

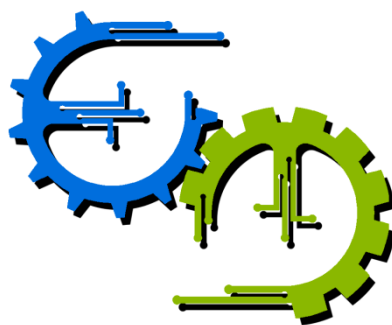
TRABALHO DE GRADUAÇÃO

**DESENVOLVIMENTO DE UM ROBÔ
MANIPULADOR SCARA**

Por,

Otavio Pellicano Moreira de Mello

Brasília, Julho de 2016



**ENGENHARIA
MECATRÔNICA**
UNIVERSIDADE DE BRASÍLIA

TRABALHO DE GRADUAÇÃO

**DESENVOLVIMENTO DE UM ROBÔ
MANIPULADOR SCARA**

Otávio Pellicano Moreira de Mello

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Prof. Walter de Britto Vidal Fiho , UnB/ENM
(Orientador)

Walter de Britto

Prof. Adolfo Bauchspiess, UnB/ENE

Bauchspiess

Prof. Guilherme Caribé de Carvalho, UnB/ENM

Guilherme Caribé

Brasília, Julho de 2016

FICHA CATALOGRÁFICA

PM527d Pellicano Moreira de Mello, Otavio
Desenvolvimento de um Robô Manipulador SCARA /
Otavio Pellicano Moreira de Mello; orientador Walter
de Britto Vidal Filho. -- Brasília, 2016.
122 p.

Monografia (Graduação - Engenharia de Controle e
Automação) -- Universidade de Brasília, 2016.

1. Robô Manipulador SCARA. 2. Projeto Mecânico. 3.
Projeto de Software. 4. Controle de Malha Aberta. I.
de Britto Vidal Filho, Walter, orient. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

MELLO, O. P. M. , (2016). Título do trabalho. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 01/2016 , Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 122p.

CESSÃO DE DIREITOS

AUTOR: Otavio Pellicano Moreira de Mello.

TÍTULO DO TRABALHO DE GRADUAÇÃO: Desenvolvimento de um Robô Manipulador SCARA.

GRAU: Engenheiro de Controle e Automação

ANO: 2016

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito do autor.

Otavio Pellicano Moreira de Mello
UnB - Faculdade de Tecnologia, Campus Universitário Darcy Ribeiro.
70910-900 Brasília – DF – Brasil.

AGRADECIMENTOS

Agradeço a todos envolvidos direto e indiretamente em minha formação acadêmica.

Otávio Pellicano Moreira de Mello.

RESUMO

O estudo da robótica de manipuladores é importante para suprir a demanda do mercado nacional de mão-de-obra especializada. O emprego de robôs manipuladores didáticos no ensino permite um maior aprendizado. Este trabalho objetiva desenvolver um robô manipulador para fins didáticos. O trabalho apresenta o projeto mecatrônico, ou seja, o projeto mecânico, eletroeletrônico e computacional de um robô manipulador SCARA de 3 graus de liberdade. No projeto mecânico calcula-se a geometria das barras da estrutura pela equação da linha elástica, do cálculo das tensões máximas e das deflexões máxima dos elementos da estrutura. Apresenta-se, também, o cálculo dinâmico do manipulador, possibilitando o cálculo dos torques nas juntas. No projeto eletroeletrônico, selecionam-se as placas de acionamento e os atuadores do sistema. Os softwares de controle e acionamento são desenvolvidos detalhadamente, ao longo do projeto computacional. Por fim, constrói-se um protótipo para que testes de funcionamento sejam feitos, validando todo o projeto mecatrônico.

Palavras Chave: Manipulador SCARA, Projeto Mecânico, Projeto Computacional, Controle Malha Aberta.

ABSTRACT

The study of robotic manipulators is important to meet the demand of the domestic market for skilled labor. The use of robot manipulators didactic teaching allows for greater learning. This study aims to develop a robot manipulator for teaching purposes. This work presents the mechatronic design, that is, mechanical, electronics and computer design of a 3 degrees of freedom SCARA robot manipulator. In mechanical design calculates the geometry of the bars of the structure by the equation of the elastic line, the calculation of the maximum stresses and maximum deflection of the structure elements. It presents also the dynamic calculation of the manipulator, allowing the calculation of torques joints. In electronics design, select up the drive plates and the system actuators. The drive control software are developed in detail, along the computational project. Finally, build up a prototype for that function tests are done, validating all the mechatronic design.

Keywords: SCARA Manipulator; Mechanic Project; Software Design; Open Loop Control.

SUMÁRIO

1	INTRODUÇÃO	1
2	REVISÃO BIBLIOGRÁFICA	3
2.1	Definição de Robôs Manipuladores	3
2.1.1	Estrutura Básica de um Robô	4
2.1.2	Robô SCARA	5
2.1.3	Especificações do Robô	6
2.2	Análise Estática	6
2.2.1	Tensões Normais e de Cisalhamentos Máximos	7
2.2.2	Equação de Linha Elástica	9
2.3	Análise Dinâmica	10
2.3.1	Mecânica Lagrangiana	11
2.4	Redes Neurais Artificiais	12
2.4.1	Perceptron Multicamadas	13
2.5	Cinemática de Robôs	15
2.5.1	Cinemática Direta	16
2.5.2	Cinemática Inversa	17
2.5.3	Cinemática Inversa por RNA	17
3	PROJETO MECÂNICO	20
3.1	Cálculos Estáticos	20
3.2	Cálculos Dinâmicos	26
3.3	Desenho e Processo de Fabricação	34
3.3.1	Processo de Fabricação do Manipulador	37
4	PROJETO ELETROELETRÔNICO	43
5	PROJETO DE SOFTWARE	46
5.1	Software de Controle do Manipulador	46
5.1.1	Interpolação Linear	50
5.1.2	Cinemática Inversa	50
5.1.3	Menor Caminho	51

5.1.4	Thetas para Passos.....	52
5.1.5	Definição de Velocidades	52
5.1.6	Gerar Comando Serial.....	53
5.1.7	Enviar Comando Serial.....	53
5.1.8	Atualizar X Atual, Y Atual e Z Atual.....	54
5.2	Software de Acionamento dos Motores	54
5.2.1	Tratar Comando Serial	55
5.2.2	Enviar Passos e Direção para o motor.....	56
5.2.3	Avisar o Término do Processamento	56
6	TESTES DE FUNCIONAMENTO	57
6.1	Repetibilidade do Manipulador	57
6.2	Interpolação Linear	59
7	CONCLUSÃO	61
	REFERÊNCIAS BIBLIOGRÁFICAS.....	62
8	ANEXOS	64
8.1	Desenhos Técnicos do Manipulador	64
8.1.1	Anel Fixação.....	64
8.1.2	Cabine	65
8.1.3	Camisa Engaste	65
8.1.4	Eixo Elo 1	66
8.1.5	Eixo Elo 2	66
8.1.6	Elo 1	67
8.1.7	Elo 2	67
8.1.8	Engaste Elo 1	68
8.1.9	Espaçador	68
8.1.10	Fixação Eixo 1 e 2	69
8.1.11	Fixação Haste.....	69
8.1.12	Haste de Sustentação	70
8.1.13	Mancal.....	70

8.1.14	Orelhas.....	71
8.1.15	Suporte Engaste 1.....	71
8.1.16	Suporte Motor 1 e 2.....	72
8.1.17	Suporte Motor 3.....	72
8.2	Software de Acionamento Arduino 1.....	73
8.3	Software de Acionamento Arduino 2.....	78
8.4	Software de Controle.....	85
8.4.1	Interpolação Linear.....	85
8.4.2	Cinemática Inversa.....	88
8.4.3	Menor Caminho.....	90
8.4.4	Thetas para Passos.....	92
8.4.5	Definição de Velocidades.....	94
8.4.6	Gerar Comando Serial.....	96
8.4.7	Enviar Comando Serial.....	98
8.4.8	Volume de Trabalho.....	99
8.4.9	Globais.....	100
8.4.10	Conversões Diversas.....	102

LISTA DE FIGURAS

Figura 2.1 Volume de trabalho de um robô SCARA [3]	5
Figura 2.2 Exemplo esquemático de um robô SCARA [5]	6
Figura 2.3 Tensão normal máxima em uma viga engastada de seção transversal retangular	7
Figura 2.4 Tensão de cisalhamento máxima em uma viga engastada de seção transversal retangular	8
Figura 2.5 Deflexão de uma viga em balanço com carregamento distribuído [7].....	9
Figura 2.6 Estrutura básica do Perceptron.....	13
Figura 2.7 Função Lógica AND de duas entradas.....	15
Figura 2.8 Função Lógica XOR de duas entradas	15
Figura 2.9 Modelo de um robô SCARA planar [1]	16
Figura 2.10 Pontos de treinamento da rede	18
Figura 2.11 Pontos de teste aplicados a rede	19
Figura 3.1 Modelo estático do primeiro e segundo elo no plano x-z.....	20
Figura 3.2 Modelo estático do primeiro e segundo elo no plano y-z.....	21
Figura 3.3 Diagrama de corpo livre	21
Figura 3.4 Localização da tensão máxima normal e de cisalhamento na viga engastada	24
Figura 3.5 Modelo SCARA para o cálculo da dinâmica.....	26
Figura 3.6 Coordenadas do ponto D.....	29
Figura 3.7 Modelo no plano x-z.....	31
Figura 3.8 Curvas de Torque	34
Figura 3.9 Vista isométrica do manipulador	36
Figura 3.10 Maquete do manipulador	36
Figura 3.11 Estrutura mecânica do manipulador.....	37
Figura 3.12 Início do processo de fabricação.....	38
Figura 3.13 Fixações dos eixos	38
Figura 3.14 Posicionamento das fixações.....	39

