UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

VITOR FERNANDO DE SOUZA GADELHA

Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

Manaus
2022

VITOR FERNANDO DE SOUZA GADELHA

Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

Versão Original

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Controle e Automação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Almir Kimura Júnior

Manaus

2022

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).
Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.

D278pp Gadelha, Vitor Fernando de Souza
Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais / Vitor Fernando de Souza Gadelha. Manaus : [s.n], 2022.
130 f.: ; 30 cm.

TCC - Graduação em Engenharia de Controle e Automação; - Universidade do Estado do Amazonas, Manaus, 2022.
Inclui bibliografia
Orientador: Junior, Almir Kimura

1. Robô. 2. Impressão 3D . 3. Modelagem CAD. 4. ROS. 5. Mapeamento. I. Junior, Almir Kimura (Orient.). II. Universidade do Estado do Amazonas. III. Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

Elaborado por Jeane Macelino Galves - CRB-11/463

Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

VITOR FERNANDO DE SOUZA GADELHA

Trabalho de Conclusão de Curso (TCC) apresentado à Escola Superior de Tecnologia da Universidade do Estado do Amazonas como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Aprovado por:

Prof. Dr. Almir Kimura Junior
Orientador (UEA)

Prof. Dr. Israel Mazaira Morales
Avaliador (UEA)

Prof. Msc. Moisés Pereira Bastos
Avaliador (UEA)

Prof. Msc. Cleto Cavalcante De Souza Leal
Avaliador (UEA)

Manaus-AM, 26 de maio de 2022

À minha família,

dedico.

AGRADECIMENTOS

À meus pais, Pierre e Mary *(in memoriam)*, que sempre me incentivaram a alcançar meus sonhos e nunca deixaram faltar nada para que eu pudesse alcançá-los. Vocês são o meu exemplo de vida e sem vocês, nada disso seria possível.

A minha irmã, Rayssa, por não só estar comigo em todos os momentos mas também por ser uma parceira e estar comigo enfrentando qualquer desafio.

A minha amiga e parceira, Rebeca, por estar comigo em todos os momentos da faculdade me apoiando e incentivando a melhorar cada vez mais.

Ao meu orientador, Prof. Dr. Almir Kimura Júnior, por todo o auxílio e confiança durante a elaboração deste projeto. Obrigado por ter aceitado o convite e me orientado da melhor maneira possível para finalizar este trabalho.

A todos os meus amigos que fiz na faculdade, em especial, Lucas e Igor por não só tornarem este curso mais divertido, mas também por não me deixarem desistir em nenhum momento deste projeto.

Aos amigos do Laboratório de Fabricação Digital, por ajudarem no desenvolvimento deste projeto e por todos os desabafos e conversas que tivemos.

Aos amigos da escola, por sempre ouvirem e apoiarem todas as ideias e planos que já tive e por terem caminhado comigo durante toda esta jornada. Vocês são incríveis.

A Universidade do Estado do Amazonas, por possibilitar a realização deste curso.

A todos que contribuíram direta ou indiretamente para realização deste trabalho.

‘"Quem diz que não pode ser feito nunca deve interromper aquele que está fazendo."

(Monkey D. Luffy - Eiichiro Oda)

RESUMO

A utilização de robôs móveis está cada vez mais presente no desenvolvimento da sociedade, desde tarefas simples como serviços domésticos até tarefas mais complexas como missões de resgate. Atualmente, o principal desafio de um robô móvel se dá quando o robô está inserido em um ambiente desconhecido, fazendo com que sua rota seja recalculada diversas vezes, a partir da detecção de informações do ambiente, até atingir seu objetivo final. Sendo assim, esse trabalho propõe o desenvolvimento de um protótipo de veículo autônomo terrestre capaz de se locomover por ambientes industriais e mapeá-lo. O modelo CAD do veículo foi primeiramente projetado no *software Solidworks* e para ter um baixo custo de produção, suas peças foram produzidas utilizando manufatura aditiva através de impressão 3D. A parte elétrica do veículo conta com um *Raspberry Pi* para processamento dos dados provenientes de diversos sensores capazes de detectar obstáculos e fazer o mapeamento da área, além de atuadores que possibilitam sua locomoção no ambiente. Para facilitar as conexões entre os dispositivos presentes, foi desenvolvida uma placa de circuito impresso (PCB) que funciona como um *shield*, conectando o microcontrolador utilizado aos dispositivos. Para a programação dos algoritmos, optou-se por utilizar o *Robot Operating System* (*ROS*) por facilitar a conexão entre os dispositivos e conter pacotes para a implementação dos algoritmos de mapeamento, tais como o *Hector SLAM*. O robô foi montado e testado em 3 ambientes com características industriais e apresentou uma média de $93,67\%$ de precisão em relação a planta baixa do local, levando aproximadamente $12,06 min/m^2$ para realizar o mapeamento em cada um dos ambientes.

Palavras-chave: Robô, Impressão 3D, Modelagem CAD, ROS, Mapeeamento.

ABSTRACT

The use of mobile robots is increasingly present in the development of society, from simple tasks like domestic services to more complex ones like rescue missions. Currently, the main challenge of a mobile robot occurs when the robot is inserted in an unknown environment, causing its route to be recalculated several times, with the detection of information from the environment, until reaching its final goal. Therefore, this work proposes the development of a prototype of an autonomous terrestrial vehicle capable of moving around industrial environments and mapping it. The CAD model of the vehicle was first designed in the *Solidworks software* and to have a low production cost, its parts were produced using additive manufacturing through 3D printing. The electrical part of the vehicle has a *Raspberry Pi* for processing data from several sensors capable of detecting obstacles and mapping the area, in addition to actuators that allow its locomotion in the environment. To facilitate the connections between the various devices present, a printed circuit board (PCB) was developed that works as a *shield*, connecting the microcontroller used to the devices. For programming the algorithms, the *Robot Operating System* (*ROS*) was chosen to facilitate the connection between the various devices and for containing packages for implementing the mapping algorithms, such as Hector SLAM. The robot was assembled and tested in 3 environments with industrial characteristics and presented an average of 93.67% of accuracy in relation to the floor plan of the place, taking approximately $12.06 min/m^2$ to perform the mapping in each one of the environments.

Keywords: Robot, 3D Printing, CAD modeling, ROS, Mapping.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
AGV	Auto Guided Vehicle
CAD	Computer Aided Design
FDM	Fused Deposition Modeling
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GUI	Graphical User Interface
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
NASA	National Aeronautics and Space Administration
PWM	Pulse Width Modulation
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
SO	Sistema Operacional
URDF	Universal Robot Description Format

SUMÁRIO

1 INTRODUÇÃO

A habilidade de navegar, monitorar ambientes e se locomover por ambientes é de suma importância para o desenvolvimento da sociedade. No entanto, nem sempre é possível fazer esse monitoramento em determinados locais, visto a dificuldade de acesso em determinados ambientes ou a hostilidade do ambiente ao ser humano. Sendo assim, muitas vezes se faz necessária a utilização de tecnologias como robôs móveis para solucionar esse problema.

Matarić (2007) define robô como: "um sistema autônomo que existe no mundo físico, pode sentir o seu ambiente e pode agir sobre ele para alcançar determinado objetivo". Dudek e Jenkin (2010) definem a robótica móvel com uma área de pesquisa que lida com o controle de sistemas autônomos e veículos semiautônomos. A diferença da robótica móvel para outras áreas como a robótica de manipuladores convencional é a ênfase em solcuionar problemas que exigem a compreensão do espaço em grande escala, ou seja, regiões do espaço onde não é possível observar de um único ponto. (DUDEK; JENKIN, 2010)

Um dos principais desafios dos robôs móveis se dá quando um robô está inserido em um ambiente desconhecido e com obstáculos dinâmicos, o que faz com que ele tenha que recalcular sua rota diversas vezes (e rapidamente) para chegar ao seu objetivo, evitando colisões. Como o terreno muitas vezes é desconhecido, se faz necessário encontrar uma forma com que o robô se adapte ao ambiente ao qual está inserido.

Dessa forma, este trabalho tem por hipótese a ideia de que ambientes industriais podem ser mapeados utilizando um veículo terrestre autônomo que possa se locomover neste lugar. Nesse contexto, busca-se desenvolver um protótipo de veículo terrestre autônomo capaz de se locomover e mapear ambientes industriais.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um protótipo de veículo terrestre autônomo capaz de se locomover em ambientes industriais e mapeá-los.

1.1.2 Objetivos Específicos

Visando atingir o objetivo principal, alguns objetivos específicos são requeridos, dentre eles:

- Desenvolver o projeto mecânico do robô visando ser possível a locomoção em ambientes industriais;
- Projetar o modelo cinemático do robô em simulação para validar as coordenadas de navegação;
- Desenvolver PCB (Placa de Circuito Impresso) para comandos elétricos e controle do robô de maneira mais eficiente;
- Programar os dispositivos presentes no robô de forma a deixá-lo autônomo;
- Programar e testar o mapeamento gerado pelo robô a fim de validar se os dados obtidos estão corretos;
- Testar o robô em ambiente controlado para verificar seu funcionamento e validar os dados in loco para fazer possíveis ajustes necessários.

1.2 Divisão do Trabalho

O trabalho está organizado da seguinte maneira: após a presente introdução, o Capítulo 2 aborda o Referencial Teórico, o qual aborda todos os assuntos necessários para a implementação do projeto. O Capítulo 3 consiste na implementação do projeto, indicando os materiais e métodos utilizados na elaboração do mesmo. O capítulo 4 apresenta os resultados e discussões obtidos no decorrer do projeto. Por fim, no capítulo 5, apresentam-se as considerações finais e sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

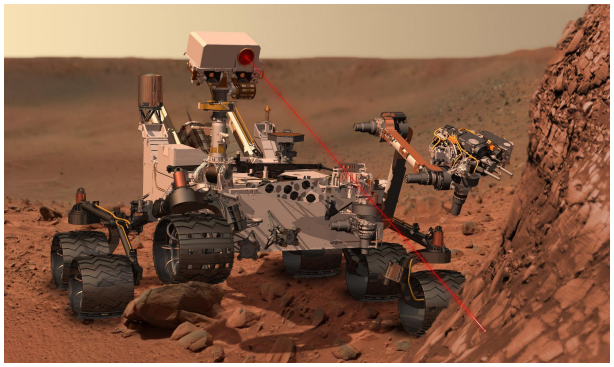
Neste capítulo, serão apresentadas as teorias necessárias para modelagem e implementação do protótipo do robô. Inicialmente serão apresentados conceitos de robótica móvel, seguidos de conceitos e técnicas de navegação e mapeamento. Posteriormente, serão apresentados conceitos para programação de robôs, fabricação digital e a teoria de controle clássico.

2.1 Robótica Móvel

Segundo Matarić (2007), um robô é um sistema autônomo que existe no mundo físico, pode sentir o seu ambiente e pode agir sobre ele para alcançar determinado objetivo. Segundo Dudek e Jenkin (2010), a robótica móvel é uma área de pesquisa que lida com o controle de sistemas autônomos e veículos semiautônomos. O que diferencia a robótica móvel de outras áreas de pesquisa como robótica de manipuladores convencional, inteligência artificial e visão computacional é a ênfase nos problemas relacionados à compreensão do espaço em grande escala, ou seja, regiões do espaço substancialmente maiores do que aquelas que podem ser observadas de um único ponto.

Atualmente, robôs móveis são muito utilizados em ambientes inóspitos e/ou remotos, onde se torna inviável enviar um humano para desenvolver alguma atividade. Utilizando os dados obtidos pelos sensores, o robô pode navegar autonomamente em um ambiente dinâmico e desconhecido, sem a assistência de humanos por exemplo, o robô Curiosity da NASA (figura 1).

Figura 1: Robô Curiosity da NASA



Fonte: NASA (2021)

2.2 Tipos de Robôs Móveis

Na literatura, podemos encontrar algumas classificações para robôs móveis segundo suas funcionalidades, como por exemplo:

a)Robôs Industriais - são aqueles que atuam em ambientes industriais, possuem conhecimento do ambiente em que estão circulando por meio da sensoriamento da sua própria posição e de objetos no ambiente. Estes robôs geralmente trabalham com o transporte de cargas dentro da indústria, seguem um caminho pré-determinado por uma linha no chão e são denominados AGVs (Automated Guided Vehicles), como o da figura 2 (PIERI, 2002);

Figura 2: AGV da Lisen Automation

Fonte: Lisen Automation (2021)

b)Robôs de serviço - são robôs para trabalho em ambientes como residências e laboratórios que realizam tarefas com certa autonomia, utilizando dados provenientes de sensores para detecção de obstáculos, como o da figura 3. São aplicáveis a diversas atividades tais como limpeza, transporte, vigilância, etc. (PIERI, 2002);

Figura 3: Robô Aspirador Samsung

Fonte: Samsung (2021)

c)Robôs de campo - são robôs que executam tarefas em ambientes externos, pouco conhecidos e muitas vezes perigosos. Podem ser usados em exploração espacial, mapeamento, resgate, dentre outras aplicações. Devido ao ambiente em que atuam é necessário um alto grau de autonomia e processamento para gerenciar os dados de diversos sensores presentes no mesmo, um exemplo é o modelo encontrado na figura 4; (PIERI, 2002)

Figura 4: Robô SpaceBok

Fonte: Spacebok (2021)

d)Robôs para Pesquisa - são robôs geralmente utilizados em estudos e pesquisas nas universidades, como o NAO da figura 5 (PIERI, 2002)

Figura 5: Robô NAO

Fonte: RobotLab (2021)

e)Robôs para Entretenimento - são robôs geralmente associados ao entretenimento tendo um público variado. Geralmente apresentam um comportamento ou atividades já programadas, como a interação com usuário ou brinquedos inteligentes, por exemplo o robô Miposaur da figura 6; (PIERI, 2002)

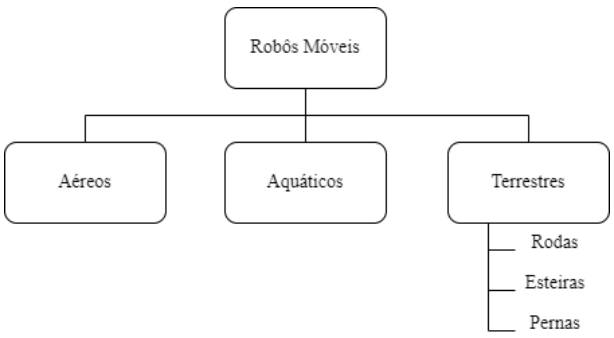
Figura 6: Robô Miposaur

Fonte: WowWee (2021)

2.3 Classificação de Robôs Móveis quanto a Mobilidade

Os robôs móveis podem ser encontrados na literatura sob algumas classificações, como visto em Pieri (2002), Siegwart, Nourbakhsh e Scaramuzza (2011) e Bräunl (2008). Quanto a mobilidade, Pieri (2002) classifica os robôs em três grandes grupos, sendo eles: aéreos, aquáticos e terrestres conforme a figura 7.

Figura 7: Classificação segundo a Mobilidade



Fonte: Pieri (2002)

A definição dos tipos de robôs móveis terrestres segundo a classificação da figura 7 pode ser, segundo autores da área:

a) Robôs com Rodas - Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), são

robôs que utilizam uma ou mais rodas para sua locomoção, isso faz com que o projeto e construção do modelo possam ser mais simples e de custo menor. Como passam todo tempo em contato com a superfície em que estão percorrendo, necessitam de uma menor preocupação com equilíbrio. Com isso, o foco no projeto passa a ser na tração e controle do robô. Um exemplo é o RoboMaster S1 da DJI encontrado na figura 8.

Figura 8: RoboMaster S1

Fonte: DJI (2021)

b) Robôs com Esteira - robôs que apresentam boa manobrabilidade em terrenos irregulares por possuirem direção diferencial e possuírem um maior contato físico com o solo, um exemplo é o robô Wall-e do filme de mesmo nome da Pixar que pode ser visto na figura 9. (BRäUNL, 2008)

Figura 9: Wall-e

Fonte: Pixar (2021)

c) Robôs com Pernas - São robôs que possuem um conjunto de pernas para locomoção. São inspirados no movimento de animais e possuem apenas alguns pontos de contato com o chão. Adaptam-se melhor em terrenos irregulares e tem uma maior adaptabilidade e manobrabilidade, porém necessitam de um certo nível de liberdade nas articulações para realizarem os movimentos, como o robô Hexa da figura 10. (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

Figura 10: Robô Hexa

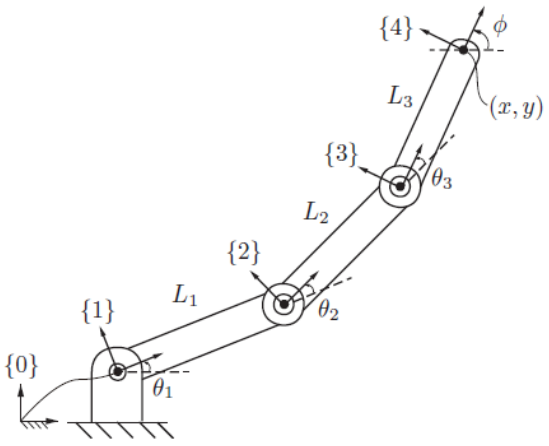
Fonte: Vincross (2021)

2.4 Cinemática de robôs

Como visto em Siegwart, Nourbakhsh e Scaramuzza (2011) a cinemática é uma área da mecânica que estuda o movimento dos corpos e como sistemas mecânicos atuam. Na robótica móvel, é necessário entender como o sistema mecânico irá atuar para então ser possível o design apropriado do robô e controle do mesmo.

Em manipuladores fixos, a cinemática parte do princípio que pelo menos uma extremidade do robô está fixa em relação ao local de trabalho, sendo possível então definir todas as outras coordenadas a partir deste ponto. Um exemplo pode ser visto na figura 11 em que como a base do robô está fixada no solo é possível determinar todas as posições dos elos e juntas do robô. No entanto, em se tratando de robôs móveis, não há nenhuma maneira de se definir essa posição instantaneamente o que faz com que seja necessário o estudo do movimento do robô ao longo do tempo.

Figura 11: Modelo Cinemático de um Manipulador Robótico



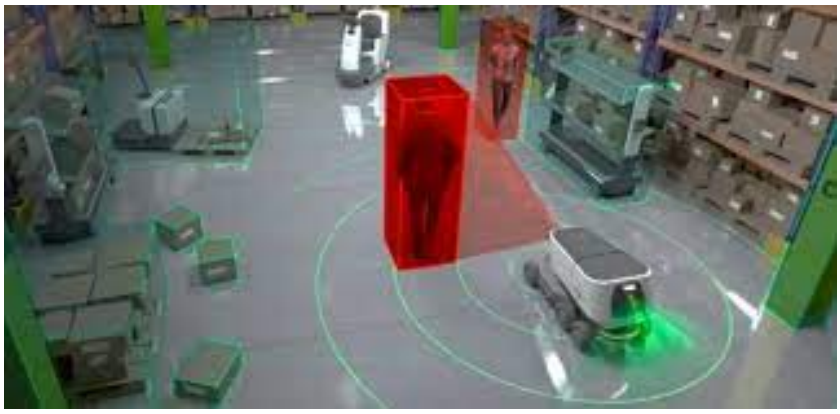
Fonte: Lynch e Park (2017)

Segundo Craig (1977) podemos definir a cinemática em dois grandes problemas: a cinemática direta, que consiste em se utilizar a posição dos atuadores para chegar a uma certa posição e orientação no espaço, e a cinemática inversa, em que a partir das posições e orientação desejadas, define a movimentação do robô para atingi-las.

2.5 Navegação de robôs

Franz e Mallot (2000) definem o termo navegação como: levar um navio até um destino final. Este processo consiste em três etapas: o navegador determina a posição e orientação do barco seguindo uma carta náutica, após isso determina a posição do destino e possíveis obstáculos e a partir disso define o trajeto que o navio irá percorrer. Aplicando esse conceito à robótica, este mesmo processo pode ser aplicado, como na figura 12 em que o robô deve se locomover no ambiente enquanto detecta as posições das pessoas ao redor.

Figura 12: Robôs Móveis durante processo de navegação em ambiente industrial



Fonte: Santora (2018)

Já para Trullier et al. (1997), a palavra navegação se refere a todas as estratégias aplicadas pelo robô para se mover até determinado objetivo. Essas estratégias consistem em sair de uma posição a outra até navegações mais complexas em que o robô possa desviar de obstáculos.

As principais abordagens utilizadas nas técnicas de navegação de robôs móveis são, segundo Rodrigues (2010):

a) Abordagem Sensorial / Reativa - nessa abordagem, o robô utiliza os dados provenientes de diversos sensores acoplados a ele para poder planejar o seu trajeto. A

principal vantagem dessa abordagem é que o robô não tem a necessidade de conhecer antecipadamente seu ambiente e pode atuar mais rapidamente sobre alguma situação inesperada.

b) Abordagem Roadmap - essa abordagem consiste em reduzir as informações do ambiente em um grafo que representa os possíveis caminhos existentes. No geral, esse método é mais simples de implementar, porém não fornece uma boa forma de representação do ambiente.

c) Abordagem utilizando Matrizes - essa abordagem utiliza matrizes em que cada célula da matriz possui atributos para auxiliar na navegação do robô, indicando por exemplo, um obstáculo ou caminho livre. O uso de matrizes facilita a determinação de caminhos até o destino, porém necessitam de um certo tempo de processamento para atualização global do sistema.

2.6 Técnicas de Navegação

Trullier e Meyer (1997) propõem uma classificação inspirada nas técnicas de navegação usadas pelos animais, que podem ser definidas em cinco categorias, que são:

a) Aproximação de um objeto: o robô tem a capacidade de se dirigir a determinado objeto visível a partir de uma posição inicial. No entanto, é uma estratégia local que só funciona quando o objeto é visível ao robô.

b) Guiagem: o robô é capaz de seguir referências presentes no ambiente para se locomover, por exemplo, uma fita no chão. Também é uma estratégia local que não precisa de um modelo do ambiente.

c) Resposta associada a um local: que permite o robô atingir um objetivo a partir de posições e referências não visíveis. Necessita que o robô já possua uma representação interna do ambiente para definir sua rota.

d) Navegação topológica: uma extensão da técnica anterior que armzena as informações dos locais e obstáculos presentes. Dessa forma, o modelo interno pode ser definido como um grafo que permite calcular diversas rotas entre dois locais arbitrários.

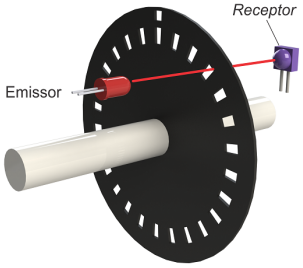
e) Navegação métrica: extensão da estratégia anterior cujo robô consegue planejar sua rota passando por zonas desconhecidas do seu ambiente. Para isso, o robô

memoriza suas posições no espaço permitindo uma trajetória calculada.

Quando nos referimos à navegação de robôs móveis autônomos, descrevemos técnicas que possibilitam a locomoção do robô de forma segura a determinada localização do espaço. Borenstein et al. (1997) apresenta uma clasificação em dois grandes grupos das técnicas utilizadas em navegação, sendo eles:

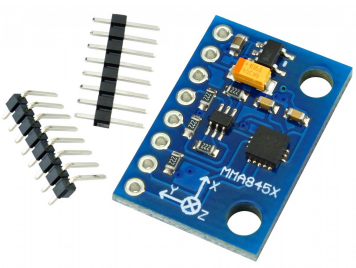
a) Navegação baseada em medidas relativas da posição: nesse método o robô utiliza sensores para determinar sua posição e orientação no espaço, fazendo com que ele possa conhecer sua localização naquele ambiente. Geralmente podem ser utilizandos encoders (figura 13) para uma navegação odométrica, em que o robô se localiza a partir da rotação de suas rodas, ou unidades de medição inerciais (IMU) (figura 14), em que o robô determina sua localização por meio das medidas de rotação e aceleração do veículo.

Figura 13: Exemplo de Encoder



Fonte: Hitec Tecnologia (2021)

Figura 14: Exemplo de Acelerômetro



Fonte: Easytronics (2021)

b) Navegação baseada em medidas absolutas da posição: neste método o robô utiliza sensores tais como laser scanners (figura 15), receptores de radiofrequência e módulos *GPS* (figura 16) para determinar sua posição absoluta em relação ao ambiente. Tal método faz com que seja mais fácil de determinar sua localização global no ambiente e traçar trajetórias num ambiente desconhecido.

Figura 15: Exemplo de Lidar

Fonte: Slamtec (2021)

Figura 16: Exemplo de GPS

Fonte: Eletrogate (2021)

2.7 *Raspberry Pi*

Conforme a *Raspberry Pi* Foundation (2021a), o *Raspberry Pi* é um computador de pequeno porte e que contém todas as funcionalidades de um computador normal, sendo muito utilizado para projetos de prototipagem e aplicações embarcadas. O modelo 4B conta com versões com 2GB, 4 GB e 8 GB de memória RAM, cujas especificações podem ser vistas na tabela 1.

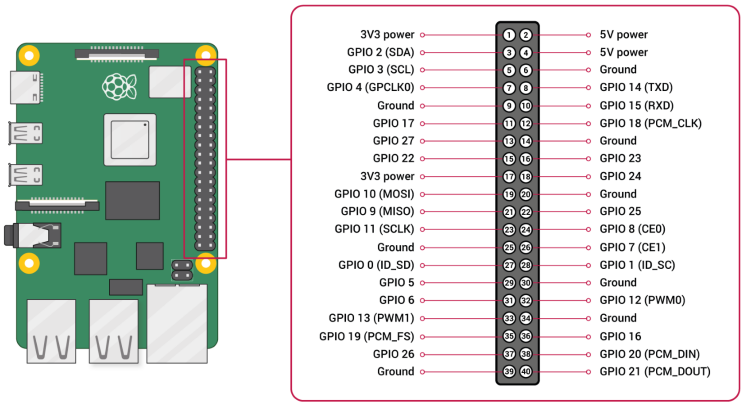
Tabela 1: Especificações técnicas do *Raspberry Pi* 4B

Recurso	**Especificação**
Processador	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memória	2,4 ou 8 GB LPDDR4
Conectividade	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet, 2 portas × USB 3.0, 2 portas × USB 2.0
Acessos	Cabeçalho GPIO padrão de 40 pinos
Vídeo	2 entradas × micro HDMI, 1 porta para Display, 1 porta para Câmera
Suporte a Cartão SD	Slot de Cartão SD para carregar o Sistema Operacional e armazenamento de dados

Fonte: *Raspberry Pi* Foundation (2021a)

A pinagem do *Raspberry Pi* pode ser vista na figura 17. O pinos 2 e 4 são pinos de alimentação 5V e não podem entrar em contato com nenhuma outra entrada do *Raspberry Pi*. O *Raspberry* considera 3,3V como sendo o nível lógico alto, e pelo fato do dispositivo não possuir um sistema de proteção, caso uma tensão superior a 3,3V entre em contato com qualquer outra entrada do raspberry, ocorrerá a queima da entrada ou da placa inteira

Figura 17: *GPIO* do *Raspberry Pi* 4B



Fonte: *Raspberry Pi* Foundation (2021a)

2.7.1 *Raspicam*

Segundo a *Raspberry Pi* Foundation (2021b), existem vários módulos de câmera oficiais do *Raspberry Pi*. O modelo original tem 5 megapixels e foi lançado em 2013, já o modelo mais utilizado atualmente é o módulo de câmera v2 que possui 8 megapixeels e foi lançado em 2016. Os dois modelos contam com versões de luz visível e infravermelho. As especificações técnicas do modelo v2 podem ser vistas na tabela 2.

Tabela 2: Especificações técnicas da *Raspicam* v2

Recurso	**Especificação**
Resolução	8 MP
Modos de Vídeo	1080p30, 720p60 e 640 × 480p60/90
Sensor	Sony IMX219
Resolução do Sensor	3280 × 2464 pixels
Abertura focal	1/4"

Fonte: *Raspberry Pi* Foundation (2021b)

Todas as câmeras podem capturar imagens de alta resolução e tem resolução de vídeo full HD (1080p), podendo ser controladas via software por algoritmos gerados pelo usuário, o que faz com que sejam muito utilizadas em atividades de Processamento Digital de Imagens e captura de vídeo em robôs. O modelo *Raspicam v2*, pode ser visto na figura 18.

Figura 18: Raspicam modelo v2

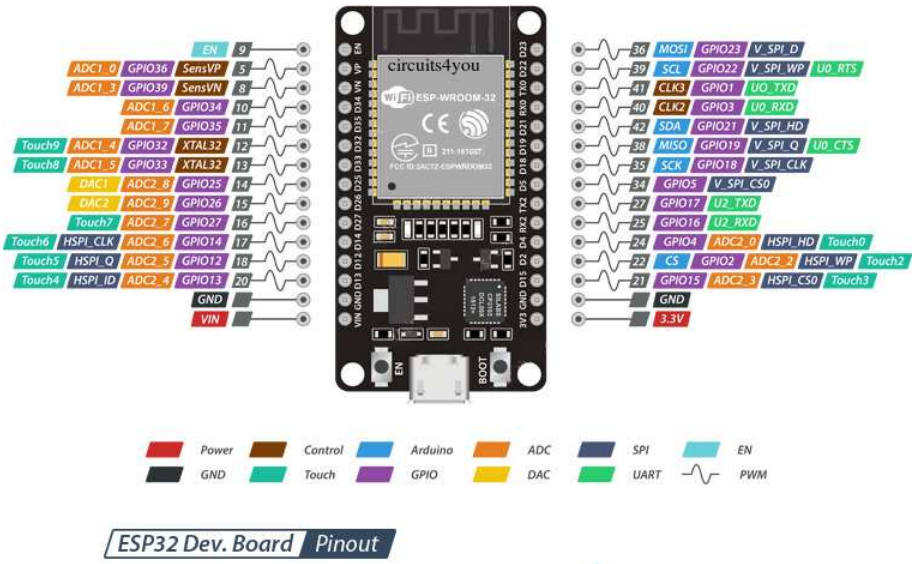
Fonte: *Raspberry Pi* Foundation (2021b)

2.8 *ESP32*

Conforme visto em *Espressif* Systems (2022), o módulo *ESP32* é um módulo de alta performance para dispositivos móveis, wearables e aplicações de *IoT*, com *WiFi* e Bluetooth integrado.

