

ROBÔ COM NAVEGAÇÃO AUTÔNOMA, DETECÇÃO VISUAL E MANIPULADOR

Relatório do Projeto Robô Autônomo

Autores:

Anderson Queiroz do Vale
Jéssica Lima Motta
Mateus Santos Cerqueira

Facilitadores:

Lucas Cruz da Silva
Rebeca Tourinho Lima
Tiago Pereira de Souza
Marco Antonio dos Reis

**Salvador
Bahia, Brasil**

Novembro de 2020

Título: Robô com Navegação Autônoma, detecção visual e manipulador	
PROD. TEC. CCRoSA - 001 / 2020	Versão
Classificação: () Confidencial (X) Restrito () Uso Interno () Público	01

Informações Confidenciais - Informações estratégicas para o CCRoSA e Senai Cimatec.

Seu manuseio é restrito a usuários previamente autorizados pelo Gestor da área.

Informações Restritas - Informação cujo conhecimento, manuseio e controle de acesso devem estar limitados a um grupo restrito de pesquisadores que necessitam utilizá-la para exercer suas atividades profissionais.

Informações de Uso Interno - São informações destinadas à utilização interna por pesquisadores e parceiros.

Informações Públicas - Informações que podem ser distribuídas ao público externo, o que, usualmente, é feito através dos canais apropriados.

Dados Internacionais de Catalogação na Publicação (CIP)

Anderson Queiroz do Vale Jéssica Lima Motta Mateus Santos Cerqueira
000 Lucas Cruz da Silva Rebeca Tourinho Lima Tiago Pereira de Souza Marco Antonio dos Reis
Robô com Navegação Autônoma, detecção visual e manipulador Salvador Bahia, Brasil Novembro de 2020
Keywords: 1. Autonomous Navigation . 2. Manipulator. 3. Computer vision. 000

SUMÁRIO EXECUTIVO

O projeto de Robô Autônomo- Desafio 3.0, também conhecido como **Saci** configura-se: sob o Programa de Formação de Novos Talentos do [Centro de Competência de Robótica e Sistemas Autônomos \(CCRoSA\)](#) do Serviço Nacional de Aprendizagem Industrial, Departamento Regional da Bahia - Senai/DR/BA, sendo este o principal fomentador do programa. O presente trabalho tem como impulsionador principal a capacitação de novos pesquisadores preparados para solucionar os mais diversos problemas relacionados a robótica e sistemas autônomos.

O projeto foi considerado como início técnico no dia 04 de Novembro de 2020.

O prazo de execução planejado foi de 30 dias.

RESUMO

O Saci integra o robô da *Clearpath Robotics Warthog* equipado com sensores (câmeras, LiDAR e GPS) e o manipulador robótico JeRoTIMON, com o propósito de transformá-lo em um robô autônomo. Este robô foi construído com o intuito de que o mesmo tivesse navegação autônoma para realizar investigação em ambiente externo e construir um mapa deste, detectasse a bomba escondida nesse ambiente, e realizasse o desarme da bomba através do manipulador. Para a simulação foram utilizados o software *Gazebo* e a ferramenta de visualização *Rviz*, e para o manipulador foi utilizado *MoveIt*. Este robô também possui sua versão real onde foi possível realizar testes e verificar seu desempenho em campo.

ABSTRACT

PUT IN ENGLISH

LISTA DE FIGURAS

Figura 1:	Elos e junta de um manipulador robótico.	18
Figura 2:	Arquitetura geral do sistema.	21
Figura 3:	Configuração D-H do manipulador JeRoTIMON.	24
Figura 4:	<i>Workspace</i> do manipulador no plano X-Y.	25
Figura 5:	<i>Workspace</i> do manipulador no plano X-Z.	25
Figura 6:	<i>Workspace</i> do manipulador no plano Y-Z.	26
Figura 7:	Estrutura analítica do protótipo.	27
Figura 8:	Fluxograma do sistema de escanamento.	28
Figura 9:	Fluxograma do sistema de planejamento e execução de trajetória.	30
Figura 10:	Modelos simulados do manipulador e do painel elétrico.	32
Figura 11:	Imagen capturada pela câmera RGB.	32
Figura 12:	Identificação dos motores.	33
Figura 13:	Base do manipulador.	34
Figura 14:	Vista lateral do <i>link</i> 2.	34
Figura 15:	Vista superior do <i>link</i> 2.	35
Figura 16:	Teledyne Genie Nano C2590.	36
Figura 17:	Lente 16mm C Series VIS-NIR.	36
Figura 18:	Suporte para fixação da câmera.	37
Figura 19:	Integração suporte-câmera-lente.	37
Figura 20:	UWE-12/10-Q12PB-C.	38
Figura 21:	U2D2.	39
Figura 22:	Caixa objetivo.	40
Figura 23:	Manipulador na posição home.	41
Figura 24:	Histograma do conjunto de dados sem detecção da <i>tag</i> .	47
Figura 25:	Boxplot do conjunto de dados sem detecção da <i>tag</i> .	47
Figura 26:	<i>Boxplot</i> da variância do tempo de busca entre os pontos A e B.	50
Figura 27:	<i>Boxplot</i> da variância do tempo total da missão entre os pontos A e B.	51
Figura 28:	Modelo completo teste R&R para tempo de busca.	52
Figura 29:	Modelo completo teste R&R para tempo total da missão.	52
Figura 30:	Gráficos do estudo R&R para tempo de busca.	53
Figura 31:	Árvore de falhas do sistema.	57
Figura 32:	Coordenadas.	75
Figura 33:	Link 0.	75
Figura 34:	Link 1.	76
Figura 35:	Link 2.	77

Figura 36:	Link 3	77
Figura 37:	Link 4	78
Figura 38:	Link 5	79
Figura 39:	Desenho técnico do JeRoTIMON, vistas: frontal, lateral esquerda e superior	84
Figura 40:	Desenho técnico do JeRoTIMON, vista isométrica.	85
Figura 41:	Analise estática - condição de maior esforço exigido.	86
Figura 42:	Analise estática - condição de maior esforço exigido (cargas no centróide).	86

LISTA DE TABELAS

Tabela 1:	Especificações técnicas do manipulador JeRoTIMON.	23
Tabela 2:	Parâmetros D-H para o manipulador JeRoTIMON.	24
Tabela 3:	Parâmetros dos motores.	33
Tabela 4:	Parâmetros dos motores.	38
Tabela 5:	Torque das juntas.	39
Tabela 6:	Ângulos dos motores na posição home.	41
Tabela 7:	Modelo de experimentos para análise de 3 variáveis.	43
Tabela 8:	Modelo de experimentos ANOVA sem detecção da <i>tag</i> .	44
Tabela 9:	Modelo de experimentos ANOVA com detecção da <i>tag</i>	44
Tabela 10:	Resultados das amostras coletadas.	44
Tabela 11:	Resultados da análise de regressão linear.	45
Tabela 12:	Resultados da análise de regressão linear (variável algoritmo).	45
Tabela 13:	Resultados dos experimentos sem detecção da <i>tag</i> .	46
Tabela 14:	Resultados dos experimentos sem detecção da <i>tag</i> .	48
Tabela 15:	Resumo dos conjuntos de dados das variáveis de saída.	50
Tabela 16:	Tabela dos valores de ρ -valor do teste ANOVA.	50
Tabela 17:	<i>FMECA</i> do sub-sistema de potência	55
Tabela 18:	<i>FMECA</i> do sub-sistema aquisição	55
Tabela 19:	<i>FMECA</i> do sub-sistema estrutural	56
Tabela 20:	<i>FMECA</i> do sub-sistema de atuação	56
Tabela 21:	<i>FMECA</i> do sub-sistema de processamento	56
Tabela 22:	Lições aprendidas	59
Tabela 23:	Tabela de suportes	81
Tabela 24:	Especificações do motor <i>Dynamixel HP42-020-S300-R</i>	119
Tabela 25:	Especificações do motor <i>Dynamixel PH54-100-S500-R</i>	123
Tabela 26:	Especificações do motor <i>Dynamixel PH54-200-S500-R</i>	127

LISTA DE SÍMBOLOS E ABREVIATURAS

AGV Automated Guided Vehicle

CAD Computer Aided Design

CCRoSA Centro de Competência de Robótica e Sistemas Autônomos

DoF Degrees of Freedom

FMECA Failure Modes, Effects and Critically Analysis

OMPL Open Motion Planning Library

OpenCV Open Source Computer Vision

ROS Robot Operating System

SOTA Study Of The Art

URDF Unified Robot Description Format

SUMÁRIO

1 INTRODUÇÃO	13
1.1 Objetivos	13
1.2 Justificativa	14
1.3 Organização do relatório	14
2 CONCEITO DO SISTEMA	17
2.1 Parâmetros básicos	17
2.1.1 Requisitos do cliente	17
2.1.2 Requisitos técnicos	17
2.1.3 Estudo do estado da arte	18
3 DESENVOLVIMENTO DO SISTEMA	21
3.1 Descrição do sistema	21
3.1.1 Arquitetura geral	21
3.1.2 Especificação técnica	22
3.1.3 Ambiente de operação	24
3.1.4 Estrutura analítica do protótipo	26
3.2 Especificação funcional	27
3.2.1 Escaneamento	27
3.2.1.1 Descrição	27
3.2.1.2 Premissas necessárias	28
3.2.1.3 Dependências	29
3.2.1.4 Saídas	29
3.2.2 Planejamento e Execução de Trajetória	29
3.2.2.1 Descrição	29
3.2.2.2 Premissas necessárias	30
3.2.2.3 Dependências	30
3.2.2.4 Saídas	31
3.3 Arquitetura de software	31
3.4 Simulação do sistema	31

4 IMPLEMENTAÇÃO	33
4.1 Parametrização dos motores	33
4.2 Estrutura física	33
4.2.1 Base	34
4.2.2 Elos ou <i>links</i>	34
4.2.3 Suportes	35
4.2.4 Câmera	35
4.2.5 Atuadores	37
4.3 Sistema de Potência	38
4.4 Comunicação	38
4.5 Análise de esforços	39
4.6 Configurações	40
5 RESULTADOS E ANÁLISES	43
5.1 Caracterização do problema e determinação do modelo	43
5.2 Planejamento dos experimentos	43
5.3 Resultados alcançados	44
5.3.1 Análise de regressão linear	44
5.3.2 ANOVA	46
5.3.2.1 Análise de variância sem detecção da <i>tag</i>	46
5.3.2.2 Análise de variância com detecção da <i>tag</i>	48
5.3.3 Análise de repetibilidade e reproduzibilidade	51
5.4 Conclusões dos testes estatísticos	54
6 CONFIABILIDADE DO SISTEMA	55
6.1 Análise dos modos e efeitos de falhas	55
6.2 Análise da árvore de falhas	56
7 GESTÃO DO CONHECIMENTO	59
7.1 Lições aprendidas	59
7.2 Guia de uso para simulação	59
7.3 Guia de uso para o modelo real	62
8 CONCLUSÃO	65

REFERÊNCIAS	67
APÊNDICE A Árvores de TF desconectadas	70
APÊNDICE B Árvores de TF conectadas	71
APÊNDICE C Diagrama de nós do sistema	74
APÊNDICE D Propriedades de Massa do JeRoTIMON	75
APÊNDICE E Lista de suportes.	81
APÊNDICE F Projeto mecânico	83
F.1 Desenho mecânico	83
F.2 Analise estática	86
APÊNDICE G Algoritmo de busca e acionamento do painel	89
APÊNDICE H Diagrama elétrico do JeRoTIMON	101
APÊNDICE I Diagrama de conexão do JeRoTIMON	109
ANEXO A Especificações da câmera Dalsa Genie Nano	115
ANEXO B Especificações do motor <i>Dynamixel PH42-020-S300-R</i>	119
ANEXO C Especificações do motor <i>Dynamixel PH54-100-S500-R</i>	123
ANEXO D Especificações do motor <i>Dynamixel PH54-200-S500-R</i>	127

1 INTRODUÇÃO

A robótica é um campo relativamente jovem da tecnologia moderna que atravessa os limites da engenharia tradicional ([SPONG; HUTCHINSON; VIDYASAGAR, 2005](#)). O estudo da robótica é um ramo da tecnologia que engloba área de mecânica, eletrônica e computação, com graus de teoria de controle, microeletrônica, inteligência artificial, fatores humanos e de produção ([PIMENTA, 2009](#)). Segundo ([ERTHAL, 1992](#)), o crescimento da robótica na indústria é justificado em face das exigências de maior qualidade, produtividade e flexibilidade nos processos fabris. Na área industrial, a robótica evoluiu devido ao aumento de uso de robôs e manipuladores industriais.

O estudo do desenvolvimento de manipuladores robóticos foi iniciado por volta de 1954 com George Devol, quando foi desenvolvido o primeiro robô programável e desde então grandes desenvolvimentos nessa área foram atingidos. Como resultado desse avanço, o investimento de empresas foi em torno de 16,5 bilhões de dólares em 2018, chegando a marca de 420 mil unidades enviadas mundialmente, com perspectiva de crescimento médio de 12% ao ano entre 2020 e 2022 ([INDUSTRIAL..., 2019](#)).

Um manipulador robótico é um dispositivo mecânico composto de elementos rígidos (elos) que proporcionam a sustentação e alcance do braço. A inevitabilidade de apresentar algum grau de flexibilidade faz com que esses elos necessitem ser projetados para apresentar elevada rigidez aos esforços que o manipulador será submetido. Esses elos são conectados entre si através de articulações (juntas), que oferecem graus de liberdade ao manipulador e controle de movimento relativo entre os elos. Essas juntas podem ser basicamente divididas em dois grupos: juntas prismáticas e juntas de rotação. Neste projeto foram utilizadas juntas de rotação. A disposição dessas juntas determina a classificação dos manipuladores como Cartesianos, Esféricos, Cilíndricos, entre outros ([ROMANO, 2002](#)).

Este relatório descreve o processo de construção de um manipulador robótico desenvolvido no Centro de Competência em Robótica e Sistemas Autônomos do SENAI CIMATEC e é destinado ao programa de formação Novos Talentos. São descritas as etapas de concepção, simulação, testes e implementação física.

1.1 Objetivos

O propósito deste projeto é construir um manipulador robótico capaz de identificar um marcador visual por meio de uma câmera RGB e proceder com a função de acionar um painel elétrico. Para isso, os objetivos específicos são:

- Realizar estudo do Estado da Arte([SOTA](#)) sobre manipuladores.
- Realizar testes e parametrização dos servomotores.

- Propor um modelo de manipulador robótico.
- Parametrizar o pacote de reconhecimento de marcadores visuais.
- Desenvolver um pacote de configuração do *MoveIt*.
- Desenvolver o código para realizar a missão.
- Realizar a simulação do protótipo do manipulador em software.
- Implementar a versão física do protótipo.
- Realizar testes e estudos estatísticos.

1.2 Justificativa

Apesar da crescente demanda, há falta de profissionais habilitados à desenvolver pesquisas e aplicações na área da robótica. O presente trabalho tem como impulsionador principal a capacitação de novos pesquisadores preparados para solucionar os mais diversos problemas relacionados a robótica e sistemas autônomos.

A busca por melhor eficiência e precisão na realização de atividades em locais que a presença do ser humano torna-se difícil, arriscado e até mesmo impossível, vem se tornando cada vez maior no cotidiano dos ambiente industriais. Além disso, é importante a capacidade do robô de interagir com o ambiente a partir da captura e análise de estímulos visuais ([LEITE, 2005](#)).

Este projeto traz, dentre os benefícios, a utilização de manipuladores robóticos autônomos que sejam capazes de identificar marcadores visuais e realizar tarefas que possam ser perigosas e/ou repetitivas para o ser humano. Espera-se que este projeto seja continuado e que seus resultados sejam compartilhados na comunidade científica, contribuindo para a construção de outros manipuladores com características e/ou objetivos semelhantes.

1.3 Organização do relatório

O presente relatório está organizado em oito capítulos, sendo este de introdução e descrição da justificativa/motivação dos objetivos e da organização do documento.

No capítulo [2](#), Conceito do Sistema, são descritos parâmetros básicos do projeto, dentre eles os requisitos do cliente, requisitos técnicos e o estudo do estado da arte.

O capítulo [3](#), Desenvolvimento do Sistema, apresenta a descrição do sistema onde serão apresentados a arquitetura geral, especificações técnicas, o ambiente de operação do manipulador e a estrutura analítica do protótipo. Além disso, trará as especificações funcionais que compõem o sistema, sua arquitetura de software e o que foi desenvolvido para simulação.

O capítulo 4, Implementação, explica a construção física do manipulador. São expostos seus parâmetros de configuração e sua estrutura.

O capítulo 5, Resultados e Análises, são apresentados os resultados alcançados e a análise dos dados amostrados através de estudos estatísticos.

O capítulo 6, Confiabilidade do Sistema, detalha a análise dos modos e efeitos de falhas.

No capítulo 7, Gestão do Conhecimento, é feito um estudo sobre as lições aprendidas além de apresentar o guia de uso.

Por fim, o capítulo 8 apresenta a conclusão do relatório.

2 CONCEITO DO SISTEMA

A norma técnica (ISO-8373, 2012) criada para padronizar o vocabulário referente aos robôs e dispositivos robóticos operando em ambientes industriais e não industriais, define o manipulador como uma máquina na qual o mecanismo, geralmente, consiste em uma série de segmentos, articulados ou deslizantes entre si, com o objetivo de empunhar e/ou mover objetos (peças ou ferramentas) em vários graus de liberdade. Em outras palavras, um manipulador é um equipamento programável baseado em atuadores, com um certo grau de liberdade, projetado para realizar uma variedade de atividades, assim como realização de diversos processos industriais (ISO-8373, 2012).

Neste capítulo serão tratados os requisitos solicitados pelo cliente, os requisitos técnicos do projeto, o estudo do estado da arte sobre manipuladores e o ambiente de operação em que este manipulador realizará a atividade.

2.1 Parâmetros básicos

Nesta seção encontram-se os requisitos solicitados pelo cliente, ou seja, a tarefa que precisa ser realizada e em qual ambiente o manipulador será simulado e concebido. Além disso, são exibidos os requisitos técnicos que tratam das especificações do sistema e uma breve revisão teórica de conceitos relacionados ao manipulador.

2.1.1 Requisitos do cliente

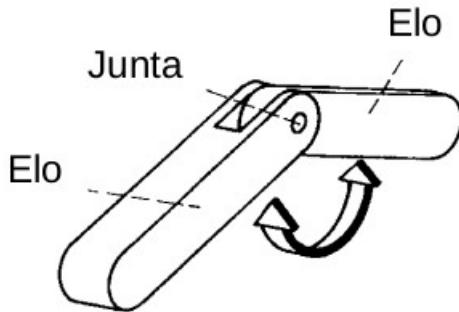
Requisitos predeterminados pelo cliente consistem em exigências de funcionamento que devem ser observadas ao final do projeto, para que se considere um sucesso a concepção deste. Para tal, foram determinadas algumas características desejáveis no projeto:

1. Desenvolver um manipulador robótico.
2. Realizar a tarefa de detecção de um marcador visual e acionamento do painel elétrico na orientação vertical e horizontal.
3. Realizar a simulação do manipulador no ambiente *ROS* utilizando o software *Gazebo*.
4. Utilizar o *framework MoveIt* para o controle do manipulador.
5. Realizar estudo estatístico para verificação da performance do manipulador.

2.1.2 Requisitos técnicos

Os requisitos técnicos de um projeto são especificações necessárias para o funcionamento esperado do projeto. Podem ser sobrepostos aos requisitos do cliente em caso de conflito

Figura 1: Elos e junta de um manipulador robótico.



Fonte: ([SANTOS, 2004](#)).

entre o esperado pelo cliente e o necessário para que o projeto seja bem sucedido, objetivando manter o projeto o mais eficiente dentro do escopo planejado. Os requisitos foram:

1. O manipulador deve estar acoplado em uma base fixa situada a 0,28 m de uma das extremidades da bancada e centralizada com a mesma.
2. Utilizar servo motores Dynamixel PH54-200-S500-R, PH54-100-S500-R e PH42-020-S300-R da Robotis.
3. Conter uma câmera RGB Teledyne Dalsa Genie Nano C2590.
4. Deverá possuir 5 graus de liberdade.
5. Suportar uma carga máxima de 2 kg.
6. Ser capaz de acionar um painel elétrico.

2.1.3 Estudo do estado da arte

Com o advento do exponencial crescimento da tecnologia há um foco crescente na pesquisa e comercialização de robôs ([HERNÁNDEZ-ORDOÑEZ et al., 2018](#)). Estes, por sua vez, são classificados em três grupos: manipuladores, veículos auto-guiados ([AGV](#)) e robôs móveis. Neste projeto, o objeto de interesse são os manipuladores robóticos, sistemas que possuem estrutura física similar a um braço humano. Estes robôs são compostos por partes rígidas, denominado de elos, conectados entre si por juntas ([SANTOS, 2004](#)). Esta estrutura encontra-se descrita na Figura 1.

Muitas pesquisas têm sido realizadas na área de manipuladores robóticos. Em ([HERNANDEZ-MENDEZ et al., 2017](#)) é descrito o desenvolvimento de um manipulador robótico com 3 *DoF* e dois dedos independentes. Este robô foi desenvolvido com o propósito de manipular objetos, cujas localizações são conhecidas, e transportá-los de uma localidade para outra

utilizando *ROS*. Os autores utilizaram o *MoveIt* para tratar do planejamento de trajetória e o *Rviz* como ferramenta de visualização. Além disso, foi desenvolvido um controlador de posição e força para o *endeffecter*¹ durante o processo de *pick and place*². Os experimentos realizados trouxeram bons resultados para o que foi proposto, sendo ressaltada a necessidade de adicionar um sistema de visão que permita identificar e localizar o objeto alvo.

O planejamento de trajetória para um manipulador de 5 *Dof* a partir do *MoveIt* é exibido em (ZHANG; LIN; WU, 2019). A partir de um modelo *CAD* já existente foi construído o modelo *URDF* utilizado para simulação no *ROS*. O processo de planejamento foi visualizado a partir do *Rviz* e o caminho planejado foi transmitido para os servo-motores possibilitando ao manipulador executar a sua rotina. O robô tem como tarefa manipular um objeto alvo de um local para outro, identificado a partir de técnicas de visão computacional que combinam os algoritmos *SIFT* e *RANSAC*. Os resultados experimentais mostraram que o uso do *MoveIt* reduz as dificuldades de operação para manipuladores e oferece vantagens em termos de validação de algoritmos e exploração de funções, sendo possível utilizar o método proposto para o controle em tempo real do robô.

A aplicação de técnicas de visão computacional em um manipulador do tipo *Robai Cyton Gamma 3000* é exibida em (KHAN; KONDA; RYU, 2018). O robô é conectado com uma câmera externa via *ROS*, possui um *endeffecter* em forma de garra que segura uma estrutura similar a um prato, e tem como tarefa equilibrar uma bola localizada no centro do prato. Para realizar o controle da tarefa de equilíbrio, a bola é identificada a partir de um algoritmo escrito em C++ e que utiliza bibliotecas do *OpenCV*. As juntas do manipulador são atuadas a partir de servo-motores *Dynamixel* que são diretamente controlados por um algoritmo. O sistema foi desenvolvido de forma que os componentes se comunicassem entre si para receber respostas do sistema de visão e dos motores, computá-las e enviar comandos de controle que permitissem executar a tarefa. Foram realizados testes e os resultados experimentais foram satisfatórios, sendo o manipulador capaz de equilibrar a bola em uma pequena vizinhança do centro do prato.

O uso de marcadores visuais do tipo *ArUco* associados ao planejamento de trajetória para um manipulador robótico é abordado em (JAVEED; PRAKASH; KULKARNI, 2019). O manipulador encontra-se acoplado a uma plataforma móvel e tem como objetivo o transporte de objetos de forma autônoma em um determinado espaço. Os autores também descrevem a integração dos mecanismos de detecção do marcador visual, navegação do robô e planejamento de trajetória, realizados no *ROS*. O robô é capaz de estimar a pose do marcador, aproximar-se e realizar sua tarefa. Os testes realizados mostraram a

¹ Na robótica, um *endeffecter* é o dispositivo no final de um braço robótico, projetado para interagir com o meio ambiente.

² Sequência de movimentos na qual o manipulador robótico pega determinado objeto e o transfere a uma pose alvo.

eficiência do sistema, sendo apontada a necessidade de um estudo futuro para possibilitar o planejamento de trajetória em um ambiente com obstáculos.

3 DESENVOLVIMENTO DO SISTEMA

Nesta seção serão explicitadas as características do manipulador JeRoTIMON, abordando os sistemas que o compõem em *software* e em *hardware*. Suas funcionalidades principais são abordadas e a conexão entre as mesmas é exibida.

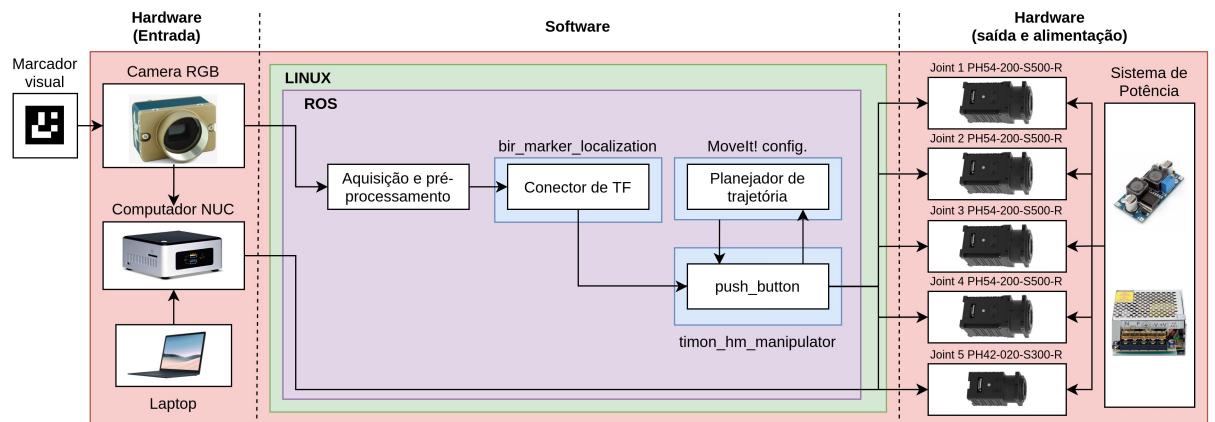
3.1 Descrição do sistema

JeRoTIMON é um manipulador desenvolvido para atender às demandas relacionadas ao reconhecimento de marcadores visuais e, a partir desta identificação, realizar o acionamento de um painel elétrico. Os pacotes que constituem este robô foram concebidos através do *software* de simulação *Gazebo*, da ferramenta de visualização *Rviz* e do *framework*¹ para planejamento de trajetória *MoveIt*. O uso dessas ferramentas possibilita que uma grande variedade de atividades que venham a ser realizadas no mundo real tenham sido previamente testadas em ambiente computacional.

3.1.1 Arquitetura geral

A Figura 2 ilustra a estruturação do sistema e a relação entre *software* e *hardware*. As cores representam o sistema geral(salmão), sistema operacional(verde), *framework*(roxo) e pacotes(azul).

Figura 2: Arquitetura geral do sistema.



Fonte: Autoria própria.

Um laptop, conectado via acesso remoto, dá início a aplicação no computador *NUC*² que possui instalado o software do protótipo. Com o sistema iniciado, a câmera RGB é

¹ São conjuntos de aplicações dentro de um projeto que interagem entre si e com isso se alcança resultados como uma determinada função de um programa.

² Computador pequeno, completo e altamente eficiente energeticamente.

capaz de obter dados visuais do ambiente e enviá-los para o *ROS*. Ao encontrar o marcador visual, o pacote *bir_marker_localization* é capaz de unir as árvores de *TF*³ do painel elétrico e do manipulador, que antes encontravam-se desconectadas. Esta conexão, garante que sejam conhecidos os dados de posição ao nó *push_button* possibilitando o planejamento de trajetória para o ponto desejado. Com o caminho planejado, *push_button* pode enviar os comandos para cada junta do manipulador, onde encontram-se os motores *Dynamixel* que são alimentadas pelo sistema de potência.

3.1.2 Especificação técnica

Na Tabela 1 estão elencadas as especificações técnicas do manipulador robótico JeRoTIMON. O numero de Graus de Liberdade foi definido baseado na capacidade de movimentação necessária para a realização dos desafios propostos. A carga útil, peso e o alcance máximo foram calculados com o auxilio do software *Onshape*, uma alternativa ao cálculo manual. A faixa de operação dos motores foi obtida segundo os limites de segurança observados, o que acrescenta proteção principalmente ao cabeamento do sistema. Informações a respeito de componentes como câmeras e motores seguem o seu padrão original de fabricação.

³ Pacote do *ROS* que permite verificar as relações entre os *frames* na estrutura de árvore.

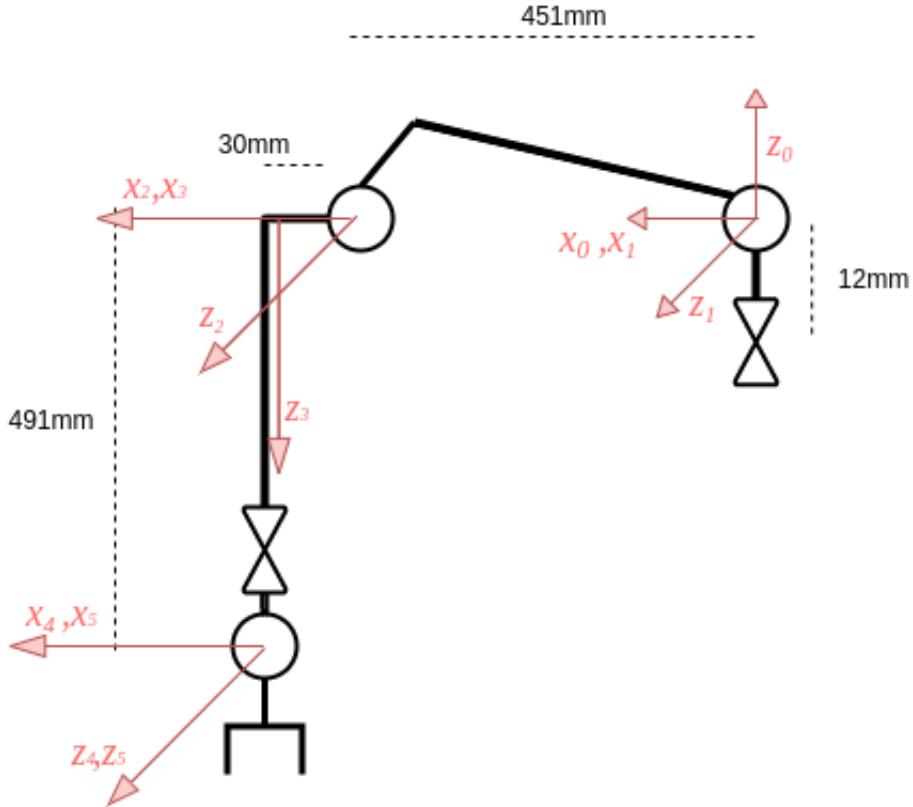
Tabela 1: Especificações técnicas do manipulador JeRoTIMON.