Figura 3.15 Conjunto mancal/rolamento	39
Figura 3.16 Acoplamento flexível e anel de sustentação	40
Figura 3.17 Junta prismática do manipulador	41
Figura 3.18 Suporte para canetas.....	42
Figura 3.19 Manipulador construído.....	42
Figura 4.1 Módulo driver motor com dupla ponte H - L298N [12]	44
Figura 4.2 Circuito Integrado L298 [11].....	44
Figura 4.3 Fluxograma de Controle do Sistema	45
Figura 5.1 Interface do manipulador	47
Figura 5.2 Fluxograma do programa de controle	47
Figura 5.3 Arduino 1 e 2 desconectados.....	48
Figura 5.4 Formatação do arquivo de entrada da interface	48
Figura 5.5 Fluxograma do botão: "Pontos Individuais"	49
Figura 5.6 Fluxograma do botão: "Conjunto de Pontos".....	50
Figura 5.7 Determinação do menor caminho	51
Figura 5.8 Janela de pontos fora do volume de trabalho.....	52
Figura 5.9 Erro ao escrever no arduino 2.....	54
Figura 5.10 Fluxograma da implementação de cada arduino.....	55
Figura 6.1 Pontos para avaliar a repetibilidade	57
Figura 6.2 Repetibilidade do manipulador.....	58
Figura 6.3 Detalhe da repetibilidade do manipulador	58
Figura 6.4 Trajetória, (0,22) a (0,28), gerada pelo software de controle	59
Figura 6.5 Trajetória, (0,22) a (0,28), gerada pelo manipulador	59

LISTA DE TABELAS

Tabela 2.1 Melhores respostas da rede para 2002 pontos de teste	19
Tabela 3.1 Substituição de variáveis estáticas.....	25
Tabela 3.2 Tensões normal e de cisalhamento máximos.....	26
Tabela 3.3 Substituição de variáveis dinâmicas.....	33
Tabela 3.4 Torques máximos.....	33
Tabela 3.5 Alcance do manipulador.....	34
Tabela 3.6 Componentes do Manipulador	35

LISTA DE SÍMBOLOS

Símbolos Latinos

x_i	Posição x do plano cartesiano de índice i	[m]
y_i	Posição y do plano cartesiano de índice i	[m]
z_i	Posição z do plano cartesiano de índice i	[m]
l_i	Comprimento do elo i do manipulador	[m]
$ M _{max}$	Módulo do momento fletor máximo	[N.m]
c	Distância entre o eixo neutro a fibra mais afastada desse eixo	[m]
I	Momento de inércia da seção transversal em relação a linha neutra	[m ⁴]
Q	Momento estático da área localizada acima ou abaixo da linha neutra	[m ³]
t	Largura da seção transversal na linha neutra	[m]
M	Momento fletor	[N.m]
E	Módulo de elasticidade	[Pa]
L	Lagrangiano	[kg.m ² /s ²]
K	Energia cinética	[kg.m ² /s ²]
Pot	Energia potencial	[kg.m ² /s ²]
T_i	Torque de índice i	[N.m]
F_i	Força de índice i	[N]
q_i	Carregamento distribuído de índice i	[kg/m]
P	Peso	[N]
h_i	Altura externa da seção transversal de índice i	[m]
\tilde{h}_i	Altura interna da seção transversal de índice i	[m]
b_i	Base externa da seção transversal de índice i	[m]
\tilde{b}_i	Base interna da seção transversal de índice i	[m]
I_{y_i}	Momento de área em relação ao eixo y de índice i	[m ⁴]
r_1	Distância entre a primeira junta ao centro de massa do elo 1	[m]

r_2	Distância entre a segunda junta ao centro de massa do elo 2	[m]
I_B	Momento de inércia do primeiro elo em relação ao centro de massa	[m ⁴]
I_D	Momento de inércia do segundo elo em relação ao centro de massa	[m ⁴]
m_i	Massa de índice i	[kg]
v_B	Velocidade linear do centro de massa do primeiro elo	[m/s]
v_D	Velocidade linear do centro de massa do segundo elo	[m/s]
g	Aceleração gravitacional	[m/s ²]

Símbolos Gregos

σ_m	Tensão normal ao longo do eixo x	[Pa]
τ_m	Tensão de cisalhamento máxima	[Pa]
θ_i	Deslocamento angular da junta i do manipulador	[rad]
$\dot{\theta}_i$	Velocidade angular da junta i do manipulador	[rad/s]
$\ddot{\theta}_i$	Aceleração angular da junta i do manipulador	[rad/s ²]

Grupos Adimensionais

$g(.)$	Função de ativação
--------	--------------------

Subscritos

max	Máximo
-------	--------

Sobrescritos

- Primeira derivada em relação ao tempo
- Segunda derivada em relação ao tempo
- Representação vetorial

Siglas

RNA	Rede Neural Artificial
PMC	Perceptron Multicamadas
SCARA	Selective Compliant Assembly Robot Arm
JIRA	Japanese Industrial Robot Association
RIA	Robot of Institute of America

1 INTRODUÇÃO

O estudo da robótica de manipuladores é importante para suprir a demanda do mercado nacional de mão-de-obra especializada. Os robôs estão cada vez mais presentes na indústria de hoje. São utilizados para realizar tarefas que exigem precisão e podem trabalhar, com as manutenções adequadas, por longos períodos sem a necessidade de elementos de conforto requerido aos seres humanos. Trabalhos repetitivos, que geram bastante estresse a quem executa, são facilmente executados por robôs manipuladores. Pela aplicabilidade desses manipuladores na indústria, o emprego de robôs manipuladores didáticos no ensino permite um maior aprendizado.

Este trabalho objetiva desenvolver um robô manipulador para fins didáticos. Para tanto, apresentará o projeto mecatrônico (mecânico, eletroeletrônico e computacional) de um robô SCARA de 3 graus de liberdade.

O projeto mecânico será dividido em duas análises, estática e dinâmica. Na estática, os conceitos de resistências dos materiais como a tensão normal, de cisalhamento, momento fletor e equação da linha elástica serão aplicados. Para a dinâmica, serão utilizados conceitos da Mecânica de Lagrange na determinação dos torques exigidos das juntas rotacionais e a força necessária na junta prismática do robô manipulador SCARA.

O projeto eletroeletrônico consiste na seleção de uma placa de potência adequada aos atuadores que serão definidos no projeto mecânico, assim como, a utilização de microcontroladores que enviarão sinais de controle para o sistema.

Para o projeto computacional, um software de controle (desenvolvido em C++) e o software de acionamento (embarcado nos microcontroladores) serão minuciosamente detalhados. Conceitos de cinemática direta e inversa, interpolação linear e controle de velocidade serão implementados no software de controle.

O trabalho pode ser dividido em:

- **Revisão Bibliográfica:** serão abordadas as referências necessárias ao acompanhamento do trabalho;
- **Projeto Mecânico:** cálculo dos esforços exigidos pela estrutura mecânica e a definição de componentes que se adequem a esses esforços;

- **Projeto Eletroeletrônico:** seleção de placas de potência para que os controladores possam ser efetivos no envio de seus sinais de controle;
- **Projeto Computacional:** elaboração dos softwares de controle e acionamento do robô.
- **Desenvolvimento do Protótipo:** com a definição de todos os componentes, através das etapas subsequentes, será construído um protótipo que atenda as definições do projeto.
- **Testes de Funcionamento:** Vários testes serão feitos para avaliar o funcionamento do manipulador.

2 REVISÃO BIBLIOGRÁFICA

2.1 Definição de Robôs Manipuladores

O conceito de robô manipulador, ainda hoje, não está bem definido, apresentando diferentes conotações em diferentes regiões do mundo. Por exemplo, a RIA (Robot of Institute of America) define um robô como um manipulador reprogramável, multifuncional projetado para mover material, partes, ferramentas, ou peças especializadas através de um algoritmo reprogramável para executar uma variedade de tarefas (conforme citado em [1]).

Já os japoneses, através da JIRA (Japanese Industrial Robot Association), classificam os robôs em 6 classes [2]:

- Dispositivo de Movimentação Manual: é um dispositivo comandado manualmente por um operador (teleoperação);
- Robô de Sequência Fixa: um dispositivo programado com um determinado método que executa etapas sucessivas de tarefas sem a possibilidade, ou com muita dificuldade, da alteração do método;
- Robô de Sequência Variável: segue o princípio do robô de sequência fixa, porém fácil de modificar;
- Robô de Reprodução: um operador humano realiza passo a passo o caminho em que robô deve seguir, gravando as informações para que o robô reproduza o trajeto;
- Robô de Controle Numérico: As operações a serem realizadas pelo robô são inseridas através de um programa com as movimentações desejadas;
- Robô Inteligente: um robô com a capacidade de compreender seu ambiente para concluir suas tarefas com êxito, apesar de possíveis alterações no ambiente em que a tarefa deverá ser executada.

Relacionando a definição americana com a japonesa, percebem-se divergências. A RIA não considera como um robô o “Dispositivo de Movimentação Manual” e o “Robô de Sequência Fixa” pelo fato de o primeiro ser acionado por um operador (esse tipo de dispositivo é classificado, pelos americanos, como manipulador [2]) e, o segundo, ser extremamente específico, não havendo assim (ou se houver, mínima), a possibilidade de alteração da tarefa originalmente programada.