Na placa *ESP32 DevKit v1*, temos o chip *ESP32* com antena embutida, uma interface *USB*-serial e regulador de tensão 3.3V. Sua programação pode ser feita através da *IDE* do Arduino utilizando a interface serial e sua pinagem pode ser vista na figura 19.

Figura 19: *ESP32 DevKit v1*



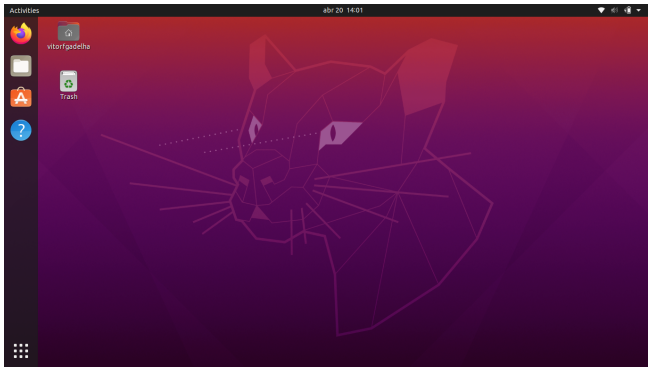
Fonte: *Espressif* Systems (2022)

2.9 Ubuntu

Uma das principais decisões ao se projetar um robô é a escolha do sistema operacional utilizado a fim de que o robô seja sustentável e seguro (CANONICAL, 2018). Com isso, o Linux se tornou o principal sistema operacional para o desenvolvimento de sistemas embarcados, sendo amplamente utilizado em projetos de automação e robótica.

Uma das principais distribuições Linux utilizada é a Ubuntu, produzida pela empresa Canonical, sendo um sistema operacional completo que pode ser utilizado em qualquer computador, cujo ambiente pode ser visto na figura 20. Além disso, a Canonical oferece diversas variantes com ambientes gráficos otimizados e diversos conjuntos de aplicativos para melhor performance como o Ubuntu MATE ou o Kubuntu.

Figura 20: Área de Trabalho do Ubuntu 20.04 LTS



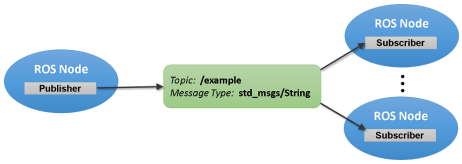
Fonte: Autor (2022)

2.10 *ROS* - Robot Operating System

Segundo Quigley et al. (2009), o *ROS* é um *framework* para desenvolvimento de software para robôs e é constituído por ferramentas e bibliotecas que simplificam a tarefa de criar rotinas e comportamentos em diversas plataformas.

Zaman, Slany e Steinbauer (2011) mostram que uma das principais características do *ROS* é ser um sistema modular em que vários programas trabalham em conjunto (denominados nós), enviando e recebendo mensagens, no formato de tópicos, para alcance do objetivo programado. Um nó que envia uma mensagem é chamado de *publisher*, enquanto que o nó que recebe a mensagem é denominado *subscriber*, o que pode ser visto na figura 21.

Figura 21: Exemplo de Comunicação por Mensagens em ROS



Fonte: Mathworks (2021a)

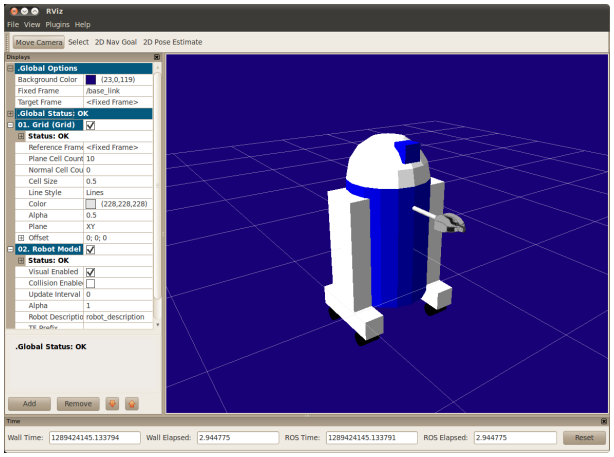
Como o *ROS* é um projeto de código aberto em constante desenvolvimento, existem diversas versões disponíveis, sendo a mais recente a versão *Noetic Ninjemys* (figura 22), lançada em maio de 2020. Além disso, é possível encontrar facilmente projetos paralelos mantidos pela comunidade que tem funcionalidades específicas ou pacotes para se utilizar diversos sensores e atuadores específicos.

Figura 22: ROS Noetic Ninjemys

Fonte: Open Robotics (2021a)

O *framework* apresenta ainda diversas ferramentas de visualização, tais como *Rviz* e *rqt*, que permitem gerar representações gráficas de diversos elementos, tais como dados de sensores ou posição e movimento do robô que auxiliam o trabalho do desenvolvedor. A partir de um *XML* denominado *URDF* (Universal Robot Description Format) é possível gerar uma representação gráfica simplificada do robô desejado, um exemplo é o robô R2-D2 de Star Wars da figura 23.

Figura 23: Exemplo de um modelo simplificado do robô R2-D2 de Star Wars



Fonte: Open Robotics (2021b)

2.11 *SLAM* - Simultaneous Localization and Mapping

Segundo as informações encontradas no site da Mathworks (2021b), *SLAM* é um método utilizado por veículos autônomos que permite construir um mapa e se localizar nesse mapa ao mesmo tempo. Isso faz com que seja possível que o veículo mapeie ambientes desconhecidos assim podendo ser possível um planejamento de rotas e desvio de obstáculos.

Atualmente, as técnicas de *SLAM* tem sido objeto de pesquisas técnicas pelo mundo todo. Soma-se isso a uma maior disponibilidade de sensores de baixo custo e uma melhoria na velocidade de processamento o que faz com que o método de *SLAM* seja utilizado em um número crescente de campos.

Os principais métodos de *SLAM* são:

a) Visual *SLAM*: que utiliza imagens adquiridas por câmeras ou sensores podendo ser implementado a um mais baixo custo. Como as câmeras fornecem um grande volume de informações, podem ser usadas para detectar diversos pontos de referência e otimizadas com algoritmos de visão computacional.

b) Lidar *SLAM*: os sensores Lidar (Light Detection and Ranging) são capazes de detectar distâncias e possuem maior precisão em relação às câmeras. Os valores de saída desse tipo de sensor são denominados *Point Clouds* ou nuvem de pontos, podendo ser 2D ou 3D. Ao processar essa nuvem de pontos são utilizados algoritmos de registro como o *ICP* (Iterative Closest Point) para determinar a localização do veículo o que

faz com que o robô necessite de um maior poder de processamento.

Devido a isso, a localização para veículos autônomos geralmente envolve a fusão dos dois métodos além da utilização de outros sensores como *GPS* e *IMU* para um resultado mais preciso do mapa gerado.

2.12 Fabricação Digital

2.12.1 Modelagem *CAD*

Segundo o site da Autodesk (2022), modelagem 3D é o processo de se desenhar um objeto ou forma tridimensional utilizando softwares. O objeto é chamado modelo 3D e pode ser usado para diversas aplicações, tais como televisão, videogames, jogos, arquitetura e engenharia.

A modelagem 3D consiste na utilização de formas primitivas como cubos, esferas e outras formas e a partir de ferramentas computacionais se definem novas formas a partir da primitiva, sendo possível gerar qualquer tipo de corpo para posterior manufatura.

Na engenharia, é comum iniciar o processo de desenvolvimento de novos produtos ou máquinas a partir da modelagem 3D, visto que a partir de simulações no modelo podem garantir o bom desenvolvimento do objeto real.

2.12.2 Manufatura aditiva

De acordo com os websites da Lab (2021), Mecalux (2021), Technologies (2021), a manufatura aditiva pode ser descrita como o conjunto de tecnologias que utiliza desenhos assistidos por computador (*CAD*) e produz objetos 3D a partir de uma impressora 3D, por exemplo. Com ela, é possível produzir peças complexas otimizando recursos, sendo que cada tipo de impressão 3D tem uma forma característica de se moldar o objeto desejado.

As vantagens de se utilizar manufatura aditiva são:

• Custo: pois permite produzir peças em pequenas quantidades e geralmente com custo menor do que de usinagem.

• Velocidade: pois é possível produzir eficientemente um modelo digital de ma-

neira mais rápida.

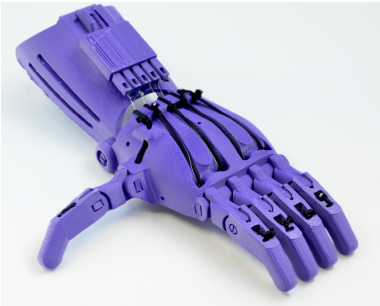
• Complexidade: as impressoras 3D permitem a criação de peças e modelos bastante complexos.

• Customização: as peças e modelos são totalmente customizáveis de acordo com a necessidade do projetista.

• Economia: como podem ser feitas peças de acordo com a necessidade é possível diminuir o uso de material, gerando menos resíduo.

Atualmente, o número de empresas que utiliza a manufatura aditiva em seus processos de produção vem crescendo. Indústrias como as de saúde, aeroespacial e automobilística já vem implementando o processo de impressão 3D em seus produtos, um exemplo é o projeto Raptor Reloaded visto na figura 24.

Figura 24: Prótese Raptor Reloaded



Fonte: UFSC (2021)

As principais tecnologias para Impressão 3D são: a Modelagem por Fusão e Deposição (*FDM*) que extrusa um material plástico derretido acrescentando camada por camada até formar o objeto final, a estereolitografia que faz a solidifação de resina através de um feixe de laser ultravioleta e a manufatura aditiva de metal que sintetiza pó de metal em um objeto por meio de um laser.

2.12.3 Eletrônica para fabricação digital

Eletrônica é a ciência que estuda a utilização de circuitos formados por componentes elétricos e eletrônicos. Os principais componentes encontrados na robótica são os sensores, dispositivos que podem fazer com que o robô tenha informações sobre ele mesmo e o ambiente ao qual está inserido, e atuadores, que são dispositivos capazes de entrar em contato com o ambiente e atuar sobre o mesmo.

Segundo Bräunl (2008) e Nehmzow (2003), do ponto de vista da robótica, é mais interessante classificar sensores da seguinte maneira:

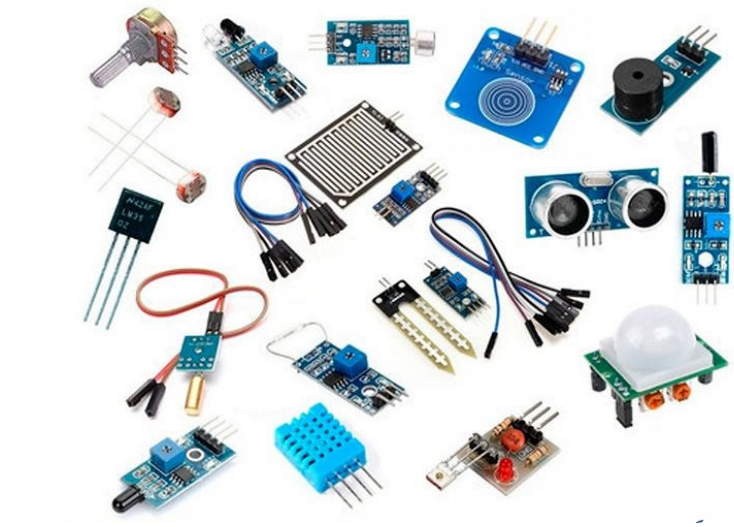
a) Proprioceptivos - que fazem um monitoramento interno de dados internos do robô, por exemplo: sensor de nível de bateria e acelerômetros;

b) Exteroceptivos – que adquirem informação do ambiente ao qual o robô está inserido, por exempo: lasers e sonares.

c) Passivos - Monitoram o ambiente sem interferir no ambiente, por exemplo câmeras e sensores de toque.

d) Ativos - interagem com o ambiente emitindo energia e assim, adquirindo as informações, como por exemplo o GPS;

Figura 25: Exemplos de Sensores



Fonte: Multcomercial (2021)

O funcionamento de alguns tipos de sensores serão mais detalhados nos itens de 2.12.3.1 a 2.12.3.3.

Segundo Matarić (2007), os atuadores podem ser classificados, do ponto de vista da robótica, da seguinte maneira:

a) Locomoção: atuadores que tem a função de locomover o robô pelo ambiente, por exemplo, motores com rodas, cujo cálculo e funcionamento podem ser vistos no item 2.12.3.4

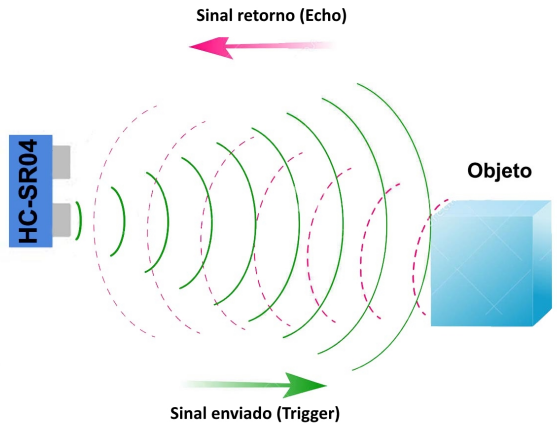
b) Manipulação: atuadores que tem a função de manipular objetos, como é o caso

de braços robóticos, que serão vistos no item 2.12.3.5

2.12.3.1 Sensor Ultrassônico

Conforme visto em Balluff (2022), o sensor ultrassônico funciona bem semelhante ao sonar de morcegos, em que uma onda sonora de alta frequência é emitida e ao ser detectada em algum objeto capaz de refletir essa onda, gera um eco detectado pelo sensor. Com isso, identifica-se a presença de um obstáculo ou objeto e a distância até ele. Esse funcionamento pode ser visto na figura 26.

Figura 26: Funcionamento de um sensor ultrassônico



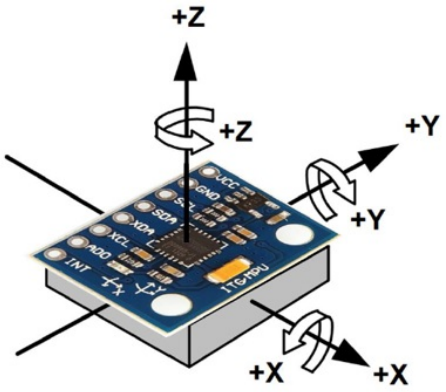
Fonte: Thomsen (2022)

2.12.3.2 Módulo IMU

Uma Unidade de Medição Inercial (IMU) é um dispositivo que contém geralmente giroscópios para medir mudanças angulares e acelerômetros para medição de acelerações, conforme visto em Vectornav (2022).

Como cada sensor individual só consegue detectar as informações relativas a apenas um eixo, comumente se utiliza módulos que consistem na junção de diversos sensores podendo chegar a um sistema de 6 eixos. Cada um dos 3 eixos, contém informações de aceleração e rotação, o que faz com que seja ideal na localização e ajuste de orientação de robôs móveis. O funcionamento de um sensor IMU pode ser visto na figura 27.

Figura 27: Funcionamento de um sensor IMU



Fonte: Castro (2021)

2.12.3.3 Sensor Lidar

Segundo Espaciais (2022), o LIDAR (Light Detection and Ranging) é um sensor remoto acoplado a plataformas (tripuladas ou não), tendo um laser como fonte própria de luz. O LIDAR emite feixes de luz na frequência do infravermelho próximo (IV) e é capaz de modelar a superfície do terreno bidimensionalmente ou tridimensionalmente.

Atualmente, é um dos sensores mais utilizados em carros autônomos pela rápida resposta, chegando a atender requisitos de resposta em tempo real, e pela alta qualidade nas informações. O funcionamento do sensor é bem parecido com o do sensor ultrassônico visto no item 2.12.3.1, em que a emissão do feixe de luz gera um eco na superfície do objeto retornando uma distância.

Um exemplo de LIDAR tridimensional pode ser visto na figura 28, em que o carro ao mesmo tempo que está navegando pela rota, conhece detectar obstáculos nas três dimensões.

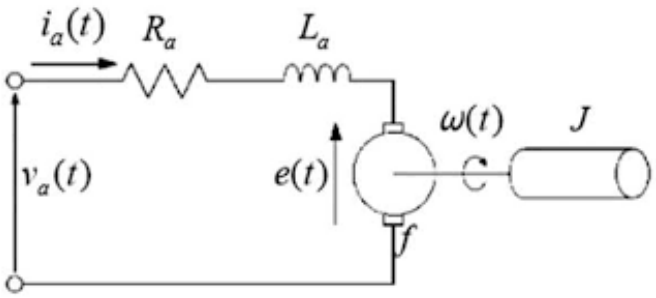
Figura 28: Funcionamento de um Lidar 3D

Fonte: Walford (2019)

2.12.3.4 Motor DC

Como visto em Chapman (1989) e em Castrucci, Bittar e Sales (2011), um motor de corrente contínua (CC) é um dispositivo eletromecânico que transformam uma tensão elétrica em um movimento de rotação. A modelagem matemática de um motor de corrente contínua pode ser visto na figura 29.

Figura 29: Esquema Elétrico de um motor CC



Fonte: Canal, Valdiero e Reimbold (2017)

O procedimento para escolha dos motores de tração de um robô móvel são, segundo Guimarães (2007):

1. Estimar o peso do robô;
2. Estimar os coeficientes de fricção entre as rodas e o solo;
3. Determinar a velocidade, a aceleração e o tamanho e quantidade de roda desejados;
4. Determinar a velocidade angular do motor;
5. Determinar o torque do motor.

Para calcular a velocidade angular do motor pode ser utilizada a equação 2.1.

$$f = \frac{30 * v}{\pi * r} \tag{2.1}$$

Sendo:

- f é a velocidade angular em rpm;

- v é a velocidade linear média do robô em m/s;
- r é o raio da roda do robô em m.

Enquanto que para calcular o torque do motor será utilizada a equação 2.2.

$$\tau = \frac{[(m * a + \mu * m * g) * r]}{n} \tag{2.2}$$

Sendo:

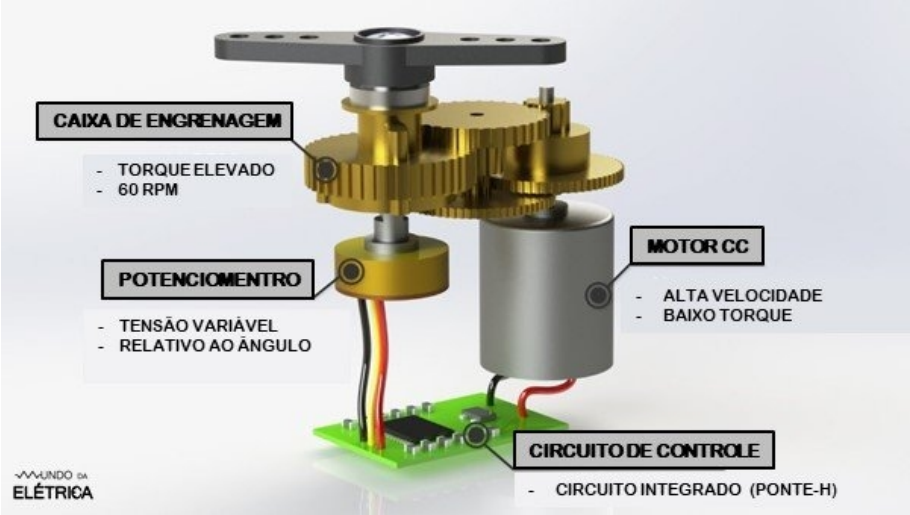
- τ é o torque necessário em N/m;
- a é a aceleração do robô em m/s^2;
- μ é o coeficiente de fricção do solo;
- g é a aceleração gravitacional em m/s^2;
- r é o raio da roda do robô em m;
- n é o número de motores de tração no robô.

2.12.3.5 Servo Motor

Como visto em Kalatec (2022), os servomotores não pertencem a uma classe específica de motores, podendo ser tanto de corrente contínua quanto de corrente alternada. Os servos são projetados para aplicações onde é necessário uma precisão no posicionamento do motor com controle de velocidade.

Seu funcionamento acontece em um sistema de malha fechada, em que o circuito interno do motor manda um sinal de controle sobre a posição atual do motor e ajusta seu movimento para a posição desejada. A estrutura interna de um servo motor pode ser vista na figura 30

Figura 30: Estrutura Interna de um Servo Motor

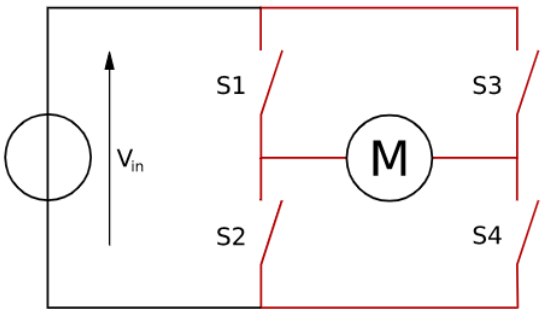


Fonte: Mattede (2022)

2.13 Ponte H

Conforme visto em Almeida (2014), um circuito muito utilizado no acionamento de motores DC é o circuito da ponte H, em que quatro chaves são conectadas em forma de "H"para inverter a polaridade de uma carga sem a necessidade de alterar a polaridade da fonte. O circuito da Ponte H pode ser visto na figura 31.

Figura 31: Circuito da Ponte H



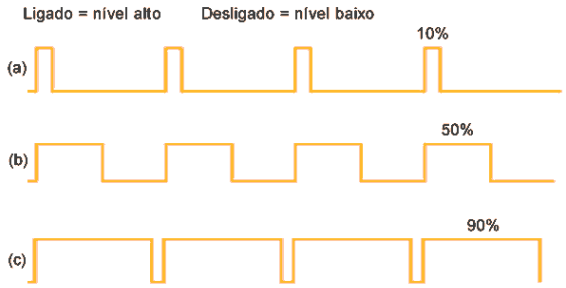
Fonte: Almeida (2014)

Fechando as chaves S1 e S4, tem-se a corrente circulando em um sentido, enquanto que fechando as chaves S2 e S3, o sentido da corrente se inverte, fazendo a carga girar no sentido contrário. Além disso, ao se utilizar o circuito em motores convencionais,

fechando as chaves S1 e S3 ou S2 e S4, provoca-se um curto nos terminais do motor, atuando como freio para motor, impedindo que ele gire em qualquer sentido. Vale ressaltar que não se pode fechar simultaneamente as chaves S1 e S2 ou S2 e S4, pois resultaria em um curto-circuito na fonte de alimentação do circuito.

Como visto em Silveira (2016), para controle da tensão média aplicada ao circuito, utiliza-se uma sinal com *PWM* (Pulse Width Modulation), que consiste em uma técnica em que o sinal de tensão é pulsado rapidamente. A figura 32, mostra três sinais *PWM* com ciclos de trabalho diferentes que resultariam em velocidades diferentes para a carga conectada.

Figura 32: Exemplo de Sinais com *PWM*



Fonte: Silveira (2016)

Nota-se no exemplo da figura 32, que o sinal de tensão funciona apenas em alta ou baixo, fazendo com que qualquer valor analógico possa ser codificado como um sinal *PWM*. A utilização do *PWM* na ponte H, faz com que seja possível controlar a velocidade da carga do circuito, fazendo com que o motor gire de acordo com a frequência de pulso desejada.

3 MATERIAIS E MÉTODOS

Neste capítulo, serão abordados todos os procedimentos para realização do trabalho, assim como os materiais utilizados. Para isso, o capítulo será dividido em seções da seguinte forma:

1. **Levantamento de Requisitos e Projeto Mecânico:** aborda os requisitos necessários para a escolha dos materiais utilizados e cálculos necessários para a elaboração do design mecânico do protótipo.
2. **Impressão das Peças e Montagem do protótipo:** aborda o processo de impressão 3D das peças modeladas e a montagem dos componentes.
3. **Projeto Elétrico e Design da PCB:** apresenta as ligações e esquemas elétricos do protótipo assim como a elaboração da PCB utilizada.
4. **Programação e Controle:** apresenta a lógica de programação utilizada e os principais processos de controle presentes no protótipo.
5. **Algoritmos e Testes de Mapeamento:** apresenta os ajustes feitos nos algoritmos de mapeamento e a metodologia empregada para os testes.

3.1 Levantamento de Requisitos e Projeto Mecânico

Um dos principais requisitos para a elaboração do robô foi que o mesmo conteria quatro rodas que pudessem tracioná-lo e pudessem ter angulações diferentes. Sendo assim, estimando a velocidade média do robô em $v = 1m/s$ e considerando uma roda de raio $r = 68,5mm$, podemos utilizar a equação 2.1 para calcular a velocidade de rotação do motor de tração.

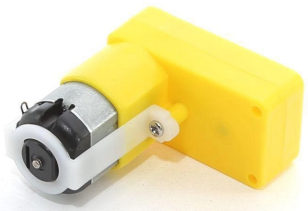
$$f = \frac{30 * 1}{\pi * 0,0685} = 139rpm \tag{3.1}$$

Além disso, utilizando a equação 2.2 e considerando a massa do robô $m = 5kg$, coeficiente de atrito das rodas com o solo $\mu = 0,015$ (borracha sobre piso de tinta epóxi), aceleração da gravidade $g = 9,8m/s^2$ e um total de $n = 4$ rodas tracionárias, temos:

$$\tau = \frac{[(5 * 0,5 + 0,015 * 5 * 9,8) * 0,0685)}{4} = 0,05539N/m \tag{3.2}$$

Com isso, escolheu-se o motor CC da figura 33, muito utilizado para robôs pequenos e que atendia as especificações calculadas. O motor possui uma velocidade de rotação de $f = 150rpm$ e um torque $\tau = 0.08Nm$.

Figura 33: Motor escolhido para o protótipo



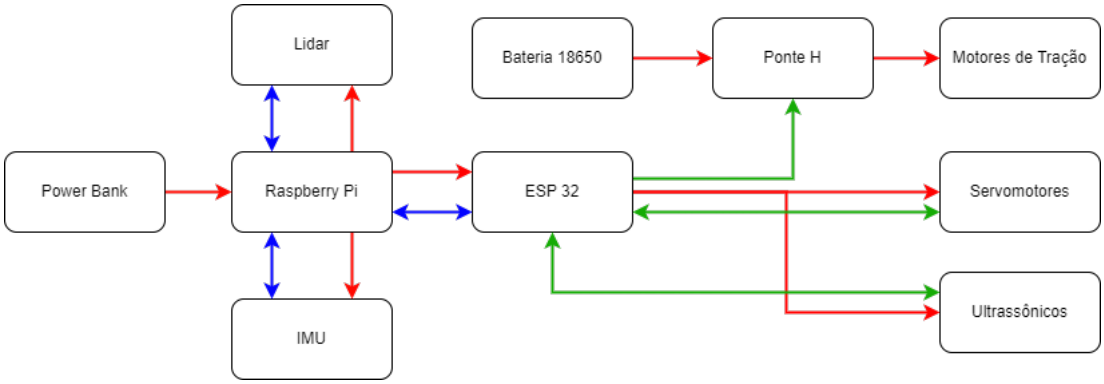
Fonte: Autor

Além disso, optou-se por utilizar quatro servomotores modelo MG-90 para fazer o controle da direção de cada roda, visto que é um servo de pequeno tamanho e com torque suficiente para mudar a direção do robô no solo.

Quanto aos sensores, optou-se por utilizar um LIDAR modelo A1M8 para fazer o mapeamento, um sensor MPU-6050 como IMU para a medição das translações e rotações do robô, dois ultrassônicos HC-SR04 para detecção de obstáculos e uma câmera *Raspicam v2* para monitorar o caminho do robô.

Como unidade de processamento, optou-se por utilizar um *Raspberry Pi 4B* com 4Gb de RAM para que tivesse uma resposta mais rápida quanto ao mapeamento. Já para o acionamento dos motores optou-se por utilizar um *ESP32* devido a grande quantidade de GPIO com PWM além de ter um *hardware* que permite uma resposta mais rápida que outros microcontroladores, o que facilita no acionamento dos motores. A arquitetura do sistema pode ser vista na figura 34.

