Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Luiz Henrique Buzzi

Projeto e concepção de uma plataforma robótica móvel integrada com o ROS

Blumenau
2019

Luiz Henrique Buzzi

Projeto e concepção de uma plataforma robótica móvel integrada com o ROS

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Controle e Automação.
Orientador: Prof. Dr. Ebrahim Samer El Youssef
Coorientador: Prof. Dr. Marcelo Roberto Petry

Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Blumenau
2019

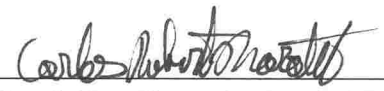
Luiz Henrique Buzzi

Projeto e concepção de uma plataforma robótica móvel integrada com o ROS

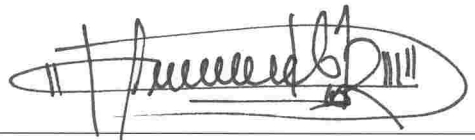
Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação.

Comissão Examinadora

Prof. Dr. Ebrahim Samer El Youssef
Universidade Federal de Santa Catarina
Orientador



Prof. Dr. Carlos Roberto Moratelli
Universidade Federal de Santa Catarina



Prof. Dr. Leonardo Mejia Rincon
Universidade Federal de Santa Catarina

Blumenau, 9 de julho de 2019

Dedico este trabalho a todos aqueles que, de alguma forma,
auxiliaram para a concretização desta etapa.

Agradecimentos

Gostaria de agradecer minha família. Especialmente, meu pai que sempre me deu suporte e minha mãe que sempre acreditou em mim. A minha namorada Priscila, uma pessoa que foi compreensiva. Ao meu coorientador Petry por todo suporte, compreensão e acreditar em mim para o desenvolvimento deste projeto. Ao meu orientador Ebrahim, que me apoiou durante a escrita deste trabalho. A todos os professores que fizeram parte desta jornada e ao INESC P&D e CNPq pelos auxílios durante a pesquisa.

"O insucesso é apenas uma oportunidade para recomeçar com mais inteligência."
(Henry Ford)

Resumo

Com o aumento do incentivo em pesquisas na área da robótica nos últimos anos, busca-se o desenvolvimento de robôs para as mais diversas tarefas. Nesse contexto a robótica móvel tem ganho espaço. O desenvolvimento de máquinas com a capacidade de se locomover em ambientes não controlados e desconhecidos abre oportunidade para aplicações muito além da robótica fixa. Cria uma área de estudos com problemas cada vez maiores. Não basta um robô possuir a capacidade de desviar obstáculos e seguir uma trajetória, pois em ambientes dinâmicos e com tarefas mais complexas, o robô precisa possuir um grau de inteligência, conseguir identificar o ambiente, localizar objetos e entender a sua posição no espaço. Para o desenvolvimento de métodos e algoritmos para solucionar os problemas na área da robótica móvel é preciso ter a disponibilidade de um robô. Muitos dos robôs utilizados em pesquisa possuem um alto custo de aquisição e implementação. Deste modo, muitos estudos acabam se limitando ao desenvolvimento teórico ou simulações devido a indisponibilidade de uma plataforma física. Este trabalho se propõe a estudar, projetar e construir uma plataforma robótica móvel de custo viável. O trabalho apresenta todos os procedimentos, métodos e ferramentas de projeto que foram utilizados para o desenvolvimento do robô. Explora seus subsistemas mecânico, eletrônico e de software, além de disponibilizar todo o detalhamento e algoritmos que possibilita a reconstrução da plataforma. O robô construído foi integrado ao Robotic Operating System (ROS) uma plataforma de código aberto que permite o desenvolvimento de projetos comerciais e não comerciais, facilita o trabalho de roboticista proporcionando ferramentas já validadas para a implementação de funções para o robô, por exemplo a navegação. Ao final são feitos testes para validar as especificações do projeto com o robô construído. Para validar a integração com o ROS, o robô foi parametrizado para desenvolver a exploração de um ambiente e criar o mapa de forma autônoma.

Palavras-Chave: 1. Robótica. 2. ROS. 3. Robô Móvel. 4. Projeto de Robô.

Abstract

With the increase of the incentive for robotics research in the last years, the development of robots for the most diverse tasks is sought. In this context mobile robotics has gained space. The development of machines with the ability to move around in uncontrolled and unknown environments opens up opportunities for applications far beyond fixed robotics. It creates an area of study with increasing problems. It is not enough for a robot to have the ability to divert obstacles and follow a trajectories, because in dynamic environments and with more complex tasks, the robot needs to possess a degree of intelligence, to be able to identify the environment, to locate objects and to understand its position in space. For the development of methods and algorithms to solve the problems in the area of mobile robotics it is necessary to have the availability of a robot. Many of the robots used in research have a high acquisition and implementation cost. In this way, many studies end up being limited to the theoretical development or simulations due to the unavailability of a physical platform. This work proposes to study, design and build a viable mobile robotic platform. This work presents all the procedures, methods and design tools that were used to develop the robot. It explores its mechanical, electronic and software subsystems, as well as providing all the details and algorithms that make it possible to rebuild the platform. The built robot was integrated with the Robotic Operating System (ROS), an open source platform that allows the development of commercial and non-commercial projects, facilitates the work of roboticist providing tools already validated for the implementation of functions for the robot, for example navigation. At the end, tests are done to validate the design specifications with the built robot. To validate the integration with the ROS, the robot was parameterized to develop the exploration of an environment and create the map autonomously.

Keywords: 1. Robotic. 2. ROS. 3. Mobile Robot. 4. Robot Design.

Lista de figuras

Lista de tabelas

Lista de Siglas e Abreviaturas

API	*Application Programming Interface*
ARP	*Autonomous Robot Platform*
AGV	*Automated Guided Vehicle*
AUV	*Autonomous Underwater Vehicle*
AMR	*Autonomous Mobile Robot*
BSD	*Berkeley Software Distribution*
CAD	*Computer-Aided Design*
CCD	*Charge-Coupled Device*
CMOS	*Complementary Metal Oxide Semiconductor*
CPR	*Counts Per Revolution*
DC	*Direct Current*
FEA	*Finite Element Analysis*
LIDAR	*Light Detection And Ranging*
LiPo	*Lithium Polymer*
NASA	*National Aeronautics and Space Administration*
PPR	*Points Per Revolution*
ROS	*Robot Operating System*
RPM	*Rotações Por Minuto*
RUR	*Rossum's Universal - Robots*
SLAM	*Simultaneous Localization and Mapping*
UAV	*Unmanned Aerial Vehicle*
UFSC	*Universidade Federal de Santa Catarina*
URDF	*Universal Robotic Description Format*
XML	*Extensible Markup Language*

Sumário

1 Introdução

Quando falamos de robô nossa mente traz imagens de filmes de ficção cientifica, caracterizando-o como um humanoide, com capacidades e aparência de um ser humano. Esta é uma imagem vista por muitas pessoas que não se encontram neste meio de estudo e mostra o robô como algo ruim, denegrindo a imagem da robótica. De fato, a robótica tem feitos grandes avanços nos últimos anos, mas estamos longe de alcançar o estereótipo construído pelo cinema.

Com o aumento do incentivo em pesquisas na área de robótica nos últimos anos, busca-se o desenvolvimento de robôs para as mais diversas tarefas. Os pesquisadores desenvolvem formas de facilitar a vida das pessoas em atividades com alto grau de periculosidade, em locais de difícil acesso ou tarefas que comprometam a saúde física de operadores, como trabalho repetitivo ou transporte de cargas.

Nesse contexto a robótica móvel tem ganhado espaço, o desenvolvimento de máquinas com a capacidade de se locomover em ambientes não controlados e desconhecidos abre oportunidade para aplicações muito além da robótica fixa e cria uma área de estudos com problemas cada vez maiores. Não basta um robô possuir a capacidade de desviar obstáculos e seguir trajetória, pois em ambientes dinâmicos e com tarefas mais complexas, o robô precisa possuir um grau de inteligência, conseguir identificar o ambiente, localizar objetos, entender a sua posição no meio físico.

1.1 Contexto Histórico

O termo *robô* foi introduzido pelo dramaturgo Karel Capek em 1921 com a peça Rossum‘s Universal - Robots (R.U.R.) [1]. A palavra resulta da combinação das palavras tcheca *robota* (trabalho obrigatório) e *robotnik* (servo) [2]. Popularizou-se rapidamente entre escritores de ficção cientifica, sendo utilizada para referenciar seres mecânicos.

Em 1940, o escritor Issac Asimov considerado pai da robótica, em conjunto com outros escritores de ficção cientifica formularam as três leis da robótica de Asimov, que apareceram pela primeira vez em sua coleção *Runaround* [3] e são apresentadas na sequencia:

1. Um robô não pode ferir um ser humano ou, por omissão, permitir que um humano seja ferido.

2. Um robô deve obedecer às ordens dadas por humanos, exceto quando essas ordens se conflitarem com a Primeira Lei.

3. Um robô deve proteger sua própria existência a menos que isso infrinja a Primeira e Segunda Lei.

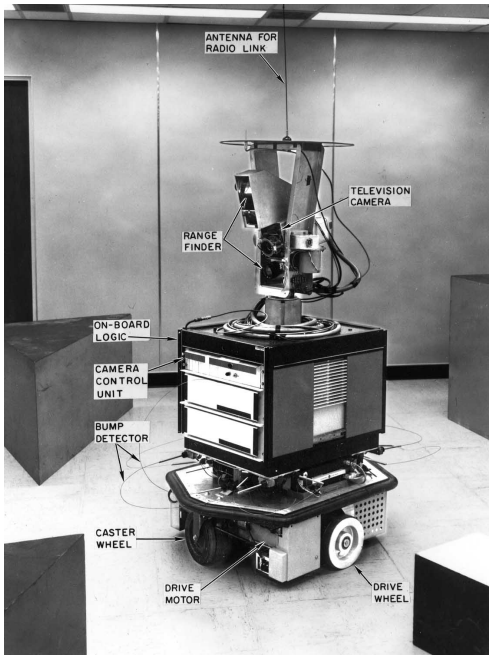


Figura 1 – Robô Shakey [Fonte: Robot globe[1]].

O escritor popularizou o termo robô em 1940, com seu livro *I, robot*. A ficção cientifica influenciou a percepção da humanidade sobre robôs, que continua os imaginando como sendo humanoides que podem falar, andar, ver e ouvir assim como mostra o filme *I, robot*, inspirado nos livros de Asimov [4].

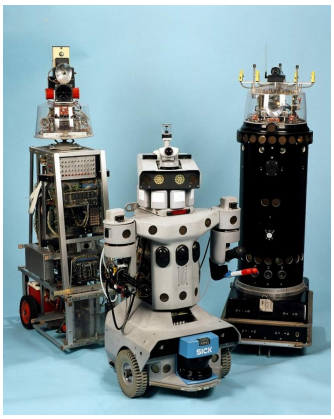
Em 1959, Devol e Joseph F. Engelberger desenvolveram o primeiro robô industrial pela Unimation Inc. Este robô tinha como função uma variedade de tarefas executadas automaticamente e diferia dos autômatos já que podia ser reprogramado e remodelado para outras tarefas com um nível de custo pouco elevado [5].

Em 1969, Nilsson descreve o primeiro sistema robótico móvel com planejamento de trajetória [6]. Em 1972, o robô Shakey - resultado do estudo de Nilsson (Figura 1), foi apresentado com primeiro robô móvel controlado por inteligencia artificial [7].

A partir da década de 70 e início dos anos 80, começaram a aparecer projetos para aplicações de segurança, exploração, transporte de carga, em sua maioria patrocinados por entidades governamentais, militares ou empresas [8].

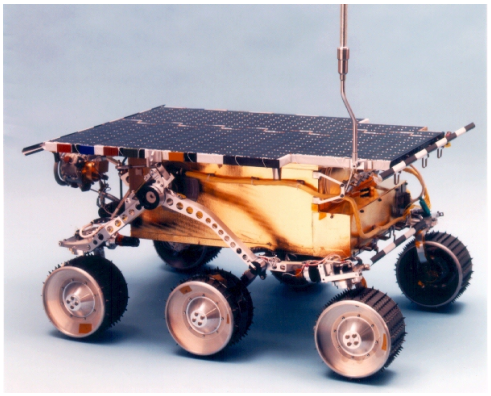
Vários projetos foram desenvolvidos desde então: *Robart I* (1980-1982), *Robart II* (1982-1992) e *Robart III* (1992-2009), Figura 2a, desenvolvido pela marinha americana, tinha como objetivo detectar potenciais intrusos [9]. Em 1994, a NASA (National Aeronautics and Space Administration) e seus parceiros desenvolvem o robô hexápode *Dante*, Figura 2b, para inspeção de vulcões [7]. Em 1997, o robô *Sojourner*, Figura 2c, é usado na missão *Pathfinder* para explorar Marte [10].

[1] http://robotglobe.org/wp-content/uploads/2015/05/Shakey-Robot-763x1024.jpg



(a) Família Robart.

(b) Dante II.



(c) Sojourner.

Figura 2 – Robôs desenvolvidos no decorrer da história [Fonte: (a) The old robot[1] (b) Science Friday[2] (c) Nasa[3]].

1.2 Robô

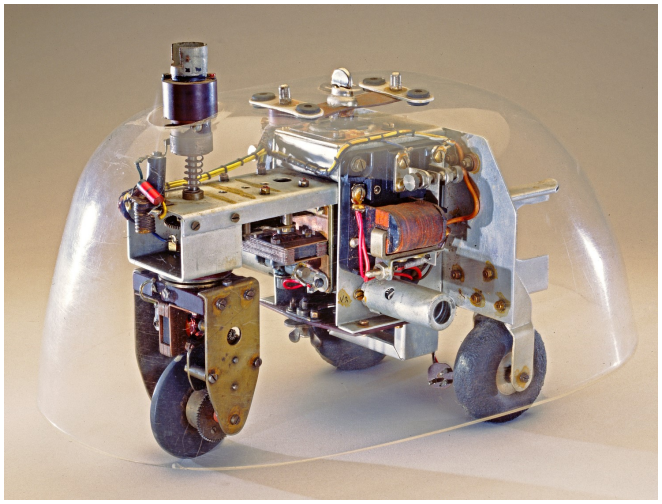
Segundo Mataric [2] "Robô é um sistema autônomo que existe no mundo físico, pode sentir o seu ambiente e pode agir sobre ele para alcançar alguns objetivos". Um robô deve ser capaz de tomar suas próprias decisões, portanto máquinas tele operadas não entram nesta definição, existem casos de robôs que recebem instruções de tarefas, mas tomam suas próprias decisões para alcançá-las. Para ser capaz de tomar decisões o robô precisa conhecer o mundo ao seu redor, para isto utiliza de sensores. Por fim o robô deve ser capaz de interagir com o ambiente, isso é possível com a presença de atuadores acoplados ao robô.

Para Latombe robô é um dispositivo mecânico versátil, equipado com atuadores e sensores sobre o controle de um sistema computacional. Opera em um espaço de trabalho no mundo real, onde é composto por objetos físicos e regido pelas leis da natureza. O robô executa tarefas se movimentando no espaço de trabalho [11].

Perceba que diferente do conceito popular robô não é descrito como um humanoide, mas como um sistema que existe no mundo físico e possui a capacidade de interagir com os elementos presentes nele. Um robô é desenvolvido para operar em um espaço determinado, para isso são feitas considerações durante o projeto para que ele possua recursos para atuar nas condições propostas. E mais, um robô possui um sistema que promove a capacidade de tomar decisões, o que permite o robô executar tarefas e cumprir objetivos determinados pelo roboticista. Por fim, um robô deve possuir a capacidade de sentir o ambiente, para isso utiliza de sensores que provem informações de acordo com as necessidades do projeto.

[1] http://www.theoldrobots.com/images79/RobartGroup_123.JPG
[2] http://www.sciencefriday.com/wp-content/uploads/2016/07/Dante-cover.jpg
[3] https://mars.jpl.nasa.gov/MPF/roverpwr/blue_up1.jpg

(a) Manipulador de seis graus de liberdade da Kuka.



(b) Elise, a tartaruga de Grey Walter, possui a capacidade de procurar luz e desviar obstáculos.

Figura 3 – Exemplo de robô fixo e móvel [Fonte: (a) Pinterest[1] (b) Smithsonian[2]].

Portanto, um robô não precisa ser móvel ou inteligente. Ele pode ser fixo como um manipulador robótico (Figura 3a) e responder a algumas situações em condições particulares, como a Tartaruga de Grey Walter (Figura 3b) [12].

1.3 Robô móvel

Segundo Pieri[7] "um robô móvel é um dispositivo mecânico montado sobre uma base não fixa, que age sob o controle de um sistema computacional, equipado com sensores e atuadores que o permitem interagir com o ambiente". Sob este aspecto um robô deve ser capaz de perceber o ambiente através de sensores, processar estas informações e atuar sobre o meio através de atuadores.

Para Corke[13] um robô móvel é uma classe de robô que tem a capacidade de se locomover pelo ambiente e sua principal função é se deslocar para algum lugar. Este lugar deve ser especificado e referenciado a alguma característica do ambiente, por exemplo mova para a luz, ou em termos de coordenadas geométricas ou de um mapa de referência.

Com uma visão mais cientifica Nehmzow[14] define robô móvel autônomo como o loop fechado entre percepção e ação. É capaz de sentir o ambiente pelos sensores, processar estas informações utilizando um computador on-board, e responder com movimento. A premissa de um robô móvel é ser capaz de se mover no ambiente e, de forma autônoma ser capaz de executar algumas funções, como reagir a obstáculos ou seguir uma trajetória. Porém em uma visão macro, pode ser controlado por um operador.

[1] https://i.pinimg.com/originals/88/e2/89/88e289a9f358cb8a5544b0024db9297d.jpg
[2] http://ids.si.edu/ids/deliveryService?id=NMAH-94-3261

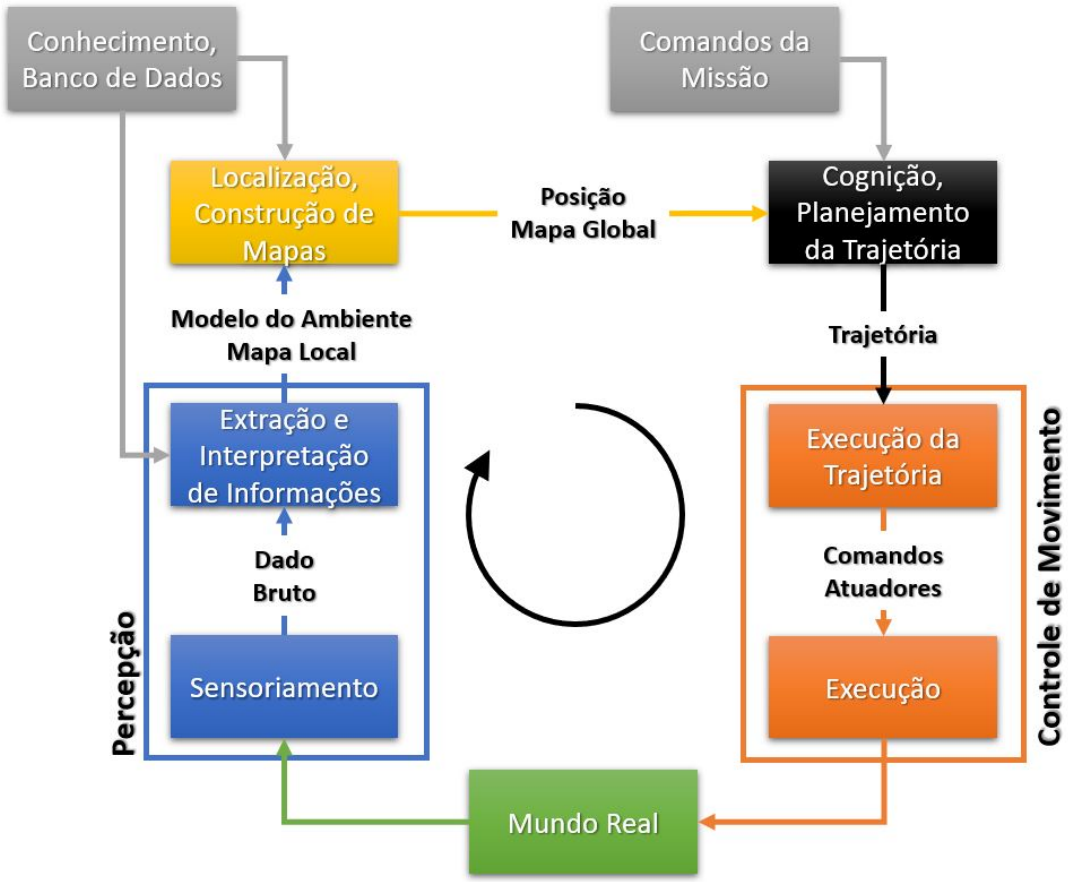
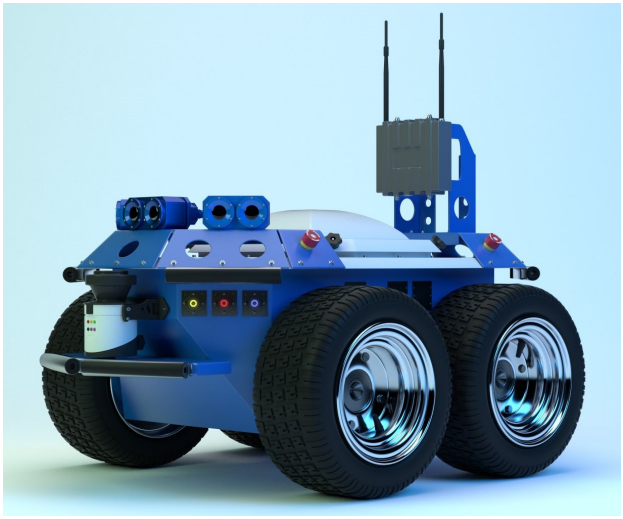


Figura 4 – Diagrama de operação de um robô móvel [Fonte: Adaptado e traduzido de Siegwart, Scaramuzza e Nourbakhsh[10]].

Desta forma é possível afirmar que, o sucesso dos projetos de robótica móvel envolve o conhecimento e desenvolvimento de diferentes áreas do conhecimento, tornando este um campo multidisciplinar. Para a solução dos problemas de locomoção é preciso conhecimento de cinemática, dinâmica e teoria de controle. Para desenvolver um sistema perceptivo robusto é preciso conhecimento da área de sinais e sensores. Os problemas de localização e navegação necessitam de conhecimento de algoritmos, inteligência artificial e estatística [10]. A Figura 4 apresenta de forma simplificada as principais funcionalidades de um robô móvel e a forma como este interage entre os seus subsistemas, mostrando a iteração entre as áreas citadas.

Desta forma, o projeto de um robô móvel depende da escolha das características que serão implementadas com base na aplicação determinada. Os atributos mecânicos e os parâmetros de projeto vão depender da aplicação que o robô vai executar, do tipo do ambiente onde vai atuar e das tarefas que deve cumprir, como evitar obstáculos ou se posicionar em algum local específico. Da mesma forma, os programas que o robô vai executar e o seu grau de autonomia também será determinado pela aplicação.



(a) Seekur Jr. da Omron Adept MobileRobots.

(b) Husky da Clearpath Robotics.

Figura 5 – Plataformas móveis comerciais [Fonte: (a) Alhamidi[1] (b) Clearpath Robotics[2]].

1.4 Trabalhos relacionados

Como o objetivo deste trabalho é o desenvolvimento de um produto, com características comerciais, a análise do estado da arte se concentrou na busca de produtos comerciais já disponíveis no mercado e feito um levantamento de suas principais característica, que auxiliaram na definição do escopo do projeto.

1.4.1 Seekur Jr

O Seekur Jr. é um robô comercial desenvolvido pela Adept MobileRobots, possui 77kg distribuídos em 835x1198x494mm com 108mm de distância do chão. Esta plataforma conta com certificação IP-54 e rodas de D:400x172mm. Dispõem de dois motores, com uma redução de 70:1. Utiliza encoders de 1024 counts e acelerômetro para estimar o movimento e a posição do robô. Possui um chassi monocoque desenvolvido em alumínio e é alimentado com uma bateria de 24V de NiMH com capacidade de 10Ah [15].

