



Federação das Indústrias do Estado da Bahia

CENTRO UNIVERSITÁRIO SENAI CIMATEC

Engenharia de Controle e Automação

Engenharia Mecânica

Trabalho de Conclusão de Curso - Metodologia TheoPrax

Projeto *open source* para aprendizagem de robótica móvel e inteligência artificial

Apresentada por:

Caio Amaral
Élisson Oliveira
Iure Pinheiro
Mateus Meneses

Orientador:

Prof. Marco Reis, M.Eng.

Dezembro de 2019

Caio Amaral
Élisson Oliveira
Iure Pinheiro
Mateus Meneses

Projeto *open source* para aprendizagem de robótica móvel e inteligência artificial

Trabalho de Conclusão de Curso - Metodologia TheoPrax apresentada ao Programa de Graduação em Engenharia de Controle e Automação e Engenharia Mecânica do Centro Universitário SENAI CIMATEC, como requisito parcial para a obtenção do título de **Bacharel em Engenharia**.

Área de conhecimento: Robótica
Orientador: Prof. Marco Reis, M.Eng.

Salvador
Centro Universitário SENAI CIMATEC
2019

Centro Universitário SENAI CIMATEC

Engenharia de Controle e Automação

Engenharia Mecânica

A Banca Examinadora, constituída pelos professores abaixo listados, leram e recomendam a aprovação do Trabalho de Conclusão de Curso - Metodologia TheoPrax, intitulado “Projeto *open source* para aprendizagem de robótica móvel e inteligência artificial”, apresentada no dia 12 de Dezembro de 2019, como requisito parcial para a obtenção do título de **Bacharel em Engenharia**.

Orientador:

Prof. Me. Marco Antonio dos Reis

Centro Universitário SENAI CIMATEC

Orientador TheoPrax:

Prof. MBA João Lucas da Hora

Centro Universitário SENAI CIMATEC

Coordenador de curso:

Prof. Dr. Guilherme Oliveira de Souza

Centro Universitário SENAI CIMATEC

Coordenador de curso:

Prof. Me. Taniel Silva Franklin

Centro Universitário SENAI CIMATEC

Agradecimentos

Agradecemos primeiramente ao Brazilian Institute of Robotics (BIR), através da figura de Leonardo Nardy e Marco Reis, pela oportunidade e confiança em nossa equipe na realização deste projeto; e ainda mais diretamente ao professor Marco por todo suporte que nos foi oferecido como orientador ao longo desse TheoPrax. Ao professor João da Hora, por da mesma forma ter acompanhado nossas atividades e pela sua cooperação em nos conciliar com a metodologia TheoPrax, mesmo com as especificidades trazidas por esta Equipe. Aos coordenadores de curso Guilherme Souza e Taniel Franklin, pela agilidade e disponibilidade, mesmo em final de semestre, em possibilitarem a execução tanto da banca quanto a pré-banca. Por fim, a Equipe agradece ao SENAI Cimatec por toda sua infraestrutura, seja pelo Laboratório TheoPrax, que nos serviu como base para a montagem do robô, ou pela cooperação com seus demais setores que nos concedeu parte dos materiais necessários para a confecção dos nossos protótipos, a vocês nosso muito obrigado!

Eu, Élisson Oliveira, agradeço à minha família, pelo apoio durante toda essa jornada. Agradeço pela compreensão de minhas duas mães, Dona Zina e Sandra, durante os momentos de minha ausência. Agradeço ao meu pai, Sr. Roberto, sem o qual eu não teria conquistado nada do que conquistei até aqui. Agradeço à Alana, pela parceria, incentivo e conforto nos momentos difíceis. Por fim, agradeço à todos os meus amigos, os mais próximos e os que estão mais distantes, por todo o aprendizado e companhia durante esse percurso.

Eu, Iure Pinheiro, gostaria de agradecer a toda minha família pelo apoio durante todo o processo, em especial ao meus pais Romilda e Antônio que sempre me incentivaram na vida acadêmica e que sem eles eu não conseguiria ter chegado nesse momento. Agradeço à Tais por nunca ter me deixado desanimar durante os momentos mais difíceis e por ser um exemplo de determinação para mim. Gostaria de agradecer também à Mateus pelos conselhos e encaminhamentos no decorrer do curso e a Vitor pelas conversas que foram necessárias para retirar o stress diário. Agradeço também ao meu primo Davi e meu amigo de longa data Leandro pelas palavras de motivação e conforto. Por último e não menos importante, agradeço a todos os meus amigos que não foram citados mas que contribuíram mesmo que indiretamente para a conclusão deste ciclo.

Eu, Mateus Meneses, agradeço especialmente a meus pais Arnaldo e Maria por todo apoio emocional e incentivo ao longo dessa longa jornada. Agradeço também aos meus irmãos e irmã por oferecer um ambiente familiar tão sólido e motivador, o que foi de extrema importância para lidar com os momentos difíceis ao longo da graduação. Agradeço ao apoio desde a infância dos meus amigos Renato e Walter e suas respectivas famílias. Agradecimentos a Iure e Vitor pelos momentos de descontração e as conversas aleatórias nos corredores do SENAI CIMATEC. A todos os colegas do BIR, muito obrigado por estarem sempre dispostos a compartilharem o conhecimento técnico, profissional e pessoal de vocês. Meus agradecimentos também a Fred por compartilhar as experiências e desafios do TheoPrax. Por fim, agradeço a todos aos meus amigos que não foram citados diretamente aqui mas que de alguma forma me ajudaram durante os anos de graduação.

Resumo

Estudos em Inteligência Artificial tem crescido e se diversificado em diferentes aplicações, e muitas vezes associadas à robótica, necessitando cada vez mais de profissionais capacitados para seu uso e exploração em diferentes ambientes. Por conta dessa necessidade, plataformas abertas de robótica podem ser uma potencial solução para o aprendizado introdutório dessas duas áreas. Portanto, o objetivo deste trabalho é desenvolver uma plataforma *open source* baseada nos modelos utilizados na competição Micromouse. Essa plataforma será composta de um protótipo, ambiente de simulação e um labirinto para testes. Foi utilizada uma metodologia dividida em quatro partes: conceitual, design, desenvolvimento e conclusão. As fases dessa metodologia geraram materiais que culminaram na elaboração de dois protótipos com guias de montagem e configuração, ambiente de simulação utilizando ROS e Gazebo, proposta de labirinto modular e um pacote de software para controle do robô. Por fim, é apresentado ao final do trabalho as conclusões obtidas, possíveis abordagens de uso da plataforma e trabalhos futuros.

Palavras-chave: Inteligência Artificial, Robótica Móvel, Ensino de Robótica

Abstract

Studies in Artificial Intelligence have grown and diversified in different applications, and often associated with robotics, increasingly requiring trained professionals for its use and exploration in different environments. Because of this need, open robotics platforms can be a potential solution for introductory learning in these two areas. Therefore, the objective of this work is to develop an open source platform based on the models used in the Micromouse competition. This platform will consist of a prototype, a simulation environment and a test maze. A methodology divided into four parts was used: conceptual, design, development and conclusion. The phases of this methodology generated materials that culminated in the elaboration of two prototypes with assembly and configuration guides, a simulation environment using ROS and Gazebo, a modular maze proposal and a robot control software package. Finally, it is presented at the end of the work the conclusions obtained, possible approaches to use the platform and future work.

Keywords: Artificial Intelligence, Mobile Robotics, Robotics Teaching

Sumário

| | |
|---|------------|
| Lista de abreviatura e siglas | vii |
| 1 Introdução | 1 |
| 1.1 Objetivo | 2 |
| 1.1.1 Objetivos Específicos | 3 |
| 1.2 Justificativa | 3 |
| 2 Fundamentação Teórica | 5 |
| 2.1 Inteligência artificial e algoritmos de busca | 5 |
| 2.1.1 Algoritmo de Dijkstra | 6 |
| 2.1.2 Algoritmo A* | 6 |
| 2.1.3 Flood Fill | 7 |
| 2.2 Micromouse | 8 |
| 2.3 Robótica e <i>Frameworks</i> | 9 |
| 2.4 Estudo do Estado da Arte | 13 |
| 2.4.1 GreenGiant | 13 |
| 2.4.2 WPISmartMouse | 15 |
| 2.4.3 Kumamoto National College | 16 |
| 2.4.4 WolfieMouse | 18 |
| 2.4.5 Raspberry Pi Mouse V2 | 20 |
| 2.4.6 Matriz de Comparação | 21 |
| 3 Materiais e Métodos | 22 |
| 3.1 Metodologia | 22 |
| 3.1.1 Conceitual | 23 |
| 3.1.2 Design | 23 |
| 3.1.3 Desenvolvimento | 24 |
| 3.1.4 Conclusão | 25 |
| 3.2 Requisitos do projeto | 25 |
| 3.3 Descrição do sistema | 26 |
| 3.3.1 Interface de usuário | 27 |
| 3.3.2 Interface de Hardware | 28 |
| 3.3.3 Controle do sistema | 28 |
| 3.4 Especificação dos componentes | 29 |
| 3.4.1 Estrutura analítica do protótipo | 29 |
| 3.4.2 Raspberry Pi Zero W v1.1 | 31 |
| 3.4.3 Motor, encoder e ponte H | 31 |
| 3.4.4 Emissor e fotorreceptor Infravermelho | 33 |
| 3.4.5 IMU | 33 |
| 3.4.6 Conversor Analógico/Digital e Multiplexador Analógico | 34 |
| 3.4.7 Lista de componentes | 35 |
| 3.5 Modelo mecânico | 35 |
| 3.5.1 Plataforma Robótica | 35 |
| 3.5.2 Labirinto | 37 |
| 3.6 Modelo esquemático de alimentação e comunicação | 38 |

| | | |
|--------------------|---|------------|
| 3.6.1 | Esquemáticos eletrônicos | 39 |
| 3.7 | Especificação das funcionalidades | 40 |
| 3.7.1 | Localização | 41 |
| 3.7.1.1 | Objetivo | 41 |
| 3.7.1.2 | Dependências | 42 |
| 3.7.1.3 | Premissas | 42 |
| 3.7.1.4 | Saídas | 42 |
| 3.7.2 | Percepção | 43 |
| 3.7.2.1 | Objetivo | 43 |
| 3.7.2.2 | Dependências | 43 |
| 3.7.2.3 | Premissas | 44 |
| 3.7.2.4 | Saídas | 44 |
| 3.7.3 | Navegação | 45 |
| 3.7.3.1 | Objetivo | 45 |
| 3.7.3.2 | Dependências | 45 |
| 3.7.3.3 | Premissas | 45 |
| 3.7.3.4 | Saída | 46 |
| 3.7.4 | Usabilidade | 46 |
| 3.7.4.1 | Objetivo | 47 |
| 3.7.4.2 | Dependências | 47 |
| 3.7.4.3 | Premissas | 47 |
| 3.7.4.4 | Saídas | 47 |
| 3.8 | Simulação | 48 |
| 4 | Resultados | 52 |
| 4.1 | Protótipo | 52 |
| 4.2 | Ambiente de simulação | 57 |
| 4.2.1 | Comparativo de Malhas | 57 |
| 4.2.2 | Modelos da Simulação | 58 |
| 4.3 | Estrutura de Software | 60 |
| 5 | Conclusão | 65 |
| 5.1 | Trabalhos futuros | 65 |
| A | Matriz de Comparação dos robôs <i>micromouse</i> estudados | 67 |
| B | Requisitos do cliente x Requisitos técnicos | 68 |
| C | Lista de componentes | 69 |
| D | Desenhos técnico do Doogie Mouse | 70 |
| E | Desenhos técnico do labirinto | 72 |
| F | Esquemáticos eletrônicos | 75 |
| G | Guia de montagem do Doogie Mouse | 80 |
| H | Guia de configuração de software do Doogie Mouse | 102 |
| Referências | | 114 |

Lista de Tabelas

| | | |
|-----|---|----|
| 2.1 | Atributos do robô Green Giant | 14 |
| 2.2 | Atributos do robô WPISmartmouse | 16 |
| 2.3 | Atributos do robô Kumamoto | 17 |
| 2.4 | Atributos do robô WolfieMouse | 19 |
| 2.5 | Atributos do robô RaspiMouse | 21 |
| 3.1 | Especificações técnicas do motor <i>Micro Metal Gearmotor HP 6V with Extended Motor Shaft</i> | 32 |
| 4.1 | Comparativo entre malhas carregadas no Gazebo | 57 |
| 4.2 | Comparativo com simplificação da malha de colisão | 57 |

Lista de Figuras

| | | |
|------|---|----|
| 2.1 | Moonlight Special - Primeiro modelo <i>micromouse</i> a ganhar uma competição. | 8 |
| 2.2 | Biblioteca x <i>Framework</i> . | 11 |
| 2.3 | Estrutura de comunicação. | 13 |
| 2.4 | Robô Green Giant. | 14 |
| 2.5 | Robô WPISmartMouse | 15 |
| 2.6 | Robô Kumamoto. | 17 |
| 2.7 | Robô WolfieMouse. | 18 |
| 2.8 | Robô RaspiMouse. | 20 |
| 3.1 | Metodologia empregada no desenvolvimento do projeto solução. | 22 |
| 3.2 | Arquitetura Geral do robô Doogie. | 27 |
| 3.3 | <i>Prototype Breakdown Structure</i> do Doogie Mouse | 30 |
| 3.4 | Modelo 3D do Doogie Mouse | 36 |
| 3.5 | Modelo 3D Renderizado do Doogie Mouse | 37 |
| 3.6 | Modelo mecânico do labirinto | 38 |
| 3.7 | Representação elétrica do Doogie Mouse. | 39 |
| 3.8 | <i>Layout</i> das PCIs do Doogie Mouse | 40 |
| 3.9 | Fluxo de informações das funcionalidades Localização, Percepção, Navegação e Usabilidade. | 41 |
| 3.10 | Fluxograma ilustrativo da funcionalidade de Localização. | 43 |
| 3.11 | Fluxograma ilustrativo da funcionalidade de Percepção. | 44 |
| 3.12 | Fluxograma ilustrativo da funcionalidade de Navegação. | 46 |
| 3.13 | Fluxograma ilustrativo da funcionalidade de Usabilidade. | 48 |
| 3.14 | Modelo do Doogie Mouse para Simulação. | 49 |
| 3.15 | Descrição Mecânica do Doogie Mouse. | 50 |
| 3.16 | Modelo do Labirinto para Simulação. | 51 |
| 4.1 | Componentes da placa superior do Doogie Mouse | 53 |
| 4.2 | Componentes da placa inferior do Doogie Mouse | 54 |
| 4.3 | Protótipos do Doogie Mouse confeccionados | 54 |
| 4.4 | Recorte do guia de instruções de montagem do Doogie Mouse | 55 |
| 4.5 | Recorte do guia de instruções de configuração de software do Doogie Mouse | 56 |
| 4.6 | Doogie Mouse no Gazebo | 58 |
| 4.7 | Labirinto na Simulação | 59 |
| 4.8 | Organização dos pacotes do Doogie Mouse. | 61 |
| 4.9 | Relação de dependência das <i>stacks</i> do Doogie Mouse. | 62 |
| 4.10 | Diagrama de comunicação entre processos. | 63 |

Lista de abreviatura e siglas

A/D Analógico/Digital

ABDI Agência Brasileira de Desenvolvimento Industrial

APEC *Applied Power Electronics Conference*

API *Application Programming Interface*

BIR *Brazilian Institute of Robotics*

CAD *Computer-Aided Design*

COLLADA *COLLAborative Design Activity*

CPR *Counts Per Revolution*

FPS *Frames per Second*

GPIO *General Purpose Input/Output*

IA Inteligência Artificial

IBM *International Business Machines*

IEEE *Institute of Electrical and Electronics Engineers*

IMU *Inertial Measurement Unit*

IoT *Internet of Things*

PBS *Prototype Breakdown Structure*

PCI Placa de Circuito Impresso

PWM *Pulse Width Modulation*

QFD *Quality Functional Deployment*

ROS *Robot Operating System*

RTF *Real Time Factor*

SLAM *Simultaneous Localization and Mapping*

SOs Sistemas Operacionais

SOTA *State Of The Art*

SSH *Secure Shell*

STL *Stereolithography*

URDF *Unified Robot Description Format*

Introdução

Por muito tempo a humanidade conjectura sobre o seu futuro. Cidades inteligentes, Indústria 4.0, veículos voadores, carros autônomos e a Inteligência Artificial (IA) são só alguns dos conceitos frequentes que permeiam esse cenário. Mas justamente a IA, popularmente relacionada como uma versão artificial da inteligência humana, tem se mostrado uma eficiente ferramenta no desenvolvimento de novas tecnologias que compõe o ideário do mundo moderno.

Contudo, a ideia de criações humanas dotadas de inteligência não é nova. Mesmo em culturas mais antigas é possível encontrar sua presença dentro das mitologias, conforme visto em [Smith \(1849\)](#) as histórias dos automatas gregos desenvolvidos pelo inventor ateniense, Dédalo. No entanto, o desenvolvimento de IA, no formato em que é utilizada atualmente, só começou a ganhar real destaque a partir do final do século 20. A própria década de 90 foi um marco para a popularização da IA, a partir da vitória histórica do Deep Blue, desenvolvido pela *International Business Machines* (IBM), contra o ex-campeão Garry Kasparov em uma partida de xadrez ([MCCORDUCK, 2004](#)).

Desde então, os estudos em IA tem crescido e se diversificado em diferentes aplicações, e muitas vezes associadas à robótica, necessitando cada vez mais de profissionais capacitados para seu uso e exploração em diferentes ambientes. Exploração espacial, fábricas inteligentes, veículos autônomos, reconhecimento visual são só algumas das possíveis aplicações de IA. Porém, toda essa diversidade pode tornar-se um empecilho para aqueles que desejam iniciar seus estudos no segmento.

Por conta dessa necessidade, plataformas abertas de robótica, como Turtlebot ([FOUNDATION, 2019b](#)) e ROSbot 2.0 ([FOUNDATION, 2019a](#)), podem ser uma potencial solução para o aprendizado introdutório dos conceitos, ou mesmo no desenvolvimento inicial de soluções em IA. Além disso, a partir do uso das plataformas, há abertura para a introdução das próprias disciplinas da robótica, como visão e robótica móvel, fortalecendo mais a formação do estudante.

Todavia, conforme [Neto et al. \(2015\)](#) 53,48% dos robôs usados em pesquisas nacionais são construídos no Brasil, devido aos altos custos na exportação dessas plataformas vindas do exterior. Por isso, faz-se necessário a existência de uma plataforma de robótica nacional que possa ser facilmente adquirida pelos estudantes e pesquisadores do Brasil. Em verdade, o contato inicial com a robótica durante a graduação é geralmente feito por

exemplo com base na ótica ”Currículo por Objetivo/Competição, na qual os estudantes tem o seu aprendizado a partir do desenvolvimento de atividades visando a participação em uma das diversas competições de robótica ([CAMPOS, 2017](#)), muitas delas inclusive organizadas pelo *Institute of Electrical and Electronics Engineers* (IEEE), como os robôs seguidor de linha, sumô e micromouse, por exemplo. Esses robôs de competição já são conhecidos pelos estudantes de graduação e por isso uma plataforma inspirada em alguma dessas configurações voltadas para as competições IEEE serão mais familiares aos estudantes, facilitando o seu aprendizado.

Em contrapartida, um *middleware* — interface de tradução, responsável por realizar comunicação e gerenciamento de dados — pode enriquecer ainda mais o aprendizado a partir da plataforma.. Pois, um *middleware* de robótica permite, conforme descrito por [Elkady e Sobh \(2012\)](#):

[...]lidar com a complexidade e heterogeneidade encontrada em aplicações de hardware, promovendo integração de novas tecnologias, simplificando o design de software, ocultando a complexidade da comunicação de baixo nível e da heterogeneidade dos sensores, melhorando a qualidade dos softwares, reutilizando a infraestrutura do software de robótica em diferentes esforços de pesquisa e reduzindo os custos de produção.

Dessa forma, o *middleware*, a partir de sua coleção de ferramentas e bibliotecas, facilita o desenvolvimento de robôs de diferentes fabricantes, com diferentes infraestruturas de software, criando um ambiente de desenvolvimento comum na robótica, inclusive aproveitando soluções anteriores em novas aplicações, com diferentes robôs. E justamente por isso, os *middlewares* abertos mais populares, como o RT-Middleware e o *Robot Operating System* (ROS) ([ELKADY; SOBH, 2012](#)), que contam com uma grande comunidade de colaboradores no seu desenvolvimento com novas aplicações, são mais atrativos para novos usuários por conta de sua versatilidade que cresce junto com a popularidade da ferramenta.

Logo, a combinação de uma plataforma aberta acessível com o uso de um *middleware* de robótica, mostra-se uma boa estratégia no ensino tanto de conceitos introdutórios da IA, quanto das próprias disciplinas da robótica.

1.1 *Objetivo*

O objetivo deste trabalho é desenvolver uma plataforma *open source* baseado nos modelos utilizados na competição regulada pelo IEEE intitulada Micromouse. A plataforma será dotada de sensores, atuadores e de interfaces de usuário. O robô utilizará o *framework*

ROS utilizado mundialmente em projetos de robôs de cunho organizacional e acadêmico. Será desenvolvido também um ambiente de simulação no simulador Gazebo o qual possui integração com o ROS. Portanto, essa plataforma pode ser utilizada em ambientes acadêmicos de nível superior tanto para ensino quanto para competições.

1.1.1 *Objetivos Específicos*

- Estudar teoria de inteligência artificial e robótica móvel;
- Desenvolver e integrar com *framework* ROS ambiente de simulação do robô utilizando o simulador Gazebo;
- Confeccionar plataforma física, contemplando sensores, atuadores, *display* e elementos mecânicos e eletrônicos;
- Desenvolver pacotes de software e *drivers* para controle, sensoriamento, planejamento e interação com usuário do robô utilizando o *framework* ROS;
- Elaborar documentação da plataforma tais como: guia do usuário, tutoriais online, esquemáticos eletrônico, desenhos mecânicos e diagramas em geral;
- Confeccionar dois labirintos, sendo o primeiro para testes em bancada e o segundo para validação da plataforma e uso em competições;
- Realizar testes para validação do funcionamento da plataforma.

1.2 *Justificativa*

O cenário industrial mundial passou por grandes mudanças desde a primeira revolução industrial no final do século XVIII até os dias atuais. Atualmente, com o avanço da tecnologia e a sua inserção na indústria, uma nova revolução intitulada Indústria 4.0 está em andamento. Segundo a [CNI \(2016, p. 11\)](#):

A incorporação da digitalização à atividade industrial resultou no conceito de Indústria 4.0, em referência ao que seria a 4^a revolução industrial, caracterizada pela integração e controle da produção a partir de sensores e equipamentos conectados em rede e da fusão do mundo real com o virtual, criando os chamados sistemas ciberfísicos e viabilizando o emprego da inteligência artificial.

Essa nova revolução irá ocasionar impactos na economia nacional. De acordo com um levantamento da Agência Brasileira de Desenvolvimento Industrial (ABDI), o Brasil irá

reduzir no mínimo R\$ 73 bilhões/ano, sendo que serão R\$ 34 bilhões com ganhos na eficiência no processo produtivo, R\$ 31 bilhões com redução de custos de manutenção de máquina e R\$ 7 bilhões com economia de energia ([ROTTA, 2017](#)).

Destaca-se algumas tecnologias com maior influência nessa revolução tais como a internet das coisas (*Internet of Things (IoT)*), *big data*, computação em nuvem, inteligência artificial e robótica avançada. Para [Mies e Zentay \(2017\)](#) o avanço da indústria em direção a fábricas mais inteligentes está vinculado com a evolução da automação, que devido a isso, torna a robótica um elemento crucial para a Indústria 4.0.

Robôs móveis, dotados de inteligência artificial, são capazes de calcular a melhor rota do ponto A ao ponto B. Num ambiente industrial — que pode perfeitamente ser modelado como um labirinto — tal habilidade garante que entregas de materiais sejam feitas de forma eficiente, reduzindo custos e evitando paradas na produção.

Para [CNI \(2016\)](#) a consolidação da Indústria 4.0 no Brasil virá com possíveis consequências, dentre elas, o surgimento de novas atividades e novas profissões, que demandarão adaptações no padrão de formação de recursos humanos. A previsão é de que suas implicações alcançarão tanto a pequena e média indústria, quanto os segmentos mais amplos como as indústrias química e de processamento de óleo e gás ([GONZALEZ; CALDERON, 2018](#)).

Entretanto, as universidades, que são os principais centros de formação de mão de obra, trouxeram poucas implementações em suas metodologias de ensino quando se trata da Indústria 4.0. Um estudo realizado por [Neto et al. \(2015\)](#) revela que, dos 65 artigos publicados no Brasil entre os anos 2004 e 2014 a respeito de Robótica Educacional, apenas 25% tinham como foco o ensino superior.

Deste modo, no cenário em que as expertises e qualificações dos trabalhadores serão diretamente responsável pelo sucesso dos empreendimentos na nova indústria, as companhias e as universidades deverão buscar estratégias para o treinamento e capacitação de seus recursos humanos que encurtem a distância entre a teoria e a prática, conforme sustentado nos trabalhos de ([BENESOVA; TUPA, 2017](#)) e ([GONZALEZ; CALDERON, 2018](#)).

Portanto, visto que, conforme trazido por ([KHOMECHENKO; GEBEL; PESHKO, 2018](#)), os conceitos necessários para implementar a Indústria 4.0 são bastante similares, diferindo-se em especial quanto a escala das aplicações, é desejável a elaboração de um projeto em que conceitos de robótica móvel, inteligência artificial, simulação e *frameworks* de robótica possam ser compreendidos dentro de processos colaborativos de capacitação ou em ambientes acadêmicos, principalmente em cursos de engenharia.

Fundamentação Teórica

2.1 *Inteligência artificial e algoritmos de busca*

Na medida em que o mundo se torna mais complexo, novas técnicas de resolução de problemas são necessárias. Nesse sentido, a IA vem sendo de grande ajuda: atividades que duram dias para um ser humano comum realizar podem ser finalizadas em apenas alguns segundos por um computador. Desde diagnósticos de doenças raras à concepção do carro autônomo, a IA está se tornando cada vez mais uma realidade.

O objetivo principal das técnicas de inteligência artificial é construir agentes inteligentes que agem de forma racional. Segundo [Winston \(1992, p. 5\)](#), “Inteligência artificial é o estudo de computações que permitem perceber, raciocinar e agir”. Sobre essa ótica, as máquinas devem ser capazes de tomar diferentes decisões, com base em suas próprias experiências.

Boa parte dos problemas que podem ser resolvidos com IA podem ser caracterizados como um “problema de busca”. Planejamento de rotas, ordenação, navegação de robôs e resolução de quebra-cabeças são alguns dos vários exemplos que podem ser citados. Todos esses problemas possuem a mesma estrutura e consistem de: um espaço de estados, um estado inicial e um objetivo final. O espaço de estados é o conjunto de todos os estados em que se pode estar. O estado inicial é onde a busca se inicia. E, por fim, o objetivo final é onde se quer chegar ([AKHTAR, 2019](#)).

Durante a realização de uma busca, diversas possibilidades de ações a serem tomadas podem estar disponíveis ao mesmo tempo (virar à esquerda, direita, ou seguir em frente, por exemplo). O papel do algoritmo é analisar todas as possibilidades para conseguir definir a sequência de ações que fará o agente sair do estado inicial até o objetivo final. Em determinadas situações, cada ação pode estar associada também à um custo. Dessa forma, a maior parte dos agentes inteligentes irá procurar não só o objetivo final, mas também a melhor forma de se chegar até ele ([ENDRISS, 2015](#)).

Usualmente, o espaço de estados pode ser representado por nós, onde cada nó é um estado. A representação dos nós varia a depender do problema. No contexto de planejamento de rotas, os nós são as posições no mapa, por exemplo. Para a resolução de um labirinto, o espaço de estados pode ser representado por uma matriz, onde cada célula da matriz é uma posição do labirinto em que o agente pode estar.

Grande parte dos algoritmos de busca podem ser adaptados para a resolução de um labirinto. Porém, conforme demonstrado por [Tjiharjadi, Wijaya e Setiawan \(2017\)](#), destacam-se: Flood Fill e A*. O A*, por sua vez, pode ser considerado uma derivação do algoritmo de Dijkstra.

2.1.1 Algoritmo de Dijkstra

O método de Dijkstra foi concebido em 1956 por Edsger W. Dijkstra. Foi um dos primeiros a serem publicados. Nele, inicialmente, considera-se todos os nós como não visitados. Atribui-se valor zero para a estimativa de custo do nó inicial e infinito para todos os outros nós. Em seguida, é calculada a estimativa de custo de deslocamento para todos os nós vizinhos, passando pelo nó atual. O nó atual é marcado como “visitado” e o mesmo processo é repetido para o próximo nó vizinho, que possui o menor custo. O próximo nó terá a sua estimativa de custo somada com a do nó anterior (caso a soma seja menor do que a estimativa de custo já associada a ele), uma vez que para chegar até ele houve um custo associado. Nós que já foram visitados nunca serão checados novamente. O processo é repetido até que o nó final seja alcançado.

2.1.2 Algoritmo A*

A* (pronunciado A-estrela) foi publicado inicialmente por Peter Hart, Nils Nilsson e Bertram Raphael em 1968. Seus criadores provaram que tal método garante que um caminho será encontrado, se o mesmo existir ([HART; NILSSON; RAPHAEL, 1968](#)). Pode ser considerado como uma extensão do algoritmo de Dijkstra, utilizando heurística para melhorar sua performance.

Nele, são utilizadas duas listas: aberta e fechada. A lista aberta é composta pelos nós que foram visitados, mas não expandidos. Ou seja: os vizinhos ainda não foram explorados. Pode ser considerada como uma lista de pendências. A lista fechada consiste nos nós que que foram visitados e já expandidos.

O algoritmo procura minimizar $f(n) = g(n) + h(n)$, onde $g(n)$ é o custo do inicio do caminho até o nó atual e $h(n)$ é a função heurística que estima o menor custo do nó atual até o objetivo. Inicialmente, o nó atual é adicionado à lista aberta e os seguintes passos são repetidos ([CUI; SHI, 2010](#)):

- Olha para o nó com o menor custo de $f(n)$ na lista aberta e se refere a ele como o nó atual;

- Transfere o nó atual para a lista fechada;
- Para cada nó vizinho do nó atual:
 - Se está na lista fechada, ignora o mesmo;
 - Se não está na lista aberta, adiciona-o a mesma. Faz com que o nó atual seja o “nó pai” do mesmo e grava f, g e h para ele;
 - Se já está na lista aberta, verifica se há um melhor caminho. Se sim, muda o pai para o nó atual e recalcula os valores de f e g.
- O algoritmo termina quando:
 - O nó alvo é adicionado à lista fechada;
 - A lista aberta está vazia e não foi possível encontrar o nó alvo.

Fazendo o caminho contrário do nó alvo até o nó inicial, é possível achar o caminho.

2.1.3 Flood Fill

O Flood Fill é um dos melhores e mais utilizados algoritmos de resolução de labirinto. Ele é capaz de descobrir paredes e analisar a melhor rota, ao mesmo tempo.

Cada célula do labirinto é preenchida com um valor que indica a distância (em quantidade de células percorridas) para chegar ao alvo. Esse processo é feito, inicialmente, considerando que o labirinto não possui nenhuma parede. O menor peso é atribuído a célula de destino e, a partir dela, os pesos das outras células adjacentes vão sendo definidos. Em seguida, uma sequência de 4 passos é repetida até que o robô alcance o objetivo final ([JABBAR, 2016](#)). Para cada nova célula que o robô passar, o mesmo deverá:

1. Definir os pesos das células: o processo de enchente começa do destino final até o início. Os pesos são atribuídos de forma crescente.
2. Verificar as paredes da célula: antes de decidir para onde vai, o robô deve identificar possíveis obstáculos em seu caminho.
3. Definir qual a próxima célula, baseando-se nas paredes ao seu redor, nos pesos das células vizinhas e na quantidade de giros que o mesmo precisa realizar.
4. Deslocar-se para a próxima célula.

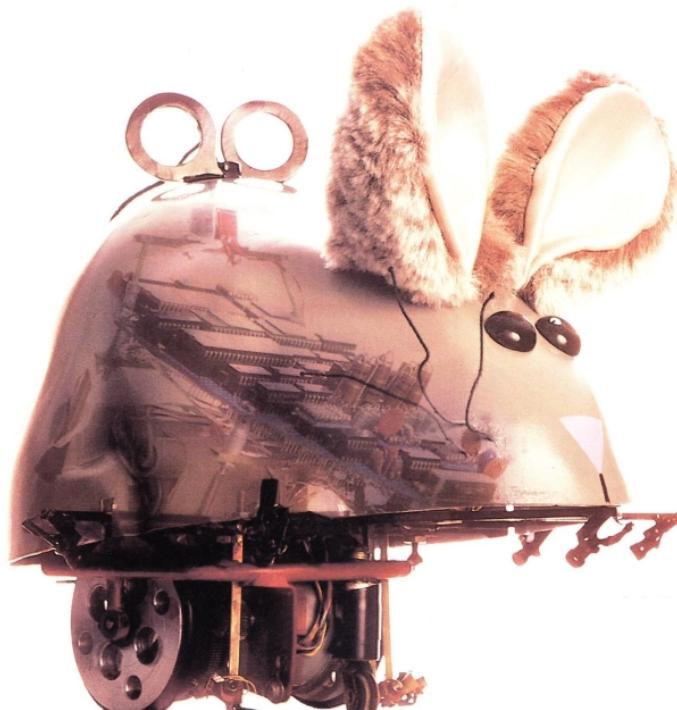
Toda vez que o robô identifica uma parede, o processo de enchente irá atribuir pesos diferentes para cada célula, uma vez que a presença da parede irá impedir a “água” de inundar o labirinto.

Uma vez que o agente encontra o destino final, o mesmo processo é repetido com posição inicial e objetivo invertidos. Ao fim, serão comparados os caminhos de ida e volta: o menor será considerado a melhor opção.

2.2 Micromouse

A competição Micromouse é um concurso anual na qual estudantes do mundo todo desenvolvem pequenos robôs autônomos, chamados *micromouse*, postos a correr dentro de um labirinto. Dessa forma, o *micromouse* que mais rápido chegar ao seu centro é o vencedor da competição([FOUNDATION, 2019a](#)).

Figura 2.1: Moonlight Special - Primeiro modelo *micromouse* a ganhar uma competição.



Fonte: [Reuben \(2010\)](#)

Sua ideia surge em 1977, quando a *IEEE Spectrum Magazine* trouxe pela primeira vez o conceito de robôs autônomos para resolução de labirintos. Pouco tempo depois, sua primeira competição foi realizada, em junho de 1979, na primeira *IEEE Amazing Micromouse Maze Contest* organizada na cidade de Nova York. Rapidamente, o conceito

da competição se espalhou e, já no começo da década de 90, vários clubes voltados para Micromouse surgiam em escolas e universidades do mundo todo ([DE; HALL, 2004](#)).

Atualmente, a *IEEE Micromouse Competition* adota uma configuração que consiste em um labirinto de 16 x 16 blocos. Cada bloco possui 18 x 18 cm. As paredes, que possuem 5 cm de altura, são pintadas de branco de modo a ser reflexiva à luz infravermelha. O chão, por outro lado, é pintado de preto, para que não seja reflexivo. Além disso, os competidores sabem previamente que o *micromouse* tem seu ponto de partida localizado em um dos cantos do labirintos, devendo alcançar o seu centro para terminar o desafio. Com base nisso, os participantes devem usar de algorítimos de busca para explorar o labirinto e encontrar a rota mais otimizada para o ponto de chegada estabelecido pela competição. O robô por sua vez, não pode ter suas dimensões maiores que uma seção de 25 x 25 cm.

2.3 Robótica e *Frameworks*

A palavra robô foi utilizada pela primeira vez em 1921 em uma peça teatral pelo dramaturgo checoslovaco Karel Capek. A origem vem da palavra checa robota que significa “trabalho forçado” e tornou-se popular através de filmes como Metrópolis (1926). O dia em que a terra parou (1951) e Planeta proibido (1956). Apesar da inserção do termo robô ser datada de 1921, registros históricos da antiga civilização grega já propunha o desenvolvimento dos primeiros modelos de robôs. Esses tinham aspecto visual semelhante ao de um humano e ou animal, e utilizavam sistemas de pesos e bombas pneumáticas, porém não tinham nenhuma funcionalidade prática, social, e como a época ainda não existia sistemas de produção complexos, esse robôs também não tinham função voltada à produtividade. Era um mecanismo de simples movimentação e sem nenhuma finalidade real. Algum tempo depois, cientistas árabes se dedicaram em fomentar a idéia e pesquisar sobre atribuir possíveis funções a esses robôs, com objetivo de facilitar as necessidades humanas. Esse pensamento revolucionário foi um grande marco na história da robótica. Segundo (PAPERT, 2008) citado por [Mahmud \(2017\)](#), documentos históricos datados de 1495 revelam que Leonardo Da Vinci ao desenvolver uma grande investigação sobre a anatomia humana, permitiu o desenvolvimento de exemplares de bonecos que obtinham articulações mecânicas, sendo capazes de mover as mãos, cabeça, olhos e pernas, conseguiam até realizar algumas ações mais simples, tal como, escrever ou tocar alguns instrumentos.

