

MANIPULADOR ROBÓTICO TIMON-HM

Relatório Parcial do Projeto Manipuladores Inteligentes

Autores:

Jéssica Lima Motta

Leonardo Mendes de Souza Lima

Miguel Felipe Nery Vieira

Vinicius José Gomes de Araújo Felismino

Facilitadores:

Lucas Cruz da Silva

Tiago Pereira de Souza

Marco Antonio dos Reis

Salvador

Bahia, Brasil

Abril de 2020

Título: Manipulador Robótico Timon-HM	
PROD. TEC. BIR - 001 / 2020	Versão 01
Classificação: () Confidencial (X) Restrito () Uso Interno () Público	

Informações Confidenciais - Informações estratégicas para o BIR 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)

Jéssica Lima Motta
Leonardo Mendes de Souza Lima
Miguel Felipe Nery Vieira
000 Vinicius José Gomes de Araújo Felismino

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

Manipulador Robótico Timon-HM
Salvador
Bahia, Brasil
Abril de 2020

Keywords:
1. Manipulator. 2. Simulation. 3. Computer vision.

000

SUMÁRIO EXECUTIVO

O projeto de Manipuladores - Desafio 2, também conhecido como **Timon-HM Manipulator** configura-se: sob o o Programa de Formação de Novos Talentos do Serviço Nacional de Aprendizagem Industrial, Departamento Regional da Bahia - Senai/DR/BA, sendo este o principal fomentador do programa.

O projeto foi considerado como início técnico no dia 27 de Fevereiro de 2020.

O prazo de execução planejado foi de 50 dias.

RESUMO

Timon-HM é um manipulador projetado, simulado e construído com o intuito de atender às demandas relacionadas ao reconhecimento de marcadores visuais e acionamento de interruptores, chaves ou botões. Posteriormente, este mesmo manipulador robótico será integrado ao robô *Warthog*, desenvolvido pela *Clearpath Robotics*, com o propósito de realizar a atividade de investigação em ambiente externo, desta vez participando de uma simulação de busca e desarme de bombas. O pacote de simulação do manipulador foi construído através do software *Gazebo* aliado ao *MoveIt* e à ferramenta de visualização *Rviz*, possibilitando assim que as atividades realizadas no mundo real tenham sido previamente testadas em ambiente simulado onde é possível analisar os movimentos e limitações do manipulador.

ABSTRACT

Timon-HM is a manipulator designed, simulated and built in order to meet demands related to the recognition of visual markers and activation of interrupters, switches or buttons. Posteriorly, this manipulator robot arm will be attached in *Warthog*, robot designed by *Clearpath Robotics*, with the purpose of carrying out the research activity in an external environment, this time, integrating a simulation for searching and defusing bombs. The manipulator simulation package was built using software *Gazebo*, allied to *Moveit* and *Rviz* visual tool, allowing that activities practiced in the real world have been previously tested in a computer environment where it is possible to analyze manipulator's movements and limitations.

LISTA DE FIGURAS

Figura 1:	Elos e junta de um manipulador robótico.	14
Figura 2:	Arquitetura geral do sistema.	17
Figura 3:	Configuração D-H do manipulador Timon-HM.	19
Figura 4:	<i>Workspace</i> do manipulador no plano X-Y.	20
Figura 5:	<i>Workspace</i> do manipulador no plano X-Z.	21
Figura 6:	<i>Workspace</i> do manipulador no plano Y-Z.	21
Figura 7:	Estrutura analítica do protótipo.	22
Figura 8:	Fluxograma do sistema de escanamento.	23
Figura 9:	Fluxograma do sistema de planejamento e execução de trajetória. . .	25
Figura 10:	Modelos simulados do manipulador e do painel elétrico.	27
Figura 11:	Imagem capturada pela câmera RGB.	27
Figura 12:	Pose 1 testada para botão.	29
Figura 13:	Pose 4 testada para botão.	29
Figura 14:	Timon-HM pressionando botão localizado em pose 2.	31
Figura 15:	Timon-HM pressionando botão localizado em pose 6.	31
Figura 16:	Árvore de falhas do sistema.	35
Figura 17:	Coordenadas.	51
Figura 18:	Link 0.	51
Figura 19:	Link 1.	52
Figura 20:	Link 2.	53
Figura 21:	Link 3.	53
Figura 22:	Link 4.	54
Figura 23:	Link 5.	55

LISTA DE TABELAS

Tabela 1:	Especificações técnicas do manipulador Timon-HM.	18
Tabela 2:	Parâmetros D-H para o manipulador Timon-HM.	19
Tabela 3:	Poses testadas para o botão.	30
Tabela 4:	Testes realizados para painel elétrico com orientação vertical.	30
Tabela 5:	Testes realizados para painel elétrico com orientação horizontal.	30
Tabela 6:	<i>FMECA</i> do sub-sistema de potência	33
Tabela 7:	<i>FMECA</i> do sub-sistema de aquisição	33
Tabela 8:	<i>FMECA</i> do sub-sistema estrutural	34
Tabela 9:	<i>FMECA</i> do sub-sistema de atuação	34
Tabela 10:	<i>FMECA</i> do sub-sistema de processamento	34
Tabela 11:	Lições aprendidas.	37
Tabela 12:	Especificações do motor <i>Dynamixel</i> MX-106	75
Tabela 13:	Especificações do motor <i>Dynamixel</i> H54-S500-R	79

LISTA DE SÍMBOLOS E ABREVIATURAS

DoF Degrees of Freedom

SOTA Study Of The Art

ROS Robot Operating System

AGV Automated Guided Vehicle

URDF Unified Robot Description Format

CAD Computer Aided Design

FMECA Failure Modes, Effects and Critically Analysis

OpenCV Open Source Computer Vision

OMPL Open Motion Planning Library

SUMÁRIO

1	INTRODUÇÃO	11
1.1	Objetivos	11
1.2	Justificativa	11
1.3	Organização do relatório	12
2	CONCEITO DO SISTEMA	13
2.1	Parâmetros básicos	13
2.1.1	Requisitos do cliente	13
2.1.2	Requisitos técnicos	13
2.1.3	Estudo do estado da arte	13
3	DESENVOLVIMENTO DO SISTEMA	17
3.1	Descrição do sistema	17
3.1.1	Arquitetura geral	17
3.1.2	Especificação técnica	18
3.1.3	Ambiente de operação	19
3.1.4	Estrutura analítica do protótipo	22
3.2	Especificação funcional	22
3.2.1	Escaneamento	22
3.2.1.1	Descrição	23
3.2.1.2	Premissas necessárias	23
3.2.1.3	Dependências	24
3.2.1.4	Saídas	24
3.2.2	Planejamento e Execução de Trajetória	24
3.2.2.1	Descrição	24
3.2.2.2	Premissas necessárias	25
3.2.2.3	Dependências	25
3.2.2.4	Saídas	26
3.3	Arquitetura de software	26
3.4	Simulação do sistema	26

4	RESULTADOS E ANÁLISES	29
4.1	Resultados alcançados	30
5	CONFIABILIDADE DO SISTEMA	33
5.1	Análise dos modos e efeitos de falhas	33
5.2	Análise da árvore de falhas	34
6	GESTÃO DO CONHECIMENTO	37
6.1	Lições aprendidas	37
6.2	Guia de uso	37
7	CONCLUSÃO	41
	REFERÊNCIAS	43
	APÊNDICE A Árvores de TF desconectadas	45
	APÊNDICE B Árvores de TF conectadas	48
	APÊNDICE C Diagrama de nós do sistema	50
	APÊNDICE D Propriedades de Massa do Timon-HM	51
	APÊNDICE E Algoritmo de busca e acionamento do painel	57
	APÊNDICE F Diagrama elétrico do Timon-HM	63
	APÊNDICE G Diagrama de conexão do Timon-HM	67
	ANEXO A Especificações da câmera Dalsa Genie Nano	71
	ANEXO B Especificações do motor Dynamixel MX-106	75
	ANEXO C Especificações do motor Dynamixel H54-S500-R	79

1 INTRODUÇÃO

Os robôs estão cada vez mais presentes tanto no dia-a-dia das pessoas quanto na indústria. O aumento da produtividade, aliado à qualidade final do produto, maior confiabilidade e repetibilidade, tornam evidentes a importância das pesquisas na área de robótica. Esta, é oriunda de uma fusão disciplinar, em que diversos ramos de conhecimento como cinemática, dinâmica, estática, controle, inteligência artificial, programação, computação gráfica e visão computacional estão sendo estudados para serem implementados nos sistemas robóticos e melhorar a performance dos mesmos (OLIVEIRA et al., 2012).

Este relatório tem por finalidade descrever o processo de construção de um manipulador robótico realizado no Laboratório de Robótica e Sistemas Autônomos do SENAI-CIMATEC-BA para o programa de formação de Novos Talentos. Serão descritas as etapas de concepção, simulação e testes realizados que viabilizarão sua futura 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 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 modelo do manipulador robótico.
- Parametrizar pacote de reconhecimento de marcadores visuais.
- Desenvolver pacote de configuração do *MoveIt*.
- Desenvolver nó para realizar missão.
- Realizar simulação do protótipo do manipulador em software.

1.2 Justificativa

Apesar da crescente demanda, há uma falta de profissionais habilitados em 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 uma 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 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 sete capítulos, sendo este de Introdução com vista à descrição da justificativa/motivação, os objetivos e a organização do relatório.

No capítulo 2, intitulado Conceitos 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, intitulado Desenvolvimento , 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 compõe o sistema, sua arquitetura de software e o que foi desenvolvido para simulação.

Já no capítulo 4, intitulado Resultados e análises, são apresentados os resultados alcançados e a análise dos dados amostrados.

O capítulo 5, intitulado Confiabilidade do sistema, detalha a análise dos modos e efeitos de falhas além de mostrar análise da árvore de falhas.

No capítulo 6, intitulado Gestão do conhecimento, é feito um estudo sobre as lições aprendidas e o guia uso.

Por fim, o capítulo 7 apresenta a Conclusão do relatório.

2 CONCEITO DO SISTEMA

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 precisa ser simulado. 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

1. Desenvolver um manipulador robótico.
2. Realizar a tarefa de detecção de um marcador visual e acionamento do painel elétrico.
3. Realizar a simulação do manipulador no ambiente *ROS* utilizando o software *Gazebo*.
4. Gerar pacote de configuração no *MoveIt* de acordo com o manipulador.

2.1.2 Requisitos técnicos

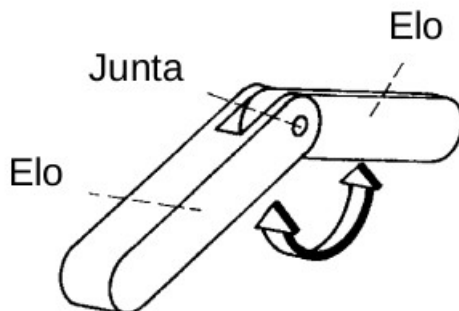
1. A base deve estar a 0,25 m de uma das extremidades da bancada.
2. Deve suportar uma carga máxima de 2 kg.
3. O manipulador deverá acionar um painel elétrico.

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 vem sido realizadas na área de manipuladores robóticos. Em ([HERNANDEZ-MENDEZ et al., 2017](#)) é descrito o desenvolvimento de um manipulador robótico com 3

Figura 1: Elos e junta de um manipulador robótico.



Fonte: (SANTOS, 2004).

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 *endeffector*¹ durante o processo de "escolher e colocar". 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 *endeffector* 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

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

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 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 Timon-HM, abordando os sistemas que o compõem em *software* e em *hardware*. Suas funcionalidades principais são abordadas e a conexão entre as mesmas é exibida.

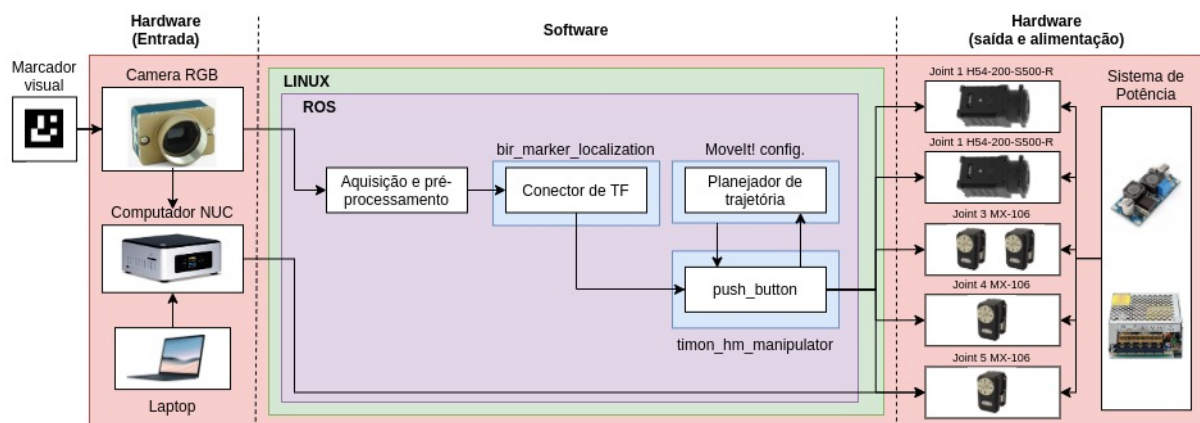
3.1 Descrição do sistema

Timon-HM 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*(salmão) 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 Timon-HM. 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.

Tabela 1: Especificações técnicas do manipulador Timon-HM.

