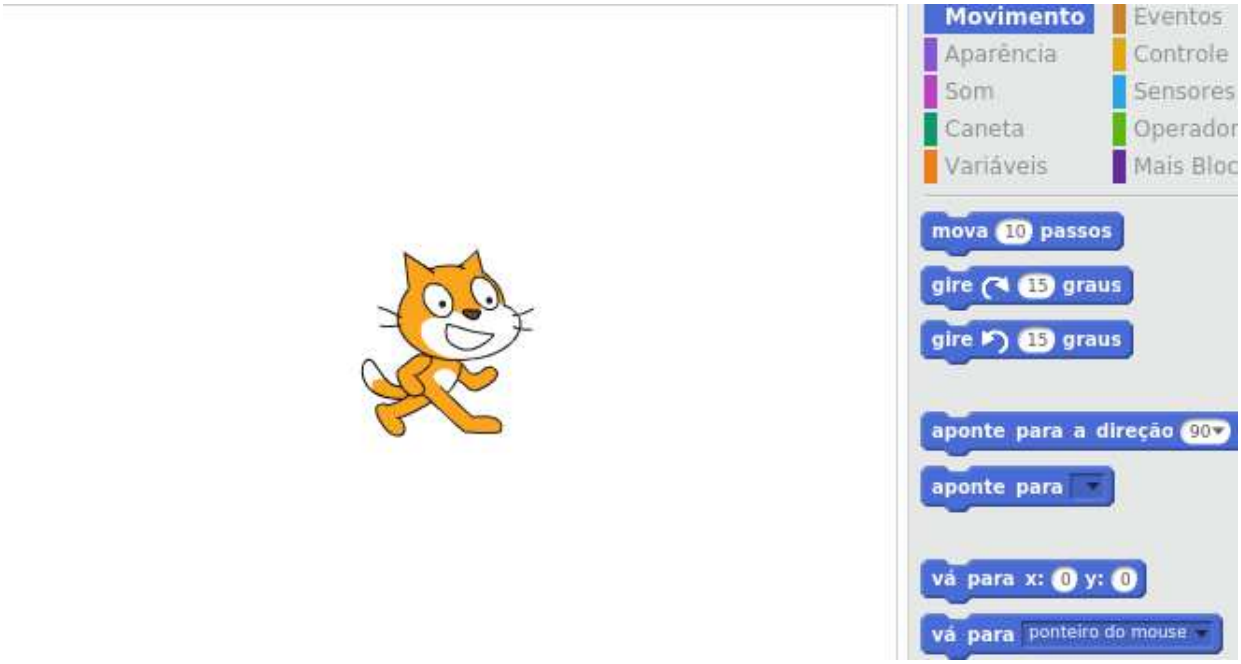

Figura 7 – Funções Básicas Scratch

3.2.3 leJOS NXJ

A ferramenta leJOS NXJ[12] foi utilizada para desenvolvimento, na linguagem Java, de programas a serem executados no Brick NXT. Sua utilização possibilitou o acesso a diversas funcionalidades presentes na API leJOS NXJ, como funções de controle de motores e sensores, por exemplo.

3.2.4 BlueCove

A ferramenta BlueCove[13] foi utilizada neste trabalho para apoiar a comunicação entre o PC e robô. Sua utilização foi necessária devido a necessidade de processamento remoto devido a diversas características que são apresentadas ao longo deste trabalho.

3.2.5 Mindstorms Lego

Os kits de robótica Mindstorms da Lego, diferentemente de todas as ferramentas apresentadas acima, disponibiliza um ambiente real de desenvolvimento. Envolve a implementação de ações dos atuadores presentes em cada robô, onde estas ações serão executadas baseando-se nas informações advindas dos sensores. A especificação desta ferramenta se encontra na seção 2.3.1.

[12] http://www.lejos.org/nxj.php
[13] http://bluecove.org/

3.3 Considerações Parciais

Este capítulo buscou apresentar as ferramentas e tecnologias utilizadas para apoiar o desenvolvimento desta pesquisa, desde a realização da revisão sistemática e documentação do projeto, até a análise de abordagens utilizadas na Robótica Educacional. Analisando as ferramentas utilizadas, observa-se que a base tecnológica do trabalho é resumida em componentes gratuitos e, se possível, *open source*.

A análise das ferramentas citadas se deu com o objetivo de analisar maneiras de se trabalhar o contexto educacional, utilizando ferramentas de apoio e abordagens de ensino. Dentre as ferramentas apresentadas, se encontram ferramentas que apoiam o ensino de diversas áreas de conhecimento, como programação e matemática. Estas ferramentas podem ser reais ou virtuais, porém, como foi apresentado ao longo do trabalho, a ferramenta escolhida como foco deste trabalho foi kit Mindstorms NXT, da Lego.

4 Metodologia

De acordo com [Oliveira 2011], a metodologia de pesquisa adotada neste trabalho pode ser classificada nas seguintes categorias: classificação quanto aos objetivos da pesquisa, quanto à natureza da pesquisa, quanto à escolha do objeto de estudo, quanto à técnica de coleta de dados, e quanto à técnica de análise de dados. A classificação escolhida em cada categoria busca garantir conhecimento amplo sobre diferentes técnicas de solução do problema de SLAM, possibilitando a adaptação de técnicas para um contexto de robôs simples. Nas seções seguintes, são apresentadas as categorias e suas respectivas classificações de pesquisa adotadas para este trabalho.

4.1 Classificação da Pesquisa

De acordo com [Gerhardt e Silveira 2009], uma pesquisa científica pode ser classificada nas categorias *abordagem da pesquisa*, *natureza da pesquisa*, *objetivos da pesquisa*, e *procedimentos*, que são apresentadas nas seções 4.1.1, 4.1.2, 4.1.3 e 4.1.4, respectivamente.

4.1.1 Abordagem da Pesquisa

A classificação deste trabalho em relação à abordagem da pesquisa é definida como *qualitativa*. Segundo [Oliveira 2011], abordagens de cunho qualitativo buscam o significado dos dados obtidos, capturando, além da aparência do fenômeno estudado, suas essências. Dessa forma, a pesquisa qualitativa envolve a obtenção e a análise dos dados descritivos por meio do contato direto entre o pesquisador e o fenômeno estudado [Oliveira 2011].

A abordagem qualitativa não busca definir resultados de maneira quantificável, como mostra [Gerhardt e Silveira 2009], buscando compreender de forma subjetiva o evento estudado. Nesse trabalho, os dados foram coletados em ambiente controlado e analisados de forma não quantificável, buscando caracterizar os resultados como, por exemplo, satisfatórios ou não, caracterizando, assim, segundo [Gerhardt e Silveira 2009], uma pesquisa qualitativa.

Sua utilização, durante este trabalho, teve como objetivo, entender detalhadamente o funcionamento de diversas técnicas de resolução do problema de SLAM, viabilizando a adaptação das mesmas para um contexto limitado. Além disso, a proximidade do pesquisador com o fenômeno a ser estudado possibilitou uma análise descritiva dos dados, de forma que viabilizou uma adaptação de técnicas para contextos limitados.

4.1.2 Natureza da Pesquisa

A classificação deste trabalho em relação à natureza da pesquisa é definida como *pesquisa aplicada*, devido ao fato de buscar solucionar um problema recorrente no contexto mundial da robótica. Segundo [Gerhardt e Silveira 2009], *"a pesquisa aplicada busca gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos"*.

4.1.3 Objetivos da Pesquisa

A classificação deste trabalho em relação aos objetivos da pesquisa é definida como *exploratória*. De acordo com [Oliveira 2011], a pesquisa exploratória tem como objetivo ampliar os conhecimentos do pesquisador sobre o tema. A pesquisa exploratória é, normalmente, o primeiro passo quando o pesquisador não conhece suficientemente a área que pretende abordar [Moresi 2003].

O uso da pesquisa exploratória, neste trabalho, se dá pela necessidade do conhecimento sobre as diversas técnicas de resolução do problema de SLAM, com o objetivo de analisar a viabilidade da auto-localização no contexto de robôs simples.

4.1.4 Procedimentos

Quanto aos procedimentos da pesquisa, a mesma foi dividida em duas etapas. Durante a primeira etapa, o procedimento seguido foi *pesquisa bibliográfica*. Nesse caso, segundo [Gerhardt e Silveira 2009], esse procedimento é feito a partir do levantamento de referências teóricas publicadas por meio de livros, artigos, paginas da web, entre outros.

Já na segunda etapa do trabalho, o procedimento seguido foi o de *pesquisa-ação*. Como [Gerhardt e Silveira 2009] afirma, *"a pesquisa-ação pressupõe uma participação planejada do pesquisador na situação problemática a ser investigada"*. Além da participação do pesquisador, o objeto de estudo da pesquisa-ação não pode ser representado por um conjunto de variáveis que poderiam ser analisadas separadamente [Gerhardt e Silveira 2009].

A utilização da *pesquisa-ação* durante este trabalho foi devido ao conhecimento que precisava ser obtido pelo pesquisador durante a participação ativa na situação estudada, o que possibilitou ampla análise de todo o contexto.

De acordo com a classificação da pesquisa nas diferentes categorias apresentadas acima, a Tabela 1 apresenta, de maneira resumida, a abordagem definida para realização deste trabalho.

Tabela 1 – Metodologia de Pesquisa.

Abordagem da pesquisa	**Natureza da pesquisa**	**Objetivos da pesquisa**	**Procedimentos**
Qualitativa	Aplicada	Exploratória	Pesquisa bibliográfica/ Pesquisa-ação

De acordo com [Marconi e Lakatos 2003], a partir da classificação da pesquisa, fica fácil estabelecer o *caminho* a ser seguido para realização da mesma. Portanto, definiu-se o processo metodológico, o qual é apresentado na seção 4.2.

4.2 Processo Metodológico

O TCC, segundo as regras da UnB, é realizado em duas fases: uma chamada de TCC_01 e outra chamada de TCC_02. Durante o TCC_01, o foco desse trabalho foi estabelecer os pilares teóricos para embasamento do projeto como um todo, bem como desenvolver uma prova de conceito visando estudar a viabilidade da proposta. Já durante o TCC_02, o trabalho esteve focado em atingir o objetivo geral *"Analisar técnicas de resolução do problema de SLAM para o contexto de robôs simples, utilizando os kits de robótica Mindstorms da Lego, em um primeiro momento"*, com base nas fundamentações teóricas e nos resultados acordados na prova de conceito, ambos - fundamentações e resultados - conquistados ao longo do TCC_01.

O TCC_01 pôde ser sub-dividido em oito atividades: *selecionar tema*, *realizar pesquisa bibliográfica*, *definir proposta*, *escrever referencial teórico*, *estabelecer suporte tecnológico*, *evoluir metodologia*, *realizar prova de conceito* e *apresentar TCC 1*. As mesmas estão distribuídas de acordo com o cronograma disposto na Tabela 2.

Tabela 2 – Cronograma de Atividades TCC_1.

Cronograma	**Março**	**Abril**	**Maio**	**Junho**
Selecionar Tema	X			
Realizar pesquisa bibliográfica	X	X	X	X
Definir proposta	X			
Escrever referencial teórico		X	X	
Estabelecer suporte tecnológico		X	X	X
Evoluir metodologia		X	X	
Realizar prova de conceito				X
Apresentar TCC 1				X

Já a segunda etapa do trabalho, durante a realização do TCC_2, o cronograma de atividades seguiu o apresentado na Tabela 3.

Tabela 3 – Cronograma de Atividades TCC_2.

Cronograma	**Março**	**Abril**	**Maio**	**Junho**	**Julho**
Desenvolver provas de conceito	X				
Configurar Montagem e Ambiente	X	X			
Implementar Navegação		X			
Aplicar Filtro de Partículas		X	X		
Analisar Auto-localização			X	X	
Analisar Viés Educacional				X	
Analisar Resultados				X	
Apresentar TCC_2					X

Com o objetivo de definir e acompanhar o processo de desenvolvimento da primeira etapa do trabalho de conclusão de curso, foi modelado um processo metodológico, o qual se encontra na Figura ??.

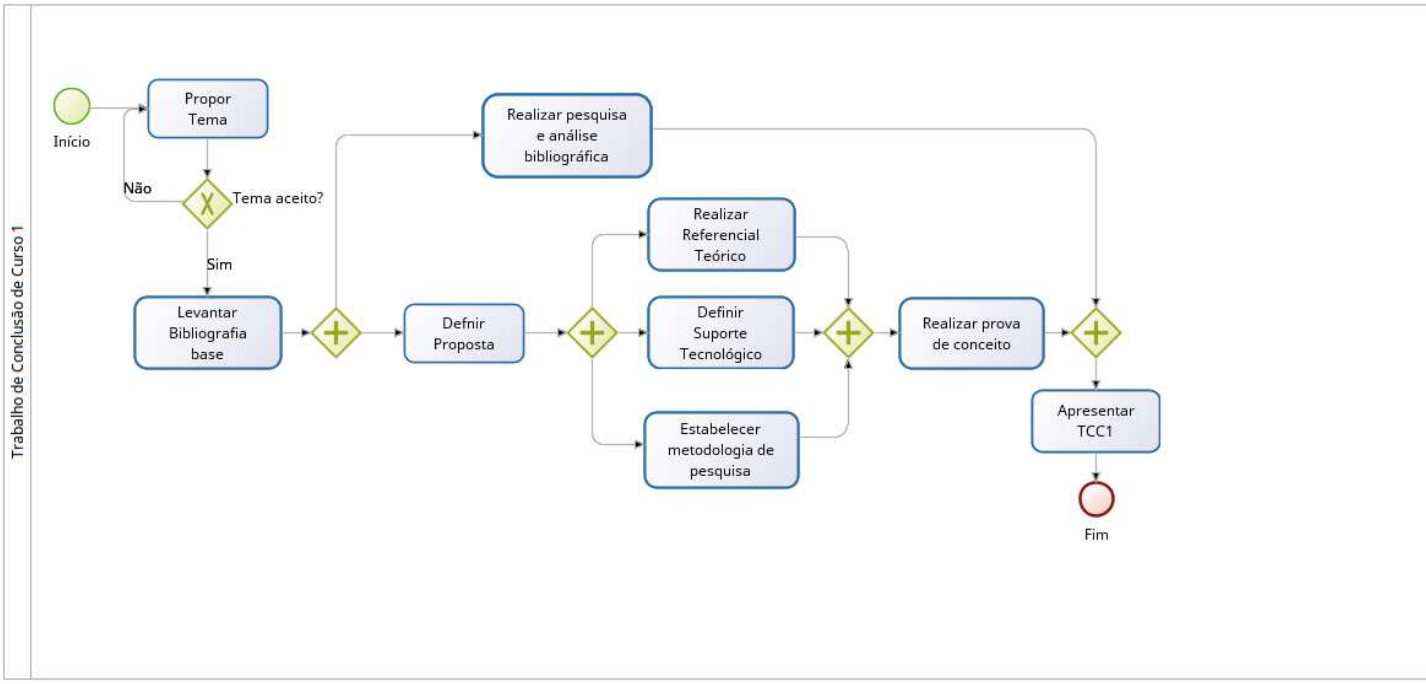

Figura 8 – Processo Metodológico

- **Propor tema**:

A atividade de propor tema engloba desde a escolha do contexto em que se deseja trabalhar, até a definição dos orientadores do trabalho. Após a escolha do contexto e dos orientadores, buscou-se definir um escopo que será abordado durante o trabalho, ou seja, o tema. Os orientadores devem validar o tema escolhido para concluir a atividade.

- **Levantar bibliografia base**:

  Esta atividade refere-se à definição de pilares para o estudo proposto, ou seja, estabelecer o marco teórico do trabalho. Este levantamento garante o entendimento do contexto trabalhado e as possibilidades de atuação, especificando mais adequadamente o escopo.

- **Definir proposta**:

  Documentar a proposta de pesquisa para este trabalho. A proposta inclui, não apenas, mas, principalmente, uma introdução com a contextualização do tema, o objetivo geral e específicos, a justificativa e uma metodologia de pesquisa.

- **Realizar pesquisa e análise bibliográfica**:

  A pesquisa bibliográfica foi feita a partir da utilização da técnica de *revisão sistemática*, com o objetivo de ampliar os conhecimentos em relação ao tema, conhecendo pesquisas em diferentes contextos e de diversas bases científicas, como *IEEE, Springer* e *CAPES*. O detalhamento da revisão sistemática encontra-se no capítulo **??**.

- **Realizar referencial teórico**:

  Trata-se da escrita do capítulo dois deste trabalho. O mesmo descreve o referencial teórico do trabalho em andamento. Como insumos para esta atividade, encontram-se todas as pesquisas bibliográficas obtidas durante a atividade de *Realizar pesquisa e análise bibliográfica*. No segundo capítulo deste trabalho, o tema é especificado com mais detalhe.

- **Definir suporte tecnológico**:

  Nesta atividade, são definidas as principais ferramentas e tecnologias utilizadas para a execução deste trabalho.

- **Estabelecer metodologia de pesquisa**:

  Durante esta atividade, a metodologia de pesquisa inicial, apresentada durante a *proposta*, foi evoluída, com o objetivo de adequar as formas de atuação ao longo da realização do trabalho proposto.

- **Realizar prova de conceito**:

Durante esta atividade, foi realizada a implementação de uma prova de conceito que buscou avaliar a viabilidade da realização deste trabalho. Durante a prova de conceito, que se encontra detalhada na seção B, ferramentas e maneiras de mapear ambientes foram estudadas.

- **Apresentar TCC 1**:

  Apresentar os resultados obtidos até o momento para a banca examinadora.

As atividades de *Levantar bibliografia base* e *Realizar pesquisa e análise bibliográfica* foram complementadas pelo emprego da técnica de revisão sistemática, que segundo [Kitchenham et al. 2006], tem como objetivo identificar, avaliar e interpretar o máximo de pesquisas disponíveis relacionadas a um tema em específico. O planejamento e condução da revisão sistemática é detalhado no Apêndice A.

Durante a realização da segunda etapa desta pesquisa, o processo de desenvolvimento seguiu o apresentado na Figura 9.

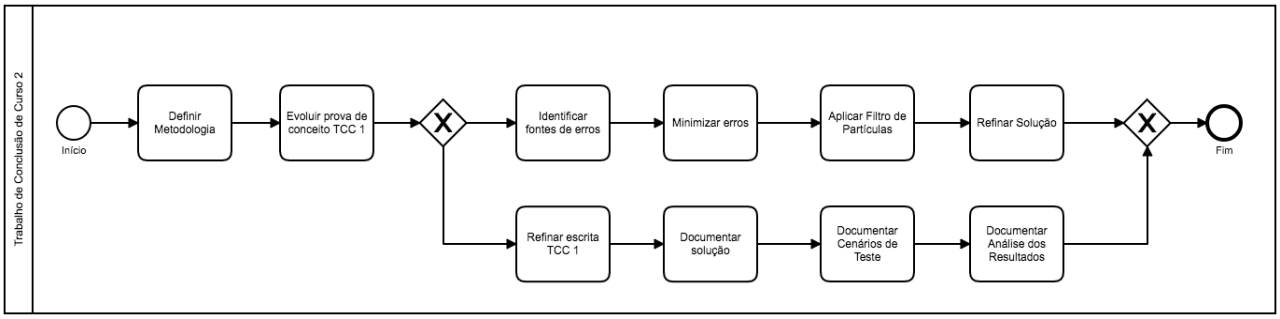

Figura 9 – Processo de Desenvolvimento TCC 2

- **Definir Metodologia:**

  Esta atividade teve como objetivo definir a maneira com que a pesquisa foi feita. Foi definido durante esta atividade, por exemplo, que a segunda estapa do trabalho seria dividida em *sprints* de 2 (duas) semanas cada.

- **Evoluir Prova de Conceito TCC 1:**

  Durante esta atividade, a prova de conceito desenvolvida durante a primeira etapa do trabalho foi evoluída, possibilitando que o robô navegasse no ambiente, identificando obstáculos. O principal objetivo desta atividade foi identificar características do kit Mindstorms NXT, da Lego, para que fosse possível apontar possíveis fontes de erros. Buscando, assim, a minimização da margem de erro da solução final.

- **Identificar Fontes de Erros:**

A partir da experiência obtida durante a evolução da prova de conceito, atividade anterior, foi possível identificar fontes específicas de erros, como a montagem do robô, por exemplo. O levantamento das fontes de erro estão descritas na seção **??**.

- **Refinar escrita TCC 1:**

  Esta atividade teve como objetivo corrigir detalhes e melhorar escrita do TCC 1, de acordo com o *feedback* da banca examinadora.

- **Minimizar Erros:**

  Durante esta atividade, foram feitos alguns testes que buscaram minimizar ao máximo os erros de navegação utilizando a prova de conceito evoluída.

- **Documentar Solução:**

  Durante esta atividade iniciou-se o processo de documentação de toda a solução, assim como da Metodologia seguida. Esta atividade seguiu em paralelo durante todo o período de desenvolvimento da solução.

- **Aplicar Filtro de Partículas:**

  Durante esta atividade o filtro de partículas começou a ser analisado mais a fundo, buscando maneiras de aplicá-lo. Uma primeira versão foi gerada, porém ainda sem resultados concretos.

- **Refinar Solução:**

  Durante esta atividade a solução foi refinada, o Filtro de Partículas passou a convergir da maneira esperada, como apresentado na seção **??**.

- **Documentar Cenários de Teste:**

  Durante esta atividade os cenários de teste passaram a ser documentados, registrando os resultados de diferentes situações de utilização do Filtro de Partículas, em diferentes ambientes e posições.

- **Documentar Análise dos Resultados:**

  Durante esta atividade os resultados foram documentados. Registrou-se informações relevantes que foram obtidas ao longo da pesquisa, como fontes de erro, margem de erro e quantidade de partículas necessárias, por exemplo.

4.2.1 Revisão Sistemática

Durante décadas, as pesquisas se encontravam carentes de métodos científicos que detalhassem o processo de revisão de literatura, como mostra [Kitchenham et al. 2006]. Esta carência acaba por impossibilitar a realização futura da mesma busca, já que sem

detalhamento da pesquisa, o interessado não poderá aplicar a busca, realizando comparações, por exemplo. Desse modo, entre as décadas de 1970 e 1980, psicólogos e cientistas sociais buscaram definir métodos de sistematizar revisões de litetarura, as chamadas *revisões sistemáticas*.

Com o objetivo de realizar uma ampla pesquisa bibliográfica, buscando conhecer diferentes técnicas de auto-localização e, mais especificamente, o problema de SLAM, utilizou-se da técnica de revisão sistemática. Uma revisão sistemática busca identificar e analisar o máximo de pesquisas relacionadas a um tema em específico, como define [Moraes e Souza 2011, p. 8]:

> *"Uma revisão literária sistemática é um meio de identificar, avaliar e interpretar todas as pesquisas disponíveis relevantes a uma determinada questão de pesquisa, ou área de um tópico, ou fenômeno de interesse. Estudos individuais que contribuem para uma revisão sistemática são chamados estudos primários; uma revisão sistemática é uma forma de estudo secundário."*

Como afirma [Moraes e Souza 2011], a técnica de revisão sistemática é bastante utilizada quando se busca identificar soluções propostas para resolver o problema levantado. Neste trabalho, a revisão sistemática busca identificar técnicas de resolução do problema de SLAM em diferentes contextos, inclusive educacional.

4.2.1.1 Processo de Revisão Sistemática

O processo utilizado durante a pesquisa segue o processo de revisão sistemática apresentado por [Kitchenham et al. 2006]. Desse modo, o processo é dividido em três etapas: *planejamento da revisão*, *condução da revisão* e a *documentação da revisão*. As atividades que estão distribuídas entre essas etapas podem ser observadas na Figura 10.

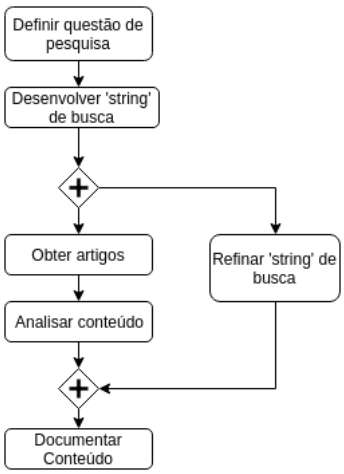

Figura 10 – Processo de Revisão Sistemática. Processo adaptado de [Kitchenham et al. 2006]

A descrição de cada atividade do processo da Figura 10 encontra-se na Tabela 4.

Tabela 4 – Processo de Revisão Sistemática

Atividade	Descrição
Definir questão de pesquisa	Atividade relacionada à definição do objetivo final da pesquisa, o que o pesquisador deseja alcançar com a mesma. A definição da questão de pesquisa possibilita o estabelecimento de critérios de busca durante os ciclos de revisão.
Desenvolver *string* de busca	Atividade de desenvolvimento. Nessa atividade, o objetivo é definir uma *string* de busca que represente todo o trabalho pesquisado. A *string* será refinada ao longo de toda pesquisa, buscando maximizar a efetividade das buscas.
Obter artigos	A partir da definição de um critério de busca, os artigos que se enquadram neste critério são selecionados para análise. A obtenção deve ser realizada em diferentes bases científicas, como IEEE, Springer e CAPES.
Analisar conteúdo	A análise do conteúdo busca, além de estudar o tema pesquisado, conhecer novas pesquisas e, com isso, refinar a *string* de busca.
Refinar a *string* de busca	Esta atividade é considerada uma das atividades mais importantes do processo de revisão sistemática. Busca maximizar a efetividade do processo de busca, identificando artigos importantes para a pesquisa.
Documentar conteúdo	A documentação do conteúdo, basicamente, é definida como o registro das informações necessárias à pesquisa, obtidas com os ciclos de pesquisa. Os resultados da revisão sistemática são frutos desta atividade.

Todo o processo de revisão sistemática é visto como um resultado parcial do trabalho, obtido durante a primeira etapa do mesmo. Dess modo, o detalhamento do desenvolvimento da revisão sistemática, como os ciclos de busca e o processo de refinamento da *string*, consta no apêndice A.

4.3 Considerações Parciais

Devido a necessidade principal deste trablho, que é identificar, analisar, adaptar e implementar técnicas para soluciona o problema de SLAM em um contexto simplificado, buscou-se definir uma metodologia de pesquisa que possibilitasse uma ampla visão do contexto, com a identificação de diversas soluções em diferentes situações. Desse modo, optou-se pela utilização da técnica de revisão sistemática, em um primeiro momento, para levantamento e análise de diversas técnicas utilizadas atualmente pela comunidade.

Com o objetivo de complementar o conhecimento obtido com a revisão sistemática, esta pesquisa de natureza *aplicada* e objetivos *exploratórios* garante uma visão prática importante, possibilitando a análise *qualitativa* de diferentes soluções ao longo da implementação da segunda etapa deste trabalho.

5 Simple SLAM: Auto-localização Simplificada

Neste capítulo, são apresentados todos os componentes presentes no *Simple SLAM*, incluindo a etapa de montagem do robô, a configuração do ambiente de programação e do ambiente de navegação do robô, buscando padronizar ao máximo as características da pesquisa para que a mesma possa ser aplicada e analisada por outro pesquisador.

5.1 Contextualização

O principal incentivo do pesquisador em relação a esta proposta de trabalho faz referência às metodologias de ensino de matemática, física e programação, tanto no âmbito da graduação, quanto nos ensinos Básico, Fundamental e Médio. Assim como já foi apresentado durante o trabalho, mais especificamente na seção 2.3, as metodologias de ensino geralmente aplicadas nos Centros Educacionais possuem uma característica de ensino passiva e ultrapassada. Desse modo, buscando garantir maior interesse dos alunos nos conteúdos apresentados, a Robótica Educacional é vista como uma ferramenta eficiente de ensino, como apresentam [Galvan et al. 2006], [Nunes e Santos 2013] e [Benitti et al. 2009].

Com isto em mente, o *Simple SLAM* busca apresentar diversas técnicas de auto-localização que podem ser utilizadas como uma atividade Educacional, em um contexto de ensino. Além disso, o *Simple SLAM* tem como objetivo a implementação de técnicas de auto-localização utilizadas no alto nível da robótica mundial em um contexto simplificado, ou seja, em um cenário tipicamente associado à Robótica Educacional. Nesse cenário, estão presentes a falta de recursos, o uso de kits mais padronizados e até mesmo limitados em termos de recursos e/ou de sensores e atuadores. No caso desse trabalho, foram utilizados equipamentos e recursos disponíveis no kit de Robótica da LEGO - NXT.

Em relação as técnicas de alto nível da Robótica Móvel, esta pesquisa busca implementar e analisar a técnica de Localização a partir da utilização do Filtro de Partículas, ou Filtro de Monte Carlo. Como foi apresentado na seção 2.2.2, o mesmo envolve a manipulação de técnicas probabilísticas com o intuito de obter a localização atual do robô.

Durante a primeira etapa desta pesquisa, foram analisadas algumas das técnicas mais utilizadas no contexto mundial da robótica móvel, como o Filtro de Kalman e o Filtro de Partículas. Estas duas técnicas possuem como objetivo possibilitar a análise da posição atual do robô a partir do uso da probabilidade. Durante a realização da Revisão Sistemática, documentada na seção A, foi observado que a técnica do Filtro de Kalman se encontra como a técnica probabilística mais utilizada no contexto mundial da robótica

móvel, estudada e difundida desde a década de 60, com exemplos de aplicações nos mais diversos contextos, geralmente ligados a sistemas de controle.

Já o Filtro de Partículas, de acordo com [Bukhori, Ismail e Namerikawa 2015], é uma técnica que, apesar de ter surgido há muitos anos, com pesquisas relacionadas à mesma durante a Segunda Guerra Mundial, só vem se tornando um foco maior de pesquisas e análises relacionadas a Robótica Móvel ao longo da última década. Desse modo, como o intuito deste trabalho é pesquisar e analisar a utilização de técnicas de auto-localização em um contexto limitado da robótica movel, optou-se pela utilização da técnica do Filtro de Partículas, uma técnica consideravelmente nova no contexto da robótica mundial.

Além da importância da realização de pesquisas relacionadas a técnicas que vêm buscando espaço na comunidade de robótica, a escolha da técnica dá-se, ainda, pela possibilidade de resolução do problema do *sequestro do robô*, apresentado por [Bukhori, Ismail e Namerikawa 2015], o qual pode ser solucionado ao se utilizar a técnica do Filtro de Partículas, diferentemente da utilização do Filtro de Kalman, como foi descrito na seção 2.2.2.

5.2 Arquitetura do Robô

De acordo com [Vieira 2005], a arquitetura de robôs móveis pode ser sub-dividida em cinco camadas: *percepção*, *decisão*, *planejamento de caminho*, *geração de trajetória* e *sistema de controle*.

A primeira camada, denominada *camada de percepção*, foi o grande foco deste trabalho, onde a identificação de obstáculos como pontos de referência é uma atividade essencial para a possibilidade de auto-localização, utilizando como base o Filtro de Partículas, por exemplo. Esta camada é responsável por adquirir informações sobre o ambiente ao seu redor, viabilizando a navegação e auto-localização no mesmo.

Já a segunda camada, *decisão*, que foi o foco de trabalho de [Ramalho 2015], tem como responsabilidade processar decisões do robô. Nesta camada, se encontra o verdadeiro cérebro do robo, como afirma [Vieira 2005].

Na terceira camada, *planejamento de caminho*, será utilizado, como ferramenta de apoio, o *framework* Traveller, desenvolvido por [Rincon 2015]. Sua utilização se refere ao planejamento da navegação no ambiente após o mapeamento, mesmo que parcial, do ambiente. Ou seja, enquanto o robô navega e mapeia o ambiente, os locais já percorridos (conhecidos e mapeados) possibilitarão a utilização do *framework*. Por outro lado, em locais ainda desconhecidos, o planejamento do caminho será feito de forma aleatória, buscando mapear o ambiente como um todo.

A quarta camada, *geração da trajetória*, utiliza o planejamento realizado na camada anterior para definir quais açoes devem ser realizadas no *hardware* do robô, levando em consideração as características físicas do mesmo. Com isso, a quinta camada, *sistema de controle*, é chamada para verificar o recebimento adequado dos sinais, assim como sua execução nos atuadores.

A Figura 11 apresenta as cinco camadas, nas quais, em seu nível mais alto, encontra-se a camada de percepção, que é o foco deste trabalho.

Figura 11 – Arquitetura do Robô. Fonte [Vieira 2005].

5.3 Montagem do Robô

Esta seção tem como objetivo apresentar a forma de montagem do robô utilizada durante a pesquisa. O padrão utilizado foi escolhido após a análise e comparação da margem de erro presente em dois tipos de montagem. Durante a primeira etapa deste trabalho (TCC 1), foi utilizado o robô montado com esteiras para movimentação, seguindo o padrão encontrado em tanques de guerra, como pode ser observado na Figura 12.

Figura 12 – Montagem do Robô Durante o TCC 1

Entretanto, após algumas análises e pesquisas, concluiu-se que a utilização de esteiras maximiza a margem de erro durante a navegação, prejudicando a qualidade do movimento. Isto se dá, segundo [Bagnall 2011], devido a área de contato entre a esteira e o chão. Quanto maior a área de contato, mais crítico é o resultado de derrapagens durante a movimentação, inviabilizando sua utilização para este objetivo. Desse modo, a partir da segunda etapa desta pesquisa, passou-se a utilizar um robô do tipo *Carpet* (de acordo com a nomenclatura da LEGO), no qual estão presentes duas rodas motorizadas e uma terceira roda para equilíbrio do robô.

Além do padrão relacionado às rodas, deve-se atentar a localização do sensor de distância. Este deve estar localizado exatamente no centro do robô, ou o mais próximo desse ponto. As Figuras 13 e 14 apresentam o robô utilizado durante a segunda etapa desta pesquisa.

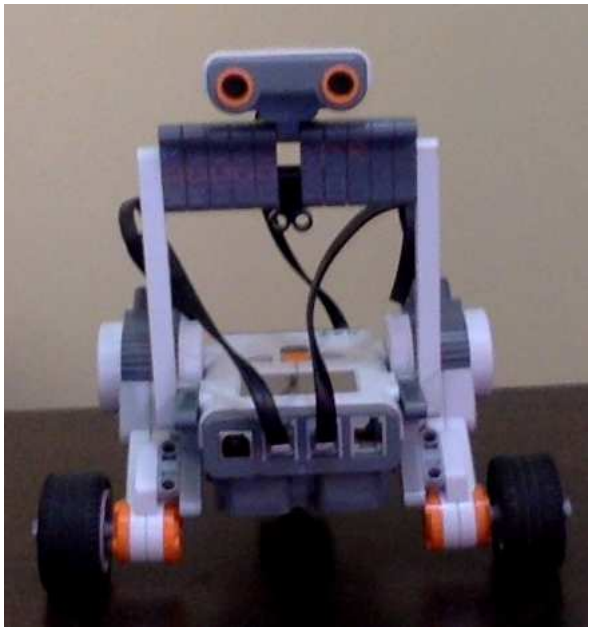

Figura 13 – Montagem do Robô Durante o TCC 2 - Frente

A terceira roda, utilizada apenas como roda de apoio, é feita a partir de uma simples montagem de peças LEGO, como pode ser observado na Figura 14.