Item	JeRoTIMON
Graus de liberdade	5
Carga útil	2 [kg]
Alcance	981 [mm]
Peso (sem base)	6,4 [kg]
Peso (com base)	10 [kg]
Tensão de operação	24 [V]
Resolução	Junta 1, Junta 2, Junta 3, Junta 4: 1,003,846 [pulsos/rev]
	Junta 5: 607,500 [pulsos/rev]
Motores	Junta 1, Junta 2, Junta 3: PH54-200-S500-R (200 W)
	Junta 4: PH54-100-S500-R (100W)
	Junta 5: PH42-020-S300-R (20 W)
Faixa de operação	Junta 1: $-45^\circ \sim 45^\circ$
	Junta 2: $-90^\circ \sim 90^\circ$
	Junta 3: $-43^\circ \sim 173^\circ$
	Junta 4: $-90^\circ \sim 90^\circ$
	Junta 5: $-90^\circ \sim 90^\circ$
Câmera	Teledyne Genie Nano C2590
Tipo de sensor de posição	Posse inicial: Codificador Absoluto
	Controle: Codificador Incremental
Comunicação	USB
Padrão elétrico	RS485
Taxa de transmissão	57,600 [bps]

Fonte: Autoria própria.

Para estabelecer uma relação entre o *endeffector* e a base do manipulador é necessário descrever o seu sistema de coordenadas em relação ao sistema de coordenadas de origem. Para isto, é utilizada a notação de *Denavit-Hartenger*(D-H). A partir da configuração D-H exibida na Figura 3 foram retirados os parâmetros exibidos na tabela 2.

Figura 3: Configuração D-H do manipulador JeRoTIMON.



Fonte: Autoria própria.

Tabela 2: Parâmetros D-H para o manipulador JeRoTIMON.

Link	a (mm)	$\alpha(^{\circ})$	d (mm)	$\theta(^{\circ})$
1	0	90	12	0
2	452	0	0	$90 - \arctan(30/451)$
3	30	-90	0	$45 + \arctan(30/451)$
4	0	90	491	0
5	0	0	0	0

Fonte: Autoria própria.

3.1.3 Ambiente de operação

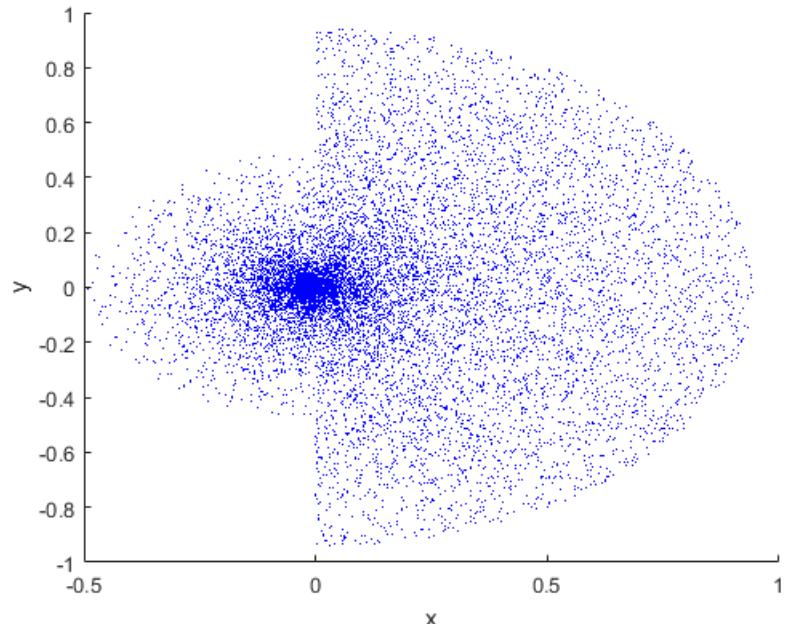
O ambiente físico para a realização do desafio, onde serão incluídos o manipulador e um painel elétrico, é uma mesa com 1.7 m de comprimento por 0.8 m de largura. É necessário então que verifique-se quais os pontos deste ambiente de trabalho que estão dentro da área de alcance (*workspace*)⁴ do manipulador.

A partir da tabela 2 foi desenvolvido um código utilizando o software *Matlab R2020a* capaz de gerar pontos que populem o *workspace* do manipulador nas projeções dos planos

⁴ É a área de alcance do manipulador, região na qual este consegue operar.

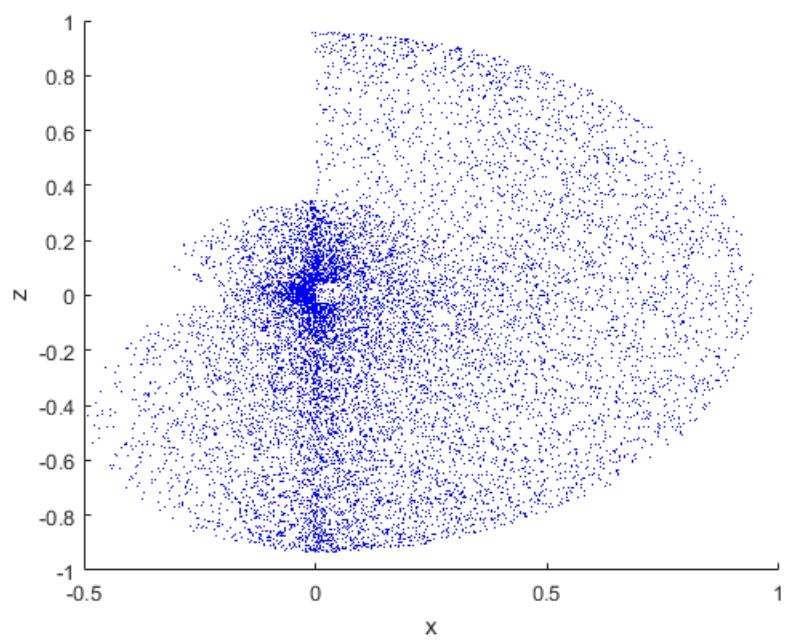
X-Y, X-Z e Y-Z, conforme exibido nas figuras 4, 5 e 6. A região em azul indica o alcance do robô e, a partir destas imagens, é possível observar que o manipulador possui restrições de operação devidas às limitações existentes em cada junta.

Figura 4: *Workspace* do manipulador no plano X-Y.



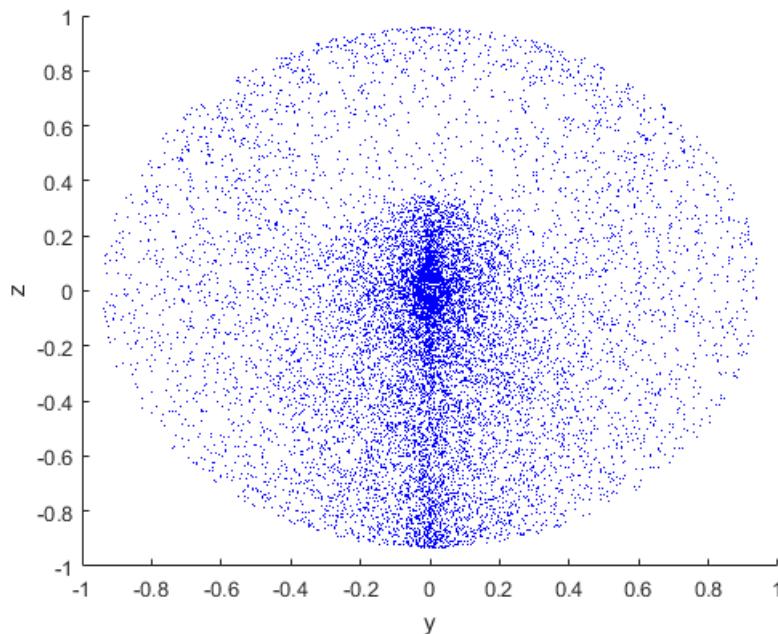
Fonte: Autoria própria.

Figura 5: *Workspace* do manipulador no plano X-Z.



Fonte: Autoria própria.

Figura 6: *Workspace* do manipulador no plano Y-Z.

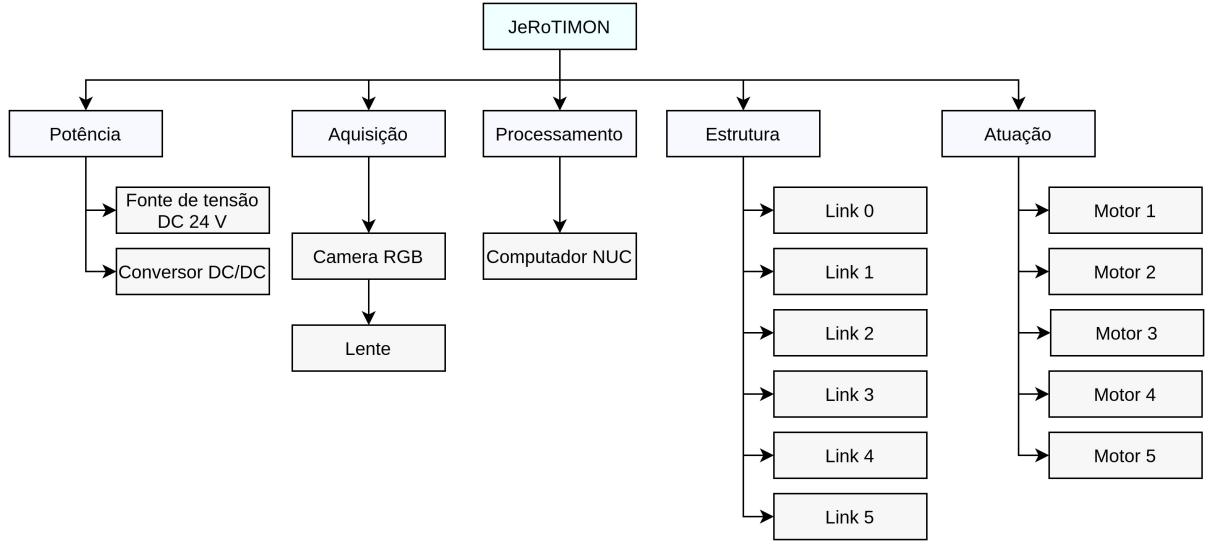


Fonte: Autoria própria.

3.1.4 Estrutura analítica do protótipo

A estrutura analítica do protótipo mostrada na Figura 7 exibe as relações sistemáticas entre as partes que compõem o manipulador. A estrutura hierárquica possui três níveis: o primeiro, referente ao sistema principal JeRoTIMON; o segundo, que é composto pelos sub-sistemas de potência, aquisição, processamento, estrutura e atuação; e o terceiro, composto pelos itens que fazem parte de cada um destes sub-sistemas.

Figura 7: Estrutura analítica do protótipo.



Fonte: Autoria própria.

3.2 Especificação funcional

O manipulador descrito trabalha acionando os botões encontrados pelo sistema de aquisição. Seu software funciona baseado na troca de mensagens entre duas funcionalidades principais: Escaneamento e Planejamento/Execução de trajetória. O Escaneamento corresponde à detecção de um marcador visual, que uma vez detectado, informa a posição no espaço de um painel elétrico que precisa ser acionado. O planejamento e execução de trajetória utiliza cálculos de cinemática direta e inversa para definir a trajetória de movimentação que permitirá ao manipulador realizar sua tarefa.

3.2.1 Escaneamento

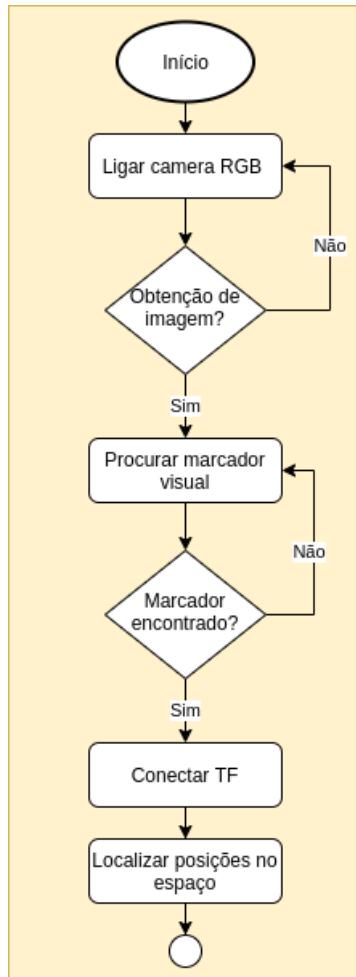
Uma câmera RGB *Teledyne Genie Nano C2590* equipada com lente *kowa LM8FC* foi acoplada ao manipulador JeRoTIMON. Através da mesma é realizada a detecção do marcador visual, utilizando a biblioteca *ArUco* e o pacote *Bir Marker Localization*, hospedado no site do Github no perfil do BIR - Brazilian Institute of Robotics ([BIR... , 2020](#)).

3.2.1.1 Descrição

A Figura 8 exibe o fluxograma que descreve o funcionamento do sistema de escaneamento integrado ao manipulador. Após a captura da imagem, a partir da câmera RGB, é feito um processamento dos dados obtidos afim de localização da *tag ArUco*, cujos tamanho e ID foram previamente estabelecidos. Caso o marcador seja encontrado, as árvores de *TFs* do painel elétrico onde encontra-se o marcador, e do manipulador robótico são conectadas,

possibilitando a localização no espaço da pose alvo. Os apêndices A e B exibem as árvores de *TFs* antes e após a conexão realizada.

Figura 8: Fluxograma do sistema de escanamento.



Fonte: Autoria própria.

3.2.1.2 Premissas necessárias

- Conter um marcador visual anexado ao painel elétrico.
- A orientação dos elementos envolvidos no escaneamento, câmera RGB e marcador visual, devem ser definidos conforme estabelecido pelo pacote *bir_marker_localization*.
- Não haver oclusão do marcador visual.
- Nível de iluminação do ambiente suficiente para a execução da detecção.
- O marcador visual deverá possuir tamanho adequado para detecção.

3.2.1.3 Dependências

Para realizar a etapa de detecção é necessária a instalação do *OpenCV* versão 3.3.1, *driver GigE-V Framework* e a inserção dos pacotes *bir_marker_localization* e *def_cam_tedelyne_nano* no *workspace* do manipulador.

3.2.1.4 Saídas

É fornecida ao sistema resposta por meio de uma sequência de mensagens publicadas no tópico */timon/camera/image_raw*. Estes dados são analisados pelo detector *ArUco*, possibilitando seu uso em um pacote desenvolvido na linguagem C++ que determina a posição do painel elétrico associado ao marcador visual.

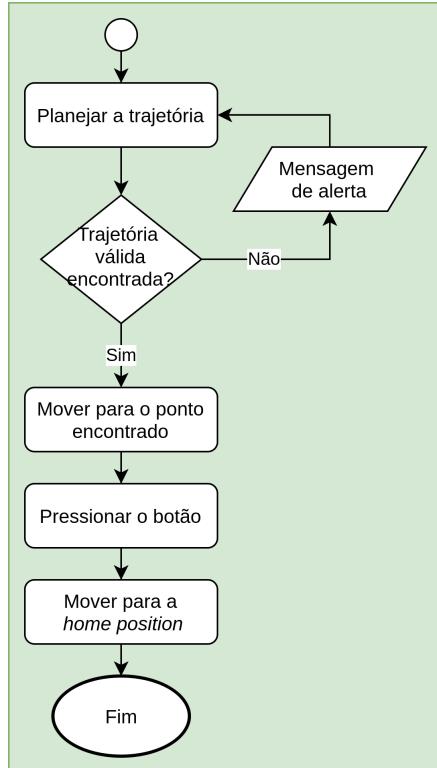
3.2.2 Planejamento e Execução de Trajetória

As equações cinemáticas são a base que possibilitam a pesquisa do movimento dos manipuladores. A cinemática inversa provê um conjunto de valores para as juntas do manipulador com o intuito de alcançar uma determinada pose pré-estabelecida do seu *endeffector*. Para resolver as equações da cinemática inversa do JeRoTIMON, optou-se por utilizar o plugin TRAC-IK, um método alternativo ao padrão da inversa Jacobiana utilizado pelo *MoveIt*. Este método se adequa bem a manipuladores que possuem limitações em suas juntas, ao contrário de algoritmos baseados no teorema de Newton ([BEESON; AMES, 2015](#)). Para o planejamento de trajetória foi utilizada a biblioteca *OMPL*, uma coleção de algoritmos de planejamento utilizada por padrão no *MoveIt* ([SUCAN; MOLL; KAVRAKI, 2012](#)).

3.2.2.1 Descrição

A Figura 9 exibe o funcionamento do sistema de planejamento e execução de trajetória aplicado ao manipulador. A pose do painel elétrico, determinada a partir do que foi mostrado em 3.2.1, é enviada como entrada para o *MoveIt*. Caso seja possível, é realizado o planejamento de uma trajetória para cada uma das juntas do manipulador, a fim de movê-lo para a pose desejada. Esta trajetória é então enviada para os atuadores das juntas, que passam a executá-la. Após a realização da rotina para o pressionamento do painel elétrico, o manipulador retorna para sua posição inicial. Caso alguma condição impeça o planejamento de trajetória, como por exemplo o posicionamento do painel elétrico fora da área de trabalho do manipulador ou falhas nas soluções para as equações da cinemática inversa, uma mensagem de alerta é exibida e o robô realiza uma nova tentativa de planejamento.

Figura 9: Fluxograma do sistema de planejamento e execução de trajetória.



Fonte: Autoria própria.

3.2.2.2 Premissas necessárias

- Viabilidade das soluções para cinemática inversa.
- Painel elétrico estar posicionado na área de trabalho do manipulador.

3.2.2.3 Dependências

O sistema de movimentação é dependente da versão Melodic Morenia do *framework ROS* e da plataforma *MoveIt*. Além destes, uma lista de pacotes deve ser instalada previamente para o funcionamento correto do sistema:

- ros-melodic-ros-control
- ros-melodic-gazebo-ros-control
- ros-melodic-controller-manager
- ros-melodic-joint-trajectory-controller
- ros-melodic-joint-state-controller
- ros-melodic-position-controllers

- ros-melodic-trac-ik-kinematics-plugin

3.2.2.4 Saídas

As respostas fornecidas pelo sistema de planejamento e execução são publicadas nos motores *Dynamixel* integrados às juntas do manipulador. A trajetória gerada pelo *MoveIt* pôde ser visualizada a partir do tópico */move_group/display_planned_path* e é publicada nos motores a partir do tópico */timon_arm_controller/dynamixel_state*.

3.3 Arquitetura de software

O robô JeRoTIMON foi desenvolvido para atuar em conjunto com o *ROS*, isto é, segue o propósito de conectar diferentes módulos, como câmeras, motores, sensores e códigos. A proposta é conectar um programa de visão capaz de conectar árvores de TF, através da identificação de marcadores visuais, com um sistema de movimentação que posiciona o manipulador para acionar o painel elétrico.

O apêndice C exibe o diagrama do *software* do sistema, obtido via *rqt_graph*. Observa-se a comunicação entre o manipulador (*/timon/robot_state_publisher*) e o painel elétrico (*/box/box_state_publisher*) a partir da conexão das árvores de *TFs* realizadas pelo */marker_localization* que recebe e analisa os dados da câmera (*/timon/camera_image_raw*). Pode-se verificar também que o nó */arm* comunica-se com o nó */move_group* e com o nó */timon_arm_controller* a fim de enviar a trajetória planejada pelo *MoveIt* para os motores *dynamixel*.

3.4 Simulação do sistema

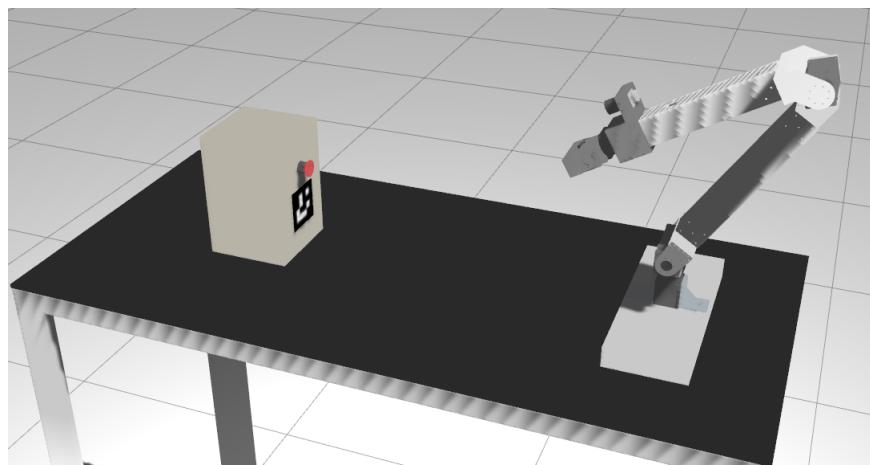
A simulação do robô JeRoTIMON inserido em seu ambiente de trabalho foi realizada na plataforma *Gazebo*. Para isto, realizou-se o desenvolvimento dos arquivos *timon_arm.urdf* e *box.urdf*, modelos *URDF* que descrevem o robô e o painel elétrico a ser acionado. Foram levados em consideração características de massa, inércia e dimensão para cada componente destes modelos, para que houvesse maior fidelidade possível com o que representam fisicamente, de forma a garantir que os testes realizados possam ser validados em ambiente real.

O pacote *ros_control* dispõe de uma lista de controladores disponíveis. Para JeRoTIMON, foram utilizados controladores do tipo *position_controllers/JointTrajectoryController*, e as transmissões para cada junta do manipulador foram definidas em seu modelo *URDF*.

Para simulação dâ câmera RGB, foi desenvolvido o arquivo *camera.xacro*. O plugin padrão do *Gazebo* foi utilizado, *ligazebo_ros_camera.so* e as especificações da câmera RGB *Teledyne Genie Nano C2590* foram levados em consideração, garantindo maior realismo à simulação. A figura 10 exibe os modelos simulados do manipulador e do painel elétrico,

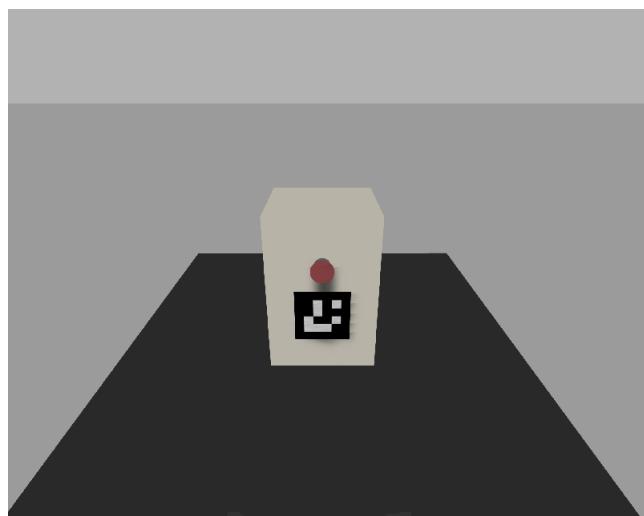
enquanto a figura 11 exibe imagem capturada pela câmera instalada no manipulador.

Figura 10: Modelos simulados do manipulador e do painel elétrico.



Fonte: Autoria própria.

Figura 11: Imagem capturada pela câmera RGB.



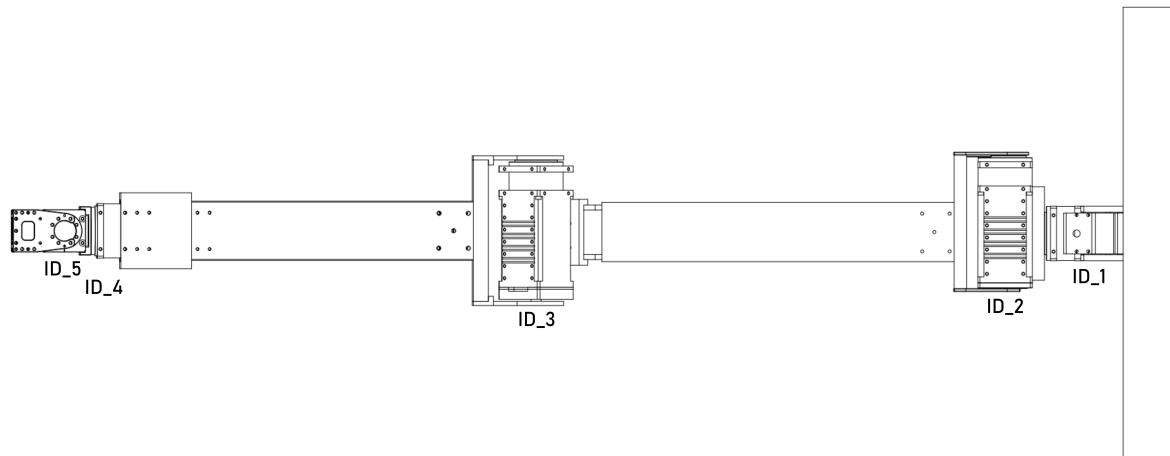
Fonte: Autoria própria.

4 IMPLEMENTAÇÃO

4.1 Parametrização dos motores

Na Tabela 3 encontram-se as configurações para os motores, que estão identificados na Figura 12. Nesta Tabela estão especificados os ângulos e as posições mínimas e máximas definidas para cada motor. Estes parâmetros foram escolhidos após testes e verificações de acordo com a capacidade do manipulador interagir com o ambiente de trabalho, pensando nas possíveis localizações que a caixa poderá estar situada.

Figura 12: Identificação dos motores.



Fonte: Autoria própria.

Tabela 3: Parâmetros dos motores.

Motor	Ângulo (min)	Posição (min)	Ângulo (máx)	Posição (máx)
ID_1	-45°	-125000	45°	125000
ID_2	-90°	-250962	90°	250962
ID_3	-43°	-119095	173°	483855
ID_4	-170°	-475464	170°	475464
ID_5	-90°	-151875	90°	151875

Fonte: Autoria própria.

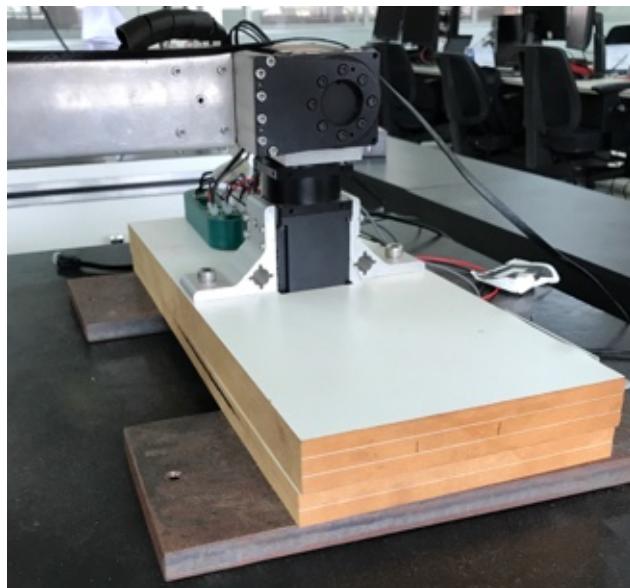
4.2 Estrutura física

Após montagem física, o manipulador obteve um alcance de aproximadamente 980 mm. Levou-se em consideração o mínimo alcance necessário para que a missão possa ser realizada conforme as especificações do cliente. Então foram projetadas cinco juntas rotacionais como mostra a Figura 12.

4.2.1 Base

Foi decidido utilizar uma base de madeira com dimensões de aproximadamente 450×180 mm e espessura de 50 mm (Figura 13). Esta base será utilizada para fixar o manipulador, garantindo estabilidade durante a execução da tarefa.

Figura 13: Base do manipulador.



Fonte: Autoria própria.

4.2.2 Elos ou *links*

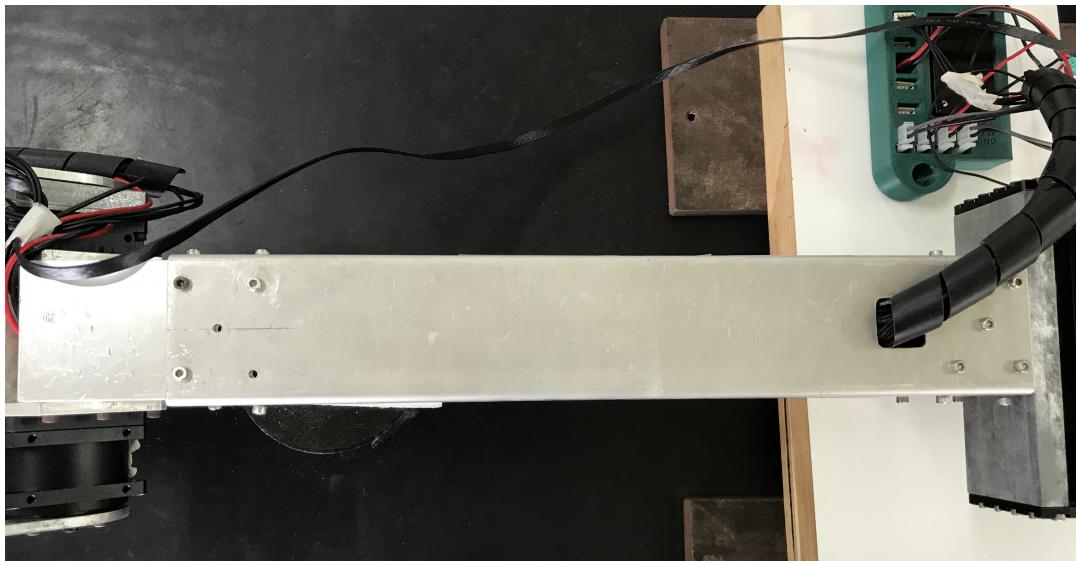
Para estruturação dos elos do robô são utilizados dois perfis de alumínio vazados com comprimento, largura e altura de $350 \times 58.7 \times 58.7$ mm, respectivamente (Figuras 14 e 15). Estas dimensões foram escolhidas levando em consideração a disponibilidade dos materiais, capacidade de alcance do braço para realizar a tarefa e capacidade estrutural do manipulador, para assim suportar esforços de natureza estática e dinâmica.

Figura 14: Vista lateral do *link* 2.



Fonte: Autoria própria

Figura 15: Vista superior do *link* 2.



Fonte: Autoria própria

4.2.3 Suportes

Foram utilizados suportes com finalidade de unir os elos do manipulador e distribuir movimentos rotativos para as juntas seguintes, feitos em alumínio. Alguns suportes foram adquiridos através da ROBOTIS, enquanto outros foram confeccionados pelo laboratório CCRoSA. A Tabela com os suportes utilizados e suas imagens e medidas pode ser vista no apêndice E.