Item	Timon-HM
Graus de liberdade	5
Carga útil	1,94 kg
Alcance	979 mm
Peso	7,38 kg
Tensão de operação	24 V
Resolução	Junta 1, Junta 2: 1.003.846 pulsos/rev
	Junta 3, Junta 4, Junta 5: 4096 pulsos/rev
Motores	Junta 1, Junta 2: H54-200-S500-R (200 W)
	Junta 3(2), Junta 4, Junta 5: MX-106 (65 W)
Faixa de operação	Junta 1: $-90^{\circ} \sim 90^{\circ}$
	Junta 2: $-90^{\circ} \sim 90^{\circ}$
	Junta 3: $-90^{\circ} \sim 135^{\circ}$
	Junta 4: $-180^{\circ} \sim 180^{\circ}$
	Junta 5: $-90^{\circ} \sim 90^{\circ}$
Camera	Teledyne Genie Nano C2590
Tipo do sensor de posição	Pose inicial: Codificador Absoluto
	Controle: Codificador Incremental
Comunicação	RS485
Taxa de transmissão	1 Mbps

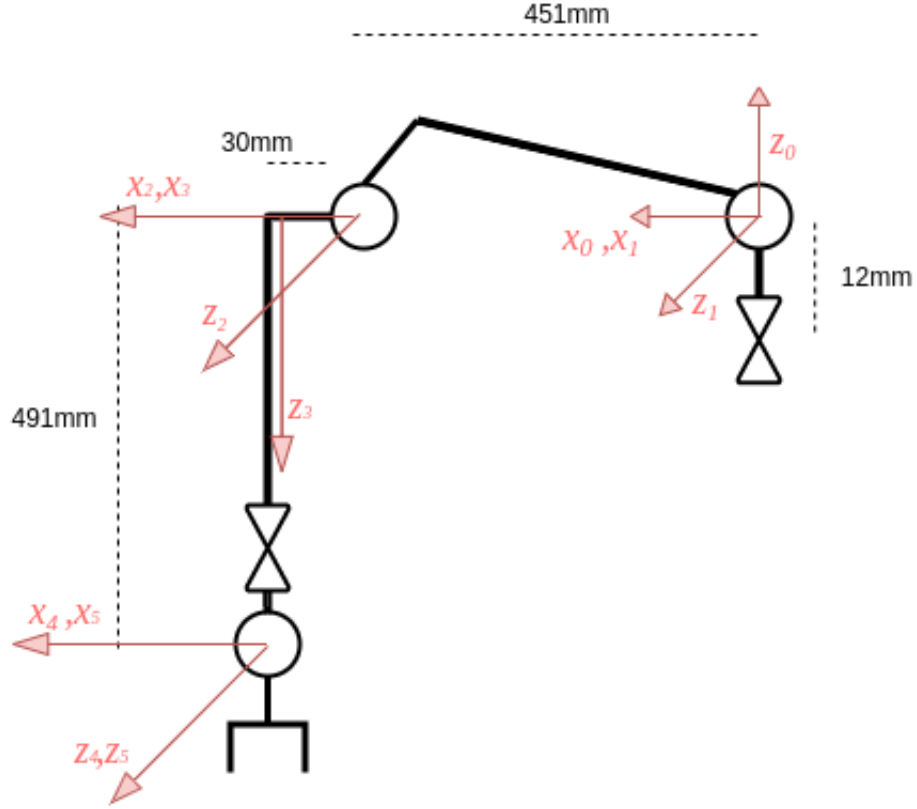
Fonte: Autoria própria.

Para poder relacionar a pose do *endeffector* com a base do manipulador é necessário

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

descrever o seu sistema de coordenadas em relação ao sistema de coordenadas de origem. Para isto, é utilizada a notação de *Denavit-Hartenberger*(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 Timon-HM.



Fonte: Autoria própria.

Tabela 2: Parâmetros D-H para o manipulador Timon-HM.

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.

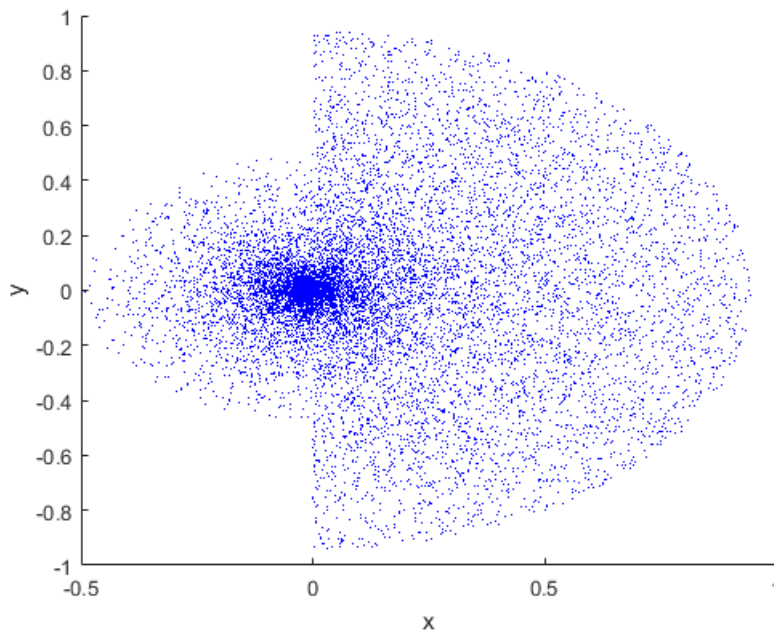
3.1.3 Ambiente de operação

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 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.

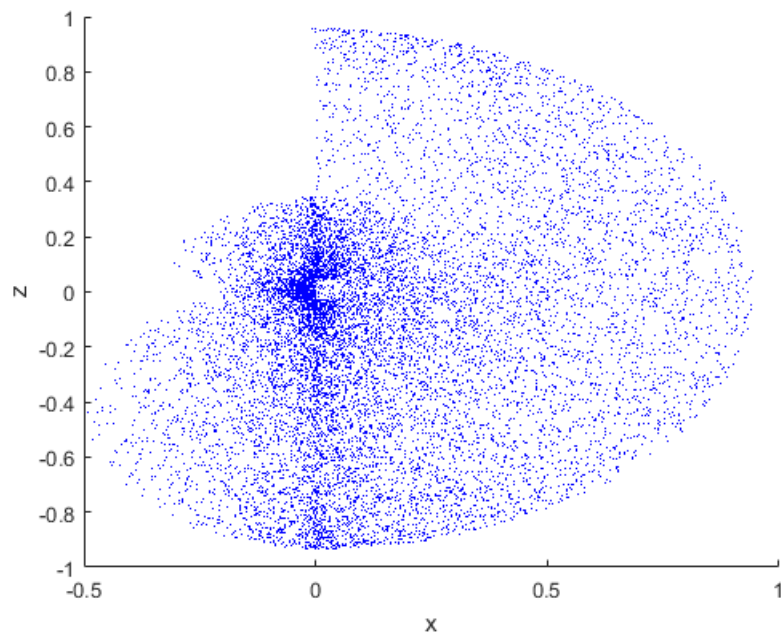
Figura 4: *Workspace* do manipulador no plano X-Y.



Fonte: Autoria própria.

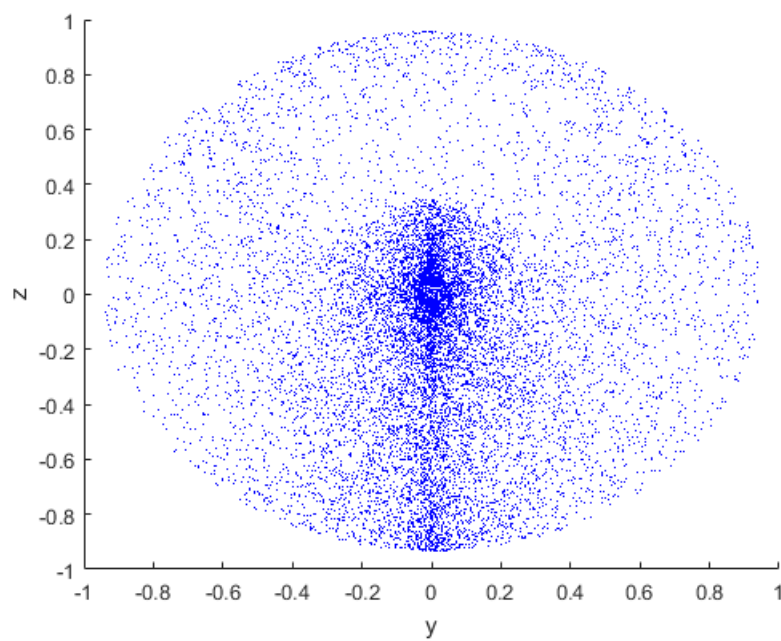
⁴ É a área de alcance do manipulador, região na qual este consegue operar.

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



Fonte: Autoria própria.

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

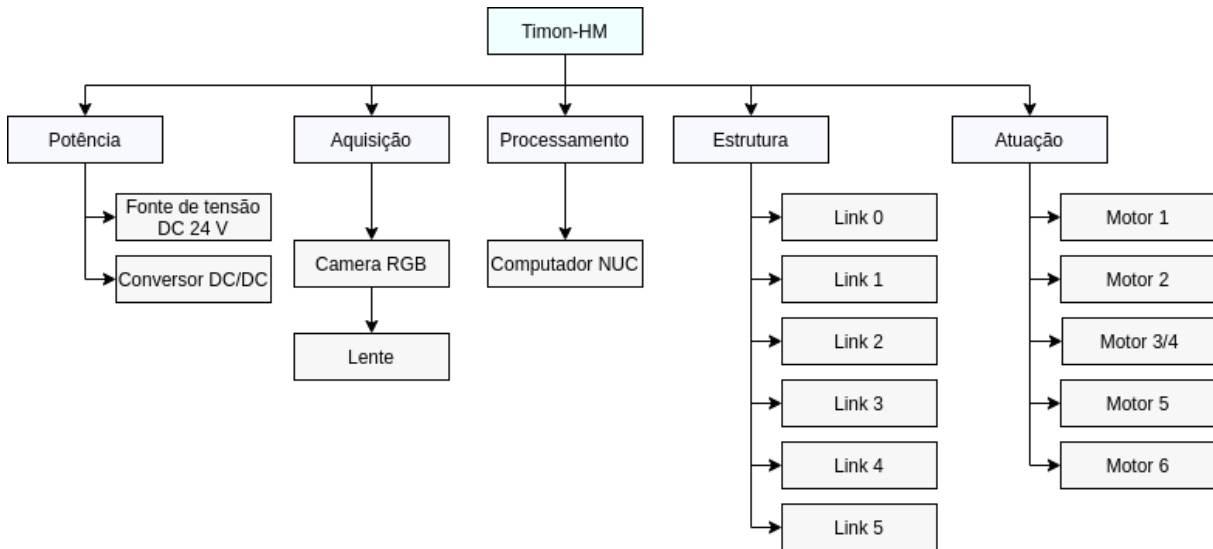


Fonte: Autoria própria.

3.1.4 Estrutura analítica do protótipo

A estrutura analítica do protótipo mostrada na Figura 7 exhibe as relações sistemáticas entre as partes que compõem o manipulador. A estrutura hierárquica possui três níveis: o primeiro, referente ao sistema principal Timon-HM; 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.

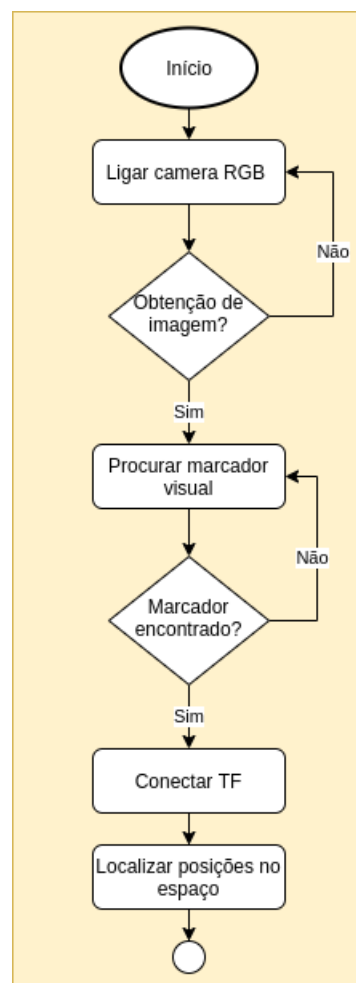
3.2.1 Escaneamento

Uma câmera RGB *Teledyne Genie Nano c2590* equipada com lente *kowa LM8FC* foi acoplada ao manipulador Timon-HM. 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 *TF* 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 *TF* antes e após a conexão realizada.

Figura 8: Fluxograma do sistema de escaneamento.



Fonte: Autoria própria.

3.2.1.2 Premissas necessárias

- Deverá haver um marcador visual associado 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.
- 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 e a inserção do pacote *Bir Marker Localization* no *workspace* do manipulador.

3.2.1.4 Saídas

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

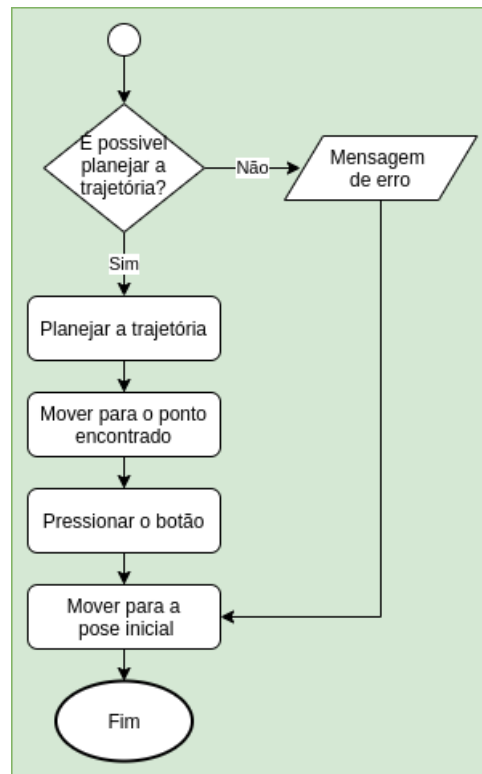
3.2.2 Planejamento e Execução de Trajetória

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 Timon-HM, 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 possuam 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 erro é exibida e o robô retorna para sua posição inicial.

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



Fonte: Autoria própria.

3.2.2.2 Premissas necessárias

- Detecção do marcador visual.
- 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 movimentação são entregues aos motores *Dynamixel* integrados às juntas do manipulador. A trajetória gerada pelo *MoveIt* pôde ser visualizada a partir do tópico `/move_group/display_planned_path`.

3.3 Arquitetura de software

O robô Timon-HM 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 TF realizadas pelo `/marker_localization` que recebe e analisa os dados da câmera (`/timon/camera_image_raw`). Pode-se verificar também que o nó `/push_button` comunica-se com o nó `/tf` a fim de adquirir a pose do painel elétrico e enviá-la ao nó `/move_group` que realiza o planejamento e execução da trajetória.

3.4 Simulação do sistema

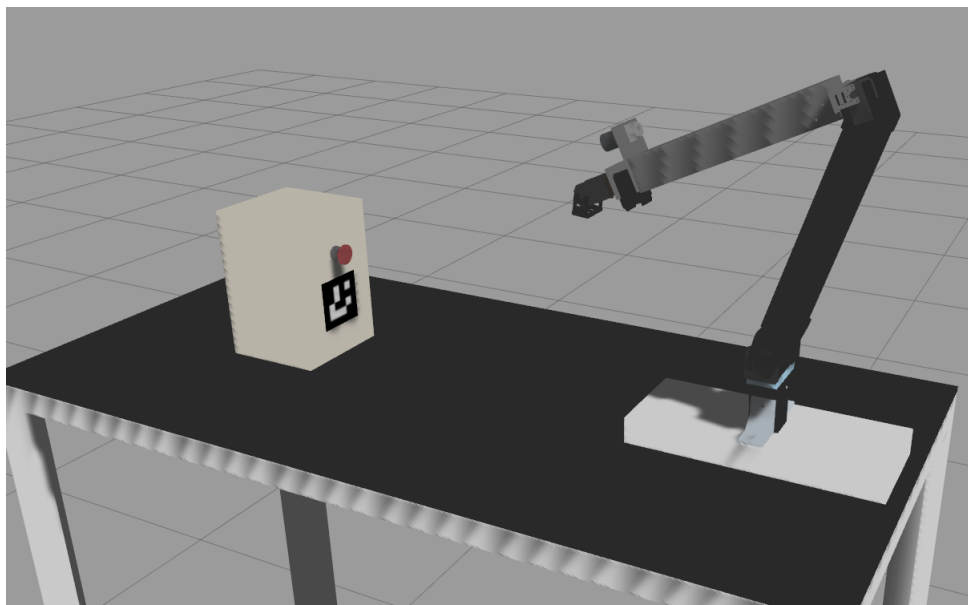
A simulação do robô Timon-HM inserido no seu ambiente de trabalho foi realizada na plataforma *Gazebo*. Para isto, realizou-se o desenvolvimento dos arquivos *timon_arm_hand_base.urdf* e *box.urdf*, modelos *URDF* que descrevem o robô e o painel elétrico a ser acionado. Foram levados em consideração 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 no mundo real.