2.1.1 Estrutura Básica de um Robô

Empregando a definição de um robô manipulador da RIA, esses manipuladores apresentam estruturas básicas que devem ser consideradas ao se projetar um robô. São elas [2]:

- Manipulador: representa o corpo principal do robô, que constitui nos elementos estruturais do robô.
- Órgão terminal: é a ferramenta utilizada para realizar um trabalho específico (como um maçarico de solda, uma pistola de pintura, uma garra, entre outros), ligado à articulação final do robô.
- Atuadores: funcionam como os músculos do robô. O controlador envia os sinais aos atuadores para que eles possam movimentar uma ou mais articulações. Exemplos de atuadores empregados: servomotor, motor de passo, atuador pneumático, atuador hidráulico. A escolha do atuador depende da aplicação pretendida.
- Sensores: são utilizados para obtenção de dados do ambiente, ou do próprio manipulador, em que o robô está inserido, como possíveis obstáculos, ou a posição das juntas.
- Controlador: é utilizado para controlar a movimentação do robô. O controlador recebe informações do computador, controlando os atuadores e coordenando os movimentos, através de uma realimentação dos sensores envolvidos.
- Processador: usado para calcular os movimentos do robô, definindo o movimento das articulações, assim como o local e a velocidade desejada. O processador supervisiona o controlador e os sensores. Ele é, normalmente, um computador dedicado a essa finalidade.
- Software: pode ser dividido em três grupos: um sistema operacional que opera o computador; um que calcula a movimentação com base na cinemática do robô; e o terceiro grupo é o conjunto de rotinas definidas pelo usuário para que o robô possa realizar uma tarefa específica

O manipulador é composto de elos conectados com juntas, definindo a geometria e a cinemática do robô. As juntas são tipicamente de revolução (para movimento de rotação), ou prismáticas (para movimentos lineares) [1]. Os tipos de juntas, assim como a geometria, é uma das maneiras utilizadas na classificação de robôs, referenciado a partir da base até a última articulação (não levando em consideração o órgão terminal). Alguns exemplos de robôs classificados dessa maneira são: articulados (RRR); esféricos (RRP), cilíndricos (RPP), cartesianos (PPP), SCARA (RRP).

Todos os manipuladores possuem uma região alcançável, limitado pela sua configuração. Essa região alcançável é o volume de trabalho do robô (Figura 2.1 apresenta um exemplo de volume de trabalho para um robô SCARA). O volume de trabalho pode ser determinado matematicamente, através das equações que definem as ligações e articulações do robô, ou experimentalmente, movendo cada articulação ao longo de sua excursão. [2].

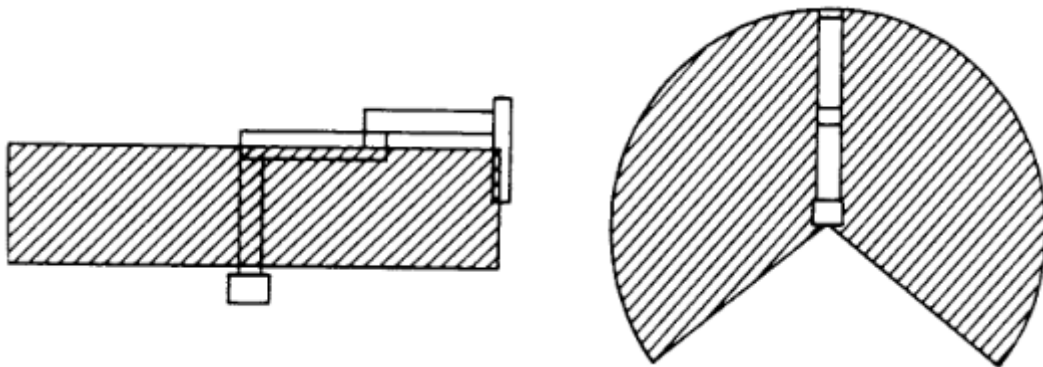


Figura 2.1 Volume de trabalho de um robô SCARA [3]

2.1.2 Robô SCARA

O robô SCARA (*Selective Compliant Assembly Robot Arm*) foi desenvolvido no Japão em 1979 pela Universidade Yamanashi, para uma demanda específica da indústria. Pelo sucesso de sua configuração, deixou de ser uma configuração especial [4].

Robôs SCARA possuem duas ou três articulações de revolução que são paralelas, permitindo que o robô se mova em um plano horizontal, além de articulações prismáticas, permitindo o movimento vertical [2]. Esse tipo de configuração torna os adequados em processos de montagem. Eles são mais compatíveis no plano x-y, mas são muito rígidos no plano z, proporcionando complacência seletiva (uma questão importante na montagem) [2]. A *Figura 2.2* apresenta um exemplo esquemático de um robô SCARA.

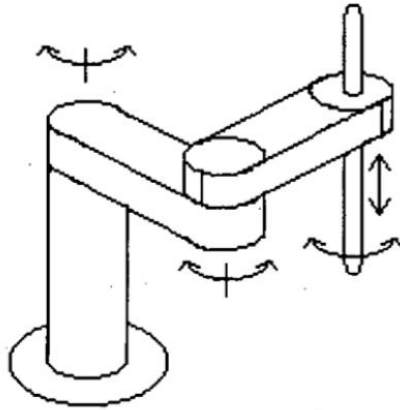


Figura 2.2 Exemplo esquemático de um robô SCARA [5]

2.1.3 Especificações do Robô

Algumas especificações são essenciais para caracterizar um robô. São elas: carga útil; alcance; precisão e repetibilidade.

- Carga Útil: é o peso em que o robô pode carregar sem comprometer o Alcance, Precisão e a repetibilidade [2].
- Alcance: é a distância máxima em que um robô pode alcançar dentro de um volume de trabalho [2].
- Precisão: é a capacidade de um robô de posicionar seu punho ou o órgão terminal em um ponto desejado [6].
- Repetibilidade: é a precisão em que a mesma posição pode ser alcançada diversas vezes pelo robô [2].

2.2 Análise Estática

A análise estática do robô é necessária para que se possa selecionar o tipo de material e geometria mais adequado para o sistema. Nessa análise são calculados esforços como tensão normal e de cisalhamento, deflexão da viga, fadiga, entre outros. No presente trabalho apenas foram considerados para o projeto do robô SCARA as tensões normal e de cisalhamento máximas e a deflexão máxima, aceitáveis para o projeto. Com o cálculo dessas variáveis, pode-se selecionar o material (em conjunto com a geometria) que atenda as especificações do projeto.

2.2.1 Tensões Normais e de Cisalhamentos Máximos

Uma viga de seção prismática submetida a um carregamento transversal, pontual ou distribuído, ou momentos fletores, apresentam tensões normais e de cisalhamento que podem variar de intensidade ao longo da viga.

No projeto de uma viga, os máximos dessas tensões são utilizados para se determinar o material e a geometria que serão utilizados. Essas tensões máximas não podem ultrapassar as tensões admissíveis do projeto.

A tensão normal ao longo do eixo x , σ_x , em uma seção varia linearmente com a distância y do eixo neutro (onde σ_x é nulo, também conhecida como linha neutra) e o maior valor de tensão ocorre nas fibras mais afastadas desse eixo neutro [7]. A tensão normal máxima (σ_m) pode ser expressa como segue:

$$\sigma_m = \frac{|M|_{max} c}{I} \quad (1)$$

em que $|M|_{max}$ é o módulo do momento fletor máximo, c é a distância entre o eixo neutro a fibra mais afastada desse eixo, I é o momento de inércia da seção transversal em relação à linha neutra. A Figura 2.3 mostra a localização desse σ_m para uma viga prismática engastada de seção transversal retangular com carregamento distribuído e pontual, ilustrando a tração e a compressão.

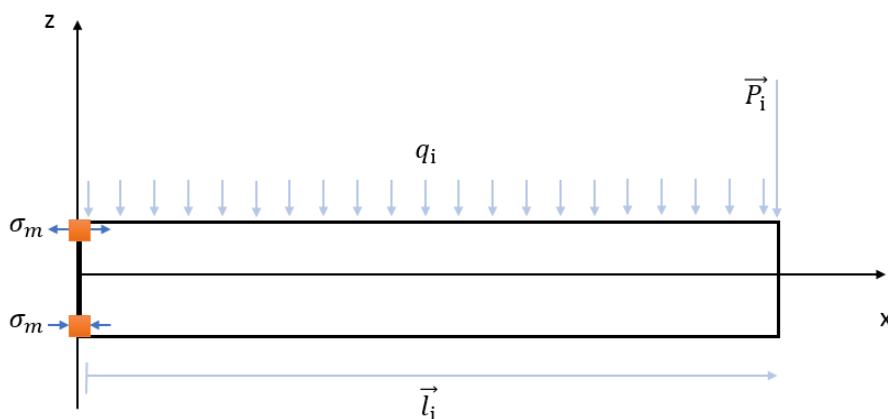


Figura 2.3 Tensão normal máxima em uma viga engastada de seção transversal retangular

A tensão de cisalhamento, para uma seção retangular, é máxima na linha neutra [7], podendo ser expressa como

$$\tau_m = \frac{|V|_{max} Q}{It} \quad (2)$$

onde τ_m é a tensão de cisalhamento máxima, $|V|_{max}$ é o módulo da força de cisalhamento máximo, t é a largura da seção transversal na linha neutra, I é o momento de inércia da seção transversal em relação ao eixo neutro, Q é o momento estático da área localizada acima ou abaixo da linha neutra. Esse momento estático é representado pela equação:

$$Q = \int_0^c y dA \quad (3)$$

A Figura 2.4 mostra a localização da tensão de cisalhamento máxima em uma viga prismática engastada de seção transversal retangular com carregamento distribuído e pontual.

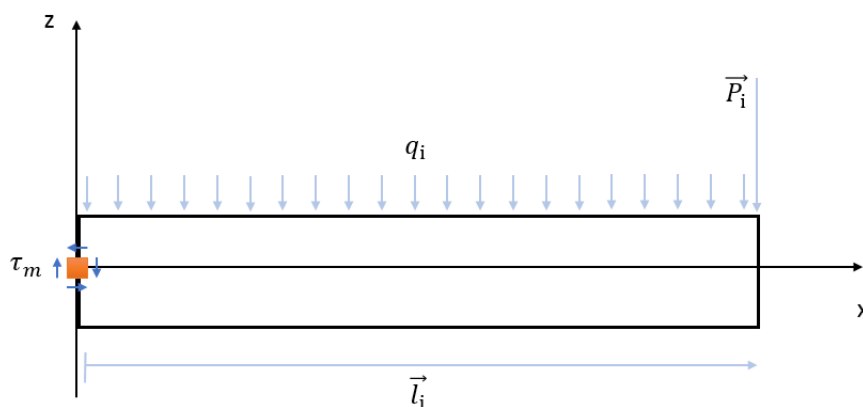


Figura 2.4 Tensão de cisalhamento máxima em uma viga engastada de seção transversal retangular

Como o I depende da geometria da seção transversal que, no caso de uma seção retangular, depende da base e da altura, pode-se determinar os valores da base e da altura de tal forma que a tensão normal máxima e de cisalhamento máxima não ultrapassem a tensão normal e de cisalhamento admissível.

