



Federação das Indústrias do Estado da Bahia

CENTRO UNIVERSITÁRIO SENAI CIMATEC

Engenharia Elétrica

Trabalho de Conclusão do Curso

Desenvolvimento do robô de inspeção.

Apresentada por: Carlos Alberto Pereira
 Cleber Couto Filho
 Davi Costa
 Ícaro Nascimento

Orientador: Prof. Marco Reis, M.Eng.

Setembro de 2018

Carlos Alberto Pereira
Cleber Couto Filho
Davi Costa
Ícaro Nascimento

Desenvolvimento do robô de inspeção.

Trabalho de Conclusão do Curso apresentada
ao , Curso de Engenharia Elétrica do Centro
Universitário SENAI CIMATEC, como requi-
sito parcial para a obtenção do título de **Ba-
charel em Engenharia.**

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

Salvador
Centro Universitário SENAI CIMATEC
2016

Dedico este trabalho a ...

Agradecimentos

Salvador, Brasil
dia de Setembro de 2018

Carlos Alberto Pereira
Cleber Couto Filho
Davi Costa
Ícaro Nascimento

Resumo

Este documento contempla a descrição das etapas do desenvolvimento do projeto de Theoprax de Conclusão de curso, ELIR, robô autônomo de inspeção de linhas de transmissão, atendendo aos objetivos gerais e específicos e aos requisitos estabelecidos pelo cliente, tendo em vista a necessidade do projeto num cenário tanto comercial quanto acadêmico. Durante o desenvolvimento do projeto foi necessário realizar o estudo de conceitos de robótica, bem como os softwares necessários para implementação das funcionalidades, também estudadas e definidas pelo grupo. Em paralelo ao desenvolvimento das funcionalidades diversos testes foram realizados para validação dos conceitos e verificação de erros, em etapas de testes individuais partindo para a etapa de testes integrados. Os conceitos estudados e desenvolvidos pelo grupo durante todo o projeto fazem parte de uma grande contribuição tecnológica para a área de robótica e engenharia, sendo um projeto enriquecedor tanto para a equipe envolvida no desenvolvimento quanto para as gerações futuras interessadas no desenvolvimento tecnológico em robótica.

Palavras-chave: Robô de Inspeção, Linhas de Transmissão, Navegação, Cinemática Inversa, Manipuladores

Abstract

This document contain a description of the development stages of the Theoprax end of course project, the ELIR, a autonomous robot for inspection transmission lines, meeting the general and specific objectives and the requirements established by the client, considering the need of the project in a scenario both commercial and academic. During the development of the project, it was necessary to carry out the study of robotics concepts, as well as the software required to implement the functionalities, also studied and defined by the group. In parallel with the development of the functionalities, several tests were carried out to validate the concepts and verify errors, in individual test stages, starting to the integrated testing stage. The concepts studied and developed by the group throughout the project are part of a great technological contribution to the area of robotics and engineering, being a project enriching both for the team involved in in development and for the team involved in development and for future generations interested in the technological development in robotics.

Keywords: Inspection Robot,Transmission Lines,Navigation,Inverse Kinematics,Manipulators

Sumário

1	Introdução	1
1.1	Objetivos	2
1.1.1	Objetivos Específicos	2
1.2	Justificativa	2
1.3	Organização do Trabalho de Conclusão do Curso	4
2	Fundamentação Teória	6
2.1	Cinemática	7
2.1.1	Cinemática Direta	7
2.1.2	Cinemática Inversa	8
2.2	Modelagem Cinemática de um Braço Planar	9
2.3	Desenvolvimento de Robôs	10
2.3.1	<i>Framework</i>	10
2.3.2	Simulação	11
2.3.3	Odometria	12
2.3.4	Gestão de Energia	13
2.3.5	Conceito de segurança e Integridade	13
2.3.6	Comunicação em sistemas robóticos	14
3	Metodologia	15
3.1	Especificação dos componentes	15
3.1.1	Lista de componentes	15
3.1.1.1	Servomotores	15
3.1.1.2	Placa de Gerenciamento de Energia (Gerenciamento de Energia)	16
3.1.1.3	<i>ROS</i> (Robot Operating System)	17
3.1.1.4	<i>MoveIt!</i>	18
3.1.1.5	<i>Gazebo</i>	19
3.1.1.6	<i>Visual Studio Code</i>	19
3.1.1.7	<i>PlatformIO</i>	20
3.2	Diagramas elétricos do sistema de Movimentação	20
3.2.1	Esquemas eletrônicos	21
3.3	Especificação das funcionalidades	21
3.3.1	Planejamento de Movimento	21
3.3.1.1	Dependências	22
3.3.1.2	Premissas Necessárias	23
3.3.1.3	Descrição da Funcionalidade	23
3.3.1.4	Saídas	23
3.3.2	Atuação	24
3.3.2.1	Dependências	24
3.3.2.2	Premissas Necessárias	25
3.3.2.3	Descrição da Funcionalidade	25
3.3.2.4	Saídas	25
3.3.3	Gerenciamento de Energia	26
3.3.3.1	Dependências	26

3.3.3.2	Premissas Necessárias	27
3.3.3.3	Descrição da Funcionalidade	27
3.3.3.4	Saídas	29
3.3.4	Sistema de Verificação da Integridade	29
3.3.4.1	Dependências	29
3.3.4.2	Premissas Necessárias	30
3.3.4.3	Descrição da Funcionalidade	30
3.3.4.4	Saídas	30
3.4	Simulação do sistema	30
4	Desenvolvimento e testes	33
4.1	Análise das Funcionalides	33
4.1.1	Atuação	33
4.1.2	Planejamento de Movimento	33
4.1.3	Gerenciamento de Energia	33
4.1.4	Checagem da Integridade do Sistema	33
4.2	Soluções Mecatrônicas para o sistema robótico	33
4.3	Estudo da Movimentação	33
4.4	Simulação	33
4.5	Testes de Movimentação Física	33
4.6	Integração com os subsistemas	33
4.7	Análise Preliminar	33
5	Conclusão	34
5.1	Considerações finais	34
A	QFD	35
B	Diagramas mecânicos	40
C	Diagramas eletro-eletrônicos	55
D	Logbook	56
E	Lista de componentes	79
	Referências	82

Lista de Tabelas

3.1 Especificações Motor Robotis <i>Dynamixel MX-28R</i>	15
3.2 Especificações Motor Robotis <i>Dynamixel MX-106R</i>	16

Lista de Figuras

1.1	Instalação típica de uma linha de transmissão	3
1.2	Inspeção em linhas de transmissão por veículos aéreos tripulados.	4
2.1	Parâmetros de Denavit-Hartenberg	7
2.2	Braço planar do tipo RR	9
2.3	Diagrama de funcionamento de um processo de simulação	11
3.1	Motor Robotis <i>Dynamixel MX-28R</i>	16
3.2	Motor Robotis <i>Dynamixel MX-106R</i>	16
3.3	Esquema das saídas do HUB.	20
3.4	Esquema das conexões do HUB para os motores.	21
3.5	Esquema HUB2.	21
3.6	Esquema HUB2.	21
3.7	Esquema HUB2.	22
3.8	Esquema HUB2.	22
3.9	Fluxograma de funcionamento da funcionalidade de Planejamento de Movimento	24
3.10	Fluxograma da funcionalidade Atuação	26
3.11	Fluxograma de funcionamento da funcionalidade de Gerenciamento de Energia	27
3.12	Fluxograma da rotina para checagem do sistema	31
3.13	Simulação do <i>ELIR</i> no <i>Gazebo</i>	32

Lista de Siglas

THEOPRAX
WWW World Wide Web

Lista de Simbolos

Introdução

O Brasil apresenta uma matriz energética diferente da do resto do mundo, onde as fontes renováveis representam uma grande parte da geração da energia. Segundo a (??), em 2016, a matriz energética mundial contava com somente 14,1% da matriz energética constituída por fontes renováveis, enquanto o brasil já apresentava 82% da sua matriz vindas de fontes renováveis, onde a geração hidrelétrica corresponde a 70% dessa geração.

A expectativa é de que a energia hidrelétrica continue sendo cada vez mais utilizada no país, devido ao crescimento previsto da demanda energética brasileira, onde segundo o (??) o consumo atual é de 405 TWh e a demanda esperada em 2030 é de 950 e 1.250 TWh/ano (??). Mesmo com a grande participação da geração hidrelétrica , somente 23% dos 260 GW totais de potencial hidrelétrico são aproveitados (??).

A concentração de demanda energética no Brasil está concentrada principalmente na região Sudeste devido a densidade populacional e elevada industrialização, isso faz com que dois terços do total da capacidade instalada estejam localizadas na Bacia do Rio Paraná que é a bacia mais próxima da região, enquanto as bacias com potencial menos aproveitado são as localizadas no norte e nordeste do país (??).

Com desenvolvimento do país é esperado um aumento na demanda de energia elétrica e consequentemente um aumento na geração de energia hidrelétrica, isso faz com que seja esperado um crescimento considerável na quantidade das linhas de transmissão, de acordo com (??), em setembro de 2018 o sistema elétrico brasileiro já atingiu 144.828 km de linhas de transmissão. Esse aumento na quantidade de linhas tende a ser amplificado pela tendência à exploração da geração de energia na região Norte, assim sendo necessário a construção de novas linhas para distribuir essa energia para as outras partes do País.

Quanto mais linhas de transmissão e maiores distâncias entre os centros geradores, maiores tendem a ser as perdas. Isso faz com que seja necessária um controle da qualidade dessa transmissão, o que se dá por meio de inspeções. A estrutura já existente apresenta precariedade em alguns aspectos, onde segundo (??) “no Brasil, há uma quantidade considerável de linhas de transmissão que já ultrapassou os 40 anos de idade. Com o envelhecimento das linhas de transmissão, a manutenção preventiva é um fator de extrema relevância para garantir o perfeito funcionamento dos sistemas.” A necessidade da constante manutenção e a alta periculosidade que os operadores são expostos faz com novas alternativas e tecnologias sejam aplicadas para a manutenção, o uso de Drones pilotados

remotamente, com câmeras e sensores já é uma realidade em alguns países do mundo. O desenvolvimento de um robô próprio para inspeção de linha configura uma dessas novas alternativas, e possibilita uma expansão dos horizontes para as tecnologias aplicadas.

1.1 *Objetivos*

O objetivo do trabalho é implementar o sistema de movimentação do robô ELIR (*Electrical Line Inspection Robot*). Onde esse sistema é complementar aos outros existentes no robô, onde o conjunto dessas soluções busca fundamentar a implementação de uma Inspeção autônoma da linha.

1.1.1 *Objetivos Específicos*

Para o desenvolvimento do sistema é necessário realizar o estudo da movimentação, gestão de energia, controle e elaboração de trajetória para sistemas robóticos. A operação na linha faz com que seja necessário a gestão de energia do robô, assim como a integração com os outros subsistemas já desenvolvidos. De forma a garantir a operação na linha, os dispositivos e ferramentas utilizadas devem estar integradas no ROS (*Robot Operating System*), onde é necessário também a integração com outros pacotes já desenvolvidos para o Robô.

1.2 *Justificativa*

Tendo em vista a crescente demanda de energia elétrica do país bem como a previsão , a necessidade de um processo confiável de transmissão se torna amplamente necessário, afinal, diversas unidades consumidoras são alimentadas diariamente, além de instalações que exercem atividades críticas, como hospitais. As unidades geradoras de energia elétrica se encontram em regiões distantes de seus consumidores finais, portanto se faz necessário a utilização de linhas de transmissão de energia elétrica.

Uma linha de transmissão é uma linha composta por cabos condutores de energia elétrica, utilizada para a transmissão de energia em alta tensão, saindo da origem geradora e indo até às cargas consumidoras.

A garantia da distribuição em condições favoráveis se dá pela confiabilidade das linhas de transmissão e os procedimentos de manutenção aplicados à elas, para isso, é

realizada constantemente a rotina de inspeção nas linhas. A rotina de inspeção, se dá através da análise da integridade da estrutura das torres, a condição em que se encontram os isoladoras e as conexões das linhas de transmissão, uma vez que o tempo e a exposição a umidade e ao sol, além de diversos eventos climáticos, fazem com diversas falhas referentes ao desgaste do material venham a aparecer.

Estas análises têm como principal objetivo a detecção de eventuais pontos de ruptura. Outro meio para a localização dos eventuais pontos de ruptura se dá pelo uso de câmeras térmicas, onde existe o aumento da temperatura pontual devido à elevação na resistência elétrica.



Figura 1.1: Instalação típica de uma linha de transmissão

Fonte: (??)

Segundo (??), as rotinas de inspeção de linhas de transmissão se dão principalmente por dois métodos: inspeções por aeronaves e inspeções terrestres. A inspeção realizada por aeronaves, se dá tipicamente com o uso de helicópteros, que executam voos em baixa altitude, extremamente próximos das linhas de transmissão.

Em alguns casos as condições climáticas podem dificultar o procedimento de inspeção e controle da aeronave, além do risco inerente da atividade para os tripulantes, principalmente devido ao fato de que as aeronaves tipicamente operam na região de “homem-morto”, uma zona de altura que representa perigo para os operadores a bordo das aeronaves em caso de uma queda.

A inspeção por vias terrestres possui uma grande dificuldade devido à dependência do terreno do local, o qual pode ser de difícil acesso devido às características geográficas. Diversos fatores tornam a inspeção de linhas de transmissão um procedimento não só perigoso, mas também altamente custoso.

Segundo (??) as principais desvantagens dos meios convencionais de inspeção de li-



Figura 1.2: Inspeção em linhas de transmissão por veículos aéreos tripulados.

Fonte: (??)

nhas de transmissão são os riscos de acidentes, devido a periculosidade do procedimento de inspeção; o alto custo, uma vez que é necessário a locação e deslocamento de equipamentos específicos para o transporte e inspeção das linhas de transmissão; a alta dependência das condições climáticas e geográficas, uma vez que se torna muito difícil realizar rotinas de inspeção em tempos chuvosos ou em locais de difícil acesso.

Outra grande desvantagem dos procedimentos de inspeção definida por (??) é justamente a necessidade de uma mão de obra qualificada para realização destes procedimentos. Se estes procedimentos de inspeção de linha de transmissão fossem realizados em linhas desenergizadas, o processo seria bem mais simples e rápido, porém existem diversos problemas atrelados ao fato de que existem inúmeras cargas consumidoras que necessitam da energia elétrica gerada.

1.3 Organização do Trabalho de Conclusão do Curso

O documento está organizado em cinco capítulos, seguindo a seguinte estrutura:

Capítulo 1 - Introdução: Faz a contextualização do âmbito no qual a pesquisa proposta está inserida. Apresenta, portanto, a problemática, objetivos e como este projeto Theoprax de conclusão de curso está estruturado

Capítulo 2 - Referencial Teórico: Apresenta a base teórica necessária para o desenvolvimento do projeto.

Capítulo 3 - Metodologia: Define o método adotado para o desenvolvimento do projeto, explicitando seu fluxo de atividades e premissas necessárias para aplicar a metodologia.

Capítulo 4 - Desenvolvimento e testes: Exibe os resultados obtidos durante o desenvolvimento do projeto, apresentando-os em testes unitários e integrados, assim como as datas de quando foram realizado, e os responsáveis pela execução.

Capítulo 5 - Conclusão: Apresenta as conclusões, contribuições e algumas sugestões de atividades de pesquisa a serem desenvolvidas futuramente.

Fundamentação Teória

O termo robô vem da palavra tcheca robota que tem como uma das possíveis traduções “trabalhador forçado” e ganhou o significado atual após o escritor tcheco Karel Capek (1890 - 1938), na sua obra de ficção científica “R.U.R. Rossumovi Univerzální Roboti”, associar o termo às máquinas criadas pelo personagem principal para servi-lo. Mas a ideia de algo que desenvolva atividades de maneira autônoma é apresentada ao mundo muito tempo antes. Aristóteles em (??) diz: “Se cada instrumento pudesse realizar sozinho a sua tarefa, obedecendo ou antecipando a nossa vontade, [...] os feitores não precisariam de servos, nem os senhores de escravos.”

Diversas obras da ficção retratam diferentes tipos de robôs criados de forma a reproduzir comportamentos semelhantes aos de um ser humano. Com o passar do tempo, juntamente com o avanço tecnológico nas áreas da eletrônica, mecânica e informática, a construção dessas máquinas se tornou possível. A indústria observou nos robôs, o potencial para automatizar e otimizar as linhas de processo, onde atividades que pudessem demandar mais tempo se fossem executadas por seres humanos, seriam executadas de forma muito mais rápida e precisa com a utilização de máquinas programadas e autônomas, aumentando a produção.

A (??) define um robô como “mecanismo programável atuado em dois ou mais eixos com um grau de autonomia, movendo-se dentro do seu ambiente, para executar tarefas pretendidas”. É resultado da integração de componentes como: Sensores; atuadores; unidade de controle; unidade de potência e manipulador mecânico. Sensores são os componentes que fornecem parâmetros sobre o ambiente em que o robô se encontra e sobre o comportamento do próprio sistema robótico. Já os atuadores são os dispositivos que movimentam as partes, quando convertem energia elétrica, hidráulica ou pneumática em mecânica. A energia necessária para o funcionamento dos atuadores é fornecida pela unidade de potência.

O gerenciamento dos parâmetros necessários para que o robô realize suas tarefas é de responsabilidade da unidade de controle. De onde também são emitidos os comandos para a movimentação. O manipulador mecânico é o conjunto de componentes estruturais do robô, elos ou links, conectados entre si por articulações comumente denominadas de juntas. Graus de liberdade, segundo (??) “É o número mínimo de variáveis independentes de posição que precisam ser especificadas para se definir inequivocamente a localização de todas as partes de um mecanismo”.

Nada é tão maravilhoso que não possa existir,
se admitido pelas leis da Natureza.

(Michael Faraday)

2.1 Cinemática

A cinemática é o ramo da física que descreve o movimento de um corpo, determinando características como posição, velocidade e aceleração. Na robótica, o estudo cinemático resulta em um conjunto de equações que caracterizam o movimento do robô, a complexidade da solução varia com a quantidade de graus de liberdade que esse robô tem. Em um manipulador mecânico composto por links que são conectados por juntas, cada conjunto link-junta caracteriza um grau de liberdade. Dessa maneira, um robô com n conjuntos link-junta tem n graus de liberdade, sendo o primeiro link a base de sustentação do robô no mundo e o último, onde está a seu end-effector.

2.1.1 Cinemática Direta

A cinemática direta é a solução para a movimentação de um robô com cálculo da posição e orientação do end-effector a partir de dadas posições das juntas. A notação de Denavit-Hartenberg é uma ferramenta utilizada para coordenar a descrição cinemática de sistemas mecânicos articulados com n graus de liberdade.

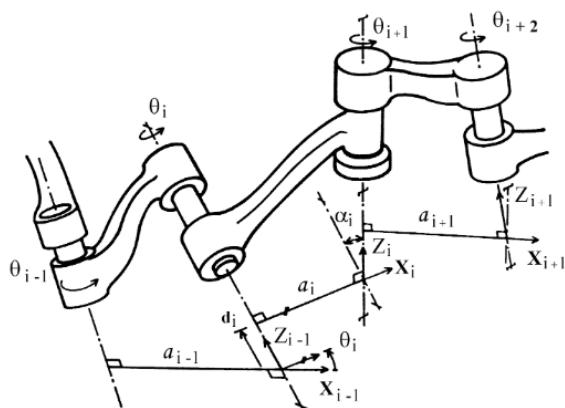


Figura 2.1: Parâmetros de Denavit-Hartenberg

Fonte: (??)

