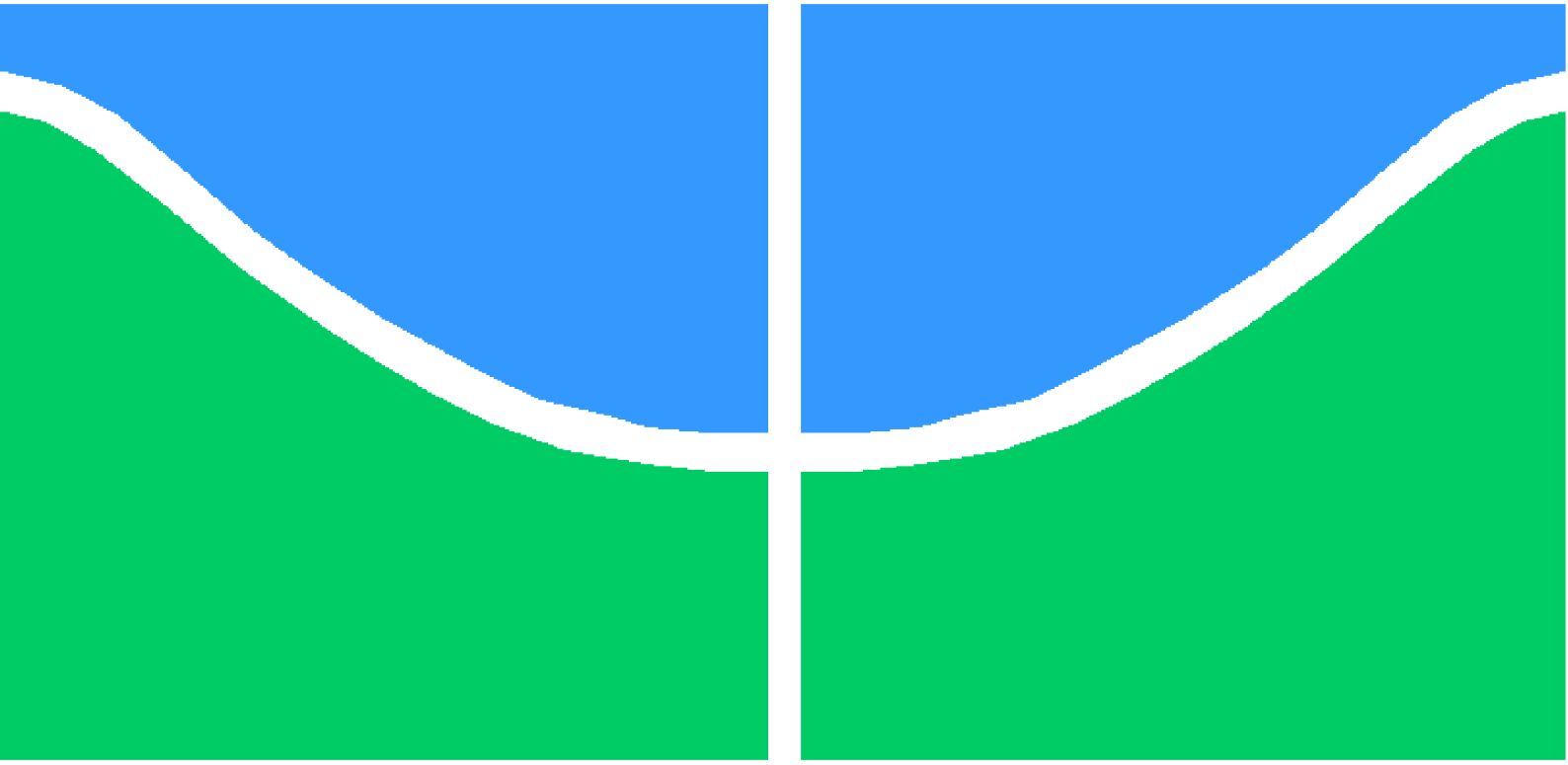

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Máquina de Raciocínio Lógico para Tomada de Decisões Estratégicas em Robotica Educacional

Autor: Carolina Barros Ramalho
Orientador: Profª. Drª. Milene Serrano

Brasília, DF
2015

Carolina Barros Ramalho

Máquina de Raciocínio Lógico para Tomada de Decisões Estratégicas em Robotica Educacional

Monografia submetida ao curso de graduação em (Engenharia de Software) da 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: Profª. Drª. Milene Serrano

Coorientador: Prof. Dr. Maurício Serrano

Brasília, DF

2015

Carolina Barros Ramalho
Máquina de Raciocínio Lógico para Tomada de Decisões Estratégicas em Robotica Educacional/ Carolina Barros Ramalho. – Brasília, DF, 2015-
85 p. : il. (algumas color.) ; 30 cm.

Orientador: Profª. Drª. Milene Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. robótica educacional. 2. programação dinâmica. I. Profª. Drª. Milene Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Máquina de Raciocínio Lógico para Tomada de Decisões Estratégicas em Robotica Educacional

CDU 02:141:005.6

Carolina Barros Ramalho

Máquina de Raciocínio Lógico para Tomada de Decisões Estratégicas em Robotica Educacional

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

Trabalho aprovado. Brasília, DF, 08 de julho de 2015:

Profª. Drª. Milene Serrano
Orientador

Prof. Dr. Maurício Serrano
Coorientador

Profª. Dra. Fabiana Freitas Mendes
Convidado 1

Prof. Dr Daniel Mauricio Muñoz Arboleda
Convidado 2

Brasília, DF
2015

Este trabalho é dedicado à minha mãe, Maria das Graças Barros Ramalho, ao meu pai, Francisco de Assis Ramalho e à minha irmã, Maysa Ramalho Lima, que fizeram de mim tudo o que sou hoje. Dedico também ao meu namorado, Wisley Rocha da Silva, por estar ao meu lado e me apoiar durante essa longa e árdua jornada.

Agradecimentos

Agradeço primeiramente à Deus, por ter me guiado até aqui e ter me dado força para continuar, mesmo quando eu achava que ja não as tinha.

Agradeço à minha mãe, Maria das Graças Barros Ramalho, pelo amor, carinho, compreensão e por todo sacrifício que fez para que hoje eu estivesse aqui.

Agradeço ao meu pai, Francisco de Assis Ramalho, por ter me ensinado todos os valores da vida e por me amar incondicionalmente.

Agradeço à minha irmã, Maysa Ramalho Lima, por me amar como uma filha, por sempre me apoiar e me dar condições de chegar onde cheguei, acreditando em mim a todo momento.

Agradeço ao meu namorado, Wisley Rocha da Silva, por estar ao meu lado, por multiplicar minhas alegrias, dividir minhas tristezas e sonhar meus sonhos comigo.

Agradeço à Profª. Milene Serrano pela atenção e carinho, pelas orientações ao longo desse trabalho, pelo conhecimento que comigo compartilhou. Você é uma referência de pessoa e profissional que quero ser.

Agradeço ao Prof. Maurício Serrano pela coorientação nesse trabalho, pelas horas que disponibilizou para me ajudar, pelos "puxões de orelha"que me deu ao longo do curso. Tenho certeza que serei uma engenheira de software melhor agora.

Agradeço ao Prof. Paulo Meirelles e ao Prof. Hilmer Neri por me darem a oportunidade de abrir novos horizontes na minha formação acadêmica.

Agradeço ao Prof. Ricardo Chaim pelos projetos desenvolvidos ao longo do curso.

Resumo

A robótica está cada vez mais inserida no âmbito educacional, desde a escola até a universidade, proporcionando aos alunos uma experiência única de aprendizagem. Com o lançamento do kit LEGO Mindstorms, a robótica ficou cada vez mais interessante aos olhos dos jovens, pois montar um robô com sensores, motores e programá-lo ficou mais acessível. Ao se popularizar, o kit LEGO Mindstorms tornou-se um dos principais integrantes dos torneios de robótica, até chegar ao ponto de existirem torneios somente para ele, como o Torneio de Robótica FIRST LEGO League. Nesses torneios, a maior preocupação é conseguir ganhar a maior quantidade de pontos possíveis num espaço de tempo relativamente curto, sendo assim, necessária a elaboração de estratégias para otimizar o gasto do tempo. Através da programação dinâmica, o objetivo desse trabalho é a criação de uma máquina de raciocínio para otimizar o ganho de pontos, durante a quantidade de tempo estabelecida, priorizando, dentre as missões não realizadas, quais devem ser executadas pelo robô. Este trabalho será *open-source*, desenvolvido na linguagem de programação Prolog, e testado no robô LEGO Mindstorms NXT, através de uma conexão *bluetooth*.

Palavras-chaves: robótica educacional. máquina de raciocínio. prolog. programação dinâmica. NXT. LEGO.

Abstract

The robotics is more and more inserted in the education extent, since the school until the university, providing to the pupils a single learning experience. With the launch of the kit LEGO Mindstorms, the robotics is more and more interesting to the young people's eyes, because to mount a robot with sensors, motors and to plan it is more accessible. With its popularizing, the kit LEGO Mindstorms became one of the main integrants of the tournaments of robotics, until reaching the point of create tournaments only for it, like the Tournament of Robotics FIRST LEGO League. In these tournaments, the biggest preoccupation is to win the biggest quantity of possible points in short time, and it's necessary the preparation of strategies to optimize the waste of the time. Through the dynamic programming, the objective of this work is the creation of a machine of reasoning to optimize to the profit of points during the established quantity of time, prioritizing, among the missions that weren't realized, those that must be executed by a robot. This work will be open-source, developed in the language of programming Prolog, and tested in the robot LEGO Mindstorms NXT, through a connection bluetooth.

Key-words: educational robotics. reasoning engine. prolog. dynamic programming. NXT. LEGO.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Program Interface
ARM	Advanced RISC Machine
BPMN	Business Process Model and Notation
GPL	General Public License
HDMI	High Definition Multimedia Interface
IDE	Integrated Development Environment
KB	KiloByte
MB	MegaByte
MHz	MegaHertz
MIT	Massachusetts Institute of Technology
PD	Programação dinâmica
RAM	Random Access Memory
RCA	Radio Corporation of America
RIS	Robotics Invention System
RMA	Robôs Móveis Autonomos
SLD	Selection-rule driven Linear resolution for Definite clauses
SMPA	Sense-Model-Plan-Act
USB	Universal Serial Bus

Lista de símbolos

$\subseteq$	Contido em ou igual a
$<$	Menor
$\in$	Pertence
$\cup$	União
$\emptyset$	Vazio

Sumário

1 Introdução

A robótica costuma ser uma área bem atrativa para qualquer pessoa, seja ela conhecedora ou não do assunto. Devido a esse interesse, o ramo da robótica vem se tornando particularmente popular nos últimos anos. A robótica móvel, em especial, vem agaranhando fãs ao redor do mundo, que buscam a possibilidade de contruir seus próprios robôs e programá-los para uma função específica. Os kits de robótica ganharam espaço no mercado, devido à sua completude, pois contém motores e sensores diversos, e devido ao seu baixo custo, se comparado com robôs mais avançados. Com a popularização do uso dos kits de robótica, vários torneios de robótica foram iniciados, atraindo jovens estudantes, universitários e profissionais da área.

1.1 Contextualização

O termo robô surgiu em meados do século XX, derivado da palavra tcheca *robota*, que significa trabalhador forçado (ou escravo) (SILVA, 2009). Desde esta época, a propagação do conceito de robôs está acontecendo de forma acelerada, seja por filmes de ficção científica, ou documentários, ou desenhos animados. Os robôs sairam, de fato, da ficção científica em 1961, quando Joseph Engelberger desenvolveu o primeiro robô comercial, o UNIMATE, e desde então estão cada dia mais inseridos em meio a sociedade, seja como elevadores, caixas eletrônicos, robôs de entretenimento ou de chão de fábrica (MURPHY, 2000 apud SILVA, 2009). Segundo (SILVA, 2009), um robô deve ter, idealmente, os seguintes elementos:

- atuadores: são os meios utilizados para que o robô se locomova e/ou altere a forma de seu corpo. Exemplo: pernas, rodas, articulações, garras, dentre outros;
- sensores: são os meios utilizados pelo robô para medir e conhecer o ambiente, detectando objetos, calor ou luz, e convertendo essa informação em símbolos processados por computadores;
- computador: é o responsável por controlar o robô através de algoritmos nele implementados;
- equipamentos ou mecanismos: são ferramentas ou equipamentos mecânicos.

Um robô pode ser categorizado em um dos três grupos, atualmente conhecidos: manipuladores, móveis e híbridos. Os robôs manipuladores são fixos ao seu local de

trabalho; os móveis se locomevem por meio dos atuadores, e os híbridos são um composto das duas categorias anteriores (RUSSEL; NORVIG, 2004 apud SILVA, 2009).

A robótica é uma ciência, em rápida ascenção, que envolve áreas do conhecimento como: microeletrônica, computação, engenharia mecânica, inteligência artificial, física, neurociência, entre outras. Portanto, estuda tecnologias associadas ao projeto, fabricação, teoria e aplicação dos robôs.

Sendo a robótica uma área tão interdisciplinar, usá-la como instrumento de aprendizagem é um tanto quanto benéfico, pois ensina à criança e/ou ao jovem a trabalhar em equipe, desenvolver o raciocínio lógico através de problemas concretos, e estimula a leitura, exploração, investigação, criatividade e organização. Além de aprimorar a parte motora do indivíduo ao trabalhar com o hardware do robô, também é aprimorado o raciocínio lógico e a abstração ao programar o software do robô.

Em meados dos anos 60, Seymourt Papert, reconhecido matemático, educador e pesquisador do MIT[1] (Instituto de Tecnologia de Massachusetts), criou a linguagem de programação LOGO. Essa linguagem foi utilizada nos kits educacionais da LEGO, conferindo o início do sistema educacional LEGO-LOGO. Nesse sistema, as crianças têm a possibilidade de construir seus robôs protótipos com os blocos de montagem e outros recursos do kit educacional da LEGO bem como programar com a linguagem LOGO, gerando o comportamento desejado nesses protótipos.

Um dos kits de robótica mais populares criados pela LEGO é o Mindstorms, que combina um computador programável, NTX ou RCX dependendo da versão, com motores elétricos, engrenagens, peças de encaixe, polias, roscas, dentre outros. Este kit contêm cerca de mil peças LEGO, incluindo o computador, o CD-ROM do software Mindstorms, um transmissor infravermelho para envio de programas para o robô (apenas para a versão RCX), um guia do construtor, motores, sensores, rodas, pneus, conectores e outros. Para aprimorar o aprendizado, a LEGO disponibiliza, além do kit de peças para montagem dos robôs, um tapete de missões a serem realizadas pelos mesmos. Cada missão do tapete tem uma pontuação máxima, que poderá ser alcançada se a missão for completada dentro do tempo e sem penalidades. É possível completar 'n' missões dentro do tempo limite.

1.2 Justificativa

Retomando aspectos apontados na contextualização, seguem algumas preocupações intrínsecas desse contexto, com as quais procura-se justificar as necessidades: (i) de uma investigação mais detalhada quanto à literatura associada; (ii) da elicitação de soluções candidatas, apoiadas na Engenharia de Software, na Inteligência Artificial e no

[1] http://web.mit.edu

projeto e análise de algoritmos, e (iii) da implementação de uma solução dentre as elicitadas.

O curso de Engenharia de Software da UnB/FGA oferece uma disciplina chamada Fundamentos de Robótica. Dentre os objetivos dessa disciplina, tem-se a intenção de formar grupos de granduandos para competições em robótica, âmbito regional, nacional e internacional. Nesse contexto, são estabelecidos desafios aos grupos de alunos matriculados na disciplina. Esses desafios orientam-se pelas propostas de atividades da First LEGO League (KAMEN; KRISTIANSEN, 2010). Uma preocupação intrínseca dos grupos participantes consiste em lidar com diferentes variáveis, em tempo de competição. No caso, destacam-se: tempo total estabelecido para conclusão das missões, localização atual do robô e pontuação das missões.

No intuito de colaborar na formação de uma equipe competitiva, têm-se como preocupações a serem investigadas nesse Trabalho de Conclusão de Curso, principalmente:

- O fato do tempo total para a conclusão das missões ser relativamente curto para que todas sejam realizadas, logo são necessários algoritmos que permitam a seleção dessas missões considerando o tempo como um fator impactante.

- Não apenas o tempo, mas para realizar essa seleção de forma apropriada, outras variáveis devem ser consideradas, tais como:

  - Localização atual do robô, pois dependendo da posição do robô em relação ao tapete o tempo de locomoção para realizar uma missão irá mudar.

  - Pontuação das missões, pois pode existir uma missão 'X' que gaste o mesmo tempo para ser realizada que uma missão 'Y', porém 'X' tem uma maior pontuação que 'Y', logo será mais valoroso executar a missão 'X' do que a missão 'Y'.

Diante do exposto, acredita-se que a elaboração de uma base de conhecimento orientada ao Paradigma Lógico (TUCKER; NOONAN, 2009, chap. 15), especificamente implementada com base em algoritmos da Inteligência Artificial, conferirá ao robô a capacidade de gerar um roteiro de missões, sendo esse o de maior pontuação possível de ser realizada dentro do tempo restante.

Pretende-se, com tal esforço, agregar valor na formação de uma equipe competitiva em robótica, disponibilizando uma máquina de raciocínio apoiada em algoritmos avançados e instigando os membros da equipe a refinarem essa máquina de forma evolutiva e continuamente.