Figura 34: Arquitetura dos Dispositivos

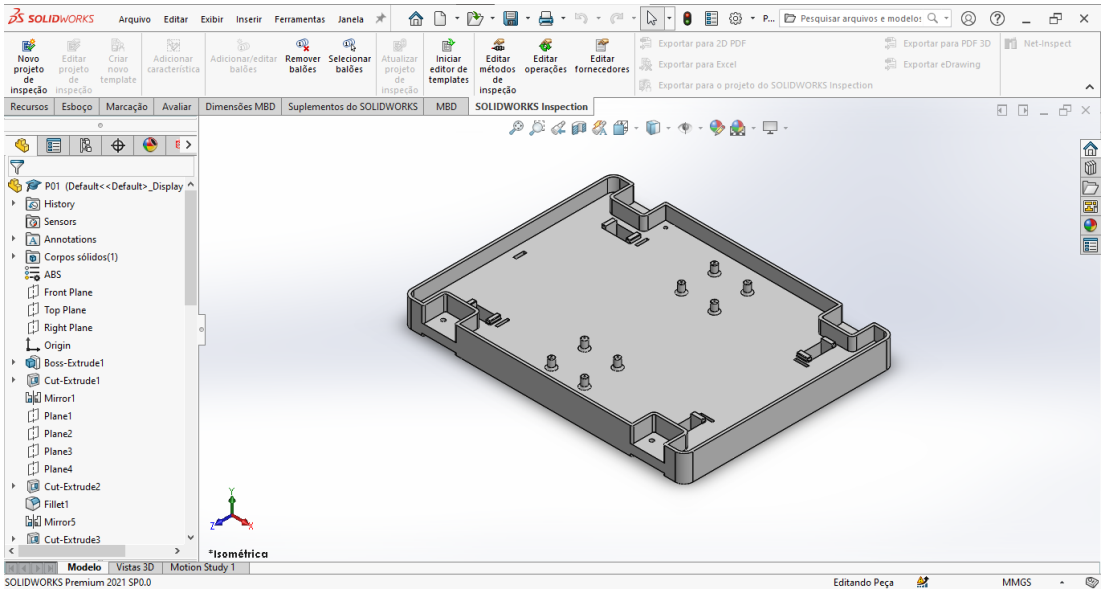


Fonte: Autor

Nota-se na figura 34 que as setas vermelhas indicam a alimentação do dispositivo, as setas verdes indicam comunicação via GPIO e as setas azuis indicam comunicação serial.

Após o levantamento dos dispositivos e suas respectivas comunicações, começou-se a projetar o protótipo utilizando o *software SolidWorks* versão 2021. A *interface* do *software* pode ser vista na figura 35

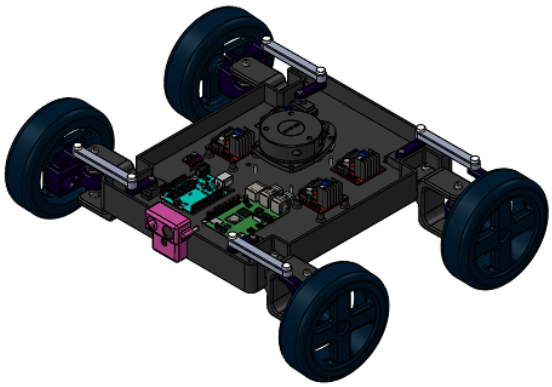
Figura 35: Tela Inicial do *software SolidWorks*



Fonte: Autor

Todas as peças foram projetadas para serem impressas utilizando impressoras 3D no Laboratório de Fabricação Digital, o que será visto com mais detalhes na seção 3.2. Com isso, a primeira versão do protótipo pode ser vista na figura 36.

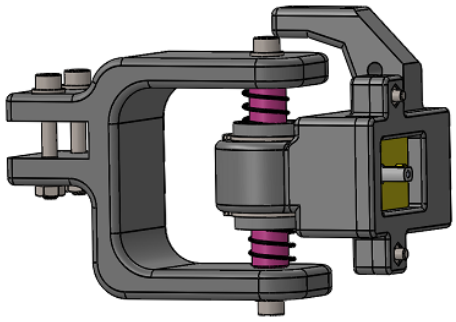
Figura 36: Modelo Inicial do Protótipo



Fonte: Autor

Nota-se na figura 37, que foi projetado um sistema de molas que faz com que seja possível a locomoção em ambientes com terreno irregular. As molas fazem com que o robô consiga se adaptar ao solo de forma a diminuir o desnível entre as rodas, melhorando o desempenho do mesmo.

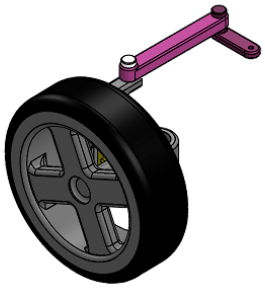
Figura 37: Sistema de Molas Projetado



Fonte: Autor

Para a direção das rodas, foi desenvolvido um mecanismo de dois pequenos "braços", conectados através de pinos (também impressos), que fazem a transmissão de movimento do servomotor para a peça de suporte do motor DC, alterando a direção das rodas conforme a angulação dos servos. O sistema pode ser visto na figura 38.

Figura 38: Sistema de direção projetado



Fonte: Autor

Após a impressão das peças, o robô foi montado conforme o modelo 3D, sendo necessárias algumas correções manuais para o melhor encaixe entre as peças. Para os braços de direção, foi usado o mesmo encaixe do servo motor, colando e parafusando a peça no braço do robô para melhor fixação como visto na figura 39.

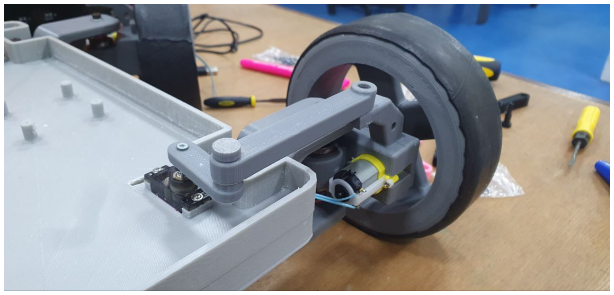
Figura 39: Ajuste do braço do robô

Fonte: Autor

As peças de suporte dos motores de tração foram montadas utilizando parafusos de fixação e encaixando as molas pelo eixo principal. Para um melhor deslocamento das molas e do rolamento utilizados, utilizou-se graxa no eixo de forma a mantê-lo mais liso deixando o movimento mais suave.

O encaixe das rodas foi pensado de acordo com o motor escolhido, sendo assim apenas foi necessário um pequeno ajuste para abrir uma "folga"e manter o encaixe fixo mesmo com o movimento. Como as rodas projetadas eram muito lisas, não conseguindo tracionar o solo devido ao atrito, optou-se por utilizar uma câmara de ar para bicicletas envolvendo as rodas. A câmara, feita com um *blending* de elastômetros, foi colada revestindo as rodas como visto na figura 40.

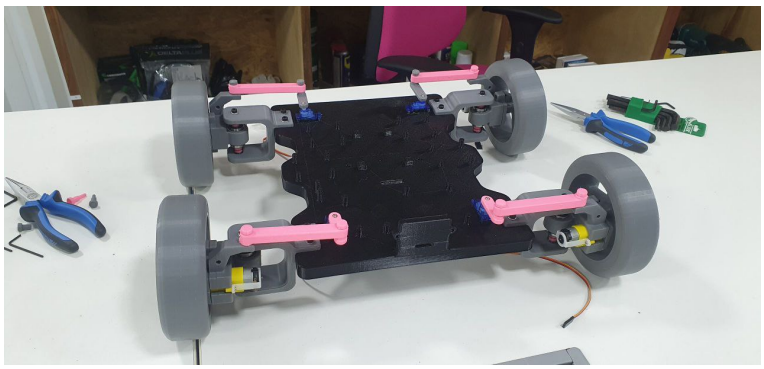
Figura 40: Ajuste das rodas de tração



Fonte: Autor

Com isso, o protótipo foi montado e testado no solo num piso plano e com coeficiente de atrito igual ao calculado (piso pintado com tinta epoxi), o que pode ser visto na figura 41.

Figura 41: Modelo Inicial do Protótipo Montado



Fonte: Autor

No entanto, após a montagem notou-se que devido a grande quantidade de cabos e dispositivos presentes no sistema, seria necessário reprojetar algumas peças para que fosse possível uma melhor manutenção. Além disso, caso o robô fosse "aberto"como projetado, o mesmo teria dificuldade em se locomover por algumas áreas que poderiam por em risco seu funcionamento. Sendo assim, o modelo passou por alterações e o design final será visto no capítulo 4.

3.2 Impressão das peças e Montagem do Protótipo

Como visto na seção 3.1, as peças foram projetadas para serem impressas usando duas impressoras 3D, sendo elas: uma *Sethi 3D S4X* e uma *GTMax Core H5*. As impressoras podem ser vistas nas figuras 42 e 43.

Figura 42: Impressora *Sethi 3D S4X*

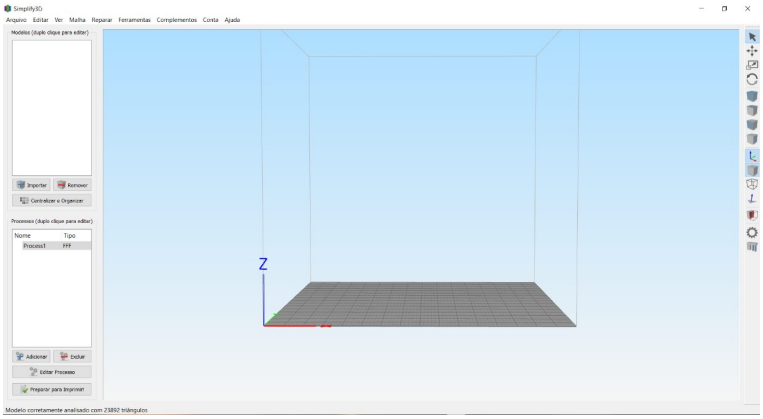
Fonte: Sethi3D (2022)

Figura 43: Impressora *GTMax Core H5*

Fonte: GTMax3D (2022)

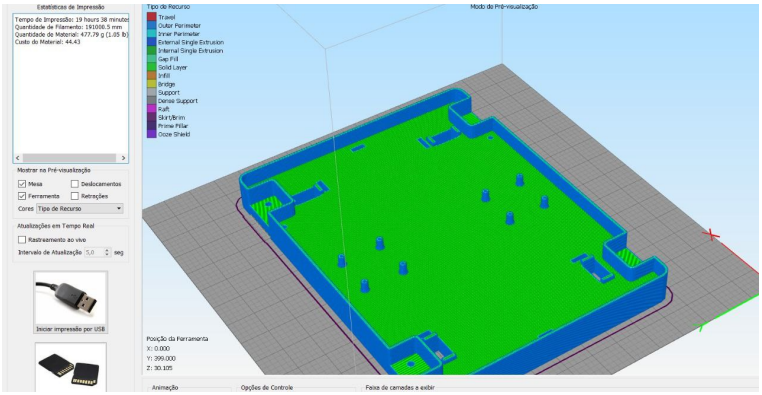
Com isso, utilizou-se o *software Simplify*, cuja interface pode ser vista na figura 44 para fazer o fatiamento de todas as peças que fossem impressas. Na figura 45, pode-se ver o processo de fatiamento de uma das peças do chassi.

Figura 44: Interface do *Simplify*



Fonte: Autor

Figura 45: Base do chassi no *Simplify*



Fonte: Autor

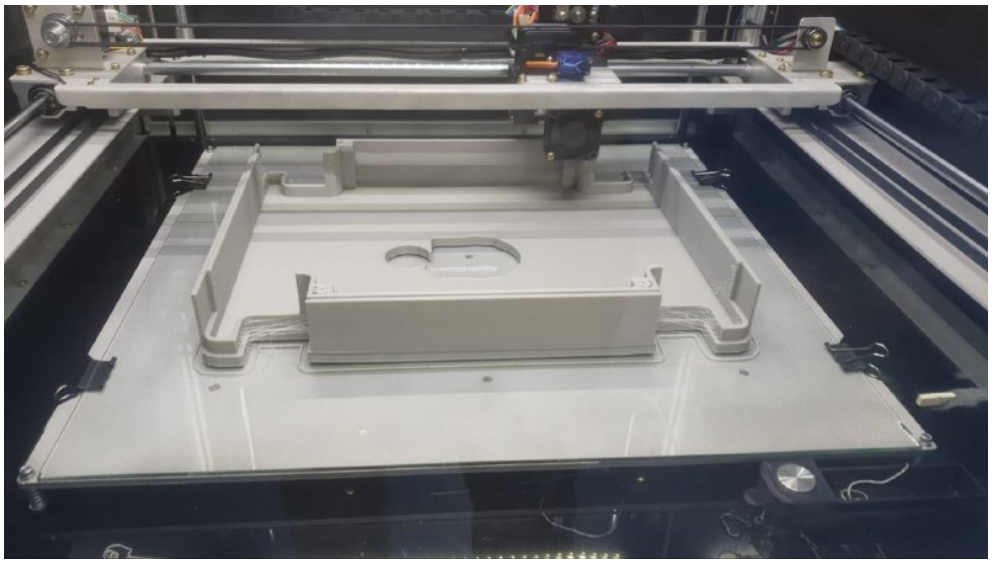
A quantidade de peças impressas e utilizadas no decorrer do projeto pode ser vista na tabela 3. Algumas peças por serem menores, tiveram que ser impressas mais de uma vez com os mesmos parâmetros de impressão para que fossem remontadas. Na figura 46, pode-se ver o topo do chassi sendo impresso na impressora Sethi 3D S4x.

Tabela 3: Quantidade de peças produzidas

Peça	Quant. Peças Produzidas
Tampa do Chassi	1
Base do Chassi	1
Roda	1
Suporte do Motor	4
"Braço"Longo	4
"Braço"Curto	4
Pinos de Fixação	8
Encaixe dos Motores	4
Suporte do sensor Ultrassônico	2
Suporte Câmera	1

Fonte: Autor

Figura 46: Topo do chassi durante processo de impressão



Fonte: Autor

Na tabela 4 encontram-se a quantidade de material por peça impressa, em gramas de filamento, e o tempo de impressão de cada peça, em minutos, e os parâmetros

de impressão das peças pode ser visto na tabela 5. Como a mesa de impressão da impressora Sethi 3D S4X tem uma área grande de impressão (400mmx400mm), foi possível realizar a impressão de diversas peças simultaneamente o que diminuiu o tempo total de trabalho da máquina.

Tabela 4: Quantidade de material e tempo de impressão das peças do robô

Peça	**Quantidade de Material (g)**	**Tempo de Impressão (m)**
Tampa do Chassi	444,92	1145
Base do Chassi	477,79	1178
Roda	146,84	352
Suporte do Motor	26,48	83
"Braço"Longo	6,49	18
"Braço"Curto	2,06	8
Pinos de Fixação	0,57	7
Encaixe dos Motores	34,16	105
Suporte Ultrassônico	10,47	42
Suporte Câmera	14,07	50

Fonte: Autor

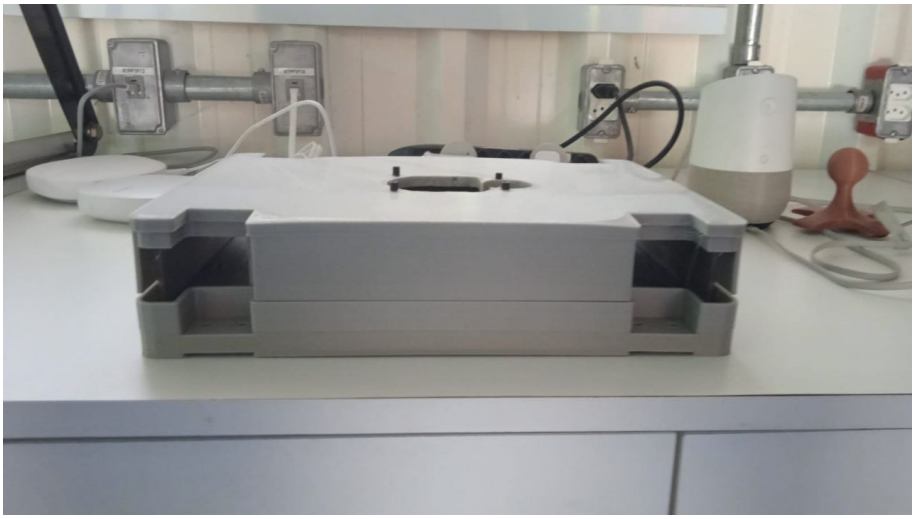
Tabela 5: Parâmetros de impressão das peças

Parâmetro	**Especificação**
Preenchimento	20%
Altura de camada	$0,3mm$
Velocidade de impressão	$80mm/s$
Largura da parede	$0,48mm$
Perímetro	4 paredes

Fonte: Autor

Como as peças foram impressas, foi necessário fazer uma folga entre algumas peças manualmente para evitar que a peça ficasse muito justa ou muito folgada. Com isso, a montagem do chassi pode ser vista na figura 47. A montagem final do robô será vista com maiores detalhes no capítulo 4.

Figura 47: Montagem e encaixe das peças do chassi



Fonte: Autor

3.3 Projeto Elétrico e Design da PCB

Para a parte elétrica do projeto, optou-se por usar duas fontes de alimentação diferentes, sendo um *Power Bank* de celular para a alimentação do *Raspberry Pi* e duas baterias *LiPo* 18650 para a alimentação dos motores. Isso se deu pelo fato de o *Raspberry Pi* necessitar de uma tensão de $5V$ e corrente de $3A$ constantes para funcionamento, tendo que ser projetados circuitos auxiliares para as baterias manterem o funcionamento do sistema.

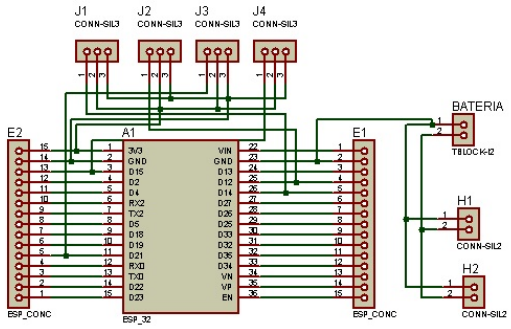
Como o *Raspberry Pi* é responsável pelo processamento de todos os dados do robô, além da alimentação de alguns dispositivos (figura 34), optou-se por utilizar um *Power Bank* de $10000mAh$, o que dá autonomia suficiente para aproximadamente 4 horas de funcionamento.

Com isso, para facilitar a conexão de todos os dispositivos dentro do robô, uma placa de circuito impresso (PCB) foi projetada. Como a alimentação do *Raspberry Pi*, do *ESP32*, do LIDAR e do *Power Bank* são via cabo USB, optou-se por tirar essa alimentação da placa para diminuir a complexidade da mesma. A placa foi projetada como um *shield* de forma que o *ESP32* é encaixado na placa para controlar os dispositivos.

O projeto da placa foi feito utilizando o software *Proteus 8* e o esquemático pode ser visto na figura 48. Nota-se que todos os pinos foram considerados no design da

placa, mesmo aqueles que não seriam utilizados no projeto. Além disso, a placa possui duas entradas de alimentação independentes para facilitar a conexão da bateria com os módulos de ponte H e quatro entradas para conexão dos servomotores.

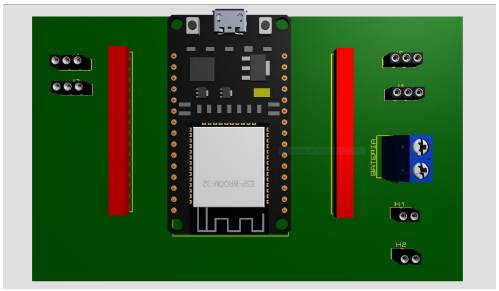
Figura 48: Esquema da placa no *Proteus*



Fonte: Autor

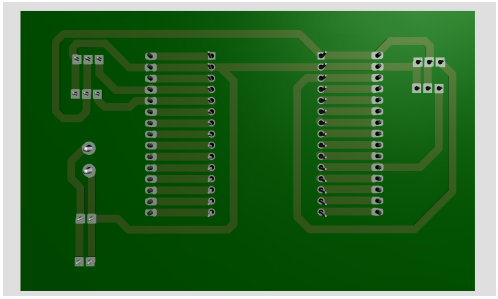
O design 3D da placa e dos componentes no *Proteus* pode ser visto nas figuras 49 e 50. Na figura 50 é possível ver também todas as trilhas e furo da placa.

Figura 49: Face superior da placa no *Proteus*



Fonte: Autor

Figura 50: Face inferior da placa no *Proteus*



Fonte: Autor

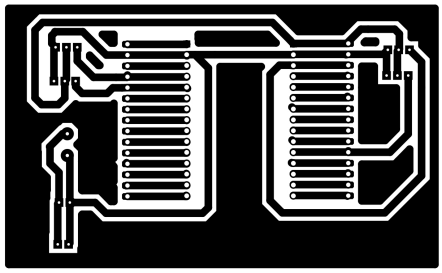
Para a produção da placa utilizou-se uma fresadora 3D de mesa Monofab modelo SRM-20 que pode ser vista na figura 51.

Figura 51: Monofab SRM-20

Fonte: Roland (2022)

Para se trabalhar com a Monofab, primeiro é necessário gerar um arquivo .svg da placa indicando onde serão indicadas as trilhas, os furos e o corte da placa. O arquivo .svg da placa pode ser visto na figura 52

Figura 52: Arquivo .svg da placa



Fonte: Autor

Utilizou-se o software *VCarve* para realizar a simulação do trajeto que a fresadora irá fazer, além de selecionar os parâmetros de broca e velocidade do processo. As brocas escolhidas e o tempo de cada processo podem ser vistos na tabela 6.

Tabela 6: Especificações das brocas utilizadas

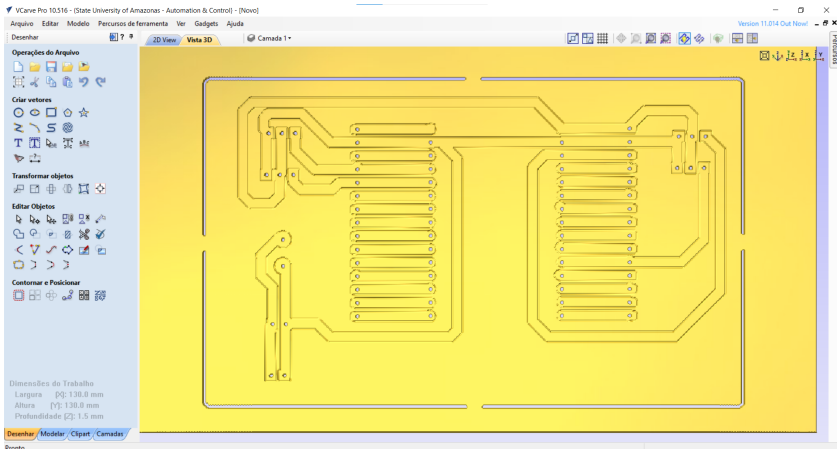
Processo	**Tipo de Broca**	**Tempo de Processo (m)**
Trilhas	fresa de gravação de 0, 1*mm*	159
Furos	fresa de topo raso de 1*mm*	8
Corte	fresa de topo raso invertido de 2*mm*	51

Fonte: Autor

Uma simulação de como será o resultado final da placa após os três processos pode

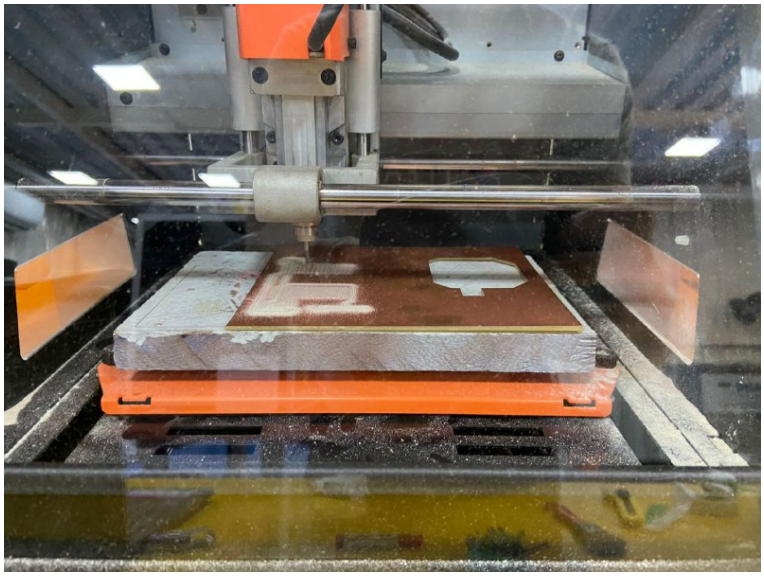
ser vista na figura 53. Na figura 54, é possível ver a placa no processo de fresagem das trilhas.

Figura 53: Simulação da placa no software *Vcarve*



Fonte: Autor

Figura 54: Processo de fresagem das trilhas



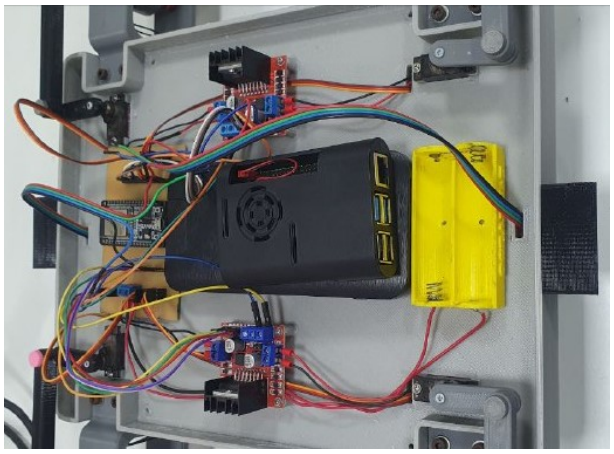
Fonte: Autor

Por fim, passou-se uma máscara de tinta própria para PCBs a fim de evitar corrosão e oxidação das placas. A placa montada é mostrada no Capítulo 4, com todos os detalhes.

As conexões foram feitas utilizando *jumpers* comuns. Para os motores DC e os sensores ultrassônicos que precisavam de cabos mais extensos, devido a distância das conexões, utilizou-se cabos comuns soldados diretamente nos dispositivos. Para os

servomotores, optou-se por utilizar os cabos já soldados no componente. As conexões podem ser vistas na figura 55.

Figura 55: Cabos e conexões do robô



Fonte: Autor

Nota-se que ao fechar o chassi do robô, o próprio desenho das peças faz com que os cabos se organizem dentro do chassi, impedindo que ocorra desconexão entre os dispositivos. Além disso, para os cabos de alimentação foram utilizados conectores com borne parafuso para que não ocorra nenhuma desconexão quando em movimento, mantendo o robô seguro.

3.4 Programação e Controle

Para a programação do robô, utilizou-se o *ROS Noetic Ninjemys*, por ser a distribuição mais atual do *framework*. Além disso, para maior compatibilidade com o *ROS* e outros sistemas embarcados, optou-se por utilizar o Ubuntu Mate instalado no *Raspberry Pi*.

A instalação e configuração do SO e do *ROS*, foi feita seguindo os tutoriais dos próprios websites da Canonical e da Open Robotics, respectivamente. Como o *ROS*trabalha utilizando *pacotes* que contém os códigos e instruções necessários para o funcionamento do robô, criou-se um pacote chamado *tcc* para o desenvolvimento deste projeto.

Com isso, os pacotes utilizados no desenvolvimento da programação são:

- **tcc:** pacote contendo todos os códigos de movimentação autônoma do robô e

configurações de mensagens e tópicos enviados.

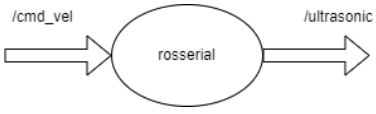
- **tcc_robot_description:** pacote contendo a descrição do robô em *URDF*, com todas as transformações lineares para a cinemática direta do robô e posicionamento dos sensores para cinemática inversa.
- **rplidar_ros:** pacote público disponibilizado pela empresa Slamtec contendo todas as configurações e códigos necessários para se utilizar o LIDAR A1M8.
- **rosserial_python:** pacote público disponibilizado pela Open Robotics para se utilizar dispositivos, tais como o Arduino e o ESP32 via serial com o *ROS*.
- **mpu_6050_driver:** pacote público disponibilizado pela Open Robotics para se trabalhar com o sensor MPU6050 contendo filtros para estabilizar os dados do sensor.
- **hector_slam:** pacote público disponibilizado pela Open Robotics contendo todos os dados necessários para realizar o processo de *SLAM* utilizando o método Hector SLAM, que utiliza o próprio LIDAR como fonte de odometria para o mapeamento como visto em Kohlbrecher et al. (2011).

Uma das principais vantagens em se utilizar o *ROS*no desenvolvimento deste projeto é a fácil integração dos dispositivos utilizados. Sendo assim, o foco maior no desenvolvimento de algoritmos ficou na programação do *ESP32*, programação da navegação autônoma, configuração do *URDF* e configuração dos outros pacotes.

Para a programação do *ESP32*, utilizou-se a Arduino IDE por tornar a programação mais simples, além de já conter as bibliotecas necessárias para um bom funcionamento dos sensores e motores.

A linguagem de programação utilizada para a programação do *ESP32* foi a própria do Arduino, que é baseada em C. O algoritmo foi programado de forma a ser um nó em *ROS*que vai funcionar conforme figura 56. O nó vai ser responsável pela movimentação do robô em todos os sentidos e pela detecção de obstáculos através dos sensores ultrassônicos.

Figura 56: Funcionamento do Algoritmo do *ESP32*



Fonte: Autor

Nota-se que apenas dois tópicos serão utilizados: sendo o tópico "/cmd_vel"um tópico comumente utilizado que contém informações de movimentação de robôs nos três eixos. O tópico "/ultrasonic"é um tópico criado para este projeto que contém as informações tanto do sensor frontal quanto do sensor traseiro para detecção de obstáculos. Optou-se por utilizar apenas um tópico para os dois sensores para aprimorar o processamento dos dados e diminuir a complexidade geral do sistema. O trecho de código que declara todos os tópicos pode ser visto na figura 57.