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

Para simulação da câmera RGB, foi desenvolvido o arquivo *camera.xacro*. O plugin padrão do *Gazebo* foi utilizado, *libgazebo_ros_camera.so* e as especificações da câmera RGB *Teledyne Genie Nano c2590* foram levados em consideração, garantindo maior realismo à

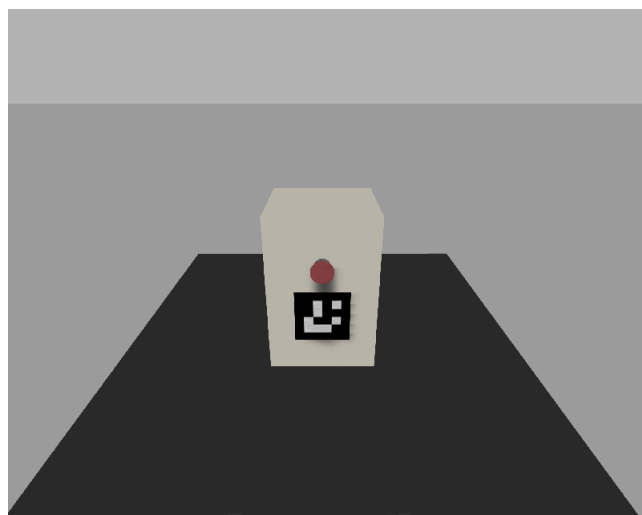
simulação. A figura 10 exibe os modelos simulados do manipulador e do painel elétrico, enquanto a figura 11 exibe imagem capturada pela câmera instalada no manipulador.

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



Fonte: Autoria própria.

Figura 11: Imagem capturada pela câmera RGB.



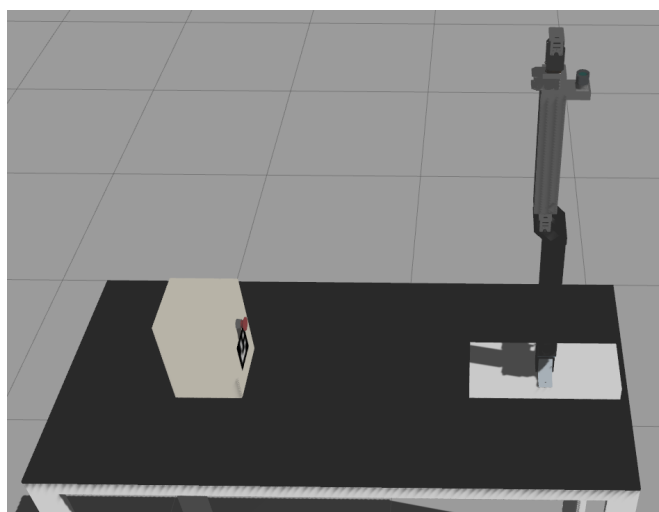
Fonte: Autoria própria.

4 RESULTADOS E ANÁLISES

Os testes para a amostragem dos dados exibidos nesta seção foram realizados em um computador com sistema operacional Ubuntu 18.04, processador Intel Core i7-4790 3.60GHZ, 16GB de memória RAM e placa de vídeo NVIDIA GeForce GT 730.

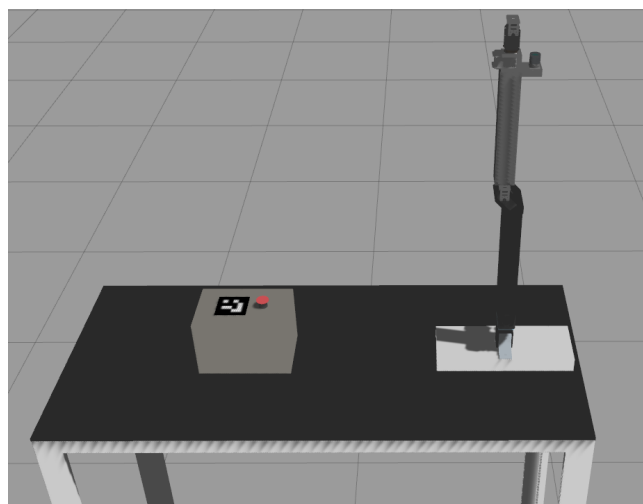
Foram estipuladas 6 diferentes posições para o botão, considerando orientações na vertical e na horizontal para o painel elétrico, conforme Tabela 3. Duas destas posições estão exibidas nas Figuras 12 e 13.

Figura 12: Pose 1 testada para botão.



Fonte: Autoria própria.

Figura 13: Pose 4 testada para botão.



Fonte: Autoria própria.

Tabela 3: Poses testadas para o botão.

Orientação do painel elétrico	Pose	Distância no eixo x (m)	Distância no eixo y (m)
Vertical	1	0.9	0
	2	0.6	0.2
	3	0.95	-0.2
Horizontal	4	0.9	0
	5	0.6	0.2
	6	0.8	-0.2

4.1 Resultados alcançados

Foram realizados 10 ensaios consecutivos para cada posição exibida na Tabela 3. Os dados referentes a estas simulações podem ser visualizados nas Tabelas 4 e 5.

Tabela 4: Testes realizados para painel elétrico com orientação vertical.

Painel elétrico com orientação vertical								
Pose 1			Pose 2			Pose 3		
N	Tempo(s)	Sucesso	N	Tempo(s)	Sucesso	N	Tempo(s)	Sucesso
1	22	sim	1	25	sim	1	26	sim
2	24	sim	2	29	sim	2	24	sim
3	20	sim	3	23	sim	3	19	sim
4	23	sim	4	32	sim	4	19	sim
5	22	sim	5	28	sim	5	33	não
6	24	sim	6	27	sim	6	24	sim
7	21	sim	7	23	sim	7	26	sim
8	22	sim	8	23	sim	8	29	não
9	21	sim	9	24	sim	9	21	sim
10	22	sim	10	21	sim	10	29	sim

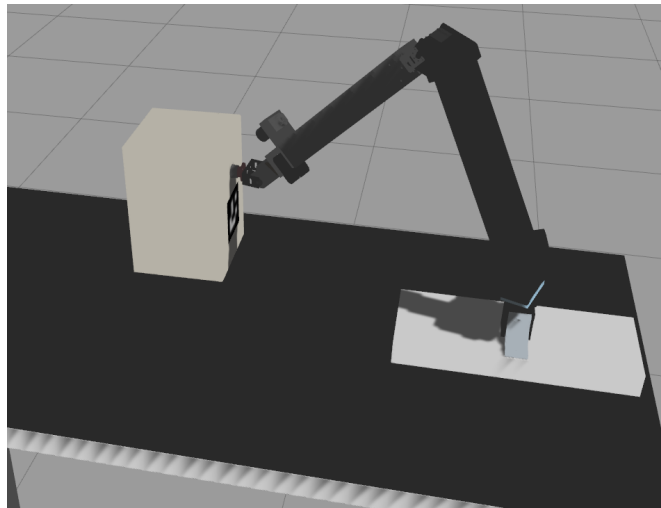
Tabela 5: Testes realizados para painel elétrico com orientação horizontal.

Painel elétrico com orientação horizontal								
Pose 4			Pose 5			Pose 6		
N	Tempo(s)	Sucesso	N	Tempo(s)	Sucesso	N	Tempo(s)	Sucesso
1	24	sim	1	27	sim	1	26	sim
2	21	sim	2	17	sim	2	27	não
3	24	sim	3	25	sim	3	25	não
4	26	sim	4	21	sim	4	18	sim
5	26	sim	5	22	sim	5	21	sim
6	17	sim	6	23	sim	6	25	sim
7	23	sim	7	22	sim	7	20	sim
8	25	sim	8	25	sim	8	21	sim
9	21	sim	9	24	sim	9	23	sim
10	27	sim	10	23	sim	10	23	sim

Conforme os dados exibidos verificou-se que o manipulador obteve sucesso em 56 das 60 tentativas realizadas, resultando em uma taxa total de 90% de aproveitamento com tempo médio de execução da rotina de 24 segundos. Nas 4 tentativas em que houve falha, o *endeffector* chegou à posição desejada, porém, devido a erros de planejamento para a sua orientação, não foi capaz de pressionar o botão.

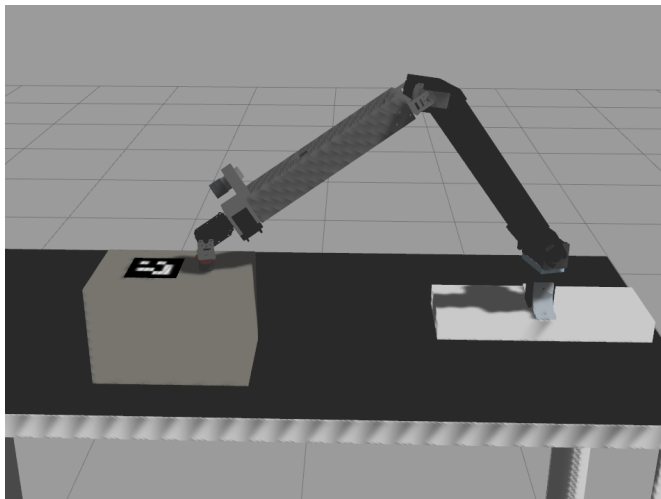
As Figuras 14 e 15 exibem o robô Timon-HM no exato momento em que foi capaz de pressionar o botão para duas das poses desejadas, após localização do mesmo via identificação da *tag ArUco*. Pode se constatar, a partir das mesmas e dos dados exibidos, que os resultados foram satisfatórios no que diz respeito ao reconhecimento da *tag ArUco*, planejamento e execução de trajetória para um determinado ponto.

Figura 14: Timon-HM pressionando botão localizado em pose 2.



Fonte: Autoria própria.

Figura 15: Timon-HM pressionando botão localizado em pose 6.



Fonte: Autoria própria.

5 CONFIABILIDADE DO SISTEMA

No presente capítulo para análise da confiabilidade do sistema, foi realizado levantamento de cada componente que constitui o manipulador robótico e dessa forma possibilitar o estudo detalhado das prováveis falhas. Para isso, foi utilizado o *FMECA* afim de analisar cada sub-sistema, como pode ser visto nas tabelas da seção 5.1.

E na seção 5.2 foi desenvolvida a árvore de falha, que permite através de um processo lógico dedutivo chegar-se às causas-raiz de uma determinada falha.

5.1 Análise dos modos e efeitos de falhas

Nas Tabelas 6, 7, 8, 9 e 10 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 6: *FMECA* do sub-sistema de potência

Análise do Tipo e Efeito de Falha								
Sub-sistema de potência								
Função(ões)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(ões) 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	Verificar continuidade na placa do circuito

Fonte: Autoria própria.

Tabela 7: *FMECA* do sub-sistema de aquisição

Análise do Tipo e Efeito de Falha								
Sub-sistema de aquisição								
Função(ões)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(ões) recomendada(s)
Aquisição de dados visuais	Incapacidade de obter dados visuais	Perda da capacidade de obter dados visuais	7	Obstrução do campo de visão da câmera	2	1	14	Verificar condições do ambiente antes das missões
	Ruptura da estrutura de fixação da câmera			Baixa visibilidade do ambiente	2	5	70	
	Coleta de dados inconsistentes ou insuficientes	Comprometimento dos dados	6	Set-up incorreto	2	7	84	Verificar cabeamento do sistema
				Falha no canal de comunicação	4	7	168	

Fonte: Autoria própria.

Tabela 8: *FMECA* do sub-sistema estrutural

Análise do Tipo e Efeito de Falha								
Sub-sistema estrutural								
Função(ões)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(ões) 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	
		Trinca do conector	6	Set-up incorreto	4	6	144	Verificar as condições de peso aplicadas
				Defeito de fabricação	5	6	180	
Compor estrutura mecânica	Danificação estrutural	Ruptura do perfil	8	Sobrecarga	2	4	64	Cuidado no manuseamento dos perfis
	Folga entre conexões	Empeno do perfil	6	Colisão	4	2	48	Definir área de trabalho

Fonte: Autoria própria.