1.3 Questão de pesquisa

Este TCC buscará responder ao seguinte questionamento: É possível implementar uma inteligência artificial no robô LEGO Mindstorms, i.e. uma máquina de raciocínio lógico, de tal forma que dada uma posição (X0, Y0) e um tempo restante, esse robô consiga gerar um roteiro de missões a serem executadas para alcançar a maior pontuação possível?

1.4 Objetivos

Os objetivos deste trabalho foram classificados quanto à objetivo geral e objetivos específicos, sendo ambos apresentados a seguir.

1.4.1 Objetivo geral

Prover uma máquina de raciocínio lógico, considerando um algoritmo específico para tomada de decisões, com a qual o robô deverá ser capaz de selecionar as missões, compondo um roteiro, de forma que a pontuação seja maximizada em relação ao tempo disponível, previamente estabelecido.

1.4.2 Objetivos específicos

Com base nas colocações apresentadas nos tópicos anteriores, são objetivos específicos deste TCC:

- Investigar algoritmos candidatos à solução da questão de pesquisa considerando, em um primeiro escopo dessa atividade, o estudo de algoritmos gulosos, programação dinâmica e Paradigma Lógico;

- Implementar uma solução que agregue ao robô a capacidade de gerar roteiros de missões de máxima pontuação a partir de um ponto inicial e um tempo restante.

- Estudar o funcionamento do kit educacional LEGO Mindstorms visando a realização de uma pesquisa exploratória/experimental que permita a verificação quanto a pertinência da solução estabelecida para o contexto investigado.

- Estabelecer uma metodologia de desenvolvimento, orientada às boas práticas da Engenharia de Software, no intuito de conduzir o processo investigativo, a implementação da solução bem como a verificação dessa solução no contexto acordado.

1.5 Organização dos capítulos

Este trabalho de conclusão de curso está organizado em seis capítulos, sendo o primeiro dedicado à Introdução. Os demais são brevemente descritos a seguir:

- Referencial teórico: explana sobre a evolução da robótica educacional, principais conceitos do Paradigma Lógico e da programação dinâmica.
- Suporte tecnológico: descreve as tecnologias utilizadas no desenvolvimento tanto da máquina de raciocínio quanto da prova de conceito, apresentando ferramentas, *plugins* e *kits* utilizados.
- Metodologia: aborda, dentre outros detalhes, o fluxo das atividades, o cronograma bem como modalidades de pesquisa e desenvolvimento e a prova de conceito elaborada.
- Proposta: descreve em detalhes a proposta definida a partir do referencial teórico levantado e da prova de conceito elaborada.
- Considerações finais: efetua uma breve reflexão sobre o trabalho e lista atividades previstas para a continuação do mesmo.

2 Referencial teórico

Neste capítulo, serão apresentados alguns conceitos importantes para a melhor compreensão do tema abordado por este projeto. O Capítulo de referencial teórico foi organizado em dois escopos, um mais voltado à bibliografia investigada no domínio da Robótica Educacional e outro mais voltado à Engenharia de Software.

2.1 Robótica educacional

O uso dos robôs pelos seres humanos é datado de antes de Cristo, mas a definição da palavra robótica é recente. A robótica como instrumento de ensino está tomando força nos últimos anos, e vem apoiando muitos educadores na tarefa de fazer o aluno se sentir mais atraído pelo conteúdo, e aprender através de observações e manipulações. A seguir é abordado brevemente a história da robótica, bem como sua definição, seu uso no âmbito educacional e os fundamentos da robótica móvel.

2.1.1 Perspectiva histórica da robótica

Os robôs são máquinas que podem substituir o homem na execução das tarefas, e consequentemente, destinados a melhorar a produção e a qualidade de vida (BARRIENTOS et al., 1997 apud SANTOS, 2002). Desde os primórdios, o homem sente-se atraído por esse mundo das máquinas, como os antigos Egípcios que adicionaram braços mecânicos às estátuas de seus deuses para serem operados por sarcedotes; e os Gregos que construíram estátuas movidas hidraulicamente; e os Chineses que entre os séculos XVIII e XIX contruíram bonecas que transportavam chá (SANTOS, 2002). Os autômatos projetados pelos Gregos tinham uma função mais estética e contemplativa, já os projetados pelos Árabes tinham funcionalidade, deixando a par preocupações estéticas e de entretenimento (SANTOS, 2002).

É perceptível que os homens, há muitos séculos, utilizam os robôs, mas sem o conhecimento da palavra, que foi divulgada a primeira vez em 1921, pelo checoslovaco Karel Capek, no seu romance *Rossum's Universal Robots*. Esse autor descreveu os robôs como máquinas com braços trabalhando duas vezes mais que os humanos, de forma incansável, eficiente e obediente, mas que se tornariam malévolos e dominariam o mundo (SANTOS, 2002). Porém em 1950, Isaac Asimov defendeu, em sua obra literária *I Robot*, que a contrução de robôs seguiria uma linha positiva e benéfica (NOF, 1999 apud SANTOS, 2002), concebendo-os como autômatos de aparência humana mas desprovidos de sentimentos (SANTOS, 2002). Asimov, em sua obra, fala sobre a inteligência dos robôs

de acordo com as seguintes leis da robótica (SANTOS, 2002):

- 1ª Lei: Um robô não pode prejudicar um ser humano, ou quando inativo, deixar um ser humano exposto ao perigo;
- 2ª Lei: Um robô deve obedecer às ordens dadas pelo ser humano, exceto se tais ordens estiverem em contradição com a 1ª Lei.
- 3ª Lei: Um robô deve proteger a sua própria existência desde que essa proteção não entre em conflito com a 1ª e a 2ª Leis.

Em 1985, Asimov acrescentou uma 4ª Lei que diz que "o robô não pode causar mal à humanidade ou por falta de ação permitir que a humanidade sofra perigos". Dessa forma, Asimov popularizou o termo robô e idealizou as três leis fundamentais da robótica (SILVA et al., 2008).

2.1.2 Definição de robô e robótica

A Divisão Internacional de Robótica da Sociedade de Engenharia de Manufatura define um robô como sendo um manipulador reprogramável, multifunções, utilizado para deslocar outros materias ou objetos específicos através da programação de movimentos (SILVA et al., 2008). Um computador é manipulado pelo homem, logo não é considerado um robô, pois este é um mecanismo inteligente que funciona de forma autônoma (CURCIO, 2008).

(PIO; CASTRO; JÚNIOR, 2006) definem a robótica como:

> a ligação inteligente entre a percepção e a ação. Trabalhar em Robótica significa estudar, projetar e implementar sistemas ou dispositivos que, com a utilização de percepção e de certo grau de “inteligência”, sejam úteis na realização de uma determinada tarefa, pré-definida ou não, que envolva interação física entre o sistema (ou dispositivo) e o meio onde a tarefa está sendo realizada.

Porém, uma definição mais formal para o termo robótica seria "uma ciência da engenharia aplicada, tida como uma combinação da tecnologia de máquinas operatrizes e ciência da computação"(GROOVER, 1989 apud REDEL; HOUNSELL, 2004). Assim, a robótica envolve várias disciplinas como engenharia mecânica, elétrica, inteligência artificial e utiliza o robô como principal instrumento (CURCIO, 2008). Os robôs podem ser categorizados em primeira, segunda e terceira geração, sendo progressivamente mais inteligentes (BARRIENTOS et al., 1997 apud SANTOS, 2002).

- Primeira geração: A única função inteligente desses robôs é a apredizagem de uma sequência de ações de manipulação, coordenadas por um operador humano usando

uma unidade de comando. Os robôs dessa geração repetem sistematicamente as tarefas e ignoram possíveis alterações no meio externo, causando restrições no seu uso, como posicionamento no espaço, relacionamento com outras máquinas e a segurança das pessoas que ficam próximas ao robô.

- Segunda geração: esses robôs tiveram a adição de um processador à sua configuração, possibilitando a adaptabilidade ao ambiente, mesmo que de forma mínima, com utilização de sensores para auxiliar na localização, detecção de esforços e adaptação de seus movimentos às informações coletadas. Geralmente a área de atuação destes robôs está ligada à manufatura autômata.

- Terceira geração: é uma versão mais recente dos robôs, onde é incorporado à sua configuração processadores múltiplos em que, cada operação em sicronia desempenha uma tarefa diferente, tornando um mesmo robô multitarefas.

2.1.3 Robótica como instrumento educacional

A róbótica educacional é um meio de inserir a tecnologia no meio educativo como forma de aprendizado, oferencendo aos alunos uma oportunidade de vivenciarem experiências semelhentes às que vivem na vida real.

2.1.3.1 Teorias de aprendizagens

A teoria de aprendizado que fundamentou a róbotica educacional foi a construcionista, criada por Seymour Papert, baseada na teoria construtivista criada por Jean Piaget. Dentre as várias teorias de aprendizado existentes, algumas são brevemente descritas a seguir, segundo (ZILLI et al., 2004).

1. **Teoria das inteligências múltiplas:** Howard Gardner, professor e psicólogo da *Harvard Graduate School of Education*, dedicou-se, desde a década de 80, ao estudo das capacidades simbólicas em crianças com QI normal e alto, desenvolvendo o que ele chamou de "inteligências múltiplas". Essas podem ser vista como contendo três princípios fundamentais:

   - A inteligência não é algo que pode ser visto como tendo múltiplas habilidades. Existem múltiplas inteligências, cada uma com habilidades distintas entre si.
   - As inteligências são interdependentes. Logo, caso seja avaliada a competência de uma pessoa em uma área, esta avaliação não servirá para outras áreas.
   - Apesar de serem interdependentes, as inteligências podem sim trabalhar em conjunto; caso contrário nada seria feito e nenhum problema seria resolvido.

   Gardner identificou essas inteligências distintas e classificou da seguinte forma:

- Inteligência linguística: é relacionada à habilidade da pessoa em produzir a liguagem escrita e falada.
- Inteligência lógico-matemática: é relacionada à habilidade para explorar relações, categorias e padrões, através da manipulação de objetos ou símbolos, e à habilidade de resolver problemas através do raciocínio.
- Inteligência musical: é relacionada, como o próprio nome já diz, ao reconhecimento da estrutura musical, à sensibilidade dos sons, à criação de melodias e ritmos, à percepção das qualidades dos tons e à habilidade para tocar instrumentos.
- Inteligência espacial: é relacionada à habilidade do indivíduo de visualizar um objeto e ter uma percepção acurada de diferentes ângulos, relações de espaço, dentre outros.
- Inteligência cinestésica: é representada pela capacidade de manipular objetos habilmente e controlar os movimentos do corpo.
- Inteligência interpessoal: é relacionada à capacidade de diferenciar e dar uma resposta aos estados de humor, temperamentos, desejos e motivações das outras pessoas.
- Inteligência intrapessoal: é relacionada à habilidade para ter acesso aos próprios sentimentos, aos estados interiores do ser, e saber utilizá-los na solução de problemas pessoais.

Segundo Gardner, a escola valoriza mais a inteligência linguística e a lógico-matemática, e o uso do computador pode colaborar para o amadurecimento das outras inteligências, por ser uma ferramenta adaptável às mais diversas formas de uso.

2. **Teoria do ensino por competências:** Philippe Perrenoud, professor e sociólogo suiço, é referência para diversos educadores com suas idéias pioneiras e vanguardistas sobre profissionalização de professores e avaliação de alunos. Perrenoud afirma que "a trilogia das habilidades de ler, escrever, contar, que fundou a escolaridade obrigatória no século XIX não está mais a altura das exigências da nossa época". Assim, Perrenoud propôs o ensino por competências que considera que os saberes são ferramentas para a ação e que se aprende a usá-los. O Centro de Referência Educacional elencou dez competências que devem ser trabalhadas pela escola:

   - Respeitar as identidades e as diferenças;
   - Utilizar-se das linguagens como meio de expressão, comunicação e informação;
   - Inter-relacionar pensamentos, ideias e conceitos;
   - Desenvolver o pensamento crítico e flexível e a autonomia intelectual;

- Adquirir, avaliar e transmitir informações;
- Compreender os princípios das tecnologias e suas relações integradoras;
- Entender e ampliar fundamentos científicos e tecnológicos;
- Desenvolver a criatividade;
- Saber conviver em grupo;
- Aprender a aprender.

3. **Teoria do construtivismo:** O suiço Jean Piaget estudou como o aprendiz passa de um estado de menor conhecimento para outro de maior conhecimento, e desenvolveu a teoria do construtivismo que enfoca o conhecimento científico na perspectiva do indivíduo que aprende. Nessa teoria, o sujeito é um ser ativo que estabelece relação de troca com o meio-objeto, relações essas que devem ser vivenciadas e significativas, assim, o indivíduo incorpora novas informações, que passa a tornar parte do conhecimento. Segundo (ZILLI et al., 2004),

> O construtivismo defende que o conhecimento não é uma cópia da realidade, mas sim uma construção do ser humano em conseqüência da sua interação com o ambiente e resultado de suas disposições internas.

   Piaget classifica os períodos de inteligência em estágios, organizados do nascimento até a fase adulta. Esses estágios indicam saltos bruscos nas capacidades do indivíduo, pois as capacidades cognitivas sofrem uma forte reestruturação.

4. **Teoria do construcionismo:** Seymour Papert, psicólogo do Laboratório de Inteligência Artificial do MIT, adaptou os princípios do construtivismo definido por Piaget e criou a teoria construcionista, que considera o computador como uma ferramenta para a construção do conhecimento e desenvolvimento do aluno. Duas ideias principais sobre a construção do conhecimento fazem com que o construcionismo se diferencie do construtivismo:

   - O aprendiz é quem constrói o seu conhecimento, atravéz das coisas que faz;
   - O aprendiz constrói algo do seu interesse e que o motiva.

   Segundo Papert, a criança aprende de forma mais eficaz quando, por ela mesma, atinge o conhecimento específico de que precisa. O contrucionismo tem como meta ensinar de forma a produzir o maior conhecimento possível com o mínimo de ensino.

2.1.3.2 Conceito da robótica educacional

(MAISONNETTE, 2002) define róbotica como sendo

> o controle de mecanismos eletro-eletrônicos através de um computador, transformando-o em uma máquina capaz de interagir com o meio ambiente e executar ações decididas por um programa criado pelo programador a partir destas interações.

e diz que a robótica educacional é a aplicação dessa tecnologia como mais uma ferramenta de ensino para que os alunos vivenciem de forma real o que estudam, tendo a oportunidade de propor e solucionar problemas difíceis ao invés de apenas observar a solução. Através da robótica educacional os alunos podem explorar ideias e descobrir novos caminhos na aplicação de conceitos que aprendem em sala de aula, assim adquirem a capacidade de produzir hipóteses, averiguar soluções, estabelecer relações e inferir conclusões (BENITTI et al., 2009). Além de benéfico ao aluno, a robótica educacional colabora com o professor no ensino de muitos conceitos teóricos de difícil compreensão, motivando o aluno a observar, abstrair e inventar. O conhecimento adquirido através de observação, abstração e do esforço do aluno tem muito mais significado para ele, e se adapta melhor às suas estruturas mentais (ZILLI et al., 2004).

2.1.3.3 Objetivos da robótica educacional

Segundo (ZILLI et al., 2004), além de propiciar o conhecimento da tecnologia atual, a robótica educacional pode desenvolver as seguintes competências:

- raciocínio lógico;
- habilidades manuais e estéticas;
- relações inter e intrapessoais;
- utilização dos conceitos aprendidos em diversas áreas do conhecimento para o desenvolvimento de projetos;
- investigação e compreensão;
- representação e comunicação;
- trabalho com pesquisa;
- resolução de problemas por meio de tentativa e erro;
- aplicação das teorias às atividades concretas;
- utilização da criatividade em diversas situações, e
- capacidade crítica.

(ZILLI et al., 2004) apresenta ainda uma classificação dos objetivos:

- Objetivos Gerais
  - Contruir maquetes que utilizem lâmpadas, motores e sensores;
  - Trabalhar conceitos de desenho, física, álgebra e geometria;
  - Conhecer e aplicar princípios de eletrônica digital, e
  - Construir ou adaptar elementos dinâmicos como engrenagens, redutores de velocidade de motores, entre outros.
- Objetivos Psicomotores
  - Desenvolver a motricidade fina;
  - Proporcionar a formação de habilidades manuais;
  - Desenvolver a concentração e observação, e
  - Motivar a precisão de seus projetos.
- Objetivos Cognitivos
  - Estimular a aplicação das teorias formuladas às atividades concretas;
  - Desenvolver a criatividade dos alunos;
  - Analisar e entender o funcionamento dos mais diversos mecanismos físicos;
  - Ser capaz de organizar suas idéias a partir de uma lógica mais sofisticada de pensamento;
  - Selecionar elementos que melhor se adequam à resolução dos projetos;
  - Reforçar conceitos de matemática e geometria;
  - Desenvolver noções de proporcionalidade;
  - Desenvolver noções topológicas;
  - Reforçar a aprendizagem da liguagem Logo;
  - Introduzir conceitos de robótica;
  - Levar à descoberta de conceitos de física de forma intuitiva;
  - Utilizar conceitos aprendidos em outras áreas do conhecimento para o desenvolvimento de um projeto, e
  - Proporcionar a curiosidade pela investigação levando ao desenvolvimento intelectual do aluno.
- Objetivos Afetivos
  - Promover atividades que geram a cooperação em trabalhos em grupo;
  - Estimular o crescimento individual através da troca de projetos e ideias;

- Garantir que o aluno se sinta interessado em participar de discussões e trabalhos em grupo;
- Desenvolver o senso de responsabilidade;
- Despertar a curiosidade;
- Motivar o trabalho de pesquisa;
- Desenvolver a autoconfiaça e a auto-estima, e
- Possibilitar a resolução de problemas por meio de erros e acertos.