1.4.2 Husky

O Husky é um robô comercial desenvolvido pela Clearpath Robotics, possui 50kg distribuídos em 670x990x390mm com 130mm de distância do chão. Esta plataforma conta com certificação IP-44 e rodas de D:330x114mm. Dispõe de dois motores e compartimento de carga com 296x411x155mm. Utiliza encoders em quadratura com 78,000 pulsos/m e é alimentado com uma bateria de 24V de Li-ion com capacidade de 20Ah [16].

[1] https://alhamidi.net/project/seekur-jr
[2] https://www.clearpathrobotics.com/wp-content/uploads/2015/08/A008_C036_0730G1.jpg

(a) RMP 440 LE da Segway. (b) Projeto FROG.

Figura 6 – Plataformas móveis comerciais [Fonte: (a) Manu Systems[1] (b) FROG: Fun Robotic Outdoor Guide[2]].

1.4.3 RMP 440 LE

O RMP 440 LE é um robô comercial desenvolvido pela Segway Robotics, possui 120kg distribuídos em 842x1105x533mm com 120mm de distância do chão. Esta plataforma conta com certificação IP-66 e rodas de D:533x168 mm. Dispões de quatro motores com drivers individuais e redundantes. É alimentado por cinco baterias de 73,6V de LiFePO4 com capacidade de 5.2Ah [17].

1.4.4 FROG

O FROG (Fun Robot Outdoor Guide) é um projeto desenvolvido por um consórcio de empresas e universidades para desenvolver um robô guia. Seu sistema base, utiliza de uma plataforma skid steer para se movimentar. Possui 32kg sem bateria distribuídos em 593x974x300mm com 58mm de distância do chão. Conta com rodas de D:260x90mm. Dispõe de dois motores de 24V 200W com caixa de redução de 19:1 e encoder de 500 pulsos (HEDS 5540). É alimentado com até 8 baterias de 12V de chumbo-ácido totalizando 45kg a mais no projeto e com capacidade de 20Ah cada [18].

[1] https://en.manu-systems.com/images/L/RMP-2316100001.jpg
[2] https://www.frogrobot.eu/wordpress/wp-content/uploads/2014/06/image6.jpg

Tabela 1 – Especificações de robôs comerciais.

Robô	**Seekur Jr**	**Husky**	**RMP**	**Frog**
Massa total do robô [kg]	77	50	120	80
Num. motores [#]	2	2	4	2
Diâmetro da roda [m]	0,4	0,33	0,533	0,26
Velocidade do robô [m/s]	1,2	1	8	1,5
Inclinação da superfície [º]	37	45	30	14
Tensão motor [V]	24	24	24	24
Aceleração [m/s^2]	0,5	0,5	0,5	0,5
Tempo de operação [h]	3	3	4	4
Eficiência [%]	65	65	65	65

1.4.5 Análise

Com base na pesquisa foi montada a Tabela 1 com os principais parâmetros dos robôs comerciais. Isto permite uma visão comercial das características desejadas para o projeto do robô. A tabela foi montada com os dados obtidos do site dos fabricantes.

1.5 Motivação

Robôs foram e ainda são assunto de inúmeras pesquisas, devido a sua aplicabilidade em tarefas perigosas e/ou de difícil desenvolvimento pelo o ser humano. Conforme já foi apresentado, tanto a robótica como a robótica móvel possuem desafios em diversas áreas do conhecimento, oferecendo oportunidade de pesquisa e desenvolvimento de novas soluções nos seus diversos campos.

Neste contexto a robótica móvel promove uma vantagem, a locomoção, que permite também o deslocamento do robô para executar tarefas em locais perigosos, não controlados, ruidosos e desconhecidos, tornando o desafio ainda maior. Com estas especificações é preciso o desenvolvimento de robôs que possuam um grau de "inteligência" e autonomia, não basta apenas seguir trajetória e desvia obstáculos, mas sim identificar objetos, locais, tomar suas próprias decisões e ser dotado de percepção.

Para o desenvolvimento de estudo de métodos e algoritmos para solucionar os problemas citados é preciso ter a disponibilidade de um robô. Muitos dos robôs utilizados em pesquisa possuem um alto custo de aquisição e implementação. Deste modo, muitos estudos acabam se limitando a desenvolvimento teórico ou simulações devido a indisponibilidade de uma plataforma física.

Em sua maioria, as plataformas comerciais possuem um projeto fechado, sendo difícil a modificação ou a substituição dos elementos de hardware. As modificações estão limitadas ao software de configuração fornecido pelo fabricante. Em termos de integração, normalmente não são disponibilizados os códigos utilizados para programar a plataforma,

apenas uma API (Application Programming Interface) limitada para o desenvolvimento da integração com outros sistemas e os protocolos utilizados para comunicação.

Desta forma, o desenvolvimento de uma plataforma móvel com objetivo de servir de suporte para outros projetos dentro da universidade que envolvam mobilidade se torna necessário. Fornecendo, ao final do projeto, um sistema para estudo e expertise no desenvolvimento de estrutura mecânica e algoritmos que suprem as necessidades de um projeto de robótica.

1.6 Objetivos

O objetivo geral deste trabalho é estudar, projetar e construir uma plataforma robótica móvel de custo viável. O trabalho apresenta os seguintes objetivos específicos:

- Este projeto deve dar suporte para o desenvolvimento de estudos e pesquisas que envolvam mobilidade; algoritmos para operar e gerenciar robôs; e métodos para construção de robôs.
- Deve operar em terrenos uniformes e admitir pequenas irregularidades.
- Desenvolver uma plataforma robusta, modelo skid steer.
- Criar um projeto modular para facilitar a integração, montagem e manutenção.
- Elaborar um projeto flexível para permitir a utilização da plataforma em diferentes estudos e aplicações.
- Especificar o robô com materiais que sejam facilmente encontrados no mercado.
- Desenvolver um projeto detalhado para possibilitar a reconstrução por outros usuários.
- Programar um driver que integre o robô ao ROS (Robot Operating System).
- Parametrizar uma aplicação que envolva os componentes do ROS para validar a integração;

1.7 Estrutura do documento

Neste capítulo foi apresentada uma introdução sobre robótica e robótica móvel, além de contextualizar o problema que será tratado nesta monografia. No Capítulo 2 serão tratados os conceitos necessários para o desenvolvimento deste trabalho. No Capítulo 3 são apresentadas as atribuições e características desejadas para o protótipo do robô assim como as escolhas tomadas e suas justificativas para o desenvolvimento do projeto. No

Capítulo 4 são apresentados os resultados do projeto, que neste trabalho resultou em um produto, são abordados os estudos que validam as características determinadas na fase de projeto e que corroboram com os objetivos traçados nas fases iniciais. Por fim no Capítulo 5 são apresentadas as considerações finais do trabalho, assim como uma visão de como o desenvolvimento do robô pode continuar e se aperfeiçoar em projetos futuros.

2 Fundamentos sobre robôs móveis

Neste capítulo serão abordados os fundamentos teóricos para o desenvolvimento deste trabalho. Inicialmente, são discutidos classificação e aspectos construtivos de robôs móveis, explicando os principais tipos de sistemas de locomoção empregados. Na sequência, é apresentada uma introdução ao sistema operacional para robôs (ROS), focando em alguns dos pacotes que são empregados neste trabalho.

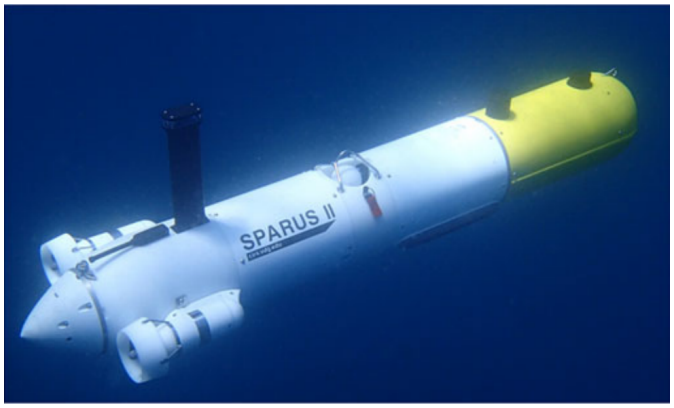
2.1 Classificação de robôs móveis

Segundo Pieri[7] é possível agrupar os robô móveis de acordo com três características: anatomia, tipo de controle e funcionalidades. Quanto a anatomia os robôs móveis podem ser classificados em três grandes grupos: os robôs aéreos, os aquáticos e os terrestres.

Os robôs aéreos, geralmente são assas fixas ou rotativas e entram na classificação de UAVs (Unmanned Aerial Vehicle), um exemplo é o Xaircraft P30, Figura 7a. Recentemente começaram a ser mais explorados para uso comercial e militar. Possuem diversas aplicações, coletar dados de inteligência, vigilância, reconhecimento, busca e salvamento, entrega de mercadorias, agricultura de precisão, dentre outros [19].

Os robôs subaquáticos, geralmente são AUVs (Autonomous Underwater Vehicle), um exemplo é o Sparus II, Figura 7b. Suas aplicações incluem pesquisas oceanográfica, indústria de gás e petróleo, operações militares. Executam atividades como recolhimento

(a) Xaircraft P30: drone utilizado na agricultura de precisão.



(b) Sparus II - AUV para águas rasas, até 200m.

Figura 7 – Robôs aéreos e subaquáticos [Fonte: (a) Mashable[1] (b) Carreras et al.[20]].

[1] https://mondrian.mashable.com/uploads%252Fcard%252Fimage%252F425154%252F424abc76-ffd1-4840-995f-16bd23c4ebe9.jpg%252F950x534__filters%253Aquality%252890%2529.jpg

(a) Pionner P3DX.

(b) Moose UGV desenvolvido pela Clearpath Robots.

Figura 8 – Robôs terrestres com rodas [Fonte: (a) Afswaw[1] (b) Clearpath Robotics[2]].

de dados, mapeamento do fundo do mar, coleta de amostras, trabalhos de manutenção e salvamento [20][21].

Os robôs terrestres são subdivididos em outros três grupos cada qual associado ao seu dispositivo de locomoção, sendo eles: rodas, esteiras e pernas. Sendo os robôs terrestres sobre rodas os mais populares.

Os robôs com rodas (Figura 8a e 8b) são os mais simples, pois não necessitam de um hardware tão complexo quanto os demais. Possui um desempenho inferior ao operar em terrenos irregulares, visto que são projetados para manter pleno contato das rodas com o solo, caso contrário, pode haver perda de tração ou área de contato insuficiente, resultando em perda de movimento e erro de odometria. De forma geral, o robô deve possuir a roda maior ou igual ao obstáculo que ele irá enfrentar [7][10].

Os robôs com esteira (Figura 9a e 9b) são os mais utilizados em ambientes irregulares com solo arenoso, pedregoso ou de superfície solta. Sua estrutura aumenta o contato entre a superfície do terreno com o robô, o que melhora o desempenho em solo arenoso, contribui para a distribuição do torque e reduz o arrasto do robô se comparado ao com rodas. Sua construção mecânica é mais complexa que o robô com rodas, sendo uma das suas desvantagem a dissipação de energia ao efetuar movimentos de rotação, os pontos de contato não estão colineares ao centro de massa, o que exige que o robô deslize para efetuar manobras, tornando difícil prever a sua orientação utilizando apenas odometria de rodas [7][10].

Por fim, os robôs com pernas (Figura 10a) são bastante complexos de controlar, visto que cada uma das pernas deve ter pelo menos dois graus de liberdade. Possuem um alto

[1] https://afsyaw.files.wordpress.com/2017/01/pioneer3dx.png

[2] https://s3.amazonaws.com/assets.clearpathrobotics.com/wp-content/uploads/2019/01/24151000/Moose_DSO_VTR-7.jpg

(a) J5T XTR desenvolvido pela Argo.

(b) Talon desenvolvido pela Foster-Miller.

Figura 9 – Robôs terrestres com esteiras [Fonte: (a) Argo XRT[1] (b) Federal resources[2]].

custo de implementação devido ao custo dos atuadores e a complexibilidade mecânica, além de possuir um maior gasto energético [7]. Contudo, sua grande vantagem está na adaptabilidade em diferentes terrenos, como o Bigdog (Figura 10b). O contato entre o robô e o terreno é pontual, facilitando o trabalho de desviar buracos, navegar em terrenos acidentados, subir escadas, dentre outros. Outra vantagem é a possibilidade de utilizar as patas como atuadores, assim como os insetos, um robô hexápode, consegue se equilibrar em quatro pernas e utilizar as outras duas para arrastar um objeto, ou cumprir uma determinada tarefa [10].

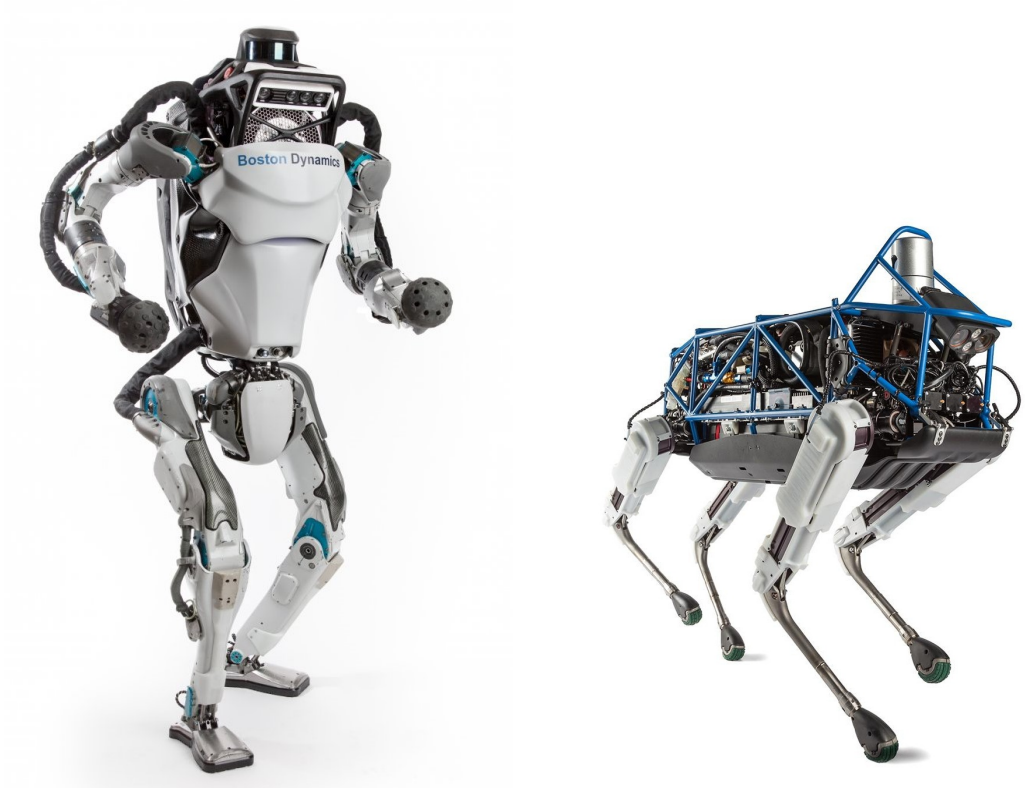
É importante salientar que estas categorias são difusas, uma vez que se pode combinar as características de dois grupos na construção de um robô para uma tarefa específica.

Quanto ao tipo de controle os robôs podem ser separados em três categorias: tele operados, semiautônomos e autônomos. Os tele operados necessitam de um operador para realizar todos seus movimentos. Estes robôs são justificados pelas aplicações em tarefas incomodas ou hostis para o ser humano [7]. Também estão sendo aplicados em procedimentos cirúrgicos em que é exigida a execução de movimentos precisos.

Os robôs semiautônomos, são robôs programados para executar sozinho uma tarefa determinada previamente. São robôs que recebem um comando macro do operador, exemplo robôs de montagem.

Os robôs autônomos são capazes de tomar suas próprias decisões baseados nos dados dos sensores, ou seja, executar a tarefa por conta própria. Conforme apresentado na Figura 4, este modelo deve ser capaz de sentir o ambiente, extrair informações, processar e reagir de acordo com a sua tarefa [10]. A autonomia em algumas tarefas de robôs móveis

[1] http://www.argo-xtr.com/wp-content/uploads/2016/05/j5t-banner-xtr.png
[2] https://www.federalresources.com/wp-content/uploads/2017/02/qinetic-Talon-Hazmat-2.jpg



(a) Atlas. (b) Bigdog.

Figura 10 – Robôs terrestres com pernas desenvolvidos pela Boston Dynamics [Fonte: (a) Boston Dynamics[1] (b) Blogspot[2]].

pode ser interessante, caso este possua tele operação com atraso de comunicação.

Por fim, os robôs móveis podem ser classificados conforme sua funcionalidade, Pieri[7] aponta quatro grupos: robôs industriais, robôs de serviço, robôs de campo, e robôs pessoais. Contudo existe uma sobreposição entres os três primeiros, determinada pela diferença dos locais onde atuam e a necessidade de uma maior autonomia.

Os robôs industriais são utilizados em linhas de produção. Atuam em um ambiente modelado para execução da tarefa, em que estão disponíveis a posição e orientação exata do robô e a de objetos em seu entorno. São comumente utilizados para o transporte de cargas pesadas em sistemas de manufatura. Estes robôs podem ser AGVs (Automated Guided Vehicle), são programados para seguir uma linha no chão.

Robôs de Serviço Industrial possuem características de serviço, mas atuam em um ambiente estruturado. Estes robôs podem ser AMRs (Autonomous Mobile Robot), que possui um sistema com sensores que permite sua localização em tempo de execução, é utilizado para transporte de cargas em linhas estruturadas.

Os robôs de serviço possuem a finalidade de executar serviços gerais. O robô atua em

[1] https://www.bostondynamics.com/sites/default/files/styles/max_1300x1300/public/2019-01/HD-01%20Front.jpg

[2] https://4.bp.blogspot.com/-tIxY36b5cUo/VtnKlxxw7mI/AAAAAAAAASk/_v1zdbZEq_w/s1600/spot2.jpg

um ambiente estruturado e possui um mapa deste ambiente. Precisa possuir uma certa autonomia para a execução de tarefas e atuar em situações imprevistas, como desviar de um objeto ou pessoa. É comum este robô atuar de forma macro, recebendo comando do que fazer e de forma autônoma ser capaz de cumprir a tarefa. É utilizado para vigilância, transporte de materiais, limpeza, dentre outros.

Robôs de serviço de campo atuam em ambientes abertos que podem ter sido previamente mapeados ou não. São equipados com sistemas sensoriais para gerar o modelo ou complementar o modelo existente. São utilizados na realização de tarefas agrícolas e para a navegação em estradas.

Os robôs de campo trabalham em ambientes não estruturados, pouco conhecidos e comumente perigosos. Suas principais funções são a exploração, limpeza de acidentes nucleares e mineração.

Por fim os robôs pessoais são utilizados para tarefas especificas, porém interagem com seres humanos. São bastante utilizados para entretenimento.

2.2 Componentes

Segundo Mataric[2] um robô é composto por um corpo físico, para que possa existir e trabalhar no mundo físico; sensores para que possa sentir e perceber o ambiente; efetuadores e atuadores para que possa interagir com seu meio ambiente; e um sistema de processamento para que possa ser autônomo. Portanto, as capacidades e habilidades do robô são intrinsecamente ligadas a estes componentes e a correta especificação permite que o robô cumpra suas tarefas

Um robô deve possuir um corpo físico, pois um agente computacional, pode conter todos os outros elementos, mas não é considerado um robô. Um robô deve possuir um corpo material, que obedece às leis da física, não podendo estar em mais de um lugar ao mesmo tempo. O corpo do robô é que vai determinar suas características, o modelo mecânico a ser utilizado, podendo seguir as classificações apresentadas na Seção 2.1 para efetuar seu deslocamento, e ainda incorporar funcionalidades adicionais, como braços que o ajudaram a cumprir tarefas.

Possuir um corpo permite ao roboticista entender como este robô se porta no ambiente, conhecer suas fronteiras para modelar o sistema de controle de forma coesa, evitando colidir em objetos presentes no espaço de trabalho. E mais, o corpo do robô é dimensionado com um olhar nas tarefas que este deve cumprir, a forma como é modelado pode determinar o quão rápido ele pode se deslocar ou por onde pode passar.

Como citado previamente, além do corpo físico, o robô precisa de sensores, atuadores e um sistema de processamento. Estes elementos serão tratados nas seções seguintes, onde é feita uma análise mais detalhada dos demais componentes e citando exemplos de aplicação.

2.3 Sensores

Os sensores são uma parte fundamental para qualquer robô, permite não só ter conhecimento do ambiente como de si próprio. Para um robô móvel é importante ter conhecimento de vários estados como posição, orientação, velocidade, deslocamento, temperatura, dentre outros. Estes dados são fundamentais para mover a plataforma pelo ambiente. Também é importante ter conhecimento de dados do ambiente como a posição dos objetos e outros fatores que dizem respeito a tarefa a ser cumprida.

Para Siegwart, Scaramuzza e Nourbakhsh[10] os sensores podem ser classificados em proprioceptivos/exteroceptivos e ativos/passivos. Os sensores proprioceptivos atuam na leitura interna do robô como velocidade do motor, carga na roda, ângulo do atuador e tensão nas baterias. Já os exteroceptivos adquirem informações do ambiente como, distância, intensidade luminosa e amplitude sonora, estes dados são processados pelo robô para interpretar as características do ambiente.

Os sensores passivos são aqueles que medem a energia no ambiente, podendo citar como exemplo os microfones, sensores de temperatura e câmeras CCD (Charge-Coupled Device) ou CMOS (Complementary Metal Oxide Semiconductor). Os sensores ativos são aqueles que emitem energia no ambiente e medem a reação, possuem desempenho superior pois controlam sua iteração com o ambiente, porém, devido a emitir energia estão sujeitos a maior interferência, podendo citar como exemplo dois robôs com sensores parecidos em meio ao mesmo espaço de trabalho. Como exemplos de sensores ativos temos os encoders de quadratura, sensores ultrassônicos e laser de área.

2.4 Atuadores

Os atuadores ou efetuadores são dispositivos que permitem que o robô haja [2]. Os atuadores usam mecanismos subjacentes como músculos artificiais, motores DC (Direct Current), motores de passo, servos motores, pistões, dentre outros para locomover o robô ou atuar sobre um objeto. Dentre as principais atividades de um atuador em robótica móvel podemos citar a locomoção e a manipulação.

Segundo Pieri[7] é possível classificar os atuadores conforme a energia que utilizam para atuação, os três principais são os pneumáticos, hidráulicos e elétricos. Os hidráulicos e pneumáticos são fundamentados no deslocamento de fluidos, podendo ser fluido hidráulico ou ar comprimido respectivamente. Em ambos os casos é necessário a utilização de bombas, reservatórios, filtros, válvulas, dentre outros, o que torna esta implementação desvantajosa frente aos elétricos. Contudo, em aplicações específicas pode tornar-se requisito de projeto.

Os atuadores elétricos são os mais utilizados na robótica. Este fato é atribuído a grande variedade de modelos presente no mercado, indiferente de tamanho ou aplicação. São

mais simples de implementar que os demais, além das ótimas características de controle, precisão e confiabilidade.

2.5 Sistema operativo

O sistema operativo torna o robô autônomo, usam informações perceptivas e informações pré-carregadas, como mapas e referencias, para decidir o que fazer. Em seguida enviar estes dados de forma coesa para os atuadores e efetuadores executarem a ação [2]. Existe sistemas operativos para robôs desenvolvidos no mercado e de código aberto, exemplo ROS, que facilita o trabalho do roboticista proporcionando ferramentas já validadas para a implementação de funções para o robô.