O desenvolvimento produtivo em larga escala dos robôs aconteceu no início do século XVIII com a ascensão da revolução industrial. Segundo (SANTOS; MENEZES, 2005) citado por [Mahmud \(2017\)](#), a indústria têxtil utilizava teares mecânicos e com o contínuo avanço da revolução industrial, as fábricas começaram a substituir algumas tarefas me-

canizadas por máquinas capazes de reproduzir automaticamente estas ações. Os anos 30 foram marcados no campo da robótica pela produção de um robô humanoide, denominado de Elektro, produzido pela empresa Westinghouse Electric Corporation. Elektro foi apresentado em 1939 e 1940 na World's Fair, uma feira de exposição internacional de produtos manufaturados. Segundo (SILVA, 2009) citado por [Mahmud \(2017\)](#), o termo “robótica” foi enunciado pela primeira vez em 1942 pelo cientista Isaac Asimov, em uma pequena história chamada “Runaround”. Asimov também publicou uma compilação de histórias em 1950 chamada “I Robot”, na qual ele propunha a existência de 3 leis fundamentais aplicadas à robótica, e, posteriormente ele adicionou mais uma, a lei zero. Essas leis são:

1. Um robô não pode ferir um ser humano ou, por omissão, permitir que um ser humano sofra algum mal;
2. Um robô deve obedecer as ordens que lhe sejam dadas por seres humanos, exceto nos casos em que tais ordens contrariem a primeira lei;
3. Um robô deve proteger sua própria existência desde que tal proteção não entre em conflito com a primeira e segunda lei.

zero. Um robô não pode fazer mal à humanidade e nem por inação, permitir que ela sofra algum mal.

Porém, atualmente estas leis são interpretadas de uma perspectiva puramente ficcional, afinal, no tempo em que foram formuladas, não se tinha dimensão do desenvolvimento vertiginoso que iria ocorrer na área da robótica.

Apesar de que robôs industriais não se assemelham ao físico humano, ainda assim eles os substituem ao desempenhar tarefas nocivas para as pessoas. O conceito de robô industrial foi patenteado por George Devol, em 1954. O avanço tecnológico permitiu que os robôs passassem a desempenhar funções muito mais complexas e em várias esferas profissionais, por exemplo, indústria automotiva, nuclear, aplicação espacial, aplicação na medicina, em ambientes adversos, onde a presença humana se torna difícil e até mesmo impossível.

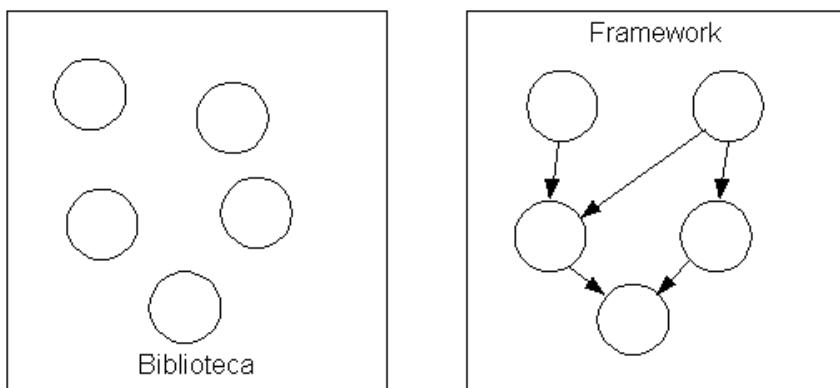
Se tratando de avanço tecnológico, o campo de engenharia de software está ligado diretamente com o desenvolvimento dos robôs atualmente e umas das principais ferramentas de auxílio na programação dos mais diversos tipos de robôs, é o *framework*. Um *framework* pode ser conceituado de algumas formas.

Segundo Fayad et al. (1999) e Johnson & Foote (1988), citado por [Junior \(2006\)](#), “um *framework* é um conjunto de classes que constitui um projeto abstrato para a solução de uma família de problemas.”

Para Mattsson (1992, 2000), citado por [Junior \(2006\)](#) “um *framework* é uma arquitetura desenvolvida como objetivo de atingir máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização.” Já para Buschmann et al. (1996), Pree (1995) e Pinto (2000), citado por [Junior \(2006\)](#) “um *framework* é definido como um software parcialmente completo projetado para ser instanciado.”

Apesar de diferentes as definições encontradas nas literaturas elas não são contraditórias. A ideia de reutilização de softwares é fundamental para a definição de *framework*, afinal uma ferramenta propõe a execução de uma tarefa de maneira mais prática e rápida. Uma comparação deixando claro as diferenças entre um *framework* e uma biblioteca de classe, pode permitir um melhor entendimento sobre o tema. Numa biblioteca de classes, cada classe é única e independente das outras. Já em um *framework*, as dependências/colaborações estão embutidas, conforme ilustrado na Figura 2.2.

Figura 2.2: Biblioteca x *Framework*.



Fonte: [Sauvé \(2001\)](#)

O ROS é um exemplo de *framework*. Ele é uma plataforma de software de código aberto, desenvolvido em 2007, projetado para suportar uma nova geração de robôs. A comunidade ROS desenvolveu o *framework* para várias plataformas, disponibilizando-os assim para diferentes Sistemas Operacionais (SOs), como Windows, Linux e Mac OS. O ROS é um conjunto de ferramentas que gerencia de forma eficiente a mediação entre o SO e demais aplicações, possui uma estrutura flexível permitindo a criação de software de robôs. Detém uma coleção de bibliotecas e convenções com objetivo de simplificar tarefas complexas e criar estruturas mais robustas em uma ampla gama de plataformas robóticas. Pode-se afirmar que o ROS representa uma coleção muito útil de softwares voltados exatamente para ajudar na percepção, controle e modelagem dos dispositivos de hardware que serão lidos, fornecendo serviços que são esperados de um SO, incluindo a abstração de hardware em baixo nível, o que permite a leitura e o controle do mesmo, passagem de mensagens entre processos e gerenciamento de pacotes (Quigley et al. 2009, apud [Vargas e Tavares](#)

(2013)). O ROS também disponibiliza ferramentas de aplicação de robótica nas áreas de navegação, simulação, visão, percepção e controle, sendo exemplo as bibliotecas de processamento de imagem (opencv, pcl) e simuladores (stage, gazebo).

Por ser uma plataforma com código aberto, o ROS possui uma ampla documentação, bem detalhada, em sua maioria na língua inglesa. Na página da comunidade ROS, existe um extenso material de apoio, oferecendo suporte a compartilhamento de códigos fonte e experiências realizadas com o *framework*. A comunidade do ROS permite que o usuário faça parte dela, mediante a um cadastro pessoal no site. Dessa forma, o usuário pode deixar eventuais perguntas e/ou dúvidas que poderão ajudá-lo a solucionar algum problema.

O framework ROS foi projetado para atender a um conjunto específico de desafios encontrados durante o desenvolvimento de robôs. O ROS é muito mais do que apenas um serviço oferecido a robôs móveis e de manipulação (Alexander et al. 2012, apud Vargas e Tavares (2013)).

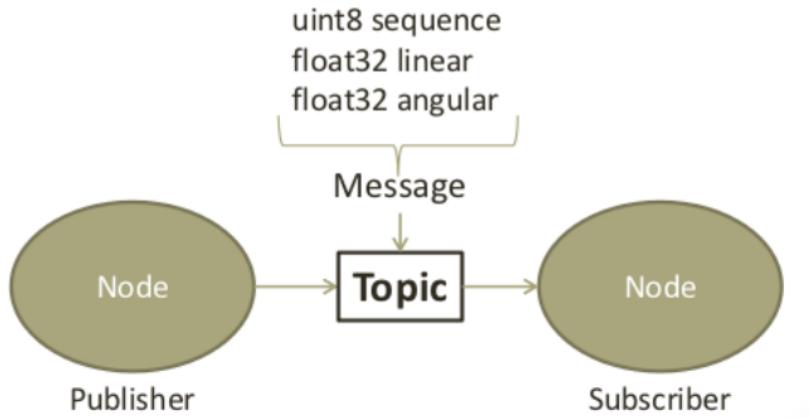
O ROS possui uma estrutura distribuída de processos, a qual apoia a reutilização de código em robótica. Suas bibliotecas possuem funcionalidade para diversas linguagens de programação, foi projetado com um idioma neutro, suportando C, C++, Python, LISP e Octave, apesar de que C++ e Python são as duas linguagens principais e com maior número de pacotes disponíveis. Além disso, O ROS organiza seu conjunto de software em *packages* ou pacotes em tradução livre. De acordo com o guia de utilização do ROS,

[...]um pacote pode conter nós ROS, uma biblioteca independente de ROS, um conjunto de dados, arquivos de configuração, um software de terceiros ou qualquer outra coisa que constitua logicamente um módulo útil. O objetivo desses pacotes é fornecer essa funcionalidade útil de maneira fácil de consumir, para que o software possa ser facilmente reutilizado. Em geral, os pacotes ROS seguem o princípio "Goldilocks": funcionalidade suficiente para ser útil, mas não muito que o pacote seja pesado e difícil de usar em outro software. (SAITO, 2019, tradução nossa)

A comunicação do ROS é baseada em eventos, utiliza processos para publicar/subscrever nós a tópicos. Conceitualiza-se que os nós publicam e subscrevem os tópicos, permitindo assim uma comunicação entre nós, onde recebem mensagens, apenas os que estiverem subscritos ao tópico.

A Figura 2.3 demonstra uma simples publicação e subscrição de dois nós a um tópico. Desta forma sempre que o nó publicar novas mensagens os nós que estejam subscritos ao tópico irão receber a informação.

Figura 2.3: Estrutura de comunicação.

Fonte: [Santos \(2013\)](#)

Os nós correspondem à execução de um programa e são criados no sistema, geralmente, via console e sempre com o servidor mestre ROS (roscore) iniciado. O servidor roscore é responsável por inicializar várias dependências, bibliotecas do ROS e o sistema de comunicação entre nós. Devida as vantagens descritas neste tópico, será utilizado o *framework* ROS para o desenvolvimento do Doogie Mouse.

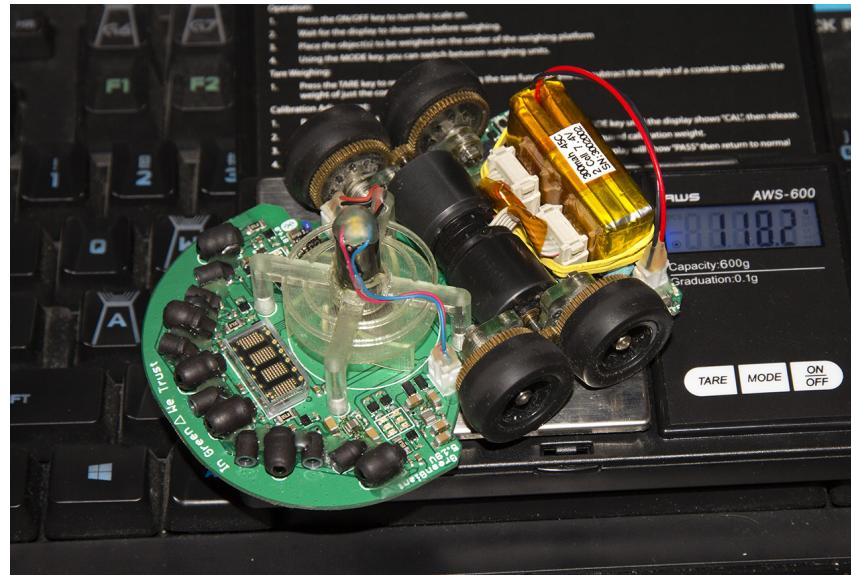
2.4 Estudo do Estado da Arte

Nessa seção, será realizado um estudo comparativo de alguns modelos de *micromouse*, buscando avaliá-los sobre diferentes critérios a partir de uma matriz de comparação. O estudo auxiliou na realização do design da paltaforma proposta por esse trabalho uma vez que, através dele, verifica-se quais recursos são mais importantes para o comprimento dos requisitos do projeto, além de permitir um entendimento generalizado de como são feitos os robôs *micromouse*.

2.4.1 GreenGiant

A Green Giant é uma desenvolvedora de múltiplas plataformas de robótica, especializada em eletrônica embarcada, tendo como seu carro-chefe o *micromouse*. Seu modelo mais recente, 2016 - 2017, é voltado para alto desempenho em competições, alcançando a posição de quarto lugar durante a *Applied Power Electronics Conference* (APEC) de 2016.

Figura 2.4: Robô Green Giant.

Fonte: [Luzhou \(2016\)](#)

Sua interface de usuário possui *Dot Matrix Display*, sinalizadores LED, botões, buzzer, além de possuir um sistema de comunicação Bluetooth 4.0. Ademais, o modelo usa um sistema de ventoinhas de sucção para aumentar o nível de aderência das rodas, permitindo alcançar maiores velocidades sem derrapar.

Tabela 2.1: Atributos do robô Green Giant

| Green Giant 5.19V | |
|--------------------------|---------------------------------------|
| Fabricante | Green Ye |
| Ano | 2017 |
| Linguagem de Programação | C/C++ |
| Sensores | IR, MPU, IE |
| Controlador | STM32 |
| Simulador | - |
| Bateria | LiPo 300mAh (7,4V) |
| Rodas | 3D printed mount&wheel + mini-z tyres |
| Motor | DC-Motor 6.540RPM 0,21N.m (6V) |
| User Interface | DMD 5x7, LEDs, Botão, Bluetooth |
| Outros | sistema de ventoinhas de sucção |

Fonte: Adaptado de [Luzhou \(2016\)](#)

Pontos Positivos:

- Produto de alto desempenho em competições;
- Sistema de ventoinhas de sucção.

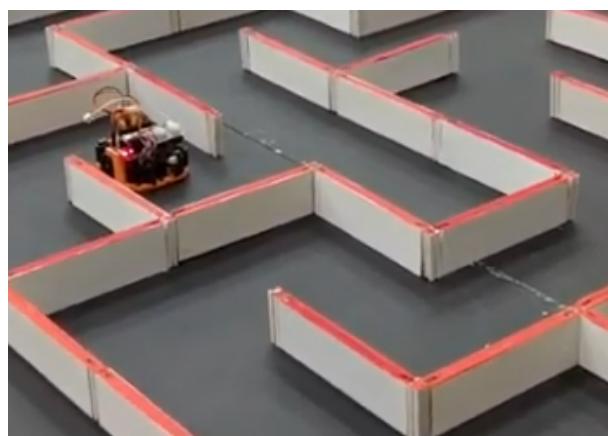
Pontos Negativos:

- Não possui suporte à simulação;
- Projeto pouco documentado;
- Não possui guia do usuário;
- Não possui suporte nativo para ambiente ROS.

2.4.2 WPISmartMouse

A organização estudantil, WPI CollabLab, compartilham um espaço de laboratório entre seus membros para projetos com vies colaborativo a sociedade. Nesse espaço desenvolveu-se o Smartmouse, projeto *micromouse* voltado para a competição Micromouse *Brown IEEE Robotic Olympiad*. O projeto também se estendeu para o desenvolvimento de um ambiente de simulação apartir dos projetos Gazebo e Ignition, não possuindo entretanto suporte para ROS.

Figura 2.5: Robô WPISmartMouse



Fonte: [WPISmartmouse \(2016\)](#)

Tabela 2.2: Atributos do robô WPISmartmouse

| Smartmouse | |
|--------------------------|----------------------------|
| Fabricante | WPI CollabLab |
| Ano | 2018 |
| Linguagem de Programação | Arduino/C, Python, BASCOM |
| Sensores | IR, Magnetic Encoder |
| Controlador | Teensy 3.6 |
| Simulador | Gazebo |
| Bateria | LiPo 1500mAh (7,4V) |
| Rodas | Solarbotics RW2i Wheel |
| Motor | DC-Motor 650RPM 2,35Nm(6V) |
| User Interface | LEDs |
| Outros | documentação no git |

Fonte: Adaptado de [WPISmartmouse \(2016\)](#)

Pontos Positivos:

- Provê ambiente de simulação;
- Documentação disponível no GitHub;
- Possui portabilidade para mais de uma linguagem de programação.

Pontos Negativos:

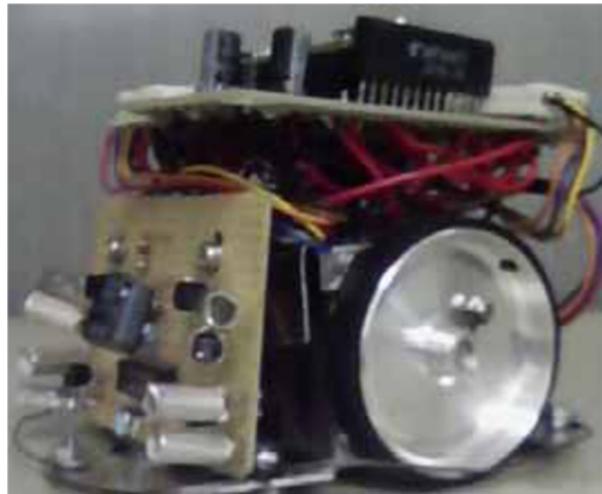
- Não possui suporte nativo para ambiente ROS;
- Pouca variedade de sensores;
- Não possui guia do usuário;
- Poucos recursos na interface com o usuário.

2.4.3 Kumamoto National College

O Instituto Nacional de Tecnologia de Kumamoto, *Kumamoto National College*, apresentou no ano de 2008 um projeto de desenvolvimento de ferramentas educacionais voltada para integração de sistemas e suas implementações. O projeto é direcionado aos seus

estudantes do 5º ano de engenharia, através da produção de um *micromouse* para a competição do ramo de *Kyushu*.

Figura 2.6: Robô Kumamoto.



Fonte: [Hayama e Matsumoto \(2010\)](#)

O hardware do robô foi bastante simplificado, visando facilitar o desenvolvimento pelos estudantes ainda não familiarizados com a robótica e eletrônica, além de buscar reduzir os custos de produção do robô. Como ferramenta educativa, o projeto conseguiu que seus estudantes produzissem o *micromouse* em um semestre de atividades. Contudo, conceitos da robótica (ex: robótica móvel, fusão de sensores, visão, navegação) não foram trabalhados ou não foram citados no artigo gerado a partir do projeto.

Tabela 2.3: Atributos do robô Kumamoto

| Kumamoto National College | |
|----------------------------------|---------------------------------------|
| Fabricante | Kiyoteru Hayama and Tsutomu Matsumoto |
| Ano | 2008 |
| Linguagem | C |
| Sensores | IR |
| Controlador | H8Tiny-3664 |
| Simulador | - |
| Bateria | LiPo 900mAh (7,4V) |
| Rodas | wheels, tires |
| Motor | Step Motor 0,78Nm (5,6V) |
| User Interface | LEDs |
| Outros | Documentação em artigo |

Fonte: Adaptado de [Hayama e Matsumoto \(2010\)](#)

Pontos Positivos:

- Projeto com fins educacionais;
- Fácil desenvolvimento.

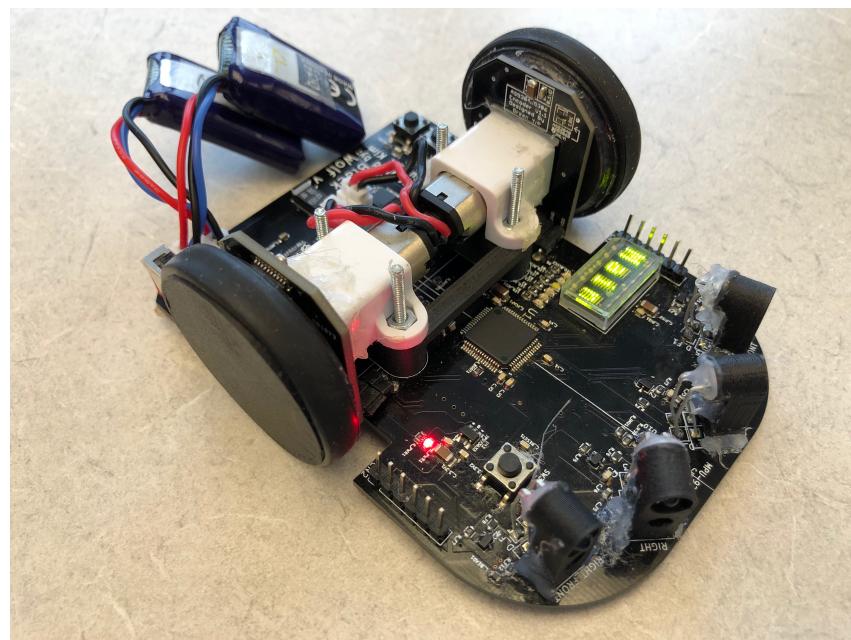
Pontos Negativos:

- Não possui suporte nativo para ambiente ROS;
- Pouca variedade de sensores;
- Não possui guia para usuário;
- Poucos recursos na interface com o usuário;
- Não possui nenhum ambiente de simulação.

2.4.4 WolfieMouse

O WolfieMouse é um projeto de robótica que desenvolveu um *micromouse* para competir na *2018 Region 1 Robotics Competition*.

Figura 2.7: Robô WolfieMouse.



Fonte: [Bryant, Bum e Choi \(2018\)](#)

Além da plataforma robótica, que conta com hardwares programados em baixo nível, para melhor otimização de seus controles, a equipe também realizou um ambiente de simulação baseado em C++ emulado no terminal do computador. Toda documentação foi disponibilizada em um repositório git, que também possui tutoriais para o desenvolvimento do robô.

Tabela 2.4: Atributos do robô WolfieMouse

| WolfieMouse | |
|--------------------|------------------------------------|
| Fabricante | Bryant Gonzaga, Bum Kim, Hyun Choi |
| Ano | 2018 |
| Linguagem | C++, C, ARM Assembly, Python |
| Sensores | MPU, ToF, Magnetic Encoder |
| Controlador | STM32 |
| Simulador | Terminal-based |
| Bateria | * |
| Rodas | * |
| Motor | DC-Motor |
| User Interface | DMD 5x7, botões |
| Outros | documentação e tutoriais no git |

Fonte: Adaptado de [Bryant, Bum e Choi \(2018\)](#)

Pontos Positivos:

- Projeto bem documentado;
- Possui tutoriais;
- Possui ambiente de simulação.

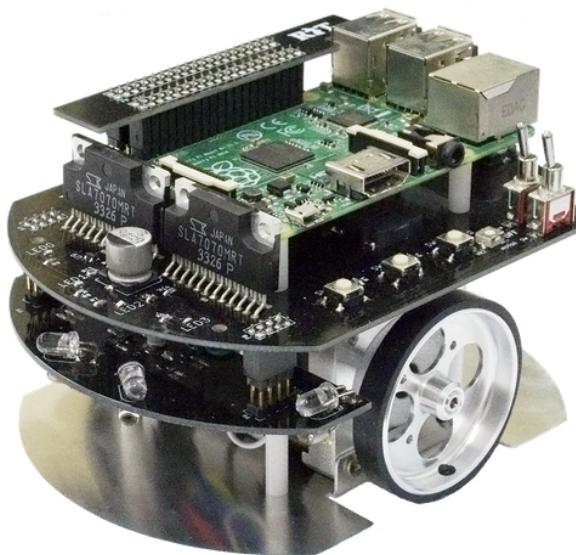
Pontos Negativos:

- Não possui suporte nativo para ambiente ROS;
- Pouco foco em finalidades educativas com o produto;
- Não possui guia para usuário;
- Poucos recursos na interface com o usuário.

2.4.5 Raspberry Pi Mouse V2

A RT Corporation Micromouse é uma desenvolvedora japonesa de plataformas robóticas voltada para aplicações voltadas de pesquisas à hobistas. Um de seus segmentos é voltado para *micromouse*, fortemente representado pelo seu produto Raspberry Pi Mouse V2, citado em "Learning ROS robot programming with Raspberry Pi" (Nikkei BP, June 2018).

Figura 2.8: Robô RaspiMouse.



Fonte: [RTCorporation \(2016\)](#)

O modelo citado, é um robô de plataforma baseado em *micromouse* que utiliza uma Raspberry Pi como sua placa principal. Dessa forma o robô pode ser controlado pelos principais *middleware* de robótica (ROS/RTM), possuindo inclusive pacotes publicados na wiki do ROS voltados para navegação e simulação do *micromouse*;

Pontos Positivos:

- Projeto bem documentado;
- Disponível no GitHub;
- Possui tutoriais;
- Possui ambiente de simulação;
- Suporte aos principais middleware de robótica (ROS/RTM);
- Possui pacotes do ROS para seu controle;

Tabela 2.5: Atributos do robô RaspiMouse

| Raspberry Pi Mouse V2 | |
|------------------------------|-------------------------------------|
| Fabricante | RT Corporation |
| Ano | 2016 |
| Linguagem | Python, Shell |
| Sensores | IR |
| Controlador | RaspberryPi3 |
| Simulador | Gazebo |
| Bateria | LiPo 1000mAh (7,4V) |
| Rodas | wheels, tires |
| Motor | Step-Motor 400step/rev (4 fases) |
| User Interface | Terminal, LEDs, Botão, Buzzer |
| Outros | documentação no Git, mas em japonês |

Fonte: Adaptado de [RTCorporation \(2016\)](#)

- Plataforma é expansível.

Pontos Negativos:

- Toda documentação do produto está em japonês;
- O robô é pouco compacto.

2.4.6 Matriz de Comparação

Através do estudo conforme tópico anteriores, montou-se uma matriz de comparação de forma a quantificar atributos considerados mais significativos para o robô. Levou- se em conta, portanto, a existência da documentação disponível e seu nível de clareza; o uso de algum *framework* de robótica; se faz uso ou suporta algum ambiente de simulação; diversidade de linguagens que a plataforma pode ser programada; como é realizada a interface do usuário; quantidade de diferentes sensores e se a plataforma é expansível, podendo acrescentar a ela outros recursos (seja em hardware ou em software). Essa matriz pode ser visualizada no Apêndice A deste documento.

Materiais e Métodos

O metodologia empregada nos trabalhos de conclusão de curso do Centro Universitário SENAI CIMATEC é executado com base na metodologia TheoPrax que foi desenvolvida pelo instituto Fraunhofer de Tecnologia Química, situado na Alemanha. A sistemática TheoPrax tem como principal objetivo incrementar a motivação da aprendizagem através do desenvolvimento de projetos reais voltados para empresas, proporcionando a integração entre o conhecimento técnico e sua aplicação prática. Para isto, esta estrutura envolve a identificação de uma situação problema ou de uma melhoria no processo ou no produto da empresa, seu estudo e, por fim, a definição de uma proposta técnica-financeira para implementação da solução.

3.1 Metodologia

A utilização da metodologia TheoPrax se restringe apenas ao gerenciamento macro do projeto e não define como a solução proposta deve ser desenvolvida. Sendo assim, o desenvolvimento do projeto, proposto no tópico 1.1, foi realizado utilizando o procedimento ilustrado na figura 3.1 que foi adaptado da metodologia empregada no *Brazilian Institute of Robotics* (BIR) para desenvolvimento de projetos de robótica.

Figura 3.1: Metodologia empregada no desenvolvimento do projeto solução.



Fonte: Própria Autoria

Conforme Figura 3.1, a metodologia utilizada neste projeto possui 4 etapas: Conceitual, Design, Desenvolvimento e Conclusão. Por conseguinte, cada etapa possui entradas e saídas, que vão se complementando ao longo do desenvolvimento do projeto, as quais serão explicadas nos tópicos seguintes.

3.1.1 Conceitual

A primeira etapa, designada como **Conceitual**, embora não explicitada no diagrama, possui como entradas as informações provenientes do cliente. Essas informações, tais como o problema proposto e os seus requisitos são utilizadas para o entendimento do projeto, servindo como ponto de partida para formulação da proposta de solução. Diante dessas informações, é possível definir os requisitos técnicos com base nos desejos do cliente (requisitos do cliente); a base do design, que consiste na definição do escopo e o que será necessário para desenvolver o projeto: meios, padrões e os principais componentes (hardware e software); e a arquitetura geral, que fornece uma visão macro de como será a interação entre o hardware e o software do robô.

Após o entendimento do projeto, passa-se para a criação de ideias. Nesta subetapa, algumas pesquisas são realizadas para ajudar no processo de criatividade e evitar a reimplementação do que já existe no mercado. Assim, utiliza-se o *State Of The Art* (SOTA), documento que aponta as principais pesquisas e estudos sobre o tema do projeto, referenciando pesquisas acadêmicas já realizadas; e o *Benchmarking*, que é uma relação oriunda do mercado na qual aponta os competidores para o sistema projetado, incluindo para cada competidor critérios de avaliações importantes para o projeto. Finalizando a subetapa de criação de ideias, tem-se a Matriz *Quality Functional Deployment* (QFD), em que os requisitos do cliente são confrontados com os requisitos técnicos, fornecendo à equipe de desenvolvimento do projeto os principais pontos que deverão receber maior atenção durante a elaboração do projeto.

Por fim, após o entendimento do projeto e a formulação da ideia, parte-se para a etapa de seleção dos principais componentes do sistema, em que primeiro elabora-se o *Prototype Breakdown Structure* (PBS), uma representação do projeto com uma visão de subsistemas, apresentado através de um fluxo estruturado.

3.1.2 Design

Com o conceito pré-estabelecido do sistema que será desenvolvido, parte-se para a etapa de **Design**. Nesta fase, define-se o sistema de maneira mais clara, tendo a especificação funcional como principal elemento. Este documento comprehende a explicação detalhada de cada funcionalidade do robô, contendo a definição, o objetivo, as premissas e as suas entradas e saídas. Devido ao nível de detalhes da especificação funcional, revisões em documentos anteriores, principalmente a arquitetura geral, são realizadas durante esta fase.

No final da etapa de Design, começa-se o planejamento para o desenvolvimento técnico do projeto. Durante o planejamento é elaborado a arquitetura elétrica do robô (uma visão mais detalhada da arquitetura geral), descrevendo as formas de conexão e os protocolos de comunicação entre os elementos que compõem o sistema; os equemáticos eletrônicos, os quais são utilizados para confecção das Placas de Circuito Impresso (PCIs) que comporão o sistema; a arquitetura mecânica, apresentando os elementos mecânicos do robô; e por fim, os desenhos técnicos mecânicos, utilizados posteriormente para fabricação das peças do protótipo.

Em conclusão, esta fase é de extrema importância, pois possibilita a geração de documentos que podem ser utilizados para a replicação do projeto. Além do mais, permite mais fluidez no desenvolvimento técnico do robô.

3.1.3 Desenvolvimento

Após a finalização do planejamento, começa-se a etapa de **Desenvolvimento** do projeto, em que o conceito e as ideias provenientes das etapas anteriores tornam-se concretas. Nesta fase são desenvolvidos e documentados os pacotes de software, os quais englobam tanto a unidade de controle do robô quanto os *drivers* dos sensores, atuadores e elementos de interação com usuário. Muito dos pacotes aqui desenvolvidos, usam a especificação funcional como guia.

Para auxiliar no processo de desenvolvimento, um ambiente de simulação torna-se elemento vital. O ambiente de simulação permite que ideias sejam propostas e testadas sem a necessidade do uso da plataforma física, acelerando o processo de desenvolvimento num âmbito em que se possui diversas pessoas trabalhando no mesmo sistema. Não menos importante, os simuladores evitam que danos sejam causados ao robô em caso de má implementação de algum algoritmo.

Entretanto, os ambientes de simulação não refletem completamente os aspectos físicos do mundo real. Diante disso, na fase de desenvolvimento torna-se necessário também a confecção de um ambiente real para testes. Com essa estrutura, também chamada de *Mockup*, pode-se realizar testes para validar o que foi desenvolvido e produzir relatórios que podem ser entregues ao cliente como forma de acompanhamento do desenvolvimento técnico do projeto.

3.1.4 Conclusão

O projeto é finalizado na etapa de **Conclusão**, em que a solução proposta é entregue ao cliente. Nesta entrega é realizado a demonstração do funcionamento do robô, utilizando um ambiente real. Compõe também a entrega documental um documento em formato de Guia do Usuário contendo as instruções para manipulação e replicação do protótipo desenvolvido.

3.2 Requisitos do projeto

Como mencionado no início deste capítulo, uma das etapas da metodologia TheoPrax envolve a identificação de uma situação problema. No projeto em questão, foi solicitado ao cliente os requisitos que o Doogie Mouse deveria cumprir. Entretanto, tais requisitos refletem a vontade do cliente de forma não técnica. Assim, a partir dos requisitos do cliente, foram levantados por parte da equipe de desenvolvimento os requisitos técnicos, os quais fornecem objetivos específicos que o projeto deve atingir. Por consequência, o cumprimento dos requisitos técnicos decorre na aprovação dos requisitos do cliente. Abaixo estão listados os requisitos do cliente e técnicos do Doogie Mouse de forma hierárquica, isto é, para cada requisito do cliente estão listados seus respectivos requisitos técnicos associados.

1. Documentação de fácil acesso e entendimento:
 - (a) Disponibilização de todo código desenvolvido no GitHub;
 - (b) Disponibilização guia do usuário como Wiki no GitHub;
 - (c) Disponibilização em um repositório no GitHub esquemáticos eletrônicos, desenhos técnicos mecânicos e seus respectivos arquivos para possível edição.
2. Uso de um sistema microprocessado comercial:
 - (a) Utilização de uma Raspberry Pi como sistema microprocessado;
3. Interface intuitiva:
 - (a) Permissão ao usuário acesso remoto do terminal do sistema operacional do robô;
 - (b) Visualização do mapa do Labirinto no RViz;
 - (c) Disponibilização de botões e display para interação com usuário na ausência de acesso remoto.
4. Tutoriais na internet para entendimento do funcionamento do robô:

- (a) Desenvolvimento de tutorial na ROS Wiki com os primeiros passos com robô, explicando ao usuário comandos básicos de locomoção;
 - (b) Desenvolvimento de tutorial na ROS Wiki explicando ao usuário como implementar no robô seu próprio algoritmo de resolução de labirinto.
5. Autonomia suficiente para utilização em ao menos duas aulas consecutivas:
- (a) Bateria recarregável;
 - (b) Autonomia superior a 1h40min.
6. Uso de componentes de fácil manipulação e com facilidade de aquisição:
- (a) Uso de conectores polarizados;
 - (b) Utilização de padrão de cores para cabos de conexão;
 - (c) Especificação de componentes disponíveis no mercado nacional;
 - (d) Desenvolvimento de shield de interface entre a plataforma de processamento e os sensores e atuadores.
7. Estrutura física compatível com as regras estabelecidas em competições do IEEE:
- (a) Dimensões do robô devem ser menores que 15 x 15 x 10 cm.

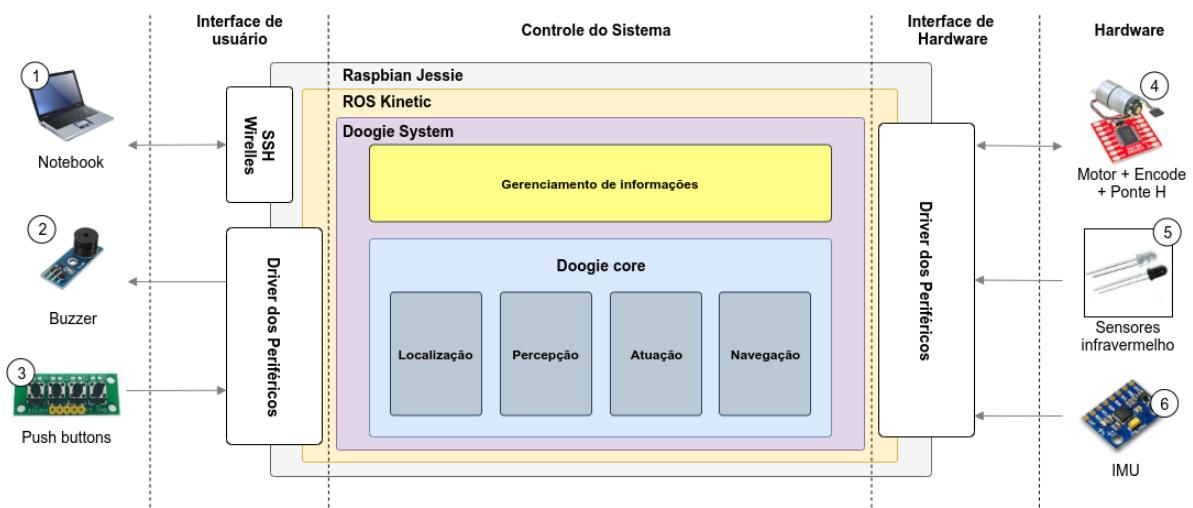
Por fim, para a proporcionar a equipe de desenvolvimento um guia em relação a aplicação dos esforços para que os requisitos do cliente e técnico fossem atingidos, foi realizado uma confrontação de tais requisitos. Esta análise foi derivada de uma ferramenta comumente utilizada em desenvolvimento de produto, denominada matriz QFD. A confrontação dos requisitos do projeto proposto pode ser visualizada no Apêndice B. Observa-se na linha 5 da matriz que o requisito técnico ”Permissão ao usuário de acesso remoto do terminal do sistema operacional do robô” possui apenas uma relação forte com o requisito do cliente ”Interface intuitiva”. Por outro lado, ”Dimensões do robô devem ser menores que 15 x 15 x 10 cm”, na linha 16, possui 2 relações fracas, uma média e uma forte com diferentes requisitos do cliente. Portanto, essas associações funcionam como direcionadores de esforços e prioridades das tarefas desenvolvidas ao longo do projeto.