4.2.4 Câmera

Para prover o sistema com capacidade de detecção da *tag* e assim obter dados necessários para aquisição de pose e orientação relativa, utilizou-se uma câmera de vídeo modelo Teledyne Genie Nano C2590 (Figura 16) e foi acoplada a lente 16mm C Series VIS-NIR (Figura 17). A comunicação entre a câmera e o sistema é feita através de um cabo categoria 6 RJ45, e sua ligação pode ser vista no apêndice I.

Figura 16: Teledyne Genie Nano C2590.



Fonte: ([TELEDYNE...](#), s.d.)

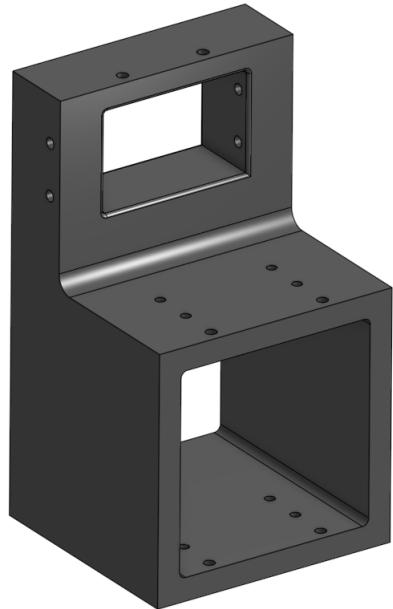
Figura 17: Lente 16mm C Series VIS-NIR.



Fonte: ([16MM...](#), s.d.)

Este conjunto pode ser fixado próxima a extremidade do manipulador robótico utilizando um suporte impresso em ABS (Figura 18), facilitando a detecção da *tag* e não comprometendo a estrutura do manipulador. O suporte é conectado ao final do *link 3*, como mostra o apêndice D. A integração pode ser visualizada na Figura 19.

Figura 18: Suporte para fixação da câmera.



Fonte: Autoria própria

Figura 19: Integração suporte-câmera-lente.



Fonte: Autoria própria

4.2.5 Atuadores

Os atuadores do manipulador são motores de corrente contínua, integrados com redutor de velocidade, controlador e driver. Foram utilizados atuadores *Dynamixel* da fabricante

ROBOTIS. Entre os modelos figuram o PH54-200-S500-R, presente nas juntas 0, 1 e 2, o motor PH54-100-S500-R para a junta 3 e o motor PH42-020-S300-R foi utilizado para a junta 4. As especificações do fabricante mais relevantes no projeto estão apresentadas nas tabelas 4 e 5. As folhas de dados estão disponíveis nos anexos B, C e D para os atuadores PH42-020-S300-R, PH54-100-S500-R, PH54-200-S500-R, respectivamente.

Tabela 4: Parâmetros dos motores.

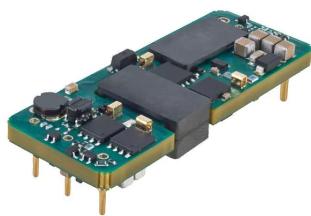
Tipo	Torque (N.m)	Tensão (V)	Corrente (A)	Velocidade de Rotação (rpm)	Dimensões (mm)
PH54-200-S500-R	44.7	24.0	9.3	29.0	54.0 X 126.0 X 54.0
PH54-100-S500-R	25.3	24.0	5.5	29.2	54.0 X 108.0 X 54.0
PH42-020-S300-R	5.1	24.0	1.5	29.2	42.0 X 84.0 X 42.0

Fonte: ([ROBOTIS, s.d.](#))

4.3 Sistema de Potência

O sistema necessitará ser energizado em dois níveis de tensão, 12 V e 24 V. Durante a realização dos testes, utilizou-se uma fonte com níveis de tensão de 0-30 V e fornecimento de corrente de 0-10 A, para atender os requisitos necessários de potência dos motores. Foi utilizado um conversor DC-DC modelo UWE-12/10-Q12PB-C (Figura 20) na saída da fonte para atingir o nível de tensão de 12 V e energizar a câmera. Um esquema elétrico é fornecido no apêndice H, indicando as conexões dos cabos entre os motores e a câmera.

Figura 20: UWE-12/10-Q12PB-C.



Fonte: ([UWE-12/10-Q12PB-C, s.d.](#))

4.4 Comunicação

O manipulador é conectado via USB através de um dispositivo denominado U2D2 ([U2D2, s.d.](#)). Ele consegue parametrizar e enviar comandos para os motores através dos softwares originais da ROBOTIS chamados Dynamixel Workbench e Dynamixel Wizard. O padrão elétrico utilizado é RS-485, ou seja, utiliza-se de 4 fios de conexão que transmitem níveis de tensão e dados. A conexão de entre o computador e os motores, utilizando o U2D2 de

intermédio, pode ser vista no apêndice I. A velocidade de transmissão (*baud rate*) definida para todos os motores no sistema é de 57600 bps.

Figura 21: U2D2.



Fonte: ([U2D2, s.d.](#))

4.5 Análise de esforços

A análise de esforços mecânicos aos quais o manipulador está exposto foi realizada para embasamento e confirmação da capacidade da estrutura e dos atuadores suportarem os esforços de momento torsor. Para isso foram analisadas individualmente as juntas em seus estados críticos. Na Tabela 5 é apresentada a análise de esforços nas juntas e comparados com os máximos esforços suportados pelos atuadores (conforme vistos nos apêndices B, C e D), para evitar falhas mecânicas.

Levando em consideração os valores obtidos nas análises de esforços, tem-se as juntas 0 e 1 como juntas críticas do sistema, pois elas sofrem a maior força do sistema e tem o maior torque exigido. Na configuração apresentada, as juntas 0 e 1 suportam a carga máxima de 45.22 N sem chegar a falha mecânica, logo o manipulador possui um limite (*payload*) de aproximadamente 2.1 kg na sua extremidade.

Tabela 5: Torque das juntas.

Junta	Torque máximo fornecido pelos motores (Nm)	Força resultante na junta (N)	Torque exigido pela junta (Nm)
0	44.7	45.22	25.8
1	44.7	45.22	25.8
2	44.7	21.44	6.9
3	25.3	4.8	0.12
4	5.1	0.87	0.05

Fonte: Autoria própria.

4.6 Configurações

O conjunto formado pelo manipulador, seu espaço de trabalho e os objetos com os quais ele deverá interagir compõem o sistema abordado neste trabalho. O espaço de trabalho foi representado como uma bancada de $1.70\text{ m} \times 0.80\text{ m} \times 0.028\text{ m}$. Há apenas um objeto com o qual o manipulador deverá interagir, uma caixa de dimensões $0.30\text{ m} \times 0.20\text{ m} \times 0.20\text{ m}$.

Para o desafio foi utilizado um marcador visual *ArUco* com dimensões $58\times 58\text{ mm}$ de id 4, conforme mostrado na Figura 22. Este tem 8 cm do centro do *ArUco* para o botão e 7 cm do centro do *ArUco* para a lâmpada, esta identifica se o botão foi acionado ou não pelo manipulador.

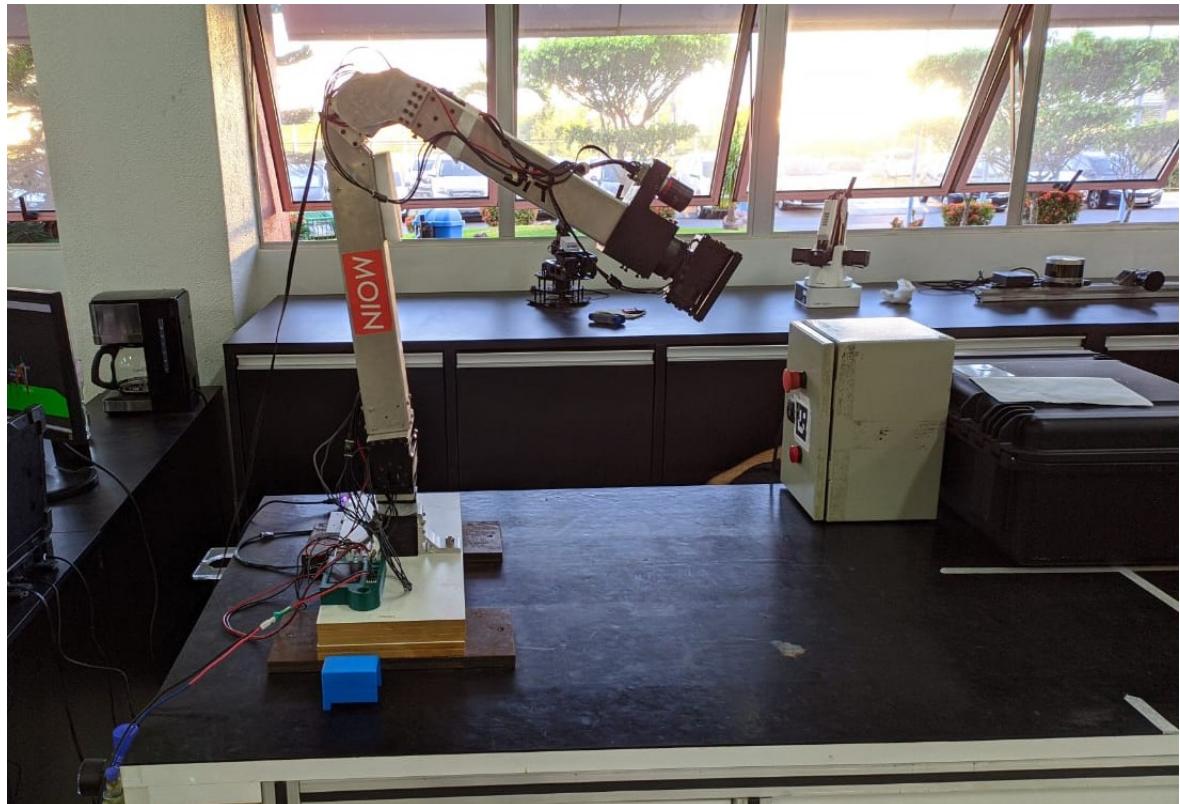
Figura 22: Caixa objetivo.



Fonte: Autoria própria

O manipulador possui a posição home definida como da Figura 23 onde o ângulo de cada motor está descrito na Tabela 6.

Figura 23: Manipulador na posição home.



Fonte: Autoria própria.

Tabela 6: Ângulos dos motores na posição home.

Motor	Ângulo
ID_1	0°
ID_2	-24°
ID_3	141°
ID_4	0°
ID_5	0°

Fonte: Autoria própria.

5 RESULTADOS E ANÁLISES

Os experimentos apresentados nesta seção possuem como objetivo avaliar o desempenho do manipulador robótico desenvolvido.

5.1 Caracterização do problema e determinação do modelo

Para análise da performance do manipulador, deseja-se avaliar inicialmente a interação de algumas variáveis que podem influenciar no desempenho do manipulador. Em um segundo momento, após estabelecidas as melhores configurações de algoritmo e velocidade para o robô, os testes devem seguir uma análise de repetibilidade e de variância do sistema.

No planejamento dos experimentos, foram levantadas quais variáveis seriam de interesse para avaliar a performance do robô, com isso foram selecionadas como variáveis de saída a precisão do manipulador, ou seja, a capacidade do mesmo de chegar a uma posição determinada com o menor erro possível, tempo de busca do marcador visual, e o tempo necessário para o acionamento do botão (missão imposta ao manipulador).

Também foram selecionadas as variáveis independentes, ou seja, as variáveis de entrada do processo, que serão analisadas quanto às suas influências no sistema de forma isolada ou a interação entre as mesmas. As variáveis selecionadas para o estudo foram: velocidade de operação dos motores, algoritmo de planejamento de trajetória utilizado e a posição da caixa no ambiente de trabalho do manipulador.

5.2 Planejamento dos experimentos

Definido o problema e as variáveis a serem estudadas, foi realizado um planejamento dos experimentos, para assim gerar dados e resultados confiáveis acerca da melhor configuração para o manipulador. As tabelas 7, 8 e 9 apresentam os modelos dos experimentos.

Tabela 7: Modelo de experimentos para análise de 3 variáveis.

Variáveis Independentes	Níveis		Variáveis dependentes
Algoritmo de planejamento de trajetória	RRT-CONNECT	Kpiece	Erro de posição
Velocidade dos motores	300	500	Tempo total
Posição da caixa	A	B	

Fonte: Autoria própria.

Tabela 8: Modelo de experimentos ANOVA sem detecção da *tag*.

Ponto	Número de repetições	Variáveis dependentes
A	10	Erro de posição
B	10	Erro de posição

Fonte: Autoria própria.

Tabela 9: Modelo de experimentos ANOVA com detecção da *tag*

Operador	Pose da caixa	Número de regravações	Sucesso	Variáveis dependentes
1	A	10	-1,1	Erro de posição
2	B	10	-1,1	Tempo de busca
1	B	10	-1,1	Tempo total
2	A	10	-1,1	

Fonte: Autoria própria.

5.3 Resultados alcançados

A partir dos testes realizados sob orientação do planejamento, foram coletados os resultados do tempo de busca do marcador visual, tempo da missão do manipulador e o erro de posição em cada experimento. As amostras dos testes coletadas foram analisados da seguinte forma: Análise de regressão linear, análise de variância e teste R&R, para os testes com e sem a detecção do marcador visual .

5.3.1 Análise de regressão linear

A partir do planejamento de experimentos realizados, foram tomados três dados de cada condição estabelecida e utilizada a média entre os três resultados como valor do experimento. As amostras coletadas estão representadas na tabela 10. O algoritmo de planejamento RRT CONNECT é representado pelo sinal de subtração (-) enquanto o sinal de adição (+) representa o algoritmo Kpiece. Já para as velocidades, (+) representa o valor de 500 rpm e (-) representa 300 rpm.

Tabela 10: Resultados das amostras coletadas.

Ordem	Algoritmo	Velocidade	Erro	Tempo
1	-	-	0.0004759353	175.25667
2	+	-	0.0009316073	82.24667
3	-	+	0.0002889380	139.39333
4	+	+	0.0008311057	90.34333

Fonte: Autoria própria.

O resultado obtido a partir da análise linear aplicada aos dados é disposto na tabela 11. O primeiro modelo dessa análise pode ser visualizado na Equação 5.1. Esta equação

mostra um modelo que consegue explicar o valor do erro de posição a partir da relação entre os parâmetros.

Tabela 11: Resultados da análise de regressão linear.

	Estimate Std.	Error	t value	Pr(> t)
(Intercept)	4.543e-04	3.745e-05	12.130	0.0524
algoritmo+	4.989e-04	4.325e-05	11.536	0.0550
velocidade+	-1.437e-04	4.325e-05	-3.324	0.1860

Fonte: Autoria própria.

$$erro = 4.543.(10^{-04}) + 4.989.(10^{-04}).\text{algoritmo}(+) - 1.437.(10^{-04}).\text{velocidade}(+) \quad (5.1)$$

Observando os dados apresentados na tabela 11, o ρ -valor da variável velocidade é superior a 0,05, aceitando a hipótese nula, ou seja, a variável velocidade possui pouca ou nenhuma influência na tomada de dados do sistema. Porém, percebe-se que o ρ -valor da variável algoritmo é próxima de 0.05, implicando que a mesma pode ser analisada de forma individual, rejeitando-se a hipótese nula para esta variável. Desta forma refazendo-se a análise linear do erro de posição com relação apenas a variável algoritmo obtém-se o novo modelo representado na equação 5.2. O resultado dos coeficientes pode ser visto na tabela 12.

Tabela 12: Resultados da análise de regressão linear (variável algoritmo).

	Estimate Std.	Error	t value	Pr(> t)
(Intercept)	3.824e-04	7.506e-05	5.095	0.0364
algoritmo+	4.989e-04	1.062e-04	4.700	0.0424

Fonte: Autoria própria.

$$erro = 3.824.(10^{-04}) + 4.989.(10^{-04}).\text{algoritmo}(+) \quad (5.2)$$

Como apresentado na tabela 12 o algoritmo apresenta ρ -valor menor que 0.05, ou seja, rejeita-se a hipótese nula para esta variável, mostrando que a mesma tem influência no erro de posição do manipulador.

Após esta análise serão realizados novos testes levando em consideração as variáveis examinadas neste estudo, sendo configurado o sistema com o algoritmo RRT CONNECT, uma vez que o algoritmo Kpiece teve influência positiva, ou seja, aumentou o erro de posição do manipulador, o que não é desejável para este sistema. A velocidade dos motores não apresentou influência considerável nos testes, portanto foi configurada com a menor velocidade por motivos de segurança.

5.3.2 ANOVA

Com o algoritmo de planejamento e velocidade estabelecidas para a continuidade dos testes, foram realizados dois tipos de experimentos: análise de erro de posição do *end-effector* sem a detecção do marcador visual, ou seja, o manipulador recebe comandos de posição em x, y e z dentro da sua área de trabalho e então é observado o erro ao executar o comando de posição. O segundo teste foi semelhante, porém, este contou com a detecção da *tag*, recebendo assim o comando de posição a partir da detecção do marcador visual. Com isso foram tomados dados em duas diferentes poses para assim analisar a variância do erro em cada pose e entre elas.

5.3.2.1 Análise de variância sem detecção da *tag*

Para esta análise foram estabelecidas, na área de trabalho do manipulador, duas posições, onde Pose A equilava a $x = 0.314445$ $y = 0.730867$ $z = 0.233471$ e Pose B equilava a $x = -0.153158$ $y = 0.755479$ $z = 0.238906$, ambas referentes ao *endeffecter* do manipulador, e então foram realizados 10 execuções de movimento para cada pose estabelecida. A tabela 13 mostra os erros de posição do *endeffecter* para cada uma das execuções.

Tabela 13: Resultados dos experimentos sem detecção da *tag*.

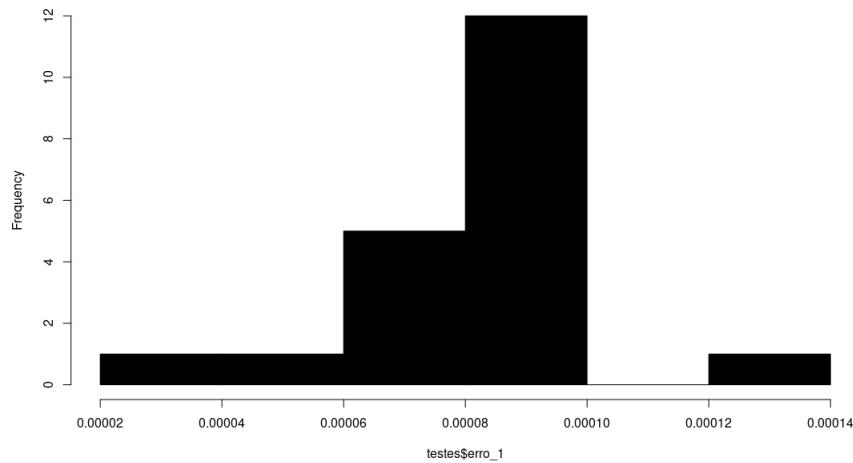
	Pose	Erro
1	A	0.000135296
2	A	0.000095000
3	A	0.000090747
4	A	0.000097944
5	A	0.000081093
6	A	0.000068184
7	A	0.000088978
8	A	0.000074532
9	A	0.000036373
10	A	0.000086499
11	B	0.000095900
12	B	0.000085300
13	B	0.000064900
14	B	0.000089500
15	B	0.000089000
16	B	0.000048300
17	B	0.000096700
18	B	0.000090200
19	B	0.000077800
20	B	0.000079800

Fonte: Autoria própria.

Para avaliar os dados, inicialmente foi realizado um teste de normalidade, como forma

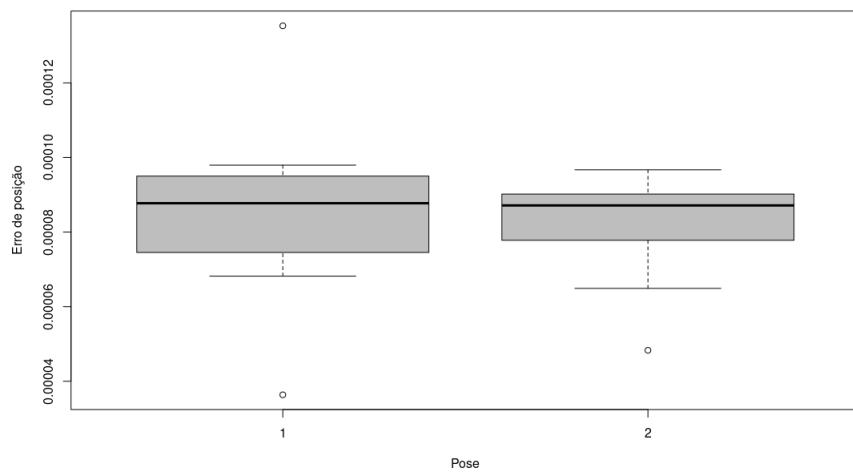
de verificar se os dados são bem modelados por uma distribuição normal ou não. O teste de Shapiro-Wilk foi utilizado para esse conjunto de dados, retornando um ρ -valor de 0.05569, o que indica que a hipótese nula não pode ser rejeitada, logo os dados seguem uma distribuição normal. A distribuição dos mesmos pode ser melhor visualizada através de um histograma, como mostra a figura 24.

Figura 24: Histograma do conjunto de dados sem detecção da *tag*.



Segundo a análise dos testes, foi realizado um teste ANOVA com a amostra de dados coletada. O objetivo foi avaliar a variância dos dados em cada ponto e entre eles, analisando se estes diferem estatisticamente entre si. Com esta análise, obteve-se um ρ -valor de 0.6912, com isso tem-se embasamento estatístico para afirmar que os dados no ponto "A" e no ponto "B" são estatisticamente iguais, os erros de pose apresentados nos dois pontos não diferem entre si. Esta análise pode ser visualizada na figura 25.

Figura 25: Boxplot do conjunto de dados sem detecção da *tag*.



Observando a figura 25 a pose 1 (A) possui maior variação dos seus dados em relação a pose 2 (B), porém as duas não diferem estatisticamente entre si.

5.3.2.2 Análise de variância com detecção da *tag*

Foi tomado um novo conjunto de dados, como mostra a tabela 14. Este experimento foi realizado com variação do operador e posição da caixa. Foram tomados dados de tempo de busca da *tag*, tempo total da missão e o erro de posição do *end-effector*. Também foi observado a porcentagem de sucesso na conclusão da missão, sendo o índice '1' correspondente ao sucesso na missão e o índice '-1' correspondente à falha.

Tabela 14: Resultados dos experimentos sem detecção da *tag*.

	Operador	Pose	Erro de pose	Tempo de busca	Tempo total	Sucesso da missão
1	1	A	0.010451819	27.1022	93.9316	1
2	1	A	0.009745817	29.1912	91.8192	1
3	1	A	0.010017589	27.3904	95.6437	1
4	1	A	0.010819201	28.4745	90.5214	1
5	1	A	0.011133634	27.7250	98.7845	1
6	1	A	0.010562436	28.8388	94.6226	1
7	1	A	0.010132297	29.1519	101.8420	1
8	1	A	0.010693537	30.2186	90.1885	1
9	2	A	0.010905047	30.2317	94.0104	1
10	2	A	0.010469991	28.9793	96.2312	1
11	2	A	0.010355747	29.0720	103.6160	1
12	2	A	0.010268870	28.6952	93.0651	1
13	2	A	0.010230915	28.6420	87.2120	1
14	2	A	0.004079350	26.8585	86.4156	1
15	2	A	0.004427968	26.1808	85.7109	1
16	2	A	0.009985857	30.0858	90.4296	1
17	1	A	0.009754491	28.2042	86.8758	1
18	1	A	0.010334966	28.6983	92.2869	1
19	1	A	0.010058060	29.4320	94.3545	1
20	1	A	0.010153228	28.1812	90.6696	1
21	1	A	0.010120659	29.2442	87.1852	1
22	1	A	0.010105795	28.9948	88.3569	1
23	1	A	0.010105795	28.9948	88.3569	-1
24	1	A	0.011557242	32.9496	102.3640	1
25	2	A	0.011245238	34.7259	98.7636	1
26	2	A	0.011190817	31.1780	113.0500	1
27	2	A	0.010174117	28.7923	88.9735	1
28	2	A	0.010007278	28.8178	91.8702	1
29	2	A	0.009963924	29.0347	86.7183	1
30	2	A	0.010475097	28.2422	99.1881	1
31	2	A	0.010741058	28.2118	91.7459	1
32	2	A	0.004469904	28.4826	88.1583	-1
33	1	A	0.010086673	28.7112	87.9372	1
34	1	A	0.010515376	28.8792	92.9002	1
35	1	A	0.010062717	29.1420	91.5910	1
36	1	A	0.010422717	29.0783	94.3296	1
37	1	A	0.010185682	27.9274	88.6971	1
38	1	A	0.010281946	29.7805	100.1800	1
39	1	A	0.013133281	20.0540	82.3651	-1
40	1	A	0.012344603	19.1706	81.7740	1

	Operador	Pose	Erro de pose	Tempo de busca	Tempo total	Sucesso da missão
41	2	B	0.005245180	20.4070	84.7338	1
42	2	B	0.007158873	20.4408	84.3043	1
43	2	B	0.011482759	20.7824	83.2411	1
44	2	B	0.087866430	23.4152	87.9248	-1
45	2	B	0.008514302	19.9797	86.4344	1
46	2	B	0.005144988	19.8502	84.5138	1
47	2	B	0.013200185	19.9392	85.7667	-1
48	2	B	0.004874412	20.2860	80.6176	1
49	1	B	0.012512453	20.7894	85.9667	1
50	1	B	0.014427715	19.4977	81.7383	-1
51	1	B	0.007453902	20.5577	82.4847	1
52	1	B	0.006078841	20.4143	81.6851	1
53	1	B	0.009550672	19.5227	84.7312	1
54	1	B	0.006123619	16.9920	74.6466	1
55	1	B	0.003705978	18.6949	80.1175	1
56	1	B	0.006896455	19.9972	76.2407	1
57	2	B	0.006869856	18.8696	79.7714	1
58	2	B	0.002917472	18.4679	73.9755	1
59	2	B	0.004802752	19.4849	80.2423	1
60	2	B	0.004557056	19.5606	77.7174	1
61	2	B	0.006353444	18.5702	76.3386	1
62	2	B	0.007120232	19.6487	78.9122	1
63	2	B	0.004832266	17.0441	77.3120	1
64	2	B	0.004535360	18.0856	77.5731	1
65	1	B	0.007897063	17.1585	78.1917	1
66	1	B	0.007330963	20.5918	81.1574	1
67	1	B	0.008284862	20.2515	79.0624	1
68	1	B	0.008096144	19.9652	81.4914	1
69	1	B	0.013425916	19.0627	79.8650	-1
70	1	B	0.007966466	18.6261	75.1340	1
71	1	B	0.008486843	18.1093	72.8817	1
72	1	B	0.007759457	18.7750	77.1361	1
73	2	B	0.012489658	20.1018	77.2422	1
74	2	B	0.008122309	18.6224	77.0045	1
75	2	B	0.012097601	20.3431	79.8706	1
76	2	B	0.087776215	17.5113	77.7186	1
77	2	B	0.006999748	17.3791	74.1548	1
78	2	B	0.008223493	18.3666	76.5340	1
79	2	B	0.008721295	19.4229	73.9818	1
80	2	B	0.008375202	18.6691	73.7671	1

Fonte: Autoria própria.

Observando a tabela 14, os testes apresentaram 91.25% de sucesso, ou seja, dos 80 testes realizados, em 73 obteve-se sucesso no pressionamento do botão. A tabela 15 resume alguns indicadores importantes das três saídas do sistema (tempo de busca, tempo total da missão e erro de pose).

Os testes de normalidade apresentaram valores de ρ -valor inferiores a 0.05, ou seja, a hipótese nula não pode ser rejeitada, logo os dados seguem uma distribuição normal.

Tabela 15: Resumo dos conjuntos de dados das variáveis de saída.

	Média	Desvio padrão	ρ -valor (Shapiro-Wilk)
Tempo de busca	23.95025	5.035711	1.103e-07
Tempo total	86.06149	8.343288	0.02409
Erro de pose	0.01095061	0.0126478	2.2e-16

Fonte: Autoria própria.

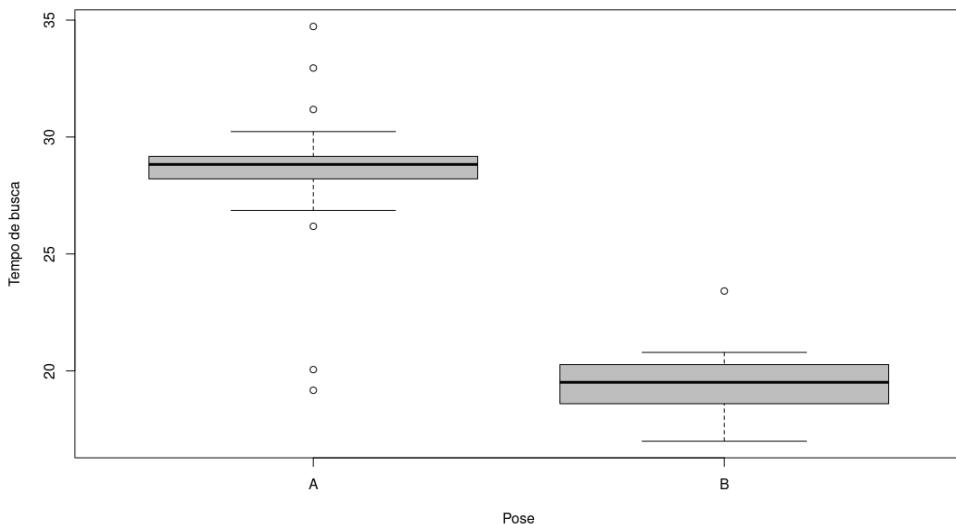
Tabela 16: Tabela dos valores de ρ -valor do teste ANOVA.

	p-valor
Erro de pose	0.525
Tempo de busca	2.2e-16
Tempo total	2.2e-16

Fonte: Autoria própria.