2.1.4 Fundamentos da robótica móvel

O foco da robótica vem se modificando ao longo dos anos. Tempos atrás, os braços mecânicos, também chamados de manipuladores, tinham um maior destaque junto a mídia e a sociedade em geral (WOLF et al., 2009) sendo até, um dos principais fatores para a Revolução Industrial. Nos tempos atuais, o destaque é a robótica móvel, que trata de máquinas capazes de se movimentar de forma independente, podendo ser terrestre, aquático, voador ou até espacial (VIEIRA, 2005 apud RINCON, 2014), movidos por rodas, esteiras, patas, hélices e dentre outros.

Segundo (WOLF et al., 2009) os Robôs Móveis Autônomos (RMA) possuem como características fundamentais

> as capacidades de locomoção e de operação de modo semi ou completamente autônomo. Também deve ser considerado que maiores níveis de autonomia serão alcançados somente à medida que o robô passe a integrar outros aspectos considerados da maior importância, como: *capacidade de percepção* (sensores que conseguem "ler" o ambiente onde ele atua), *capacidade de agir* (atuadores e motores capazes de produzir ações, tais como o deslocamento do robô no ambiente), *robustez e inteligência* (capacidade de lidar com as mais diversas situações, de modo a resolver e executar tarefas por mais complexas que sejam).

O termo Controle Robótico Inteligente é referente ao uso de técnicas de planejamento e controle para a navegação e operação autônoma dos robôs, que permitem que os RMAs tenham a capacidade de executar as mais diversas e complexas tarefas. Esse controle inteligente é possível devido ao uso de diversos sensores e atuadores que combinados conferem ao robô a possibilidade de planejar e realizar o acionamento de seus dispositivos de modo a executar a ação desejada. Alguns sensores e atuadores para robôs móveis são listados respectivamente, nas Figuras 1 e 2, a seguir apresentadas.

Sensor	Principal Função	Exemplos
De Posição e Orientação	Determinar a posição absoluta	GPS (Sistema de Posicionamento Global)
	ou direção de orientação do robô	Bússola [Compass]
		Inclinômetro
		Triangulação usando marcas (Beacons)
De Obstáculos	Determinar a distância até um objeto	Sensor Infra-Vermelho (IR - Infrared)
	ou obstáculo	Ultrassom (Sonar)
		Radar
		Sensor Laser (Laser rangefinder)
		Sistemas de Visão Estéreo (Stereo Vision)
De Contato	Determinar o contato com um objeto	Sensores de Contato (Bumbers, Switches)
	ou posição de contato com marcação	Antenas e "bigodes" (Animal whiskers)
		Marcações (barreiras óticas e magnéticas)
De Deslocamento	Medir o deslocamento do robô	Inercial (Giroscópio, Acelerômetros)
e Velocidade	Medidas relativas da posição e	Odômetro (Encoders: Optical, Brush)
	orientação do robô	Potenciômetros (Angular)
		Sensores baseados em Visão
Para Comunicação	Envio e recepção de dados e sinais	Sistemas de Visão e Sensores Óticos
	externos (troca de informação)	Sistemas de Comunicação (RF)
Outros tipos	Sensores magnéticos, indutivos, capacitivos, reflexivos	
	Sensores de temperatura, carga (bateria), pressão e força, etc.	
	Detectores: detector de movimento, de marcações, de gás/odores	

Figura 1 – Sensores para robôs móveis (WOLF et al., 2009)

Atuador	Principal Tipo/Função	Exemplos
Base Fixa	Braço robótico com base fixa	Robôs industriais PUMA
Base Móvel: Rodas	2 Rodas independentes (diferencial)	Robôs Khepera e Pioneer P3-DX
	3 Rodas (triciclo, omni-directionais)	Robô BrainStem PPRK
	4 Rodas (veículos robóticos - ackermann)	Stanley - Stanford (Darpa Challenge)
Base Móvel: Esteira	Esteira (Slip/Skid locomotion - tracks)	Tanques e veículos militares
Base Móvel: Juntas e	Bípedes	Robôs Humanóides
Articulações	4 Patas (quadpods)	Robôs Sony Aibo, BigDog
	6 Patas (hexapods)	Robôs Inseto (Lynxmotion Hexapods)
Base Móvel: Propulsão	Veículos aéreos com hélices	Aviões, Helicópteros e Dirigíveis
Hélices ou Turbinas	Veículos aquáticos com hélices	Barcos autônomos
	Veículos sub-aquáticos	Submarinos autônomos
Outros tipos	Braços manipuladores com base móvel	Garras (Grippers) embarcadas
	Garras com ou sem feed-back sensorial	Mão robótica
	Mecanismos de disparo	Disparo do chute (futebol de robôs)

Figura 2 – Atuadores para robôs móveis (WOLF et al., 2009)

Os sensores conferem ao robô uma visão parcial, incompleta e sujeita a erros, sendo tarefa do controle inteligente adquirir, unificar e tratar as informações providas pelos sensores. Os comandos dos atuadores também são imprecisos, pois estes estão sujeitos a erros de posicionamento do robô e acionamento dos motores além das forças externas, como fricção, gravidade, colisão com obstáculos, derrapagem de rodas, dentre outros. Nesse contexto, cabe ao controle inteligente prover técnicas que permitam contornar estes cenários, corrigindo os erros de modo que as tarefas do robô sejam executadas corretamente (WOLF et al., 2009).

Um projeto de sistema de controle de robô deve levar em conta uma série de quesi-

tos como: i)tipo de tarefa do robô; ii)tipo e precisão dos sensores embarcados, e iii)tipo e precisão dos atuadores. Além de adotar uma arquitetura específica de controle que seja capaz de realizar algumas tarefas específicas, ou no melhor dos casos todas as tarefas, sendo elas: i) fusão de sensores; ii)desviar de obstáculos; iii)auto-localização; iv)mapeamento do ambiente v)planejamento de trajetórias; vi)planejamento de ações; vii)navegação robótica e viii)interação e comunicação. O detalhamento de cada tópico pode ser conferido na íntegra na obra de (WOLF et al., 2009). Para que essas tarefas sejam planejadas e executadas gerenciando todos os dispositivos embarcados no robô é necessário fazer o uso das arquiteturas computacionais de controle de robôs móveis autômatos. Essas arquiteturas computacionais são relativas à percepção, raciocínio/decisão e ação em relação ao ambiente (WOLF et al., 2009). Dentre os diversos tipos de arquiteturas computacionais, as mais conhecidas e reconhecidas são descritas nos tópicos a seguir.

2.1.4.1 Controle reativo

O controle reativo consiste em um sistema de reação sensorial-motora que considera apenas as leituras sensoriais realizadas no presente para fins de tomada de decisão e geração de comandos de ação (WOLF et al., 2009). Esse tipo de arquitetura computacional tem por objetivo possibilitar a implementação de sistemas de controle com rápida resposta a um grande número de ocorrências ou situações de ambiente, permitindo que os robôs atuem em ambientes dinâmicos, devido a simplicidade no tratamento das informações sensoriais e a forma direta pela qual a percepção está associada com a ação (GRASSI, 2006 apud SANTOS, 2009).

(WOLF et al., 2009) defende que o sistema reativo é muito útil para implementar comportamentos.

> Um sistema reativo é bastante útil para implementar comportamentos elementares como: desviar de obstáculos (*avoid collision behaviour*: reage a presença de um obstáculo), seguir um objeto (*wall-following behaviour*: acompanhar um elemento guia) e seguir uma fonte luminosa (*phototaxis behaviour*: mover em direção a uma fonte de luz).

2.1.4.2 Controle deliberativo

A arquitetura deliberativa consiste em aplicar um mecanismo de planejamento e tomada de decisão, que estabeleça um plano prévio de execução de uma sequência de ações, baseado no modelo interno de conhecimento do mundo que o robô possui. Esse conhecimento prévio do mundo confere ao robô a possibilidade de otimizar seu desempenho em relação ao modelo interno.

O robô pode ter seu modelo interno representado de duas formas diferentes, simbólico e geométrico. O modelo simbólico é baseado em lógica, utilizando, por muitas vezes,

a inteligência artificial. Já no modelo geométrico o mundo é representado espacialmente, por espaços livres e com obstáculos (SANTOS, 2009).

Tendo em vista que o robô possui esse modelo interno de mundo previamente estabelecido, ele possui limitações quando colocado em um mundo dinâmico, pois será necessário reconhecer o novo obstáculo, acrescentar ao seu modelo e reestabelecer o plano de execução (SANTOS, 2009). Devido a essa limitação, (WOLF et al., 2009) diz que o melhor seria a combinação dos modelos reativos e deliberativos, assim o robô teria capacidade de reação e de planejamento de tarefas complexas.

2.1.4.3 Controle hierárquico

O controle hierárquico consiste na combinação de múltiplos módulos de controle (reativo e/ou deliberativo) dispostos em forma hierárquica, podendo ser com decomposição vertical ou horizontal para definir um esquema de prioridades em relação às múltiplas camadas do sistema (WOLF et al., 2009).

O controle hierárquico com decomposição vertical possui os módulos de controle reativo dispostos de forma vertical, mantendo a prioridade de baixo para cima, sendo a camada mais baixa a mais prioritária. Um exemplo de uso desse modelo é quando a ação de desviar de um obstáculo, que está mais baixa, tem prioridade à ação de seguir em frente, que está mais acima.

O controle hierárquico com decomposição horizontal é adotado para controle de módulos deliberativos, implementando um "pipeline"que decompõe as funções em camadas executadas em cascata, formando uma linha de ação horizontal. Um exemplo desse modelo é o tipo Sense-Model-Plan-Act (SMPA) que possui um longo ciclo de resposta ao estímulo de entrada devido as suas quatro camadas horizontais que serão executadas em cascada para cada estímulo.

2.1.4.4 Controle híbrido

A arquitetura de controle híbrido nada mais é do que um controle reativo sendo planejado por um controle deliberativo. Inicialmente, é realizado um plano de execução pelo módulo deliberativo. Depois de pronto, o plano é executado por um módulo reativo de maior prioridade, o qual pode intervir nesse plano quando encontra algum obstáculo em meio à execução das ações. Nesse controle, pode-se ainda ter um módulo de auto-localização que monitora constantemente a posição atual do robô e a corrige, se necessário.

Este tipo de arquitetura é uma das mais sofisticadas e adotadas na implementação dos sistemas de controle dos robôs móveis autônomos modernos, devido a sua eficiência em atingir os objetivos do robô.

2.2 Engenharia de software

O estudo da robótica na Universidade tem um foco considerável na inteligência artificial, e uma das áreas desta é a otimização. A seguir são brevemente descritos os dois algoritmos de otimização abordados por este trabalho, o algoritmo guloso e a programação dinâmica, bem como o paradigma lógico, também tema deste trabalho.

2.2.1 Algoritmo guloso

Os algoritmos gulosos tomam uma decisão levando em consideração apenas a informação disponível no dado momento, sem levar em conta as consequências da decisão, nunca reconsiderando uma decisão já tomada. Dessa forma, o algoritmo guloso toma uma decisão ótima naquele momento (decisão ótima local), e espera que essa decisão seja também a ótima global (ROCHA, 2004).

2.2.1.1 Características do algoritmo guloso

Segundo (ROCHA, 2004), os problemas e os algoritmos gulosos que os resolvem são caracterizados pelos itens a seguir.

- Para que exista uma solução ótima de um problema deve existir um conjunto de candidatos;
- Ao executar um algoritmo guloso dois conjuntos são criados: um que contêm os elementos que foram analisados e rejeitados e outro os elementos que foram analisados e escolhidos;
- Um algoritmo guloso contêm quatro funções:
  - a primeira avalia se o conjunto candidato produz uma solução para o problema;
  - a segunda verifica a viabilidade do conjunto de candidatos, ou seja, se é possível acrescentar mais candidatos a esse conjunto;
  - a terceira função, chamada de função seleção, avalia qual dos candidatos restantes é o melhor para ser acrescentado no conjunto;
  - a quarta e última função, chamada função objetivo, retorna o valor da solução encontrada.

É possivel dizer que um algoritmo guloso trabalha da seguinte forma: inicialmente contém um conjunto S que está vazio, ou seja, nenhum candidato foi escolhido. A cada passo a função seleção é utilizada para determinar qual o melhor candidato. Se o elemento analisado não é viável, ignora-se o termo que está sendo avaliado no momento. Se for viável, adiciona o elemento ao conjunto S. O elemento avaliado, sendo ele aceito ou rejeitado,

será avaliado apenas uma vez, não havendo reconsiderações. A cada vez que o conjunto de candidatos escolhidos (S) é ampliado, é verificado se a solução do problema foi obtida (ROCHA, 2004).

2.2.1.2 Elementos da estratégia gulosa

Como exposto, a estratégia utilizada pelo algoritmo guloso para encontrar a solução ótima do problema em questão é em cada passo escolher a solução que parece ser ótima naquele momento. Nem sempre uma estratégia gulosa é capaz de chegar à solução de um problema, mesmo esta existindo. Existem duas características que podem indicar se o problema pode ou não ser resolvido através de uma estratégia gulosa.

- **Propriedade da escolha gulosa:** Segundo a propriedade da escolha gulosa, uma solução globalmente ótima pode ser alcançada escolhendo uma solução localmente ótima. Para se provar que uma escolha localmente ótima em cada um dos passos irá levar a uma solução ótima é necessário examinar uma solução ótima global para algum subproblema. Posteriormente, deve-se mostrar que essa solução pode ser modificada em uma solução gulosa. Tal estratégia irá resultar em um subproblema menor, mas similiar.
- **Subestrutura ótima:** Esta característica também é importante quando se utiliza a programação dinâmica, vide seção 2.2.2. Um problema possui uma subestrutura ótima quando uma solução ótima para o problema contêm, dentro dela, soluções ótimas para os subproblemas.

2.2.1.3 Fundamentos do algoritmo guloso

A teoria de algoritmos gulosos envolve o conceito de matróide que por definição é um par (E,I) onde E é um conjuto finito e $I \subseteq 2^E$, que satisfaz os seguintes axiomas (CASTALONGA; LEMOS et al., 2007):

(i) $\emptyset \in I$;

(ii) Se $X \subseteq Y$ e $Y \in I$, então $X \in I$;

(iii) Se $X, Y \in I$ e $|X| < |Y|$ então existe $e \in Y - X$ tal que $X \cup e \in I$.

Esse tipo de estrutura não cobre todo tipo de problema que pode ser resolvido pelo algoritmo guloso, mas cobre muitos casos de interesse prático.

Lema: todo conjunto independente maximal em um matróide tem o mesmo tamanho. **Prova:** Por contradição. Se existir dois conjuntos independentes maximais A e B e $|A| < |B|$, deve existir um elemento $x \in (B - A)$ tal que $(A + x) \in I$. Isso contradiz o fato de que A é um conjunto maximal.

2.2.2 Programação dinâmica

A programação dinâmica, assim como os métodos de "dividir para conquistar", combina as soluções dos subproblemas para resolver um problema. Esses métodos particionam os problemas em subproblemas de forma que resolvendo os subproblemas, recursivamente, resolve-se o problema. Já a programação dinâmica aplica-se quando os subproblemas se particionam em subsubproblemas. Um algoritmo comum resolve repetidamente essas partições, trabalhando mais que o necessário. Em contrapartida, o algoritmo dinâmico resolve cada subsubproblema apenas uma vez, salvando o resultado em uma tabela, evitando calcalar repetidas vezes o mesmo problema (CORMEN, 2009). Um problema pode ter várias soluções possíveis, cada uma com um valor, e a solução com o melhor valor (maior ou menor) é dita a solução ótima para o problema. Segundo (CORMEN, 2009) para se desenvolver uma solução de programação dinâmica devem ser seguidos quatro passos:

- Caracterizar a estrutura de uma solução ótima;
- Recursivamente definir o valor de uma solução ótima;
- Calcular o valor de uma solução ideal, normalmente de forma ascendente, e
- Construir uma solução ótima a partir de informações computadas.

Os três primeiros itens formam a base de uma solução dinâmica. Mas, quando utilizar a programação dinâmica para resolver um problema? O primeiro passo para resolver um problema de otimização por programação dinâmica é caracterizar uma subestrutura ótima, sempre que um problema exibi-la. Para caracterizar o espaço de subproblemas, deve-se manter o espaço o mais simples possível e, em seguida, expandi-lo, se necessário. Uma subestrutura ótima varia entre os domínios de problema de duas formas:

(i) quantos subproblemas uma solução ideal utiliza para o problema, e

(ii) quantas escolhas existem para determinar quais subproblemas usar em uma solução ótima.

Ou seja, pode ser que um problema tenha apenas um subproblema (de tamanho n - i), mas deve-se considerar n escolhas para i, afim de determinar quais escolhas produzem uma solução ótima.

2.2.2.1 *Memoization* vs Iteração sobre subproblemas

A palavra *memoization* vem do termo *memoize* que deriva de um termo latim *memorandum* que quer dizer relembrando. Essa é uma técnica *top-down* que armazena em

uma tabela a solução de cada subsubproblema, evitando assim o retrabalho e aumentando a eficiência. Essa técnica utiliza um algoritmo recursivo.

A iteração sobre subproblemas é uma técnica *buttom-up* que utiliza um algoritmo iterativo para calcular os subproblemas. Essa técnica começa a calcular do menor subproblema para o maior.