Como nessa abordagem não é levada em consideração a deflexão devido aos esforços submetidos à viga. Tem-se então a necessidade de uma formulação adicional que se leve em conta essa deflexão: a equação da linha elástica, que será abordada a seguir.

2.2.2 Equação de Linha Elástica

Em muitos projetos a determinação da deflexão da estrutura mecânica é crucial para que o sistema se comporte como o esperado. A deflexão máxima exprime a máxima distância entre a linha neutra antes e depois da aplicação de carregamentos na estrutura. A Figura 2.5 ilustra a variação dessa distância ao longo do comprimento de uma viga.

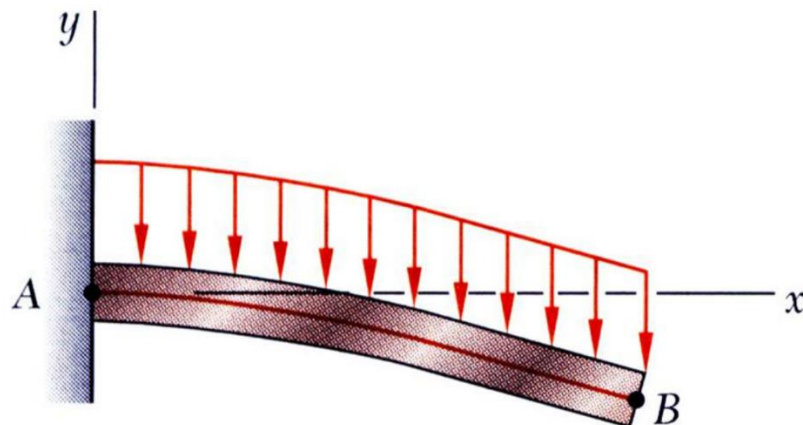


Figura 2.5 Deflexão de uma viga em balanço com carregamento distribuído [7]

A equação que define a deflexão de uma viga em um determinado ponto, é denominada como equação diferencial da linha elástica [7] (ou simplesmente equação da linha elástica) e que dentro do regime elástico pode ser expressa como

$$\frac{d^2y}{dx^2} = \frac{M(x)}{EI} \quad (4)$$

onde $M(x)$ é o momento fletor ao longo do eixo x , E é o módulo de elasticidade e I é o momento de inércia da seção transversal em relação a linha neutra.

Ao se integrar duas vezes a equação da linha elástica, determina-se a deflexão y em qualquer ponto da viga. As duas constantes de integração, oriundas das integrações sucessivas, são determinadas pelas condições de contorno. Para uma viga em balanço (ilustrada na Figura 2.5) essas condições de contorno são:

$$\left. \frac{dy}{dx} \right|_A = 0 \quad (5)$$

$$y|_A = 0 \quad (6)$$

Analisando a máxima variação de y , pode-se determinar a máxima deflexão de uma viga. E, assim, determinar o material e a geometria a ser utilizado para que seja cumprido o requisito de máximo valor admissível de deflexão do projeto.

2.3 Análise Dinâmica

A análise dinâmica do robô se faz necessária para que se possa selecionar os atuadores mais adequados para o sistema. Nessa análise, são calculados os esforços para que o sistema se comporte como esperado, levando em consideração velocidade e aceleração exigidas no projeto.

A análise dos esforços pode ser feita utilizando a Mecânica Newtoniana em que, tomando como ponto de partida a segunda lei de Newton, calculam-se os esforços necessários para as restrições do projeto (aceleração e velocidade). Porém, a medida que a complexidade do robô aumenta, a Mecânica Newtoniana se torna dispendiosa, tornando-se mais simples a utilização de outra ferramenta para a modelagem do sistema. Assim, a maior parte das referências utilizadas em robótica fazem o modelamento através da Mecânica de Lagrange (ou Mecânica Lagrangiana) [2].

2.3.1 Mecânica Lagrangiana

A Mecânica Lagrangiana utiliza das ferramentas do Cálculo Variacional junto ao princípio de Hamilton para obter um método de cálculo dos esforços que dependa das energias cinéticas e potenciais envolvidas no sistema.

O princípio de Hamilton (também chamado de mínima ação) é um dos mais abrangentes princípios da Física, estabelecendo que a cada teoria clássica está associado um certo funcional S , denominado ação, que satisfaz a equação abaixo [8].

$$\delta S = 0 \tag{7}$$

Euler, utilizando o princípio de Hamilton, obteve em 1.744 a relação

$$\frac{\partial F}{\partial y} - \frac{d}{dx} \frac{\partial F}{\partial y'} = 0 \tag{8}$$

conhecida como equação de Euler-Lagrange que representa a condição de extremo para um funcional S (para um sistema sem vínculos, como por exemplo dois pontos ligados a uma haste), em que $F = F(y(x), y'(x), x)$ e a notação $y'(x)$ é definido como se segue:

$$y'(x) = \frac{dy}{dx} \tag{9}$$

Ao se adicionarem vínculos ao sistema a equação de Euler-Lagrange pode ser expressa como

$$\frac{\partial}{\partial y} (F + \lambda G) - \frac{d}{dx} \frac{\partial}{\partial y'} (F + \lambda G) = 0 \tag{10}$$

em que λ é uma quantidade livre chamada de multiplicador de Lagrange e $G = G(y(x), y'(x), x)$ está associado aos vínculos do sistema. A dedução completa da equação de Euler-Lagrange pode ser estudada na referência [8].

Como os robôs manipuladores apresentam vínculos (elos que ligam as juntas), utiliza-se a equação de Euler-Lagrange com vínculos para determinar os conjugados (juntas rotacionais) e forças (juntas prismáticas) através do lagrangiano para o sistema:

$$L = K - Pot \quad (11)$$

em que L é o lagrangiano, K e Pot são as energias cinéticas e potenciais do sistema, respectivamente. A equação que define os conjugados e as forças, deduzidas através da equação de Euler-Lagrange com vínculos, são definidas como [2]

$$T_i = \frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{\theta}_i} \right) - \frac{\partial L}{\partial \theta_i} \quad (12)$$

$$F_i = \frac{\partial}{\partial t} \left(\frac{\partial L}{\partial \dot{x}_i} \right) - \frac{\partial L}{\partial x_i} \quad (13)$$

em que T_i e F_i são os somatórios de todos os conjugados e forças externas para o movimento de rotação e translação, respectivamente e, θ_i e x_i , são as variáveis do sistema.

A aplicação desse método ficará clara quando o mesmo for utilizado na modelagem dinâmica do robô proposto por este trabalho.

2.4 Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) são modelos matemáticos que visam simular a capacidade de aprendizagem das redes de neurônios. Elas possuem a habilidade de reconhecer padrões, através de um treinamento para um fim específico. Esse treinamento é feito estimulando a rede com os padrões requeridos que a rede aprenda.

Os padrões não precisam ser lineares, podendo assim serem utilizadas em problemas mais complexos em que a não-linearidade do sistema não possa ser desconsiderada.

A RNA não se limita a apenas em aprender os padrões utilizados no treinamento. Após o treinamento da rede, ela é capaz de generalizar o conhecimento adquirido, possibilitando estimar soluções que eram até então desconhecidas [9]. Pode-se assim, utilizá-las como aproximadores universais.

A característica das RNAs de aproximadores universais a tornam propícias na aplicação do cálculo da cinemática inversa de um robô manipulador, utilizando a cinemática direta para a elaboração de padrões de treinamento. Obtendo assim a cinemática inversa do robô dependendo apenas da cinemática direta

2.4.1 Perceptron Multicamadas

O *Perceptron* Multicamadas (PMC) é formado a partir de *Perceptrons* com pelo menos uma camada intermediária entre os neurônios de entrada e saída. Esses *Perceptrons* são as unidades fundamentais dessa rede neural (o neurônio individual). A Figura 2.6 apresenta a estrutura básica de um *Perceptron*.

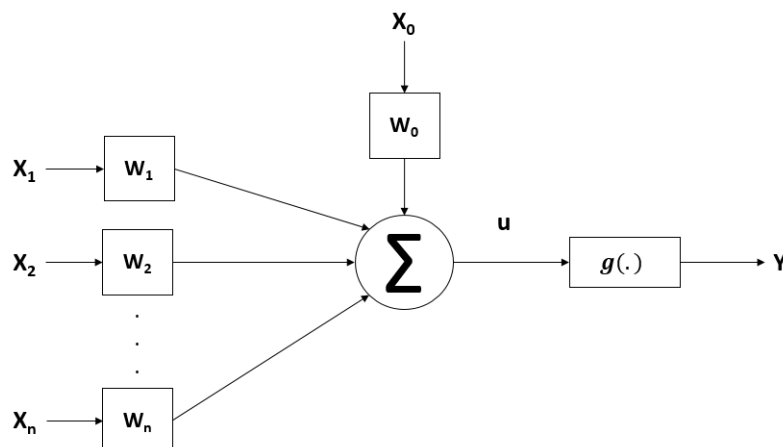


Figura 2.6 Estrutura básica do Perceptron

A configuração do *Perceptron* visa simular o funcionamento de um neurônio. Nele os sinais seguem apenas um sentido analogamente aos neurônios: dendritos, corpo celular, axônios.