Figura 57: Declaração dos Tópicos e Bibliotecas utilizados

```
#include <ros.h>
#include <ESP32Servo.h>
#include <NewPing.h>
#include <tcc/Ultrasonic.h>
#include <geometry_msgs/Twist.h>
```

Fonte: Autor

Para que o robô possa ter o controle em cada roda, os servos foram "zerados"no ângulo de 90° visto que assim pode girar até 0° ou 180° conforme a necessidade. O trecho de código que mostra a função de angulação dos braços do robô pode ser visto na figura 58.

Figura 58: Código para controle dos servomotores

```
void angleServo(int angle){
  left_front_arm.write(90+angle);
  left_back_arm.write(90+angle);
  right_front_arm.write(90+angle);
  right_back_arm.write(90+angle);
}
```

Fonte: Autor

O código para movimentação dos motores faz uso do tópico "/cmd_vel"que possui informações para translação e rotação nos três eixos. Como estamos lidando com um robô terrestre, as rotações acontecem apenas no eixo z. Sendo assim, quando o robô recebe uma velocidade linear em x, o mesmo se movimenta para frente ou para trás conforme o sinal enviado. Se sua velocidade linear for diferente de 0 e receber uma velocidade angular, o robô irá acionar os servos para fazer uma curva na angulação desejada. Se sua velocidade linear for igual a 0, o robô gira no próprio eixo. O trecho de código que define o comportamento do robô pode ser visto na figura 59.

Figura 59: Trecho de código de acionamento dos motores

```
void moveRobot(const geometry_msgs::Twist& velocity_msg){
  int speed = round(velocity_msg.linear.x);
  int angulo = round(velocity_msg.angular.z);

  if(speed<0){
    backward();
    angleServo(angulo);
    velocityMotor(speed);
  }
  else if(speed>0){
    forward();
    angleServo(angulo);
    velocityMotor(speed);
  }
  else{
    if(angulo<0){
      rotateLeft();
      velocityMotor(speed);
    }
    else if(angulo>0){
      rotateRight();
      velocityMotor(speed);
    }
    else{
      brake();
    }
  }
}
```

Fonte: Autor

Para a declaração dos tópicos utilizados no *ESP32*, apenas é necessário declarar quais as informações transmitidas pelo *publisher* e quais informações o *subscriber* deve receber. O trecho desse código pode ser visto na figura 60.

Figura 60: Declaração do *publisher* e *subscriber*

```
ros::NodeHandle  nh;

tcc::Ultrasonic ultrasonic_msg;

ros::Publisher pub_distance("/ultrasound", &ultrasonic_msg);
ros::Subscriber<geometry_msgs::Twist> sub_motors("cmd_vel", moveRobot);
```

Fonte: Autor

Para o sensor ultrassônico funcionar corretamente, é necessário que o mesmo esteja em loop, coletando as informações de distância a todo momento. Sendo assim, o trecho de código para esse *loop* pode ser visto na figura 61.

Figura 61: Código para funcionamento do sensor ultrassônico

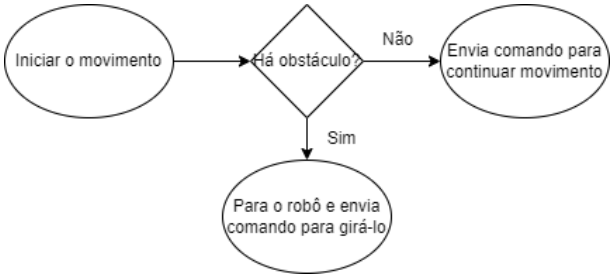
```
void loop(){
  unsigned long currentMillis = millis();

 if (currentMillis>=range_timer + intervalR)
 {
   range_timer = currentMillis+intervalR;

   ultrasonic_msg.front_distance = front_returnDistance();
   ultrasonic_msg.back_distance = back_returnDistance();
   pub_distance.publish(&ultrasonic_msg);
   }
 nh.spinOnce();
 delay(1);
}
```

Fonte: Autor

O código completo utilizado no *ESP32*poderá ser encontrado no apêndice B, contendo as informações de pinos utilizados e funcionamento de cada pino.

O algoritmo de movimentação autônoma (desenvolvido no pacote *tcc*), recebe as informações do tópico "/ultrasonic"e faz os cálculos necessários para enviar os comandos de movimentação ao tópico "/cmd_vel". Esse algoritmo, foi desenvolvido em *Python 3* e a lógica de programação pode ser vista de forma simplificada na figura 62.

Figura 62: Funcionamento do Algoritmo de Movimentação Autônoma



Fonte: Autor

Na figura 63, pode-se ver um trecho do código em *python* em que são declarados os tópicos e mensagens utilizados naquele código, além da função para o sensor ultrassônico criada.

Figura 63: Trecho do código em *python* de movimentação autônoma

auto.py

tcc > scripts > auto.py

```python
#!/usr/bin/env python3
import rospy
from sensor_msgs.msg import Range
from geometry_msgs.msg import Twist
from tcc.msg import Ultrasonic
import math
import time

def sonar_callback(sonar_data):
    global front_distance
    global back_distance
    front_distance = sonar_data.front_distance
    back_distance = sonar_data.back_distance
    auto_move()
```

Fonte: Autor

Para a movimentação do robô, é usada a função *auto_move* que declara as condições para o robô desviar dos obstáculos e controlar os motores ligados no *ESP32*, conforme a figura 64.

Figura 64: Algoritmo de Movimentação Autônoma

tcc > scripts > auto.py

```python
def auto_move():
    velocity_publisher = rospy.Publisher("/cmd_vel", Twist, queue_size=10)

    forward = Twist()
    backward = Twist()
    rotate_right = Twist()
    rotate_left = Twist()
    stop = Twist()

    forward.linear.x=50
    backward.linear.x=50
    rotate_right.angular.z=50
    rotate_left.angular.z=-50
    stop.linear.x = 0

    loop_rate = rospy.Rate(10)
    if(front_distance >= 0 and front_distance <=20):
        velocity_publisher.publish(stop)
        rospy.sleep(0.5)
        velocity_publisher.publish(backward)
        rospy.sleep(0.5)
        if(random(2) == 0):
            velocity_publisher.publish(rotate_left)
        else:
            velocity_publisher.publish(rotate_right)
        loop_rate.sleep()
    else:
        velocity_publisher.publish(forward)
        loop_rate.sleep()
```

Fonte: Autor

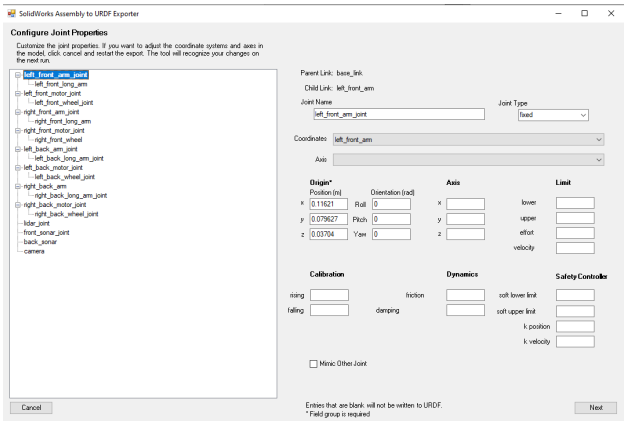
O código completo será visto no apêndice B, contendo todas as informações necessárias para funcionamento do robô de maneira autônoma e conexão com o ROS.

O pacote *tcc_robot_description* foi gerado por um *plugin* no *SolidWorks* disponibilizado pela Open Robotics, que permite gerar o *URDF*de um robô projetado no *software*. Utilizando o *plugin*, é possível definir as peças que serão os elos do robô e definir como cada junta do mesmo atuará, além de todas as posições no espaço. No

entanto, os projetos gerados no *SolidWorks* tem bastante precisão no encaixe de peças, o que faz com que sejam necessários ajustes no código gerado.

A interface do plugin pode ser vista na figura 65. A partir do *plugin*, utilizou-se apenas as informações de posição das juntas e elos, além de informações de massa e inércia de cada peça. Para a movimentação das juntas, foram feitos ajustes manuais no código do *URDF* para um melhor processamento do *plugin*.

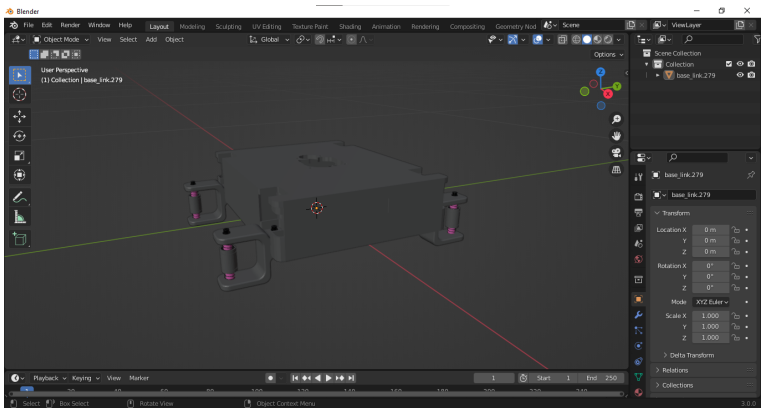
Figura 65: Interface do *plugin SW to URDF*



Fonte: Autor

No pacote gerado, as peças do robô na simulação são geradas na extensão *.stl*, que é um formato de arquivo mais simples que armazena dados de forma, mas não armazena informações de cores. Sendo assim, para ter uma simulação mais real, utilizou-se o *software Blender* para transformar as peças em arquivo *.dae*. A interface do *Blender* com o chassi do robô pode ser vista na figura 66.

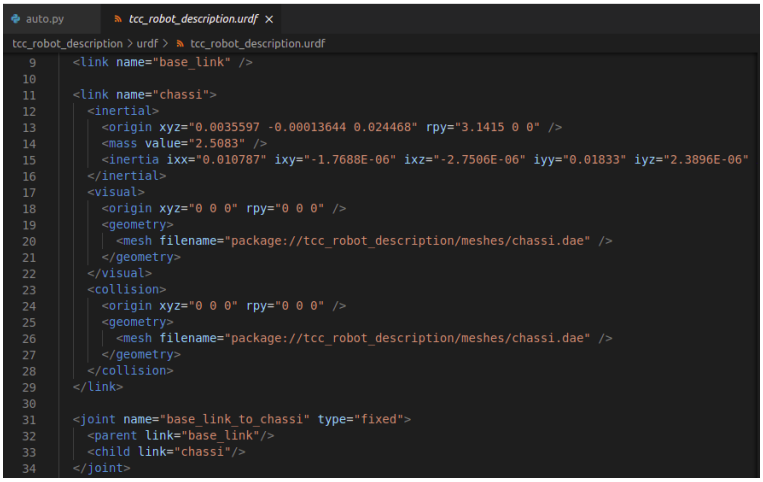
Figura 66: Interface do *Blender* com o chassi do robô



Fonte: Autor

Os outros pacotes utilizados usam informações referentes ao pacote **tcc_robot_description** para localização dos sensores em relação ao robô e principalmente para localização do robô em relação ao mapa. Um trecho do *URDF* do robô pode ser visto na figura 67 e o código inteiro pode ser visto no apêndice B.

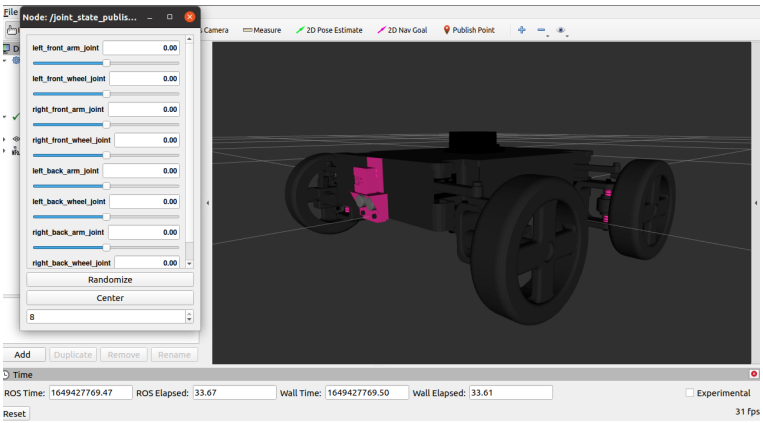
Figura 67: Declaração do chassi no código do *URDF*



Fonte: Autor

Com os arquivos *.dae* é possível simular o robô usando o *software Rviz*, uma interface gráfica do *ROS* para visualização de dados de diversos formatos. Ao se declarar um modelo de robô, pode-se importar um modelo *URDF* como na figura 68, que mostra o modelo inteiro do robô.

Figura 68: Modelo *URDF* no *Rviz*



Fonte: Autor

3.5 Algoritmos e Testes de Mapeamento

Para o mapeamento utilizou-se os pacotes **rplidar_ros**, **mpu_6050_driver** e **hector_slam**, vistos na seção anterior. Como os pacotes são feitos pelas respectivas empresas para um uso geral dos sensores, não foi necessário programar novos algoritmos, modificando apenas configurações de cada pacote.

O pacote **rplidar_ros**, possui todos os algoritmos necessários para utilizar o sensor RPLidar A1M8, o que faz com que sua comunicação com o *Raspberry Pi* seja mais simples. Para utilização do pacote, o mesmo apenas foi copiado para o *workspace* do *ROS*, fazendo com que fosse compilado ao mesmo tempo que outros pacotes.

Já o pacote **mpu_6050_driver**, precisou de diversas modificações, visto que o mesmo não detectava o sinal do sensor MPU-6050 inicialmente. Para isso, redefiniu-se os registradores utilizados na comunicação *I2C* e acrescentando um *queue_size* para os *publishers* do algoritmo, fazendo com que não sejam perdidas as informações.

Com isso, o sensor IMU serviu como uma realimentação ao algoritmo de mapeamento, visto que o mesmo detecta as acelerações e rotações do robô no espaço, sendo mais preciso que o Lidar para detecção das posições do robô mantendo o mapeamento em malha fechada.

Para o mapeamento, o pacote **hector_slam** foi escolhido visto que o mesmo não necessita da odometria das rodas para um funcionamento eficiente. Sendo assim, modificou-se no código apenas a localização do Lidar para que o robô pudesse entender a localização do mesmo, o trecho do código pode ser visto na figura 69.

Figura 69: Argumentos do Laserscan no código

```
  <node pkg="tf" type="static_transform_publisher"
name="base_to_laser_broadcaster" args="0.0012639 0.0077901
0.08975 0 0 0 base_link laser 100"/>
</launch>
```

Fonte: Autor

Vale ressaltar também que alguns parâmetros de mapeamento tem que ser declarados no código, sendo eles: "base_frame"deve ser declarado como "base_link"que é a base do robô no *URDF*, "odom_frame"deve ser declarado como "imu_link"que é a posição relativa ao IMU, o "scan_topic"deve ser declarado como "scan"visto que é o tópico gerado pelo pacote **rplidar_ros** e o "map_size"deve ser modificado de acordo

com a qualidade do mapa. Foi escolhido um mapa de 2048x2048 pixels para dar uma alta qualidade ao mapa como visto na figura 70.

Figura 70: Parâmetros de mapeamento

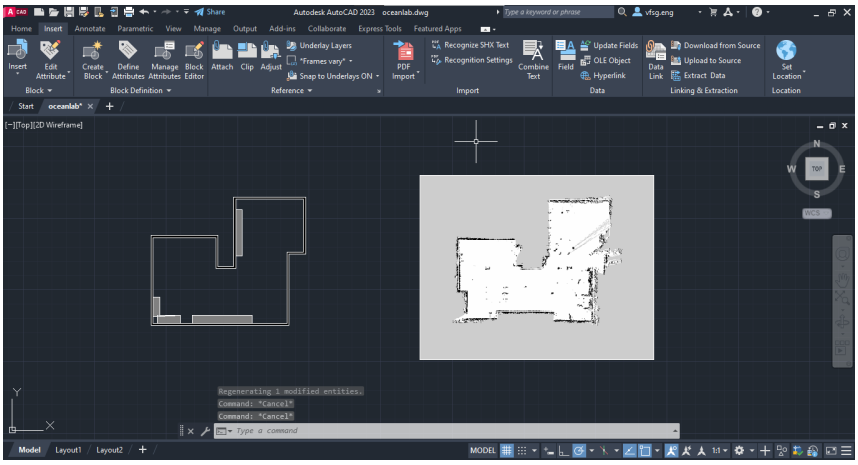
```
<launch>
  <arg name="tf_map_scanmatch_transform_frame_name" default="scanmatcher_frame"/>
  <arg name="base_frame" default="base_link"/>
  <arg name="odom_frame" default="imu_link"/>
  <arg name="pub_map_odom_transform" default="true"/>
  <arg name="scan_subscriber_queue_size" default="5"/>
  <arg name="scan_topic" default="scan"/>
  <arg name="map_size" default="2048"/>
```

Fonte: Autor

Ao final do processo de mapeamento, para comparação dos resultados foi feita a planta baixa dos lugares mapeados a fim de detectar possíveis falhas no processo de mapeamento utilizando o *software AutoCAD 2023*.

Para identificar a precisão do processo de mapeamento, o mapa gerado foi anexado ao modelo da planta baixa feita no *AutoCAD* e foi refeita a planta de forma a seguir as coordenadas geradas pelo mapeamento. A partir da nova planta baixa gerada, calculou-se a área mapeada comparando com a área da planta baixa original do local, conforme figura 71.

Figura 71: Comparação do mapa gerado com a planta baixa do local



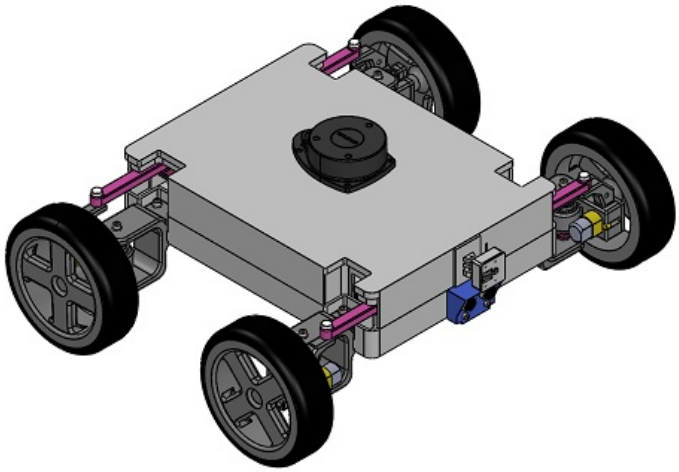
Fonte: Autor

Além disso, calculou-se também o tempo de mapeamento em cada local, de forma a encontrar um valor quantitativo para a qualidade do mapa em relação ao tempo de mapeamento. Os testes realizados e os resultados após o processo de mapeamento estão descritos em detalhes no capítulo 4.

4 RESULTADOS E DISCUSSÃO

Como visto no capítulo 3, o modelo passou por diversas alterações até chegar ao design final para "fechar"o chassi e reorganizar os componentes dentro da estrutura. O modelo final pode ser visto na figura 72. O desenho de cada uma das peças pode ser visto no apêndice A.

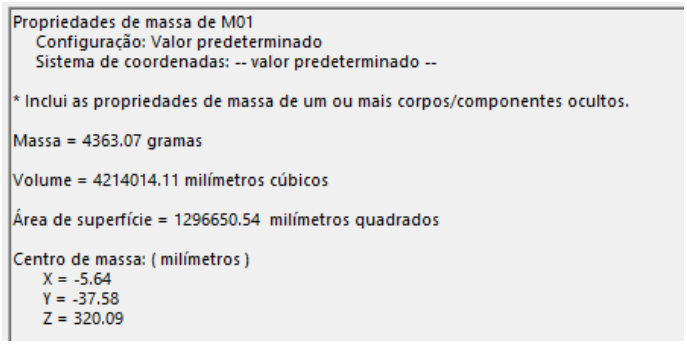
Figura 72: Projeto do Modelo Final



Fonte: Autor

Esse modelo apresenta um chassi mais robusto em ABS e por ser fechado consegue dar uma resistência maior ao protótipo. Algumas propriedades do modelo final podem ser vistas nas figuras 73 e 74. Nota-se a partir dos relatórios gerados pelo *Solidworks* que a maassa ficou do protótipo ficou próxima ao valor estimado para cálculo dos motores de tração. Essa diferença se dá devido a falta da fiação no desenho e pelo uso de peças com impressão 3D, em que a impressora cria uma malha interna dentro das peças.

Figura 73: Propriedades de massa do modelo final



Propriedades de massa de M01
Configuração: Valor predeterminado
Sistema de coordenadas: -- valor predeterminado --

* Inclui as propriedades de massa de um ou mais corpos/componentes ocultos.

Massa = 4363.07 gramas

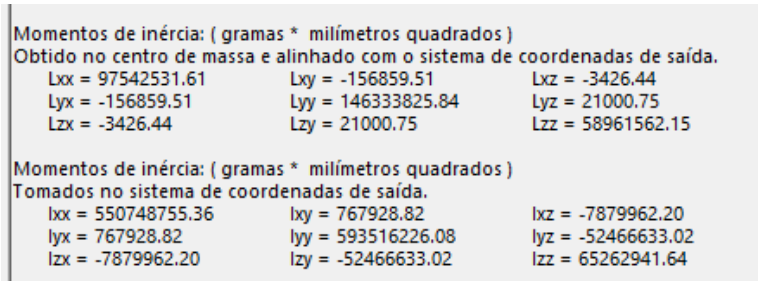
Volume = 4214014.11 milímetros cúbicos

Área de superfície = 1296650.54 milímetros quadrados

Centro de massa: (milímetros)
X = -5.64
Y = -37.58
Z = 320.09

Fonte: Autor

Figura 74: Momento de Inércia do Modelo Final



Momentos de inércia: (gramas * milímetros quadrados)
Obtido no centro de massa e alinhado com o sistema de coordenadas de saída.

Lxx = 97542531.61	Lxy = -156859.51	Lxz = -3426.44
Lyx = -156859.51	Lyy = 146333825.84	Lyz = 21000.75
Lzx = -3426.44	Lzy = 21000.75	Lzz = 58961562.15

Momentos de inércia: (gramas * milímetros quadrados)
Tomados no sistema de coordenadas de saída.

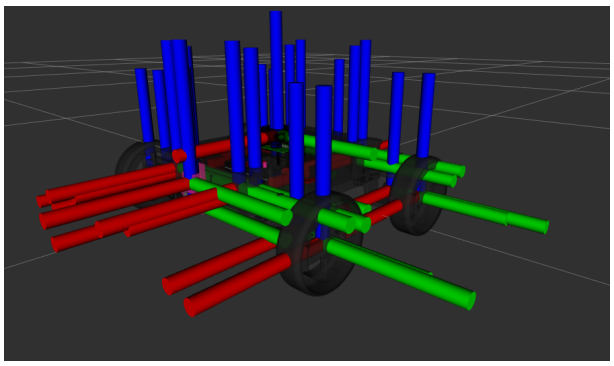
Ixx = 550748755.36	Ixy = 767928.82	Ixz = -7879962.20
Iyx = 767928.82	Iyy = 593516226.08	Iyz = -52466633.02
Izx = -7879962.20	Izy = -52466633.02	Izz = 65262941.64

Fonte: Autor

A partir dos dados apresentados no relatório, pode-se afirmar que o protótipo atenderá as especificações de locomoção em ambientes industriais e foi projetado com motores de tração bem dimensionados para essa atividade.

A partir do modelo gerado, foi possível obter um modelo cinemático do mesmo utilizando o *URDF* do robô, que pode ser visto na figura 75 que mostra o modelo do robô no software Rviz com todos os eixos referentes a localização de cada junta.

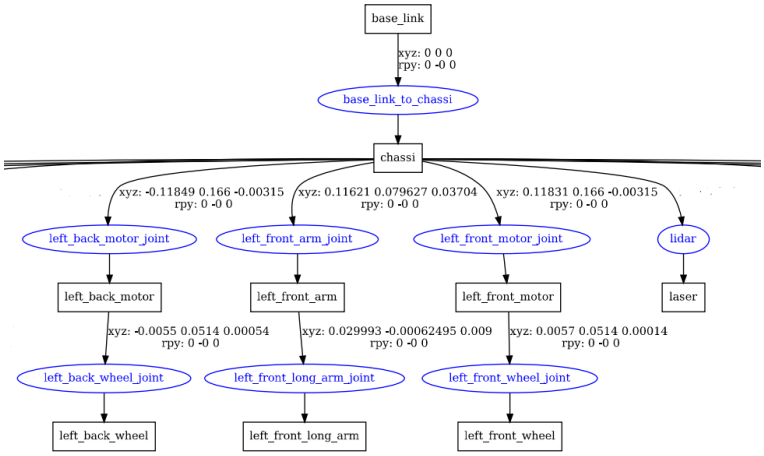
Figura 75: Localização dos elos e juntas no *URDF*



Fonte: Autor

O modelo gerado pode ser visto através de suas transformações lineares na figura 76, que mostra uma parte dos elos e juntas do robô de maneira gráfica. Nota-se que algumas juntas foram criadas apenas para efeitos de simulação, o que é o caso das quatro juntas "motor", devido a uma limitação do próprio *URDF*, o que não interfere em nada ao funcionamento do sistema real.

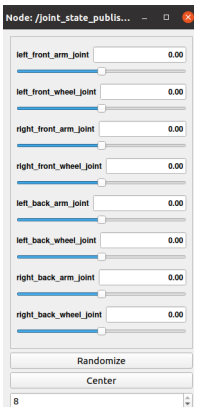
Figura 76: Relatório das posições no *URDF*



Fonte: Autor

É possível ainda controlar o modelo apenas usando uma GUI (Interface Gráfica do Usuário) como visto na figura 77. Vale ressaltar também que o modelo *URDF*foi utilizado apenas como fins de simulação, localização das juntas e ajuste de transformações para os pacotes de mapeamento e navegação, não interferindo diretamente na dinâmica do robô. Com isso, o modelo atendeu as necessidades do projeto sendo possível ajustar todos os parâmetros de mapeamento a partir deste.

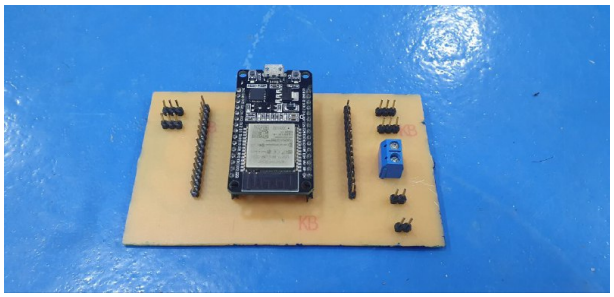
Figura 77: GUI de controle do *URDF*



Fonte: Autor

Para finalizar a montagem, a placa projetada passou por diversos testes de condutividade antes de ser soldada para avaliar possíveis problemas de fresagem. A placa finalizada e montada pode ser vista na figura 78.

Figura 78: Placa Finalizada e montada

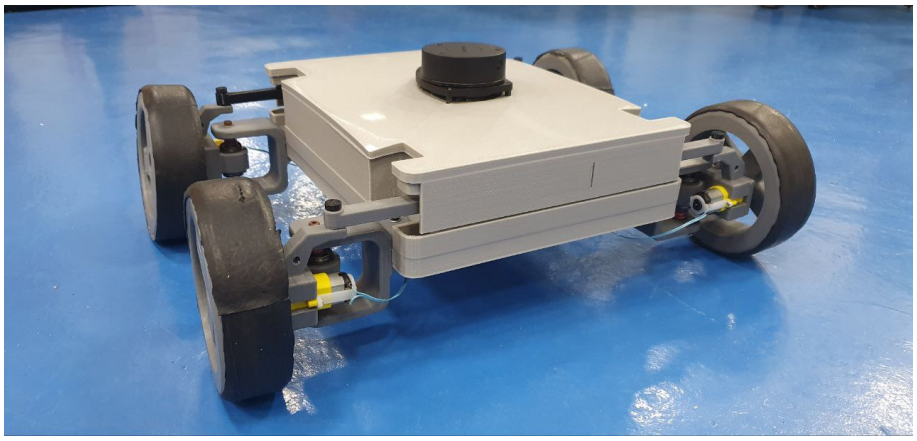


Fonte: Autor

Alguns dos conectores da placa tiveram que ser soldados novamente por desconectarem depois de um certo tempo de uso. Como o projeto foi pensado para se usar *jumpers* comuns de prototipagem, a placa atendeu as expectativas mantendo um bom funcionamento em todos os testes feitos. Para fixação da placa no robô, foi usada apenas fita adesiva, que fez uma boa aderência e mantendo a placa fixa mesmo com o robô em movimento.

Com isso, o protótipo foi montado e testado no solo num piso plano e com coeficiente de atrito igual ao calculado (piso pintado com tinta epóxi), o que pode ser visto na figura 79. Pode-se notar que o robô atendeu aos requisitos de projeto ao se locomover sem dificuldades pelo piso. Além disso, todos os cabos utilizados apresentaram um bom resultado, não ocorrendo nenhum tipo de desconexão durante os testes.