Tabela 9: *FMECA* do sub-sistema de atuação

Análise do Tipo e Efeito de Falha								
Sub-sistema de atuação								
Função(ões)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(ões) recomendada(s)
Movimentação do manipulador	Rotação inconsistente com o desejado	Perda de desempenho (velocidade; controle)	6	Emperramento parcial	4	6	144	Inspeções 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 10: *FMECA* do sub-sistema de processamento

Análise do Tipo e Efeito de Falha								
Sub-sistema de processamento								
Função(ões)	Modo(s) de falha em potencial	Efeito(s) potencial(is) da falha	S	Causa(s) potenciais	O	D	C	Ação(ões) 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	Inspeções periódicas em bancada

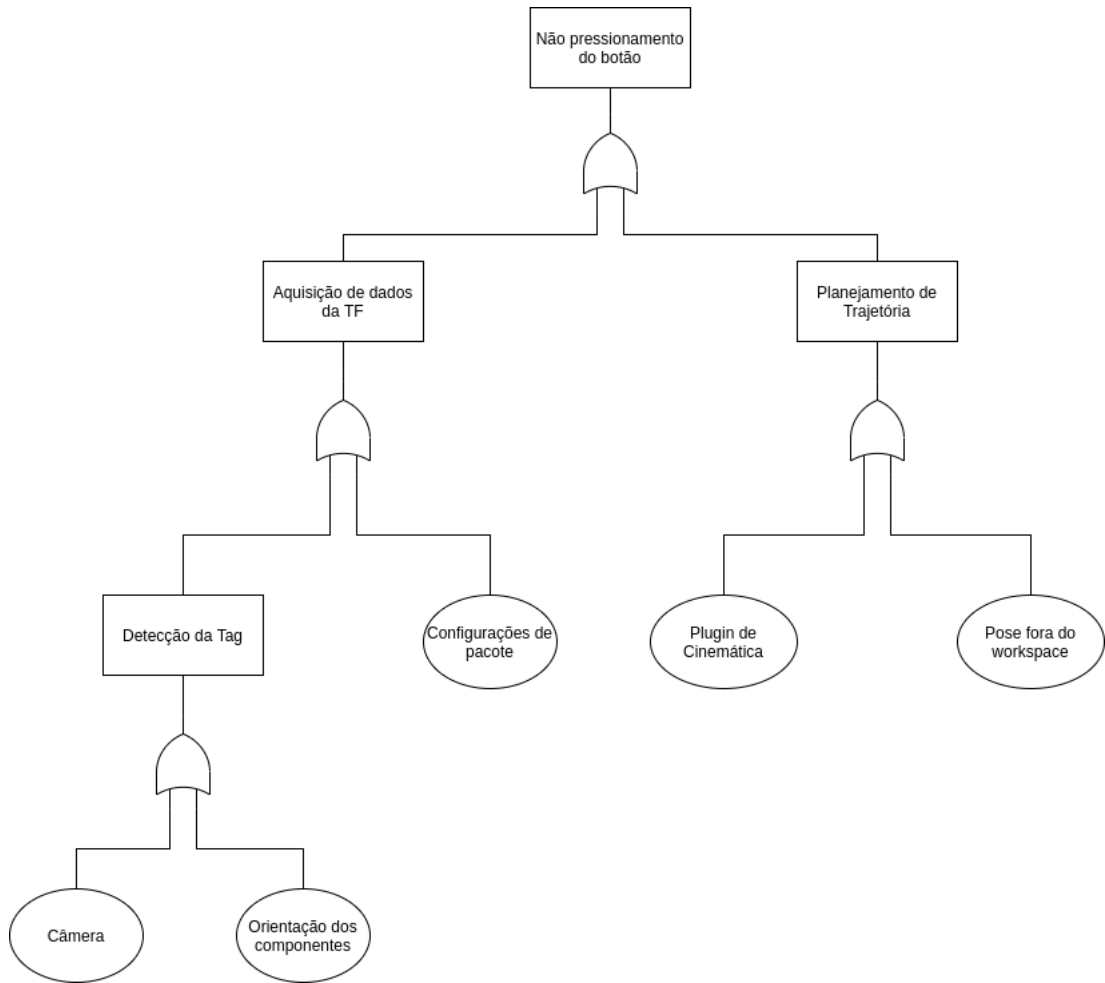
Fonte: Autoria própria.

5.2 Análise da árvore de falhas

Na Figura 16 encontra-se a árvore de falhas do sistema do manipulador Timon-HM. 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 16: Árvore de falhas do sistema.



Fonte: Autoria própria.

6 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.

Além das lições aprendidas, a seção 6.2 traz o guia de uso com o propósito de auxiliar o usuário na replicação dos experimentos realizados neste relatório.

6.1 Lições aprendidas

A Tabela 11 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.

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 definindo 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 Trac-IK	Maior eficiência de planejamento	Pesquisa e consultoria prévia do pacote mais adequado para o projeto

Tabela 11: Lições aprendidas.

6.2 Guia de uso

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
```

- 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-*
```

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

- Compilar o *workspace*:

```
$ catkin_make
```

- Ativar o ambiente virtual do *workspace*:

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

Após realizar os procedimentos citados o *workspace* estará configurado para executar a simulação, com os seguintes comandos:

- 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.launch
```

¹ Pasta onde contém os arquivos fonte.

7 CONCLUSÃO

O presente relatório descreveu a idealização e simulação de Timon-HM, 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 definida e modelos *URDF* descrevendo a mesma e o ambiente no qual o robô estava inserido foram desenvolvidos.

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*.

Um pacote capaz de comunicar o sistema de escaneamento, composto por uma câmera RGB e o pacote *Bir Marker Localization*, e o sistema de planejamento/execução de trajetória foi desenvolvido utilizando a linguagem C++. A partir da identificação da *tag ArUco*, o sistema é capaz de localizar o painel elétrico e planejar uma trajetória que leve o *endeffector* do manipulador até o alvo estabelecido.

Foram realizados 60 testes considerando diferentes posições e orientações para o painel elétrico. Os resultados alcançados mostram que Timon-HM foi capaz de realizar a tarefa em 90% dos casos, com tempo médio de execução de 24 segundos, resultados estes considerados satisfatórios para o prosseguimento do projeto.

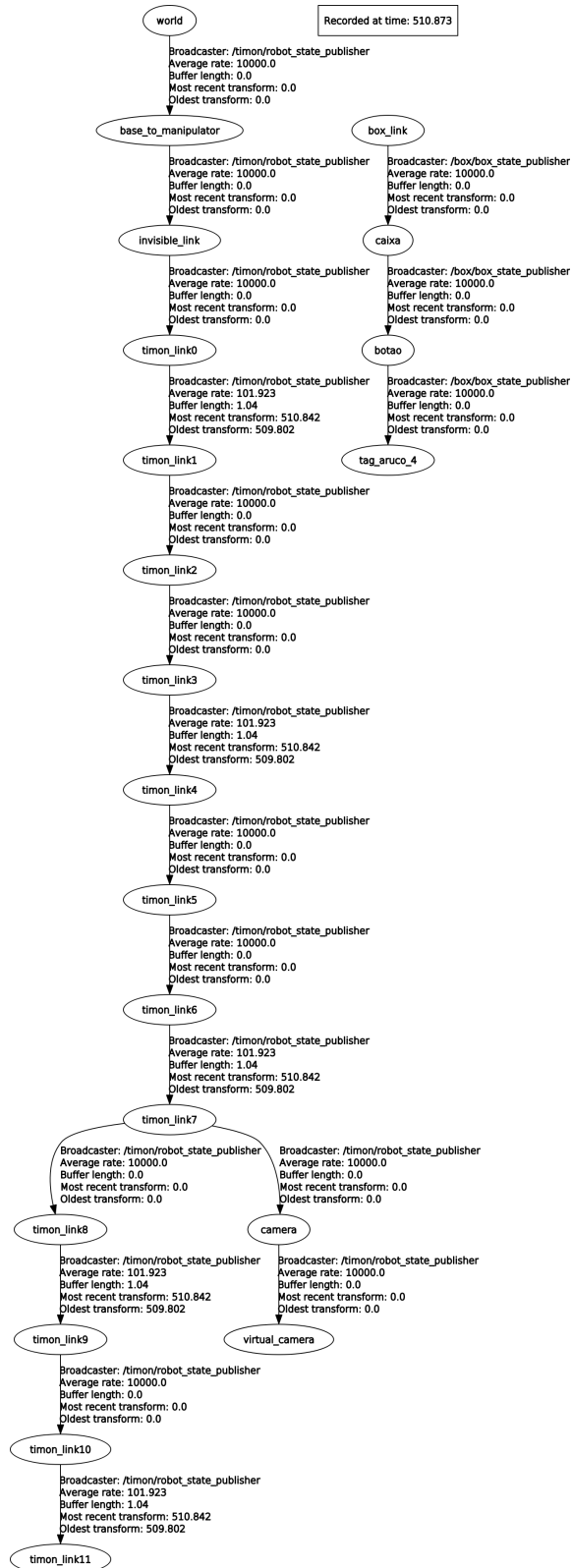
Futuramente a implementação do modelo físico será realizada em laboratório e novos testes serão executados a fim de comprovar o bom funcionamento do sistema. Por fim, o manipulador será instalado em um robô móvel *Warthog* para autonomamente localizar e desarmar uma bomba hipotética instalada em ambiente aberto.

REFERÊNCIAS

- BEESEON, 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 24.
- BIR Marker Localization. 2020. <https://github.com/Brazilian-Institute-of-Robotics/bir_marker_localization/>. Accessed: 2020-04-24. Citado na página 22.
- 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 13.
- HERNÁNDEZ-ORDOÑEZ, 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 13.
- 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 15.
- 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 14.
- 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 12.
- OLIVEIRA, G. T. d. S. et al. Projeto ótimo de robôs manipuladores 3r considerando a topologia do espaço de trabalho. Universidade Federal de Uberlândia, 2012. Citado na página 11.
- SANTOS, V. M. Robótica industrial. *Universidade de Aveiro-Departamento de Engenharia Mecânica*, 2004. Citado 2 vezes nas páginas 13 e 14.
- 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 24.
- 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 14.

APÊNDICE A

Árvores de TF desconectadas



APÊNDICE B

Árvores de TF conectadas

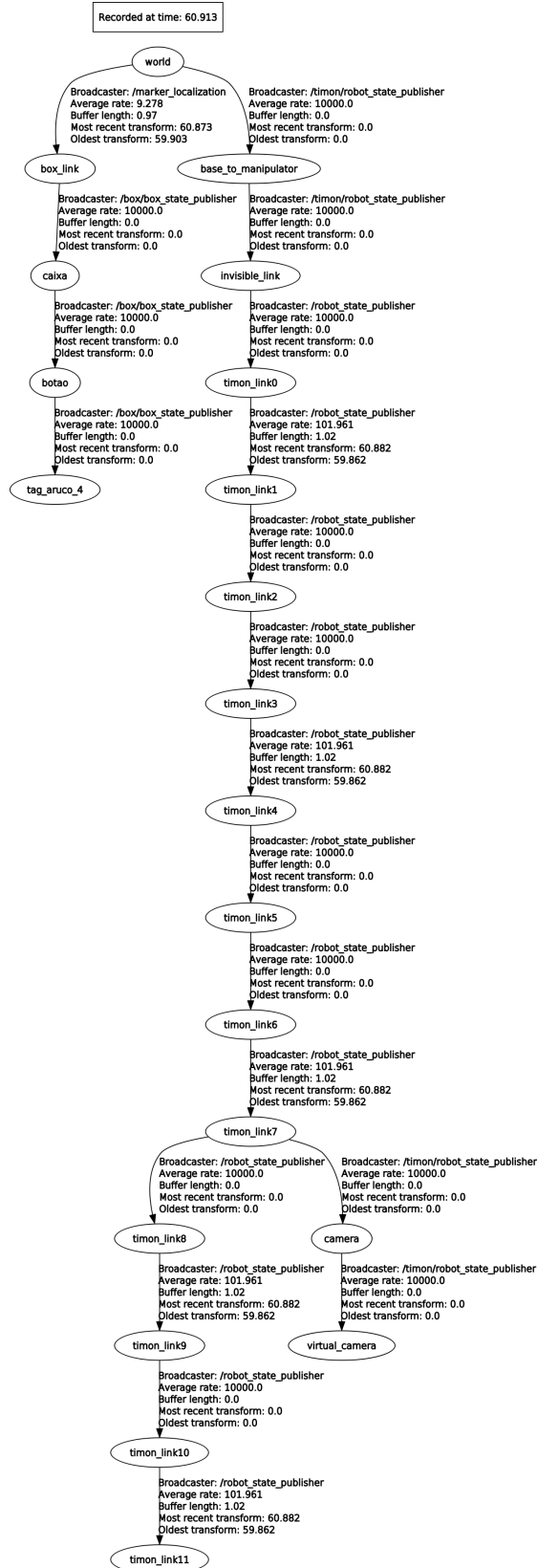
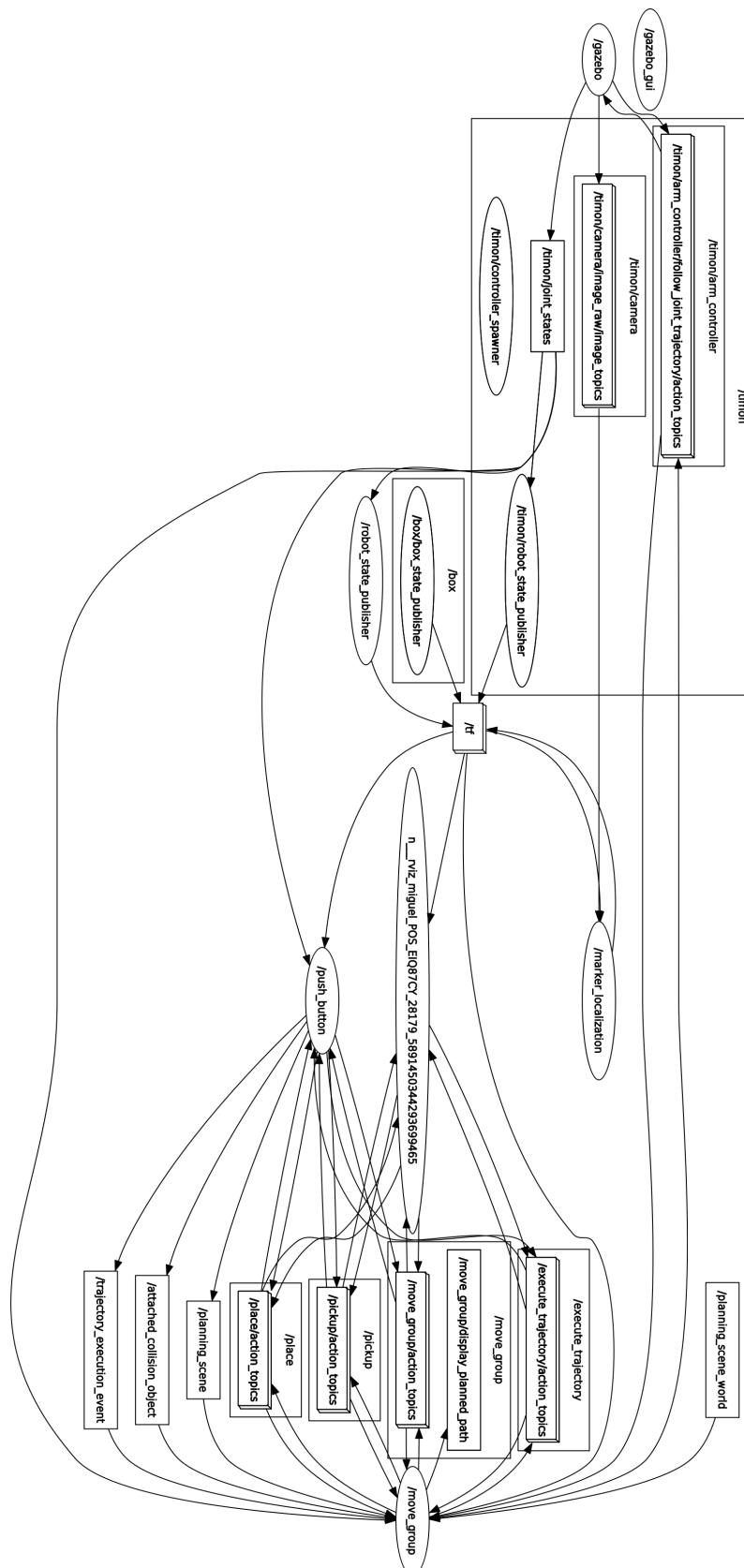


Diagrama de nós do sistema



APÊNDICE D

Propriedades de Massa do Timon-HM

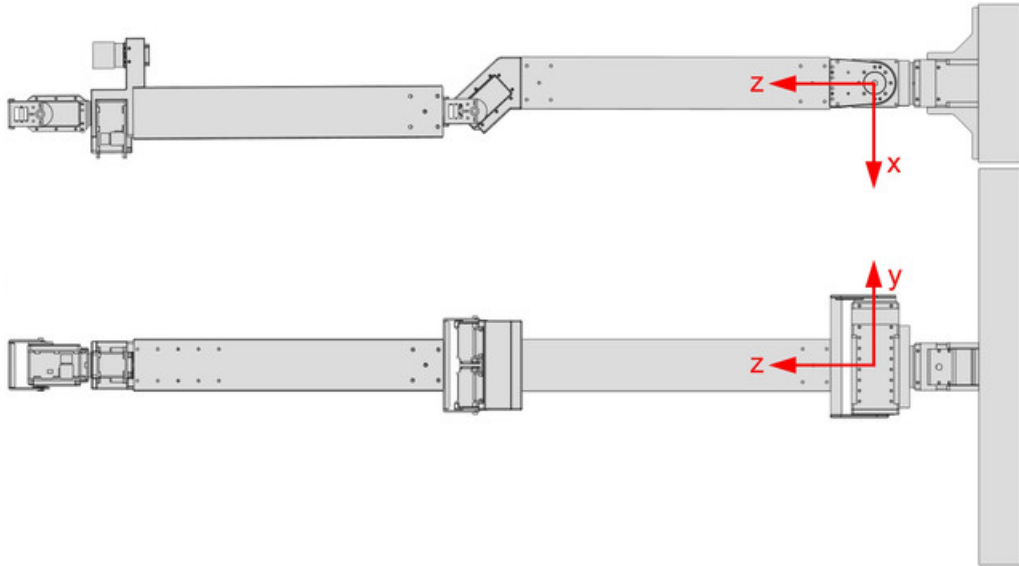


Figura 17: Coordenadas.

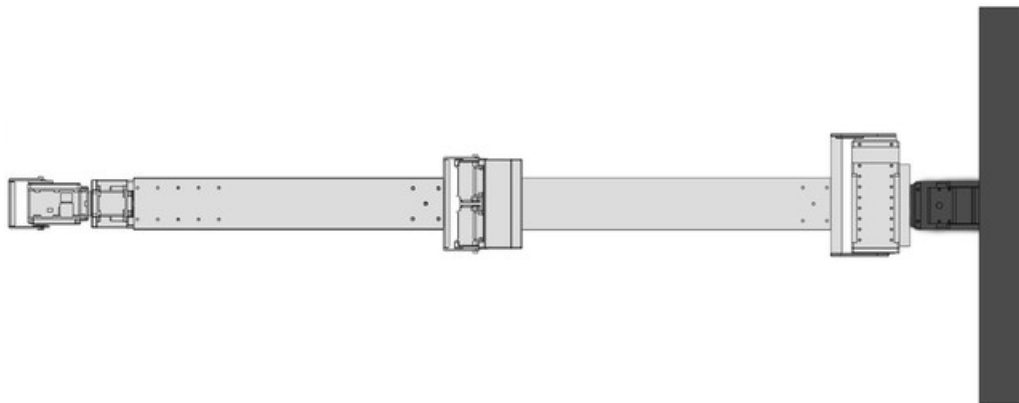


Figura 18: Link 0.

- **Massa:** 3.72264467 kg
- **Volume:** $0.00412764m^3$
- **Área:** $0.28500950m^2$
- **Centro de Massa:**
 - X: -0.00000012 m
 - Y: 0.00000000 m

– Z: 0.03490035 m

- **Momento de inércia:** $\text{kg } m^2$

– **Lxx:** 1.06821989e-2 **Lxy:** -2.59829620e-6 **Lxz:** 1.68828926e-8

– **Lyx:** -2.59829620e-6 **Lyy:** 4.86112355e-2 **Lyx:** 0.00000000e+0

– **Lzx:** 1.68828926e-8 **Lzy:** 0.00000000e+0 **Lzz:** 5.38837109e-2

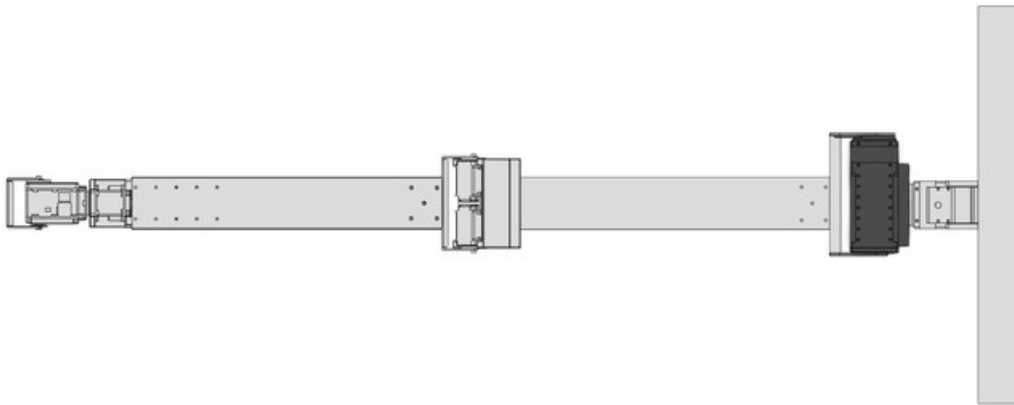


Figura 19: Link 1.

- **Massa:** 1.00643107 kg

- **Volume:** 0.00040209 m^3

- **Área:** 0.05733417 m^2

- **Centro de Massa:**

– X: 0.01039500 m

– Y: 0.00005146 m

– Z: 0.16006794 m

- **Momento de inércia:** $\text{kg } m^2$

– **Lxx:** 5.52840983e-4 **Lxy:** -4.82304098e-6 **Lxz:** -5.15873870e-5

– **Lyx:** -4.82304098e-6 **Lyy:** 1.52709797e-3 **Lyx:** -2.55376135e-7

– **Lzx:** -5.15873870e-5 **Lzy:** -2.55376135e-7 **Lzz:** 1.41817064e-3

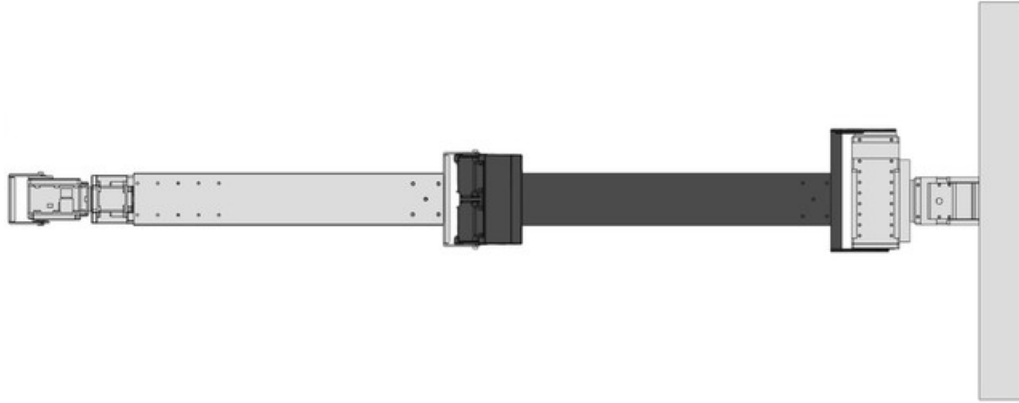


Figura 20: Link 2.

- **Massa:** 1.43140941 kg
- **Volume:** 0.00072217 m^3
- **Área:** 0.30338280 m^2
- **Centro de Massa:**
 - X: 0.00402379 m
 - Y: -0.00489519 m
 - Z: 0.42191352 m
- **Momento de inércia:** kg m^2
 - **Lxx:** 4.01177388e-2 **Lxy:** -1.99033225e-6 **Lxz:** 7.25766892e-4
 - **Lyx:** -1.99033225e-5 **Lyy:** 4.07927622e-2 **Lyz:** 1.35014731e-3
 - **Lzx:** 7.25766892e-4 **Lzy:** 1.35014731e-3 **Lzz:** 1.93910351e-3

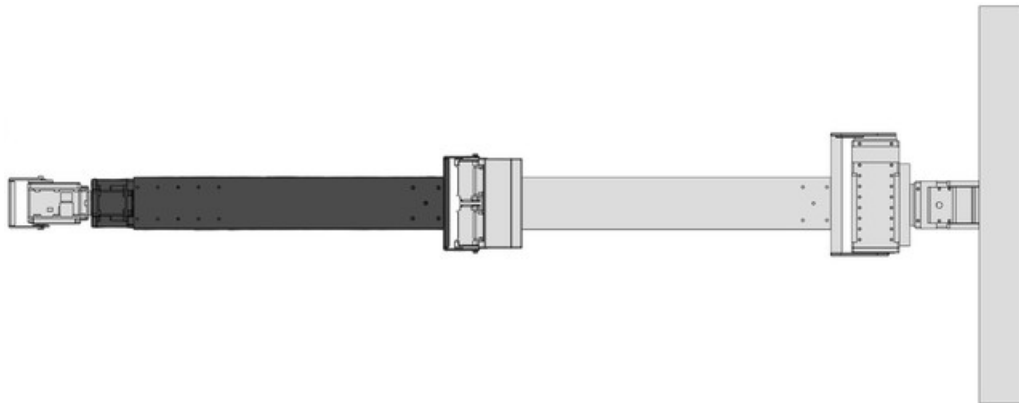


Figura 21: Link 3.

- **Massa:** 1.02430185 kg

- **Volume:** 0.00048651 m^3
- **Área:** 0.27643743 m^2
- **Centro de Massa:**
 - X: 0.00177442 m
 - Y: -0.02543863 m
 - Z: 0.88744337 m
- **Momento de inércia:** kg m^2
 - **Lxx:** $2.15420518\text{e-}2$ **Lxy:** $4.48113214\text{e-}6$ **Lxz:** $7.29467793\text{e-}6$
 - **Lyx:** $4.48113214\text{e-}6$ **Lyy:** $2.09581064\text{e-}2$ **Lyz:** $-9.20042318\text{e-}4$
 - **Lzx:** $7.29467793\text{e-}6$ **Lzy:** $-9.20042318\text{e-}4$ **Lzz:** $1.37743730\text{e-}3$

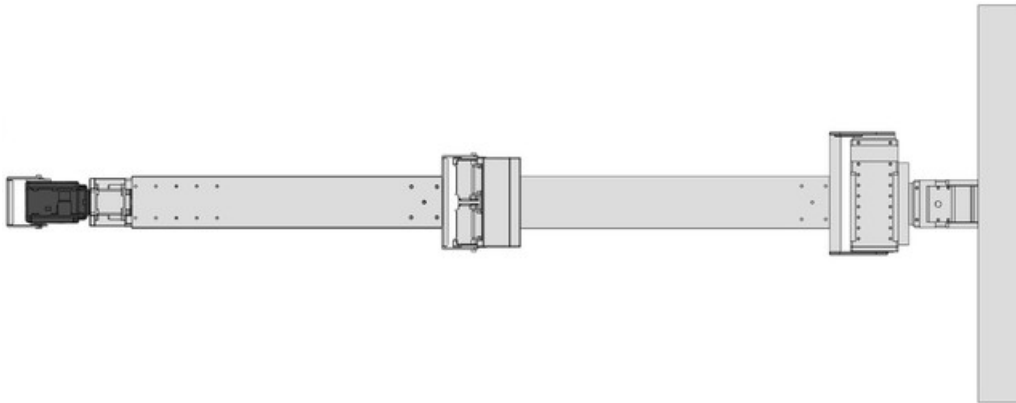


Figura 22: Link 4.

- **Massa:** 0.17387384 kg
- **Volume:** 0.00008979 m^3
- **Área:** 0.02133374 m^2
- **Centro de Massa:**
 - X: 0.00204423 m
 - Y: -0.03416008 m
 - Z: 1.09140950 m
- **Momento de inércia:** kg m^2
 - **Lxx:** $7.27594009\text{e-}5$ **Lxy:** $3.60335785\text{e-}7$ **Lxz:** $-1.43417976\text{e-}6$

- **Lyx:** 3.60335785e-7 **Lyy:** 9.11402947e-5 **Lyz:** 1.03937291e-8
- **Lzx:** -1.43417976e-6 **Lzy:** 1.03937291e-8 **Lzz:** 4.72120463e-5

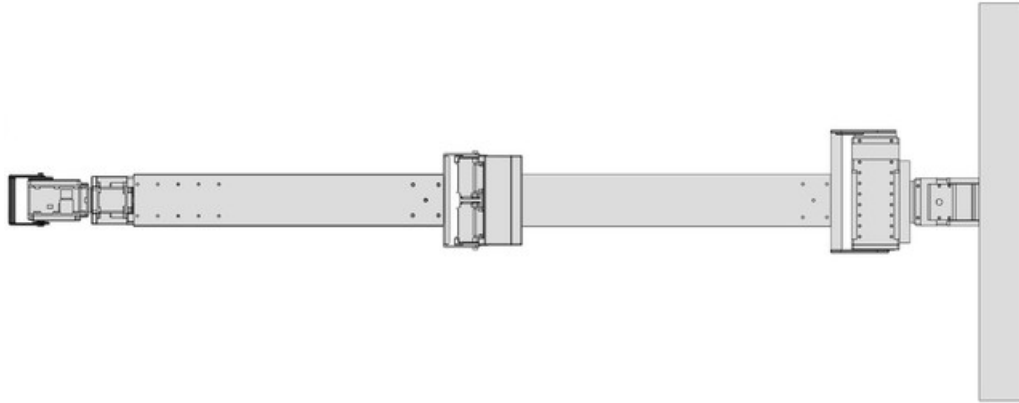


Figura 23: Link 5.

- **Massa:** 0.01873963 kg
- **Volume:** 0.00000694 m^3
- **Área:** 0.00883439 m^2
- **Centro de Massa:**
 - X: 0.00184227 m
 - Y: -0.03451643 m
 - Z: 1.13452531 m
- **Momento de inércia:** kg m^2
 - **Lxx:** 4.49832477e-6 **Lxy:** 1.27964175e-7 **Lxz:** 9.56680130e-10
 - **Lyx:** 1.27964175e-7 **Lyy:** 1.12015451e-5 **Lyz:** 4.61798902e-9
 - **Lzx:** 9.56680130e-10 **Lzy:** 4.61798902e-9 **Lzz:** 1.08114418e-5

APÊNDICE E

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>

int main(int argc, char** argv)
{
    ros::init(argc, argv, "push_button");
    ros::NodeHandle nh;
    ros::AsyncSpinner spinner(2);
    spinner.start();
    static const std::string PLANNING_GROUP = "timon_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, "Timon push_button
    demonstration", 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());

std::cout << move_group.getCurrentPose();

//ADDING FLOOR
// Define a collision object ROS message.
moveit_msgs::CollisionObject collision_object;
collision_object.header.frame_id = move_group.getPlanningFrame
    ();
moveit_msgs::CollisionObject collision_object1;
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.
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;
shape_msgs::SolidPrimitive primitive1;
primitive1.type = primitive.BOX;
primitive1.dimensions.resize(3);
primitive1.dimensions[0] = 0.2;
primitive1.dimensions[1] = 0.2;
primitive1.dimensions[2] = 0.3;
// 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 = -0.11;
    geometry_msgs::Pose box_pose1;
box_pose1.orientation.w = 0.0;
box_pose1.position.x = 0.0;
box_pose1.position.y = -1.06;
box_pose1.position.z = 0.0;

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);
planning_scene_interface.addCollisionObjects(collision_objects
);

//GET MANIPULATOR STATE
robot_state::RobotState start_state(*move_group.
    getCurrentState());
move_group.setStartState(start_state);
//BEGIN POSE
    geometry_msgs::Pose target_pose;
    target_pose.orientation.w = 0.474989;
    target_pose.orientation.x = -0.522257;
    target_pose.orientation.y = 0.47662;
    target_pose.orientation.z = -0.523895;
    target_pose.position.x = 0.000603714;
    target_pose.position.y = -0.335967;
    target_pose.position.z = 0.305978;
    move_group.setPoseTarget(target_pose);

moveit::planning_interface::MoveGroupInterface::Plan my_plan;
move_group.plan(my_plan);
move_group.execute(my_plan);