2.2.3 Paradigma lógico

Os literais são fórmulas atômicas positivas ou negativas, que em conjunto formam uma cláusula. Um tipo de cláusula é a chamada cláusula de Horn (ou cláusula definitiva), onde cada cláusula tem no máximo um literal positivo. Essas cláusulas são utilizadas pela resolução SLD (Selection-rule driven Linear resolution for Definite clauses), que restringe o uso de literais para tornar as implementações mais eficientes. Essa resolução recebe como entrada um programa lógico P e uma consulta N, e efetua a resolução entre a consulta N e alguma regra de P. O ganho de eficiência vem do fato de que, para cada regra é possível a seleção de apenas um literal para a resolução com N (RODRIGUES, 2010). A programação lógica surgiu da aplicação da SLD no processamento da linguagem natural da inteligência artificial, e esse processamento serviu como primeira aplicação e motivo imediato para o desenvolvimento do Prolog (Programation en Logique) (RODRIGUES, 2010).

2.2.3.1 Prolog

O Prolog é uma linguagem de programação utilizada para resolver problemas que envolvam objetos e relações entre objetos. O termo objeto em Prolog não se refere a uma estrutura de dados que pode herdar variáveis e métodos de uma classe, mas se refere às coisas que podemos representar usando termos (CLOCKSIN; MELLISH, 2003). Um programa Prolog consiste em um conjunto de cláusulas, onde cada cláusula ou é um fato sobre a informação dada ou uma regra sobre como a solução pode relacionar ou ser inferida a partir dos fatos dados (CLOCKSIN; MELLISH, 2003). Segundo (CLOCKSIN; MELLISH, 2003), a programação em Prolog consiste em:

- especificar algum fato sobre os objetos e seus relacionamentos;
- definir algumas regras sobre objetos e seus relacionamentos, e
- fazer perguntas sobre os objetos e seus relacionamentos.

O Prolog pode fazer muito mais do que apenas responder sim ou não às perguntas feitas, ele permite que um computador seja usado como um armazém de fatos e regras, fornecendo maneiras de se fazer inferências a partir de um fato ou outro e encontrar os valores das

variáveis que levam a uma dedução lógica. Algumas definições base da linguagem Prolog são descritas a seguir.

- **Fato:** um fato é a definição das relações entre os objetos. Para implementar essas definições, algumas regras devem ser seguidas i) os nomes de todas as relações e objetos devem começar com letra minúscula; ii) a relação é escrita primeiro seguida do(s) objeto(s) entre parênteses (caso tenha mais de um separar com vírgula); iii) um fato deve terminar com ponto-final. A relação entre os objetos é chamada de predicado, e os objetos da relação são chamados de argumentos.

- **Questão:** a questão tem a escrita muito parecida com o fato, exceto que começa com um ponto de interrogação. Para cada pergunta feita ao interpretador de Prolog, ele irá procurar na sua base de conhecimento fatos ou regras que comprovem aquele questionamento.

- **Variáveis:** é um nome com letra maiúscula. Quando utilizado em uma pergunta, faz com que o intepretador de PROLOG procure na base de conhecimento um ou mais objetos que sejam equivalentes àquela variável. Como por exemplo, nessa base de conhecimento exitem três fatos:

  ```
  gosta(maria, pedro).
  gosta(maria, joao).
  gosta(laura, joao).
  ```

  Quando for perguntado ao prolog:

  ```
  ?- gosta(maria, X).
  ```

  O interpretador Prolog responderá:

  ```
  X = pedro;
  X = joao.
  ```

- **Conjunção:** é o operador vírgula (,) tendo função de "e". Quando utilizado na pergunta, por exemplo, o interpretador de Prolog procura na base de conhecimento fatos que comprovem as duas perguntas. Usando a base de conhecimento do item anterior, se perguntar ao interpretador de Prolog:

  ```
  ?- gosta(maria, X) , gosta(laura, X).
  ```

  ele responderá:

  ```
  X = joao.
  ```

  Para responder a esse questionamento, o interpretador de Prolog procura primeiramente qual o objeto que pode ser a variável X, achando inicialmente o argumento

pedro. Em seguida, ele procura se existe algum fato que Laura goste de Pedro. Caso não exista esse fato na base de conhecimento, ele volta para buscar outro objeto que possa ser a variável X, encontrando o argumento joão. Ao buscar por algum fato que comprove que Laura gosta de João, o interpretador encontra e retorna "sim"para a pergunta, ou seja, Maria gosta de João e Laura também. Caso existissem mais pessoas em comum que Maria e Laura gostassem, era só digitar ";"seguido de Enter e o interpretador de Prolog buscaria por outros objetos para X.

- **Regras:** uma regra é uma declaração geral sobre objetos e seus relacionamentos. Em Prolog, uma regra é comumente utilizada para dizer que um fato depende de um conjunto de outros fatos. Em Prolog, uma regra contém uma cabeça e um corpo, ligados por pelo símbolo ":- ", que é constituído por dois pontos e um hífen, e é pronunciado "se".

2.3 Considerações parciais

Trabalhar a robótica dentro do âmbito escolar colabora no aprendizado de forma multidisciplinar. A robótica móvel está sendo muito utilizada pelas escolas, principalmente o kit da LEGO, para trabalhar nas crianças a parte motora, ao montar o robô, a parte intelectual, para construir os programas, e a parte social, para aprender a trabalhar em grupo.

Já na universidade, a robótica está muito associada à ideia de inteligência artificial, tendo um foco maior para as áreas de automação e otimização. Na área de otimização, o algoritmo guloso e a programação dinâmica são conceitos bem trabalhados e conhecidos. A utilização da linguagem de programação Prolog na área da robótica não é tão vasta quanto de outras linguagens, como C e C++, mas está sendo cada vez mais adotada.

Diante do conteúdo estudado neste capítulo, o foco desse trabalho se dará na utilização da programação dinâmica em conjunto com a linguagem de programação Prolog e o kit de robótica móvel Lego Mindstorms NXT.

3 Suporte tecnológico

Para dar suporte ao desenvolvimento do projeto foi feito um levantamento de ferramentas candidatas, sendo este refinado e descrito nesse capítulo. Assim como o capítulo anterior, este capítulo, de suporte tecnológico, foi organizado em dois escopos, um mais voltado ao suporte tecnológico investigado no domínio da Robótica Educacional e outro mais voltado à Engenharia de Software.

3.1 Robótica educacional

Um dos maiores precursores da robótica no âmbito educacional foi Seymour Papert ao criar a linguagem LOGO com o intuito de incentivar a aprendizagem da matemática, baseado nas idéias do suiço Jean Piaget que dizia, *"as funções essenciais da inteligência consistem em compreender e inventar, em outras palavras, construir estruturas estruturando o real"*, ou seja, é essencial para a formação da inteligência a ação sobre objetos, e por meio dessa descobrir propriedades através de abstração. Desde então, a robótica educacional vêm se tornando uma plataforma atraente para criar envolvimento nos estudantes, incentivando o estudo da ciência e da tecnologia. As plataformas iterativas/lúdicas vêm ganhando espaço ao longo dos anos como ferramentas auxiliares às metodologias de ensino que utilizam a robótica educacional. Dentre essas plataformas podemos citar: Robomind[1], Scratch[2], Arduino[3], Raspberry Pi[4] e o Kit Lego Mindstorms[5]. Essas plataformas serão brevemente descritas a seguir.

3.1.1 Robomind

O Robomind é uma IDE para a programação dos movimentos de um robô em um mundo bidimensional, através de uma linguagem de programação bem simples e intuitiva, ideal para iniciantes. A interface da IDE é composta por quatro partes: à cima contêm o menu e os atalhos para as funções mais utilizadas, como inserir qualquer movimento; à esquerda encontra-se a área de edição do código; à direita contém o mundo bidimensional, onde é possível visualizar o robô realizando os movimentos, previamente determinados pelo código, em um mapa; e na parte inferior encontra-se o controle de execução do programa juntamente à área de mensagens utilizadas pela IDE para informar erros sintáticos ou durante a execução do código, e a situação na qual o robô se encontra. O Robomind

[1] www.robomindacademy.com
[2] www.scratch.mit.edu
[3] www.arduino.cc
[4] www.raspberrypi.org
[5] mindstorms.lego.com

é gratuito para teste durante 30 dias. Após esse periodo, é possível comprá-lo por USD 14.00 por ano. Ele pode ser instalado em vários idiomas, dentre eles o Português[6], além de ser multiplataforma, Linux, Mac e Windows. Com o Robomind só é possível simular o comportamento do que é implementado, não trabalhando a montagem de um robô, programação embarcada e utilização de sensores.

3.1.2 Scratch

O Scratch é um ambiente e uma linguagem de programação multimídia, produzido pelo *MediaLab*[7] do MIT, para ser utilizado por crianças a partir dos oito anos de idade na criação de histórias iterativas, animações, jogos, músicas, dentre outros, ditos projetos Scratch. Um projeto Scratch contém personagens programáveis, que podem tratar eventos vindos do teclado ou do mouse, para mudança de posição e direção, dentre outras funcionalidades. Assim como o Robomind, ele utiliza uma linguagem de programação em blocos, com encaixe seletivo ideal para programadores iniciantes. O Scratch é gratuito, *online* e também tem a opção de utilizá-lo em português. Similar ao Robomind, no Scratch só é possível simular de forma limitada a implementação, não sendo possível testar em um robô físico nem utilizar sensores.

3.1.3 Arduino

O Arduino é uma plataforma de prototipagem eletrônica *open-source*, criada em 2005 pelo italiano Massimo Banzi, para auxiliar no ensino de eletrônica visando o baixo custo para os alunos. A plataforma é composta por um *hardware* e um *software* bastante flexíveis. O *hardware* é constituído de uma placa com um microprocessador; porta USB, usada para comunicação serial com o computador; pinos digitais, utilizados para detecção ou transmissão de controles digitais; pinos analógicos, usados para leitura de sinais de sensores; e pinos de alimentação, usados para alimentação de circuitos externos. O *software* é uma IDE, onde será programado o código, conhecido como *sketch*, e por meio desta será passado à placa através de uma comunicação serial. Nesta IDE, é utilizada a linguagem de programação Arduino, mas quando é passado para a placa, esta linguagem é traduzida para a linguagem estruturada C. A IDE é multiplataforma, podendo ser instalada em Linux, Mac e Windows. O Arduino tem a vantagem de utilizar programação embarcada no robô e possuir entradas para diversos sensores, porém a montagem do robô fica a critério do desenvolvedor não tendo padronização alguma.

[6] www.robomind.net/pt/
[7] www.media.mit.edu

3.1.4 Raspberry Pi

O Raspberry Pi é um computador versátil e flexível, criado pela *Raspberry Pi Foundation*[8] em conjunto com a Universidade de Cambrigde, no ano de 2012, com o objetivo de estimular o ensino da ciência da computação em jovens do ensino básico. O Raspberry Pi possui dois modelos, ambos equipados com processador multimídia *Broadcom* BCM2835 *system-on-chip* (SoC) de 700 Mhz com placa gráfica integrada VideoCore IV; entrada para cartão de memória, que se faz necessário pois o Raspberry Pi não possui armazenamento interno; interface HDMI; entrada USB e RCA; processador ARM 7 com capacidade de processamento de 32 bits, não sendo possível a instalação do Windows, mas aceitando qualquer distribuição Linux, como o Raspbian que é baseado no sistema operacional Debian. O que difere nas duas versões é a memoria RAM. Um modelo é oferecido com 256 MB e o outro com 512 MB. Assim como o Arduino, a desvantagem do Raspberry Pi é a não padronização da montagem dos robôs. Entretanto, é possível a utilização de sensores e programação embarcada.

3.1.5 Kit educacional Lego Mindstorms

A LEGO lançou em janeiro de 2006, na feira *Consumer Electronics Show*[9] em Las Vegas, a linha Mindstorms NXT, que é uma versão mais avançada que a já consagrada RCX, possuindo um processador Atmel ARM 32 bits com *clock* de 48MHz, HD de 256KB de memória *flash*, memória RAM de 64KB, software próprio, sensores de luz, toque e som. Esse suporte confere ao robô noções de distância, sendo inclusive capaz de reagir a movimentos, ruídos e cores. Essa nova versão também executa movimentos com maior grau de precisão que seu antecessor. O kit Lego Mindstorms é bem flexível quanto à montagem do robô, porém a Lego disponibiliza um manual de montagem de um robô padrão; uma vantagem ao se comparar com os concorrentes mencionados anteriormente. A escolha do kit Lego Mindstorms para o dado projeto se deu, além da vantagem já conhecida da padronização do robô, por ele já conter os sensores necessários no kit e por esse material ser utilizado na matéria Princípios de Robótica Educacional, ministrada na FGA pelo prof. Dr. Maurício Serrano, co-orientador deste projeto. Como o kit tem uma liguagem de programação própria, e a liguagem de programação escolhida por este trabalho foi o Prolog, se fez necessário a busca de uma ferramenta que tornasse possível essa integração, sendo esta ferramenta brevemente descrita a seguir.

3.1.6 Traveller

O *framework* Traveller (RINCON, 2014), contém algoritmos para definir qual caminho o robô deve seguir de modo a chegar a um destino de forma rápida e eficiente.

[8] www.raspberrypi.org/about/
[9] www.cesweb.org

Utilizando o robô Lego Mindstorms NXT, o Traveller definirá qual será a trajetória para o robô alcançar determinado ponto no mapa desviando dos obstáculos. A trajetória definida pelo Traveller dará suporte à máquina de raciocínio para o cálculo do tempo necessário para executar cada missão do tapete de missões.

3.1.7 PLNXT

O PLNXT é uma plataforma baseada na *Prolog API for Mindstorms NXT*, desenvolvida dentro do projeto *HeKatE*[10], e visa proporcionar uma solução para a programação de alto nível no robô LEGO Mindstorms NXT. Proporciona a comunicação via *bluetooth* ou USB com o robô e possibilita a programação das funcionalidades do robô, como virar, andar para frente, andar para trás, leitura dos sensores, dentre outros, na linguagem de programação Prolog. Segundo o site da plataforma[11], o PLNXT é de fácil programação, tem uma média dificuldade de instalação, possui a capacidade de controlar o NXT sem ter que carregar um programa na memória do robô e possui uma boa legibilidade.

3.2 Engenharia de software

De modo a gerenciar o processo de desenvolvimento deste trabalho, bem como desenvolver, manter e disponibilizar o código e artefatos gerados, o uso de algumas ferramentas se faz necessário. Tais ferramentas são brevemente descritas a seguir, bem como a distribuição do sistema operacional utilizada.

3.2.1 Ubuntu

O Linux foi criado por Linus Torvalds em 1991 (TORVALDS; BY-DIAMOND, 2001) e desde então amplamente aderido pela comunidade. Com base nesse kernel vários outros sistemas operacionais foram criados, dentre eles o Ubuntu. Baseado no Debian e patrocinado pela Canonical Ltda., o Ubuntu é licenciado por General License Public (GPL), podendo ser então modificado e distribuído de acordo com os termos da licença. Robustez e confiabilidade, por meio de código aberto, compatibilidade, segurança e rapidez são algumas características que o fazem tão popular.

3.2.2 SWIProlog

O SWIProlog é uma implementação, de código aberto, para a linguagem de programação Prolog, executando em modo texto, através de comandos no terminal do sistema. Sob a licença *Lesser GNU Public License*[12], pode ser utilizado nas plataformas Windows,

[10] hekate.ia.agh.du.pl
[11] http://ai.ia.agh.edu.pl/wiki/mindstorms:nxt_prolog_api
[12] http://www.gnu.org/licenses/lgpl.html

Linux e MacOS. Possui diversas ferramentas de edição gráfica, tais como: J-Prolog Editor e SWI-Prolog-Editor. Permite ainda a utilização da linguagem Prolog por outras linguagens, tais como: C/C++ e Java.

3.2.3 Bizagi Process Modeler

O BizAgi Process Modeler[13] é um aplicativo gratuito utilizado para criar e documentar modelos de processos em BPMN[14] (*Business Process Model and Notation*). Esse aplicativo faz parte de uma suíte de software, chamada Bizagi, composta por dois produtos: o Bizagi Process Modeler e o BizAgi BPM Suite.

3.2.4 Git

O Git[15] é um dos mais consagrados e utilizados sistemas de controle de versão. É *open-source* e gratuito, distribuído sob a licença GNU GPLv2[16], e projetado para lidar com qualquer tamanho de projeto mantendo a rapidez e a eficiência.

3.2.5 Github

O GitHub é um repositório web planejado para utilizadores do sistema de controle de versão Git. Possui opções de utilização de repositórios privados e públicos, sendo o pago e gratuito, respectivamente. No GitHub, é possível, dentre tantas outras funcionalidades, visualizar o código no navegador, criar *issues*, revisar e aceitar mudanças, baixar o repositório como *.zip*, criar *branchs* ou até mesmo fazer o *fork* de outro repositório.

3.2.6 Waffle

O Waffle[17] é uma solução de gerenciamento de projeto *online*, que se integra ao repositório do GitHub e cria um *board*, muito semelhante a um *kanban*, com as *issues* que estão cadastradas, e as agrupa de acordo com seus status, que pode ser Backlog, Ready, In Progress e Done. Cada *issue*, se estiver associada a um Milestone no GitHub, será também associada a um Milestone no Waffle. Qualquer alteração feita no GitHub é automaticamente compatibilizada no Waffle.

[13] http://www.bizagi.com/en/bpm-suite/bpm-products/modeler
[14] www.bpmn.org
[15] http://www.git-scm.com/
[16] http://www.gnu.org/licenses/gpl-2.0.html
[17] www.waffle.io

3.2.7 LaTeX

O Latex[18] é um sistema de preparação de documentos para composição tipográfica de alta qualidade, desenvolvido inicialmente por Leslie Lamport, em 1985, baseado na linguagem TeX criada por Donald E. Knuth, no final da década de 70, na Universidade de Stanford. Atualmente, o Latex é mantido e desenvolvido pelo *The Latex3 Project*[19], sendo disponibilizado gratuitamente. O Latex é mais amplamente utilizado no meio técnico-científico para escrita de documentos de médio a grande porte, devido a sua facilidade de produzir fórmulas e símbolos matemáticos.