Figura 79: Montagem final do robô



Fonte: Autor

Os testes de mapeamento foram feitos em três ambientes com características diferentes, sendo detalhados posteriormente, que são:

1. **Ambiente 1: Local com área menor que 30m^2 sem obstáculos**
2. **Ambiente 2: Local com área maior que 30m^2 com poucos obstáculos**
3. **Ambiente 3: Local com área maior que 50m^2 com maior quantidade obstáculos**

4.1 Ambiente 1: Loja de Material Esportivo

Como local com área menor que 30m^2 sem obstáculos foi selecionada uma loja de material esportivo, situada na Av. Efigênio Salles, s/n, em Manaus. O local pode ser visto na figura 80.

Figura 80: Loja de Material Esportivo usada pra testes

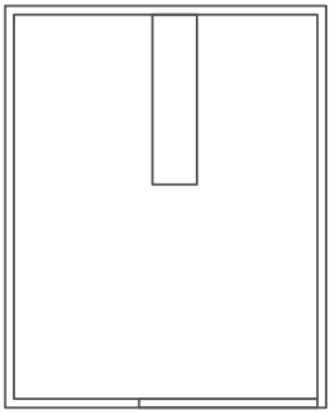
Fonte: Autor

Como o local não tinha tantos obstáculos para a navegação do robô, a loja foi selecionada para ser o primeiro local de testes. Para conexão do robô, foi utilizado o *wi-fi* conectando um notebook ao *Raspberry Pi* via *ssh* para enviar os comandos de iniciar o mapeamento e finalizar o mapeamento. A navegação apenas deve ser iniciada e ao fim do processo, o robô para e volta a sua posição inicial.

A planta baixa da loja feita no *AutoCAD* pode ser vista na figura 81 e o mapa gerado, na figura 82. Nota-se que o robô não conseguiu identificar com acurácia o que

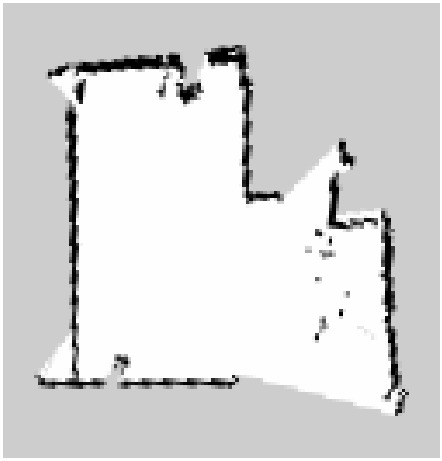
estava por trás do balcão (parte superior direita da figura), isso aconteceu porque era o local onde estavam sendo rodados os testes, o que fez com que o robô não conseguisse entrar.

Figura 81: Planta Baixa da loja



Fonte: Autor

Figura 82: Mapa da loja gerado pelo robô



Fonte: Autor

Outro ponto importante de se notar é a angulação na "parede"no canto inferior direito da figura 82 que ocorreu devido a porta da loja ser de vidro (figura 83), o que causa uma certa refração no sinal infravermelho do Lidar, gerando um erro no sensor.

Figura 83: Porta de vidro mencionada

Fonte: Autor

Os pixels em branco são considerados pixels que estão livres e não possuem obstá-

culos, os pixels em preto são as paredes e obstáculos, enquanto que os pixels em cinza são pixels cuja informação é desconhecida. O mapa gerado ficou bem semelhante à planta baixa da loja. Após a comparação das áreas entre as duas imagens, obteve o resultado que pode ser visto na tabela 7.

Tabela 7: Área calculada pelo *AutoCAD* da Loja mapeada

Imagem	**Área (m^2)**
Planta Baixa projetada	$11,105$
Mapa gerado pelo robô	$10,9897$

Fonte: Autor

Com isso, pode-se perceber que o mapa gerado possui $98,96\%$ de precisão em relação a planta baixa do local mapeado o que representa um resultado satisfatório para esse tipo de local, visto que o ambiente é pequeno e sem muitos obstáculos.

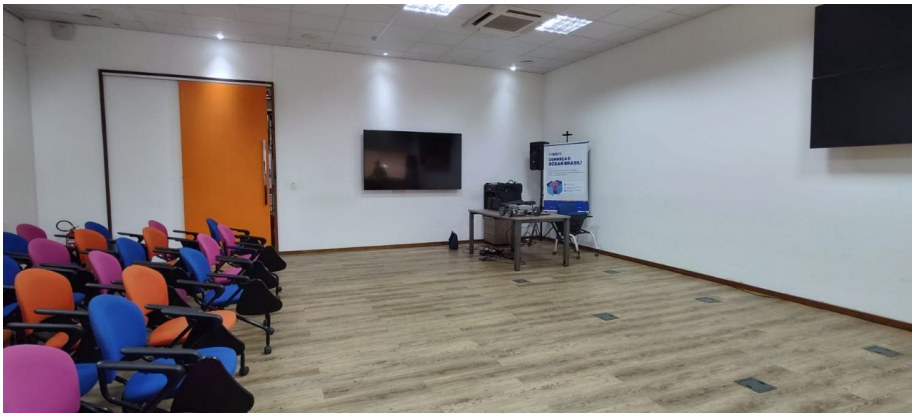
4.2 Ambiente 2: Auditório do Ocean

Para os testes em local com área maior que $30m^2$ com poucos obstáculos foi selecionado o auditório do Ocean Manaus, situado na Av. Darcy Vargas, 1200 - Parque Dez, dentro da Escola Superior de Tecnologia (EST) da Universidade do Estado do Amazonas (UEA). O auditório foi escolhido por ser um local bem maior que o local antes mapeado, contendo diversas cadeiras no local fazendo com que o robô tenha que desviar de algumas, além de detectar a presença ou não delas no mapa. O local pode ser visto nas figuras 84 e 85.

Figura 84: Visão frontal do auditório do Ocean

Fonte: Autor

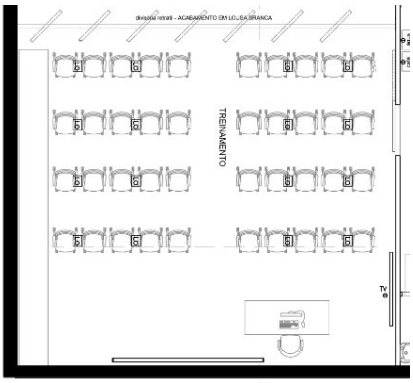
Figura 85: Visão lateral do auditório do Ocean



Fonte: Autor

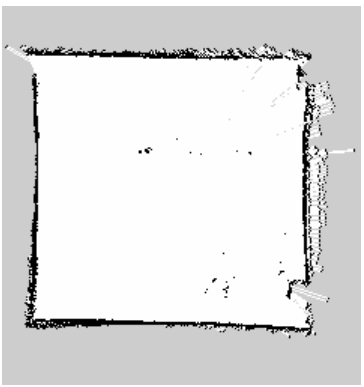
O robô navegou prioritariamente pela parte da sala que estava livre, sendo necessário andar apenas por baixo da primeira fileira de cadeiras para melhorar a qualidade do mapa. A planta baixa da sala pode ser vista na figura 86 e o mapa gerado pode ser visto na figura 87.

Figura 86: Planta Baixa do Ocean



Fonte: Autor

Figura 87: Mapa do Ocean gerado pelo robô



Fonte: Autor

Nota-se a partir das figuras 86 e 87 que o robô excluiu as cadeiras do mapeamento, isso ocorre pois como as pernas das cadeiras eram muito finas em relação ao tamanho da parede, o robô entendeu a cadeira como um ruído no sinal, corrigindo-o. No canto inferior direito, pode-se notar também um certo desnível no mapa que ocorre devido a grande junção de objetos no canto da sala como visto na figura 85.

A comparação dos dados da planta baixa projetada foi comparado ao mapa gerado. Os resultados podem ser vistos na tabela 8.

Tabela 8: Área calculada pelo *AutoCAD* do Auditório do Ocean

Imagem	**Área (m^2)**
Planta Baixa projetada	83, 72
Mapa gerado pelo robô	76, 7548

Fonte: Autor

O mapa gerado apresentou 91, 68% de precisão em relação a planta baixa do auditório mapeado. Sendo assim, o resultado foi considerado satisfatório visto que o robô conseguiu navegar autonomamente pela sala mesmo diante de diversos obstáculos difíceis de serem detectados, no caso as cadeiras, e conseguiu apresentar um mapa com bastante acurácia em relação a planta baixa do local. Uma imagem do robô durante o processo de mapeamento pode ser vista na figura 88.

Figura 88: Robô durante processo de mapeamento no auditório do Ocean

Fonte: Autor

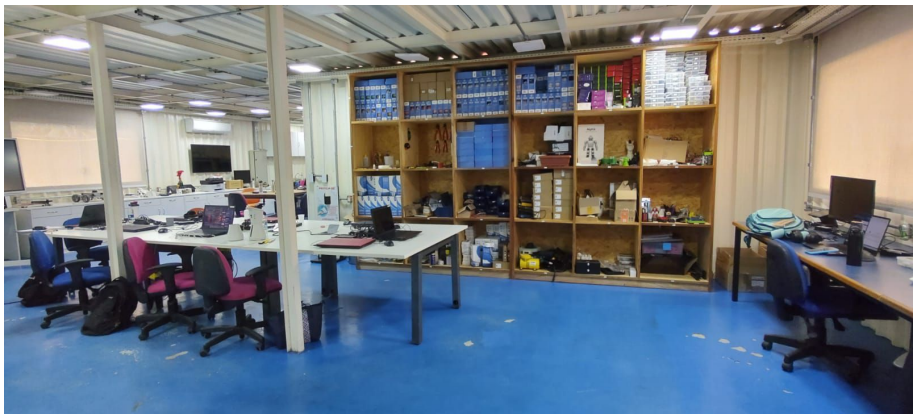
4.3 Ambiente 3: Laboratório de Fabricação Digital do Ocean

Para um lugar com área maior que $50m^2$ e com maior quantidade de obstáculos, optou-se pelo Laboratório de Fabricação Digital do Ocean, que é um prédio anexo ao prédio do Ocean Manaus. O laboratório foi utilizado não só para desenvolvimento de todo o projeto como também para testes de mapeamento no local. Algumas fotos do laboratório podem ser vistas nas figuras 89 e 90.

Figura 89: Salão principal do Laboratório de Fabricação Digital

Fonte: Autor

Figura 90: Visão lateral do Laboratório de Fabricação Digital

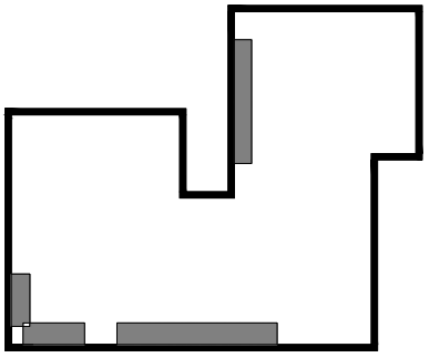


Fonte: Autor

O laboratório foi utilizado como teste final do projeto visto que possui diversos obstáculos para o robô se preocupar, além de ter uma área maior que os outros locais mapeados. Vale ressaltar que o piso do laboratório é de tinta epóxi, sendo o mesmo piso utilizado em diversos ambientes industriais atualmente, sendo assim o resultado pode ser espelhado para locomoção em ambientes industriais. O laboratório também possui diversas áreas diferentes, sendo assim o robô precisa realizar uma maior quantidade de movimentos para realizar o mapeamento.

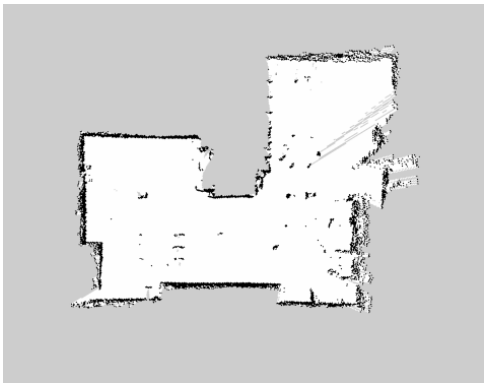
Como o laboratório não havia documentação de planta baixa, primeiramente foi feita a planta baixa de uma maneira simplificada no *AutoCAD*. Optou-se por pintar os móveis de cinza para melhor comparação com o mapa gerado. Com isso, pode-se ver a planta baixa e o mapa gerado pelo robô nas figuras 91 e 92.

Figura 91: Planta Baixa do Laboratório de Fabricação Digital



Fonte: Autor

Figura 92: Mapa do Laboratório de Fabricação Digital gerado pelo robô



Fonte: Autor

Nota-se que o robô conseguiu fazer um mapa muito próximo ao visto na planta baixa projetada. No entanto no lado direito do mapa, pode-se notar que houver muitas falhas de detecção, que ocorreram devido a presença de alguns puffs que estavam posicionados no local.

Por se tratar de uma área maior, o processo todo de mapeamento levou em torno de 20 minutos. A área de cada imagem pode ser vista na tabela 9.

Tabela 9: Área calculada pelo *AutoCAD* do Laboratório de Fabricação Digital

Imagem	**Área (m^2)**
Planta Baixa projetada	$97,8583$
Mapa gerado pelo robô	$88,4507$

Fonte: Autor

Pode-se notar que por se tratar de uma área muito maior e com maior quantidade de obstáculos no local, a eficácia do mapeamento foi menor apresentando uma precisão de $90,38\%$ em relação a planta baixa do local. Mesmo com uma acurácia mais baixa em relação aos outros locais mapeados o mapa foi considerado satisfatório por conter todos os obstáculos principais para autonomia de movimentos.

Por fim, comparou-se o tempo de mapeamento em cada local e os resultados podem ser vistos na tabela 10.

Tabela 10: Tempos de processo para cada local

Local	**Tempo (min)**	**Tempo de Processo (s/m^2)**
Loja de Material Esportivo	2	10,91
Auditório do Ocean	15	11,72
Laboratório de Fabricação Digital	20	13,56

Fonte: Autor

Como os tempos de processo não são exatos, visto o tempo de realimentação do algoritmo de mapeamento, optou-se por utilizar a média dos valores que é $12,06 s/m^2$. Nota-se que é um valor satisfatório visto que as características dos ambientes mapeados se aproximam de ambientes industriais.

O custo total de montagem do projeto pode ser visto na tabela 11. Nota-se que a quantidade de filamento para impressão 3D foi arredondada visto que os rolos vendidos no mercado são apenas de $0,5kg$ ou de $1kg$, e que alguns itens como parafusos e porcas foram comprados em uma quantidade maior que a projetada.

Tabela 11: Custo de Montagem do Protótipo

Item	**Quantidade**	**Valor Un. (R$)**	Valor Total(R$)
Filamento PLA	$1kg$	108,90	108,90
Filamento ABS	$1kg$	108,90	108,90
RPLIDAR A1M8	$1un$	492,33	492,33
Ultrassônico HC-SR04	$2un$	18,00	36,00
Motores DC	$4un$	16,90	67,60
Servomotores	$4un$	10,93	43,72
MPU-6050	$1un$	18,90	18,90
Raspberry Pi 4	$1un$	376,68	376,68
ESP32 Dev Kit	$1un$	85,40	85,40
Raspicam v2	$1un$	69,00	69,00
Parafusos, Porcas e Arruelas	-	93,93	93,93
Baterias 18650	$2un$	33,45	66,90
Power Bank	$1un$	177,90	177,90
Módulo Ponte H	$2un$	23,50	23,50
		Total	**1769,66**

Fonte: Autor

5 CONSIDERAÇÕES FINAIS

Neste trabalho, foi desenvolvido o protótipo de veículo autônomo terrestre para mapeamento de ambientes industriais. Para isso, foi necessário projetar o robô de maneira que fosse robusto o suficiente para realizar o mapeamento, projetar uma placa eletrônica para acionamento e controle do mesmo e por fim, programar os algoritmos de navegação autônoma e mapeamento.

Para se desenvolver o protótipo, iniciou-se pelo desenvolvimento do projeto mecânico, visando a locomoção em ambientes industriais. Verificou-se que o projeto atendeu aos requisitos de projeto, visto que o protótipo conseguiu se movimentar sem dificuldades pelos ambientes de testes e mantendo uma boa dinâmica durante o movimento. Vale ressaltar que o terceiro ambiente de testes possui o mesmo tipo de piso utilizado em diversos ambientes industriais, fazendo com que o local possa ser considerado como ambiente industrial e os resultados encontrados validam o processo de mapeamento em ambientes industriais.

O modelo cinemático do robô, apresentou uma boa validação para o sistema de navegação do protótipo. Os sensores e atuadores presentes no robô foram validados, indicando que o modelo cinemático do robô funcionou da maneira correta dentro da simulação. Com isso, o robô pôde fazer o mapeamento da maneira correta, através dos algoritmos posteriormente desenvolvidos.

A placa projetada atendeu as necessidades do projeto, facilitando a conexão dos dispositivos e melhorando a utilização do cabeamento. Vale ressaltar ainda que, apesar de ser um *shield* em que o microcontrolador é apenas encaixado, a placa pode ser utilizada novamente para outros projetos.

Em relação aos algoritmos de movimentação autônoma e mapeamento desenvolvidos, observou-se que o desenvolvimento destes fez com que o robô conseguisse alcançar o principal objetivo. Os testes realizados indicaram que o robô é capaz de fazer

o mapeamento em ambientes industriais, com uma acurácia média de $93,67\%$ e um tempo de processo médio de $12,06 s/m^2$.

Portanto, comprova-se a hipótese de que é possível desenvolver um protótipo de veículo autônomo terrestre capaz de mapear ambientes industriais. Também se observa que a etapa de projeto mecânico e de desenvolvimento dos algoritmos de mapeamento foram fundamentais para alcance deste objetivo.

Por fim, para dar continuidade à pesquisa e melhorar o projeto apresentado, faz-se a seguir algumas sugestões e considerações para trabalhos futuros.

a) Utilizar *encoders* para um melhor controle da velocidade do robô, além de ajudar na odometria do próprio mapeamento, podendo melhorar a qualidade do mesmo;

b) Projetar e desenvolver peças de maior resistência para conseguir dar maior robustez ao robô, fazendo com que o mesmo consiga se locomover em ambientes mais "perigosos";

c) Utilização de outros microcontroladores e baterias com maior capacidade de autonomia para aumentar a capacidade de circulação do robô dentro de um ambiente;

d) Desenvolvimento de algoritmo capaz de não só mapear o ambiente como também processar dados referentes ao mesmo, como por exemplo, gerar a planta baixa de um ambiente através apenas dos dados dos sensores;

e) Utilização de um ou mais sensores que possam auxiliar o sensor Lidar no mapeamento para evitar que ocorram certos desvios no mapeamento;

f) Testes em ambientes industriais com obstáculos dinâmicos para ajustes e melhoria do sistema de mapeamento implementado.

REFERÊNCIAS

ALMEIDA, R. **Ponte H com bootstrap para acionamento de motores DC**. 2014. Disponível em: <https://www.embarcados.com.br/ponte-h-bootstrap-acionamento-motores-dc/>. Acesso em: 02 de maio de 2022.

AUTODESK. **Software de Modelagem 3D**. 2022. Disponível em: <https://www.autodesk.com.br/solutions/3d-modeling-software>. Acesso em: 10 de março de 2022.

AUTOMATION, L. **Mobile Robot AGV**. 2021. Disponível em: <https://www.lisenautomation.com>. Acesso em: 25 de novembro 2021.

BALLUFF. **O que é Servo motor e como funciona?** 2022. Disponível em: <https://balluffbrasil.com.br/sensor-ultrassonico-como-ele-funciona-e-de-que-modo-pode-ajudar-a-sua-industria/>. Acesso em: 25 de abril de 2022.

BORENSTEIN, J. et al. Mobile robot positioning: Sensors and techniques. **Journal of Robotic Systems**, v. 14, 1997.

BRäUNL, T. **Embedded robotics (third edition): Mobile robot design and applications with embedded systems**. [S.l.: s.n.], 2008.

CANAL, I. P.; VALDIERO, A. C.; REIMBOLD, M. P. Modelagem matemática de motor de corrente contínua e análise dinâmica. In: . [S.l.: s.n.], 2017.

CANONICAL. **Key considerations when choosing a robot's operaitng system**. 2018. Disponível em: <https://ubuntu.com/blog/from-ros-prototype-to-production-on-ubuntu-core>. Acesso em: 20 de abril de 2022.

CASTRO, L. **Entendo o funcionamento do sensor inclinação (Giroscópio) com Arduino**. 2021. Disponível em: <https://blog.arduinoomega.com/5-passos-para-utilizar-o-sensor-de-inclinacao-giroscopio/>. Acesso em: 29 de abril de 2022.

CASTRUCCI, P.; BITTAR, A.; SALES, R. M. **Controle Automático**. Rio de Janeiro: LTC, 2011.

CHAPMAN, S. J. **Fundamentos de Máquinas Elétricas**. [S.l.: s.n.], 1989. v. 53. ISSN 1098-6596.

CRAIG, J. J. **Robótica**. [S.l.: s.n.], 1977. v. 1. ISSN 0140-6736.

DJI. **RoboMaster S1**. 2021. Disponível em: <https://www.dji.com>. Acesso em: 26 de novembro 2021.

DUDEK, G.; JENKIN, M. **Computational Principles of Mobile Robotics**. [S.l.: s.n.], 2010.

EASYTRONICS. **Acelerômetro**. 2021. Disponível em: <https://www.easytronics.com.br>. Acesso em: 27 de novembro 2021.

ELETROGATE. **Módulo GPS NEO-6M com Antena**. 2021. Disponível em: <https://www.eletrogate.com>. Acesso em: 28 de novembro 2021.

ESPACIAIS, I. N. de P. **LIDAR**. 2022. Disponível em: <http://www.dsr.inpe.br/DSR/areas-de-atuacao/sensores-plataformas/lidar>. Acesso em: 29 de abril de 2022.

FOUNDATION, R. P. **Raspberry Pi Documentation**. 2021. Disponível em: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-4-model-b>. Acesso em: 27 de março de 2022.

____. **Raspberry Pi Documentation**. 2021. Disponível em: <https://www.raspberrypi.com/documentation/accessories/camera.html>. Acesso em: 27 de março de 2022.

FRANZ, M. O.; MALLOT, H. A. Biomimetic robot navigation. **Robotics and Autonomous Systems**, v. 30, 2000. ISSN 09218890.

GTMAX3D. **Impressora 3D Pro - GTMAX3D Core H5**. 2022. Disponível em: <https://www.gtmax3d.com.br/impressora-3d-pro/gtmax3d-core-h5-simplify3d>. Acesso em: 02 de maio de 2022.

GUIMARãES, F. de A. Desenvolvimento de robô móvel utilizado para a exploração de ambientes hostis. 2007.

KALATEC. **O que é servomotor, como funciona e quais as vantagens?** 2022. Disponível em: <https://blog.kalatec.com.br/o-que-e-servo-motor>. Acesso em: 25 de abril de 2022.

KOHLBRECHER, S. et al. A flexible and scalable slam system with full 3d motion estimation. In: IEEE. **Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)**. [S.l.], 2011.

LAB, D. **Manufatura Aditiva: saiba o que é e o que ela representa.** 2021. Disponível em: <https://3dlab.com.br/>. Acesso em: 30 de novembro 2021.

LYNCH, K. M.; PARK, F. C. **Modern robotics**. [S.l.]: Cambridge University Press, 2017.

MATARIć, M. J. **The Robotics Primer**. [S.l.]: MIT Press, 2007.

MATHWORKS. **Exchange Data with ROS Publishers and Subscribers.** 2021. Disponível em: <https://la.mathworks.com/help//ros/>. Acesso em: 30 de novembro 2021.

____. **What is SLAM? 3 Things you need to know**. 2021. Disponível em: <https://www.mathworks.com/discovery/slam.html>. Acesso em: 30 de novembro 2021.

MATTEDE, H. **O que é Servo motor e como funciona?** 2022. Disponível em: <https://www.mundodaeletrica.com.br/o-que-e-servo-motor-e-como-funciona/>. Acesso em: 25 de abril de 2022.

MECALUX. **Manufatura aditiva: a impressão 3D digitaliza a manufatura.** 2021. Disponível em: <https://www.mecalux.com.br/>. Acesso em: 30 de novembro 2021.

MULTCOMERCIAL. **Tipos de sensores: saiba quais são e as aplicações de cada um**. 2021. Disponível em: <https://blog.multcomercial.com.br/tipos-de-sensores/>. Acesso em: 30 de novembro 2021.

NASA. **Curiosity**. 2021. Disponível em: <https://www.nasa.gov>. Acesso em: 24 de novembro 2021.

NEHMZOW, U. **Mobile Robotics: A Practical Introduction**. [S.l.: s.n.], 2003.

PIERI, E. R. de. **Curso de Robótica Móvel**. [S.l.]: UFSC, 2002.

PIXAR. **Wall-e**. 2021. Disponível em: <https://www.pixar.com>. Acesso em: 26 de novembro de 2021.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: . [S.l.: s.n.], 2009. v. 3.

ROBOTICS, O. **ROS**. 2021. Disponível em: <https://www.ros.org/>. Acesso em: 30 de novembro 2021.

____. **URDF**. 2021. Disponível em: <https://www.ros.org>. Acesso em: 30 de novembro 2021.

ROBOTLAB. **NAO**. 2021. Disponível em: <https://www.robotlab.com>. Acesso em: 25 de novembro 2021.

RODRIGUES, D. P. Técnicas de navegação. 2010.

ROLAND. **MonoFab SRM-20**. 2022. Disponível em: <https://www.rolanddg.com.br/produtos/3d/srm-20-fresadora-3d-compacta>. Acesso em: 04 de maio de 2022.

SAMSUNG. **Robô Samsung Powerbot VR7200**. 2021. Disponível em: <https://www.samsung.com/br>. Acesso em: 25 de novembro 2021.

SANTORA, M. **5 Challenges When Creating Autonomous Navigation Systems**. 2018. Disponível em: <https://www.therobotreport.com/autonomous-navigation-design-challenges/>. Acesso em: 10 de maio 2021.

SETHI3D. **Impressora Sethi3D S4X**. 2022. Disponível em: <https://www.sethi3d.com.br/s4x>. Acesso em: 02 de maio de 2022.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to Autonomous Mobile Robots, Second Edition**. [S.l.: s.n.], 2011. v. 23. ISSN 0263-5747.

SILVEIRA, C. B. **O que é PWM e Para que Serve?** 2016. Disponível em: <https://www.citisystems.com.br/pwm/>. Acesso em: 10 de maio 2021.

SLAMTEC. **RPLidar-A1**. 2021. Disponível em: <https://www.slamtec.com>. Acesso em: 27 de novembro 2021.

SPACEBOK. **Spacebok**. 2021. Disponível em: <https://www.spacebok.ethz.ch>. Acesso em: 25 de novembro 2021.

SYSTEMS, E. **ESP32 Documentation**. 2022. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 29 de março de 2022.

TECHNOLOGIES, W. **Manufatura Aditiva: entenda o que é e como ela funciona.** 2021. Disponível em: <https://www.wishbox.net.br>. Acesso em: 30 de novembro 2021.

TECNOLOGIA, H. **O que é Encoder? Para que serve? Como escolher? Como interfacear?** 2021. Disponível em: <https://www.hitecnologia.com.br>. Acesso em: 29 de novembro 2021.

THOMSEN, A. **Como conectar o Sensor Ultrassônico HC-SR04 ao Arduino**. 2022. Disponível em: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/>. Acesso em: 25 de abril de 2022.

TRULLIER, O.; MEYER, J. A. Biomimetic navigation models and strategies in animats. **AI Communications**, v. 10, 1997. ISSN 09217126.

TRULLIER, O. et al. Biologically based artificial navigation systems: Review and prospects. **Progress in Neurobiology**, v. 51, 1997. ISSN 03010082.

UFSC, L. de Robótica da. **Projeto Prótese Mão Amiga**. 2021. Disponível em: <https://robotica.ufsc.br/projeto-protese-mao-amiga/>. Acesso em: 30 de novembro 2021.

VECTORNAV. **What is an Inertial Measurement Unit**. 2022. Disponível em: <https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu>. Acesso em: 28 de abril de 2022.

VINCROSS. **Hexa**. 2021. Disponível em: <https://www.vincross.com>. Acesso em: 26 de novembro 2021.

WALFORD, L. **The Who, What, When, Where, Why, and How of Lidar**. 2019. Disponível em: <https://www.autofutures.tv/2019/02/11/the-who-what-when-where-why-and-how-of-lidar/>. Acesso em: 29 de abril de 2022.

WOWWEE. **Miposaur**. 2021. Disponível em: <https://www.wowwee.com>. Acesso em: 25 de novembro 2021.

ZAMAN, S.; SLANY, W.; STEINBAUER, G. Ros-based mapping, localization and autonomous navigation using a pioneer 3-dx robot and their relevant issues. In: . [S.l.: s.n.], 2011.