```

```

tf::TransformListener listener;
//GETTING BUTTON POSE
tf::StampedTransform transform;
try{
// ros::Time now = ros::Time::now();
listener.waitForTransform("/fake_botao", "/invisible_link",
                        ros::Time(0), ros::Duration(2.0));
listener.lookupTransform("/fake_botao", "/invisible_link",
                        ros::Time(0), transform);
}
catch (tf::TransformException &ex) {
ROS_ERROR("%s", ex.what());
// ros::Duration(1.0).sleep();
}
//CHECK BOX ORIENTATION
//HORIZONTAL
if (transform.getRotation().w() >= 0.99) {
    robot_state::RobotState start_state6(*move_group.
        getCurrentState());
    move_group.setStartState(start_state6);

    //POSE
    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(0.3);
    move_group.setPlanningTime(20);

    move_group.setPoseTarget(target_pose);
    move_group.plan(my_plan);
    move_group.execute(my_plan);

    //PRESS BUTTON
    robot_state::RobotState start_state7(*move_group.
        getCurrentState());
    move_group.setStartState(start_state7);
    target_pose.position.z -=0.050;
    move_group.setPoseTarget(target_pose);

```



```

move_group.plan(my_plan);
move_group.execute(my_plan);

robot_state::RobotState start_state8(*move_group.
    getCurrentState());
move_group.setStartState(start_state8);
target_pose.position.z +=0.050;
move_group.setPoseTarget(target_pose);
move_group.plan(my_plan);
move_group.execute(my_plan);
}

//VERTICAL
else{
robot_state::RobotState start_state2(*move_group.
    getCurrentState());
move_group.setStartState(start_state2);
//POSE
target_pose.orientation.w = 0;
target_pose.orientation.x = 0.707;
target_pose.orientation.y = 0;
target_pose.orientation.z = -0.707;
target_pose.position.x = transform.getOrigin().x();
target_pose.position.y = -transform.getOrigin().z();
target_pose.position.z = -transform.getOrigin().y();

move_group.setGoalPositionTolerance(0.001);
move_group.setGoalOrientationTolerance(0.3);
move_group.setPlanningTime(20);
move_group.setPoseTarget(target_pose);

move_group.plan(my_plan);
move_group.execute(my_plan);

//PRESS BUTTON
robot_state::RobotState start_state3(*move_group.
    getCurrentState());
move_group.setStartState(start_state3);
target_pose.position.y -=0.051;
move_group.setPoseTarget(target_pose);

```

```

    move_group.plan(my_plan);
    move_group.execute(my_plan);

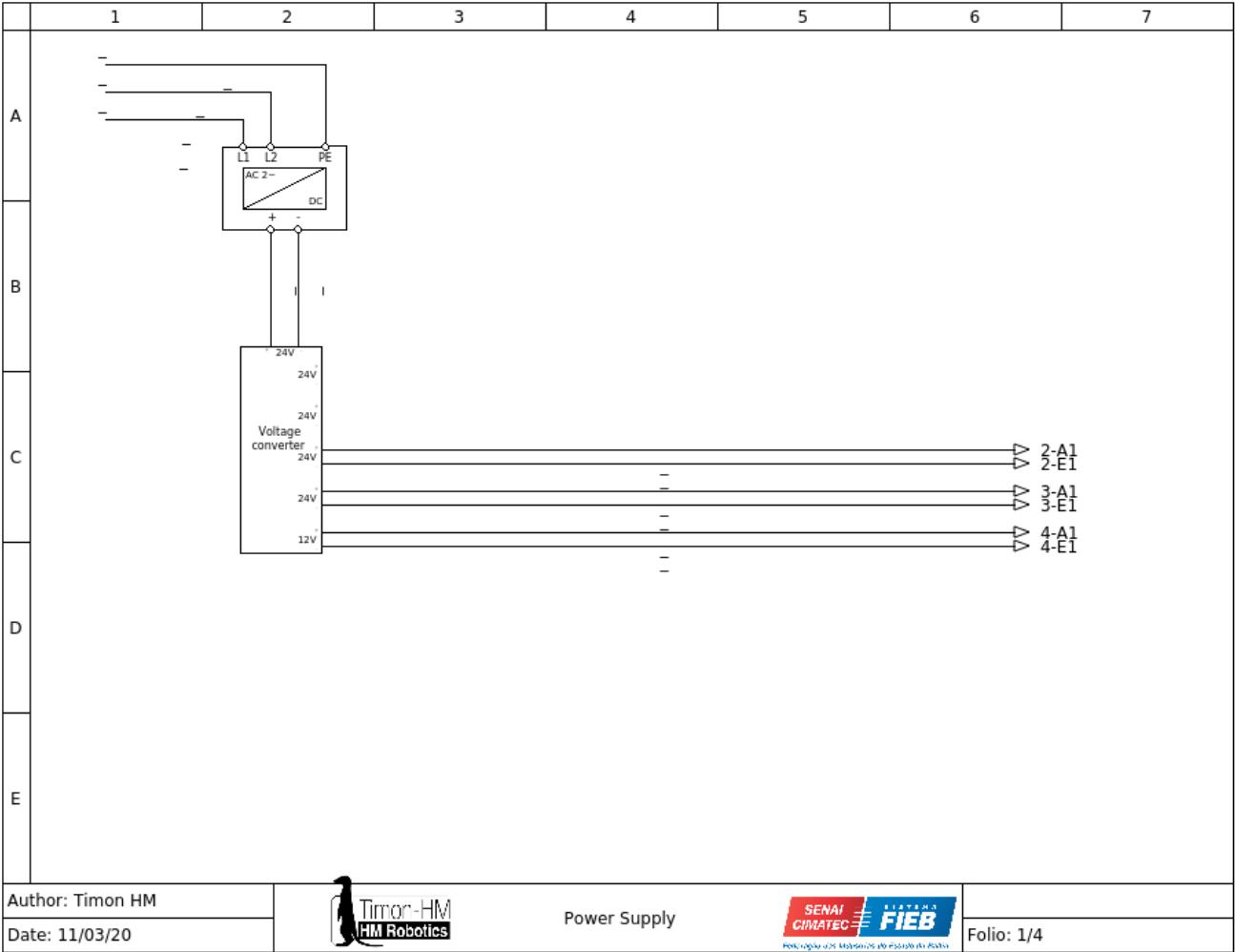
    robot_state::RobotState start_state4(*move_group.
        getCurrentState());
    move_group.setStartState(start_state4);
    target_pose.position.y +=0.051;
    move_group.setPoseTarget(target_pose);
    move_group.plan(my_plan);
    move_group.execute(my_plan);
}
//BACK TO UP POSITION
robot_state::RobotState start_state5(*move_group.
    getCurrentState());
move_group.setStartState(start_state5);
moveit::core::RobotStatePtr current_state3 = move_group.
    getCurrentState();
std::vector<double> joint_group_positions3;
current_state3->copyJointGroupPositions(joint_model_group,
    joint_group_positions3);
joint_group_positions3[0] = 0;
joint_group_positions3[1] = 0;
joint_group_positions3[2] = 0;
joint_group_positions3[3] = 0;
joint_group_positions3[4] = 0;
move_group.setJointValueTarget(joint_group_positions3);
move_group.plan(my_plan);
move_group.execute(my_plan);
//REMOVE OBJECTS
std::vector<std::string> object_ids;
object_ids.push_back(collision_object.id);
object_ids.push_back(collision_object1.id);
planning_scene_interface.removeCollisionObjects(object_ids);

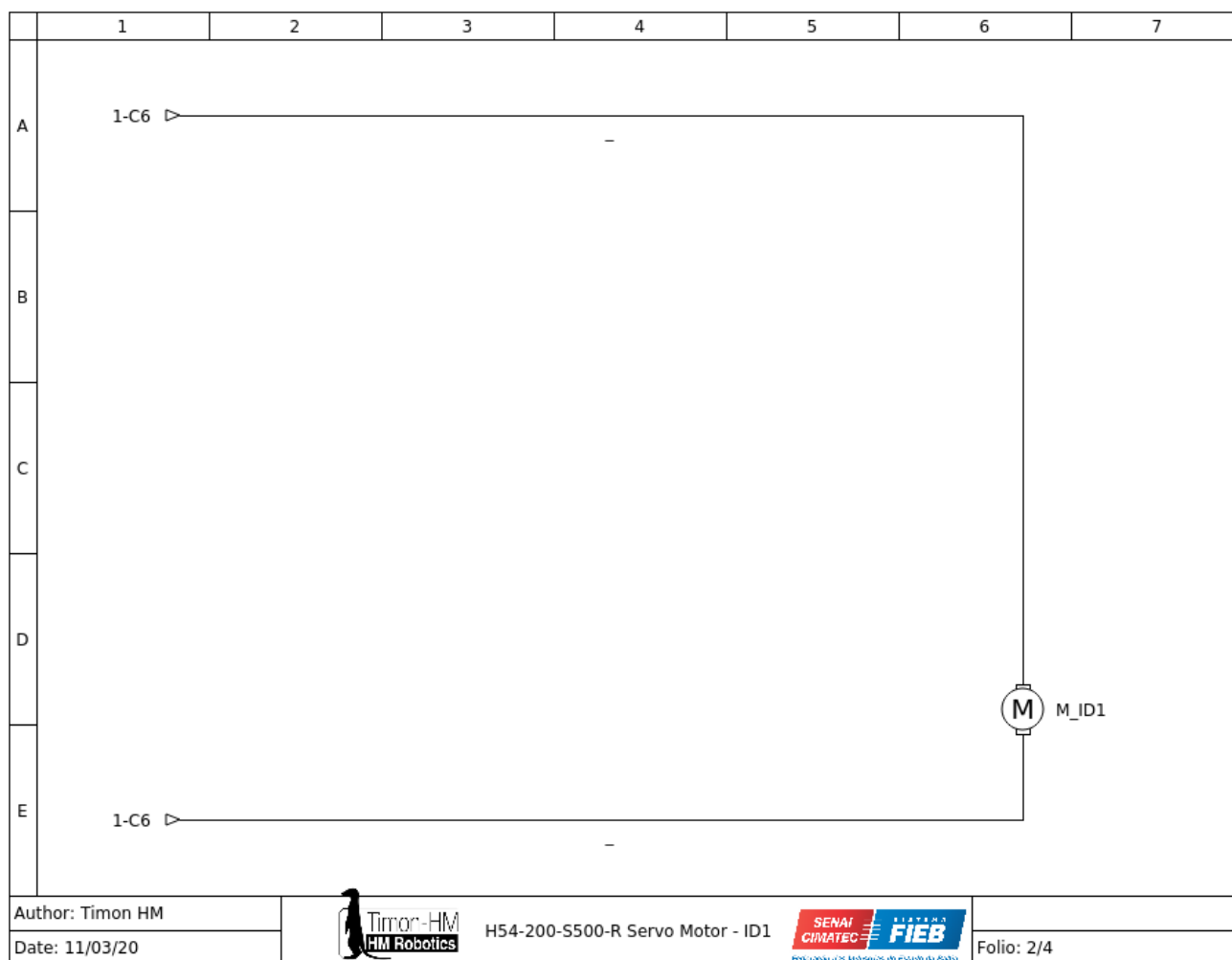
ros::shutdown();
    return 0;
}

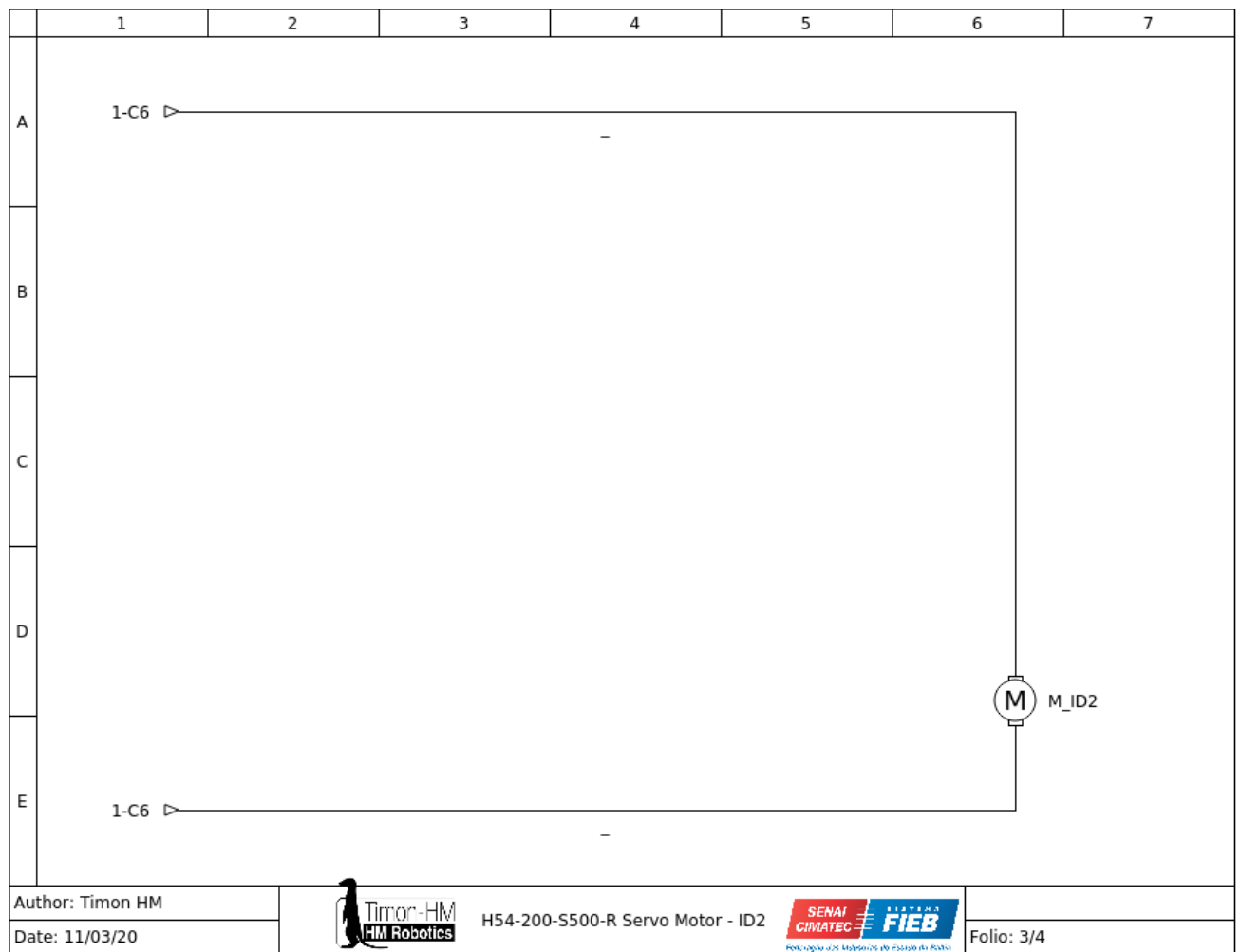
```