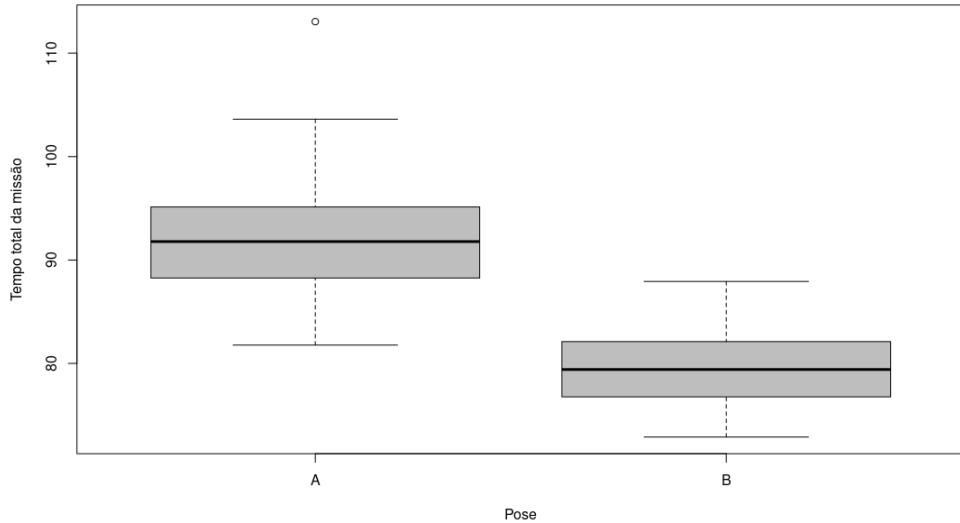
Realizando o teste do ANOVA, podemos visualizar a variância dos pontos A e B em relação ao erro de pose, tempo de busca e tempo total da missão. A tabela 16 apresenta o ρ -valor para cada conjunto de dados, e a partir dessa análise podemos chegar a conclusão que a hipótese nula não pode ser rejeitada para o erro de pose assim como visto na análise passada, ou seja, os dados obtidos do erro de posição do *end-effector* nos pontos A e B são estatisticamente iguais. Já os dados dos tempo de busca e tempo total da missão apresentam ρ -valor menor que 0.05, logo a hipótese nula é rejeitada e pode-se concluir que os dados diferem entre si, como mostram os *boxplots* das figuras 26 e 27.

Figura 26: *Boxplot* da variância do tempo de busca entre os pontos A e B.



Fonte: Autoria própria.

Figura 27: *Boxplot* da variância do tempo total da missão entre os pontos A e B.



Fonte: Autoria própria.

5.3.3 Análise de repetibilidade e reproduzibilidade

Como forma de avaliação do sistema robótico, também foi adotado um estudo de repetibilidade e reproduzibilidade do sistema. Com isso, foram adotados os dados da tabela 14. Foram tomadas 20 medições de cada operador para cada pose da caixa, totalizando 80 testes e assim foram obtidos os seguintes dados apresentados na figura 28 e 29.

Observando a figura 28, correspondente ao teste de R&R para o tempo busca do marcador visual, neste teste percebe-se o número de categorias igual a 6, número considerável e acima de valores de referências, o que demonstra que o sistema de medição pode ser utilizado para análise dos dados. Outro ponto importante para observado é o indicador "*Total Gage R&R*", que foi calculado em 20.37, inferior a 30% (Valor máximo aceitável adotado em algumas literaturas).

O sistema sendo aceitável para análise, pode-se observar outros indicadores importantes para análise, como exemplo do ρ -valor, que mostra a maior influência na variação devido a pose da caixa, não ocorrendo influência do operador ou da interação entre operador e pose. O indicador "*Part-to-Part*" indica que aproximadamente 98% da variação do sistema deve-se a própria variação de pose da caixa, o que demonstra que o sistema possui repetibilidade e reproduzibilidade, quanto a variável de tempo de busca.

Figura 28: Modelo completo teste R&R para tempo de busca.

```

Complete model (with interaction):
      Df Sum Sq Mean Sq F value Pr(>F)
pose_rep       1 1902.1 1902.1 403.599 0.0317
operador       1    0.1    0.1   0.012 0.9308
pose_rep:operador 1    4.7    4.7   2.450 0.1215
Repeatability  80 153.9    1.9
Total          83 2060.7

alpha for removing interaction: 0.05

Reduced model (without interaction):
      Df Sum Sq Mean Sq F value Pr(>F)
pose_rep       1 1902.1 1902.1 971.443 <2e-16
operador       1    0.1    0.1   0.029 0.866
Repeatability  81 158.6    2.0
Total          83 2060.7

Gage R&R

      VarComp %Contrib
Total Gage R&R  1.957994  4.15
  Repeatability  1.957994  4.15
  Reproducibility 0.000000  0.00
    operador     0.000000  0.00
Part-To-Part    45.240977 95.85
Total Variation  47.198971 100.00

      StdDev StudyVar %StudyVar
Total Gage R&R  1.399283  8.39570   20.37
  Repeatability  1.399283  8.39570   20.37
  Reproducibility 0.000000  0.00000   0.00
    operador     0.000000  0.00000   0.00
Part-To-Part    6.726141 40.35685   97.90
Total Variation  6.870151 41.22090   100.00

Number of Distinct Categories = 6

```

Fonte: Autoria própria.

Figura 29: Modelo completo teste R&R para tempo total da missão.

```

Complete model (with interaction):
      Df Sum Sq Mean Sq F value Pr(>F)
pose_rep       1   3659    3659 431.235 0.0306
operador       1      2      2   0.272 0.6941
pose_rep:operador 1      8      8   0.351 0.5550
Repeatability  80   1932      24
Total          83   5602

alpha for removing interaction: 0.05

Reduced model (without interaction):
      Df Sum Sq Mean Sq F value Pr(>F)
pose_rep       1   3659    3659 152.736 <2e-16
operador       1      2      2   0.096 0.757
Repeatability  81   1941      24
Total          83   5602

Gage R&R

      VarComp %Contrib
Total Gage R&R  23.95875  21.68
  Repeatability  23.95875  21.68
  Reproducibility 0.00000   0.00
    operador     0.00000   0.00
Part-To-Part    86.55716  78.32
Total Variation 110.51591 100.00

      StdDev StudyVar %StudyVar
Total Gage R&R  4.894768 29.36861   46.56
  Repeatability  4.894768 29.36861   46.56
  Reproducibility 0.000000  0.00000   0.00
    operador     0.000000  0.00000   0.00
Part-To-Part    9.303610 55.82166   88.50
Total Variation 10.512655 63.07593   100.00

Number of Distinct Categories = 2

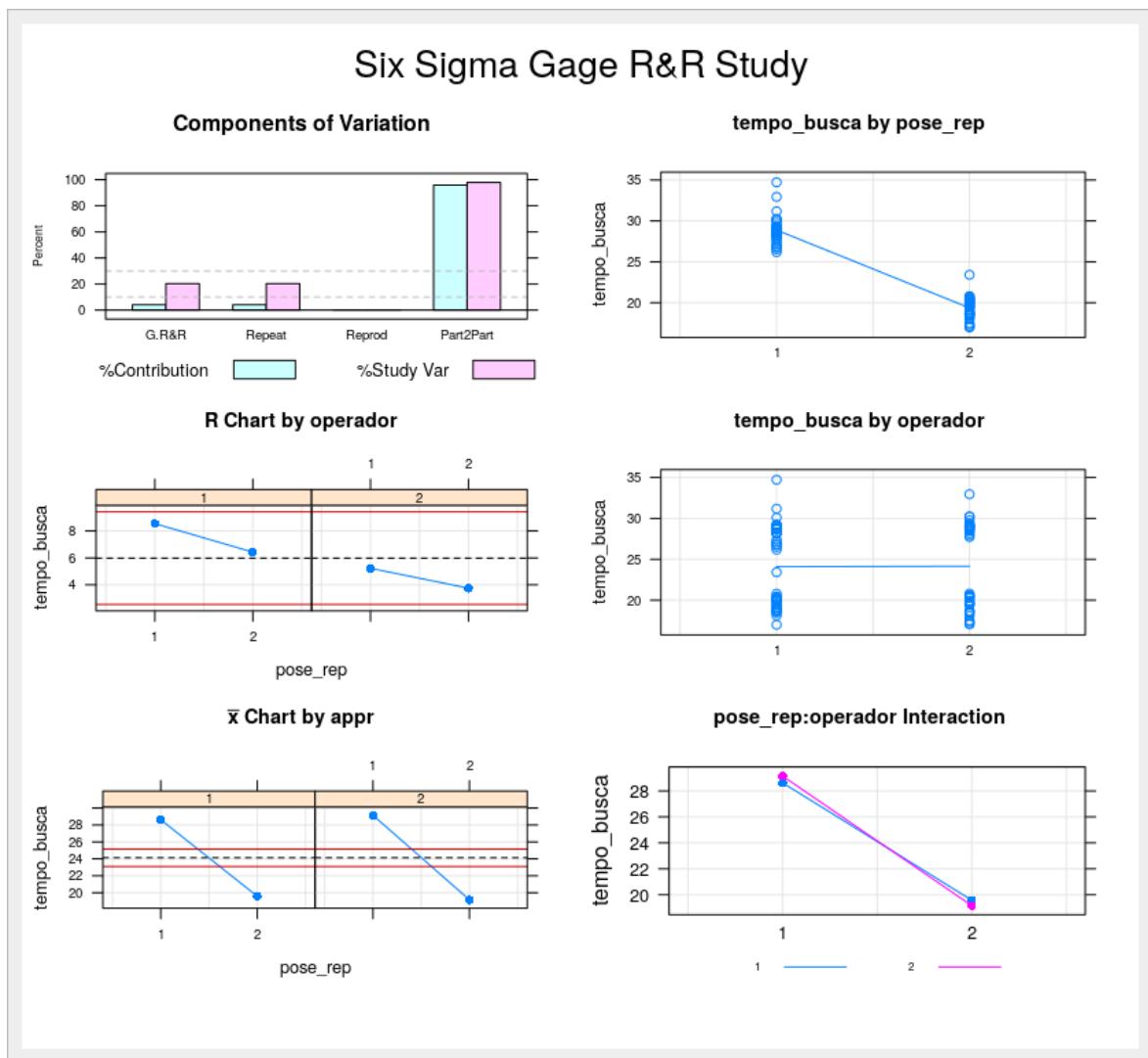
```

Fonte: Autoria própria.

O mesmo não pode ser observado na figura 29, onde o número de categorias apresentados é baixo e o índice "Total Gage R&R" foi calculado em 46.56%, acima do valor máximo de referência (30%), o que mostra que o sistema de medição não é aceitável quando observado a variável de tempo total da missão.

Segundo a análise dos dados obtidos no estudo de repetibilidade e reprodutibilidade do sistema para o tempo de busca, visualizamos cartas na figura 30 que possibilita um melhor entendimento do sistema de medição.

Figura 30: Gráficos do estudo R&R para tempo de busca.



Fonte: Autoria própria.

A carta "Componentes de variação" representa atribuições da variação do sistema, e como observado na figura 30, a variação deve-se principalmente a "peça por peça" ou "parte por parte", o que é desejável, demonstrando que o sistema de medição é aceitável para análise do tempo de busca do manipulador.

A carta R é uma carta de controle de amplitudes que representa graficamente a consistência do operador. Os pontos apresentados na figura 30 estão dentro dos limites máximos e mínimos, ou seja, é sinal de que os operadores estão realizando os testes de forma consistente.

Os pontos apresentados na carta Xbar, na figura 30, os dados estão acima ou abaixo dos limites de controle. Estes resultados indicam que a variação entre as poses são maiores que a variação do dispositivo de medição.

Outro ponto importante de ser destacado é o gráfico de interação entre o operador e o tempo de busca (figura 30), a barra horizontal visualizada, demonstra que os operadores estão realizando medições semelhantes entre si, com variações semelhantes entre as medidas máximas e mínimas do tempo de busca.

5.4 Conclusões dos testes estatísticos

Após estudo dos dados obtidos a partir de testes no manipulador robótico, verifica-se que os dados apresentaram distribuição normal, o que possibilita uma análise confiável dos mesmos, comprovando isso através de histogramas apresentados durante a seção.

Também é possível observar nos testes realizados que o algoritmo de planejamento de trajetória RRT CONNECT apresentou melhor resultado em relação ao Kpiece, bem como a velocidade pouco influenciou nos testes, mostrando que a melhor configuração para o sistema para as missões impostas é utilizando o algoritmo RRT.

Não houve diferença no erro de posicionamento do *end-effector* quando observado as poses A e B e com ou sem detecção do marcador visual, ou seja, pode-se concluir que o manipulador robótico apresenta erro constante nas poses analisadas. É observado também que o tempo de busca e tempo total da missão diferem entre as poses.

Por fim, conclui-se que a análise de repetibilidade e reprodutibilidade foi positiva quando observados os dados da variável tempo de busca do marcador visual, uma vez que o sistema de medição atendeu ao teste; enquanto que para a variável tempo total da missão os resultados não foram de todo satisfatórios, indicando possíveis direções de melhoria no sistema.

6 CONFIABILIDADE DO SISTEMA

A fim de verificar-se a confiabilidade do sistema, foi realizado o levantamento de cada componente que constitui o manipulador robótico e o estudo detalhado das prováveis falhas. Para isso, foi utilizado o *FMECA* com o propósito de analisar cada sub-sistema, como pode ser visto nas tabelas da seção 6.1.

Na seção 6.2 foi desenvolvida a análise da árvore de falhas do sistema, método que permite através de um processo lógico dedutivo chegar-se às causas básicas de um problema de proporções maiores.

6.1 Análise dos modos e efeitos de falhas

Nas Tabelas 17, 18, 19, 20 e 21 estão representadas informações referentes ao estudo sistemático e estrutura das falhas potenciais para os sub-sistemas de potência, de aquisição, estrutural, de atuação e de processamento, respectivamente.

Tabela 17: *FMECA* do sub-sistema de potência

Análise do Tipo e Efeito de Falha								
Sub-sistema de potência								
Função(es)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(es) recomendada(s)
Transmissão de dados	Incapacidade de transmissão de dados	Comprometimento dos dados transmitidos	6	Fissura dos fios	3	5	90	Manutenções preventivas
		Perda da capacidade de transmissão de dados	8	Desgaste causado pelo tempo	2	1	16	
Transmissão de energia	Incapacidade de transmissão de energia	Perda da capacidade de transmissão de energia	8	Derretimento por temperatura elevada	1	1	8	Manutenções preventivas
		Má qualidade da energia transmitida	6	Torção dos fios	3	5	90	
Alimentação do sistema	Incapacidade em fornecer energia	Não energização do sistema	9	Má conexão com fonte de tensão	3	6	162	Verificar conexão com fonte
				Desgaste nas soldas causado pelo tempo	4	7	252	

Fonte: Autoria própria.

Tabela 18: *FMECA* do sub-sistema aquisição

Análise do Tipo e Efeito de Falha								
Sub-sistema de aquisição								
Função(es)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(es) recomendada(s)
Aquisição de dados visuais	Incapacidade de obter dados visuais	Perda da capacidade de obter dados visuais	8	Obstrução do campo de visão da câmera	2	1	16	Verificar condições do ambiente antes das missões
	Ruptura da estrutura de fixação da câmera			Baixa visibilidade do ambiente	2	5	80	
	Coleta de dados inconsistentes ou insuficientes			Calibração incorreta da câmera	3	1	24	
	Comprometimento dos dados	Set-up incorreto	6	Set-up incorreto	2	7	84	Verificar cabeamento do sistema
				Falha no canal de comunicação	4	7	168	

Fonte: Autoria própria.

Tabela 19: *FMECA* do sub-sistema estrutural

Análise do Tipo e Efeito de Falha									
Sub-sistema estrutural									
Função(es)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(es) recomendada(s)	
Ligar o frame ao perfil de alumínio	Incapacidade de sustentar o perfil	Ruptura do conector	8	Excesso de peso aplicado	2	1	16	Verificar a fabricação da peça	
				Desgaste pelo tempo	2	5	80		
	Compor estrutura mecânica	Trinca do conector	6	Set-up incorreto	4	6	144	Verificar as condições de peso aplicadas	
				Defeito de fabricação	5	6	180		
Danificação estrutural	Ruptura do perfil	8	Sobrecarga	2	4	64	Cuidado no manuseamento dos perfis		
	Folga entre conexões	6	Colisão	4	2	48	Definir área de trabalho		

Fonte: Autoria própria.

Tabela 20: *FMECA* do sub-sistema de atuação

Análise do Tipo e Efeito de Falha									
Sub-sistema de atuação									
Função(es)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(es) recomendada(s)	
Movimentação do manipulador	Rotação inconsistente com o desejado	Perda de desempenho (velocidade; controle)	6	Emperramento parcial	4	6	144	Inspecções periódicas	
Fornecer o torque mínimo de sustentação	Curto circuito	Parada do motor	8	Cabeamento incorreto	2	5	80	Medir corrente nos terminais	
	Perda de comunicação	Perda de desempenho	6	Falha na alimentação	4	5	120		
	Tensão insuficiente								

Fonte: Autoria própria.

Tabela 21: *FMECA* do sub-sistema de processamento

Análise do Tipo e Efeito de Falha								
Sub-sistema de processamento								
Função(es)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(es) recomendada(s)
Processamento dos dados	Não processamento dos dados do sistema principal	Não funcionamento	9	Queima/ danificação de componentes	3	8	216	Manutenção preditiva
				Falha no sistema operacional	2	8	144	Executar teste de verificação do software
				Falha na porta de comunicação	2	2	36	Executar teste de entrada e saída de dados com <i>NUC</i>
				Problemas de contato elétrico	2	3	54	Inspecções periódicas em bancada

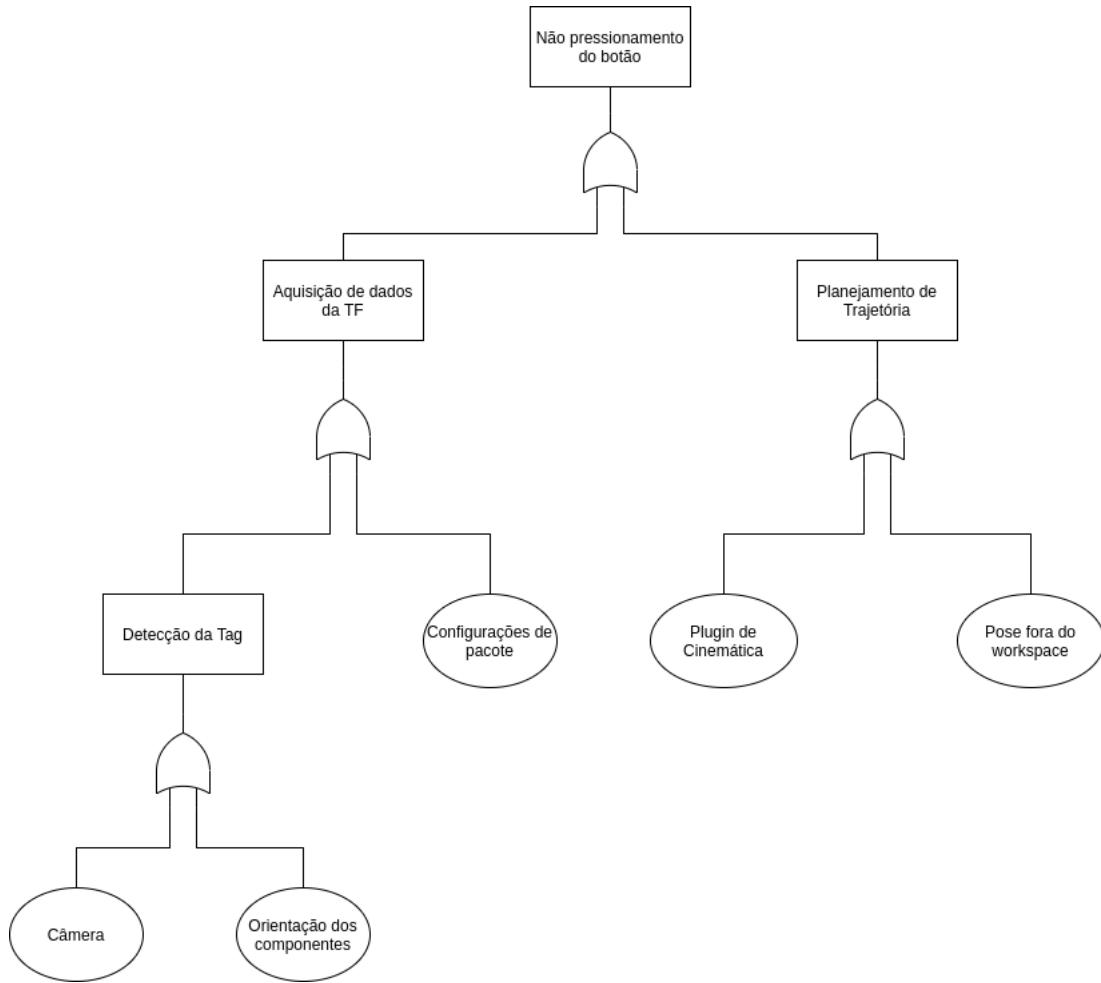
Fonte: Autoria própria.

6.2 Análise da árvore de falhas

Na Figura 31 encontra-se a árvore de falhas do sistema do manipulador JeRoTIMON. Observa-se que caso o manipulador não consiga realizar a tarefa de pressionar o botão, duas causas principais devem ser investigadas: a conexão entre as TF's, realizada a partir do pacote *Bir Marker Localization*, ou o planejamento de trajetória, realizado pelo *Moveit*.

Para a primeira causa, devem ser verificadas as configurações do pacote e da câmera, bem como a orientação dos componentes envolvidos no processo de detecção. Já para solucionar os problemas correspondentes ao planejamento de trajetória, deve-se observar se a pose desejada está dentro do *workspace* do manipulador e se o *plugin* de cinemática inversa utilizado está sendo capaz de realizar as conversões necessárias para a pose desejada.

Figura 31: Árvore de falhas do sistema.



Fonte: Autoria própria.

7 GESTÃO DO CONHECIMENTO

Neste capítulo estão descritas as lições aprendidas durante o processo de desenvolvimento do protótipo que foram criadas a partir da comparação entre o que era esperado e o que realmente aconteceu em cada etapa do projeto. Esta sub-seção engloba desde a fase inicial do planejamento até a construção do modelo real.

Além das lições aprendidas, as seções 7.2 e 7.3 trazem os guias de uso para a simulação e o modelo real, respectivamente, com o propósito de auxiliar o usuário na replicação dos experimentos realizados neste relatório.

7.1 Lições aprendidas

A Tabela 22 mostra como foi estruturada cada lição aprendida abordando os seguintes aspectos: Tema, Fase, Impacto, O que ocorreu?, Como resolveu?, Resultados e Recomendações para os próximos projetos. O objetivo deste estudo é a correção dos impactos negativos para os projetos subsequentes.

Tabela 22: Lições aprendidas

LIÇÕES APRENDIDAS						
Tema	Fase	Impacto	O que ocorreu?	Como resolveu?	Resultados	Recomendações para os próximos projetos
Gestão	Planejamento	Negativo	Ausência de uma metodologia de trabalho	Reunião com foco em definir metodologia de projeto	Evolução na comunicação dos membros	Antes de começar o projeto realizar reunião para definição de metodologia
Gestão	Planejamento	Negativo	Ausência de uma ferramenta para gestão de projeto	Escolha de uma ferramenta gratuita e de fácil aplicação	Melhoria na organização das atividades	Definição prévia de uma ferramenta de gestão de projeto
Gestão	Execução	Negativo	Montagens e desmontagens do manipulador	Planejamento prévio das atividades	Otimização do tempo	Cronograma definido as atividades a serem realizadas
Tecnológico	Execução	Negativo	Falha no dimensionamento das peças	Consultoria sobre materiais aplicados na confecção	Obtenção de peças com maior resistência mecânica	Pesquisa e consultoria prévia antes da modelagem das peças
Tecnológico	Execução	Negativo	Dificuldade em planejamento de trajetória para determinadas poses do robô	Utilização do plugin Track-IK	Maior eficiência de planejamento	Pesquisa e consultoria prévia do pacote mais adequado para o projeto
Tecnológico	Execução	Negativo	Motor dynamixel (H54-200-S500-R PRO) parou de funcionar	Realizado a Análise 8D para fazer investigação do ocorrido	Compreensão de que há procedimentos a serem feitos antes de utilizar um produto	Realizar leitura meticolosa do manual do produto para saber quais são os procedimentos

Fonte: Autoria própria.

7.2 Guia de uso para simulação

Para replicar a simulação do manipulador robótico Timon-HM é necessário seguir os passos descritos nesta seção. Recomenda-se a utilização do Ubuntu 18.04 LTS e o *ROS Melodic Morenia*.

Antes de inserir o pacote do manipulador no *workspace* é fundamental que sejam instalados os pacotes requeridos para este. Primeiramente, no terminal, segue-se os comandos listados:

- Instalar MoveIt:

```
$ sudo apt-get install ros-melodic-moveit
```

- Instalar pacote de ferramentas visuais do MoveIt:

```
$ sudo apt-get install ros-melodic-moveit-visual-tools
```

- Instalar TRAC-IK para resolução da cinemática:

```
$ sudo apt-get install ros-melodic-trac-ik-kinematics-plugin
```

- Instalar pacote de controle no *Gazebo ROS*:

```
$ sudo apt-get install ros-melodic-gazebo-ros-control
```

- Instalar pacote do controlador do *ROS*:

```
$ sudo apt-get install ros-melodic-controller-*
```

- Instalar pacote controlador de posição do *ROS*:

```
$ sudo apt-get install ros-melodic-position-controller
```

- Instalar pacote controlador de esforço do *ROS*:

```
$ sudo apt-get install ros-melodic-effort-controller
```

- Instalar pacote de juntas do *ROS*:

```
$ sudo apt install ros-melodic-joint-*
```

- Instalar pacote de controle do *ROS*:

```
$ sudo apt install ros-melodic-ros-control
```

Antes de instalar o pacote *bir_marker_localization* é necessária a instalação do *OpenCV* versão 3.3.1, este possui o guia de instalação próprio disponível em:

<https://www.learnopencv.com/install-opencv3-on-ubuntu/>.

Após a instalação do *OpenCV*, para clonar o repositório do *bir_marker_localization* segue-as as seguintes etapas, no terminal:

- Criar um *workspace* para adicionar dentro deste os pacotes que serão usados na simulação.

```
$ mkdir nomedoworkspace_ws
```

- Entrar no *workspace*:

```
$ cd nomedoworkspace_ws
```

- Criar a pasta *source*¹:

```
$ mkdir src
```

- Entrar no *source*:

```
$ cd src
```

- Clonar o repositório *Bir Marker Localization* para *workspace* (Verificar qual a *branch* estável):

```
$ git clone https://github.com/Brazilian-Institute-of-Robotics/bir_marker_localization.git
```

- Clonar o pacote do manipulador para dentro da pasta src:

```
$ git clone -b feature/simulation https://github.com/Brazilian-Institute-of-Robotics/timon_hm_manipulator.git
```

- Clonar o pacote do *Open Manipulator* para dentro da pasta src:

```
$ git clone https://github.com/ROBOTIS-GIT/open_manipulator_msgs.git
```

- Retornar para a raiz do *workspace*:

```
$ cd ..
```

- Compilar o *workspace*:

```
$ catkin_make
```

- Ativar o ambiente virtual do *workspace*:

```
$ source devel/setup.bash
```

¹ Pasta onde contém os arquivos fonte.

Após realizar os procedimentos citados o *workspace* estará configurado para executar a simulação, com os seguintes comandos (cada comando terá que ser inserido em uma aba do terminal):

- Iniciar a simulação no *Gazebo*:

```
$ roslaunch manipulator_gazebo gazebo.launch
```

- Iniciar pacote MoveIt do Timon-HM:

```
$ roslaunch manipulator_gazebo moveit_demo.launch
```

- Iniciar o *bir_marker_localization*:

```
$ roslaunch timon_demo bir_marker_localization.launch
```

- Iniciar o algoritmo de busca do marcador visual e para acionar o painel elétrico:

```
$ roslaunch timon_demo push_button_simulation.launch
```

7.3 Guia de uso para o modelo real

Para replicar o modelo real do manipulador a versão do Ubuntu, do *ROS* e os pacotes necessários são os mesmos descritos na seção 7.2. Após a instalação dos pacotes, a seguir estão descritas as etapas subsequentes:

- Criar *workspace* e a pasta *source*:

```
$ mkdir -p catkin_ws/src
```

- Entrar no *workspace* e no *source*:

```
$ cd catkin_ws/src
```

- Clonar para dentro da pasta *source* o repositório do manipulador:

```
$ git clone https://github.com/Brazilian-Institute-of-Robotics/timon_hm_manipulator.git
```

- Clonar para dentro do *source* o repositório do *Bir Marker Localization*:

```
$ git clone -b final_settings https://github.com/Brazilian-Institute-of-Robotics/bir_marker_localization.git
```

- Clonar para dentro do *source* o repositório do *Open Manipulator*:

```
$ git clone https://github.com/ROBOTIS-GIT/
open_manipulator_msgs.git
```

- Clonar para dentro do *source* o repositório do *Dynamixel workbench*:

```
$ git clone https://github.com/ROBOTIS-GIT/dynamixel-
workbench.git
```

- Clonar para dentro do *source* o repositório da câmera *Teledyne*:

```
$ git clone -b refactor_code https://github.com/Brazilian
-Institute-of-Robotics/def_cam_teledyne_nano.git
```

- Retornar para a raiz do *workspace*:

```
$ cd ..
```

- Compilar o *workspace*:

```
$ catkin_make
```

- Ativar o ambiente virtual do *workspace*:

```
$ source devel/setup.bash
```

Para executar a aplicação é necessário realizar os seguintes comandos (cada comando terá que ser inserido em uma aba do terminal):

- Iniciar os controladores do manipulador:

```
$ roslaunch timon_arm_controller dxl_controllers.launch
```

```
$ roslaunch timon_arm_controller moveit.launch
```

```
$ roslaunch timon_arm_controller dxl_moveit_bridge.launch
```

- Iniciar a câmera:

```
$ roslaunch def_cam_teledyne_nano camera_example.launch
```

- Iniciar o *Bir Maker Localization*:

```
$ roslaunch timon_demo bir_marker_localization.launch
```

- Executar a missão:

```
$ roslaunch timon_demo push_button_real.launch
```

8 CONCLUSÃO

O presente relatório descreveu a idealização, simulação e construção do JeRoTIMON, um manipulador robótico com 5 *DoF*, integrado ao *ROS*, capaz de acionar um painel elétrico a partir da localização de um marcador visual *ArUco*. A estrutura física do robô foi concebida de acordo com os modelos *URDF* projetados.

Para possibilitar o uso da ferramenta *Moveit*, arquivos de configuração foram gerados e sua comunicação com o modelo simulado no *Gazebo* foi estabelecida. Para resolver as equações de cinemática inversa optou-se pelo plugin TRAC-IK e para o planejamento de trajetória foi utilizada a biblioteca *OMPL*.

Foi desenvolvido em linguagem C++ um pacote capaz de comunicar o sistema de escaneamento e o sistema de planejamento/execução. A utilização da câmera Teledyne Genie Nano C2590 possibilitou a identificação da *tag ArUco* com precisão, tornando o sistema capaz de localizar o painel elétrico e planejar uma trajetória que leve o *endeffector* do manipulador até o alvo estabelecido.

Foram realizados 80 testes considerando diferentes algoritmos, posições e orientações para o painel elétrico. Os resultados alcançados mostram que JeRoTIMON foi capaz de realizar a tarefa em 91.25% dos casos, com tempo médio de busca de 23.95 segundos e o tempo médio de execução de 86.06 segundos, resultados estes considerados satisfatórios para o prosseguimento do projeto.