A figura mostra dois *links* ligados por uma junta de superfícies deslizantes uma sobre a outra. Um eixo de uma junta estabelece a conexão de dois *links*. Segundo (??), os eixos das juntas devem ter duas normais conectadas a eles, uma para cada um dos

links. Assim a posição relativa destes dois *links* conectados ($i - 1$ e i) é dada por d_i , que é a distância medida ao longo do eixo da junta entre suas normais. O ângulo de junta θ_i entre as normais é medido em um plano normal ao eixo da junta. Dessa forma, d_i e θ_i são a distância e o ângulo entre os *links* adjacentes. Determinam a posição relativa de *links* vizinhos.

Um *link* pode apenas ser conectado a dois outros *links* ($i - 1$ e $i + 1$). Assim, dois eixos de juntas são estabelecidos em ambos terminais de conexão. Os *links* mantém uma configuração fixa entre as juntas e podem ser caracterizados pelos parâmetros a_i e α_i . O parâmetro a_i é a menor distância medida ao longo da normal comum entre os eixos da junta, chamado de comprimento de *twist*, já o α_i é o ângulo de *twist*. Esses quatro parâmetros determinam a estrutura do *link*, parâmetros da junta e a posição relativa aos *links* vizinhos.

A representação de Denavit-Hartenberg tem como resultado uma matriz 4 x 4 representando cada sistema de coordenadas do *link* na junta em relação ao *link* anterior. Essa matriz é obtida através do produto das transformações: Translação de uma distância d_i ao longo do eixo Z_{i-1} para trazer os eixos X_{i-1} e X_i na coincidência; Rotação no eixo Z_{i-1} de um ângulo θ_i para alinhar os eixos X_{i-1} e X_i ; Translação ao longo do eixo X_i de uma distância a_i para trazer as duas origens na coincidência; Rotação do eixo X_i um ângulo α_i para trazer os dois sistemas de coordenadas na coincidência. Isso resulta na matriz de transformação homogênea ${}^{i-1}A_i$.

$${}^{i-1}A_i = T_{z,d}T_{z,\theta}T_{x,a}T_{z,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

2.1.2 Cinemática Inversa

Segundo (??) os robôs estão em um espaço onde o objeto a ser manipulado tem sua posição expressa no sistema de coordenadas do ambiente. Com o objetivo de controlar a posição e orientação do *end-effector* do robô, a solução da cinemática inversa é mais adequada. A cinemática inversa consiste em, partindo de uma posição e orientação desejada, calcula-se as posições das juntas para que o robô alcance esse objetivo, é o processo inverso da cinemática direta. Há de se observar que a cinemática inversa pode ou não ter solução, caso a posição de interesse esteja fora do espaço de trabalho do robô, não há

posições de juntas que execute a tarefa. Nos momentos em que a posição desejada pode ser alcançada, podem existir mais de uma solução. Um ponto importante na solução da cinemática inversa é, quando há mais de uma solução deve-se atentar para qual delas é a melhor opção, levando em consideração o ambiente em que o robô se encontra, principalmente os obstáculos à sua volta. A demanda energética para a execução dos possíveis movimentos e o esforço qual as juntas serão submetidas nesta ação, é crucial para o planejamento da movimentação do robô.

2.2 Modelagem Cinemática de um Braço Planar

O robô ELIR tem na sua estrutura, braços que se movimentam apenas em dois eixos, x e z , através da atuação de duas juntas, podendo assim ser modelado cinematicamente como um braço planar do tipo RR. A figura a seguir mostra um exemplo desse braço, RR por ter duas juntas rotativas, que se movimenta no plano $x - y$:

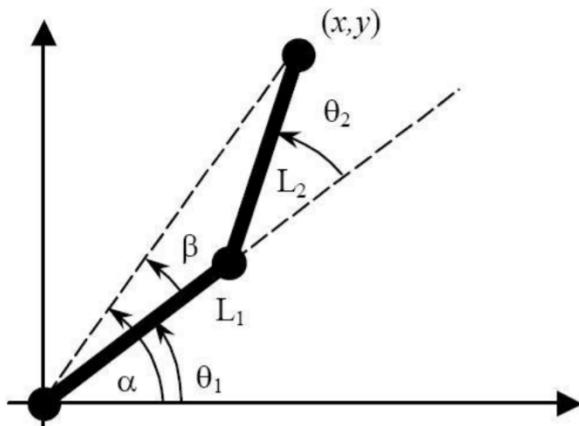


Figura 2.2: Braço planar do tipo RR

Fonte: (??)

Usando a análise da cinemática direta, consegue-se determinar a posição do *end-effector* com base nos ângulos θ_1 e θ_2 e nas dimensões L_1 e L_2 . Logo tem-se que:

$$x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \quad (2.2)$$

$$y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \quad (2.3)$$

Aplicando a lei dos cossenos ao triângulo formado pelo braço e pela linha entre a origem do braço e o seu *end-effector* obtém-se:

$$\theta_2 = \pm \arccos \frac{(x^2 + y^2 - (L_1)^2 - (L_2)^2)}{2L_1L_2} \quad (2.4)$$

Para determinar o θ_1 considera-se a relação trigonométrica:

$$\tan(A - B) = \frac{\tan(A) - \tan(B)}{1 + \tan(A)\tan(B)} \quad (2.5)$$

e tomando:

$$\tan(\beta) = \frac{L_2 \sin \theta_2}{L_1 + L_2 \cos \theta_2} \quad (2.6)$$

tem-se que:

$$\theta_1 = \arctan \left[\frac{y(L_1 + L_2 \cos \theta_2) - x L_2 \sin \theta_2}{x(L_1 + L_2 \cos \theta_2) - y L_2 \sin \theta_2} \right] \quad (2.7)$$

Assim é possível fazer a solução da cinemática inversa para um braço planar RR.

2.3 Desenvolvimento de Robôs

Para o desenvolvimento de sistemas robóticos, é necessária a integração de vários dispositivos, assim sendo necessário utilizar ferramentas e tecnologias que poupem tempo no desenvolvimento, de forma a facilitar o processo de comunicação entre as diversas camadas de abstração. As camadas de abstração se referenciam ao alto e baixo nível da máquina, onde baixo nível é uma referência para aplicações mais simples, que estão mais próximas da linguagem da máquina, como por exemplo aplicação de comunicação somente via *bytes*. Um exemplo de um elemento que está numa camada de abstração de alto nível é uma Interface Homem-Máquina , onde o usuário consegue interagir com a máquina diretamente, sem ter que necessariamente entender o seu funcionamento interno.

2.3.1 Framework

Em ambientes computacionais, a utilização de ferramentas para realização de atividades e desenvolvimento de soluções é de extrema importância. Estas ferramentas podem ser softwares específicos para execução de uma determinada atividade ou *frameworks*. Segundo (??) “*Frameworks* são estruturas de classes que constituem implementações incompletas que, estendidas, permitem produzir diferentes artefatos de software”. Os *frameworks* em geral permitem o desenvolvimento de soluções computacionais baseadas em determinadas funcionalidades, seguindo uma estrutura definida pelo *framework*. De acordo com (??) os *frameworks* definem uma arquitetura para um conjunto de subsistemas, dando os construtores necessários para a sua criação. A principal característica de um *framework* é a sua capacidade de reutilização, afinal a sua utilização permite que diversos conjuntos de produtos possam ser gerados partindo de uma única estrutura que possua os conceitos mais gerais. Segundo (??) *frameworks* podem ser classificados em dois tipos principais: *Frameworks* de Aplicações Orientado a Objetos e *Frameworks* de Componente.

Os *frameworks* orientados a objetos geram famílias de aplicações orientadas a objetos e seus pontos de extensão são definidos como classes abstratas ou interfaces, onde se estendem por cada instância da família de aplicações. Para *frameworks* de componentes, o suporte é previsto para componentes que sigam um determinado modelo, possibilitando que as instâncias destes componentes sejam acopladas ao *framework*. Também são estabelecidas as condições necessárias para que um componente seja executado, regulando a sua interação entre as instâncias de outros componentes. Os *frameworks* utilizados para robótica, são extremamente importantes, pois o uso de suas ferramentas possibilita o desenvolvimento e criação das soluções computacionais e códigos necessários para cada funcionalidade de um robô, de forma que o funcionamento delas em conjunto seja otimizado pela natureza do *framework* de realizar a compatibilização entre as estruturas.

2.3.2 Simulação

Em sistemas complexos, onde diversas variáveis definem o seu funcionamento, e por consequência as suas respostas a determinados estímulos, torna-se extremamente difícil e irresponsável executar a sua fabricação antes de realizar uma validação prévia de seu funcionamento. Este tipo de procedimento de análise prévia do comportamento de um sistema é chamado de simulação. De acordo com (ção_{de}simulação + +IMPL_{ção}_{de}simulação + +YEAR_{ção}_{de}simulação + +IMPL,)\Simulação refere-se a uma ampla coleção de métodos aplicados

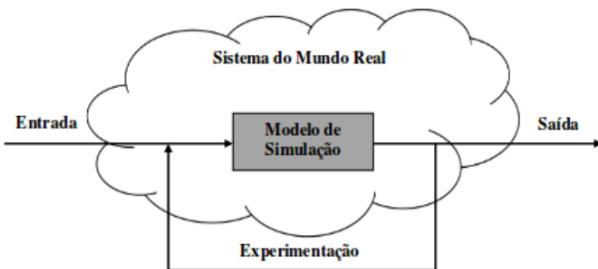


Figura 2.3: Diagrama de funcionamento de um processo de simulação

Fonte: (ção_{de}simulação + +IMPL_{ção}_{de}simulação + +YEAR_{ção}_{de}simulação + +IMPL,)

Em sistemas robóticos, uma ferramenta extremamente útil e bastante utilizada, é a simulação. Segundo (ção+ +IMPL_{ção}+ +YEAR_{ção}+ +IMPL,) “Quando se trabalha com robótica, o uso de uma simulação é de importância significante. Por um lado, ela permite a validação de diferentes alternativas durante o design do sistema robótico, levando assim, a melhores decisões e preservação de custos. Por outro lado, auxilia o processo de desenvolvimento de *software*, disponibilizando uma reposição para robôs que não estejam em mãos”. Através do uso de *softwares* de simulação é possível criar uma representação computacional não só o modelo físico de um robô, mas também os parâmetros referentes à objetos do ambiente no qual o mesmo será posto em funcionamento. A avaliação prévia da execução das tarefas e do funcionamento do robô, permite a observação do compor-

tamento do sistema em determinadas situações, facilitando assim, a tomada de decisões mais efetivas no processo de desenvolvimento do protótipo real.

2.3.3 Odometria

A odometria consiste no cálculo para estimar a mudança de posição do robô no tempo, onde isso pode se dar por meio de diversos dispositivos que possibilitem o cálculo de deslocamento. Onde segundo (??), "Odometria - a medição da distância - é um método fundamental usado por robôs para navegação". A medição de tempo é fácil utilizando o *clock* interno do computador embutido. Medir velocidade é mais difícil: em alguns robôs educacionais utilizam codificadores são usados para contar as rotações da rodas, enquanto em outros a velocidade é estimada das propriedades dos motores. No caso da análise de deslocamento do robô na linha por meio de roldanas, o movimento é caracterizado como linear, já que o deslocamento ocorre em somente uma direção, analogamente a odometria utilizada é a linear, onde o deslocamento pode ser calculado simplesmente pela equação 2.8 onde s representa o espaço caminhando, v a velocidade e t o tempo.

$$s = v * t \quad (2.8)$$

Utilizando o medidor de tempo interno do computador embutido nos sistemas robóticos, pode se calcular a variação de espaço para um tempo muito pequeno, onde esses pequenos incrementos são somados ou subtraídos para encontrar o deslocamento do robô. A velocidade de deslocamento das roldanas pode ser encontrada utilizando a equação 2.9 com as informações do raio da roldana r e a sua velocidade de giro w em radianos por segundo. A informação da velocidade de giro da roldana geralmente é extraída dos servomotores utilizados para tração.

$$v = 2\pi * r * w \quad (2.9)$$

O cálculo da odometria por meio da velocidade das rodas é denominado no âmbito da robótica como odometria de roda, *wheel odometry* em inglês, porém, outras técnicas são utilizadas, já que existem diversos tipos de deslocamento diferentes. Outro tipo aplicação muito encontrada é a odometria visual, que segundo (??) "Odometria Visual (OV) é o processo de estimação do deslocamento de um agente (ex: veículo, humano e robô) utilizando a entrada de uma ou múltiplas câmeras conectadas a ele". Os domínios da aplicação incluem robótica, realidade aumentada, automotiva e 'computadores vestíveis'.

2.3.4 Gestão de Energia

O conceito de gestão de energia se dá pela forma como a energia elétrica é utilizada em um sistema composto de diversos dispositivos elétricos e eletrônicos. Para sistemas robóticos, este conceito representa um fator importante para garantir uma operação autônoma de qualidade. Os robôs quando nesse tipo de operação, geralmente não dispõem de uma fonte de energia constante, e portanto, são geralmente alimentados por baterias e tendo interação por meio de conexões sem fio. O uso de diversos dispositivos eletrônicos de baixo consumo energético, como sensores e interfaces microcontroladas, podem não se mostrar um problema para um curto período de operação, porém, para maximizar o tempo da atividade exercida pelo robô, é necessário encontrar uma forma eficiente de gerir a operação dos dispositivos conectados na rede de alimentação. Segundo (??) “Gestão de energia é um conceito importante em redes de sensores, porque uma estrutura de energia cabeadas geralmente não está disponível e um conceito óbvio é utilizar a energia disponível da bateria de forma eficiente”. Quanto mais atividades diferentes o robô desempenha maior será a demanda de energia entre os dispositivos interconectados, isso faz com que seja necessário que os desenvolvedores busquem uma forma de otimizar o custo de energia individual das atividades e do fluxo de operação como um todo. Os diversos dispositivos utilizados em sistemas robóticos fazem com que o mesmo se utilize de diferentes níveis de tensão e corrente, já que comumente, os dispositivos utilizados são comerciais, e devido às diferenças das suas características e parâmetros, definidos por empresas diferentes, responsáveis pela produção e fabricação das ferramentas, é necessário que a gestão de energia leve em consideração a compatibilidade entre diferentes dispositivos.

2.3.5 Conceito de segurança e Integridade

Em diversas áreas, é comum a verificação das condições antes da execução de atividades, a aviação é um grande exemplo de uso desse conceito. Neste seguimento, o *checklist* é utilizado toda vez antes de um avião decolar, assim é possível verificar se os sistemas vitais para o voo estão em ordem. O principal objetivo dessa ação é identificar os riscos que existem para o cumprimento da atividade. Para que um dispositivo robótico execute as tarefas para as quais ele foi desenvolvido, deve-se verificar se os seus sistemas, como um todo, e os componentes individualmente, estão em condições de funcionamento, garantindo assim a integridade do sistema como um todo. Essa análise deve ser feita levando em conta a importância de cada sistema e de cada componente desses sistemas, a fim de aumentar a capacidade de operação em condições adversas do robô. Podem existir sistemas que, mesmo quando não estão operando adequadamente, não comprometem a execução da missão do robô.

2.3.6 Comunicação em sistemas robóticos

Dispositivos eletrônicos são capazes de realizar transmissão de dados, afinal, a interconexão entre eles é de extrema importância em sistemas em que existam diversos dispositivos responsáveis por funções distintas. Dispositivos que se comunicam entre si, são capazes de criar uma rede em todo o sistema, permitindo um aumento na confiabilidade das funções do sistema, através da troca de informações de parâmetros que venham a ser importantes para o funcionamento do sistema como um todo. Para que os dispositivos possam se comunicar entre si, os mesmos adotam o que se chama de protocolos de comunicação. Protocolos de comunicação são arquiteturas que estabelecem a troca de dados entre dispositivos eletrônicos. Os dispositivos comerciais possuem diferentes tipos de protocolos de comunicação e por isso, torna-se extremamente importante se atentar a qual protocolo utilizar durante a conceituação de um projeto que se tenha a necessidade da interconexão de dispositivos. Uma das formas mais comuns de se realizar a transmissão de dados entre dispositivos embarcados é a comunicação serial. (??) define a comunicação serial como um envio de *bits* de forma serial, similar a uma fila. Possuindo dois canais principais: o canal *TX* para envio e o canal *RX* para recebimento. Dentro desse processo de comunicação alguns parâmetros devem ser levados em conta, como a taxa de transmissão de dados (*BaudRate*); bits de paridade, para assegurar que o número de bits no campo de dados é par ou ímpar; bits de parada para indicar o início ou fim de uma comunicação. Outro meio de comunicação muito utilizado é o USB (*Universal Serial Bus*), criado com a intenção de tornar a comunicação serial mais simplificada e com uma taxa de transmissão muito mais elevada. Os cabos conectores USB possuem geralmente quatro fios condutores, sendo dois deles para alimentação e dois outros cabos de dados. Os cabos de dados são nomeados como D+ e D-, onde a comunicação entre os dispositivos se dá pela variação de tensão entre estes dois sinais. Dentro de ampla complexidade como um robô, onde diversos dispositivos necessitam estar trocando informações, a utilização de protocolos de comunicação serial se tornam extremamente importantes para a garantia da confiabilidade na execução de tarefas e operações.

Metodologia

Esta seção destaca o que é necessário para construção do projeto, contendo a lista de materiais, especificação dos componentes, funcionalidades, modelo esquemático de comunicação e de alimentação.

3.1 Especificação dos componentes

Serão detalhados os componentes utilizados para confecção do protótipo, sendo eles físicos ou digitais.

3.1.1 Lista de componentes

3.1.1.1 Servomotores

Os servomotores são responsáveis pelo atuação do robô, realizando os movimentos das juntas dos braços e das garras, além de atuarem as rodas que deslocam o robô na linha. São utilizados os servomotores da Robotis, *Dynamixel MX-106R* e *MX-28*. Esses motores foram escolhidos pois apresentam drivers prontos para o *ROS*, que possibilitam uma integração mais fácil com as ferramentas de controle, apresentando bom torque, peso reduzido e fácil integração para controle conjunto.

Robotis Dynamixel MX-28R

Peso (g)	153
Dimensões (mm)	40.2 x 65.1 x 46
Torque (N.m)	8.0 (em 11.1V), 8.4 (em 12V) e 10.0 (em 14.4V)
Temperatura de operação (°C)	-5 até +80
Tensão de operação (V)	10 até 14.8 (Tensão recomendada: 12V)
Baud rate	8000bps até 4.5Mbps
Protocolo de comunicação	RS485
Resolução	0.088º
ID	254 ID (0 até 253)
Feedback	Posição, temperatura, carga, tensão de alimentação, etc.

Tabela 3.1: Especificações Motor Robotis *Dynamixel MX-28R*

Figura 3.1: Motor Robotis *Dynamixel MX-28R*.

Fonte: (??)

Figura 3.2: Motor Robotis *Dynamixel MX-106R*

Fonte: (??)