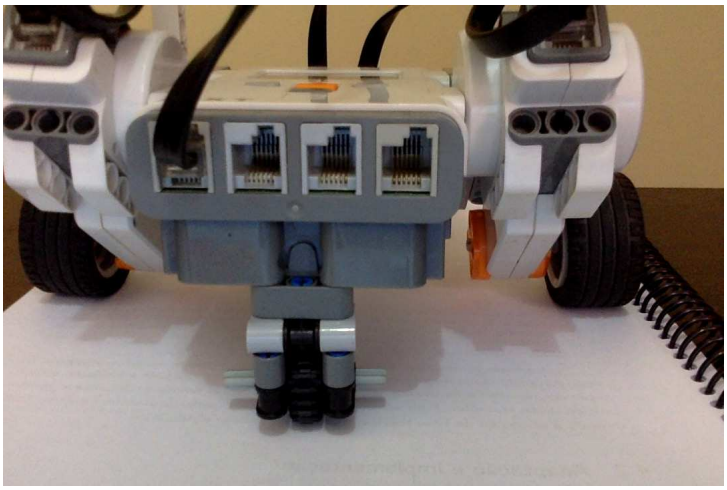

Figura 14 – Montagem do Robô Durante o TCC 2 - Roda de Apoio

Em relação ao código utilizado durante a pesquisa, as portas utilizadas para acesso aos motores e sensores seguem o padrão apresentado na Tabela 5.

Tabela 5 – Configuração dos Componentes Utilizados

Porta	**Componente**
B	Motor esquerdo
C	Motor direito
S1	Sonar

5.4 Ambiente de Navegação

Esta seção tem como objetivo apresentar o ambiente de navegação utilizado durante a realização da pesquisa. Ao longo da mesma, diversos ambientes foram utilizados, possibilitando a escolha da melhor configuração para análise efetiva dos resultados.

Sabe-se que, independente do piso selecionado, erros estarão presentes, seja devido a deslizes entre as rodas e o piso, ou devido a margem de erro presente nos sensores odométricos. Desse modo, buscou-se selecionar um piso que possibilite uma margem de erro mínima, dentre as opções presentes em um contexto simplificado, de uma escola, por exemplo.

Deve-se optar sempre por pisos lisos, ou seja, sem irregularidades, as quais podem 'travar' a roda de apoio, que não possui um formato tão adequado quanto as rodas motorizadas. De acordo com análises empíricas durante a realização da pesqisa, identificou-se que este formato diferenciado da roda de apoio, o qual pode ser visualizado na Figura 14, pode gerar diversas interferências durante a navegação, principalmente em pisos que possuem 'rachaduras', como as presentes em pisos feitos com azulejo, por exemplo.

Devido a esta característica da roda de apoio, pisos feitos com azulejo devem ser evitados, optando sempre por pisos tão lisos quanto possível. Um exemplo do piso utilizado

durante esta pesquisa pode ser visualizado na Figura 15.

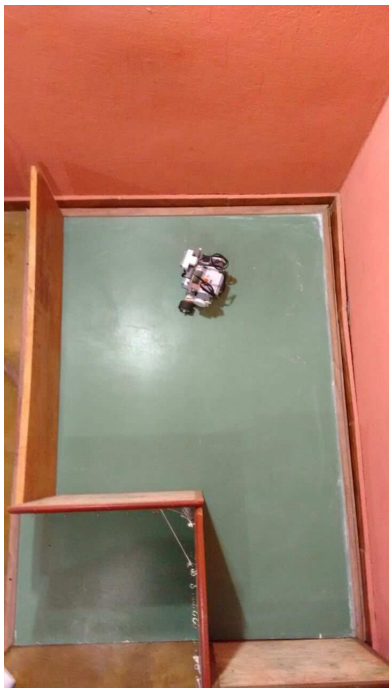

Figura 15 – Exemplo de Piso.

Com o piso definido, deve-se definir o padrão de obstáculos no ambiente. Como o objetivo desta pesquisa é analisar a viabilidade da utilização de técnicas de auto-localização e mapeamento de ambientes em robôs simples, utilizando apenas um sensor de distância do tipo *sonar*, deve-se adequar o padrão de obstáculos no ambiente, buscando minimizar a margem de erro presente na análise.

Para que o robô seja capaz de se localizar no ambiente, o mesmo precisa identificar características únicas no mesmo. Isto é feito utilizando os obstáculos presentes no ambiente. Desse modo, a presença de obstáculos no mapa é essencial para a viabilidade do processo de auto-localização. Para que a identificação dos obstáculos como *pontos de referência* seja possível, os mesmos precisam se adequar ao máximo às características presentes nos sonares. A principal característica do sonar utilizado é apresentada na Figura 35, a qual envolve o modelo de emissão do sinal utilizado pelo sonar, em formato de cone.

Para que um sonar funcione, o mesmo emite um sinal que irá refletir no primeiro obstáculo encontrado, retornando ao robô e, a partir da análise do tempo gasto para percorrer este caminho, chega-se a conclusão da distância entre o robô e o objeto que refletiu o sinal. Porém, este sinal não é emitido de forma unidimensional, sua emissão é feita no formato de cone, devido as características físicas relacionadas à tramissão do som no ar. Por esse motivo, alguns erros podem ser adicionados à solução, visto que em determinadas situações, o objeto que refletiu o sinal sonoro não se encontra exatamente à frente do robô.

Isto significa que o robô pode enxergar obstáculos mesmo quando não existe ne-

nhum obstáculo diretamente à sua frente, enxergando obstáculos presentes nas laterais e tratando-os como obstáculos frontais. Infelizmente, não foi possível encontrar uma solução exata para esse problema. Isso se deve ao fato de usar um sonar para detecção de distâncias, com o qual é intrínseco esse tipo de problema. Desse modo, buscando minimizar a taxa de ocorrência desse *'engano'*, o ambiente foi pensado e criado utilizando paredes lisas e em formatos uniformes, retilíneos.

Como o objetivo da pesquisa é analisar a possibilidade da auto-localização dentro de um ambiente, optou-se pela não utilização de obstáculos entre as paredes. Fazendo com que o ambiente seja composto apenas de piso e paredes, como apresenta a Figura 16.

Como o robô deverá analisar os pontos de referência (paredes), a partir de determinadas características, e se auto-localizar em relação ao mapa, o ambiente utilizado não pode ser simétrico. Dessa forma, deve-se evitar a utilização de ambientes quadrados ou retangulares, por exemplo, já que o robô não conseguiria identificar características únicas em cada local do ambiente. A Figura 16 apresenta uma boa opção de ambiente a ser utilizado, já que o mesmo possui características que viabilizam a comparação e conclusão de unicidade.

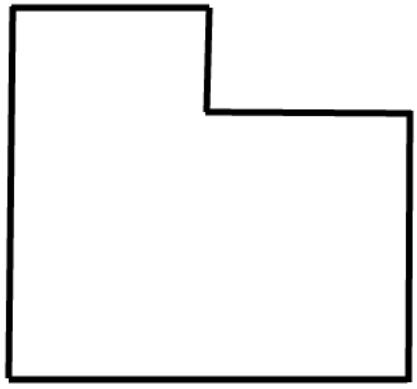

Figura 16 – Exemplo de Ambiente.

5.5 Configuração e Integração das Tecnologias

Nesta seção, será apresentado o passo a passo para configuração e integração das tecnologias utilizadas durante este trabalho. A principal tecnologia utilizada é o pacote leJOS (Lego Java Operating System) NXJ, o qual possibilita a utilização da linguagem Java para implementação dos algoritmos durante a realização do trabalho. A tecnologia leJOS NXJ envolve um ambiente Java, um *firmware* e uma API, como descrito na seção 3.2.3.

O ambiente java disponibilizado pelo leJOS NXJ possibilita integração com a IDE Eclipse, facilitando o *upload* de códigos, atualização de *firmware* e utilização da API.

O *firmware* é necessário, pois cria uma máquina virtual Java no NXT, possibilitando a execução de códigos Java no *Brick*. Além disso, a API leJOS envolve uma variedade de classes e métodos com algoritmos já produzidos e disponibilizados para utilização durante o desenvolvimento. Com a utilização da API, o desenvolvedor não se preocupa com a implementação de algoritmos comuns, como de navegação e obtenção de dados dos sensores.

A plataforma leJOS NXJ é multiplataforma, possibilitando sua utilização no *Windows*, *Linux* e *MAC OS*. Nas seções a seguir apresentadas, são descritos os passos para instalação e configuração do leJOS NXJ nas três plataformas.

5.5.1 Instalação no Windows

Primeiramente, para possibilitar a comunicação entre o PC e o NXT, deve-se instalar um *USB Driver*, o qual pode ser encontrado aqui, optando pela versão de seu Brick (no caso deste trabalho, o NXT). Após a instalação do USB Driver, deve-se instalar o JDK (Java Development Kit) versão 1.8, disponivel aqui. Vale ressaltar que os *hiperlinks* presentes neste capítulo, assim como em todo o texto, foram verificados durante o primeiro semestre de 2017, os quais são passíveis de alterações pelos responsáveis, resultando, futuramente, em possíveis *links* indisponíveis.

Após a instalação do JDK, devem ser especificadas/configuradas as variáveis de ambiente do Java, adicionando o caminho do diretório do JDK instalado no *PATH*. Neste sentido, devem ser seguidos os passos:

1. Clicar em iniciar
2. Ir em Painel de Controle
3. Ir em Sistema
4. Entrar em Configurações Avançadas
5. Clicar na aba "Avançado"
6. Ir em Variáveis de Ambiente
7. Procurar a variável JAVA_HOME (Caso não exista esta variável, criar uma)
8. Editar a variável PATH e adicionar o seguinte ao valor da mesma: $\%java\_home\%\backslash bin$

Uma vez realizados esses passos, basta testar se o Java está funcionando. Para isso, basta abrir o *Command Prompt* (CMD) e exeutar o comando *java*. Caso esteja funcionando, serão apresentadas algumas opções de utilização do comando *java*.

Uma vez o Java já instalado e funcionando, basta instalar o NXJ, o qual se encontra aqui. Baixar o arquivo *Setup.exe*, executar o arquivo e seguir os comandos de instalação apresentados pelo próprio instalador. Ao final da instalação, será executado um programa para *upload* de um novo *firmware* para o NXT. Uma vez realizado tal procedimento, o leJOS, provavelmente, estará instalado e em funcionamento.

5.5.2 Instalação no Linux

Seguindo a mesma lógica de instalação apresentada na seção anterior, primeiramente deve-se instalar o USB Driver e o JDK. Por motivo de organização, são apresentados, primeiramente, os passos de instalação do JDK, seguido do USB Driver.

Para instalar o JDK, realizar o *download* da última versão do JDK, disponível aqui. Instalar o JDK, de acordo com sua distribuição linux, e editar o PATH, adicionando o caminho do /bin do JDK instalado utilizando a variável $\%java\_home\%\backslash\ bin$. Para verificar se a instalação e a configuração foram feitas corretamente, executar o comando *java* no terminal e observar se informações de utilização do comando são apresentadas.

Visando a preparação e a instalação do USB Driver, considerar os seguintes passos:

1. Instalar a lib *libusb*, disponível aqui, para que as ferramentas do leJOS possam acessar as portas USB do PC.

2. É preciso reparar se há acesso de leitura e de escrita no dispositivo NXT USB. Para isso, verificar o arquivo /dev/bus/usb (especificidades em cada distribuição linux).

3. Usar regras Udev. Para isto, deve-se criar um arquivo chamado */etc/udev/rules.d/70-lego.rules* e adicionar o seguinte conteúdo no mesmo:

```
1  # Lego NXT brick in normal mode
2  SUBSYSTEM=="usb", DRIVER=="usb",
3  ATTRS{idVendor} == "0694",
4  ATTRS{idProduct} == "0002", GROUP="lego",
5  MODE = "0660"
6
7  #Lego NXT brick in firmware update mode (Atmel
       SAM-BA mode)
8
9  SUBSYSTEM == "usb", DRIVER == "usb",
10 ATTRS{idVendor} == "03eb",
11 ATTRS{idProduct} == "6124", GROUP="lego",
12 MODE = "0660"
```

Realizado tal procedimento, basta instalar o leJOS NXJ, de acordo com os passos a seguir:

1. Baixar o arquivo *tar.gz* aqui.
2. Descompactar o arquivo no diretório */opt/lejos_nxj/*.
3. Criar a variável de ambiente *NXJ_HOME* apontando para o diretório no qual foi descompactado o leJOS.
4. Adicionar o diretório */bin* da variável *NXJ_HOME* no *PATH*.
5. Caso sejam percebidos problemas com permissões, devem ser alteradas as permissões de execução neste diretório.
6. O *PATH* deve conter também o caminho para o binário *ant*.
7. Agora, basta gerar a distribuição. Trocar para o diretório de criação e rode *ant*.

Feito isto, basta atualizar o *firmware* no NXT.

5.5.3 Instalação no MAC OS

Primeiramente, deve-se instalar o JDK no sistema, fazendo o download do mesmo aqui. Instalar o JDK e editar o PATH, adicionando o caminho do /bin do JDK instalado utilizando a variável $\%java\_home\%\backslash bin$. Para verificar se a instalação e a configuração foram feitas corretamente, executar o comando *java* no terminal e observar se informações de utilização do comando são apresentadas.

Para instalar o leJOS no MAC OS, deve-se primeiro configurar o ambiente, de acordo com os seguintes passos:

1. Instalar o Software LEGO, o qual irá instalar também os Drivers USB.
2. Baixar o instalador aqui.
3. Extrair os arquivos para uma nova localização, como */Applications/lejos_nxj*.
4. Caso seja utilizado o login de administrador, é necessário criar (ou editar) o arquivo *.tcshrc* no diretório *home*.
5. Adicionar as seguintes linhas no diretório, no qual o leJOS foi extraído:

```
1 setenv NXJ_HOME /Applications/lejos_nxj
2 setenv PATH ${PATH}:${NXJ_HOME}/bin
```

6. Salvar o arquivo em */users/administrator*.

7. Abrir um terminal e executar tcsh, sendo possível visualizar o tcsh shell.

8. Executar *setenv* para ter certeza que as variáveis de ambiente estão configuradas corretamente.

5.5.4 Utilizando leJOS NXJ

Após a instalação do leJOS, apresentada nas seções anteriores, tudo está pronto para desenvolvimento e utilização das ferramentas presentes no pacote. Esta seção tem como objetivo apresentar as formas de utilização do pacote.

O USB é a única forma de *upar* o *firmware* para o NXT. Para isto, pode-se utilizar o modo gráfico, o modo por linha de comando do terminal ou até o modo de integração com a IDE Eclipse. Para fazê-lo, os seguinte passos devem ser seguidos:

1. Conectar seu NXT ao PC via USB.

2. Executar o arquivo *nxj-flashg*, presente no diretório */bin* do leJOS para utilizar o modo gráfico, ou apenas executar *nxjflash*.

3. No modo gráfico, clicar em *upload firmware* e aguardar alguns instantes até que o processo seja concluído. Não retirar o NXT durante o *upload* do *firmware*.

Feito isso, o NXT estará, provavelmente, pronto para executar código Java.

5.5.4.1 Conflitos de Tecnologias e Soluções

A tecnologia leJOS é utilizada, em sua grande maioria, por pesquisadores e estudantes interessandos no contexto de Robótica. Desse modo, interessados na tecnologia podem ainda utilizar e pesquisar conceitos em outras tecnologias, como o Arduino, por exemplo.

Desse modo, esta seção tem como objetivo apresentar um possível problema encontrado durante a utilização do leJOS por pesquisadores e/ou estudantes que também costumam utilizar a tecnologia Arduino. Os *drivers* utilizados para acesso do NXT por parte do computador podem conflitar com os *drivers* utilizados para o mesmo propósito na tecnologia Arduino, principalmente ao utilizar o Sistema Operacional Windows. Este conflito pode gerar falhas na comunicação, impossibilitando o *upload* de *firmware* e códigos do computador ao NXT.

Para solucionar este problema, pode-se simplesmente desinstalar o *software* do Arduino ou, caso deseje trabalhar com as duas tecnologias, devem ser realizados os seguintes passos:

1. Desconectar o computador da Internet;
2. Abrir o gerenciador de dispositivos (Botão direito em "Meu Computador");
3. Conectar o NXT via USB (O gerenciador de dispositivos será atualizado);
4. Ir até Portas e selecionar **Bossman**;
5. Clicar em "Atualizar Driver"e selecionar a opção de "Atualização Manual";
6. Selecionar o Driver da Lego, dentre as opções disponíveis;
7. Feito isto, basta realizar o *upload* do *firmware* ou código ao NXT.

5.5.5 Utilizando IDE Eclipse

É possível trabalhar com a API leJOS e suas ferramentas utilizando apenas um editor de texto e comandos via terminal. Entretanto, o trabalho pode se tornar cansativo e desgastante, já que a implementação se torna lenta e ineficiente, assim como o processo de *debug*. Para facilitar este processo, pode-se utilizar a IDE (*Integrated Development Environment*) do Eclipse, possibilitando a implementação, compilação e *upload* de códigos e de *firmware* para o NXT.

Para utilizar a IDE Eclipse para a implementação dos códigos com a API leJOS, basta realizar os seguintes passos.

1. Fazer o download do Eclipse aqui;
2. Descompactar o arquivo no diretório permanente do Eclipse;
3. Executar o arquivo eclipse;
4. Com o Eclipse aberto, selecionar Help > Install New Software;
5. Em frente a caixa de dialógo apresentada, selecione a opção Add;
6. Em Add, na opção Name, digitar "leJOS NXJ";
7. Na localização, digitar: "http://lejos.sourceforge.net/tools/eclipse/plugin/nxj/";
8. Clicar em Ok e selecionar o *checkbox* do plugin, clicando em Next em seguida;
9. Ler e aceitar os termos da licença, clicando em Next;
10. Por fim, o plugin, provavelmente, estará instalado. Portanto, basta reiniciar o Eclipse e utilizá-lo normalmente.

Durante a implementação, caso seja necessária ajuda sobre determinado método ou classe, basta consultar os arquivos de Ajuda do leJOS, que se encontram em Help > Help Contents no Eclipse, na seção leJOS NXJ.

5.6 Arquitetura da Solução

Nesta seção, será apresentada a arquitetura utilizada na solução desenvolvida durante esta pesquisa, a qual envolve a aplicação do Filtro de Partículas como uma forma de se auto-localizar no ambiente, partindo de um ponto desconhecido.

A solução geral é dividida em duas grandes áreas: (i) a obtenção de informações e execução de comandos, presentes no robô, executando localmente; e (ii) o processamento do Filtro de Partículas, o qual é realizado, remotamente, em um computador. A comunicação entre os dois projetos é feita com a tecnologia *bluetooth*, utilizando a ferramenta *bluecove*. As seções seguintes apresentam a arquitetura utilizada em cada uma dessas áreas, assim como algumas características de implementação.

5.6.1 Módulo Local - NXT

Para que o robô possa se localizar no ambiente, o mesmo precisa se locomover buscando identificar pontos de referência no ambiente. Como foi descrito na seção 2.2.2, o Filtro de Partículas utiliza a informação de distância de obstáculos em relação ao robô para identificação do local em que o mesmo se encontra. Desse modo, para que o robô identifique um local único, o mesmo precisa se locomover pelo ambiente de maneira que a margem de erro não inviabilize sua auto-localização.

Visto isso, buscou-se uma maneira de abstrair a navegação do robô, utilizando o conceito de Piloto, como apresenta [Bagnall 2011]. Este conceito faz referência a todas as informações e todos os processamentos referentes à navegação serem integrados em apenas um responsável, o Piloto. Este Piloto deve conhecer algumas informações e funções que garantem a abstração da navegação, como:

- **Distância entre Rodas:**

  É necessário o conhecimento da distância ente cada roda, pois a realização de curvas depende da rotação em cada roda e a distância entre as mesmas. Quanto maior a distância entre as rodas, maior deverá ser o número de voltas necessárias em cada roda para contemplar a curva desejada.

- **Diâmetro da Roda:**

Com a informação do diâmetro da roda é possível calcular a quantidade de giros necessários para contemplar determinadas distâncias, assim como determinadas curvas.

- **Controle de Cada Motor:**

  Com o Piloto possuindo controle de cada roda, o mesmo pode calcular e realizar a navegação de maneira integrada, abstraindo todo o processo de navegação. No caso da solução, foram utilizados os motores B e C, como esquerdo e direito, respectivamente.

- **Dados dos sensores:**

  O Piloto precisa do acesso aos dados dos sensores de distância (sonar) e odométricos para que possa calcular e executar trajetos durante a navegação no ambiente. Nesta solução, foi utilizado o sensor de distância (sonar) na porta S1, como foi descrito na seção 5.3.

Para implementação desta abstração foram utilizadas as classes *DifferentialPilot*, *RangeFinder* e *Navigator*, presentes na API leJOS. A integração destas classes segue o apresentado na Figura 17.

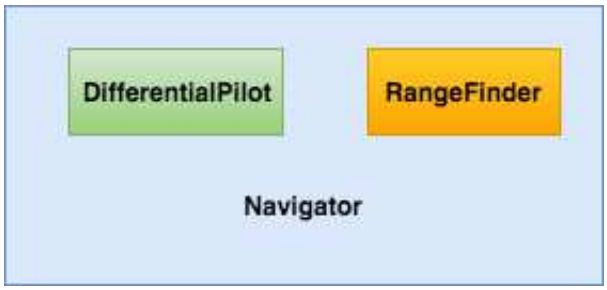

Figura 17 – Classes que contemplam o Piloto da solução.

5.6.2 Módulo Remoto - PC

A parte da solução que é executada remotamente envolve o processamento do Filtro de Partículas bem como a geração e o envio de comandos ao robô, seja para navegação ou obtenção de informações dos sensores. A solução possui um mecanismo de conexão *bluetooth* com dispositivos, utilizando a ferramenta *Bluecove*, bastando apenas informar o nome do dispositivo, que, no caso desta pesquisa, é **GTX**.

A forma de *input* desta informação é a partir da utilização do componente gráfico presente na API leJOS, o qual possibilita a apresentação do ambiente de navegação, as partículas e o resultado da filtragem, por exemplo, além da possibilidade do contato com o usuário por meio de *inputs* textuais e botões.

A Figura 18 apresenta um exemplo do componente gráfico utilizado para contato com o usuário durante esta pesquisa.

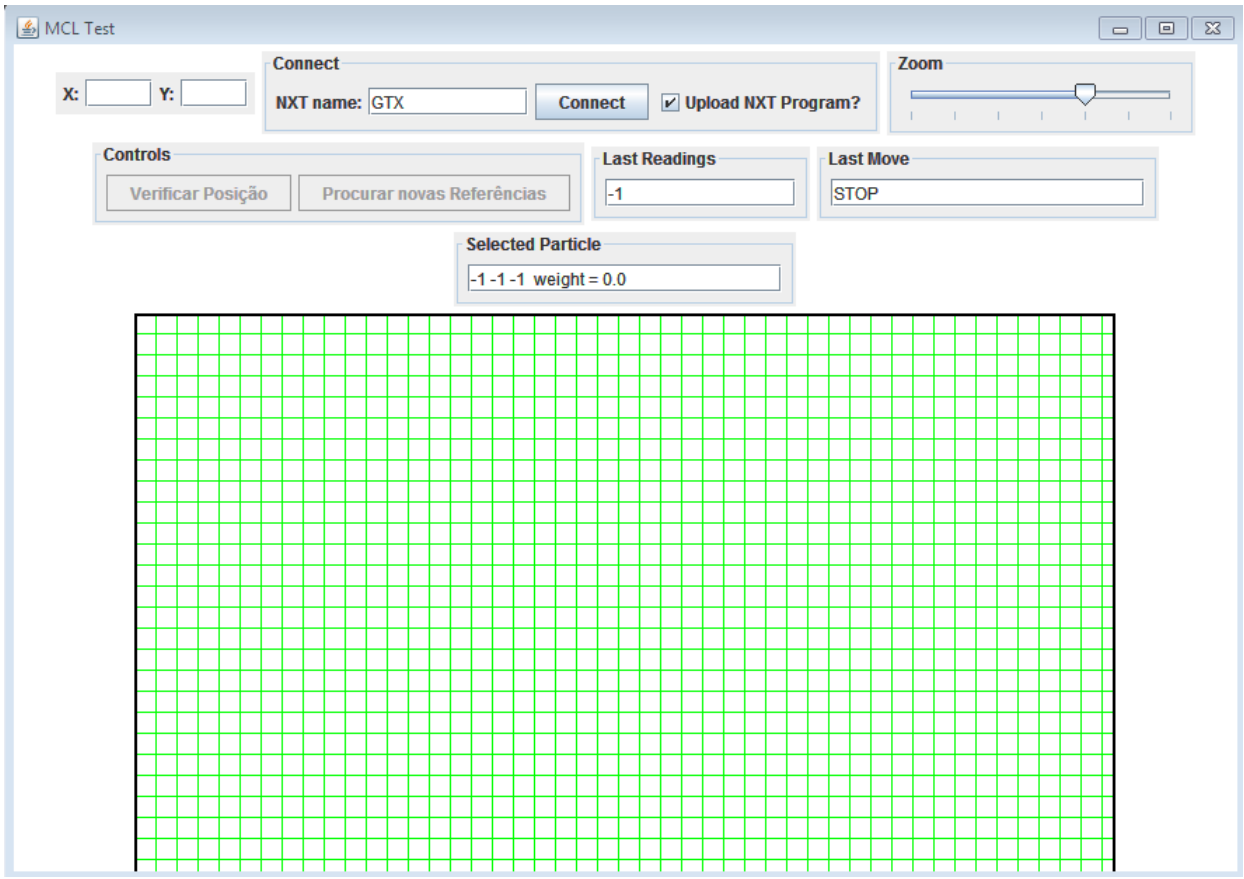

Figura 18 – Interface gráfica utilizada durante a pesquisa.

A arquitetura de comunicação entre os projetos e o usuário segue o padrão apresentado na Figura 19. O módulo *Localization* faz referência à parte da solução que é executada remotamente, em um computador. E o módulo *Navigator* é executado localmente, no NXT.

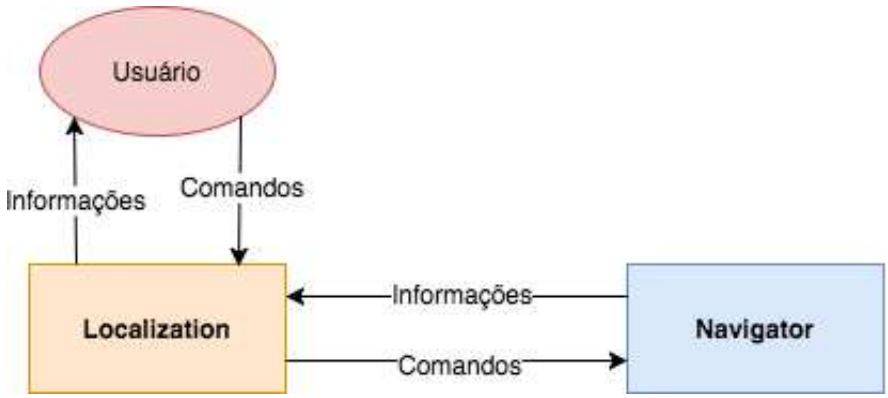

Figura 19 – Arquitetura de Comunicação entre projetos e usuário.

O Fluxo de informações entre os módulos está de acordo com o apresentado na

Figura 20, a qual apresenta os três módulos participantes, o PC, o usuário (pesquisador) e o robô.

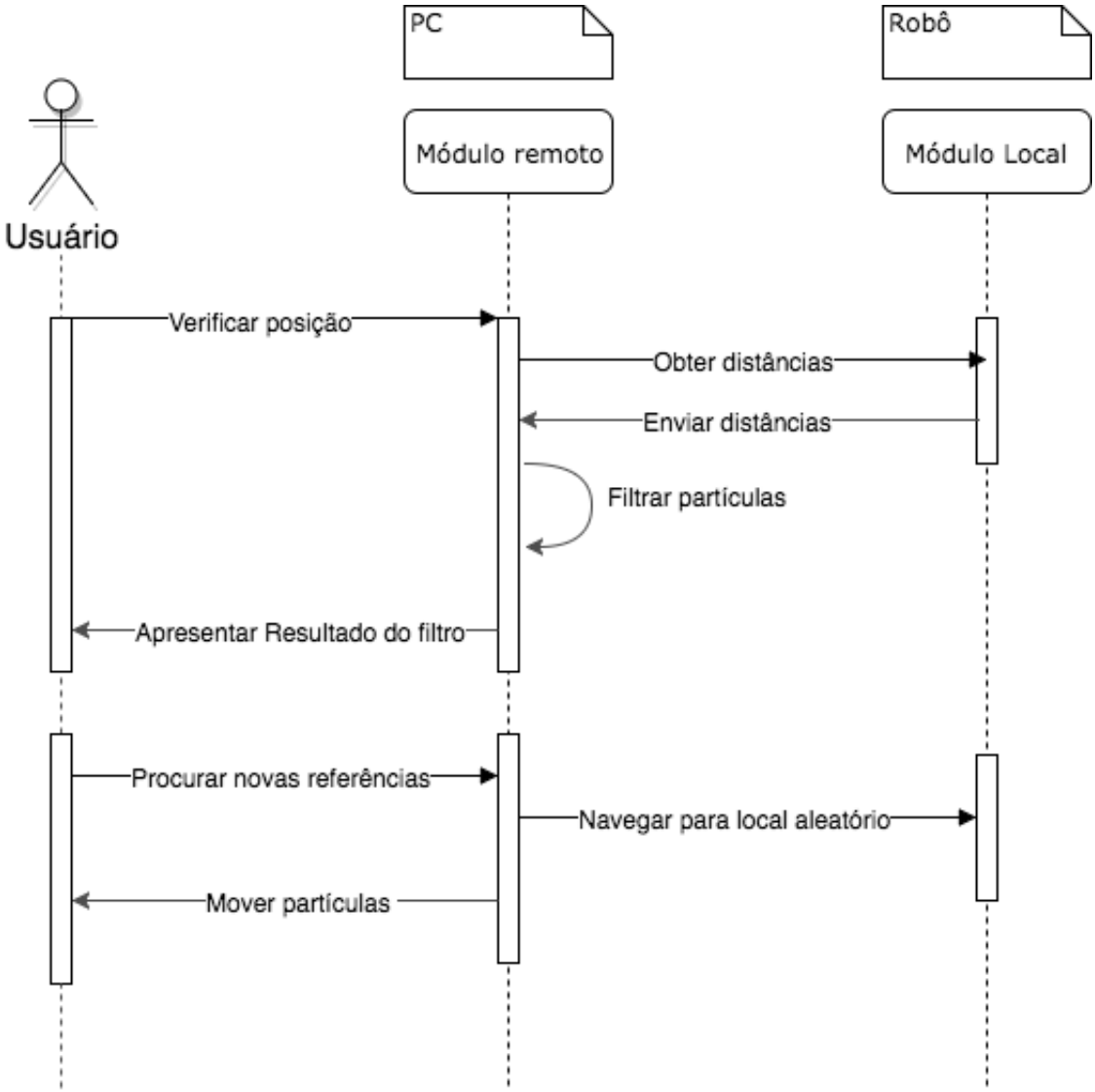

Figura 20 – Diagrama de Sequencia da Solução.

O processamento do Filtro de Partículas é executado utilizando métodos da API leJOS, de acordo com toda a teoria apresentada na seção 2.2.2. A geração das informações de distância de cada partícula necessita de um processamento bastante elevado, quando se tem como referência a capacidade de *hardware* dos robôs NXT. Desse modo, todo esse processamento é feito remotamente, executando em um computador. A quantidade de partículas utilizadas foi variável ao longo da pesquisa, buscando estudar o impacto desta variável durante a execução dos testes. Dessa forma, conduziu-se um refinamento baseado em estudos empíricos.

5.7 Considerações Parciais

Durante este capítulo, foram apresentadas as caracteríticas técnicas da solução desenvolvida durante esta pesquisa, assim como o ambiente utilizado para teste e o processo de configuração e instalação das ferramentas necessárias.

Esta solução foi desenvolvida e melhorada de forma que o principal objetivo foi analisar a viabilidade da utilização de uma técnica de auto-localização utilizando o Kit de Robótica Mindstorms da Lego. Por esse motivo, foram utilizadas apenas ferramentas presentes no Kit e em sua API, possibilitando a replicação do estudo e sua utilização em um ambiente Educacional por parte de qualquer interessado no assunto.

6 Análise dos Resultados

Este capítulo tem como objetivo discutir os cenários de teste utilizados durante a pesquisa, assim como os resultados obtidos durante a realização da mesma. Dentre estes resultados, estão presentes as fontes de erros levantadas com a experiência da pesquisa, suas respectivas correções, dentro do possível, e a análise da viabilidade da utilização do Filtro de Partículas no contexto simplificado da Robótica Educacional. Desse modo, este capítulo está dividido em três seções: 6.1, 6.2 e 6.3.

6.1 Cenários de Teste

O principal objetivo dos cenários de teste é apoiar a observação e a análise sistemática da técnica de auto-localização utilizada durante o trabalho, assim como as diferentes configurações do ambiente e do filtro utilizado.

Após a realização de todos os cenários de teste propostos, foi possível chegar a uma conclusão quanto à viabilidade da utilização da técnica do Filtro de Partículas em um contexto típico da Robótica Educacional, a qual será apresentada na seção 6.3.

Os cenários de teste fazem parte de 9 conjuntos, chamados de casos de teste, nas quais a solução foi analisada, variando entre elas o ambiente e a configuração do filtro. Foram executados 45 cenários de teste, distribuídos com 9 casos de teste. Os quais têm como objetivo analisar a exatidão da auto-localização obtida em cada contexto, reiterando que o objetivo da pesquisa é analisar a viabilidade da utilização de uma técnica de SLAM em um contexto típico da Robótica Educacional.

O Filtro de Partículas é uma das duas principais técnicas que buscam solucionar o problema de SLAM. Por este motivo, os cenários de teste desta pesquisa buscam analisar o Filtro de Partículas, sendo esse executado no contexto da Robótica Educacional. A intenção é viabilizar, em parte, a aplicação da solução de SLAM nesse contexto.

Os dados obtidos a partir de cada cenário de teste está organizado de acordo com a Tabela 6.

Tabela 6 – Organização dos Dados

Cenário	**Partículas**	**Rotação**	**Deslocamento**	**Ciclos**	**Precisão**

Onde,

- **Cenário** faz referência ao cenário de teste. Cada caso de teste possui 5 (cinco)

cenários, registrando a variabilidade dos resultados de acordo com a variabilidade dos parâmetros.

- **Partículas** informa a quantidade de partículas utilizada durante este teste;
- **Rotação** informa a velocidade de rotação do robô, em graus por segundo;
- **Deslocamento** informa a velocidade de deslocamento em linha reta, em centímetros por segundo;
- **Ciclos** apresenta o número de ciclos de filtragem que foram necessários para obtenção da localização;
- **Posição** informando qual a posição estimada do robô, informando a margem de erro da posição estimada, tanto no eixo X quanto no eixo Y. Basicamente, este campo informa de que ponto a que ponto nos eixos X e Y, o robô pode estar localizado, utilizando o centímetro como unidade, seguindo o padrão do restante da pesquisa.
- **Precisão** informando qual a precisão obtida no teste, classificada em *Ótima*, *Boa*, *Mediana*, *Baixa* e *Baixíssima*. A classificação da precisão é feita a partir da comparação entre a posição estimada e a posição real. Como a posição estimada possui uma margem de erro, destacada como uma elipse no local indicado, a comparação utiliza como referência esta elipse, assim como a partícula definida como sendo a que possui os dados mais próximos aos obtidos pelo robô real. A classificação segue a tabela 7.