Para sua próxima aplicação, o manipulador será instalado em um robô móvel *Warthog* que estará equipado com sensores de localização, mapeamento e detecção de obstáculos. De forma autônoma, este conjunto irá localizar e desarmar uma bomba hipotética instalada em ambiente aberto.

REFERÊNCIAS

- 16MM C Series VIS-NIR Fixed Focal Length Lens. s.d. <<https://bit.ly/35zzQMq>>. Acessado: 26-10-2020. Citado na página 36.
- BEESON, P.; AMES, B. Trac-ik: An open-source library for improved solving of generic inverse kinematics. In: IEEE. *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. [S.l.], 2015. p. 928–935. Citado na página 29.
- BIR Marker Localization. 2020. <https://github.com/Brazilian-Institute-of-Robotics/bir_marker_localization/>. Accessed: 2020-04-24. Citado na página 27.
- ERTHAL, J. L. Estudo de métodos para a solução da cinemática inversa de robôs industriais para implementação computacional. *UFSC, Programa de Pós-Graduação*, 10 1992. Citado na página 13.
- HERNANDEZ-MENDEZ, S. et al. Design and implementation of a robotic arm using ros and moveit! In: IEEE. *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*. [S.l.], 2017. p. 1–6. Citado na página 18.
- HERNÁNDEZ-ORDÓÑEZ, M. et al. An education application for teaching robot arm manipulator concepts using augmented reality. *Mobile Information Systems*, Hindawi, v. 2018, 2018. Citado na página 18.
- INDUSTRIAL Robots: Robot Investment Reaches Record 16.5 billion USD. 2019. <<https://ifr.org/ifr-press-releases/news/robot-investment-reaches-record-16.5-billion-usd/>>. Acesso em: 15-04-2020. Citado na página 13.
- ISO-8373. *Robots and robotic devices — Vocabulary*. Geneva, CH, 2012. Citado na página 17.
- JAVEED, A.; PRAKASH, V. G.; KULKARNI, S. P. Autonomous service robot. In: *Proceedings of the Advances in Robotics 2019*. [S.l.: s.n.], 2019. p. 1–6. Citado na página 19.
- KHAN, K. A.; KONDA, R. R.; RYU, J.-C. Ros-based control for a robot manipulator with a demonstration of the ball-on-plate task. *Advances in robotics research*, v. 2, n. 2, p. 113, 2018. Citado na página 19.
- LEITE, A. C. *Controle Híbrido de Força e Visão de um Manipulador Robótico Sobre Superfícies Desconhecidas*. Tese (Doutorado) — Tese de Mestrado, Departamento de Engenharia Elétrica, 2005. Citado na página 14.
- PIMENTA, T. T. Controle de manipuladores robóticos. *PUC-Rio de Janeiro, Departamento de Engenharia de Controle e automação*, 12 2009. Citado na página 13.
- ROBOTIS. s.d. <<http://www.robotis.us/>>. Acessado: 19-10-2020. Citado 2 vezes nas páginas 38 e 81.
- ROMANO, V. F. *Robótica Industrial: Aplicação na indústria de manufatura e processos*. São Paulo: Edgar Blucher, 2002. v. 256. Citado na página 13.

SANTOS, V. M. Robótica industrial. *Universidade de Aveito-Departamento de Engenharia Mecânica*, 2004. Citado na página 18.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. Robot modeling and control, jon wiley & sons. Inc., ISBN-100-471-649, 2005. Citado na página 13.

SUCAN, I. A.; MOLL, M.; KAVRAKI, L. E. The open motion planning library. *IEEE Robotics & Automation Magazine*, IEEE, v. 19, n. 4, p. 72–82, 2012. Citado na página 29.

TELEDYNE Dalsa. s.d. <<https://www.teledynedalsa.com/en/products/imaging/cameras/genie-nano-1gige/>>. Acessado: 19-10-2020. Citado na página 36.

U2D2. s.d. <<https://emanual.robotis.com/docs/en/parts/interface/u2d2/>>. Acessado: 26-10-2020. Citado 2 vezes nas páginas 38 e 39.

UWE-12/10-Q12PB-C. s.d. <<https://bit.ly/31i2SyP>>. Acessado: 19-10-2020. Citado na página 38.

ZHANG, S.; LIN, Z.; WU, G. Motion planning of a 5-dof anthropomorphic robotic arm under ros environment. In: SPRINGER. *IFToMM International Conference on Mechanisms, Transmissions and Applications*. [S.l.], 2019. p. 409–418. Citado na página 19.

APÊNDICE A

Árvores de TF desconectadas



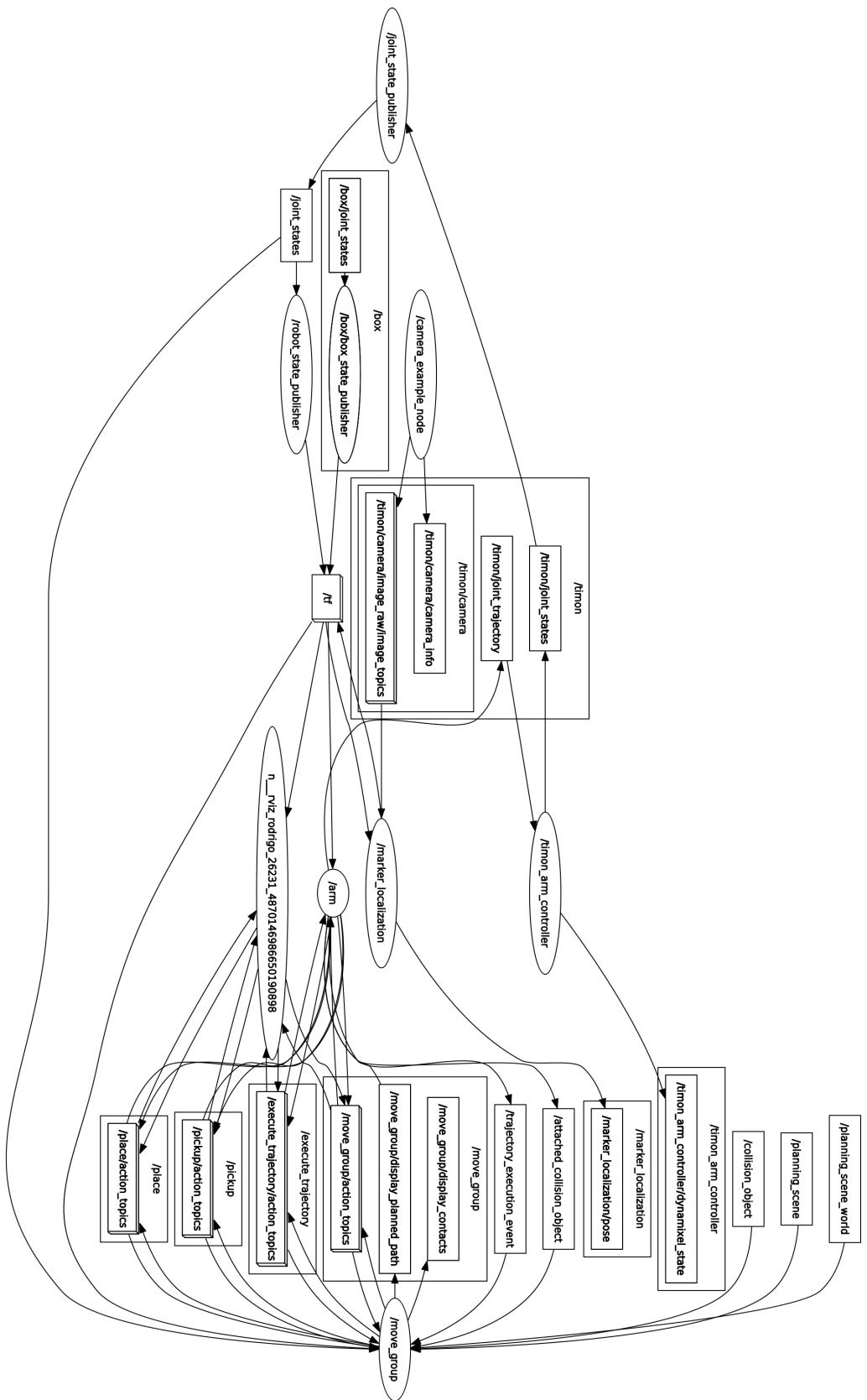
APÊNDICE B

Árvores de TF conectadas



APÊNDICE C

Diagrama de nós do sistema



APÊNDICE D

Propriedades de Massa do JeRoTIMON

Figura 32: Coordenadas.

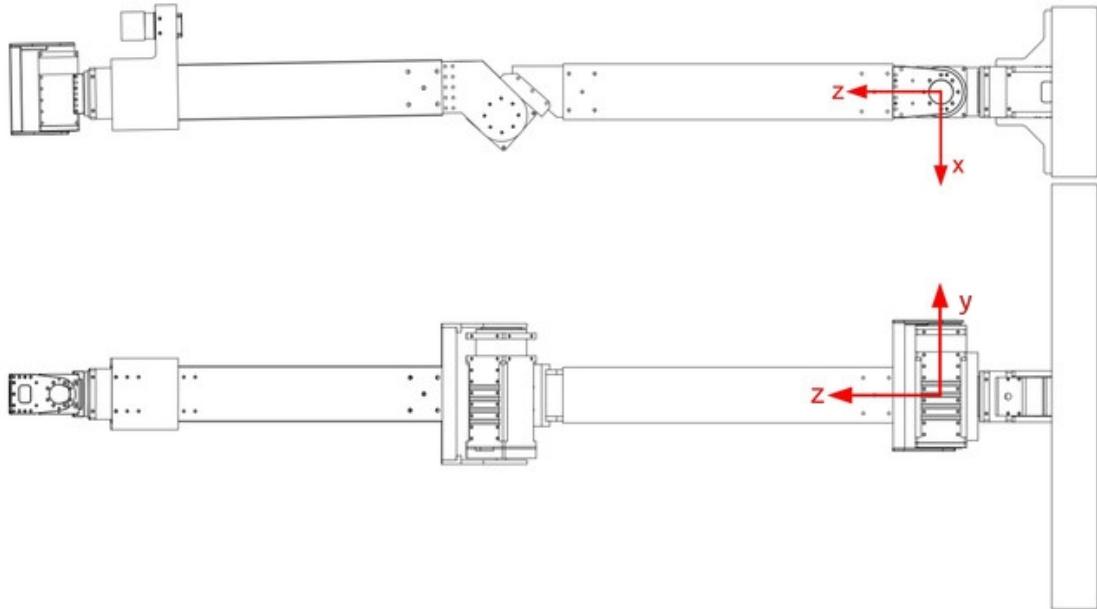
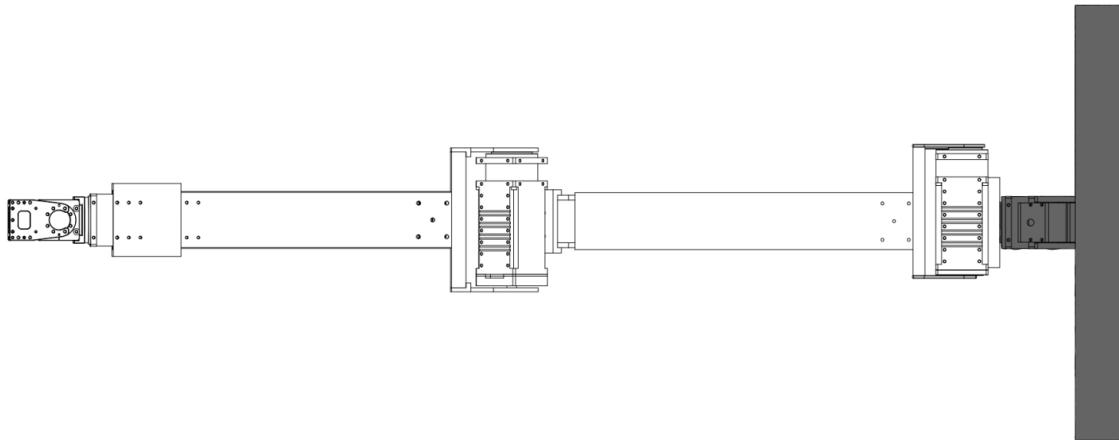


Figura 33: Link 0.



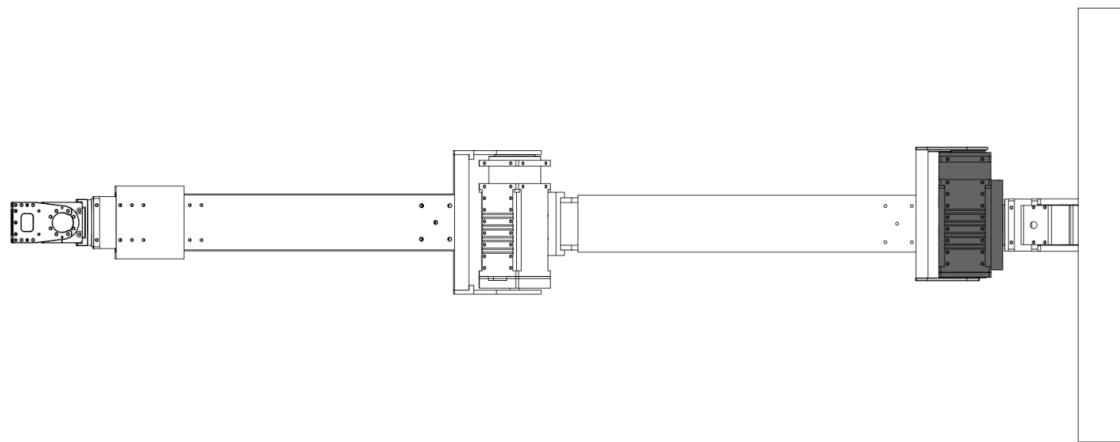
- **Massa:** 3.72384654 kg
- **Volume:** 0.00413037 m^3
- **Área:** 0.28622123 m^2
- **Centro de Massa:**

- X: -0.00000147 m
- Y: 0.00000000 m
- Z: 0.03504069 m

- **Momento de inércia:** kg m^2

- **L_{xx}:** 0.01072644 **L_{xy}:** -9.465e-9 **L_{xz}:** -3.247e-8
- **L_{yx}:** -9.465e-9 **L_{yy}:** 0.04865651 **L_{yz}:** 6.830e-13
- **L_{zx}:** -3.247e-8 **L_{zy}:** 6.830e-13 **L_{zz}:** 0.05388213

Figura 34: Link 1.



- **Massa:** 1.03781084 kg

- **Volume:** 0.00040169 m^3

- **Área:** 0.05853078 m^2

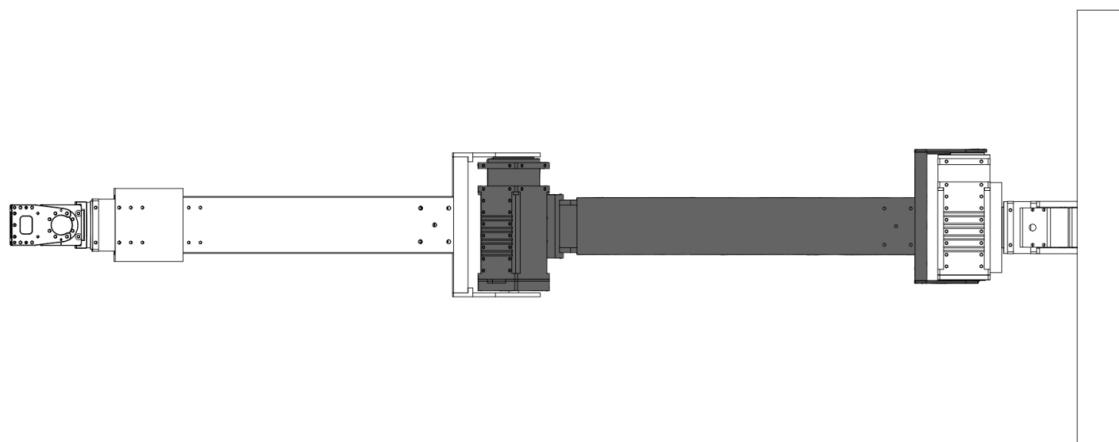
- **Centro de Massa:**

- X: 0.00814457 m
- Y: 4.45597047e - 8 m
- Z: 0.16022275 m

- **Momento de inércia:** kg m^2

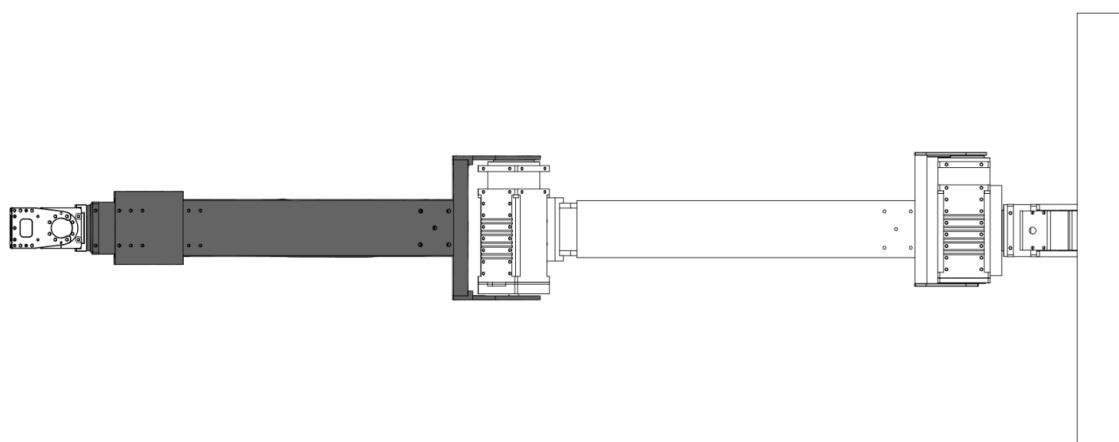
- **L_{xx}:** 0.0005687 **L_{xy}:** 2.602e-10 **L_{xz}:** -0.00004027
- **L_{yx}:** 2.602e-10 **L_{yy}:** 0.00166759 **L_{yz}:** -2.222e-10
- **L_{zx}:** -0.00004027 **L_{zy}:** -2.222e-10 **L_{zz}:** 0.00155695

Figura 35: Link 2.



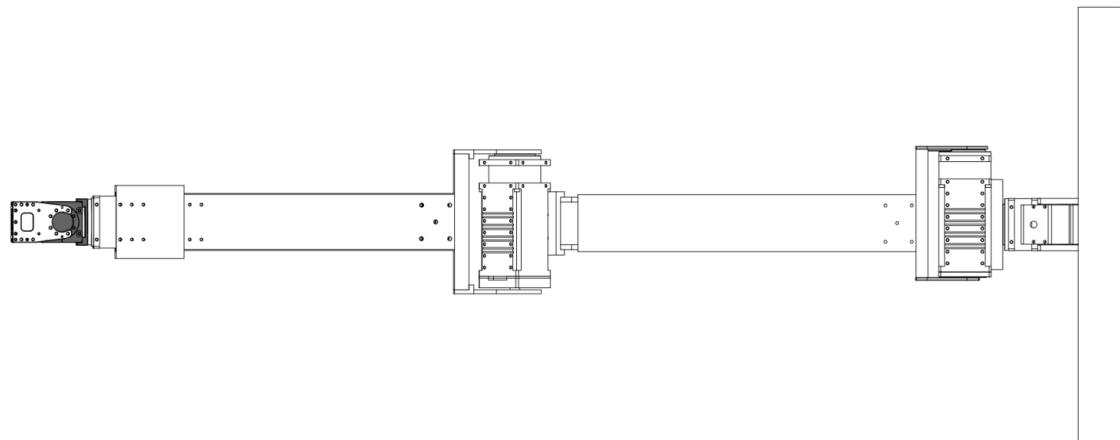
- **Massa:** 2.05026959 kg
- **Volume:** 0.00077667 m^3
- **Área:** 0.29790369 m^2
- **Centro de Massa:**
 - X: 0.00294129 m
 - Y: -0.01058166 m
 - Z: 0.48960322 m
- **Momento de inércia:** kg m^2
 - **L_{xx}:** 0.06174738 **L_{xy}:** -0.00003476 **L_{xz}:** 0.00098626
 - **L_{yx}:** -0.00003476 **L_{yy}:** 0.06319525 **L_{yz}:** 0.00311701
 - **L_{zx}:** 0.00098626 **L_{zy}:** 0.00311701 **L_{zz}:** 0.0034029

Figura 36: Link 3.



- **Massa:** 1.83580505 kg
- **Volume:** 0.00081683 m^3
- **Área:** 0.32055793 m^2
- **Centro de Massa:**
 - X: 0.00257384 m
 - Y: 0.00207951 m
 - Z: 0.91287054 m
- **Momento de inércia:** kg m^2
 - **L_{xx}:** 0.03439469 **L_{xy}:** -0.00000261 **L_{xz}:** -0.00001194
 - **L_{yx}:** -0.00000261 **L_{yy}:** 0.03489631 **L_{yz}:** -0.0001288
 - **L_{zx}:** -0.00001194 **L_{zy}:** -0.0001288 **L_{zz}:** 0.0023687

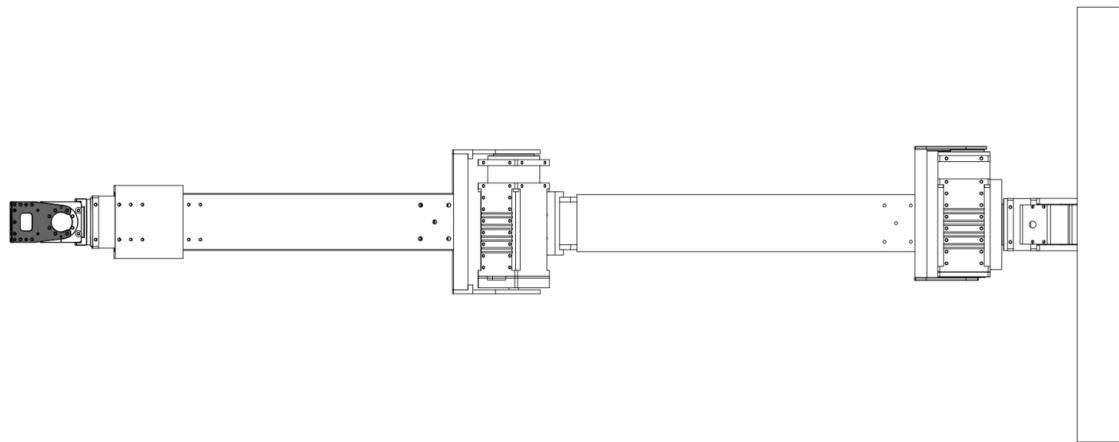
Figura 37: Link 4.



- **Massa:** 0.39425655 kg
- **Volume:** 0.00014602 m^3
- **Área:** 0.03018022 m^2
- **Centro de Massa:**
 - X: 0.00257156 m
 - Y: 0.00133073 m
 - Z: 1.09673426 m
- **Momento de inércia:** kg m^2

- **Lxx:** 0.00028795 **Lxy:** 2.010e-10 **Lxz:** -8.166e-13
- **Lyx:** 2.010e-10 **Lyy:** 0.00011981 **Lyz:** -0.00000446
- **Lzx:** -8.166e-13 **Lzy:** -0.00000446 **Lzz:** 0.0002842

Figura 38: Link 5.



- **Massa:** 0.08694786 kg
- **Volume:** 0.0000322 m^3
- **Área:** 0.02435944 m^2
- **Centro de Massa:**
 - X: 0.00257155 m
 - Y: 0.00684212 m
 - Z: 1.13578562 m
- **Momento de inércia:** kg m^2
 - **Lxx:** 0.00012963 **Lxy:** -5.235e-13 **Lxz:** -9.996e-14
 - **Lyx:** -5.235e-13 **Lyy:** 0.00003703 **Lyz:** 0.00000247
 - **Lzx:** -9.996e-14 **Lzy:** 0.00000247 **Lzz:** 0.00012613

APÊNDICE E

Lista de suportes.

Tabela 23: Tabela de suportes

Imagen	Junta	Fornecedor	Tipo	Modelo	Quantidade	Medidas (mm)
	Id_1	CCRoSA	Cantoneira	Original	2	59 x 59 x 40.5
	Id_2	CCRoSA	Frame Perpendicular	Original	1	92.6 x 51.1 x 12
	Id_2	ROBOTIS	Rolamento	FRP54-I110K	1	54 x 54 x 5
	Id_3	ROBOTIS/CCRoSA	Frame de Rotação	FRP54-H221K/Original	1	105.5 x 138 x 54.3
	Id_3	CCRoSA	Fixador	Original	1	95 x 54 x 54
	Id_3	CCRoSA	Fixador	Original	1	60 x 65.2 x 17
	Id_3	CCRoSA	Calço	Original	1	54 x 54 x 2
	Id_3	CCRoSA	Rolamento	Original	1	54 x 54 x 10
	Id_4	CCRoSA	Frame de Rotação	Original	1	85.03 x 89.46 x 148.2
	Id_5	ROBOTIS	Frame Perpendicular	FRP42-A110K	1	48 x 42 x 11.5
	Id_5	ROBOTIS	Rolamento	FRP42-I110K	1	42 x 42 x 5
	Id_6	ROBOTIS	Frame de Rotação	FRP42-H121K	1	96 x 42 x 56.6

Fonte: Adaptado de ([ROBOTIS, s.d.](#))

APÊNDICE F

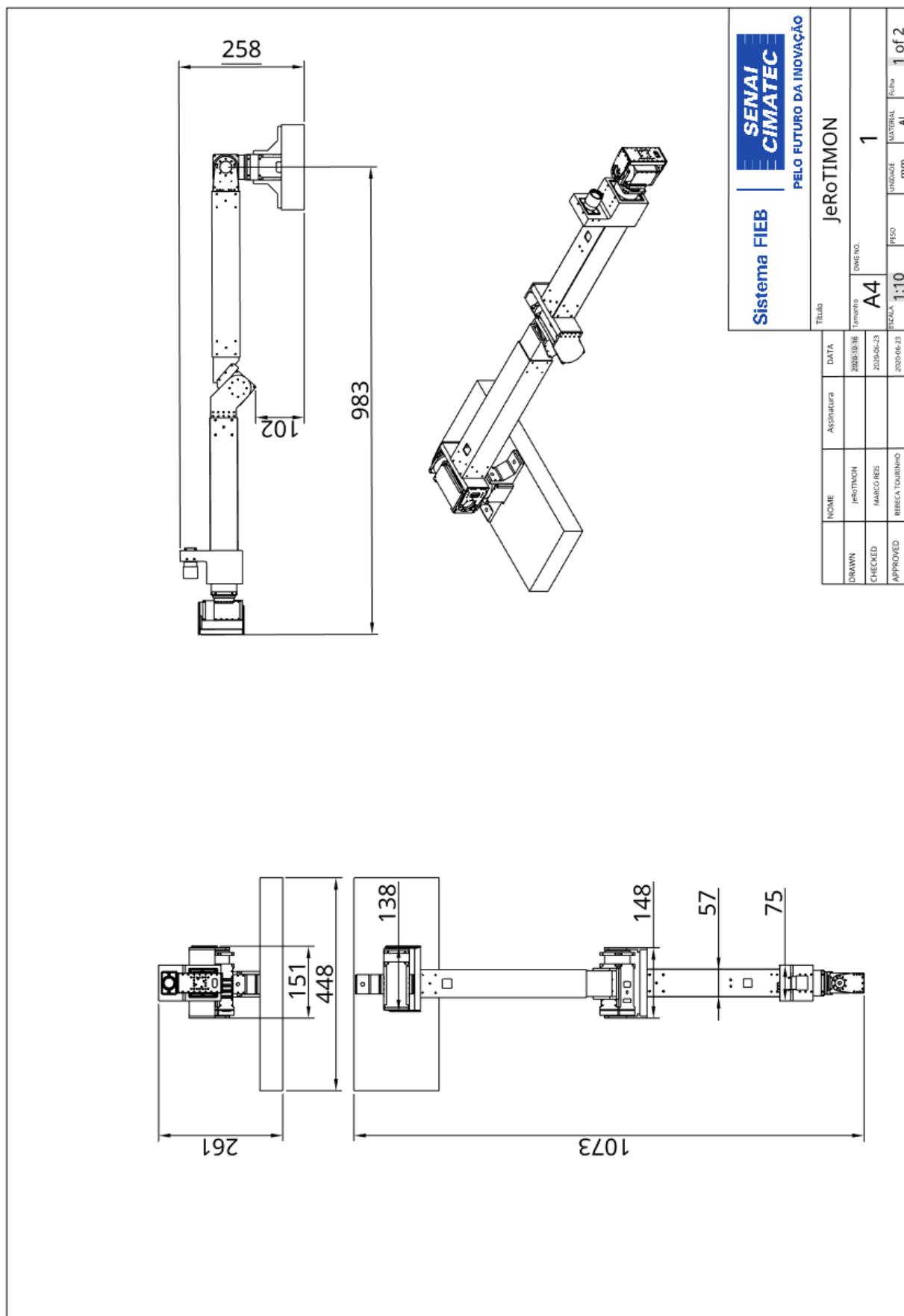
Projeto mecânico

O projeto mecânico do manipulador JeRoTIMON foi desenvolvido em duas etapas: Modelagem 3D e Cálculos Estáticos. A etapa de modelagem levou em consideração, além da análise estrutural e requisitos do cliente, as restrições físicas para que o manipulador pudesse ser anexado no UGV *Warthog*. Por fim, foi realizado os cálculos de esforços estáticos em que analisou-se os esforços sofridos pelo manipulador, na situação em que exige uma maior carga dos atuadores, e com isso garanta a funcionalidade do mesmo. A seguir, será visto o desenho mecânico final do manipulador e como foi realizado a análise estática.