Robotis Dynamixel MX-28R

Peso (g)	153
Dimensões (mm)	40.2 x 65.1 x 46
Torque (N.m)	8.0 (em 11.1V), 8.4 (em 12V) e 10.0 (em 14.4V)
Temperatura de operação (°C)	-5 até +80
Tensão de operação (V)	10 até 14.8 (Tensão recomendada: 12V)
Baud rate	8000bps até 4.5Mbps
Protocolo de comunicação	RS485
Resolução	0.088°
ID	254 ID (0 até 253)
Feedback	Posição, temperatura, carga, tensão de alimentação, etc.

Tabela 3.2: Especificações Motor Robotis *Dynamixel MX-106R*3.1.1.2 *Placa de Gerenciamento de Energia (Gerenciamento de Energia)*

A placa de gerenciamento de energia é responsável pela distribuição de corrente e de tensão para todos os componentes elétricos e eletrônicos do robô, além de monitorar os níveis de tensão e corrente demandados durante a operação.

Além de realizar o monitoramento do consumo em cada porta individualmente, a placa possui um sistema de proteção, cortando a alimentação em casos de surto de corrente. A placa funciona através da alimentação de 14.4 Volts provenientes da placa multiplexadora, responsável por transmitir a carga de duas baterias *Li-Ion* de 14.4 Volts e 6Ah.

Na placa de *Gerenciamento de Energia* existem conversores DC/DC responsáveis por fazer a conversão dos níveis de 14.4 Volts para 12 Volts em cada porta de saída da placa. As 7 portas de saída possuem sensores de tensão e corrente individuais, feitos com amplificadores de instrumentação INA226. Existem duas portas de saída que disponibilizam tensões em valores menores, de 5 Volts.

O monitoramento dos níveis de tensão e corrente se dá principalmente pela inteligência do sistema, um firmware embarcado em um microcontrolador Atmega32u4, responsável por fazer as leituras dos parâmetros em cada uma das portas, verificando se os seus níveis estão de acordo com os limites configurados, cortando a alimentação via relés digitais caso esses valores sejam ultrapassados.

3.1.1.3 ROS (*Robot Operating System*)

Framework, no ambiente de programação, é um espaço onde compatibiliza códigos comuns a fim de otimizar o trabalho e tempo, muito utilizado na área de desenvolvimento. A abstração de hardware, códigos de baixo nível, drivers de sensores, simuladores, etc - são as grandes vantagens de se utilizar essa aplicação, podendo assim, fazer com que o desenvolvedor foque somente nas soluções de problemas específicos do seu projeto.

Foi utilizado durante todo o desenvolvimento do *ELIR* o *framework ROS*, já que reúne uma série de ferramentas importantes para o desenvolvimento de um robô. “O Sistema Operacional de Robótica é um flexível framework para escrita de softwares para robótica. É uma coleção de ferramentas, bibliotecas e convenções que serve para simplificar a tarefa de criar complexos e robustos comportamentos de robôs diante a uma variedade de plataformas (??).”

A grosso modo, cada câmera, motor ou periférico ligado ao *ROS*, estão associados ao um nó. A comunicação entre os nós se dá através de tópicos ou de serviços, a diferença é que o primeiro a informação é trocada de forma constante com certo intervalo de tempo e o segundo somente quando solicitado.

Assim é feita toda a comunicação e interligação entre os periféricos no *ROS*, forma simples de integração dos componentes.

3.1.1.4 MoveIt!

Durante a inspeção de linha, é necessário que o robô realize a ultrapassem dos diferentes tipos de obstáculos que existem nas linhas de transmissão. O *MoveIt!* é um ferramenta que funciona de forma integrada com o *ROS*, apresentando funcionalidades de planejamento de movimento, percepção 3D, controle, manipulação e cinemática inversa.

A cinemática é o estudo do movimento, no âmbito da robótica designa o estudo do controle da posição do robô no espaço. Esse controle pode representar do robô como um todo, sua posição geográfica, ou controle de alguma parte sua em específico, como seu braço e a posição relativa desse braço e o robô. A cinemática direta é o cálculo onde se encontra a posição do robô para determinado valor de velocidade ou posição de suas juntas. Analogamente, na cinemática inversa, se encontra os valores de velocidade ou posição das juntas para uma posição no espaço, onde essa posição é denominada *end-effector*, geralmente sendo definido como a parte do robô que interage com o mundo, como por exemplo a garra no caso de manipuladores. O cálculo da cinemática inversa envolve equações complexas e retorna diferentes soluções, assim sendo necessário encontrar a solução que melhor atende às diretrizes do movimento, o *MoveIt!* já realiza esse cálculo e fornece uma trajetória otimizada baseada em parâmetros do usuário.

Outra das suas vantagens é utilizar o modelo *URDF* do robô. *URDF* é uma sigla para Unified Robot Description Format, e designa um arquivo com extensão *.urdf* e sintaxe em XML. É um dos tipos de modelos mais utilizados na robótica atual, sendo escolhido pois apresenta uma sintaxe simples e dinâmica, proporcionando conversões em outros formatos de forma fácil. Define o robô como um conjunto de partes, chamadas de *links* onde a união entre essas partes é uma junta. Onde cada *link* vai ter um *link* pai, que é determinado pela definição da junta, assim o modelo apresenta uma estrutura em árvore, onde todos os *links* vão ter um pai até chegar ao *link* da raiz, essa definição é importante pois a partir disso é feita a cinemática inversa do robô. Um erro no modelo *URDF* acarreta em uma mudança no comando que é mandado para o robô original.

Durante o desenvolvimento do projeto, foi possível realizar a integração do *MoveIt!* com o robô real, sendo possível realizar movimentos físicos utilizando a ferramenta de visualização para posicionamento de end-effector pelo usuário. Porém, ao tentar se enviar um comando com o valor de posição para o end-effector se mover, o programa falhou em encontrar soluções para a cinemática. Mesmo se utilizando os diversos solucionadores providos e enviando valores possíveis de se calcular, o programa sempre estava falhando em encontrar uma solução.

O *MoveIt!* é designado para funcionar com robôs de 6 graus de liberdade, onde cada grau de liberdade indica uma coordenada que o end-effector pode se mover e um

dos 3 eixos de referência (x,y,z) que ele pode girar. A grande quantidade de graus de liberdade faz com que os solucionadores utilizem formas de cálculos complexas, assim robôs que possuem menos que 6 graus de liberdade precisam ser compatibilizados, já que a solução leva em consideração todas as direções e giros. O Elir possui somente 2 graus de liberdade, e as soluções que antes funcionavam não estavam se utilizando especificamente da solução de cinemática inversa provida pelo *MoveIt!*. Assim, decidiu-se realizar o cálculo da cinemática inversa por meio de um código python, que utiliza a equação de cinemática inversa específica para o tipo de braço do robô e fornece os ângulos de junta necessários para o end-effector especificado.

Com o ângulo de junta em mãos, é feita a integração do robô com a ferramenta, de forma que o planejamento de movimento ocorre, só que com o software recebendo um ângulo desejado. Conhecendo os possíveis valores de end-effector, o que pode ser encontrado pela ferramenta de visualização, é possível realizar o movimento no robô enviando somente um comando de coordenadas.

Implementar o controle de movimento nessa plataforma possibilita uma série de implementações futuras que aumentam a autonomia do robô e robustez do sistema, como odometria, ferramentas de percepção e mapeamento.

3.1.1.5 Gazebo

O software *Gazebo* é software utilizado para simulação de robôs. Tem uma licença de uso livre e apresenta diversas formas de integração com o *ROS*, sendo o principal simulador utilizado em conjunto com essa plataforma, possibilitando a inserção de plugins como câmeras e sonares, que se comunicam com o *ROS* de forma fiel a dispositivos reais.

Nele é possível simular também o ambiente do robô, definindo parâmetros físicos como aceleração da gravidade e vento. Oferece suporte para a inserção de modelos 3D de softwares CAD, assim podendo ser inseridas diversas estruturas que já foram modeladas para outros propósitos no software.

3.1.1.6 Visual Studio Code

Para que todas as funcionalidades do robô sejam configuradas e desenvolvidas de forma correta a nível de software, é necessário o desenvolvimento de diversos códigos em diferentes linguagens para a configuração de aspectos específicos do projeto.

Foi utilizada durante o desenvolvimento do projeto a ferramenta *Visual Studio Code*. Trata-se de um editor de códigos open source desenvolvido pela *Microsoft* em 2015, sendo possível desenvolver códigos em diversas linguagens como C++, C, Python entre outros. Durante todo o desenvolvimento do *ELIR* a ferramenta fora utilizada para o desenvolvimento de arquivos nas extensões .py, .yaml, .launch e .urdf.

Por ter uma interface simples e amigável, o *VSCode* mostrou-se uma ferramenta extremamente útil para a escrita e desenvolvimento de códigos durante todas as fases do projeto.

3.1.1.7 PlatformIO

Na interface do existem diversas extensões que podem ser instaladas para adicionar novas funcionalidades na plataforma.

Uma das extensões utilizadas fora o *PlatformIO*, um ecossistema desenvolvido especificamente para o desenvolvimento de códigos e firmwares em plataformas microcontroladas, sendo extremamente versátil, tendo suporte para diversas plataformas como STM, MSP430, Arduino entre outras, tornando desnecessário o uso de uma IDE específica para se realizar a configuração e desenvolvimento de firmwares durante o projeto.

A necessidade de se utilizar essa extensão se deu principalmente pela necessidade de se embarcar o firmware de Gerenciamento de Energia na placa. O *PlatformIO* possui as funcionalidades de debug e gravação, sendo assim, todos os procedimentos necessários para atualizar o firmware são atendidos na extensão.

3.2 Diagramas elétricos do sistema de Movimentação

Devido à quantidade de motores presentes no robô e a forma com que eles estão distribuídos na estrutura, foram desenvolvidos dois modelos de hubs para a conexão dos motores na rede.

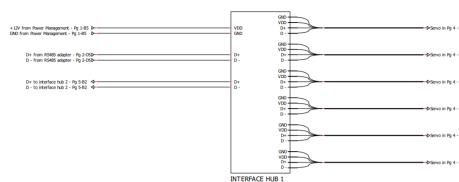


Figura 3.3: Esquema das saídas do HUB.

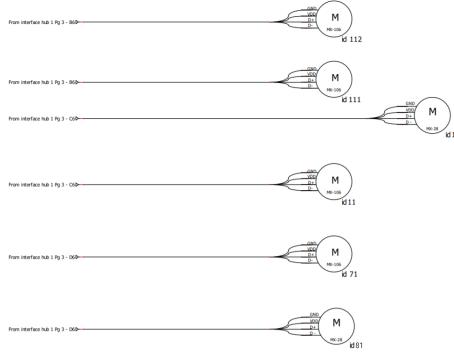


Figura 3.4: Esquema das conexões do HUB para os motores.

3.2.1 Esquemas eletrônicos

Nas unidades de tração do robô, os hubs contam com um conector de alimentação, um para a entrada de dados e 6 de saída para os motores.

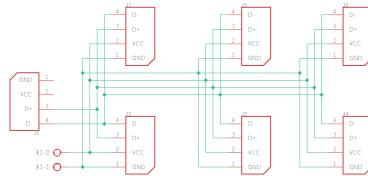


Figura 3.5: Esquema HUB2.

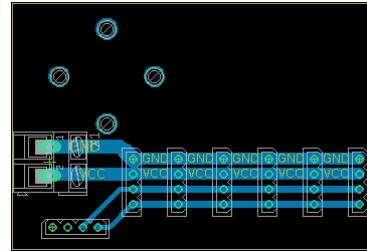


Figura 3.6: Esquema HUB2.

Já na unidade central, o hub além de conectar os seis motores ali presentes, também é responsável pela conexão dos hubs das unidades de tração e do conversor rs485 que está ligado à *NUC*.

3.3 Especificação das funcionalidades

3.3.1 Planejamento de Movimento

A funcionalidade de *Planejamento de Movimento* é responsável por realizar o planejamento da trajetória do Robô, utilizando o software *MoveIt!* que realiza o cálculo da cinemática inversa para encontrar a melhor forma de ultrapassar os obstáculos.

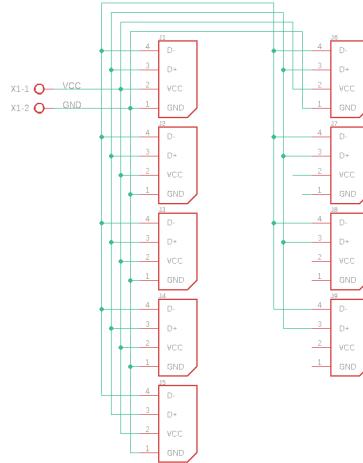


Figura 3.7: Esquema HUB2.

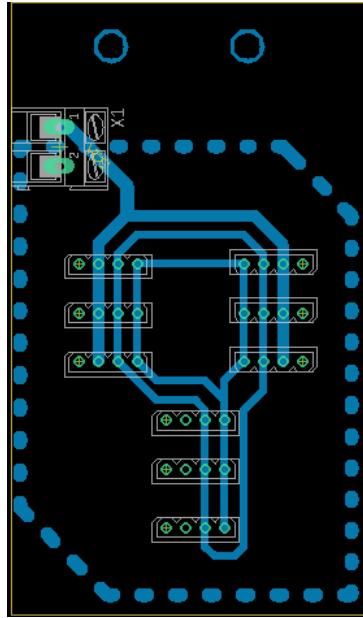


Figura 3.8: Esquema HUB2.

3.3.1.1 Dependências

O software *MoveIT!* pode utilizar o modelo matemático da cinemática inversa do robô ou um arquivo do tipo *URDF*. O nome *URDF* é uma sigla para *Unified Robot Description Format*, esse arquivo é uma especificação em XML utilizada para descrever robôs. Modelos em *URDF* apresentam uma simplicidade na descrição do robô, e para o caso do Robô *Elir*, utilizar o modelo *URDF* possibilitará uma aproximação fiel ao modelo real do robô, assim para o cálculo da cinemática inversa será utilizado o seu modelo *URDF* e não o seu modelo matemático.

3.3.1.2 Premissas Necessárias

Para o correto funcionamento dessa funcionalidade as seguintes premissas são necessárias:

- A configuração dos limites de giro das juntas do robô estarão compatíveis com os comandos enviados
- O modelo *URDF* do robô estará adequado com o modelo físico
- O pacote gerado pelo *MoveIt! Setup Assistant* estará configurado adequadamente

3.3.1.3 Descrição da Funcionalidade

A movimentação do robô na linha acontecerá por movimentos de translação e transposição de obstáculos. A translação na linha será feita por controladores de torque nas rodas do robô, enquanto a transposição do obstáculos utilizará o *MoveIT!*. Por meio da ferramenta *MoveIt! Setup Assistant*, se utiliza o modelo do robô para criar um pacote do *ROS* com os principais arquivos pelo *MoveIT!*. A configuração correta do *MoveIT!* possibilita que se utilizem as funções da sua biblioteca para o cálculo da trajetória, levando em consideração também obstáculos no caminho.

O *MoveIT!* fornece uma *user interface* que recebe o end-effector, a nomenclatura atribuída ao node feito em python que recebe o *end-effector* é `moveit_commander`. O *node* responsável por fazer a integração da user interface com os parâmetros recebidos pelo *ROS Parameter Server* com o *end-effector* para fazer os cálculos é denominado `move_group`. O *node move_group* também pode receber parâmetros como leituras dos sensores do robô e nuvens de pontos.

3.3.1.4 Saídas

Por meio da compatibilização do *MoveIt!* com o *ROS*, a saída dessa funcionalidade são os comandos de velocidade, esforço e posição para cada junta do robô.

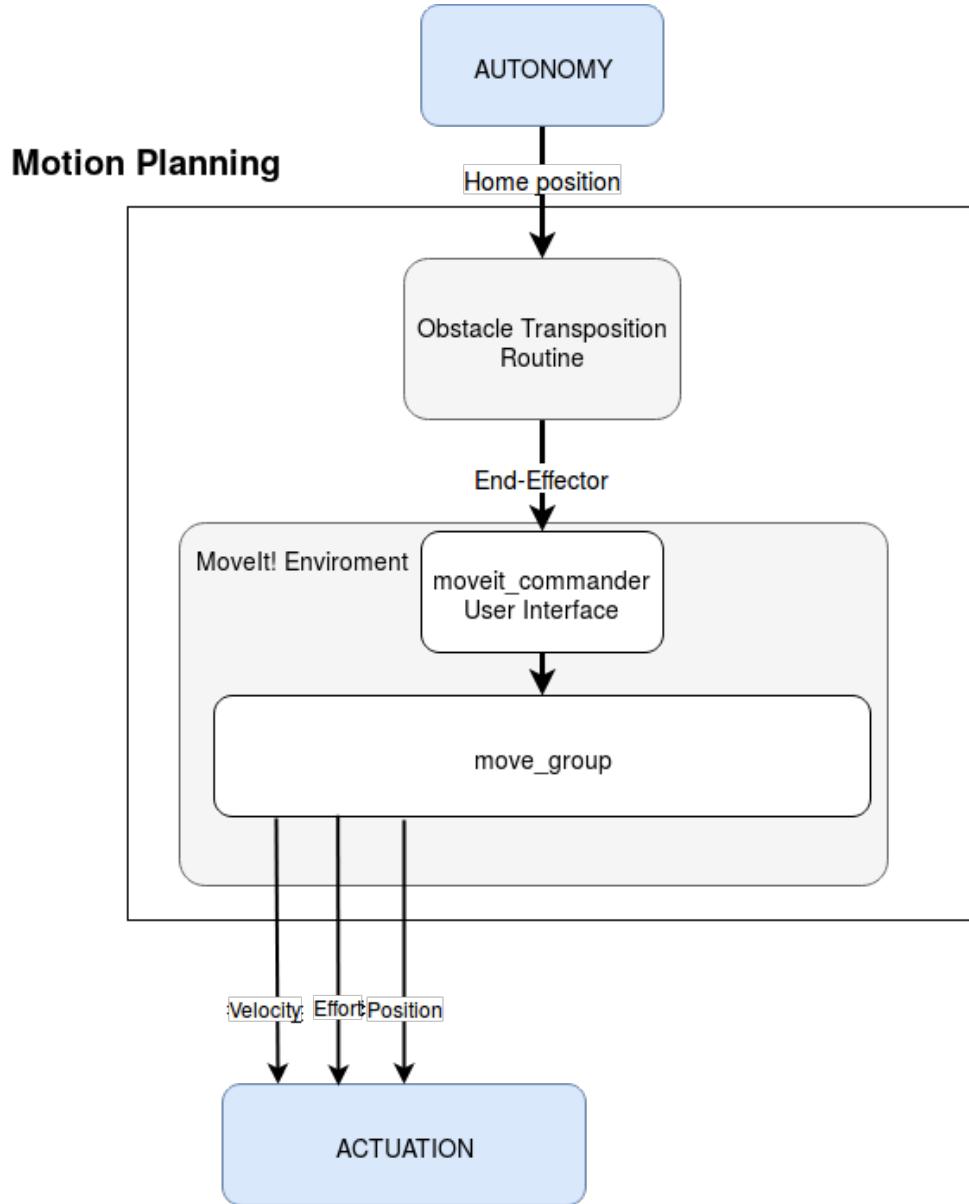


Figura 3.9: Fluxograma de funcionamento da funcionalidade de Planejamento de Movimento

Fonte: Própria

3.3.2 Atuação

A funcionalidade de Atuação tem como objetivo mover a estrutura física do robô, possibilitando o controle dos movimentos das juntas, garras e unidades de tração.