Tabela 7 – Classificação da Precisão.

Precisão	**Descrição**
Ótima	Posição real se encontra a menos de 5 centímetros da partícula final.
Boa	Posição real se encontra a uma distância entre 6 e 10 centímetros da partícula real, se encontrando, ainda, dentro da elipse de erro.
Mediana	Posição real se encontra próxima as fronteiras da elipse de erro, com mais de 10 centímetros de distância da partícula final.
Baixa	Posição real se encontra fora da elipse de erro, porém a até 10 centímetros de distância da sua fronteira.
Baixíssima	Posição real se encontra fora da elipse de erro, a uma distância de mais de 10 centímetros da fronteira da elipse.

Durante os quatro primeiros e quatro últimos casos de teste, o ambiente utilizado foi o apresentado na Figura 21, na qual todos os valores se encontram em centímetros (cm). O quinto caso, 6.1.5, utilizou um ambiente sem características específicas, como o

apresentado na Figura 22, buscando analisar o comportamento do algorítmo em mapas quadrados ou retangulares.

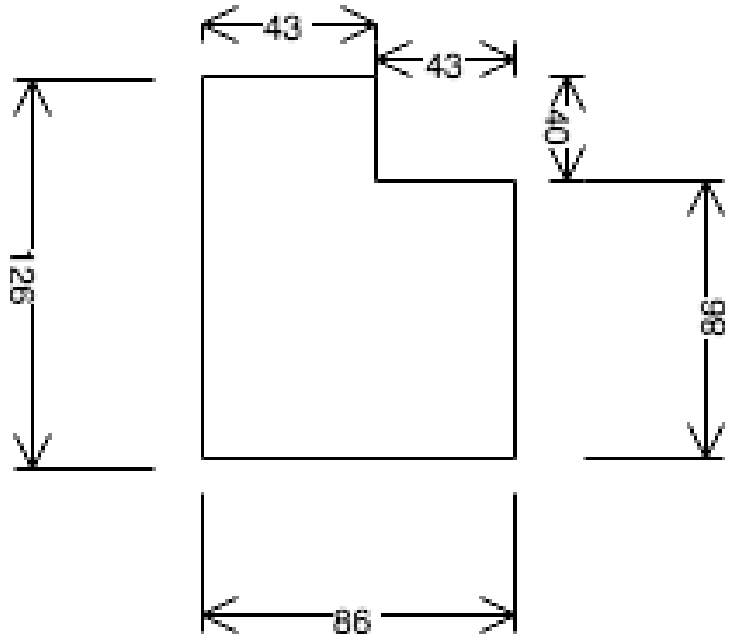

Figura 21 – Mapa Utilizado Durante Primeiro Caso de Teste.

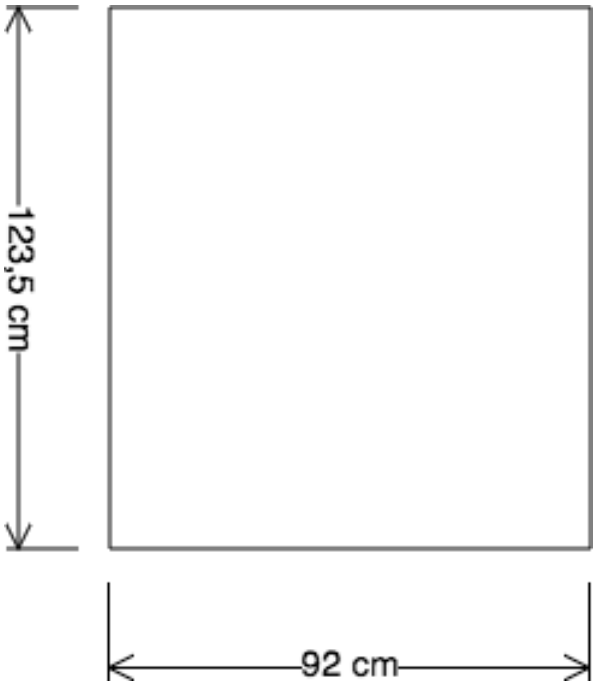

Figura 22 – Mapa Utilizado Durante Quinto Caso de Teste.

Cada cenário de teste tem como objetivo observar o comportamento do Filtro de Partículas, levando em consideração o erro obtido durante o processo de auto-localização. A variação entre cada cenário se dá pela alteração de três variáveis do processo de auto-localização:

- Quantidade de partículas:

  Como o contexto trabalhado envolve recursos de *hardware* limitados, o buscou-se selecionar uma quantidade relativamente baixa de partículas para execução do filtro, garantindo uma a ausência de grandes "furos"no mapa. A partir da execução do primeiro cenário de teste, com resultados disponíveis na Tabela 8, foi fixado o valor de 150 (cento e cinquenta) partículas distribuídas aleatoriamente pelo mapa.

  Esta escolha se deu a partir de análise empírica, utilizando como parâmetros comparativos o processamento necessário para execução do Filtro de Partículas, a uniformidade da distribuição das partículas pelo mapa, a quantidade de ciclos necessários para conclusão da posição atual do robô e o erro final do processo.

- Velocidade de Rotação

  A seção 6.2 apresenta como uma das principais fontes de erros o deslize entre a roda e o piso ao se movimentar. Desse modo, a escolha da velocidade se deu, principalmente, buscando minimizar o erro final. A velocidade de rotação foi fixada, a partir do cenário 6.1.3 no valor de 30 graus por segundo, a qual apresentou o menor erro final nos cenários anteriores.

- Velocidade de Deslocamento

  Assim como o parâmetro discutido anteriormente, este envolve a potencialização de uma fonte de erros para o processo de auto-localização, o deslize entre as rodas e o piso, como é apresentado na seção 6.2. O valor deste parâmetro não foi fixado durante os cenários com o mapa 21, foi apenas reutilizado o valor de 30 diâmetros de roda por segundo, o que significa forçar a maior velocidade do robô.

  Utilizar a maior velocidade possível se torna possível devido a estratégia utilizada para minimizar esta fonte de erro, como é apresentado na seção 6.2.

  Estas informações podem ser obtidas na seção 6.1.1 e no apendice C.

As seções 6.1.1, 6.1.2, 6.1.3, 6.1.4 e 6.1.5 apresentam a primeira fase de testes, na qual são executados os cenários variando cada um dos parâmetros até identificar a melhor configuração. A segunda fase de testes, registrada nas seções 6.1.6, 6.1.7, 6.1.8 e 6.1.9, apresenta a execução dos cenários também variando cada um dos parâmetros separadamente. Porém, na fase 2, os parâmetros fixos fazem referência ao melhor valor obtido durante a primeira fase de testes.

6.1.1 Caso de Teste 1

Os resultados obtidos com o caso de teste estão dispostos na Tabela 8. Deve-se atentar que o exemplo 3 possui um agravante, o robô teve sua roda direita presa na parede,

enquanto se locomovia. Desse modo, o robô perdeu-se e teve que se encontrar novamente, comportando-se da maneira esperada em um caso básico de "sequestro do robô".

Este caso de teste faz referência à variação da quantidade de partículas. Com o objetivo de apresentar a forma de registro dos casos de teste, o primeiro cenário do caso de teste 1 é apresentado nas Figuras 23 e 24.

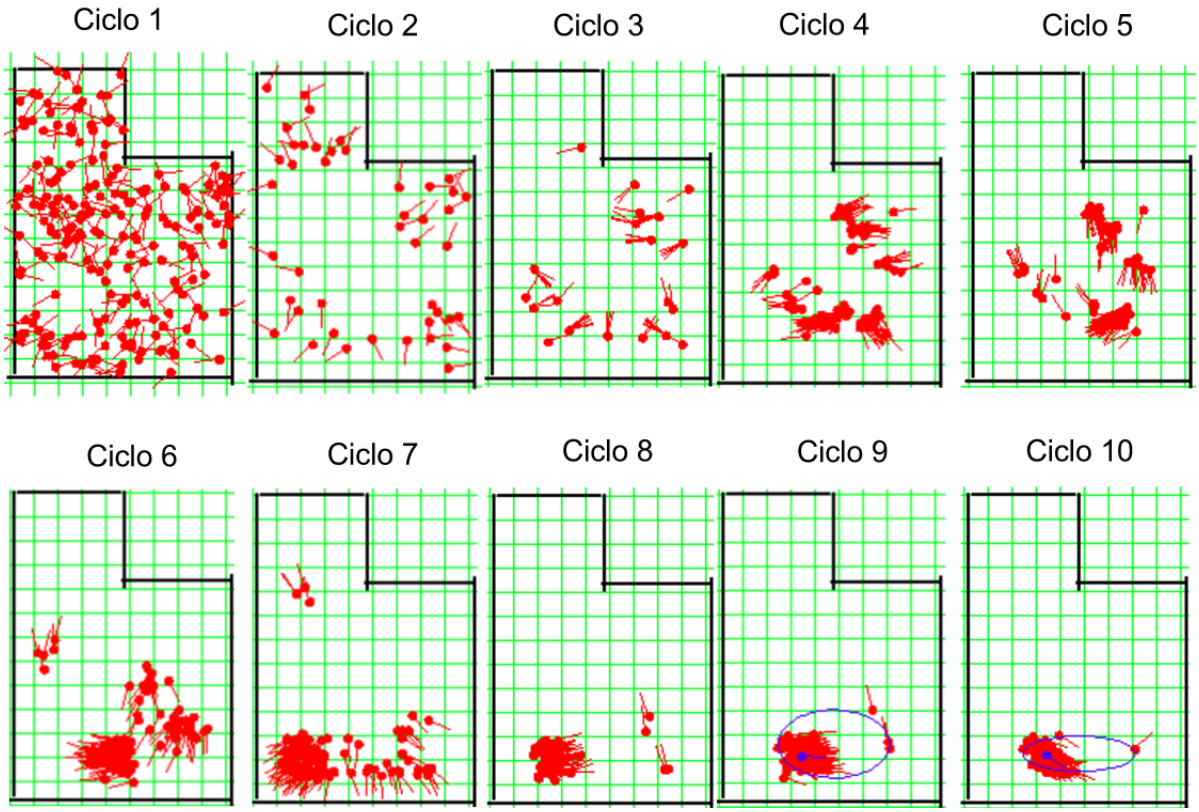

Figura 23 – Caso de Teste 1 - Cenário 1.

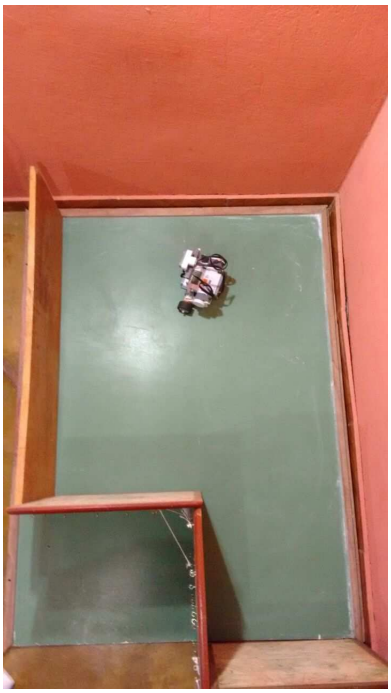

Figura 24 – Posição Real do Caso de Teste 1 - Cenário 1.

Como pode ser observado nas Figuras 23 e 24, o erro final obtido foi pequeno, o robô foi capaz de se auto-localizar com precisão.

Para evitar poluição de imagens no registro dos resultados de cada cenário dos casos de teste, os cenários e casos a seguir possuem seus registros disponíveis apenas no Apêndice C. A compilação final dos resultados se encontra nas Tabelas 8, 9, 10, 11 e 12.

Tabela 8 – Resultados Obtidos - Caso de Teste 1

Cenário	**Partículas**	**Rotação**	**Deslocamento**	**Ciclos**	**Posição**	**Precisão**
1	200	100	100	9	X: 25 a 70 Y: 15 a 30	Ótima
2	400	100	100	10	X: 17 a 37 Y: 39 a 66	Boa
3	500	100	100	21	X: 35 a 54 Y: 34 a 82	Mediana
4	100	100	100	5	X: 50 a 63 Y: 39 a 69	Mediana
5	150	100	100	3	X: 74 a 76 Y: 66 a 69	Ótima

A partir da análise empírica deste caso de teste, foi possível identificar que o número de partículas necessárias para realização, com maior qualidade, do processo de auto-localização seria de 150 partículas. Isto se dá devido à relação entre o processamento necessário para execução do filtro e a uniformidade da distribuição das partículas pelo mapa.

A partir do caso de teste 6.1.2, a quantidade de partículas utilizadas foi fixada em 150 partículas, devido ao apresentado anteriormente. Os dados que levaram a esta escolha podem ser observados em C.1.

6.1.2 Caso de Teste 2

Os resultados obtidos com o caso de teste estão dispostos na Tabela 9. Este caso de teste faz referência à variação da velocidade de rotação do robô, a qual é mensurada em graus por segundo.

Tabela 9 – Resultados Obtidos - Caso de Teste 2

Cenário	Partículas	Rotação	Deslocamento	Ciclos	Posição	Precisão
1	150	10	100	7	X: 30 a 72 Y: 47 a 71	Boa
2	150	30	100	6	X: 20 a 38 Y: 42 a 67	Ótima
3	150	50	100	3	X: 29 a 63 Y: 30 a 62	Boa
4	150	70	100	3	X: 23 a 66 Y: 29 a 68	Mediana
5	150	90	100	8	X: 36 a 61 Y: 15 a 35	Mediana

A partir da análise empírica destes resultados, fixou-se o valor da velocidade de rotação em 30 graus por segundo. Esta definição se deu devido a comparação dos resultados obtidos, utilizando o erro final como variável comparativa. Desse modo, a partir do caso de teste 6.1.3, a velocidade de rotação foi fixada no valor de 30 graus por segundo.

Estes registros podem ser visualizados na seção C.2.

6.1.3 Caso de Teste 3

Os resultados obtidos com o caso de teste estão dispostos na Tabela 10. Este caso de teste faz referência à variação da velocidade de deslocamento do robô, em unidades do diâmetro das rodas por segundo.

Tabela 10 – Resultados Obtidos - Caso de Teste 3

Cenário	Partículas	Rotação	Deslocamento	Ciclos	Posição	Precisão
1	150	30	2	4	X: 30 a 70 Y: 28 a 62	Baixa
2	150	30	5	6	X: 31 a 59 Y: 32 a 70	Mediana
3	150	30	10	3	X: 30 a 73 Y: 20 a 78	Baixa
4	150	30	15	6	X: 82 a 89 Y: 73 a 82	Boa
5	150	30	30	5	X: 23 a 54 Y: 40 a 73	Mediana

Durante a execução dos cenários dispostos na Tabela 10, foi possível identificar uma solução simples para a fonte de erro 6.2.3, a qual possibilitou a fixação da velocidade como 15 diâmetros por segundo. Este valor significa, assim como o valor de 30 diâmetros por segundo, a máxima velocidade do kit, dado que é uma velocidade inalcançável por parte do kit Mindstorm NXT.

Esta solução envolve, como é descrito na seção 6.2, a utilização de um processo de aceleração, até alcançar a velocidade definida. Desse modo, foi possível obter melhores resultados durante o processo de auto-localização, de acordo com análises empíricas do processo como um todo.

6.1.4 Caso de Teste 4

Os resultados obtidos com o caso de teste estão dispostos na Tabela 11. Este caso de teste faz referência à análise da solução utilizando a configuração que obteve os melhores resultados nos cenários anteriores, ou seja:

- **Partículas:** 150
- **Rotação:** 30 graus por segundo
- **Deslocamento:** 15 diâmetros por segundo (velocidade máxima)

O objetivo principal deste caso de teste é verificar a variabilidade dos resultados utilizando configuração fixa durante os exemplos.

Tabela 11 – Resultados Obtidos - Caso de Teste 4

Cenário	Partículas	Rotação	Deslocamento	Ciclos	Posição	Precisão
1	150	30	15	8	X: 30 a 54 Y: 28 a 45	Mediana
2	150	30	15	4	X: 62 a 87 Y: 23 a 58	Baixa
3	150	30	15	5	X: 27 a 75 Y: 10 a 32	Baixíssima
4	150	30	15	7	X: 66 a 70 Y: 27 a 33	Mediana
5	150	30	15	8	X: 37 a 67 Y: 34 a 67	Boa

O caso de teste 6.1.5 apresenta o processo de auto-localização em um mapa retangular, buscando analisar o comportamento do algoritmo em um ambiente com referenciais de baixa unicidade.

6.1.5 Caso de Teste 5

Os resultados obtidos com o caso de teste estão dispostos na Tabela 12.

- **Partículas:** 150

- **Rotação:** 30
- **Deslocamento:** 15

O objetivo deste caso de teste é verificar a viabilidade da utilização do Filtro de Partículas em um ambiente sem pontos de referência únicos.

Tabela 12 – Resultados Obtidos - Caso de Teste 5

Cenário	Partículas	Rotação	Deslocamento	Ciclos	Posição	Precisão
1	150	30	15	6	X: 21 a 63 Y: 40 a 84	Mediana
2	150	30	15	11	X: 37 a 70 Y: 47 a 90	Mediana
3	150	30	15	6	X: 27 a 75 Y: 10 a 32	Mediana
4	150	30	15	7	X: 66 a 70 Y: 27 a 33	Mediana
5	150	30	15	11	X: 37 a 67 Y: 34 a 67	Mediana

De acordo com o resultado obtido durante a execução dos cenários apresentados na Tabela 12, a qual possui seus registros dispostos na seção C.5, o robô é capaz de se auto-localizar no ambiente retangular. A análise deste resultado se encontra na seção 6.3.

Após a identificação dos melhores resultados obtidos com a variação de cada parâmetro, deve-se executar alguns cenários que variam cada parâmetro deixando os demais fixos em seus melhores resultados. Dessa forma, a análise da melhor combinação de parâmetros se torna mais correta. Os casos de teste 6.1.6, 6.1.7, 6.1.8 e 6.1.9 apresentam a execução de cada combinação de parâmetros faltante.

6.1.6 Caso de Teste 6

Os resultados obtidos com o caso de teste estão dispostos na Tabela 13. Este caso de teste faz referência à análise da solução utilizando a configuração apresentada a seguir:

- **Partículas:** Número de partículas variado.
- **Rotação:** 30 graus por segundo.
- **Deslocamento:** 15 diâmetros por segundo.

O objetivo principal deste caso de teste é verificar a variabilidade dos resultados utilizando a configuração das velocidades fixa, variando apenas a quantidade de partículas usadas.

Tabela 13 – Resultados Obtidos - Caso de Teste 6

Cenário	Partículas	Rotação	Deslocamento	Ciclos	Posição	Precisão
1	100	30	15	7	X: 34 a 52 Y: 20 a 35	Baixa
2	150	30	15	4	X: 40 a 47 Y: 34 a 59	Boa
3	200	30	15	6	X: 24 a 61 Y: 10 a 46	Mediana
4	250	30	15	9	X: 31 a 70 Y: 22 a 62	Mediana
5	300	30	15	5	X: 28 a 60 Y: 29 a 68	Boa

A partir dos resultados obtidos durante a execução do caso de teste 6.1.6, foi possível identificar que a quantidade de partículas impacta o resultado obtido, principalmente em relação a quantidade de ciclos de filtragem necessários e o tempo de processamento de cada ciclo. A utilização de 300 partículas se viu desnecessária, já que o processamento é pesado e o resultado não trás grandes ganhos em relação a utilização de menos partículas. Desse modo, optou-se por fixar a quantidade de partículas em 150, a qual apresentou o melhor resultado do cenário, envolvendo, ainda, baixo processamento, se comparado com a utilização de 300 partículas.

6.1.7 Caso de Teste 7

Os resultados obtidos com o caso de teste estão dispostos na Tabela 14. Este caso de teste faz referência à análise da solução utilizando a configuração apresentada a seguir:

- **Partículas:** 150.
- **Rotação:** Velocidade de rotação variada (de 10 a 50 graus por segundo).
- **Deslocamento:** 15 diâmetros por segundo (velocidade máxima).

O objetivo principal deste caso de teste é verificar a variabilidade dos resultados utilizando a quantidade de partículas, de acordo com o resultado obtido no caso de teste 6.1.6, e a velocidade de deslocamento fixas. Variando, dessa forma, apenas o parâmetro relacionado a velocidade de rotação.

Tabela 14 – Resultados Obtidos - Caso de Teste 7

Cenário	Partículas	Rotação	Deslocamento	Ciclos	Posição	Precisão
1	150	10	15	7	X: 32 a 50 Y: 40 a 85	Baixa
2	150	20	15	6	X: 44 a 69 Y: 22 a 48	Ótima
3	150	30	15	7	X: 55 a 80 Y: 26 a 50	Mediana
4	150	40	15	6	X: 35 a 72 Y: 82 a 109	Baixíssima
5	150	50	15	10	X: 72 a 75 Y: 41 a 43	Baixíssima

A partir dos resultados obtidos ao longo da execução do caso de teste 6.1.7, observou-se que velocidades de rotação elevadas podem prejudicar a precisão da navegação do robô. Isto se dá, principalmente, devido aos deslizes entre o piso e as rodas. Observou-se que a utilização de 20 graus por segundo gerou o melhor resultado do cenário, desse modo, fixou-se a velocidade de rotação em 20 graus por segundo.

6.1.8 Caso de Teste 8

Os resultados obtidos com o caso de teste estão dispostos na Tabela 15. Este caso de teste faz referência à análise da solução utilizando a configuração apresentada a seguir:

- **Partículas:** 150.
- **Rotação:** 20 graus por segundo.
- **Deslocamento:** Velocidade de deslocamento variada.

O objetivo principal deste caso de teste é verificar a variabilidade dos resultados utilizando a quantidade de partículas, de acordo com o resultado obtido no cenário 6.1.6, e a velocidade de rotação fixas. Variando, dessa forma, apenas o parâmetro relacionado a velocidade de deslocamento.

Tabela 15 – Resultados Obtidos - Caso de Teste 8

Cenário	Partículas	Rotação	Deslocamento	Ciclos	Posição	Precisão
1	150	20	5	4	X: 40 a 60 Y: 46 a 52	Boa
2	150	20	10	5	X: 32 a 48 Y: 42 a 68	Ótima
3	150	20	15	5	X: 29 a 71 Y: 25 a 59	Baixa
4	150	20	20	9	X: 30 a 74 Y: 37 a 58	Mediana
5	150	20	25	7	X: 61 a 77 Y: 42 a 55	Baixíssima

A partir dos resultados obtidos com o caso de teste 6.1.8, fixou-se a velocidade de deslocamento em 10 diâmetros por segundo, já que foi obtido o melhor resultado com esta configuração. O caso de teste 6.1.9 apresenta o comportamento do robô utilizando a melhor configuração obtida, de acordo com os 8 primeiro casos de teste.

6.1.9 Caso de Teste 9

Os resultados obtidos com o caso de teste estão dispostos na Tabela 16. Este caso de teste faz referência à análise da solução utilizando a configuração apresentada a seguir:

- **Partículas:** 150.
- **Rotação:** 20 graus por segundo.
- **Deslocamento:** 10 diâmetros por segundo.

O objetivo principal deste caso de teste é verificar a variabilidade dos resultados utilizando a quantidade de partículas, a velocidade de deslocamento e a velocidade de rotação fixas, de acordo com o resultado obtido nos casos de teste anteriores. Este conjunto de parâmetros contempla a melhor configuração obtida com os casos de teste.

Tabela 16 – Resultados Obtidos - Caso de Teste 9

Cenário	Partículas	Rotação	Deslocamento	Ciclos	Posição	Precisão
1	150	20	10	4	X: 41 a 79 Y: 20 a 53	Boa
2	150	20	10	7	X: 29 a 64 Y: 32 a 78	Boa
3	150	20	10	7	X: 28 a 30 Y: 42 a 45	Baixíssima
4	150	20	10	3	X: 21 a 60 Y: 35 a 75	Boa
5	150	20	10	4	X: 29 a 64 Y: 31 a 71	Mediana

A partir da observação e análise empírica durante a realização dos casos de teste apresentados anteriormente, foi possível identificar fontes específicas de erros. Assim como obter uma conclusão sobre a viabilidade da resolução parcial do problema de SLAM em um contexto limitado da Robótica Educacional. As seções 6.2 e 6.3 registram estes resultados obtidos durante a pesquisa.

6.2 Fontes de Erros

Como foi apresentado ao longo do trabalho, o mundo da robótica está envolto em erros. Desse modo, deve-se identificar o motivo do erro para que o mesmo possa ser minimizado ao máximo, buscando garantir uma navegação adequada. Por este motivo, durante a realização desta pesquisa, buscou-se identificar o maior número de fontes de erros possível para que as mesmas fossem atacadas, minimizando a margem de erro durante o processo de auto-localização.

Cada fonte de erro identificada foi listada, com o objetivo de descrevê-la e apresentar uma possível solução que minimize o erro advindo desta fonte. Esta listagem se encontra nas sub-seções 6.2.1, 6.2.2, 6.2.3, 6.2.4, 6.2.5 e 6.2.6.

6.2.1 Distância entre Rodas

De acordo com o apresentado na seção 5.3, o robô utilizado é composto de duas rodas motorizadas que controlam a movimentação do robô, e uma roda solta para apoio. Utilizando esta arquitetura de montagem, é possível realizar rotações sobre seu próprio eixo, fazendo com que uma roda gire para frente e outra para tras. Esta técnica é utilizada a todo momento para movimentação no ambiente, já que uma rotação sobre o próprio eixo possibilita o acompanhamento da direção em que o robô se locomove com maior clareza.

Utilizando a rotação sobre o próprio eixo, toda a navegação pode ser feita com base em cálculos trigonométricos, o que possibilita a sua utilização no contexto educacional, seja no ensino superior, médio ou até fundamental. Desse modo, durante a realização desta pesquisa, busco-se a utilização de rotações sobre o próprio eixo para realização de curvas. A Figura 25 apresenta de forma clara o funcionamento da rotação.

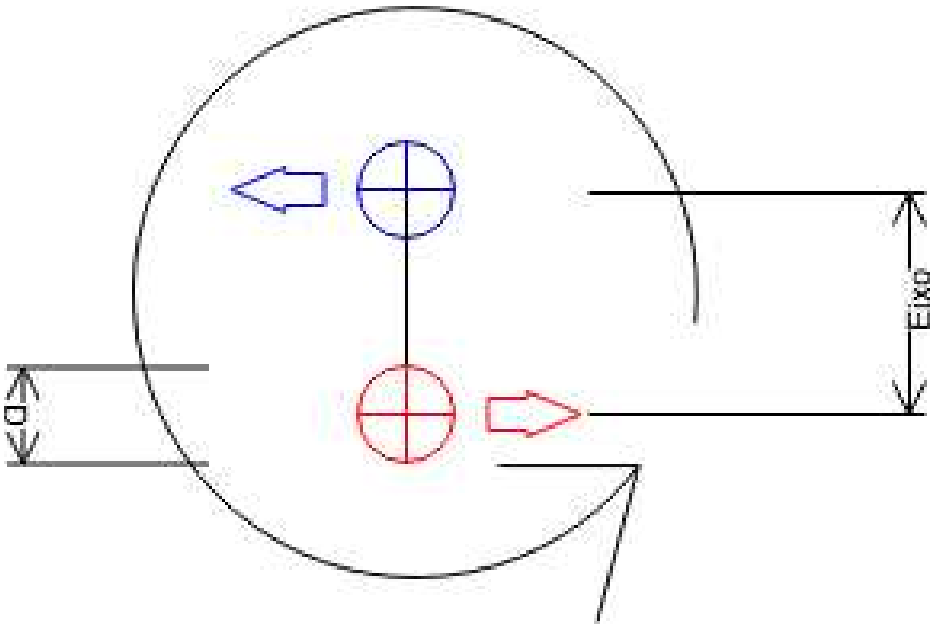

Figura 25 – Rotação sobre o Próprio Eixo.

Visto que que a solução utilizada envolve rotações sobre o próprio eixo, buscou-se minimizar ao máximo a margem de erro presente nesta técnica. Para que a rotação funcione da maneira desejada, deve-se conhecer, com o máximo de precisão, a distância entre cada roda, ou o eixo do robô (como pode ser observado na Figura 25) e o diâmetro de cada roda, representado por "D"na Figura 25.

A distância entre cada roda é necessária pois quanto maior a distância, maior o número de giros em cada roda para que a mesma angulação seja alcançada. Dado que a distância entre cada roda é representada pela variável "eixo", o diâmetro de cada roda é representado pela variável "d"e a angulação, pela variável "a", a rotação é uma função que segue a assinatura *f(a, eixo, d)*. Desse modo, cada um desses parâmetros é essencial para a garantia da precisão das rotações durante a movimentação do robô pelo mapa.

A partir da experiência obtida durante a realização da pesquisa, pôde ser observado que a melhor forma de definir a distância entre as rodas e o diâmetro das mesmas, como descreve a seção 6.2.2, é a definição por e calibração a partir de testes. Estas variáveis podem ser utilizadas como calibradores, que possibilitam a minimização de erros advindos de deslizes, como apresenta a seção 6.2.3. Desse modo, é possível esconder erros a partir da configuração destas variáveis. O sugerido é a configuração destas variáveis obervando resultados simples, como a realização de rotações completas e o erro final obtido.

6.2.2 Diâmetro da Roda

Assim como a comunidade de robótica mundial, esta pesquisa se baseia na utilização de sensores odométricos durante a navegação. Ou seja, as rotações de cada roda são registradas buscando calcular a distância e o sentido percorrido por cada roda. Esta informação possibilita que o robô saiba o quanto andou, para onde andou e, desse modo, possa concluir onde se encontra atualmente.

A odometria se baseia na matemática básica, mais especificamente, na trigonometria. Possuindo a informação do tamanho de cada roda é possível saber a distância do pneu ao centro da roda, ou ponto de ligação ao eixo. Esta distância, ou raio da roda, possibilita o calculo da quantidade de rotação necessária para percorrer determinada distância por parte da roda. A Figura 26 apresenta o fundamento da odometria, o que justifica a necessidade precisa do raio de cada roda.

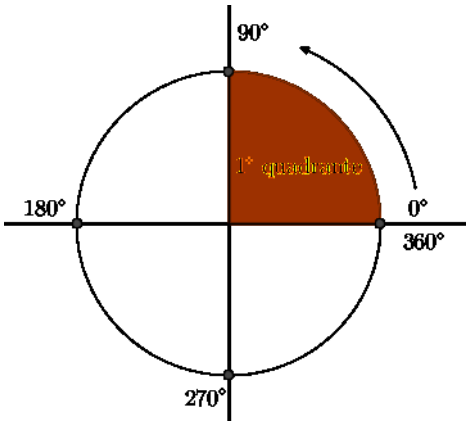

Figura 26 – Representação da Estratégia Odométrica.

A utilização da Odometria se baseia em um cálculo bastante simples, como apresenta a Figura 26. A distância percorrida na margem da circunferência destacada como primeiro quadrante faz referencia à distância percorrida pela roda em relação ao piso. A obtenção desta distância se torna fácil com o conhecimento do raio e da rotação feita pela roda.

Deste modo, deve-se registrar o tamanho de cada roda com precisão, levando em consideração a distância entre cada margem da roda, com a reta passando pelo centro. Assim como apresentado na seção 6.2.1, a obtenção desta variável deve ser feita a partir da realização de diversos testes, buscando minimizar ao máximo o erro final obtido.

6.2.3 Deslizes entre a Roda e o Piso

Durante a pesquisa, diferentes materiais foram utilizados como piso do ambiente de navegação, como azulejo, madeira e cimento. Cada material possui suas características específicas, porém todos geram erros advindos de deslizes ou fixações em buracos e rachaduras.

Devido, principalmente, à característica da roda solta utilizada pela pesquisa, a qual é feita a partir de uma engrenagem da LEGO, como foi apresentado na Figura 14, o principal problema ao qual deve-se estar atento é a fixação da mesma em buracos ou rachaduras pelo mapa. Esta situação se torna bastante crítica pois não possui uma uniformidade e, muito menos, uma previsibilidade, o que inviabiliza a utilização de técnicas que minimizem esta fonte de erro.

Visto isso, buscou-se a utilização de mapas com pisos lisos e uniformes, minimizando a ocorrência de fixações da roda traseira. Entretanto, como foi argumentado anteriormente, mapas lisos ainda incluem a possibilidade de erros advindos de deslizes entre as rodas e o piso. Este erro se dá, principalmente, nos momentos de arranque e frenagem, devido a brusca mudança de estado da roda, como descreve a Primeira Lei de Newton.

A partir de análises empíricas, obteve-se a conclusão de que a utilização de uma aceleração progressiva no início e término da movimentação minimiza consideravelmente a margem de erro final na navegação. Desse modo, os cenários de teste apresentados anteriormente foram realizados utilzando a técnica de acelaração progressiva, fazendo parte da solução final da pesquisa.

6.2.4 Precisão dos Sensores Odométricos

Como foi descrito na seção 6.2.2, o funcionamento da Odometria se dá com base na contagem de rotações de cada roda. Visto que esta rotação pode ser parcial, possuindo uma unidade mínima específica, de 1 grau, no caso do kit Mindstom. Desse modo, os cálculos odométricos são baseados na precisão inicial de 1 grau para cada medição. Porém, cálculos odométricos envolvem centenas e até milhares de medições e manipulações de variáveis derivadas, o que pode gerar um grande erro ao final da navegação.

O funcionamento deste tipo de sensor segue o apresentado na Figura 27. De acordo com o apresentado, o sensor alterna o sinal de saída entre alto e baixo (1 e 0, por exemplo), de acordo com o processo de rotação. A variação de tempo entre uma mudança de sinal para a próxima mudança viabiliza a obtenção da distância percorrida, de acordo com [Spielmann et al. 2013].

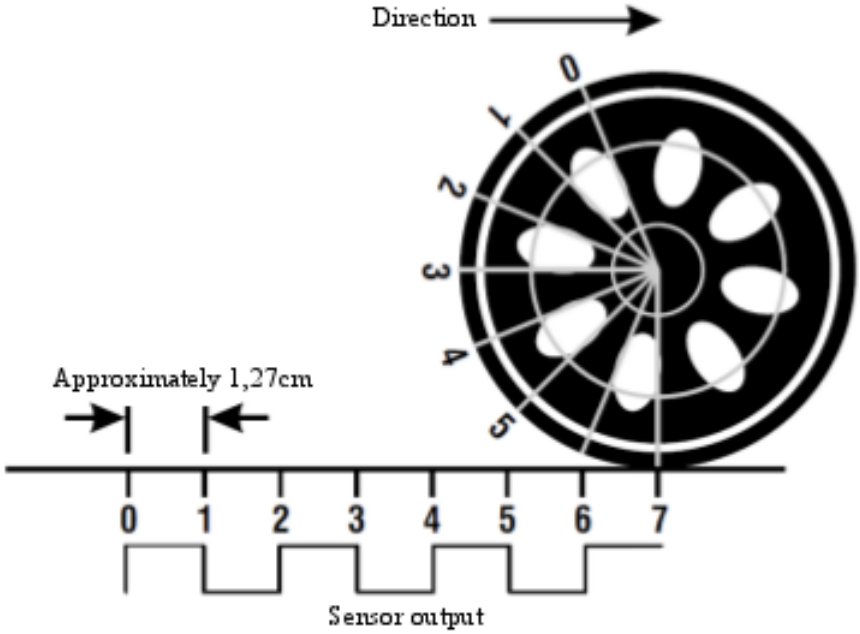

Figura 27 – Funcionamento da Odometria.