F.1 Desenho mecânico

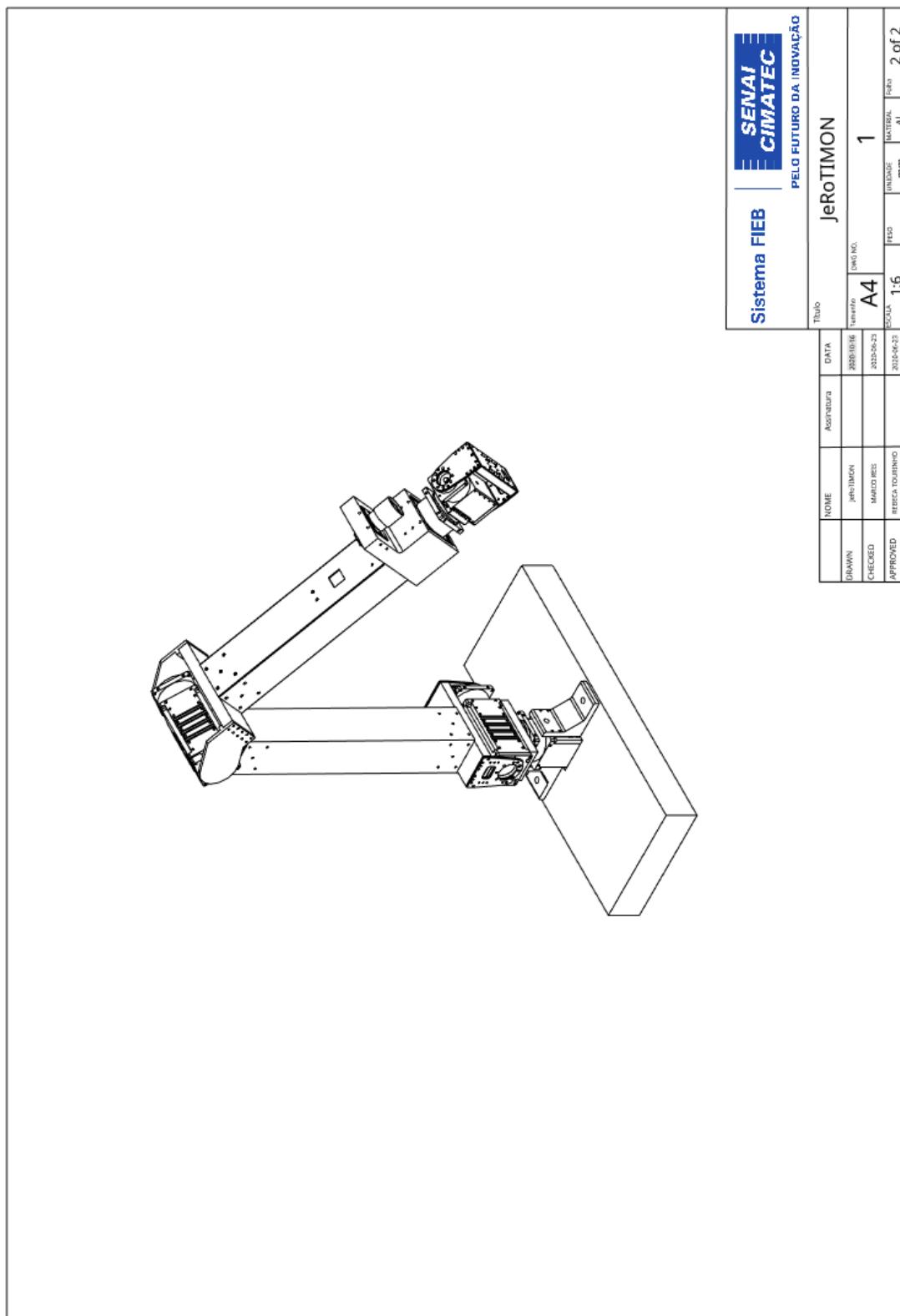
O desenho do manipulador foi todo elaborado utilizando o *software* “*OnShape*”. A Figura 39 mostra o desenho técnico do manipulador com as vistas: frontal, lateral esquerda, superior e isométrica, contendo todos os detalhes necessários para execução do projeto. A figura 40 mostra a vista isométrica expandida do manipulador em sua posição *home*.

Figura 39: Desenho técnico do JeRoTIMON, vistas: frontal, lateral esquerda e superior.



Fonte: Autoria própria.

Figura 40: Desenho técnico do JeRoTIMON, vista isométrica.

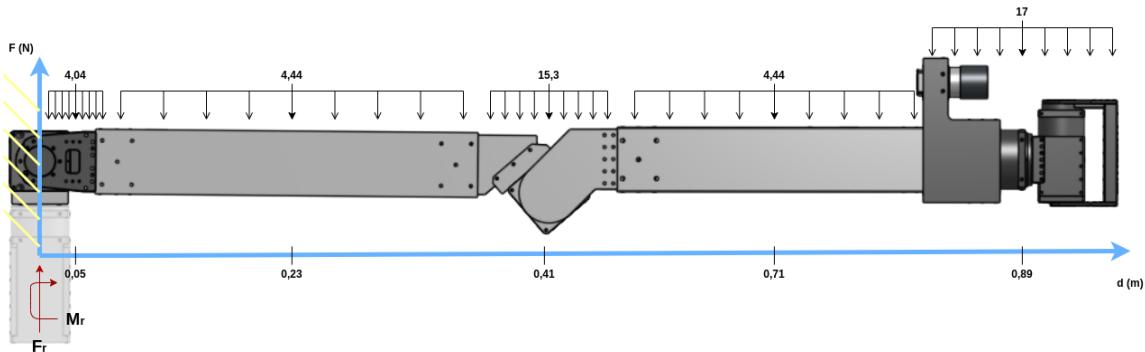


Fonte: Autoria própria.

F.2 Analise estática

Para calcular a carga máxima suportada pelo manipulador foi realizada uma análise estática. Para isso, o mesmo foi posto na situação em que exige um maior esforço das juntas 0 e 1, conforme visto na Figura 41.

Figura 41: Analise estática - condição de maior esforço exigido.



fonte: Autoria Própria

Para analisar os esforços sofridos pela junta 1 (motor 2), admiti-se a extremidade esquerda como engastada e em sua situação de equilíbrio estático, como visto na Figura 41. A força resultante no eixo y é calculada da seguinte forma:

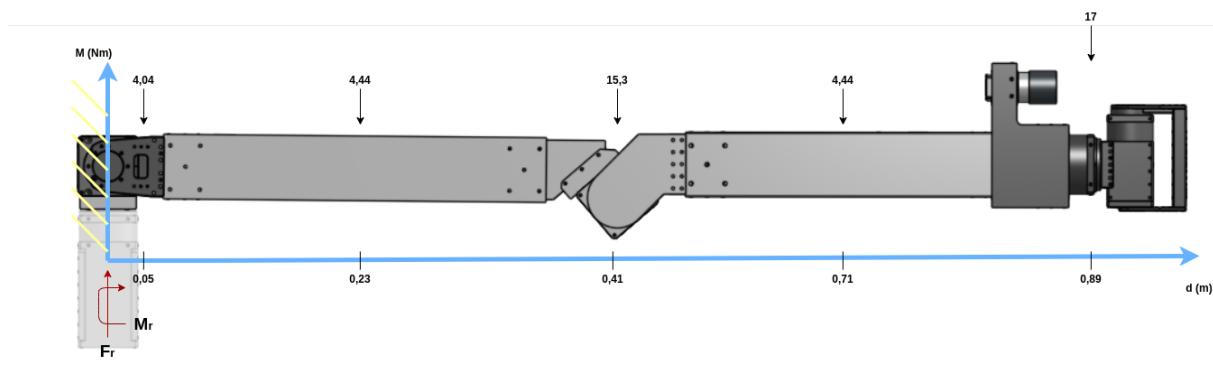
$$\sum F_{ry} = 0$$

$$F_{ry} - 4,04 - 4,44 - 15,3 - 4,44 - 17 = 0$$

$$F_{ry} = 45,22 \text{ [N]}$$

Para calcular o torque sofrido pela junta, as cargas distribuídas de cada componente do robô foram colocadas no seu centróide, conforme visto na Figura 42.

Figura 42: Analise estática - condição de maior esforço exigido (cargas no centróide).



fonte: Autoria Própria

Em seguida, foi calculado o momento torsor gerado nos motores 1 e 2 por cada carga pontual da seguinte maneira:

$$\sum M_r = 0$$

$$M_r - (4,04 \times 0,05) - (4,44 \times 0,23) - (15,3 \times 0,41) - (4,44 \times 0,71) - (17 \times 0,89) = 0$$

$$M_r = \mathbf{25,8} \text{ [N x m]}$$

Por fim, foi calculada a carga máxima suportada pelo manipulador, também conhecida por *payload*. Para este cálculo, foi necessário obter e comparar as informações sobre o torque máximo fornecido pelo motor que compõe a junta 1 (PH54-200-S500-R) que é de 44,7 N x m, segundo o *datasheet* disponível no anexo D. Sendo o alcance máximo de 0,89 m, temos:

$$T_{max/motor} = T_{sofrido} - T_{payload}$$

$$44,7 = 25,8 - (F_{payload} \times 0,89)$$

$$F_{payload} = 21,23[N]$$

De acordo com a Segunda Lei de Newton e considerando a aceleração da gravidade como 10 m/s^2 , obtemos uma carga máxima suportada pelo manipulador de:

$$F_r = m \times a$$

$$21,23 = m \times 10$$

$$m = \mathbf{2,1} \text{ [kg]}$$

APÊNDICE G

Algoritmo de busca e acionamento do painel

```
#include <ros/ros.h>
#include <geometry_msgs/PoseWithCovarianceStamped.h>
#include <moveit/move_group_interface/move_group_interface.h>
#include <moveit/planning_scene_interface/
    planning_scene_interface.h>
#include <moveit_msgs/DisplayRobotState.h>
#include <moveit_msgs/DisplayTrajectory.h>
#include <moveit_msgs/AttachedCollisionObject.h>
#include <moveit_msgs/CollisionObject.h>
#include <moveit_visual_tools/moveit_visual_tools.h>
#include <trajectory_msgs/JointTrajectory.h>
#include <trajectory_msgs/JointTrajectoryPoint.h>
#include <tf/transform_listener.h>
#include <tf/tf.h>
#include <geometry_msgs/Pose.h>
#include <moveit_msgs/PlanningScene.h>
#include <moveit_msgs/ApplyPlanningScene.h>

int main(int argc, char** argv)
{
    ros::init(argc, argv, "push_button_real");
    ros::NodeHandle nh;
    ros::AsyncSpinner spinner(2);
    spinner.start();
    static const std::string PLANNING_GROUP = "arm";
    moveit::planning_interface::MoveGroupInterface move_group(
        PLANNING_GROUP);
    moveit::planning_interface::PlanningSceneInterface
        planning_scene_interface;
    const robot_state::JointModelGroup* joint_model_group =
        move_group.getCurrentState()->getJointModelGroup(
            PLANNING_GROUP);
    namespace rvt = rviz_visual_tools;
    moveit_visual_tools::MoveItVisualTools visual_tools(
        "invisible_link");
```

```

visual_tools.deleteAllMarkers();
visual_tools.loadRemoteControl();
Eigen::Isometry3d text_pose = Eigen::Isometry3d::Identity();
text_pose.translation().z() = 1.75;
visual_tools.publishText(text_pose, "MoveGroupInterface Demo",
    rvt::WHITE, rvt::XLARGE);
visual_tools.trigger();
ROS_INFO_NAMED( "Reference frame: %s", move_group.
    getPlanningFrame().c_str());
ROS_INFO_NAMED( "End effector link: %s", move_group.
    getEndEffectorLink().c_str());

ros::Time start_time = ros::Time::now();

//JUNTAS
moveit::core::RobotStatePtr current_state = move_group.
    getCurrentState();
std::vector<double> joint_group_positions;
current_state->copyJointGroupPositions(joint_model_group,
    joint_group_positions); //COMANDO NO DOMÍNIO DAS JUNTAS
joint_group_positions[0] = 0.0;
joint_group_positions[1] = -0.12;
joint_group_positions[2] = 2.2;
joint_group_positions[3] = 1.57;
joint_group_positions[4] = 0.68;
move_group.setJointValueTarget(joint_group_positions);
moveit::planning_interface::MoveGroupInterface::Plan my_plan;
move_group.plan(my_plan);
while(move_group.plan(my_plan).val == -1){
    move_group.plan(my_plan);
}
//move_group.execute(my_plan)

sleep(5);

// Define variables for xyz for the box
double transform_fake_button_x;
double transform_fake_button_y;
double transform_fake_button_z;

```

```

double transform_fake_button_or_x;
double transform_fake_button_or_y;
double transform_fake_button_or_z;
double transform_fake_button_or_w;
// Define a collision object ROS message.
moveit_msgs::CollisionObject collision_object; //floor
collision_object.header.frame_id = move_group.getPlanningFrame()
();
moveit_msgs::CollisionObject collision_object1; //box
collision_object1.header.frame_id = move_group.
getPlanningFrame();
// The id of the object is used to identify it.
collision_object.id = "floor";
collision_object1.id = "box";
// Define a box to add to the world.
// floor
shape_msgs::SolidPrimitive primitive;
primitive.type = primitive.BOX;
primitive.dimensions.resize(3);
primitive.dimensions[0] = 2.0;
primitive.dimensions[1] = 3.0;
primitive.dimensions[2] = 0.0;
// Define a pose for the box (specified relative to frame_id)
geometry_msgs::Pose box_pose;
box_pose.orientation.w = 0.0;
box_pose.position.x = 0.0;
box_pose.position.y = 0.0;
box_pose.position.z = 0.0;

tf::TransformListener listener;
//GETTING FAKE BUTTON POSE
tf::StampedTransform transform, transform_button_;
int i=2;
posicao:
try{
    std::cout << i;
    i = i+1;
    listener.waitForTransform("/invisible_link", "/fake_botao",

```

```

        ros::Time(0), ros::Duration(2.0));
listener.lookupTransform("/invisible_link", "/fake_botaao",
                       ros::Time(0), transform);

transform_fake_button_x = transform.getOrigin().x();
transform_fake_button_y = transform.getOrigin().y();
transform_fake_button_z = transform.getOrigin().z();
transform_fake_button_or_x = transform.getRotation().x();
transform_fake_button_or_y = transform.getRotation().y();
transform_fake_button_or_z = transform.getRotation().z();
transform_fake_button_or_w = transform.getRotation().w();

}

catch (tf::TransformException &ex) {

if(i == 1){
moveit::core::RobotStatePtr current_state = move_group.
    getCurrentState();
std::vector<double> joint_group_positions;
current_state->copyJointGroupPositions(joint_model_group,
    joint_group_positions);
joint_group_positions[0] = 0;
joint_group_positions[1] = -0.006;
joint_group_positions[2] = 2.052;
joint_group_positions[3] = 1.57;
joint_group_positions[4] = 0.68;
move_group.setJointValueTarget(joint_group_positions);
move_group.plan(my_plan);
while(move_group.plan(my_plan).val == -1){
    move_group.plan(my_plan);
}
// move_group.execute(my_plan);
sleep (5.0);
}

if(i == 2){
moveit::core::RobotStatePtr current_state = move_group.
    getCurrentState();
std::vector<double> joint_group_positions;

```

```

current_state->copyJointGroupPositions(joint_model_group ,
    joint_group_positions);
joint_group_positions[0] = 0;
joint_group_positions[1] = -0.4999;
joint_group_positions[2] = 2.499;
joint_group_positions[3] = 1.57;
joint_group_positions[4] = 0.68;
move_group.setJointValueTarget(joint_group_positions);
move_group.plan(my_plan);
while(move_group.plan(my_plan).val == -1){
    move_group.plan(my_plan);
}
// move_group.execute(my_plan);
sleep(5.0);
}

if(i == 3){
moveit::core::RobotStatePtr current_state = move_group.
    getCurrentState();
std::vector<double> joint_group_positions;
current_state->copyJointGroupPositions(joint_model_group ,
    joint_group_positions);
joint_group_positions[0] = 0.4;
joint_group_positions[1] = -0.769;
joint_group_positions[2] = 2.757;
joint_group_positions[3] = 1.57;
joint_group_positions[4] = 0.68;
move_group.setJointValueTarget(joint_group_positions);
move_group.plan(my_plan);
while(move_group.plan(my_plan).val == -1){
    move_group.plan(my_plan);
}
// move_group.execute(my_plan);
sleep(5.0);
}

if(i == 4){
moveit::core::RobotStatePtr current_state = move_group.
    getCurrentState();
}

```

```

std :: vector<double> joint_group_positions;
current_state->copyJointGroupPositions(joint_model_group ,
    joint_group_positions);
joint_group_positions[0] = -0.4;
joint_group_positions[1] = -0.769;
joint_group_positions[2] = 2.757;
joint_group_positions[3] = 1.57;
joint_group_positions[4] = 0.68;
move_group.setJointValueTarget(joint_group_positions);
move_group.plan(my_plan);
while(move_group.plan(my_plan).val == -1){
    move_group.plan(my_plan);
}
// move_group.execute(my_plan);
i=0;
sleep(5.0);
}

goto posicao;

}

//GETTING BUTTON POSE
try{
// ros::Time now = ros::Time::now();
listener.waitForTransform("/invisible_link", "/botao_press",
    ros::Time(0), ros::Duration(2.0));
listener.lookupTransform("/invisible_link", "/botao_press",
    ros::Time(0), transform_button_);
std::cout<<"Pose do botão: x: "<< transform_button_.getOrigin()
().x()<<
    " y: "<< transform_button_.getOrigin()
().y()<<
    " z: "<< transform_button_.getOrigin()
().z();

}

catch (tf::TransformException &ex) {
i=0;
goto posicao;
}

```

```

}

ROS_INFO( "Aruco Detectado" );
ros :: Duration delta_t1 = ros :: Time :: now() - start_time;
double delta_t1_sec = delta_t1.toSec();

std :: cout << "Tempo de busca: " << delta_t1_sec;

if ( transform .getRotation () .x () <= 0.5) {
    ROS_INFO( "Caixa na Horizontal" );
    // box
    shape_msgs :: SolidPrimitive primitive1;
    primitive1 .type = primitive .BOX;
    primitive1 .dimensions .resize (3);
    primitive1 .dimensions [0] = 0.2;
    primitive1 .dimensions [1] = 0.3;
    primitive1 .dimensions [2] = 0.2;
    // Define a pose for the box (specified relative to frame_id
    )
    geometry_msgs :: Pose box_pose1;
    box_pose1 .position .x = transform_fake_button_x ;
    box_pose1 .position .y = transform_fake_button_y;
    box_pose1 .position .z = transform_fake_button_z -0.354;
    box_pose1 .orientation .w = transform_fake_button_or_w;
    box_pose1 .orientation .x = transform_fake_button_or_x;
    box_pose1 .orientation .y = transform_fake_button_or_y;
    box_pose1 .orientation .z = transform_fake_button_or_z;
    //Collision box
    collision_object .primitives .push_back ( primitive );
    collision_object .primitive_poses .push_back ( box_pose );
    collision_object .operation = collision_object .ADD;
    collision_object1 .primitives .push_back ( primitive1 );
    collision_object1 .primitive_poses .push_back ( box_pose1 );
    collision_object1 .operation = collision_object .ADD;

    std :: vector<moveit_msgs :: CollisionObject> collision_objects ;
    collision_objects .push_back ( collision_object );
    collision_objects .push_back ( collision_object1 );
}

```

```

planning_scene_interface.addCollisionObjects(
    collision_objects);

geometry_msgs::Pose target_pose;
target_pose.orientation.w = transform.getRotation().w();
target_pose.orientation.x = transform.getRotation().x();
target_pose.orientation.y = transform.getRotation().y();
target_pose.orientation.z = transform.getRotation().z();
target_pose.position.x = transform.getOrigin().x();
target_pose.position.y = transform.getOrigin().y();
target_pose.position.z = transform.getOrigin().z();
move_group.setGoalPositionTolerance(0.001);
move_group.setGoalOrientationTolerance(0.35);
move_group.setPlanningTime(25);
move_group.setPoseTarget(target_pose);

move_group.plan(my_plan);
while (move_group.plan(my_plan).val != 1){
    move_group.plan(my_plan);
}
// move_group.execute(my_plan);

sleep(16.0);

moveit::core::RobotStatePtr current_state_1 = move_group.
    getCurrentState();
std::vector<double> joint_group_positions_garra;
current_state_1->copyJointGroupPositions(joint_model_group,
    joint_group_positions_garra);
joint_group_positions_garra[3] = 1.57;
joint_group_positions_garra[4] = 1.21;
move_group.setJointValueTarget(joint_group_positions_garra);
moveit::planning_interface::MoveGroupInterface::Plan my_plan
;
move_group.plan(my_plan);
while(move_group.plan(my_plan).val == -1){
    move_group.plan(my_plan);
}

```

```

}

sleep(11.0);
// move_group.execute(my_plan);

//PRESS BUTTON

target_pose.position.z -=0.175;
std::vector<geometry_msgs::Pose> waypoints;
waypoints.push_back(target_pose);
moveit_msgs::RobotTrajectory trajectory;
const double jump_threshold = 0.0;
const double eef_step = 0.1;
move_group.computeCartesianPath(waypoints, eef_step,
                               jump_threshold, trajectory);
sleep(5.0);

geometry_msgs::Pose pose_real;
pose_real = move_group.getCurrentPose().pose;
std::cout << "Pose Real: " << pose_real.position;

target_pose.position.z +=0.11;
move_group.setPoseTarget(target_pose);
move_group.plan(my_plan);
while (move_group.plan(my_plan).val != 1){
    move_group.plan(my_plan);
}

else
{ // box
    shape_msgs::SolidPrimitive primitive1;
    primitive1.type = primitive.BOX;
    primitive1.dimensions.resize(3);
    primitive1.dimensions[0] = 0.2;
    primitive1.dimensions[1] = 0.3;
    primitive1.dimensions[2] = 0.2;
    // Define a pose for the box (specified relative to frame_id
        )
}

```

```

geometry_msgs::Pose box_pose1;
    box_pose1.orientation.w = transform_fake_button_or_w;
    box_pose1.orientation.x = transform_fake_button_or_x;
    box_pose1.orientation.y = transform_fake_button_or_y;
    box_pose1.orientation.z = transform_fake_button_or_z;
    box_pose1.position.x = transform_fake_button_x -0.1;
    box_pose1.position.y = transform_fake_button_y+0.318;
    box_pose1.position.z = transform_fake_button_z - 0.1;
//Collision box
collision_object.primitives.push_back(primitive);
collision_object.primitive_poses.push_back(box_pose);
collision_object.operation = collision_object.ADD;
collision_object1.primitives.push_back(primitive1);
collision_object1.primitive_poses.push_back(box_pose1);
collision_object1.operation = collision_object.ADD;

std::vector<moveit_msgs::CollisionObject> collision_objects;
collision_objects.push_back(collision_object);
collision_objects.push_back(collision_object1);

planning_scene_interface.addCollisionObjects(
    collision_objects);

geometry_msgs::Pose target_pose;

target_pose.orientation.w = transform.getRotation().w();
target_pose.orientation.x = transform.getRotation().x();
target_pose.orientation.y = transform.getRotation().y();
target_pose.orientation.z = transform.getRotation().z();
target_pose.position.x = transform.getOrigin().x();
target_pose.position.y = transform.getOrigin().y();
target_pose.position.z = transform.getOrigin().z();
move_group.setGoalPositionTolerance(0.001);
move_group.setGoalOrientationTolerance(1.0);
move_group.setPlanningTime(25);
move_group.setPoseTarget(target_pose);
move_group.plan(my_plan);
while (move_group.plan(my_plan).val != 1){

```

```

move_group . plan ( my_plan ) ;
}
// move_group . execute ( my_plan ) ;
sleep ( 10.0 ) ;
moveit :: core :: RobotStatePtr current_state_1 = move_group .
    getCurrentState () ;
std :: vector<double> joint_group_positions_garra ;
current_state_1 ->copyJointGroupPositions ( joint_model_group ,
    joint_group_positions_garra ) ;
joint_group_positions_garra [ 3 ] = - 2.12 ;
joint_group_positions_garra [ 4 ] = 1.21 ;
move_group . setJointValueTarget ( joint_group_positions_garra ) ;
moveit :: planning_interface :: MoveGroupInterface :: Plan my_plan
;
move_group . plan ( my_plan ) ;
while ( move_group . plan ( my_plan ) . val == - 1 ){
    move_group . plan ( my_plan ) ;
}
sleep ( 10.0 ) ;

//PRESS BUTTON
move_group . setGoalOrientationTolerance ( 0.9 ) ;
float or_w , or_x , or_y , or_z , x , y ;
or_x = move_group . getCurrentPose ( ) . pose . orientation . x ;
or_y = move_group . getCurrentPose ( ) . pose . orientation . y ;
or_z = move_group . getCurrentPose ( ) . pose . orientation . z ;
or_w = move_group . getCurrentPose ( ) . pose . orientation . w ;
target_pose . orientation . w = or_w ;
target_pose . orientation . x = or_x ;
target_pose . orientation . y = or_y ;
target_pose . orientation . z = or_z ;
target_pose . position . y += 0.172 ;
target_pose . position . z += 0.018 ;
move_group . setPoseTarget ( target_pose ) ;
// moveit :: planning_interface :: MoveGroupInterface :: Plan
my_plan4 ;
move_group . plan ( my_plan ) ;
while ( move_group . plan ( my_plan ) . val != 1 ){
    move_group . plan ( my_plan ) ;
}

```

```

    }

    sleep(10.0);

    target_pose.position.y -=0.2;
    move_group.setPoseTarget(target_pose);
    move_group.plan(my_plan);
    while (move_group.plan(my_plan).val != 1){
        move_group.plan(my_plan);
    }
    // move_group.execute(my_plan);

}

//BACK TO HOME POSITION

current_state->copyJointGroupPositions(joint_model_group,
    joint_group_positions);
joint_group_positions[0] = 0.0;
joint_group_positions[1] = -0.12;
joint_group_positions[2] = 2.2;
joint_group_positions[3] = 0;
joint_group_positions[4] = 0.0;
move_group.setJointValueTarget(joint_group_positions);

move_group.plan(my_plan);
while (move_group.plan(my_plan).val != 1){
    move_group.plan(my_plan);
}
// move_group.execute(my_plan);
sleep(5.0);
ros::Duration delta_t = ros::Time::now() - start_time;
double delta_t_sec = delta_t.toSec();

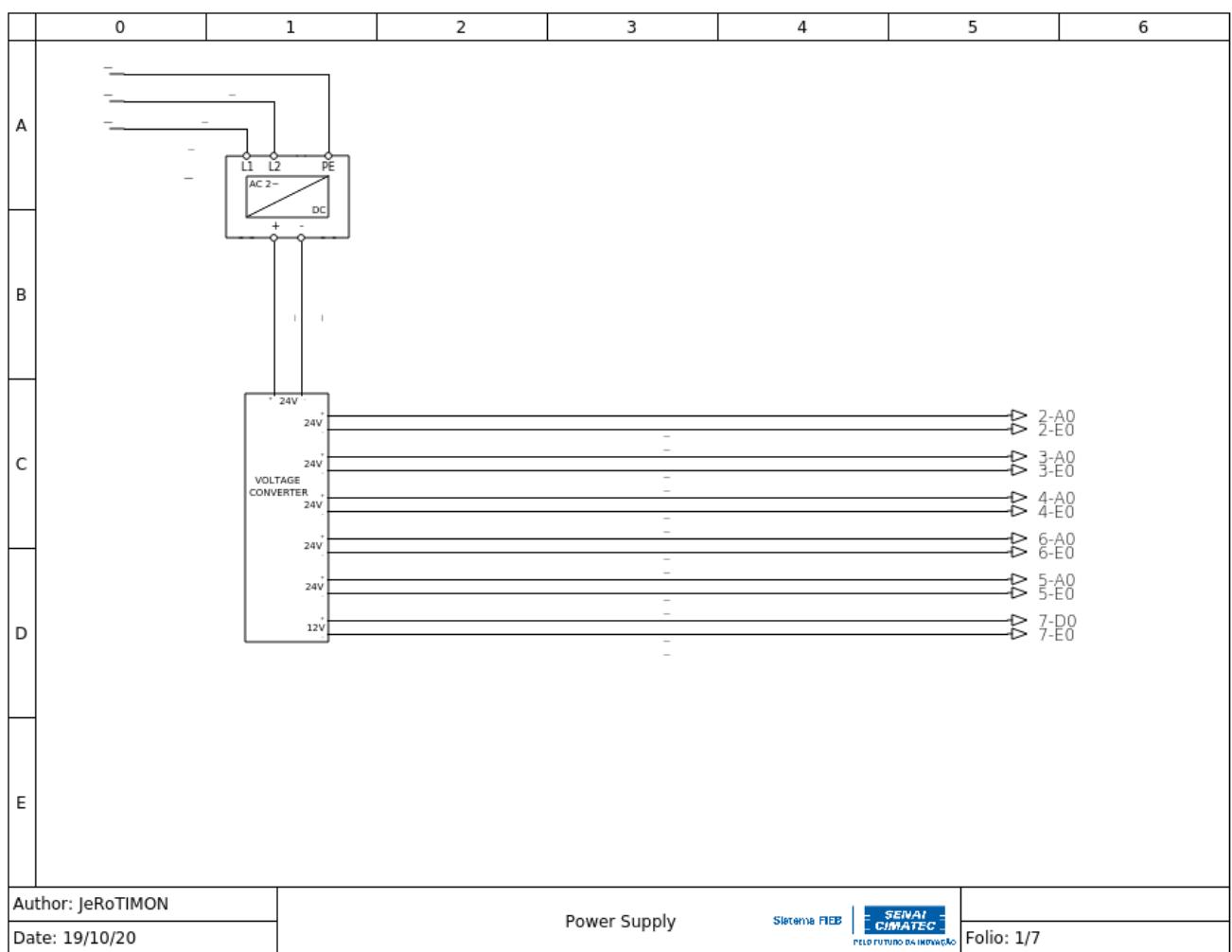
std::cout<< "Tempo de missão: " << delta_t_sec;

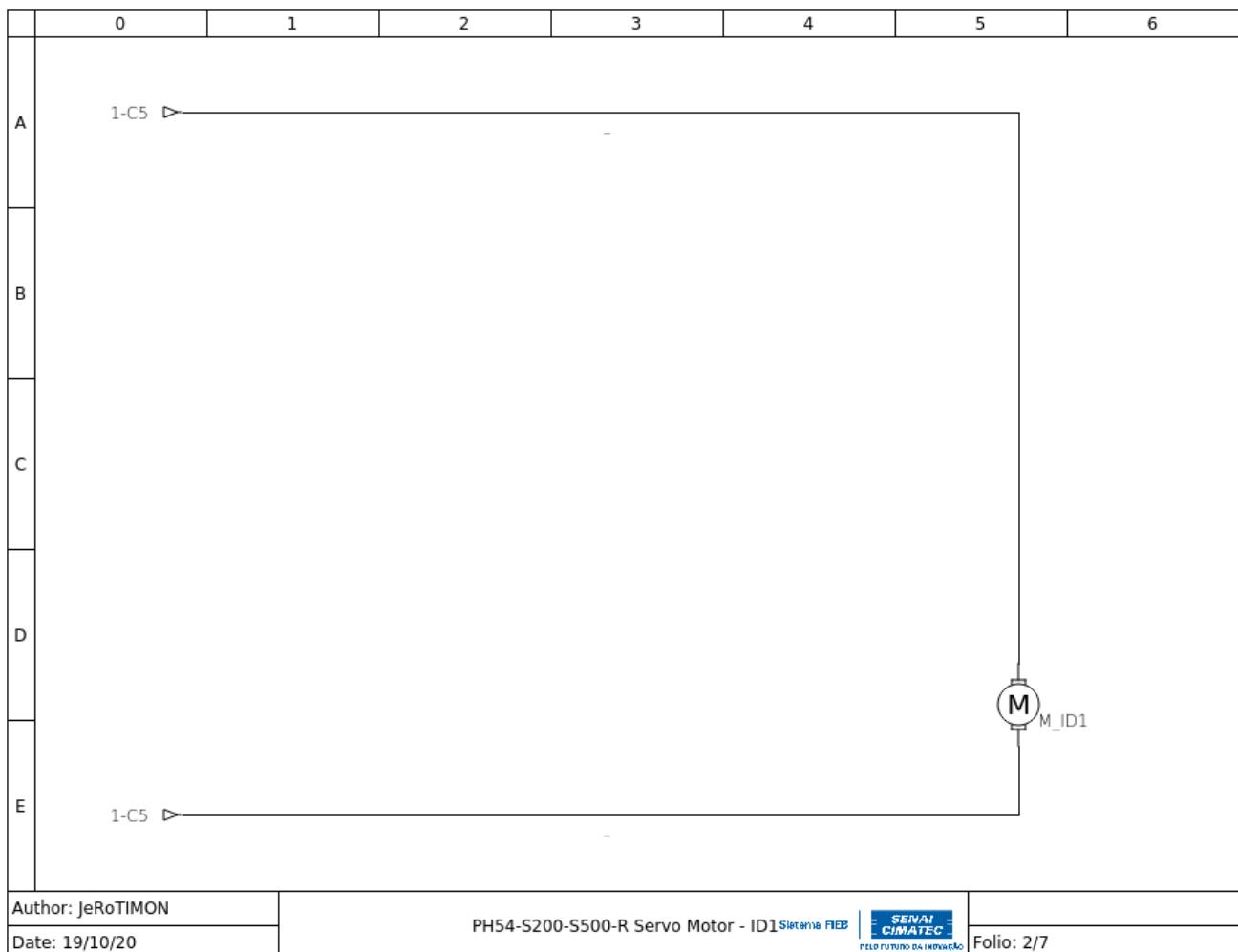
ros::shutdown();
return 0;
}

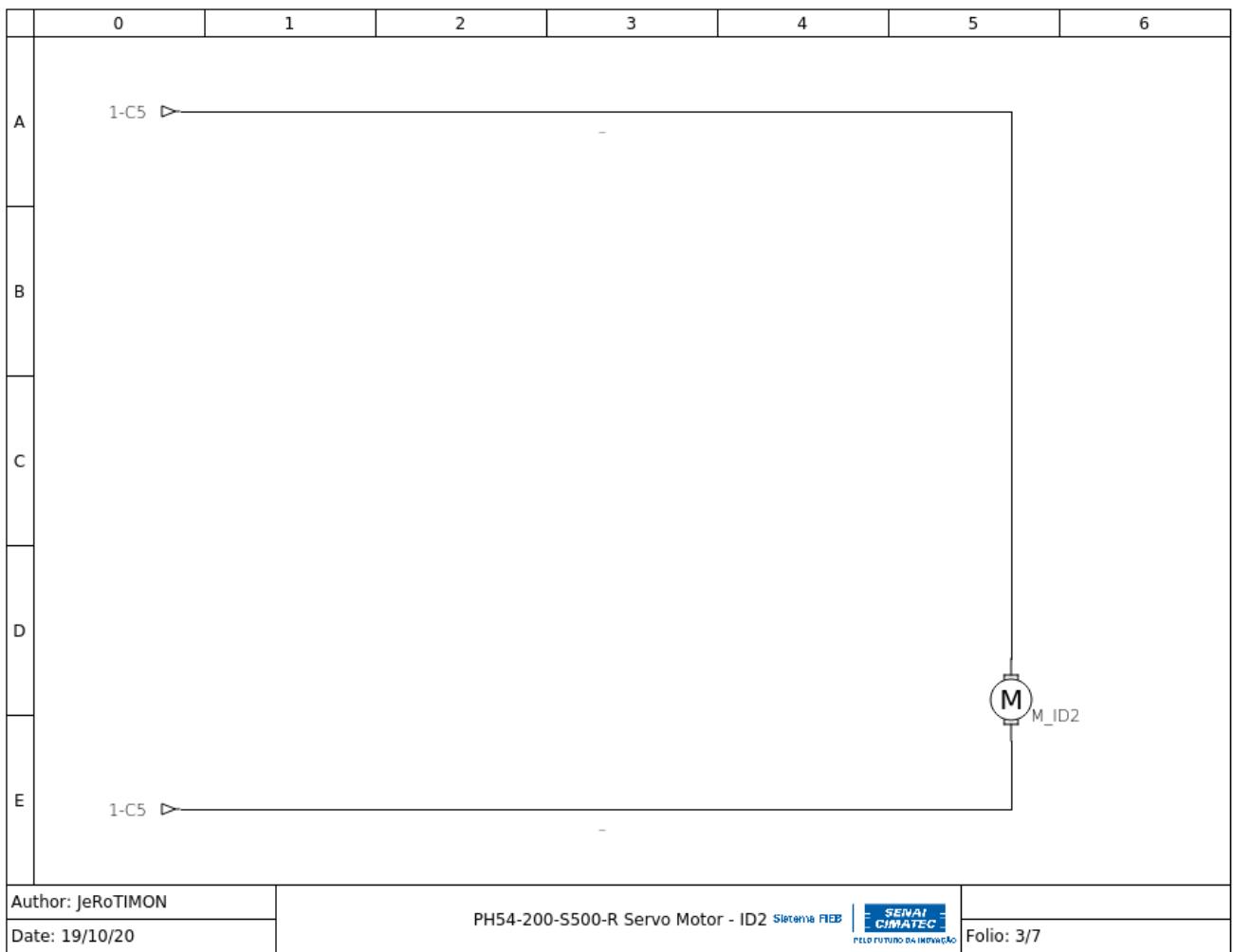
```