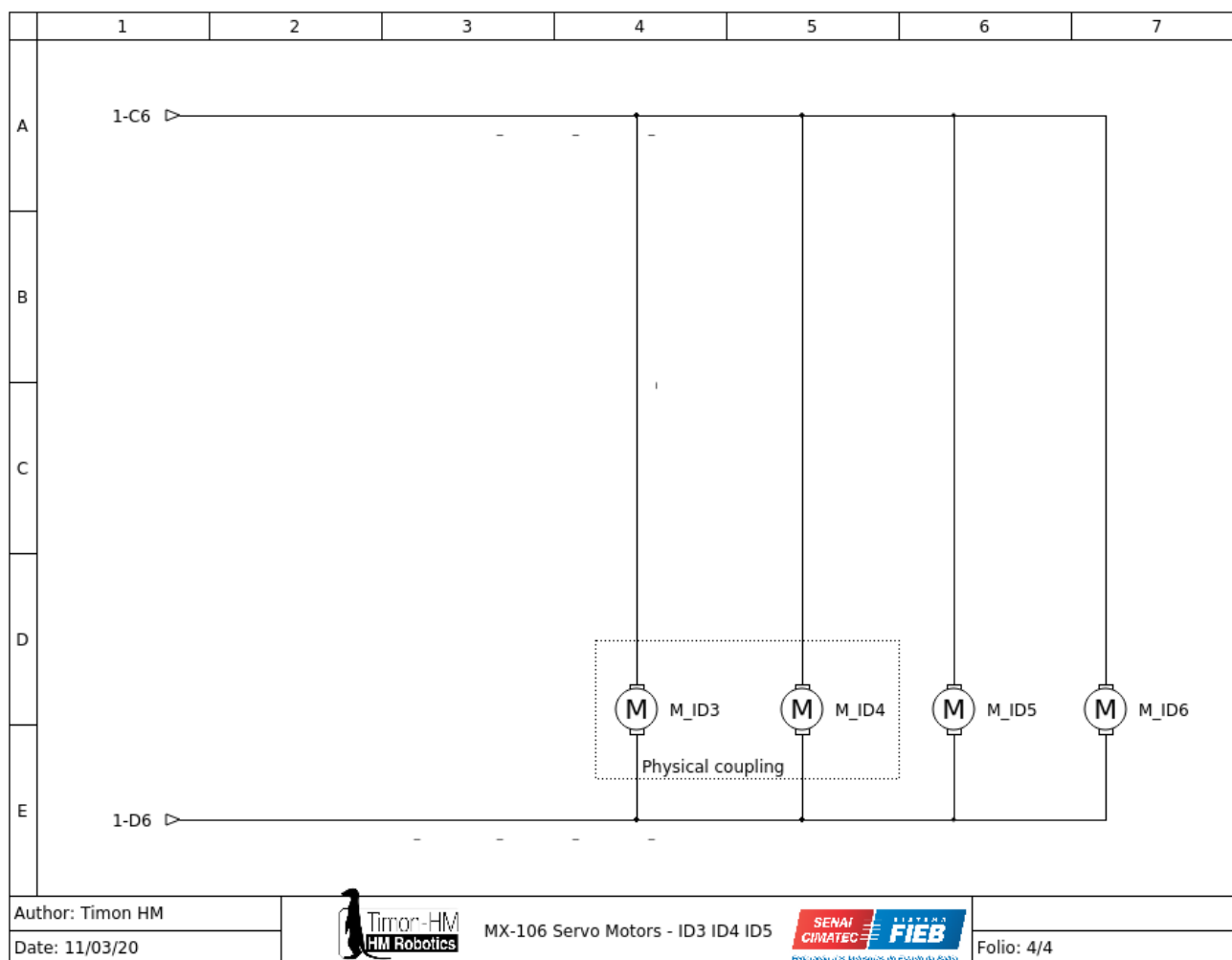
APÊNDICE F

Diagrama elétrico do Timon-HM



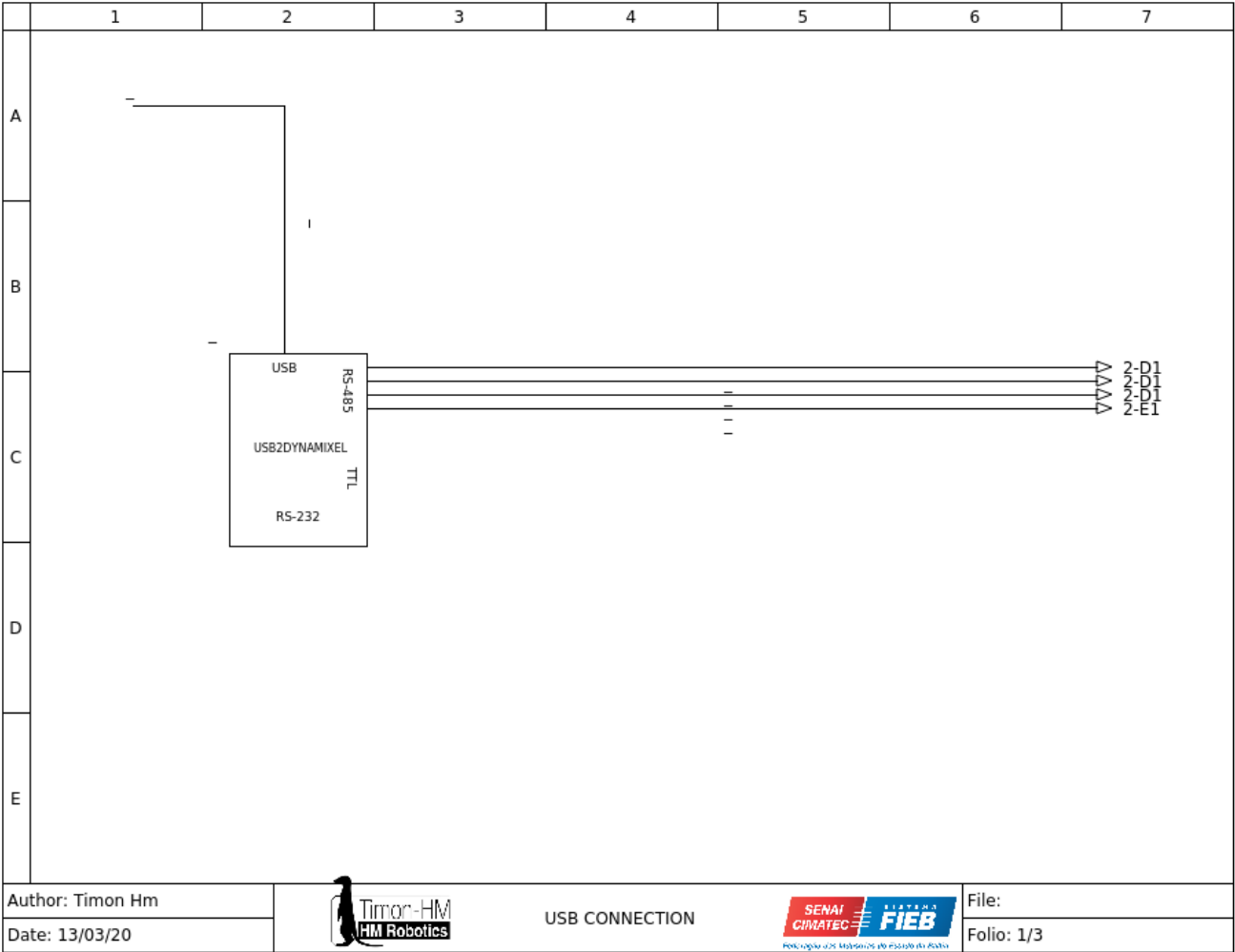


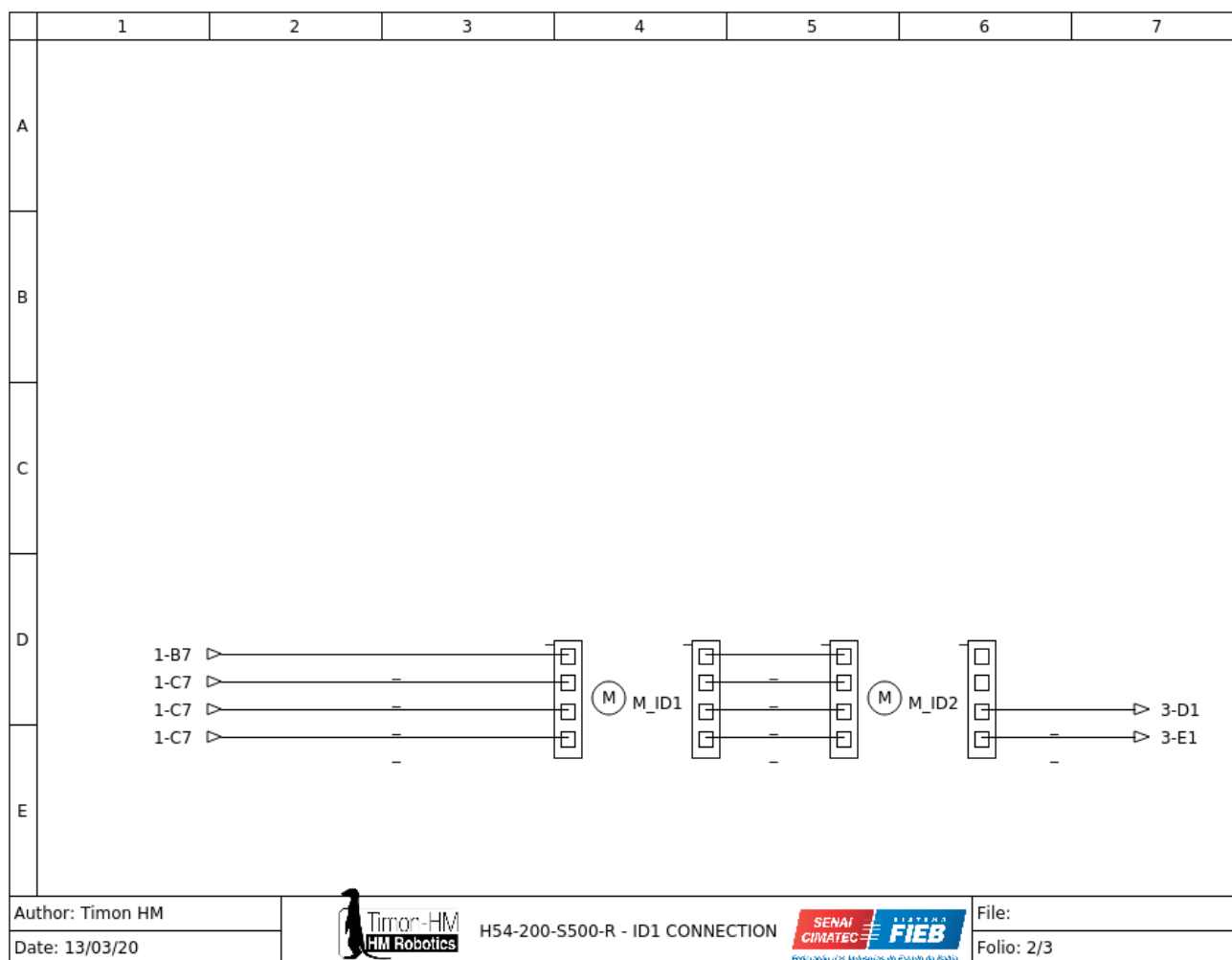


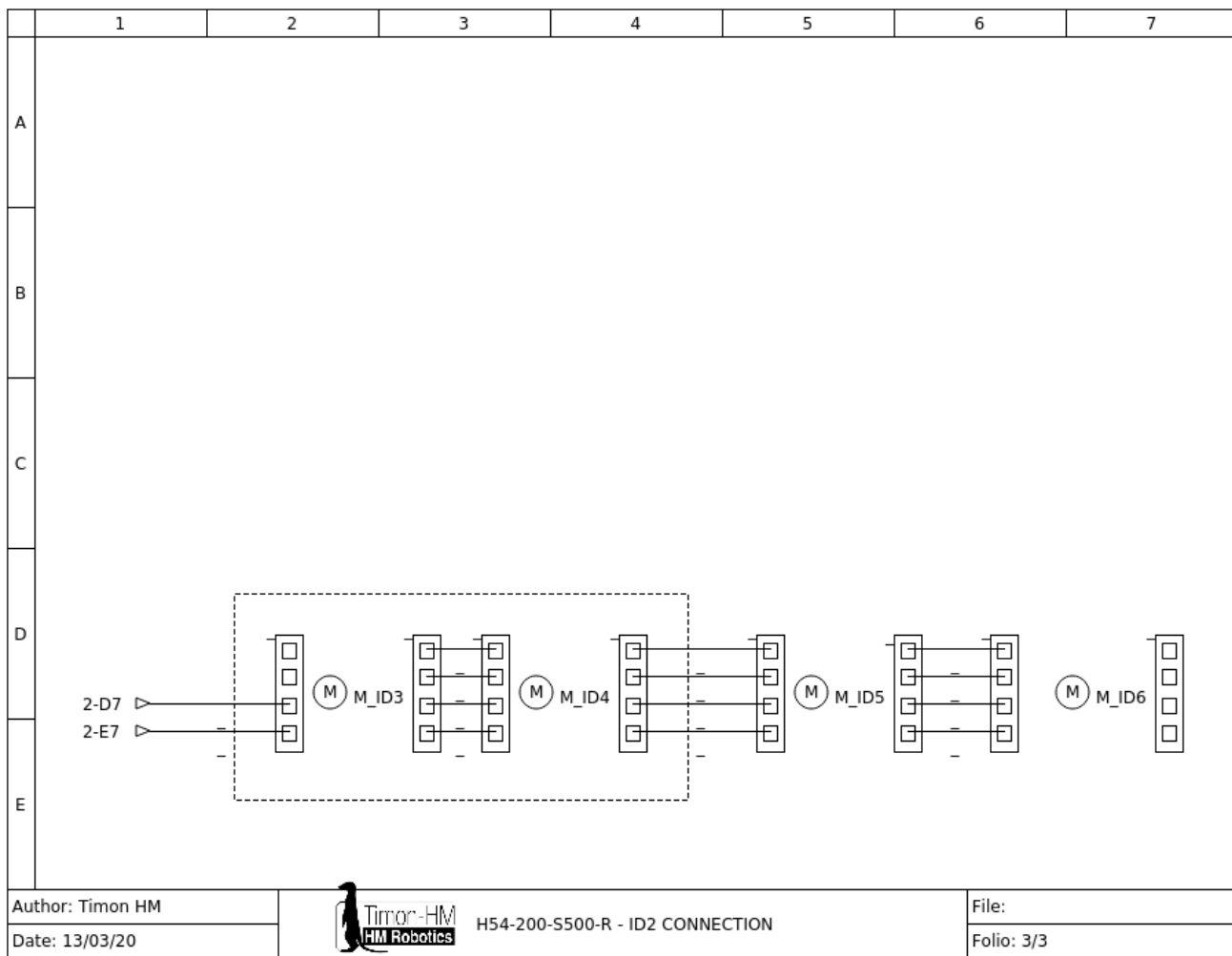


APÊNDICE G

Diagrama de conexão do Timon-HM







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 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 μ s 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"