Para que um neurônio (binário) possa ser ativado, existe um potencial elétrico que deve ser superado, um limiar de ativação. Esse limiar é representado na Figura 2.6 como W_0 (bias), que é multiplicado por X_0 possuindo um valor resultante negativo (X_0 pode ser representado com o valor de 1 ou -1 [9]). Os valores de X_1 a X_n representam as entradas do neurônio e W_1 a W_n os peso sinápticos correspondentes. $g(.)$ é uma

função de ativação que transforma o resultado do somatório em uma saída binária ou valores correspondentes em uma determinada curva (determinado pela função de ativação), Y . A Figura 2.6 pode ser representada em forma de equação da seguinte forma:

$$u = \sum_{i=1}^n W_i X_i - W_0 \quad (14)$$

$$Y = g(u) \quad (15)$$

O princípio de funcionamento do *Perceptron* é muito simples. Primeiro há uma fase de treinamento em que são inicializados pesos sinápticos (e o *bias*) aleatoriamente. Se a saída gerada pelo *Perceptron* não for a saída desejada, os pesos sinápticos e o limiar de ativação são incrementados (conhecido como ajuste excitatório), ou decrementados (ajuste inibitório) [9]. O processo é repetido até que a saída seja a saída desejada.

O *Perceptron* possui uma grande limitação: ele apenas consegue classificar padrões cuja a função é dividir classes linearmente separáveis [9]. Ou seja, uma função lógica simples como XOR (ou exclusivo) não seria possível obter a resposta desejada através do *Perceptron*. A Figura 2.7 e a Figura 2.8 ilustram duas funções lógicas (AND e XOR) com o intuito de demonstrar a diferença entre classes linearmente e não linearmente separáveis, respectivamente.

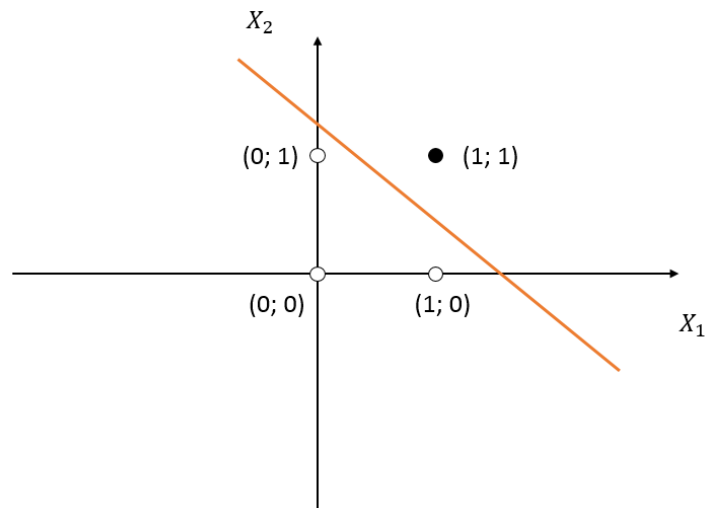


Figura 2.7 Função Lógica AND de duas entradas

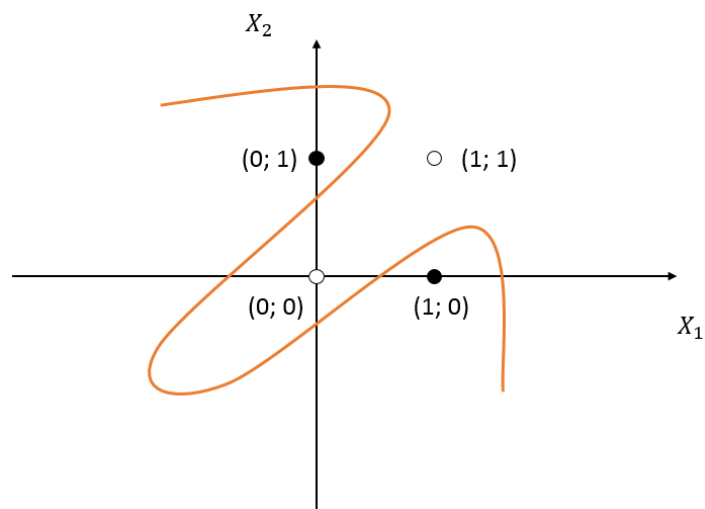


Figura 2.8 Função Lógica XOR de duas entradas

O *Perceptron* Multicamadas não possui essa limitação de só poder classificar padrões linearmente separáveis, permitindo uma ampla utilização desse tipo de estrutura em diversas áreas da ciência, inclusive na robótica.

2.5 Cinemática de Robôs

Para um robô em que o tamanho dos elos e os ângulos articulares são variáveis conhecidas, é possível determinar a posição do efetuador no espaço através das posições das juntas (cinemática direta) ou a posição das juntas através da posição no espaço (cinemática inversa).

Na cinemática direta, é necessário desenvolver um conjunto de equações que se relacionem com a configuração do robô de tal forma que, ao substituir as variáveis articulares e de elos nessas equações, calcula-se a posição e a orientação do efetuador [2]. Pode-se derivar a equação da cinemática inversa através da cinemática direta.

2.5.1 Cinemática Direta

Com as posições das juntas, pode-se determinar a posição do efetuador terminal (ou órgão terminal) geometricamente, ou através de técnicas de transformações lineares. A técnica geométrica é aplicável quando a configuração do robô é relativamente simples, evitando assim as transformações lineares.

A *Figura 2.9* ilustra o modelo de um robô SCARA com dois graus de liberdade, desconsiderando a junta prismática. Para esse robô, pode-se calcular facilmente sua cinemática direta utilizando a geometria como mostrado na equação (16).

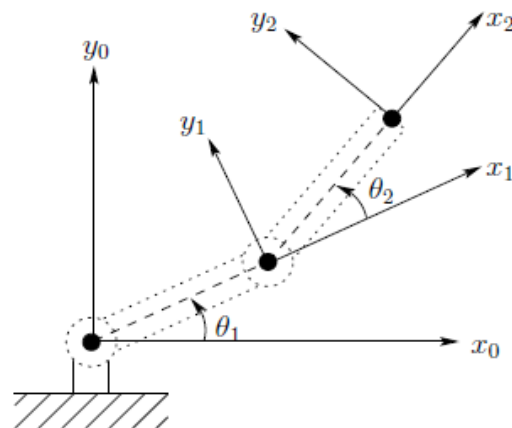


Figura 2.9 Modelo de um robô SCARA planar [1]

$$\begin{cases} x_2 = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ y_2 = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{cases} \quad (16)$$

em que l_1 representa o elo com uma inclinação θ_1 referente ao plano x_0y_0 e l_2 o outro elo indicado na *Figura 2.9*.

2.5.2 Cinemática Inversa

Na cinemática inversa, deseja-se obter a posição das juntas através da posição do efetuador terminal. Novamente, pode-se utilizar o método geométrico ou por transformações lineares.

Utilizando novamente a *Figura 2.9* como exemplo, através de uma das técnicas citadas, as seguintes equações são obtidas

$$\theta_2 = \pm \cos^{-1} \left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2} \right) \quad (17)$$

$$\theta_1 = \tan^{-1} \left(\frac{y}{x} \right) - \tan^{-1} \left(\frac{l_2 \sin \theta_2}{l_1 + l_2 \cos \theta_2} \right) \quad (18)$$

que definem a cinemática inversa do robô (a dedução completa pode ser encontrada em [1]).

2.5.3 Cinemática Inversa por RNA

Um estudo comparativo foi feito calculando a cinemática inversa, através da direta (mais facilmente obtida), por RNA, mais especificamente *perceptron* multicamadas. Para esse estudo, utilizou-se o software *MATLAB®*. As RNAs foram construídas utilizando a toolbox: *MathWorks® - Neural Network Toolbox*. Com essa toolbox é possível definir diversos parâmetros da rede como: número de entradas e saídas; quantidade de camadas ocultas de neurônios, assim como a quantidade de neurônios em cada camada; a função de ativação utilizada pela rede; entre outras.

Primeiramente, construíram-se 900 configurações de rede *perceptron* multicamadas: 2 camadas ocultas (variando as duas camadas de 1 a 30 neurônios); função de ativação tangente hiperbólica; algoritmo levenberg-marquardt para o treinamento.

No treinamento das redes, foram utilizados 105 pontos. Esses pontos foram gerados a partir da cinemática direta do manipulador, equação (16). Após o treinamento, a resposta da rede foi avaliada para esses mesmos pontos treinados e, em seguida, para outros 2002 pontos, com o intuito de avaliar a interpolação da rede.

Com os resultados das redes, escolheu-se a que alcançou a menor média de erro para os 2002 pontos de teste avaliando se, para pontos sem interpolação, o erro era admissível, ou seja, menor que o erro de resolução dos motores de um manipulador SCARA planar simulado para esse estudo. Considerou-se os erros de resolução das juntas rotacionais de um grau e dois elos ligados a essas juntas de 10,5 cm, apresentado um erro de resolução de 0,1511 cm.

Os pontos de treinamento e de testes utilizados no treinamento da rede e na interpolação são apresentados na Figura 2.10 e Figura 2.11, respectivamente. A Tabela 2.1 mostra as 5 configurações que obtiveram o menor erro (distância euclidiana média).