3.2.8 TexMaker

O TeXMaker[20] é um editor de texto Latex gratuito, multiplataforma para Linux e MacOSX, com suporte a unicode, verificação ortográfica, auto-completar e contém um visualizador de pdf embutido. Com uma interface simples, o TexMaker contém botões de atalho para as funções mais utilizadas como: estruturas (part/chapter/section), referências (ref/cite), tamanho e estilo de letra, negrito, itálico, alinhamento (direta/esquerda, centro) e inserção de diversos símbolos matemáticos.

3.3 Considerações parciais

As ferramentas descritas nesse capítulo foram investigadas com o intuito de verificar as diversas formas de se trabalhar com a robótica na área educacional e auxiliar na compreensão do domínio do problema. O estudo dessas ferramentas serviu de insumo para a elaboração da proposta do algoritmo de programação dinâmica, produto de trabalho desse TCC. Já as ferramentas relacionadas à engenharia de software citadas nesse capítulo visam auxiliar no desenvolvimento do trabalho, tanto na parte escrita, quanto na de planejamento e implementação.

[18] http://www.latex-project.org/
[19] http://latex-project.org/latex3.html
[20] http://www.xm1math.net/texmaker/

4 Metodologia

Com o intuito de guiar a pesquisa de forma adequada, esse capítulo aborda sobre os diversos tipos de metodologias de pesquisa, de modo a definir qual se adequa melhor ao projeto. Este capítulo traz também a modelagem da metodologia de pesquisa que será utilizada, além do cronograma da pesquisa e a prova de conceito.

4.1 Classificação da pesquisa

As pesquisas podem ser classificadas quanto ao seus objetivos gerais ou quanto aos procedimentos técnicos utilizados. A classificação das pesquisas quanto aos seus objetivos gerais é muito importante para estabelecer uma visão teórica, ou seja, possibilitar uma aproximação conceitual. Porém, é necessário confrontar essa visão teórica com dados realistas, fornecendo uma visão empírica. Desta forma, faz-se necessário traçar um modelo conceitual e operativo, chamado delineamento, e que tem como parte mais importante a definição do procedimento que será utilizado na coleta de dados. Logo, as pesquisas podem ser classificadas de acordo com seu delineamento, ou seja, quanto aos procedimentos técnicos utilizados (GIL, 2002).

4.1.1 Quanto aos objetivos gerais

As pesquisas classificadas pelos objetivos gerais podem ser divididas em 3 (três) grandes grupos: pesquisas exploratórias, pesquisas descritivas e pesquisas explicativas (GIL, 2002). Cada um desses grupos será detalhado nas subseções a seguir apresentadas.

4.1.1.1 Pesquisa exploratória

A pesquisa exploratória tem um caráter investigativo sobre um assunto pouco conhecido. Tem por objetivo facilitar a delimitação do tema de pesquisa através de informações proporcionadas pela exploração (PRODANOV; FREITAS, 2013). Por isso, seu planejamento é bastante flexível, de modo que considere os mais diversos aspectos acerca do tema estudado. Em grande maioria, as pesquisas exploratórias assumem a forma de pesquisa bibliográfica ou estudo de caso, envolvendo levantamento bibliográfico, entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado e/ou análise de exemplos que estimulem a compreensão (GIL, 2002).

4.1.1.2 Pesquisa descritiva

Na pesquisa descritiva, o pesquisador apenas observa, registra, analisa e ordena os dados dos fatos observados sem interferir de forma alguma no processo. Envolve o uso padronizado de coleta de dados podendo ser questionário e/ou observação sistemática. Em geral, assume a forma de levantamento (GIL, 2002). A pesquisa descritiva tem por objetivo descrever as características de determinada população, fenômeno ou estabelecimento de relações entre variáveis procurando classificar, explicar e interpretar os fatos que ocorrem, diferindo da pesquisa experimental que pretende demonstrar o modo ou as causas de um dado fato ocorrido (PRODANOV; FREITAS, 2013).

4.1.1.3 Pesquisa explicativa

A pesquisa explicativa, assim como o nome sugere, busca explicar os motivos pelos quais ocorrem os fatos observados, por meio do registro, análise, classificação e interpretação dos fenômenos observados. Em geral, quando realizada nas ciências naturais, requer o uso do método experimental que possibilita a manipulação e controle de variáveis, e quando realizada nas ciências sociais, requer o uso do método observacional. Dessa forma, a pesquisa explicativa assume a forma de pesquisa experimental ou pesquisa *ex-post facto*. (PRODANOV; FREITAS, 2013)

4.1.2 Quanto aos procedimentos

As pesquisas classificadas quanto aos procedimentos técnicos utilizados podem ser divididas em 2 (dois) grupos: pesquisas que se valem das fontes de "papel", sendo elas classificadas em bibliográfica ou documental; e as pesquisas cujos dados são fornecidos por pessoas, sendo classificadas em experimental, *ex-post facto*, levantamento ou estudo de caso (GIL, 2002). Cada uma dessas classificações será detalhada a seguir.

4.1.2.1 Pesquisa bibliográfica

Uma pesquisa bibliográfica é desenvolvida exclusivamente a partir de fontes bibliográficas, principalmente de livros e artigos científicos. As fontes bibliográfcas podem ser classificadas de acordo com a Figura 3.

Os livros são fontes bibliográficas de excelência, classificados como de leitura corrente ou de referência. Os livros de literatura corrente são aqueles que objetivam conferir conhecimentos científicos ou técnicos ao leitor, já os de referência tem por objetivo proporcionar ao leitor a rápida obtenção das informações requeridas (informativa) ou a referência às obras que as contenham (remissiva) (GIL, 2002). A pesquisa bibliográfica é muito vantajosa ao pesquisador, já que este pode ter acesso a uma gama de informações muito maior do que aquelas que ele conseguiria pesquisando diretamente. Essa pesquisa também é de

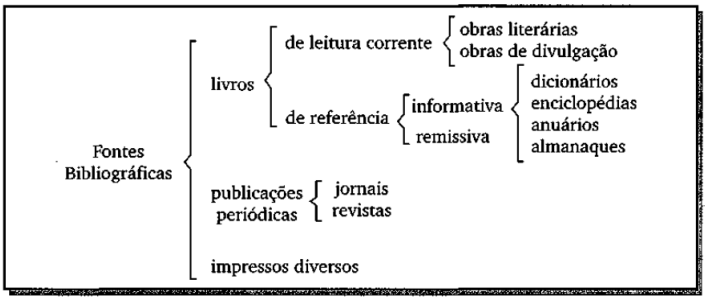

Figura 3 – Classificação das pesquisas bibliográficas (GIL, 2002)

suma importância nas pesquisas históricas, pois a maioria dos fatos só é possível conhecer por esse tipo de pesquisa. A desvantagem dessa pesquisa é a possível propagação de erros, pois se a fonte bibliográfica utilizada estiver com algum dado equivocado, este será refletido no trabalho do pesquisador.

4.1.2.2 Pesquisa documental

A pesquisa documental é difícil de distinguir da pesquisa bibliográfica, pois ambas utilizam material impresso para um determinado público como fonte de pesquisa. A principal diferença que se pode destacar é quanto ao tratamento análitico das fontes. A pesquisa documental utiliza fontes de pesquisa que não passaram por um tratamento análitico, ou seja, são matéria-prima como cartas pessoais, fotos, diários, memorandos e outros, a partir da qual o pesquisador vai desenvolver a sua pesquisa. A pesquisa bibliográfica se utiliza de fontes de pesquisa já analisadas como livros e periódicos (GIL, 2002).

4.1.2.3 Pesquisa experimental

A pesquisa experimental consiste em determinar o objeto de estudo, as possíveis variáveis capazes de influenciá-lo, as formas de controle e observação dos efeitos, fazendo com que o pesquisador seja o agente ativo da pesquisa. Uma pesquisa experimental pode ser desevolvida em qualquer lugar, desde que ela possa ser manipulada e controlada pelo pesquisador e que haja uma distribuição aleatória dos elementos que irão participar do experimento. As pesquisas experimentais são as mais utilizadas para se provar hipóteses de causa-efeito, porém é um ambiente com variáveis controladas, logo, pode haver variáveis de controle difícil ou até mesmo impossível, tornando o experimento inviável. (GIL, 2002)

4.1.2.4 Pesquisa *ex-post facto*

A tradução literal de *ex-post facto* é "de um fato passado", ou seja, a pesquisa *ex-post facto* é realizada após a ocorrência de variáveis no objeto de estudo. A diferença entre a pesquisa *ex-post facto* e a pesquisa experimental é que esta verifica a causa-efeito com o

controle de variáveis, enquanto aquela verifica a causa-efeito sem o controle de variáveis, já que o fato já ocorreu. Esse tipo de pesquisa é muitas vezes chamada de correlacional, pois nela é possível se identificar a existência de relação entre variáveis. (GIL, 2002)

4.1.2.5 Levantamento

O levantamento é utilizado quando se interroga diretamente as pessoas que se quer conhecer o comportamento, procura ser representativo de universo definido, identificando as características dos componentes desse universo e oferecendo resultados pela precisão estática, através da caracterização dos segmentos. Um questionário pode ser utilizado para coletar os dados, e depois é feita a análise quantitativa dessa coleta. Quando um levantamento é feito sobre um grupo muito grande de pessoas, este é chamado de "censo". (PRODANOV; FREITAS, 2013)

4.1.2.6 Estudo de caso

O estudo de caso busca o aprofundamento nas questões propostas, estudando um único grupo ou comunidade, utilizando mais a observação direta do que a interrogação, captando as explicações e interpretações do que ocorre no grupo. No estudo de campo, o pesquisador está inserido no grupo, para poder entender melhor as regras, os costumes e as convenções.

4.2 Planejamento da pesquisa

O modelo híbrido de pesquisa exploratório/experimental será utilizado nesse TCC para avaliar os vários tipos de algoritmos que podem ser utilizados como solução para a questão de pesquisa. Após a escolha do algoritmo ter sido feita, este será adaptado ao contexto do LEGO Mindstorms, utilizando o modelo híbrido de pesquisa exploratório/-descritivo. Pretende-se ainda, nas atividades específicas de desenvolvimento, fazer uso de uma metodologia orientada aos princípios ágeis, uma possível adaptação do Scrum, com sprints de uma semana (em média), as quais ocorrerão em ambiente controlado e terão seus resultados quantificados. Dado que a pesquisa será exploratória/experimental e exploratória/descritiva, muito provavelmente será utilizado o modelo de pesquisa-ação ao longo do projeto. Esse modelo permitirá, dentre outras contribuições, a retroalimentação da solução com base nos experimentos realizados ao longo da condução do trabalho.

4.3 Modelagem da metodologia

A modelagem da metodologia utilizada neste trabalho demonstra, de forma clara, o fluxo das atividades tanto do trabalho de conclusão de curso 1 quanto do trabalho de

conclusão de curso 2, visto que um é a continuação do outro.

Inicialmente foi realizado um levantamento bibliográfico preliminar sobre robótica, algoritmos de otimização, como o algoritmo guloso, e programação dinâmica, a fim de dar suporte à definição do escopo do projeto. A seguir, foi realizado um planejamento da abordagem para definir se a mesma seria ou não viável. Após a aprovação do escopo, foi definido um referencial teórico, um suporte tecnólogico e uma metodologia de pesquisa. Concluídos esses passos, foi elaborada a prova de conceito onde buscou-se avaliar a viabilidade da proposta, abordando a integração do robô com a linguagem Prolog, e uma avaliação dos algoritmos de otimização (algoritmo guloso e programação dinâmica), vide seção 4.5. Em seguida, foi elaborada a parte escrita da proposta.

A apresentação para a banca avaliadora findará a fase do TCC 01, dando início ao TCC 02. Este começa com a coleta e realização das sugestões de melhoria feitas pela banca avaliadora, a seguir serão definidos cenários de uso que para serem utilizados na avaliação dos resultados da implementação da máquina de raciocínio. A *First Lego League* utiliza em cada torneio um tapete de missões diferente. Para a criação de cada cenário de uso, será utilizado um tapete diferente, e cada tapete contêm missões diferentes. O primeiro cenário de uso será utilizando o tapete de missões *Nature's Fury*, vide seção 5.3, e o ideal será a máquina de raciocínio contemplar no mínimo 3 cenários de uso. Em *sprints* de uma semana, durante três meses, a máquina de raciocínio será desenvolvida. A cada duas *sprints*, uma *release*, a máquina de raciocínio será testada no robô e seus resultados serão avaliados de acordo com os cenários de uso anteriormente definidos. Caso o resultado não seja satisfatório, uma nova *release* será necessária. Quando a avaliação dos resultados for satisfatória (ou seja, os objetivos geral e específicos do TCC forem atingidos), será elaborada a parte escrita do projeto.

A definição temporal de uma *release* ter duas *sprints* e cada *sprint* ter duas semanas é uma média, podendo haver *sprints* ou *releases* maiores ou menores, de acordo com a necessidade. Uma *release* é dada por encerrada quando o código é testado no robô e seus resultados são avaliados.

Com intuito de ilustrar como será realizada a condução deste projeto, segue a Figura 4 que apresenta a modelagem do processo metodológico com base na metodologia descrita nesta seção.

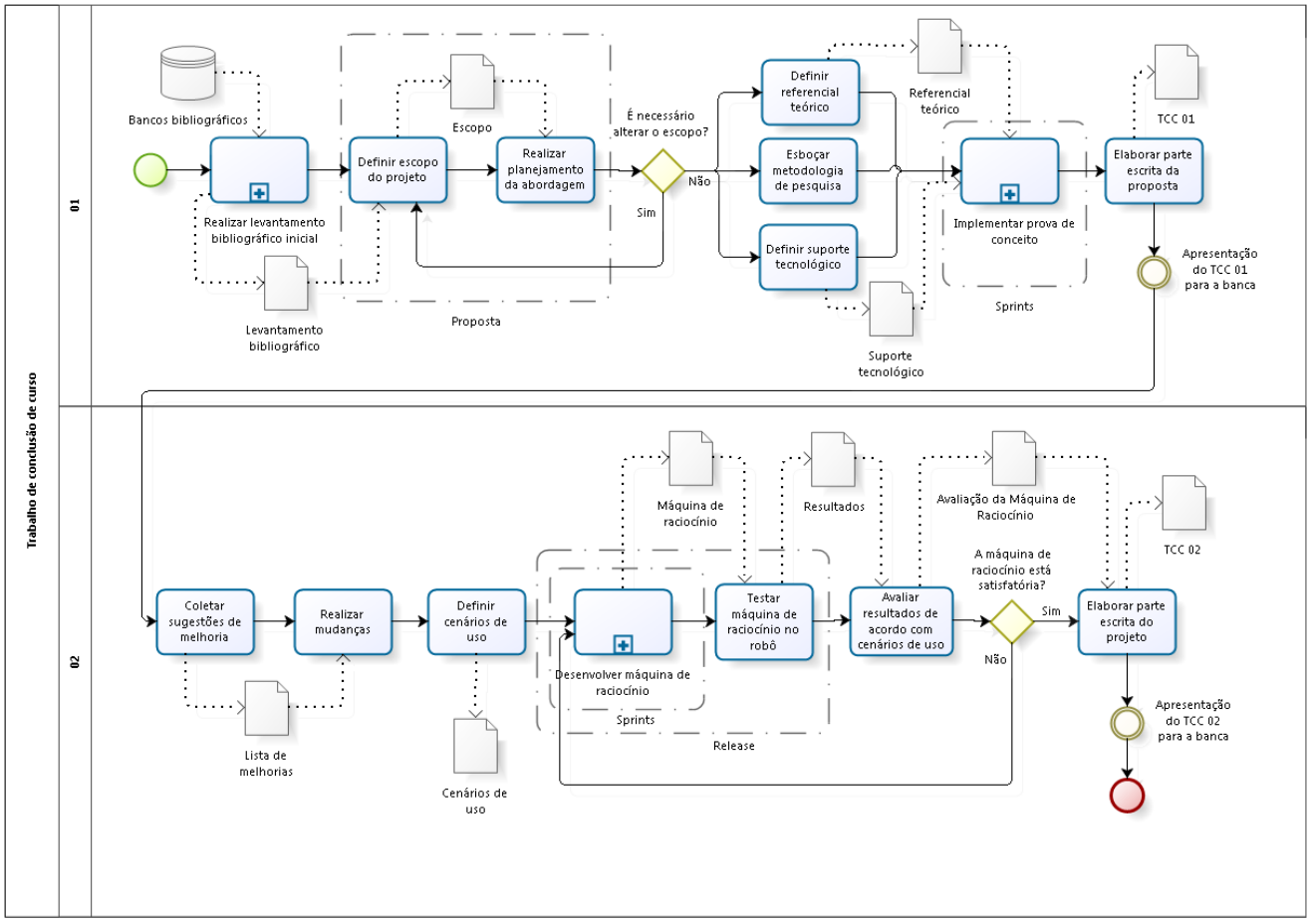

Figura 4 – Modelagem do processo utilizado no TCC

4.4 Cronograma da pesquisa

O cronograma, detalhado nas Tabelas 1 e 2, visa conferir uma noção temporal acerca das atividades definidas para o processo. Trata-se de uma visão preliminar, logo, pode haver modificações ao longo do projeto de acordo com as necessidades.