3.3.2.1 Dependências

Essa funcionalidade depende das funcionalidades de *Gerenciamento de Energia* e *Planejamento de Movimento*. O *Gerenciamento de Energia* será responsável por fazer

alimentação dos motores, possibilitando controlar a corrente máxima fornecida para cada grupo. A dependência em relação à funcionalidade de *Planejamento de Movimento* está atrelada principalmente com o software *MoveIt!*, que ao receber um *end-effector*, realiza o cálculo de trajetória e envia os comandos de velocidade, esforço e posição para os controladores das juntas, garras e unidades de tração.

3.3.2.2 Premissas Necessárias

Para o correto funcionamento desse módulo, devem ser consideradas as seguintes premissas:

- Os motores devem estar configurados de acordo com o padrão de ID determinado pela equipe, fazendo parte da mesma malha de controle;
- Os controladores das juntas, garras e unidades devem estar configurados de acordo com os comandos que serão recebidos pelo *MoveIt!*;
- Os 3 grupos de motores estarão em malhas de alimentação de 12V individuais.

3.3.2.3 Descrição da Funcionalidade

O *ROS* disponibiliza uma série de drivers para compatibilização dos motores dy-namixel, possibilitando a criação de controladores específicos no seu ambiente. Serão criados os controladores referentes as juntas e unidades de tração do robô. Os controladores receberão comandos de *velocity* e *position* do *MoveIt!* junto com os comandos para movimentar o robô na linha. Após os comandos serem recebidos pelos controladores, eles serão enviados para o *hardware* do robô, de acordo do padrão de comunicação dos motores, por meio de comunicação serial.

3.3.2.4 Saídas

A saída desta funcionalidade é o movimento da estrutura física do robô, que estará de acordo com o planejamento de trajetória do *MoveIt!* e com as instruções para operação na linha

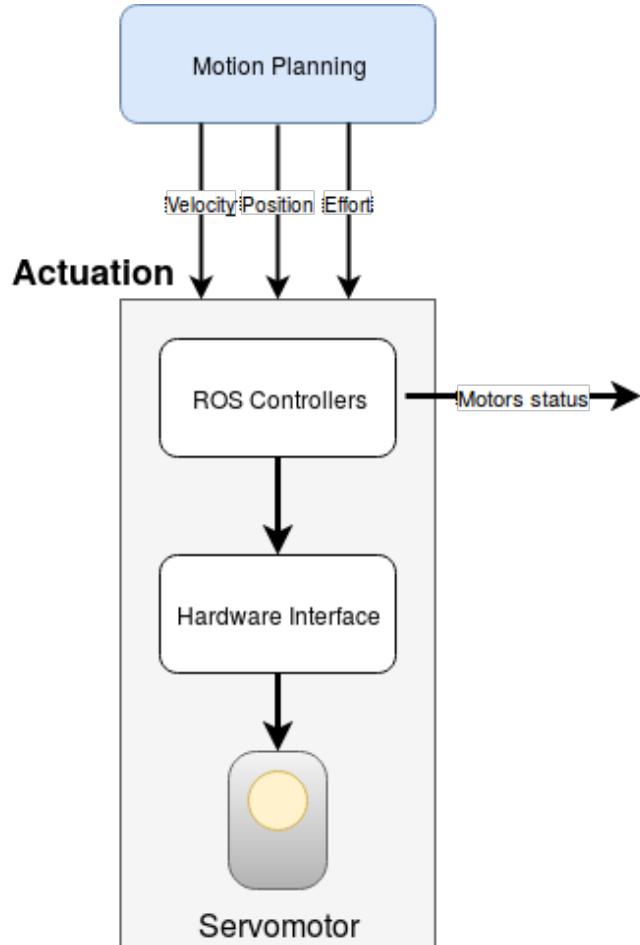


Figura 3.10: Fluxograma da funcionalidade Atuação

Fonte: Própria

3.3.3 Gerenciamento de Energia

A funcionalidade de *Gerenciamento de Energia* é responsável pelo gerenciamento de alimentação elétrica dos componentes elétricos e eletrônicos do robô, através da integração das funcionalidades de seu firmware no ambiente *ROS*.

3.3.3.1 Dependências

Essa funcionalidade depende da comunicação serial por meio da biblioteca `rosserial` para compatibilização e integração das funcionalidades de firmware no ambiente *ROS*. Operacionalização e customização do firmware embarcado no hardware de acordo com as necessidades do projeto e da alimentação fornecida pela placa multiplexadora, por meio de baterias Li-Ion NH2054 14.4 volts.

3.3.3.2 Premissas Necessárias

Para o correto funcionamento desse módulo de *Gerenciamento de Energia*, devem ser consideradas as seguintes premissas:

- A placa multiplexadora estará conectada diretamente ao módulo de *Gerenciamento de Energia*
- Todos os dispositivos estarão conectados nas suas respectivas entradas
- A placa deverá ser alimentada por 2 baterias de 14.4 Volts e 3 Amperes, totalizando um fornecimento de até 6 Amperes
- A placa estará conectada diretamente na NUC, por meio de uma USB

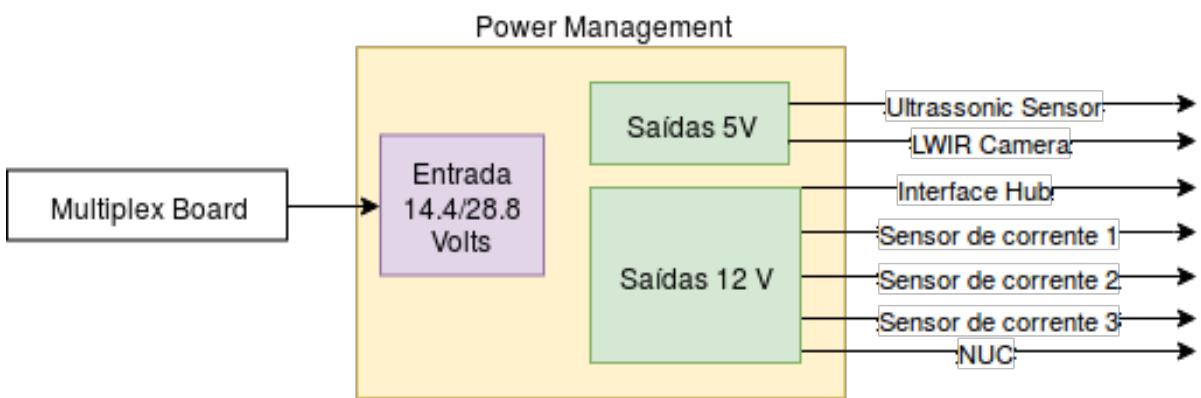


Figura 3.11: Fluxograma de funcionamento da funcionalidade de Gerenciamento de Energia

Fonte: Própria

3.3.3.3 Descrição da Funcionalidade

A funcionalidade *Gerenciamento de Energia* é responsável fornecer diversos recursos em sua totalidade. O hardware utilizado (placa *Zord*) possui um sensor de corrente e tensão para cada porta de saída, permitindo o monitoramento individual de cada uma das portas. O microcontrolador utilizado Atmega32U4 possui um firmware embarcado onde toda a compatibilização com o ambiente *ROS* é realizada, o que torna essencial o uso do pacote rosserial para o seu funcionamento. O firmware é responsável pela ativação dos relés digitais em caso de surtos de corrente para proteção dos dispositivos elétricos.

Os limites nos valores de corrente funcionam justamente para que o hardware interrompa a alimentação em um possível caso de surto de corrente. Todos os aspectos importantes para o funcionamento do sistema de gerenciamento de energia pode ser configurado tanto via *ROS*, por meio das configurações dos serviços, ou por meio do firmware,

modificando os parâmetros do tempo de duração dos picos de corrente. Os principais serviços e tópicos criados pela funcionalidade Gerenciamento de Energia no *ROS* são:

- *Tópicos*
 - *PowerOutput* Este tópico disponibiliza os valores de tensão e corrente de todas as portas da placa em tempo real.
 - *TakeStatus* Disponibiliza o estado de cada porta da placa, informando os eventos ocorridos e a porcentagem de corrente demandada durante a ocorrência do evento.
- *Serviços*
 - *GetCurrentLimitCommand* Este comando retorna o valor de corrente máxima de saída configurado para a porta escolhida
 - *SetCurrentLimitCommand* Este comando realiza a configuração do valor máximo de corrente de saída em uma determinada porta
 - *PowerOnOffCommand* Este comando realiza a ação de ativação ou desligamento de uma determinada porta.

A placa de Gerenciamento de energia irá receber a carga das baterias pela placa multiplexadora e irá realizar o controle de alimentação dos seguintes componentes:

- Grupos de servo motores
- Grupo de sensores de corrente
- NUC
- Interface HUB
- Câmera LWIR
- Sensor ultrassônico
- Phidgets
- STM Nucleo
- Módulo GPS

3.3.3.4 Saídas

A funcionalidade irá disponibilizar a energia para o robô e as seguintes estruturas no ambiente *ROS*:

- Tópicos com informações de tensão e corrente nas portas
- Tópico para aviso de sobre-corrente
- Tópico para informar disponibilidade da placa
- Serviços para ler e configurar limite de corrente das portas
- Serviço para ligar ou desligar energia em uma porta

3.3.4 Sistema de Verificação da Integridade

É a funcionalidade responsável por checar a integridade do sistema antes do início da missão, verificando os subsistemas e suas variáveis.

3.3.4.1 Dependências

A funcionalidade receberá informações dos seguintes componentes

- Sensor de Temperatura
- Servomotores
- Câmera IR
- Câmera Stéreo
- IMU
- Sensor de Proximidade
- Placa de Power Management
- Sonar
- Baterias

Todas as informações serão enviadas por meio do ambiente *ROS*, na forma de *Services* ou *Publishers*.

3.3.4.2 Premissas Necessárias

As premissas necessárias para o funcionamento dessa funcionalidade são:

- Os subsistemas do robô irão disponibilizar o seu status no ambiente *ROS* por meio de tópicos ou serviços
- A checagem fará parte do planejamento de missão

3.3.4.3 Descrição da Funcionalidade

A checagem da integridade do sistema é uma funcionalidade essencial para garantir o sucesso da missão e preservar a integridade do robô. O *ROS* facilita essa comunicação entre os subsistemas, possibilitando que seja criada uma rotina de checagem antes de cada missão.

Será disponibilizado no sistema uma rotina para iniciar a missão. Ao receber o comando para início de missão, os sistemas serão checados sequencialmente, utilizando estrutura de *Services* e *Publishers* do *ROS*. Caso algum sistema apresente falha, a missão não se iniciará e o erro será mostrado no *terminal* e registrado no arquivo de *log*. Se todos os sistemas estiverem em funcionamento, se iniciará a missão. O fluxograma da funcionalidade está ilustrado na figura [3.12](#).

3.3.4.4 Saídas

No início da rotina de inspeção, a funcionalidade será responsável por enviar o sinal inicia a missão. Caso todos os sistemas checados estejam funcionando, a inspeção ocorrerá normalmente, se algum sistema apresentar defeitos, o defeito será mostrado no *terminal*, registrado em *log* e a missão será abortada.

3.4 Simulação do sistema

A simulação de sistemas robóticos consistem em um dos pilares para o desenvolvimento de projetos. Com a simulação é possível testar aplicações sem a necessidade de adquirir componentes, os membros da equipe de projeto conseguem trabalhar de forma simultânea no robô enquanto o protótipo físico fica reservado para testes específicos.

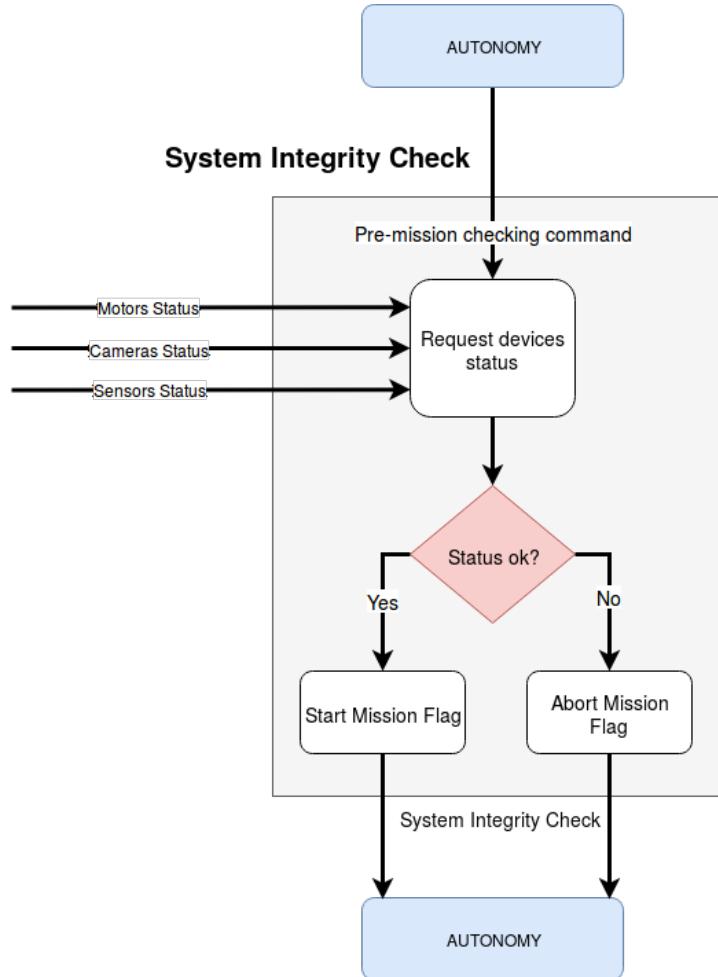


Figura 3.12: Fluxograma da rotina para checagem do sistema

Fonte: Própria

O *ROS* oferece ferramentas de visualização já integradas no seu sistema, o RVIZ, que possibilita o usuário visualizar os modelos do robô e também administrar plugins, como de mapeamento e planejamento de movimento, que é o caso do *MoveIt!*.

Para a simulação do robô no ambiente aberto, é utilizado o software *Gazebo*. A integração entre *ROS* e *Gazebo* consegue fazer com que o modelo *URDF*, por mais que não seja o nativo do *Gazebo*, seja aceito na simulação. Parâmetros do mundo podem ser ajustados e a integração de plugins como câmeras e sensores faz com que a simulação consiga ser utilizada em diferentes estudos. Algoritmos de imagem podem ser testados com os plugins de câmera já implementados, proporcionando um auxílio para demonstrar conceitos e teorias de funcionamentos.

A simulação fornecida possui os controladores de juntas já implementados, fazendo com que testes de códigos de movimentação e testes de controles já pudessem ser previamente testados, poupando riscos de dano ao protótipo e possibilitando trabalho simultâneo.

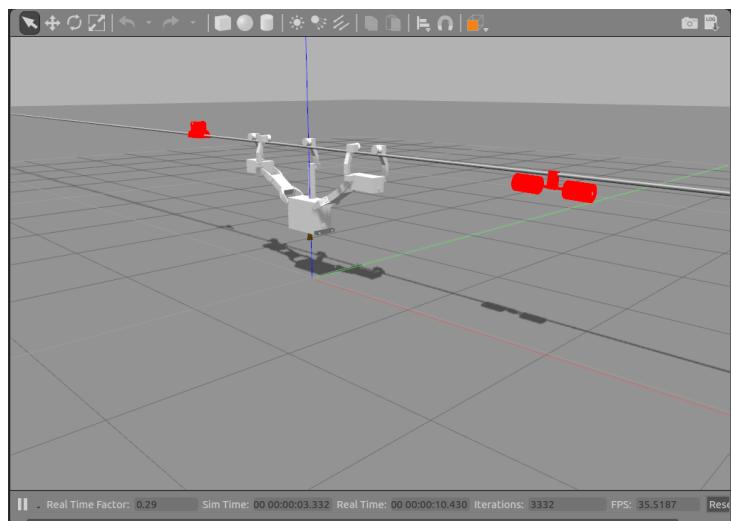


Figura 3.13: Simulação do *ELIR* no *Gazebo*.

Desenvolvimento e testes

4.1 Análise das Funcionalidades

4.1.1 Atuação

4.1.2 Planejamento de Movimento

4.1.3 Gerenciamento de Energia

4.1.4 Checagem da Integridade do Sistema

4.2 Soluções Mecatrônicas para o sistema robótico

4.3 Estudo da Movimentação

4.4 Simulação

4.5 Testes de Movimentação Física

4.6 Integração com os subsistemas

4.7 Análise Preliminar

Conclusão

O resultado do projeto alcançou as expectativas. Os problemas que aconteceram conseguiram ser contornados e o tempo gasto para sua solução conseguiu se adequar ao esperado pelo cronograma das tarefas. O material produzido atende às demandas do cliente e os pacotes produzidos foram organizados buscando facilitar o uso por terceiros.

5.1 Considerações finais

A gestão do projeto do robô como um todo, fez com que o resultado produzido alcançasse as expectativas. O nível de desenvolvimento aumentou progressivamente de forma que o projeto foi conduzido, adicionando paulatinamente diversos conhecimentos específicos que não seriam vistos normalmente durante a graduação, mas também fortalecendo conhecimentos já formados.