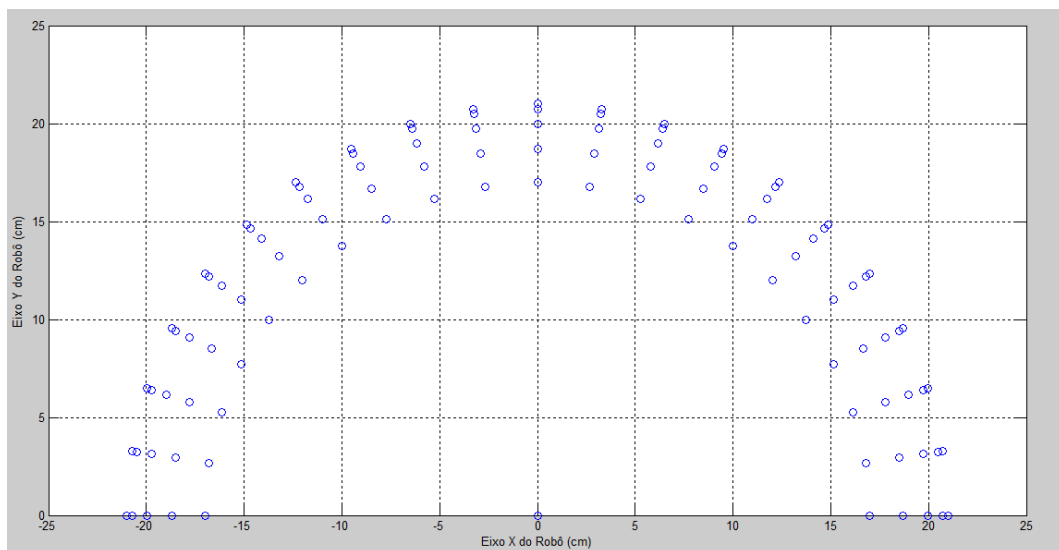


Figura 2.10 Pontos de treinamento da rede

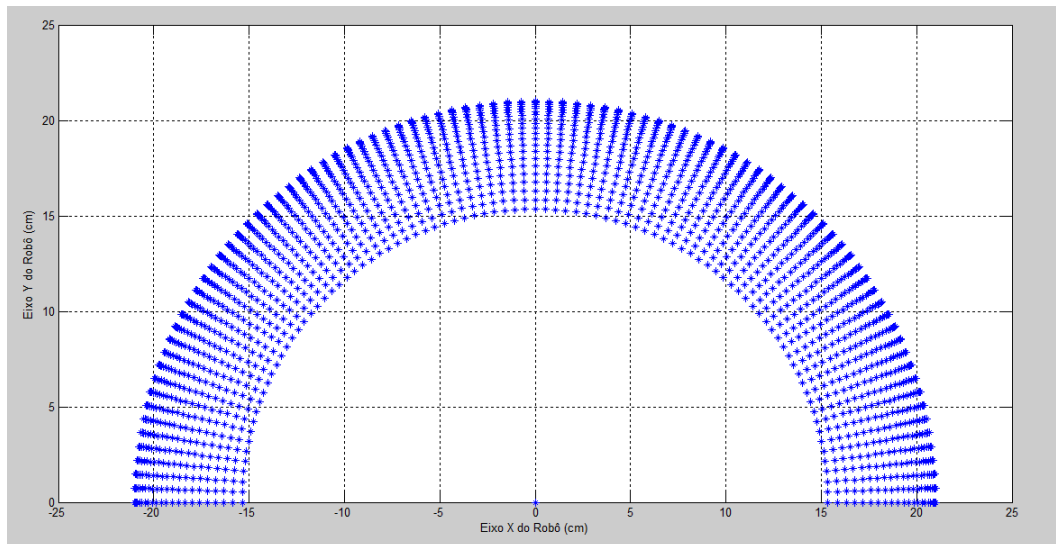


Figura 2.11 Pontos de teste aplicados a rede

Tabela 2.1 Melhores respostas da rede para 2002 pontos de teste

Configurações	Neurônios 1 ^a camada oculta	Neurônios 2 ^a camada oculta	Distância euclidiana média [cm]
1	11	18	0,113052582
2	7	10	0,114765243
3	12	10	0,122492845
4	7	16	0,132826435
5	10	6	0,133279735

Embora o erro médio (distância euclidiana média) dos 2002 pontos seja menor que o erro de resolução, alguns pontos apresentaram erros muito maiores que o admissível. Necessitando assim, em uma análise futura, a otimização do algoritmo de treinamento para esses pontos discrepantes. Um treinamento da RNA por reforço, surge como uma tentativa da diminuição dessas diferenças ou, uma outra alternativa, aumentar os pontos de treinamento até que todos os pontos alcançados estejam dentro da faixa de resolução do robô.

Como as RNAs construídas (e testadas da maneira apresentada) não alcançaram um resultado satisfatório, optou-se por utilizar na implementação do software de controle a cinemática inversa obtida nas equações (17) e (18).

3 PROJETO MECÂNICO

O projeto mecânico do manipulador foi desenvolvido em 3 fases: Cálculos Estáticos, Cálculos Dinâmicos e Desenho. Na fase Desenho, levou-se em consideração, além da análise estrutural, as restrições geométricas para a elaboração de todas as peças constituintes do manipulador. Em Cálculos Estáticos, calculou-se a deflexão máxima para os esforços sofridos pelo manipulador, considerando a exigência de que as tensões máximas sejam inferiores a tensão de escoamento do material escolhido (alumínio), para que a equação da linha elástica possa ser aplicada. Nos Cálculos Dinâmicos, desenvolveu-se a dinâmica do manipulador através da Mecânica de Lagrange. Assumiram-se acelerações e velocidades máximas para o projeto deduzindo, assim, os torques exigidos em cada junta.

3.1 Cálculos Estáticos

O desenvolvimento do cálculo estático do robô SCARA será apresentado a seguir. A Figura 3.1 mostra o modelo do primeiro e do segundo elo no plano x-z e, a Figura 3.2, mostra o plano y-z do robô. Onde q_i é o carregamento distribuído; \vec{P}_i é o peso do servo motor ou do órgão terminal (a depender do elo); \vec{l}_i é o comprimento do elo; h_i e b_i são a altura e a largura da secção transversal da viga em balanço. O índice i varia de acordo com o elo em que se refere.

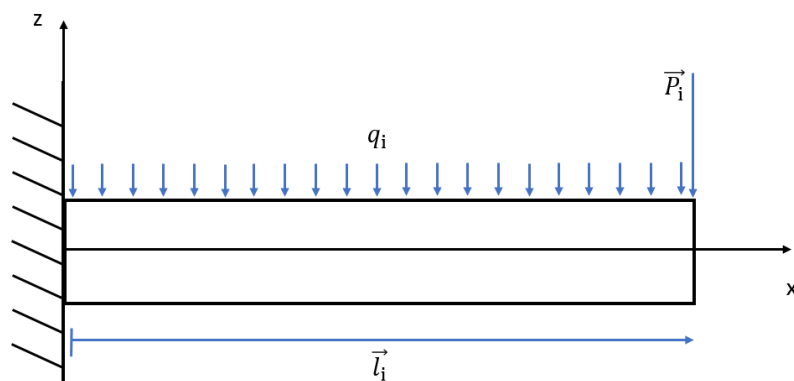


Figura 3.1 Modelo estático do primeiro e segundo elo no plano x-z

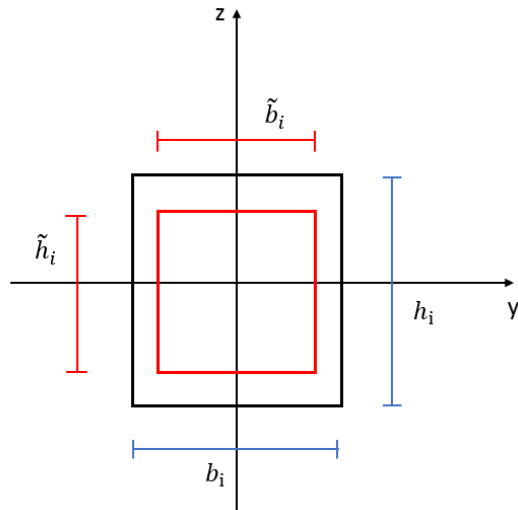


Figura 3.2 Modelo estático do primeiro e segundo elo no plano y-z

A partir do modelo, o diagrama de corpo livre da viga no plano x-z (Figura 3.3) é desenhado, evidenciando as reações do engaste: M_a e V_a , momento fletor e tensão de cisalhamento, respectivamente.

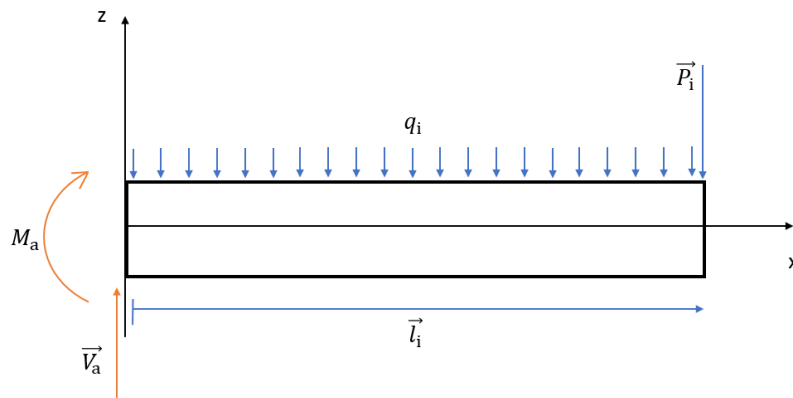


Figura 3.3 Diagrama de corpo livre

Utilizando a segunda lei de newton para estática, temos

$$V_{a_i} = P_i + q_i l_i \quad (19)$$

$$M_{a_i} = -\frac{q_i l_i^2}{2} - P_i l_i \quad (20)$$

que, ao analisar uma secção de comprimento x_i , obtêm-se o momento fletor e a tensão de cisalhamento em função desse x_i :

$$V_i(x_i) = -q_i x_i + P_i + q_i l_i \quad (21)$$

$$M_i(x_i) = -\frac{q_i}{2} x_i^2 + (P_i + q_i l_i) x_i + \left(-\frac{q_i l_i^2}{2} - P_i l_i \right) \quad (22)$$

em que $0 \leq x_i \leq l_i$.