3.3 Descrição do sistema

O Doogie é um robô autônomo capaz de mapear um labirinto e descobrir qual o menor caminho do seu ponto de partida até um ponto de chegada, utilizando algoritmos de busca, como por exemplo o *flood fill*. Sua arquitetura, ilustrada na Figura 3.2, pode ser dividida em três partes: interação com o usuário, controle do sistema e interface de hardware. Para a interação com o usuário, o robô possuirá um Buzzer e dois *Push Buttons*. Além disso,

há a possibilidade de acessar o sistema do robô via *Secure Shell* (SSH), por intermédio da conexão WiFi. A interface de hardware é composta por motores de corrente contínua, responsáveis pela movimentação, trabalhando em conjunto com uma ponte H e encoders; sensores infravermelhos, dispostos na frente e dos lados, a fim de identificar paredes; e uma *Inertial Measurement Unit* (IMU), responsável por fornecer a aceleração linear e a velocidade angular da plataforma móvel para complementar os dados de Odometria. O controle do sistema é embarcado dentro de uma Raspberry Pi Zero que utiliza o *framework* ROS para gerenciar os diversos subsistemas do micromouse.

Figura 3.2: Arquitetura Geral do robô Doogie.



Fonte: Própria Autoria

3.3.1 Interface de usuário

O acesso à Raspberry Pi Zero é feito via SSH permitindo maior acessibilidade e segurança nos dados. O mesmo é feito remotamente através de conexão *wireless*, possibilitando o acesso a linha de comandos da Raspberry bem como seu sistema de arquivos, de um outro computador.

O robô tem uma interface de interação com o usuário através de dois *Push Buttons* e um *Buzzer*. Os botões são utilizados para execução de tarefas tais como iniciar e parar a execução da rotina principal do robô. O acesso a tais dispositivos é feito através do driver do periférico *General Purpose Input/Output* (GPIO) que oferece *Application Programming Interface* (API) nas linguagens de programação Python e C++.

3.3.2 Interface de Hardware

O deslocamento do robô utiliza 2 motores de corrente contínua, acoplados à encoders, possibilitando a obtenção de informações de posição e velocidade, com objetivo de otimizar o controle e acionamento dos motores. Além disso, para permitir que os motores girem em ambas as direções, são utilizados circuitos de ponte H.

Os sensores infravermelho são responsáveis por identificar obstáculos no trajeto do robô. Eles são dispostos de modo que o micromouse possa indentificar as paredes do labirinto à sua direita, esquerda e frente. Também é necessário auxílio da IMU para obtenção de dados através dos sensores acelerômetro e giroscópio para estimar a posição com maior precisão. De forma similar a interface de usuário, serão utilizados drivers para estabelecer a comunicação entre o ambiente ROS e o hardware acima descrito.

3.3.3 Controle do sistema

A Raspberry Pi Zero é um mini computador de baixo custo e fácil acesso, capaz de reunir diversas funcionalidades em uma única placa de tamanho reduzido. No mesmo, está instalado o sistema operacional Raspbian Jessie, possibilitando a utilização do *framework* ROS, responsável por gerenciar os subsistemas do robô.

Os principais subsistemas desenvolvidos para o Doogie foram:

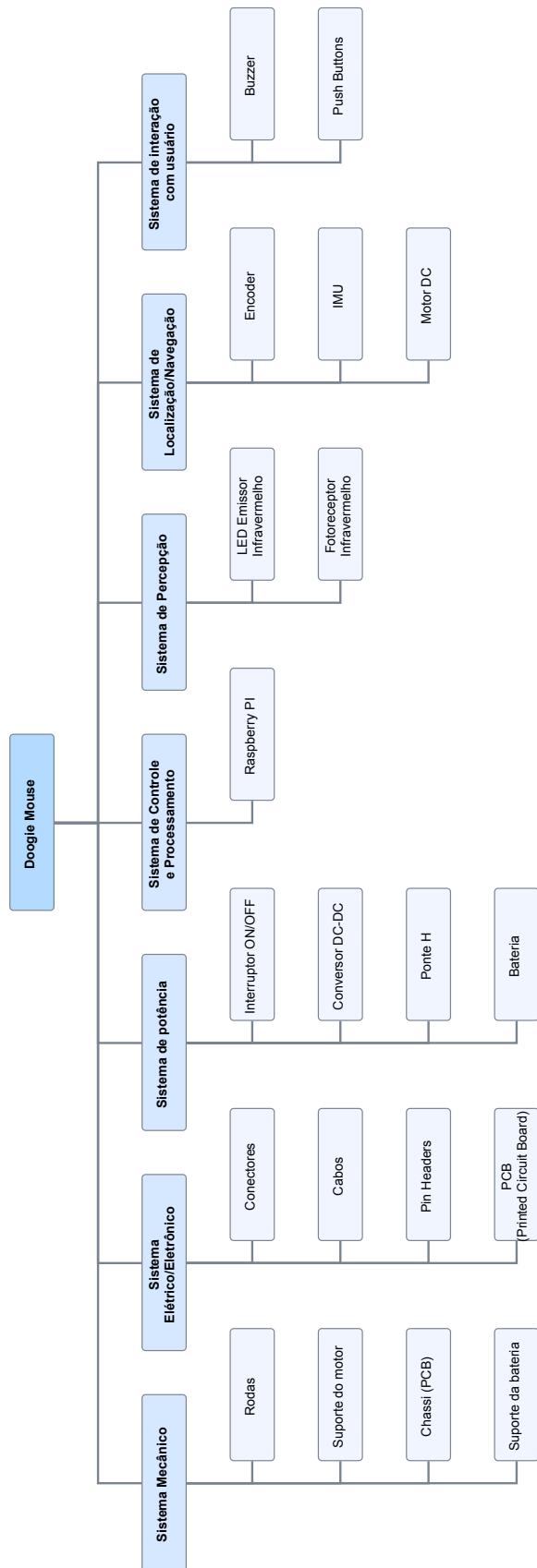
- **Localização:** o labirinto pode ser modelado em uma matriz, geralmente de tamanho 16x16. Esse subsistema tem o objetivo de prover informações sobre em qual posição dessa matriz o robô se encontra (linha e coluna);
- **Percepção:** com a utilização dos sensores infravermelho, o robô pode identificar se há obstáculos ao seu redor. Esse subsistema é responsável por informar se há obstáculos na frente, direita e/ou esquerda do mesmo;
- **Planejamento:** há diversas formas de mapear e completar o labirinto. Utilizando as informações obtidas dos outros subsistemas (localização e percepção), é possível realizar o algoritmo especificado e tomar as decisões necessárias para a realização da atividade;
- **Navegação:** subsistema responsável por gerenciar as ações de navegação do robô, tais como: ir para frente, virar para direita ou esquerda, parar, dentre outros.
- **Gerenciamento de Informações:** é o subsistema responsável por receber comandos através dos botões situados no chassi do robô e fornecer o mapa do labirinto a medida em que ele é percorrido.

3.4 Especificação dos componentes

O robô Doogie é composto de um sistema microprocessado compatível com distribuições Linux; sensores de percepção que permitem a detecção de obstáculos ao redor do robô; sensores de posição para o controle dos movimentos e localização dentro do labirinto; atuadores para permitir a locomoção de forma autônoma. Além disso, ele é equipado com bateria recarregável, conversor analógico/digital, multiplexador analógico, botões, LEDs e Buzzer. Os tópicos seguintes irão detalhar os principais componentes da plataforma móvel.

3.4.1 Estrutura analítica do protótipo

Para fornecer uma visão macro dos componentes que compõe o robô, foi elaborado um diagrama hierárquico, denominado PBS. Conforme Figura 3.3, é possível visualizar a dependência entre os subsistemas e os componentes do robô. Além disso, diferente da Figura 3.2, esse diagrama oferece maior detalhamento dos elementos que estão relacionados a cada subsistema do robô.

Figura 3.3: *Prototype Breakdown Structure* do Doogie Mouse

Fonte: Própria Autoria

3.4.2 Raspberry Pi Zero W v1.1

Para que o robô realize a resolução do labirinto, é necessário que ele contenha uma unidade de processamento central. Este dispositivo tem como objetivo, a partir dos dados dos sensores e de um algoritmo de resolução de labirintos, inferir qual o melhor conjunto de movimentos que permita o robô alcançar o ponto de chegada do labirinto no menor tempo possível. Como o modelo de robô Micromouse possui um baixo nível de abstração de *hardware*, é necessário também que a unidade de processamento disponibilize periféricos necessários para a comunicação com os componentes do sistema. Não menos importante, a unidade deve possuir compatibilidade com o *framework* ROS, *middleware* escolhido para o gerenciamento dos subsistemas que compõem o robô.

Portanto, foi escolhido o mini computador Raspberry PI Zero W v1.1: uma placa de baixo custo e com tamanho reduzido (apenas 6,5 x 3 cm), ótimo para projetos compactos como o robô Doogie. Esta placa é equipada com um processador *single-core* Broadcom BCM2835 que opera na velocidade de 1 GHz aliado a uma memória RAM de 512 MB, além de um *slot* para cartão microSD. A Raspberry PI Zero W v1.1 conta com *WiFi* e *Bluetooth* integrados, conector de vídeo mini HDMI e 2 portas USB para conectividade. Além disso, a placa permite que distribuições Linux como o Raspbian e Ubuntu sejam instaladas, as quais são compatíveis com *framework* ROS. A Raspberry Pi oferece também APIs nas linguagens de programação Python e C++ para controle do seu periférico GPIO de 40 pinos. Essas interfaces permitem a utilização de protocolos de comunicação como UART, SPI e I2C; controle de motores através de saídas *Pulse Width Modulation* (PWM); e pinos de entrada e saída para conexão com elementos digitais.

3.4.3 Motor, encoder e ponte H

Os robôs utilizados, em sua maioria, nas competições de Micromouse, têm como princípio de construção os robôs diferenciais. Este tipo de robô, muito comum no campo da robótica móvel, possui duas rodas conectadas a dois motores lateralmente opostos, os quais podem ser controlados independentemente. O sentido de giro e as velocidades das rodas determinam o movimento do robô. Na especificação dos motores a serem utilizados em um robô diferencial, os parâmetros mais relevantes são sua velocidade e seu torque. Em casos de robôs providos por baterias, o consumo do motor e a tensão de alimentação são também fatores primordiais na sua escolha.

O controle utilizado em robôs diferenciais se baseia no modelo cinemático do mesmo, o qual inclui parâmetros construtivos dos robôs e, principalmente, as velocidades das rodas. Por isso, é necessário que além do motor, dois componentes estejam presentes

na construção de um robô diferencial: um sensor de velocidade acoplado a cada roda, comumente utilizado um encoder para tal função e um *driver* de potência que permita controlar os motores de forma independente, isto é, controlar a velocidade e o sentido de giro de cada atuador.

Deste modo, o motor escolhido foi o *Micro Metal Gearmotor HP 6V with Extended Motor Shaft* do fabricante Pololu. Para compatibilização, este fabricante fornece alguns componentes que podem ser comprados em conjunto com o motor. Assim, optou-se por utilizar o encoder magnético desenvolvido para ser acoplado ao motor especificado, o qual possui uma resolução de 12 *Counts Per Revolution* (CPR) e faixa de operação de tensão entre 2,7 V até 18 V. Além do encoder, foi utilizado o par de rodas de 32 x 7 mm e os elementos de fixação também disponibilizados pelo mesmo fabricante. As especificações técnicas do motor podem ser visualizadas na Tabela 3.1

Tabela 3.1: Especificações técnicas do motor *Micro Metal Gearmotor HP 6V with Extended Motor Shaft*

| Descrição | Valor |
|--|-----------------|
| Tensão | 6 V |
| Tamanho | 10 x 12 x 26 mm |
| Peso | 9,5 g |
| Diâmetro do eixo | 3 mm |
| Relação de redução | 29,86:1 |
| Velocidade sem carga | 1000 RPM |
| Corrente sem carga | 0,07 A |
| Corrente com eixo parado | 1,6 A |
| Torque com eixo parado | 0,57 kg.cm |
| Máxima potência de saída1 | 1,5 W |
| Máxima eficiência | 41 % |
| Velocidade na máxima eficiência | 830 rpm |
| Torque na máxima eficiência | 0,10 kg.cm |
| Corrente na máxima eficiência | 0,36 A |
| Potência de saída na máxima eficiência | 0,89 W |

Para a ponte H, foi escolhido o módulo TB6612FNG, o qual permite o controle independente de dois motores de corrente contínua operando com tensão entre 4,5 à 13,5 V e com valores de corrente de até 1 A por canal. Não menos importante, este driver possui tensão lógica de operação entre 2,7 à 5,5 V e 100 kHz como máxima frequência de PWM permitida.

3.4.4 Emissor e fotorreceptor Infravermelho

Os robôs da modalidade micromouse têm como funcionalidade essencial a percepção. Para que o sistema que controla o robô decida qual a melhor opção de movimento, é necessário primeiro que se conheça quais as possibilidades existentes. Quem fornece informações para que essas possibilidades sejam criadas é exatamente a funcionalidade de percepção. O princípio básico dessa funcionalidade é informar, para cada célula do labirinto, quantas paredes circundam-a e onde estes obstáculos se encontram (Norte, Sul, Leste e Oeste).

De acordo com as regras estabelecidas nas competições reguladas pelo IEEE Região 9, os labirintos devem possuir cores específicas para os elementos que o compõe. Para as paredes do labirinto, a cor utilizada é a branca. A cor branca, quando comparada a cores mais frias, possui um alto valor de reflectância de luz cujo o comprimento de onda esteja dentro da faixa do infravermelho. Os sensores mais indicados que se beneficiam dessa característica são aqueles em que há alteração de propriedades física quando exposto a luz infravermelha. Sendo assim, a utilização de LEDs que operam com espectros não visíveis, como o infravermelho, em conjunto com fototransistores, é uma ótima opção para equipar robôs da modalidade micromouse.

Portanto, para instrumentar o projeto em questão, optou-se pela utilização do LED IR333C do fabricante Everlight, o qual opera na região do infravermelho, permitindo a emissão de luz com comprimento de onda de 940 ± 45 nm e com potência de dissipação e corrente máxima iguais a 150 mW e 100 mA respectivamente, ambas à temperatura ambiente. Para o receptor infravermelho, especificou-se o fototransistor PT333-3B do mesmo fabricante Everlight. Este sensor opera na faixa de espectro entre 840 a 1100 nm, possuindo maior sensibilidade quando exposto a ondas com comprimento de onda igual a 940 nm. À temperatura ambiente, o PT333-3B pode dissipar até 75 mW e drenar no seu coletor valores de correntes até 20 mA. Ambos componentes possuem tensão de operação compatível a utilizada no robô Doogie. Vale ressaltar que estes componentes podem ser substituídos por similares desde que as características física e elétrica mencionadas sejam compatíveis.

3.4.5 IMU

Os labirintos utilizados nas competições oficiais de Micromouse são formados por células de 18 x 18 cm. A estrutura física do labirinto é igual a uma matriz cuja a dimensão é 16 x 16. Sendo assim, os robôs devem saber em qual célula do labirinto ele se encontra no momento para coletar algumas informações, utilizadas posteriormente para a sua movimentação e para a otimização da solução do labirinto. Uma técnica utilizada para

obter a posição do robô dentro do labirinto é a de Odometria. Nesta estratégia, a partir das informações do encoder e das dimensões das rodas utilizadas, estima-se a distância percorrida em um intervalo de tempo, entretanto, é uma técnica vulnerável a erros acumulativos. Uma forma utilizada para melhorar a precisão deste método é a incorporação de uma IMU, as quais têm seus dados fundidos com os dados da Odometria, resultando em uma medição mais confiável.

Isto posto, o robô Doogie foi equipado com a MPU-6050. Este sensor permite medição de aceleração e velocidade angular nos 3 eixos cartesianos, totalizando 6 graus de liberdade (6DOF), com a medição de cada canal disponibilizada por conversores Analógico/Digital (A/D) com resolução de 16 bits. Além disso, a MPU-6050 realiza medição de temperaturas entre -40 à 85 °C possui interface de comunicação I2C e pode operar com tensões entre 3 à 5 V. Por fim, algumas bibliotecas nas linguagens de programação Python e C++ são disponibilizadas pela comunidade *open source* para comunicação da Raspberry Pi com esta IMU.

3.4.6 Conversor Analógico/Digital e Multiplexador Analógico

Como descrito no tópico 3.4.2, a Raspberry Pi Zero W v1.1 possui 40 pinos digitais. Entretanto, nenhum deles é habilitado para fazer conversão A/D. Como os sensores infravermelho especificados (ver tópico 3.4.4) fornecem valores analógicos de tensão em sua saída, é necessário o uso de um conversor externo. Além disso, robôs autônomos necessitam de verificação periódica da autonomia de sua bateria. Essa informação é também obtida de forma analógica. Logo, torna-se necessário que o Doogie Mouse seja equipado com um conversor A/D externo.

Mediante o exposto, foi selecionado o conversor ADS1115. Esse conversor funciona com tensões entre 2 à 5,5 V, e a tensão máxima nos pinos analógicos é igual à tensão de alimentação. Os pinos analógicos podem ser programados como 4 pinos independentes, ou dois canais diferenciais. Ademais, a interface de comunicação utilizada pelo ADS1115 é I2C.

O ADS1115 possui apenas 4 canais, entretanto, são necessários 5 canais: 4 para os sensores infravermelho e um para bateria. Para solucionar a carência de canais, foi especificado o multiplexador analógico 74HC4051. Esse módulo pode multiplexar até 8 canais, permitindo tensões de 2 à 10 V e frequência de até 170 MHz.

3.4.7 Lista de componentes

A lista contendo todos os componentes bem como suas respectivas quantidades e descrição pode ser visualizada no Apêndice C deste documento.

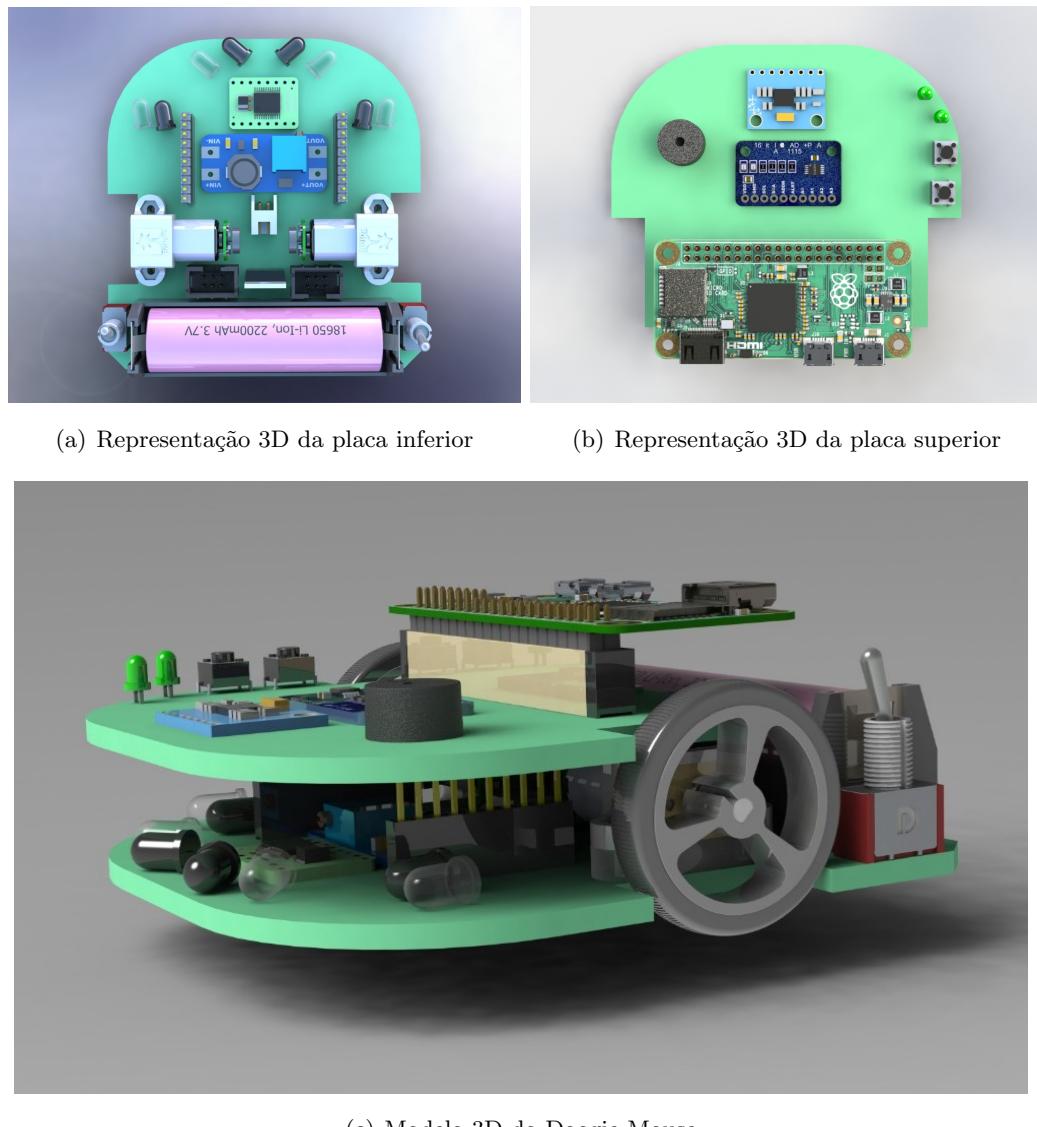
3.5 **Modelo mecânico**

3.5.1 Plataforma Robótica

Para o design do robô, foi utilizado como um ponto de partida o TON-BOT v1.1, plataforma desenvolvida pela Ioton Technology ([TECHNOLOGY, 2016](#)). Além disso, conforme ítem 7 da subseção 3.2, ele deve ter suas dimensões não superiores à uma seção retangular de 15 x 15 x 10 cm.

A partir dessas premissas e da análise feita na subseção 3.2, buscou-se um design mecânico simples e de maior leveza. Dessa forma, foi utilizado como *frame* do robô as próprias PCIs, buscando posicionar suas rodas de forma a mantê-las alinhadas ao centro de massa de todo o conjunto mecânico. Para tanto, uma modelagem em *Computer-Aided Design* (CAD) inicialmente foi realizada através da ferramenta Solidworks, buscando iterativamente a melhor disposição de seus elementos físicos (rodas, sensores e demais componentes eletrônicos das placas), para a elaboração das PCIs, vistas na subseção 3.6.1. Assim elaborou-se um esboço do modelo mecânico conforme a Figura 3.4.

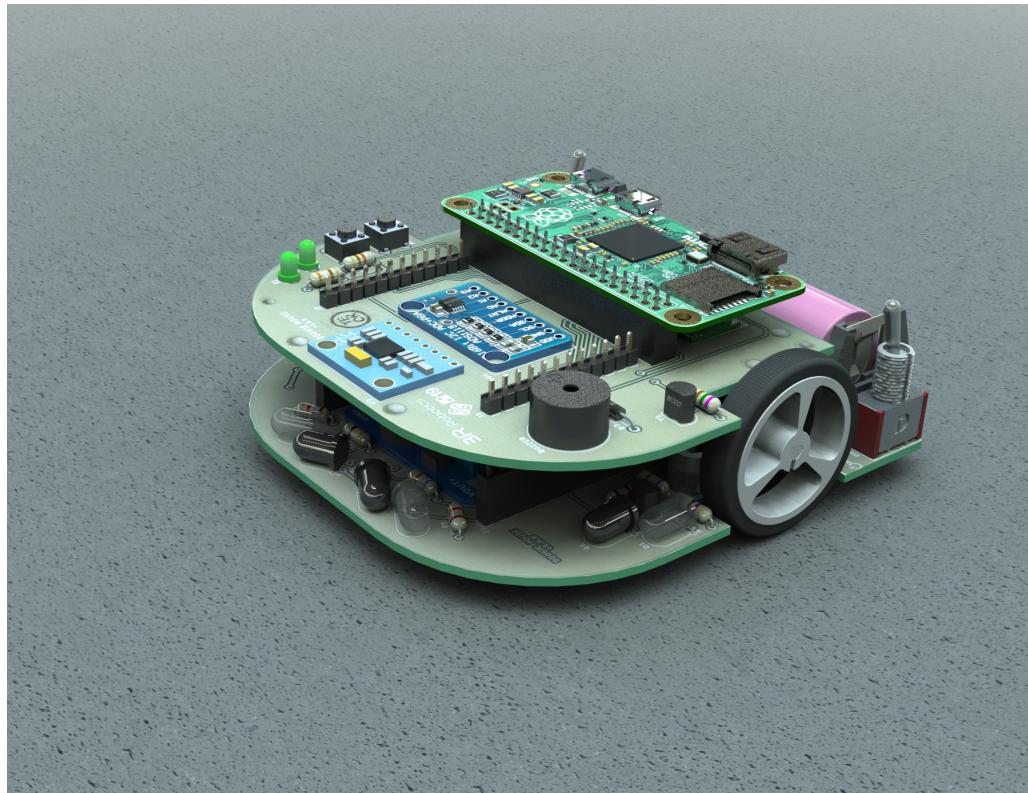
Figura 3.4: Modelo 3D do Doogie Mouse



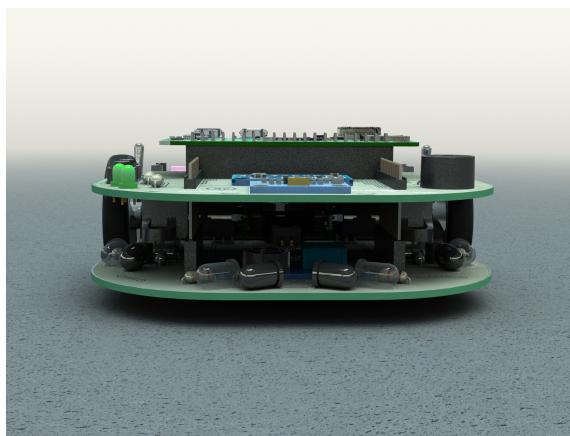
Fonte: Própria Autoria

Da esquerda para direita visualiza-se as placas inferior, superior e o modelo completo do robô visto em perspectiva. A placa inferior possui 98 mm de comprimento e 92,90 mm de largura, enquanto a placa superior possui 75 mm de comprimento e 92,90 mm de largura. Foi necessário o uso de duas placas para melhor adequação dos componentes eletrônicos sem atrapalhar eventuais manutenções no dispositivo nem dificultar sua montagem. O modelo 3D final, renderizado, pode ser visto abaixo na Figura 3.5. Os desenhos técnicos da plataforma móvel podem ser visualizados no Apêndice D.

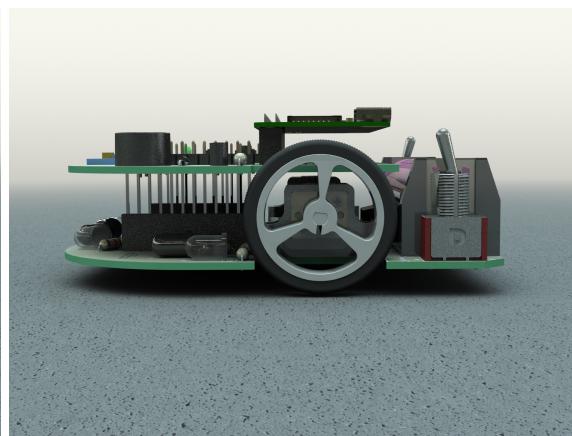
Figura 3.5: Modelo 3D Renderizado do Doogie Mouse



(a) Modelo 3D Renderizado do Doogie Mouse



(b) Vista frontal



(c) Vista Lateral

Fonte: Própria Autoria

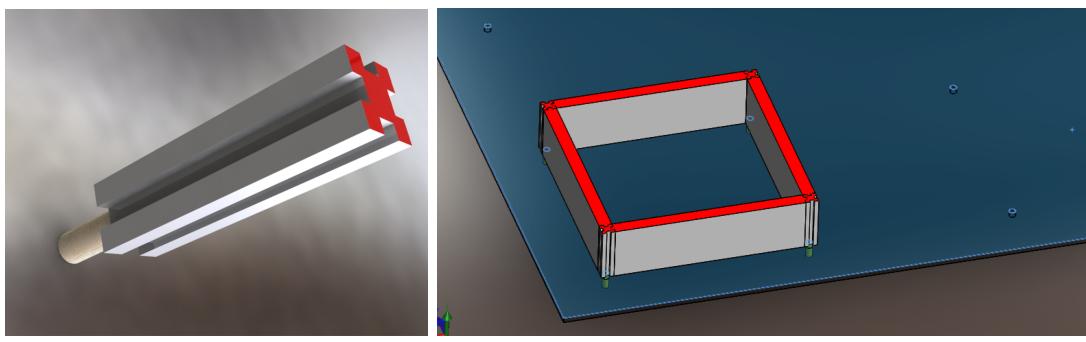
3.5.2 Labirinto

O modelo do labirinto foi desenvolvido a partir de uma adaptação do modelo IEEE ([WAN; RUBSTEIN, 2019](#)). Por ser um labirinto protótipo, preferiu-se trabalhar com uma matriz de 10x10 células quadradas, em vez de 16x16, mantendo-se as mesmas dimensões da célula

de 18 mm x 18 mm, com 50 mm de altura.

Além disso, buscou-se um design que fosse modular, de forma a facilitar seu transporte bem como múltiplas configurações do labirinto pela alteração da posição de suas paredes. Para isso, optou-se pela utilização de pivôs de madeira para a fixação das paredes na placa do labirinto, conforme visto na Figura 3.7(a). Um exemplo de uma célula é mostrado na Figura 3.7(a), enquanto seus desenhos técnicos se encontram no Apêndice E.

Figura 3.6: Modelo mecânico do labirinto



(a) Pivô de fixação

(b) Célula do labirinto montada

Fonte: Própria Autoria

3.6 *Modelo esquemático de alimentação e comunicação*

O Doogie é equipado com uma bateria do tipo Li-Ion com 3,6 V de tensão nominal. Para compatibilizar o nível de tensão da bateria com os demais componentes, são utilizados dois conversores DC-DC. O primeiro irá fornecer 6 V exclusivamente para os motores e os LEDs emissores. Já o segundo, é responsável por fornecer 5 V à Raspberry Pi. Entretanto, os componentes que estão conectados a Raspberry Pi são energizados com 3,3 V através de um conversor DC-DC interno a placa do sistema microprocessado.

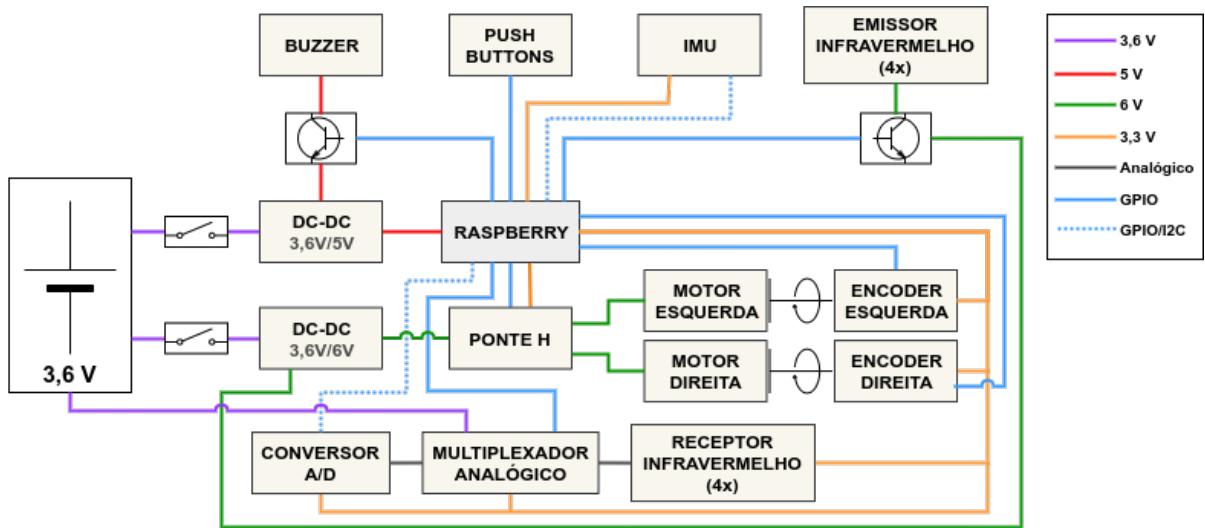
Com intuito de facilitar a replicação do robô por usuários que queiram utilizá-lo, optou-se pelo uso de *breakout boards*, que são placas eletrônicas pré-montadas. O Doogie possui seis delas: dois conversores de tensão DC-DC, um conversor A/D, uma IMU, um multiplexador analógico e uma ponte H. Os demais componentes como resistores, transistores, LEDs infravermelho, fototransistores e conectores, são soldados diretamente na PCI.

Os componentes do Doogie se comunicam e são controlados de diversas formas. O conversor A/D e a IMU se comunicam com a Raspberry Pi através de um barramento I2C. Como o conversor A/D especificado possui apenas 4 canais (um para cada sensor infravermelho), optou-se por utilizar um multiplexador analógico para permitir que o valor de

tensão da bateria também seja obtido. Este multiplexador é controlado por pinos digitais da unidade de processamento. Por fim, o controle dos motores é realizado por uma ponte H com dois canais independentes. Esse *driver* de potência permite que os sentidos de giros dos motores sejam controlados por pinos digitais da Raspberry Pi enquanto o controle de velocidade seja realizado por PWM.

A arquitetura elétrica na Figura 3.7 demonstra como esses componentes descritos estão interligados eletricamente. A propósito de melhor visualização, o referencial de tensão (Ground - GND) foi omitido.

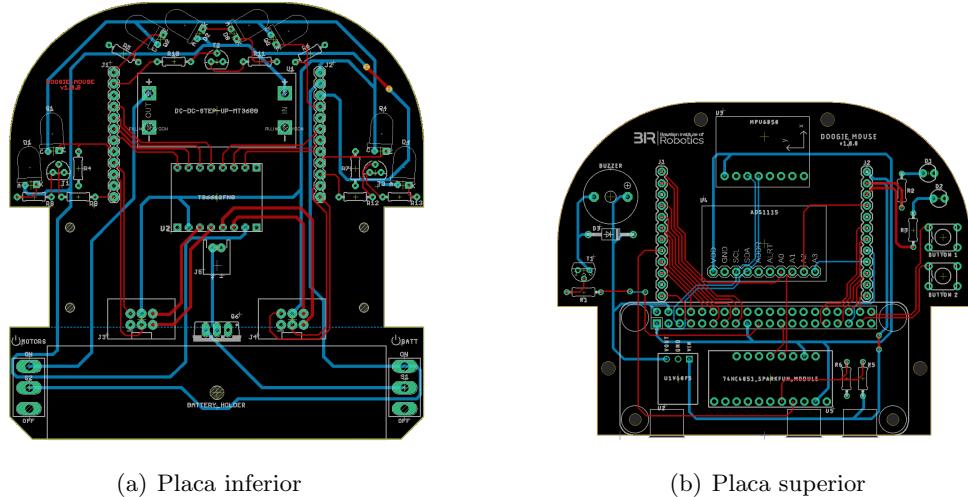
Figura 3.7: Representação elétrica do Doogie Mouse.



Fonte: Própria Autoria

3.6.1 Esquemáticos eletrônicos

Como explicado no tópico 3.5 o Doogie Mouse possui duas PCIs que são utilizadas como chassi do robô. Além disso, como também citado na subseção 3.6, os componentes eletrônicos e as *breakout boards* são soldados diretamente na PCI. Para elaboração do *layout* de tais placas, foi utilizado o software Autodesk Eagle 9.4.2. O resultado obtido pode ser visualizado na Figura 3.8. Já os esquemáticos eletrônicos, que descrevem em maior detalhe as ligações elétricas entre todos os componentes do robô bem como o valor das grandezas físicas dos elementos eletrônicos, podem ser visualizados no Apêndice F deste documento.