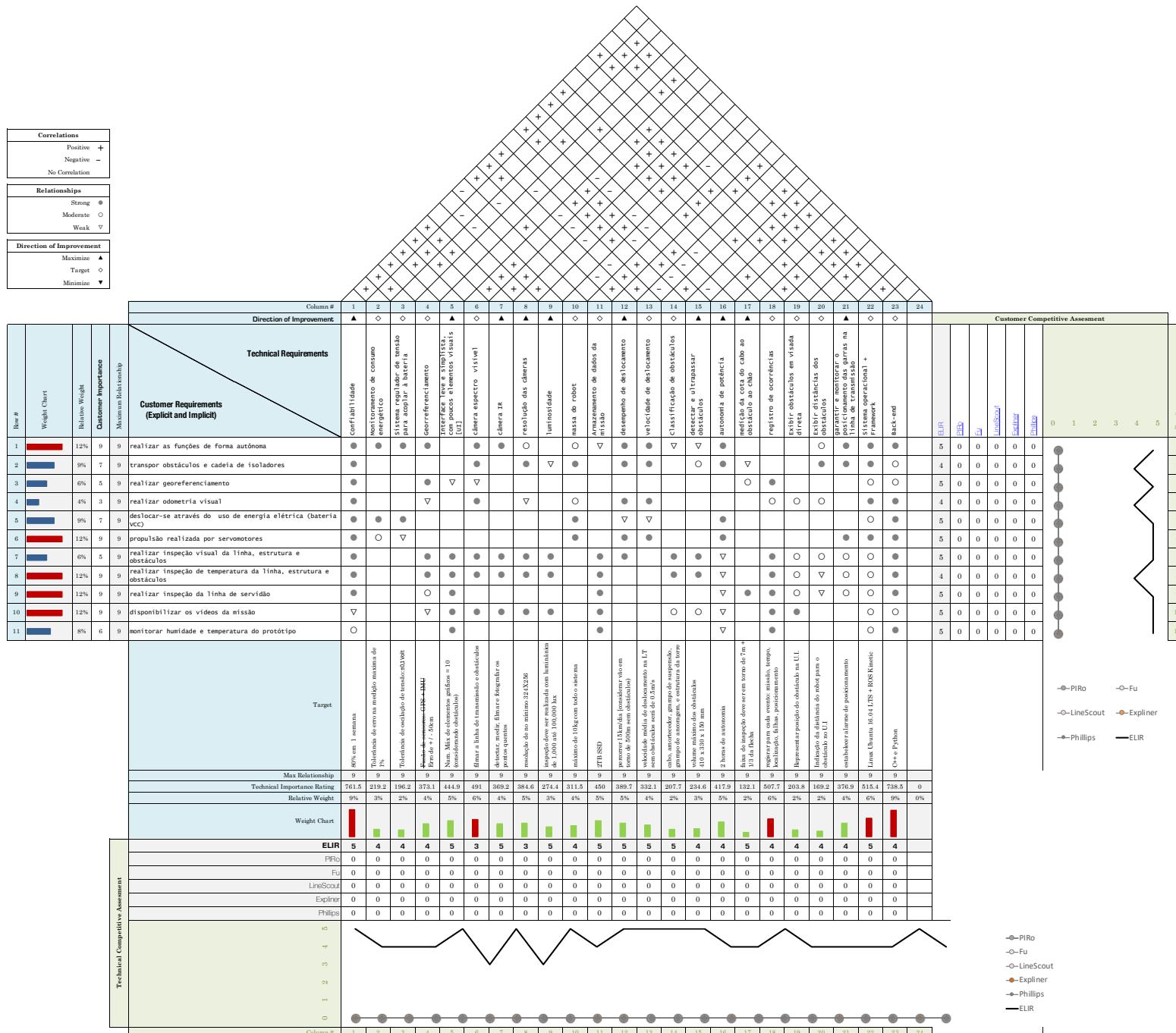
O que foi produzido para o projeto estará disponível para futuras consultas, o que impulsiona o desenvolvimento de projetos semelhantes. Novos estudos podem ser iniciados como trabalhos de graduação, ou pós-graduação, realizando provas de conceito e abrindo oportunidades para novas tecnologias.

O início do projeto se deu de forma lenta, por apresentar uma área do conhecimento nova para maior parte da equipe. A robótica necessita da integração de diversas áreas diferentes, e para a engenharia elétrica, o conhecimento de diversas camadas de abstração. Com a experiência e o desenvolver das atividades, os conhecimentos adquiridos possibilitaram atividades em paralelo e o aumento da versatilidade dos integrantes.

A experiência como um todo foi muito enriquecedora, adicionando conhecimentos que serão necessários no futuro e proporcionando um crescimento para todos os participantes.

Apêndice A

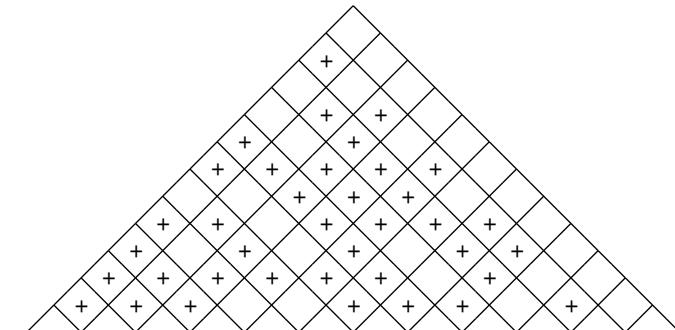
QFD



Correlations	
Positive	+
Negative	-
No Correlation	

Relationships	
Strong	●
Moderate	○
Weak	▽

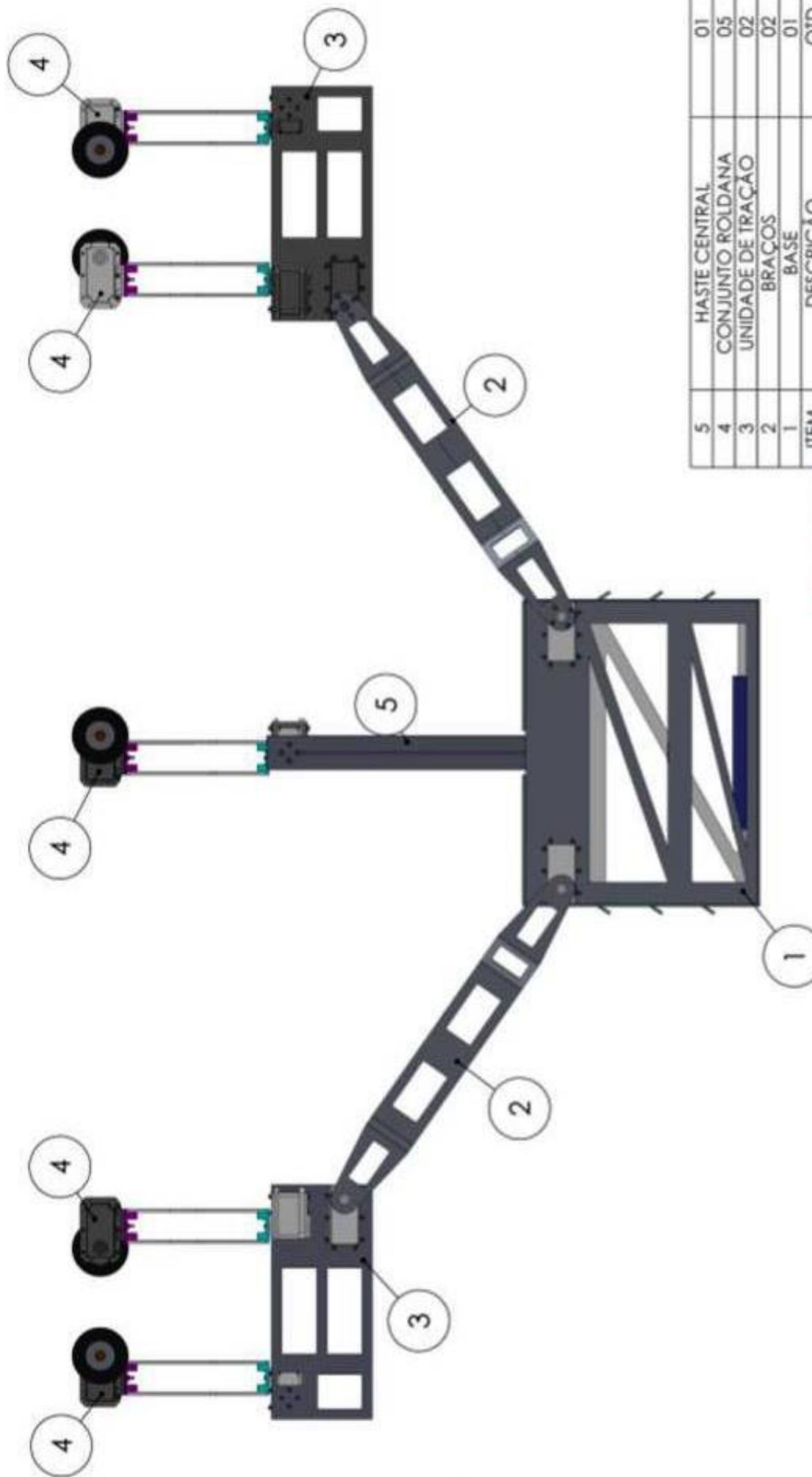
Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼



Target		verificar a integridade física do sistema antes e durante a missão	gerenciar o fornecimento de energia	realizar a comunicação e a aquisição dos dados	informar o posicionamento e orientação do sistema quando solicitado	realizar o planejamento da trajetória	mover a estrutura física e transporção dos obstáculos	classificar os objetos encontrados na linha	identificar pontos quentes e objetos na linha e na faixa de serviço	disponibilizar de forma simplificada os dados mais relevantes	fornecer parâmetro de confiabilidade e da estratégia a ser adotada	realizar a simulação da missão antes do início da mesma
Max Relationship	9	9	9	9	9	9	9	9	9	9	9	3
Functional Importance Rating	277.1	291.2	500	246	403.6	338.1	413.1	574	494.9	513.4	166.1	0
Relative Weight	7%	7%	12%	6%	10%	8%	10%	14%	12%	12%	4%	0%
Weight Chart												
Column #	1	2	3	4	5	6	20	21	22	23	24	25

Diagramas mecânicos

REV	DESCRIPTION	DEAW	DATE
0	Dogelka elaboration	Juliang Sartor	07/09/2017

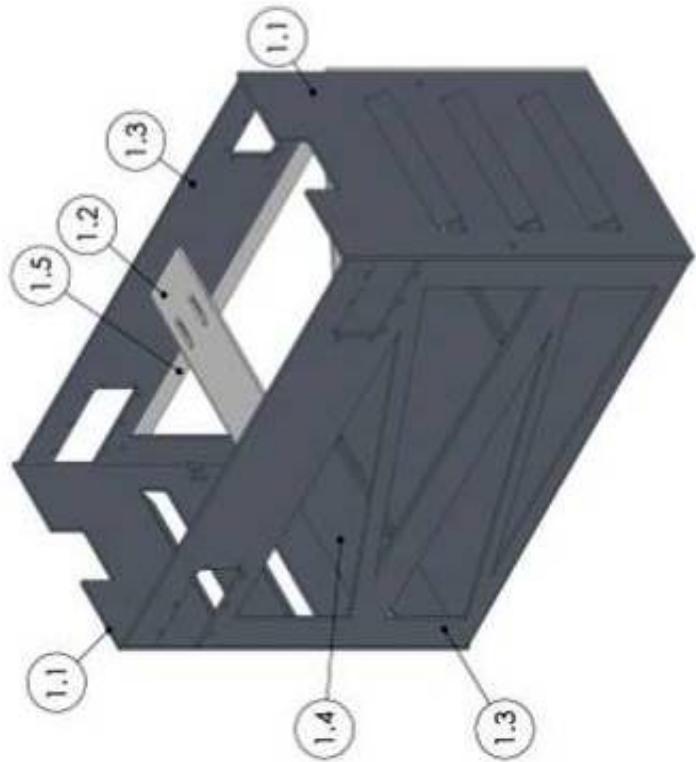


PI-RO 2.1
VISTA FRONTAL

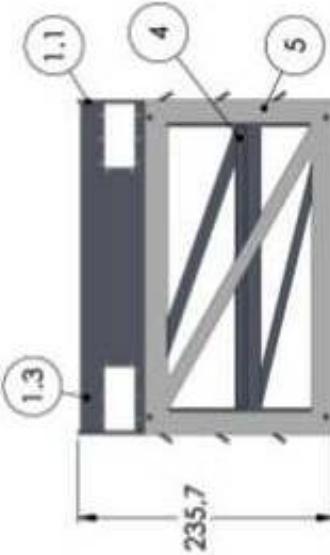


Nome	Sobrenome	Matrícula	Aluno(a)	Professor
Juliana	Sartori	20110000000000000000	Alumínio	N/A
Marcos	Röls	20110000000000000000	Alumínio	N/A

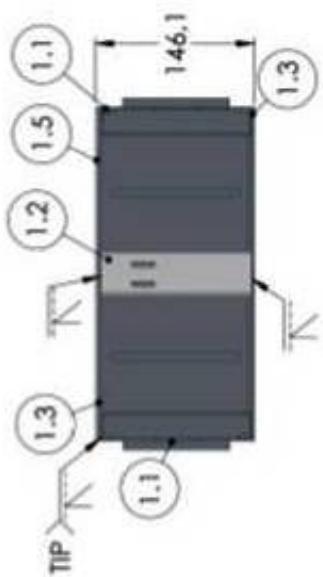
REV	DESCRIPTION	DRAW	DATE
0	Drawing elaboration	Julián Santoni	07/08/2017



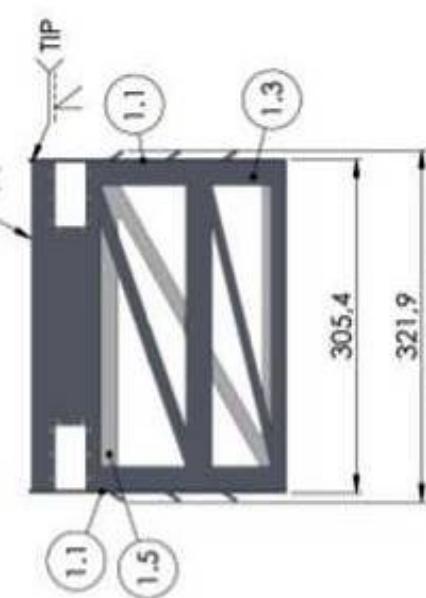
VISTA ISOMÉTRICA



VISTA TRASEIRA



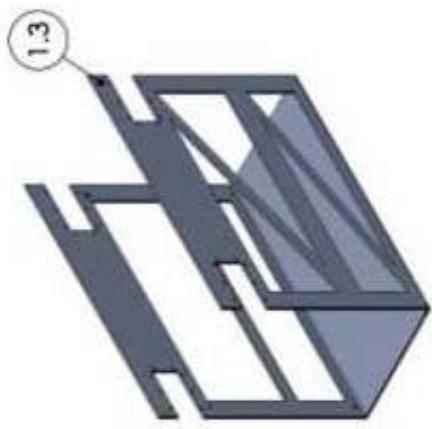
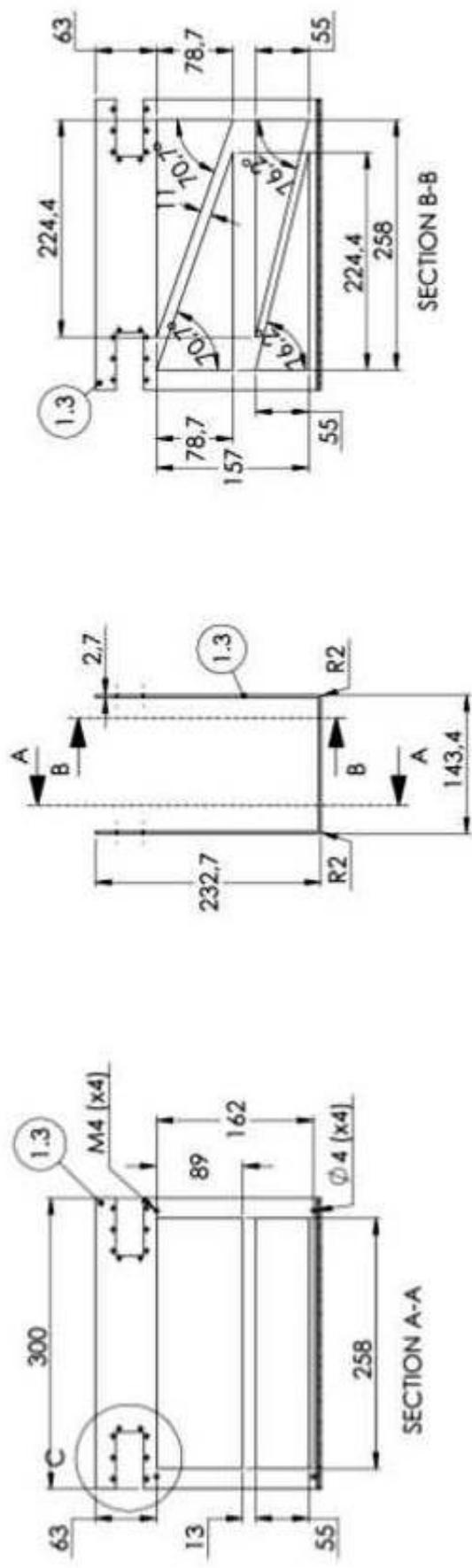
VISTA SUPERIOR



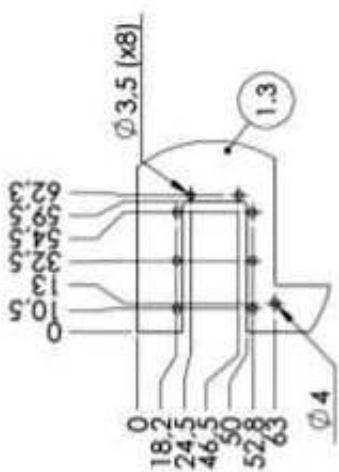
VETTA FRONTAI

ITEM	Draw Title	Project:	Draw:	Revise:
1.5	CHAPA 05: 305,4 x 175 x 2,7mm	PI-RO 2.1	Júliana Scartori	Aluminio
1.4	CHAPA 04: 148 x 254 x 2,7mm		Autor:	Marco Reis
1.3	CHAPA 03: 608,8 x 300 x 2,7mm		Date:	07/09/2017
1.2	CHAPA 01: 138 x 47 x 2,7 mm		Comments:	THE INFORMATION IN THIS DRAWING IS THE PROPERTY OF FIEB. IT IS FOR INTERNAL USE ONLY AND CANNOT BE COPIED OR USED OUTSIDE THE COMPANY'S PREMISES.
1.1	CHAPA 01: 143,4 x 235,4 x 2,7 mm			
	DESCRICAÇÃO			
	QTD			
	1:2			
	X			
	A-3			
	2/13			
BASE - ITEM 01				

REV DATE DECODED BY
0 02/09/2017 02/09/2017

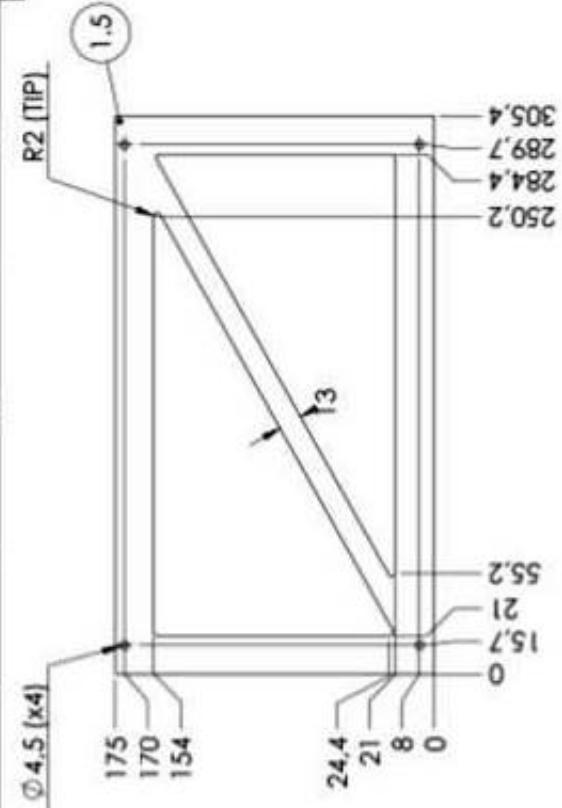


DETALHE C
ESCALA 2:5
FURAÇÃO TÍPICA PARA ENCAIXE
DO MOTOR MX-106T

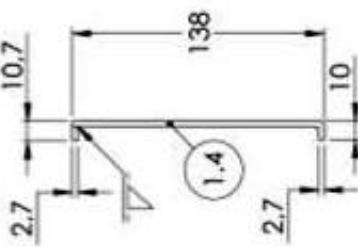
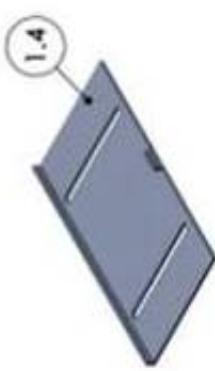
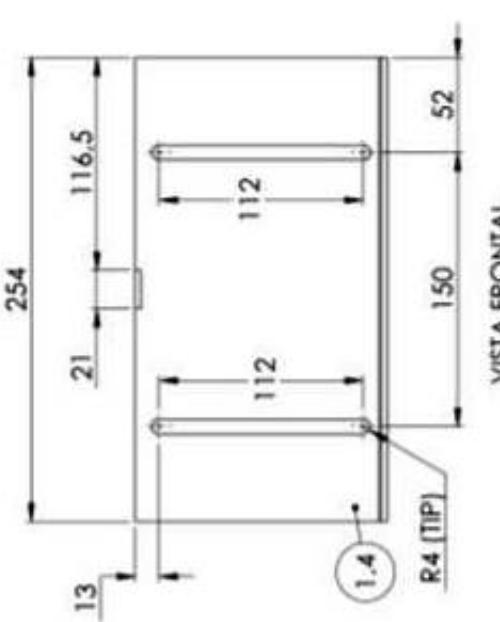


ITEM	1.3	DESCRIÇÃO	Chapa 608.8 x 300 x 2.7mm	MATERIAL	Alumínio
BASE				UNIDADE	1:5
Project	Prova	Draw	Juliana Sartori	Designer	A3
PRo 2.1		Draw	Mario Reis	Reviewer	4/13
Robotics		Draw		Reviewer	
SENAI FIEB		Draw		Reviewer	
3R		Draw		Reviewer	
Brasil é o futuro da indústria no Brasil		Draw		Reviewer	
Industrial Institute of		Draw		Reviewer	
Robotics		Draw		Reviewer	
Nome:	1. Até momento existem os três tipos de robôs que se destacam: Robôs de braço, Robôs de engenharia e Robôs de treinamento.	Assunto:	Alumínio	Responsible:	N/A
Responsible:		Date:		Comments:	

REV	DESCRIPTION	DRAW	DATE
0	Drawing elaboration	Juliana Santori	07/09/2017



VISTA FRONTAL
VISTA LATERAL



ITEM	DESCRICAÇÃO	MATERIAL	
		mm	kg
5	Chapa 305.4 x 145 x 2.7mm	Alumínio	1.2
		A3	-
		Therm	-
		N/A	5/13

Project: PRo 2.1
Draw Date: 07/09/2017
Draw Time:
Notes:
1. All measurements in mm
2. Material: Alumínio
3. Color: Branco

The information in this document must be handled strictly in accordance with the conditions of use.