Uma viga em balanço apresenta a tensão de cisalhamento e o momento fletor máximos (em módulo) em $x_i = 0$, e esses valores são:

$$|V_i|_{max} = P_i + q_i l_i \quad (23)$$

$$|M_i|_{max} = \frac{q_i l_i^2}{2} + P_i l_i \quad (24)$$

Sabe-se que a tensão normal máxima (σ_{m_i}), para uma flexão é:

$$\sigma_{m_i} = \frac{|M_i|_{max} c_i}{I_{y_i}} \quad (25)$$

em que I_{y_i} é o momento de área em relação ao eixo y e, c_i é a distância entre a base (em relação ao eixo z) e a linha neutra da viga, metade de h_i . O momento de área em relação ao eixo y é dado por:

$$I_{y_i} = \frac{(b_i h_i^3 - \tilde{b}_i \tilde{h}_i^3)}{12} \quad (26)$$

Com essas relações, a tensão normal máxima resulta em:

$$\sigma_{m_i} = \frac{3l_i h_i}{b_i h_i^3 - \tilde{b}_i \tilde{h}_i^3} (q_i l_i + 2P_i) \quad (27)$$

A tensão de cisalhamento máxima (τ_{m_i}) nessa estrutura é dada pela equação:

$$\tau_{m_i} = \frac{|V_i|_{max} Q_i}{I_{y_i} t_i} \quad (28)$$

em que t_i é o comprimento da base da secção transversal da viga (b_i) e Q_i é o momento estático de área, representado por:

$$Q_i = \int_0^{c_i} y dA \quad (29)$$

que resolvendo essa expressão para a secção transversal da Figura 3.2 tem-se:

$$Q_i = \frac{b_i h_i^2 - \tilde{b}_i \tilde{h}_i^2}{8} \quad (30)$$

resultando em um τ_{m_i} igual a:

$$\tau_{m_i} = \frac{Q(q_i l_i + P_i)}{I_{y_i} b_i} \quad (31)$$

A representação gráfica de onde estão situadas σ_{m_i} e τ_{m_i} é dada pela Figura 3.4

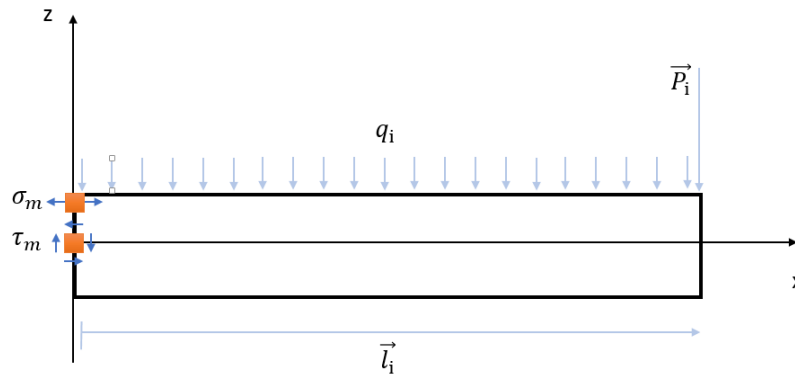


Figura 3.4 Localização da tensão máxima normal e de cisalhamento na viga engastada

Em seguida, calcula-se a equação da linha elástica para a viga da Figura 3.1, através da equação:

$$\frac{d^2 z}{dx^2} = \frac{M_i(x_i)}{E_i I_{y_i}} \quad (32)$$

em que E_i é o módulo de elasticidade da viga.

Apenas o $z_i = z_i(x_i)$ ainda não foi obtido. Para obtê-lo, integra-se duas vezes, utilizando as condições de contorno

$$\left. \frac{dz_i}{dx_i} \right|_{x_i=0} = 0 \quad (33)$$

$$z_i(x_i)|_{x_i=0} = 0 \quad (34)$$

para determinar as constantes da equação. Resultando assim na equação da linha elástica

$$z_i(x_i) = \frac{1}{24E_i I_i} [-q_i x_i^4 + 4x_i^3(P_i + q_i l_i) - 6x_i^2(q_i l_i^2 + 2P_i l_i)] \quad (35)$$

A deflexão máxima, em uma viga engastada, é obtida quando $x_i = l_i$.

Considerando os dois primeiros elos do manipulador como uma viga única de alumínio de seção transversal de 50x50 mm, espessura de 2 mm e os valores das variáveis como mostrado na Tabela 3.1, o z_i máximo pode ser obtido.

Tabela 3.1 Substituição de variáveis estáticas

Variável	Valor	Unidade de Medida
l_i	0,45	m
x_i	0,45	m
E_i	70.000	MPa
I_i	$1,47 \times 10^{-7}$	m^4
q_i	15,997	kg/m
P_i	24,53	N
$z_i(l_i)$	-0,066	mm

O comprimento dos dois elos do manipulador é mais de 20% menor que a escolha do comprimento de l_i . Essa escolha foi feita para acrescentar deformação a viga e compensar as simplificações feitas. Outra escolha, com o intuito de super dimensionar o projeto, foi a da densidade do alumínio, pois essa informação não foi disponibilizada. Optou-se, portanto, utilizar as especificações técnicas do Alumínio 2011, por possuir a maior densidade das ligas de alumínio ($2,83 \text{ g/cm}^3$) [10].

As tensões máximas (normal e cisalhamento) foram calculadas para validar o uso da equação da linha elástica, pois a mesma só pode ser utilizada se o material se encontra na região elástica, ou seja, abaixo da tensão de escoamento do material, que

no caso do Alumínio 2011, esse valor é de 245 MPa [10]. Como a Tabela 3.2 demonstra, essas tensões ficaram muito abaixo desse limite.

Tabela 3.2 Tensões normal e de cisalhamento máximas

Variável	Valor	Unidade de Medida
τ_{m_i}	0,013	MPa
σ_{m_i}	1,768	MPa

3.2 Cálculos Dinâmicos

A seguir será apresentado o desenvolvimento do cálculo dinâmico do robô SCARA através da mecânica de Lagrange. O modelo do robô, no plano x-y é apresentado na Figura 3.5 com sua legenda logo abaixo.

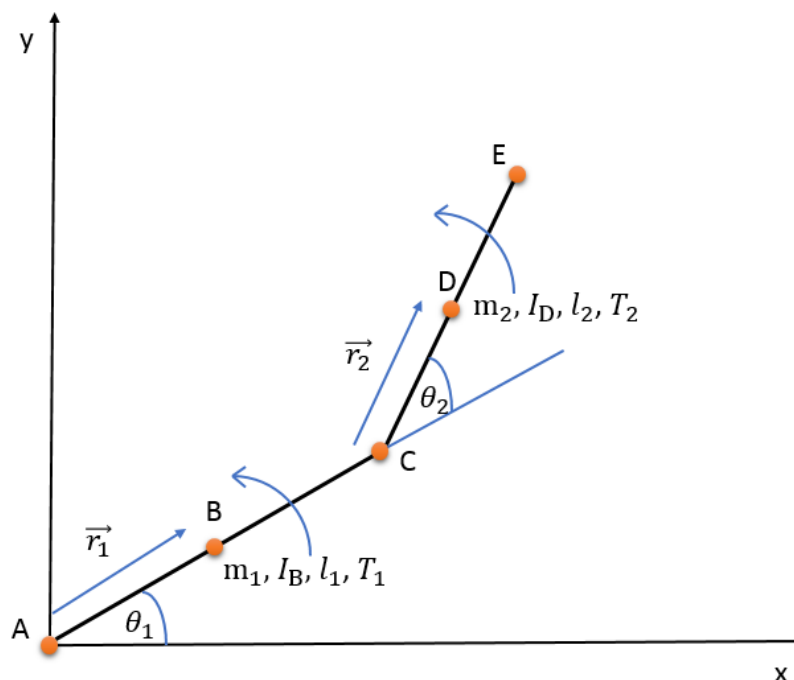


Figura 3.5 Modelo SCARA para o cálculo da dinâmica

Legenda da Figura 3.5:

- θ_1 : Deslocamento angular da primeira junta;
- θ_2 : Deslocamento angular da segunda junta;

- l_1 : Comprimento do primeiro elo;
- l_2 : Comprimento do segundo elo;
- \vec{r}_1 : Distância entre os pontos A e B;
- \vec{r}_2 : Distância entre os pontos C e D;
- Ponto A: Origem das coordenadas (primeira junta);
- Ponto B: Centro de massa do primeiro elo;
- Ponto C: Segunda junta;
- Ponto D: Centro de massa do segundo elo;
- Ponto E: Extremidade do segundo elo;
- I_B : Momento de inércia do primeiro elo em relação ao ponto B;
- I_D : Momento de inércia do segundo elo em relação ao ponto D;
- m_1 : massa do primeiro elo + massa do motor;
- m_2 : massa do segundo elo + massa da carga;
- T_1 : Conjugado do primeiro elo;
- T_2 : Conjugado do segundo elo.

Resolvendo pela Mecânica Lagrangiana, tem-se

$$L = K - Pot \quad (36)$$

$$K = K_1 + K_2 + K_3 \quad (37)$$

$$Pot = Pot_1 + Pot_2 + Pot_3 \quad (38)$$

em que K_1, K_2, K_3 são as energias cinéticas do primeiro, do segundo e do terceiro elo (junta prismática ortogonal ao plano xy) e Pot_1, Pot_2, Pot_3 as energias potenciais. O é dado por $L = L(\theta_1(t), \dot{\theta}_1(t), \theta_2(t), \dot{\theta}_2(t), z(t), \dot{z}(t), t)$, em que $z(t)$ é o deslocamento da junta prismática no sentido do eixo z.

A energia cinética do primeiro elo é dada por

$$K_1 = \frac{m_1 v_B^2}{2} + \frac{I_B \dot{\theta}_1^2}{2} \quad (39)$$

em que v_B é a velocidade linear do centro de massa do primeiro elo, que pode ser escrito como

$$K_1 = \frac{m_1 (\dot{\theta}_1 r_1)^2}{2} + \frac{I_B \dot{\theta}_1^2}{2} \quad (40)$$

ou da forma:

$$K_1 = \frac{\dot{\theta}_1^2}{2} (m_1 r_1^2 + I_B) \quad (41)$$

que pelo teorema dos Eixos Paralelos, obtém-se:

$$K_1 = \frac{\dot{\theta}_1^2 I_A}{2} \quad (42)$$

I_A é momento de inércia em relação ao ponto A.

O K_2 segue o mesmo procedimento de análise, utilizando a equação (39) como base:

$$K_2 = \frac{m_2 v_D^2}{2} + \frac{I_D (\dot{\theta}_1 + \dot{\theta}_2)^2}{2} \quad (43)$$

em que v_D é a velocidade linear do centro de massa do segundo elo, podendo ser expresso na forma vetorial

$$\vec{v}_D = \dot{x}_D \hat{i} + \dot{y}_D \hat{j} \quad (44)$$

podendo ter seu valor obtido graficamente através da posição de suas coordenadas, conforme ilustrado na Figura 3.6.