Figura 3.8: *Layout das PCIs do Doogie Mouse*

(a) Placa inferior

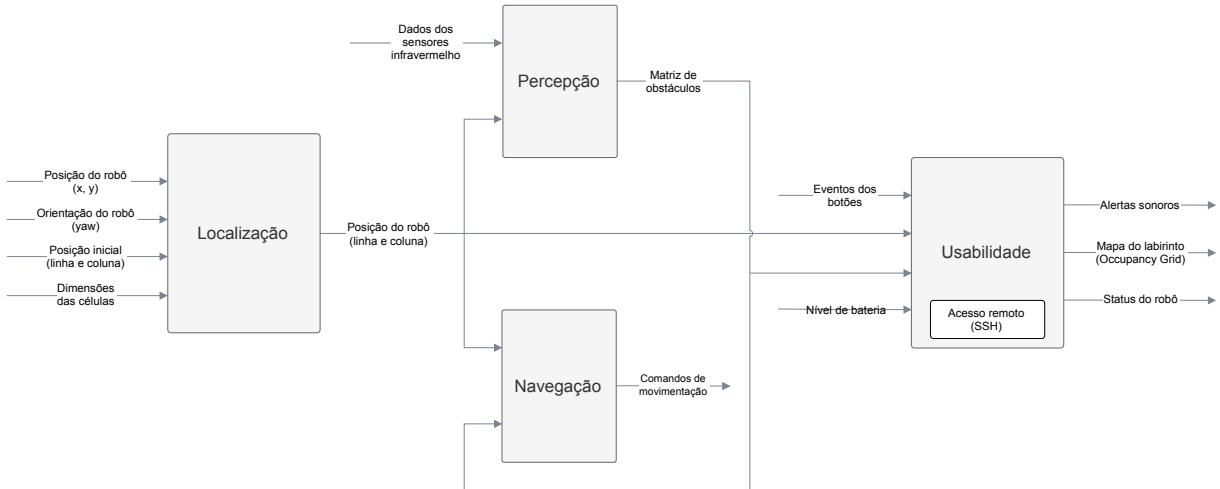
(b) Placa superior

Fonte: Própria Autoria

3.7 Especificação das funcionalidades

As funcionalidades de um robô descrevem os subsistemas e a lógica de operação dos mesmos. No Doogie, existem quatro funcionalidades principais: Localização, Percepção, Navegação e Usabilidade. O fluxo de informações de tais funcionalidades pode ser visualizado na Figura 3.9. Observa-se nesse fluxograma como cada funcionalidade está interligada com as demais e quais informações são trafegadas entre elas. Para melhor entendimento, é descrito nos tópicos sequentes cada funcionalidade individualmente.

Figura 3.9: Fluxo de informações das funcionalidades Localização, Percepção, Navegação e Usabilidade.



Fonte: Própria Autoria

3.7.1 Localização

O labirinto a ser percorrido pelo micromouse será modelado como uma matriz 16 x 16, sendo dividido em células de largura e comprimento fixos. Para que o robô consiga decidir para onde ele deve ir, primeiro é necessário saber onde ele está.

A estratégia utilizada para a obtenção da posição do Doogie utilizará a técnica de Odometria, onde, a partir das informações do encoder e das dimensões das rodas utilizadas, estima-se a distância percorrida em um intervalo de tempo. Além disso, para melhorar a precisão da medição, será utilizada uma IMU, capaz de fornecer os dados de aceleração e velocidade angular nos três eixos cartesianos.

Uma vez que o robô consiga calcular a distância percorrida, em qualquer intervalo de tempo, sabendo as dimensões de cada célula e assumindo que seu ponto de partida foi informado, é possível determinar as coordenadas do mesmo dentro do labirinto: linha e coluna.

3.7.1.1 Objetivo

Interpretar as informações fornecidas pela Odometria e identificar a posição do robô dentro do labirinto.

3.7.1.2 Dependências

Esse pacote depende da aquisição de dados publicados por:

- Pacote de driver da IMU;
- Pacote de driver do motor.

3.7.1.3 Premissas

Para que essa funcionalidade alcance o seu propósito, assume-se que:

- O ponto de partida do robô será informado pelo usuário: [linha, coluna];
- O valor de comprimento e largura das células são conhecidos pelo sistema e é diferente de zero;
- O valor de comprimento e largura (em metros) das células será previamente informado;
- O pacote de *driver* da IMU, aliado ao pacote de *driver* do motor, oferecerão a informação de orientação do robô.

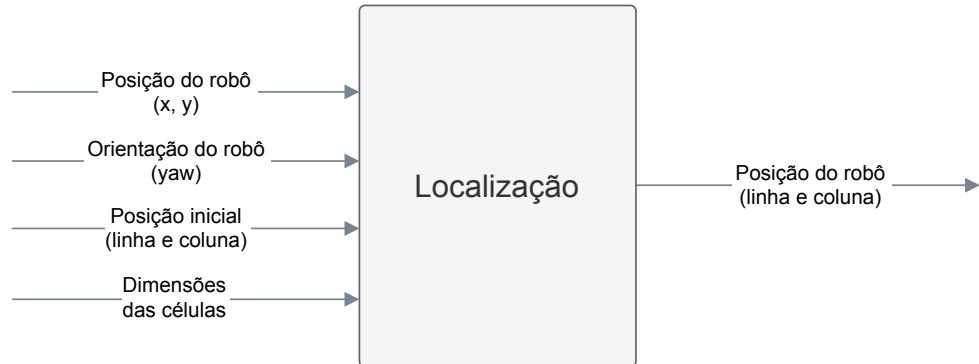
3.7.1.4 Saídas

Essa funcionalidade tem como saídas:

- Doogie position: localização do robô na matriz, no formato: [linha, coluna].

A Figura 3.10 demonstra as entradas e saídas da funcionalidade em questão.

Figura 3.10: Fluxograma ilustrativo da funcionalidade de Localização.



Fonte: Própria Autoria

3.7.2 Percepção

Para que o *micromouse* consiga se locomover pelo labirinto, é necessário reconhecer os possíveis caminhos, identificando os obstáculos ao seu redor. Utilizando informações obtidas dos sensores infravermelho, essa funcionalidade conseguirá definir a presença de paredes nas proximidades do robô.

Como mencionado anteriormente, o labirinto será modelado como uma matriz $m \times n$. Será utilizado um sistema de referência absoluto para o mesmo, definindo onde fica o norte, sul, leste e oeste. Com esse sistema de referência, a medida que o robô vai percorrendo o labirinto, em cada célula serão identificadas a presença de paredes. Dessa forma, essa funcionalidade irá publicar uma matriz, contendo a informação da presença de paredes ao norte, sul, leste e oeste de cada célula.

3.7.2.1 Objetivo

Identificar a presença de obstáculos ao norte, sul, leste e oeste de cada célula da matriz que representa o labirinto.

3.7.2.2 Dependências

Esse pacote depende da aquisição de dados publicados por:

- Pacote de driver dos sensores infravermelho;

- Posição do robô, obtida do pacote de localização.

3.7.2.3 Premissas

Para que essa funcionalidade alcance o seu propósito, assume-se que:

- Os sensores infravermelho estarão conectados à Raspberry Pi e a informação dos mesmos está sendo disponibilizada corretamente;
- O pacote de localização estará funcionando corretamente, publicando as informações de posicionamento do robô.

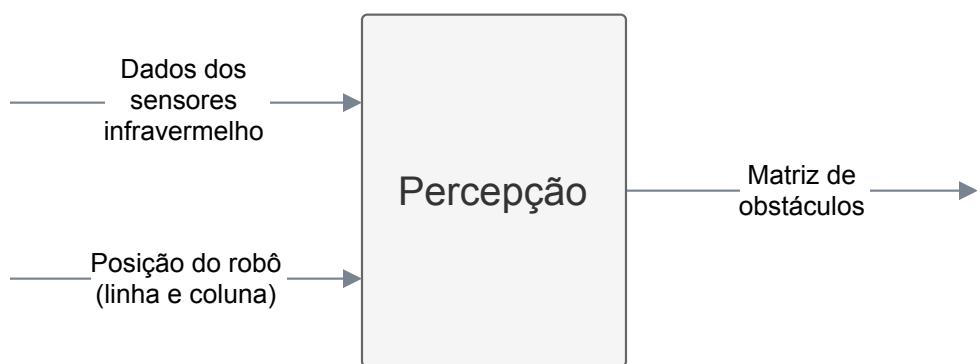
3.7.2.4 Saídas

Essa funcionalidade tem como saída:

- Matriz de dimensão $m \times n$ com cada célula contendo valores booleanos para as seguintes variáveis:
 - *north*: presença de parede ao norte da célula;
 - *south*: presença de parede ao sul da célula;
 - *east*: presença de parede ao leste da célula;
 - *west*: presença de parede ao oeste da célula.

Na Figura 3.11 pode ser visualizado quais serão as entrada e a saída da funcionalidade Percepção.

Figura 3.11: Fluxograma ilustrativo da funcionalidade de Percepção.



Fonte: Própria Autoria

3.7.3 Navegação

Para o robô seguir pelo melhor caminho dentro do labirinto é necessário antes que este seja conhecido por ele. Para isso, em um primeiro momento, será necessário que o micromouse percorra o labirinto somente para o seu conhecimento parcial. Então, será necessário a utilização das informações publicadas pelas funcionalidades de Percepção e Localização. Com base nelas, o sistema de navegação poderá definir os comandos necessários para que o robô execute o próximo movimento dentro do labirinto.

O micromouse será programado com 3 estratégias de solução do labirinto diferentes. Essas podem ser selecionadas pelo usuário através da interface de usuário. Primeiramente, o robô irá executar a funcionalidade de mapeamento com base no algoritmo de força bruta, que consiste em sempre que não houver obstáculos à direita, tomar este caminho, movimentando-se de modo diferente somente se for identificado impossibilidade de seguir pela direita. Após isso, o algoritmo de resolução fará com que o robô chegue ao destino final pelo percurso mais otimizado dentro das possibilidades conhecidas previamente pelo mapeamento do labirinto.

3.7.3.1 Objetivo

Planejar trajetórias a fim de garantir a correta locomoção do robô, possibilitando primeiramente o mapeamento do labirinto e após isso a chegada ao destino final pelo caminho mais rápido.

3.7.3.2 Dependências

Esse pacote depende da aquisição de dados publicados por:

- Matriz de obstáculos;
- Posição do robô: localização do robô na matriz, no formato: [linha, coluna].

3.7.3.3 Premissas

Para que essa funcionalidade alcance o seu propósito, assume-se que:

- Correto funcionamento dos pacotes de localização e percepção;
- Os drivers de potências dos motores estarão conectados à Raspberry Pi e o driver ROS e o sistema de controle dos movimentos do robô estejam funcionando corretamente.

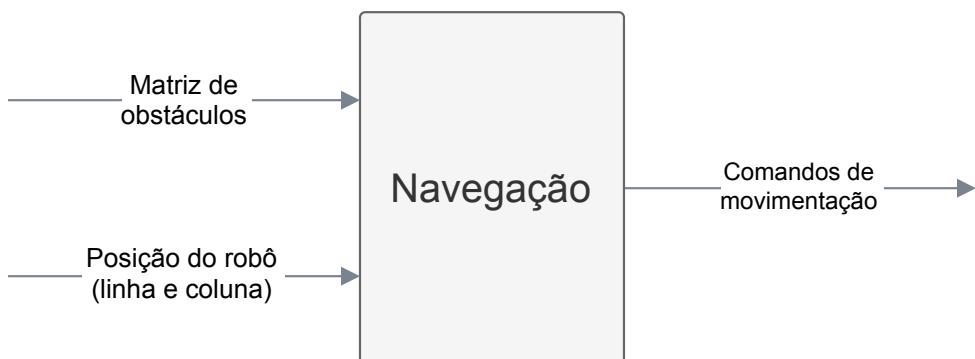
3.7.3.4 Saída

Essa funcionalidade tem como saídas:

- Comandos de movimentação.

As entradas e saída da funcionalidade Navegação podem ser visualizadas na Figura 3.12.

Figura 3.12: Fluxograma ilustrativo da funcionalidade de Navegação.



Fonte: Própria Autoria

3.7.4 Usabilidade

Para permitir que o usuário interaja com o robô, algumas interfaces são disponibilizadas. A primeira delas consiste do acesso via SSH que permite o acesso remoto do robô. Usando essa interface é possível acessar a linha de comando do robô para realizar configurações do dispositivo de processamento, compilação de códigos fonte, configuração de parâmetros do robô e execução tanto das rotinas de demonstração do robô quanto das rotinas de resolução de labirintos. Já para que o usuário exerça interação com a plataforma sem a necessidade do acesso remoto, é disponibilizado botões e um buzzer. Esses dois elementos, são utilizados principalmente em competições para facilitar o manuseio do robô.

Por fim, quando não utilizado em competições, o mapa do labirinto pode ser visualizado

no RViz (visualizador 3D do *framework* ROS) a medida em que ele é percorrido pela plataforma móvel.

3.7.4.1 Objetivo

Permitir que o usuário acesse e edite configurações, execute comandos e monitore o estado do robô.

3.7.4.2 Dependências

Esse pacote depende da aquisição de dados publicados por:

- Driver do Buzzer;
- Driver dos botões;
- Módulo de Localização;
- Módulo de Percepção.

3.7.4.3 Premissas

Para que essa funcionalidade alcance o seu propósito, assume-se que:

- Uma rede comunicação Wireless com a plataforma esteja disponível;
- O acesso via SSH seja configurado previamente;
- O Buzzer esteja conectado a Raspberry PI e o ROS driver esteja funcionando corretamente;
- Os módulos de Percepção e Localização estejam funcionando corretamente.

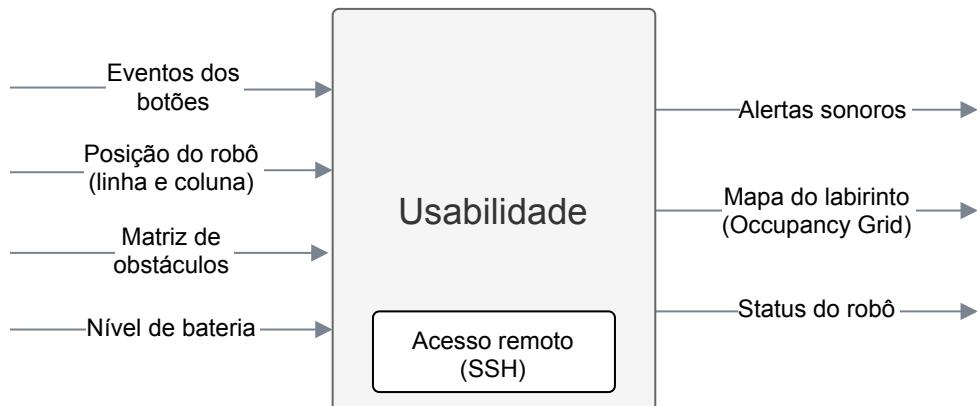
3.7.4.4 Saídas

Essa funcionalidade tem como saídas:

- Alertas sonoros;
- Mapa do labirinto (*Occupancy Grid*);
- Status do robô.

As entradas e saída da funcionalidade Navegação podem ser visualizadas na Figura 3.13.

Figura 3.13: Fluxograma ilustrativo da funcionalidade de Usabilidade.



Fonte: Própria Autoria

3.8 Simulação

O ambiente de simulação do robô foi realizado dentro do Gazebo ([KOENIG; HOWARD, 2004](#)) na versão 7.0.0, uma vez que o simulador, além de ser *open source*, já possui integração nativa com o ROS através do conjunto de pacotes providos pelo `gazebo_ros_pkgs`. Dessa forma, os mesmos recursos usado pelo ROS para controlar o *hardware* do robô, implementados através do pacote `ros_control`, são também utilizados na simulação. Essa integração proporciona uma experiência mais realista do controle do robô, sendo possível inclusive a integração do ambiente de simulação com o robô real.

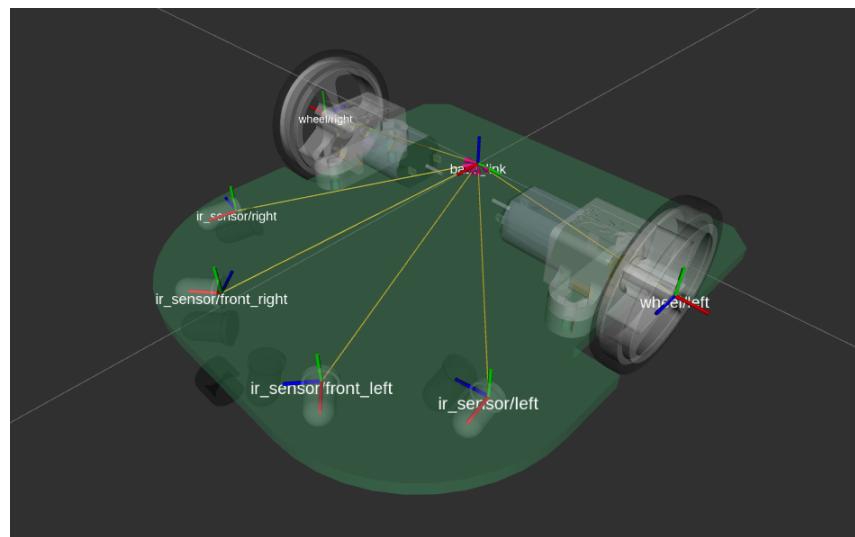
A simulação será composta tanto pelo modelo do robô Doogie Mouse, quanto pelo labirinto, que serão descritos em um *Unified Robot Description Format* (URDF), formato de arquivo padrão usado pelo ROS para descrever o modelo de um robô, definindo-se assim os links, juntas, sensores e o funcionamento dos atuadores utilizados no Doogie Mouse, e os links e juntas usados para descrever o labirinto.

Para tanto, serão aproveitados os modelos 3D gerados na seção 3.5, para a constituição da malha de efeito visual (a que de fato é renderizada na simulação) e da malha de colisão (na qual se calcula os contatos e colisões do robô com o mundo).

De forma a otimizar a simulação, tornando-a mais leve, o modelo do robô foi simplificado, mantendo-se apenas os componentes essenciais para sua representação: a placa inferior, sensores infravermelhos e as rodas. A partir disso, através do *plugin* SolidWorks to URDF Exporter, mantido pela comunidade do ROS, gerou o modelo URDF do robô.

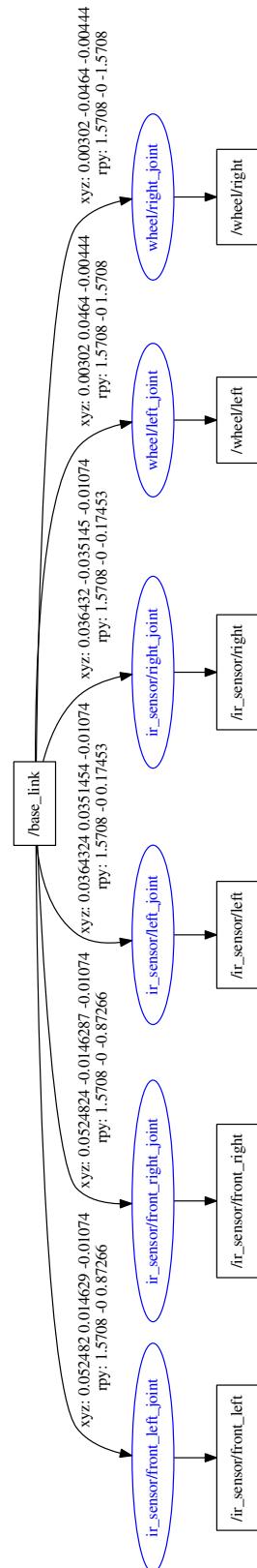
O simulador na versão utilizada suporta apenas dois formatos de arquivo para descrição de malhas: *COLLAborative Design Activity* (COLLADA) e *Stereolithography* (STL). Através do plugin utilizado no SolidWorks a malha é exportada em STL, contudo neste formato não se transporta informação de cor e textura para a simulação. Para contornar esse problema, através do SolidWorks Visualize exportou-se a malha do modelo do robô em Wavefront e então, através do Blender converteu-a para COLLADA para ser renderizada na simulação. A malha de colisão por sua vez foi simplificada em um dos formatos primitivos fornecidos pelo URDF, um paralelepípedo, de forma a conter os limites de contato do robô. Uma representação em grafos do modelo do robô pode ser vista na Figura 3.15 e na Figura 3.14, o mesmo modelo através do visualizador do ROS

Figura 3.14: Modelo do Doogie Mouse para Simulação.



Fonte: Própria Autoria

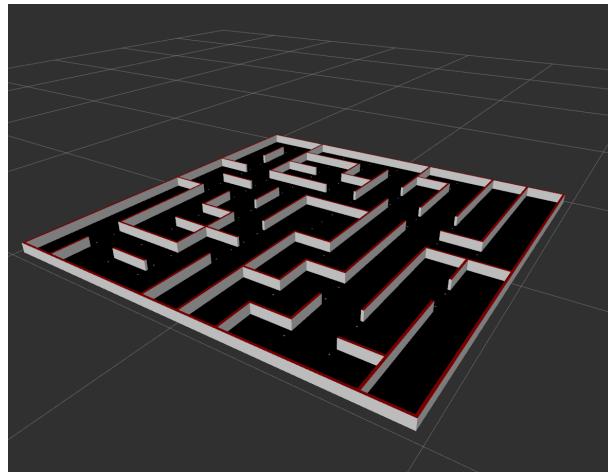
Figura 3.15: Descrição Mecânica do Doogie Mouse.



Fonte: Própria Autoria

O labirinto, por sua vez, foi descrito como um único link cuja malha descreve tanto suas paredes quanto a base. Dentro do Gazebo, o mundo simulado já possui uma base que sustenta os objetos renderizados na simulação, sendo necessário assim somente a malha de colisão das paredes, por isso supriu-se a base, mantendo-a apenas como malha visual do labirinto. O mesmo procedimento utilizado para gerar o URDF do robô foi replicado para o labirinto, visto na Figura 3.16.

Figura 3.16: Modelo do Labirinto para Simulação.



Fonte: Própria Autoria

Também foram gerados URDFs utilizando as malhas em STL providas pelo plugin afim de realizar um ensaio para verificar como cada malha interfere na simulação e estabelecer a sua melhor configuração tomando como base os indicadores de *Frames per Second* (FPS) e *Real Time Factor* (RTF) fornecidos pelo Gazebo.

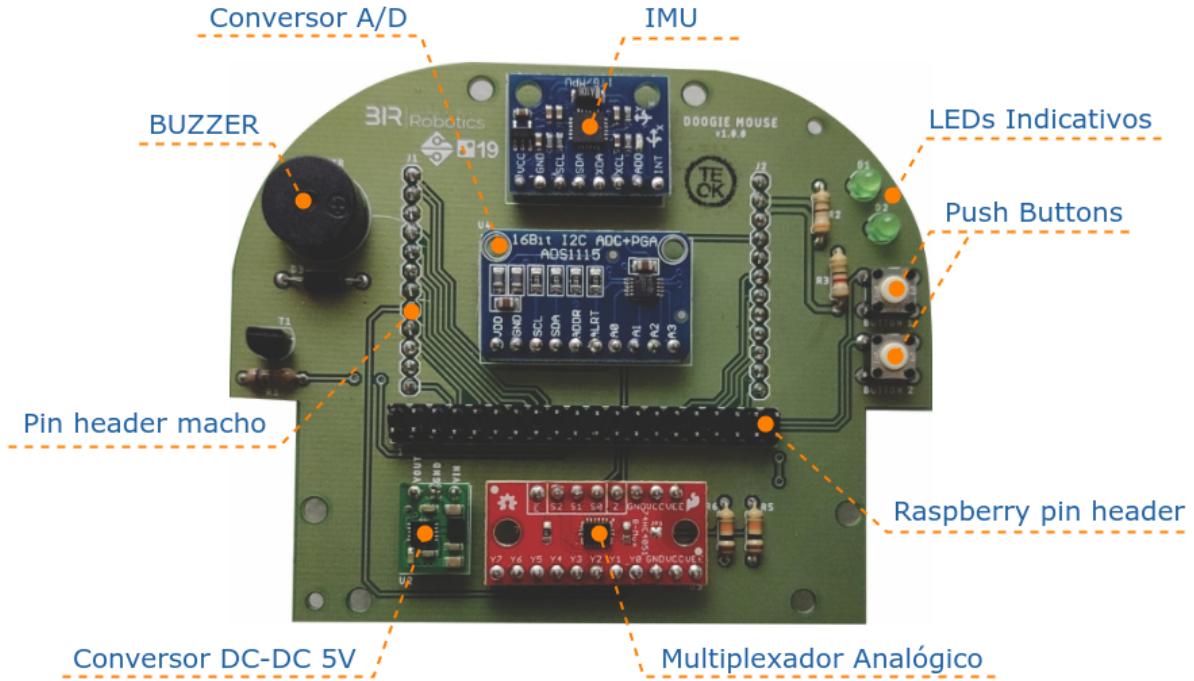
Resultados

A fase de desenvolvimento do robô Doogie culminou em 3 resultados principais: Protótipo, ambiente de simulação e a estrutura de software da plataforma móvel. Os tópicos posteriores irão detalhar cada um desses resultados.

4.1 *Protótipo*

O protótipo é composto por duas placas (placa inferior e placa superior) resultantes da integração de componentes eletrônicos. Na Figura 4.1, é possível visualizar alguns dos componentes eletrônicos que compõem a placa superior, quais sejam: a IMU que é responsável por fornecer dados que auxiliam no processo de localização do Doogie dentro do labirinto, os LEDs, Buzzer, Push Buttons que têm por finalidade garantir a interface do usuário com o robô, promovendo indicações luminosas e sonoras. O *Pin Header* da Raspberry que é a conexão física da placa com a própria Raspberry, o Multiplexador Analógico e o Conversor A/D que interagem entre si permitindo a aquisição dos dados dos sensores e da bateria em formato analógico, transformando-os em digital e, por fim, o conversor DC-DC 5V que garante que a tensão de alimentação da Raspberry mantenha-se conforme especificação (5V). Vale ressaltar que o *Pin Header* macho é responsável por permitir a conexão entre as placas inferior e superior.

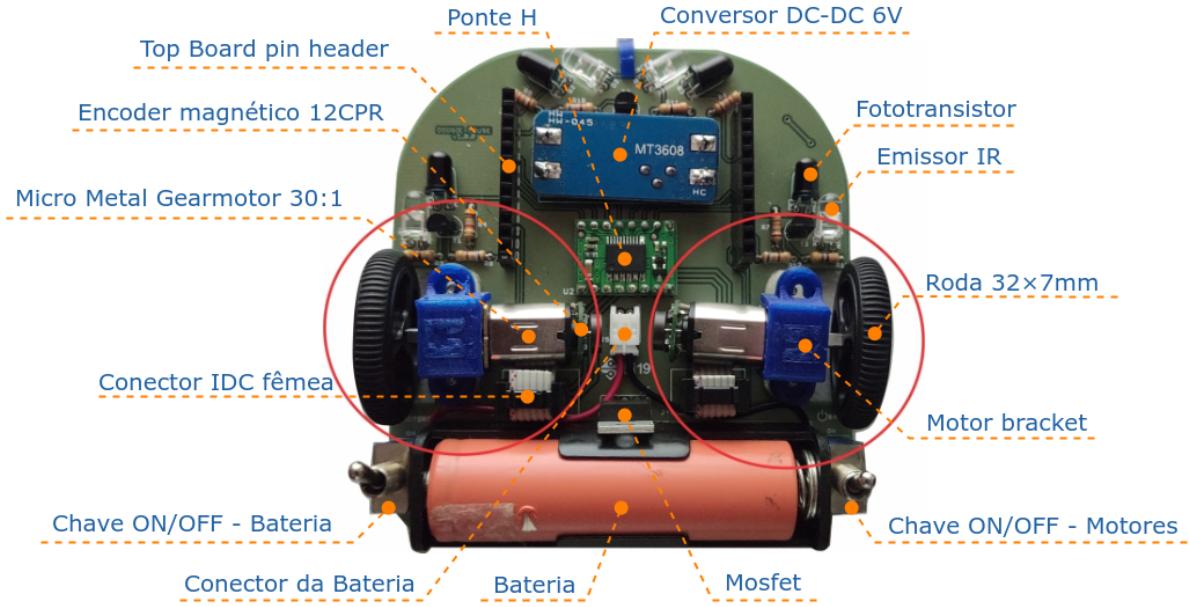
Figura 4.1: Componentes da placa superior do Doogie Mouse



Fonte: Própria Autoria

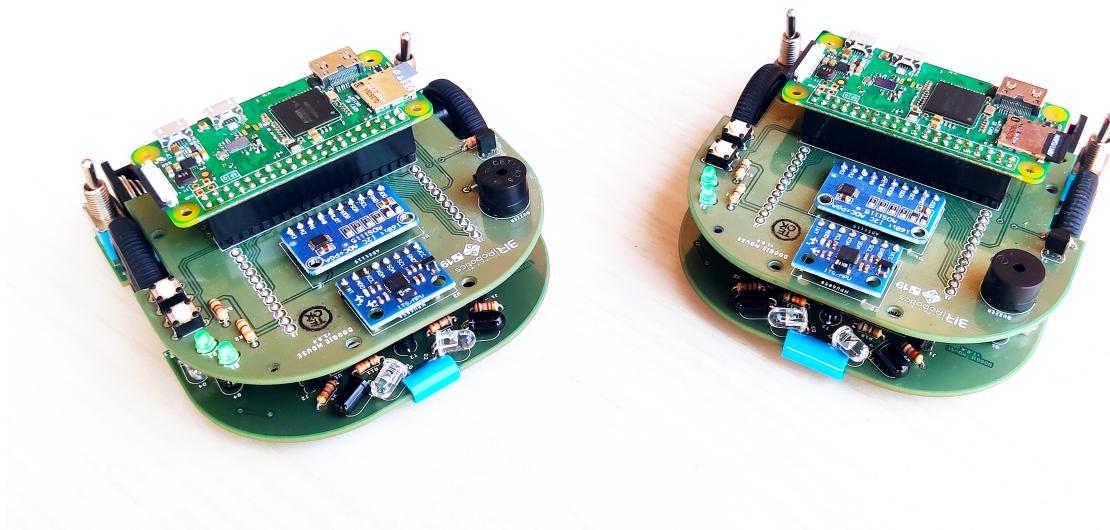
Sabendo sobre a funcionalidades dos componentes da placa superior, podemos observar agora a figura 4.2. Os componentes que compõem a placa inferior, cuja parte circulada em cor vermelha refere-se ao conjunto do motor que é formado pelas rodas, *bracket*, micro motor, encoder magnético, *flat cable* e conector IDC fêmea. Esse conjunto é responsável por garantir o correto acionamento e funcionamento do motor, promovendo suas interligações elétricas e mecânicas, fornecendo dados para aprimorar o sistema de controle do motor. É possível identificar 4 pares de sensores infra-vermelho (Emissor IR e Fototransistor) na placa inferior, que são responsáveis por identificar os obstáculos próximos ao robô. Observa-se também o conversor DC-DC 6V que tem função de fornecer para os sensores e motores a tensão necessária para o funcionamento dos mesmos, o componente Ponte H que permite o controle de sentido de giro do eixo dos motores bem com a potência elétrica, as chaves Liga/Desliga que ligam e desligam os circuitos da bateria e dos motores, o conector da bateria que é responsável pela fixação da bateria e o próprio componente bateria que é responsável por fornecer a tensão de alimentação para o funcionamento de ambas as placas. A integração das placas inferior e superior juntamente com a Raspberry é o que compõem o protótipo físico do robô Doogie Mouse (vide Figura 4.3).

Figura 4.2: Componentes da placa inferior do Doogie Mouse



Fonte: Própria Autoria

Figura 4.3: Protótipos do Doogie Mouse confeccionados



Fonte: Própria Autoria

Visando auxiliar os usuários, o protótipo dispõe de dois guias: um guia de montagem das placas e um guia de configuração de software. O guia de montagem das placas está disponível no site GitHub para acesso por qualquer usuário. Esse guia foi criado para propor-

cionar ao usuário uma experiência didática com relação à parte de hardware do Doogie e apresenta uma sequência de passos para fixação dos componentes eletrônicos, assim como algumas dicas, que tem por objetivo garantir mais facilidade no decorrer do processo de montagem. Nesse documento também são listados os requisitos físicos necessários para montagem do robô, bem como, ferramentas utilizadas, componentes eletrônicos e os equipamentos de proteção individual (EPI's), aconselháveis utilização durante todo o procedimento para evitar qualquer tipo de acidente. Com o intuito de eliminar possíveis dúvidas durante as etapas de montagem, o documento supracitado dispõe de imagens que ilustram detalhadamente todo passo a passo de fixação de cada componente. Uma parte desse guia, que possui mais de 34 passos de montagem, pode ser visualizado na Figura 4.4. O guia completo pode ser visualizado no Apêndice G.

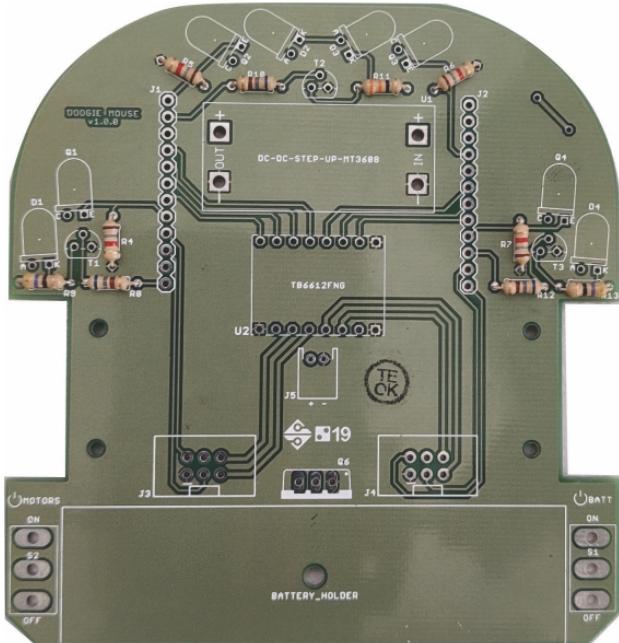
Figura 4.4: Recorte do guia de instruções de montagem do Doogie Mouse

General Instructions

1. Every steps where there direct contact with metalic parts that will be welded, is advised use the nose plier for avoid accident for burnig.
2. After every steps maked take off the excess of the terminals of the eletronic components with help of cut plier.
3. Every process of compoennts fixation will be maked with a Soldering Iron and tin.
4. Is advised make soldering process on a place with air flux for avoid the smoke resulted of the soldering process.
5. During all process you should to use the security glasses and soldering gloves for avoid accidents.

Bottom Board

1. We are going to start the board assembly by the resistors, with help of soldering iron weld them on the board, look the picture below.



Fonte: Própria Autoria

De maneira análoga ao guia de montagem das placas, o guia de configuração de software também está disponível no GitHub, conforme ilustrado pela Figura 4.5. O manual

de configuração de software informa sobre os requisitos físicos necessário para iniciar as configurações e apresenta uma lista com as instruções sobre instalação e resolução de dependência de todos os softwares necessários para o funcionamento do Doogie, que varia desde a preparação da unidade de armazenamento em que é instalado o sistema operacional até um código teste dentro do ambiente do ROS. Esse manual também dispõe de uma galeria de imagens ilustrando as telas de acordo com cada procedimento, garantindo ao usuário um ótimo entendimento das etapas. Este manual pode ser visualizado na íntegra no Apêndice H.

Figura 4.5: Recorte do guia de instruções de configuração de software do Doogie Mouse

Doogie Software Setup

A repository with software setup instructions for Doogie Mouse robot.

Introduction

This repository will help you on the setup of the Raspberry Pi Zero W v1.1 which is used in the Doogie Mouse robot. These instructions assume that you have already the Raspberry peripherals such as mouse, keyboard and monitor connected on it. See the [hardware_instructions](#) or [oficial Raspberry instructions](#) for more informations.

These instructions package has been made under Ubuntu 16.04 LTS.

How to Read?

The notes are more or less in chronological order, hence start from top the bottom. Commands are prefixed with the system on which the commands needs to be run:

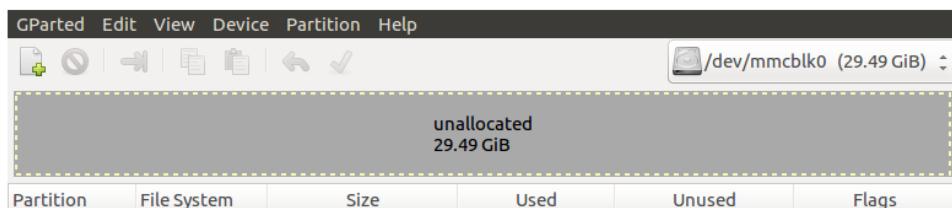
- HOST:~\$ commands executed on your machine
- RPI:~\$ commands executed on the Raspberry Pi

Prepare microSD card

The first step is to format the microSD card. If you are using an old microSD card, remember to do the backup of the media or you will lost all data. To format the card, install and run the GParted (GNOME Partition Editor):

```
HOST:~$ sudo apt-get install gparted
HOST:~$ sudo gparted
```

Delete all partitions, if any. You will see the image below.



Fonte: Própria Autoria

4.2 Ambiente de simulação

4.2.1 Comparativo de Malhas

No comparativo de malhas, analisou-se como os formatos **COLLADA** (.dae) e **STL**(.stl) interferem na simulação, bem como o uso de tipos primitivos para a representação da malha de colisão. Na Tabela 4.1, comparam-se os modelos simplificados, gerado na seção 3.8, usando malhas em COLLADA (Doogie_lite_dae) e STL (Doogie_lite_stl), e o modelo 3D sem simplificações (Doogie_real_dae) gerado na seção 3.5 usando sua malha em COLLADA.

Tabela 4.1: Comparativo entre malhas carregadas no Gazebo

| | Visual (Faces) | Colisão (Faces) | FPS | Real Time Factor |
|------------------------|---------------------------|----------------------------|------------|-----------------------------|
| Doogie_lite_dae | 70.410 | 70.410 | ~1,5 | 0,35 |
| Doogie_real_dae | 1.091.337 | 1.091.337 | ~0,6 | 0,03 |
| Doogie_lite_stl | 70.398 | 70.398 | ~4.2 | 0,33 |

Fonte: Própria Autoria

O formato glscollada possui performance na simulação ligeiramente inferior ao STL, principalmente em relação ao FPS no qual foi 36% inferior ao desempenho da outra malha. Também fica visível a necessidade da simplificação do modelo do robô, sendo inviável a simulação do Doogie_real_dae que teve seu FPS inferior a um e o RTF próximo a zero, denotando uma simulação lenta e de pouca acurácia.