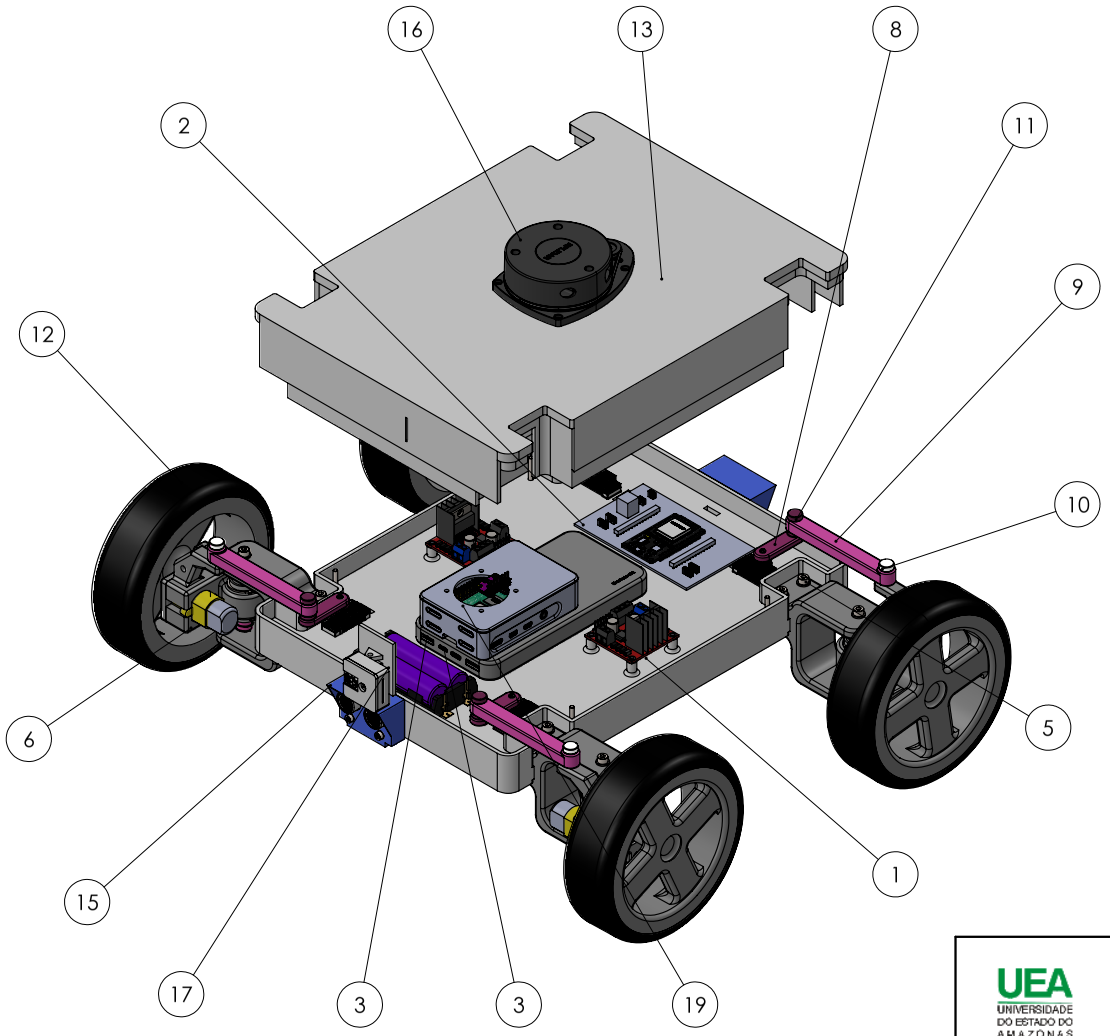
APÊNDICE A

Neste apêndice são apresentados todos os desenhos técnicos das peças projetadas no decorrer do projeto, seguindo as normas da ABNT. A documentação das peças inicia com a lista de BOM do projeto, sendo seguida da documentação geral e posteriormente as peças projetadas seguindo a ordem apresentada na tabela 12.

Tabela 12: Peças documentadas

Peça	Descrição
P01	Base do Chassi
P02	Garra de suporte do motor
P03	Eixo do motor
P04	Braço curto
P05	Braço longo
P06	Pino de fixação
P07	Pino de fixação
P08	Suporte do motor
P09	Suporte do motor
P10	Roda
P11	Tampa do suporte do Motor
P12	Molas
P13	Tampa do chassi

Fonte: Autor



Nº DO ITEM	DESCRIÇÃO	QTD.
1	Base do chassi	1
2	Placa do robô	1
3	Power bank	1
4	Servo motor MG-90	4
5	Garra de suporte do motor	4
6	Suporte do Motor	2
7	Suporte do Motor	2
8	Braço curto	4
9	Braço longo	4
10	Pino de encaixe	4
11	Pino de encaixe	4
12	Rodas	4
13	Tampa do chassi	1
15	Suporte ultrassônico	2
16	Lidar A1M8	1
17	Suporte Câmera	1
19	Raspberry Pi	1

UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas; - Chop and break sharp edges;
- Dimensões: Milimetros (mm); - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm - Tolerance when not specified: ±0.25mm;

MATERIAL: MATERIAL:

REVISÃO: REVISION: 00

DATA: DATE: 15/05/2022

DESENHISTA: DESIGNER: Vitor Gadelha

TÍTULO: TITLE: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

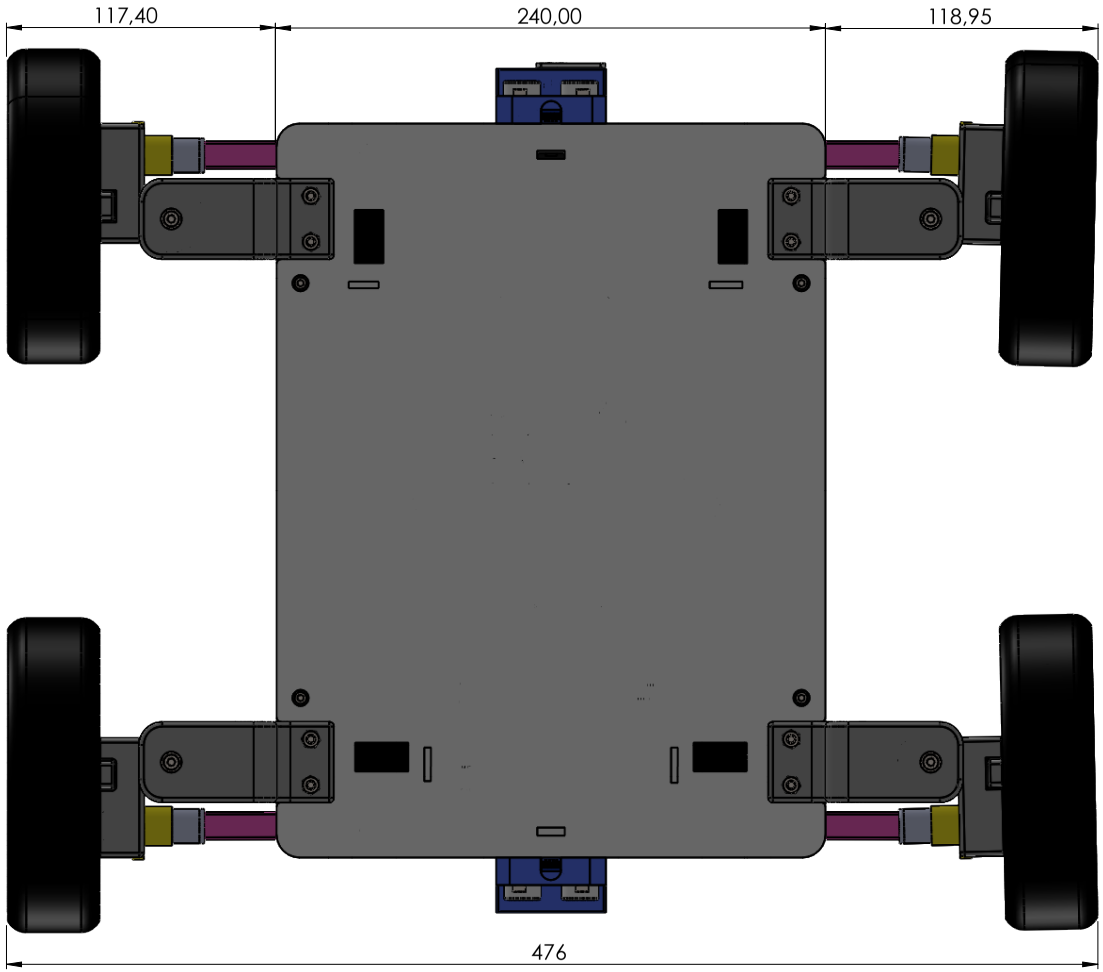
PEÇA: Part: LISTA DE MATERIAIS (BOM)

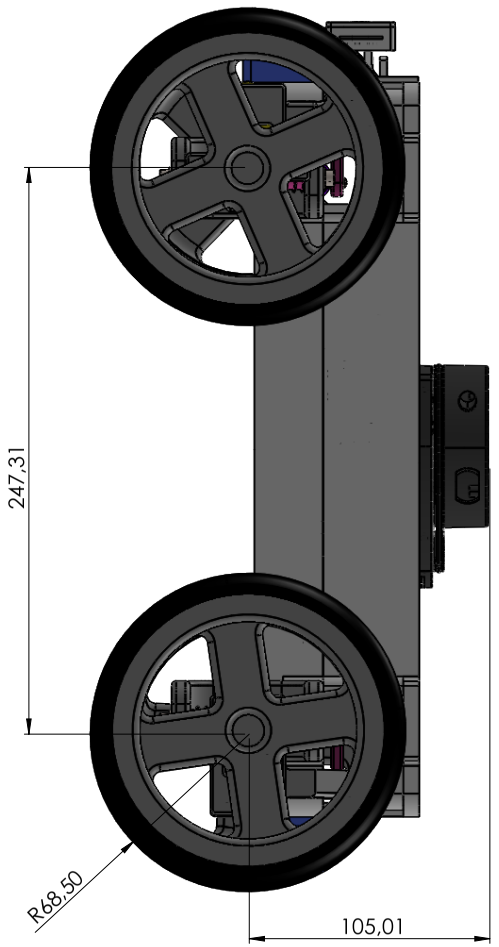
A3

PESO: WEIGHT:

ESCALA: 1:10 SCALE:

FOLHA: 1 DE 1 SHEET:





UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

DESENHISTA: DESIGNER: Vitor Gadelha

TÍTULO: TITLE: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

- Desbastar e quebrar pontas afiadas; - Chop and break sharp edges;
- Dimensões: Milimetros (mm); - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm - Tolerance when not specified: ±0.25mm;

MATERIAL: MATERIAL: VÁRIOS

REVISÃO: REVISION: 00

DATA: DATE: 15/05/2022

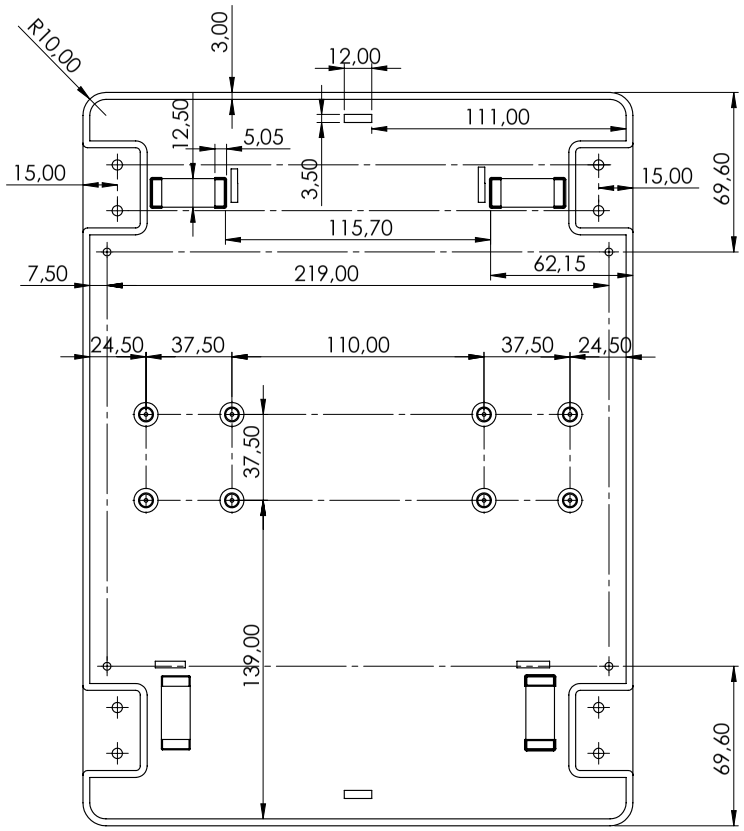
PEÇA: PART: Cotas Gerais

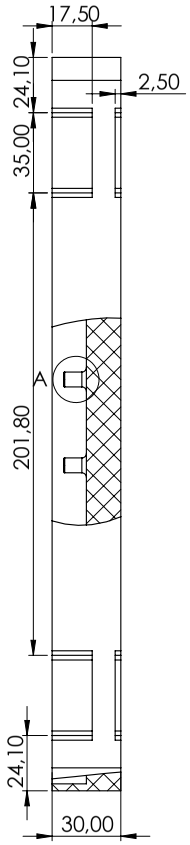
A3

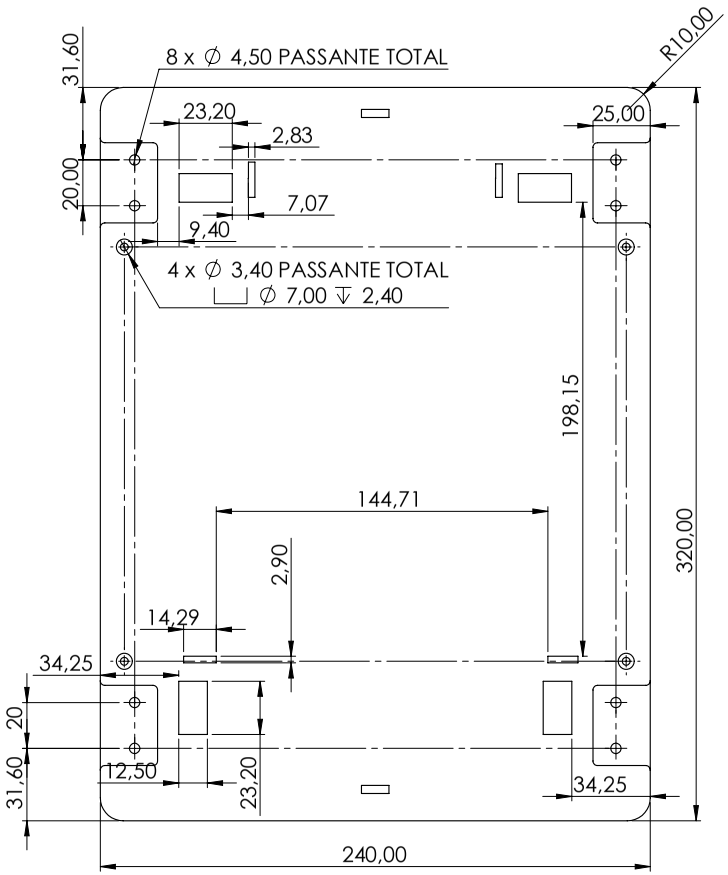
PESO: WEIGHT:

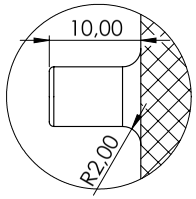
ESCALA: 1:2 SCALE:

FOLHA: 1 DE 1 SHEET:









DETALHE A
ESCALA 2 : 1

UEA UNIVERSIDADE DO ESTADO DO AMAZONAS | EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas; / - Chop and break sharp edges;
- Dimensões: Milimetros (mm); / - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm / - Tolerance when not specified: ±0.25mm;

MATERIAL: / MATERIAL:	ABS
REVISÃO: / REVISION:	00
DATA: / DATE:	15/05/2022

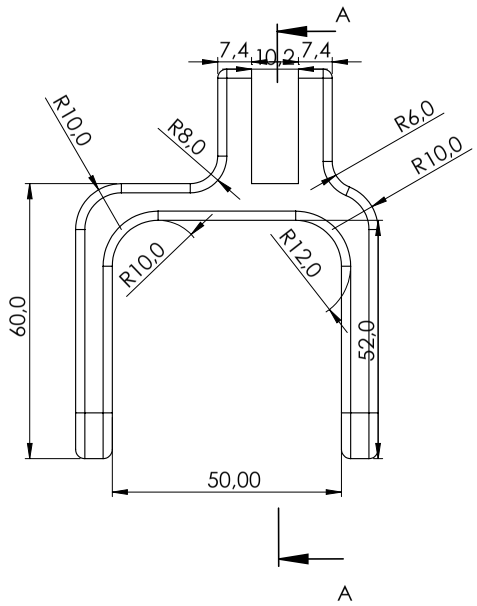
DESENHISTA: / DESIGNER: Vitor Gadelha

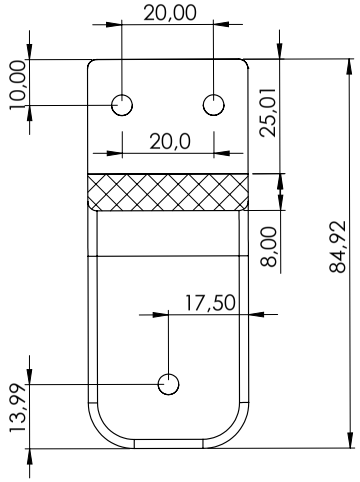
TÍTULO: / TITLE: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

PEÇA: / Part: BASE DO CHASSI

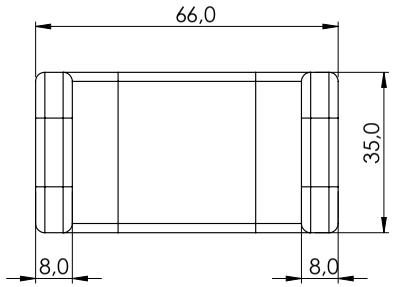
A3

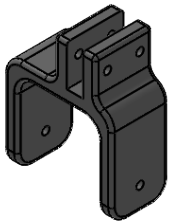
PESO: / WEIGHT: | ESCALA: 1:2 / SCALE: | FOLHA: 1 DE 1 / SHEET:





SEÇÃO A-A





UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA

REVISÃO: 00

DATA: 15/05/2022

DESENHISTA: Vitor Gadelha

TÍTULO:
Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

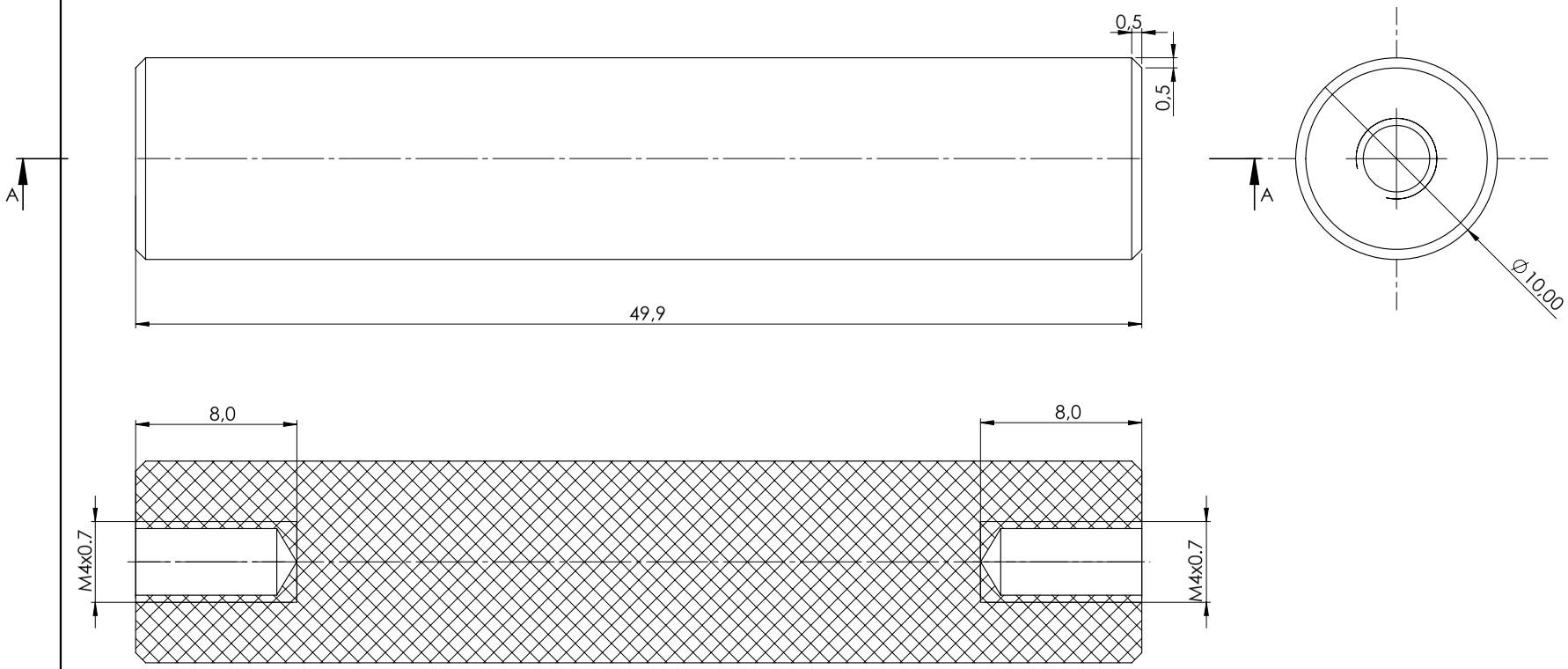
PEÇA: GARRA DE SUPORTE DO MOTOR

A3

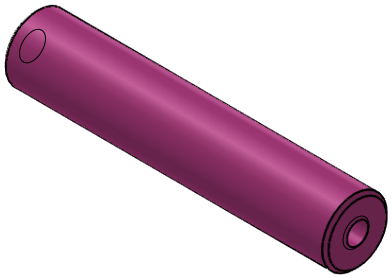
PESO:

ESCALA: 1:1

FOLHA: 1 DE 1



SEÇÃO A-A
ESCALA 5 : 1



UEA UNIVERSIDADE DO ESTADO DO AMAZONAS
EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas; - Chop and break sharp edges;
- Dimensões: Milimetros (mm); - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA
REVISÃO: 00
DATA: 15/05/2022

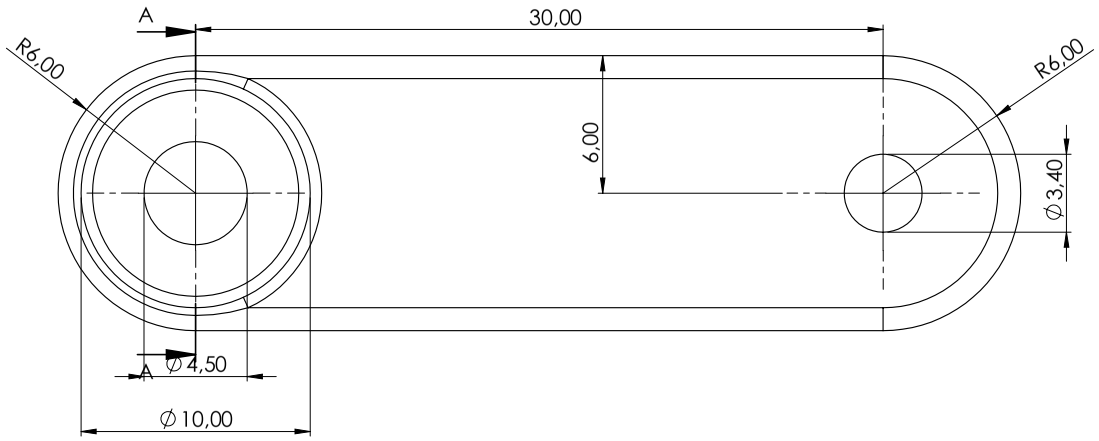
DESENHISTA: Vitor Gadelha

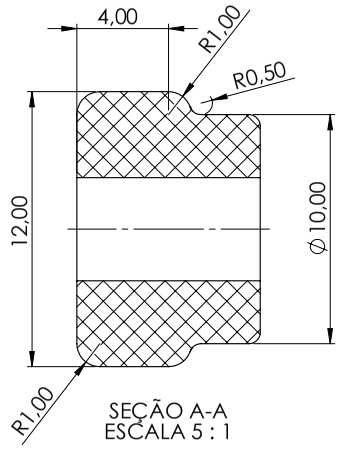
TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

PEÇA: EIXO DO SUPORTE DO MOTOR

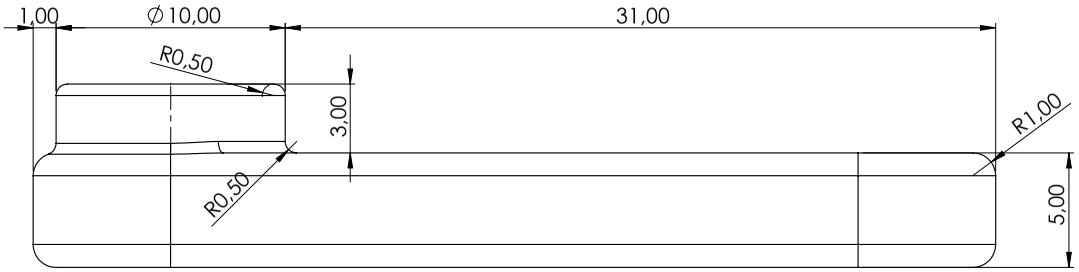
A3

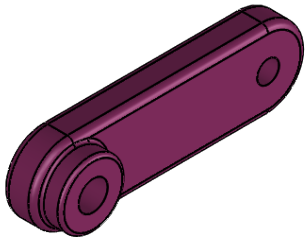
PESO: | ESCALA: 5:1 | FOLHA: 1 DE 1





SEÇÃO A-A
ESCALA 5 : 1





UEA UNIVERSIDADE DO ESTADO DO AMAZONAS
EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA

REVISÃO: 00

DATA: 15/05/2022

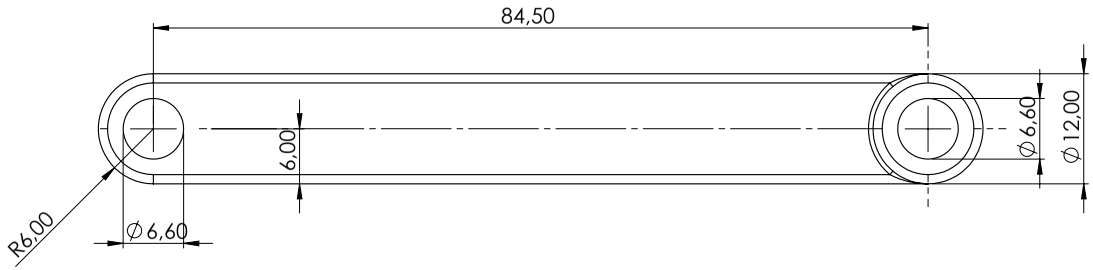
DESENHISTA: Vitor Gadelha

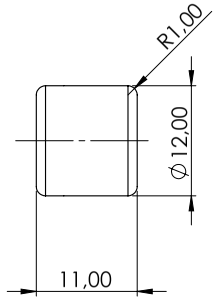
TÍTULO:
Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

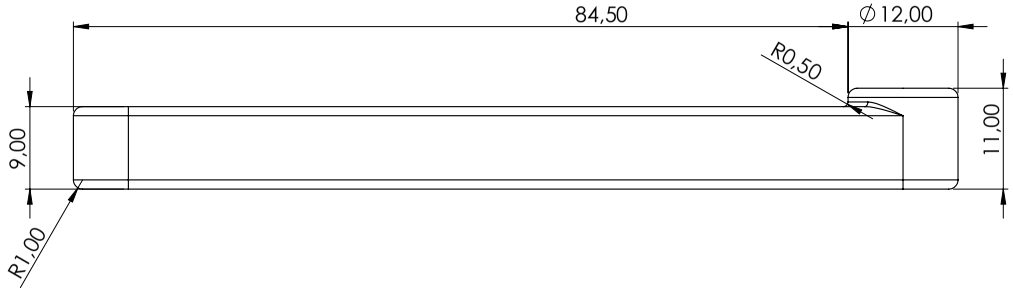
PEÇA: BRAÇO CURTO

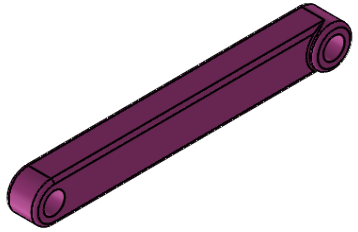
A3

PESO: | ESCALA: 5:1 | FOLHA: 1 DE 1









UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA

REVISÃO: 00

DATA: 15/05/2022

DESENHISTA: Vitor Gadelha

TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

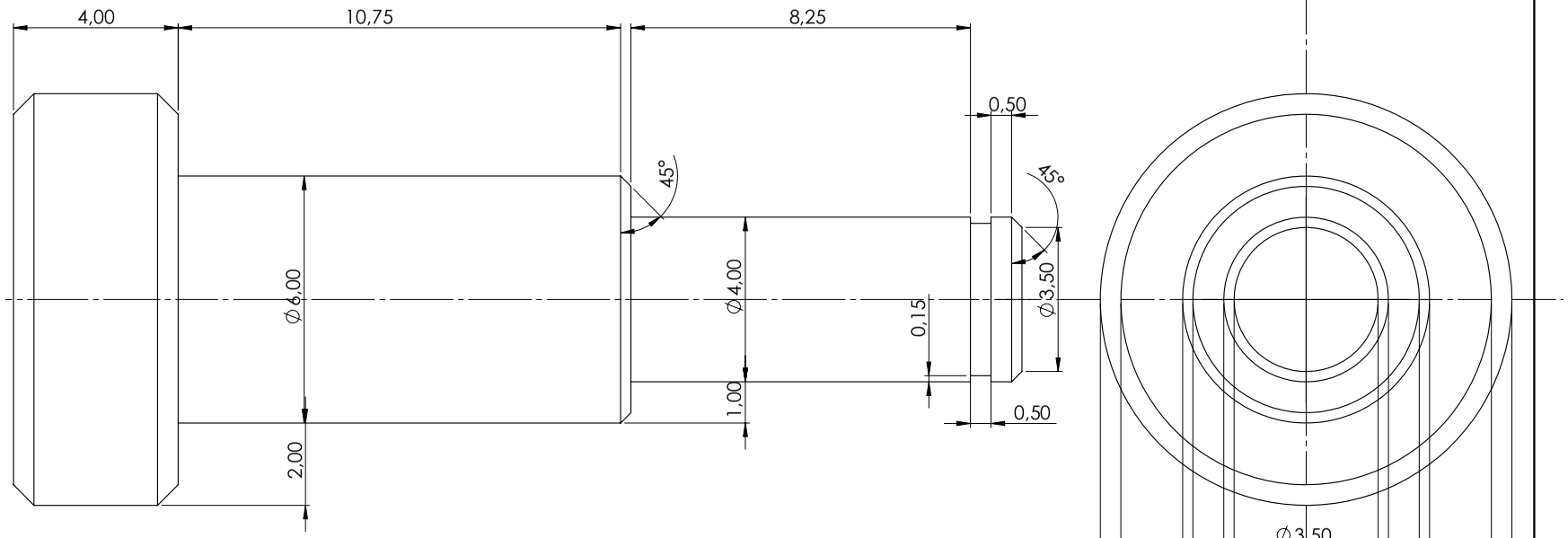
PEÇA: BRAÇO LONGO

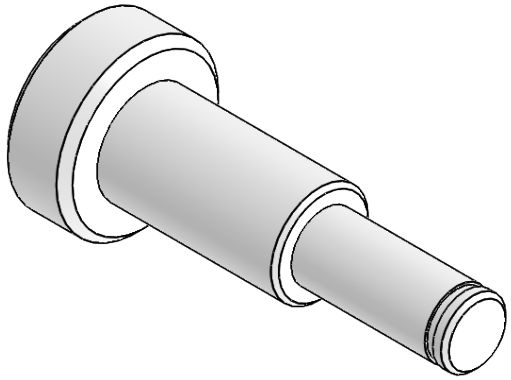
A3

PESO:

ESCALA: 2:1

FOLHA: 1 DE 1





UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA

REVISÃO: 00

DATA: 15/05/2022

DESENHISTA: Vitor Gadelha

TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

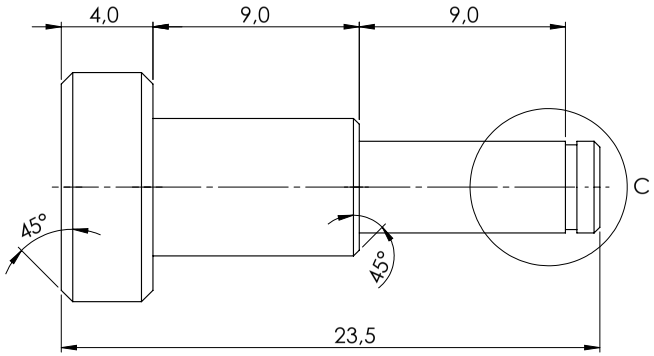
PEÇA: PINO DE ENCAIXE

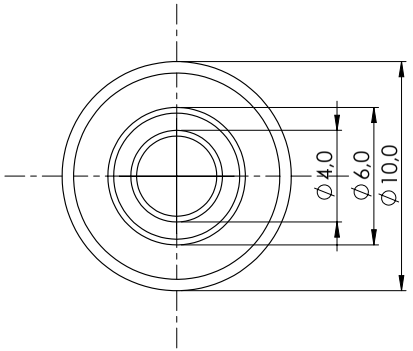
A3

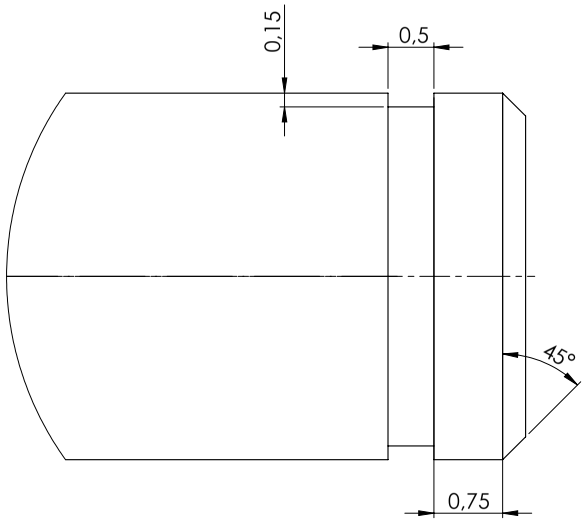
PESO:

ESCALA: 10:1

FOLHA: 1 DE 1







DETALHE C
ESCALA 20 : 1

UEA UNIVERSIDADE DO ESTADO DO AMAZONAS | EST Escola Superior de Tecnologia da UEA

DESENHISTA: Vitor Gadelha

TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

- Desbastar e quebrar pontas afiadas; - Chop and break sharp edges;
- Dimensões: Milimetros (mm); - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA

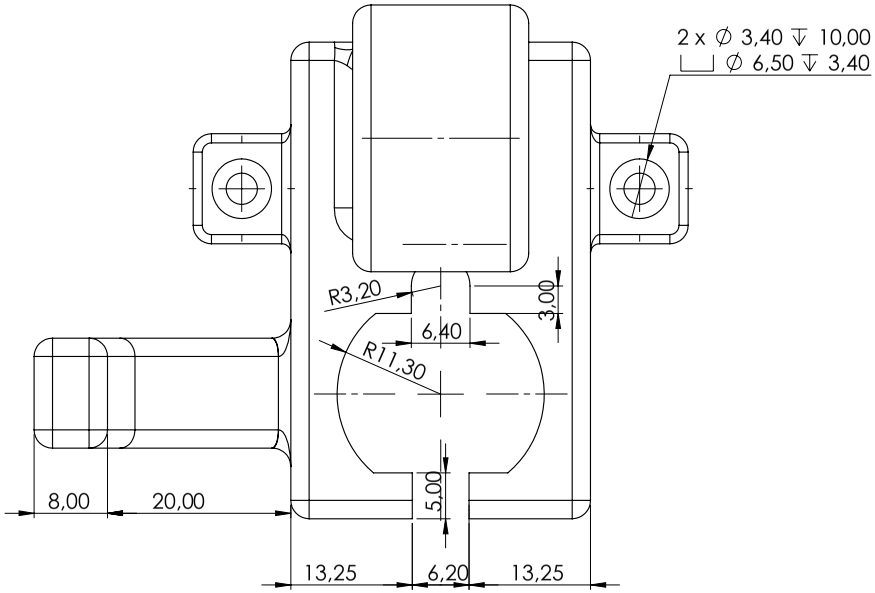
REVISÃO: 00

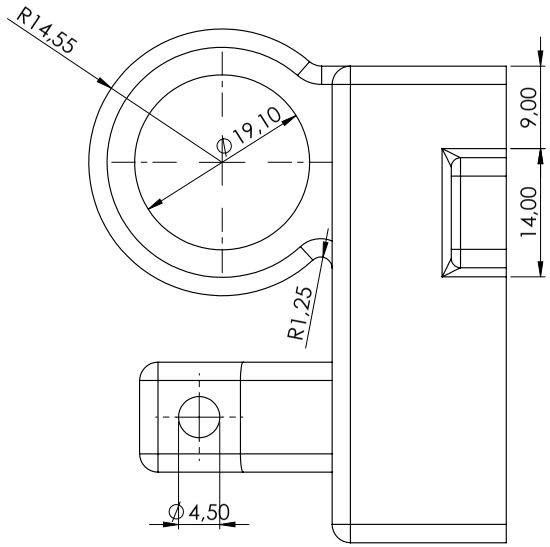
DATA: 15/05/2022

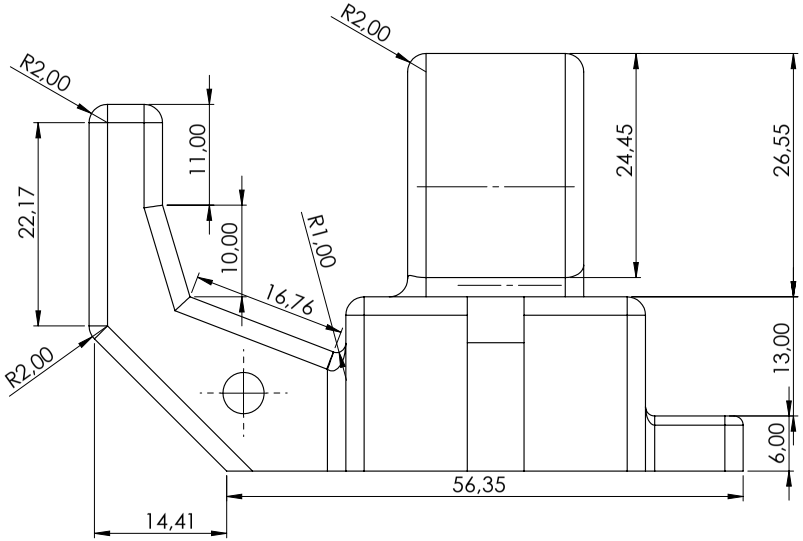
PEÇA: PINO DE ENCAIXE

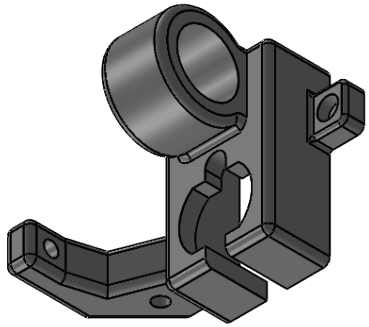
A3

PESO: | ESCALA: 5:1 | FOLHA: 1 DE 1









UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA

REVISÃO: 00

DATA: 15/05/2022

DESENHISTA: Vitor Gadelha

TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

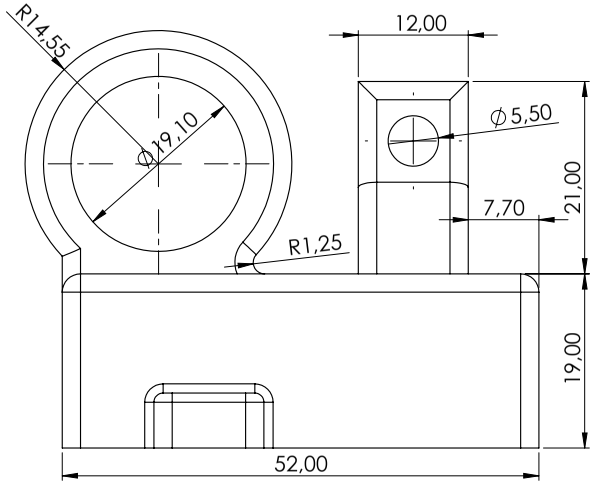
PEÇA: SUPORTE DO MOTOR

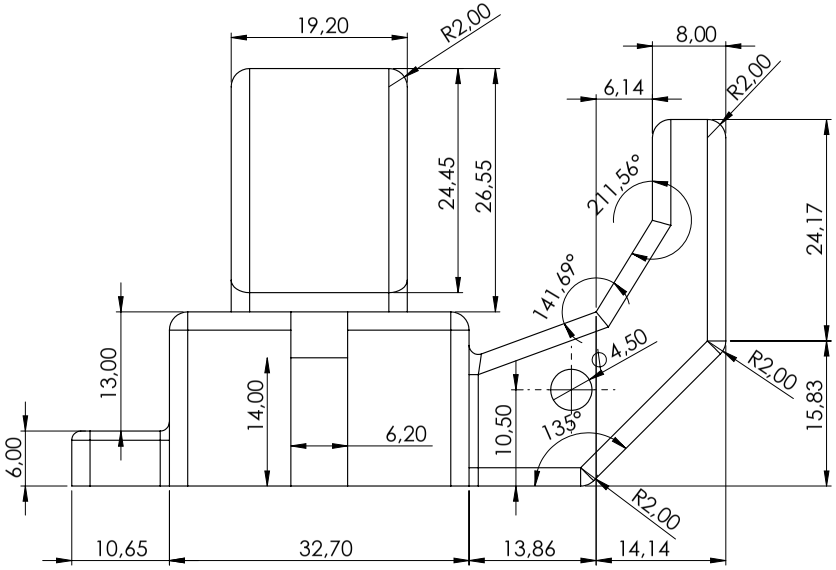
A3

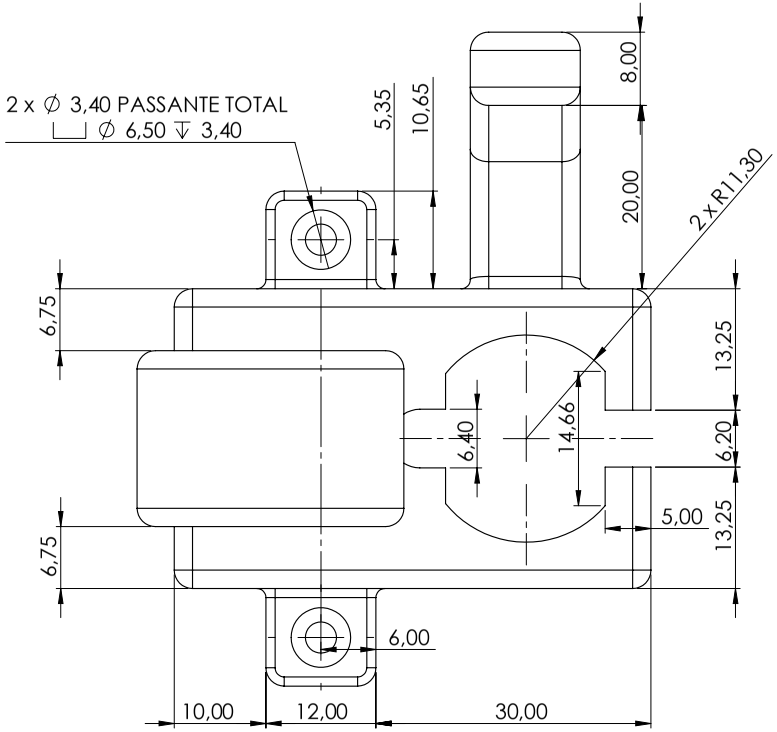
PESO:

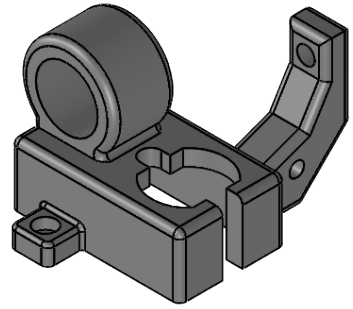
ESCALA: 2:1

FOLHA: 1 DE 1

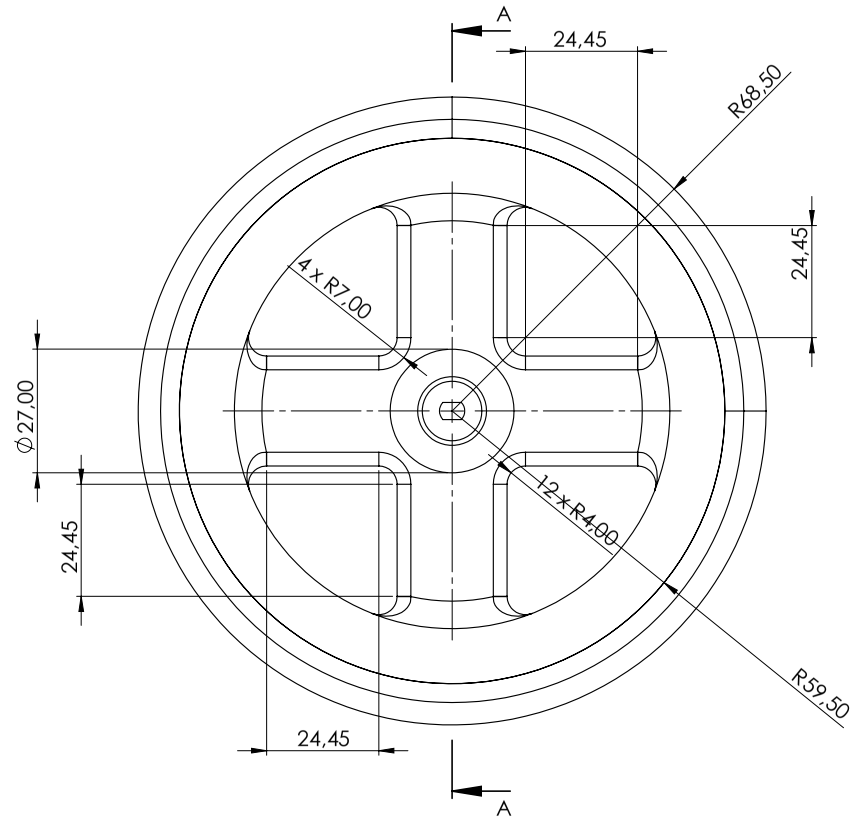


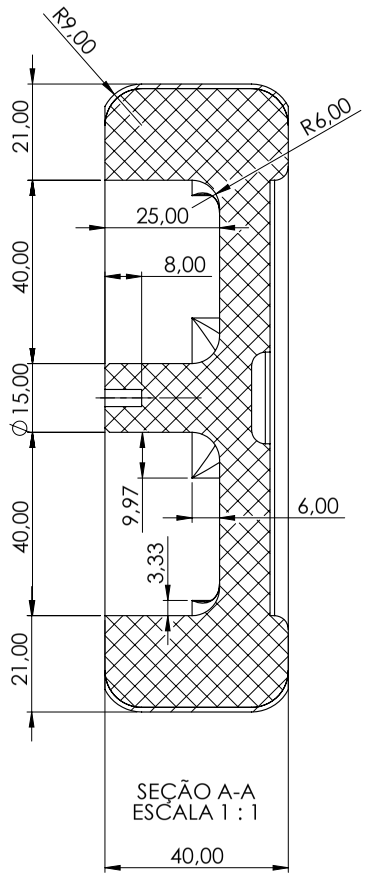






UEA Universidade do Estado do Amazonas · EST Escola Superior de Tecnologia da UEA		DESENHISTA: Vitor Gadelha	
- Desbastar e quebrar pontas afiadas; - Chop and break sharp edges; - Dimensões: Milimetros (mm); - Dimensions: Milimeters(mm); - Tolerância caso não especificado: ±0.25mm - Tolerance when not specified: ±0.25mm;	MATERIAL: PLA REVISÃO: 00 DATA: 15/05/2022	TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais PEÇA: SUPORTE DO MOTOR	A3
		PESO:	ESCALA: 2:1 · FOLHA: 1 DE 1





UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA

REVISÃO: 00

DATA: 15/05/2022

DESENHISTA: Vitor Gadelha

TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

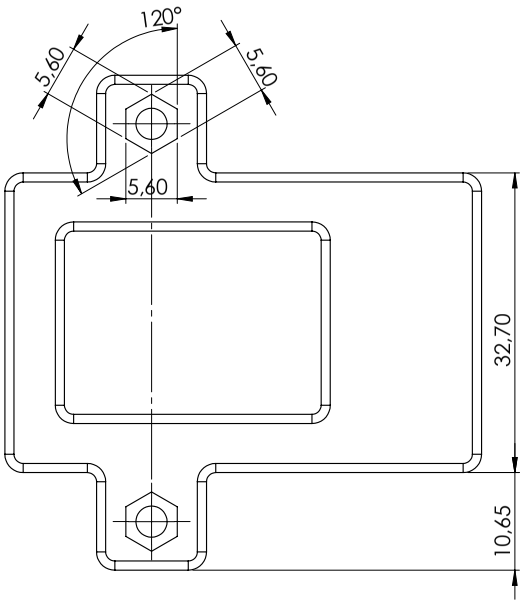
PEÇA: RODA

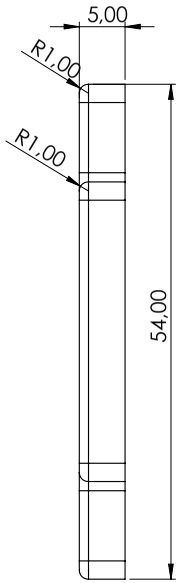
A3

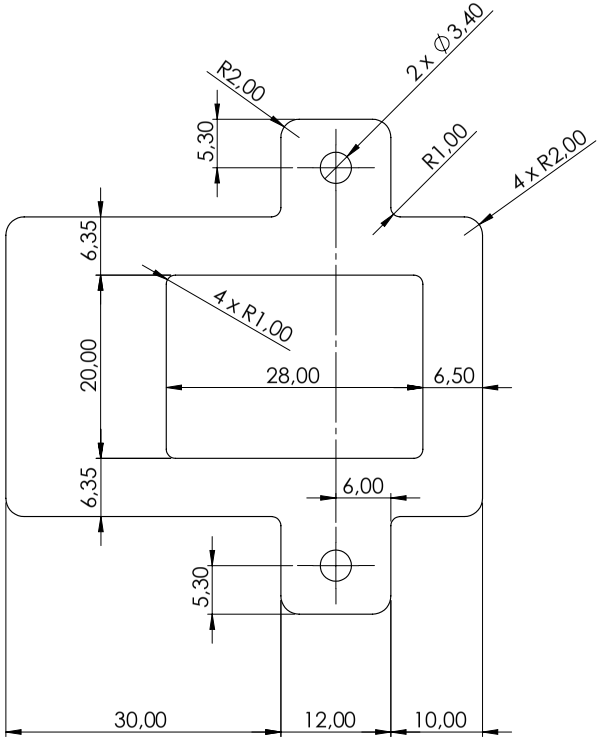
PESO:

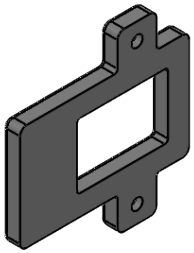
ESCALA: 1:1

FOLHA: 1 DE 1









UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: PLA

REVISÃO: 00

DATA: 15/05/2022

DESENHISTA: Vitor Gadelha

TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

PEÇA: TAMPA DO SUPORTE DO MOTOR

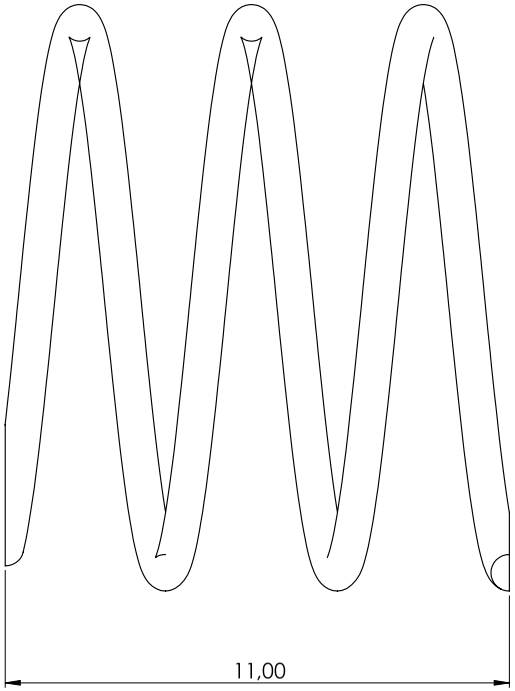
A3

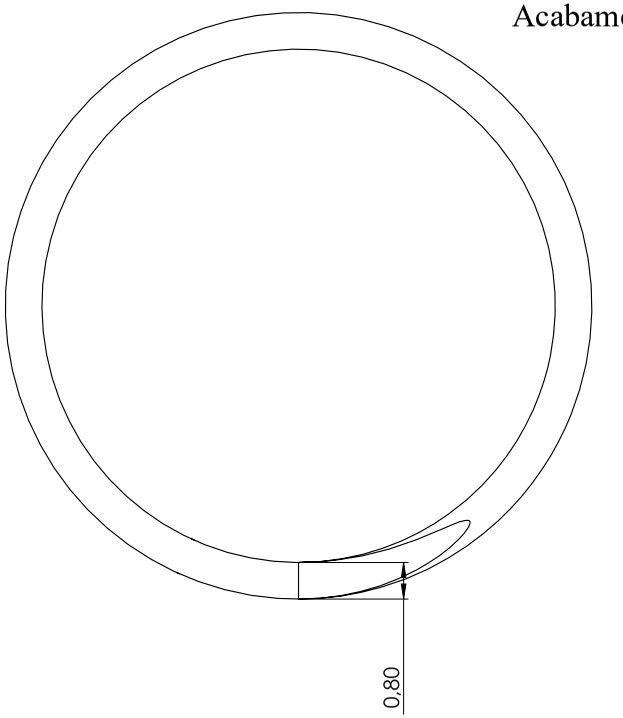
PESO:

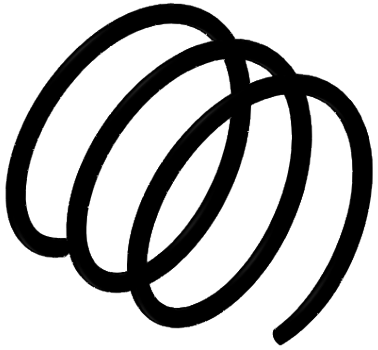
ESCALA: 2:1

FOLHA: 1 DE 1

Diâmetro externo: 12,8mm
Diâmetro do arame: 0,8mm
Passo: 3,75mm
Comprimento: 11mm
Acabamento: Natural







UEA UNIVERSIDADE DO ESTADO DO AMAZONAS

EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: AÇO MOLA

REVISÃO: 00

DATA: 15/05/2022

DESENHISTA: Vitor Gadelha

TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

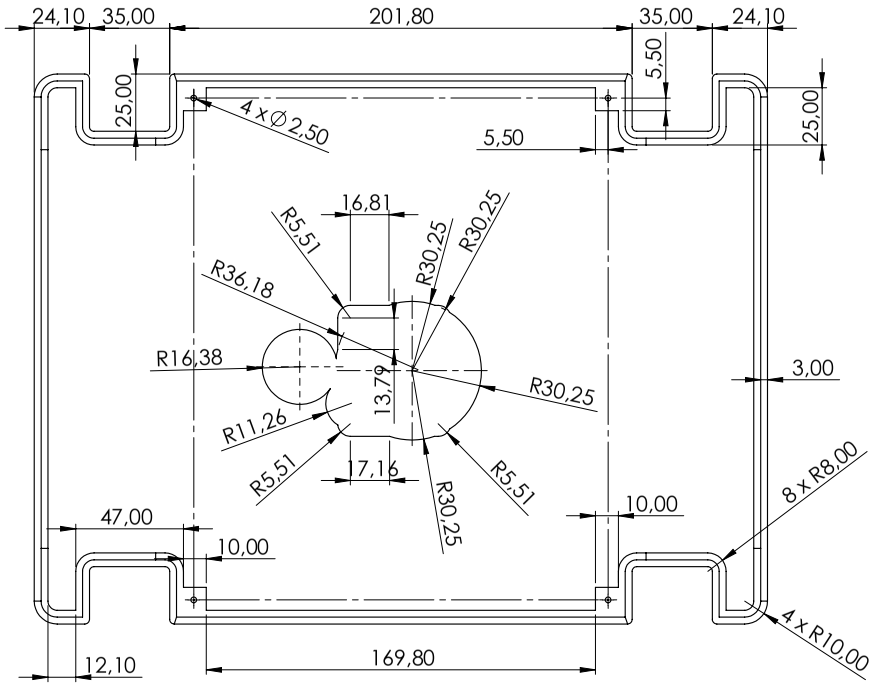
PEÇA: MOLA

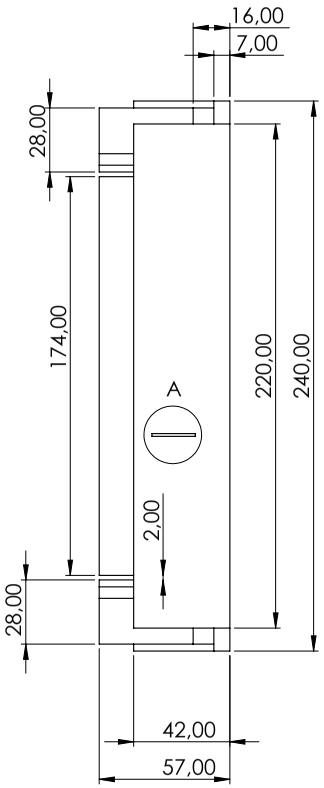
A3

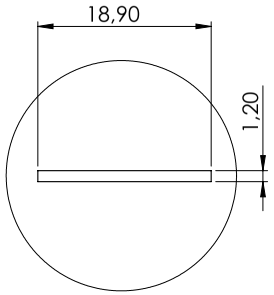
PESO:

ESCALA: 10:1

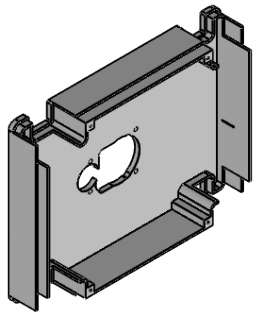
FOLHA: 1 DE 1

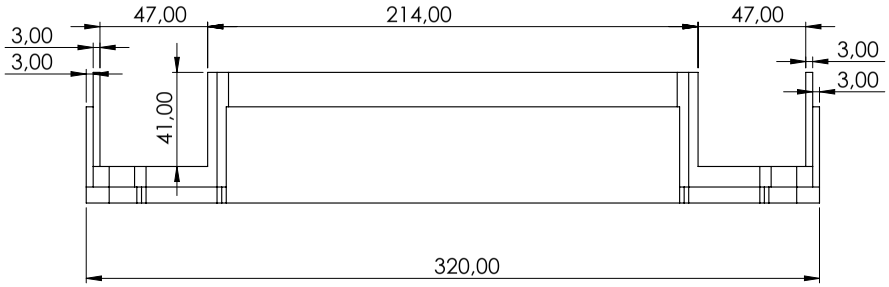






DETALHE A
ESCALA 2 : 1





UEA UNIVERSIDADE DO ESTADO DO AMAZONAS | EST Escola Superior de Tecnologia da UEA