© 2017 - SENAI - Brazilian Institute of Education and Research



Brazilian Institute of Education and Research

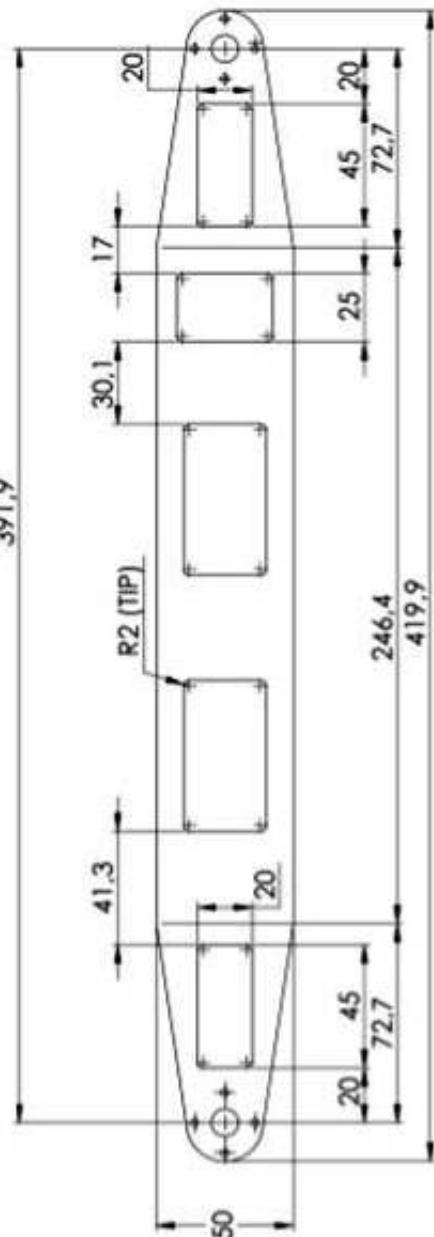
BR

Robotics

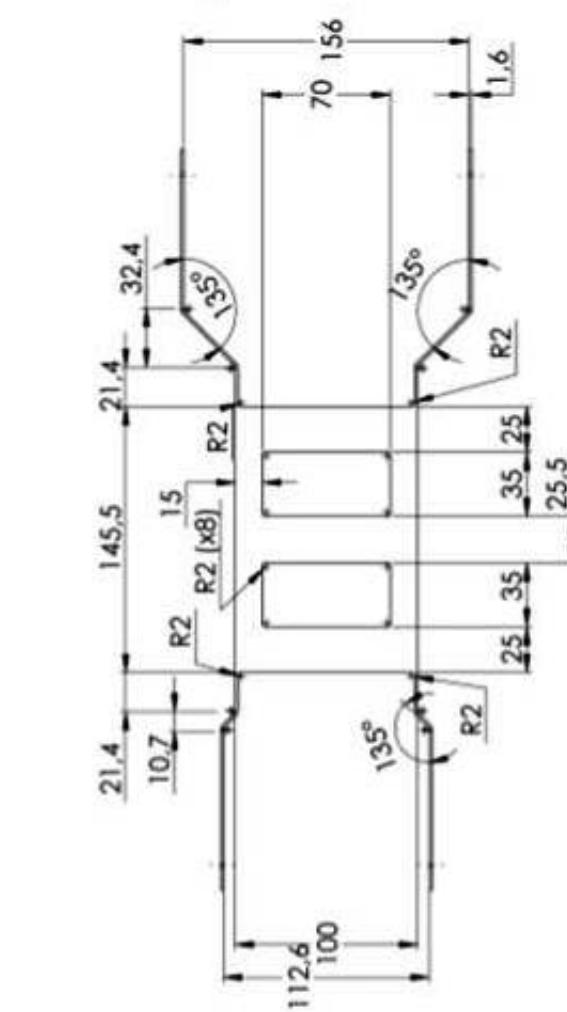
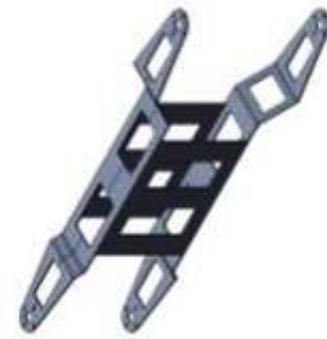
BR

REV	DESCRIPTION	DRAW	DATE
0	Drawing elaboration	Júlian Sartori	07/11/2016
A	Modificação de dimensões	Júlian Sartori	07/09/2017

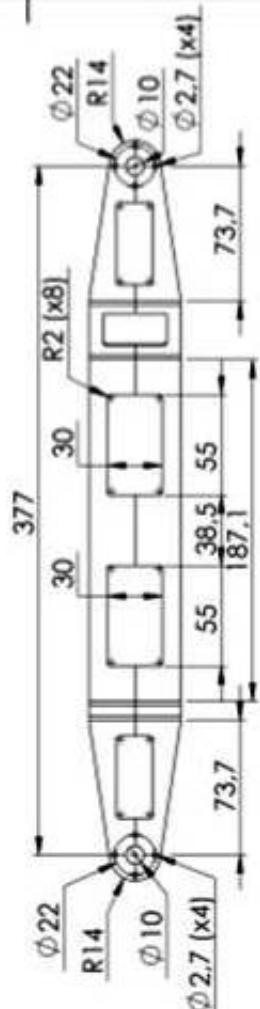
391,9



VISTA ISOMÉTRICA



VISTA SUPERIOR



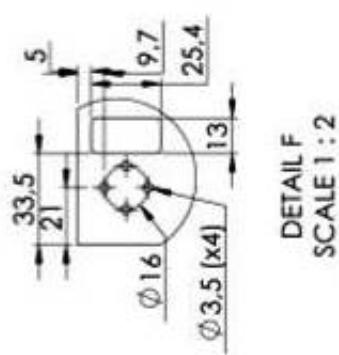
NOTA 01: CONSIDERAR QUE FILETES NÃO ESPECIFICADOS TÊM RAIO IGUAL A 3,0mm.
 NOTA 02: AS DIMENSÕES DE ABERTURA (156mm e 112,6mm) DEVEM SER MANTIDAS APÓS A REALIZAÇÃO DAS DOBRAS DO PERFIL.

SENAI FIEB		BRAÇO- ITEM 02	
Brasília - Distrito Federal	Brasília - Distrito Federal	Project:	PI-Ro 2.1
Brasília - Distrito Federal	Brasília - Distrito Federal	Date:	07/09/2017
Brasília - Distrito Federal	Brasília - Distrito Federal	Author:	Júlian Sartori
Brasília - Distrito Federal	Brasília - Distrito Federal	Reviewer:	Marco Reis
Brasília - Distrito Federal	Brasília - Distrito Federal	Base File:	Base File
Brasília - Distrito Federal	Brasília - Distrito Federal	Scale:	1:2
Brasília - Distrito Federal	Brasília - Distrito Federal	Page:	02
Brasília - Distrito Federal	Brasília - Distrito Federal	Sheet:	A3
Brasília - Distrito Federal	Brasília - Distrito Federal	Comments:	None
Brasília - Distrito Federal	Brasília - Distrito Federal	Revision:	0/13

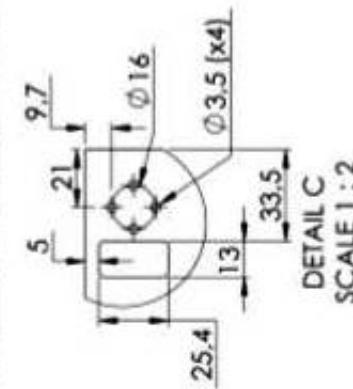
REV	DESCRIPTION	DRAW	DATE
0	Drawing elaboration	Júlio Soárez	07/11/2019
A	Modificação da dimensão	Júlio Soárez	18/11/2019



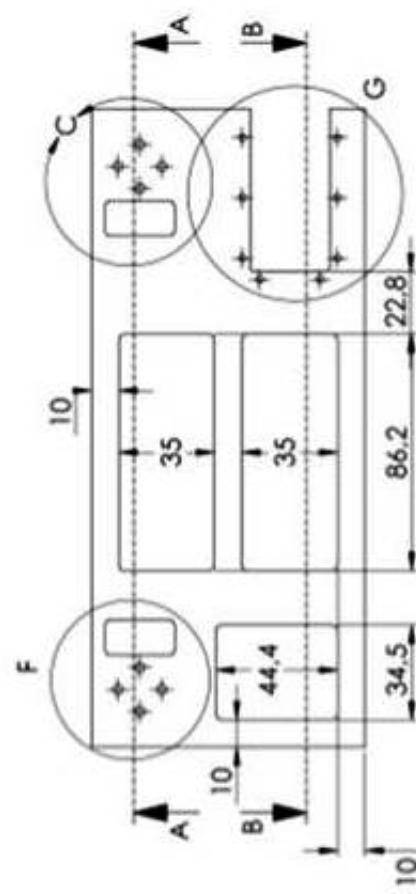
SECTION A-A



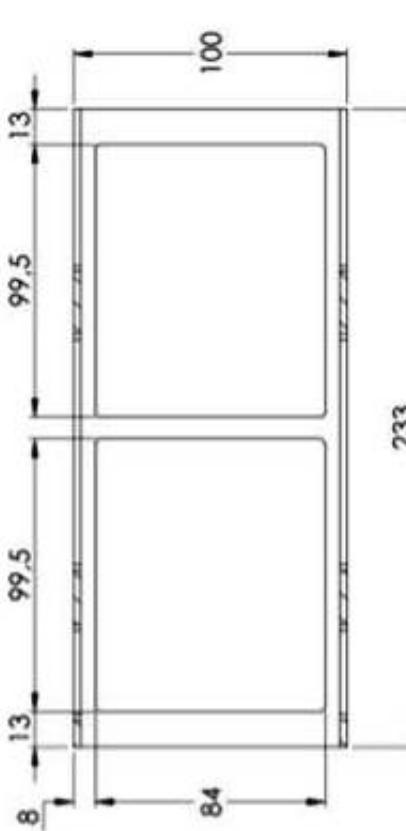
DETAIL F
SCALE 1 : 2



DETAIL C
SCALE 1:2



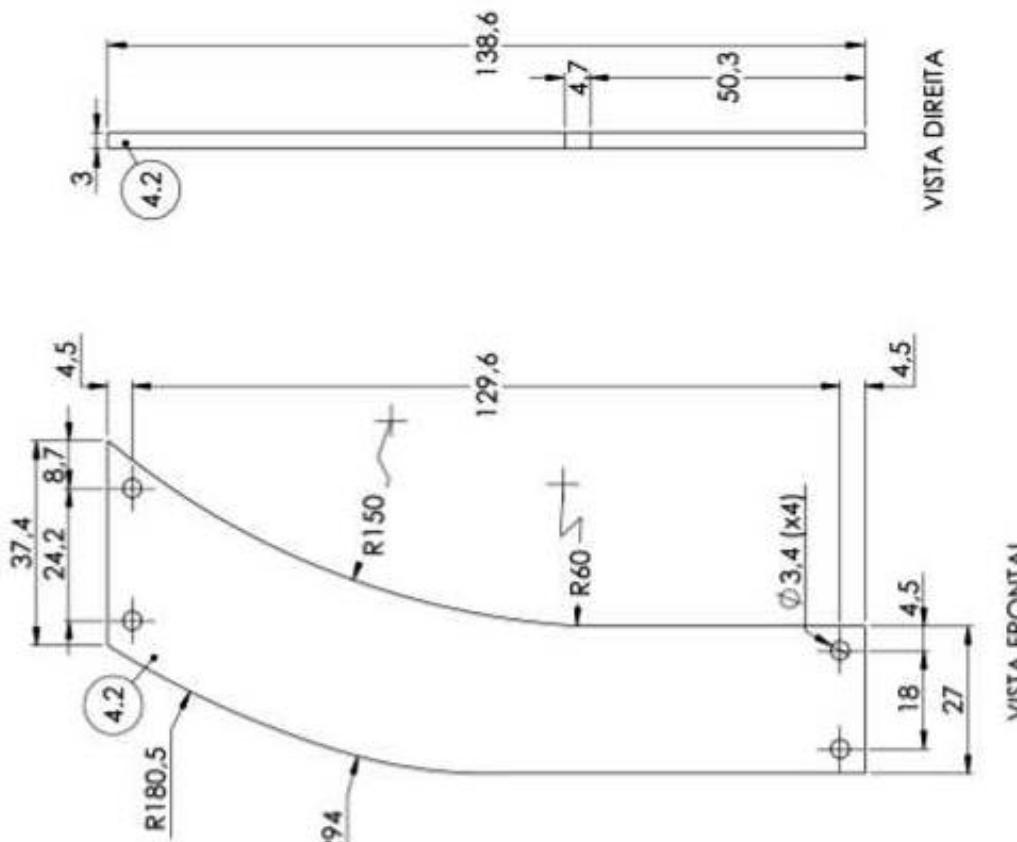
VISTA FRONTAL



233



NOTA 01: CONSIDERAR QUE FILETES NÃO ESPECIFICADOS TEM RAIO IGUAL A 2mm.



VISTA ISOMÉTRICA

4.9	Motor MX-28	05	-
4.8	Motor MX-106T-31	05	-
4.7	Garra- Chapa IV	05	Alumínio
4.6	Garra- Chapa III	05	Alumínio
4.5	Roldana externa	05	Borracha
4.4	Roldana interna	05	Alumínio
4.3	Eixo Roldana	05	Alumínio
4.2	Garra- Chapa I	10	Alumínio
4.1	Suporte garra	05	Alumínio



ITEM	DESCRÍÇÃO	QTD	MATERIAL
E	CONJUNTO ROLDANA - ITEM 04	1:2	
ITEM 4.2		X	
A3			
B			
C			
D			
E			
F			

NOTA: 1. As dimensões em milímetros.
2. Unidade utilizada: milímetro (mm).
3. Desenho feito por:
Projeto: PhRo 2.1
Nome: Juliana Santori
Assinatura: _____
Nome: Marco Reis
Assinatura: _____
Data: 07/09/2017
Local: São Paulo - SP - Brasil
NOTA: THIS DRAWING IS THE PROPERTY OF THE COMPANY THAT ISSUED IT. UNAUTHORIZED COPIES ARE FORBIDDEN.

4

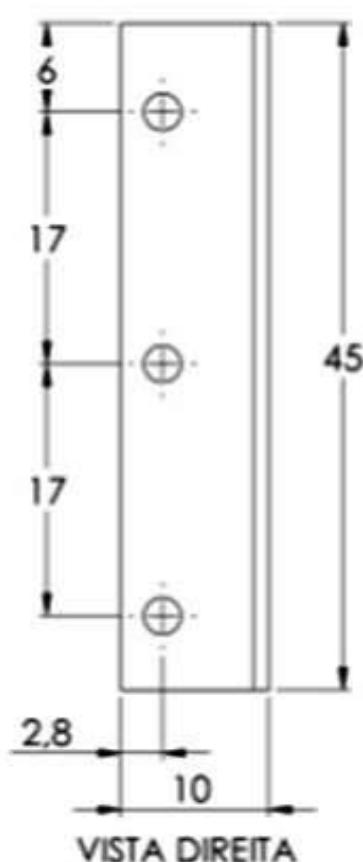
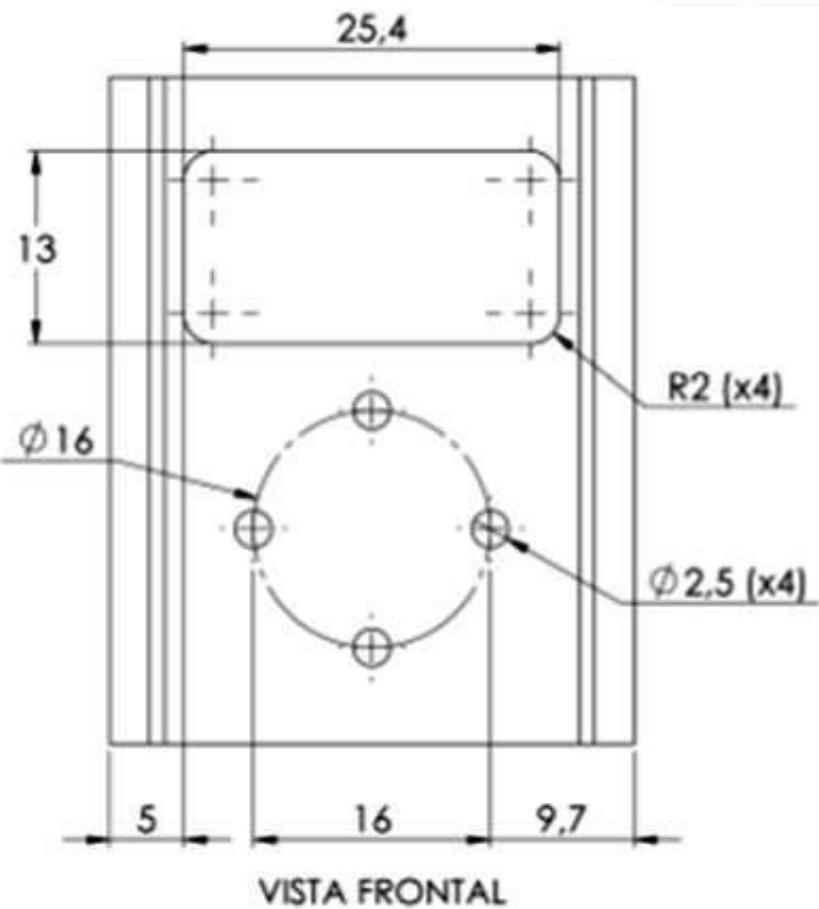
3

2

1

REV	DESCRIPTION	DRAW	DATE
0	Drawing elaboration	Juliana Sartori	07/09/2017

F



D

C

B

F

E

D

C

B

A

VISTA FRONTAL

VISTA DIREITA

VISTA INFERIOR

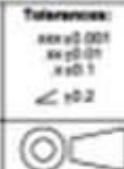


VISTA ISOMÉTRICA



Notes:
 1. All measurement in mm
 unless otherwise specified.
 2. Remove cutting edge
 3. Default finishing

Revisão:
0



Draw Title:

CONJUNTO ROLDANA ITEM 4.1

Scale: 2:1

Quantity: 05

Size: A4

Finishing:

Project: PI-Ro 2.1

Order: Juliana Sartori

Approv:

Marco Reis

Material: Alumínio

ThermalT:

N/A

Page: 9/13

Date: 07/09/2017

THE INFORMATION IN THIS DOCUMENT CANNOT BE COPIED, GIVEN AWAY OR
USED FOR OTHERS MEANS EXCEPT THOSE CONTAINED IN THE CONTRACT TERMS.