Tabela 4.2: Comparativo com simplificação da malha de colisão

| | Visual (Faces) | Colisão (Faces) | FPS | Real Time Factor |
|------------------------|---------------------------|----------------------------|------------|-----------------------------|
| Doogie_lite_dae | 70.410 | 6 | ~8,0 | 0,52 |
| Doogie_real_dae | 1.091.337 | 6 | ~3,5 | 0,52 |

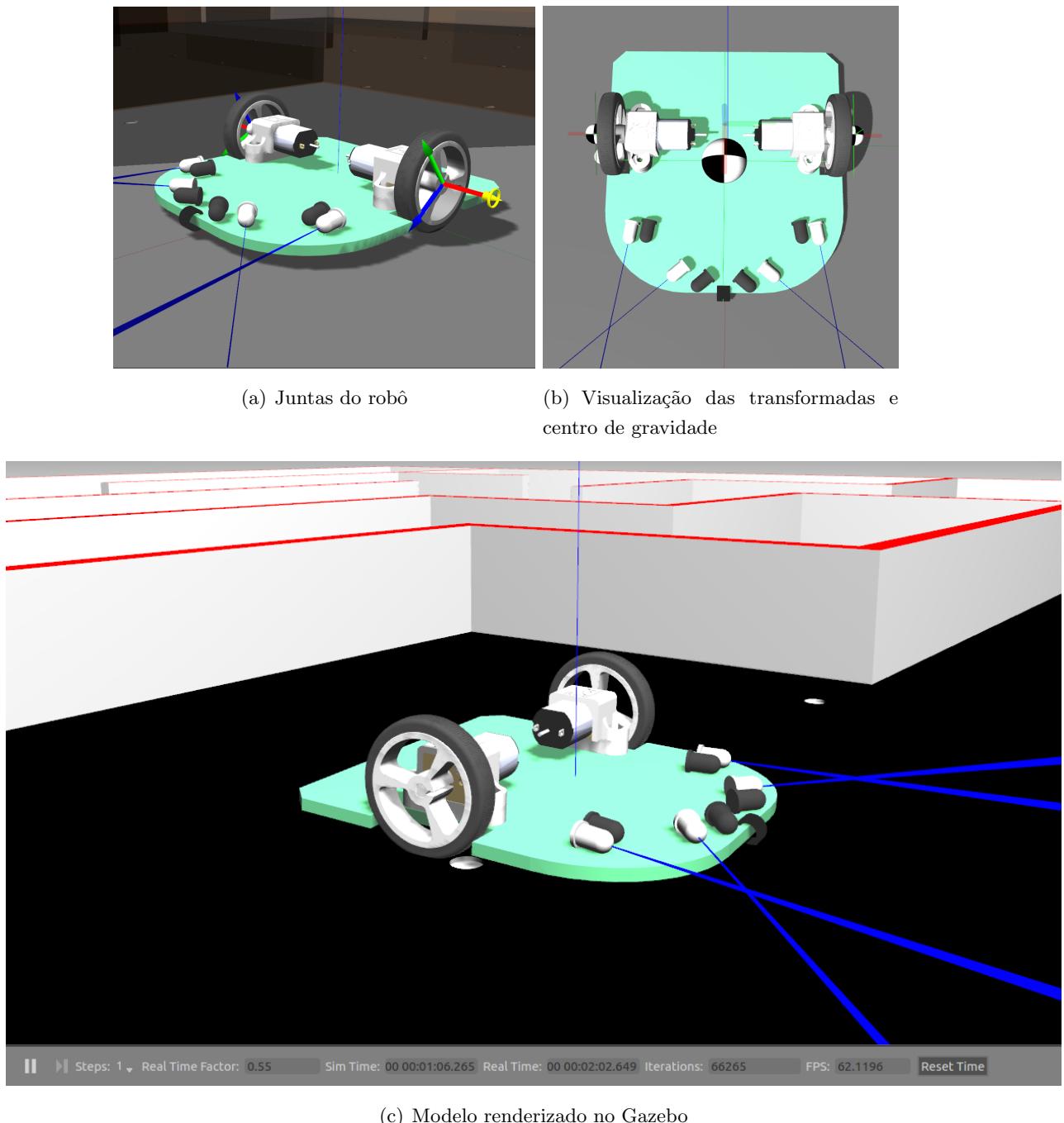
Fonte: Própria Autoria

Ao comparar o desempenho como uso do modelo simplificado e o modelo real, usando tipos primitivos para a representação da malha de colisão pode-se perceber que o gargalo da simulação se encontra no número das faces de colisão usadas para a representação dinâmica do robô. A taxa de FPS aumentou em seis vezes em relação a análise anterior, atingindo um RTF igual ao do modelo simplificado, conforme visto na Tabela 4.2. Sendo assim a configuração Doogie_lite_dae com a simplificação da malha de colisão a mais

otimizada para uso na simulação sem perda de cor e textura do modelo do robô.

4.2.2 Modelos da Simulação

Figura 4.6: Doogie Mouse no Gazebo

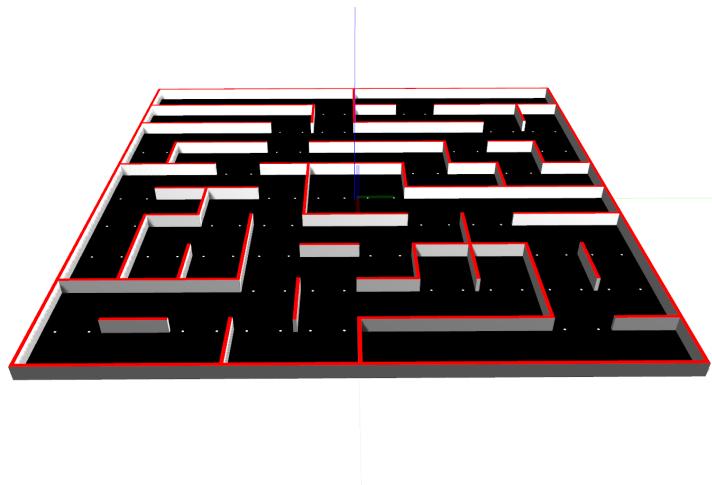


Os modelos exportados para o Gazebo, seguindo a metodologia descrita na seção 3.8, simplificando o modelo 3D gerado pelo SolidWorks, e usando a configuração otimizada

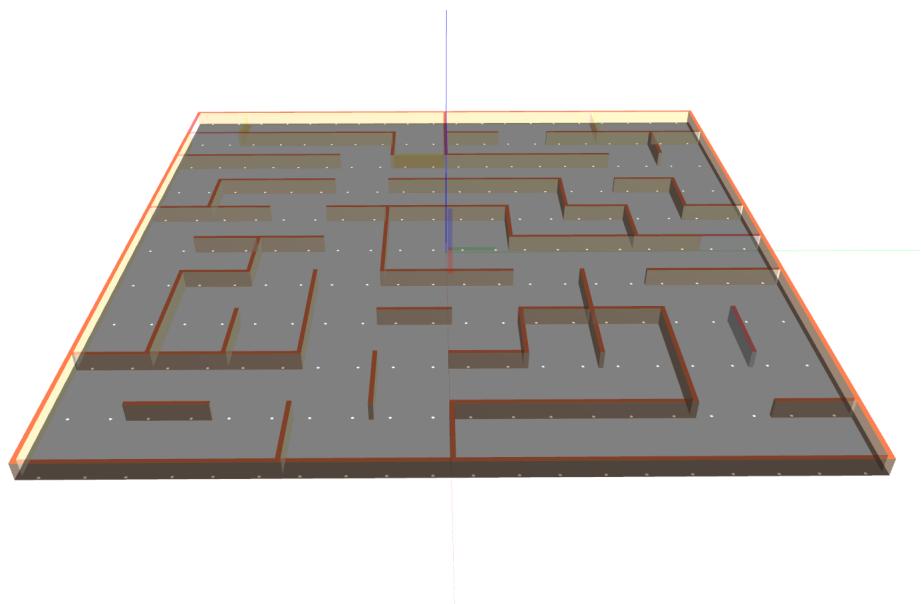
obtida no comparativo de malhas da subseção 4.2.1, do robô e do labirinto foram elaborados de forma a não comprometer a simulação em sua acurácia de ser realizada próxima do tempo real, RTF, ou na sua taxa de atualização de quadros FPS. Assim, atingiu-se a simulação simultânea de ambos os modelos, sem comprometimento desses indicadores e em máquinas mais recentes a simulação pode chegar a se manter dentro de 60 FPS, a taxa mais comum de atualização usada nos monitores comerciais.

Dessa forma, provê-se um ambiente de simulação que facilite o acesso dos usuários ao Doodie Mouse, mesmo que para esses não seja possível adquirir a plataforma física, tornando-o mais acessível em sua proposta de ensino.

Figura 4.7: Labirinto na Simulação



(a) Labirinto renderizado no Gazebo



(b) Malha de colisão das paredes

4.3 Estrutura de Software

Para contemplar o objetivo do projeto, que é ser uma plataforma para aprendizagem de robótica móvel e inteligência artificial, optou-se pelo uso do *framework* ROS devido a sua capacidade de abstração de hardware e de reuso de software como já mencionado em 2.3. Diante disso, foi criada uma estrutura de software modular que pode ser visualizada na Figura 4.8. Essa estrutura é formada por *stacks*, que são conjunto de pacotes agrupados em um mesmo local. Abaixo está descrito qual o objetivo de cada pacote dentro dessas *stacks*.

doogie_base:

- **doogie_algorithms**: Contém os algoritmos de inteligência que são utilizados pelo robô na resolução do labirinto;
- **doogie_control**: Possui arquivos de configuração e de inicialização dos processos de controle do robô;
- **doogie_description**: Abarca arquivos com a descrição do modelo do robô;
- **doogie_msgs**: Compreende as estrutura de dados (.msg e .action) criadas exclusivamente para o Doogie Mouse;
- **doogie_navigation**: Abrange os processos utilizados para a percepção, localização e navegação do robô dentro do labirinto. Fornece também APIs para permitir que usuários implementem sua própria estratégia de resolução de labirinto.

doogie_simulator:

- **doogie_gazebo**: Integra todos os arquivos de configuração necessários para executar o ambiente de simulação.

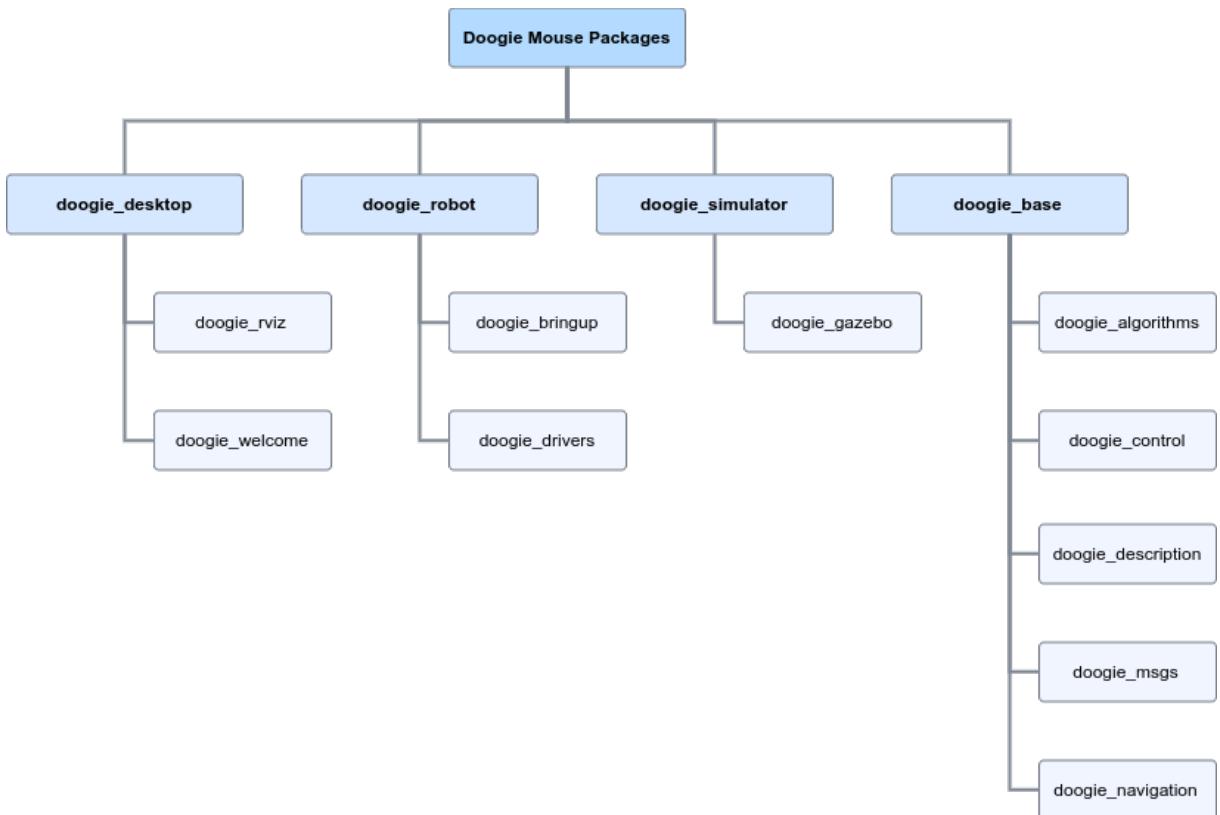
doogie_robot

- **doogie_bringup**: Agrupa arquivos de configuração e inicialização de todos os subsistemas do robô. Contém também os executáveis responsáveis pela atuação da plataforma móvel.
- **doogie_drivers**: Compreende executáveis e bibliotecas dos componentes de hardware do robô (motor, Encoder, IMU e sensores infravermelho).

doogie_desktop

- **doogie_rviz**: Engloba arquivos de configuração e inicialização para visualização de dados do robô no visualizador RViz;
- **doogie_welcome**: Contém executáveis para verificação do funcionamento do *framework* ROS na etapa de instalação.

Figura 4.8: Organização dos pacotes do Doogie Mouse.



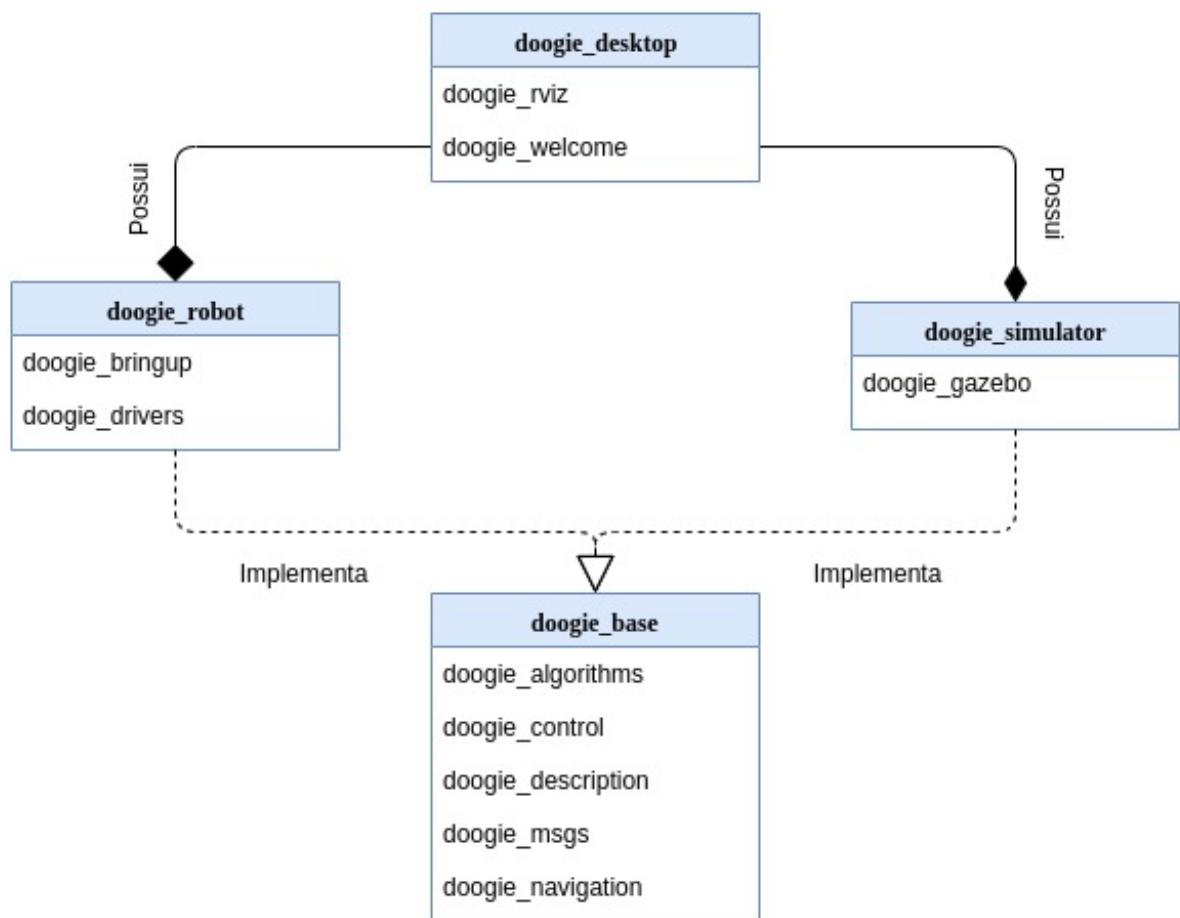
Fonte: Própria Autoria

Dentre as *stacks* e pacotes listados na Figura 4.8 apenas a *stack* doogie_desktop e o pacote doogie_rviz não foram criados. Os demais estão disponíveis para utilização no repositório remoto GitHub.

Embora todos esses pacotes tenham sido criados, nem todos eles são empregados no funcionamento da plataforma móvel. Como alguns pacotes são utilizados apenas para simulação do robô, não há a necessidade da sua utilização dentro da plataforma. Dessa forma, os pacotes foram organizados de acordo com o propósito final: utilização no ambiente de simulação, no robô físico ou em ambos. A Figura 4.9 demonstra como esses

pacotes se relacionam entre si. A *stack* doogie_base é comum tanto ao ambiente de simulação quanto ao robô físico, porém, não tem utilidade se não utilizada dentro desses dois contextos. A *stack* doogie_robot possui todos os pacotes essenciais para o funcionamento do robô físico, entretanto, possui dependência direta com a *stack* doogie_base. Em oposição está a *stack* doogie_simulator válida apenas para uso do ambiente de simulação, mas igualmente a doogie_robot, depende da doogie_base para funcionar. Englobando todas as *stacks* mencionadas bem como a reunião de toda a documentação do robô está a *stack* doogie_desktop. A relação de dependência entre esses pacotes é feita através de arquivos de configuração que são utilizados no processo de compilação dos pacotes através de ferramentas específicas do *framework* ROS. Caso uma dependência não seja satisfeita, por exemplo uma *stack* não instalada, um erro ocorrerá.

Figura 4.9: Relação de dependência das *stacks* do Doogie Mouse.

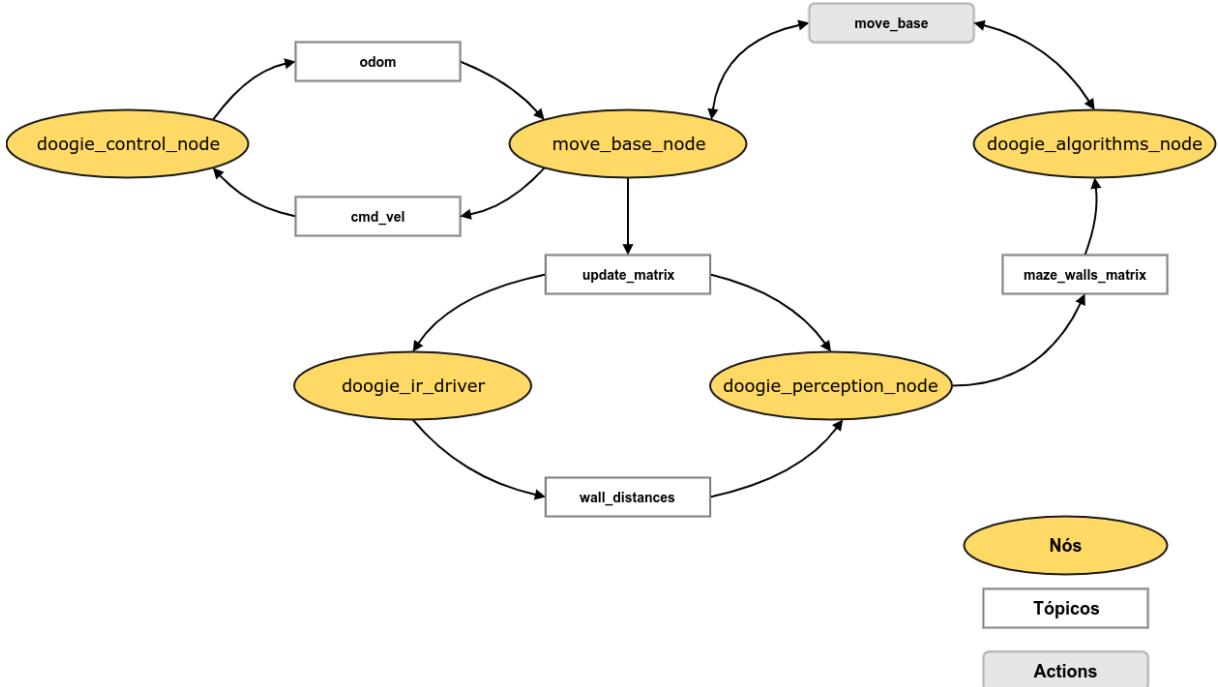


Fonte: Própria Autoria

Continuando com a proposta de modularidade de sistemas para proporcionar um grau de flexibilidade no uso de diferentes estratégias de resolução do labirinto, foi pensado, embora não testado, um modelo de comunicação entre os processos dentro do ambiente

ROS, conforme ilustrado pela Figura 4.10.

Figura 4.10: Diagrama de comunicação entre processos.



Fonte: Própria Autoria

Partindo da premissa que a posição e orientação inicial da base móvel dentro do labirinto é conhecida, a movimentação do robô começa com o doogie_algorithms_node enviando um comando de movimentação para o move_base_node. Após interpretado qual movimento a plataforma deve executar, o processo move_base_node enviará valores de velocidade linear e angular através do tópico cmd_vel para o processo responsável pelo controle de movimentação do robô. Com o *feedback* de Odometria, o move_base_node infere se o robô chegou ao destino desejado. Enquanto o robô está se movimentando, uma informação de progresso é enviada para o doogie_algorithms_node de modo que ações possam ser tomadas enquanto o robô se locomove, como emitir uma alerta sonoro quando 90% do trajeto for concluído. Quando o robô chega ao seu destino final (uma determinada célula do labirinto), uma informação de posição e orientação do robô é fornecida através do tópico update_matrix. Essa informação é armazenada pelo processo de percepção e dispara também uma requisição para que os dados dos sensores infravermelhos sejam fornecidos. Após a requisição de dados, o *driver* dos sensores infravermelho irá providenciar no tópico wall_distances as informações de distância das paredes do labirinto. Com os valores de distância, posição e orientação do robô, o processo doogie_perception_node sintetiza uma matriz de obstáculos fornecendo em relação a um referencial global quais paredes existem para cada célula do labirinto visitada. Essa matriz é então fornecida para o processo que

contém a estratégia de resolução do labirinto escolher a próxima ação de movimentação do robô. Essas etapas se repetem até que o robô encontre o centro do labirinto.

Com essa proposta de organização de comunicação entre processos e de posse das APIs fornecidas para envio de comandos de movimentação do robô e manuseio da matriz de obstáculos, o usuário pode implementar sua própria estratégia de resolução de labirinto. O fator limitante de qual abordagem dentro da IA utilizar será apenas o poder de processamento da Raspberry Pi.

Conclusão

Neste trabalho foi apresentada uma plataforma open source para ensino de robótica e introdução à inteligência artificial. Foram confeccionados dois protótipos, capazes de percorrer e solucionar um labirinto, utilizando algoritmos de busca.

O robô foi desenvolvido utilizando o *framework* ROS e o ambiente de simulação Gazebo, dando a possibilidade de se trabalhar e experimentar com os robôs reais e com o gêmeo digital.

Cerca de 90% dos pacotes e 75% das stacks foram completamente desenvolvidas pela equipe do projeto. Procurou-se ao máximo evitar a “reinvenção da roda”, utilizando estratégias do ramo da robótica que já são bem consolidadas na comunidade.

Foram desenvolvidos guias de montagem, modelo 3D, desenhos técnicos, guias de configuração de software e documentação independente dos pacotes. Tudo isso com o intuito de auxiliar o usuário durante o seu processo de aprendizagem.

Foram utilizadas boas práticas de desenvolvimento, além de padrões utilizados em outros projetos de robótica. Dessa forma, buscou-se obter um sistema que possa ser bem recebido tanto por iniciantes quanto por pessoas já familiarizadas com a área.

É possível perceber que a plataforma consegue extrapolar o seu objetivo inicial: além de ser utilizada como um robô solucionador de labirintos, nela podem ser aplicadas técnicas de *Simultaneous Localization and Mapping* (SLAM), técnicas diferentes de controle, além da utilização de técnicas de navegação de robôs autônomos como *State Machine* e *Behavior Tree*.

A realização desse projeto foi bastante enriquecedora, proporcionando aos discentes grande aprendizado. Todos os participantes conseguiram aprimorar suas habilidades, tanto na área profissional, quanto na área pessoal.

5.1 Trabalhos futuros

Para melhorar a experiência do usuário com o Doogie Mouse pretende-se no futuro realizar as seguintes ações:

- Elaborar uma Wiki no GitHub reunindo toda documentação do robô;
- Confeccionar um labirinto para testes do protótipo físico;
- Desenvolver três estratégias de resolução de labirinto para servirem como exemplo base de nova implementações;
- Realizar a fusão de dados de Odometria e IMU para oferecer uma maior precisão de movimento do robô.

Matriz de Comparação dos robôs *micromouse* estudados

| ÁREA | PESO | RECURSO | PESO (kg) | GreenGiant | Kumamoto National College | Smartmouse | WaffiMouse | Raspberry Pi Mouse V2 |
|-----------------------|------|------------------------------------|-----------|------------|---------------------------|------------|------------|-----------------------|
| Documentação | 1 | Documentação Disponível | 1 | 0,5 | 0,75 | 0,75 | 0,75 | 0,25 |
| | | Enrique Eduacional | 1 | 0 | 1 | 0 | 0 | 1 |
| | | Tutoriais | 1 | 0 | 0 | 0,25 | 0,75 | 1 |
| Middleware | 1 | Supporte a RTM | 0,7 | 0 | 0 | 0 | 0 | 1 |
| | | Supporte a ROS | 1 | 0 | 0 | 0 | 0 | 1 |
| | | Realiza Simulação | 1 | 0 | 0 | 1 | 1 | 1 |
| Ambiente de Simulação | 0,8 | Suporte a Pybullet | 0,5 | 0 | 0 | 0 | 0 | 0,5 |
| | | Suporte a V-Rep | 0,5 | 0 | 0 | 0 | 0 | 0,5 |
| | | Suporte a Gazebo | 1 | 0 | 0 | 1 | 0 | 1 |
| Linguagem | 0,5 | Programado em C/C++ | 1 | 1 | 1 | 1 | 1 | 0,2 |
| | | Programado em Python | 1 | 0 | 0 | 1 | 1 | 1 |
| | | Display | 0,8 | 1 | 0 | 0 | 1 | 0 |
| User Interface | 0,5 | LEDs | 0,2 | 1 | 1 | 1 | 0 | 1 |
| | | Buzzer | 0,2 | 0 | 0 | 0 | 0 | 1 |
| | | Botões | 0,2 | 1 | 0 | 0 | 1 | 1 |
| Sensores | 0,5 | Sistema de Diagnóstico | 1 | 0 | 0 | 0 | 0 | 1 |
| | | IR | 0,2 | 1 | 1 | 1 | 1 | 1 |
| | | MPU | 0,2 | 1 | 0 | 0 | 1 | 0 |
| Expansível | 0,8 | ToF | 0,2 | 0 | 0 | 0 | 1 | 0 |
| | | IMU | 0,8 | 0 | 0 | 0 | 0 | 0 |
| Plataforma | 1 | Plataforma é Facilmente Expansível | 1 | 0 | 0 | 0 | 0 | 1 |
| | | Somatório: | 1,8 | 2,45 | 3,8 | 4,1 | 8,25 | |

Requisitos do cliente x Requisitos técnicos

| Row # | Requisitos técnicos | Requisitos do cliente | | | | | | | | Target |
|-------|---|-----------------------|---|---|---|---|---|---|---|--|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 1 | Disponibilização de todo código desenvolvido no GitHub | ● | | | ▽ | | | | | Repositório no GitHub para cada pacote de desenvolvimento de software com instruções de utilização do pacote escritas em um arquivo README |
| 2 | Disponibilização guia do usuário como Wiki no GitHub | ● | | ○ | ● | | | | | Repositório no GitHub com Wiki explicando como utilizar o robô |
| 3 | Disponibilização em um repositório no GitHub esquemáticos eletrônicos, desenhos técnicos mecânicos e seus respectivos arquivos para possível edição | ● | | | ▽ | | | | | Repositório no GitHub contendo esquemáticos elétricos, mecânicos e documentações complementares relacionadas ao hardware do robô |
| 4 | Utilização de uma Raspberry pi como sistema microprocessado | | ● | ○ | | ● | ○ | ▽ | | Utilização da Raspberry PI Zero W |
| 5 | Permissão ao usuário de acesso remoto do terminal do sistema operacional do robô | | | ● | | | | | | Uso do protocolo SSH para acesso remoto |
| 6 | Visualização do mapa do Labirinto no Rviz | | ▽ | ● | ▽ | | | | | Mapa 2D utilizando Occupancy Grid |
| 7 | Disponibilização de botões e display para interação com usuário na ausência de acesso remoto | | ○ | ● | | ○ | ▽ | ○ | | Utilização de <i>push buttons</i> e display OLED |
| 8 | Desenvolvimento de tutorial na ROS Wiki com os primeiros passos com robô, explicando ao usuário comandos básicos de locomoção | ● | | ○ | ● | | | | | Tutorial no ROS Wiki com os comandos básicos para locomoção do robô |
| 9 | Desenvolvimento de tutorial na ROS Wiki explicando ao usuário como implementar no robô seu próprio algoritmo de resolução de labirinto | ● | | ○ | ● | | | | | Tutorial no ROS Wiki explicando ao usuário como implementar seu próprio algoritmo de resolução de labirinto |
| 10 | Bateria recarregável | | ○ | | | ● | ● | ● | | Uso de bateria recarregável |
| 11 | Autonomia superior a 1h40min | | ○ | | | ● | ▽ | | | Autonomia do robô igual a 02h00 |
| 12 | Uso de conectores intuitivos | | | ○ | | | ● | ○ | | Uso de conectores polarizados |
| 13 | Utilização de padrão de cores para cabos de conexão | | | ● | | | ● | | | Utilização de cores específicas para diferenciar funcionalidades dos cabos utilizados |
| 14 | Especificação de componentes disponíveis no mercado nacional | | ● | ▽ | | ○ | ● | ○ | | Aquisição de componentes em lojas nacionais |
| 15 | Desenvolvimento de shield de interface entre a plataforma de processamento e os sensores e atuadores | | ● | ○ | | | ● | ● | | Confecção de uma PCI (Placa de Circuito Impresso) |
| 16 | Dimensões do robô devem ser menores que 15 x 15 x 10 cm | | ○ | | | ▽ | ▽ | ● | | Estrutura mecânica com dimensões inferiores à 15 x 15 x 10 cm |