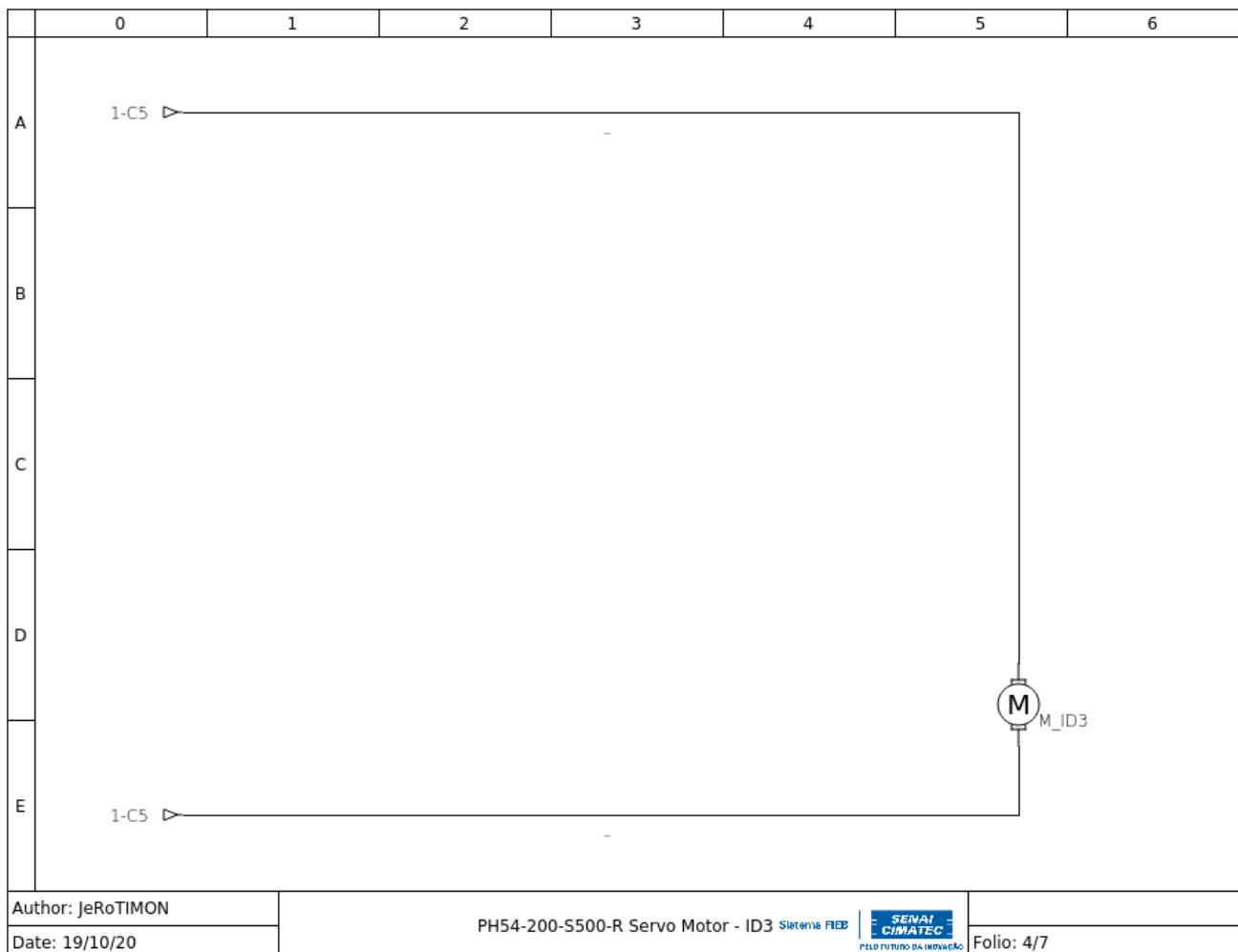
APÊNDICE H

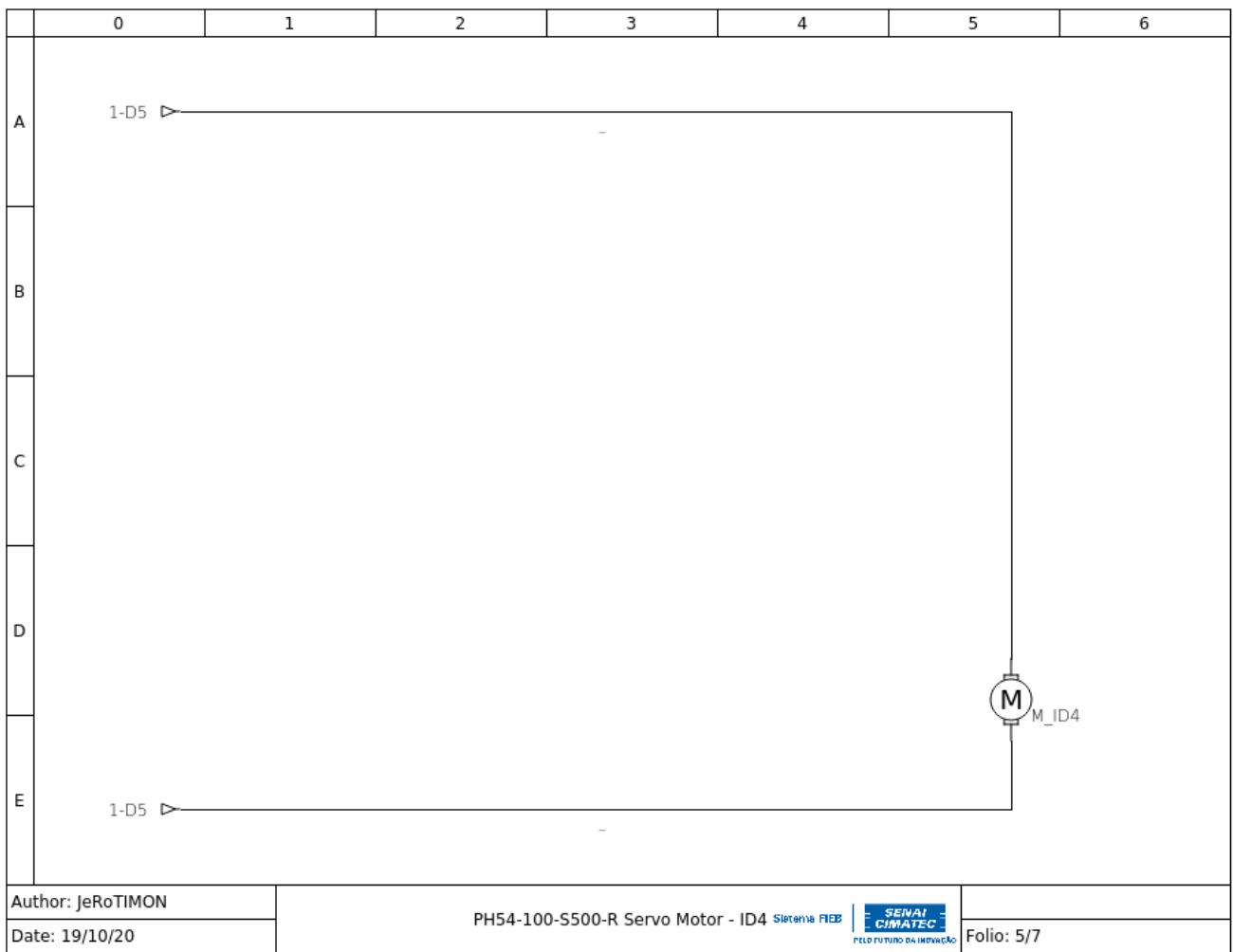
Diagrama elétrico do JeRoTIMON

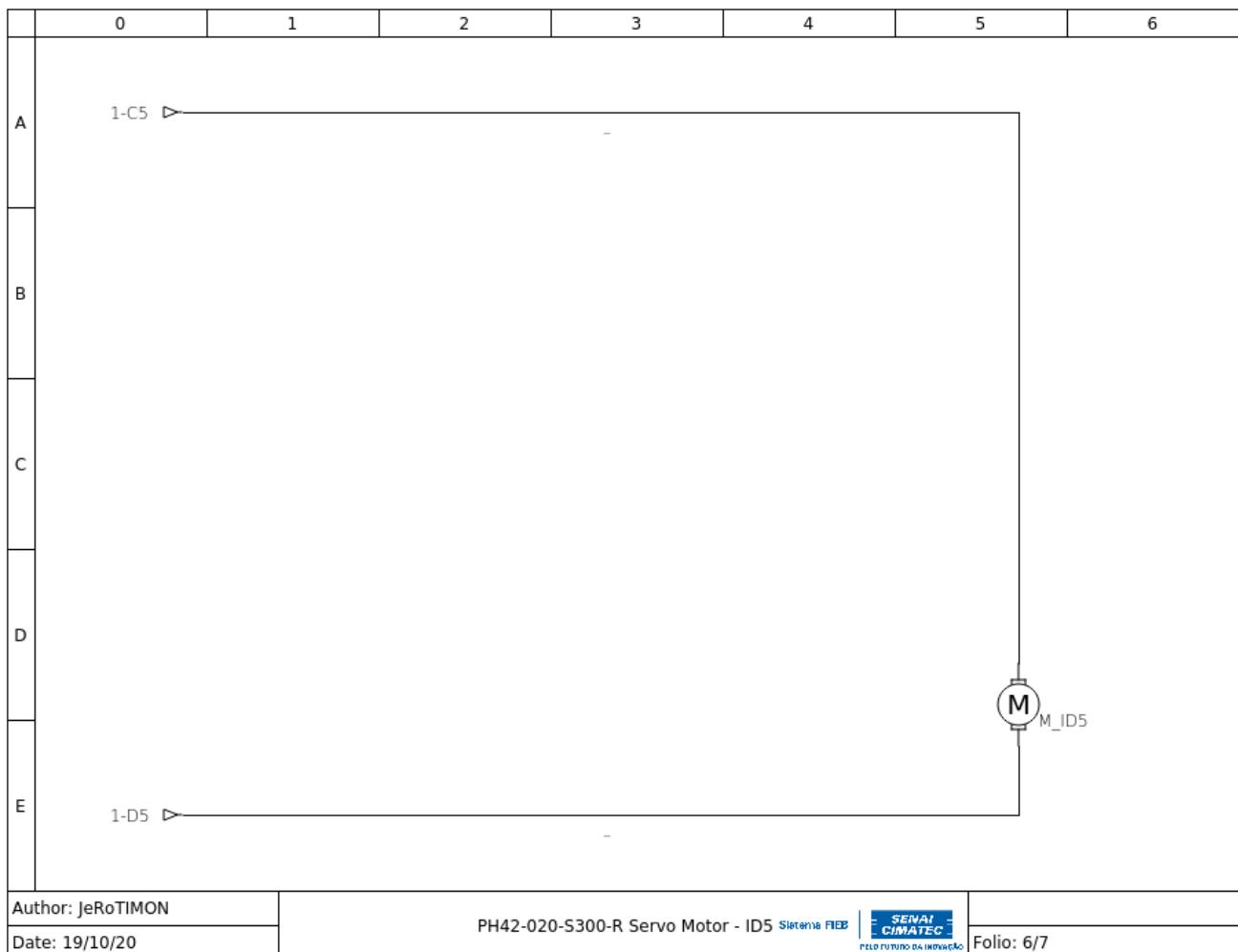








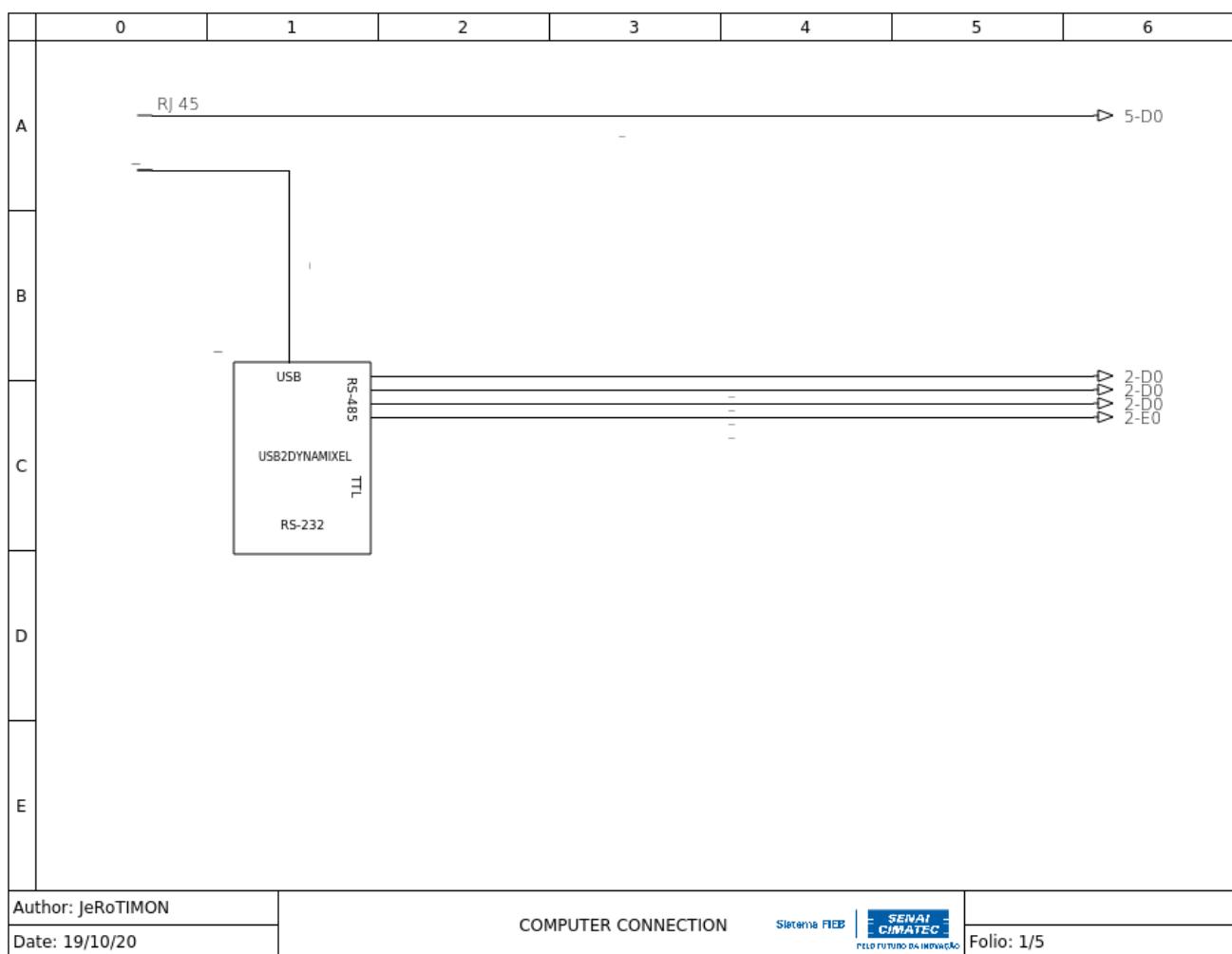


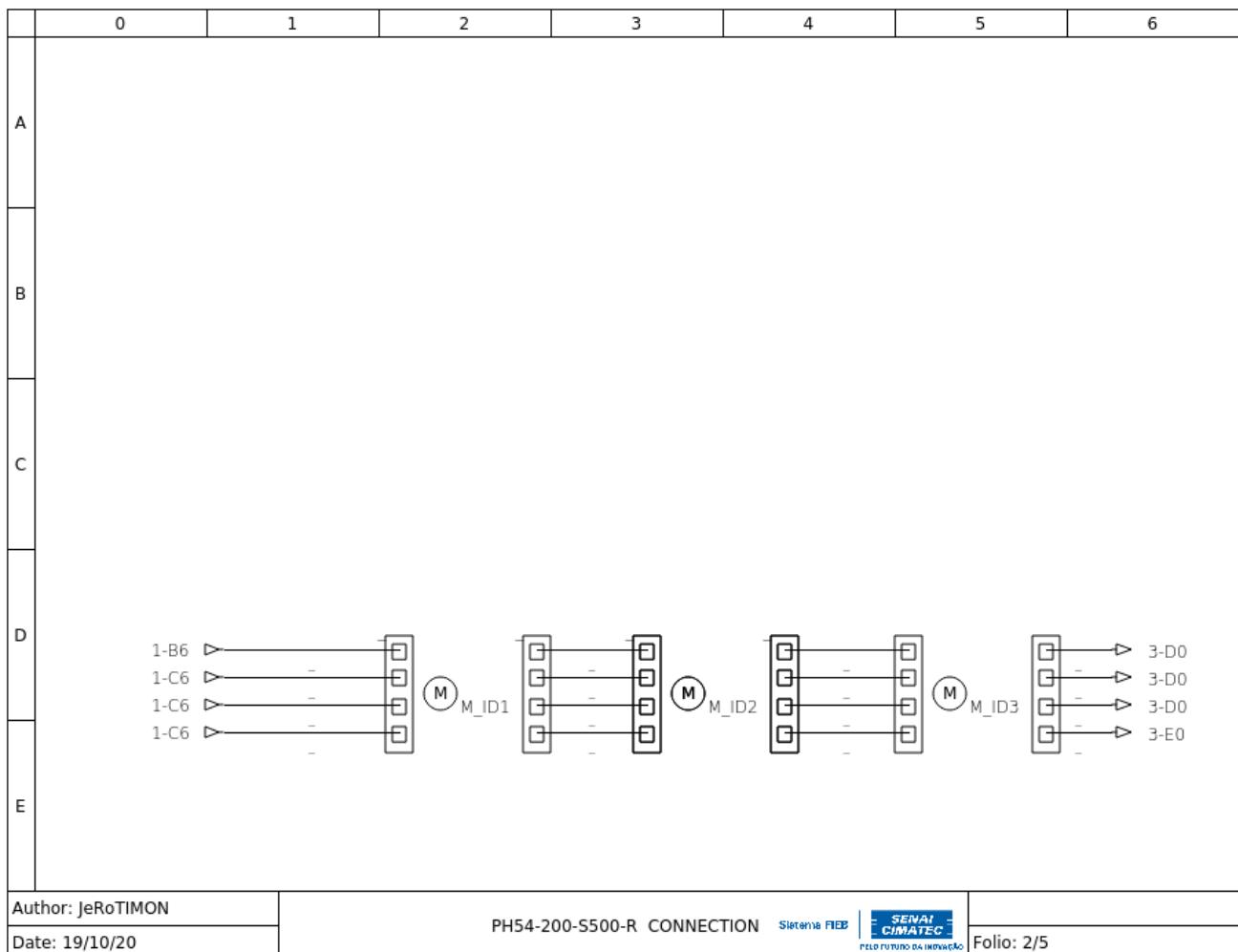


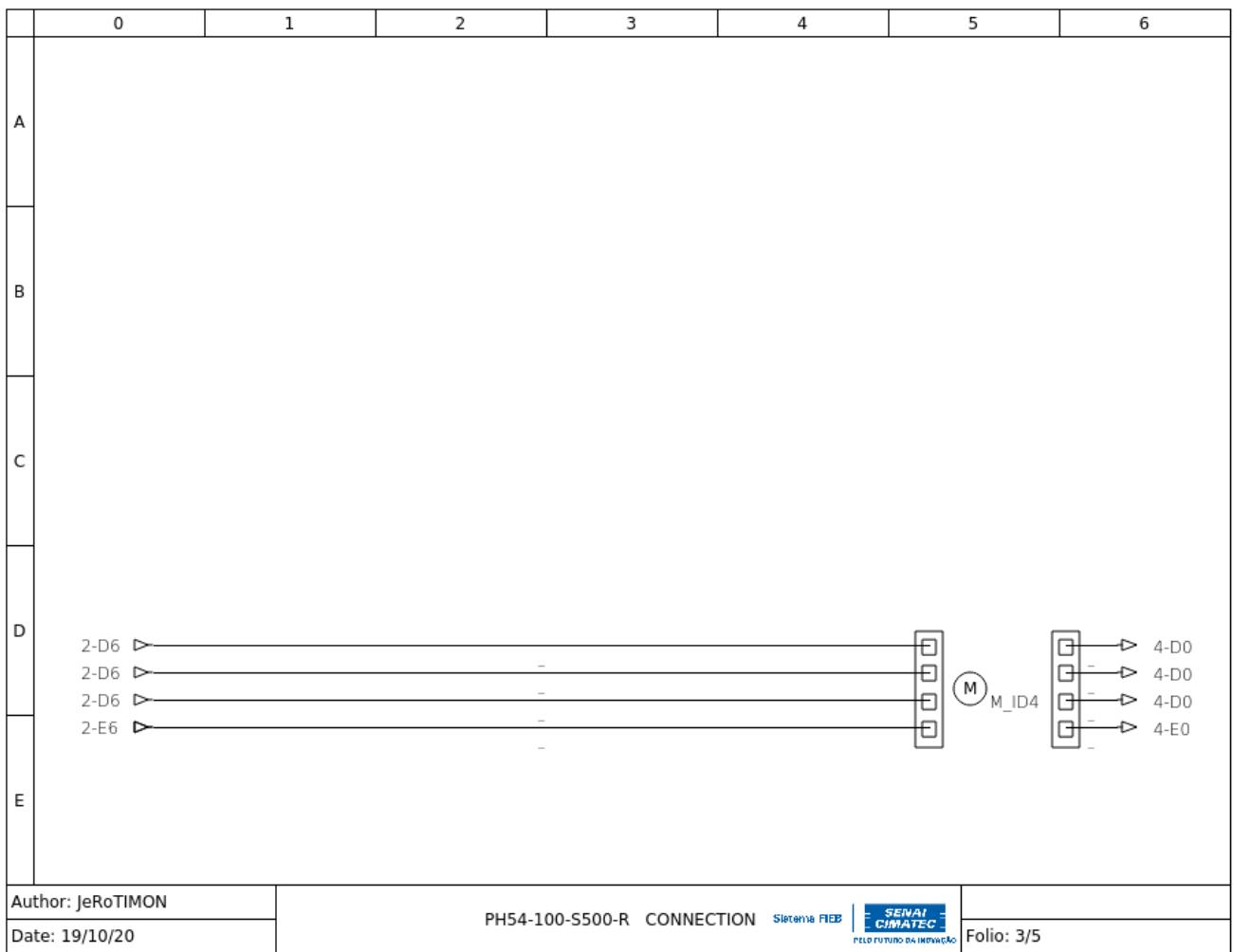
	0	1	2	3	4	5	6
A							
B							
C							
D							
1-D5	►						
1-D5	►		—				
E			—				
Author: JeRoTIMON	Camera Teledyne Genie Nano		Sistema FIEP	 SENAI CIMATEC PELO FUTURO DA INOVAÇÃO			
Date: 19/10/20						Folio: 7/7	

APÊNDICE I

Diagrama de conexão do JeRoTIMON







	0	1	2	3	4	5	6
A							
B							
C							
D	3-D6 ►						
	3-D6 ►	-					
	3-D6 ►	-					
	3-E6 ►	-					
E							
Author: JeRoTIMON							
Date: 19/10/20							
PH42-020-S300-R CONNECTION Sistema FIEB				 SENAI CIMATEC PELÔ FUTURO DA INovaçãO		Folio: 4/5	

	0	1	2	3	4	5	6
A							
B							
C							
D	1-A6	►		—			
E							
Author: JeRoTIMON	CAMERA CONNECTION		Sistema FIEP	SENAI CIMATEC	PELO FUTURO DA INovação	Folio: 5/5	
Date: 19/10/20							

ANEXO A

Especificações da câmera Dalsa Genie Nano

Specifications: M2590, M2590-NIR, C2590

Supported Features	M2590, M2590-NIR	Nano-C2590	
Resolution	2592 x 2048		
Sensor	OnSemi Python5000 P1 (5.1M)		
Pixel Size	4.8 µm x 4.8 µm		
Shutter type	Full frame electronic global shutter function		
Full Well charge	10ke (max)		
Firmware option (Field programmable)	Standard Design Monochrome (factory default)	Standard Design Bayer (factory default)	RGB-Output Design
Max. Internal Frame Rate Full Resolution (2592 x 2048)	51.8 fps (Fast Readout Enable) 24.7 fps (Normal Readout Enable)		
Maximum Sustained Frame Rate Output (with TurboDrive v1)*	42.7 fps (8-bit) 24.9 fps (10-bit)		N/A
Maximum Sustained Frame Rate Output (without TurboDrive)	22 fps (8-bit)		5.5 fps (RGBA) 8.7 fps (RGB) 11 fps (Yuv422) 22 fps (8-bit mono)
Pixel Data Formats	Mono 8-bit Mono 10-bit	Bayer 8-Bit Bayer 10-Bit	RGBA 32-bit RGB 24-bit Yuv422 16-bit Mono 8-bit
Trigger to Exposure Minimum delay (Synchronous Exposure Alignment)	8 µs if exposureAlignment = Synchronous With No Overlap between the new exposure and the previous readout 26.2 µs if exposureAlignment = Synchronous With Overlap between the new exposure and the previous readout		
Trigger to Exposure Minimum delay (Reset Exposure Alignment)	3 µs		
Trigger to Exposure Start jitter (best case with Synchronous Exposure Alignment)	Up to 1 line time		
Trigger to Exposure Start jitter (Reset Exposure Alignment) †	0 µs		
Exposure Time Minimum (see "exposureTimeActual" in Sensor Control)	87 µs (increment steps of 1µs)		
Min. Time from End of Exposure to Start of Next Exposure (second frame)	49 µs – Normal Readout 47 µs – Fast Readout		
Horizontal Line Time:	11.33 µs – Normal Readout 9.33 µs – Fast Readout		
Readout Time	23242 µs – Normal Readout for 2592 x 2048 Add 76µs when overlapping Exposure and Readout 19142 µs – Fast Readout for 2592 x 2048 Add 64µs when overlapping Exposure and Readout <i>Specifically: (Horizontal line time at current resolution * number of lines) + (3 * (line time of the 2590 model))</i>		
Auto-Brightness	Yes , with Auto-Exposure and AGC (FPGA Gain)		
Black offset control	Yes (in DN)		

Gain Control	In-sensor Analog Gain (1.0x to 8x) in 11 gain steps (1.0, 1.14, 1.33, 1.6, 2.0, 2.29, 2.67, 3.2, 4.0, 5.33, 8.0) In-sensor Digital Gain (1x to 32x) in 0.01x steps In-FPGA Digital Gain (1x to 4x) in 0.007x steps	
Binning Support	Yes In-FPGA (summing and average, 2x2, 4x4) Yes In- Sensor (averaging 2x2)	No
Color Correction Support	No	Yes
Decimation Support	No	
Defective Pixel Replacement	Yes, up to 512 positions	
Image Correction	No	
Image Flip Support	Yes, In-Sensor, Vertical Only	
Multi-ROI Support	Yes, in Sensor, up to 16 ROI (mutually exclusive with binning)	
On-Board Image Memory	90MB	
Output Dynamic Range (dB)	62.1 dB (in 10-Bit Pixel Format)	
SNR (dB)	39.8 dB (in 10-Bit Pixel Format)	

*TurboDrive internal limitation of 250MB/sec

† Note: The actual internal minimum exposure may be different than what is programmed. Use the feature "exposureTimeActual" from the [Sensor Control](#) category to read back the actual sensor exposure. The exposure start sensor event is delayed 4 µs from the actual start.