Esta fonte de erro, infelizmente, não possui soluções diretas, já que é derivada da característica do *hardware* utilizado na pesquisa. A mesma faz referência a uma das características que compõem o contexto limitado da Robótica Educacional, como foi descrito ao longo de toda a pesquisa.

6.2.5 Característica do Sensor de Distância

Durante a pesquisa foi utilizado um sensor de distância do tipo sonar, o qual possui algumas características específicas, que podem prejudicar a precisão de navegação e auto-localização. A seção B.2 apresenta algumas características técnicas do Kit utilizado, apresentando algumas informações úteis em relação ao sensor de distância.

O funcionamento de um sonar ocorre a partir da utilização da emissão e recepção de sinais sonoros para cálculo de distância entre o emissor/receptor e o obstáculo mais próximo a sua frente. Como o sinal enviado é um sinal sonoro, o mesmo se locomove no ambiente (ar) de forma tridimensional, seguindo, mais especificamente, o formato de um cone, como apresenta a Figura 28. Esta característica envolve grande parte dos problemas advindos da utilização de um sensor desta categoria.

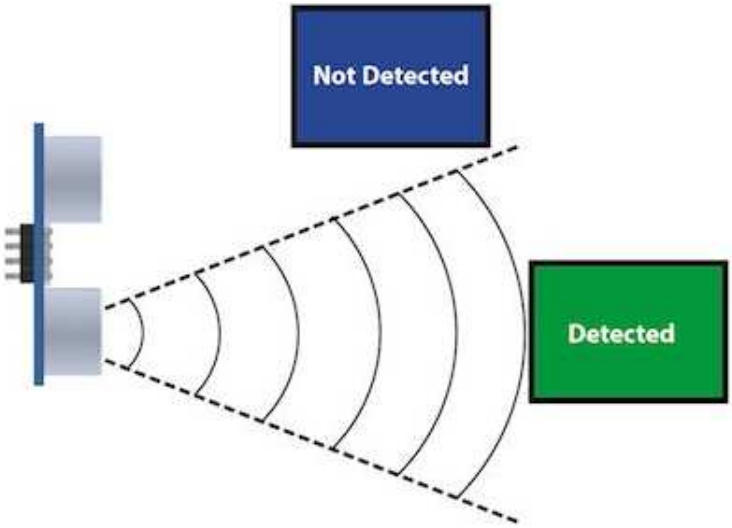

Figura 28 – Emissão do Sinal Ultrasônico.

A partir da observação da Figura 28, é possível identificar possíveis problemas gerados por esta característica. Basicamente, o sonar observa o ambiente de maneira unidimensional, em relação aos *inputs* obtidos pelo mesmo. *Inputs*, estes, que são compostos apenas da distância entre o robô e um objeto a sua frente. Entretando, estes *inputs* são gerados a partir da iteração tridimensional do som com o ambiente, incluindo no resultado uma margem de erro importante, que não deve ser ignorada.

Para analisar o impacto desta característica, deve-se compreender os detalhes técnicos do sonar utilizado, mais especificamente, a distância máxima de alcance do sonar e a angulação de emissão do sinal sonoro. Durante esta pesquisa, o sonar utilizado, presente no Kit Mindstorm NXT, possui uma angulação de emissão do sinal de 30 (trinta) graus e uma distância máxima de 255 (duzentos e cinquenta e cinco) centímetros. Com estas informações é possível padronizar o ambiente e o contexto de utilização do sonar de uma forma que os resultados obtidos possuam uma margem de erro aceitável, de acordo com o contexto trabalhado.

Com a emissão do sinal a uma angulação de 30 graus, por exemplo, obstáculos identificados a 180 centímetros podem possuir até 90 centímetros de margem de erro, gerando um resultado pouco confiável para o contexto da Robótica Móvel. Desse modo, deve-se evitar a identificação de obstáculos a distâncias longas, as quais foram fixadas, durante esta pesquisa, em aproximadamente um metro de distância.

Devido às características do sonar, a identificação de pontos de referência não se baseia em características físicas dos obstáculos, utilizando a relação de distância entre robô e obstáculo como forma de identificação de pontos de referência. A ideia não é identificar locais específicos que possuem pontos de referência, como a técnica da utilização de postes emissores de infra-vermelho, e sim a identificação da posição atual do robô em relação aos

obstáculos, a qual viabiliza o processo de auto-localização do robô.

Buscando a minimização da margem de erro, alguns detalhes do ambiente precisaram ser fixados, limitados e bem definidos. Foi fixado um tamanho máximo de 1,5 metros quadrados para o ambiente de navegação, assim como a utilização de obstáculos lisos e uniformes como pontos de referência. Basicamente, estes obstáculos se resumem a utilização de paredes na margem do ambiente de navegação, sem a utilização de obstáculos internos ao mapa.

O sensor utilizado é o sonar original do Kit, o qual pode ser observado na Figura 29.

Figura 29 – Sensor de Distância Ultrasônico.

6.2.6 Colisões em Obstáculos

A característica presente nos sensores de distância e odométricos, apresentadas na seção 6.2.5 e 6.2.4, pode gerar o acontecimento de uma colisão. Colisões podem ser ocorrer de forma frontal ou lateral, incluindo erros de grande proporção na navegação, principalmente no primeiro caso. A colisão frontal envolve o caso em que o robô tenta "empurrar"um obstáculo, ou seja, há uma colisão e o robô continua girando seus atuadores, sem sair do lugar. Neste caso, o qual pode ser considerado o pior caso, faz-se referência ao caso do *sequesto do robô*, onde o mesmo se encontra perdido após uma colisão. A Figura 30 apresenta, de forma clara, a situação levantada.

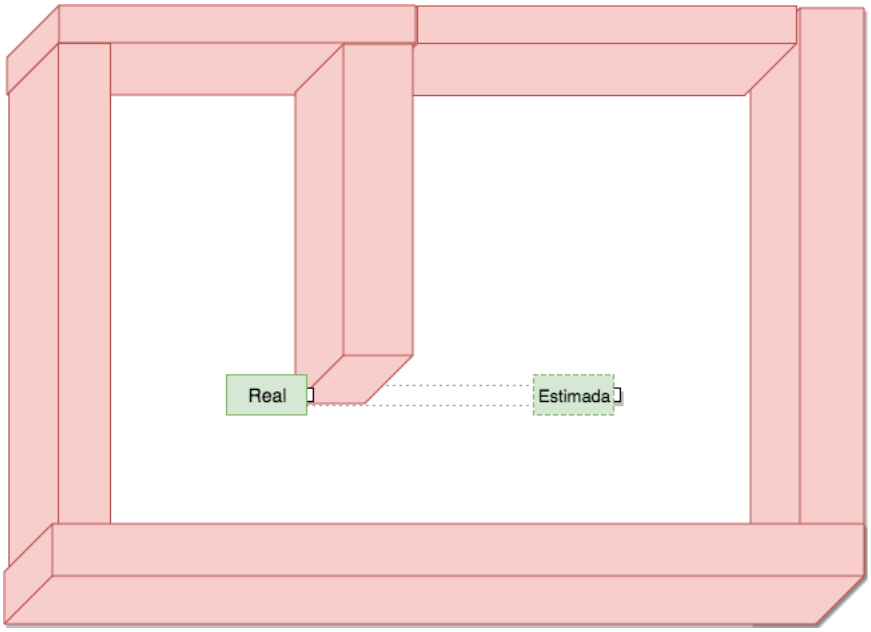

Figura 30 – Modelagem da situação de colisão frontal.

A colisão lateral, apesar de não gerar um resultado tão crítico, deve ser analisada com atenção. Neste caso, o robô encosta paralelamente em algum obstáculo, esbarrando uma de suas rodas no mesmo. O robô continua navegando, porem com adição de erros, geralmente inclinando o robô em direção ao obstáculo, podendo gerar, ainda, uma colisão frontal derivada da colisão lateral. A Figura 31 apresenta a situação discutida.

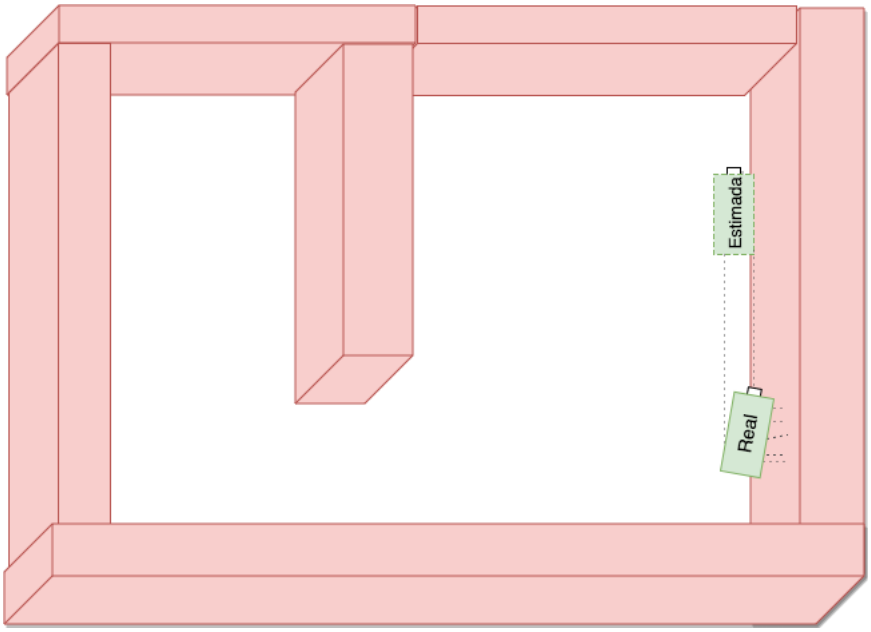

Figura 31 – Modelagem da situação de colisão lateral.

O problema de colisões em obstáculos é crítico e não possui técnicas simples para resolução mesmo. Entretanto, a técnica de Filtro de Partículas, escolhida para possibilitar

a auto-localização do robô, durante esta pesquisa, possibilita a resolução do problema do *sequestro do robô*, como apresentado na seção 2.2.2. A situação de colisões, seja frontal ou lateral, pode ser trabalhada como uma situação de sequestro do robô, possibilitando, a partir da utilização do Filtro de Partículas, a resolução, mesmo que parcial, do problema de colisões. Um exemplo claro desta situação pode ser observado na seção 6.1.1, exemplo 3. Neste caso, o robô sofreu uma colisão lateral com a parede enquanto se locomovia pelo ambiente, ainda buscando se auto-localizar. A partir de análises empíricas, observou-se que o ocorrimento de uma colisão frontal ou lateral durante o processo de auto-localização é tolerável.

Em casos de colisão, de acordo com os testes executados, o processo de auto-localização é capaz de ser concluído com sucesso, porém com margem de erro razoavelmente superior a média geral apresentada e um número de ciclos de filtragem muito superior a média de ciclos. Desse modo, vê-se que a utilização do Filtro de Partículas se torna essencial em um contexto limitado, como o da Robótica Educacional.

6.3 Análise de Viabilidade

Apesar da utilização de ferramentas simples e baratas, a técnica analisada durante esta pesquisa, o Filtro de Partículas, está comumente presente em projetos de alto custo da robótica mundial. Visto isto, foram necessárias diversas adaptações, tanto no robô, quanto no ambiente no qual o mesmo se locomove. Estas adaptações são resultado da identificação de fontes de erro, como descreve a seção 6.2, assim como a identificação de soluções que possam minimizar o erro advindo de cada fonte.

Após a identificação das fontes de erro e adaptações necessárias para o robô e o ambiente, foram executados os cenários de teste. Estes tiveram como objetivo principal, a identificação da variabilidade do resultado de acordo com a alteração de alguns parâmetros, como o número de partículas, a velocidade de deslocamento, a velocidade de rotação e o mapa utilizado.

A configuração dos parâmetros que gerou os melhores resultados foi:

- Numero de Partículas: 150;
- Velocidade de Rotação: 20 graus por segundo;
- Velocidade de Deslocamento: 10 diâmetros por segundo.

Com esta configuração, 60% dos resultados foram considerados bons, e apenas 20% foram considerados ruins. Com esta proporção, é possível chegar a uma conclusão positiva em relação a utilização do Filtro de Partículas no contexto limitado da Robótica

Educacional. Embora estes resultados tenham sido obtidos a partir de diversas adaptações no robô e ambiente, o que poderia inviabilizar sua utilização, o contexto educacional possibilita e até se beneficia deste processo de adaptação e melhorias no robô e ambiente.

Este processo de adaptação do robô e ambiente, trabalhando de maneira empírica, pode envolver a imaginação e o trabalho em equipe de uma turma inteira de robótica, o que só favorece a utilização de temas como este no contexto educacional. Desse modo, pode-se concluir que a utilização da técnica do Filtro de Partículas no contexto limitado da Robótica Educacional gera resultados positivos, tanto na aplicação quanto no processo educacional dos participantes.

7 Conclusão

Esta pesquisa buscou analisar o contexto de SLAM dentro dos limites da Robótica Educacional, com o objetivo de estudar a viabilidade da utilização de técnicas renomadas na comunidade de robótica para solucionar o problema de SLAM. De acordo com o apresentado ao longo da pesquisa, a comunidade de robótica busca solucionar o problema de SLAM utilizando diversas técnicas e estratégias. Entretanto, boa parte da comunidade concorda com a utilização de algumas estratégias, como o processamento remoto, apresentado na tabela 20. Visto isso, este trabalho buscou seguir as estratégias mais utilizadas e reconhecidas para o contexto da pesquisa, como a utilização da arquitetura de processamento remoto e a utilização do Filtro de Partículas.

Como o contexto alvo da pesquisa faz referência a Robótica Educacional, buscou-se utilizar ferramentas simples, presentes no kit de robótica Mindstorm NXT e disponíveis em ambientes educacionais, advindos de baixo investimento. Foi possível estudar, analisar e adaptar estratégias e técnicas de auto-localização no contexto limitado da Robótica Educacional. Esta análise mostrou que a utilização de técnicas como o Filtro de Partículas se torna viável quando os estudantes possuem liberdade para alterar o robô e o ambiente, trabalhando de maneira empírica para minimizar a margem de erro presente na solução.

De acordo com [Galvan et al. 2006], a solução de problemas a partir do estudo empírico (pesquisa empírica) é uma característica importante no contexto da Robótica Educacional. Esta característica favorece o trabalho em equipe e a solução de problemas de forma autodidata, incendivando o interesse e a dedicação dos alunos participantes, ainda de acordo com [Galvan et al. 2006]. Desse modo, a questão da necessidade de adaptação do robô e ambiente, que poderia ser considerada prejudicial em diversos contextos, se torna uma característica importante e positiva no contexto da Robótica Educacional.

Esta pesquisa obteve importantes resultados, tanto em relação a revisão sistemática sobre o tema pesquisado, quanto em relação a implementação da solução, identificando fontes críticas de erro, maneiras de minimizar estas fontes e a viabilidade da aplicação no contexto educacional. Para que a pesquisa se complete, deve-se analisar, ainda, a integração da auto-localização e do mapeamento de ambientes, ambos implementados, completamente e parcialmente, durante esta pesquisa. Desse modo, deve-se destacar as lacunas deixadas pela pesquisa, apresentando os trabalhos futuros para que novos interessados possam seguir e completar o estudo sobre o SLAM no contexto educacional.

Os trabalhos futuros desta pesquisa são:

- Refinar implementação do mapeamento de ambientes:

Para que esta implementação possa ser integrada com facilidade no módulo de autolocalização, a mesma deve gerar um mapa no formato *svg*, destacando as paredes presentes no ambiente em forma de linhas. De acordo com as características apresentadas na seção 6.2, é possível adiantar que a solução do mapeamento de ambientes deve incluir uma camada de filtragem do resultado, que busca linearizar a representação das paredes e obstáculos.

Esta linearização é necessária devido, principalmente, a característica do sensor de distância utilizado (sonar). A Figura 32 apresenta o resultado obtido sem a utilização de um filtro para linearizar a saída obtida. Para que o mapeamento de ambientes seja incluído na solução geral, este deve, primeiramente, minimizar a margem de erro e incorporar uma camada de filtro para linearizar o resultado do mapeamento.

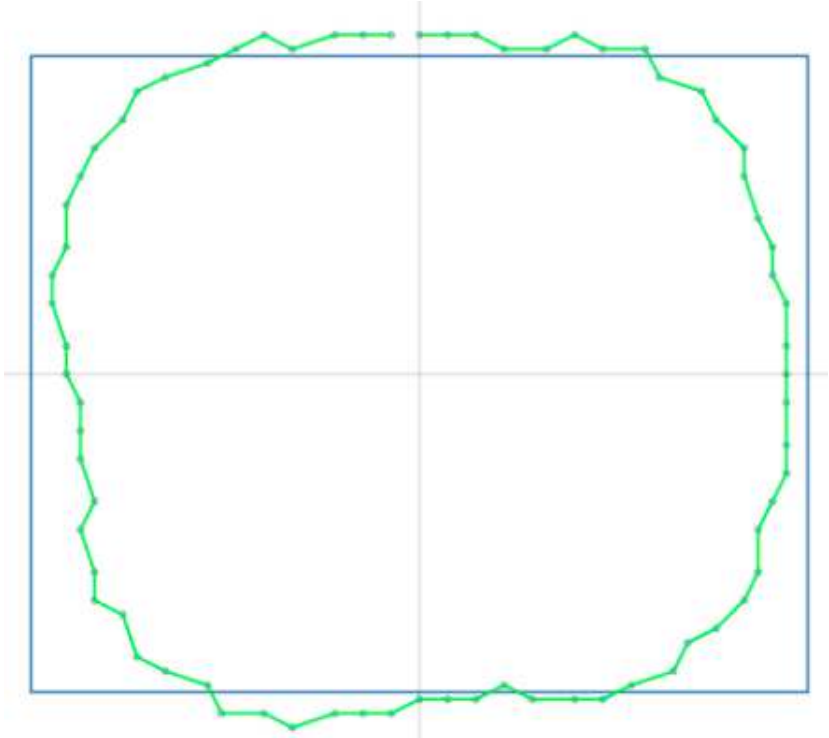

Figura 32 – Mapeamento de Ambientes - Implementação Inicial.

- Refinar solução do sequestro do robô:

  A solução do sequestro do robô, um dos motivos favoráveis a escolha do Filtro de Partículas para implementação desta solução, se encontra, no fim desta pesquisa, implementada e funcional. Entretanto, a solução utilizada durante a pesquisa envolve, apenas, a solução de um "sequestro"durante a etapa dos ciclos de filtragem, onde o robô navega pelo ambiente buscando se encontrar no mesmo.

  Como trabalho futuro, pode-se incluir a ideia de solucionar o problema do sequestro do robô a qualquer momento. Ou seja, após os ciclos de filtragem, quando o robô já conhece sua localização atual, pode ocorrer um "sequestro do robô". Neste caso, o robô deve perceber as mudanças no ambiente, chegando a conclusão de que o mesmo

não se encontra mais no local conhecido. Ao concluir que o robô não se encontra na posição conhecida, o mesmo deve iniciar novamente o processo de auto-localização, criando o conjunto de partículas e executando os ciclos de filtragem. Esta solução não precisa seguir a estratégia, para execução dos ciclos de filtragem, adotada durante esta pesquisa, já que esta estratégia envolve a iteração do pesquisador/usuário durante a execução dos ciclos. Esta execução, no caso do "sequestro do robô", pode ser automatizada, prosseguindo sem a necessidade da participação do usuário.

- Integrar mapeamento e auto-localização, completando o SLAM:

  Durante esta pesquisa foi implementada a solução da auto-localização utilizando o Filtro de Partículas. Além disso, foi iniciada a implementação da solução para mapeamento de ambientes, durante a prova de conceito desta pesquisa. Como trabalho futuro, deve-se integrar estas duas soluções, possibilitando a auto-localização e o mapeamento de ambientes simultâneo (SLAM).

  A API utilizada para solucionar o problema de SLAM exige a utilização de um mapa pronto no formato "svg", para que as partículas sejam distribuídas no mesmo. Entretando, sabe-se que, durante as primeiras fases do SLAM, o mapa se encontra parcialmente concluído, o que inviabilizaria a utilização da API leJOS, neste caso. Porém, a partir da experiência obtida durante a pesquisa, observou-se uma estratégia viável para integração destes dois módulos. Nesta estratégia, o robô constrói o mapa com as informações que possui e preenche o restante com informações fictícias, possibilitando o "fechamento"do mapa e, com isso, a utilização do mesmo no algoritmo do Filtro de Partículas.

Apêndices

APÊNDICE A – Revisão Sistemática

A revisão bibiliográfica deste trabalho foi complementada pelo emprego da técnica de revisão sistemática. Nesta seção, apresentam-se as três fases básicas da revisão sistemática definidas por [Kitchenham et al. 2006], A.0.1 *planejamento da revisão*, A.0.2 *condução da revisão* e A.0.3 *reporte dos resultados da revisão*, como mostra a Figura 33.

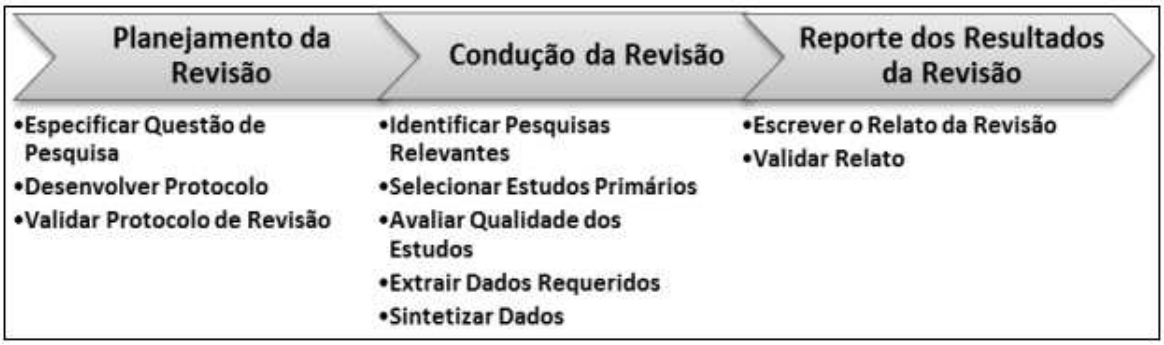

Figura 33 – Processo geral de revisão sistemática, segundo [Kitchenham et al. 2006]

A.0.1 Planejamento da Revisão

A revisão sistemática foi realizada entre os meses de março e junho de 2016, utilizando como fonte de busca as bases *IEEE*, *CAPES* e *Springer*. A partir dos modelos de revisão sistemática apresentados por [Kitchenham et al. 2006], foi desenvolvido um protocolo de revisão, o qual possibilita, a outros pesquisadores, a repetição da pesquisa.

A.0.1.1 Objetivos e Questão de Pesquisa

O objetivo inicial do trabalho é estudar a problemática da auto-localização na robótica, identificando diferentes soluções em diversos contextos. Afirmado isso, foi possível realizar uma pesquisa bibliográfica com o objetivo de identificar diferentes linhas de pesquisa nesta área. Após uma análise superficial de cada linha de pesquisa, foi selecionada a linha de pesquisa que adota, como solução para a auto-localização, a utilização da técnica de SLAM, o que pode ser considerado como o marco teórico do trabalho.

Com a identificação do marco teórico do trabalho, a definição do foco da pesquisa torna-se uma tarefa menos árdua. Como o marco teórico deste trabalho baseia-se nas linhas de pesquisa que buscam utilizar a técnica de SLAM para realizar navegação autônoma, esta revisão sistemática teve como objetivo identificar diferentes técnicas utilizadas atualmente para solucionar o problema de SLAM em diferentes contextos, desde contextos marcados pelo uso de recursos limitados, precisando, portanto, serem mais simplificados, até mesmo contextos altamente complexos.

A definição deste objetivo da revisão dá-se pela necessidade de conhecimento amplo em relação a diferentes técnicas para solucionar o problema de SLAM. Como este trabalho buscou adaptar técnicas para um contexto da Robótica Educacional, cujos recursos são mais escassos e limitados, adicionou-se aos objetivos da revisão itens relacionados à Robótica Educacional e robôs simples.

Segundo [Kitchenham et al. 2006], o primeiro passo para se realizar uma revisão sistemática é definir a sua questão de pesquisa. Desse modo, a partir da realização de uma pesquisa bibliográfica inicial, foram identificadas as seguintes questões de pesquisa:

- **Q1**. *Quais técnicas são mais utilizadas para solucionar o problema de SLAM?* e
- **Q2**. *"Como tratar o problema de SLAM no contexto simplificado da robótica educacional?"*

Além das questões de pesquisa, de acordo com [Brito et al. 2015], alguns outros itens devem ser detacados, como:

- **População**: comunidade acadêmica e de robótica.
- **Intervenção**: adaptação de técnicas para um contexto de robótica simplificado (educacional).
- **Controle**: utilização do *Quasi-gold standard* [Kitchenham et al. 2006], que será explicado mais à frente.
- **Resultados**: obtenção de técnicas adaptáveis ao contexto simplificado (Robótica Educacional).
- **Aplicação**: servir de base para implementação da proposta desse trabalho de conclusão de curso, na qual técnicas foram adaptadas, buscando solucionar, com recursos limitados/escassos - típico cenário da Robótica Educacional - o problema de SLAM.

A partir da definição das questões de pesquisa e dos objetivos da revisão, buscou-se definir a estratégia de pesquisa, apresentada no tópico A.0.1.2.

A.0.1.2 Estratégia de Pesquisa

A estratégia de pesquisa adotada para esta revisão segue recomendações de diversos autores, como [Kitchenham et al. 2006] e [Biolchini et al. 2005], utilizando o conceito de *quasi-gold standard*.

Segundo [Zhang, Babar e Tell 2010], *gold standard* representa o conjunto completo de estudos primários referentes a uma questão de pesquisa, com máxima precisão e sensitividade. Já o *quasi-gold standard*, representa um subconjunto do *gold standard*, o qual vai sendo evoluído ao longo dos ciclos de busca, com o objetivo de se aproximar do *gold standard*.

É utilizado para definir os valores de precisão e sensitividade da busca, o que possibilita a avaliação da busca realizada, verificando a necessidade de refinamento da *string*, por exemplo. [Zhang, Babar e Tell 2010] define precisão e sensitividade da busca da seguinte forma:

- $precisão = \frac{ERO}{EO}$
- $sensitividade = \frac{ERO}{TER}$

  onde:

  ERO = número de estudos relevantes obtidos,

  EO = número de estudos obtidos e

  TER = número total de estudos relevantes.

Esta definição possibilita a criação de critérios que avaliem a qualidade da busca, ou seja, da *string* de busca utilizada. Porém, a seleção dos materiais deve seguir critérios relacionados à qualidade do material, os quais são divididos em *critério de inclusão (CI)* e *critério de exclusão (CE)*, como se pode observar a seguir:

- CI 1 - Os artigos devem estar escritos em inglês ou português;
- CI 2 - Artigos referentes à auto-localização e ao mapeamento de ambientes simultâneos (SLAM);
- CI 3 - Artigos com acesso gratuito, disponíveis na *web* para *download* ou leitura;
- CE 1 - Artigos que buscam solucionar o problema de auto-localização sem a utilização da técnica de SLAM, e
- CE 2 - Artigos que não possuem o problema de SLAM como foco principal.

A partir da definição da questão de pesquisa e dos objetivos da revisão, assim como a definição dos critérios de inclusão e exclusão, foi possível desenvolver uma *string* de busca inicial. O idioma escolhido para a *string* de busca foi o inglês, devido a sua ampla utilização nas bases de conhecimento selecionadas. Buscou-se utilizar a mesma *string* de busca em todas as bases de dados pesquisadas, exceto em alguns casos em que houve a necessidade da adaptação da *string* de acordo com os padrões adotados pela base.

Com o objetivo de identificar pesquisas relacionadas à auto-localização utilizando mapeamento de ambientes simultaneamente, a *string* de busca definida foi: *auto-localization AND environment mapping*. Na Tabela 17, são apresentados os resultados obtidos a partir desta busca.

Tabela 17 – Resultados Obtidos com a *String* Inicial

String	**IEEE, Springer e CAPES**	
	Nº. Artigos	Nº. Artigos relevantes
Auto-localization AND environment mapping	29	6

Esta primeira busca foi de extrema importância para se obter uma visão inicial da pesquisa, identificando novas palavras-chave e iniciando os ciclos de busca.

A.0.1.3 Procedimento de Seleção

As buscas foram realizadas com a mesma *string* de busca (na maioria dos casos, como já foi explicado anteriormente) nas três bases de conhecimento científico utilizadas. A cada busca realizada, os artigos foram registrados, para que, posteriormente, os mesmos pudessem ser submetidos à avaliação da qualidade e, se comprovada a relevância do mesmo, à extração de dados.

A seleção dos artigos deu-se a partir da leitura dos títulos, resumos e palavras-chave, classificando o artigo como relevante ou não, em um primeiro momento. Caso fosse confirmada a relevância do mesmo, o artigo passaria por uma avaliação mais profunda, a *avaliação da qualidade*, como mostra o tópico A.0.1.4.

A.0.1.4 Avaliação da Qualidade

A avaliação da qualidade do artigo deu-se a partir da análise do conteúdo do mesmo, focando principalmente na introdução, nos resultados e conclusões dos artigos. A avaliação positiva do artigo significa uma resposta positiva para as seguintes perguntas:

1. O estudo é interessante? (em relação aos objetivos da pesquisa)
2. As evidências apresentadas são válidas?
3. As evidências apresentadas são importantes?

4. As evidências apresentadas não contradizem autor algum selecionado como pilar da pesquisa?

Com a confirmação da qualidade do material, o mesmo foi exposto à extração de dados, ou seja, à leitura completa e detalhada do artigo.

A.0.1.5 Extração de Dados

Com o objetivo de organizar os dados obtidos, facilitando o manuseio das informações, os dados extraídos de cada artigo foram registrados conforme o modelo apresentado na Tabela 18.

Tabela 18 – Exemplo de Registro de Material

Título	**Autor(es)**	**Data de publicação**	**Fonte da publicação**	**Listagem das informações importantes**
Integration of Vision based SLAM and Nonlinear Filter for Simple Mobile Robot Navigation	Dae Hee Won, Young Jae Lee, Sangkyung Sung, Taesam Kang	2008	IEEE	-Utilização de sensor de visão e encoders. -Filtro de partículas

A.0.2 Condução da Revisão

Durante a condução da revisão, a busca efetiva dos materiais é realizada, os ciclos de busca são documentados e a *string* de busca é refinada, como afirma [Mafra e Travassos 2006].

Esta pesquisa foi realizada entre os meses de março e junho de 2016. A *string* de busca apresentada no tópico A.0.1.2 resultou em uma visão considerada fraca, pelo pesquisador e seus orientadores, sobre a auto-localização e o mapeamento de ambientes, peças chave da técnica de SLAM. Além disso, essa busca possibilitou a criação do *quasi-gold standard* inicial, com apenas um artigo: *Auto-localização e construção de de mapas de ambiente para robôs móveis baseados em visão omnidirecional estéreo* [Oliveira 2008].

Desse modo, foi necessária a realização de uma pesquisa manual para obter maior conhecimento sobre o tema e as palavras-chave a serem usadas para garantir maior qualidade dos resultados obtidos. Com a realização desta pesquisa, os seguintes artigos foram adicionados ao *quasi-gold standard*:

- *The Cleaning Robot Project: Aplicação do Filtro de Kalman na Auto-Localização de um Sistema Robótico Autônomo* [Pinto 2008],

- *Integration of Vision based SLAM and Nonlinear Filter for Simple Mobile Robot Navigation* [Won et al. 2008] e
- *A Solution to the Simultaneous Localization and Map Building (SLAM) Problem* [Dissanayake et al. 2001].

A partir da análise destes artigos inciais, foi possível identificar diversas palavras-chave que levavam à pesquisa desejada, possibilitando refinamento da *string* de busca. Adicionando à mesma novos termos, como *"SLAM problem"* e *"simultaneous"*, evoluindo a *string* e obtendo o seguinte resultado: *"simultaneous AND auto-localization AND environment mapping AND SLAM problem"*.

O ciclo de busca utilizando esta *string* gerou poucos resultados, selecionando apenas um para análise: *Improved global localization of an indoor mobile robot via fuzzy extended information filtering* [Lin e Tsai 2008]. Com o objetivo ampliar a abrangência da busca, optou-se por modificar a palavra-chave *auto-localization* por apenas *localization*, obtendo a seguinte *string* de busca: *"simultaneous AND localization AND environment mapping AND SLAM problem"*.

Com a realização deste novo ciclo de busca, diversos artigos relevantes foram identificados e adicionados ao *quasi-gold standard*. Os artigos presentes no *quasi-gold* corrente foram encontrados com esta nova busca, evidenciando uma certa qualidade da *string* de busca. Os artigos adicionados ao *quasi-gold standard* durante este ciclo são:

- *A Simultaneous Localization and Mapping Algorithm in Complex Environments: SLASEM* [Sun et al. 2010],
- *A Neuro-Fuzzy Assisted Extended Kalman Filter-Based Approach for Simultaneous Localization and Mapping (SLAM) Problems* [Chatterjee e Matsuno 2007],
- *Map Management for Efficient Simultaneous Localization and Mapping (SLAM)* [Dissanayake et al. 2002] e
- *Simultaneous Localization and Map Building by Integrating a Cache of Features* [Costa, Dias e Araújo 2006].

Com o intuito de especificar mais a busca, foi adicionada, devido a uma dica do orientador prof. Dr. Maurício Serrano, a palavra-chave *"simple robots"* à *string* de busca, chegando a seguinte *string*: *"simultaneous AND localization AND environment mapping AND SLAM problem OR ("simple mobile robots"AND slam)"*.

Com esta mudança, obteve-se outra visão desta pesquisa como um todo. Dessa forma, foram identificadas diversas pesquisas que buscam solucionar problemas de locomoção, como o problema de SLAM, em contextos limitados, da mesma forma que objetivo

geral deste trabalho. Por fim, foram adicionados ao *quasi-gold standard*, os seguintes artigos:

- *BatSLAM: Simultaneous Localization and Mapping Using Biomimetic Sonar* [Steckel e Peremans 2013],
- *Neural Network-Based Multiple Robot Simultaneous Localization and Mapping* [Saeedi et al. 2011] e
- *Visual simultaneous localization and mapping: a survey* [Fuentes-Pacheco, Ruiz-Ascencio e Rendón-Mancha 2012].

Como é possível observar, os artigos selecionados para adição no *quasi-gold standard* não envolviam temas referentes a robôs simples, apesar da afirmação de que esta mudança havia modificado a visão da pesquisa. Artigos referentes a este tema não foram adicionados ao *quasi-gold standard* devido ao fato dos mesmos não atenderem ao critério de exclusão *"CE 2 - Artigos que não possuam o problema de SLAM como foco principal"*.

Em contrapartida, este ciclo possibilitou o conhecimento de novos termos, viabilizando um refinamento eficiente para o próximo ciclo de busca.