2.5.1 ROS

O *Robotic Operating System* é um framework flexível para o desenvolvimento de aplicações para robôs. É uma composição de ferramentas e bibliotecas que buscam simplificar o processo de desenvolvimento de robôs complexos e robustos para diferentes modelos de plataformas robóticas.

A plataforma é de código aberto e distribuída sob os termos da licença BSD (Berkeley Software Distribution) que permite desenvolvimento de projetos comerciais e não comerciais. Isto possibilita que roboticistas de diferentes laboratórios ao redor do mundo utilizem suas ferramentas e contribuam para o seu desenvolvimento. De fato, é possível encontras diversos algoritmos voltados para aplicações robóticas desenvolvidos e ainda em desenvolvimento nos repositórios da aplicação.

O ROS foi desenvolvido para operar em uma topologia *Peer-to-peer* que permite a comunicação entre os *hosts* sem a necessidade de um servidor central. Para seu funcionamento é preciso um sistema de pesquisa para localizar os *hosts* em tempo de execução, este serviço recebe o nome de *master* [22].

Os fundamentos do desenvolvimento em ROS são baseados em quatro componentes, que são *nodes*, *messages*, *topics* e *services*, aqui referenciados como nodos, mensagens, tópicos e serviços.

- Nodos: são processos do sistema. O ROS é desenvolvido para ser modular e escalável, cada nodo tem a responsabilidade de processar uma parte do código, pode ser visto como um módulo do software. Um típico projeto de robô possui vários nodos que podem ser processados em cada um dos *hosts* da rede. A Figura 11 exemplifica a estrutura de operação de um robô no ROS, perceba que cada nodo é um processo sendo executado de maneira concorrente no robô.

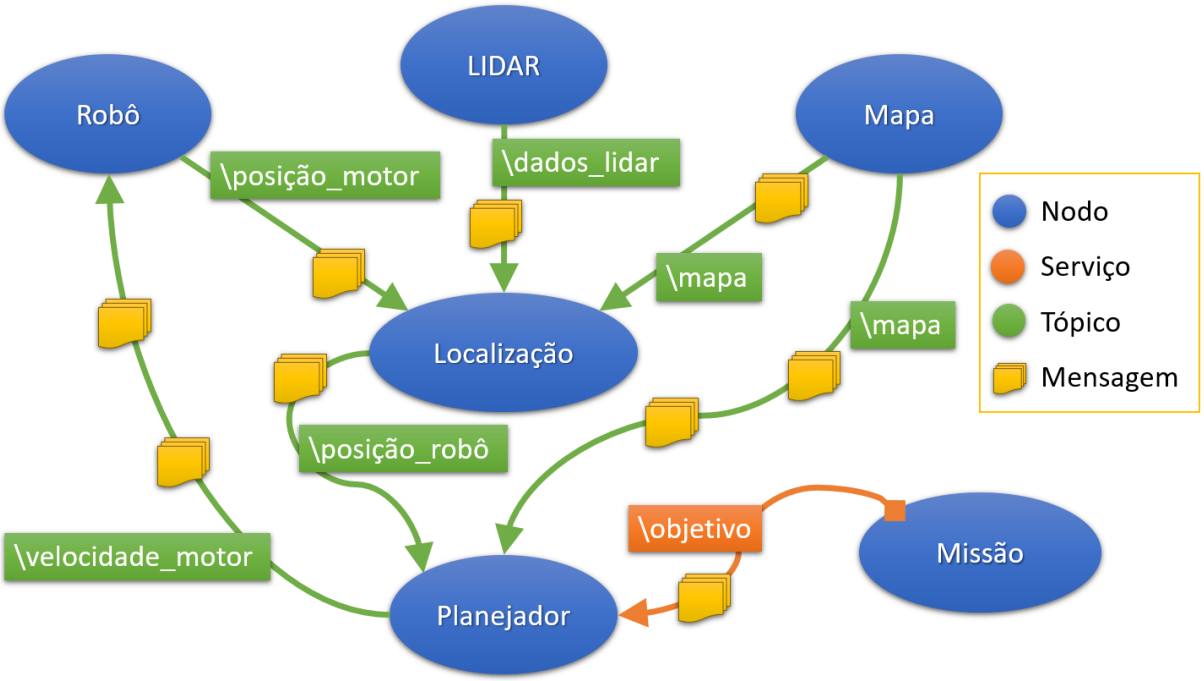


Figura 11 – Exemplo de didático de funcionamento do ROS.

- Tópico: o nodo se comunica com os demais publicando um tópico. Os tópicos possuem nomes simples como *map*, *odometry*, *controller* ou *nav*. Para um nodo receber a informação publicada pelo tópico ele precisa se subscrever a ele. Pode haver vários nodos publicando e subscritos a um mesmo tópico. Conforme é mostrado na Figura 11, os nodos se comunicam por tópicos, e mais, perceba que o nodo Mapa fornece um mesmo tópico duas vezes.

- Mensagem: os tópicos carregam um conjunto de dado, as mensagens. Uma mensagem é basicamente uma estrutura de dados, que compões os tipos primitivos (inteiro, booleano, ponto flutuante, etc.), e podem ser estruturados em vetores e constantes.

- Serviço: os tópicos não garantem a comunicação síncrona, para resolver estes problemas existe os serviços. O serviço é definido por um par de mensagem, uma para fazer uma solicitação e a outra para receber como resposta. Diferentes dos tópicos, apenas um nó pode anunciar um serviço com um nome específico. A Figura 11 exemplifica um serviço, onde, o nodo missão envia um objetivo para o planejador e aguarda uma confirmação. Se for do interesse do roboticista, o nodo missão pode parar sua execução até que a resposta seja recebida.

Nas subseções seguintes serão tratadas as ferramentas do ROS que foram utilizadas para o desenvolvimento do controle do Robô, sendo elas o ROS *Control*, o *Gmapping*, o *Move Base* e o *Exploration Lite*. Passando pelo driver do robô que permite a abstração do hardware, até os algoritmos de controle e navegação.

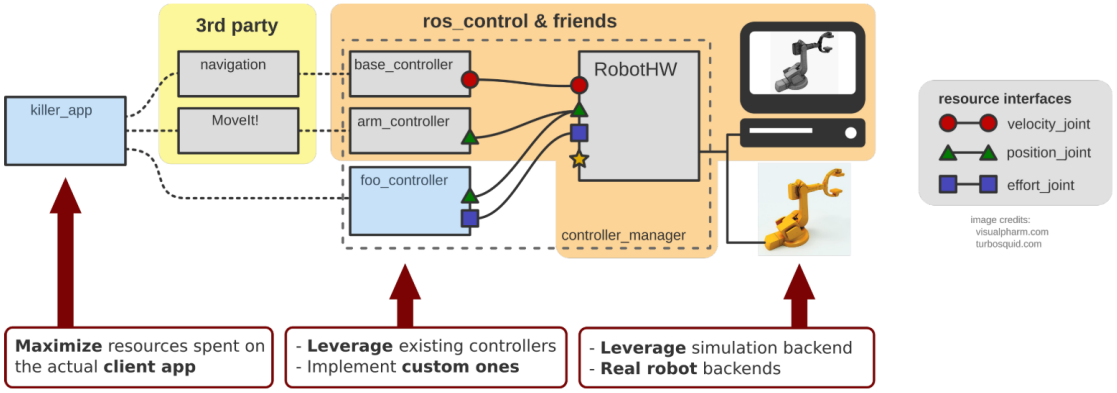


Figura 12 – Diagrama simplificado de operação do ROS control [Fonte: Chitta et al.[23]].

2.5.1.1 ROS *Control*

O *ros_control* permite a implementação e o gerenciamento dos controles do robô. É uma camada que possibilita a comunicação segura e em tempo real [23]. Sua principal função é fornecer a abstração de hardware e garantir que o ROS se comunique com robô sem distinção de desenvolvedor, ou seja, os mesmos controles podem ser utilizados para diferentes plataformas simplesmente modificando o código que descreve o robô [24]. A Figura 12 exibe as camadas que compõem o *ros_control*. No nível mais baixo, se encontra o robô e o simulador; seguindo pela classe *RobotHW* que implementa a interface *hardware_interface, esta* possibilita a abstração do hardware; passando para os controladores que determinam como o robô é movimentado; por fim os nodos que atribuem as funções do robô.

Em *hardware_interface* é construído o código que descreve o robô, denominado driver, ele é desenvolvido utilizando a interface *hardware_interface*. Esta interface permite a criação de um driver que conecta o robô desenvolvido com o ROS [25], isto permite que o robô utilize das ferramentas de controle, navegação, exploração e outras que estão desenvolvidas. Conforme apresentado no diagrama Figura 13, o *hardware_interface* implementa a descrição dos componentes de hardware como: limites de velocidade e conversão de torques; e se responsabiliza pela leitura dos sensores como encoder, para determinar a posição do motor.

As instancias desta classe permitem a integração dos recursos do robô como motores elétricos, atuadores hidráulicos, sensores de baixo nível como encoders e sensores de torque. Permite também a integração de hardware heterogêneo ou a troca de componentes de forma transparente, podendo ser aplicado a um robô real ou simulado. Desta forma desenvolvedores de robôs e pesquisadores podem desenvolver suas plataformas utilizando componentes de diferentes fornecedores, com diferentes protocolos de comunicação e manter a utilização dos algoritmos, sendo o único requisito fornecer com o robô um

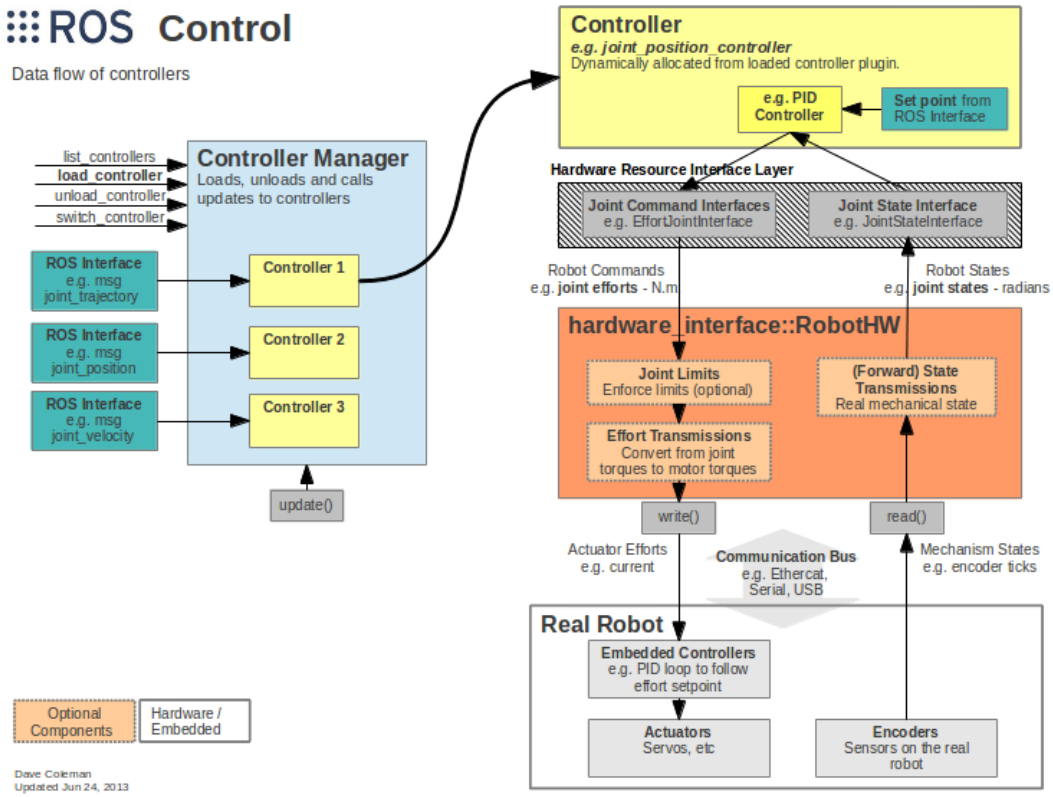


Figura 13 – Diagrama de operação do ROS control [Fonte: Chitta et al.[23]].

driver desenvolvido utilizando a interface.

Neste driver é desenvolvido as conversões de unidades de controle, para que o sistema se comunique de forma coerente com a aplicação, podendo ser aplicados os limites de operação dos componentes de forma individual como limite de velocidade, torque, aceleração e outros. Também é desenvolvido toda a comunicação entre os controladores e sensores, como sua inicialização, protocolos para escrita e leitura.

A inicialização e gerenciamento do *hardware_interface* é feita pela classe *controller_manager*, que se responsabilizada pela publicação e subscrição das mensagens pertinentes ao robô. Também inicializa os serviços e tópicos para que outras ferramentas possam controlar a plataforma. Dentro desta classe podem ser implementadas rotinas assíncronas ao controle, que em um intervalo de tempo maior, coletam dados de status do sistema e disponibiliza para o ROS. Estas informações podem variar de acordo com o robô utilizado, devido a não haver uma padronização para esta aplicação. A Figura 13, apresenta o *controller_manager* como sendo o responsável pelo gerenciamento e carregamento dos controladores.

Os *controllers* são responsáveis por ditar a dinâmica dos atuadores de acordo com o modelo de controle do robô, podendo ser um controle por posição, velocidade, trajetória, diferencial [26], torque, dentre outros. O controle é escolhido de acordo com a dinâmica

desejada para o robô, e caso não encontre um controle adequado é possível modelar um controle específico utilizando as classes relacionadas. Os controladores coletam informações do modelo físico do robô através do arquivo URDF, que é descrito na sequência.

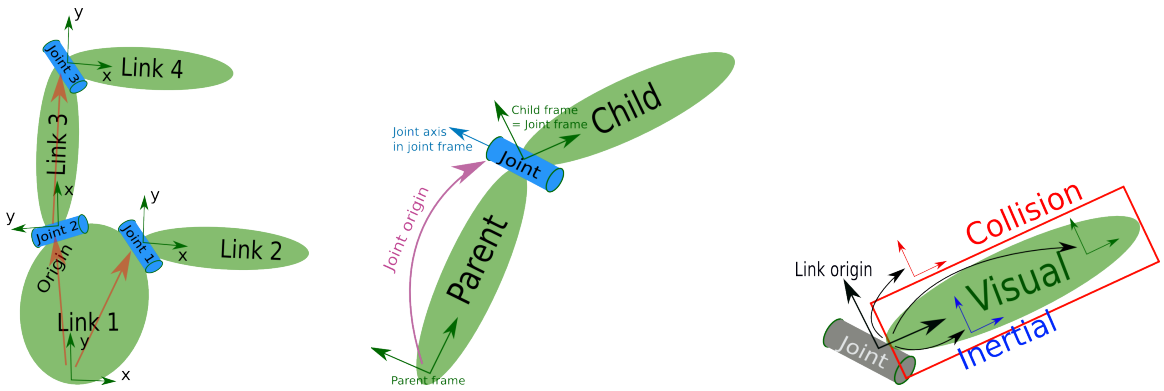
2.5.1.2 URDF

O URDF (Universal Robotic Description Format) é um arquivo XML (Extensible Markup Language) que permite descrever as características construtivas do robô para a sua utilização nas simulações do Gazebo ou nos modelos de controle do robô dentro do ROS [27]. Permite a descrição dos componentes utilizando camadas. Cada camada é determinada por um bloco de código com parâmetros que descrevem os componentes da simulação.

- Camadas físicas: para determinar os parâmetros de inércia e massa do robô.
- Camadas de colisão: para modelagem da área de conflito do robô utilizando formas simples.
- Camadas de visualização: permite o desenvolvimento de geometrias mais complexas para a representação do robô no ambiente simulado.

Os arquivos URDF são compostos pelos principais componentes listados a seguir:

- *robot*: descreve um robô com seus componentes no modelo URDF. É um parâmetro que engloba os demais itens da lista. Figura 14a.
- *sensor*: permite a implementação de sensores que serão utilizados no modelo simulado e define sua transformação de base no modelo real.



(a) *robot description.* (b) *joint description.* (c) *link description.*

Figura 14 – Camadas do URDF [Fonte: (a) ROS URDF model[1] (b) ROS URDF joint[2] (c) ROS URDF link[3]].

[1] http://wiki.ros.org/urdf/XML/model
[2] http://wiki.ros.org/urdf/XML/joint
[3] http://wiki.ros.org/urdf/XML/link

- *joint*: descreve o atuador, podendo ser prismático, de revolução, planar ou fixo. Todo *link* deve ser unido a outro *link* por um *joint*, caso não haja movimento, basta utilizar a opção fixo. Figura 14b.

- *link*: descreve um conjunto de elementos que define um corpo rígido, sendo parâmetros de inércia, de colisão e visual. Figura 14c.

- *transmission*: descreve a interface de acoplamento entre o atuador e o componente mecânico, como uma caixa de redução.

Outros componentes podem aparecer na descrição dependendo da necessidade do modelo. Os arquivos URDF são carregados na inicialização do ROS como parâmetro padrão *robot_description* e é utilizado pelos demais componentes do sistema para determinar o seu posicionamento durante a execução utilizando transformações de base fixas e moveis, e é essencial para descrever a odometria do robô.

2.5.1.3 *Gmapping*

O *gmapping* é um pacote que implementa os algoritmos utilizados no OpenSLAM [28][29]. O OpenSLAM é uma plataforma para pesquisadores de SLAM (Simultaneous Localization and Mapping). Este pacote permite o mapeamento de um ambiente utilizando o modelo SLAM, ou seja, o robô consegue gerar o mapa do ambiente enquanto se localiza.

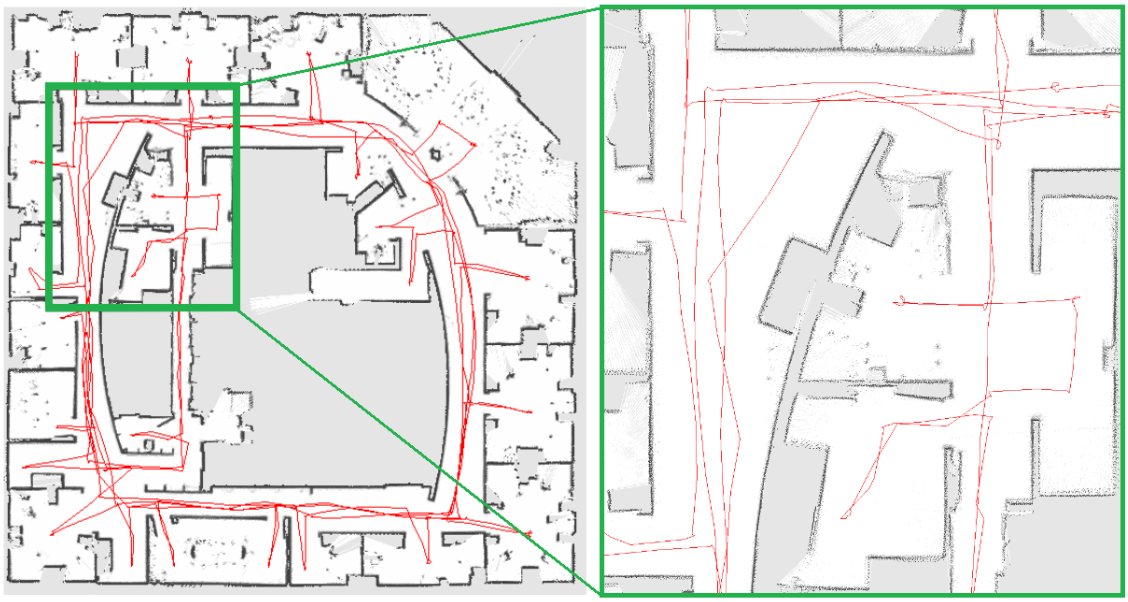


Figura 15 – Laboratório de pesquisa da Intel, mapa gerado utilizando o algoritmo. O retângulo verde é uma aproximação do mapa [Fonte: Adaptado de Grisettiyz, Stachniss e Burgard[28]].

O mapa é gerado através de uma nuvem de pontos obtida com dados coletados de um LIDAR 2D (Light Detection And Ranging), em conjunto com a odometria do robô. O LIDAR é um sensor que mede distância, porém devido as suas características mecânicas e elétricas ele faz este processo a uma frequência elevada e, consegue rotacionar em seu eixo, gerando uma nuvem de pontos 2D semelhante a um radar.

O algoritmo gera um mapa de ocupação no formato de matriz, onde identifica os obstáculos em preto, locais livres em branco e locais não explorados em cinza, conforme apresentado na Figura 15. Note que o robô circula várias vezes pelo ambiente, aumentando a precisão da leitura, e caso ocorra modificação o algoritmo é capaz de identificar e modificar o mapa.

2.5.1.4 *Move Base*

O *move_base* é um pacote que implementa um planejador de trajetória, dado um objetivo no mundo ele tentará alcançá-lo utilizando uma base móvel, neste caso o robô. O nodo permite a utilização de um planejador global e um planejador local. O planejador global olha para a tarefa final, busca dentro do mapa o melhor trajeto para cumprir o objetivo, leva em consideração obstáculos e o custo com base nas configurações definidas. O planejador local, leva em considerações as modificações que vão acontecendo durante o desenvolvimento da tarefa, ele é encarregado de desviar obstáculos fixos e móveis, prevenir colisões e caso o ambiente tenha sido modificado, remanejar a trajetória para que o robô consiga cumprir a tarefa.

Conforme apresentado no diagrama Figura 16, o *move_base* recebe informação de um objetivo, um local para onde deve ir, esta informação pode ser provida por um operador ou um outro nodo em uma camada superior. A informação é recebida primeiramente pelo planejador global, que analisa a informação com base na localização atual do robô, em conjunto com os dados dos sensores, uma função de custo global e o mapa, para fornecer o melhor trajeto para alcançar o objetivo, com base nos parâmetros estipulados. Em seguida a informação é transmitida para o planejador local, que refina os dados levando em consideração uma função de custo local, os sensores e a odometria do robô. Estes dados então são publicados como uma velocidade em x, y z, roll, pitch e yaw, que podem ser enviados para o ROS control.

Este nodo utiliza dos dados do arquivo URDF para o conhecimento das dimensões da plataforma e permite a entrada diversos parâmetros que vão refinar a ferramenta para cada caso de aplicação. É possível expandir a área do robô para aumentar a segurança e evitar colisões; escolher diferentes algoritmos de custo e planejamento que são implementados dentro da biblioteca *nav_core* fornecida junto com o pacote; maior e menor raios de rotação, velocidades, aceleração; dentre outros parâmetros.

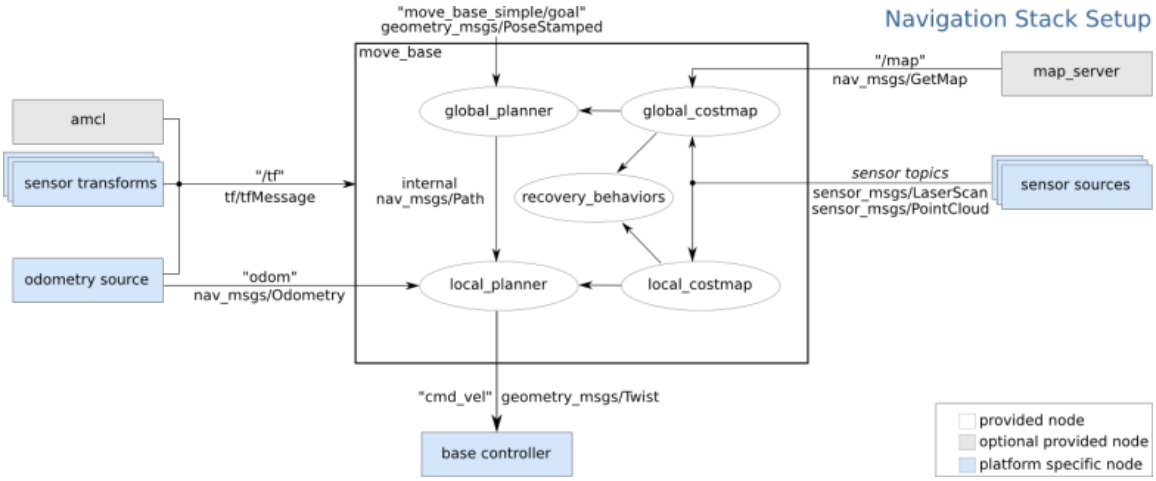


Figura 16 – Fluxograma simplificado de operação do *move_base* [Fonte: ROS move base[1]].