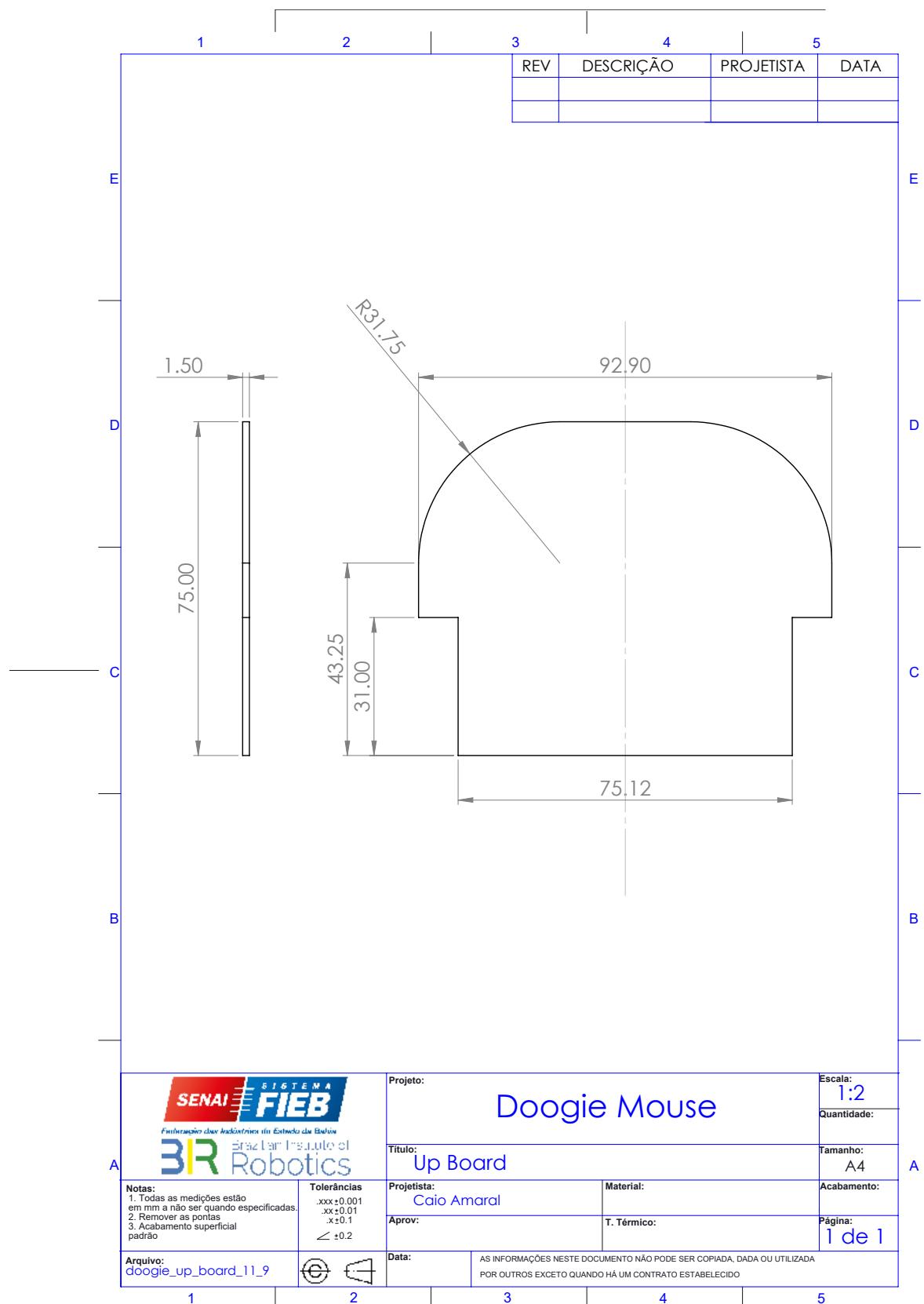
Lista de componentes

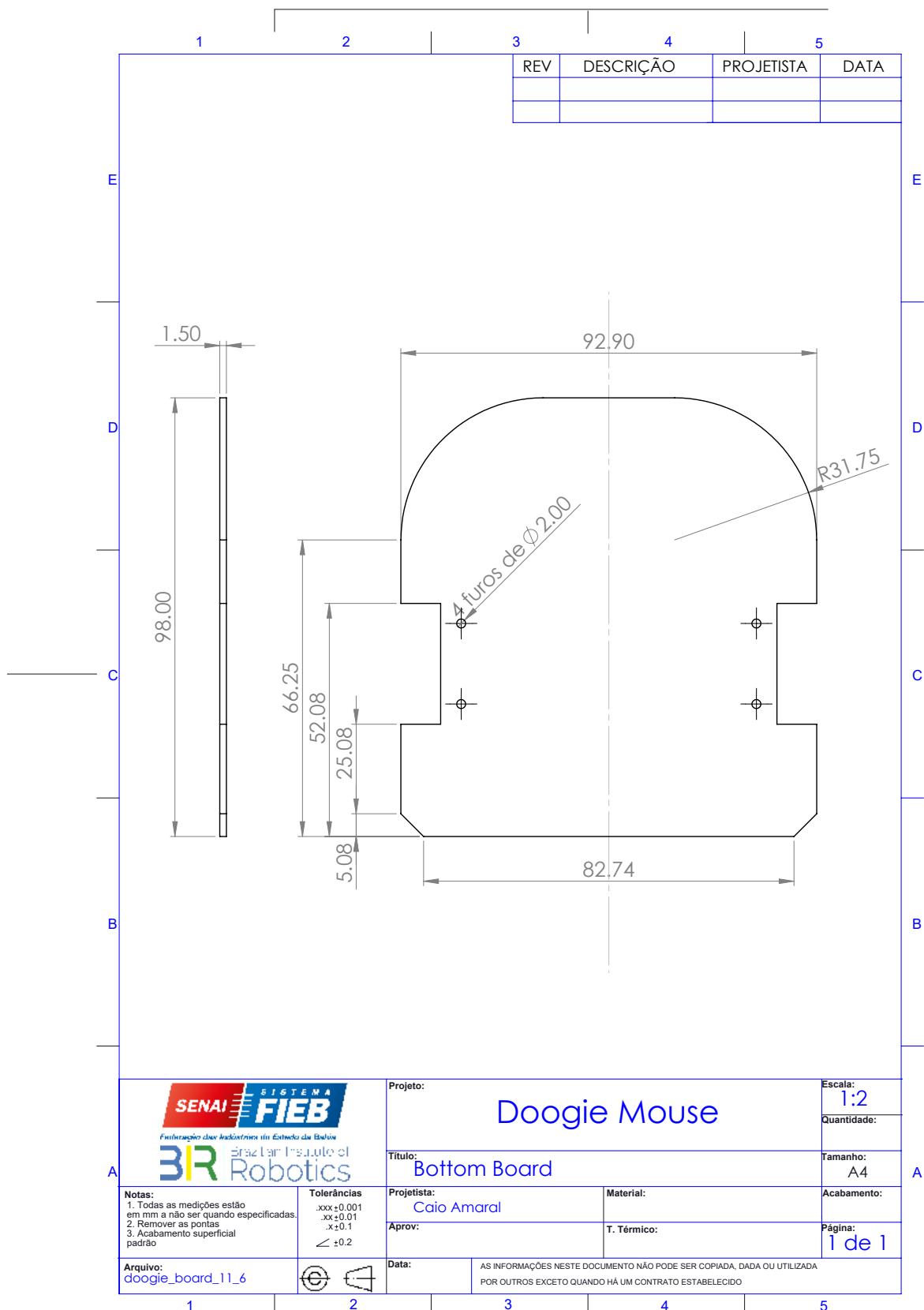
| Placa Superior | | | |
|-----------------------|----------------------------|-------------|--|
| Ítem | Elemento | Qtd. | Descrição |
| 1 | Placa de circuito impresso | 1 | doogie_top_board |
| 2 | Raspberry PI | 1 | Raspberry Pi Zero W |
| 3 | MicroSD | 1 | Cartão de Memória Classe 10 16GB MicroSd SanDisk |
| 4 | Regulador de tensão | 1 | Regulador de tensão Pololu 5V Step-Up U1V10F5 |
| 5 | IMU | 1 | Acelerômetro e Giroscópio 3 Eixos 6 DOF MPU-6050 |
| 6 | Conversor A/D | 1 | Conversor Analógico Digital 4 canais ADS1115 |
| 7 | Multiplexador Analógico | 1 | Módulo Multiplexador 8 Canais 74HC4051 Sparkfun |
| 8 | Barra de pinos macho 1x12 | 1 | Barra de Pinos 1x40 180 Graus 20mm |
| 9 | Barra de pinos Macho 2x40 | 1 | Barra de Pinos 2x40 180 Graus |
| 10 | Barra de pinos fêmea 2x20 | 1 | Barra de pinos 2x20 macho e fêmea |
| 10 | Push button | 2 | Chave Táctil Push-Button |
| 11 | LED | 2 | LEDs 3mm |
| 12 | Buzzer | 1 | Buzzer Ativo 5V |
| 13 | Transistor | 1 | Transistor NPN BC337 |
| 14 | Diodo | 1 | Diodo Retificador 1N4007 |
| 15 | Resistor | 1 | Resistor 750Ω 1/4W |
| 16 | Resistor | 1 | Resistor 390Ω 1/4W |
| 17 | Resistor | 1 | Resistor 820Ω 1/4W |
| 18 | Resistor | 2 | Resistor 10kΩ 1/4W |

| Placa Inferior | | | |
|-----------------------|----------------------------|-------------|--|
| Ítem | Elemento | Qtd. | Descrição |
| 1 | Placa de circuito impresso | 1 | doogie_bottom_board |
| 2 | Regulador de tensão | 1 | Conversor Boost DC Ajustável Step Up |
| 3 | Ponte H | 1 | TB6612FNG Dual Motor Driver Carrier |
| 4 | Barra de pinos fêmea 1x12 | 1 | Barra de Pinos 1x40 Fêmea 180 Graus |
| 5 | Barra de pinos macho 2x3 | 2 | Barra de Pinos 2x40 180 Graus (usado na top board) |
| 6 | Conector | 1 | Conector JST XH macho envolto |
| 7 | Chave 3 Terminais | 2 | Chave 3 Terminais Toggle Switch SPDT |
| 8 | Fototransistor | 4 | Fototransistor Receptor Infravermelho IR 5mm |
| 9 | Emissor Infravermelho | 4 | LED Emissor Infravermelho IR 5mm |
| 10 | Transistor | 3 | Transistor NPN BC337 |
| 11 | Mosfet | 1 | Mosfet P-Channel TO220 IRF4905 |
| 12 | Resistor | 1 | Resistor 33Ω 1/4W |
| 13 | Resistor | 2 | Resistor 47Ω 1/4W |
| 14 | Resistor | 3 | Resistor 680Ω 1/4W |
| 15 | Resistor | 4 | Resistor 1k8Ω 1/4W |
| 16 | Motor | 2 | 30:1 Micro Metal Gearmotor HP 6V com eixo do motor extendido |
| 17 | Par de Encoder Magnético | 1 | Par de encoder magnéticos para motor Micro Metal Gearmotors, 12 CPR, 2.7-18V |
| 18 | Par de rodas | 1 | Par de roda Pololu 32x7mm |
| 19 | Par de Motor Brackets | 1 | Motor Bracket - Impressão 3D |
| 20 | Conector IDC Fêmea | 2 | Conector IDC fêmea - 6 vias |
| 21 | Cabo Flat | 1 | Cabo Flat - 6 vias (1 m) |
| 22 | Bateria | 1 | Bateria Li-Ion 18650 3,7V 2200mAh 2C |
| 23 | Suporte para bateria | 1 | Suporte para bateria 18650 |

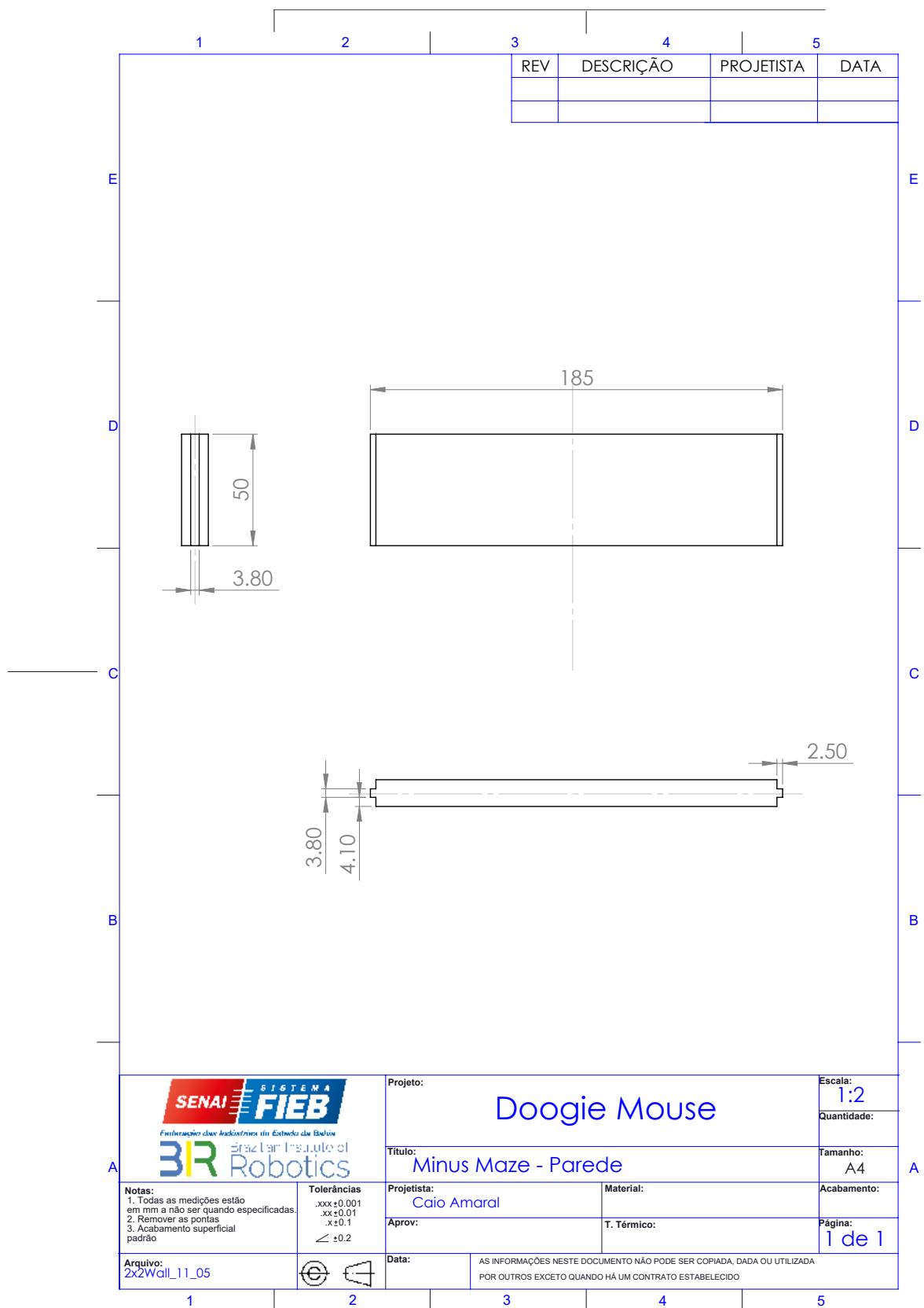
| Auxiliares | | | |
|-------------------|--------------------------|-------------|---|
| Ítem | Elemento | Qtd. | Descrição |
| 1 | Adaptador USB/Micro USB | 2 | Cabo adaptador USB para Micro USB |
| 2 | Adaptador HDMI/Mini HDMI | 1 | Adaptador HDMI Fêmea para Mini HDMI Macho |

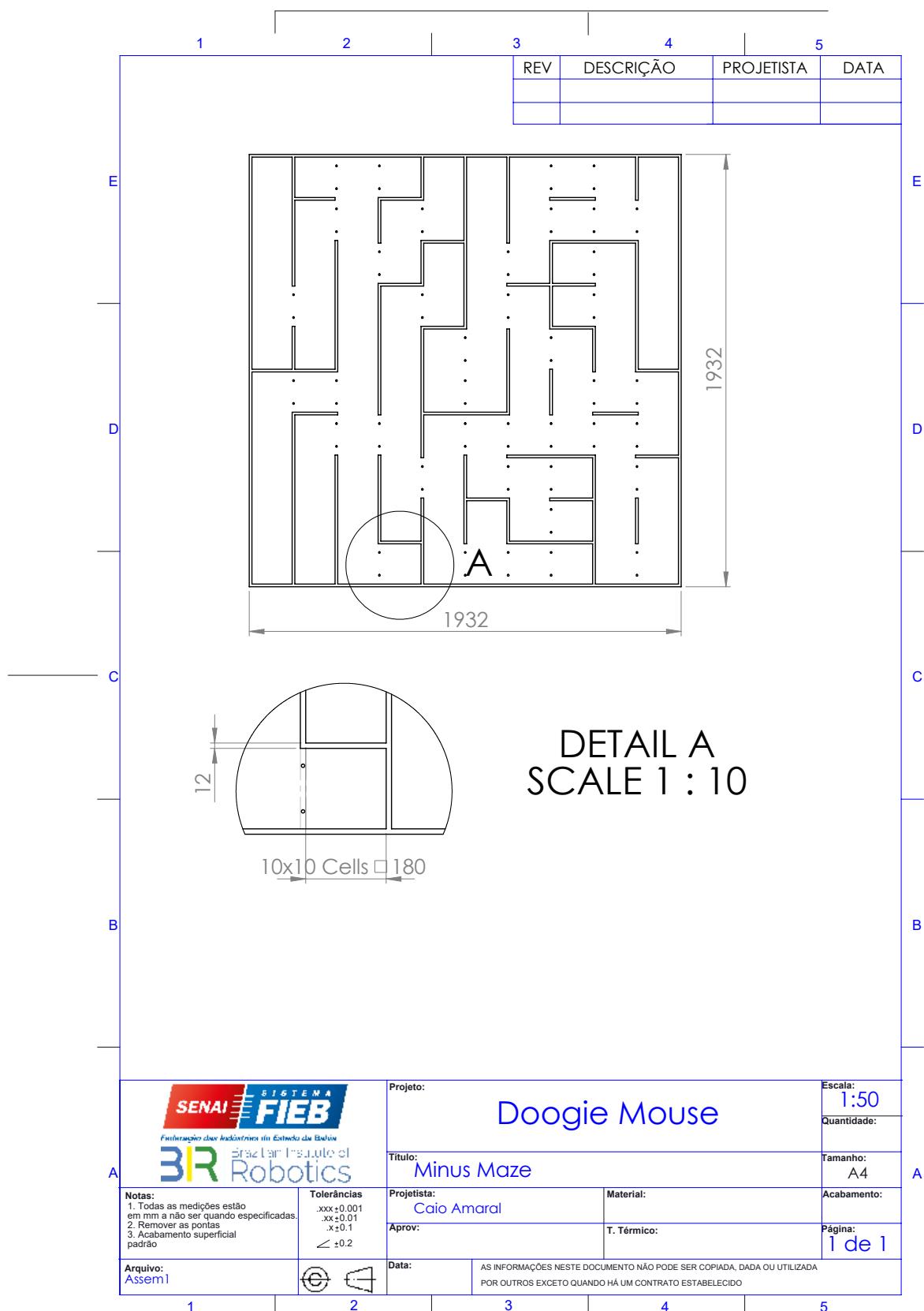
Desenhos técnico do Doogie Mouse

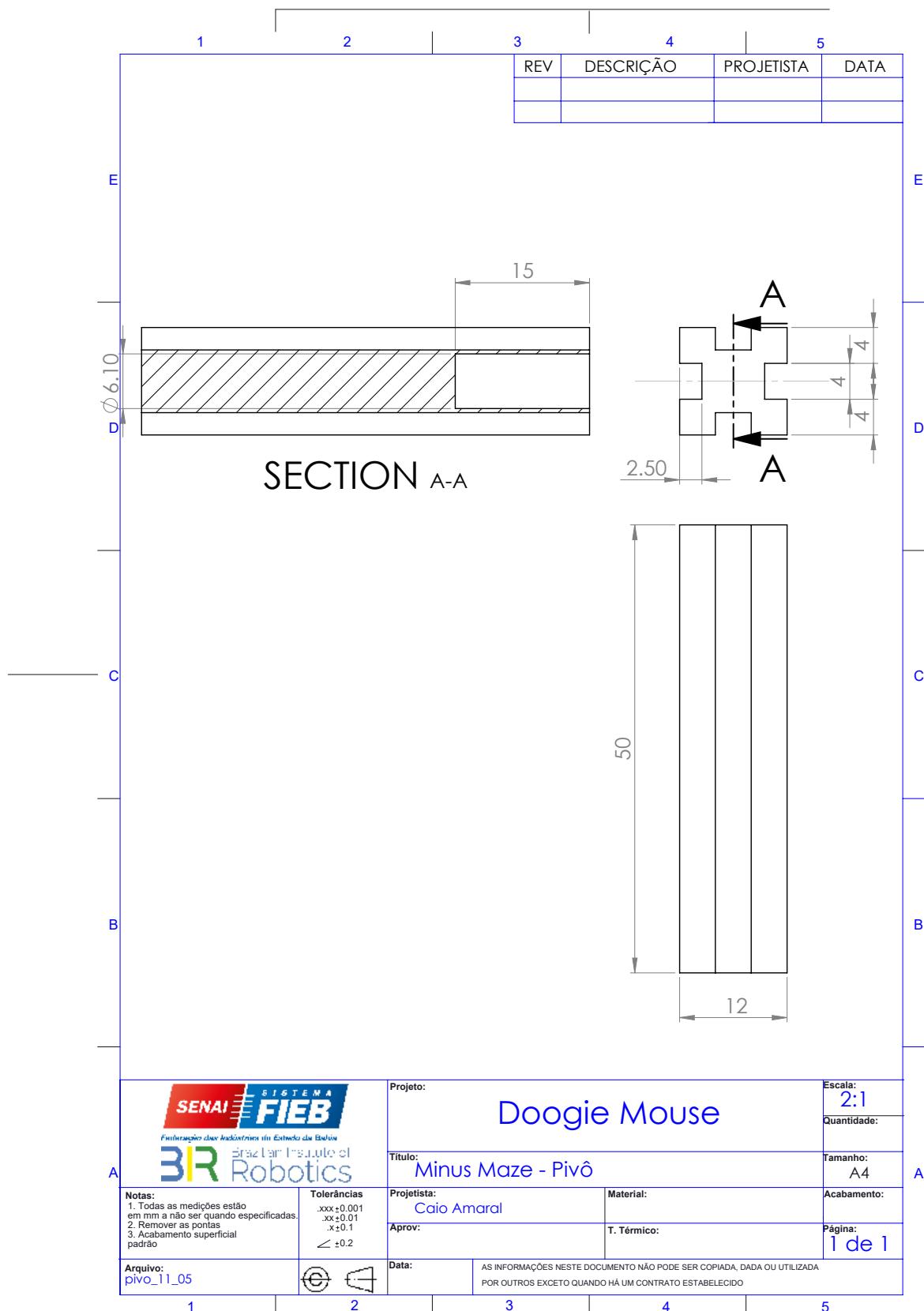




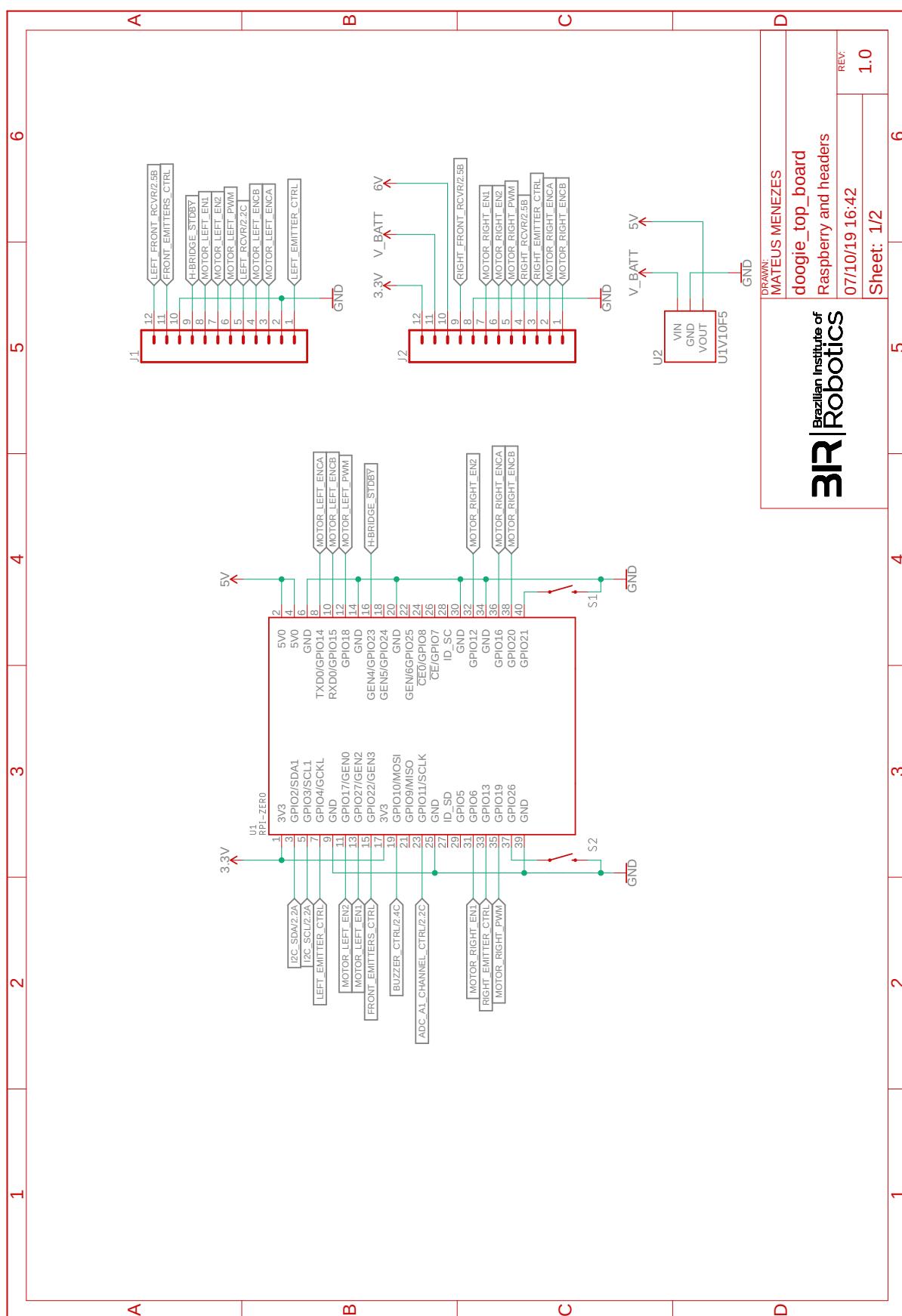
Desenhos técnico do labirinto

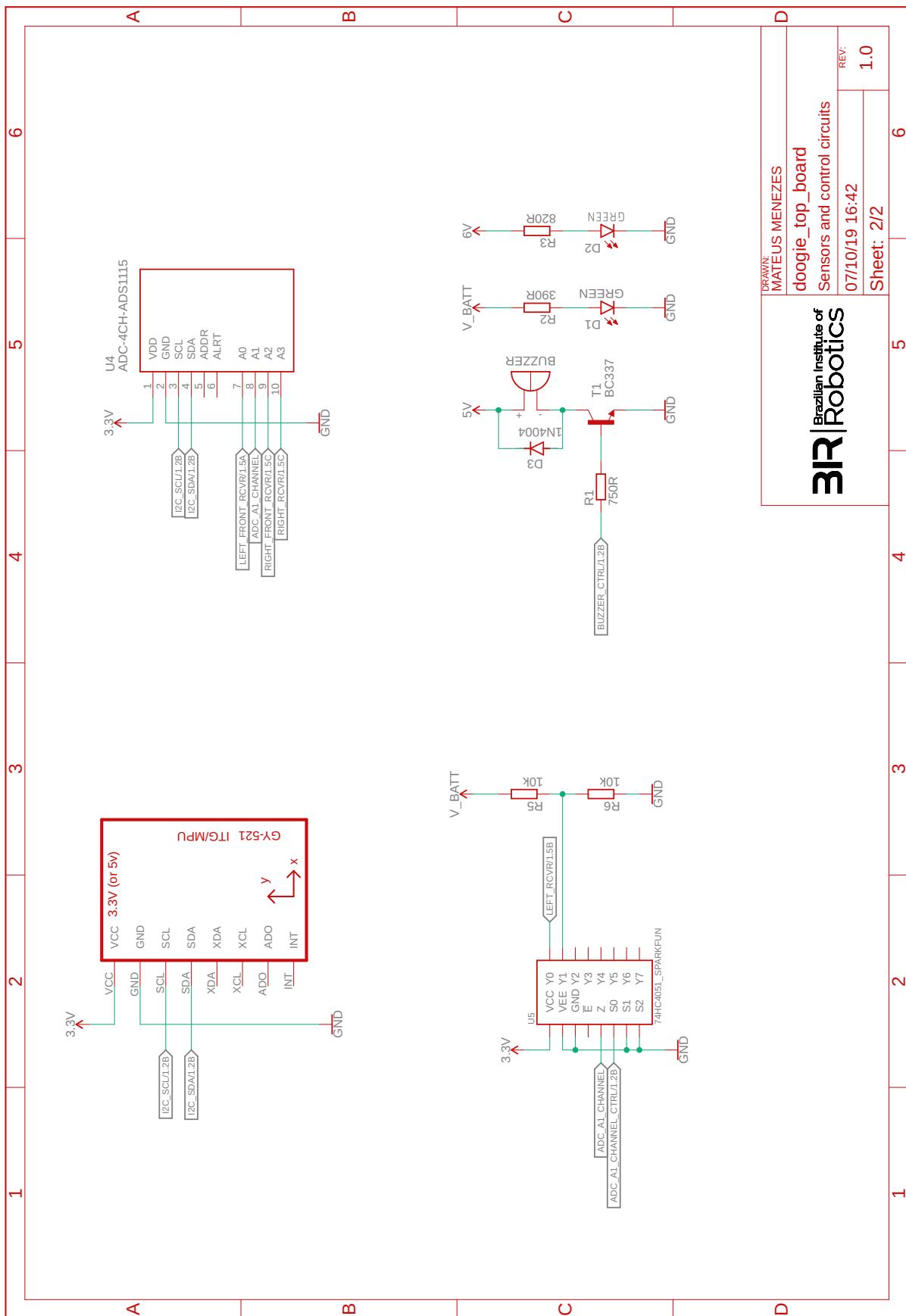


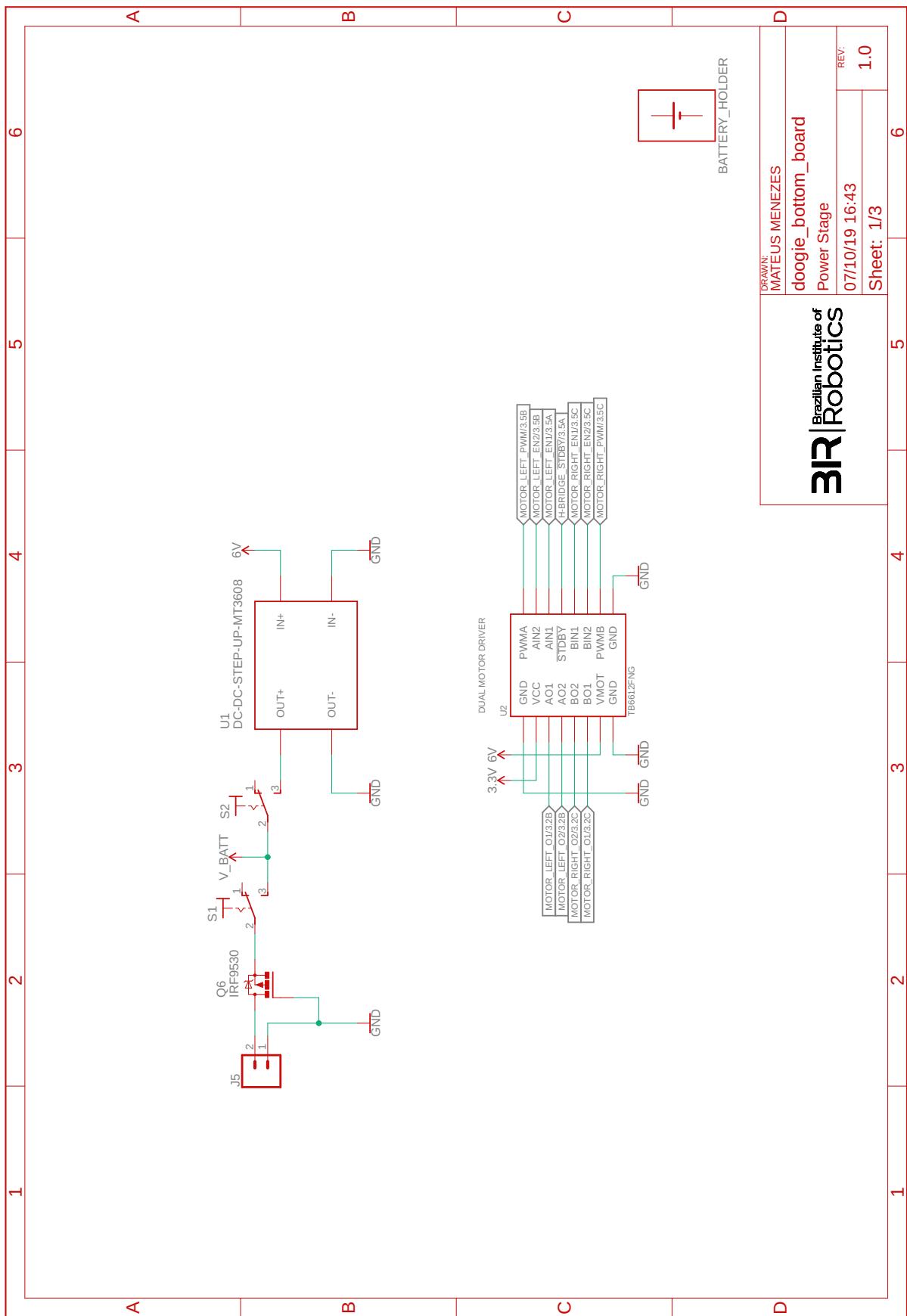


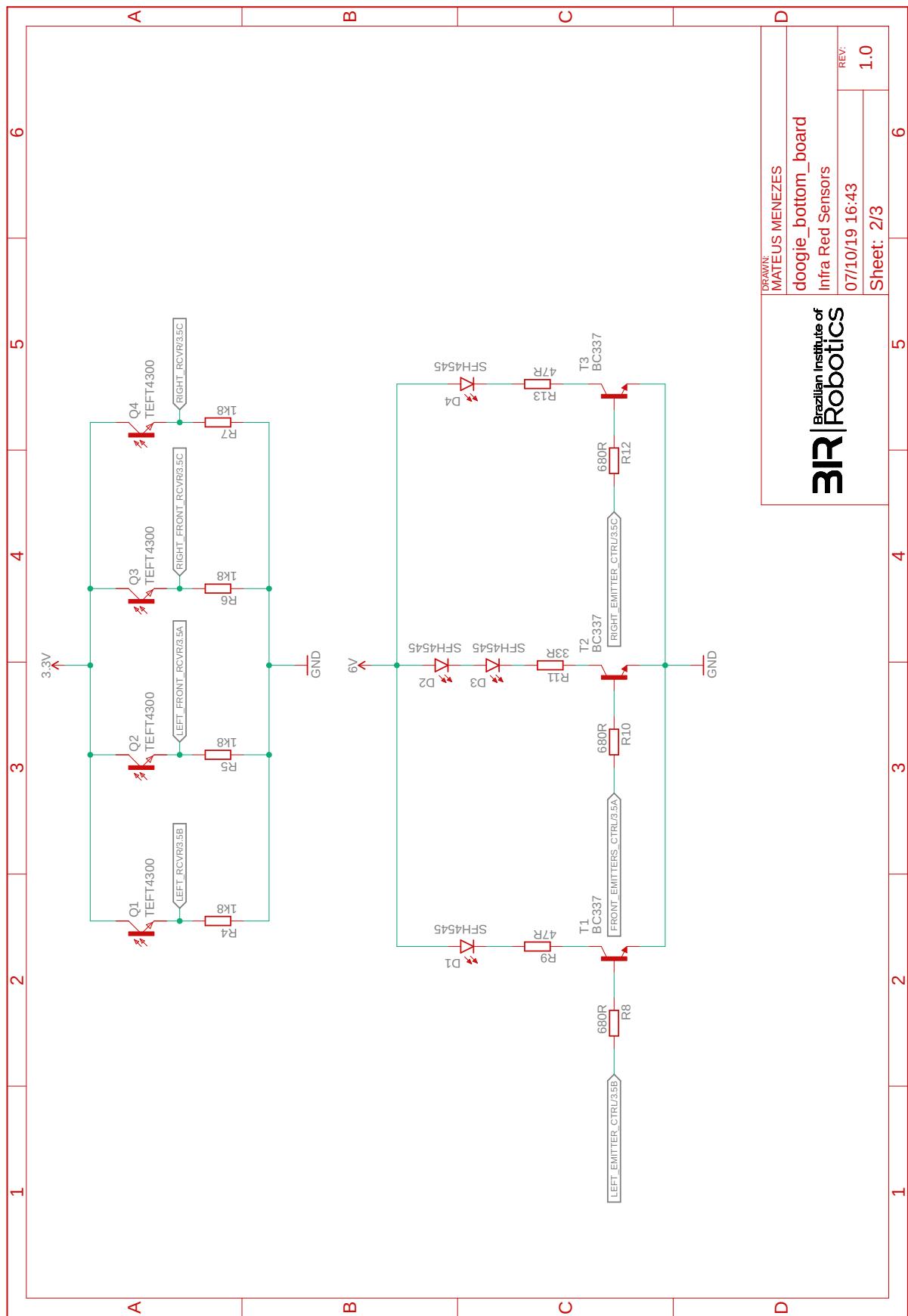


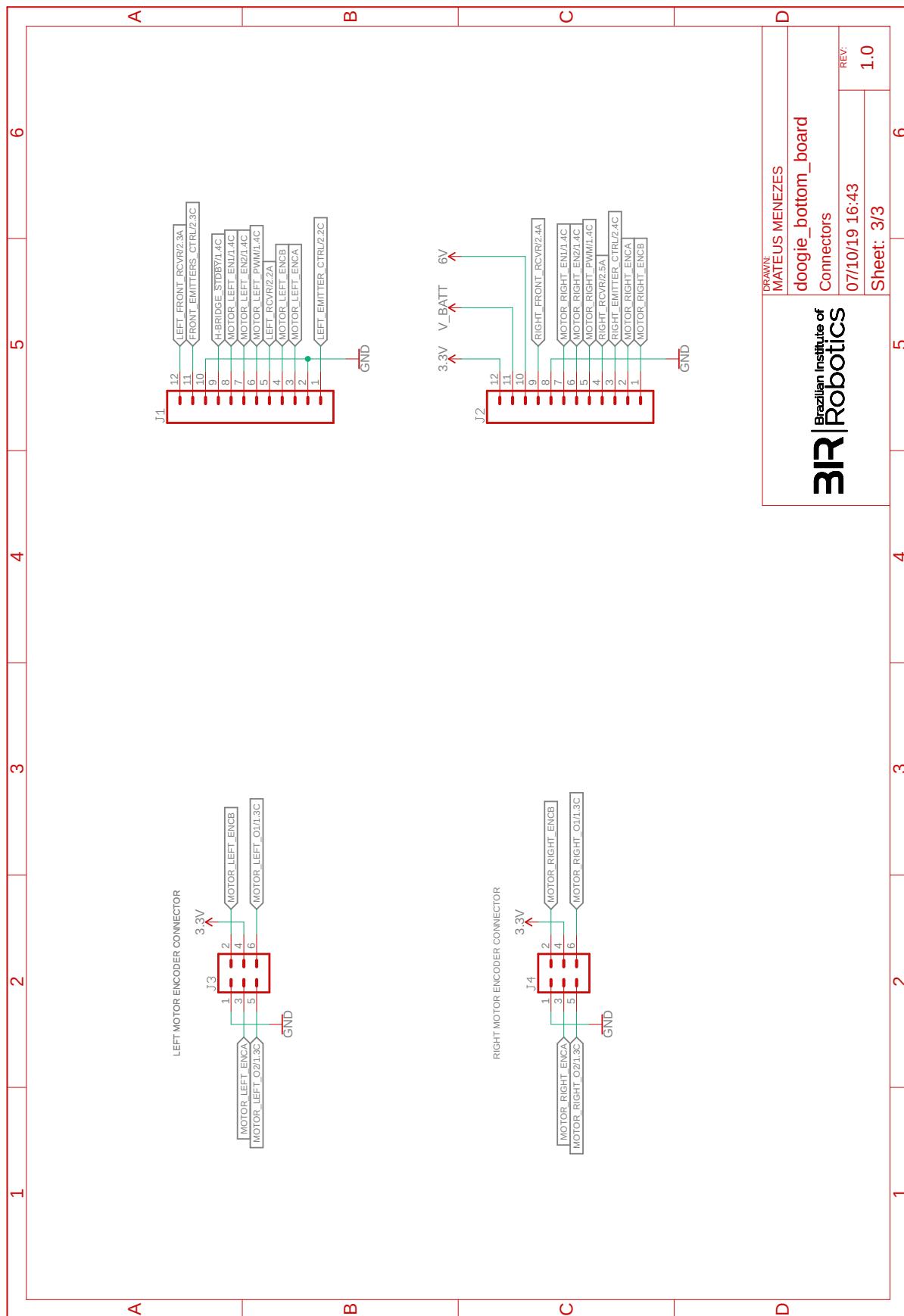
Esquemáticos eletrônicos











Guia de montagem do Doogie Mouse

README.md

12/5/2019

Doogie Hardware Setup

A repository with hardware setup instructions for Doogie Mouse robot.

Introduction

This document is intended to assist the process of assembly the Doogie Mouse robot as well as instructed about good practice on the procedure. This guide it going to promote an order of soldering of components for get more facility during the mount process. The guide it going to start the process with de bottom board and next it going to the top board.

Materials

Before starts the process you will need have the tools and security equipment:

- 01 Soldering Iron (prefer thin tip);
- 01 Tin roll of 0,3mm thin thread;
- 01 Cut plier;
- 01 Nose plier;
- 01 Protection glasses;
- 01 Pair of soldering gloves.