No próximo ciclo de busca, foram adicionadas palavras-chave referentes à Robótica Educacional e às estratégias de resolução do problema de SLAM, como mostra a *string*: *"(Simple mobile robot? AND (SLAM OR auto-localization)) AND (map* OR education* robot*) AND strateg*"*. Além de adicionar estes novos termos, utilizou-se de técnicas disponíveis nas bases, como a utilização de *'?'*, que representa qualquer caractere, e *'*'*, que significa que quaisquer caracteres precedidos dos caracteres anteriores ao *'*'* serão considerados.

Para realização desta busca, foi necessária a adaptação da *string* de busca durante a pesquisa na base de dados *Springer*, devido a diferenças nos padrões de definição da *string*. Para esta adaptação, novas palavras-chave precisaram ser selecionadas, as quais foram obtidas a partir dos resultados advindos das outras bases, com a mesma *string*.

A *string* adaptada para a base *Springer* foi: *"(Simple AND mobile AND robot AND SLAM AND localization AND mapping) AND (educational AND navigation AND simultaneous) AND strategies"*.

Ao realizar esta busca, foi observado que os resultados atendiam, em sua maioria, ao desejado pela pesquisa, sendo muitos artigos selecionados para análise e avaliação e alguns adicionados ao *quasi-gold standard*, como:

- *Incremental SLAM with Backtracking Data Association for Mobile Robots* [Ji et al. 2008],

- *Mapping and Pursuit-Evasion Strategies For a Simple Wall-Following Robot* [Katsev et al. 2011] e
- *A Simple and Parallel Algorithm for Real-Time Robot Localization by Fusing Monocular Vision and Odometry/AHRS Sensors* [Wang, Liu e Li 2014].

Com a realização desta busca, foram obtidos 27 artigos, restando apenas 20 após a avaliação da relevância dos mesmos para a pesquisa. Desde o primeiro ciclo de busca, diversos artigos foram considerados relevantes para a pesquisa, chegando a um número de 42 (quarenta e dois) artigos relevantes.

Ou seja, aplicando os conceitos de *precisão* e *sensitividade*, temos que:

- $sensitividade = \frac{20}{42}$
- $precisao = \frac{20}{27}$

Desse modo, foi obtida uma *sensitividade* de 47% e uma precisão de 74%, dando fim aos ciclos de busca com 42 artigos selecionados e analisados. A Tabela 19 apresenta, de maneira resumida, a evolução da *string* de busca, comparando a *string* inicial com a *string* final da pesquisa.

Tabela 19 – Comparação das *strings* inicial e final

String	**Fonte de busca**	**Resultados**		**Observações**
		Total	**Relevantes**	
Inicial	*IEEEXplore*	2	1	String construída sem o conhecimento necessário para utilização das palavras-chave que representam a pesquisa.
	Springer	19	2	
	CAPES	8	3	
Refinada	*IEEEXplore*	17	13	String refinada, ao longo de diversos ciclos de busca, adicionando novas palavras-chave e obtendo resultados mais específicos.
	Springer	6	3	
	CAPES	4	4	

Durante a realização de cada ciclo, os artigos foram analisados, avaliados e seus dados foram extraídos como fonte de estudo para a realização deste trabalho de conclusão de curso. Na seção A.0.3, serão apresentados os resultados de maneira organizada e simplificada.

A.0.3 Publicação dos Resultados

A primeira etapa deste trabalho de conclusão de curso pode ser vista como o resultado geral desta revisão sistemática, onde conceitos, termos, abordagens e qualquer informação utilizada no trabalho é fruto, seja direta ou indiretamente, desta revisão sistemática.

Com o objetivo de documentar os resultados diretos desta revisão sistemática, estes foram registrados como as técnicas utilizadas atualmente para solucionar o problema de SLAM em diferentes contextos.

Com a realização desta revisão, foi possível identificar *áreas* mutáveis nas diferentes soluções do problema de SLAM. As áreas identificadas são *"arquitetura da solução"* A.0.3.1, *"técnica probabilística utilizada"* A.0.3.2 e *"informações disponíveis"* A.0.3.3. Em cada solução, os autores buscaram se adequar ao contexto trabalhado, seja a partir da disponibilidade de sensores específicos ou da capacidade computacional disponível.

A.0.3.1 Arquitetura da Solução

A arquitetura da solução, na grande maioria dos estudos, foi definida a partir do requisito computacional. Ou seja, a limitação computacional presente nos robôs simples levou os autores a buscarem arquiteturas que contornassem esse problema, como mostra [Lindhorst, Lukas e Nett 2013]. Entre as diversas arquiteturas, as mais comumente utilizadas são as que buscam processar as informações em um computador, considerando o robô apenas para obtenção das informações, como ilustra a Figura 34.

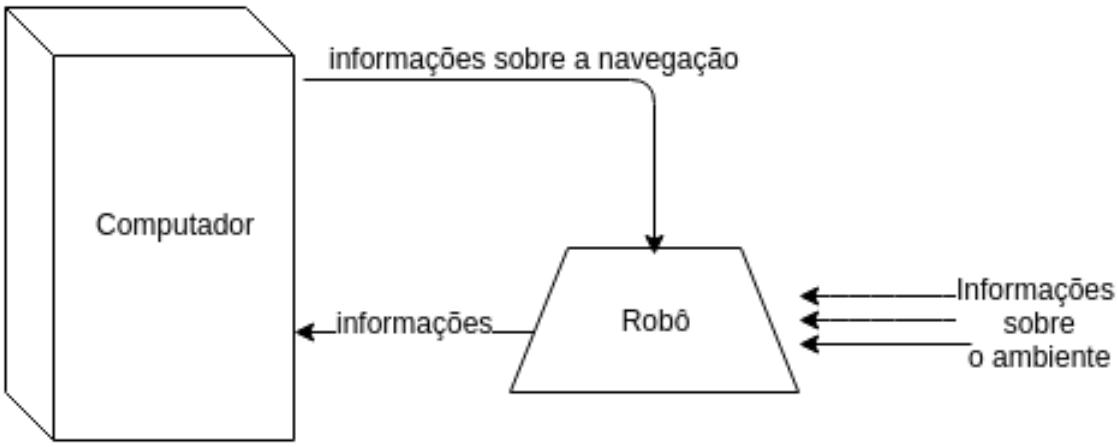

Figura 34 – Arquitetura de comunicação

De acordo com a Figura 34, o robô será responsável apenas por obter informações do ambiente, ou seja, recuperar os dados obtidos a partir dos sensores disponíveis. O computador, em posse das informações sobre o ambiente, ou seja, os pontos de referência, a quantidade de rotações em cada roda, a distância e cores de objetos, por exemplo, será responsável por processar toda a informação, construindo um mapa lógico para possibilitar a localização do robô em relação ao ambiente, como apresenta [Lindhorst, Lukas e Nett 2013].

Geralmente, são utilizados dois mapas simultâneos, um presente no robô (local) e outro, mais completo, presente no computador (remoto). O mapa local é, basicamente, um vetor de pontos *'p'* em relação a um tempo *'t'*, como explica [Adams, Mullane e Vo 2013].

O computador utiliza este mapa local, que é disponibilizado pelo robô, para completar, corrigir e atualizar o mapa remoto, mesclando informações e utilizando, geralmente, filtros probabilísticos para maximizar sua precisão [Oliveira 2008].

As decisões referentes à navegação são geradas a partir da análise do mapa remoto, já que o mapa local é incompleto e inconsistente, como afirma [Lindhorst, Lukas e Nett 2013]. A utilização desta arquitetura de mapeamento remoto e local torna prática a realização de navegações com múltiplos robôs, como apresenta, [Lindhorst, Lukas e Nett 2013], em seu trabalho sobre rede de comunicação sem fio para navegação de múltiplos robôs.

Seguindo esta arquitetura, [Saeedi et al. 2011] e [Wang, Huang e Dissanayake 2007] também desenvolveram sistemas de resolução do problema de SLAM com a utilização de múltiplos robôs, mostrando a viabilidade da sua utilização. Neste tipo de trabalho, os robôs colaboram entre si, pois, como a informação obtida é centralizada em um computador único, as decisões referentes à navegação de um determinado robô são resultados do processamento das informações obtidas por todos os robôs, maximizando a visão global de cada robô [Lindhorst, Lukas e Nett 2013].

De qualquer forma, independente da arquitetura da solução utilizada, os erros advindos dos sensores sempre serão um problema sério a ser resolvido [Adams, Mullane e Vo 2013]. Com o objetivo de solucionar este problema, a comunidade de robótica vê-se presa à utilização de estruturas matemáticas probabilísticas, como afirma [Adams, Mullane e Vo 2013]. As seções 2.2.1, 2.2.2 e A.0.3.2 apresentam as estruturas probabilísticas mais utilizadas atualmente, assim como suas vantagens e desvantagens.

A.0.3.2 Técnica Probabilistica Utilizada

Para a realização de uma navegação específica, primeiramente, o robô deverá obter informações sobre o ambiente [Romano 2002]. Para isso, é necessária a utilização de sensores que captem estas informações, seja a partir de odometria, infra-vermelho ou vídeo. Entretanto, estes sensores são munidos de uma margem de erro que, muitas vezes, prejudica a navegação e a auto-localização como um todo [Costa e Okamoto Jr. 2002]. De acordo com [Adams, Mullane e Vo 2013]:

> *Devido à natureza imperfeita dos sensores, à falta da previsibilidade em ambientes reais e à necessidade de aproximações para alcançar decisões computacionais, a robótica é uma ciência que depende de algorítmos probabilisticos.*

Desse modo, a comunidade de robótica vem buscando, na matemática, soluções probabilísticas que minimizem esta margem de erro, como afirma [Fuentes-Pacheco, Ruiz-Ascencio e Rendón-Mancha 2012]. Entre as soluções mais utili-

zadas, encontram-se, no topo da lista, os filtros de *Kalman* e *partículas*, como pode ser observado nas seções 2.2.1 e 2.2.2, respectivamente.

Alguns autores, como, por exemplo, [Neto 2015], vêm estudando as duas técnicas e comparando-as com o intúito de selecionar a *"melhor"* técnica. Entretando, o termo *"melhor"* é relativo, neste caso, a depender do contexto em que será aplicado o filtro probabilístico, como afirma [Neto 2015].

O estudo de [Neto 2015] exemplifica, com clareza, as comparações entre os filtros de partícula e de Kalman, ao utilizar as duas técnicas para solucionar o problema de SLAM com um robô móvel munido de sensores a laser. De acordo com a relatividade do termo *"melhor"*, nesta ocasião, o autor buscou apresentar apenas algumas vantagens e desvantagens dos dois filtros, como pode-se observar nas seções 2.2.1 e 2.2.2.

A.0.3.3 Informações Disponíveis

Na robótica móvel, existem diversas maneiras de se obter informações sobre o ambiente, a partir da utilização de sensores específicos para informações específicas [Costa e Okamoto Jr. 2002]. Entre os sensores mais utilizados, encontram-se os *sonares, sensores infra-vermelho, câmeras de vídeo, sensores de distância, sensores RGB e sensores odométricos*. A partir do ambiente em que se deseja navegar, faz-se necessária a seleção dos sensores adequados para o mesmo.

Além do tipo de sensor escolhido, deve-se levar em consideração algumas características do mesmo, como o alcance, desempenho e precisão, por exemplo. Conhecer a margem de erro dos sensores é essencial, como afirma [Katsev et al. 2011].

Como foi explicado no tópico A.0.3.2, os sensores possuem margens de erro que podem prejudicar a navegação e a auto-localização do robô. Em busca de tentar solucionar este problema, além da utilização de filtros probabilísticos, diversos autores buscam utilizar múltiplos sensores, integrando as informações dos mesmos para minimizar a margem de erro na informação, como mostram [Wang, Liu e Li 2014], [Saeedi et al. 2011] e [Dissanayake et al. 2001].

De acordo com [Machado 2003], há três tipos de integração entre sensores: *complementar*, onde os sensores têm apenas uma visão parcial do ambiente, unindo as informações para obter uma visão mais completa; *competitiva*, onde dois ou mais sensores competem para obter a informação com maior precisão, como a medição da distância de um único objeto utilizando sonar e sensor a laser, por exemplo; e *cooperativa*, quando os sensores cooperam entre si para obter informações que não seriam possíveis com a utilização de apenas um, como a geração de informações em três dimensões, por exemplo.

Entre os sensores citados anteriormente, o sensor odométrico, geralmente, é utilizado em qualquer sistema de navegação sobre eixos, de acordo com

[Maimone, Cheng e Matthies 2007]. Este sensor recupera os dados sobre rotações realizadas nas rodas em que o mesmo se encontra, possibilitando a mensuração da distância percorrida, assim como a identificação de curvas, por exemplo [Lin e Tsai 2008].

Entretanto, como afirma [Huang, Wang e Dissanayake 2009], sensores odométricos geram erros acumulativos, ou seja, quanto maior for a navegação, maior será o erro total do sensor. Trabalhar com sensores assim gera uma certa dificuldade. Entretanto, com a utilização de filtros probabilísticos, como apresentado na seção A.0.3.2, este erro pode ser reduzido, viabilizando sua utilização.

Já a utilização de câmeras de vídeo, segundo [Machado 2003], além de necessitar de *landmarks* verificáveis através do processamento de imagens, as mesmas exigem requisitos computacionais que, muitas vezes, acabam por inviabilizar sua utilização.

O sensor ultrasônico, que foi utilizado durante a segunda etapa deste trabalho, diferentemente das câmeras de vídeo, são simples, rápidos e baratos, de acordo com [Machado 2003]. Entretanto, sua utilização acaba por limitar o contexto de navegação, devido ao seu limite de alcance. Ou seja, em um ambiente aberto, o sensor ultrasônico não será capaz de identificar obstáculo algum. Além da sua limitação de alcance, [Machado 2003] apresenta algumas características que podem problematizar sua utilização:

1. *pobre direcionalidade*: não são capazes de identificar a direção do obstáculo identificado, apenas sua distância;

2. *passíveis de ruídos*: ruídos são comuns durante a utilização de sonares, seja devido a interferências de outros sensores, ou a partir da sua margem de erro padrão, por exemplo.

3. *reflexão especular*: O sensor emite o sinal sonoro em formato de cone, possibilitando a obtenção de erros relacionados à angulação de incidência.

Com foi definido por [Machado 2003], umas das características importantes da utilização do sonar é a reflexão especular, devido a sua emissão em formato de cone, como é apresentado de forma clara na Figura 35.

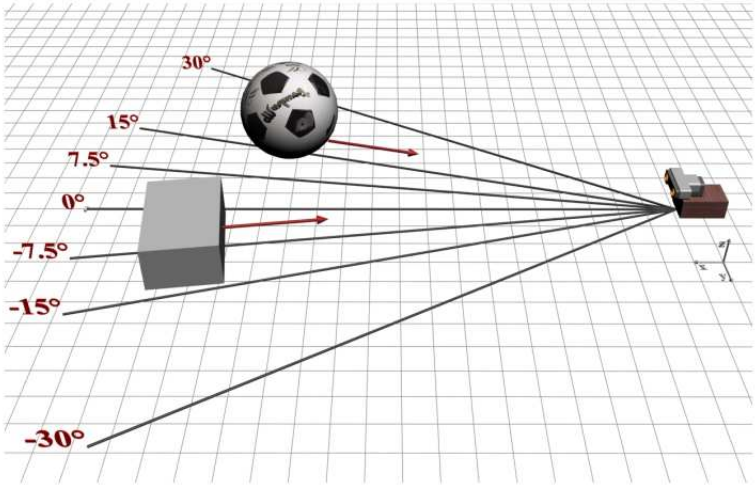

Figura 35 – Emissão do Sinal Sonoro - Sonar

Outro detalhe importante a ser analisado, segundo [Machado 2003], é a velocidade de emissão do sinal. Esta deve ser relacionada a necessidade de obtenção das informações, ou seja, dependendo da velocidade de navegação atual do robô, a velocidade de emissão deve ser atualizada. Desse modo, se o robô se movimenta rapidamente, devem ser gerados sinais a uma frequência alta, possibilitando a reação do robô a eventos inesperados [Machado 2003].

Com a emissão de sinais sonoros enquanto o robô se movimenta, como foi sugerido anteriormente, deve-se levar em consideração o atraso no mapeamento, já que, ao obter o retorno do sinal, o robô já não se encontra no mesmo local [Machado 2003], inserindo erros na navegação, o que pode prejudicá-la.

Outro problema bastante recorrente com a utilização de sonares é a identificação de cantos em paredes, por exemplo. Como a emissão do sinal é feita no formato de cone, quando a extremidade do cone alcança a parede, a mesma é refletida fornecendo a informação ao robô. O mesmo acontece do outro lado do cone, deixando o centro invisível ao sinal sonoro. O impacto deste problema é proporcional a distância entre o robô e o objeto, como já foi explicado ao longo do trabalho.

A Tabela 20 apresenta de maneira organizada a contabilização dos resultados da revisão, de acordo com as áreas destacadas anteriormente. Encontram-se nesta Tabela apenas os artigos que solucionaram, de alguma maneira, o problema de SLAM.

Tabela 20 – Contabilização de Estratégias (SLAM)

Arquitetura de processamento	**Filtro**	***Inputs***	**Nº. Artigos**
Processamento remoto	Kalman	laser, vídeo, odometria, sonar, infra-vermelho, câmera término, kinect RGB, sonar biomimético.	12
	Partículas	Câmera de vídeo,odometria, laser, sonar,magnetômetro, sensor de rádio.	5
Processamento local	Kalman	sensor de distância (laser), odometria, identificador de landmarks (simulado) e sonar.	3
	Partículas	Sonar e odometria	1
Processamento remoto + múltiplos robôs	Kalman	laser (range), vídeo e odometria	3
	Partículas	0	0
Total			24

APÊNDICE B – Prova de Conceito

Com o objetivo de obter conhecimento referente à solução prática proposta durante este trabalho, foi desenvolvida uma pequena prova de conceito durante a primeira etapa deste trabalho (TCC_1). Nesta seção, estão apresentadas as informações sobre o planejamento e condução da prova de conceito, assim como as informações importantes obtidas com a realização da mesma, caracterizadas como *resultados da prova de conceito*.

Esta seção está dividida entre *planejamento e condução* da prova de conceito, B.1, e *características técnicas* B.2.

B.1 Planejamento e Condução

Esta prova de conceito foi pensada com o objetivo de identificar algumas características básicas sobre a implementação de sistemas de navegação inteligente utilizando os kits de robótica Mindstorms, da Lego. Com este objetivo, a prova de conceito envolveu a análise de ferramentas utilizadas, como sensores e atuadores, o estudo e seleção da linguagem de programação utilizada e alguns detalhes referentes à localização relativa de obstáculos.

Para isso, buscou-se simular uma atividade básica na navegação e auto-localização na robótica. Atividade esta, que tem como objetivo identificar obstáculos ao redor do robô, levando em consideração a margem de erro, assim como todas as características presentes nos sensores e atuadores do kit. Dessa forma, o robô é colocado em um ambiente desconhecido, onde o mesmo busca identificar obstáculos à sua volta, com o objetivo de definir uma saída sem obstáculo algum.

O ambiente utilizado para a prova de conceito pode ser observado na Figura 36. A montagem do robô utilizada nesta prova seguiu o estabelecido na seção 5.3, utilizando esteiras, um sonar para identificar a distância de obstáculos, um sensor de toque e sensores odométricos em cada motor.

Figura 36 – Ambiente Prova de Conceito

Com a realização da prova de conceito, ao implementar questões referentes ao posicionamento, direcionamento e localização do robô, diversas características foram identificadas e analisadas. Ao longo da seção B.2 estão apresentadas algumas destas características técnicas.

B.2 Características Técnicas

Neste tópico, serão apresentadas algumas das características importantes identificadas durante a realização da prova de conceito.

B.2.1 Seleção da Linguagem e Ambiente de Desenvolvimento

Para realização desta prova de conceito, assim como o desenvolvimento de toda a solução proposta, ao longo deste trabalho de conclusão de curso, optou-se pela utilização da linguagem Java. Esta escolha deu-se não só devido à possibilidade, com mais facilidade, da integração desta solução com o *framework* desenvolvido por [Rincon 2015], como também por todo o apoio técnico advindo da utilização da ferramenta *leJOS NXJ*[1].

[1] http://www.lejos.org/nxj.php

A ferramenta leJOS NXJ engloba, basicamente, uma máquina virtual Java (JVM) desenvolvida em C, sendo multiplataforma, ou seja, é portável para sistemas Linux, Windows e Macintosh [Bagnall 2011]. O material de estudo sobre a ferramenta leJOS NXJ encontra-se disponível em toda a *web*, de maneira livre, e no livro [Bagnall 2011], que foi utilizado como fonte de informação durante o desenvolvimento.

O sistema operacional utilizado para realização desta prova de conceito foi o *windows*, devido a sua facilidade de configuração da comunicação entre robô/PC, diferentemente do observado em sistemas Linux. O *passo-a-passo* para instalação e configuração do ambiente utilizado para desenvolvimento encontra-se em [Bagnall 2011, p. 6].

A seleção desta ferramenta é decorrente, além dos motivos apresentados anteriormente, da existência de uma *API* que disponibiliza diversas funcionalidades referentes à navegação dos robôs. A *API* do leJOS NXJ engloba desde funções referentes ao controle de motores, até o apoio à criação de mapas e sistemas complexos de navegação.

Em paralelo à escolha da linguagem de programação, buscou-se identificar um ambiente de desenvolvimento que disponibilizasse ferramentas de apoio que facilitem o desenvolvimento da solução. Com este objetivo, optou-se pela utilização da *IDE* Eclipse[2], por possuir um *plugin* da ferramenta leJOS NXJ, como apresenta [Bagnall 2011]. O tutorial para instalação e configuração do ambiente utilizado encontra-se em [Bagnall 2011, p. 14].

B.2.2 Atuadores

Os atuadores, ou motores, contemplam a base da robótica móvel. Desse modo, o leJOS NXT disponibiliza diversas funcionalidades referentes ao controle destes atuadores. Um exemplo disso é a possibilidade de controlar a aceleração do motor, possibilitando um arranque com aceleração gradual, para minimizar as chances de derrapagem, por exemplo.

Estas possibilidades de controle de rotação são devido a existência de *encoders* óticos em cada motor, como apresenta [Bagnall 2011]. Cada *encoder* tem como objetivo registrar as rotações de cada eixo, possibilitando a navegação por odometria, como já foi explicado ao longo do trabalho.

Para acessar os atuadores, o leJOS NXJ oferece, como principal fonte de acesso, a classe Motor, que possui três instâncias estáticas: *Motor.A*, *Motor.B* e *Motor.C*. O livro [Bagnall 2011] faz uma análise detalhada de todos os métodos presentes nesta classe, os quais são, principalmente, voltados à aceleração e velocidade de rotação dos eixos.

[2] https://eclipse.org/

B.2.3 Sensores

Cada computador central do kit Mindstorms possui quatro portas para sensores, ou seja, a solução deste trabalho só pode envolver no máximo quatro sensores, que fazem parte do kit Mindstorm. Para esta prova de conceito, foram utilizados os sensores de *odometria*, a partir dos *encoders* em cada atuador, um sensor ultrasônico, como sensor de distância e um sensor de toque.

- **Sensor de toque:**

  É o sensor mais básico do kit, que retorna um valor *booleano* que indica se o sensor está pressionado ou não, a partir do botão laranja apresentado na Figura 37.

Figura 37 – Sensor de Toque.

A classe *TouchSensor*, que implementa a interface deste sensor, contém apenas um simples método:

```
1                    boolean isPressed();
```

- **Sensor ultrasônico:**

  Como apresenta a Figura 38, o sensor ultrasônico lembra bastante um par de olhos, apesar de possuir muitas características em comum com um sensor de som, em vez de uma câmera, por exemplo. Isso se dá pela estratégia de funcionamento do sensor, o qual emite um sinal sonoro que reflete em obstáculos à frente, retornando ao sensor. A partir da análise do tempo percorrido pelo sinal sonoro, é possível estimar a distância do objeto analisado, em relação ao robô.

Figura 38 – Sensor Ultrasônico.

Este sensor é capaz de identificar distâncias de até dois metros e meio, mais especificamente 255 centímetros. Porém, sua utilização em distâncias tão grandes não é recomendada por [Bagnall 2011], devido à grande margem de erro presentes em medições como esta. De acordo com [Bagnall 2011], este sensor possui, em distâncias de aproximadamente 180 centímetros, uma margem de erro de mais ou menos três centímetros (*+/- 3*). Sua margem de erro é proporcional à distância entre robô e obstáculo.

Outra característica importante do funcionamento deste sensor é a emissão de sinais no formato de cone, como apresenta a Figura 39[3].

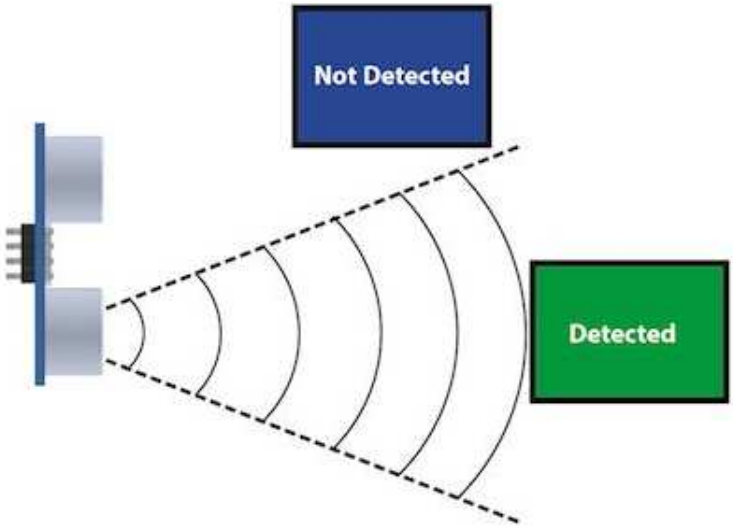

Figura 39 – Emissão do Sinal Ultrasônico.

O cone formado pela emissão do sinal segue uma angulação de 30º, ou seja, em uma distância de 180 centímetros, o cone possui um diâmetro de 90 centímetros. Desse modo, deve-se levar em consideração a incapacidade de identificar pequenas irregularidades em obstáculos, como fendas e buracos.

Durante o desenvolvimento da prova de conceito, observou-se com mais precisão a característica descrita anteriormente, levando à alteração da proposta do trabalho, em relação ao ambiente utilizado. Inicialmente, o ambiente proposto baseava-se no tapete de missões *Nature's Fury*, utilizado como um dos desafios do torneio *First Lego League*[4]. Um exemplo deste ambiente pode ser visualizado na Figura 40.

[3] http://arcbotics.com/products/sparki/parts/ultrasonic-range-finder/
[4] http://www.firstlegoleague.org/

Figura 40 – Ambiente utilizado.

Os problemas referentes aos cantos, buracos e fendas encontrados durante a realização da prova de conceito mostraram o grande impacto desta característica do sonar no mapeamento de ambientes pequenos, como o utilizado. Desse modo, optou-se por modificar o ambiente a ser utilizado durante a segunda etapa deste trabalho. O ambiente será baseado em cômodos reais, como quartos, salas ou escritórios. O critério inicial para utilização do ambiente é referente a área de navegação disponivel, devendo ser de, pelo menos, 1,5 metros quadrados.

B.2.4 Computador Central NXT (Brick)

O robô utilizado durante esta prova de conceito e durante o desenvolvimento da solução proposta, é da família Mindstorm NXT, da Lego. O computador central, de acordo com [Bagnall 2011], possui uma área de 7,2 x 11,2 centímetros, com um processador *Atmel 32-bit ARM* de 48 MHz de frequência, memória RAM de 64 KB e 256 KB de memória *flash*.

Estas limitações de memória e processamento foram contornadas a partir da utilização da arquitetura de processamento remoto, como foi definido na seção 5.6, já como um resultado da revisão sistemática apresentada na seção A.

APÊNDICE C – Cenários de Teste

Este apêndice tem como objetivo registrar os dados obtidos durante a execução dos cenários de teste. Os cenários de teste estão organizados por grupos, chamados de "cados de teste". Ao longo do apêndice, estão presentes imagens do processo de auto-localização e do resultado da localização, como pode ser observado a seguir.

C.1 Caso de Teste 1

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 41 a Figura 51.

C.1.1 Cenário 1

Cenário utilizando 200 partículas:

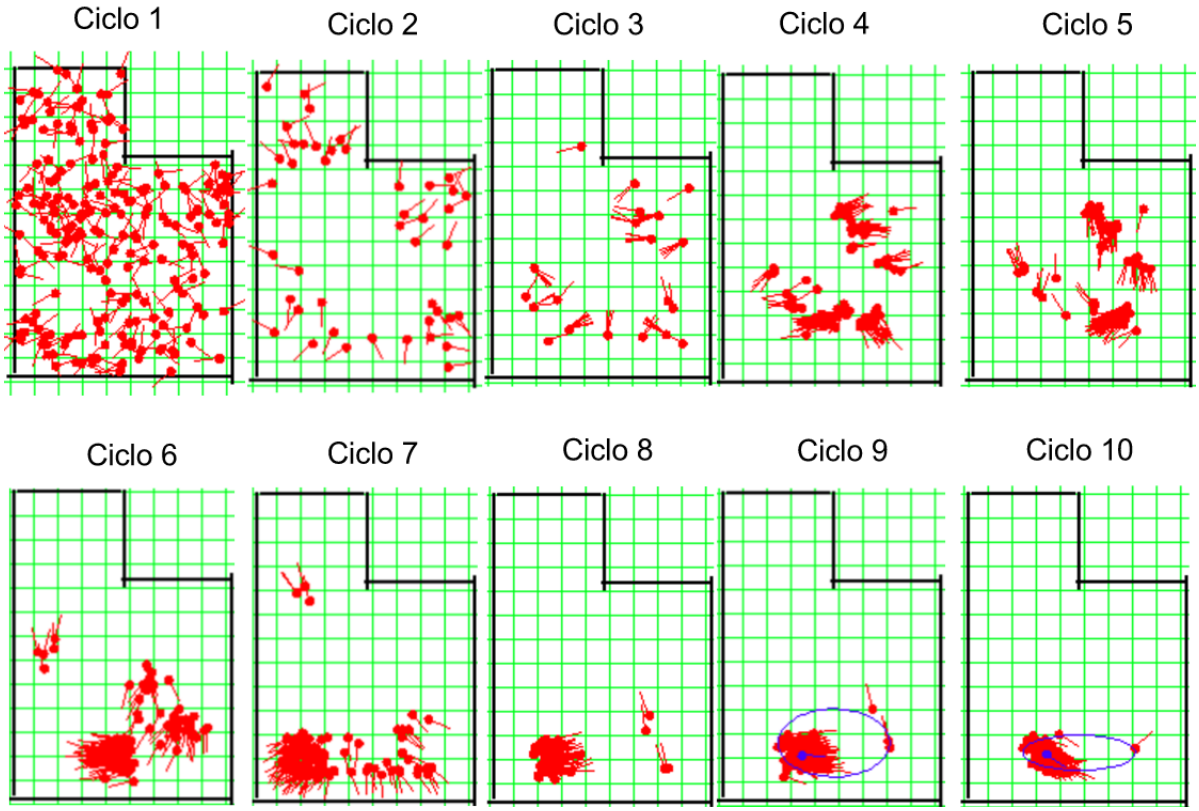

Figura 41 – Caso de Teste 1 - Cenário 1.

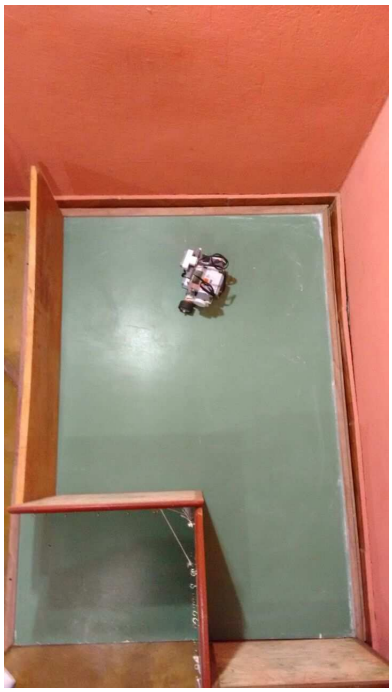

Figura 42 – Posição Real do Caso de Teste 1 - Cenário 1.

C.1.2 Cenário 2

Cenário utilizando 400 partículas:

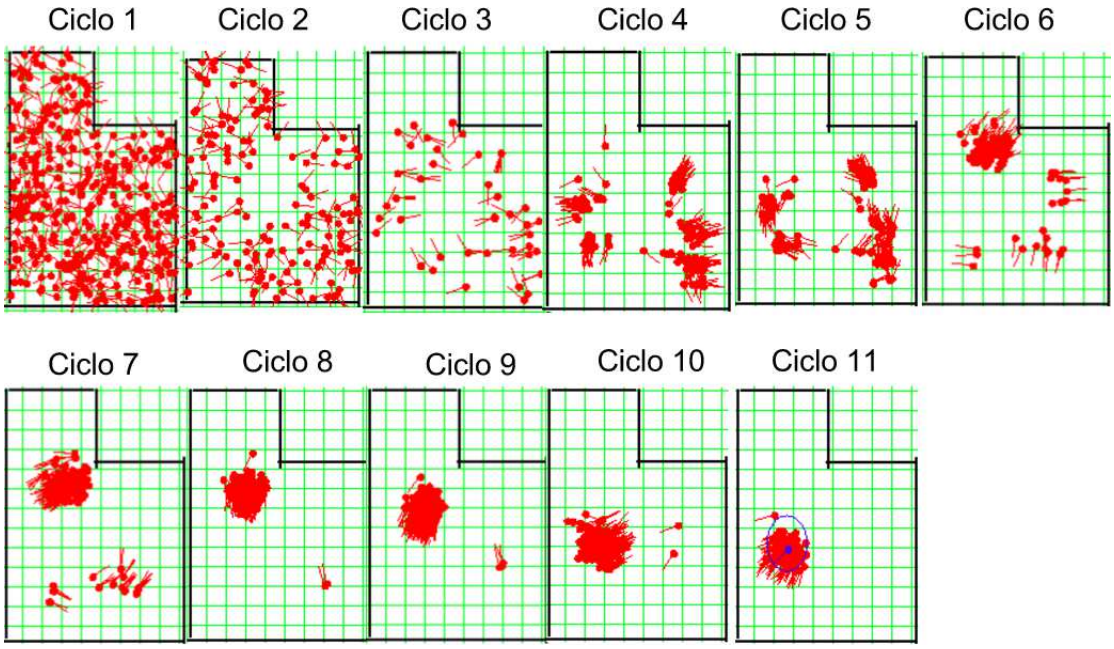

Figura 43 – Caso de Teste 1 - Cenário 2.

Figura 44 – Posição Real do Caso de Teste 1 - Cenário 2.