2.5.1.5 *Exploration Lite*

O *exploration_lite* é um pacote que, em conjunto com o *move_base* como sendo um nodo de navegação e o *gmapping* como sendo um nodo de SLAM, permite que o robô explore o universo no qual está localizado. Este algoritmo combina os dados dos nodos conforme a Figura 17, e busca por fronteiras não exploradas.

O princípio de funcionamento do algoritmo é bastante simples, o robô se encontra em um ambiente desconhecido. Ao inicializar a operação o nodo de SLAM começa a construir um mapa, identificando os obstáculos no ambiente. Neste momento o *exploration_lite* começa a identificar os locais onde não existe obstáculos, onde os locais conhecidos em cinza claro e desconhecidos em cinza escuro se encontram, nestes pontos o algoritmo entende que o mapa não está completamente explorado. Através de um algoritmo de custo ele decide qual dos pontos explorar e envia o robô para aquele ponto, utilizando o nodo de navegação. Este processo irá se repetira até que o robô explore todo o mapa e não encontre mais fronteiras abertas [30].

A Figura 18 mostra o resultado de um processo em andamento. O mapa é construído utilizado um scaner de área que identifica os obstáculos, porém devido as limitações da resolução do scaner é possível observar espaços não explorados no meio da varredura, estes

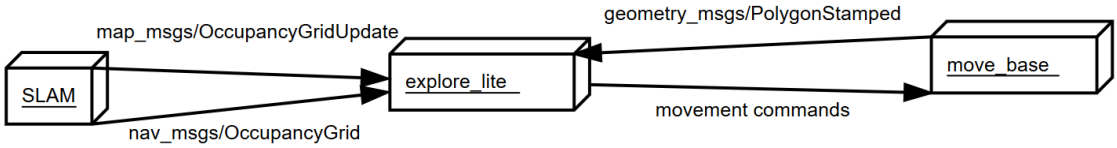


Figura 17 – Fluxograma simplificado de operação do *exploration_lite* [Fonte: ROS exploration lite[2]].

[1] http://wiki.ros.org/move_base
[2] http://wiki.ros.org/explore_lite

pontos também serão explorados pelo algoritmo. Em azul são as fronteiras inexploradas pelo robô, em preto os obstáculos localizados e em verde os pontos onde o algoritmo vai enviar o robô com base na função de custo.

Figura 18 – Resultado do *exploratio_lite* [Fonte: ROS exploration lite[1]].

[1]http://wiki.ros.org/explore_lite

3 Concepção, projeto e desenvolvimento de um robô móvel

Neste capítulo serão tratadas as considerações sobre o projeto do robô, iniciando pela estrutura mecânica, delineando e justificando as escolhas dos componentes. Na sequência será tratado o projeto eletrônico que envolve os controladores de potência e o computador que controla o robô. Por fim será apresentado a implementação do software que integra a plataforma com o ROS e abordado como é estruturado o sistema de navegação para desenvolver o mapeamento de uma área utilizando o *exploration_lite*.

3.1 Definições e metodologia

Robôs móveis sobre rodas são dispositivos com capacidade de se deslocar sobre uma superfície em um ambiente que pode ser estruturado ou desestruturado, interno ou externo e possuir ou não obstáculos fixos ou móveis. Estas e outras características são atribuídas ao robô na fase inicial do projeto e permite que o projetista consiga desenvolver o trabalho de forma objetiva. Para este projeto o robô é caracterizado de duas formas:

- Global: determina as características que envolvem todos os sistemas do robô.
- Específica: determina as características individuais de cada sistema (mecânico, eletrônico e software).

Algumas características foram definidas a partir dos objetivos do trabalho. Examinando os projetos comerciais e refinando o escopo foram traçadas as características globais que são:

- Elaborar um projeto flexível e utilizar materiais comerciais.
- Desenvolver um projeto replicável.
- Operar em terrenos regulares e irregulares, internos e externos.
- Possuir as capacidades e definições da Tabela 2.
- Ser controlado por uma estação conforme o diagrama (Figura 19), onde, o robô possui a capacidade de executar as tarefas de forma autônoma, mas definidas e parametrizadas pela estação.

Tabela 2 – Especificações esperadas para o projeto.

Massa do robô + carga [kg]	50
Num. motores [#]	4
Diâmetro da roda [m]	0,4
Velocidade do robô [m/s]	1,5
Inclinação da superfície [°]	25
Tensão motor [V]	24
Aceleração [m/s^2]	0,5
Tempo de operação [h]	3
Eficiência [%]	65

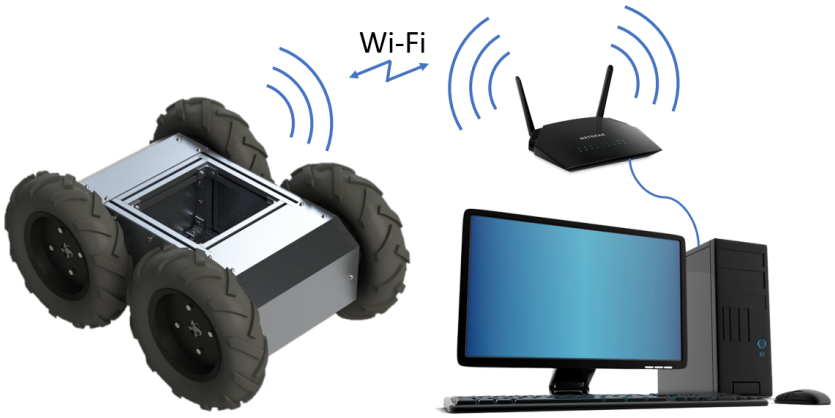


Figura 19 – Topologia de comunicação.

Estas características definem os subsistemas do robô e, em conjunto com os objetivos individuais de cada subsistemas é possível definir as características especificas e elaborar as tarefas a serem desenvolvidas para alcançá-las. Para o projeto mecânico é esperado:

- Desenvolver uma plataforma skid steer, com 4 rodas.
- Projetar e construir uma estrutura robusta.
- Mecânica modular, para permitir a fácil substituição e reestruturação de componentes.
- Desenhar um projeto de fácil manufatura, para facilitar a produção e reduzir o custo.

Para alcançar estes objetivos o projeto mecânico foi desenvolvido de forma iterativa e seguindo as seguintes atividades:

- Pesquisa e análise de componentes comerciais.
- Desenvolvimento do projeto mecânico em software CAD (Computer-Aided Design).
- Análise de montagem dos componentes.

- Efetuar estudos e análise de resistência com software FEA (Finite Element Analysis).

Da mesma forma foram definidas as características especificas para o sistema eletrônico, que são:

- Implementar sensores que permitam o robô navegar de forma autônoma.
- Possibilitar a integração de novos sensores, para poder aprimorar as funcionalidades do robô.

As atividades para o desenvolvimento do projeto eletrônico foram a pesquisa e a análise de componentes comerciais que atendiam as especificações do projeto. O sistema eletrônico não foi o foco do estudo, sendo desenvolvido apenas para dar suporte a mecânica e ao software. Por fim foram definidas as características especificas para o software, que são:

- Integrar ao ROS.
- Desenvolver uma atividade utilizando o ROS para testar a integração, a exploração autônoma.

Para alcançar os objetivos do projeto de software foram desenvolvidas as seguintes atividades:

- Desenvolvimento de um algoritmo para integrar o robô ao ROS, denotado *driver*.
- Parametrizar alguns pacotes do ROS para o desenvolvimento de exploração autônoma.

Ao final foi parametrizadas rotinas no ROS para o cumprimento de determinadas tarefas. Todo o processamento foi desenvolvido dentro do robô, para isso foi integrado um computador que roda o ROS *master*. A transmissão das tarefas, telemetria do robô e tele operação, foi desenvolvida em uma outra estação(computador), que está conectado ao robô por uma rede Wifi, está estação deve possuir o ROS instalado e consegue se comunicar com o *master*. A topologia é apresentada na Figura 19

3.2 Projeto mecânico

O projeto mecânico foi desenvolvido com base nas especificações globais e especificas. Foi determinado que a plataforma deve seguir um modelo skid steer, pois, opera em terrenos irregulares, pode operar em ambientes internos e externos e possui uma estrutura robusta. Outro requisito imposto foi utilização de quatro rodas, que simplifica a implementação, manutenção e reduz o custo se comparado com esteiras. Com base nestas

considerações somadas com a análise dos produtos já presentes no mercado foi estipulado o escopo do projeto mecânico.

- Dimensionamento dos motores.
- Análise de produtos comerciais.
- Proposta de estrutura.
- Simulação da estrutura.

Com as especificações traçadas foi iniciado o desenvolvimento do projeto que se segue nas próximas seções. Aborda o dimensionamento do sistema motriz, a escolha dos componentes mecânicos comerciais e a modelagem dos componentes estruturais.

3.2.1 Dimensionamento do motor

Um dos principais limitante do projeto de uma plataforma móvel é o sistema de tração, composto pelo motor que fornece a força para o robô, a caixa de redução que ajusta a força e o acoplamento que conecta aos componentes mecânicos. Ele é que vai indicar por quais tipos de superfície pode se deslocar e quanto peso ele pode carregar. Para o dimensionamento do sistema motriz as especificações do projeto precisam ser convertidas em especificações de motores [31]. Primeiramente encontramos a velocidade de rotação do motor utilizando a equação:

$$f_r = \frac{v_l}{2\pi r} \ [Hz] \tag{3.1}$$

onde, f_r é a frequência de rotação e v_l é a velocidade linear do robô.

Encontrando a frequência de rotação é possível encontrar a velocidade em RPM (Rotações por Minuto), que é um parâmetro mais comum em motores comerciais, para isso utiliza-se a equação a seguir:

$$V_r = 60 f_r \ [RPM] \tag{3.2}$$

onde V_r é a velocidade de rotação em RPM.

Outra informação importante na hora de dimensionar um motor é o torque. Para encontrar o torque exigido pelo motor é utilizado a seguinte equação:

$$\tau = \frac{rm(a + g\sin(\alpha))}{N_m} N_{\%} \ [Nm] \tag{3.3}$$

onde, τ é o torque exigido do motor, r o raio da roda, m a massa do robô, a a aceleração esperada para o robô, g a gravidade, α o ângulo que o robô consegue escalar, N_m o número de motores e $N_{\%}$ a eficiência do sistema considerando conjunto mecânico e motor.

Outro parâmetro encontrado na hora de escolher um motor é a potência, para o cálculo da potência segue a equação:

$$P = 2\pi\tau f_r \ [W] \tag{3.4}$$

Tabela 3 – Parâmetros necessários para o dimensionamento dos motores.

Velocidade [Hz] Eq. 3.1	1,19
Velocidade [RPM] Eq. 3.2	71,62
Torque por motor [Nm] Eq. 3.3	17,86
Potência p/ Motor [W] Eq. 3.4	133,98
Corrente p/ motor [A] Eq. 3.5	5,58

onde P é a potência em Watts. Por fim, é possível estimar a corrente esperada que o robô irá consumir, para o correto dimensionamento das baterias, para isso é utilizada a equação:

$$i_{max} = \frac{P}{v}\ [A] \tag{3.5}$$

onde i_{max} é a máxima corrente esperada e v a tenção do motor, neste caso a mesma tensão da bateria.

Com isso é obtida a Tabela 3, que exibe os parâmetros para o dimensionamento dos motores. Com base nas especificações estabelecida é preciso encontrar um motor comercial que atenda aos requisitos. Feita uma pesquisa de mercado foi encontrada a possibilidade de comprar o motor separado da caixa de redução ou montado, porém, comprar os componentes separados aumenta o preço do produto e adiciona a complexibilidade de encontrar modelos compatíveis. Deste modo foi optado pela compra dos componentes montados que são encontrados em dois modelos:

- Motores com caixa de redução de rosca sem fim: este modelo faz com que o eixo da roda fique deslocado 90º do eixo do motor, Figura 20a.
- Motores com caixa de redução planetária: este motor é inteiramente disposto em linha com a caixa de redução, Figura 20b.

Foi estudada a possibilidade de utilizar motores com caixa de redução rosca sem fim. Este modelo reduz consideravelmente a distância entre eixo do robô, pois divide seus componentes uma parte colinear ao eixo da roda e outra parte perpendicular, isto pode ser

(a) Motor e redução rosca sem fim. (b) Motor e redução rosca planetária.

Figura 20 – Motor e redução considerados para o projeto.

visto claramente na Figura 20. A distância entre eixo deve ser controlada para melhorar a dinâmica do robô devido a sua característica de arrasto. Outra vantagem deste conjunto é que seu sistema opera como um freio natural, sem deslocamento no eixo do motor não a deslocamento da roda. Porém a obtenção deste tipo de sistema no mercado é muito difícil o que inviabilizou sua utilização, devido a busca de componentes de fácil acesso.

A segunda opção foi escolhida por ser mais acessível no mercado e, o mesmo sistema já é comummente utilizado em projetos de robôs. Feita uma pesquisa em distribuidores e uma análise das especificações levara a um conjunto de motor e caixa de redução fornecida pela Maxon. Esta empresa permite a configuração do conjunto motor-redução, podendo escolher entre diferentes relações de caixa para determinado modelo de motor. O motor foi escolhido com base nas especificações da Tabela 3 com o parâmetro de potência, foi considerado o motor que atenda este parâmetro com uma margem de segurança. O motor selecionado possui o código 148867 e fornece 150W, 6940 RPM e 117mNm com carga, é apresentado na Figura 21a.

Como este motor permite a escolha da caixa de redução com diferentes relações de velocidade e torque, foram avaliadas as configurações disponibilizadas pelo fabricante, não encontrada nenhuma que atenda todos os parâmetros da Tabela 3. A questão é, a caixa de redução permite multiplicar os valores de torque do motor e inversamente modificar sua velocidade, ou seja, ao escolher uma caixa que aumenta o torque, proporcionalmente reduz a velocidade. Como apresentado nas equações:

$$\tau_{nom.eixo} = \tau_{nom.motor} N_{relação} \; [Nm] \tag{3.6}$$

$$v_{nom.eixo} = \frac{v_{nom.motor}}{N_{relação}} \; [m/s] \tag{3.7}$$

onde $N_{relação}$ é a relação de redução da caixa. O fabricante disponibiliza dois valores de caixa de redução que se enquadram nas especificações do projeto, sendo elas com relação 1:91 e 1:113. Obtendo a relação de torque, rotação, velocidade e inclinação apresentada na Tabela 4.

Cada uma das caixas de redução atende apenas uma das especificações e se aproxima da outra. Buscando um melhor rendimento em baixas velocidades e maior torque no eixo foi selecionada a caixa de redução planetária código 203126 com redução de 1:113, Figura 21b.

Tabela 4 – Parâmetros para o dimensionamento da caixa de redução.

Relação	**Ideal**	**1:91**	**1:113**
Velocidade nominal no eixo [RPM] Eq. 3.7	71,62	76,26	61,42
Torque nominal no eixo [Nm] Eq. 3.6	17,86	16,11	20,00
Velocidade nominal [m/s]	1,50	1,60	1,29
Inclinação da superfície [º]	25	22,08	28,63

(a) Motor Maxon. (b) Caixa de redução Maxon

Figura 21 – Motor e caixa de redução utilizados [Fonte: Maxon[1]].

3.2.2 Alinhamento dos Motores

Com a escolha do motor foi observado que apenas com a soma da largura dos motores é obtida uma largura de 305mm, sem a adição de outros componentes. Devido ao modelo dinâmico do robô, quanto maior a distância entre eixo, maior vai ser o arrasto gerado pelas rodas. Pensando nisso foram levantadas duas possibilidades:

- Fixação dos eixos em linha: todo o sistema motriz esquerdo e direito ficam colineares e o acoplamento do motor com eixo é feito por um acoplador de eixo Rotex 24 ST. Este modelo conta com toda a extensão do sistema motriz somada a largura do robô. Como mostra a Figura 22a.

- Fixação dos eixos deslocados: o conjunto roda e eixo, esquerdo e direito são colineares, os motores são acoplados ao eixo por correia dentada e polia. Este método permite que os motores estejam paralelos, porém deslocados, reduzindo a largura do robô no equivalente a largura de um motor. Como mostra a Figura 22b.

Modelado os sistemas foi obtido que: a primeira proposta fornece uma largura de chassis de 600mm e a segunda proposta fornece uma largura de 350mm. Como largura entre eixos influência na dinâmica do robô, a primeira opção é descartada com base na largura. Por exclusão a segunda opção é a mais interessante, porém deve-se observar que o acoplamento do eixo por correia e polia adiciona uma dificuldade a mais para o projeto, tendo que fazer seu devido alinhamento e tensionamento.

3.2.3 Mancais e eixos das rodas

O correto dimensionamento dos mancais e do eixo das rodas atribuem as características de robustez esperadas para o projeto, sendo eles os responsáveis por sustentar toda a

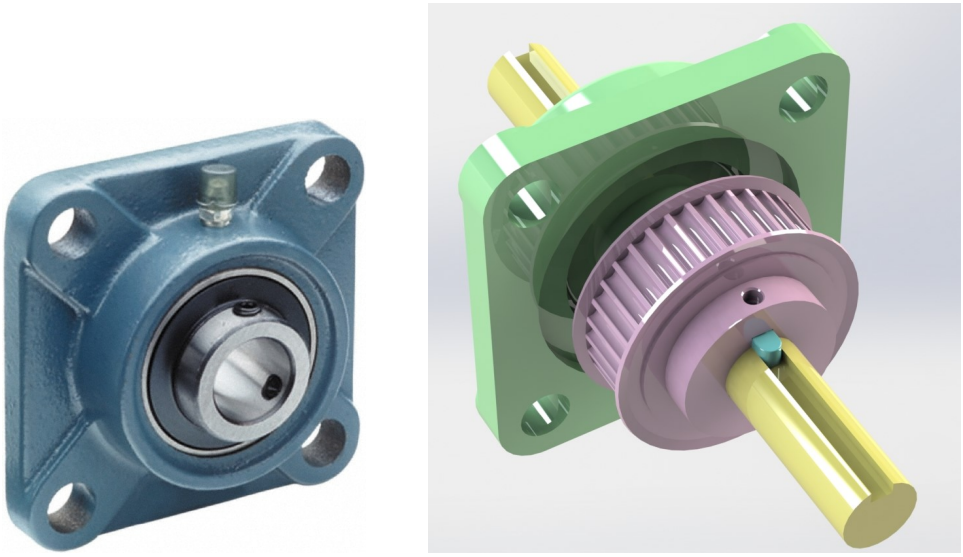
[1] https://www.maxonmotor.com/medias/sys_master/root/8824674844702/148866-RE40-GB-150W-KL-2WE.zip

(a) Motores colineares. (b) Motores deslocados com polia e correia.

Figura 22 – Alinhamento dos motores com o eixo da roda.

carga imposta sobre a estrutura. Existem mancais comerciais capazes corrigir pequenos desalinhamentos de eixo sem comprometer seu funcionamento e facilita bastante a montagem. Feita uma análise e catálogos de mancais comerciais foi observado que o modelo compatível com as características do robô comporta um eixo de 17mm. O mancal selecionado foi o UCF203, Figura 23a, é utilizado para fixação acoplado a chapas. O modelo conta com rolamento e é capaz de lidar com pequenos desalinhamentos de eixo.

Com a escolha do mancal foi definido que o projeto deve contar com um eixo de roda com diâmetro de 17mm. O eixo precisa sustentar a carga do robô e o torque imposto pelos motores. Com esta preocupação foi determinado que o acoplamento dos componentes ao eixo deve ser feito por chaveta. Este método de acoplamento garante a integra transmissão de torque e integridade do eixo, se comparado a simples utilização de parafuso. Com esta definição, foram dimensionados os eixos das rodas com rebaixo para chaveta, assim como



(a) Mancal UCF203. (b) Eixo e componentes com chaveta.

Figura 23 – Componentes mecânicos utilizados [Fonte: (a) Inficreator[1]].

[1]http://www.inficreator.com/image/cache/catalog/pi-m_4boltsf-500x500.jpg

os demais componentes que conectam ao eixo, conforme a Figura 23b, onde apresenta a chaveta em azul, o eixo em amarelo, o mancal em verde e a polia em lilás.

3.2.4 Conjunto de roda

O sistema de montagem das rodas deve ser simples. Para a manutenção as vezes é necessário remover a roda, e em uma aplicação futura, as rodas podem ser substituídas de acordo com a tarefa, podendo ser com garras, lisas, mais largas. Alguns sistemas possuem o eixo do motor soldado a roda dificultando este processo. Pensando nestas características e em modularidade, foi desenvolvido um conjunto para fixar a roda ao eixo, denotado conjunto de roda. Uma ponta de eixo (verde) é fixada ao eixo da roda (amarelo) por parafuso e chaveta (azul), esta peça tem a finalidade de transmitir o torque para a roda. Na sequência um cubo de roda (lilás) é fixado a roda (laranja). A ponta de eixo e o cubo de roda fazem a ligação entre o eixo e a roda através de parafusos (vermelho), rosqueados na ponta de eixo. O sistema pode ser observado na Figura 23b.

3.2.5 Estrutura Mecânica

O desenvolvimento de uma estrutura mecânica simples pode reduzir o custo de desenvolvimento do projeto significativamente, este não está somente relacionado a quantidade de material, mas também as técnicas de manufatura utilizadas, porém é preciso ficar atento, pois uma estrutura muito complexa pode tornar o sistema de difícil montagem e/ou manutenção. Da mesma forma, o projeto da estrutura vai definir a robustez da

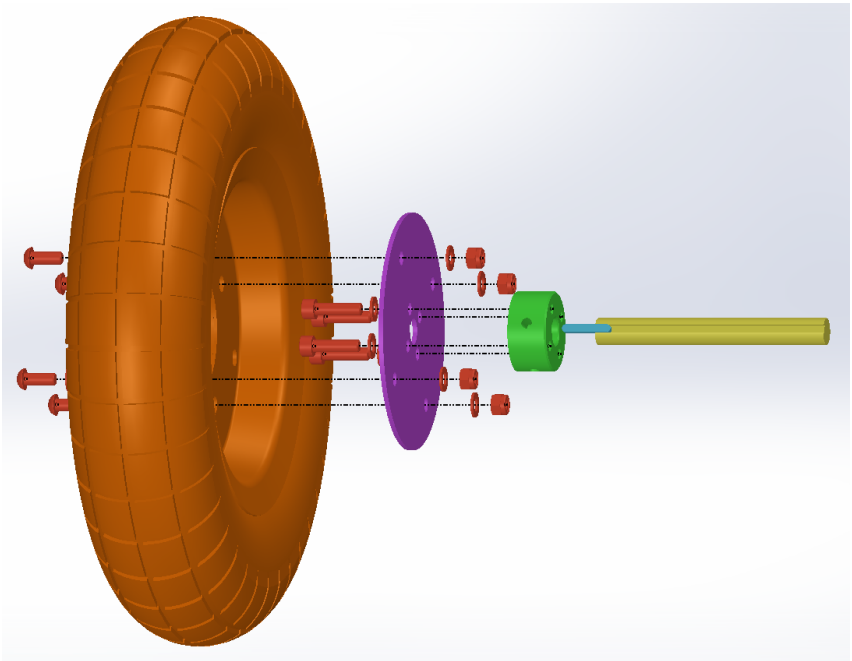
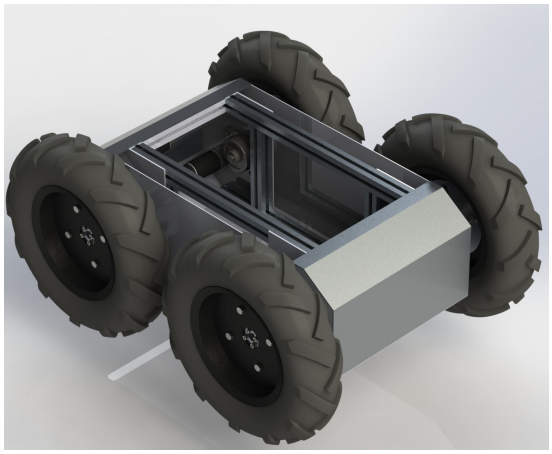
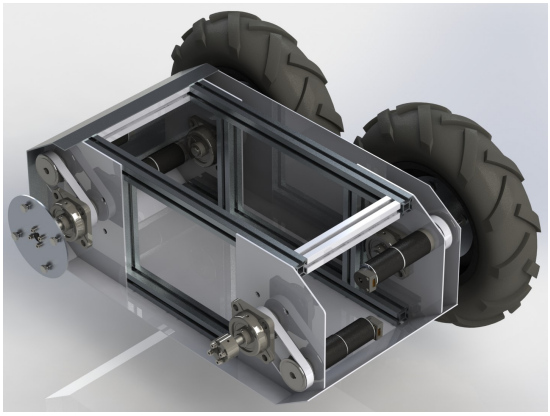


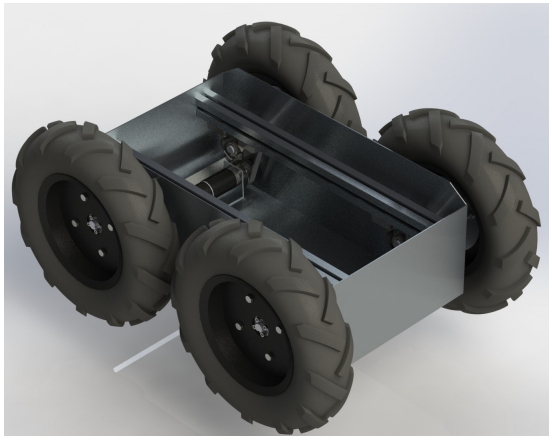
Figura 24 – Conjunto de roda com o eixo e a roda.