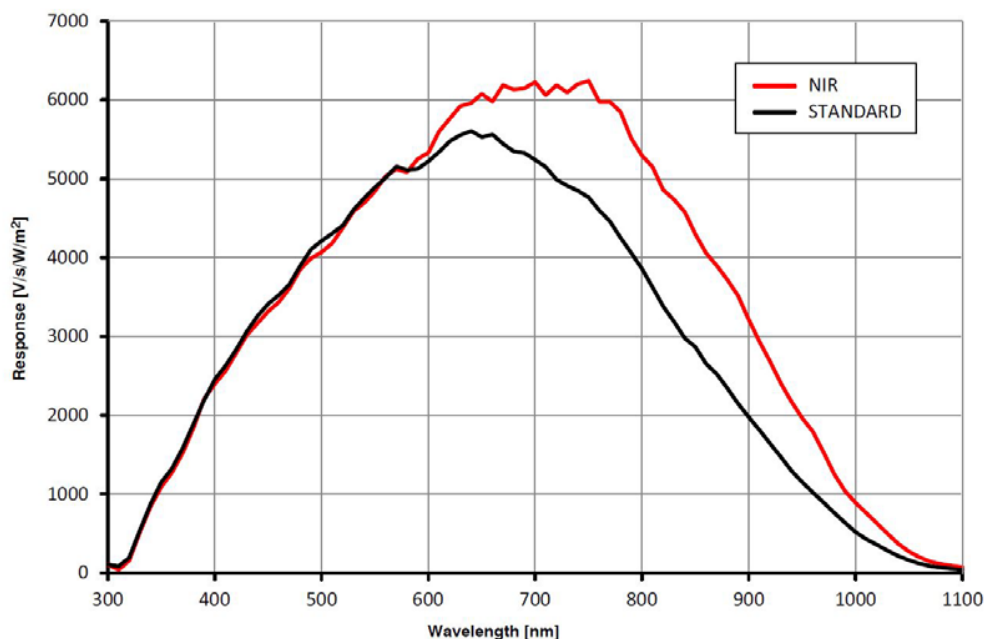
C1280, C1930, C2590

- 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"

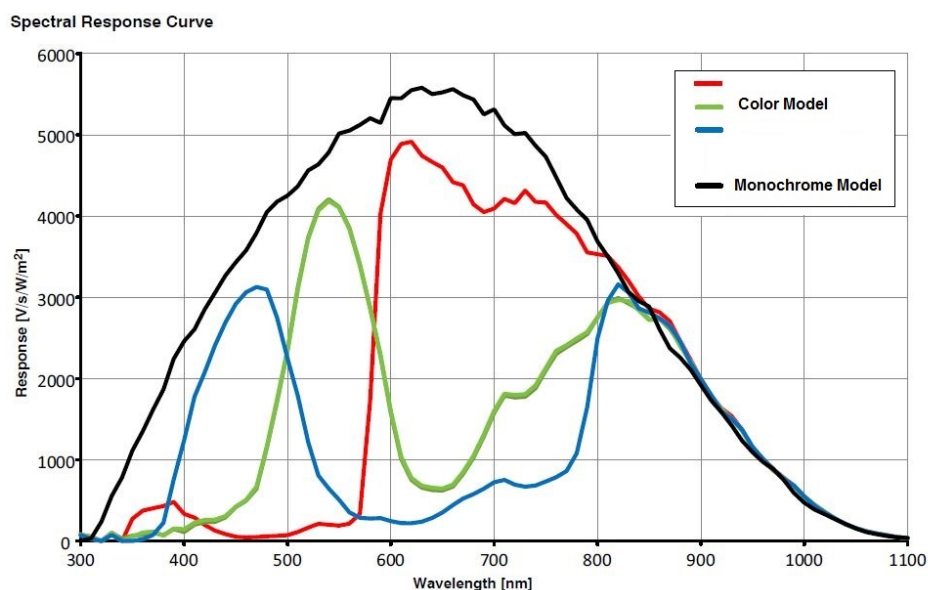
Spectral Response (Python 4.8 μm series)

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 MX-106

Item	Especificações
MCU	ARM CORTEX-M3 (72 [MHz], 32 Bit)
Sensor de posição	Enconder absoluto (sem contato) (12 Bit, 360 [°]) Fabricante: AMS , Numero da peça : AS5045
Motor	Coreless (Maxon)
Taxa de transmissão	8,000 [bps] ~4.5 [Mbps]
Algoritmo de controle	Controle PID
Resolução	4096 [pulse/rev]
Folga	20 [arcmin] (0.33 [°])
Modo operacional	Modo junta (0 ~ 360 [°]) Modo roda (Curso sem fim)
Peso	165 [g]
Dimensões (c x l x h)	40.2 x 65.1 x 46 [mm]
Relação de engrenagem	225 : 1
Torque de Parada	8.0 [Nm] (at 11.1 [V], 4.8 [A]) 8.4 [Nm] (at 12.0 [V], 5.2 [A]) 10.0 [Nm] (at 14.8 [V], 6.3 [A])
Velocidade (sem carga)	41 [rev/min] (at 11.1 [V]) 45 [rev/min] (at 12.0 [V]) 55 [rev/min] (at 14.8 [V])
Carga radial	40 [N] (10 [mm] afastado da face do eixo)
Carga axial	20 [N]
Temperatura de operação	-5 ~+80 [°C]
Tensão de entrada	10.0 ~14.8 [V] (Recomendado : 12.0 [V])
Sinal de Comando	Pacote digital
Tipo de protocolo	RS485 Comunicação serial assíncrona com 8bit, 1stop, sem paridade
Conexão física	RS485 Barramento multiponto
ID	254 ID (0 ~253)
Feedback	Posição, temperatura, carga, tensão de entrada, etc
Material	engrenagem toda de metal Plástico de engenharia (frente, meio e verso) Metal (Frente)
Corrente de repouso	100 [mA]

Tabela 12: Especificações do motor *Dynamixel* MX-106

Enter Search Terms

MX-106T/R

1. Specifications

2. Control Table

3. How to Assemble

4. Maintenance

5. Reference



Edit on GitHub



ROBOTIS



Robot Source



GitHub



MX-106R, MX-106T

NOTE : Compliance has been replaced with PID Gains.

NOTE : Although the MX-106T (TTL) and MX-106R (RS-485) differ in communications protocols both have the same features and perform equally. (TTL uses 3-pin connectors while RS-485 uses 4)

NOTE : In order to use Protocol 2.0, please update the firmware to V39 or above. ([Update firmware](#) using R+ Manager 2.0)

Dynamixel MX Series (Protocol v2.0) F/W Recovery



WARNING : For MX-106(2.0) Protocol, please refer to the [MX-106\(2.0\) Control Table](#) as they are different.

1. Specifications

Item	Specifications
MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
Position Sensor	Contactless absolute encoder (12Bit, 360 [°]) Maker : ams(www.ams.com), Part No : AS5045
Motor	Coreless(Maxon)
Baud Rate	8,000 [bps] ~ 4.5 [Mbps]
Control Algorithm	PID control
Resolution	4096 [pulse/rev]
Backlash	20 [arcmin] (0.33 [°])

Enter Search Terms

Q

MX-106T/R

1. Specifications

2. Control Table

3. How to Assemble

4. Maintenance

5. Reference

MX-106T/R	
Item	Specifications
Operating Mode	Joint Mode (0 ~ 360 [°]) Wheel Mode (Endless Turn)
Weight	165 [g]
Dimensions (W x H x D)	40.2 x 65.1 x 46 [mm]
Gear Ratio	225 : 1
Stall Torque	8.0 [Nm] (at 11.1 [V], 4.8 [A]) 8.4 [Nm] (at 12[V], 5.2 [A]) 10.0 [Nm] (at 14.8 [V], 6.3 [A])
No Load Speed	41 [rev/min] (at 11.1 [V]) 45 [rev/min] (at 12 [V]) 55 [rev/min] (at 14.8 [V])
Radial Load	① 40 [N] (10 [mm] away from the horn)
Axial Load	① 20 [N]
Operating Temperature	-5 ~ +80 [°C]
Input Voltage	10.0 ~ 14.8 [V] (Recommended : 12.0 [V])
Command Signal	Digital Packet
Protocol Type	TTL Half Duplex Asynchronous Serial Communication with 8bit, 1stop, No Parity RS485 Asynchronous Serial Communication with 8bit, 1stop, No Parity
Physical Connection	RS485 / TTL Multidrop Bus
ID	254 ID (0 ~ 253)
Feedback	Position, Temperature, Load, Input Voltage, etc
Material	Full Metal Gear Engineering Plastic(Front, Middle, Back) ① Metal(Front)
Standby Current	100 [mA]

▲TOP

① Applies to aluminum housing products(MX-28AR/AT, MX-64AR/AT, MX-106R/T).



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 polarity before wiring.



CAUTION

(May cause injury or damage to product)

- Do not operate the product at a temperature exceeding -5 ~ +80 [°C] range.
- Do not insert sharp blades nor pins during product operation.



ATTENTION

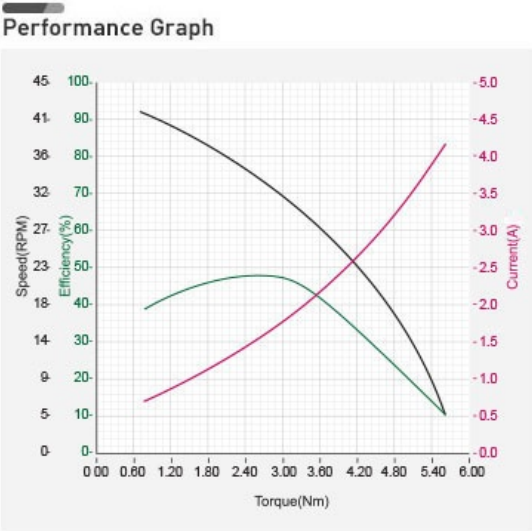
(May cause injury or damage to product)

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

Enter Search Terms
MX-106T/R
1. Specifications
2. Control Table
3. How to Assemble
4. Maintenance
5. Reference



1. 1. Performance 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.

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.

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 1.0](#) 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(24) is cleared to '0'(Off).

ANEXO C

Especificações do motor Dynamixel H54-S500-R

Item	Especificações
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 13: Especificações do motor *Dynamixel* H54-S500-R

ROBOTIS e-Manual

Enter Search Terms

Q

AM

SOFTWARE

PARTS

FAQ

H54-200-S500-R

1. Specifications


1. 1. Performance Graph


2. Control Table


3. How to Assemble


4. Maintenance


5. Reference


ROBOTIS

Robot Source

GitHub







H54-200-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
Weight	855 [g]
Dimensions (W x H x D)	54 x 126 x 54 [mm]
Resolution	501,923 [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]
① Continuous Speed	29.0 [rev/min]
① Continuous Torque	44.7 [Nm]
① Continuous Current	9.3 [A]
Output	200 [W]
Operating Temperature	5 ~ 55 [°C]
Input Voltage	24.0 [V]
Command Signal	Digital Packet

Enter Search Terms

Q

H54-200-S500-R

1. Specifications

1. 1. Performance Graph

2. Control Table

3. How to Assemble

4. Maintenance

5. Reference

Item	Specifications
Protocol Type	RS485 Asynchronous Serial Communication (8bit, 1stop, No Parity)
Physical Connection	RS485 Multidrop Bus
ID	253 ID (0 ~ 252)
Standby Current	80 [mA]

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 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.

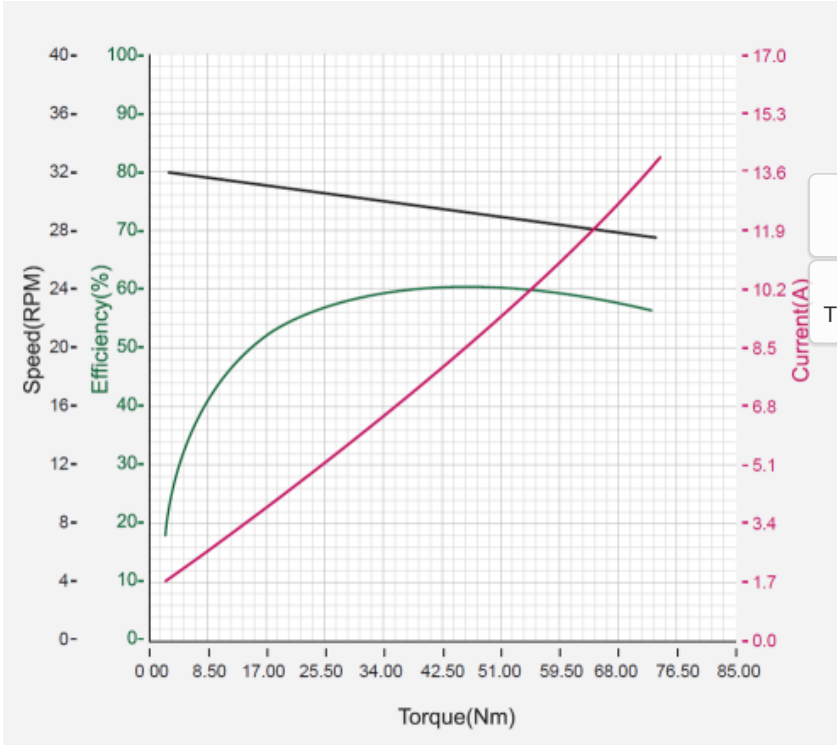


ATTENTION
(May cause injury or damage to product)

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

1. 1. Performance Graph

Enter Search Terms
H54-200-S500-R
1. Specifications
1. 1. Performance Graph
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.

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