With the tools in hands you will need to have too the electronics components according the Figure 1 and the boards according the Figure 2.

Figure 1: Components used to assembly the Doogie Mouse robo

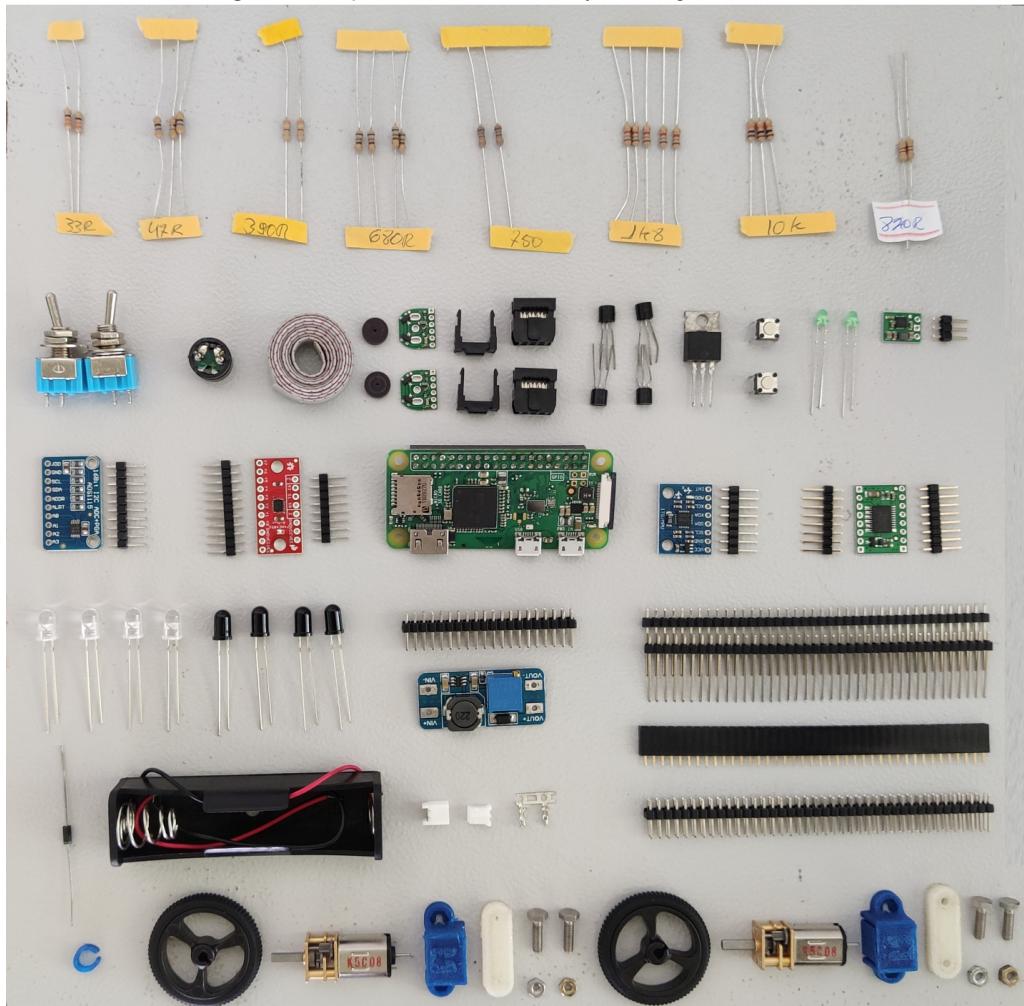
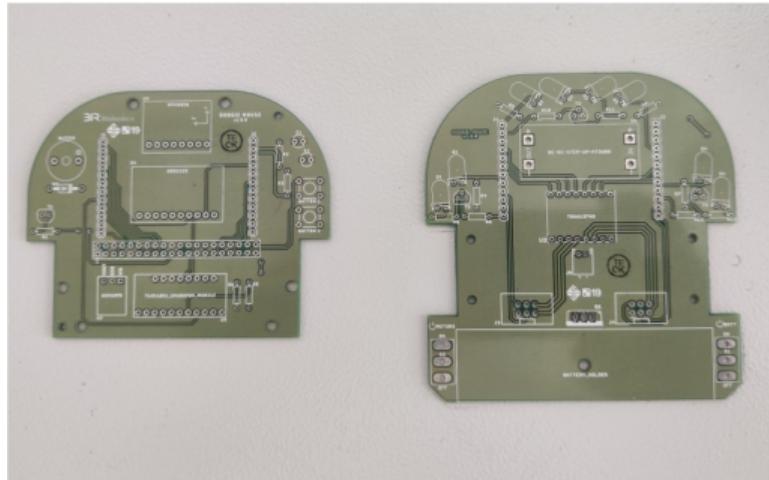


Figure 2: Top and bottom boards of Doogie Mouse robo

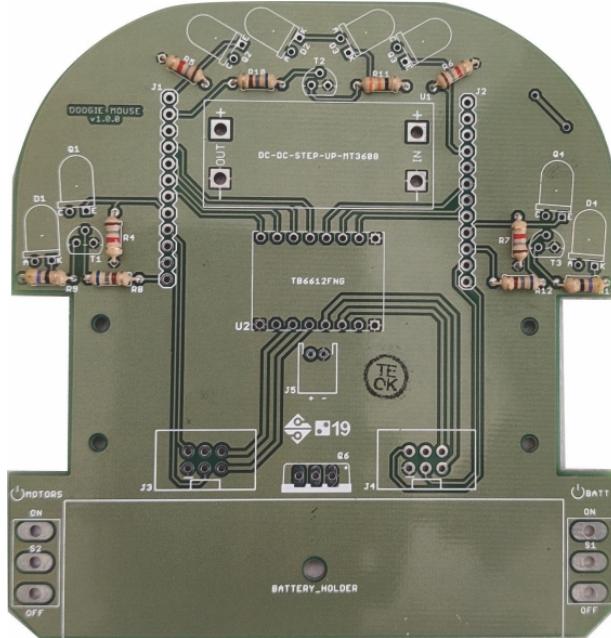


General Instructions

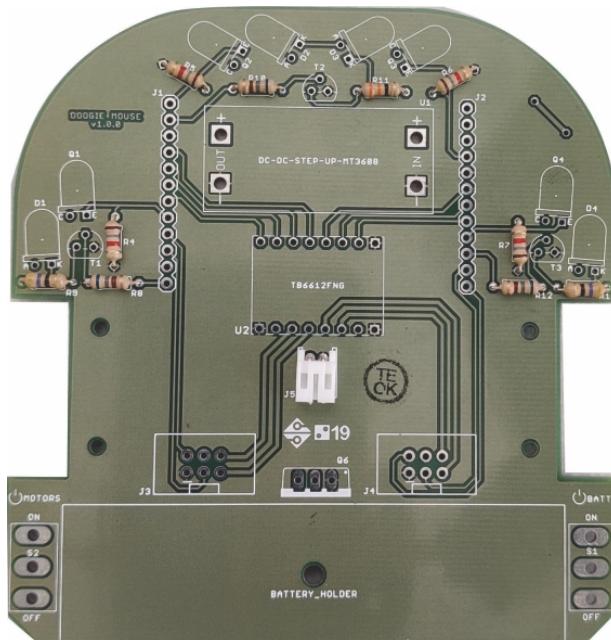
1. Every steps where there direct contact with metalic parts that will be welded, is advised use the nose plier for avoid accident for burnig.
2. After every steps maked take off the excess of the terminals of the eletronic components with help of cut plier.
3. Every process of componennts fixation will be maked with a Soldering Iron and tin.
4. Is advised make soldering process on a place with air flux for avoid the smoke resulted of the soldering process.
5. During all process you should to use the security glasses and soldering gloves for avoid accidents.

Bottom Board

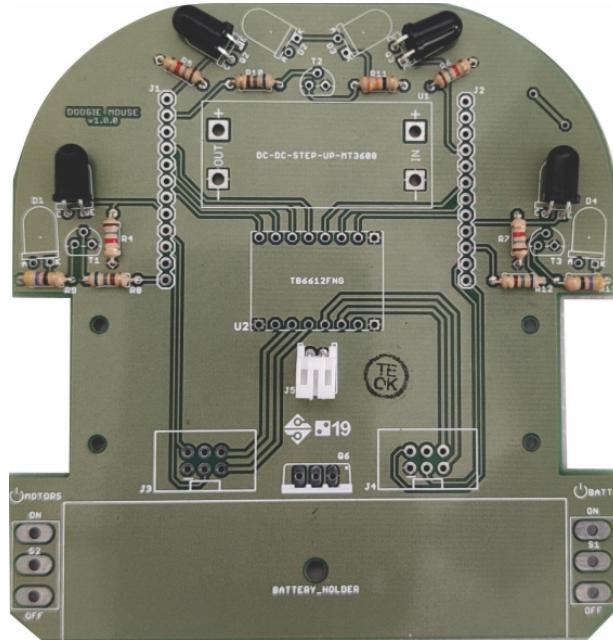
1. We are going to start the board assembly by the resistors, with help of soldering iron weld them on the board, look the picture below.



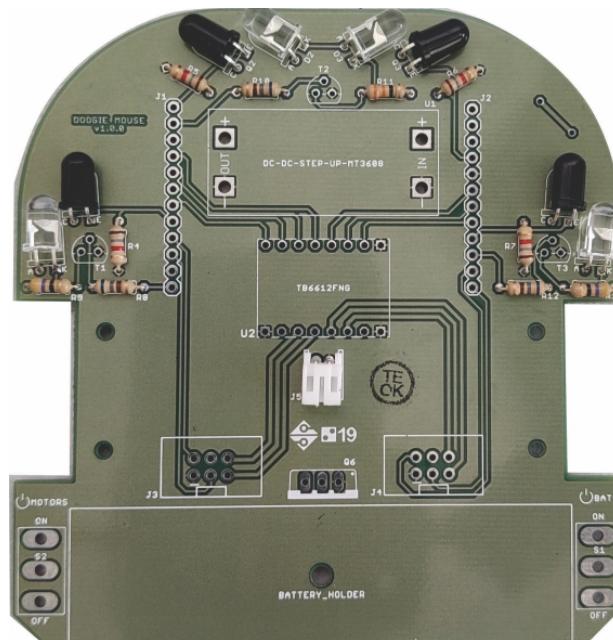
2. Now weld the batery connector.



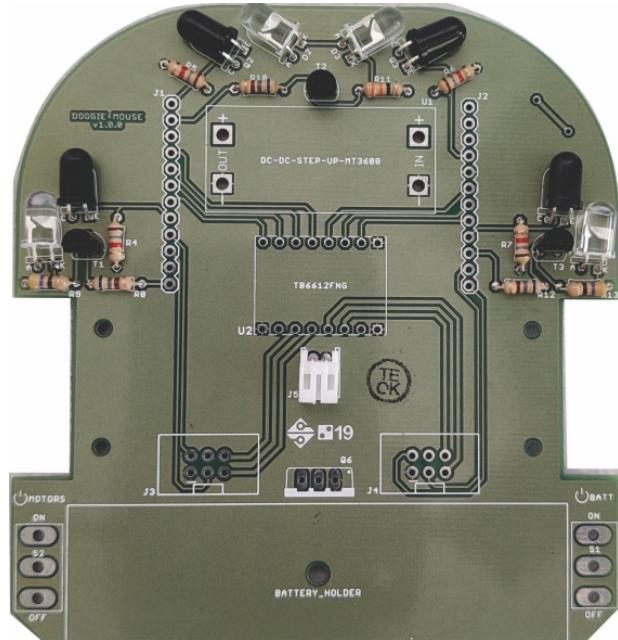
3. Put on and weld the emmiters leds of the infra red sensors.



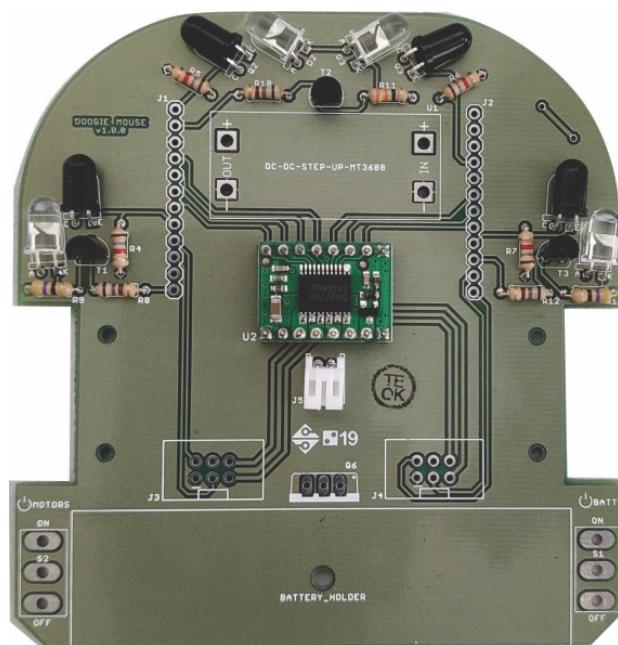
4. Next weld the receptors leds of the sensor.



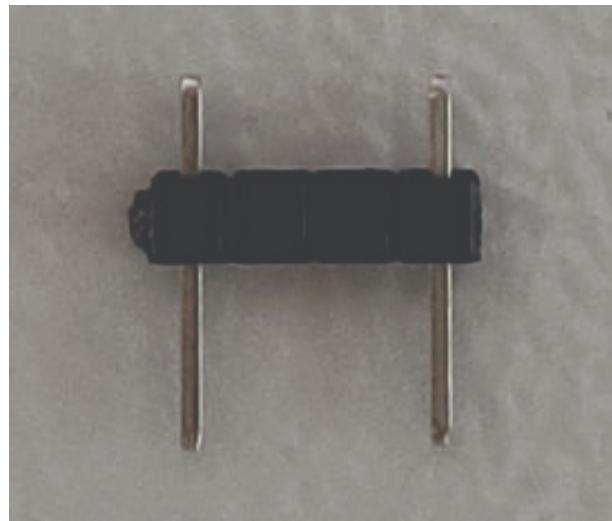
5. Weld the transistors on the board.



6. Now weld the H bridge.



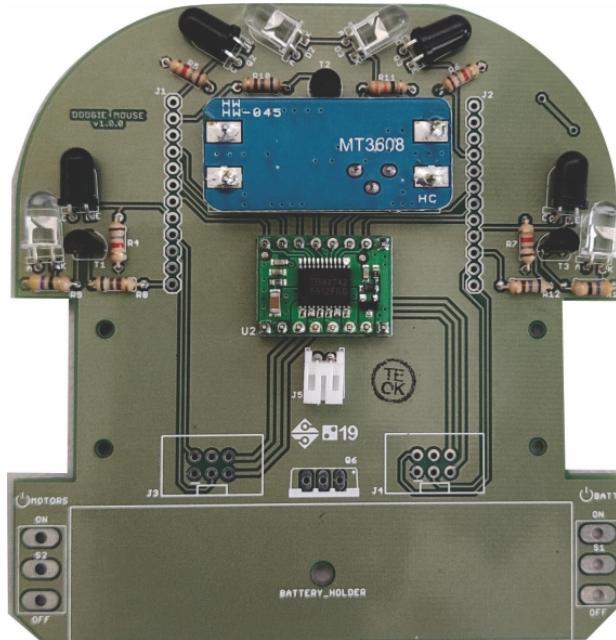
7. Using a pin bar and de cut pliers make two units of a new leaked pin bars according picture below .



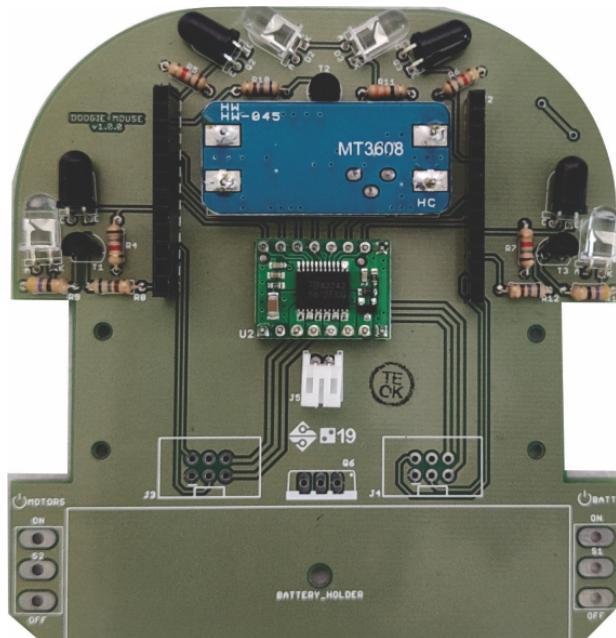
8. After make the leaked pin bars weld them on the extremity (VIN+/VIN- and VOUT+/VOUT-) of the electronic component, look the picture below.



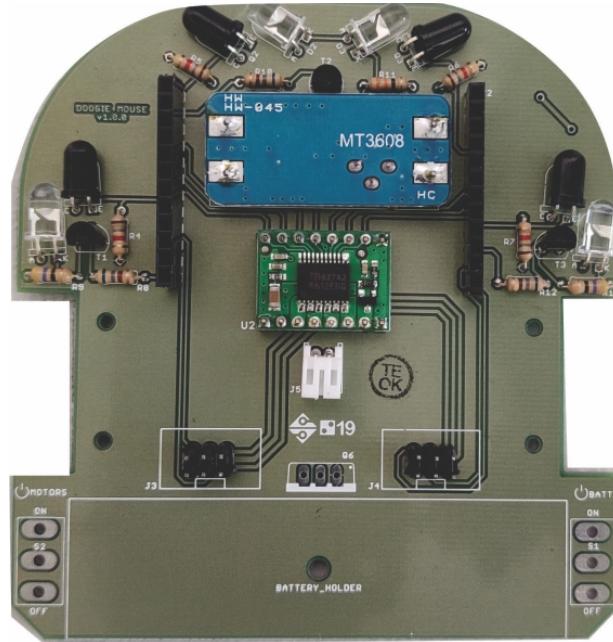
9. After the pins already to inserted in the tension regulator then weld them on de board.



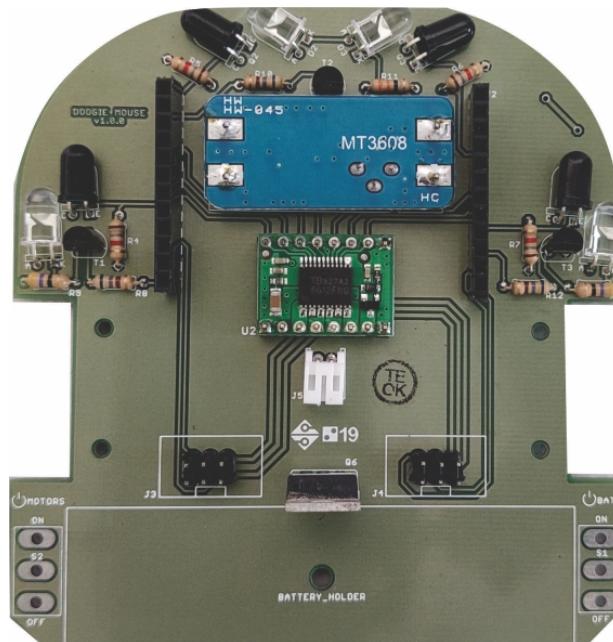
10. Weld the central pin bars on the board, use the nose plier as a helper tool.



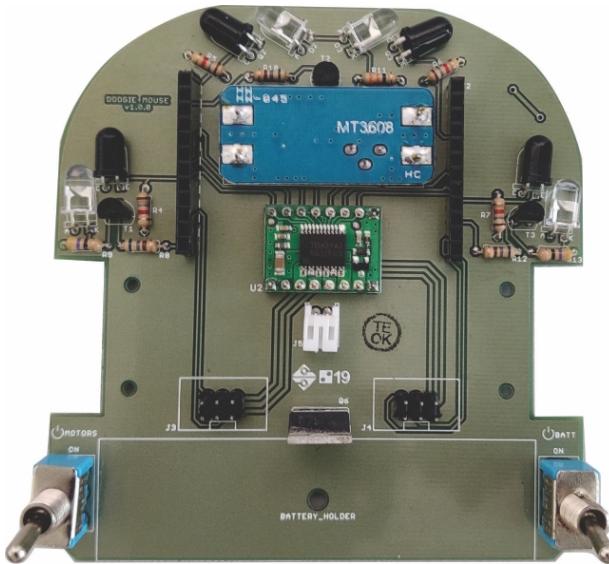
11. Weld the pin bars on the encoder and next secure the component on the board with de soldering.



12. After this attach the mosfet on the board with help of soldering iron.



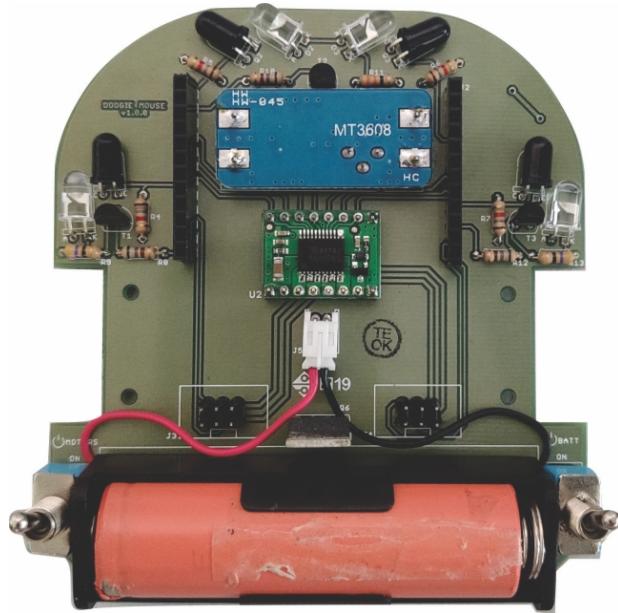
13. Then connect and weld the key (ON/OFF) on the board.



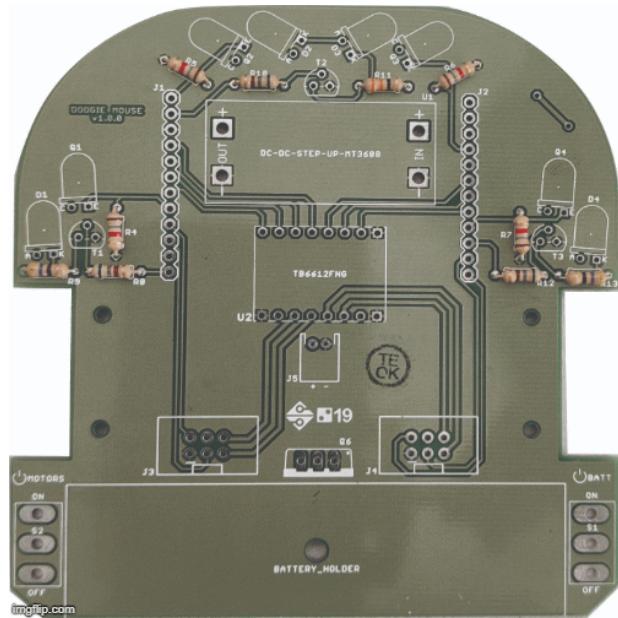
14. In this step weld the crimps and make the battery male connector.



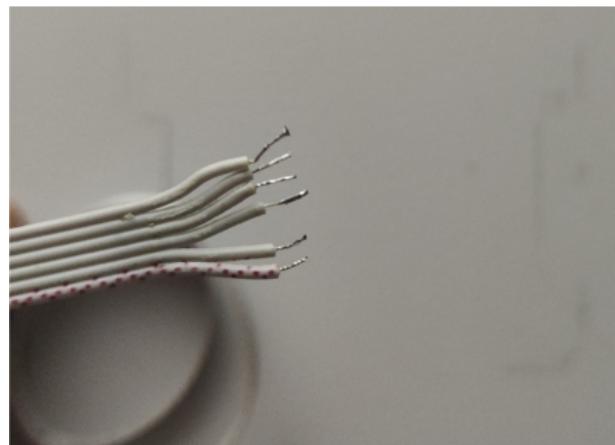
15. Now with the help of a screw nut and washer, hold the baterry support on the board.



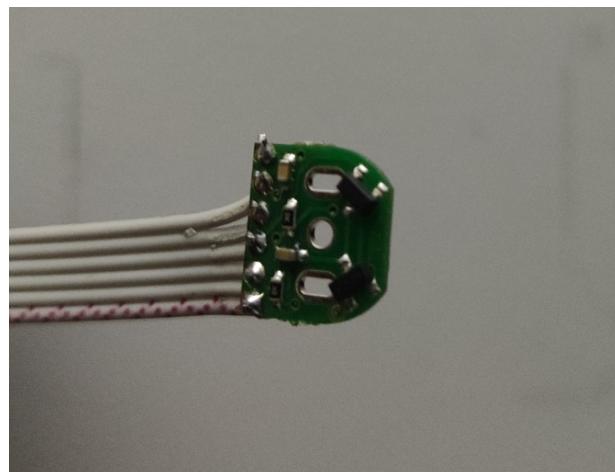
Look the full bottom board mount process in the GIF below.



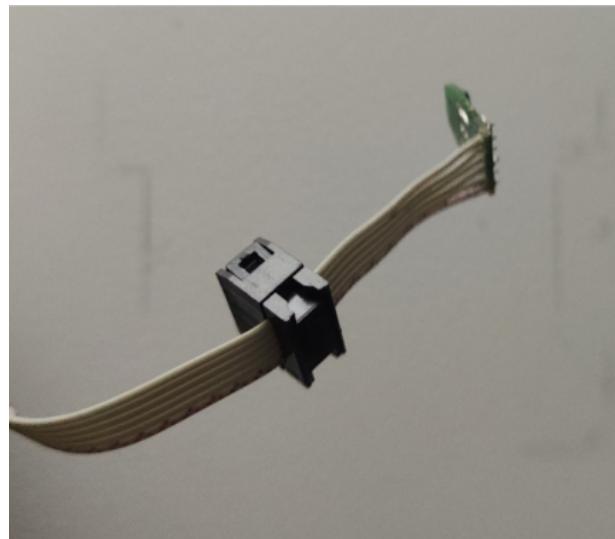
16. Now we going to start the confectionate of motor set. First tin the end of flat cable.



17. After this weld the flat cable on the magnetic encoder board. Look the picture below.



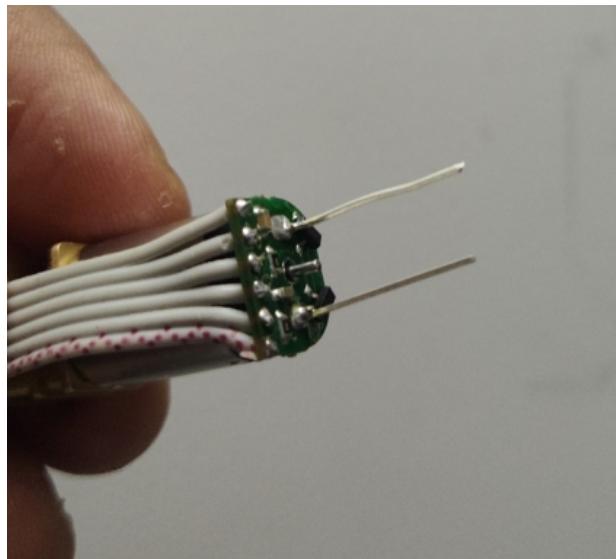
18. Then make the fixation of the female connector IDC. Be sure that the lenght of the cable are 25 mm .



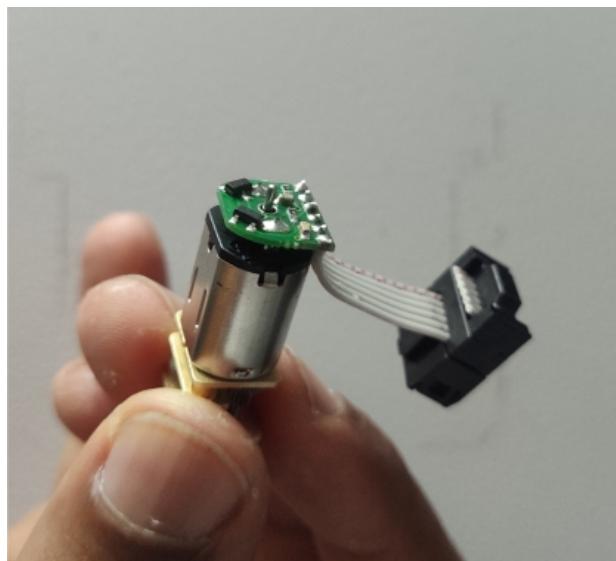
19. Next cut the excess cable.



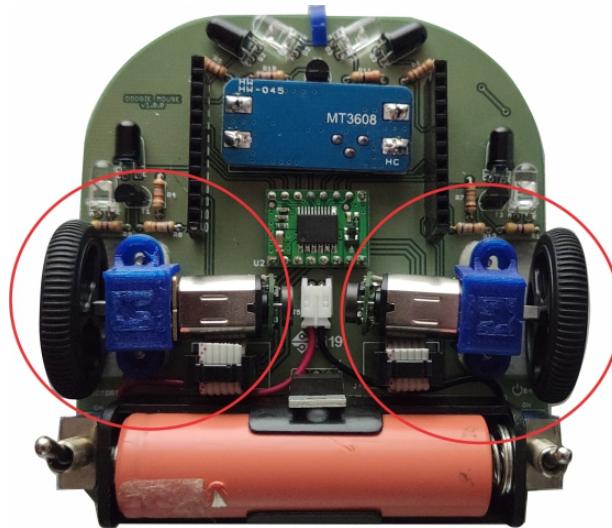
20. Use a metal guide for able weld the motor terminals on the magnetic encoder board. Pay attention on the polarity(M1 = + | M2 = -).



21. Cut the excess of the metal guide.



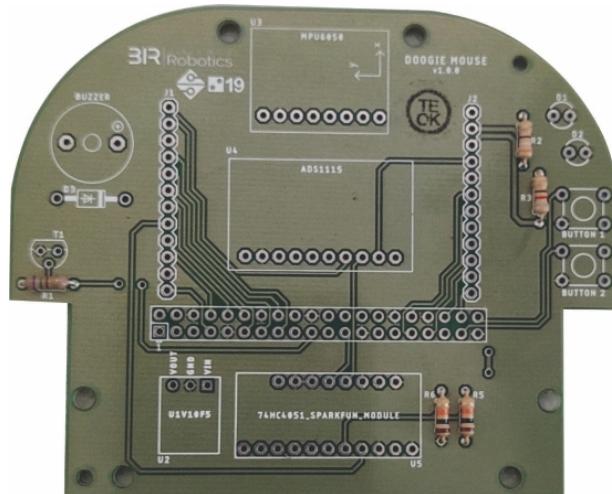
22. With help of bracket, screw and nut make the fixation of motor on the bottom board.



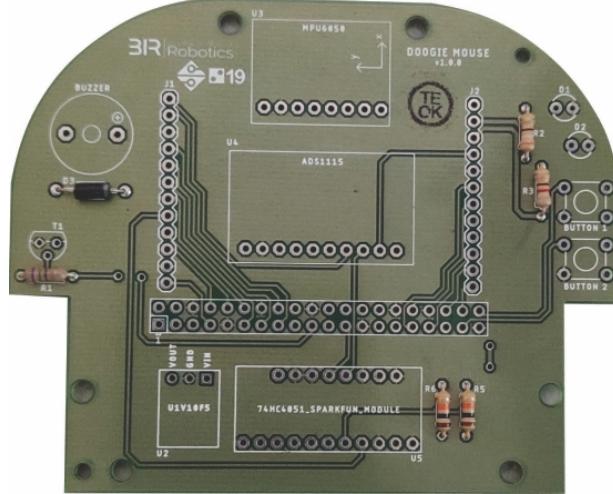
In this step the mount bottom board procedure is finished we will be now for the mount top board procedure.

Top Board

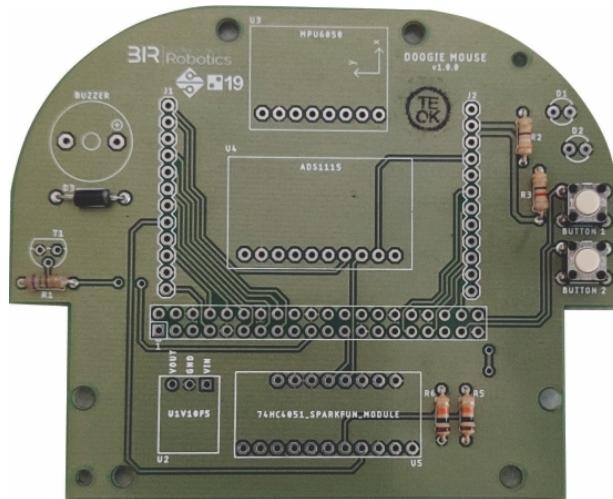
1. With the help of the soldering iron and tin connect and secure the resistors.



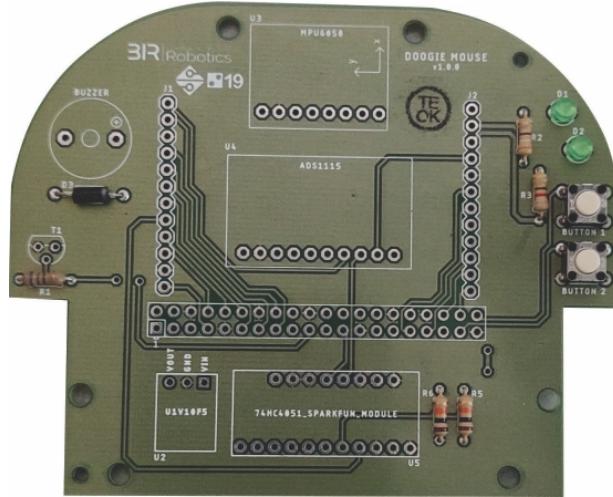
2. Weld the diode.



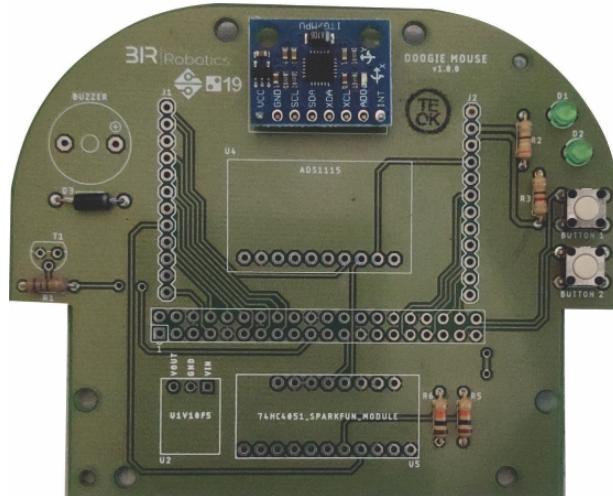
1. Then make the fixation and soldering of the push-buttons.



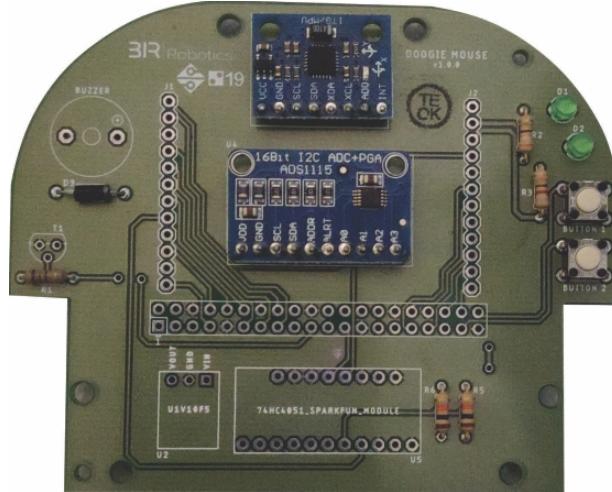
4. After this weld the leds on the right top corner of the board.



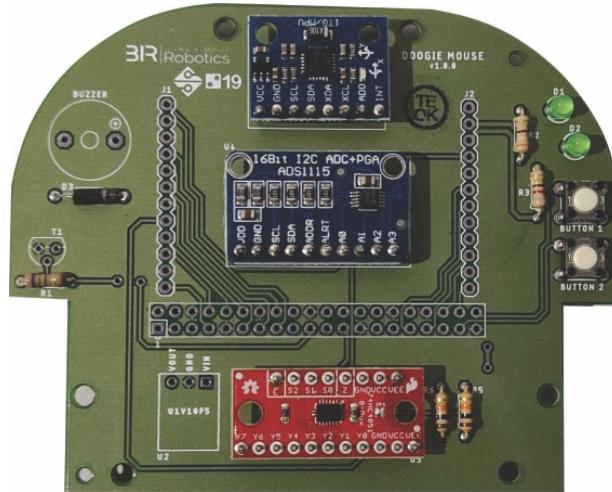
5. Weld the pin bars of the IMU on the component and then hold them on the board with soldering.