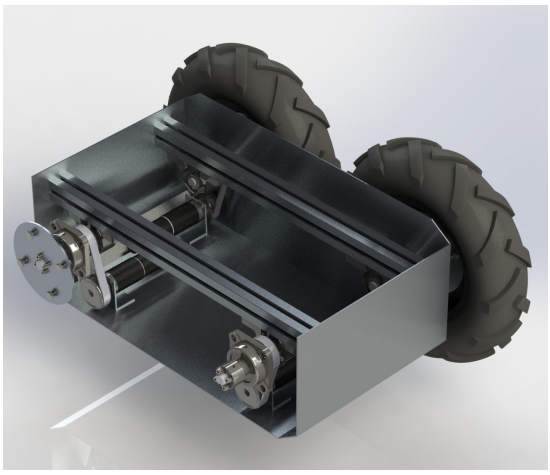
(a) Proposta chassis em perfil de alumínio.



(b) Proposta chassis em perfil de alumínio.



(c) Proposta utilizando a carenagem como base estrutural.



(d) Proposta utilizando a carenagem como base estrutural.

Figura 25 – Propostas de estrutura mecânica.

plataforma. Pensando nisso foram elaboradas duas propostas:

- Chassis em perfil de alumínio extrudado, utilizando as chapas apenas para carenagem, este modelo permite o total desacoplamento do sistema da carenagem facilitando montagem e manutenção. Como mostra a Figura 25a e 25b.
- Chassis acoplado a carenagem, utilizando de cortes e dobras mais complexos, aproveita a carenagem como suporte estrutural. Como mostra a Figura 25c e 25d.

A primeira opção apresentou um projeto mais robusto e modular. Seu sistema permite a operação do robô desacoplado da carenagem, além de abrir espaço para a reformulação de sua largura apenas substituindo os perfis centrais. Esta proposta ainda facilita a montagem e manutenção, visto que os módulos são simétricos e desacoplados, podendo efetuar a sua montagem separada do todo e unir apenas no final. Seu desenho é simétrico e permite a utilização das partes como sendo módulos, além de centralizar o centro de massa o que facilita para o controle em modelos skid steer. Todo o projeto mecânico

pode ser encontrado no repositório do projeto[1], que conta com o detalhamento das peças, processo de montagem e o modelo em 3D[2].

3.2.6 Simulação mecânica

Durante o processo de desenvolvimento do projeto, muitas dúvidas sobre a estabilidade do sistema vão surgindo, dificultando determinar alguns parâmetros cruciais para o desenvolvimento, como:

- Carga suportada nos elementos;
- Torque suportado nos eixos;
- Deformação dos componentes com a presença de carga.

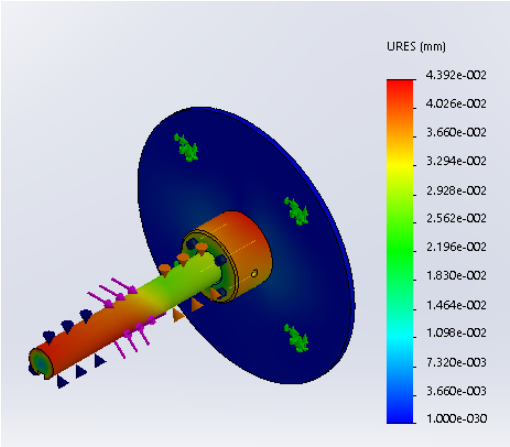
É difícil determinar, sem estudo, se o sistema atende as especificações desejadas. Uma das formas seria prototipar o projeto e fazer testes para verificar se os parâmetros projetados corroboram com os testes do protótipo, este método pode ser muito acurado porem muito custoso nas fazes iniciais do projeto. Para contornar o problema e desenvolver a plataforma de forma iterativa buscando soluções que atendam os parâmetros e com um custo reduzido de desenvolvimento, o sistema foi submetido a softwares FEA, capaz de determinar com baixa margem de erro se o sistema modelado atende as especificações.

As simulações executadas foram desenvolvidas para testar as especificações do projeto, e mais, testar o quão robusto é este frente as especificações. Esta parte do trabalho foi desenvolvida de forma iterativa, desenvolvendo o projeto mecânico e simulando, caso o resultado não fosse o esperado, o projeto era modificado e simulado novamente, até alcançar os resultados. O objetivo era o desenvolvimento de uma estrutura rígida com pouca deformação com aplicação de carga. As rotinas de simulação foram modeladas buscando avaliar os sistemas críticos do robô, sendo estes:

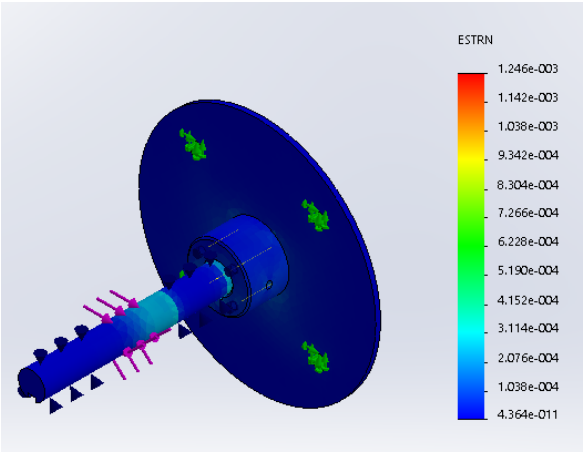
- A fixação do cubo de roda com a ponta de eixo e o eixo;
- A chapa que fixa o motor e o eixo que os conectando por uma correia;
- O perfil lateral composto pela estrutura e o conjunto dos motores;
- A montagem final.

Os testes e os procedimento adotados são descritos nas seções seguintes.

[1] https://github.com/marcelopetry/ARP2/tree/master/drawings
[2] https://1drv.ms/u/s!AvyRvYEIor_5i_Y_BG6IPfdy78uQxw?e=PGD8Ct



(a) Deslocamento do conjunto de roda com aplicação de um torque de 30Nm.



(b) Deformação do conjunto de roda com aplicação de um torque de 30Nm.

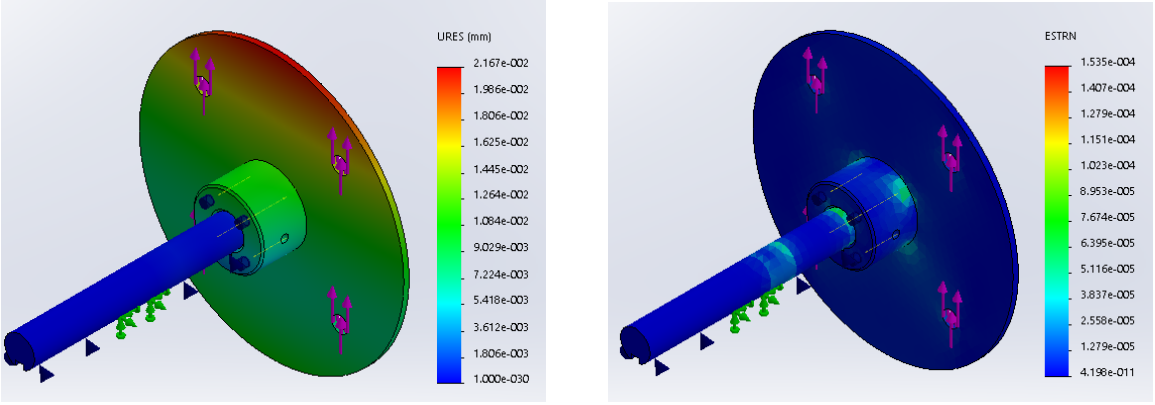
Figura 26 – Simulação do conjunto de roda com aplicação de torque.

3.2.6.1 Conjunto de roda

O conjunto de roda é uma peça fundamental para este projeto, permite a fácil remoção do conjunto Aro/Pneu para manutenção e/ou modificação. É importante garantir que este sistema consiga suportar o torque aplicado a roda sem haver deformação plástica e com uma margem de segurança. Deve ser capaz de suportar 1/4 da carga do robô sem haver deformação plástica, também com uma margem de segurança.

Inicialmente foi testado se o sistema é capaz de suportar o torque exercido pelo motor. Para este teste o eixo foi restringido nos dois pontos de contato com os mancais (triângulos), permitindo apenas o movimento de rotação axial. No ponto onde é fixada a polia (setas rosas) foi aplicado um torque de 30Nm, que é corresponde ao torque do motor mais uma margem de segurança. Nos pontos de fixação da roda ao cubo de roda foi fixado o movimento (seta verde). Este teste simula o robô parado com torque máximo do motor aplicado ao eixo da roda, uma condição onde a roda está travada e o motor aplica torque sobre o sistema. Esta simulação permite visualizar a integridade dos componentes mecânicos sobre esta condição e verificar se os componentes de fixação estão corretamente dimensionados. A Figura 26a apresenta a deformação elástica do sistema com a aplicação do torque, é possível notar que o maior deslocamento está no eixo (escala vermelha), sendo o maior deslocamento 0,00439mm. A Figura 26b mostra a deformação plástica do sistema, o ponto de maior deformação é o ponto de contato da chaveta com o cubo de roda.

Na sequência o sistema foi submetido a aplicação de uma força perpendicular ao eixo, estes testes têm o objetivo de determinar se o sistema é capaz de suportar a carga estipulada pelo projeto. Para a simulação os pontos de contato com os mancais são restringidos (triângulos), permitindo apenas o movimento de rotação axial. O ponto de



(a) Deslocamento do conjunto de roda com aplicação de uma carga de 250N.

(b) deformação do conjunto de roda com aplicação de uma carga de 250N.

Figura 27 – Simulação do conjunto de roda com aplicação de carga.

contato da polia com o eixo é fixo (setas verdes) e uma força é aplicada sobre os pontos de fixação do cubo de roda com o aro (setas rosas). Esta simulação permite avaliar a deformação do eixo sobre a condição de máxima carga, para o teste foi considerado que a carga total do robô pode chegar a 100kg, resultando em uma força de 250N por roda. A Figura 27a apresenta a deformação plástica do sistema com a aplicação da carga, fica evidente que o maior deslocamento está na periferia do cubo de roda (escala vermelha) e corresponde a 0,02167mm. Já a Figura 27b mostra a deformação plástica do sistema, neste existe uma deformação equivalente no contato do cubo de roda com os componentes de fixação.

3.2.6.2 Placa do motor

A placa do motor tem por objetivo fixar os componentes do sistema motriz (Figura 28), sendo eles o motor, o mancal interno e o sistema de correia, polia e esticador. Devido ao seu projeto mecânico, este sistema deve ser capaz de suportar a tenção sobre a correia, a carga exercida pelo eixo da roda devido a massa do robô e o torque ocasionado pelo movimento do motor. Visando facilitar a simulação cada uma das cargas foi simulada individualmente.

Para a simulação da força exercida pela correia sobre a placa, o sistema foi fixado nos pontos de fixação externos (setas verdes) e duas forças foram aplicadas (setas rosas), uma nos pontos de fixação do motor com direção ao eixo de tração e outra nos pontos de fixação do mancal com direção ao motor, cada uma com 100N, para simular a tensão da correia esticada. Como as forças são colineares e em sentidos opostos, estas vão comprimir o sistema, simulando a tensão da correia. A Figura 29a apresenta a deformação plástica do sistema com a aplicação da carga, fica evidente um deslocamento da região do motor (escala vermelha). Já a Figura 29b mostra a deformação plástica do sistema, que se concentra na região entre o motor e o mancal.

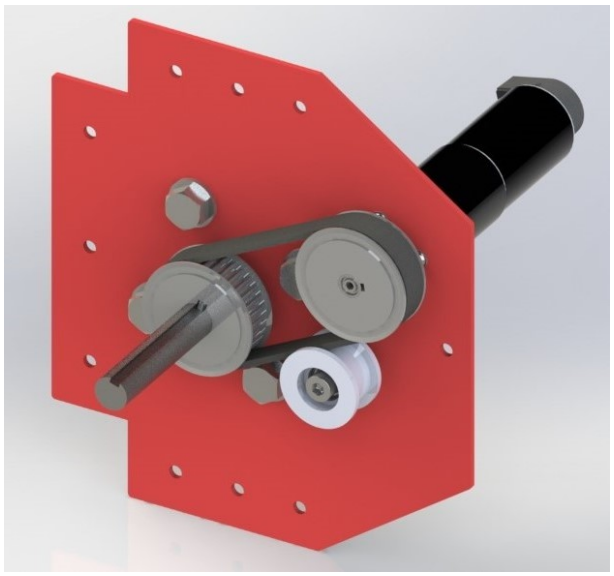
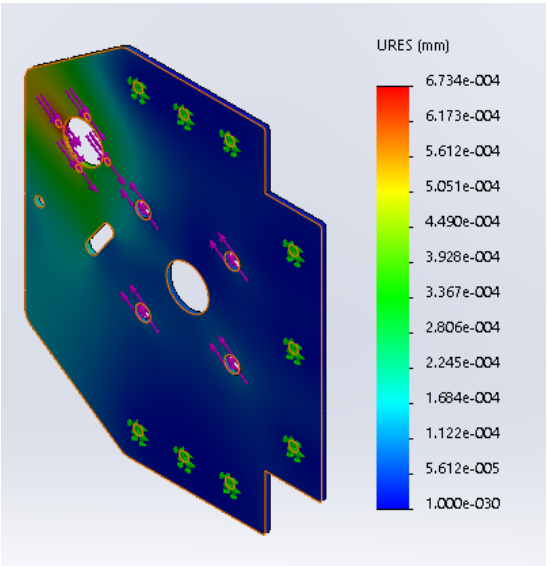
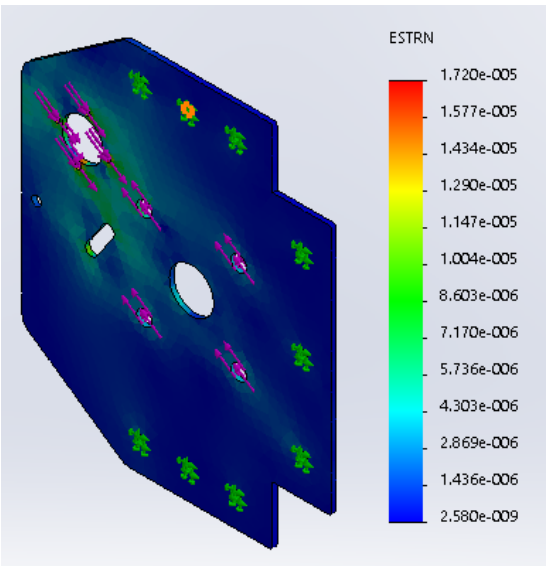


Figura 28 – Conjunto de componentes do sistema motriz.

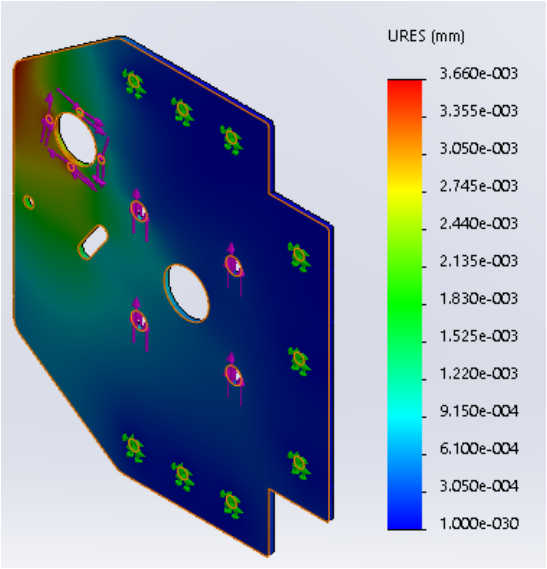


(a) Deslocamento da placa de fixação do sistema motriz com aplicação de uma carga de 200N.

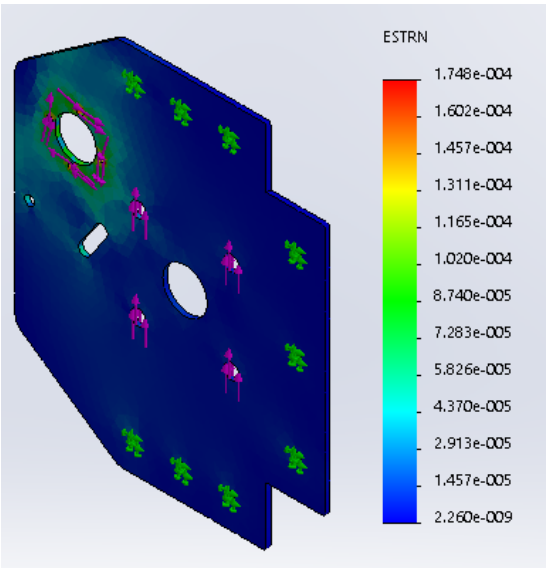


(b) Deformação da placa de fixação do sistema motriz com aplicação de uma carga de 200N.

Figura 29 – Simulação da placa de fixação do sistema motriz com aplicação de carga.



(a) Deslocamento da placa de fixação do sistema motriz com aplicação de uma carga de 250N e um torque de 30Nm sobre o motor.



(b) Deformação da placa de fixação do sistema motriz com aplicação de uma carga de 250N e um torque de 30Nm sobre o motor.

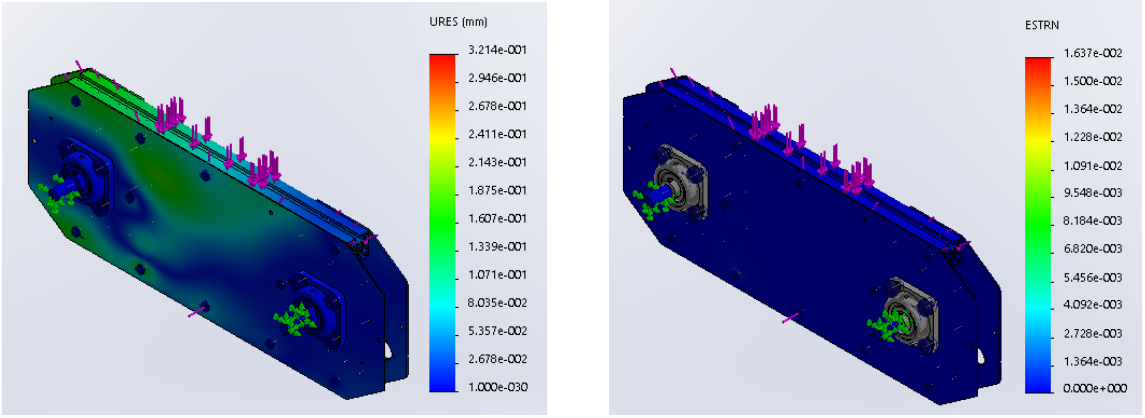
Figura 30 – Simulação da placa de fixação do sistema motriz com aplicação de carga e torque.

A força exercida pela carga do robô sobre a placa foi simulada fixando os pontos de fixação externos (setas verdes). A força é aplicada sobre os pontos de fixação do mancal (setas rosas) com direção contraria a força exercida pela massa do robô, é considerado para o teste que a massa total é de 100kg, portanto 250N por roda. Devido ao torque do motor, outro ponto a ser analisado é se a chapa é capaz de suportar o torque quando o eixo está bloqueado, suportando o torque total do motor (setas rosas), que foi considerado 30Nm. A Figura 29a apresenta a deformação plástica do sistema com a aplicação das cargas, fica evidente um deslocamento na região do motor (escala vermelha), sendo este um ponto onde o sistema não está fixo a outra estrutura. Já a Figura 29b mostra a deformação plástica do sistema, que se concentra na região que é posicionado o motor.

3.2.6.3 Conjunto de tração

O conjunto de tração compõe toda a lateral do veículo, é responsável pela fixação de dois sistemas motrizes e por suportar metade da carga do robô. Este sistema determina a robustez da plataforma e sua capacidade de carga. Devido ao seu projeto utilizando perfil de alumínio, permite o acoplamento de outros dispositivos e de carga ao robô com facilidade devido a presença de guias para este propósito.

O sistema precisa ter sustentação mínima para metade da carga do robô. Para garantir este parâmetro o sistema foi simulado, fixando os dois eixos de tração (setas verdes) e aplicando uma carga na base superior do sistema (setas rosas), a força é calculada como sendo metade da carga, que foi considerada 100kg, ou seja, 500N. A Figura 31a apresenta



(a) Deslocamento do conjunto trator com aplicação de uma carga de 500N sobre a superfície.

(b) Deformação do conjunto trator com aplicação de uma carga de 500N sobre a superfície.

Figura 31 – Simulação do conjunto trator com aplicação de carga.

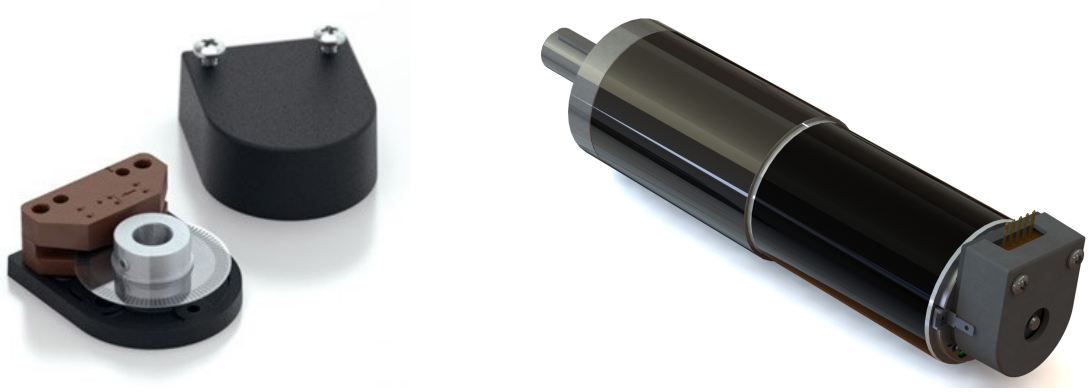
a deformação plástica do sistema com a aplicação da carga. A Figura 31b mostra a deformação elástica do sistema.