	Jan	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**
Realizar levantamento bibliográfico inicial	X	X				
Definir escopo do projeto		X				
Realizar planejamento da abordagem			X			
Definir referencial teórico				X	X	
Definir suporte tecnológico				X	X	
Esboçar metodologia de pesquisa				X	X	
Elaborar prova de conceito						X
Elaborar parte escrita da proposta						X

Tabela 1 – Cronograma do TCC 01

	Jul	**Ago**	**Set**	**Out**	**Nov**	**Dez**
Coletar sugestões de melhoria	X					
Realizar mudanças	X					
Definir cenários de uso	X					
Desenvolver máquina de racionício		X	X	X		
Testar máquina de raciocínio no robô			X	X		
Avaliar resultados de acordo com cenários de uso			X	X		
Elaborar parte escrita do projeto					X	

Tabela 2 – Cronograma do TCC 02

4.5 Prova de conceito

Com o intuito de provar a viabilidade desta proposta, foi elaborada uma prova de conceito, baseada no famoso problema de otimização chamado Problema da Mochila. Esta prova utiliza também uma ferramenta de integração do kit NXT com a linguagem de programação Prolog.

4.5.1 Ferramentas pesquisadas

Após definido a utilização do kit Lego Mindstorms por este trabalho, foi necessário a busca de ferramentas que tornasse possível a implementação do código na linguagem Prolog. Uma extensa busca foi realizada, e resultou em três ferramentas que são descritas a seguir.

4.5.1.1 LegoLog

O LegoLog[1] baseia-se em dois componentes principais: um robô construído usando RIS (*Robotics Invention System*) e um controlador robótico Golog. A idéia básica por trás do RIS é que programas sejam construídos no computador e transferidos para o robô através de uma torre de infravermelho, que está ligada à porta serial do computador. A LEGO fornece um software base, conhecido como *firmware*, que implementa uma máquina virtual que pode ser baixada para o robô. Esta máquina virtual permite que sejam escolhidos até cinco programas, cada um com até 32 variáveis, 8 tarefas e 9 sub-rotinas. Uma vez que o firmware está funcionando, é possível utilizar o próprio ambiente de programação no Windows, disponibilizado pela LEGO, para escrever programas para essa máquina virtual (LEVESQUE; PAGNUCCO, 2000).

Porém, o LegoLog funciona apenas no antigo robô da LEGO, o RCX, que era quem utilizava essa torre de infravermelho para transmitir dados do computador para o robô. Como o objeto de estudo dessa proposta é o robô LEGO Mindstorms NXT, o LegoLog foi descartado como opção de ferramenta a ser utilizada por esse projeto.

4.5.1.2 MLP

A MLP[2] é uma plataforma multiliguagens, criada em 2008, na Universidade de Madeira, por um projeto chamado DROIDE. Tem por principal objetivo criar uma biblioteca comum para a programação do kit LEGO Mindstorms NXT em diversas linguagens. Atualmente tem suporte para as seguintes linguagens de programação: C++, C#, Visual Basic, Java e Prolog, e funciona apenas no sistema operacional Windows.

Essa ferramenta consiste em um módulo base de comunicação, utilizado para todas as linguagens, e módulos separados, para cada linguagem individualmente. No módulo Prolog, é utilizada para programação a IDE SWIProlog, sendo necessário apenas, segundo a documentação[3] da plataforma, importar a biblioteca NXT.Prolog utilizando o comando use_foreign_library/1 disponível no próprio SWIProlog.

Na documentação da MLP não contêm instruções de como utilizar os predicados na plataforma, tampouco informações de como conectar a plataforma ao robô. Vários e-

[1] http://www.cs.toronto.edu/ morri/Legolog/
[2] http://www.cee.uma.pt/droide2/plataforma/index.html
[3] http://www.cee.uma.pt/droide2/plataforma/documentation/userguide.pdf

mails foram mandados para os criadores da plataforma, mas sem resposta alguma. Assim sendo, a MLP também foi descartada como opção de ferramenta a ser utilizada por esse projeto.

4.5.1.3 PLNXT

Como descrito no capítulo de suporte tecnológico, o PLNXT é uma plataforma baseada na *Prolog API for Mindstorms NXT*, desenvolvida dentro do projeto *HeKatE*[4], e visa proporcionar uma solução para a programação de alto nível no robô LEGO Mindstorms NXT. Alguns dos objetivos da plataforma são:

- Suportar todos os componentes do NXT padrão, como sensores e motores, e
- Ser uma solução multiplataforma, Windows e GNU/Linux.

A plataforma é composta por três camadas principais:

- Camada de comunicação (nxt_actions): proporciona a comunicação de baixo nível com o robô;
- Camada sensomotoric (nxt_sensomoto): permite a troca de informações com sensores e motores;
- Camada comportamental (nxt_movement): fornece funções para controle do robô.

A comunicação com o robô pode ser feita via porta USB ou via *bluetooth*. No site[5] do PLNXT, é possível encontrar um tutorial passo-a-passo de como configurar o ambiente para utilizar a plataforma, além de códigos para testar a conexão, os motores e os sensores.

A configuração do ambiente para utilizar essa plataforma neste trabalho foi feita. Porém, ao testar a conexão via *bluetooth* com o robô, utilizando o código disponibilizado no site, ocorre um problema. No caso, o robô identifica que está conectado ao computador, mas a plataforma não identifica o robô. Apesar deste contra tempo, essa platafoma foi a escolhida para ser utilizada nesse trabalho.

4.5.2 Descrição da prova de conceito

Nas bibliografias de inteligência artificial o problema da mochila é por diversas vezes citado como exemplo para otimização. Devido à semelhança desse problema com o problema tratado por esse trabalho, aquele servirá como base para a elaboração da teoria deste.

[4] hekate.ia.agh.du.pl
[5] http://ai.ia.agh.edu.pl/wiki/mindstorms:plnxt:start

4.5.2.1 Problema da mochila

O problema da mochila é um dos mais famosos problemas de otimização, onde existem vários objetos, que contêm um peso e um valor associado, e uma mochila com uma capacidade máxima. Essa mochila deve ser preenchida de forma que contenha o maior valor possível. Existem duas variações desse problema: a mochila binária e a mochila fracionária.

No caso da mochila binária é possível pegar somente todo o objeto ou não pegá-lo. Já a mochila fracionária é possível pegar parte de um objeto, mantendo a proporcionalidade do valor. Pode-se resolver esses dois problemas de várias formas, mas foi analisado por esse trabalho a resolução pelo algoritmo guloso e por programação dinâmica. Os dados do problema da mochila utilizados nesta prova de conceito são apresentados na Tabela 3.

	Item A	**Item B**	**Item C**
valor	60	100	120
peso	10	20	30
valor/peso	6	5	4

Tabela 3 – Problema da mochila

Como citado anteriormente, esta prova de conceito demonstrará a resolução do problema da mochila, em suas duas formas, tanto pelo algoritmo guloso quanto pela programação dinâmica, de forma a analisar qual algoritmo atende melhor à solução do problema abordado por este trabalho.

- **Resolução por algoritmo guloso:** Na resolução utilizando o algoritmo guloso, a melhor ordenação de entrada seria pelo valor/peso de forma decrescente. Dessa forma, para o problema da mochila fracionária, tem-se uma solução com um valor máximo alcançado de 240, de acordo com a Figura 5.

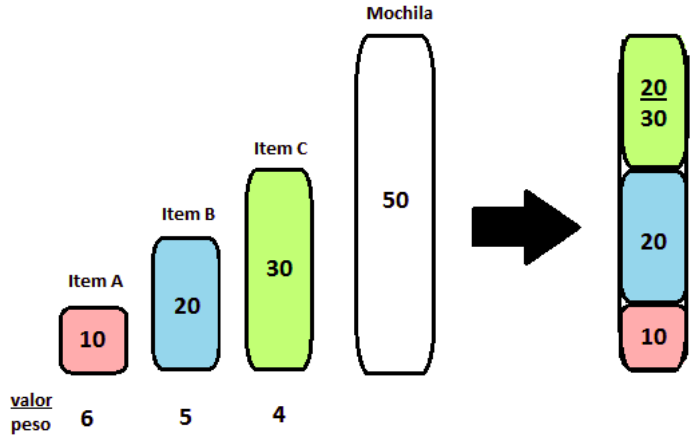

Figura 5 – Problema da mochila fracionária - Algoritmo Guloso

E para a mochila binária, o valor máximo alcançado seria de 160, como mostra a Figura 6.

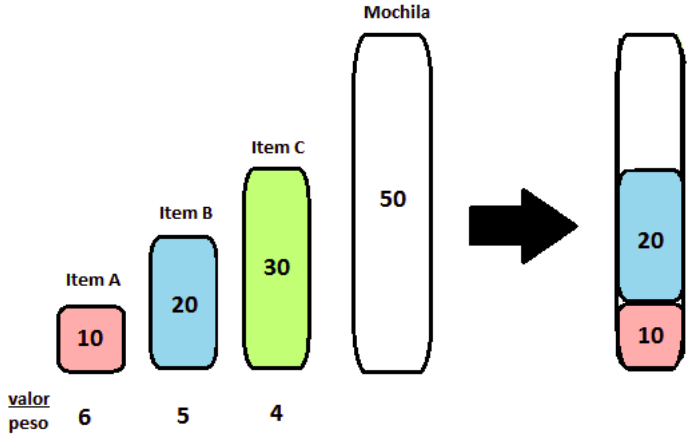

Figura 6 – Problema da mochila binária - Algoritmo Guloso

Como é possível perceber, na resolução da mochila binária, o algoritmo guloso não obtém sucesso, uma vez que a melhor solução teria o valor máximo igual 220, como demonstra a Figura 7.

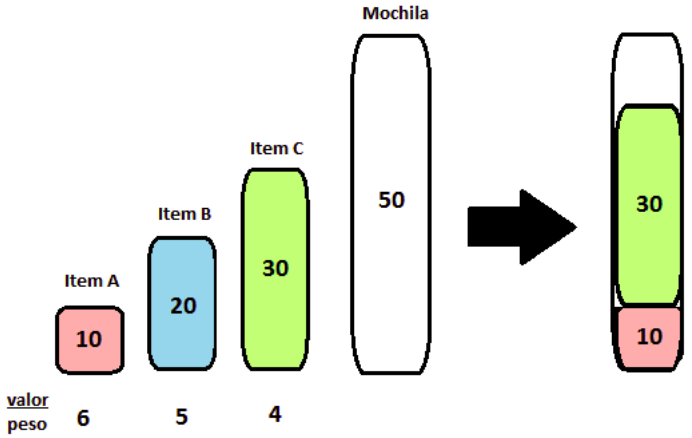

Figura 7 – Problema da mochila binária - Resposta

- **Resolução por programação dinâmica:** Na resolução da mochila binária por programação dinâmica, é criada uma matriz onde as linhas são os itens e cada coluna representa a capacidade da mochila ordenada de forma crescente. No caso do exemplo anteriormente citado, seria criada uma matriz igual a mostrada na Figura 8.

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0						
A(10)						
B(20)						
C(30)						

Figura 8 – Problema da mochila binária - Programação Dinâmica

A análise se inicia na primeira célula (linha 1, coluna 1), e a cada passo vai analisando a próxima célula da linha, quando chega ao final, volta à primeira coluna e desce para a segunda linha, caracterizando uma análise *top-down*.

A cada iteração, é analisado primeiramente o peso do item em relação a capacidade relativa da mochila (referente a coluna que se está analisando), podendo resultar em três cenários:

1. **Peso analisado é igual à capacidade relativa:** nesse cenário, o algoritmo pergunta a si próprio: O que é melhor? Tomar o valor da célula anterior, tomar o valor da célula acima ou tomar o valor do item analisado? Após a decisão, o algoritmo grava o valor na matriz.
2. **Peso analisado é inferior à capacidade relativa:** nesse cenário, o algoritmo pergunta a si próprio: É melhor tomar o valor da célula acima ou é melhor tomar o valor do item analisado somado à célula da linha acima que

complemente o peso analisado, de forma que a capacidade relativa seja alcançada? Após tomada a decisão, o algoritmo grava o valor na matriz.

3. **Peso analisado é superior à capacidade relativa:** nesse cenário, não é possível tomar o valor do item analisado. Logo, o algoritmo pergunta a si próprio: é melhor tomar o valor da célula ao lado ou da célula acima?

Dada uma matriz (NxM), a última célula da mesma (linha N, coluna M) contém a resposta para o problema analisado.

Tomando como exemplo a Tabela 3, a solução por programação dinâmica cria uma matriz como a da Figura 8 e preenche a primeira linha e a primeira coluna com zeros, como mostra a Figura 9, pois se a capacidade da mochila é zero (primeira coluna) não tem como pegar item. Como não tem item para analisar (primeira linha), não tem como tomar valor algum. Mesmo na prática não existindo uma mochila sem capacidade ou nenhum item para ser analisado, essas hipóteses são acrescentadas para que na hora da execução do algoritmo não ocorra erro, já que o algoritmo está sempre analisando o valor de outras células para a tomada de decisão.

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0					
B(20)	0					
C(30)	0					

Figura 9 – Problema da mochila binária - PD - Passo Inicial

O segundo passo é analisar a capacidade da mochila sendo 10, e havendo apenas o item A, com peso 10. Logo, tem-se o cenário do peso analisado igual à capacidade relativa, e obviamente, o melhor, nesse caso, é tomar o valor do item analisado, ou seja, escrever 60 na célula, como mostra a Figura 10.

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60				
B(20)	0					
C(30)	0					

Figura 10 – Problema da mochila binária - PD - Item A, Capacidade 10

Em seguida, é analisada a capacidade da mochila como sendo 20. Nesse casso, existe apenas o item A, de peso 10, para ser colocado na mochila. Imediatamente é percebido, por nós humanos, que o valor desse item será escrito na célula. Porém, a progração dinâmica não tem esse conhecimento, e avalia o cenário de peso analisado

inferior à capacidade relativa da mochila. Nesse cenário, é possível tomar o valor da célula acima, que é zero, ou o valor do item de peso 10 somado ao seu complemento, contido na linha acima, que o faz alcançar a capacidade da mochila. A coluna que complementa o peso do item é a de valor 10, pois assim é alcançado o valor 20 da capacidade relativa analisada. As Figuras 11 e 12 demonstram essa decisão.

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	?			
B(20)	0					
C(30)	0					

Figura 11 – Problema da mochila binária - PD - Item A, Capacidade 20 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	60			
B(20)	0					
C(30)	0					

Figura 12 – Problema da mochila binária - PD - Item A, Capacidade 20 - Resultado

Essa análise é feita para toda a linha, como mostram as Figuras 13, 14, 15, 16, 17 e 18, onde em todos os passos foi decidido, pela programação dinâmica, a soma do valor do item A, que é 60, e o seu complemento, que em todos os casos tem valor zero.

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	60	?		
B(20)	0					
C(30)	0					

Figura 13 – Problema da mochila binária - PD - Item A, Capacidade 30 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	60	60		
B(20)	0					
C(30)	0					

Figura 14 – Problema da mochila binária - PD - Item A, Capacidade 30 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	?	
B(20)	0					
C(30)	0					

Figura 15 – Problema da mochila binária - PD - Item A, Capacidade 40 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	
B(20)	0					
C(30)	0					

Figura 16 – Problema da mochila binária - PD - Item A, Capacidade 40 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	?
B(20)	0					
C(30)	0					

Figura 17 – Problema da mochila binária - PD - Item A, Capacidade 50 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0					
C(30)	0					

Figura 18 – Problema da mochila binária - PD - Item A, Capacidade 50 - Resultado

A análise do item B, de peso 20, quando a capacidade da mochila é 10, é o cenário de peso analisado superior à capacidade relativa. Nesse caso, é possível continuar com o valor contindo na mochila na capacidade relativa anterior (célula anterior), ou pode-se tomar o valor do item anterior quando a capacidade relativa da mochila é a mesma (célula acima). Dessa forma, a programação dinâmica decide pela célula acima e escreve seu valor na célula analisada. As Figuras 19 e 20 demonstram essa decisão.

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	?				
C(30)	0					

Figura 19 – Problema da mochila binária - PD - Item B, Capacidade 10 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60				
C(30)	0					

Figura 20 – Problema da mochila binária - PD - Item B, Capacidade 10 - Resultado

As demais decisões são efetuadas seguindo um dos passo-a-passo anteriormente citados, dependendo do cenário que se está analisando. As Figuras 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37 e 38 demonstram tais decisões.

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	?			
C(30)	0					

Figura 21 – Problema da mochila binária - PD - Item B, Capacidade 20 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100			
C(30)	0					

Figura 22 – Problema da mochila binária - PD - Item B, Capacidade 20 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	10	20	30	40	50
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	?		
C(30)	0					

Figura 23 – Problema da mochila binária - PD - Item B, Capacidade 30 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160		
C(30)	0					

Figura 24 – Problema da mochila binária - PD - Item B, Capacidade 30 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	?	
C(30)	0					

Figura 25 – Problema da mochila binária - PD - Item B, Capacidade 40 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	
C(30)	0					

Figura 26 – Problema da mochila binária - PD - Item B, Capacidade 40 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	?
C(30)	0					

Figura 27 – Problema da mochila binária - PD - Item B, Capacidade 50 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0					

Figura 28 – Problema da mochila binária - PD - Item B, Capacidade 50 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	?				

Figura 29 – Problema da mochila binária - PD - Item C, Capacidade 10 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60				

Figura 30 – Problema da mochila binária - PD - Item C, Capacidade 10 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60	?			