Firmware Files for Models 1280, 1930, 2590

The latest firmware files for all Nano models are available on the Teledyne DALSA support web site:
<http://www.teledynedalsa.com/imaging/support/downloads/firmware/>

The firmware files for these models are listed below. The xx denotes the build number.

M1280, M1930, M2590

- Standard
"Genie_Nano_OnSemi_Python_0.3M-0.5M-1.3M-2M-5M_Mono_STD_Firmware_5CA18.xx.cbf"

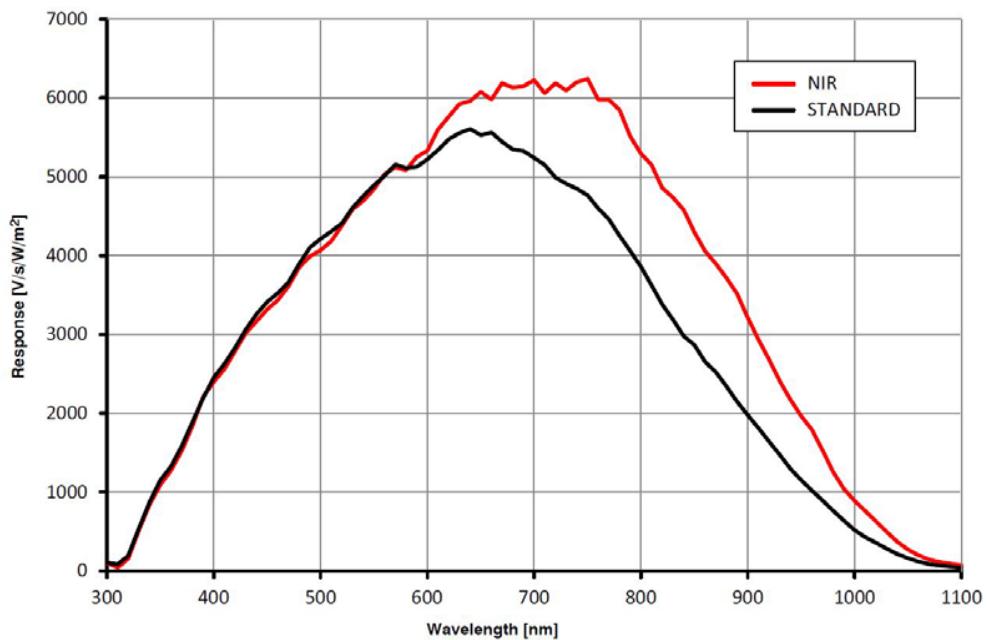
C1280, C1930, C2590

- Bayer Output
"Genie_Nano_OnSemi_Python_0.3M-0.5M-1.3M-2M-5M_Bayer_STD_Firmware_6CA18.xx.cbf"
- RGB Output
"Genie_Nano_OnSemi_Python_0.3M-0.5M-1.3M-2M-5M_RGB_Output_Firmware_6CA18.xx.cbf"

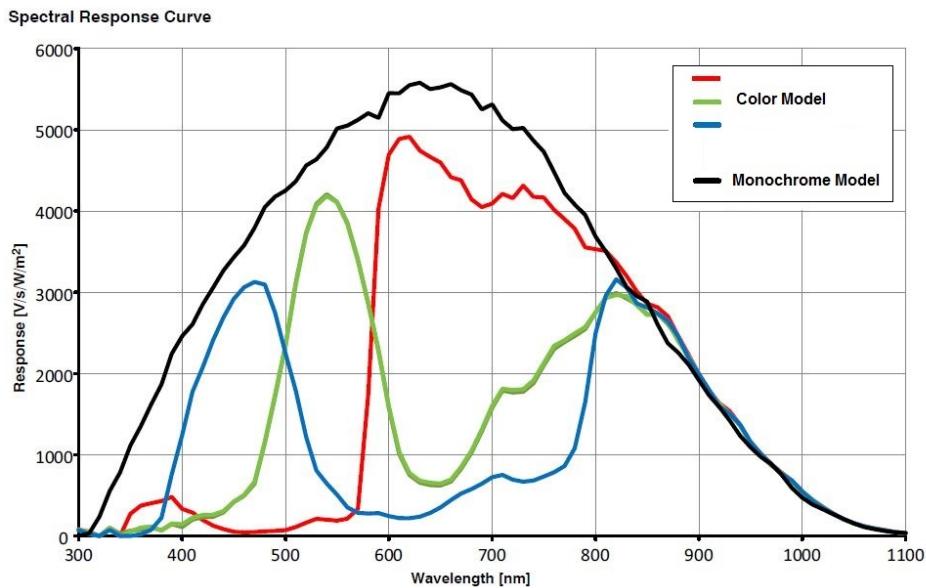
Spectral Response (Python 4.8 μm series)

Model specific specifications and response graphics for the On-Semi Python (VGA to 5M) series are provided here. The response curves describe the sensor, excluding lens and light source characteristics.

On-Semi Python Series (with 4.8 μm pixels) — Monochrome and NIR



On-Semi Python Series (with 4.8 μm pixels) — Monochrome and Color



ANEXO B

Especificações do motor *Dynamixel PH42-020-S300-R*

Item	Especificação
Microcontrolador	ARM CORTEX-M4 (168 [MHz], 32Bit)
Motor	Coreless (Maxon)
Taxa de transmissão	9,600 [bps] ~10.5 [Mbps]
Modos de operação	Torque Control Mode
	Velocity Control Mode
	Position Control Mode
	Extended Position Control Mode
	PWM Control Mode (Voltage Control Mode)
Massa	340 [g]
Dimensões (L x A x P)	42 x 84 x 42 [mm]
Resolução	607,500 [pulsos/rev]
Relação de transmissão	303.75:1
Folga	<6 [arcmin], 0.1 [°]
Carga radial	280 [N] (10 [mm] afastado da face do eixo)
Carga axial	100 [N]
Velocidade sem carga	32.7 [rev/min]
Corrente sem carga	0.57 [A]
Velocidade contínua	29.2 [rev/min]
Torque contínuo	5.1 [Nm]
Corrente contínua	1.5 [A]
Potência de saída	20 [W]
Temperatura de operação	5 ~55 [°C]
Tensão de operação	24.0 [V]
Sinal de comando	Pacotes digitais
Protocolo de comunicação	RS485 Comunicação serial asíncrona (8bit, 1 bit de parada, sem paridade)
Conexão física	RS485 Barramento multiponto
ID	253 ID (0 ~252)
Corrente de espera	80 [mA]

Tabela 24: Especificações do motor *Dynamixel HP42-020-S300-R*

Enter Search Terms



PH42-020-S300-R

- 1. Specifications >
 - 2. Control Table >
 - 3. How to Assemble >
 - 4. Maintenance
 - 5. Reference >



 Edit on GitHub



ROBOTIS



Robot Source



GitHub



TOP

PH42-020-S300-R

1. Specifications

Item	Specifications
MCU	ARM CORTEX-M4 (168 [MHz], 32Bit)
Motor	Coreless (Maxon)
Baud Rate	9,600 [bps] ~ 10.5 [Mbps]
Operating Modes	Torque Control Mode
	Velocity Control Mode
	Position Control Mode
	Extended Position Control Mode
	PWM Control Mode(Voltage Control Mode)
Weight	340 [g]
Dimensions (W x H x D)	42 x 84 x 42 [mm]
Resolution	607,500 [pulse/rev]
Gear Ratio	303.75:1
Backlash	< 6 [arcmin], 0.1 [°]
Radial Load	280 [N] (10 [mm] away from the horn)
Axial Load	100 [N]
No Load Speed	32.7 [rev/min]
No Load Current	0.57 [A]
1 Continuous Speed	29.2 [rev/min]
1 Continuous Torque	5.1 [Nm]
1 Continuous Current	1.5 [A]
Output	20 [W]
Operating Temperature	-5 ~ 55 [°C]
Input Voltage	24.0 [V]
Command Signal	Digital Packet
Protocol Type	RS485 Asynchronous Serial Communication (8bit, 1stop, No Parity)
Physical Connection	RS485 Multidrop Bus

Enter Search Terms	
PH42-020-S300-R	
1. Specifications	
2. Control Table	
3. How to Assemble	
4. Maintenance	
5. Reference	

Item	Specifications
ID	253 ID (0 ~ 252)
Standby Current	30 [mA]

 These specifications are calculated based on the specifications of the core motor. Please consult ROBOTIS for the long term use or special use, or else refer to the Performance Graph for general use.



TOP



DANGER

(May cause serious injury or death)

- Never place items containing water, flammables, and solvents near product.
- Never place fingers, arms, toes, and other body parts near product during operation.
- Cut power off if product emits strange odors or smoke.
- Keep product out of reach of children.
- Check the power's polarity before wiring.



CAUTION

(May cause injury or damage to product)

- Do not operate the product at a temperature exceeding -5 ~ 55 [°C] range.
- Do not insert sharp blades nor pins during product operation.

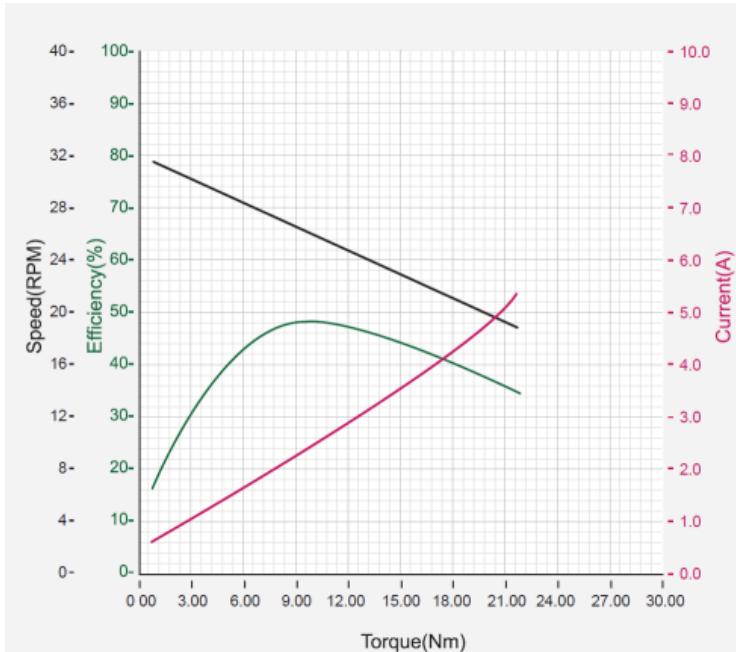


ATTENTION

(May cause injury or damage to product)

- Do not disassemble or modify product.
- Do not drop or apply strong shock to product.

1. 1. Performance Graph



[Show Enlarged Graph](#)

Enter Search Terms	
PH42-020-S300-R	
1. Specifications	
2. Control Table	
3. How to Assemble	
4. Maintenance	
5. Reference	



TOP

NOTE : The Max Torque and the Stall Torque of Performance Graph are different in measurement methods. Stall torque is a measured value of the momentary torque that it can reach. This is generally how RC servos are measured. The Performance graph is also called as N-T curves, which is measured with the gradually increasing load. The actual motor operation environment is closer to the performance graph, not stall torque method. For this reason, the performance graph is broadly used in the industrial field. Generally, Max Torque of the Performance Graph is less than the Stall Torque.

CAUTION : When supplying power

- It is recommended using ROBOTIS controller or SMPS2DYNAMIXEL.
- Do not connect or disconnect DYNAMIXEL when power is being supplied.
- In case of DYNAMIXEL PRO and DYNAMIXEL-P series, please supply power through 24V power port.

2. Control Table

The Control Table is a structure of data implemented in the device. Users can read a specific Data to get status of the device with Read Instruction Packets, and modify Data as well to control the device with WRITE Instruction Packets.

WARNING : DYNAMIXEL-P series use different Control Table from DYNAMIXEL PRO series. Please pay attention when replacing DYNAMIXEL PRO with DYNAMIXEL-P series.

2. 1. Control Table, Data, Address

The Control Table is a structure that consists of multiple Data fields to store status or to control the device. Users can check current status of the device by reading a specific Data from the Control Table with Read Instruction Packets. WRITE Instruction Packets enable users to control the device by changing specific Data in the Control Table. The Address is a unique value when accessing a specific Data in the Control Table with Instruction Packets. In order to read or write data, users must designate a specific Address in the Instruction Packet. Please refer to [Protocol 2.0](#) for more details about Instruction Packets.

NOTE : Two's complement is applied for the negative value. For more information, please refer to [Two's complement](#) from Wikipedia.

2. 1. 1. Area (EEPROM, RAM)

The Control Table is divided into 2 Areas. Data in the RAM Area is reset to initial values when the power is reset(Volatile). On the other hand, data in the EEPROM Area is maintained even when the device is powered off(Non-Volatile).

Data in the EEPROM Area can only be written to if Torque Enable(512) is cleared to '0'(Off).

2. 1. 2. Size

The Size of data varies from 1 ~ 4 bytes depend on their usage. Please check the size of data when updating the data with an Instruction Packet. For data larger than 2 bytes will be saved according to [Little Endian](#).

2. 1. 3. Access

The Control Table has two different access properties. 'RW' property stands for read and write access permission while 'R' stands for read only access permission. Data with the read only property cannot be changed by the WRITE Instruction. Read only property('R') is generally used for measuring and monitoring purpose, and read write property('RW') is used for controlling device.

ANEXO C

Especificações do motor Dynamixel PH54-100-S500-R

Item	Especificação
Microcontrolador	ARM CORTEX-M4 (168 [MHz], 32Bit)
Motor	BLDC (Maxon)
Taxa de transmissão	9,600 [bps] ~10.5 [Mbps]
	Controle de torque
	Controle de velocidade
Modos de operação	Controle de posição
	Controle de posição extendida (mais do que uma revolução)
	Controle por PWM (Controle de tensão)
Massa	740 [g]
Dimensões (L x A x P)	54 x 108 x 54 [mm]
Resolução	1,003,846 [pulsos/rev]
Relação de transmissão	501.923 : 1
Folga	<6 [arcmin], 0.1 [°]
Carga radial	370 [N] (10 [mm] afastado da face do eixo)
Carga axial	130 [N]
Velocidade sem carga	33.3 [rev/min]
Corrente sem carga	1.13 [A]
Velocidade contínua	29.2 [rev/min]
Torque contínuo	25.3 [Nm]
Corrente contínua	5.5 [A]
Potência de saída	100 [W]
Temperatura de operação	5 ~55 [°C]
Tensão de operação	24.0 [V]
Sinal de comando	Pacotes digitais
Protocolo de comunicação	RS485 Comunicação serial asíncrona (8bit, 1 bit de parada, sem paridade)
Conexão física	RS485 Barramento multiponto
ID	253 ID (0 ~252)
Corrente de espera	40 [mA]

Tabela 25: Especificações do motor *Dynamixel PH54-100-S500-R*

ROBOTIS e-Manual DYNAMIXEL PLATFORM STEAM SOFTWARE PARTS FAQ

Enter Search Terms 

[Edit on GitHub](#)  ROBOTIS  Robot Source  GitHub  G 

PH54-100-S500-R

1. Specifications >

2. Control Table >

3. How to Assemble >

4. Maintenance

5. Reference >



PH54-100-S500-R

1. Specifications

Item	Specifications
MCU	ARM CORTEX-M4 (168 [MHz], 32Bit)
Motor	BLDC (Maxon)
Baud Rate	9,600 [bps] ~ 10.5 [Mbps]
Operating Modes	Torque Control Mode Velocity Control Mode Position Control Mode Extended Position Control Mode PWM Control Mode(Voltage Control Mode)
Weight	740 [g]
Dimensions (W x H x D)	54 x 108 x 54 [mm]
Resolution	1,003,846 [pulse/rev]
Gear Ratio	501.923 : 1
Backlash	< 6 [arcmin], 0.1 [°]
Radial Load	370 [N] (10 [mm] away from the horn)
Axial Load	130 [N]
No Load Speed	33.3 [rev/min]
No Load Current	1.13 [A]
Continuous Speed	29.2 [rev/min]
Continuous Torque	25.3 [Nm]
Continuous Current	5.5 [A]
Output	100 [W]
Operating Temperature	-5 ~ 55 [°C]
Input Voltage	24.0 [V]

PH54-100-S500-R

- 1. Specifications
- 2. Control Table
- 3. How to Assemble
- 4. Maintenance
- 5. Reference

Item	Specifications
Command Signal	Digital Packet
Protocol Type	RS485 Asynchronous Serial Communication (8bit, 1stop, No Parity)
Physical Connection	RS485 Multidrop Bus
ID	253 ID (0 ~ 252)
Standby Current	40 [mA]



TOP

1 These specifications are calculated based on the specifications of the core motor. Please consult ROBOTIS for the long term use or special use, or else refer to the Performance Graph for general use.

**DANGER**

(May cause serious injury or death)

- Never place items containing water, flammables, and solvents near product.
- Never place fingers, arms, toes, and other body parts near product during operation.
- Cut power off if product emits strange odors or smoke.
- Keep product out of reach of children.
- Check the power's polarity before wiring.

**CAUTION**

(May cause injury or damage to product)

- Do not operate the product at a temperature exceeding -5 ~ 55 [°C] range.
- Do not insert sharp blades nor pins during product operation.

**ATTENTION**

(May cause injury or damage to product)

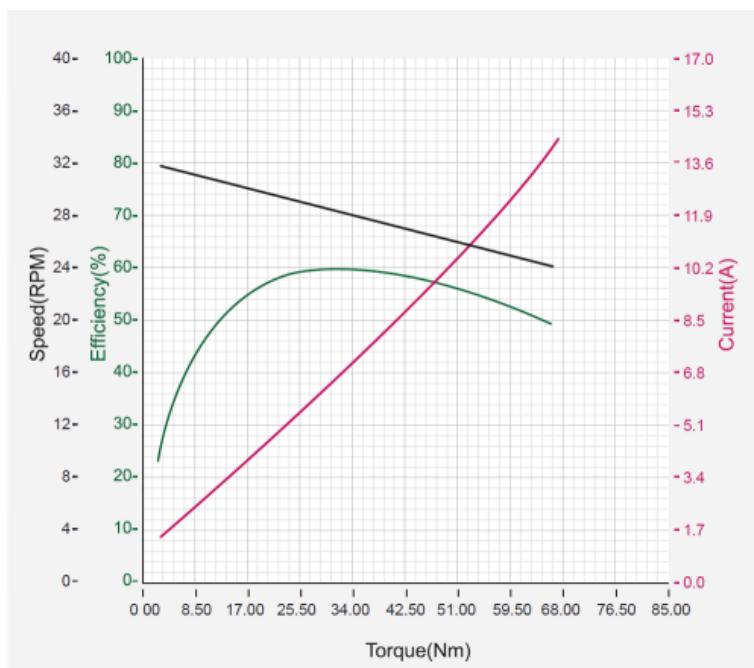
- Do not disassemble or modify product.
- Do not drop or apply strong shock to product.

1. 1. Performance Graph

🔍

PH54-100-S500-R

- 1. Specifications ➤
- 2. Control Table ➤
- 3. How to Assemble ➤
- 4. Maintenance ➤
- 5. Reference ➤



[Show Enlarged Graph](#)

NOTE : The Max Torque and the Stall Torque of Performance Graph are different in measurement methods. Stall torque is a measured value of the momentary torque that it can reach. This is generally how RC servos are measured. The Performance graph is also called as N-T curves, which is measured with the gradually increasing load. The actual motor operation environment is closer to the performance graph, not stall torque method. For this reason, the performance graph is broadly used in the industrial field. Generally, Max Torque of the Performance Graph is less than the Stall Torque.

CAUTION : When supplying power

- It is recommended using ROBOTIS controller or SMPS2DYNAMIXEL.
- Do not connect or disconnect DYNAMIXEL when power is being supplied.
- In case of DYNAMIXEL PRO and DYNAMIXEL-P series, please supply power through 24V power port.

2. Control Table

The Control Table is a structure of data implemented in the device. Users can read a specific Data to get status of the device with Read Instruction Packets, and modify Data as well to control the device with WRITE Instruction Packets.

WARNING : DYNAMIXEL-P series use different Control Table from DYNAMIXEL PRO series. Please pay attention when replacing DYNAMIXEL PRO with DYNAMIXEL-P series.

2. 1. Control Table, Data, Address

The Control Table is a structure that consists of multiple Data fields to store status or to control the device. Users can check current status of the device by reading a specific Data from the Control Table with Read Instruction Packets. WRITE Instruction Packets enable users to control the device by changing specific Data in the Control Table. The Address is a unique value when accessing a specific Data in the Control Table with Instruction Packets. In order to read or write data, users must

ANEXO D

Especificações do motor Dynamixel PH54-200-S500-R

Item	Especificação
MCU	ARM CORTEX-M4 (168 [MHz], 32Bit)
Motor	BLDC (Maxon)
Taxa de transmissão	9,600 [bps] ~10.5 [Mbps]
Modos de operação	Modo de controle de torque Modo de controle de velocidade Modo de controle de posição Modo de controle de posição estendida Modo de controle PWM (modo de controle de tensão)
Peso	855 [g]
Dimensões (c x l x h)	54 x 126 x 54 [mm]
Resolução	1,003,846 [pulse/rev]
Relação de engrenagem	501.923 : 1
Folga	<6 [arcmin], 0.1 [°]
Carga radial	370 [N] (10 [mm] afastado da face do eixo)
Carga axial	130 [N]
Velocidade (sem carga)	33.1 [rev/min]
Corrente (sem carga)	1.65 [A]
Velocidade contínua	29.0 [rev/min]
Torque contínuo	44.7 [Nm]
Corrente contínua	9.3 [A]
Saída	200 [W]
Temperatura de operação	5 ~55 [°C]
Tensão operacional	24.0 [V]
Sinal de comando	Pacote digital
Tipo de protocolo	RS485 Comunicação Serial Assíncrona (8bit, 1stop, Sem paridade)
Conexão física	RS485 Barramento multiponto
ID	253 ID (0 ~252)
Corrente de repouso	40 [mA]

Tabela 26: Especificações do motor *Dynamixel PH54-200-S500-R*

Enter Search Terms



PH54-200-S500-R

- 1. Specifications
- 1. 1. Performance Graph
- 2. Control Table
- 3. How to Assemble
- 4. Maintenance
- 5. Reference

[Edit on GitHub](#)

ROBOTIS



Robot Source



GitHub



TOP

PH54-200-S500-R

1. Specifications

Item	Specifications
MCU	ARM CORTEX-M4 (168 [MHz], 32Bit)
Motor	BLDC (Maxon)
Baud Rate	9,600 [bps] ~ 10.5 [Mbps]
Operating Modes	Torque Control Mode Velocity Control Mode Position Control Mode Extended Position Control Mode PWM Control Mode(Voltage Control Mode)
Weight	855 [g]
Dimensions (W x H x D)	54 x 126 x 54 [mm]
Resolution	1,003,846 [pulse/rev]
Gear Ratio	501.923 : 1
Backlash	< 6 [arcmin], 0.1 [°]
Radial Load	370 [N] (10 [mm] away from the horn)
Axial Load	130 [N]
No Load Speed	33.1 [rev/min]
No Load Current	1.65 [A]
Continuous Speed	29.0 [rev/min]
Continuous Torque	44.7 [Nm]
Continuous Current	9.3 [A]
Output	200 [W]
Operating Temperature	-5 ~ 55 [°C]
Input Voltage	24.0 [V]
Command Signal	Digital Packet
Protocol Type	RS485 Asynchronous Serial Communication (8bit, 1stop, No Parity)
Physical Connection	RS485 Multidrop Bus

Enter Search Terms



PH54-200-S500-R

- 1. Specifications
- 1. 1. Performance Graph
- 2. Control Table
- 3. How to Assemble
- 4. Maintenance
- 5. Reference



TOP

[1] These specifications are calculated based on the specifications of the core motor. Please consult ROBOTIS for the long term use or special use, or else refer to the Performance Graph for general use.

**DANGER**

(May cause serious injury or death)

- Never place items containing water, flammables, and solvents near product.
- Never place fingers, arms, toes, and other body parts near product during operation.
- Cut power off if product emits strange odors or smoke.
- Keep product out of reach of children.
- Check the power's polarity before wiring.

**CAUTION**

(May cause injury or damage to product)

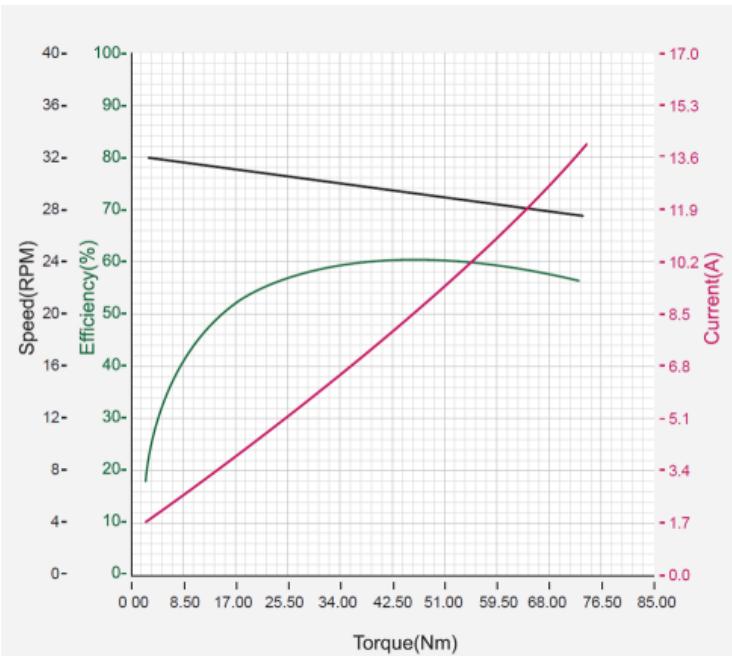
- Do not operate the product at a temperature exceeding -5 ~ 55 [°C] range.
- Do not insert sharp blades nor pins during product operation.

**ATTENTION**

(May cause injury or damage to product)

- Do not disassemble or modify product.
- Do not drop or apply strong shock to product.

1. 1. Performance Graph

[Show Enlarged Graph](#)

🔍

PH54-200-S500-R

- 1. Specifications
- 1. 1. Performance Graph
- 2. Control Table
- 3. How to Assemble
- 4. Maintenance
- 5. Reference

NOTE : The Max Torque and the Stall Torque of Performance Graph are different in measurement methods. Stall torque is a measured value of the momentary torque that it can reach. This is generally how RC servos are measured. The Performance graph is also called as N-T curves, which is measured with the gradually increasing load. The actual motor operation environment is closer to the performance graph, not stall torque method. For this reason, the performance graph is broadly used in the industrial field. Generally, Max Torque of the Performance Graph is less than the Stall Torque.



TOP

CAUTION : When supplying power

- It is recommended using ROBOTIS controller or SMPS2DYNAMIXEL.
- Do not connect or disconnect DYNAMIXEL when power is being supplied.
- In case of DYNAMIXEL PRO and DYNAMIXEL-P series, please supply power through 24V power port.

2. Control Table

The Control Table is a structure of data implemented in the device. Users can read a specific Data to get status of the device with Read Instruction Packets, and modify Data as well to control the device with WRITE Instruction Packets.

WARNING : DYNAMIXEL-P series use different Control Table from DYNAMIXEL PRO series. Please pay attention when replacing DYNAMIXEL PRO with DYNAMIXEL-P series.

2. 1. Control Table, Data, Address

The Control Table is a structure that consists of multiple Data fields to store status or to control the device. Users can check current status of the device by reading a specific Data from the Control Table with Read Instruction Packets. WRITE Instruction Packets enable users to control the device by changing specific Data in the Control Table. The Address is a unique value when accessing a specific Data in the Control Table with Instruction Packets. In order to read or write data, users must designate a specific Address in the Instruction Packet. Please refer to [Protocol 2.0](#) for more details about Instruction Packets.

NOTE : Two's complement is applied for the negative value. For more information, please refer to [Two's complement](#) from Wikipedia.

2. 1. 1. Area (EEPROM, RAM)

The Control Table is divided into 2 Areas. Data in the RAM Area is reset to initial values when the power is reset(Volatile). On the other hand, data in the EEPROM Area is maintained even when the device is powered off(Non-Volatile).

Data in the EEPROM Area can only be written to if Torque Enable(512) is cleared to '0'(Off).

2. 1. 2. Size

The Size of data varies from 1 ~ 4 bytes depend on their usage. Please check the size of data when updating the data with an Instruction Packet. For data larger than 2 bytes will be saved according to [Little Endian](#).

2. 1. 3. Access

The Control Table has two different access properties. 'RW' property stands for read and write access permission while 'R' stands for read only access permission. Data with the read only property cannot be changed by the WRITE Instruction. Read only property('R') is generally used for measuring and monitoring purpose, and read write property('RW') is used for controlling device.