4

3

2

1

4

3

2

1

REV	DESCRIPTION	DRAW	DATE
0	Drawing elaboration	Juliana Sartori	07/09/2017

F

F

E

E

D

D

C

C

B

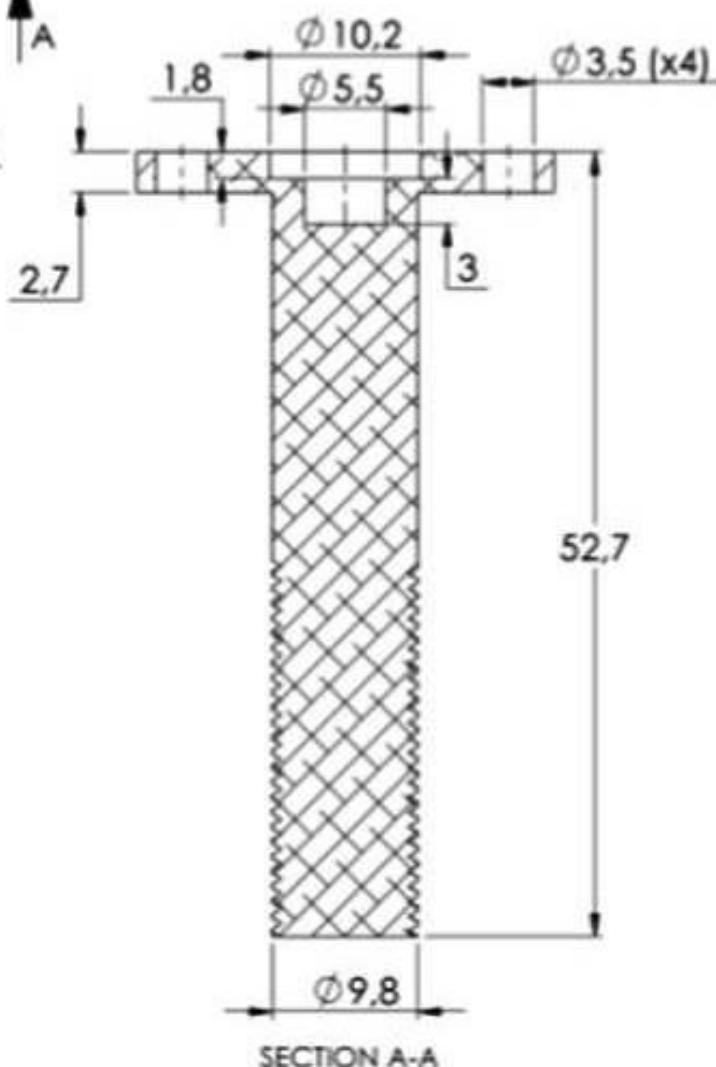
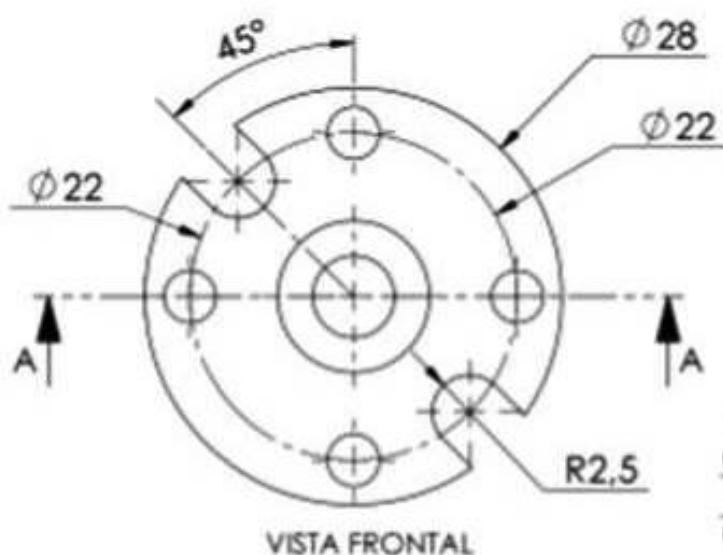
B

4

3

2

1



VISTA ISOMÉTRICA

SENAI **SISTEMA**
FIEB
Poderoso para Indústria do Estado de São Paulo

BR Brazilian Institute of
Robotics

Notes:
1. All measurement in mm
unless otherwise specified.
2. Remove cutting edge
3. Default finishing

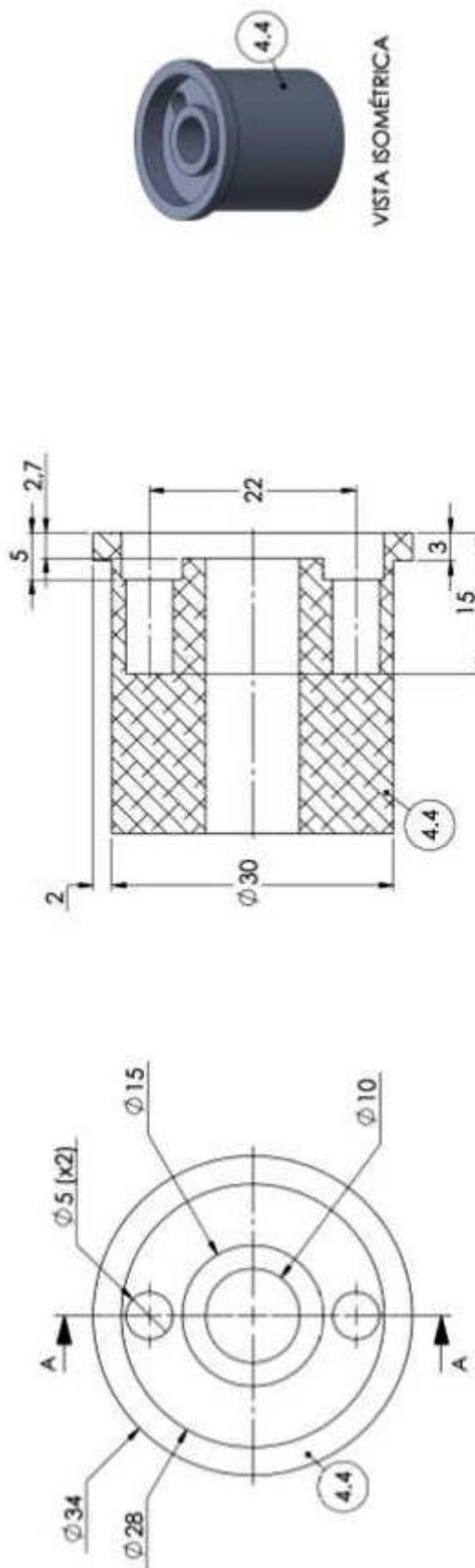
Tolerances:
 ± 0.02 (0.019)
 ± 0.01
 ± 0.1
 ± 0.2

Reviewed:
0

Draw Title: CONJUNTO ROLDANA ITEM 4.3
Project: PI-Ro 2.1
Draw: Juliana Sartori
Material: Alumínio
Approved: Marco Reis
ThermalT: N/A
Scale: 2:1
Quantity: 05
Size: A4
Finishing:
Date: 07/09/2017
Page: 10/13

THE INFORMATION IN THIS DOCUMENT CANNOT BE COPIED, GIVEN AWAY OR USED FOR OTHER MEANS EXCEPT THOSE CONTAINED IN THE CONTRACT TERMS.

REV	DESCRIPTION	DRAW	DATE
0	Drawing elaboration	Julianna Santori	07/09/2017

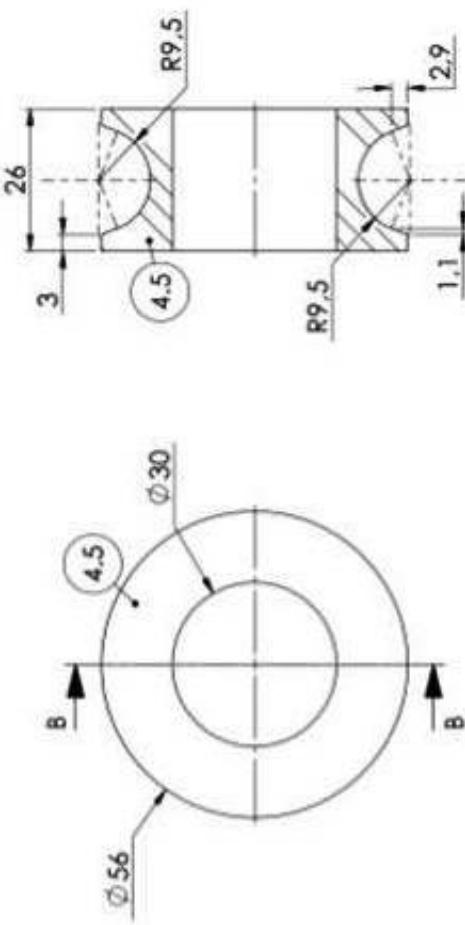


VISTA
FRONTAL

SECCION A-A



VISTA
ISOMÉTRICA



VISTA
FRONTAL
ESCALA 1:1

ESCALA 1:1

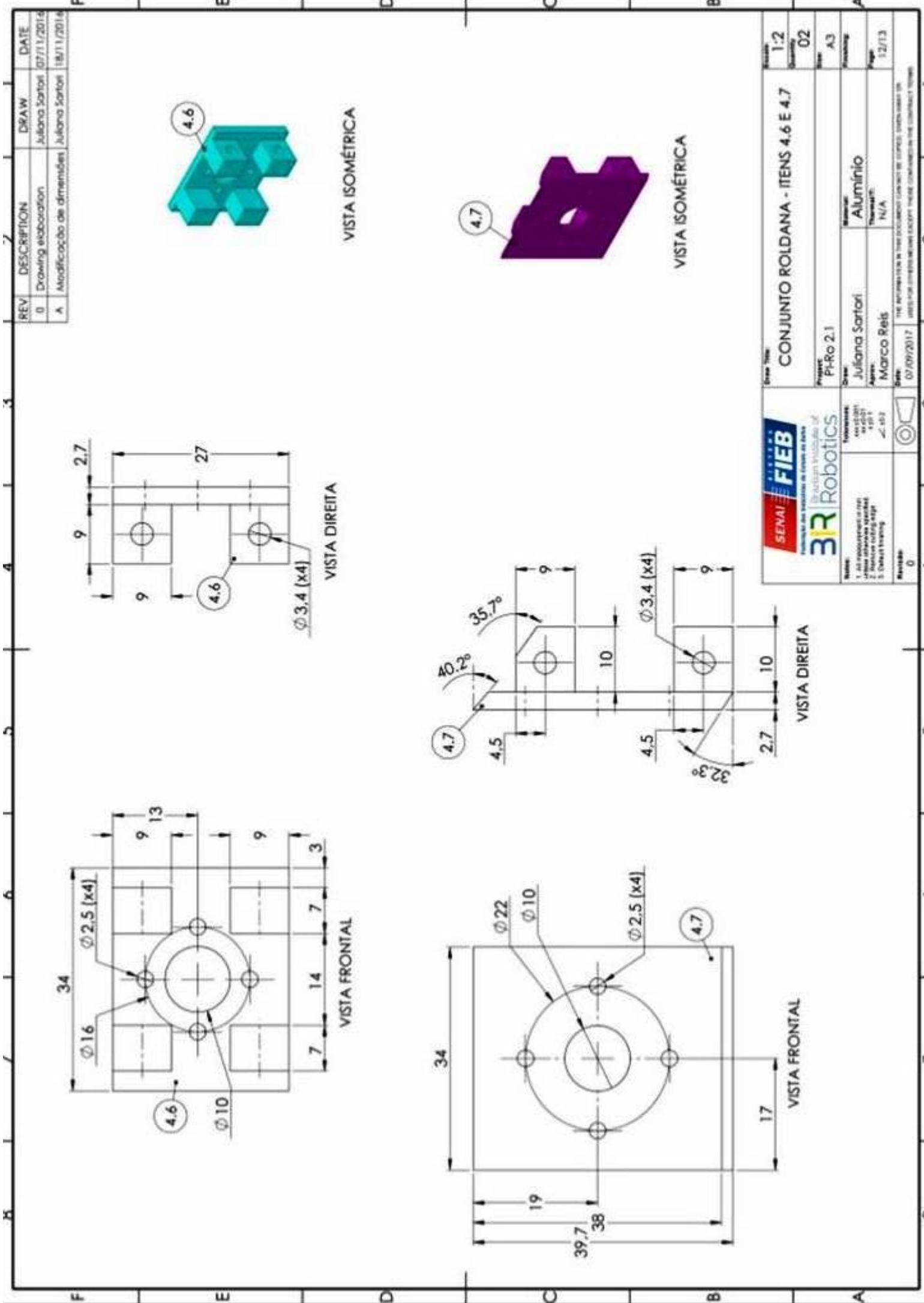
Drawing No:	CONJUNTO ROLDANA	
	ITEM 4.4 E 4.5	Rev. 2.1
Project:	PhRo 2.1	Date: 05/09/2017
Client:	SENAI FIEB	Author: Juliana Santori
Software:	Brasília 3D v10.0.0.0	Reviewer: Marco Reis
Comments:	1. Adicionar dimensionamento de espessura para o eixo central. 2. Remover cutaway regions. 3. Detalhar trapping.	Comments: N/A
Reviewer:	N/A	Date: 07/09/2017

BRASÍLIA 3D

Software para modelagem 3D

versão 10.0.0.0

versão 10.0.



4

3

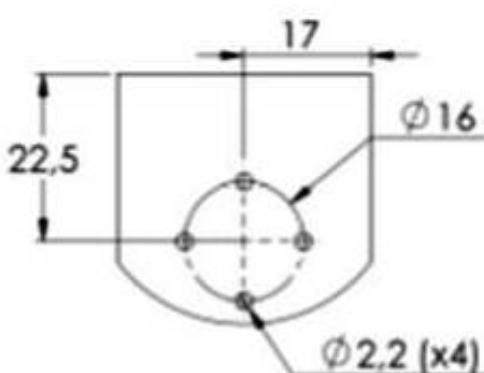
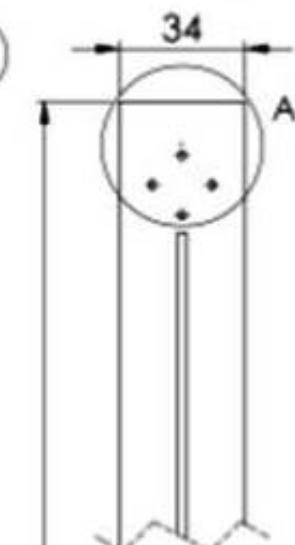
2

1

F

F

5

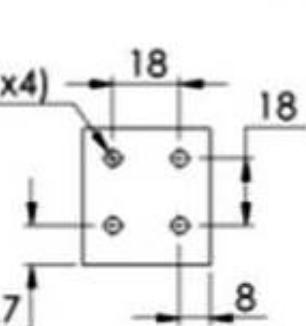


DETAIL A
SCALE 1 : 1



VISTA FRONTAL

VISTA DIREITA



VISTA INFERIOR



VISTA ISOMÉTRICA



Notes:
1. All measurements in mm unless otherwise specified.
2. Remove cutting edges.
3. Default finishing.

Revisado:
0

Drawing Title:

HASTE CENTRAL - ITEM 5

Scale:
1:2

Quantity:
01

Project:

PI-Ro 2.1

Size:
A4

Date:

07/09/2017

Page:
13/13

Draw:

Material:

Alumínio

Approve:

ThermalT:

N/A

Marco Reis

Date:

Page:

07/09/2017

13/13

THE INFORMATION IN THIS DOCUMENT CANNOT BE COPIED, GIVEN AWAY OR
USED FOR OTHERS MEANS EXCEPT THOSE CONTAINED IN THE CONTRACT TERMS.

4

3

2

1

F

E

D

C

B

A

Diagramas eletro-eletrônicos

Logbook

CONFIGURAÇÃO DOS LIMITES DE GIRO DOS MOTORES

Objetivos

O teste teve como objetivo estabelecer os limites de giro dos motores em seus controladores, com base nos limites físicos da estrutura do robô.

Descrição do teste

É criado um “Controller manager” que conecta os motores e publica em um tópico as informações destes. As juntas do robô são posicionadas manualmente em suas posições máximas e mínimas, então o tópico “motor_states” é monitorado para verificar as posições dos motores.

DATA

8 AGOSTO 2018

LOCALIDADE

SENAI CIMATEC
SALVADOR - BAHIA

Mandruvah team

Cleber
Carlos
Ícaro
Davi

17:00

Foram coletados os limites de giro dos motores com id 11, 12, 13, 21, 22 e 23.

17:05

Ajustamos as posições iniciais dos controladores das juntas com base na posição “home” da simulação no MoveIt!. Nesse momento, verificamos que o valor que é publicado no controlador para mover a junta é a posição em radianos em relação à posição inicial que foi determinada no controlador.

TESTE DE MOVIMENTAÇÃO DOS SERVO-MOTORES

Objetivos

Verificar o comportamento do robô executando alguns movimentos em um dos braços.

Descrição do teste

Os motores são alimentados e seus controladores executados. A partir daí, valores de posição são publicados e o comportamento do robô verificado.