C.1.3 Cenário 3

Cenário utilizando 500 partículas:

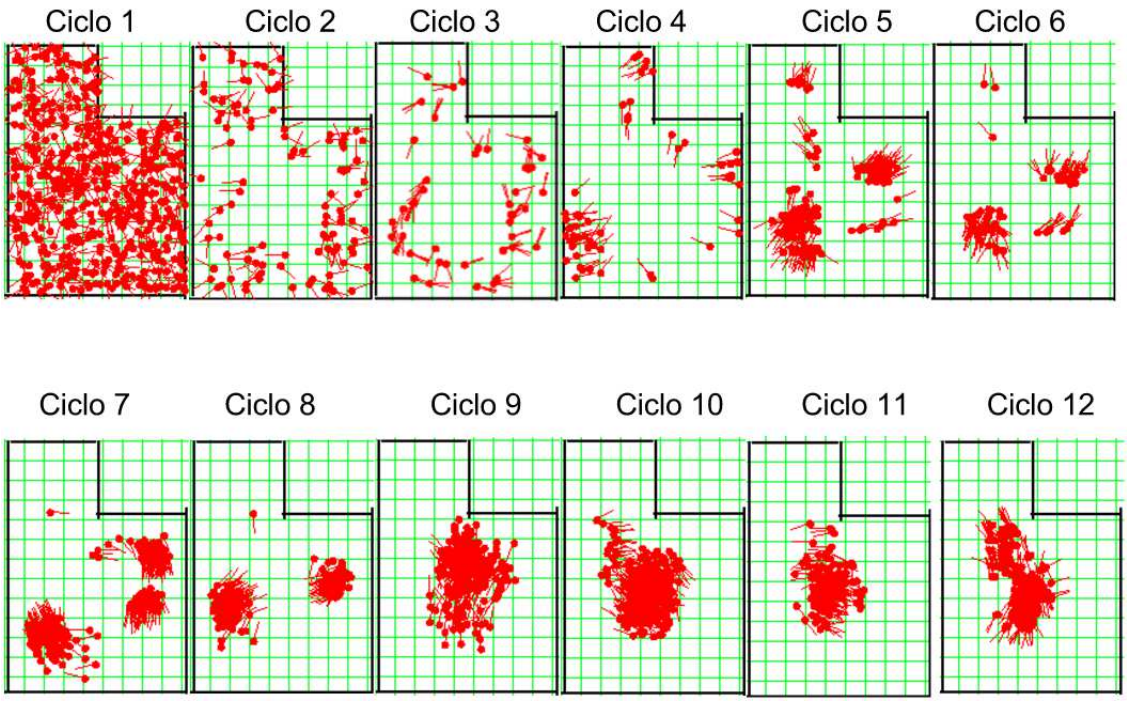

Figura 45 – Caso de Teste 1 - Cenário 3 Parte 1.

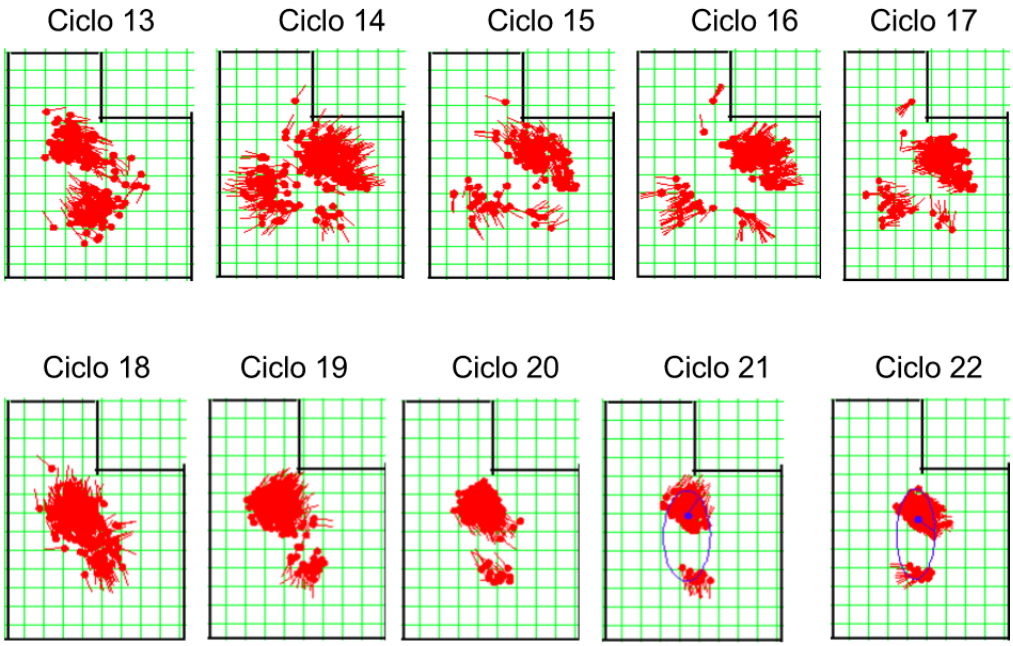

Figura 46 – Caso de Teste 1 - Cenário 3 Parte 2.

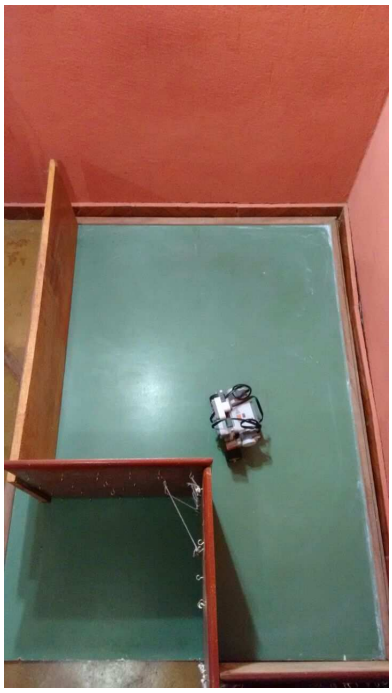

Figura 47 – Posição Real do Caso de Teste 1 - Cenário 3.

C.1.4 Cenário 4

Cenário utilizando 100 partículas:

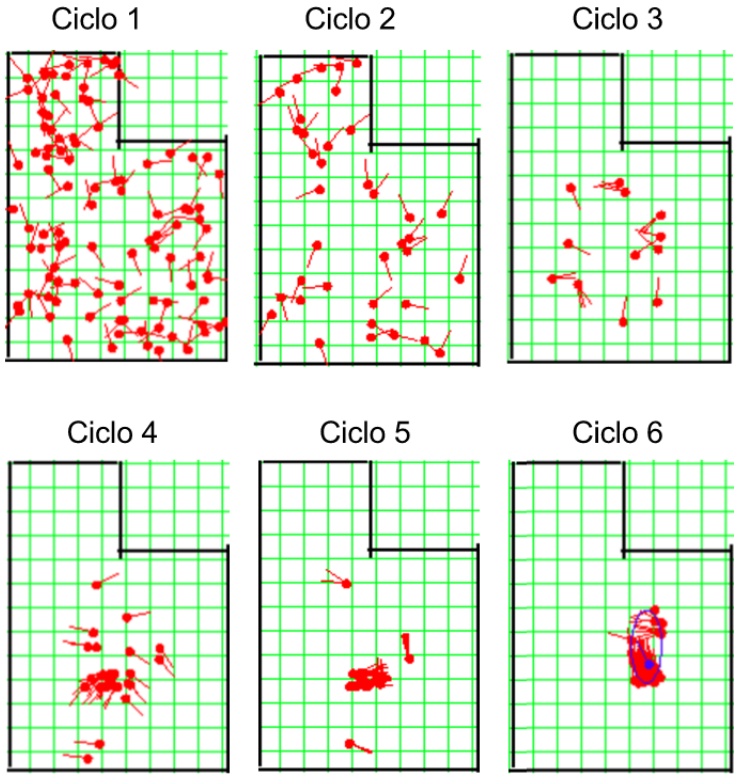

Figura 48 – Caso de Teste 1 - Cenário 4.

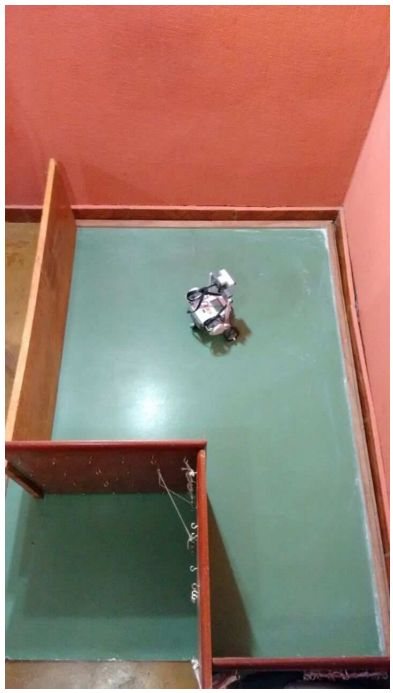

Figura 49 – Posição Real do Caso de Teste 1 - Cenário 4.

C.1.5 Cenário 5

Cenário utilizando 150 partículas:

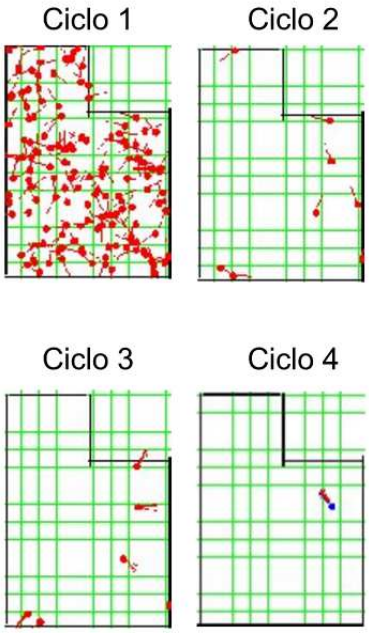

Figura 50 – Caso de Teste 1 - Cenário 5

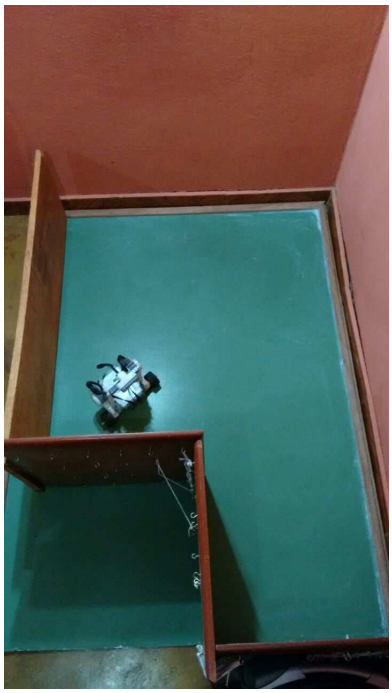

Figura 51 – Posição Real do Caso de Teste 1 - Cenário 5.

C.2 Caso de Teste 2

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 52 a Figura 61.

C.2.1 Cenário 1

Cenário utilizando velocidade de rotação em 10 graus por segundo:

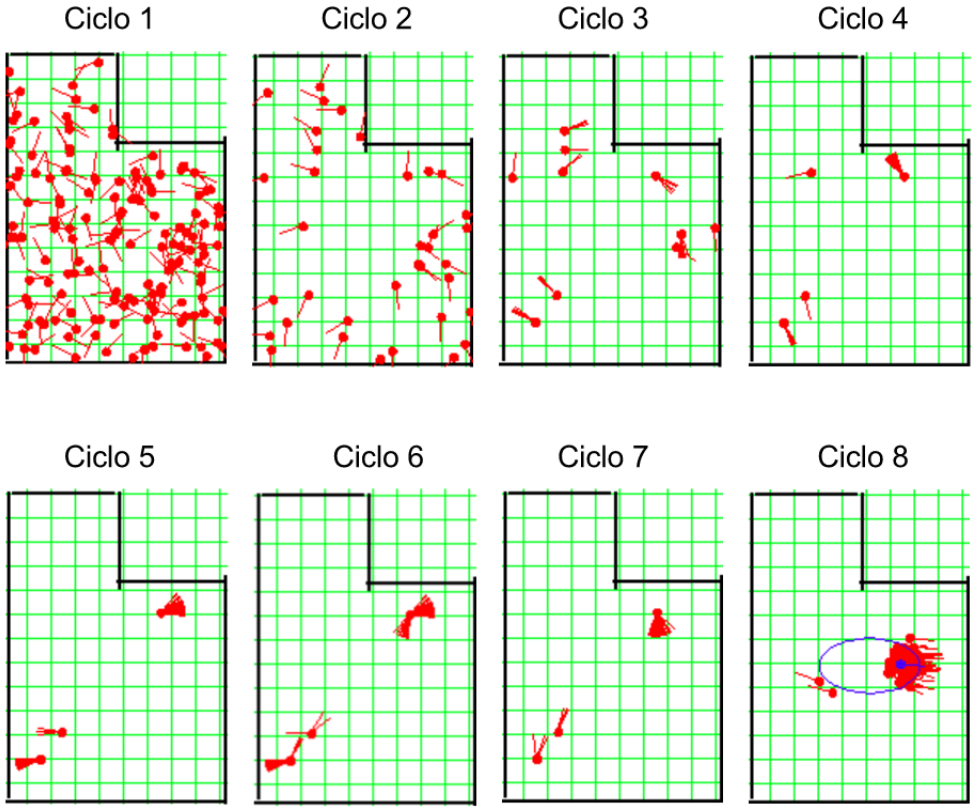

Figura 52 – Caso de Teste 2 - Cenário 1

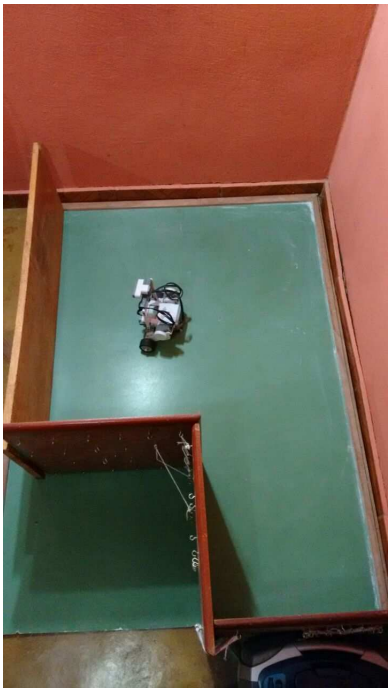

Figura 53 – Posição Real do Caso de Teste 2 - Cenário 1.

C.2.2 Cenário 2

Cenário utilizando velocidade de rotação em 30 graus por segundo:

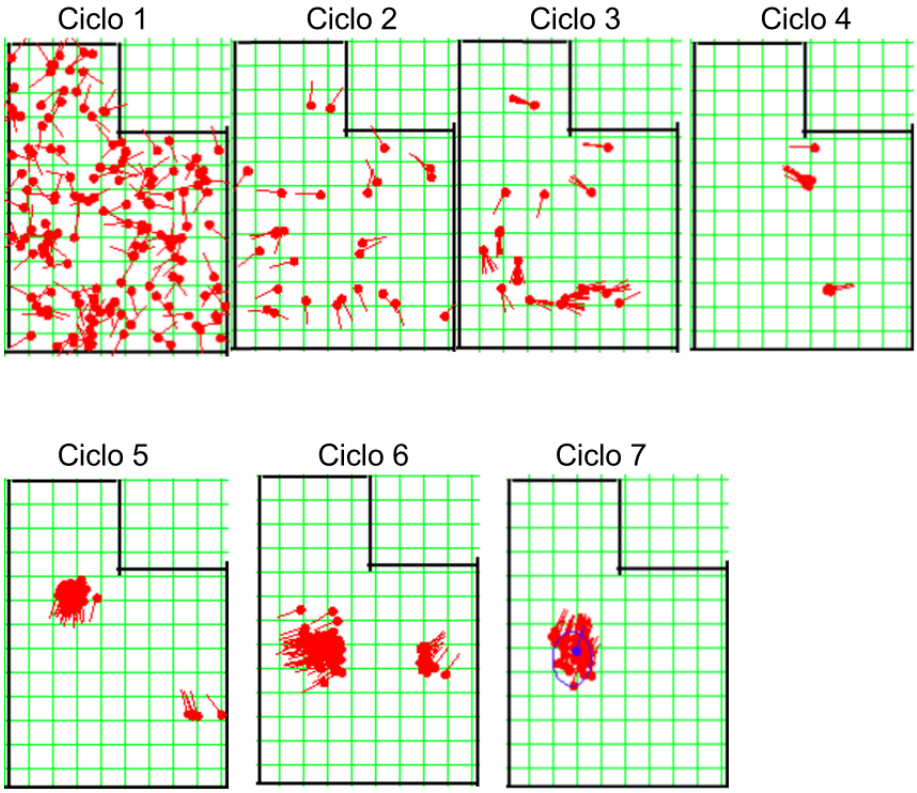

Figura 54 – Caso de Teste 2 - Cenário 2

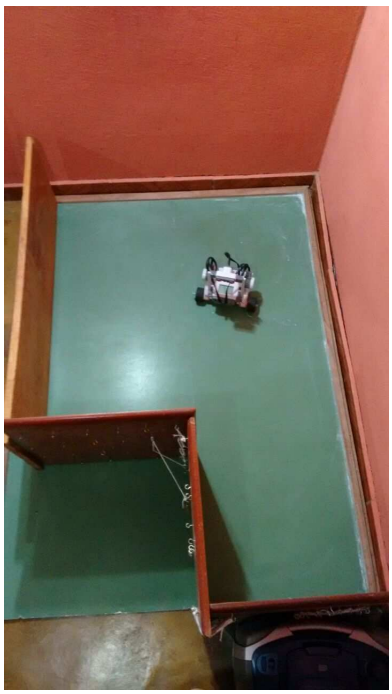

Figura 55 – Posição Real do Caso de Teste 2 - Cenário 2.

C.2.3 Cenário 3

Cenário utilizando velocidade de rotação em 50 graus por segundo:

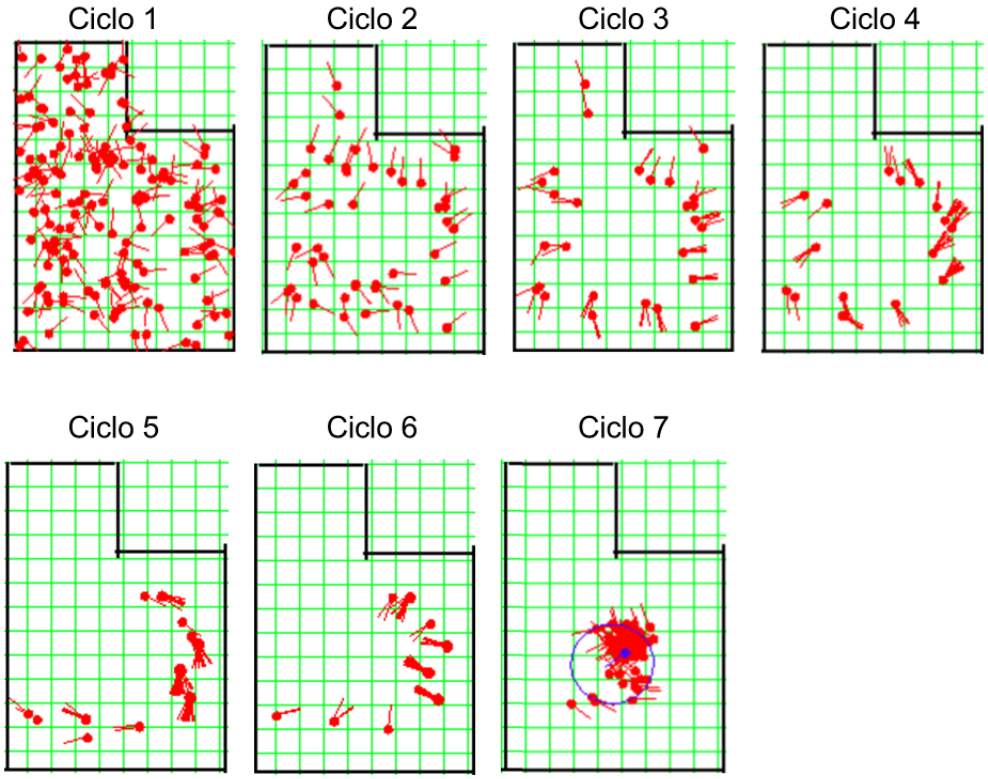

Figura 56 – Caso de Teste 2 - Cenário 3

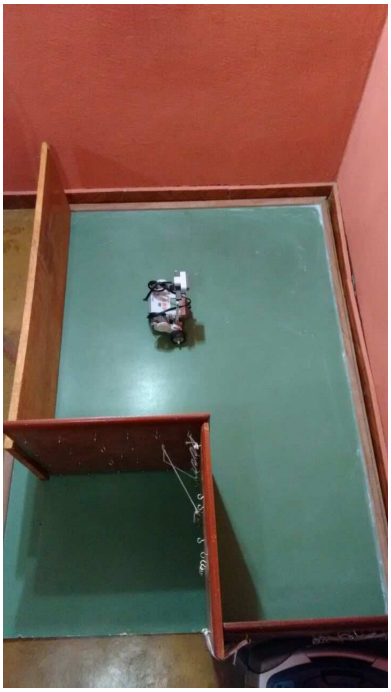

Figura 57 – Posição Real do Caso de Teste 2 - Cenário 3.

C.2.4 Cenário 4

Cenário utilizando velocidade de rotação em 70 graus por segundo:

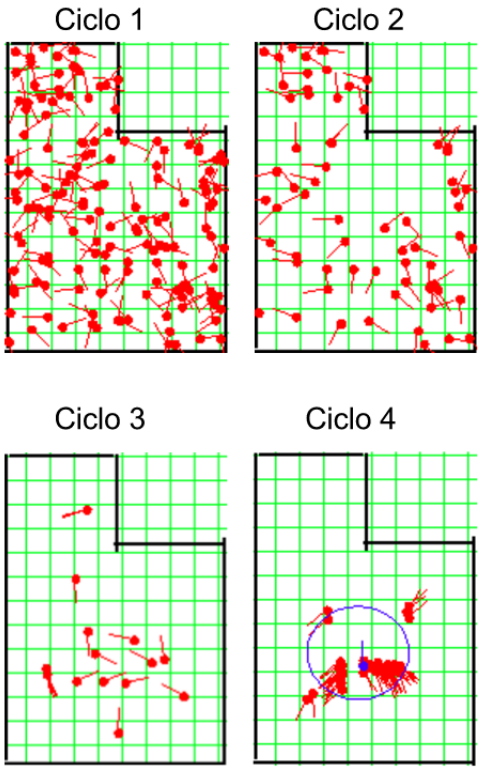

Figura 58 – Caso de Teste 2 - Cenário 4

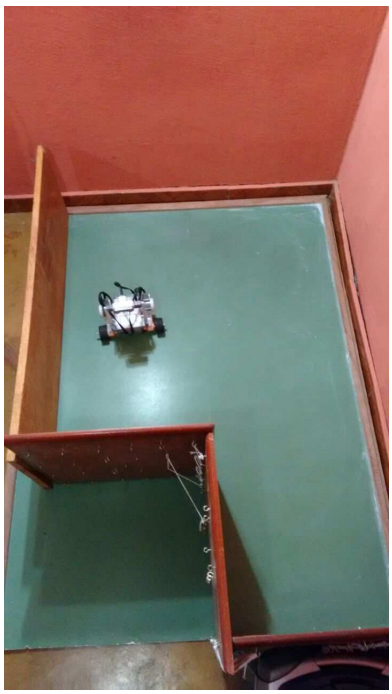

Figura 59 – Posição Real do Caso de Teste 2 - Cenário 4.

C.2.5 Cenário 5

Cenário utilizando velocidade de rotação em 90 graus por segundo:

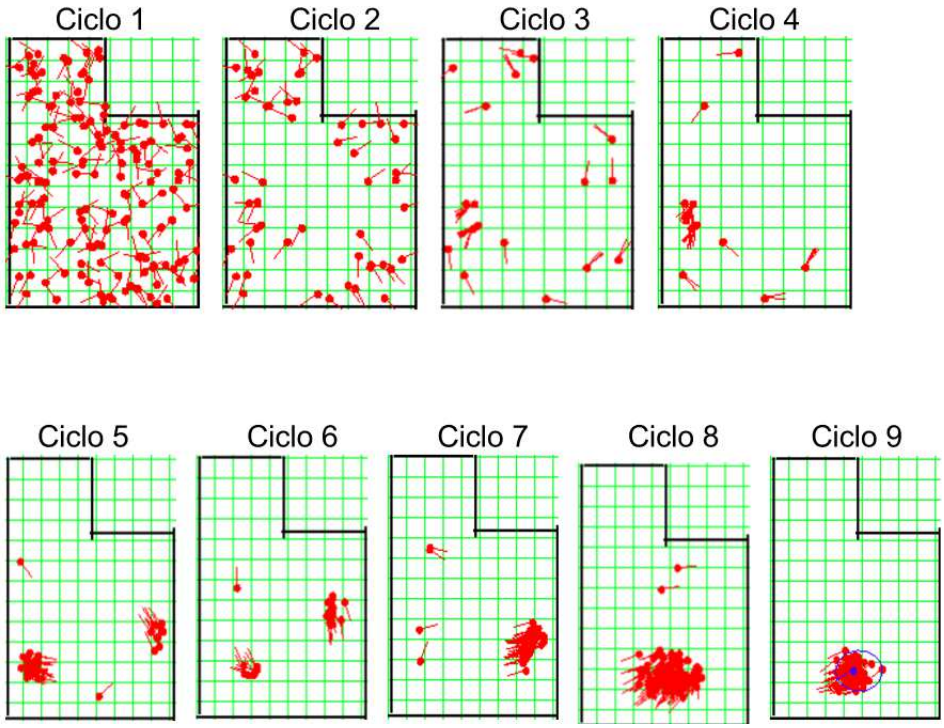

Figura 60 – Caso de Teste 2 - Cenário 5

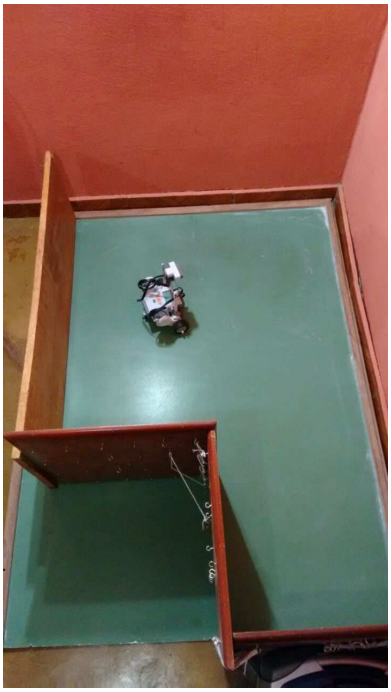

Figura 61 – Posição Real do Caso de Teste 2 - Cenário 5.

C.3 Caso de Teste 3

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 62 a Figura 71.

C.3.1 Cenário 1

Cenário utilizando velocidade de deslocamento em 2 unidades de diâmetro por segundo:

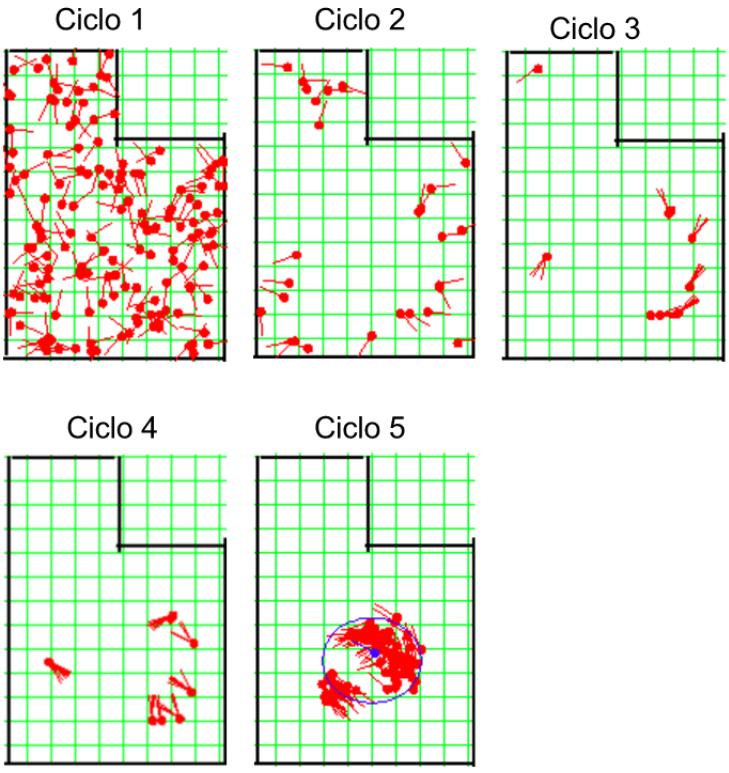

Figura 62 – Caso de Teste 3 - Cenário 1

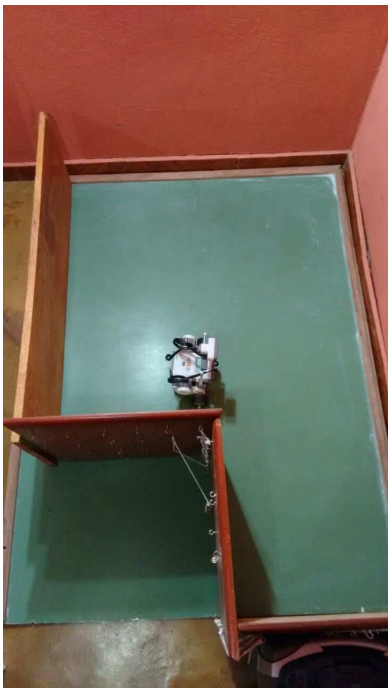

Figura 63 – Posição Real do Caso de Teste 3 - Cenário 1.

C.3.2 Cenário 2

Cenário utilizando velocidade de deslocamento em 5 unidades de diâmetro por segundo:

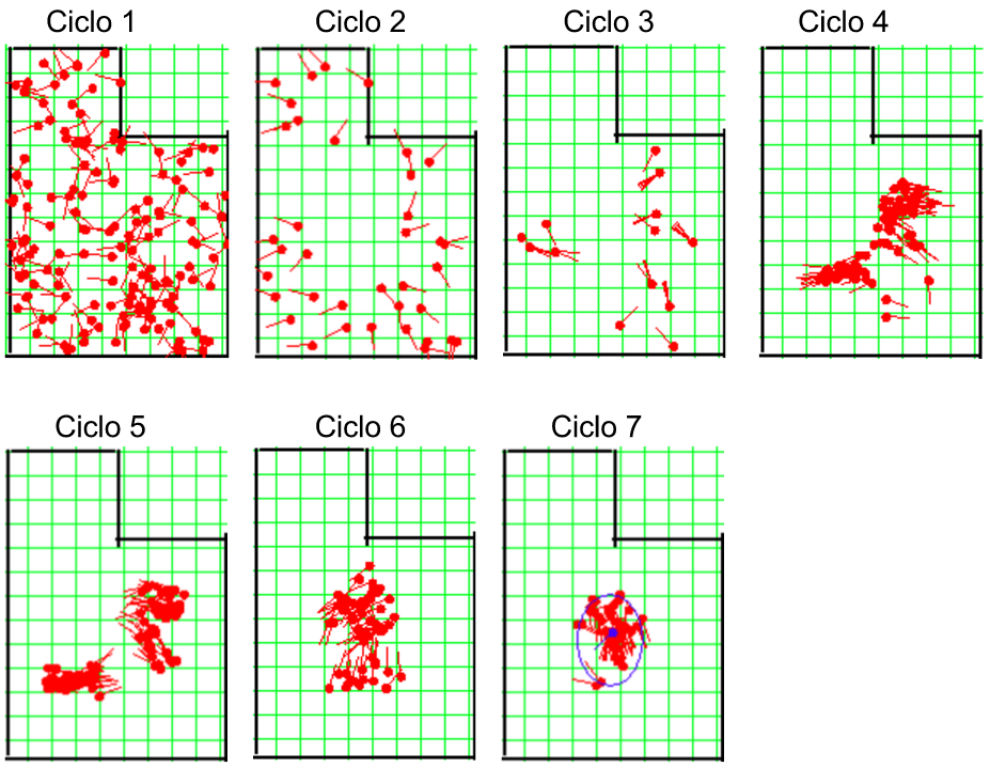

Figura 64 – Caso de Teste 3 - Cenário 2

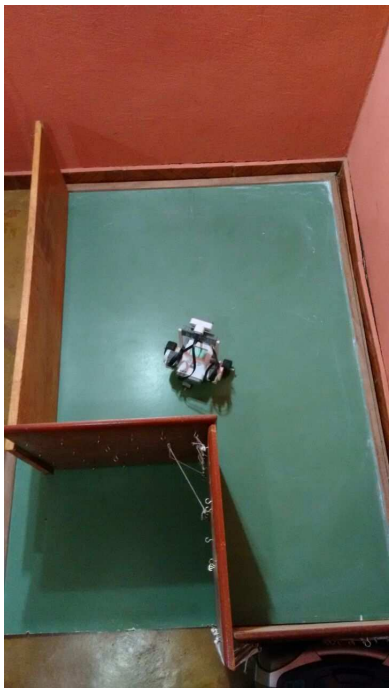

Figura 65 – Posição Real do Caso de Teste 3 - Cenário 2.

C.3.3 Cenário 3

Cenário utilizando velocidade de deslocamento em 10 unidades de diâmetro por segundo:

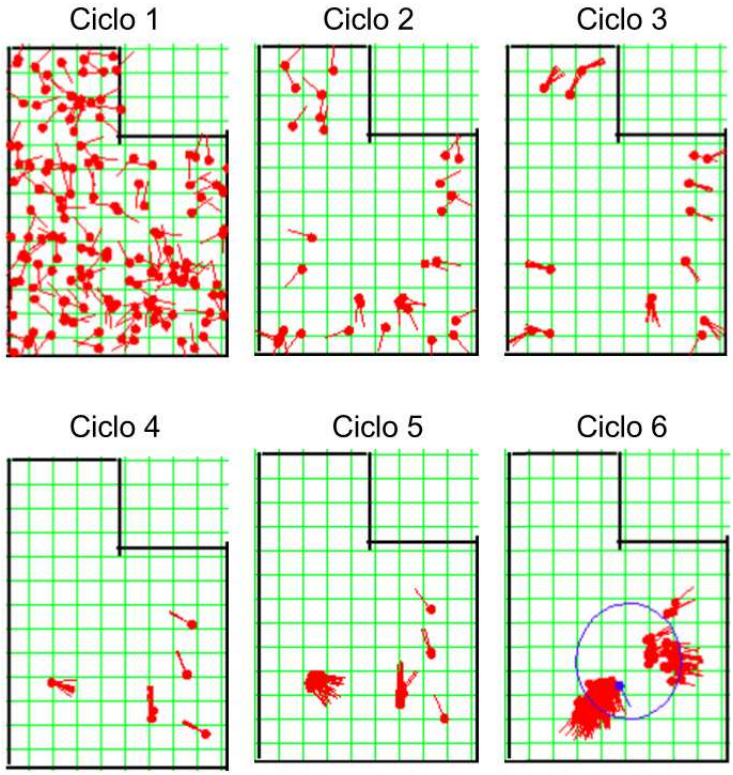

Figura 66 – Caso de Teste 3 - Cenário 3

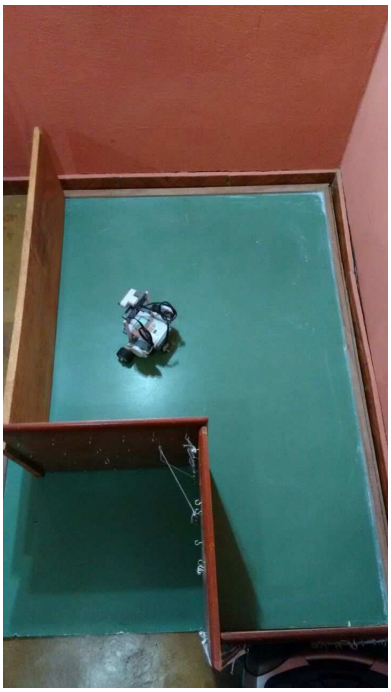

Figura 67 – Posição Real do Caso de Teste 3 - Cenário 3.