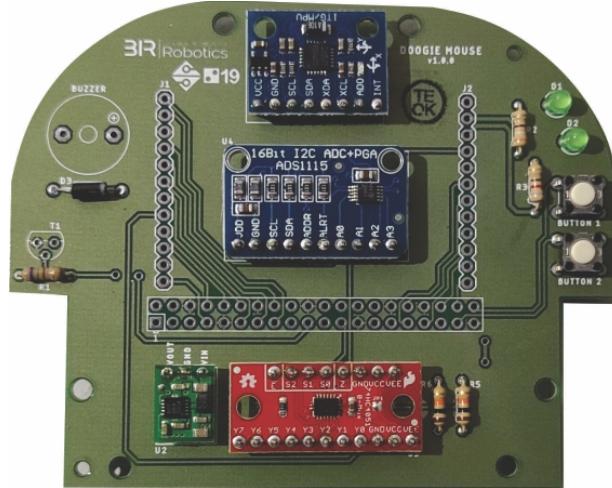
6. Like the previous step weld de pin for ADS then hold them on the board with soldering.



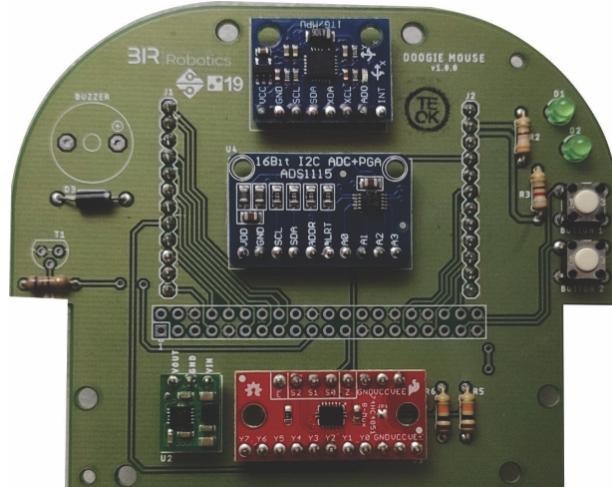
7. Of similar form make for the multiplexer.



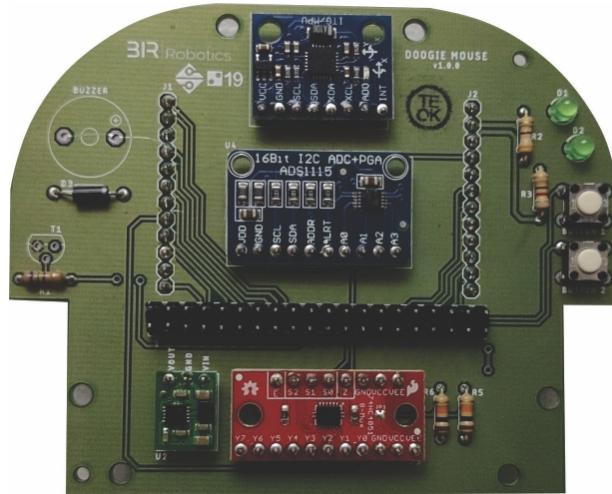
8. Repeat the step for the DC/DC conversor.



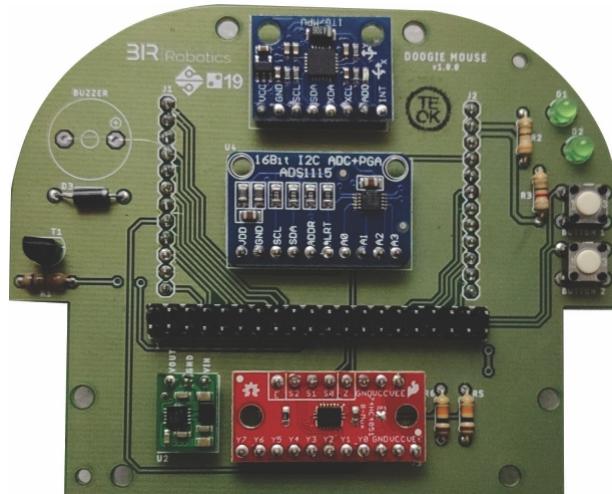
9. Weld the pin headers on the central connectors making the bigger part stay facing for the under board side side.



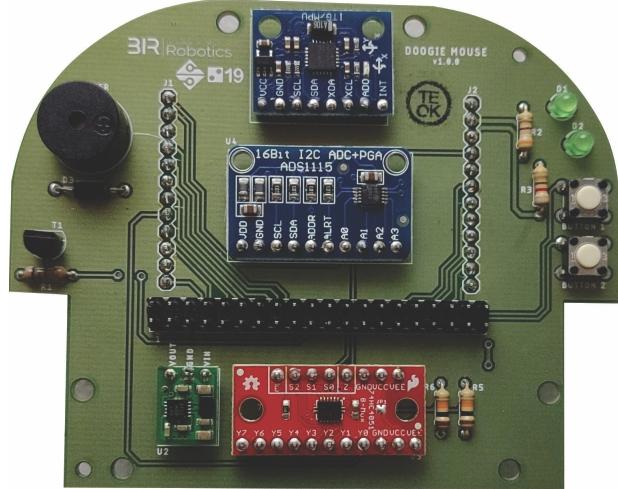
10. In this step weld the pin bar on the top board.



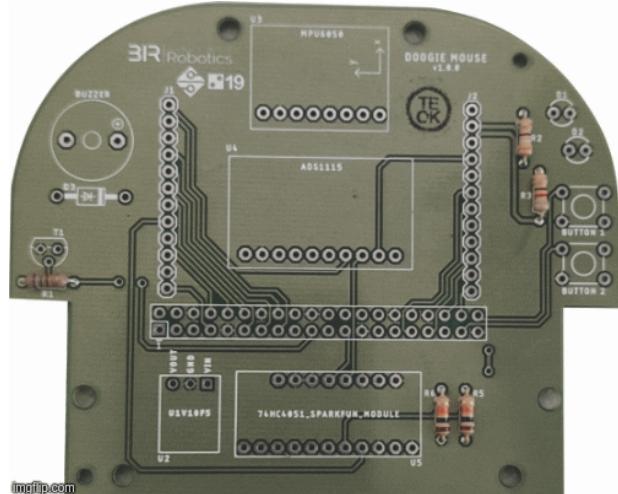
11. Now you will have to make the fixation of the transistor with soldering.



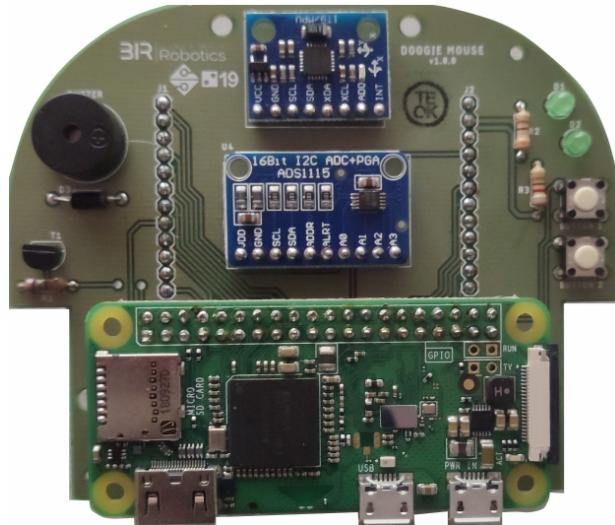
12. Weld the buzzer.



Look the full top board mount process in the GIF below.



13. For finish weld the female pin bar on de raspberry Pi zero and connect on the top board.



Guia de configuração de software do Doogie Mouse

README.md

12/5/2019

Doogie Software Setup

A repository with software setup instructions for Doogie Mouse robot.

Introduction

This repository will help you on the setup of the Raspberry Pi Zero W v1.1 which is used in the Doogie Mouse robot. These instructions assume that you have already the Raspberry peripherals such as mouse, keyboard and monitor connected on it. See the [hardware_instructions](#) or [oficial Raspberry instructions](#) for more informations.

These instructions package has been made under Ubuntu 16.04 LTS.

How to Read?

The notes are more or less in chronological order, hence start from top the bottom. Commands are prefixed with the system on which the commands needs to be run:

- **HOST:~\$** commands executed on your machine
- **RPI:~\$** commands executed on the Raspberry Pi

Prepare microSD card

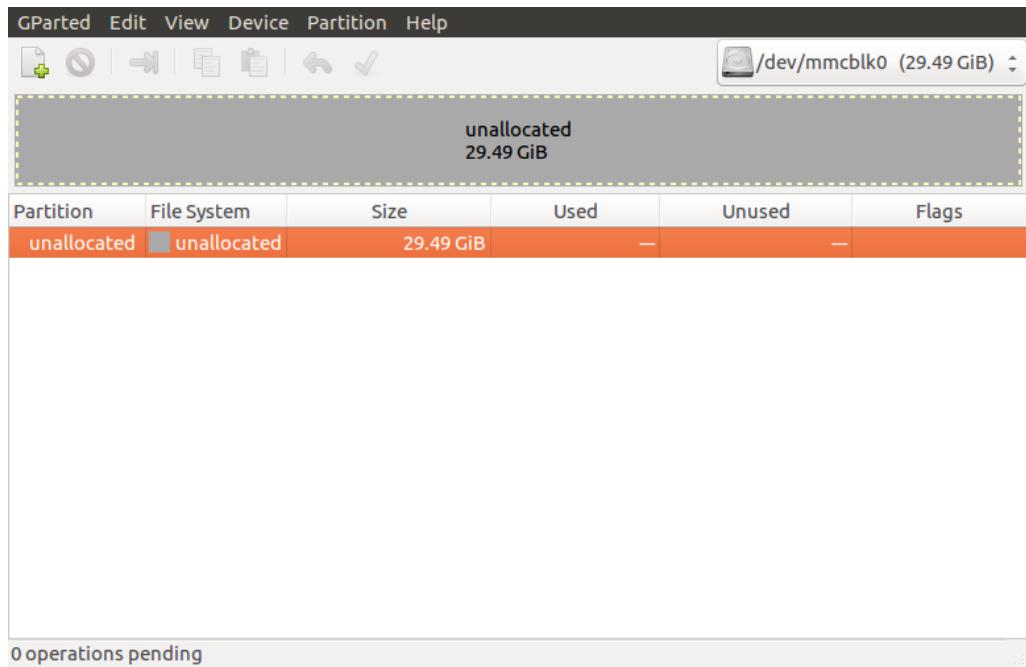
The first step is to format the microSD card. If you are using an old microSD card, remember to do the backup of the media or you will lost all data. To format the card, install and run the GParted (GNOME Partition Editor):

```
HOST:~$ sudo apt-get install gparted  
HOST:~$ sudo gparted
```

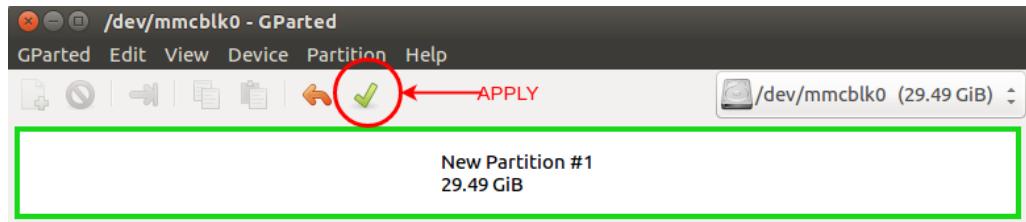
Delete all partitions, if any. You will see the image below.

README.md

12/5/2019



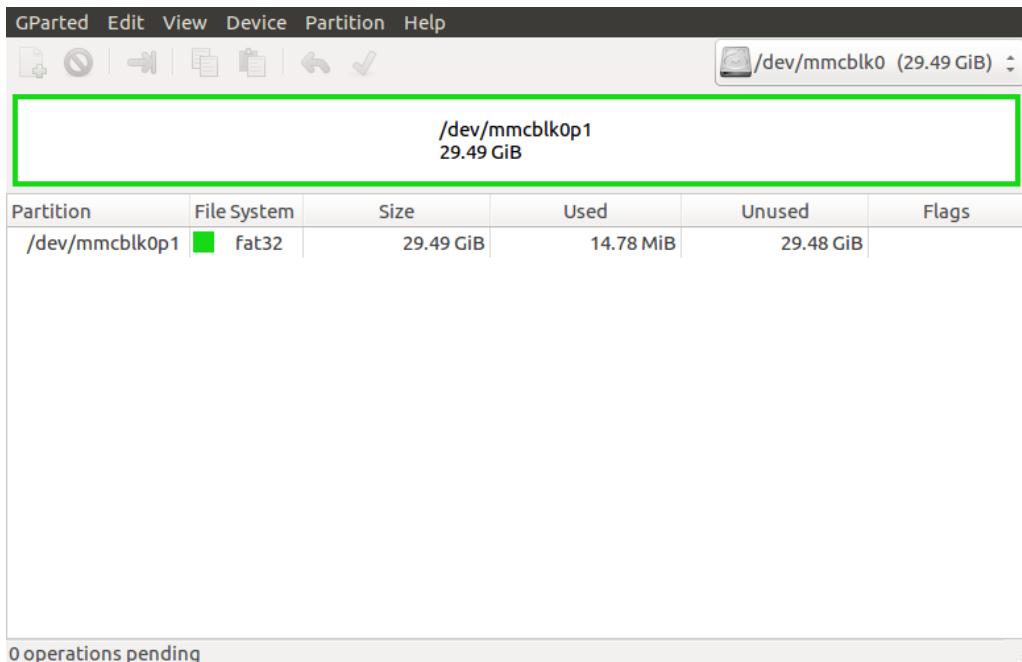
After that, create a new partition: right-click->New. Create the partition with the hole space. In the File System, choose FAT32, then click in add button and Apply All Operation.



Then the gparted will be like this:

README.md

12/5/2019



See [Installing operating system images on Linux](#) for more informations. To finish, unplug and plug the card in your computer.

Raspbian OS Installation

To begin, download the last version of the Raspbian Jessie OS. In our case, download the [Raspbian Jessie Full](#) version and unzip it.

If you want download the image using torrent, Ubuntu distribution has a native torrent client called Transmission BitTorrent Client

To copy the image to the microSD card

```
HOST:~$ sudo umount /dev/mmcblk0p1
HOST:~$ cd IMG_FOLDER
HOST:~$ sudo dd bs=4M status=progress if=IMG_FILE_NAME.img of=/dev/mmcblk0
```

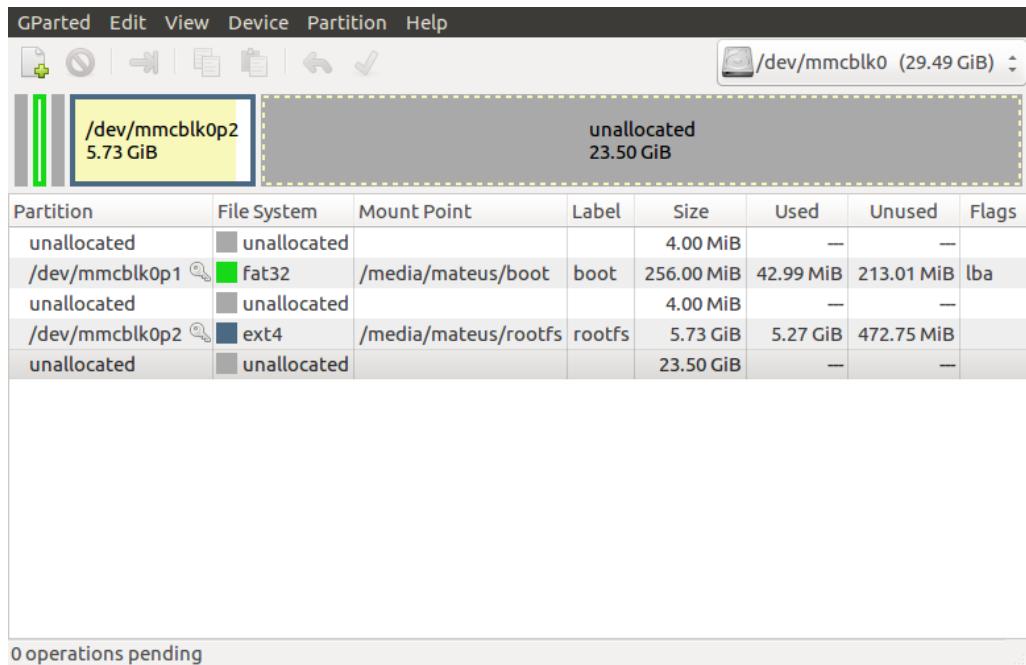
In the end of the process, you'll see

```
6425673728 bytes (6,4 GB, 6,0 GiB) copied, 253,09 s, 25,4 MB/s
1534+0 records in
1534+0 records out
6434062336 bytes (6,4 GB, 6,0 GiB) copied, 335,209 s, 19,2 MB/s
```

README.md

12/5/2019

Again, unplug and plug the card in your computer. To check if the process occurred correctly, open the gparted and you'll see the partitions of the card as the image below.



Remove the microSD card and then plug into the Raspberry.

Connect your Raspberry Pi

Make sure at least the keyboard and the monitor are connected to the Raspberry Pi. Now, power on the board using its power pins (5 V and GND) or the USB power port.

Finish the setup

When you start your Raspberry Pi for the first time, some configuration should be made. Follow the steps below to set up Raspberry:

1. On the display upper left corner, click in the Raspberry icon
2. Go to [Preferences >> Raspberry PI Configuration](#)
3. On [System](#) window, change the fields:
 - o [Password](#) (current is raspberry). Recommended [doogiemouse](#) for new password
 - o [Hostname](#). Recommended [doogie-mouse](#). **Note:** If multiples Doogie Mouse robots will be used, we recommend use a different second name, e.g., [doogie-brain](#) for the first robot and for the second robot, [doogie-pinky](#). Feel free to choose the best names...
4. On [Interfaces](#), enables SSH and I2C
5. On [Localisation](#)
 - o [Set Locale... >> Country](#). Change to [US \(USA\)](#)
 - o Set the other fields ([Timezone](#), [Keyboard](#) and [WiFi Country](#)) with your preference
6. Click in [Ok](#) to finish this setup. **Note:** Do not reboot yet!!!.

README.md

12/5/2019

7. Connect the Raspberry to an WiFi clicking on its icon on upper right corner of display
8. Click in the Raspberry icon (see step 1), **Shutdown... >> Reboot** to finish the setup

Setup remote access - SSH (Secure Shell)

Make sure your Raspberry Pi is properly set up and connected. You will need to note down the IP address of your Pi in order to connect to it later. Using the `ifconfig` command in a terminal will display information about the current network status, including the IP address, or you can use `hostname -I` to display the IP addresses associated with the device.

Skip the steps below if the SSH is already enabled

Enable SSH

1. Launch **Raspberry Pi Configuration** from the **Preferences** menu
2. Navigate to the **Interfaces** tab
3. Select **Enabled** next to **SSH**
4. Click **OK**

Alternatively, `raspi-config` can be used in the terminal:

1. Enter `sudo raspi-config` in a terminal window
2. Select **Interfacing Options**
3. Navigate to and select **SSH**
4. Choose **Yes**
5. Select **OK**
6. Choose **Finish**

SSH using Linux or Mac OS

You can use SSH to connect to your Raspberry Pi from a Linux computer, a Mac, or another Raspberry Pi, without installing additional software. You will need to know your Raspberry Pi's IP address to connect to it. To find this, type `hostname -I` from your Raspberry Pi terminal.

To connect to your Pi from a different computer, copy and paste the following command into the terminal window but replace `<IP>` with the IP address of the Raspberry Pi. Use `Ctrl + Shift + V` to paste in the terminal.

```
HOST:~$ ssh pi@<IP>
```

When the connection works you will see a security/authenticity warning. Type yes to continue. You will only see this warning the first time you connect.

```
The authenticity of host <IP> (<IP>) can't be established.  
ECDSA key fingerprint is  
SHA256:KKQF9QAU1NOFlokyQcqdRiTd9m0hck0/lcYpuJeCbqk.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added <IP> (ECDSA) to the list of known hosts.
```

README.md

12/5/2019

```
In the event your Pi has taken the IP address of a device to which your computer has connected before (even if this was on another network), you may be given a warning and asked to clear the record from your list of known devices. Following this instruction and trying the ssh command again should be successful.
```

Next you will be prompted for the password for the pi login. You should now be able to see the Raspberry Pi prompt, which will be identical to the one found on the Raspberry Pi itself.

X-forwarding

You can also forward your X session over SSH, to allow the use of graphical applications, by using the `-Y` flag:

```
HOST:~$ ssh -Y pi@<RASPBERRY_IP>
```

Now you are on the command line as before, but you have the ability to open up graphical windows. For example, typing:

```
geany &
```

For detailed explanation, see [SSH using Linux or Mac OS](#).

SSH Trick

Currently, you need to type your password each time you connect with the RPi. With the use of ssh-keys, we can automate this process.

1. Generate ssh-keys in the VM.

```
HOST:~$ cd ~/.ssh
HOST:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/<YOUR_USER>/.ssh/id_rsa):
doogie_mouse_rsa
Enter passphrase (empty for no passphrase): <empty>
Enter same passphrase again: <empty>
Your identification has been saved in rpizero_rsa.
Your public key has been saved in rpizero_rsa.pub.
...
```

Optional you can choose a different rsa-name (required if you are planning to use multie keys for different systems) and set a passphrase (increasing security). In my setup I left the passphrase empty (just hitting enter).

2. Set correct permissions of the key-set

README.md

12/5/2019

```
HOST:~$ chmod 700 doogie_mouse_rsa doogie_mouse_rsa.pub
```

3. Send a copy of the public key to the RPi so it can verify the connection

```
cat ~/.ssh/doogie_mouse_rsa.pub | ssh pi@<RASPBERRY_IP> "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"
```

4. Configure ssh connection in `ssh_config`

```
HOST:~$ sudo gedit /etc/ssh/ssh_config
```

Depending on the configuration of `dhcpcd.conf` on the RPi, add the following lines in the file end:

```
Host doogie-mouse
  HostName <RASPBERRY_IP>
  IdentityFile ~/.ssh/doogie_mouse_rsa
  User pi
  Port 22
```

5. Allow bash to invoke the configuration upon a ssh-call

```
HOST:~$ ssh-agent bash
HOST:~$ ssh-add ~/.ssh/doogie_mouse_rsa
```

6. Test connection:

```
HOST:~$ ssh -Y doogie-mouse
```

You should now be logged in onto the Raspberry Pi via SSH, without entering your password.

Installing ROS

Increase Swap Space

In order to ensure there is enough swap space to compile ROS, make the following modifications below, then reboot the Raspberry Pi Zero so the change takes affect.

```
RPI:~$ sudo nano /etc/dphys-swapfile
```

README.md

12/5/2019

and set the variable **CONF_SWAPSIZE=1024**.

Setup ROS Repositories

```
RPI:~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu buster main" > /etc/apt/sources.list.d/ros-latest.list'
RPI:~$ wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key
-O - | sudo apt-key add -'
```

Now, make sure your Debian package index is up-to-date:

```
RPI:~$ sudo apt-get update
RPI:~$ sudo apt-get upgrade
```

Install Bootstrap Dependencies

```
RPI:~$ sudo apt-get install -y python-rosdep python-rosinstall-generator
python-wstool python-rosinstall build-essential cmake
```

Initializing rosdep

```
RPI:~$ sudo rosdep init
RPI:~$ rosdep update
```

Instalation

Now, we will download and build ROS Kinetic

Create a catkin Workspace

In order to build the core packages, you will need a catkin workspace. Create one now:

```
RPI:~$ mkdir -p ~/catkin_ws_isolated
RPI:~$ cd ~/catkin_ws_isolated
```

Next we will want to fetch the core packages so we can build them. We will use wstool for this. We will install ROS-Comm (ROS package, build, and communication libraries without GUI tools).

```
RPI:~$ rosinstall_generator ros_comm ros_control sensor_msgs
diff_drive_controller joint_state_controller control_toolbox --rosdistro
```

README.md

12/5/2019

```
kinetic --deps --wet-only --exclude roslibsp --tar > kinetic-ros_comm-wet.rosinstall
RPI:~$ wstool init src kinetic-ros_comm-wet.rosinstall -j2
```

This will add all of the catkin or wet packages in the given variant and then fetch the sources into the `~/catkin_ws_isolated/src` directory. The command will take a few minutes to download all of the core ROS packages into the src folder. The `-j2` option downloads 2 packages in parallel.

Resolve Dependencies

Before you can build your catkin workspace, you need to make sure that you have all the required dependencies. We use the `rosdep tool` for this, however, a couple of dependencies are not available in the repositories. They must be manually built first.

libconsole-bridge-dev:

```
RPI:~$ sudo apt-get install libconsole-bridge-dev
```

liblz4-dev:

```
RPI:~$ sudo apt-get install liblz4-dev
```

Resolving Dependencies with rosdep

The remaining dependencies should be resolved by running rosdep:

```
RPI:~$ cd ~/catkin_ws_isolated
RPI:~$ rosdep install -y --from-paths src --ignore-src --rosdistro kinetic
      -r --os=debian:jessie
```

This will look at all of the packages in the src directory and find all of the dependencies they have. Then it will recursively install the dependencies.

The `--from-paths` option indicates we want to install the dependencies for an entire directory of packages, in this case src. The `--ignore-src` option indicates to rosdep that it shouldn't try to install any ROS packages in the src folder from the package manager, we don't need it to since we are building them ourselves. The `--rosdistro` option is required because we don't have a ROS environment setup yet, so we have to indicate to rosdep what version of ROS we are building for. Finally, the `-y` option indicates to rosdep that we don't want to be bothered by too many prompts from the package manager.

After a while rosdep will finish installing system dependencies and you can continue.

Building the catkin Workspace

9 / 12

README.md

12/5/2019

Once you have completed downloading the packages and have resolved the dependencies, you are ready to build the catkin packages.

Invoke `catkin_make_isolated`:

```
RPI:~$ sudo ./src/catkin/bin/catkin_make_isolated --install -DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic -j1
```

Source the `setup.bash` in the `~/.bashrc`, so that ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
RPI:~$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

Maintaining a Source Checkout

Adding Released Packages

You may add additional packages to the installed ros workspace that have been released into the ros ecosystem. First, a new `rosinstall` file must be created including the new packages (Note, this can also be done at the initial install). For example, if we have installed `ros_comm`, `sensor_msgs` and `ros_control`, but want to add `roscpp_tutorials`, the command would be:

```
RPI:~$ cd ~/catkin_ws_isolated
RPI:~$ rosinstall_generator ros_comm ros_control sensor_msgs roscpp_tutorials --rosdistro kinetic --deps --wet-only --tar > kinetic-custom_ros.rosinstall
```

You may keep listing as many ROS packages as you'd like separated by spaces.

Next, update the workspace with `wstool`:

```
RPI:~$ wstool merge -t src kinetic-custom_ros.rosinstall
RPI:~$ wstool update -t src
```

After updating the workspace, you may want to run `rosdep` to install any new dependencies that are required:

```
RPI:~$ rosdep install -y --from-paths src --ignore-src --rosdistro kinetic -r --os=debian:jessie
```

Finally, now that the workspace is up to date and dependencies are satisfied, rebuild the workspace:

README.md

12/5/2019

```
RPI:~$ sudo ./src/catkin/bin/catkin_make_isolated --install -  
DCMAKE_BUILD_TYPE=Release --install-space /opt/ros/kinetic
```

ROS Hello World!

To check if ROS is working properly, create a new workspace where the Doogie Mouse packages will be installed and built:

```
RPI:~$ mkdir -p ~/doogie_ws/src  
RPI:~$ cd ~/doogie_ws  
RPI:~$ catkin_make
```

Now, clone the `doogie_welcome` package into the workspace and build it:

```
RPI:~$ cd ~/doogie_ws/src  
RPI:~$ git clone https://github.com/doogie-mouse/doogie_welcome.git  
RPI:~$ cd ~/doogie_ws/  
RPI:~$ catkin_make
```

Run the nodes:

```
RPI:~$ source ~/doogie_ws/devel/setup.bash  
RPI:~$ roslaunch doogie_welcome doogie_welcome.launch
```

Note: Source the devel setup.bash in the `~/.bashrc`, so that ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
RPI:~$ echo "source ~/doogie_ws/devel/setup.bash" >> ~/.bashrc
```

In another terminal of your machine, go inside the Doogie Mouse Raspberry again using SSH and do the next steps.

You could see all the topics created by running:

```
RPI:~$ rostopic list
```

This command will reproduce a list like

README.md

12/5/2019

```
pi@doogie-mouse:~ $ rostopic list
/cpp_chatter
/python_chatter
/rosout
/rosout_agg
```

To see what is happening in the topic `cpp_chatter`, run the command:

```
RPI:~$ rostopic echo /cpp_chatter
```

Usage tools

The tools listed below are highly recommended.

Installing terminal Terminator

Run the command:

```
RPI:~$ sudo apt-get install terminator
```

See the [Terminator's documentation](#) for detailed functionalities description.

Installing htop

To see CPU and RAM memory usage by Raspberry, install htop

```
RPI:~$ sudo apt-get install htop
```

References

This software setup was made based on:

- [Installing ROS Kinetic on the Raspberry Pi](#)
- [Installing ROS Indigo on the Raspberry Pi](#)
- [SSH-Keys : pi-user](#)

Referências Bibliográficas

- AKHTAR, R. *Search Algorithms in AI*. 2019. Disponível em: [\(https://www.geeksforgeeks.org/search-algorithms-in-ai/\)](https://www.geeksforgeeks.org/search-algorithms-in-ai/). Acesso em: 01 dez. 2019. 2.1
- BENESOVA, A.; TUPA, J. Requirements for education and qualification of people in industry 4.0. *Procedia Manufacturing*, v. 11, p. 2195 – 2202, 2017. ISSN 2351-9789. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. Disponível em: [\(<http://www.sciencedirect.com/science/article/pii/S2351978917305747>\)](http://www.sciencedirect.com/science/article/pii/S2351978917305747). 1.2
- BRYANT, G.; BUM, K.; CHOI, H. *WolfieMouse*. 2018. Disponível em: [\(<https://github.com/kbumsik/WolfieMouse>\)](https://github.com/kbumsik/WolfieMouse). Acesso em: 13 out. 2019. 2.7, 2.4
- CAMPOS, F. Robótica educacional no brasil: Questões em aberto, desafios e perspectivas futuras. *Revista Ibero-Americana de Estudos em Educação*, v. 12, n. 4, p. 2108–2121, 2017. 1
- CNI. *Desafios para Indústria 4.0 no Brasil*. Rio de Janeiro, 2016. 1.2
- CUI, X.; SHI, H. A*-based pathfinding in modern computer games. v. 11, 11 2010. 2.1.2
- DE, T.; HALL, D. *The inception of chedda: a detailed design and analysis of micromouse*. [S.l.], 2004. 2.2
- ELKADY, A.; SOBH, T. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, v. 2012, p. 15, 2012. Disponível em: [\(<https://doi.org/10.1155/2012/959013>\)](https://doi.org/10.1155/2012/959013). 1
- ENDRISS, U. *Search Techniques for Artificial Intelligence*. 2015. 72 Slides. 2.1
- FOUNDATION, I. O. S. R. *ROSbot 2.0*. 2019. Disponível em: [\(<https://robots.ros.org/rosbot-2.0>\)](https://robots.ros.org/rosbot-2.0). Acesso em: 20 mai. 2019. 1, 2.2
- FOUNDATION, I. O. S. R. *TurtleBot*. 2019. Disponível em: [\(<https://www.turtlebot.com>\)](https://www.turtlebot.com). Acesso em: 20 mai. 2019. 1
- GONZALEZ, I.; CALDERON, A. J. Development of final projects in engineering degrees around an industry 4.0-oriented flexible manufacturing system: Preliminary outcomes and some initial considerations. *Education Sciences*, v. 8, n. 4, p. 15, 2018. Disponível em: [\(<https://doi.org/10.3390/educsci8040214>\)](https://doi.org/10.3390/educsci8040214). 1.2
- HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, v. 4, n. 2, p. 100–107, July 1968. ISSN 2168-2887. 2.1.2
- HAYAMA, K.; MATSUMOTO, T. Practice of creativity Education by the subject of micromouse competition. *IFAC Proceedings Volumes*, v. 42, n. 24, p. 322–325, 2010. ISSN 1474-6670. Disponível em: [\(<http://www.sciencedirect.com/science/article/pii/S1474667015316402>\)](http://www.sciencedirect.com/science/article/pii/S1474667015316402). 2.6, 2.3

JABBAR, A. Autonomous navigation of mobile robot based on flood fill algorithm. v. 12, p. 79–84, 06 2016. [2.1.3](#)

JUNIOR, C. G. B. Agregando frameworks de infra-estrutura em uma arquitetura baseada em componentes: Um estudo de caso no ambiente aulanet. julho 2006. [2.3](#)

KHOMCHENKO, V. G.; GEBEL, E. S.; PESHKO, M. S. Educational robotics as part of the international science and education project "synergy" in realizing the social needs of society on the road to the industrial revolution "industry 4.0". *EAI Endorsed Transactions on Energy Web*, European Alliance for Innovation n.o., v. 5, n. 16, p. 153816, jan. 2018. Disponível em: <<https://doi.org/10.4108/eai.30-1-2018.153816>>. [1.2](#)

KOENIG, N.; HOWARD, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, v. 3, p. 2149–2154, 2004. [3.8](#)

LUZHOU, Y. *Micromouse 2016-2017*. 2016. Disponível em: <<http://greenye.net/Pages/Micromouse/Micromouse2016-2017.htm>>. Acesso em: 10 mai. 2019. [2.4, 2.1](#)

MAHMUD, D. A. *O USO DE ROBÓTICA EDUCACIONAL COMO MOTIVAÇÃO A APRENDIZAGEM DE MATEMÁTICA*. 2017. Disponível em: <<http://webcache.googleusercontent.com/search?q=cache:http://www2.unifap.br/matematica/files/2017/07/O-USO-DE-ROB%25C3%2593TICA-EDUCACIONAL-COMO-MOTIVA%25C3%2587%25C3%2583O-A-APRENDIZAGEM-DE-MATEM%25C3%2581TICA.pdf>>. Acesso em: 30 sep. 2019. [2.3](#)

MCCORDUCK, P. *Machines Who Think*. 2nd edition. ed. [S.l.]: A. K. Peters ltd., 2004. [1](#)

MIES, G.; ZENTAY, P. Industrial robots meet industry 4.0. In: *Teste*. [s.n.], 2017. v. 2, n. 4. Disponível em: <http://hadmernok.hu/174_22_mies.pdf>. Acesso em: 15 mai. 2019. [1.2](#)

NETO, R. P. et al. Robótica na educação: Uma revisão sistemática dos Últimos 10 anos. *Simpósio Brasileiro de Informática na Educação*, n. XXVI, p. 8, 2015. [1, 1.2](#)

REUBEN, H. 1977-79 – “Moonlight Special” Battelle Inst. (American). 2010. Disponível em: <<http://cyberneticzoo.com/mazesolvers/1977-79-moonlight-special-battelle-inst-American/>>. Acesso em: 25 jun. 2019. [2.1](#)

ROTTA, F. *Industria 4.0 pode economizar R\$ 73 bilhões ao ano para o Brasil*. 2017. Disponível em: <<https://www.abdi.com.br/postagem/industria-4-0-pode-economizar-r-73-bilhoes-ao-ano-para-o-brasil>>. Acesso em: 15 mai. 2019. [1.2](#)

RTCORPORATION. *Raspberry Pi*. 2016. Disponível em: <<https://www.rt-net.jp/products/raspimouse2>>. Acesso em: 03 nov. 2019. [2.8, 2.5](#)

SAITO, I. (Ed.). *Packages - ROS Wiki*. [S.l.], 2019. Disponível em: <<http://wiki.ros.org/Packages>>. [2.3](#)

SANTOS, N. M. B. dos. Atualização de demonstrador robótico para utilização do ros. junho 2013. [2.3](#)

- SAUVÉ, J. P. *Frameworks*. 2001. Disponível em: [\(<http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>\)](http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm). Acesso em: 30 sep. 2019. 2.2
- SMITH, W. Dictionary of greek and roman biography and mythology. In: . [S.l.: s.n.], 1849. 1
- TECHNOLOGY, I. *TON-BOT V1.1*. 2016. Disponível em: [\(<https://github.com/iotontech>\)](https://github.com/iotontech). Acesso em: 17 ago. 2019. 3.5.1
- TJIHARJADI, S.; WIJAYA, M. C.; SETIAWAN, E. Optimization maze robot using A* and flood fill algorithm. *International Journal of Mechanical Engineering and Robotics Research*, v. 6, n. 5, p. 366–372, 2017. ISSN 22780149. 2.1
- VARGAS, I. G.; TAVARES, D. M. *Estudo de Caso Usando o Framework Robot Operating System (ROS)*. 2013. Disponível em: [\(<https://www.enacomp.com.br/2013/anais/pdf/44.pdf>\)](https://www.enacomp.com.br/2013/anais/pdf/44.pdf). Acesso em: 30 set. 2019. 2.3
- WAN, S.; RUBSTEIN, C. Micromouse Competition Guidelines. *2019 IEEE Region 1 Student Conference*, n. 1, p. 45, 2019. 3.5.2
- WINSTON, P. H. *Artificial Intelligence*. 3. ed. Reading, MA: Addison-Wesley, 1992. ISBN 978-0-201-53377-4. 2.1
- WPISMARTMOUSE. *WPI Smartmouse 2018*. 2016. Disponível em: [\(<https://github.com/WPISmartmouse/Smartmouse_2018>\)](https://github.com/WPISmartmouse/Smartmouse_2018). Acesso em: 27 ago. 2019. 2.5, 2.2