3.3 Projeto eletrônico

O projeto eletrônico do robô não apresenta grande dificuldade. Os motores foram definidos na seção anterior como sendo responsáveis por determinar as especificações de carga e movimento do robô. O projeto eletrônico é constituído pela definição do controlador que fará a atuação dos motores, dos sensores que integram o robô, o dimensionamento das baterias e do computador que será utilizado para instalar o ROS. O projeto eletrônico não foi o foco deste trabalho, sendo que os componentes foram selecionados apenas para atender os objetivos mecânicos e de software.

O controlador de potência precisa alimentar quatro motores de escovas de 150W cada, conforme definido na seção anterior. Feita uma busca no mercado, foi encontrado o controlador SDC2130 da Roboteq, que permite controlar dois motores. Este controlador fornece integração com o ROS através de um algoritmo disponibilizado pela empresa, que facilita o processo de desenvolvimento do projeto. Possui parâmetros de configuração para o motor como máxima corrente, máxima velocidade e realimentação com PID e encoder. Com esta definição, dois controladores foram necessários para o robô, que controlam um par de motores cada.

Definido os controladores, é feita uma pesquisa de encoders. Para o dimensionamento dos encoders é preciso fazer uma análise do controlador que possui uma frequência máxima de leitura. Quando utilizados com dois motores fornecem uma limitação de 30.000



(a) Encoder série E2 da US Digital. (b) Montagem do conjunto motor, caixa de redução e encoder.

Figura 32 – Montagem do encoder [Fonte: (a) US Digital[1]].

contagens/segundo por motor. Utilizando a equação fornecida no manual da Roboteq:

$$30.000 = \frac{RPM}{60} * PPR * 4 \tag{3.8}$$

onde PPR é número de pontos por revolução e RPM é a quantidade de rotações por minuto. A equação foi solucionada com as especificações do projeto e foi obtido um valor máximo de 68,7 PPR ou 274,7 CPR. Ao verificar a compatibilidade de encoders no site da Maxon, não foi encontrado nenhum que seja compatível com as especificações. Uma nova busca resultou em um fornecedor que produz encoder para substituição, portanto, possui as mesmas especificações e dimensões dos fornecidos pela Maxon, além de permitir a configuração dele. A empresa em questão, US Digital, possui o encoder E2-50-157-NE-H-D-B, que atende todas as especificações, o encoder e a montagem pode ser vista na Figura 32.

Com o objetivo de fazer mapeamento e navegação foi incorporado ao robô um LIDAR 2D. Este sensor permite a captura de uma nuvem de pontos que mede a distância do sensor a obstáculos. Esta nuvem permite o robô fazer o mapeamento de ambientes internos e assim auxilia a navegação e a localização do robô utilizando mapas conhecidos e/ou em construção. Com pouco orçamento para a compra do sensor, foi utilizado um projeto Kickstarter, que resultou em um LIDAR a um custo menor que o de mercado. O sensor utilizado foi o Sweep desenvolvido pela Scanse, permite o mapeamento em 360^0, com 1000 amostras/segundo a uma distância de 40 metros. O sensor é apresentado na Figura 33a.

O computador que foi utilizado, precisava ser capaz de comportar o ROS para fazer o controle do robô; possuir uma boa comunicação Wifi para se conectar ao computador do usuário; possuir um consumo de energia baixo para ser alimentado pela bateria; ser pequeno para embarcar no robô. Devido a disponibilidade no mercado e atender aos requisitos foi utilizado o Intel Nuc NUC6i5SYB, com processador i5, 16Gb de RAM e 256Gb de SSD. O computador é apresentado na Figura 33b.

[1] https://www.usdigital.com/products/encoders/incremental/rotary/kit/E2

(a) Sweep LIDAR. (b) Intel Nuc.

Figura 33 – Componentes eletrônicos utilizados [Fonte: (a) Scanse[1] (b) Amazon[2]].

Para o sistema de alimentação foram consideradas duas baterias, uma com maior capacidade, responsável por fornecer energia para os motores. Esta bateria é diretamente ligada a um botão de emergência que fica na carenagem do robô. Em um caso de emergência o botão é pressionado interrompendo a alimentação dos motores, embora os controladores e computador continuem operacionais. Devido a disponibilidade de mercado e atender aos requisitos foi utilizada uma bateria de 16.000mAh e 24v de LiPo (Lithium Polymer).

A outra bateria alimenta o sistema de controle. Esta bateria mantém o sistema operacional, e caso a bateria dos motores esteja sem energia ou foi pressionado o botão de

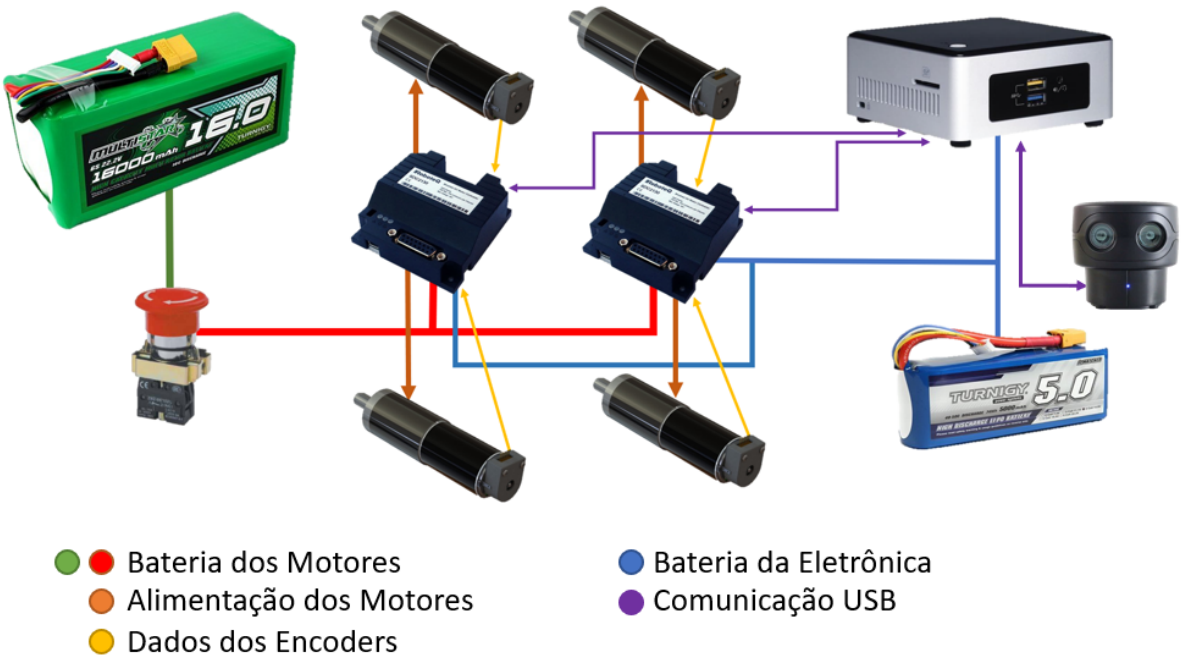


Figura 34 – Diagrama eletrônico simplificado.

[1] https://scanse.io/

[2] https://m.media-amazon.com/images/S/aplus-media/sota/b7cc2608-287b-4e36-b158-461e97b49ab7.png

emergencial o sistema de controle continua operacional, e sinaliza um erro ao ROS. Esta bateria foi escolhida devido a disponibilidade de mercado e possui 12V e 5.000mAh de LiPo. Ambas as baterias precisam de um carregador especial para serem carregadas.

Na Figura 34 é apresentado o diagrama eletrônico simplificado, demonstrando todos os componentes eletrônicos do robô. As conexões elétricas de carga e as conexões de dados.

3.4 Software

O projeto do software é uma parte importante do desenvolvimento, através dele é possível controlar o robô de forma inteligente. Até então o projeto mecânico em conjunto com a eletrônica, permitiu receber comandos de velocidade para cada motor, com garantia de seguimento de referência, atribuída pela utilização de realimentação no controlador.

Apesar de, até este ponto, do robô já conseguir realizar movimentos, é necessário que exista uma camada que transforme o comando de operação do robô em sinais de velocidade para os motores. Ainda é preciso que o robô seja capaz de cumprir algum objetivo, neste caso, ele deve ter destreza para calcular os passos que o levarão a alcançar o objetivo e transformar isto em comandos de operação, que por sua vez, são transformados em velocidades para os motores.

O diagrama da Figura 35 resume as tarefas que o software precisa cumprir. Na primeira camada (verde), deve ser capaz de receber os dados de velocidade dos motores e transmitir para o robô. Esta camada é denotada *hardware_interface*. Na segunda camada se encontra o controle do robô, nesta parte do algoritmo o sistema transforma a velocidade linear e angular do robô em comandos de velocidade angular para os motores. Esta camada é denotada *controllers* (laranja). No final, na terceira camada é colocada a aplicação, podendo ser implementado o sistema de navegação, mapeamento, dentre outros

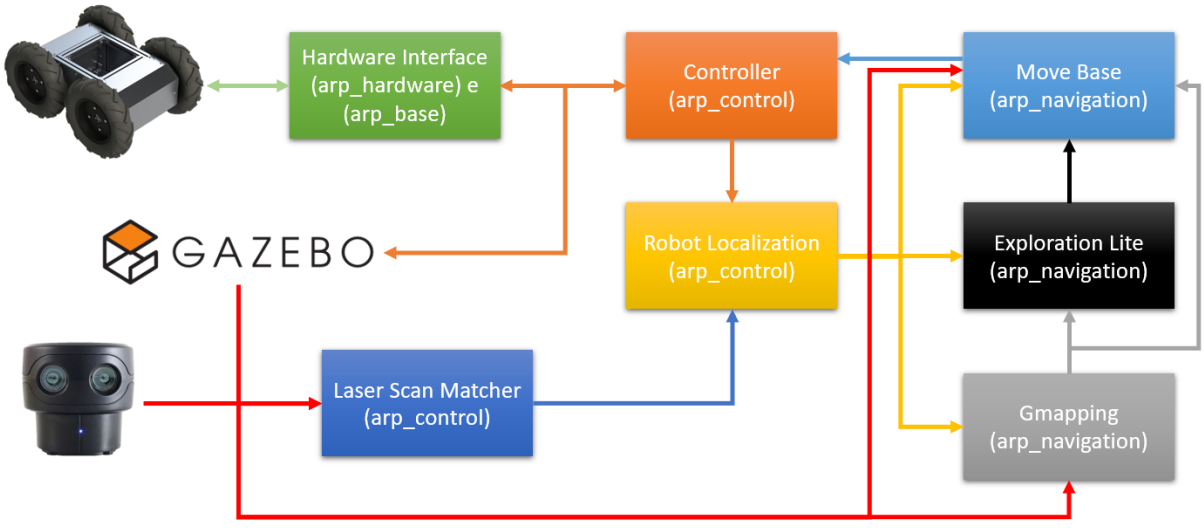


Figura 35 – Diagrama simplificado do software.

algoritmos de alto nível que estão relacionados com a necessidades da tarefa, este nível pode ser dividido em n camadas, podendo ser utilizada uma camada de navegação em conjunto com um planejador de trajetória.

Conforme proposto, a plataforma será integrada ao ROS. Cumprir algumas tarefas de forma autônoma, com suporte para o desenvolvimento de diversas aplicações. A implementação de novas tarefas dentro do escopo do projeto está sujeita às restrições mecânicas, a modificações relacionadas a terceira camada de software e a adição de sensores e/ou manipuladores ao robô.

O primeiro passo para o desenvolvimento do software é fazer uma análise das camadas, e estruturar o projeto em partes. É visível que o projeto de software se subdivide em três principais partes, o *hardware_interface*, o *controllers* e a camada de aplicação. Além disso, é desenvolvido uma simulação capaz de testar em ambiente computacional os algoritmos desenvolvidos na segunda e terceira camada do software. Com o objetivo de testar a primeira e segunda camada e a integração com o ROS, foi testado o robô para o mapeamento autônomo utilizando os algoritmos *exploration_lite* (preto), *gmapping* (cinza), *move_base* (azul claro), *laser_scan_matcher* (azul) e o *robot_localization* (amarelo). Abaixo é apresentado o escopo do projeto de software:

- Desenvolvimento da interface de hardware, permitindo o controle do robô real com a segunda e terceira camada do ROS (*hardware_interface*);
- Desenvolvimento do URDF, utilizado na simulação e nos controladores da segunda e terceira camada de software;
- Implementação e configuração da simulação, para facilitar o desenvolvimento das camadas dois e três;
- Configuração da segunda camada (*controllers*);
- Configuração dos algoritmos da terceira camada (mapeamento interno).

Na sequência é apresentado o desenvolvimento do software que controla o robô e a parametrização dos algoritmos de terceiros utilizados do repositório do ROS. Para o desenvolvimento do algoritmo foi determinado o nome do robô como sendo ARP (Autonomous Robot Platform), portanto durante o desenvolvimento do software são criados tópicos e pastas com o nome arp. Todo o algoritmo desenvolvido e atualizado pode ser encontrado no repositório da aplicação[1].

[1] https://github.com/marcelopetry/arp2

3.4.1 Hardware Interface

A primeira tarefa a ser desenvolvida é permitir que os algoritmos do ROS consigam controlar o robô, conforme visto no Capitulo 2, a tarefa é desenvolvida pelo *hardware_interface*, uma interface que implementa as funções necessária para o desenvolvimento de um driver capaz te conectar o robô ao ROS, e permitir que conversem entre si. As principais atribuições do algoritmo desenvolvido são:

- Enviar e receber os dados dos controladores seguindo o protocolo de comunicação da Roboteq;
- Inicializar a interface para possibilitar os controladores da segunda camada atuarem sobre o robô;
- Disponibilizar os dados como uma mensagem de status para que as outras camadas tenham acesso;
- Possibilitar o envio recepção de dados das entradas e saídas digitais/analógicas do controlador;
- Converter os dados recebidos pela segunda camada em dados compatíveis com o controlador;
- Gerenciar situações críticas, enviar comando de parada de emergência para os controladores, como recurso de software para parar o robô.

Para solucionar os tópicos listados, foi implementado um driver para o ROS, desenvolvido pelo fabricante do controlador e modificado para as necessidades do projeto. Na sequência são apresentadas etapas e o desenvolvimento.

3.4.1.1 Roboteq *Interface*

A Roboteq disponibiliza uma interface com o ROS, que facilitou o desenvolvimento das tarefas. A solução disponibilizada atua recebendo mensagens com os comandos por tópicos e publicando uma mensagem de status. A ideia do *hardware interface* é operar como um serviço e não como um tópico, desta forma, o código fornecido pela Roboteq foi alterado para manter compatibilidade com o ROS.

O algoritmo fornecido por eles é dividido em duas partes: uma responsável pela conexão com o controlador, onde é feita a comunicação, a recepção dos dados, a gravação do script, a configuração do controlador e o tratamento da mensagem recebida; a segunda parte trata dos canais, cada controlador é responsável por dois motores, para o algoritmo cada motor é tratado como um canal. Cada canal tem a possibilidade de enviar um setpoint, uma forma de operação e retornar o feedback relacionado ao canal. Cada

canal possui um watchdog, ou seja, caso o canal não comunique por um intervalo de tempo, o controlador envia um sinal de parada de emergência imediatamente, este tempo é configurado como parâmetro do ROS e também do controlador.

Foi citado a gravação do script como uma das atribuições do algoritmo. Para o controlador se comunicar com o ROS, é preciso gravar um script interno, responsável por definir as variáveis de controle do controlador, recepcionar os dados enviados pela serial e enviar os dados em intervalos regulares de tempo. O script é gerado pelo software da Roboteq, o RoboRun, que facilita a implementação e parametrização dele. O script usa como referência um modelo fornecido pelo fabricante.

A plataforma é projetada para operar com dois controladores, cada qual com dois motores. Para garantir o cumprimento do tempo e não haver problema com atraso de mensagens ou operação com atraso, cada controlador é inicializado como um thread, responsável por todas as atribuições listadas acima, para citar as mais importantes, receber o status e enviar o setpoint.

3.4.1.2 Driver ROS

O gerenciamento dos recursos do robô é feito pela classe *arp_hardware*, é uma classe que faz a integração dos componentes do robô e o tratamento dos dados, é o link entre o hardware e o ROS. Já a outra classe o *arp_base* é responsável por fazer a chamada do *arp_hardware* e inicializar os threads necessária para o seu funcionamento. Em conjunto as duas formam o *Hardware Interface*, o driver do robô.

A classe *arp_hardware* inicializa fazendo a configuração dos parâmetros de comunicação com os controladores e inicializa a conexão. Caso houver problemas o algoritmo apresenta um erro e para a inicialização. Na continuidade, lê os parâmetros do controlador. Após concluída as operações ele registra o *controller_interface*, gerando o serviço que vai permitir que o ROS controle o robô, caso não haja problemas, ele inicializa o tópico responsável por publicar o status do robô.

Dentro da classe *arp_hardware* também são implementadas funções que permitem atualizar os dados de setpoint com base nos comandos do ROS, atualizar os dados dos motores no ROS com base no feedback dos controladores e publicar o tópico de status. Porém estas funções não são inicializadas nesta classe, mas sim na classe *arp_base*, que fica responsável por chamar as funções e iniciá-las.

A classe *arp_base* é responsável por carregar os parâmetros de configuração do pacote fornecidos pelo *launch* do ROS e inicializar a classe *arp_hardware* e segue uma rotina de inicialização:

- Inicializa um thread para cada controlador, com a função de ler e escrever os dados sempre que necessário, está implementação segue o modelo fornecido pela Roboteq.

- Inicializa um thread de controle que em intervalos de tempo definidos nos parâmetros, chama a função responsável por publicar os dados dos motores no ROS e enviar os setpoints para os controladores.

- Por fim, é inicializada um thread de diagnóstico, está opera em intervalo de tempo diferente da thread de controle, sendo este um parâmetro. O loop de diagnóstico publica o estado do robô (temperaturas, tensão das baterias, corrente, estado dos controladores) e faz uma varredura da integridade do equipamento, em caso de problema pode enviar comando de parada de emergência.

3.4.1.3 Inicialização do nodo

Compilando o pacote com suas respectivas dependências é criado o nodo *arp_node*, este é o algoritmo responsável pela primeira camada do software, a integração do hardware com o ROS, o *Hardware Interface*. Com este nodo é possível conectar os controladores e passar o setpoint para cada motor e receber o feedback dos canais e do robô, abstraindo o hardware. Para enviar os comandos para o robô basta criar um simples *publisher* para o serviço criado ou implementar a segunda camada.

Dentro do ROS é possível inicializar o nó com o comando *rosrun* ou criar um *lauch* que permite a configuração dos parâmetros de forma mais simples, como se segue:

```
<?xml version="1.0"?>
<launch>
  <node pkg="arp_base" type="arp_node" name="arp_node">
    <rosparam subst_value="true">
      control_frequency: 10.0
      diagnostic_frequency: 1.0
      ctrl_name: ['front','rear']
      channel_name: ['left','right']
      polling_timeout: 0.15
    </rosparam>
  </node>
</launch>
```

Onde os parâmetros são:

- *control_frequency*: frequência do loop de controle em Hz;

- *diagnostic_frequency*: frequência do loop de diagnóstico em Hz;

- *ctrl_name*: vetor com o nome dos controladores, podendo ser um ou mais. O algoritmo está configurado para procurar a porta e os parâmetros dos controladores da Roboteq;

- *channel_name*: nome dos canais de cada controlador, podendo ser um ou mais;

- *polling_timeout*: watchdog, tempo sem resposta para que o controlador de parada de emergência.

3.4.2 Simulador e URDF

Para o controle do robô utilizando o *controller* é preciso descrevê-lo utilizando o URDF, que vai fornecer as informações físicas e mecânicas do robô para o modelo, e a posição dos sensores. O URDF conforme foi tratado no Capítulo 2, é um arquivo XML e contém componentes que auxiliam na descrição do robô em formato de código.

O simulador descreve os mesmos parâmetros do robô e fornece a capacidade de testar os algoritmos em um ambiente virtual e seguro antes de sua aplicação no robô físico. Convenientemente o simulador é disponibilizado junto com a distribuição do ROS, o Gazebo, e utiliza o mesmo arquivo URDF criado para os controladores. Basta adicionar alguns componentes que melhoram a visualização e descrevem as características físicas do robô e do universo para o simulador.

3.4.3 Configuração

A partir deste ponto, toda a configuração necessária já foi realizada, ou seja, o robô está descrito utilizando o arquivo URDF, que permite a simulação e coleta de características para os controladores; e permite controlar o robô físico através de um driver, que envia os comando para o robô da mesma forma que envia para o simulador, fornecendo a abstração do hardware e a possibilidade de integração com as demais ferramentas disponibilizadas pelo ROS. Dentro deste trabalho, os demais algoritmos foram apenas parametrizados e não desenvolvidos, porém são integrados de tal forma que permite o robô fazer o mapeamento interno de um ambiente fechado utilizando SLAM.

Para o controle do robô transformar os dados de velocidade linear e angular em dados para os motores é utilizado o *diff_drive_controller*, que permite a utilização de dois ou mais motores e controlar plataformas do tipo *skid steer*. Este algoritmo é um segue um modelo diferencial, que permite o robô fazer curvas aplicando diferentes velocidades nos motores. Na Figura 35 é apontado em laranja, o *controller* se conecta com o robô real e simulado, devido a abstração de hardware, ele recebe os feedbacks dos motores e envia sinais de controle. Também é responsável por publicar a odometria utilizada pelo *robot localization*.

O *laser scan matcher* é um algoritmo que consegue gerar uma localização do robô com base nos dados do laser. Devido ao erro gerado pelo modelo skid steer, que utiliza do arrasto das rodas para fazer curvas, é preciso utilizar mais que um sensor para fazer a localização do robô além da odometria. Este algoritmo funciona muito bem em ambientes populoso, onde o laser encontra diversos objetos para tomar como referência. Os dados do *laser scan matcher* são fundidos pelo *robot localization*, que permite escolher quais serão

os sensores predominantes em cada medida. Para este projeto é utilizado a odometria como medida predominante para deslocamentos e velocidade linear em x e o laser para deslocamento e velocidade linear em y e angular em z.

O algoritmo de navegação (*move_base*) é parametrizado utilizando os limites de operação do robô, sendo eles: velocidade máxima e mínima linear e angular em ambientes abertos e para situações críticas; aceleração máxima linear e angular permitida; as dimensões de operação do robô, levando em conta uma margem de segurança, estas são responsáveis por delimitar o máximo que o robô pode se aproximar dos obstáculos; e frequência de atualização do algoritmo de navegação. Outros parâmetros são utilizados, porém se referem a comunicação interna dos nodos. Na Figura 35 é apontado com azul claro, e envia dados de diretamente para o *controller*.

Este algoritmo fornece todas as funções de navegação, sendo elas: o desvio de obstáculos globais, quando são localizados no mapa já construído; o desvio de obstáculos locais, obstáculos que são identificados pelo laser, mas ainda não estão mapeados. Um exemplo clássico é uma pessoa passando na frente do robô; por fim, determinar a rota do robô para alcançar o objetivo. Esta leva em consideração o menor caminho, e utiliza as informações que já foram mapeadas para fazer isto.

O algoritmo de mapeamento (*gmapping*) possui apenas dois parâmetros principais, um tamanho inicial do mapa, que pode ser ampliado a medida que o robô explora o ambiente; e as informações do LIDAR, que são utilizadas para determinar a distância que o robô se encontra dos objetos e determinar a localização deles em relação ao robô. Este algoritmo utiliza dos dados de odometria em fusão com os dados do laser para criar o mapa tendo como referência a posição do robô em relação a sua posição inicial. Na Figura 35 é apontado em cinza.