- Desbastar e quebrar pontas afiadas;
  - Chop and break sharp edges;
- Dimensões: Milimetros (mm);
  - Dimensions: Milimeters(mm);
- Tolerância caso não especificado: ±0.25mm
  - Tolerance when not specified: ±0.25mm;

MATERIAL: ABS

REVISÃO: 00

DATA: 15/05/2022

DESENHISTA: Vitor Gadelha

TÍTULO: Protótipo de Veículo Autônomo Terrestre para Mapeamento de Ambientes Industriais

PEÇA: TAMPA DO CHASSI

A3

PESO: | ESCALA: 1:2 | FOLHA: 1 DE 1

APÊNDICE B

Listing B.1: tcc.ino

```
#include <ros.h>
#include <ESP32Servo.h>
#include <NewPing.h>
#include <tcc/Ultrasonic.h>
#include <geometry_msgs/Twist.h>

Servo left_front_arm;
Servo right_front_arm;
Servo left_back_arm;
Servo right_back_arm;

//Left Front Arm
int left_front_armPin = 15;
int left_front_motorPin1 = 2;
int left_front_motorPin2 = 4;
int left_front_motorEna = 16;

//Right Front Arm
int right_front_armPin = 12;
int right_front_motorPin1 = 27;
int right_front_motorPin2 = 26;
int right_front_motorEna = 25;

//Left Back Arm
int left_back_armPin = 21;
int left_back_motorPin1 = 5;
int left_back_motorPin2 = 18;
int left_back_motorEna = 19;
```

```
29
30 //Right Back Arm
31 int right_back_armPin = 14;
32 int right_back_motorPin1 = 33;
33 int right_back_motorPin2 = 32;
34 int right_back_motorEna = 13;
35
36 #define front_trigPin 22
37 #define front_echoPin 39
38
39 #define back_trigPin 1
40 #define back_echoPin 36
41
42 #define maxDistance 400
43 #define intervalR 200
44
45 NewPing front_sonar(front_trigPin, front_echoPin,
      maxDistance);
46 NewPing back_sonar(back_trigPin, back_echoPin,
      maxDistance);
47
48 //variables
49 float front_range;
50 unsigned long range_timer;
51 float front_duration, front_distance;
52
53 float back_range;
54 float back_duration, back_distance;
55
56 float front_returnDistance(){
57   digitalWrite(front_trigPin, LOW);
58   delayMicroseconds(2);
59   digitalWrite(front_trigPin, HIGH);
60   delayMicroseconds(10);
61   digitalWrite(front_trigPin, LOW);
```

```
62
63
64    front_duration = pulseIn(front_echoPin, HIGH);
65
66    // convert the time into a distance
67    return front_duration/58; //    duration/29/2, return
        centimeters
68  }
69
70  float back_returnDistance(){
71
72    digitalWrite(back_trigPin, LOW);
73    delayMicroseconds(2);
74    digitalWrite(back_trigPin, HIGH);
75    delayMicroseconds(10);
76    digitalWrite(back_trigPin, LOW);
77
78
79    back_duration = pulseIn(back_echoPin, HIGH);
80
81    // convert the time into a distance
82    return back_duration/58; //    duration/29/2, return
        centimeters
83  }
84
85  void forward(){
86   zeroServo();
87   digitalWrite(left_front_motorPin1, HIGH);
88   digitalWrite(left_front_motorPin2, LOW);
89   digitalWrite(right_front_motorPin1, HIGH);
90   digitalWrite(right_front_motorPin2, LOW);
91   digitalWrite(left_back_motorPin1, HIGH);
92   digitalWrite(left_back_motorPin2, LOW);
93   digitalWrite(right_back_motorPin1, HIGH);
94   digitalWrite(right_back_motorPin2, LOW);
```

```
 analogWrite(left_back_motorEna,180);
 analogWrite(left_front_motorEna,180);
 analogWrite(right_back_motorEna,180);
 analogWrite(right_front_motorEna,180);
}

void backward(){
 zeroServo();
 digitalWrite(left_front_motorPin1, LOW);
 digitalWrite(left_front_motorPin2, HIGH);
 digitalWrite(right_front_motorPin1, LOW);
 digitalWrite(right_front_motorPin2, HIGH);
 digitalWrite(left_back_motorPin1, LOW);
 digitalWrite(left_back_motorPin2, HIGH);
 digitalWrite(right_back_motorPin1, LOW);
 digitalWrite(right_back_motorPin2, HIGH);
 analogWrite(left_back_motorEna,150);
 analogWrite(left_front_motorEna,150);
 analogWrite(right_back_motorEna,150);
 analogWrite(right_front_motorEna,150);
}

void brake(){
 zeroServo();
 digitalWrite(left_front_motorPin1, HIGH);
 digitalWrite(left_front_motorPin2, HIGH);
 digitalWrite(right_front_motorPin1, HIGH);
 digitalWrite(right_front_motorPin2, HIGH);
 digitalWrite(left_back_motorPin1, HIGH);
 digitalWrite(left_back_motorPin2, HIGH);
 digitalWrite(right_back_motorPin1, HIGH);
 digitalWrite(right_back_motorPin2, HIGH);
}

void rotateRight(){
```

```
130  zeroServo();
131  digitalWrite(left_front_motorPin1, HIGH);
132  digitalWrite(left_front_motorPin2, LOW);
133  digitalWrite(right_front_motorPin1, LOW);
134  digitalWrite(right_front_motorPin2, HIGH);
135  digitalWrite(left_back_motorPin1, HIGH);
136  digitalWrite(left_back_motorPin2, LOW);
137  digitalWrite(right_back_motorPin1, LOW);
138  digitalWrite(right_back_motorPin2, HIGH);
139  analogWrite(left_back_motorEna,180);
140  analogWrite(left_front_motorEna,200);
141  analogWrite(right_back_motorEna,180);
142  analogWrite(right_front_motorEna,180);
143 }
144
145 void rotateLeft(){
146  zeroServo();
147  digitalWrite(left_front_motorPin1, LOW);
148  digitalWrite(left_front_motorPin2, HIGH);
149  digitalWrite(right_front_motorPin1, HIGH);
150  digitalWrite(right_front_motorPin2, LOW);
151  digitalWrite(left_back_motorPin1, LOW);
152  digitalWrite(left_back_motorPin2, HIGH);
153  digitalWrite(right_back_motorPin1, HIGH);
154  digitalWrite(right_back_motorPin2, LOW);
155  analogWrite(left_back_motorEna,150);
156  analogWrite(left_front_motorEna,150);
157  analogWrite(right_back_motorEna,150);
158  analogWrite(right_front_motorEna,150);
159 }
160
161 void moveRobot(const geometry_msgs::Twist& velocity_msg){
162   int speed = round(velocity_msg.linear.x);
163   int angulo = round(velocity_msg.angular.z);
164   if(angulo<0){
```

```
165     rotateLeft();
166   }
167   else if(angulo>0){
168     rotateRight();
169   }
170   else{
171    if(speed<0){
172     backward();
173   }
174    else if(speed>0){
175     forward();
176   }
177    else{
178     brake();
179   }
180  }
181 }
182
183 ros::NodeHandle  nh;
184
185 tcc::Ultrasonic ultrasonic_msg;
186
187 ros::Publisher pub_distance("/ultrasound", &
      ultrasonic_msg);
188 ros::Subscriber<geometry_msgs::Twist> sub_motors("cmd_vel
      ", moveRobot);
189
190 void zeroServo(){
191   left_front_arm.attach(15);
192   right_front_arm.attach(12);
193   left_back_arm.attach(21);
194   right_back_arm.attach(14);
195
196   left_front_arm.write(90);
197   left_back_arm.write(90);
```

```
198   right_front_arm.write(90);
199   right_back_arm.write(90);
200 }
201
202 void setup(){
203  pinMode(left_front_armPin, OUTPUT);
204  pinMode(left_front_motorPin1, OUTPUT);
205  pinMode(left_front_motorPin2, OUTPUT);
206  pinMode(left_front_motorEna, OUTPUT);
207  pinMode(right_front_armPin, OUTPUT);
208  pinMode(right_front_motorPin1, OUTPUT);
209  pinMode(right_front_motorPin2, OUTPUT);
210  pinMode(right_front_motorEna, OUTPUT);
211  pinMode(left_back_armPin, OUTPUT);
212  pinMode(left_back_motorPin1, OUTPUT);
213  pinMode(left_back_motorPin2, OUTPUT);
214  pinMode(left_back_motorEna, OUTPUT);
215  pinMode(right_back_armPin, OUTPUT);
216  pinMode(right_back_motorPin1, OUTPUT);
217  pinMode(right_back_motorPin2, OUTPUT);
218  pinMode(right_back_motorEna, OUTPUT);
219  pinMode(front_trigPin, OUTPUT);
220  pinMode(front_echoPin, INPUT);
221  pinMode(back_trigPin, OUTPUT);
222  pinMode(back_echoPin, INPUT);
223
224
225   nh.initNode();
226   nh.subscribe(sub_motors);
227   nh.advertise(pub_distance);
228   zeroServo();
229   }
230
231
232  void loop(){
```

```
233    unsigned long currentMillis = millis();
234
235   if (currentMillis>=range_timer + intervalR)
236   {
237     range_timer = currentMillis+intervalR;
238
239     ultrasonic_msg.front_distance = front_returnDistance
          ();
240     ultrasonic_msg.back_distance = back_returnDistance();
241     pub_distance.publish(&ultrasonic_msg);
242     }
243   nh.spinOnce();
244   delay(1);
245  }
```

Listing B.2: auto.py

```
#!/usr/bin/env python3
import rospy
from sensor_msgs.msg import Range
from geometry_msgs.msg import Twist
from tcc.msg import Ultrasonic
import math
import time
import random

def sonar_callback(sonar_data):
    global front_distance
    global back_distance
    front_distance = sonar_data.front_distance
    back_distance = sonar_data.back_distance
    auto_move()

def auto_move():
    velocity_publisher = rospy.Publisher("/cmd_vel",
       Twist, queue_size=10)
```

```
    forward = Twist()
    backward = Twist()
    rotate_right = Twist()
    rotate_left = Twist()
    stop = Twist()

    forward.linear.x=1
    backward.linear.x=-1
    rotate_right.angular.z=1
    rotate_left.angular.z=-1
    stop.linear.x = 0

    velocity_publisher.publish(forward)
    while(front_distance <=30 and front_distance >=0):
        velocity_publisher.publish(stop)
        rospy.Duration(50)
        velocity_publisher.publish(rotate_left)
    else:
        rospy.sleep(10)


if __name__ == '__main__':

    rospy.init_node('auto_nav_node')
    rospy.Subscriber("/ultrasound", Ultrasonic,
       sonar_callback)
    rospy.loginfo("Starting Autonomous Navigation")
    rospy.spin()
```

Listing B.3: tcc_robot_description.urdf

```
<?xml version="1.0" encoding="utf-8"?>
<robot name="tcc_robot_description">

<gazebo>
```

```
  <plugin name="gazebo_ros_control" filename="
     libgazebo_ros_control.so">
  </plugin>
</gazebo>

  <link name="base_link" />

  <link name="chassi">
    <inertial>
      <origin xyz="0.0035597 -0.00013644 0.024468" rpy="
         3.1415 0 0" />
      <mass value="2.5083" />
      <inertia ixx="0.010787" ixy="-1.7688E-06" ixz="
         -2.7506E-06" iyy="0.01833" iyz="2.3896E-06" izz=
         "0.028471" />
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://tcc_robot_description/
           meshes/chassi.dae" />
      </geometry>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://tcc_robot_description/
           meshes/chassi.dae" />
      </geometry>
    </collision>
  </link>

  <joint name="base_link_to_chassi" type="fixed">
    <parent link="base_link"/>
    <child link="chassi"/>
```

```
</joint>

<link name="left_front_arm">
  <inertial>
    <origin xyz="0.01923 -0.00040067 0.0017501" rpy="0
       0 0" />
    <mass value="0.003059" />
    <inertia ixx="6.4546E-08" ixy="5.9255E-09" ixz="
       -9.9141E-09" iyy="3.4881E-07" iyz="2.0657E-10"
       izz="3.4326E-07" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_front_arm.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_front_arm.dae" />
    </geometry>
  </collision>
</link>

<joint name="left_front_arm_joint" type="revolute">
  <origin xyz="0.11621 0.079627 0.03704" rpy="0 0 0" />
  <parent link="chassi" />
  <child link="left_front_arm" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="100
     " velocity="0.5"/>
</joint>
```

```
<link name="left_front_long_arm">
  <inertial>
    <origin xyz="-5.075E-05␣0.045892␣0.0002588" rpy="0␣
       0␣0" />
    <mass value="0.01033" />
    <inertia ixx="6.6757E-06" ixy="7.1446E-09" ixz="
       3.7631E-11" iyy="2.1836E-07" iyz="-3.502E-08"
       izz="6.695E-06" />
  </inertial>
  <visual>
    <origin xyz="0␣0␣0" rpy="0␣0␣0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_front_long_arm.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0␣0␣0" rpy="0␣0␣0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_front_long_arm.dae" />
    </geometry>
  </collision>
</link>

<joint name="left_front_long_arm_joint" type="revolute"
   >
  <origin xyz="0.029993␣-0.00062495␣0.009" rpy="0␣0␣0"
     />
  <parent link="left_front_arm" />
  <child link="left_front_long_arm" />
  <axis xyz="0␣0␣1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
```

```
  <mimic joint="left_front_arm_joint" multiplier="-1"/>
</joint>

<link name="left_front_motor">
  <inertial>
    <origin xyz="0.016457 0.01696 0.0036997" rpy="0 0 0
       " />
    <mass value="0.060115" />
    <inertia ixx="1.8154E-05" ixy="8.4227E-08" ixz="
       -3.0253E-06" iyy="2.3008E-05" iyz="3.0552E-07"
       izz="1.9991E-05" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_front_motor.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_front_motor.dae" />
    </geometry>
  </collision>
</link>

<joint name="left_front_motor_joint" type="revolute">
  <origin xyz="0.11831 0.166 -0.00315" rpy="0 0 0" />
  <parent link="chassi" />
  <child link="left_front_motor" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
```

```
  <mimic joint="left_front_arm_joint" />
</joint>

<link name="left_front_wheel">
  <inertial>
    <origin xyz="8.5521E-10 0.001165 2.329E-11" rpy="0
       0 0" />
    <mass value="0.34934" />
    <inertia ixx="0.00058126" ixy="-3.1106E-12" ixz="
       3.6514E-12" iyy="0.0010764" iyz="9.8911E-14" izz
       ="0.00058126" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_front_wheel.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_front_wheel.dae" />
    </geometry>
  </collision>
</link>

<joint name="left_front_wheel_joint" type="continuous">
  <origin xyz="0.0057 0.0514 0.00014" rpy="0 0 0" />
  <parent link="left_front_motor" />
  <child link="left_front_wheel" />
  <axis xyz="0 1 0" />
</joint>
```

```
<link name="right_front_arm">
  <inertial>
    <origin xyz="0.019232 0.00024039 0.0017501" rpy="0
       0 0" />
    <mass value="0.003059" />
    <inertia ixx="6.4467E-08" ixy="-3.5555E-09" ixz="
       -9.9155E-09" iyy="3.4889E-07" iyz="-1.2393E-10"
       izz="3.4326E-07" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_front_arm.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_front_arm.dae" />
    </geometry>
  </collision>
</link>

<joint name="right_front_arm_joint" type="revolute">
  <origin xyz="0.116209886437567 -0.0793727029687257
     0.0370399999999999" rpy="0 0 0" />
  <parent link="chassi" />
  <child link="right_front_arm" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
</joint>
```

```
<link name="right_front_long_arm">
  <inertial>
    <origin xyz="-5.3063E-05 -0.045892 0.0002588" rpy="
       0 0 0" />
    <mass value="0.01033" />
    <inertia ixx="6.6757E-06" ixy="-7.4625E-09" ixz="
       4.1607E-11" iyy="2.1836E-07" iyz="3.502E-08" izz
       ="6.695E-06" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_front_long_arm.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_front_long_arm.dae" />
    </geometry>
  </collision>
</link>

<joint name="right_front_long_arm_joint" type="revolute
   ">
  <origin xyz="0.029998 0.00037494 0.009" rpy="0 0 0" /
     >
  <parent link="right_front_arm" />
  <child link="right_front_long_arm" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
  <mimic joint="right_front_arm_joint" multiplier="-1"/
```

```
    >
</joint>

<link name="right_front_motor">
  <inertial>
    <origin xyz="0.016245 -0.017067 0.0037085" rpy="0 0
       0" />
    <mass value="0.060166" />
    <inertia ixx="1.8035E-05" ixy="-7.6046E-08" ixz="
       -2.9257E-06" iyy="2.2805E-05" iyz="-2.978E-07"
       izz="1.9931E-05" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_front_motor.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_front_motor.dae" />
    </geometry>
  </collision>
</link>

<joint name="right_front_motor_joint" type="revolute">
  <origin xyz="0.11831 -0.166 -0.00315" rpy="0 0 0" />
  <parent link="chassi" />
  <child link="right_front_motor" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
```

```
  <mimic joint="right_front_arm_joint" />
</joint>

<link name="right_front_wheel">
  <inertial>
    <origin xyz="8.5521E-10 -0.001165 -2.3292E-11" rpy=
       "0 0 0" />
    <mass value="0.34934" />
    <inertia ixx="0.00058126" ixy="3.1107E-12" ixz="
       -3.6514E-12" iyy="0.0010764" iyz="9.8911E-14"
       izz="0.00058126" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_front_wheel.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_front_wheel.dae" />
    </geometry>
  </collision>
</link>

<joint name="right_front_wheel_joint" type="continuous"
   >
  <origin xyz="0.00549999999999998 -0.0514
     0.000540000000000065" rpy="0 0 0" />
  <parent link="right_front_motor" />
  <child link="right_front_wheel" />
  <axis xyz="0 1 0" />
```

```
</joint>

<link name="left_back_arm">
  <inertial>
    <origin xyz="-0.01923 -0.00038461 0.0017501" rpy="0
       0 0" />
    <mass value="0.003059" />
    <inertia ixx="6.4536E-08" ixy="-5.6881E-09" ixz="
       9.9143E-09" iyy="3.4882E-07" iyz="1.9829E-10"
       izz="3.4326E-07" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_back_arm.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_back_arm.dae" />
    </geometry>
  </collision>
</link>

<joint name="left_back_arm_joint" type="revolute">
  <origin xyz="-0.11617 0.079602 0.03704" rpy="0 0 0" /
     >
  <parent link="chassi" />
  <child link="left_back_arm" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
```

```
</joint>

<link name="left_back_long_arm">
  <inertial>
    <origin xyz="-7.1121E-05␣0.045892␣0.0002588" rpy="0
       ␣0␣0" />
    <mass value="0.01033" />
    <inertia ixx="6.6757E-06" ixy="1.0011E-08" ixz="
       5.318E-11" iyy="2.1837E-07" iyz="-3.502E-08" izz
       ="6.695E-06" />
  </inertial>
  <visual>
    <origin xyz="0␣0␣0" rpy="0␣0␣0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_back_long_arm.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0␣0␣0" rpy="0␣0␣0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_back_long_arm.dae" />
    </geometry>
  </collision>
</link>

<joint name="left_back_long_arm_joint" type="revolute">
  <origin xyz="-0.029994␣-0.0005999␣0.009" rpy="0␣0␣0"
     />
  <parent link="left_back_arm" />
  <child link="left_back_long_arm" />
  <axis xyz="0␣0␣1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
```

```
  <mimic joint="left_back_arm_joint" multiplier="-1"/>
</joint>

<link name="left_back_motor">
  <inertial>
    <origin xyz="-0.016245 0.017067 0.0037085" rpy="0 0
       0" />
    <mass value="0.060166" />
    <inertia ixx="1.8035E-05" ixy="-7.6046E-08" ixz="
       2.9257E-06" iyy="2.2805E-05" iyz="2.978E-07" izz
       ="1.9931E-05" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_back_motor.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_back_motor.dae" />
    </geometry>
  </collision>
</link>

<joint name="left_back_motor_joint" type="revolute">
  <origin xyz="-0.11849 0.166 -0.00315" rpy="0 0 0" />
  <parent link="chassi" />
  <child link="left_back_motor" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
```

```
  <mimic joint="left_back_arm_joint" />
</joint>

<link name="left_back_wheel">
  <inertial>
    <origin xyz="-2.329E-11 0.001165 8.5521E-10" rpy="0
       0 0" />
    <mass value="0.34934" />
    <inertia ixx="0.00058126" ixy="-9.891E-14" ixz="
       -3.6514E-12" iyy="0.0010764" iyz="-3.1107E-12"
       izz="0.00058126" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_back_wheel.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/left_back_wheel.dae" />
    </geometry>
  </collision>
</link>

<joint name="left_back_wheel_joint" type="continuous">
  <origin xyz="-0.00550000000000003 0.0514
     0.000540000000000141" rpy="0 0 0" />
  <parent link="left_back_motor" />
  <child link="left_back_wheel" />
  <axis xyz="0 1 0" />
</joint>
```

```
<link name="right_back_arm">
  <inertial>
    <origin xyz="-0.01923 0.00038461 0.0017501" rpy="0
       0 0" />
    <mass value="0.003059" />
    <inertia ixx="6.4536E-08" ixy="5.6881E-09" ixz="
       9.9143E-09" iyy="3.4882E-07" iyz="-1.9829E-10"
       izz="3.4326E-07" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_back_arm.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_back_arm.dae" />
    </geometry>
  </collision>
</link>

<joint name="right_back_arm_joint" type="revolute">
  <origin xyz="-0.11617 -0.079598 0.03704" rpy="0 0 0"
     />
  <parent link="chassi" />
  <child link="right_back_arm" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
</joint>
```

```
<link name="right_back_long_arm">
  <inertial>
    <origin xyz="-7.1171E-05 -0.045892 0.0002588" rpy="
       0 0 0" />
    <mass value="0.01033" />
    <inertia ixx="6.6757E-06" ixy="-1.001E-08" ixz="
       5.5428E-11" iyy="2.1837E-07" iyz="3.502E-08" izz
       ="6.695E-06" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_back_long_arm.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_back_long_arm.dae" />
    </geometry>
  </collision>
</link>

<joint name="right_back_long_arm_joint" type="revolute"
   >
  <origin xyz="-0.029994 0.0005999 0.009" rpy="0 0 0" /
     >
  <parent link="right_back_arm" />
  <child link="right_back_long_arm" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
     1000" velocity="0.5" />
```

```
  <mimic joint="right_back_arm_joint" multiplier="-1"/>
</joint>

<link name="right_back_motor">
  <inertial>
    <origin xyz="-0.016457 -0.01696 0.0036997" rpy="0 0
       0" />
    <mass value="0.060115" />
    <inertia ixx="1.8154E-05" ixy="8.4227E-08" ixz="
       3.0253E-06" iyy="2.3008E-05" iyz="-3.0552E-07"
       izz="1.9991E-05" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_back_motor.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_back_motor.dae" />
    </geometry>
  </collision>
</link>

<joint name="right_back_motor_joint" type="revolute">
  <origin xyz="-0.118490113562428 -0.165997702968727
     -0.00315000000000138" rpy="0 0 0" />
  <parent link="chassi" />
  <child link="right_back_motor" />
  <axis xyz="0 0 1" />
  <limit lower="-0.785398" upper="0.785398" effort="
```

```
      1000" velocity="0.5" />
  <mimic joint="right_back_arm_joint" />
</joint>

<link name="right_back_wheel">
  <inertial>
    <origin xyz="8.5521E-10 -0.001165 -2.3292E-11" rpy=
       "0 0 0" />
    <mass value="0.34934" />
    <inertia ixx="0.00058126" ixy="3.1107E-12" ixz="
       -3.6514E-12" iyy="0.0010764" iyz="9.8911E-14"
       izz="0.00058126" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_back_wheel.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/right_back_wheel.dae" />
    </geometry>
  </collision>
</link>

<joint name="right_back_wheel_joint" type="continuous">
  <origin xyz="-0.0057 -0.0514 0.00014" rpy="0 0 0" />
  <parent link="right_back_motor" />
  <child link="right_back_wheel" />
  <axis xyz="0 1 0" />
</joint>
```

```
<link name="laser">
  <inertial>
    <origin xyz="-0.0069916 7.523E-06 -0.029398" rpy="0
       0 0" />
    <mass value="0.0015917" />
    <inertia ixx="8.0308E-08" ixy="6.8467E-23" ixz="
       -6.3017E-23" iyy="8.0308E-08" iyz="-7.0791E-23"
       izz="4.8073E-09" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/lidar.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
         meshes/lidar.dae" />
    </geometry>
  </collision>
</link>

<joint name="lidar" type="fixed">
  <origin xyz="0.0012639 0.0077901 0.08975" rpy="0 0 0"
      />
  <parent link="chassi" />
  <child link="laser" />
</joint>

<link name="front_sonar">
  <inertial>
```

```
    <origin xyz="-0.0039566 -1.7867E-05 -0.00064317"
        rpy="0 0 0" />
    <mass value="0.0083608" />
    <inertia ixx="2.8756E-06" ixy="3.1812E-10" ixz="
        -1.9099E-07" iyy="1.2074E-06" iyz="3.8706E-09"
        izz="2.0537E-06" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
          meshes/front_sonar.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
          meshes/front_sonar.dae" />
    </geometry>
  </collision>
</link>

<joint name="front_sonar_joint" type="fixed">
  <origin xyz="0.17191 5.7844E-05 0.010412" rpy="0 0 0"
      />
  <parent link="chassi" />
  <child link="front_sonar" />
  <axis xyz="0 0 0" />
</joint>

<link name="back_sonar">
  <inertial>
    <origin xyz="-0.0039566 -1.7867E-05 -0.00064317"
        rpy="0 0 0" />
```

```
    <mass value="0.0083608" />
    <inertia ixx="2.8756E-06" ixy="3.1812E-10" ixz="
      -1.9099E-07" iyy="1.2074E-06" iyz="3.8706E-09"
      izz="2.0537E-06" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
        meshes/back_sonar.dae" />
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://tcc_robot_description/
        meshes/back_sonar.dae" />
    </geometry>
  </collision>
</link>

<joint name="back_sonar" type="fixed">
  <origin xyz="-0.17209 -5.325E-05 0.010412" rpy="0 0
    3.1416" />
  <parent link="chassi" />
  <child link="back_sonar" />
  <axis xyz="0 0 0" />
</joint>

<link name="camera">
  <inertial>
    <origin xyz="-0.0097445 -0.0066011 0.0010967" rpy="
      0 0 0" />
    <mass value="0.010254" />
    <inertia ixx="1.831E-06" ixy="2.5252E-09" ixz="
```

```
        1.5565E-08" iyy="1.1093E-06" iyz="1.683E-10" izz
        ="8.6011E-07" />
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://tcc_robot_description/
            meshes/camera.dae" />
      </geometry>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
        <mesh filename="package://tcc_robot_description/
            meshes/camera.dae" />
      </geometry>
    </collision>
  </link>

  <joint name="camera" type="fixed">
    <origin xyz="0.18195 0.011291 0.047303" rpy="0 0 0" /
        >
    <parent link="chassi" />
    <child link="camera" />
    <axis xyz="0 0 0" />
  </joint>
</robot>
```

Listing B.4: low_control.launch

```
<launch>
  <node pkg="rosserial_python" type="serial_node.py" name
      ="serial_node">
    <param name="port" value="/dev/ttyUSB0"/>
    <param name="baud" value="57600"/>
  </node>
```

```
  <node pkg="tcc" type="auto.py" name="auto_nav_node"/>
  <include file="$(find␣rplidar_ros)/launch/rplidar.
     launch" />
  <include file="$(find␣hector_slam_launch)/launch/
     tutorial.launch">
</launch>
```

APÊNDICE C

A tabela contendo os pinos utilizados e sua função no ESP pode ser vista em 13.

Tabela 13: Pinagem utilizada no ESP

PINO	GPIO	Função
3,3V		
GND		
D15	15	acionar braço esquerdo dianteiro
D2	2	acionar motor esquerdo dianteiro
D4	4	acionar motor esquerdo dianteiro
RX2	16	controlar velocidade do motor esquerdo dianteiro
TX2	17	
D5	5	acionar motor esquerdo traseiro
D18	18	acionar motor esquerdo traseiro
D19	19	controlar velocidade do motor esquerdo traseiro
D21	21	acionar braço esquerdo traseiro
RX0	3	
TX0	1	acionar sensor ultrassônico traseiro
D22	22	acionar sensor ultrassônico frontal
D23	23	
VIN		
GND		
D13	13	controlar velocidade do motor direito traseiro
D12	12	acionar braço direito dianteiro
D14	14	acionar braço direito traseiro
D27	27	acionar motor direito dianteiro
D26	26	acionar motor direito dianteiro
D25	25	controlar velocidade do motor direito dianteiro
D33	33	acionar motor direito traseiro
D32	32	acionar motor direito traseiro
D35	35	
D34	34	
VN	39	receber dados do sensor ultrassônico frontal
VP	36	receber dados do sensor ultrassônico traseiro
EN		

Fonte: Autor