Figura 31 – Problema da mochila binária - PD - Item C, Capacidade 20 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60	100			

Figura 32 – Problema da mochila binária - PD - Item C, Capacidade 20 - Resultado

As Figuras 33 e 34 mostram a decisão da programação dinâmica quando um cenário de peso analisado é igual a capacidade relativa da mochila, onde a melhor opção foi não foi tomar o valor do item, que seria 120, mas tomar o valor do item anterior na mesma capacidade relativa (célula acima), que é 160.

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60	100	?		

Figura 33 – Problema da mochila binária - PD - Item C, Capacidade 30 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60	100	180		

Figura 34 – Problema da mochila binária - PD - Item C, Capacidade 30 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60	100	180	?	

Figura 35 – Problema da mochila binária - PD - Item C, Capacidade 40 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60	100	180	180	

Figura 36 – Problema da mochila binária - PD - Item C, Capacidade 40 - Resultado

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60	100	180	180	?

Figura 37 – Problema da mochila binária - PD - Item C, Capacidade 50 - Análise

Item(peso)	Capacidade relativa da mochila					
	0	**10**	**20**	**30**	**40**	**50**
0	0	0	0	0	0	0
A(10)	0	60	60	60	60	60
B(20)	0	60	100	160	160	160
C(30)	0	60	100	180	180	220

Figura 38 – Problema da mochila binária - PD - Item C, Capacidade 50 - Resultado

O valor da última célula é a resposta para o problema. Como é possível ver na Figura 38, o valor encontrado pela programação dinâmica foi 220, demonstrando que para o problema da mochila binária, o algoritmo é eficaz.

4.5.2.2 Adaptação do problema da mochila para o contexto do trabalho

Para o completo entendimento do trabalho proposto, algumas definições sobre termos amplamente utilizados no kit LEGO Mindstorms NXT se fazem necessárias e são feitas a seguir.

- **Missão:** uma missão é composta por uma tarefa que deve ser executada pelo robô em um determinado tempo, valendo uma pontuação, e por um trajeto que deve ser efetuado pelo robô para executar a missão;
- **Pontuação da missão:** cada missão tem uma pontuação previamente estabelecida. Essa pontuação é alcançada se a missão for concluída com sucesso;
- **Pontuação total:** é o somatório das pontuações de todas as missões concluídas com sucesso;
- **Mapa:** um mapa é onde o robô deve executar as missões. Cada mapa contém diferentes missões associados a ele;
- **Tempo total:** é uma quantidade de tempo, previamente estabelecida, geralmente de dois a três minutos, utilizada para executar o máximo de missões possíveis almejando a maior pontuação total;
- **Tempo restante:** é a quantidade de tempo que falta para chegar ao fim do tempo total;
- **Tempo da tarefa:** é a quantidade de tempo necessária para que o robô execute uma tarefa;
- **Tempo do trajeto:** é a quantidade de tempo que o robô utiliza para chegar em um local com coordenadas (X, Y) partindo de um local com coordenadas (X0, Y0);
- **Tempo da missão:** é a quantidade de tempo utilizada para concluir a missão com sucesso.
- **Base:** local no mapa de onde o robô deve partir para executar a primeira missão.

O objetivo do trabalho é alcançar a maior pontuação dentro de um determinado espaço de tempo. Supondo que existam três missões para o robô executar, missão da árvore, missão dos pets e missão do avião, descritas a seguir, quais missões devem ser executadas, dentro de um tempo restante de 50 segundos, para se alcançar a maior pontuação?

- **Missão árvore:** Essa missão consiste de uma tarefa em que o robô tem que derrubar o galho de uma árvore, sem deixar que ele encoste nos fios de alta tensão. Tempo

de execução da tarefa: 5 segundos; Tempo da trajetória: 5 segundos; Tempo da missão: Tempo da trajetória + tempo da tarefa = 10 segundos;

- **Missão pets:** Essa missão consiste na tarefa de resgatar os pets e levá-los à base. Tempo de execução da tarefa: 10 segundos; Tempo da trajetória: 10 segundos; Tempo da missão: 20 segundos;
- **Missão avião**: Essa missão consiste na tarefa de derrubar o avião na pista de pouso. Tempo de execução da tarefa: 10 segundos; Tempo da trajetória: 20 segundos; Tempo de execução da missão: 20 segundos;

Da mesma forma que no problema da mochila binária não é possível pegar uma fração de um objeto, no contexto desse trabalho não é possível executar uma fração de uma missão, tornando o problema binário para ganhar a pontuação da missão. Desta forma, é criada uma matriz onde o eixo X é o tempo restante e o eixo Y é o tempo de execução de cada missão, como mostra a Figura 39.

Missão(tempo)	Tempo restante					
	0	10	20	30	40	50
0						
Árvore(10)						
Pets(20)						
Avião(30)						

Figura 39 – Prolego - Programação Dinâmica

Resolvendo essa matriz com programação dinâmica é alcançado o valor de 220, como mostra a Figura 40, ou seja, as missões que devem ser executadas são: missão pets e missão avião.

Missão(tempo)	Tempo restante					
	0	10	20	30	40	50
0	0	0	0	0	0	0
Árvore(10)	0	60	60	60	60	60
Pets(20)	0	60	100	160	160	160
Avião(30)	0	60	100	180	180	220

Figura 40 – Prolego - Programação Dinâmica - Resultado

4.6 Considerações parciais

Este trabalho propõe uma implementação de um algoritmo de programação dinâmica que otimize a execução de missões pelo robô, afim de aproveitar melhor o tempo

restante para conseguir a maior pontuação possível. Trata-se de uma pesquisa híbrida exploratória/experimental para o levantamento da solução da questão de pesquisa, e exploratória-descritiva para construção do algoritmo. A fim de refinar a proposta inicial do trabalho, que antes era baseada apenas no referencial teórico estudado, foi elaborada uma prova de conceito, avaliando três ferramentas candidatas para a integração do robô com o Prolog. Adicionalmente, comparou-se o algoritmo guloso e a programação dinâmica com o propósito de demonstrar as razões pelas quais a programação dinâmica foi selecionada para ser utilizada nesse trabalho. Com a avaliação das ferramentas candidatas foi possível perceber alguns problemas em relação a cada uma delas. Alguns problemas se mostraram mais graves que outros, o que inviabilizou a utilização de duas ferramentas, elegendo assim a ferramenta PLNXT, a qual será utilizada nesse trabalho.

5 Proposta

A proposta desse trabalho é a elaboração de uma máquina de raciocínio utilizando programação dinâmica e que seja implementada na linguagem Prolog. As entradas da máquina de raciocínio serão: uma localização (X,Y), um tempo restante e uma base de conhecimento que contém as missões ainda não realizadas. A partir dessas entradas, a máquina de raciocínio avaliará quais missões devem ser executadas para que se alcance a quantidade máxima de pontos possível.

A base de conhecimento das missões que ainda não foram realizadas será desenvolvida da seguinte forma: cada fato da base de conhecimento será uma missão, e cada missão terá um tempo de execução e uma pontuação associada. Inicialmente, o tempo para realizar cada missão será fixo, pois o robô sempre partirá da base. A evolução do algoritmo se dará ao adicionar mais variáveis, como por exemplo o tempo de deslocamento do ponto atual, sendo este qualquer ponto no mapa, à outro ponto. Cada evolução será testada com o cenário de uso.

Para determinar o tempo de execução de cada missão, partindo da base, será feito um levantamento dos vídeos da matéria Princípios de Robótica Educacional, onde os alunos realizaram as missões e gravaram em vídeo de forma fiel, sem edição alguma. Serão selecionados apenas os vídeos em que o robô utilizado foi montado de acordo com o indicado pela LEGO. Para cada missão, será feita uma média dos tempos realizados pelos grupos e essa média será assumida como o tempo de execução da missão. No caso de não haver vídeos para alguma missão, a mesma será executada cinco vezes e então será feita a média dos tempos.

O algoritmo terá como saída um roteiro de missões, o qual deverá ser executado pelo robô. O comando para realizar cada missão será enviado via *bluetooth* para o robô, utilizando a ferramenta PLNXT.

5.1 Arquitetura do robô

(VIEIRA, 2005 apud RINCON, 2014) define uma arquitetura para robôs móveis baseada em cinco camadas: percepção, decisão, planejamento de caminho, geração de trajetória e sistema de controle. Essa arquitetura pode ser visualizada na Figura 41.

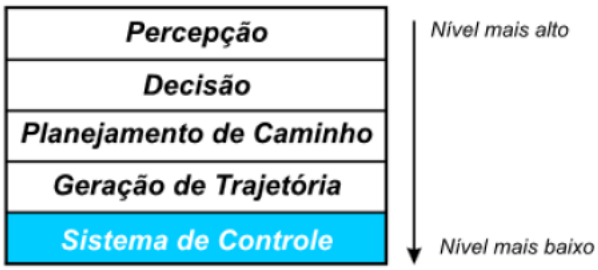

Figura 41 – Arquitetura em camadas de robôs móveis - (VIEIRA, 2005 apud RINCON, 2014)

A primeira camada, denominada Camada de Percepção, trata da percepção do robô acerca do mundo ao seu redor, utilizando os sensores que o compõe. Essa camada é responsável, por exemplo, por identificar os obstáculos contidos no tapete.

A segunda camada, denominada Camada de Decisão, é a responsável por decidir as ações que o robô irá executar. Esta camada é o cerébro do robô e será nela que o trabalho proposto irá agir. A máquina de raciocínio que atuará nesta camada terá o cálculo da trajetória, definida na quarta camada, através do *framework* Traveller. Conhecendo a trajetória será possível para a maquína de raciocínio calcular o tempo necessário de execução de cada missão e decidir quais missões deverão ser executadas.

A terceira camada define a trajetória que o robô irá executar para chegar à posição desejada, é nessa camada que o *framework* Traveller age. Com o conhecimento do ambiente anteriormente levantado pela Camada de Percepção o Traveller define um percurso por onde o robô deverá seguir e no qual não colidirá com nenhum obstáculo.

A quarta camada recebe o plano da trajetória feito pela camada anterior e define quais ações devem ser feitas sobre o *hardware* para executar o plano. Essa camada tem conhecimento das dimensões e limitações do robô. A última camada atua diretamente no *hardware* para garantir que os atuadores estão recebendo o sinal e agindo conforme planejado.

5.2 Montagem do robô

O robô que será utilizado por este trabalho é o Tribot, um tipo de montagem do robô NXT. Nessa montagem, o robô utiliza três motores, um para cada roda e um para a garra que se encontra na parte frontal do robô. O motor da esquerda deve ser ligado na porta C, o da direita na porta B e o da garra na porta A. Essa montagem utiliza também os sensores de toque, de som, de cor e de distância. O tutorial completo, contendo o passo-

a-passo para a montagem do robô pode ser encontrado no site da LEGO[1]. Na Figura 42 é possível visualizar a aparência do robô quando montado.

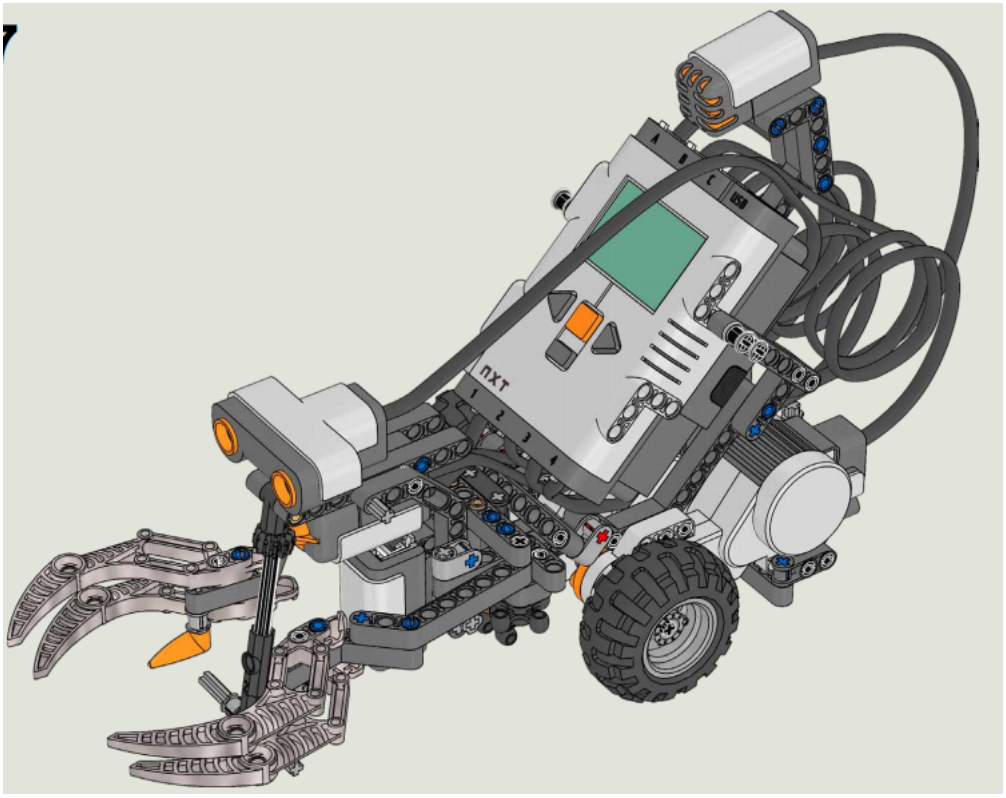

Figura 42 – Montagem do robô Tribot - Lego Mindstorms NXT

5.3 Tapete de missões

O tapete de missões escolhido para ser utilizado por esse trabalho é o *Nature's Fury*, utilizado como desafio do torneio *First Lego League* de 2013, que tem como temática uma tsunami que está devastando as cidades. Nesse tapete, as crianças são ensinadas quanto aos lugares seguros e não seguros para permanecer quando uma tsunami está em atividade. Os lugares seguros durante uma tsunami, demarcados no tapete, são bonificados com uma pontuação extra, caso o robô fique nesse lugar. Já os lugares inseguros também são demarcados no tapete. Porém, esse lugares representam penalidades, com perda de pontuação, caso o robô passe pelos mesmos. Existem missões de salvar os animais de estimação, salvar pessoas, levar água e mantimentos para os lugares seguros. O tapete *Nature's Fury* foi escolhido para ser objeto de trabalho deste projeto por ser utilizado nas aulas de Princípios de Robótica Educacional, sendo assim, mais acessível do que os outros tapetes. Na Figura 43 é possível visualizar o tapete, bem como os obstáculos que possui.

[1] https://education.lego.com/fr-fr/lesi/support/product-support/mindstorms-education-nxt/nxt-base-set-9797/building-instructions

Figura 43 – Tapete de missões *Nature's Fury* - Lego Mindstorms NXT

5.4 Descrição das missões

As missões contidas no tapete *Nature's Fury* são descritas a seguir.

- **Missão caminhão de fornecimento:** consiste em levar o caminhão até a área amarela do mapa. Pontuação: 20 pontos;
- **Missão sinal de evacuação:** consiste em levantar a placa com o sinal de evacuação. Pontuação: 30 pontos;
- **Missão avião de carga:** consiste em fazer o avião chegar à área azul ou amarela do tapete. Pontuação: 20 pontos para a área amarela, 30 pontos para a área azul;
- **Missão galho da árvore:** consiste em retirar o galho da árvore sem deixá-lo encostar nos fios de alta tensão. Pontuação: 30 pontos;
- **Missão tsunami:** consiste em fazer com que as três ondas que estão suspensas toquem o tapete. Pontuação: 20 pontos;
- **Missão ambulância:** consiste em levar a ambulância até a área amarela do tapete. Pontuação: 25 pontos;
- **Missão pista limpa:** consiste em retirar qualquer objeto da pista de pouso. Pontuação: 30 pontos;

- **Missão realocação de construção:** consiste em retirar todas as peças cinzas encontradas na construção da área verde do tapete. Pontuação: 20 pontos;
- **Missão teste da base de isolamento:** consiste em encostar na base de isolamento e derrubar apenas um dos prédios. Pontuação: 30 pontos;
- **Missão construção:** consiste em empilhar os segmentos na área rosa do tapete. Pontuação: 5 pontos para cada segmento empilhado;
- **Missão obstáculo:** consiste no robô passar por cima de obstáculos chegando às áreas coloridas do tapete. Nesse caso, para cada cor do tapete, existe uma pontuação. Pontuação: 10 pontos para a área azul, 16 pontos para a área verde, 23 pontos para a área roxa e 31 pontos para a área vermelha.
- **Missão elevar a casa:** consiste no robô abaixar a alavanca que eleva a casa. Pontuação: 25 pontos;
- **Missão progresso:** consiste em rodar um círculo de cores. Pontuação: 2 pontos para cada cor que o ponteiro passar.
- **Missão família:** consiste em juntar as pessoas que estão no tapete em uma única área colorida. Pontuação: 33 para duas pessoas juntas, ou 66 para 3 pessoas juntas;
- **Missão água:** consiste em juntar ao menos uma pessoa com uma garrafa de água na mesma região. Pontuação: 15 pontos para cada pessoa+água;
- **Missão segurança:** consiste em levar pelo menos uma pessoa à região vermelha ou amarela do tapete. Pontuação: 12 pontos para cada pessoa na área amarela ou 18 pontos para cada pessoa na área vermelha;
- **Missão *pets*:** consiste em juntar ao menos um animal à uma pessoa numa área colorida. Pontuação: 15 pontos para cada pet+pessoa;
- **Missão equipamentos e suprimentos:** consiste em levar pelo menos um item, exceto a água, para a região vermelha ou amarela do tapete. Pontuação: 3 pontos para cada item na área amarela ou 4 pontos para cada item na área vermelha;
- **Missão lugar seguro:** consiste no robô estar na região vermelha no final do jogo. Pontuação: 25 pontos.