Por fim, no nível mais alto da aplicação se encontra os algoritmos de exploração (*exploration_lite*), que aproveita de todos os serviços fornecidos pelos dois algoritmos anteriores para o desenvolvimento da exploração. Este algoritmo precisa de dois parâmetros principais: tempo de busca, este parâmetro permite determinar um tempo de persistência em uma rota, caso o algoritmo não note progresso significativo na exploração, ele traça um novo caminho; e o tamanho da fronteira, é preciso determinar para o algoritmo a distância mínima entre dois objetos não mapeados para que seja considerada uma fronteira inexplorada, quanto menor este valor mais detalhado será o mapa, mas pode influenciar na velocidade de resolução do problema. Na Figura 35 é apontado em preto, e se comunica diretamente com o *move base* passando o próximo ponto que o robô deve explorar.

4 Testes e Resultados

Neste capítulo será apresentado o robô desenvolvido nesta monografia, as suas especificações e características. Também será abordado testes desenvolvidos para validar as especificações de projeto e os parâmetros físicos. Por fim é detalhada a integração entre o robô e o ROS, o que permite o desenvolvimento de tarefas de forma autônoma.

4.1 Projeto final

A proposta final é apresentada na Figura 36, e busca um projeto modular, robusto, de fácil acesso para os equipamentos e instrumentos. Visa isolação do sistema do robô do compartimento de carga. Propõe uma estrutura de fácil implantação para novos itens de hardware. A estrutura foi montada toda com perfil de alumínio 40x40, que cria o distanciamento ideal para alojamento das polias e correia no interior do sistema motriz (Figura 36d). Criando um módulo de tração composto por dois eixos e dois motores (Figura 36f). Este módulo é igual para o lado esquerdo e direito, criando uma substituição perfeita.

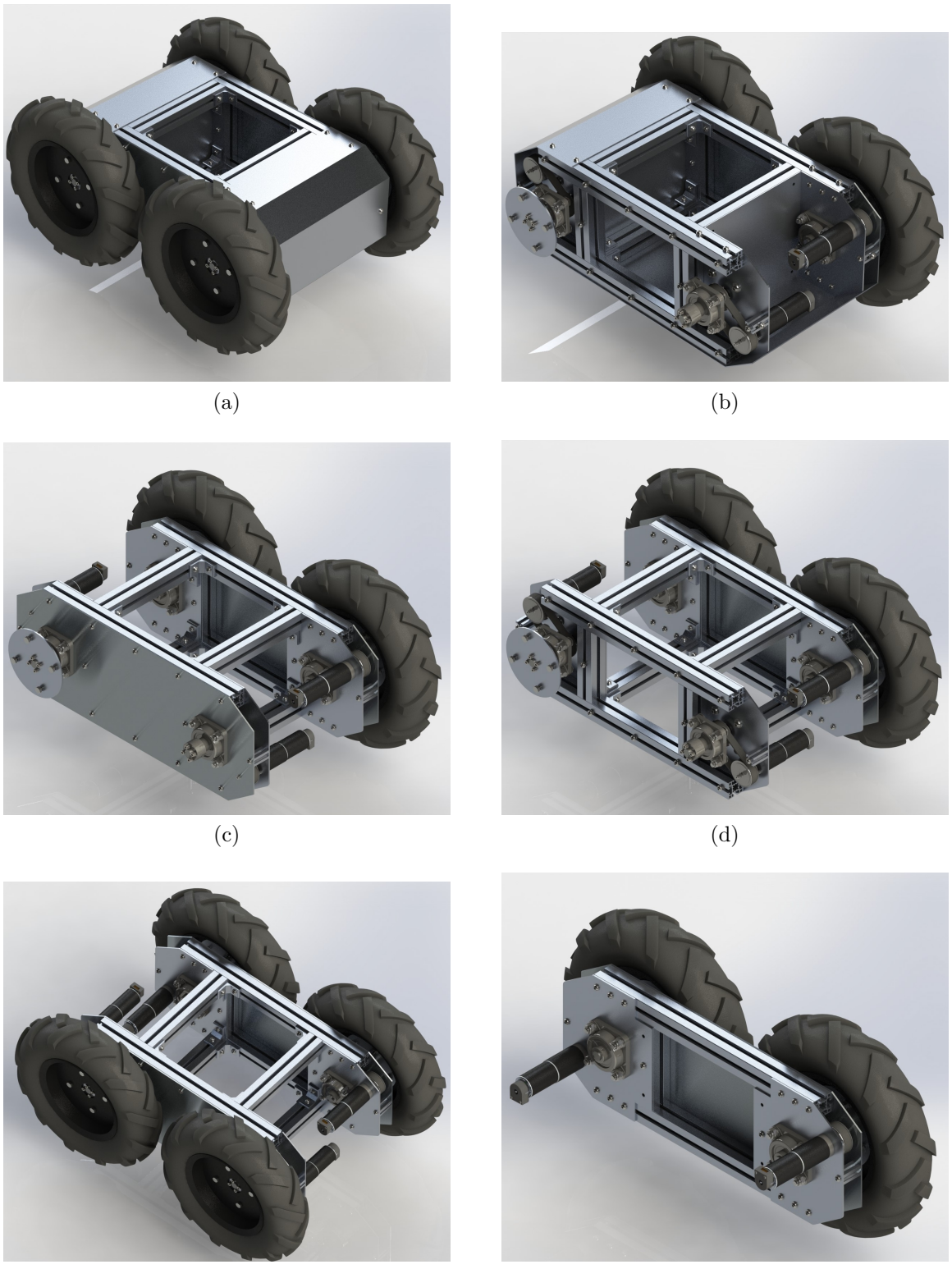
Os dois módulos são conectados por quatro perfis de 40x40 e fixados por cantoneiras de alumino. O robô pode ser remodelado em sua distância entre eixo apenas substituindo estes perfis, o tornando mais versátil e aberto a atualizações. O projeto permite a utilização do robô sem a necessidade de carenagem, podendo operar aberto. Este é mais um ponto de sua modularidade, que permite a adaptação da carenagem e a sua não utilização conforme a necessidade da aplicação.

Devido aos componentes não atenderem exatamente os parâmetros de projetos as especificações do robô sofrem pequenas alterações e estão sintetizadas na Tabela 5.

Na Figura 37 é apresentado o resultado do projeto. Foi incorporado um pórtico no robô para permitir a instalação do LIDAR e uma câmera, que possibilitou o desenvolvimento dos testes de integração com o ROS e de navegação autônoma.

Tabela 5 – Especificações finais do projeto.

Massa total do robô [kg]	43,2
Num. motores [#]	4
Diâmetro da roda [m]	0,4
Velocidade do robô [m/s]	1,3
Inclinação da superfície [º]	29
Tensão motor [V]	24
Aceleração [m/s^2]	0,5
Eficiência [%]	65

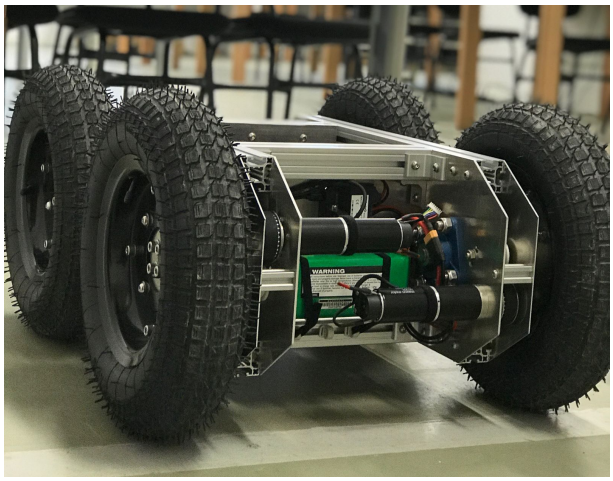


(a) (b)

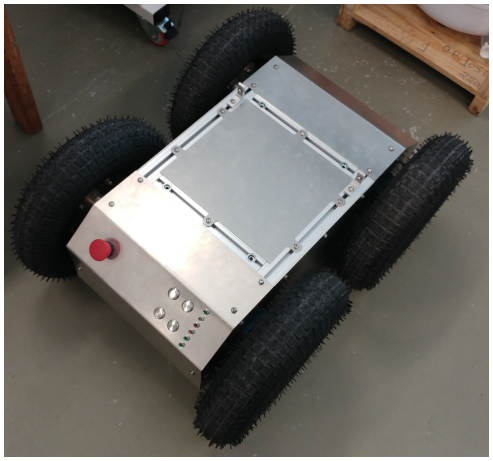
(c) (d)

(e) (f)

Figura 36 – Resultado do projeto mecânico.



(a) Robô montado sem a carenagem.



(b) [Robô montado com a carenagem.

Figura 37 – Implementação física do projeto.

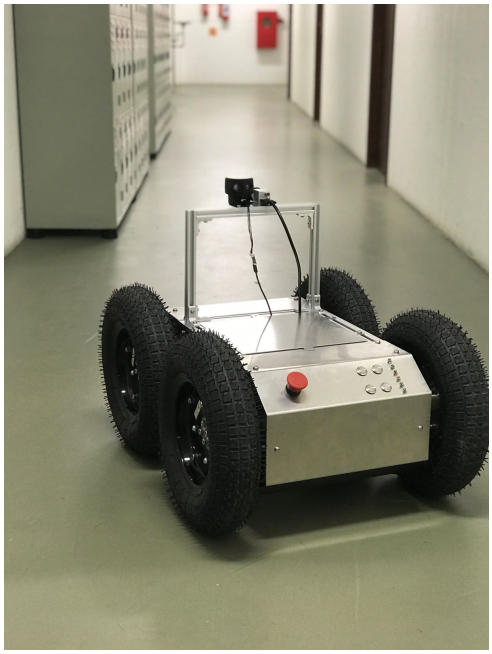
4.2 Validação de parâmetros

Para o desenvolvimento deste projeto foram traçados parâmetros que delimitaram o escopo. Estes parâmetros foram obtidos de forma analítica e através de estudos em bibliografias. Para corroborar os dados com a prática o robô foi submetido a testes físicos. Os testes foram elaborados de forma a avaliar o robô dentro das condições propostas e verificar quais são os pontos máximos de operação.

Inicialmente foram levantados os terrenos nos quais o robô deve operar, conforme o projeto é esperado que a plataforma opere em terrenos irregulares e regulares. Ambos os terrenos possuem dificuldades, em um terreno irregular o robô possui menor atrito facilitando algumas manobras, porém a falta de uniformidade do terreno não garante contato com as quatro rodas. Inversamente, terrenos regulares possuem uma maior uniformidade da superfície, aumentando a probabilidade de contato pleno das quatro rodas, porém estes ambientes possuem um maior atrito, causando maior dificuldade de o robô efetuar manobras. O robô foi testado em duas situações:

- Ambiente interno, caracterizado por um piso de concreto usinado liso e uniforme. O local de teste foi o Bloco B da UFSC (Universidade Federal de Santa Catarina), Figura 38a.
- Ambiente externo, caracterizado por terreno de pedra, levemente acidentado. O local de teste foi o estacionamento do bloco B da UFSC Blumenau, Figura 38b.

No escopo do projeto são delimitados parâmetros de velocidade e aceleração, para avaliar estes parâmetros a plataforma foi submetida a testes em linha reta em ambos os terrenos utilizando 7kg de carga totalizando 50,2Kg. O robô foi inicializado com as baterias em carga máxima, posicionado em uma reta e acelerado com um degrau de



(a) Robô em terreno caracterizado como interno.

(b) Robô em terreno caracterizado como externo.

Figura 38 – Caracterização dos terrenos onde o robô foi testado.

velocidade positivo e um degrau negativo. Os dados foram coletados utilizando os encoders presentes nos eixos de rotação de cada uma das rodas. O resultado é apresentado na Figura 39 e 40, é possível notar pelo gráfico que o robô atinge a velocidade de 1,3 m/s em ambos os terrenos e que a velocidade é proporcional a potência dos motores. Outra característica é que a potência é igual para os quatro motores.

O teste mencionado é capaz de avaliar o robô em um plano reto, mas uma das especificações presentes no escopo é a inclinação. Para avaliar a capacidade da plataforma em um terreno inclinado foi dimensionada uma rampa com a possibilidade de modificar seu angulo de inclinação (Figura 41), porém está rampa foi concebida utilizando madeira, modificando o coeficiente de atrito dos testes anteriores. O teste em um plano inclinado foi executado com uma carga de 7kg totalizando uma massa de 50,2kg.

O procedimento para o teste foi averiguar a capacidade do robô sair da inercia, a plataforma foi posicionada paralelamente sobre a rampa com o objetivo de medir os parâmetros de velocidade e aceleração durante o procedimento. O robô foi acionado utilizando degraus de velocidade e as baterias foram carregadas ao máximo. Os dados foram coletados utilizando os encoders presentes nos eixos de cada motor. Inicialmente o teste foi executado utilizando a inclinação presente no escopo, porém devido ao resultado o teste foi replicado aumentando a inclinação da plataforma para obter o máximo que o robô consegue escalar. Os gráficos da Figura 42 e 43 apresentam os resultados. Perceba que quando o robô para no instante 28s e no instante 32s os controladores tentam mantem o robô parado na rampa, ajustando a potência para compensar a força da gravidade,

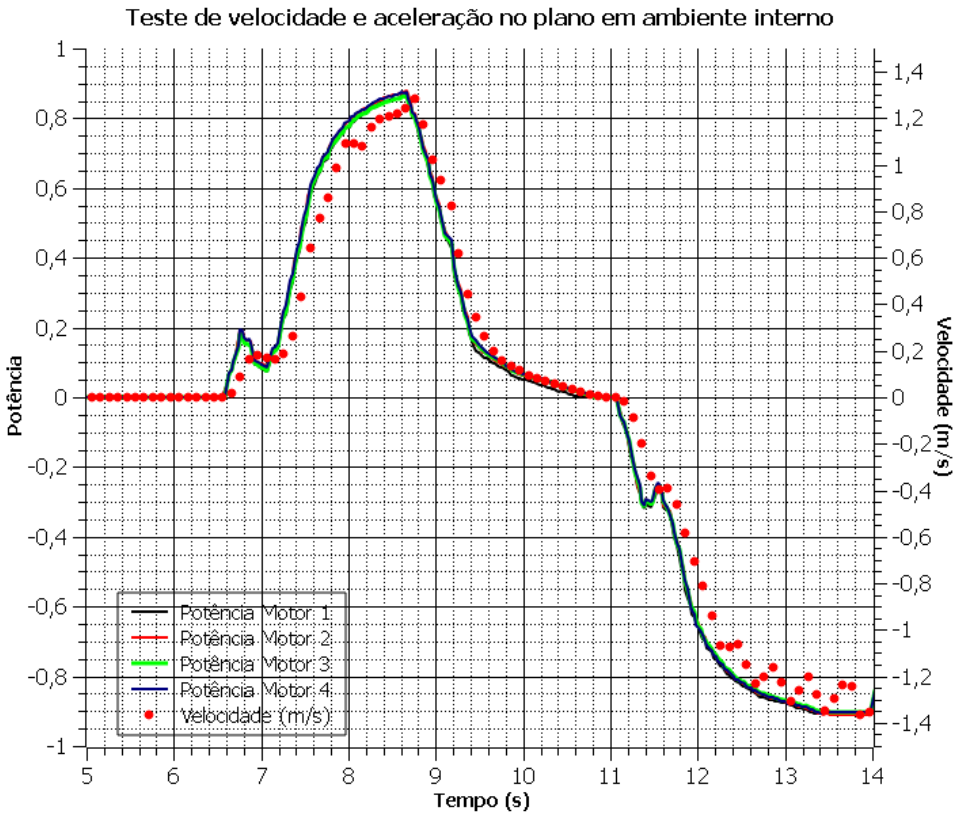


Figura 39 – Gráfico da velocidade do robô em terreno interno.

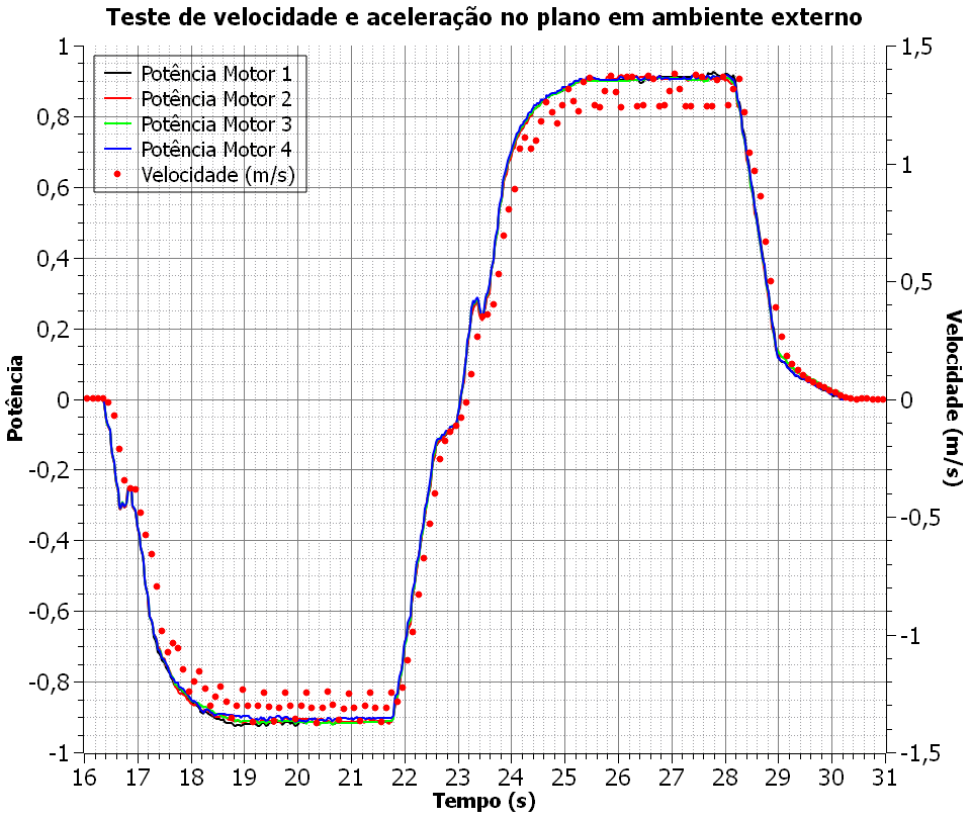


Figura 40 – Gráfico da velocidade do robô em terreno externo.

(a) (b) (c)

Figura 41 – Rampa utilizada para os testes de inclinação.

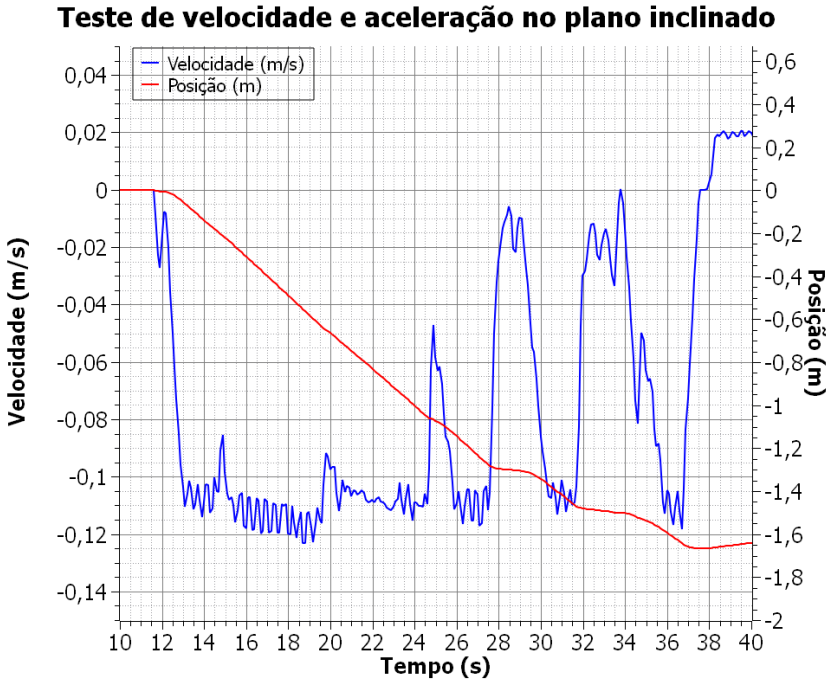


Figura 42 – Gráfico da posição e velocidade dos motores em uma rampa de 29^0.

conforme apresentado na curva da velocidade.

Outro parâmetro para avaliação é a qualidade dinâmica do robô, este teste foi desenvolvido de forma qualitativa e possui o objetivo de avaliar a facilidade de locomover a plataforma. Devido à natureza como o robô é concebido, ele possui muita força de tração, podendo mover grandes cargas. Uma das suas limitações é efetuar curvas, devido ao arrasto das rodas. Este teste foi executado utilizando o controle analógico de um X-box, onde o operador deve controlar o robô para desviar obstáculos e avaliar a facilidade de efetuar as manobras. O teste foi executado com as baterias carregadas ao máximo e com uma carga de 7kg totalizando uma massa de 50,2Kg. O teste foi efetuado por um voluntário que nunca havia operado uma plataforma robótica móvel anteriormente. Os resultados foram:

- Inicialmente o robô se demostra difícil de controlar, porém sua curva de aprendizado

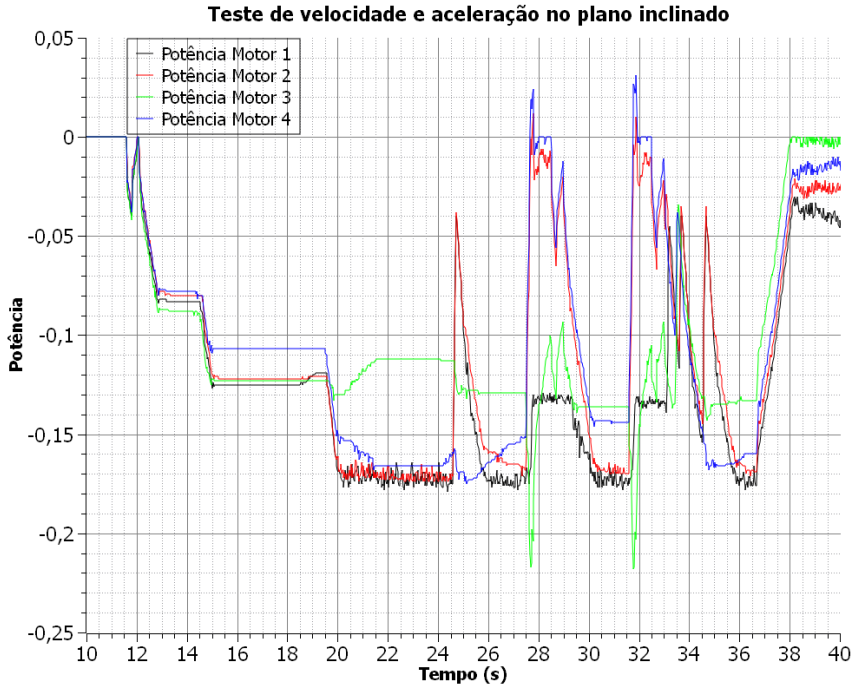


Figura 43 – Gráfico da potência dos motores em uma rampa de 29^0.

Tabela 6 – Máximos de operação.

Máxima inclinação sem carga [º]	36
Máxima inclinação com 7kg de carga [º]	32,4
Carga máxima sem efetuar curvas [kg]	210
Carga máxima efetuando curvas [kg]	15
Carga máxima no plano inclinado de 29º [kg]	20

é rápida, em poucos minutos é possível dominar o seu funcionamento;

- O robô possui um movimento suave e preciso, possibilitando cumprir manobras e posicioná-lo facilmente.

- Controlar o robô de forma assistida, olhando para ele é fácil. O mesmo não se aplica ao utilizar uma câmera frontal, isto dificulta a visualização dos elementos em volta do robô.

De forma geral, os testes desenvolvidos apresentaram resultados para validar os parâmetros apresentados no escopo. Para uma melhor avaliação, os estudos foram replicados para o robô sem carga, totalizando uma massa de 43,2kg. Também foram replicados elevando a carga de forma gradativa até que não tivesse força para operar ou inviabilizase o teste, isto permitiu avaliar a máxima capacidade de operação. Os resultados são apresentados na Tabela 6.