C.3.4 Cenário 4

Cenário utilizando velocidade de deslocamento em 15 unidades de diâmetro por segundo:

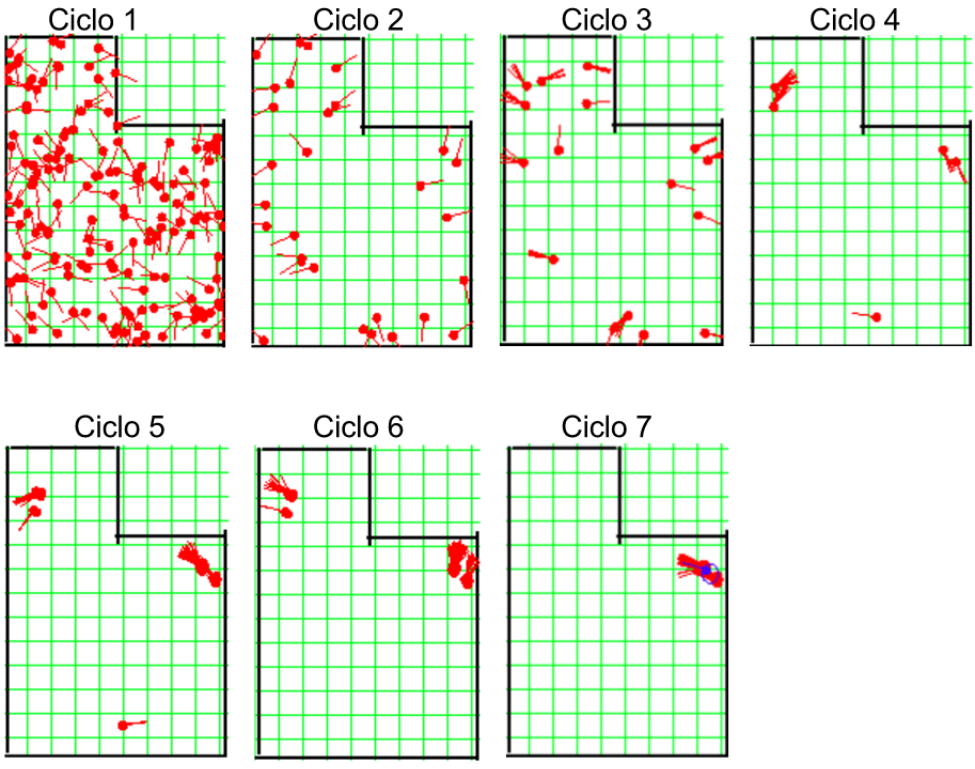

Figura 68 – Caso de Teste 3 - Cenário 4

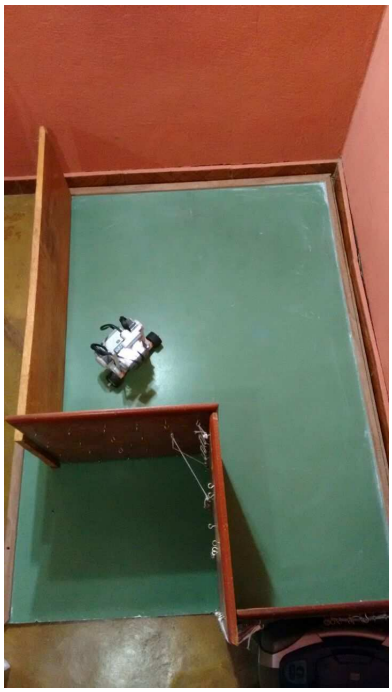

Figura 69 – Posição Real do Caso de Teste 3 - Cenário 4.

C.3.5 Cenário 5

Cenário utilizando velocidade de deslocamento em 30 unidades de diâmetro por segundo:

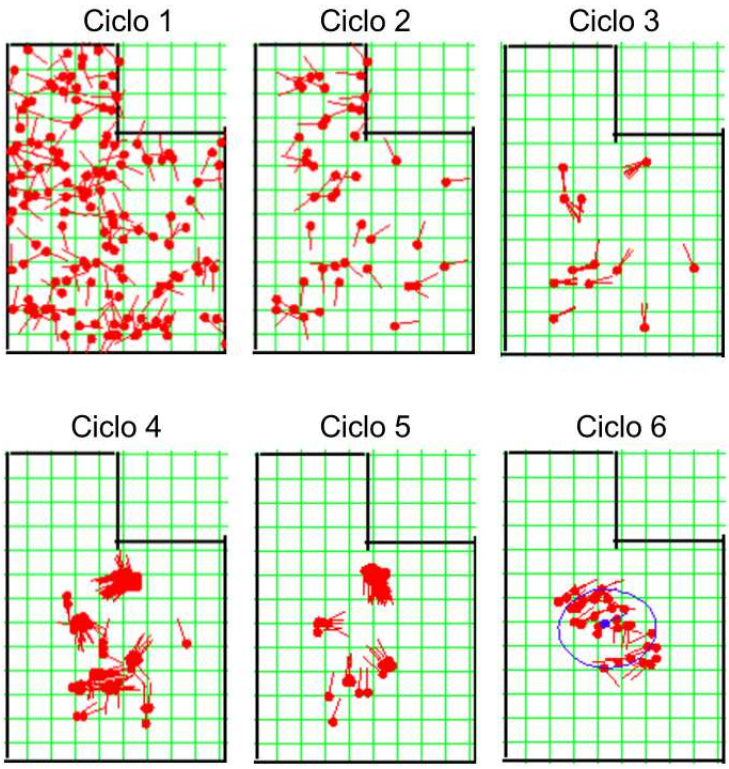

Figura 70 – Caso de Teste 3 - Cenário 5

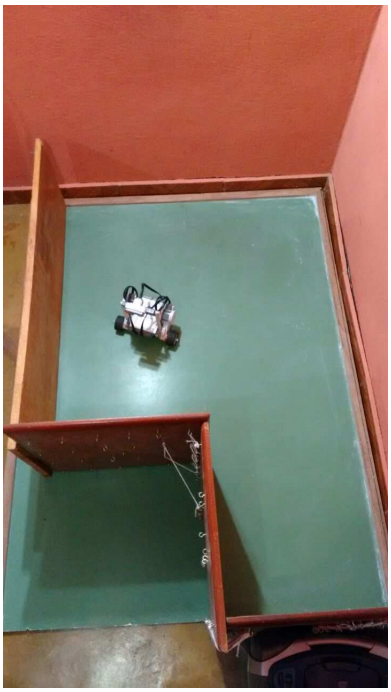

Figura 71 – Posição Real do Caso de Teste 3 - Cenário 5.

C.4 Caso de Teste 4

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 72 a Figura 81.

C.4.1 Cenário 1

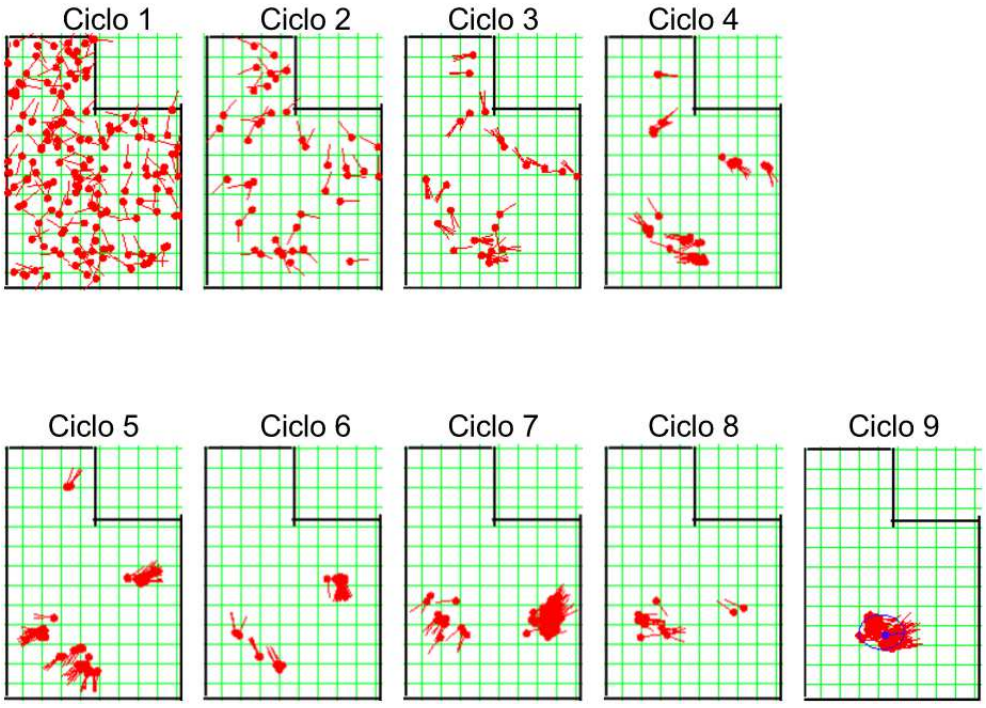

Figura 72 – Caso de Teste 4 - Cenário 1

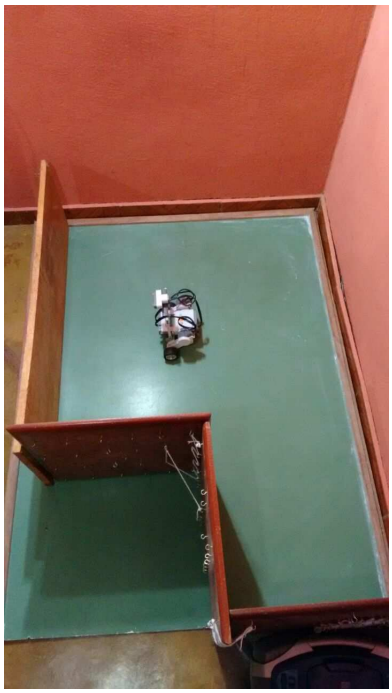

Figura 73 – Posição Real do Caso de Teste 4 - Cenário 1.

C.4.2 Cenário 2

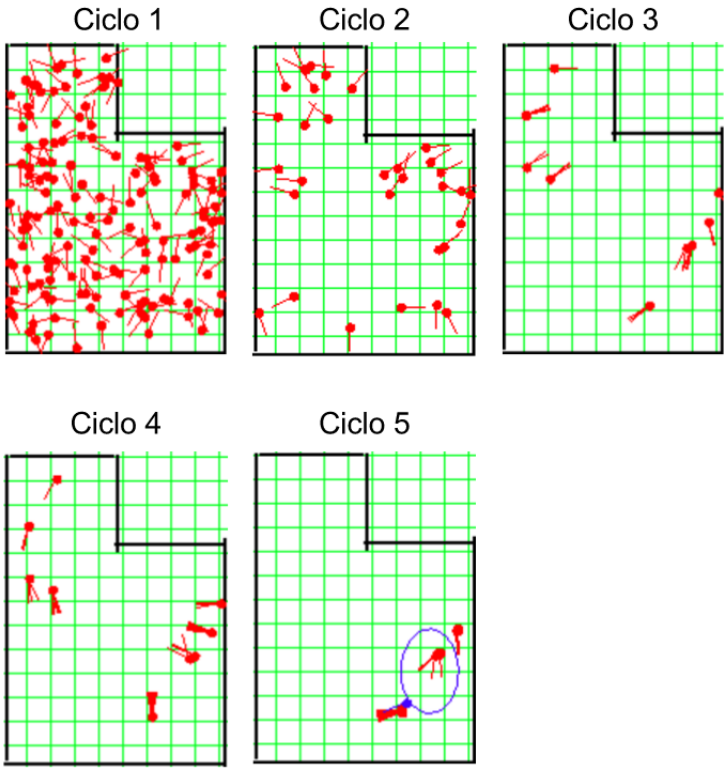

Figura 74 – Caso de Teste 4 - Cenário 2

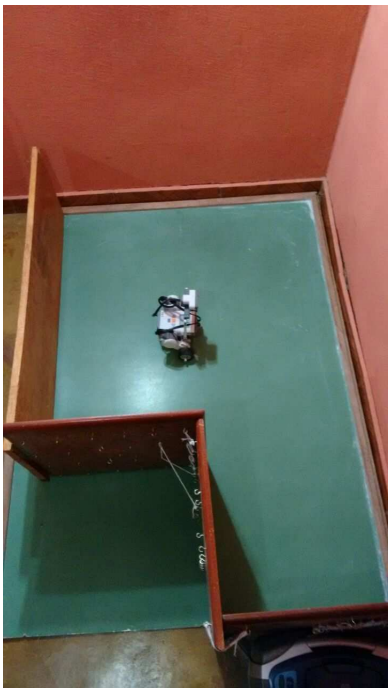

Figura 75 – Posição Real do Caso de Teste 4 - Cenário 2.

C.4.3 Cenário 3

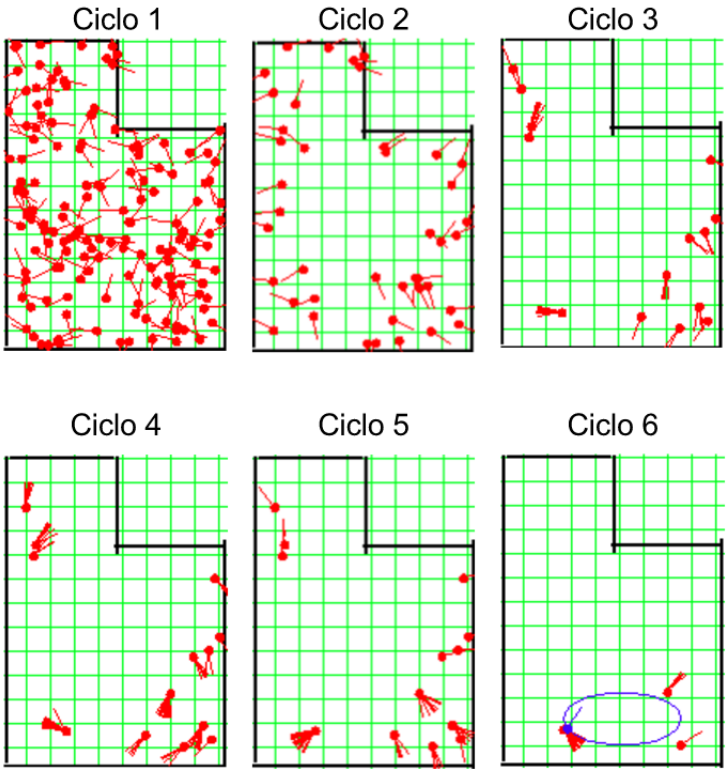

Figura 76 – Caso de Teste 4 - Cenário 3

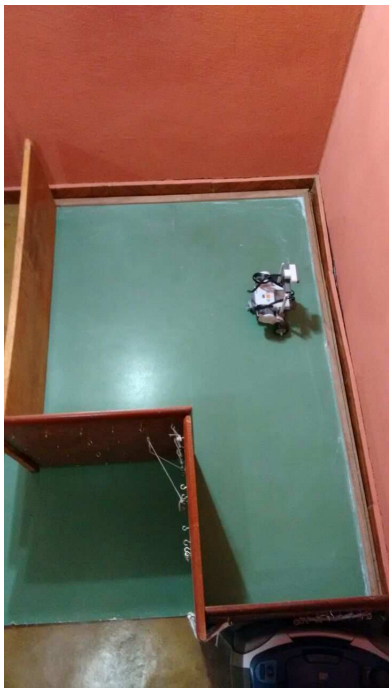

Figura 77 – Posição Real do Caso de Teste 4 - Cenário 3.

C.4.4 Cenário 4

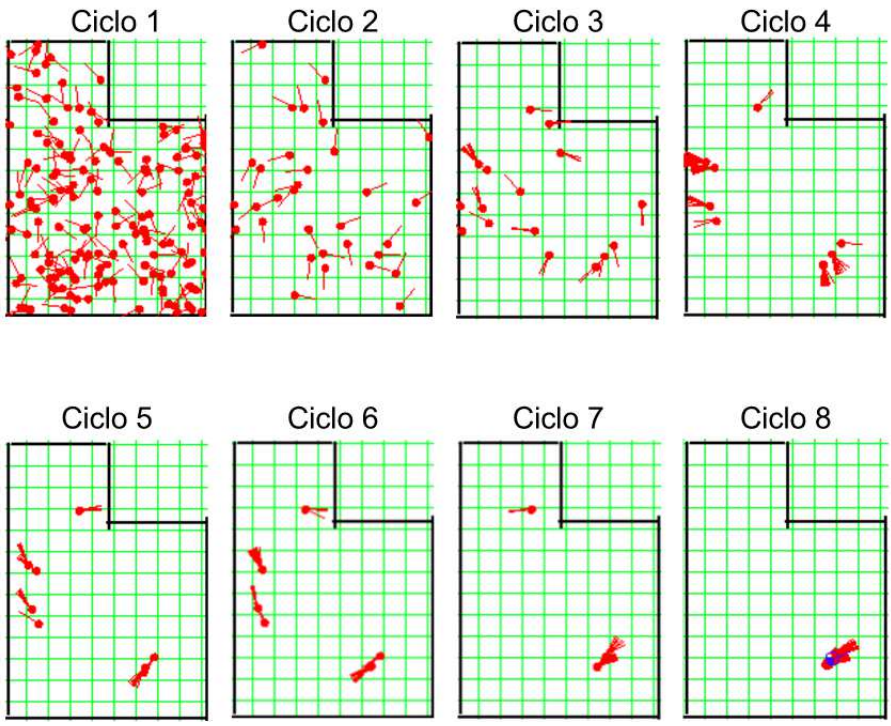

Figura 78 – Caso de Teste 4 - Cenário 4

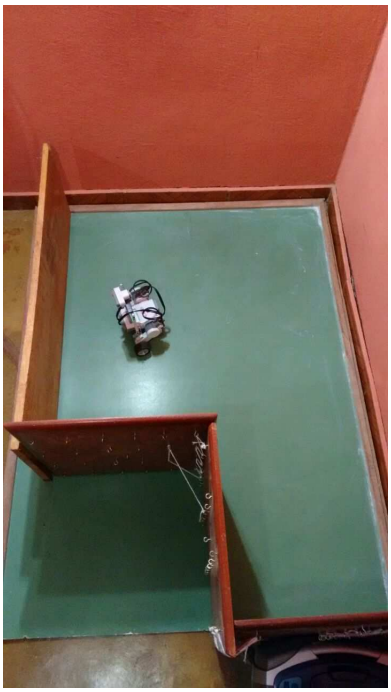

Figura 79 – Posição Real do Caso de Teste 4 - Cenário 4.

C.4.5 Cenário 5

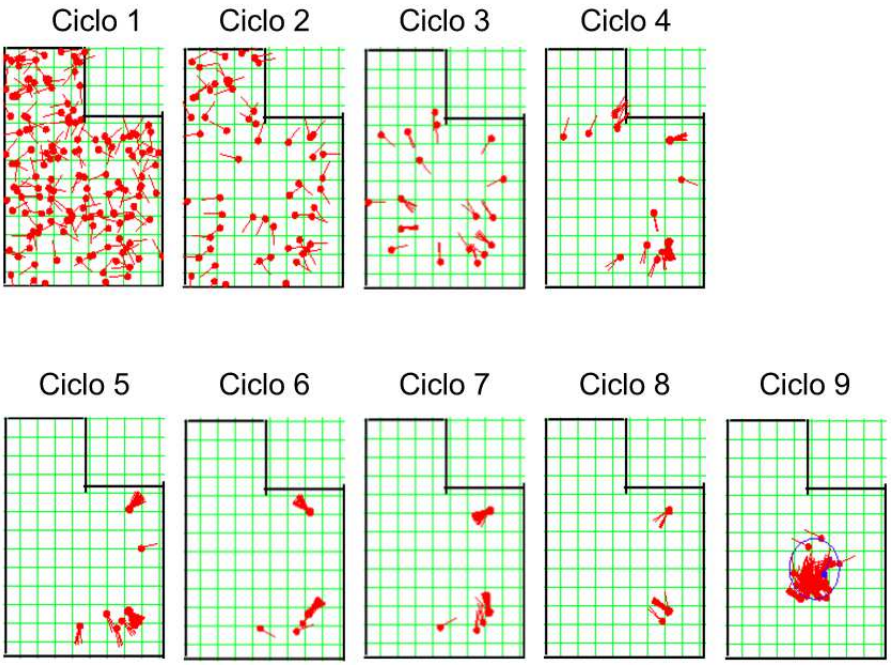

Figura 80 – Caso de Teste 4 - Cenário 5

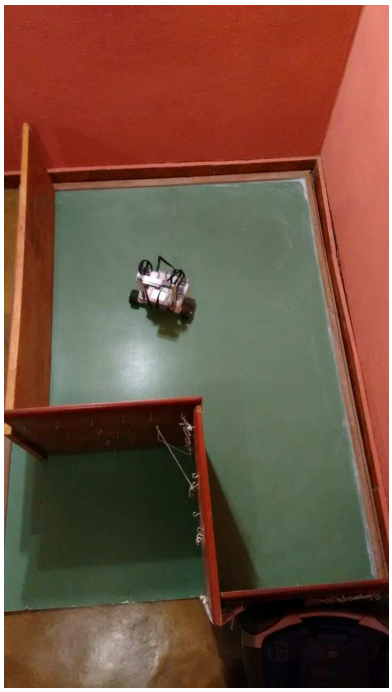

Figura 81 – Posição Real do Caso de Teste 4 - Cenário 5.

C.5 Caso de Teste 5

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 82 a Figura 91.

C.5.1 Cenário 1

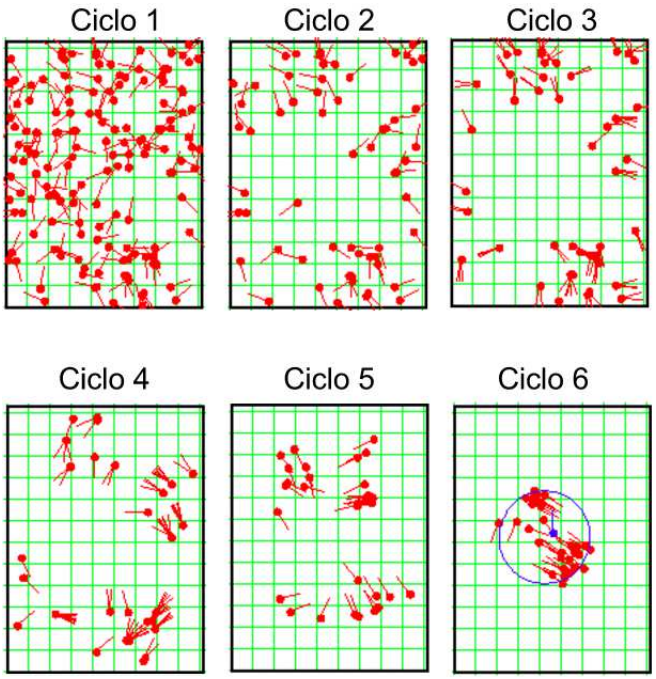

Figura 82 – Caso de Teste 5 - Cenário 1.

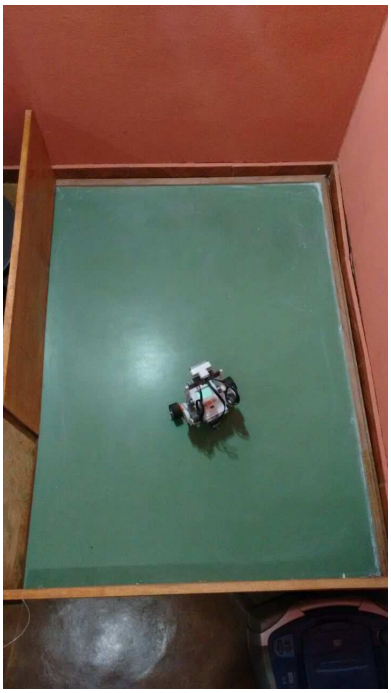

Figura 83 – Posição Real do Caso de Teste 5 - Cenário 1.

C.5.2 Cenário 2

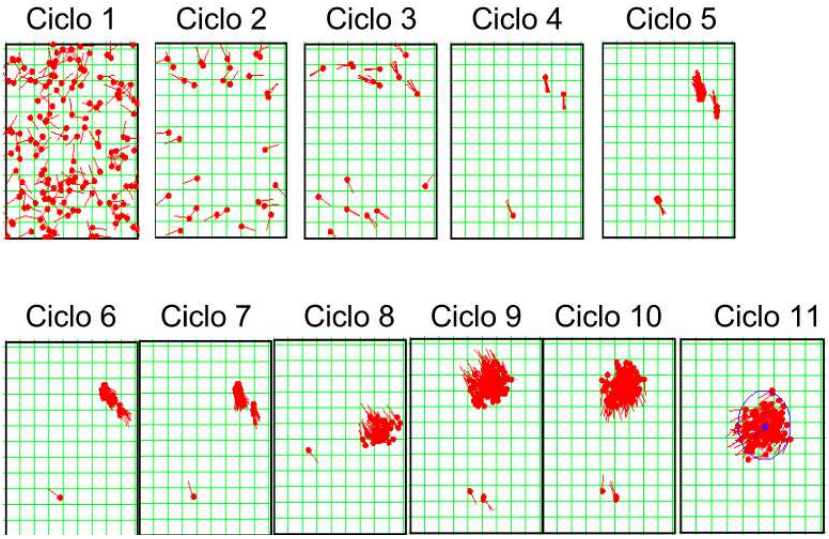

Figura 84 – Caso de Teste 5 - Cenário 2.

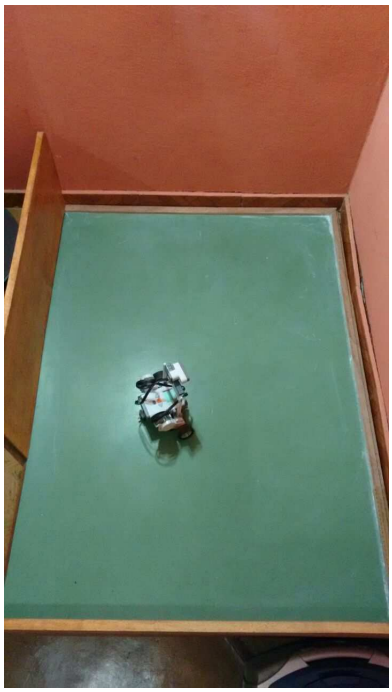

Figura 85 – Posição Real do Caso de Teste 5 - Cenário 2.

C.5.3 Cenário 3

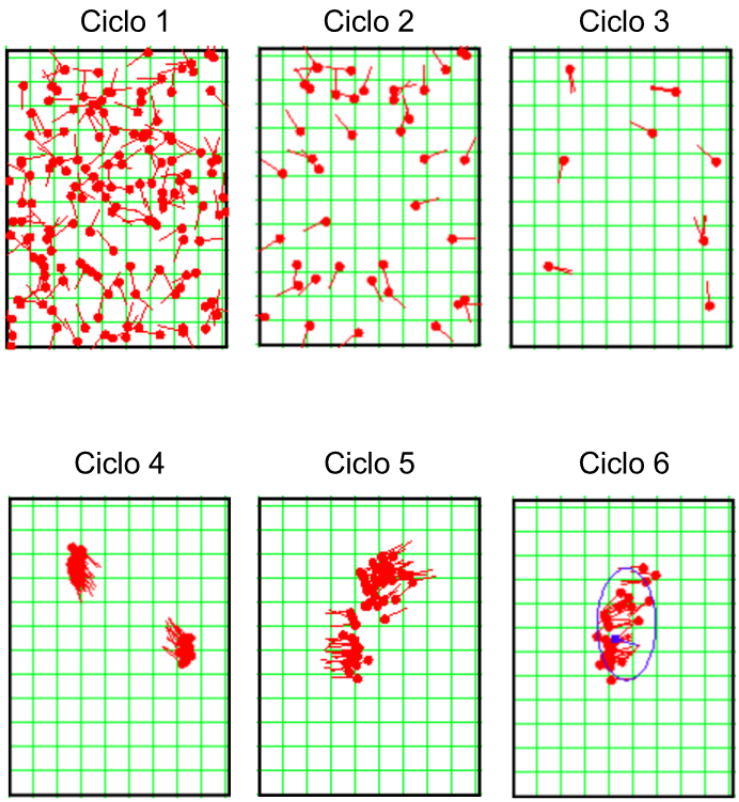

Figura 86 – Caso de Teste 5 - Cenário 3.

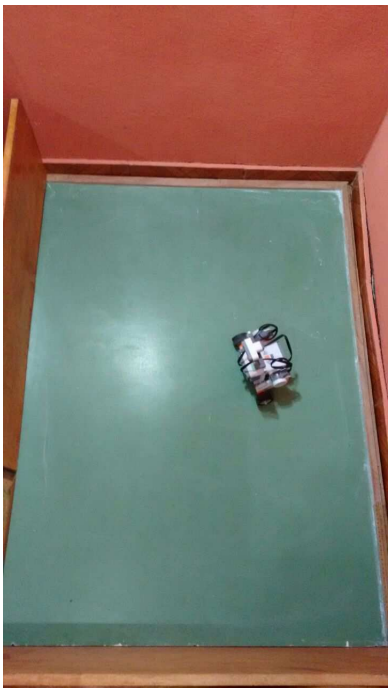

Figura 87 – Posição Real do Caso de Teste 5 - Cenário 3.

C.5.4 Cenário 4

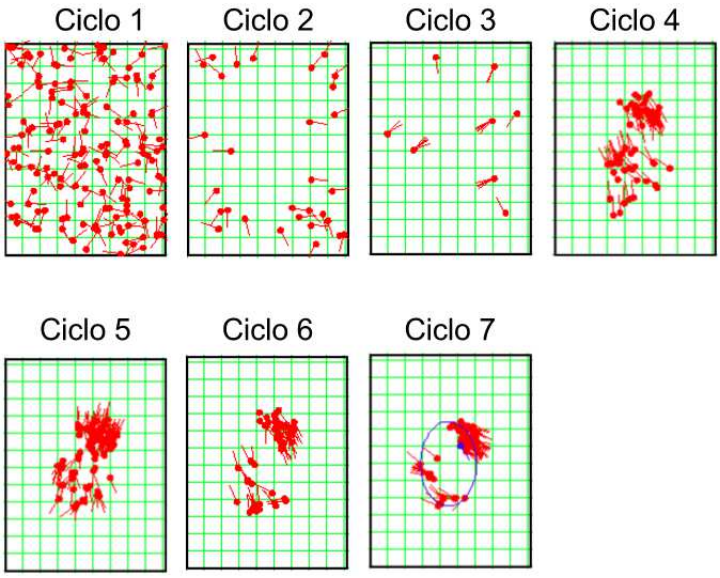

Figura 88 – Caso de Teste 5 - Cenário 4.

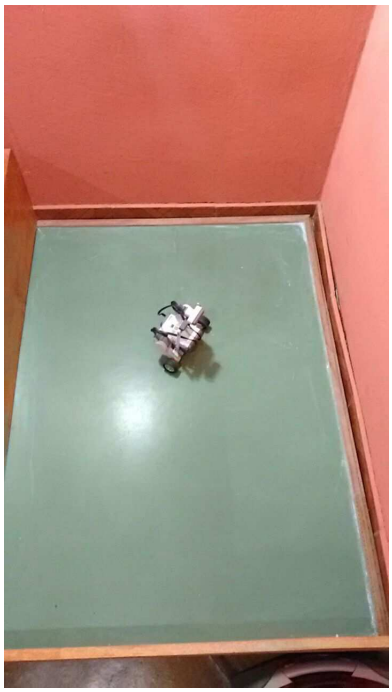

Figura 89 – Posição Real do Caso de Teste 5 - Cenário 4.

C.5.5 Cenário 5

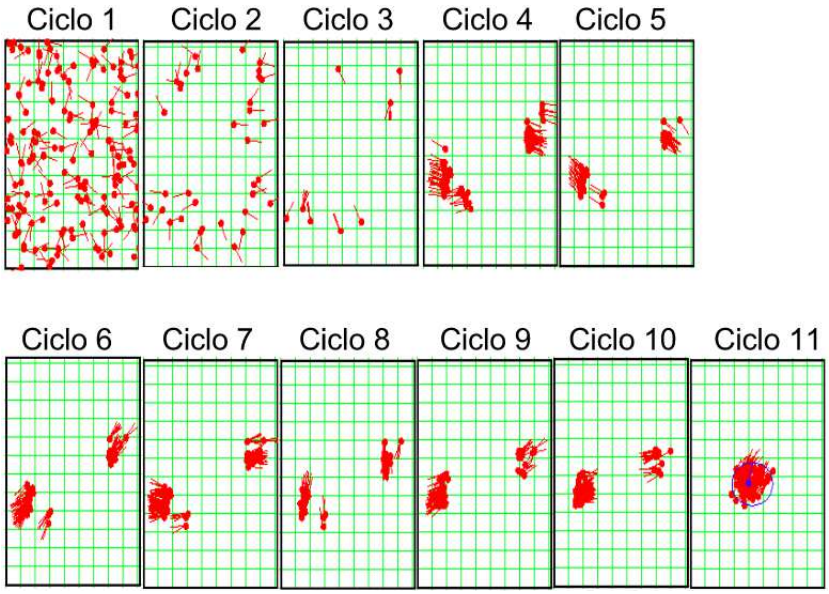

Figura 90 – Caso de Teste 5 - Cenário 5.

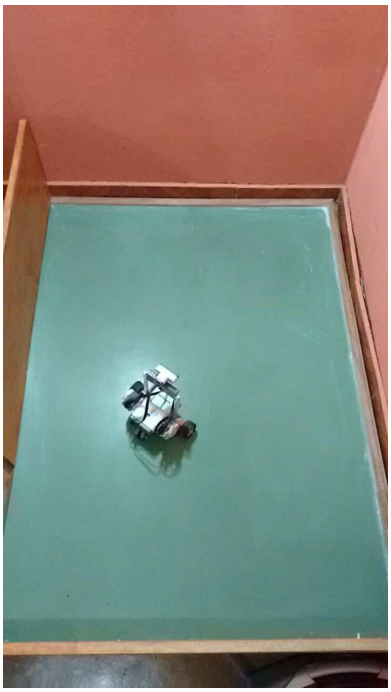

Figura 91 – Posição Real do Caso de Teste 5 - Cenário 5.

C.6 Caso de Teste 6

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 92 a Figura 101.