5.5 Informações gerais sobre a condução do trabalho

O desenvolvimento deste trabalho será feito utilizando a metodologia ágil, com *sprints* de em média duas semanas e *releases* a cada duas sprints. No início de cada

sprint, será feito o planejamento da sprint, definindo as issues que serão realizadas. Cada issue será registrada na ferramenta Waffle, que será integrada ao GitHub. Para controle de versão do código e da escrita do TCC, foi criada uma organização[2] no GitHub que contêm dois repositórios: um chamado Prolego[3], para controle de versão do código, e outro chamado Prolego_doc[4], para controle de versão da escrita do TCC. Toda a codificação será feita levando em consideração as boas práticas de programação. Na Figura 44 é possível visualizar a organização Prolego criada no GitHub, bem como os repositórios que possui.

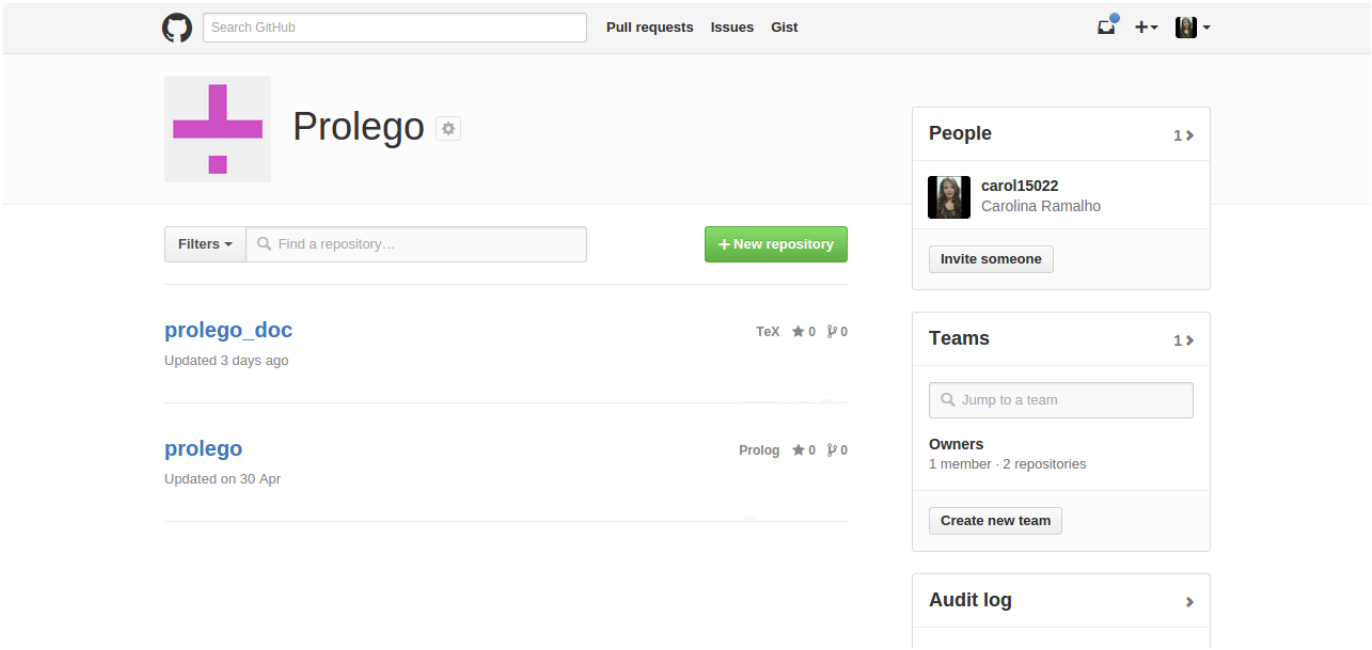

Figura 44 – Organização Prolego - GitHub

[2] https://github.com/Prolego
[3] https://github.com/Prolego/prolego
[4] https://github.com/Prolego/prolego_doc

5.6 Considerações parciais

A proposta desse trabalho consiste na criação de uma máquina de raciocínio para tomada de decisões estratégicas na robótica educacional, utilizando a linguagem de programação Prolog, a programação dinâmica e o robô Lego Mindstorms NXT.

Para auxiliar no cálculo do tempo de execução de cada missão, será utilizado o *framework* Traveller, que define a trajetória do robô dado um ponto inicial e um ponto final. A montagem do robô escolhida para ser utilizada por esse trabalho é a Tribot, uma montagem padrão do NXT. O tapete escolhido para ser trabalhado foi o *Nature's Fury*, um dos mais conhecidos e utilizados tapetes da LEGO.

Para que seja possível a programação do robô na linguagem Prolog, é necessária a utilização da IDE SWI-Prolog. Para a integração do robô com o Prolog, é necessário a instalação da plataforma PLNXT. Ambas as ferramentas são multiplataformas, Linux e Windows, porém nesse trabalho o sistema operacional escolhido foi o Linux. Todo o trabalho será versionado pelo GitHub e está disponível de forma livre, sendo possível acompanhar todo o desenvolvimento dessa máquina de raciocínio.

6 Considerações finais

Como exposto por este trabalho, a robótica educacional apoia o aprendizado do aluno de forma multidisciplinar, colaborando no desenvolvimento motor, intelectual e social. Para os universitários, a robótica é um apoio para o aprendizado de áreas relacionadas à inteligência artificial, como a otimização. O algoritmo guloso e a programação dinâmica são exemplos de algoritmos trabalhados pela inteligência artificial para a resolução de problemas de otimização. Nas bibliografias, a implementação dos algoritmos de otimização é geralmente feita em liguagens como C e C++. Pouca referência é encontrada para implementações na linguagem Prolog, sendo esse um dos obstáculos no desenvolvimento deste projeto.

A robótica é trabalhada na Universidade de forma heterogênia, com diversos tipos de hardware e software. O estudo desses vários tipos de hardware serviu de insumo para a decisão de se trabalhar com o kit Lego Mindstorms NXT neste projeto. Na definição das ferramentas utilizadas por este projeto teve-se o cuidado de buscar ferramentas que apoiassem não só à implementação, mas também a gerência de todo o projeto.

Como citado anteriormente, para o levantamento da solução da questão de pesquisa utilizou-se um modelo de pesquisa híbrida exploratória/experimental que apoiou a definição da proposta inicial de trabalho, o levantamento bibliográfico, e a prova de conceito. Para apoiar a construção da máquina de raciocínio será utilizado um modelo de pesquisa híbrido exploratória-descritiva.

A proposta de construção de uma máquina de raciocínio, implementada em Prolog segundo um algoritmo de programação dinâmica, que otimize a execução de missões pelo robô, provou-se viável através da prova de conceito exposta neste trabalho. A Figura 45 mostra a porcentagem de conclusão das tarefas previstas no cronograma.

Como na Faculdade do Gama o trabalho de conclusão de curso 02 tem um valor de créditos maior que o trabalho de conclusão de curso 01, assumiu-se que as tarefas previstas no cronograma para o TCC 01 tem um peso 1 e as previstas para o TCC 02 tem peso 1,5. Estes pesos são utilizados para calcular a porcentagem de conclusão do TCC como um todo, como mostra a Figura 46.

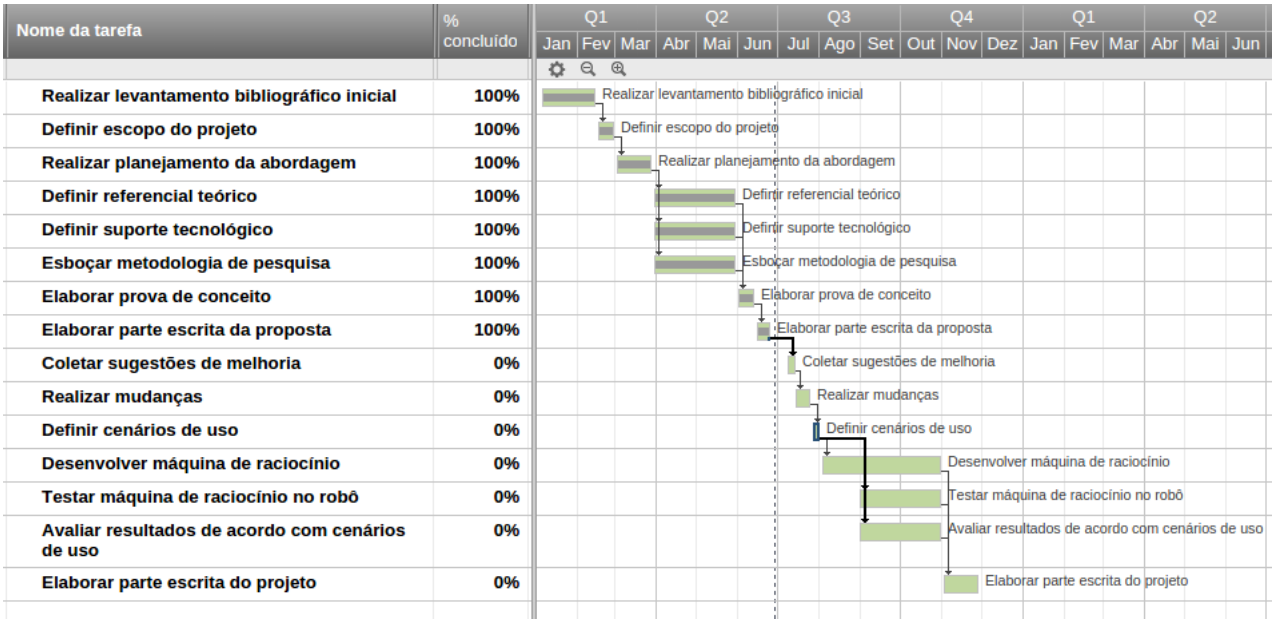

Figura 45 – Status das atividades previstas pelo cronograma

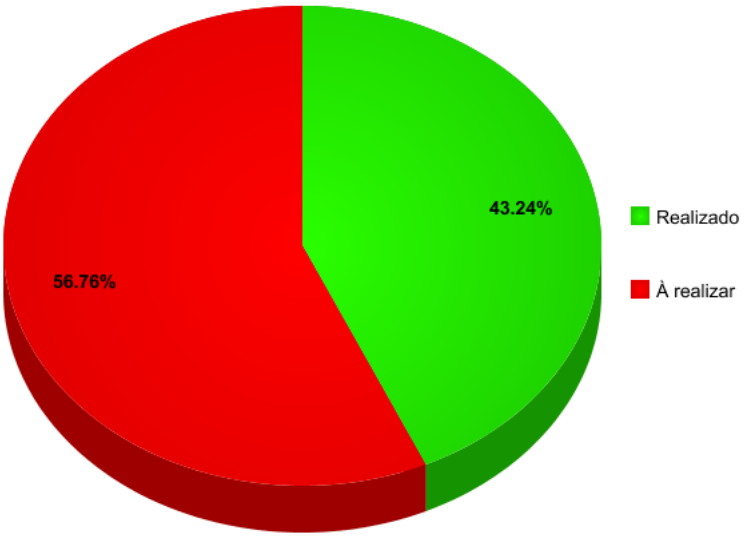

Figura 46 – Status do projeto

A preocupação do TCC agora é lidar com a conexão via *bluetooth* utilizando a ferramenta PLNXT. Assim que tal conexão estiver funcinando corretamente, será implementada a adaptação do problema da mochila para o contexto do trabalho. Em meios digitais, existem muitos códigos de exemplo da resolução do problema da mochila, utilizando programação dinâmica e implementados em Prolog que servirão como base para a implementação da máquina de raciocínio.

Referências

BARRIENTOS, A. et al. Fundamentos de robótica. *Ed. Mc Graw Hill,*, 1997. Citado 2 vezes nas páginas 30 e 31.

BENITTI, F. B. V. et al. Experimentação com robótica educativa no ensino médio: ambiente, atividades e resultados. In: *Anais do XXIX Congresso da Sociedade Brasileira de Computação, Bento Gonçalves/RS*. [S.l.: s.n.], 2009. p. 1811–1820. Citado na página 35.

CASTALONGA, J. P.; LEMOS, J. M. S. et al. Cobertura e empacotamento por circuitos através de um elemento em matróides. Universidade Federal de Pernambuco, 2007. Citado na página 42.

CLOCKSIN, W.; MELLISH, C. S. *Programming in PROLOG*. [S.l.]: Springer Science & Business Media, 2003. Citado na página 44.

CORMEN, T. H. *Introduction to algorithms*. [S.l.]: MIT press, 2009. Citado na página 43.

CURCIO, C. P. d. C. Instituto de tecnologia para o desenvolvimento (lactec) instituto de engenharia do paraná (iep) programa de pós-graduação em desenvolvimento de tecnologia (prodetec). 2008. Citado na página 31.

GIL, A. C. *Como elaborar projetos de pesquisa*. [S.l.]: São Paulo (SP): Atlas, 2002. Citado 5 vezes nas páginas 13, 53, 54, 55 e 56.

GRASSI, V. Arquitetura híbrida para robôs móveis baseada em funções de navegação com interação humana. *Escola Politécnica, USP.*, 2006. Citado na página 39.

GROOVER, M. P. *Robótica: tecnologia e programação*. [S.l.]: McGraw-Hill, 1989. Citado na página 31.

KAMEN, D.; KRISTIANSEN, K. K. First lego league. 2010. Citado na página 27.

LEVESQUE, H. J.; PAGNUCCO, M. Legolog: Inexpensive experiments in cognitive robotics. In: *Proceedings of the Second International Cognitive Robotics Workshop, Berlin, Germany*. [S.l.: s.n.], 2000. Citado na página 60.

MAISONNETTE, R. A utilização dos recursos informatizados a partir de uma relação inventiva com a máquina: a robótica educativa. *PROINFO-Programa Nacional de Informática na Educação, Curitiba-PR*, 2002. Citado na página 34.

MURPHY, R. *Introduction to AI robotics*. [S.l.]: MIT press, 2000. Citado na página 25.

NOF, S. Y. *Handbook of industrial robotics*. [S.l.]: John Wiley & Sons, 1999. Citado na página 30.

PIO, J. L. de S.; CASTRO, T. H. C. de; JÚNIOR, A. N. de C. A robótica móvel como instrumento de apoio à aprendizagem de computação. In: *Anais do Simpósio Brasileiro de Informática na Educação*. [S.l.: s.n.], 2006. v. 1, n. 1, p. 497–506. Citado na página 31.

PRODANOV, C. C.; FREITAS, E. C. de. *Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico-2ª Edição*. [S.l.]: Editora Feevale, 2013. Citado 3 vezes nas páginas 53, 54 e 56.

REDEL, R.; HOUNSELL, M. d. S. Implementação de simuladores de robôs com o uso da tecnologia de realidade virtual. In: *IV Congresso Brasileiro de Computação, Itajaí–SC. IV CBCOMP*. [S.l.: s.n.], 2004. v. 1, p. 398–401. Citado na página 31.

RINCON, R. L. Framework de definição de trajetória para robôs móveis. 2014. Citado 4 vezes nas páginas 14, 37, 75 e 76.

ROCHA, A. Algoritmos gulosos: definições e aplicacções. Campinas, SP, 2004. Citado 2 vezes nas páginas 41 e 42.

RODRIGUES, T. G. *Sobre os Fundamentos da Programação Lógica Paraconsistente*. Tese (Doutorado) — Universidade Estadual de Campinas, 2010. Citado na página 44.

RUSSEL, S.; NORVIG, P. Inteligência artificial. *Editora Campus*, 2004. Citado na página 26.

SANTOS, C. C. d. Robótica na construção: uma aplicação prática. Universidade de Coimbra, 2002. Citado 2 vezes nas páginas 30 e 31.

SANTOS, K. *SISTEMA DE NAVEGAÇÃO AUTÔNOMA PARA ROBÔS MÓVEIS BASEADO EM ARQUITETURA HÍBRIDA: TEORIA E APLICAÇÃO*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DE ITAJUBÁ, 2009. Citado 2 vezes nas páginas 39 e 40.

SILVA, A. F. D. *RoboEduc: Uma metodologia de aprendizado com Robótica Educacional*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE, 2009. Citado 2 vezes nas páginas 25 e 26.

SILVA, A. F. de et al. Utilização da teoria de vygotsky em robótica educativa, em ‘. In: IX CONGRESO IBEROAMERICANO DE INFORMATICA EDUCATIVA RIBIE. [S.l.], 2008. Citado na página 31.

TORVALDS, L.; BY-DIAMOND, D. R. *Just for fun: The story of an accidental revolutionary*. [S.l.]: Harper Audio, 2001. Citado na página 50.

TUCKER, A.; NOONAN, R. *Linguagens de Programação: Princípios e Paradigmas*. [S.l.]: Grupo A Educação, 2009. Citado na página 27.

VIEIRA, F. C. *Controle dinâmico de robôs móveis com acionamento diferencial*. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, 2005. Citado 4 vezes nas páginas 14, 37, 75 e 76.

WOLF, D. F. et al. Robótica móvel inteligente: Da simulação às aplicações no mundo real. In: *Mini-Curso: Jornada de Atualização em Informática (JAI), Congresso da SBC*. [S.l.: s.n.], 2009. Citado 5 vezes nas páginas 13, 37, 38, 39 e 40.

ZILLI, S. d. R. et al. A robótica educacional no ensino fundamental: perspectivas e prática. Florianópolis, SC, 2004. Citado 3 vezes nas páginas 32, 34 e 35.