O último parâmetro a ser avaliado foi a autonomia do robô. Com carga plena em ambas as baterias o robô foi operado de forma contínua com uma carga de 7Kg totalizando

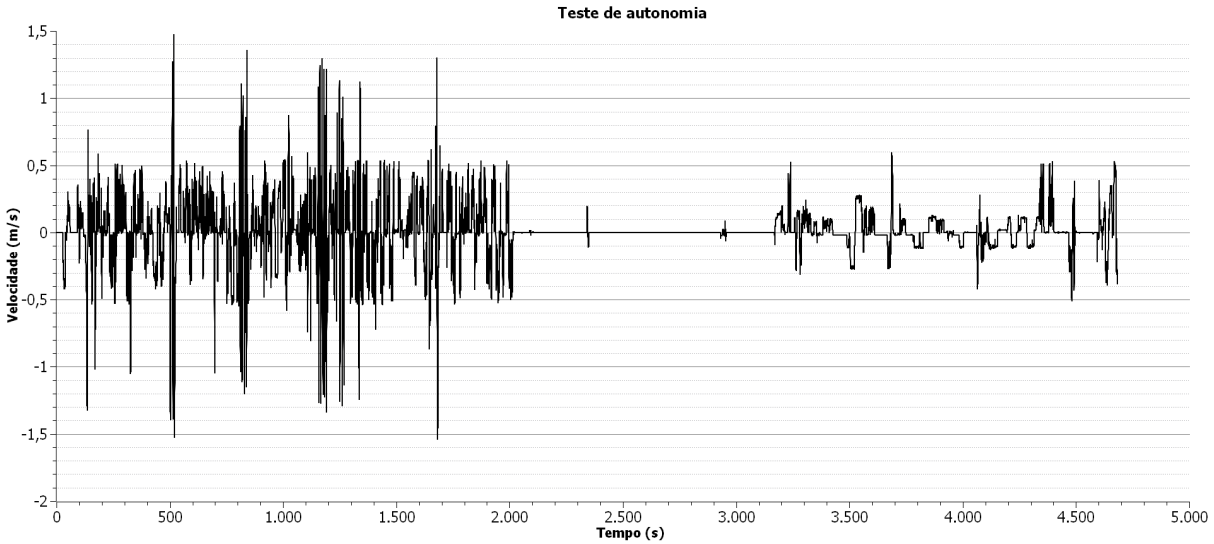


Figura 44 – Gráfico da velocidade do robô durante o teste de autonomia.

uma massa de 50,2kg. O teste foi executado por um operador, que controlou o robô durante todo o procedimento utilizando um joystick, dentre as tarefas o robô deveria se deslocar dentro de um ambiente interno, se posicionar em locais de difícil acesso e subir elevações dentro de suas capacidades. O resultado do teste é coletado quando o sistema falha, por insuficiência energética. No gráfico apresentado na Figura 44 é mostrada a atividade do robô durante o teste, o objetivo é simular utilização normal, onde o robô opera intensivamente e para pôr um intervalo, e retoma as atividades. O robô operou durante 1h e 18min, porém a bateria de potência estava com uma tenção de 23,3V, a bateria que interrompeu o teste foi a bateria da lógica, que ao final do teste estava com 11,83V. O que causou o desligamento do Nuc, interrompendo o funcionamento do ROS.

Todos os parâmetros foram coletados em ambiente controlado, os testes foram executados seguindo procedimentos pré-determinados. Todos os testes foram gravados em imagem e todas as mensagens trocadas pelo sistema podem ser replicados em ambiente real ou simulado utilizando ferramentas do ROS.

4.3 Algoritmo de exploração

O software foi inicialmente testado no simulador. Utilizando um mapa disponibilizado junto com a distribuição do ROS, o Willow Garage, Figura 45a. Este mapa recria um ambiente de escritório composto por salas e corredores. Para o robô um dos piores cenários é um corredor, pois nesta condição o deslocamento linear fica todo por conta da leitura dos encoders.

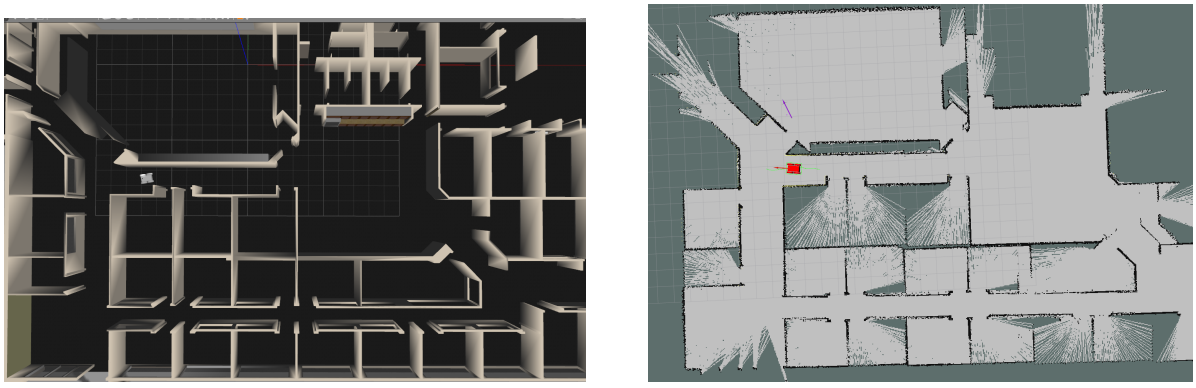
Como resultados da simulação é obtida a criação do mapa do ambiente proposto, Figura 45b. Os resultados do algoritmo de navegação e mapeamento foram mais que satisfatórios. A fusão dos dados dos encoders junto com a odometria das rodas resultaram

em uma medida bastante acurada de posição. Isto pode ser visto claramente ao observar o paralelismo das paredes, e mais, o robô precisou dar a volta no corredor para fazer o mapeamento, e como pode ser visto, não teve variação de posicionamento da parede no encontro do mapa.

Porém o algoritmo de exploração apresentou algumas particularidades, durante o processo de execução o algoritmo deixa transparecer que não utiliza uma implementação otimizada. Muitas vezes o robô está seguindo uma fronteira de forma correta, e em um determinado momento ele simplesmente começa a explorar o lado oposto do mapa, gerando um custo de tempo muito alto para a operação. Porém ele é capaz de identificar todas as fronteiras e explorar o mapa por completo.

Durantes os testes alguns dos parâmetros dos algoritmos foram alterados, visando obter o melhor resultado. Através de um método iterativo foi encontrado um ponto ótimo, onde o robô possui uma boa destreza e dinâmica durante a operação no simulador. O ROS permite que o mesmo código que roda no simulador seja replicado para o modelo físico, sem a necessidade de efetuar alterações. Após uma série de testes no simulador, foi implementado o código no modelo real. Após a primeira operação com o modelo real, foram realinhados alguns parâmetros para um ajuste fino. Os testes do modelo físico foram feitos nos corredores do Bloco B da UFSC Blumenau e no LABIND, com e sem circulação de pessoas.

Os resultados da implementação físicas foram muito parecidos, o robô se comportou bem com os algoritmos de mapeamento e de navegação. Porém uma particularidade do algoritmo de navegação que difere do modelo simulado, é que em muitos casos ele tem dificuldade de se deslocar quando muito próximo de obstáculos, por exemplo, passando por portas ou desviando um obstáculo perto da parede. O algoritmo de mapeamento se comportou de forma similar com o modelo simulado, apresentou se bastante acurado e de rápido processamento, porém devido a ruídos oriundos do sensor, em alguns casos aparecem linhas de não obstrução a uma distância considerável do alvo, como pode ser



(a) Simulação do mapa Willow Garage no Gazebo. (b) Resultados do modelo simulado

Figura 45 – Resultado do mapeamento do ambiente no simulador.

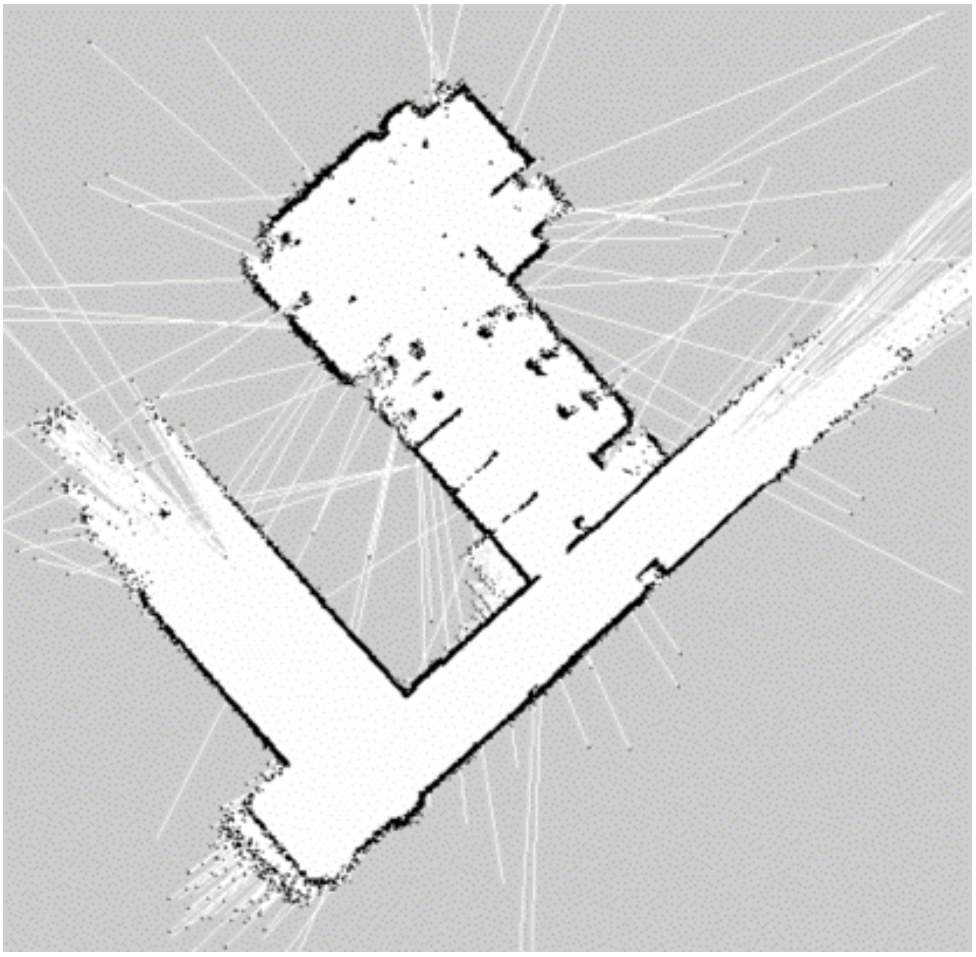


Figura 46 – Resultado do mapeamento autônomo de parte do Bloco B da UFSC Blumenau.

visto na Figura 46.

Já o algoritmo de exploração teve ainda mais dificuldades no modelo real. No modelo simulado já apresentava resultados não otimizados, mandando o robô para locais distantes sem acabar a exploração, no modelo físico isto se mostrou mais gritante. Muitas vezes o robô iniciava uma trajetória e três metros depois retornava para um ponto distante. Em alguns casos ele ficava simplesmente indo e voltando. Porém ao final é capaz de cumpri o objetivo de mapear o ambiente. Dependendo a complexibilidade do local a ser mapeado ele apresenta um melhor resultado. Durante os testes foi percebido que o pior desempenho foi ao passar por corredores.

5 Conclusões

5.1 Considerações finais

Neste trabalho foi feito um estudo sobre o desenvolvimento de um robô móvel integrado com o ROS para aplicações, estudos e pesquisas que envolvam a necessidade de mobilidade. A proposta apresenta os processos necessários para o desenvolvimento, as escolhas de projeto e os componentes utilizados, podendo ser aproveitado como referência para o desenvolvimento de novas plataformas robóticas ou ser recriado para o desenvolvimento de outros projetos que contemplem mobilidade. O projeto foi desenvolvido de forma a facilitar a integração, expansão e modificação, permitindo a adaptação do projeto conforme as necessidades da aplicação.

Os componentes mecânicos utilizados, em sua maioria são comercias e facilmente encontrados no mercado brasileiro. Alguns dos componentes precisaram ser desenvolvidos, estes foram submetidos a testes de força utilizando softwares CAD para garantir que suportem as especificações do projeto. O processo foi executado de forma iterativa até atingir as especificações esperadas. O projeto foi pensado para que a manufatura destes componentes seja simples, não agregando grandes custos ao projeto e facilitando a sua implementação como um produto, tornando simplificado o processo de replicar a plataforma em um processo produtivo. Toda a estrutura foi desenvolvida pensando na montagem, manutenção e substituição, por isto os componentes desenvolvidos possuem simetria e são projetados em módulos, garantido a fácil substituição das partes ou módulos que são iguais e se repetem no robô.

Os componentes elétricos utilizados também são comercias, porém existe uma dificuldade maior de se obter no Brasil, para este projeto os componentes foram importados. Os motores foram dimensionados para as características especificadas no escopo do projeto, porém os encoders foram dimensionados para serem compatíveis com o controlador. O robô foi desenvolvido para comportar o sistema operativo utilizando um computador interno, porém a parametrização, supervisão e atribuição de missões é feita por um computador externo que se conecta a plataforma via Wifi, entretanto o robô consegue operar desconectado caso já esteja parametrizado, devido ao ROS *master* estar operando dentro do computador interno. O projeto mecânico e elétrico está dimensionado prevendo a necessidade de aumento da capacidade das baterias, que inicialmente vislumbra a possibilidade de duplicar as baterias de potência e instalar uma bateria de controle maior, permitindo que o robô opere por mais tempo.

O conjunto elétrico e mecânico foi testado para garantir as especificações projetadas. Conforme os resultados é possível afirmar que o robô desenvolvido corrobora com os parâ-

metros de projeto calculados. A velocidade máxima esperada era de $1,3m/s$ exatamente a mesma obtida nos resultados, a inclinação máxima esperada é de 29º e o robô consegue encalar $32,7$º, a carga máxima projetada é de $7kg$ e o robô carrega uma carga de $15kg$ nas condições projetadas. Porém o tempo de operação do robô sob atividade normal é de 1h e 18m, era esperado que o robô conseguisse operar por 3h, conforme apresentado esta limitação é atribuída a bateria de controle, que foi mal dimensiona. A bateria utilizada é de 11,11V podendo chegar com plena carga a uma tensão de 12,6V, mas o computador utilizado precisa de uma tenção de alimentação de 12V a 19V, ou seja, a bateria não chegou a ser completamente descarregada, pois o robô parou quando ela atingiu uma tenção de 11,83V.

O robô foi integrado ao ROS através do *arp_base*, o driver fornece ao robô a capacidade de receber os comandos do software e transformar os dados coletados pelos sensores em dados para o programa. Este algoritmo foi desenvolvido utilizando os códigos fornecidos pelo fabricante do controlador dos motores e tem a principal função de enviar um setpoint de velocidade para cada motor. Em paralelo foi desenvolvido um simulador que permite o recriar as condições reais do robô para a programação e parametrização das aplicações utilizando o ROS. O simulador e o *arp_base* recebem os mesmos comandos isso foi validado ao desenvolver o software utilizando o simulador e após o desenvolvimento utilizado sem modificações no robô.

Para validar a integração do robô com o ROS, foi utilizado alguns algoritmos presentes no software para efetuar o mapeamento autônomo de ambientes fechados. Como resultado o robô foi capaz de mapear uma parte do bloco B da UFSC e apresentou um mapa que caracteriza de forma real a planta do prédio. Para este resultado foi feita a fusão dos dados do LIDAR com a odometria das rodas, o que permitiu a criação de um mapa que apresenta as paredes paralelas conforme esperado e com pouco desencontro. O mesmo teste também foi feito no simulador que apresentou o resultado de um ambiente mais amplo e com maior qualidade, fator que pode ser atribuído ao erro dos sensores no robô físico, que deixaram o resultado mais ruidoso.

5.2 Principais contribuições

Em geral o sistema cumpriu o esperado, utilizar de componentes comerciais para a construção de um robô capaz de carregar cargas em terreno com pequenas irregularidades e ser capaz de integrar ao ROS. O desenvolvimento foi todo documentado permitindo estudiosos de robótica e entusiastas a reconstrução da plataforma para o desenvolvimento de estudos e pesquisas que precisem de mobilidade. E, contempla um robô físico, permitindo o estudo e desenvolvimento de ferramentas para integrar ao ROS, aplicação do robô em outros projetos e estudo da plataforma.

As principais contribuições deste trabalho são: fornecer material sobre as etapas de

estudo, desenvolvimento e construção de uma plataforma robótica móvel, estruturando as decisões de projeto; fornecer material sobre a integração do robô com o ROS, visto que os documentos fornecidos pelos desenvolvedores são confusos e em muitas vezes insuficientes, resultando na necessidade estudar códigos de outros robôs desenvolvidos para conseguir desenvolver o projeto.

5.3 Trabalhos futuros

Este trabalho abre oportunidade para o desenvolvimento de diversos outros projetos que possam aproveitar de mobilidade. Porém algumas características mecânicas e elétricas podem ser aperfeiçoadas para as próximas versões. O robô foi modelado para possuir uma estrutura robusta, porém isso implica em um aumento de massa. O robô pode se beneficiar substancialmente em termos de eficiência energética e mobilidade com a redução de massa, é importante o desenvolvimento de estudos de materiais que mantenhas as características mecânicas e permita reduzir a massa do robô.

Outra característica mecânica que pode ser melhorada é a distância da estrutura até o chão. Ao tentar passar por alguns obstáculos maiores o robô encosta no obstáculo, na região entre as duas rodas. Um estudo para melhorar o projeto mecânico pode ser feito para corrigir este problema e permitir que o robô consiga acessar locais mais difíceis.

Atualmente o robô utiliza um modelo dinâmico diferencial fornecido junto com o ROS *control*, este modelo envia velocidades iguais para as duas rodas da esquerda e velocidades iguais para as duas rodas da direita. Um estudo mais aprofundado da dinâmica do robô pode ser feito, buscando desenvolver ou utilizar um modelo mais preciso favorecendo o controle do robô.

Referências

1 ČAPEK, K. *R.U.R. (Rossum's universal robots)*. NewYork: Penguin Group, 2004. ISBN 0141182083military.

2 MATARIC, M. J. *Introdução à robótica*. [S.l.: s.n.], 2014. ISBN 9788521208532.

3 CLARKE, R. Asimov's laws of robotics: implications for information technology-Part I. *Computer*, v. 26, n. 12, p. 53–61, 1993. ISSN 00189162. Disponível em: <http://ieeexplore.ieee.org/document/247652/>.

4 SICILIANO, B. et al. *Robotics*. London: Springer London, 2009. (Advanced Textbooks in Control and Signal Processing). ISBN 978-1-84628-641-4.

5 ROSÁRIO, J. M. *Robótica industrial I : modelagem, utilização e programação*. [S.l.: s.n.], 2010. ISBN 9788579231452.

6 NILSSON, N. J. A Mobile Automaton: An Application of Artificial Intelligence Techniques. v. 1969, n. May, p. 7–9, 1969. Disponível em: <http://www.dtic.mil/docs/citations/ADA459660>.

7 PIERI, E. R. de. *Curso de robótica móvel*. [S.l.], 2002. Disponível em: <http://www.adororobotica.com/CURSO_DE_ROBOTICA_MOVEL.pdf>.

8 SILVA, L. R. D. U.-P. *Análise e programação de robôs móveis autônomos da plataforma eyebot (Dissertação Mestrado)*. [S.l.: s.n.], 2003. 91 p. ISBN 9780262015356.

9 EVERETT, H. Robotics security systems. *IEEE Instrumentation & Measurement Magazine*, v. 6, n. 4, p. 30–34, dec 2003. ISSN 1094-6969. Disponível em: <http://ieeexplore.ieee.org/document/1251480/>.

10 SIEGWART, R.; SCARAMUZZA, D.; NOURBAKHSH, I. R. *Introduction to autonomous mobile robots*. [S.l.: s.n.], 2011. ISBN 978-0-262-01535-6.

11 LATOMBE, J.-C. *Robot motion planning*. [S.l.]: Springer Science & Business Media, 2012. v. 124. ISBN 1461540224.

12 SPRINGER, P. J. *Military robots and drones: a reference handbook*. [S.l.]: ABC-CLIO, 2013. ISBN 1598847325.

13 CORKE, P. *Robotics, Vision and Control*. [S.l.: s.n.], 2011. v. 73. ISBN 978-3-642-20143-1.

14 NEHMZOW, U. *Robot behaviour: Design, description, analysis and modelling*. [S.l.: s.n.], 2009. 1–252 p. ISBN 9781848003965.

15 MOBILEROBOTS, A. *Seekur Jr*. [S.l.], 2011. Disponível em: <https://www.generationrobots.com/media/SeekurJr-Datasheet-RevB.pdf>.

16 Clearpath Robotics. *Husky, Unmanned Ground Vehicle*. [S.l.], 2016. Disponível em: <http://bit.ly/1L2FtEH>.

17 SEGWAY. *RMP 440 LE*. [S.l.], 2011. Disponível em: <http://www.segway.com/media/1220/rmp440le.pdf>.

18 Universiteit Van Amsterdam. *Deliverable: D1.3 Customised Mobile Platform*. [S.l.], 2011. Disponível em: <https://www.frogrobot.eu/wordpress/wp-content/uploads/2014/03/D1.3_Final.pdf>.

19 BAEK, H.; LIM, J. Design of Future UAV-Relay Tactical Data Link for Reliable UAV Control and Situational Awareness. *IEEE Communications Magazine*, v. 56, n. 10, p. 144–150, 2018. ISSN 15581896.

20 CARRERAS, M. et al. Sparus II AUV - A Hovering Vehicle for Seabed Inspection. *IEEE Journal of Oceanic Engineering*, v. 43, n. 2, p. 344–355, 2018. ISSN 03649059.

21 RIBAS, D. et al. Girona 500 AUV: From survey to intervention. *IEEE/ASME Transactions on Mechatronics*, IEEE, v. 17, n. 1, p. 46–53, 2012. ISSN 10834435.

22 Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, A. N. ROS: an open-source Robot Operating System. n. Figure 1, p. 679–686, 2009. ISSN 01628828.

23 CHITTA, S. et al. ros_control: A generic and simple control framework for ROS. *The Journal of Open Source Software*, v. 2, n. 20, p. 456, 2017.

24 MEEUSSEN, W. *ros_control*. 2018. Disponível em: <http://wiki.ros.org/ros_control>.

25 MAGYAR, B. *Create your own hardware interface*. 2017. Disponível em: <http://wiki.ros.org/ros_control/Tutorials/Createyourownhardwareinterface>.

26 MAGYAR, B. *diff_drive_controller*. 2018. Disponível em: <http://wiki.ros.org/diff_drive_controller>.

27 SUCAN, I.; KAY, J. *urdf*. 2019. Disponível em: <http://wiki.ros.org/urdf>.

28 GRISETTIYZ, G.; STACHNISS, C.; BURGARD, W. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005. v. 2005, p. 2432–2437. ISBN 0-7803-8914-X. ISSN 10504729. Disponível em: <http://ieeexplore.ieee.org/document/1570477/>.

29 GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, v. 23, n. 1, p. 34–46, feb 2007. ISSN 1552-3098. Disponível em: <http://ieeexplore.ieee.org/document/4084563/>.

30 HÖRNER, J. *Map-merging for multi-robot system*. Tese (Bachelor's thesis) — Charles University, 2016. Disponível em: <https://is.cuni.cz/webapps/zzp/detail/174125/>.

31 BRAUN, J. *Formulae Handbook*. [S.l.]. 60 p. Disponível em: <http://maxon.blaetterkatalog.ch/b9991/catalog/index.html?data=b9991/b999145&lang=e>.