DATA

10 AGOSTO 2018

LOCALIDADE

SENAI CIMATEC
SALVADOR - BAHIA

Mandruvah team

Cleber
Carlos
Ícaro
Davi

15:20

O braço do robô foi levantado até a posição “home” com as duas juntas sendo movimentadas ao mesmo tempo. Antes de atingir a posição determinada, o motor com id 21 apresentava erro de overload.

16:10

Quando o robô começa o movimento já próximo da posição final, “home”, o braço consegue alcançar o objetivo. Depois de cerca de 5 minutos nessa posição, um erro de overheat é apresentado.

TESTE DE MOVIMENTAÇÃO DO BRAÇO

Objetivos

Verificar possíveis motivos para erro de “Overload” na junta 12-22 apresentado em teste anterior.

Descrição do teste

Os motores são alimentados e seus controladores executados. A partir daí, valores de posição são publicados e o comportamento do robô verificado.

DATA

13 AGOSTO 2018

LOCALIDADE

SENAI CIMATEC
SALVADOR - BAHIA

Mandruvah team

Cleber
Carlos
Ícaro
Davi

16:50

Foi verificado que os motores estão configurados para operar com 100% do torque, não sendo assim essa a causa do problema.

17:15

Percebemos também que o problema acontece com maior frequência quando as juntas do braço e da unidade de tração são acionadas ao mesmo tempo. Quando é acionada uma junta por vez, o “Overload” acontece menos vezes.

17:32

É levantada a suspeita de que a falta do cabo de sincronização nos motores da junta pode ser a causa da falha. Com o cabo conectado, o problema não aconteceu.

TESTE DE CONVERSOR DA PLACA DE POWER MANAGEMENT

Objetivos

Verificar possíveis problemas do conversor da placa de power management e sua resposta de saída.

Descrição do teste

O conversor é retirado da placa e então testado com fonte de alimentação e sua saída observada com um multímetro. O conversor testado é do modelo UWE-12/10-Q12PB-C

DATA

25 SETEMBRO 2018

LOCALIDADE

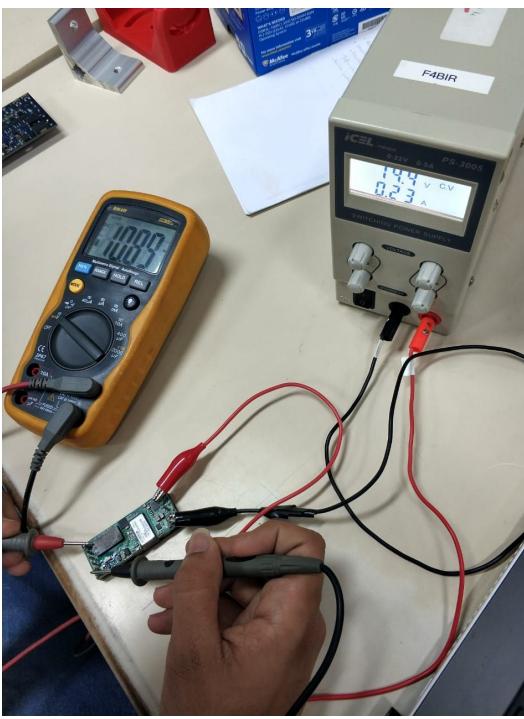
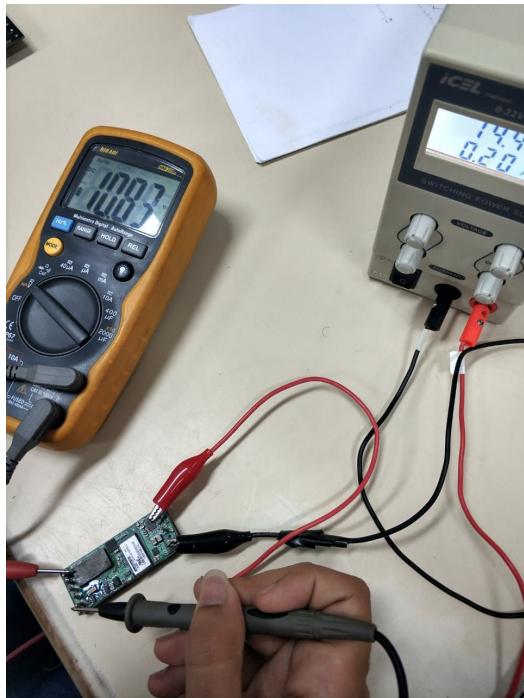
SENAI CIMATEC
SALVADOR - BAHIA

Mandruvah team

Cleber
Carlos
Ícaro
Davi

16:05

O conversor retirado da placa foi testado com alimentação de uma fonte de tensão com 14 Volts, e o mesmo consumiu um valor de cerca de 200 mA. Sua tensão de saída ficou em aproximadamente 10 Volts.

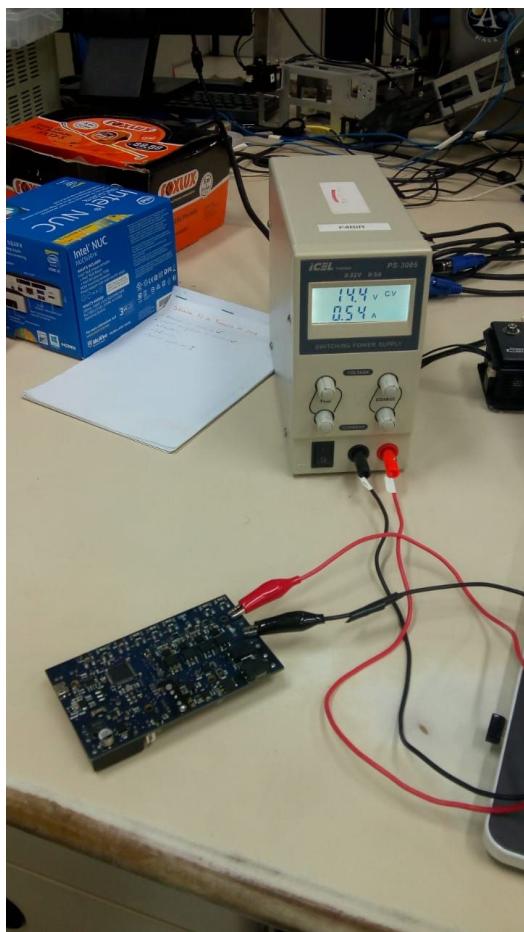


16:12

A corrente requisitada pelo conversor defeituoso oscila entre 200 mA e 250 mA. Sua temperatura, ao contrário do conversor que está funcionando corretamente, não aumenta e o conversor permanece frio.

16:28

O conversor que já está na Power Management permanece apresentando funcionamento correto. Sua temperatura aumenta quando permanece ligado.



ELIR

logbook

TESTE DOS MOTORES/CONTROLLER_MANAGER

Objetivos

Identificar se há algum motor defeituoso que pode estar “sujando” a comunicação dos motores.

Descrição do teste

Um motor é conectado e o arquivo controller_manager.launch é executado e verifica-se se o motor foi encontrado. Em seguida, são inseridos os demais motores, um a um, para que se perceba se a comunicação ainda acontece.

DATA

04 OUTUBRO 2018

LOCALIDADE

SENAI CIMATEC
SALVADOR - BAHIA

Mandruvah team

Cleber
Carlos
Ícaro
Davi

13:40

Todos os 18 motores estavam conectados, utilizando apenas os componentes (cabos e hub) da ROBOTIS. Quando o controller_manager.launch foi executado, nenhum motor foi encontrado.

13:42

Com apenas um motor conectado, o controller_manager o encontrou.

13:55

O teste prosseguiu até que, quando o motor de ID 14 foi conectado, o controller_manager não encontrou mais motores.

14:10

O motor de ID 14 foi removido e o teste continuou. O mesmo erro aconteceu quando o motor de ID 3 foi adicionado. Esse motor também foi retirado.

14:15

O teste seguiu até o último motor, nenhum motor aparentemente defeituoso foi encontrado. A comunicação funcionou bem com os 16 motores restantes.

14:45

Dois motores novos foram conectados. O controlador foi executado por volta de 20 vezes, em todos os testes todos os motores foram encontrados.

TESTE DA PLACA DE POWER MANAGEMENT

Objetivos

Verificar possíveis problemas da montagem da placa de power management sem a presença de um dos conversores DC/DC

Descrição do teste

Os capacitores de acoplamento do regulador de tensão para o Atmega 32U4 são soldados e então a placa é alimentada com 14.4 Volts. A temperatura é monitorada com um multímetro com termopar, e os níveis de tensão com um multímetro comum.

DATA

05 OUTUBRO 2018

LOCALIDADE

SENAI CIMATEC
SALVADOR - BAHIA

Mandruvah team

Cleber
Carlos
Ícaro
Davi

15:20

Capacitores de acoplamento foram soldados na placa, respeitando a polarização estabelecida no projeto de power management.

Capacitores de tântalo de 10uF de 16 Volts.

15:25

A placa foi alimentada com uma fonte de tensão à 14.4 Volts. Para que haja o funcionamento da placa é necessário realizar um curto entre os pinos -Vin e S2. Desta maneira inicia-se a operação da placa.



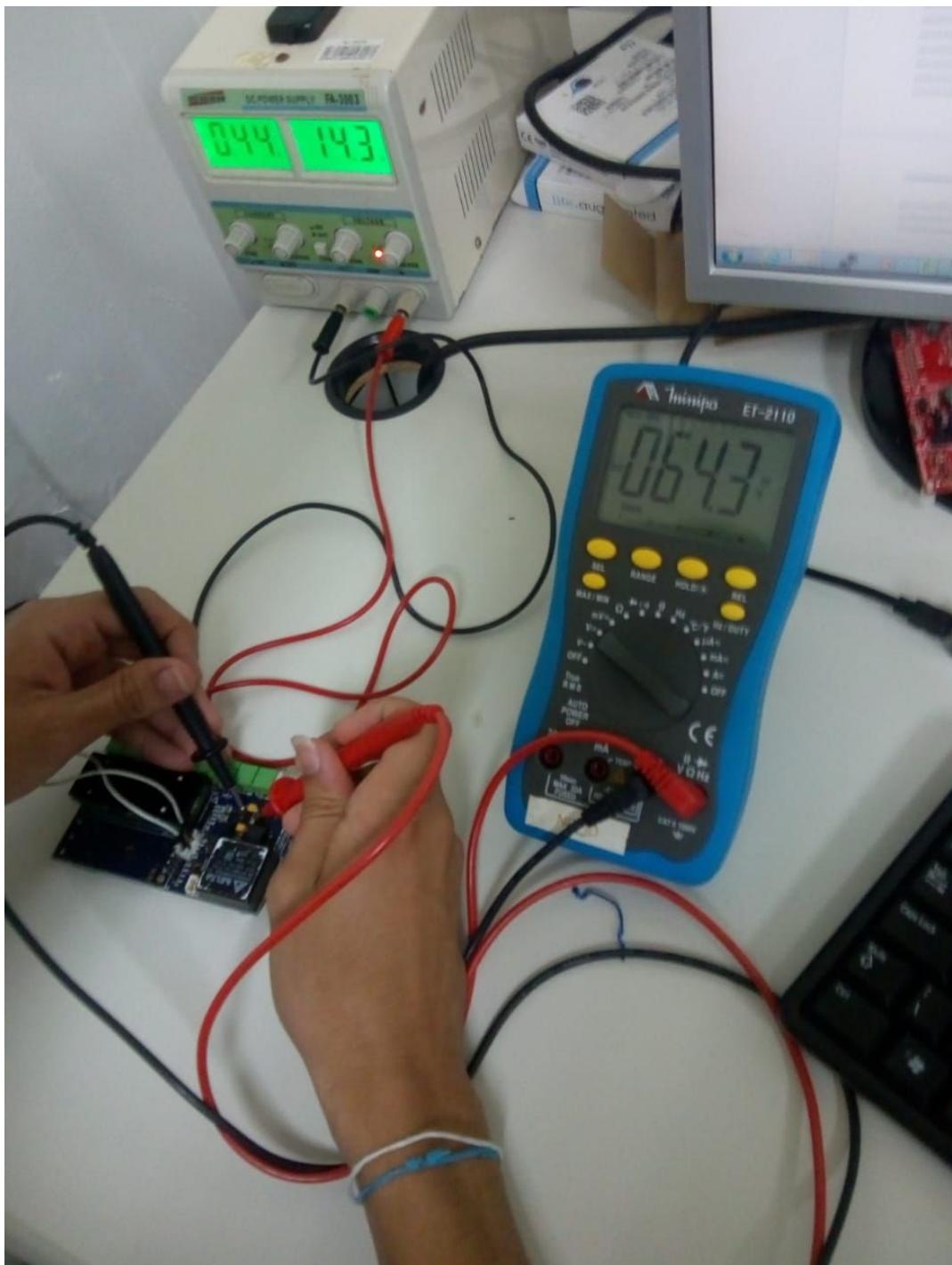
15:27

A placa apresentou um aumento da temperatura do regulador de tensão 5 Volts, chegando a alcançar temperaturas em cerca de 110°C.



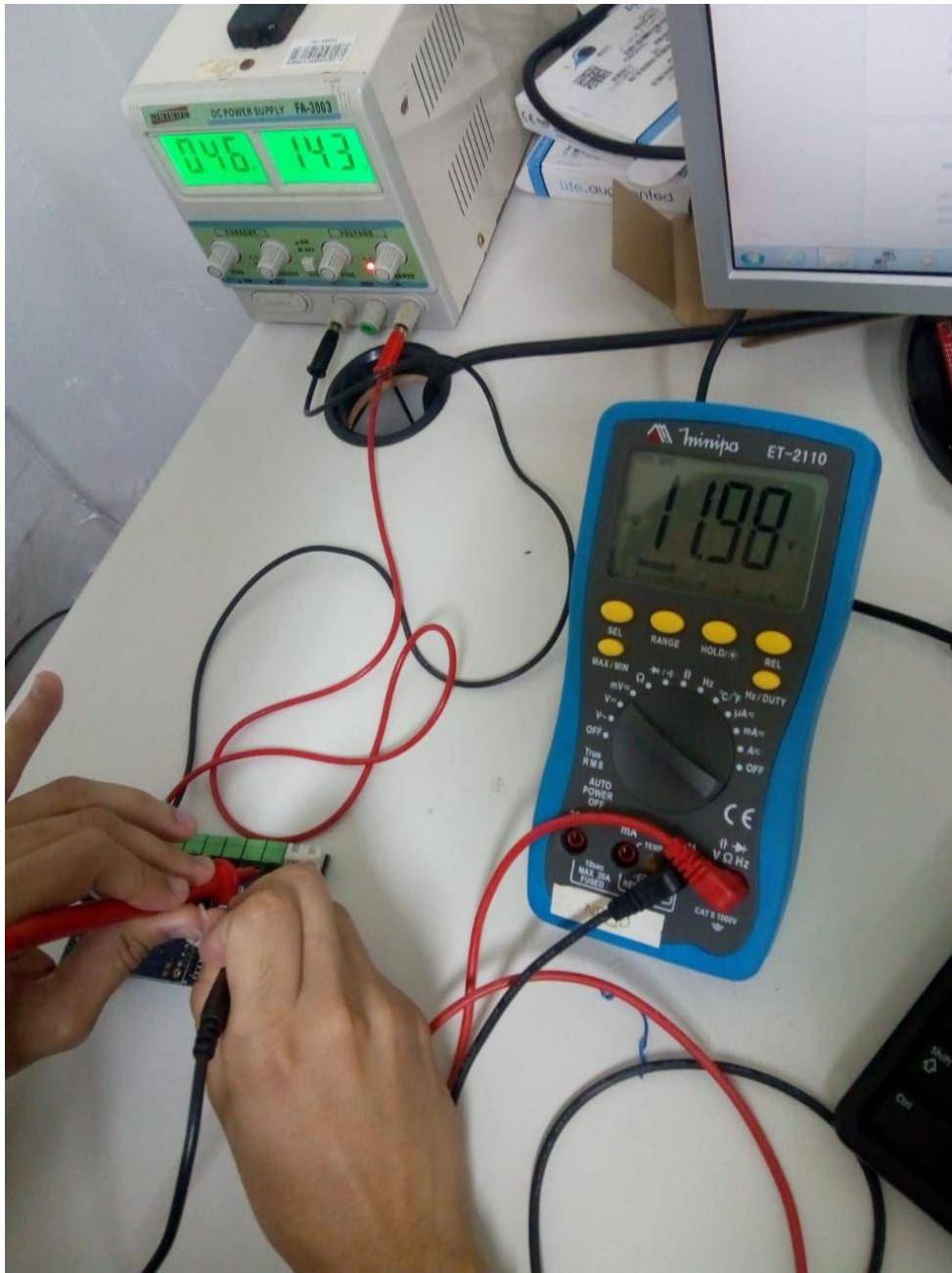
15:28

Os níveis de tensão gerados pelos reguladores também apresentaram níveis fora do esperado. O regulador de 5 volts apresentou tensões extremamente baixas, revelando um possível mal funcionamento, em torno de 70 milivolts



15:30

O conversor ainda presente na placa apresentou um funcionamento normal, gerando 12 Volts nas 3 portas de saída fixa, como esperado.



TESTE COM O ROBÔ NA LINHA

Objetivos

Montar setup do teste, acessar a NUC remotamente e movimentar o robô na linha

Descrição do teste

O robô é posicionado na linha, é estabelecido um acesso remoto, são iniciados os controladores e movimenta-se o robô.

DATA

19 OUTUBRO 2018

LOCALIDADE

SENAI CIMATEC
SALVADOR - BAHIA

Mandruvah team

Cleber
Davi
Ícaro
Carlos

10:40

O robô foi levado até o estacionamento do SENAI CIMATEC onde há uma linha de transmissão de testes. O setup foi montado com uma bateria automotiva, um inversor de frequência 12 - 110V para conectar a fonte da NUC e os motores ligados diretamente à bateria.

11:05

Com o robô na linha e a NUC ligada e conectada a uma rede local exclusiva do teste, conectamos um notebook com o robô via SSH e assim tivemos acesso remoto ao terminal do robô.

11:10

Ao tentar carregar o `controller_manager.launch` nenhum motor foi encontrado. Conferimos então todas as conexões dos hubs de comunicação, nenhum mal contato foi encontrado. Percebemos que um dos terminais do conector do conversor USB-RS485 estava solto da placa.

13:20

Corrigimos o terminal solto do conversor. Novamente carregamos o `controller_manager.launch`, aconteceram erros de “*checksum*” e de “*wrong packet prefix*” e apenas 4 dos 18 motores foram encontrados.

13:40

Após os erros de comunicação, verificamos que quando corrigimos o conversor, os fios D+ e D- foram trocados acidentalmente.

13:45

Ainda depois de corrigir a conexão invertida, não conseguimos encontrar todos os motores, foi levantada a hipótese de que a bateria automotiva era o problema. Medimos a tensão e estava em 12,15V, decidimos voltar ao laboratório e testar com a fonte de bancada.

14:20

Repetimos o teste alimentando o robô com a fonte de bancada e o teste correu bem, nenhum dos erros anteriores aconteceu.

Lista de componentes

ELIR project - BILL OF MATERIAL

Desenvolvimento do robô de inspeção.

Carlos Alberto Pereira
Cleber Couto Filho
Davi Costa
Ícaro Nascimento

Salvador, Setembro de 2018.