C.6.1 Cenário 1

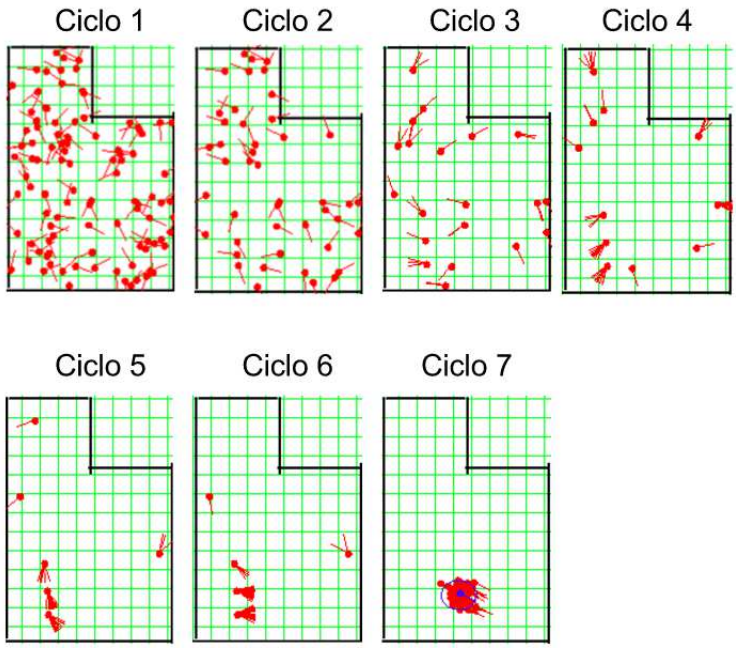

Figura 92 – Caso de Teste 6 - Cenário 1.

Figura 93 – Posição Real do Caso de Teste 6 - Cenário 1.

C.6.2 Cenário 2

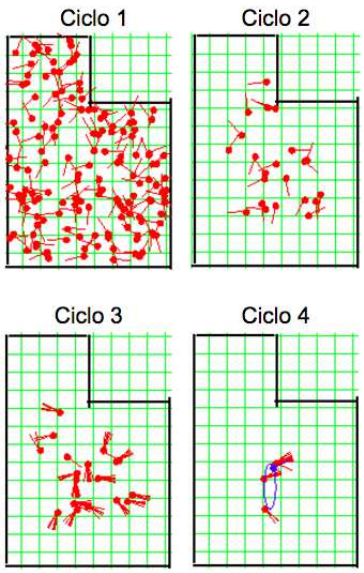

Figura 94 – Caso de Teste 6 - Cenário 2.

Figura 95 – Posição Real do Caso de Teste 6 - Cenário 2.

C.6.3 Cenário 3

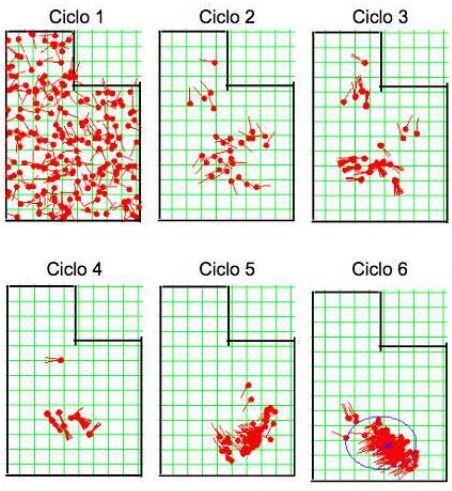

Figura 96 – Caso de Teste 6 - Cenário 3.

Figura 97 – Posição Real do Caso de Teste 6 - Cenário 3.

C.6.4 Cenário 4

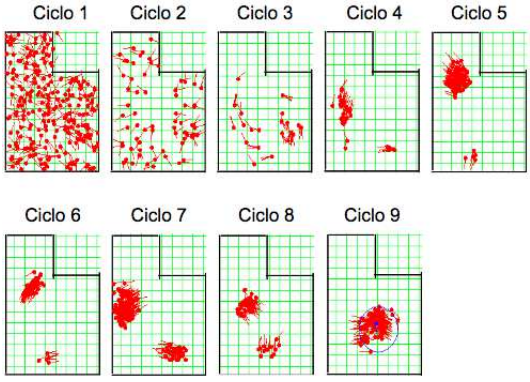

Figura 98 – Caso de Teste 6 - Cenário 4.

Figura 99 – Posição Real do Caso de Teste 6 - Cenário 4.

C.6.5 Cenário 5

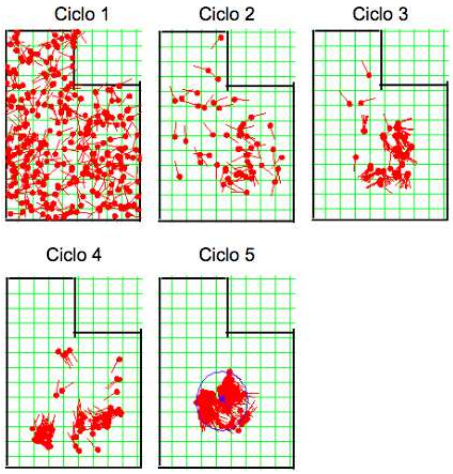

Figura 100 – Caso de Teste 6 - Cenário 5.

Figura 101 – Posição Real do Caso de Teste 6 - Cenário 5.

C.7 Caso de Teste 7

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 102 a Figura 111.

C.7.1 Cenário 1

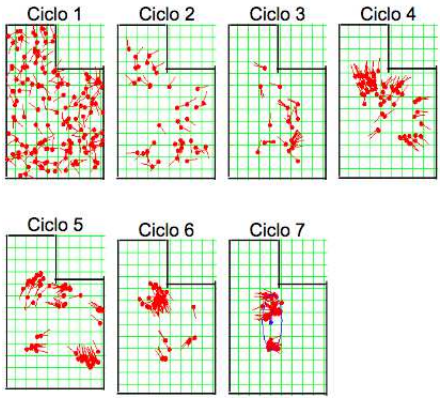

Figura 102 – Caso de Teste 7 - Cenário 1.

Figura 103 – Posição Real do Caso de Teste 7 - Cenário 1.

C.7.2 Cenário 2

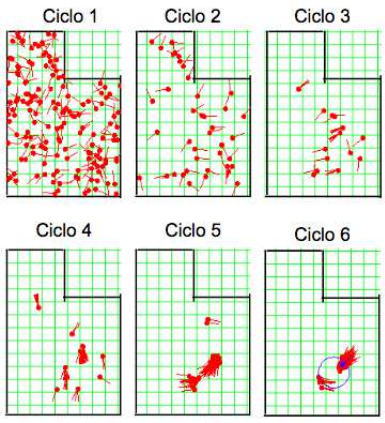

Figura 104 – Caso de Teste 7 - Cenário 2.

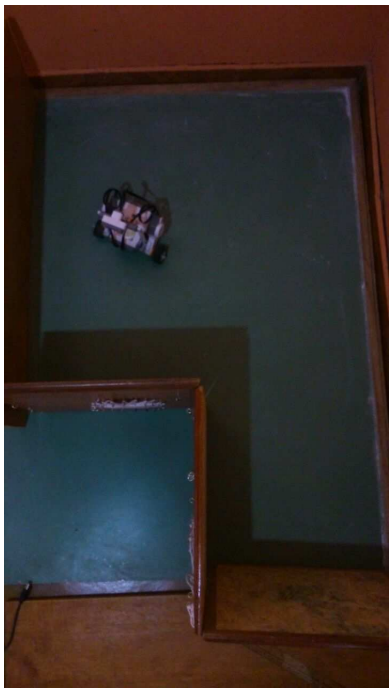

Figura 105 – Posição Real do Caso de Teste 7 - Cenário 2.

C.7.3 Cenário 3

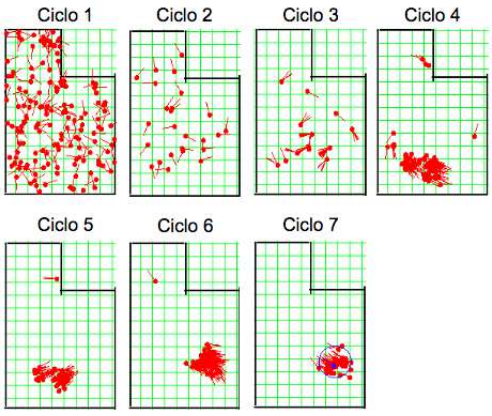

Figura 106 – Caso de Teste 7 - Cenário 3.

Figura 107 – Posição Real do Caso de Teste 7 - Cenário 3.

C.7.4 Cenário 4

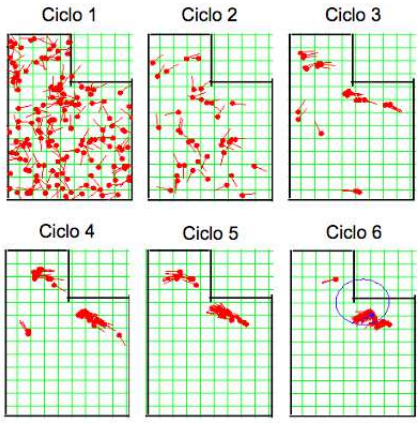

Figura 108 – Caso de Teste 7 - Cenário 4.

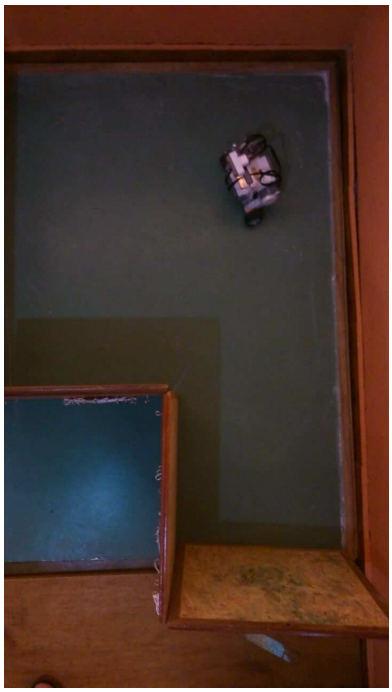

Figura 109 – Posição Real do Caso de Teste 7 - Cenário 4.

C.7.5 Cenário 5

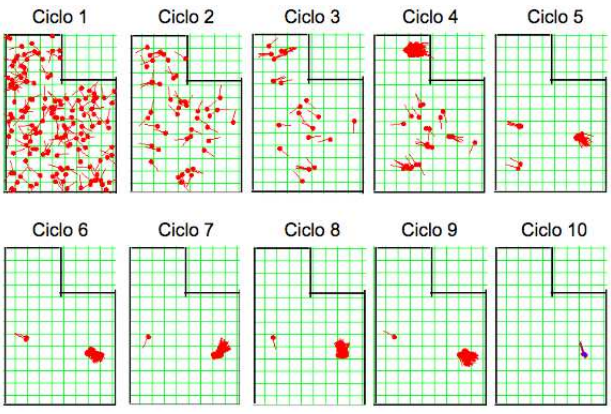

Figura 110 – Caso de Teste 7 - Cenário 5.

Figura 111 – Posição Real do Caso de Teste 7 - Cenário 5.

C.8 Caso de Teste 8

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 112 a Figura 121.

C.8.1 Cenário 1

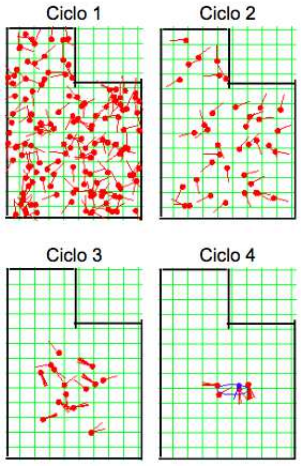

Figura 112 – Caso de Teste 8 - Cenário 1.

Figura 113 – Posição Real do Caso de Teste 8 - Cenário 1.

C.8.2 Cenário 2

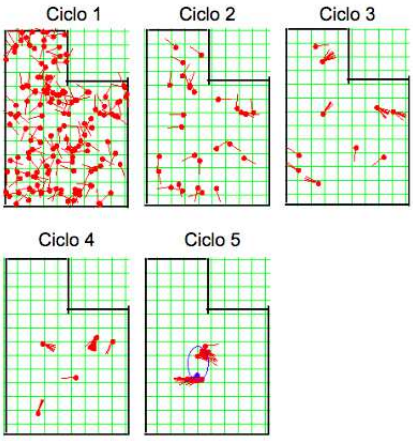

Figura 114 – Caso de Teste 8 - Cenário 2.

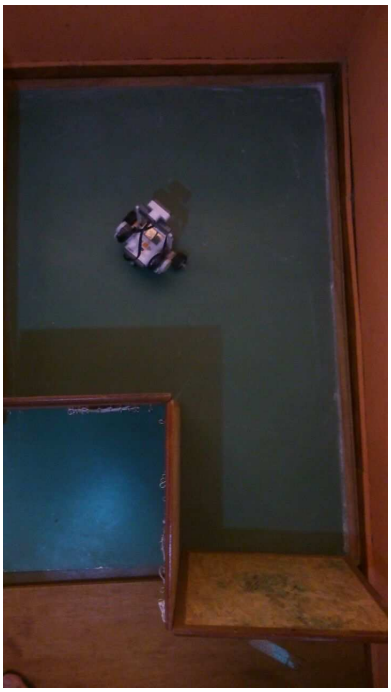

Figura 115 – Posição Real do Caso de Teste 8 - Cenário 2.

C.8.3 Cenário 3

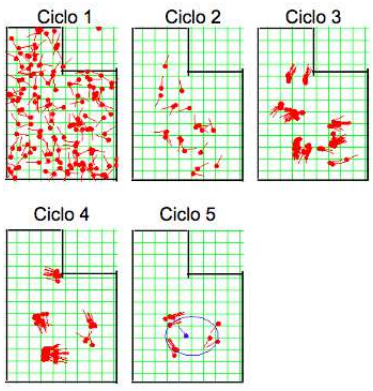

Figura 116 – Caso de Teste 8 - Cenário 3.

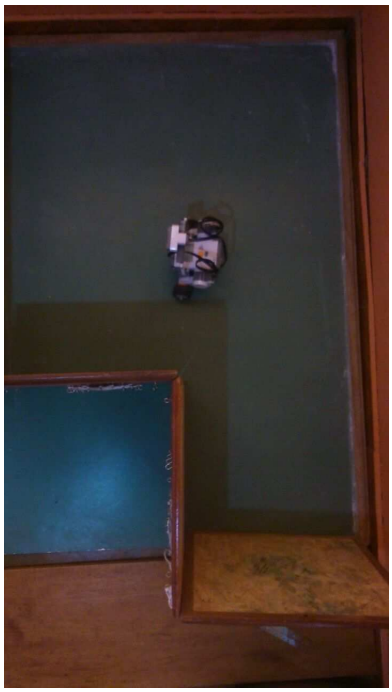

Figura 117 – Posição Real do Caso de Teste 8 - Cenário 3.

C.8.4 Cenário 4

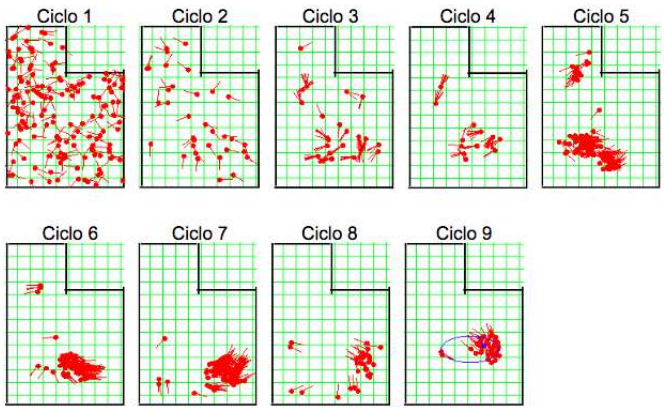

Figura 118 – Caso de Teste 8 - Cenário 4.

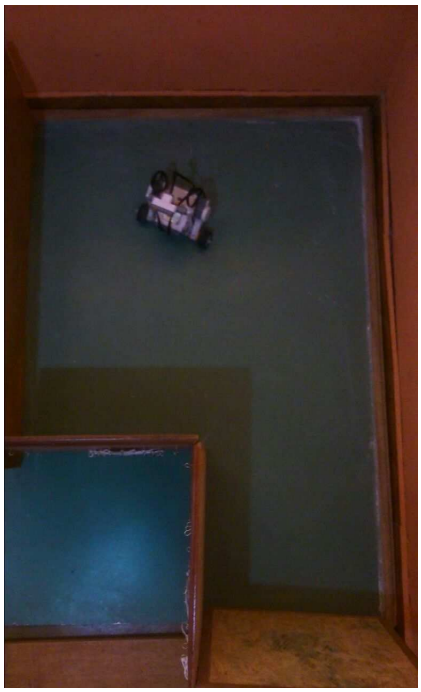

Figura 119 – Posição Real do Caso de Teste 8 - Cenário 4.

C.8.5 Cenário 5

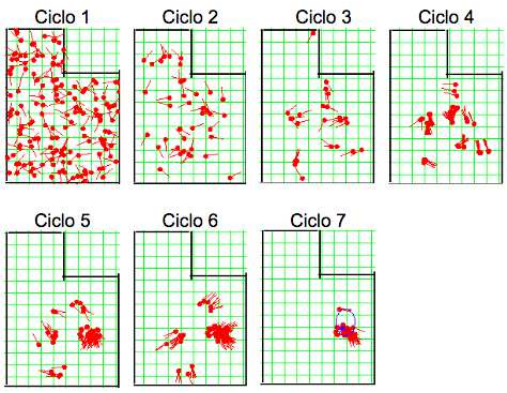

Figura 120 – Caso de Teste 8 - Cenário 5.

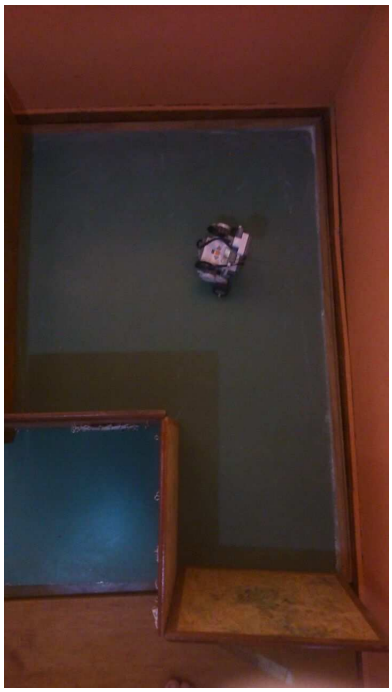

Figura 121 – Posição Real do Caso de Teste 8 - Cenário 5.

C.9 Caso de Teste 9

Este caso de teste está dividido em 5 (cinco) cenários, os quais são apresentados a seguir, contemplando todo o processo de auto-localização em cada exemplo, a partir da apresentação das imagens a seguir, Figura 122 a Figura 131.

C.9.1 Cenário 1

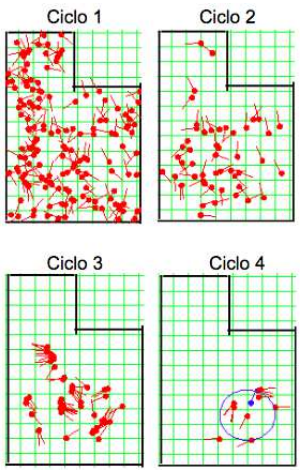

Figura 122 – Caso de Teste 9 - Cenário 1.

Figura 123 – Posição Real do Caso de Teste 9 - Cenário 1.

C.9.2 Cenário 2

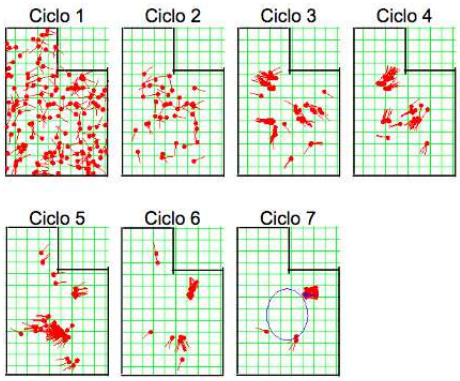

Figura 124 – Caso de Teste 9 - Cenário 2.

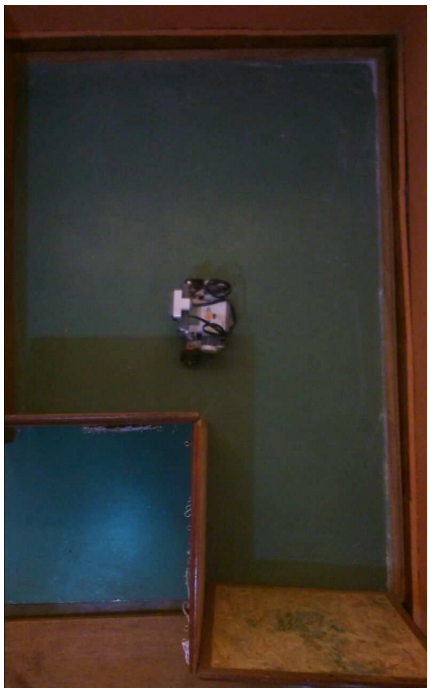

Figura 125 – Posição Real do Caso de Teste 9 - Cenário 2.

C.9.3 Cenário 3

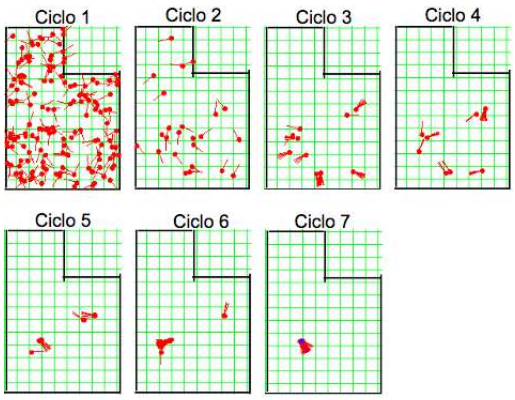

Figura 126 – Caso de Teste 9 - Cenário 3.

Figura 127 – Posição Real do Caso de Teste 9 - Cenário 3.

C.9.4 Cenário 4

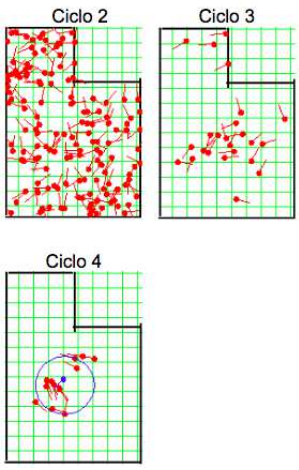

Figura 128 – Caso de Teste 9 - Cenário 4.

Figura 129 – Posição Real do Caso de Teste 9 - Cenário 4.

C.9.5 Cenário 5

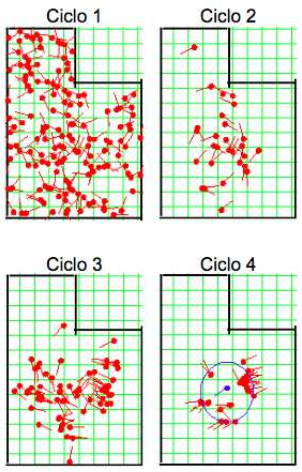

Figura 130 – Caso de Teste 9 - Cenário 5.

Figura 131 – Posição Real do Caso de Teste 9 - Cenário 5.

Referências

ADAMS, M.; MULLANE, J.; VO, B.-N. Circumventing the feature association problem in slam. IEEE Intelligent transportation systems magazine, 2013. Citado 3 vezes nas páginas 33, 113 y 114.

BAGNALL, B. *Intelligence Unleashed: Creating LEGO NXT Robots With Java*. [S.l.: s.n.], 2011. ISBN 9780986832208. Citado 5 vezes nas páginas 62, 71, 121, 123 y 124.

BECKER, F. O que é construtivismo? Universidade Federal do Rio Grande do Sul, 2009. Citado 2 vezes nas páginas 37 y 38.

BENITTI, F. B. V. et al. Experimentação com robótica educativa no ensino médio: ambiente, atividades e resultados. Universidade do Estado de Santa Catarina, 2009. Citado 2 vezes nas páginas 36 y 59.

BIOLCHINI, J. et al. Systematic review in software engineering. Universidade Federal do Rio de Janeiro, 2005. Citado na página 106.

BRITO, M. F. de et al. Knowledge transfer in outsourcing software development projects. 12th international conference on information systems and technology management, 2015. Citado na página 106.

BUKHORI, .; ISMAIL, Z. H.; NAMERIKAWA, T. Detection strategy for kidnapped robot problem in landmark-based map monte carlo localization. IEEE International Symposium on Robotics and Intelligent Sensors, 2015. Citado 2 vezes nas páginas 33 y 60.

CHATTERJEE, A.; MATSUNO, F. A neuro-fuzzy assisted extended kalman filter-based approach for simultaneous localization and mapping (slam) problems. IEEE transactions on fuzzy systems, 2007. Citado na página 110.

CHEW, M. et al. Simple mobile robots for introduction into engineering. *International Instrumentation and Measurement Technology Conference*, 2009. Citado 2 vezes nas páginas 31 y 38.

CHEW, M. T. et al. Simple mobile robots for introduction into engineering. International Instrumentation and Measurement, 2009. Citado 3 vezes nas páginas 35, 37 y 38.

COSTA, A. H. R.; Okamoto Jr., J. *Robótica Industrial - Interação de Robô no Ambiente*. [S.l.]: Edgard Blücher Ltda, 2002. ISBN 8521203152. Citado 2 vezes nas páginas 25 y 114.

COSTA, J.; DIAS, F.; ARAúJO, R. Simultaneous localization and map building by integrating a cache of features. University of Coimbra, Portugal, 2006. Citado na página 110.

COSTA, T. C. A. Uma abordagem construcionista da utilização dos computadores na educação. Universidade Federal de Pernambuco, 2010. Citado 2 vezes nas páginas 36 y 37.

DISSANAYAKE, G. et al. A solution to the simultaneous localization and map building (slam) problem. The University of Sydney, Australia, 2001. Citado 6 vezes nas páginas 27, 33, 34, 35, 109 y 115.

DISSANAYAKE, G. et al. Map management for efficient simultaneous localization and mapping (slam). University of Sydney, Sydney, NSW 2006, Australia, 2002. Citado na página 110.

FARROKHSIAR, M.; KRYS, D.; NAJJARAN, H. A teaching tool for the state-of-the-art probabilistic methods used in localization of mobile robots. Okanagan School of Engineering, University of British Columbia, Kelowna, B. C., Canada, 2010. Citado na página 31.

FUENTES-PACHECO, J.; RUIZ-ASCENCIO, J.; RENDóN-MANCHA, J. M. Visual simultaneous localization and mapping: a survey. Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México, 2012. Citado 2 vezes nas páginas 110 y 114.

GALVAN, S. et al. Innovative robotics teaching using lego sets. University of Verona, Italy, 2006. Citado 8 vezes nas páginas 28, 31, 36, 37, 38, 39, 59 y 99.

GERHARDT, T. E.; SILVEIRA, D. T. Metodos de pesquisa. Universidade Federal do Rio Grande do Sul, 2009. Citado 2 vezes nas páginas 49 y 50.

GUIZILINI, V. C. et al. Implementação do dp-slam em tempo real para robôs móveis usando sensores. Universidade Estadual de Campinas, 2004. Citado na página 35.

HUANG, S.; WANG, Z.; DISSANAYAKE, G. Iterated d-slam map joining: evaluating its performance in terms of consistency, accuracy and efficiency. Sydney, Australia, 2009. Citado na página 115.

HáMORI Ákos; LENGYEL, J.; RESKó, B. 3dof drawing robot using lego-nxt. Óbuda University, Hungary, 2011. Citado na página 27.

JI, X. et al. Incremental slam with backtracking data association for mobile robots. International Conference on Information and Automation. Zhangjiajie, China, 2008. Citado na página 111.

KATSEV, M. et al. Mapping and pursuit-evasion strategies for a simple wall-following robot. IEEE Transactions on robotics, vol. 27, no. 1, february 2011, 2011. Citado 2 vezes nas páginas 111 y 115.

KITCHENHAM, B. A. et al. Lessons from applying the systematic literature review process within the software engineering domain. Keele University, Staffordshire ST5 5BG, UK, 2006. Citado 5 vezes nas páginas 54, 55, 56, 105 y 106.

LAU, N. et al. Ciber-rato: Uma competição robótica num ambiente virtual. 2002. Citado 2 vezes nas páginas 28 y 38.

LIN, H.-H.; TSAI, C.-C. Improved global localization of an indoor mobile robot via fuzzy extended information filtering. *Robotica*, v. 26, p. 241–254, 3 2008. ISSN 1469-8668. Disponível em: <http://journals-cambridge-org.ez54.periodicos.capes.gov.br/article_S0263574707003876>. Citado 2 vezes nas páginas 110 y 115.

LINDHORST, T.; LUKAS, G.; NETT, E. Wireless mesh network infrastructure for industrial applications – a case study of tele-operated mobile robots. University of Magdeburg, 2013. Citado 2 vezes nas páginas 113 y 114.

MACHADO, K. F. Módulo de auto-localização para um agente exploratório usando filtro de kalman. Universidade Federal do Rio Grande do Sul - Instituto de Informática, 2003. Citado 5 vezes nas páginas 31, 32, 34, 115 y 116.

MAFRA, S. N.; TRAVASSOS, G. H. Estudos primáriose secundários apoiando a busca por evidência em engenharia de software. Universidade Federal do Rio de Janeiro, 2006. Citado na página 109.

MAIMONE, M.; CHENG, Y.; MATTHIES, L. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, Wiley Online Library, v. 24, n. 3, p. 169–186, 2007. Citado 2 vezes nas páginas 31 y 115.

MAJDIK, A. et al. New approach in solving the kidnapped robot problem. Technical University of Cluj-Napoca, Romania, 2010. Citado na página 33.

MALIUK, K. D. Robótica educacional como cenário investigativo nas aulas de matemática. Universidade Federal do Rio Grande do Sul, Porto Alegre, 2009. Citado 3 vezes nas páginas 28, 37 y 38.

MARCONI, M. D. A.; LAKATOS, E. M. *Métodos Científicos*. [S.l.]: Editora Atlas, 2003. ISBN 8522433976. Citado na página 51.

MARTINS, J. M. F. Melhoramento do desempenho do robot de serviço de limpeza. Faculdade de Engenharia da Universidade do Porto, 2008. Citado na página 31.

MOONEY, C. Z. *Monte carlo simulation*. [S.l.]: Sage Publications, 1997. v. 116. Citado na página 35.

MORAES, T. M.; SOUZA, A. S. de. Revisão sistemática sobre a comunicação dentro do processo de desenvolvimento de software. Universidade Federal de Goiás - GO, 2011. Citado na página 56.

MORESI, E. Metodologia da pesquisa. Universidade Católica de Brasília - DF, 2003. Citado na página 50.

NETO, A. M. M. da S. Comparação entre o filtro de kalman e filtro de partículas aplicados na robótica móvel. Universidade Estadual de Campinas, 2015. Citado 2 vezes nas páginas 35 y 114.

NUNES, S. da C.; SANTOS, R. P. dos. O construcionismo de papert na criação de um objeto de aprendizagem e sua avaliação segundo a taxionomia de bloom. IX Encontro Nacional de Pesquisa em Educação em Ciências, 2013. Citado 3 vezes nas páginas 36, 37 y 59.

OLIVEIRA, M. F. de. Metodologia cientÍfica: um manual para a realização de pesquisas em administração. Universidade Federal de Goiás, Catalão GO, 2011. Citado 2 vezes nas páginas 49 y 50.

OLIVEIRA, P. R. G. D. Auto-localização e construção de mapas de ambiente para robôs móveis baseados em visão omnidirecional estéreo. São Paulo, Brasil, 2008. Citado 5 vezes nas páginas 25, 27, 32, 109 y 113.

PAULA, L. C. M. de et al. Aplicação de processamento paralelo em método iterativo para solucão de sistemas lineares. X Encontro Anual de Computação - EnAComp, 2013. Citado na página 27.

PINTO, F. P. V. F. The cleaning robot project aplicação do filtro de kalman na auto-localização de um sistema robótico autónomo. Faculdade de Engenharia da Universidade do Porto, 2008. Citado 5 vezes nas páginas 31, 32, 33, 34 y 109.

QUARTIERO, E. M. Da máquina de ensinar à máquina de aprender: pesquisas em tecnologia educacional. Universidade do Estado de Santa Catarina, 2007. Citado na página 36.

RAMALHO, C. B. Máquina de raciocínio lógico para tomada de decisões estratégicas em robótica educacional. 2015. Citado 2 vezes nas páginas 28 y 60.

RINCON, R. L. Traveller: Um framework de definição de trajetórias para robôs móveis. 2015. Citado 3 vezes nas páginas 28, 60 y 120.

ROMANO, V. F. *Introdução à Robótica Industrial*. [S.l.]: Edgard Blücher Ltda, 2002. ISBN 8521203152. Citado 3 vezes nas páginas 25, 31 y 114.

SAEEDI, S. et al. Neural network-based multiple robot simultaneous localization and mapping. Ieee Transactions on Neural Networks, 2011. Citado 3 vezes nas páginas 110, 113 y 115.

SANTOS, F. M.; SILVA, V. F.; ALMEIDA, L. Auto-localização em pequenos robôs móveis e autônomos: O caso do robô bulldozer iv. 2002. Citado 5 vezes nas páginas 25, 26, 27, 28 y 32.

SEIFZADEH, S.; WU, D.; WANG, Y. Cost-effective active localization technique for mobile robots. International Conference on Robotics and Biomimetics, 2009. Citado na página 35.

SERRANO, M.; SERRANO, M. Análise de ferramentas para o ensino de computação na educação básica. Universidade de Brasília (UnB/FGA), 2014. Citado na página 38.

SILVA, E. G. da et al. Análise de ferramentas para o ensino de computação na educação básica. XXXIV Congresso da Sociedade Brasileira de Computação, 2014. Citado 2 vezes nas páginas 37 y 38.

SPIELMANN, R. et al. Sensor module for mobile robot. Institute of Control and Industrial Informatics, Faculty of Electrical Engineering, Slovak University of Technology, Bratislava, Slovakia, 2013. Citado na página 92.

STECKEL, J.; PEREMANS, H. Batslam: Simultaneous localization and mapping using biomimetic sonar. University of Antwerp, Antwerp, Belgium, 2013. Citado na página 110.

SUN, R. et al. A simultaneous localization and mapping algorithm in complex environments: Slasem. Chinese Academy of Sciences, Shenyang 110016, P. R. China, 2010. Citado na página 110.

VIEIRA, F. C. Controle dinâmico de robôs móveis com acionamento diferencial. Rio Grande do Norte, Brasil, 2005. Citado 2 vezes nas páginas 60 y 61.

WAHAB1, A. F. A.; AZAHARI2, M. H.; TAJUDDIN1, R. M. An evaluation of robotic education scale in enhancing science achievement. Universiti Teknologi MARA, 2015. Citado na página 28.

WANG, K.; LIU, Y.-H.; LI, L. A simple and parallel algorithm for real-time robot localization by fusing monocular vision and odometry/ahrs sensors. IEEE/ASME Transactions on Mechatronics, vol. 19, no. 4, august 2014, 2014. Citado 2 vezes nas páginas 111 y 115.

WANG, Z.; HUANG, S.; DISSANAYAKE, G. Multi-robot simultaneous localization and mapping using d-slam framework. Faculty of Engineering, University of Technology, Sydney, Australia, 2007. Citado na página 113.

WON, D. H. et al. Integration of vision based slam and nonlinear filter for simple mobile robot navigation. Konkuk University Seoul, Korea, 2008. Citado 4 vezes nas páginas 32, 33, 35 y 109.

ZHANG, H.; BABAR, M. A.; TELL, P. Identifying relevant studies in software engineering. National ICT Australia, University of New South Wales, Australia, 2010. Citado 2 vezes nas páginas 106 y 107.

ZHAO, S. et al. Research on robotic education based on lego bricks. School of Information Science and Engineering, Northeastern University, 2008. Citado 2 vezes nas páginas 28 y 36.