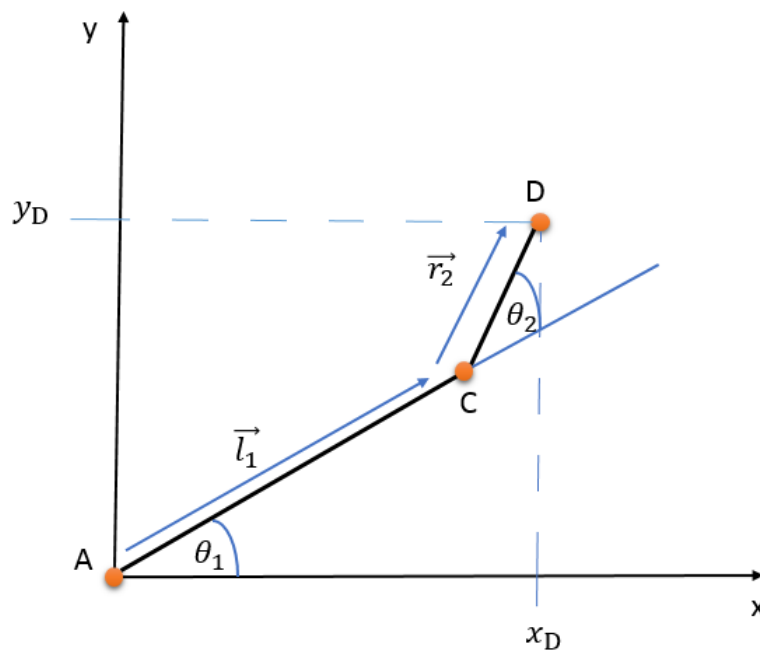


Figura 3.6 Coordenadas do ponto D

$$x_D = l_1 \cos \theta_1 + r_2 \cos(\theta_1 + \theta_2) \quad (45)$$

$$y_D = l_1 \sin \theta_1 + r_2 \sin(\theta_1 + \theta_2) \quad (46)$$

derivando x_D e y_D em relação ao tempo, obtém-se a velocidade em relação a x e y:

$$\dot{x}_D = -l_1 \dot{\theta}_1 \sin \theta_1 - r_2 (\dot{\theta}_1 + \dot{\theta}_2) \sin(\theta_1 + \theta_2) \quad (47)$$

$$\dot{y}_D = l_1 \dot{\theta}_1 \cos \theta_1 + r_2 (\dot{\theta}_1 + \dot{\theta}_2) \cos(\theta_1 + \theta_2) \quad (48)$$

a norma de \vec{v}_D ao quadrado é dada por

$$v_D^2 = \dot{x}_D^2 + \dot{y}_D^2 \quad (49)$$

substituindo os valores de \dot{x}_D^2 e \dot{y}_D^2 :

$$v_D^2 = (l_1 \dot{\theta}_1)^2 + 2l_1 \dot{\theta}_1 r_2 (\dot{\theta}_1 + \dot{\theta}_2) (\sin \theta_1 \sin(\theta_1 + \theta_2) + \cos \theta_1 \cos(\theta_1 + \theta_2)) + r_2^2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \quad (50)$$

com o intuito de simplificar v_D^2 , é utilizada a propriedade de funções par e ímpar descrita a seguir:

- Par : $f(-x) = f(x)$;
- Ímpar: $f(-x) = -f(x)$

fazendo aparecer a soma de dois arcos do cosseno:

$$v_D^2 = (l_1 \dot{\theta}_1)^2 + 2l_1 \dot{\theta}_1 r_2 (\dot{\theta}_1 + \dot{\theta}_2) (-\sin -\theta_1 \sin(\theta_1 + \theta_2) + \cos -\theta_1 \cos(\theta_1 + \theta_2)) + r_2^2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \quad (51)$$

resultando assim em uma fórmula mais reduzida de v_D^2 :

$$v_D^2 = (l_1 \dot{\theta}_1)^2 + 2l_1 \dot{\theta}_1 r_2 (\dot{\theta}_1 + \dot{\theta}_2) \cos \theta_2 + r_2^2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \quad (52)$$

que substituindo na equação na equação (43)

$$K_2 = \frac{m_2 \left[(l_1 \dot{\theta}_1)^2 + 2l_1 \dot{\theta}_1 r_2 (\dot{\theta}_1 + \dot{\theta}_2) \cos \theta_2 + r_2^2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \right]}{2} + \frac{I_D (\dot{\theta}_1 + \dot{\theta}_2)^2}{2} \quad (53)$$

Para o cálculo de K_3 , utiliza-se o modelo no plano x-z, conforme ilustrado pela Figura 3.7.

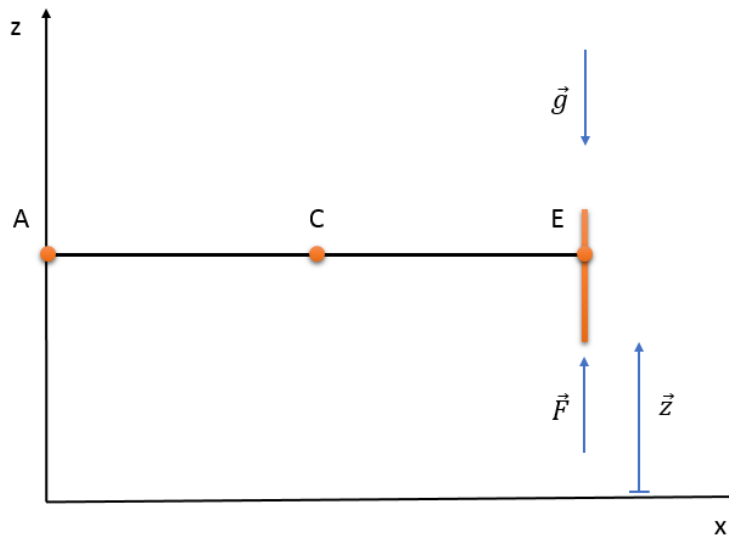


Figura 3.7 Modelo no plano x-z

Legenda da Figura 3.7:

- \vec{g} : gravidade, considerada como uma constante ao longo de todo o braço;
- \vec{z} : Deslocamento do órgão terminal em relação ao eixo z;
- \vec{F} : Força aplicada paralela ao eixo z;
- m_3 : Massa a ser deslocada pela força \vec{F}

O K_3 é então apresentado, diretamente, na forma:

$$K_3 = \frac{m_3 \dot{z}^2}{2} \quad (54)$$

A energia potencial do primeiro e do segundo elo (Pot_1 e Pot_2) é zero, uma vez em que não há movimentação desses elos ao longo do eixo z, enquanto a energia potencial do terceiro elo é facilmente encontrada como:

$$Pot_3 = m_3gz \quad (55)$$

Com os valores das energias cinéticas e potenciais, utiliza-se a equação (36) para o cálculo do Lagrangiano, resultando na equação (56).

$$L = \frac{\dot{\theta}_1^2 I_A}{2} + \frac{m_2 \left[(l_1 \dot{\theta}_1)^2 + 2l_1 \dot{\theta}_1 r_2 (\dot{\theta}_1 + \dot{\theta}_2) \cos \theta_2 + r_2^2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \right]}{2} \quad (56)$$

$$+ \frac{I_D (\dot{\theta}_1 + \dot{\theta}_2)^2}{2} + \frac{m_3 \dot{z}^2}{2} - m_3gz$$

Agora, através das equações. (12) e (13), determinam-se os valores de T_1 , T_2 e F , apresentados nas equações (57) e (58).

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = A \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + B \begin{bmatrix} \dot{\theta}_1 \dot{\theta}_2 \\ \dot{\theta}_2 \dot{\theta}_1 \end{bmatrix} + C \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \end{bmatrix} + D \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \quad (57)$$

$$F = m_3(\ddot{z} + g) \quad (58)$$

onde:

$$A = \begin{bmatrix} I_A + I_D + m_2 l_1 + m_2 r_2^2 + 2m_2 l_1 r_2 \cos \theta_2 & m_2 r_2^2 + m_2 l_1 r_2 \cos \theta_2 + I_D \\ m_2 r_2^2 + m_2 l_1 r_2 \cos \theta_2 + I_D & m_2 r_2^2 + I_D \end{bmatrix} \quad (59)$$

$$B = \begin{bmatrix} -m_2 l_1 r_2 \sin \theta_2 & -m_2 l_1 r_2 \sin \theta_2 \\ -m_2 l_1 r_2 \sin \theta_2 & 0 \end{bmatrix} \quad (60)$$

$$C = \begin{bmatrix} 0 & -m_2 l_1 r_2 \sin \theta_2 \\ 0 & 0 \end{bmatrix} \quad (61)$$

$$D = \begin{bmatrix} 0 & 0 \\ m_2 l_1 r_2 \sin \theta_2 & m_2 l_1 r_2 \sin \theta_2 \end{bmatrix} \quad (62)$$

Substituindo os valores de A , B , C , D e as velocidades e acelerações angulares pelos valores calculados, mostrado na Tabela 3.3 (com θ_2 variando: $0^\circ \leq \theta_2 \leq 1080^\circ$), obteve-se a curva de torque de T_1 e T_2 conforme ilustrado na Figura 3.8, com seus valores máximos destacados na Tabela 3.4.

Tabela 3.3 Substituição de variáveis dinâmicas

Variável	Valor	Unidade de Medida
I_A	0,0001772	kg.m ²
I_D	0,0573374	kg.m ²
m_2	2,56	kg
l_1	0,168	m
r_2	0,1014	m
$\dot{\theta}_1$	0,52	rad/s
$\dot{\theta}_2$	0,52	rad/s
$\ddot{\theta}_1$	0,52	rad/s
$\ddot{\theta}_2$	0,52	rad/s

Tabela 3.4 Torques máximos

Torque	Módulo do Máximo [N.m]	Módulo do Máximo [kgf.cm]
T_1	0,3879	3,8056
T_2	0,1276	1,2521

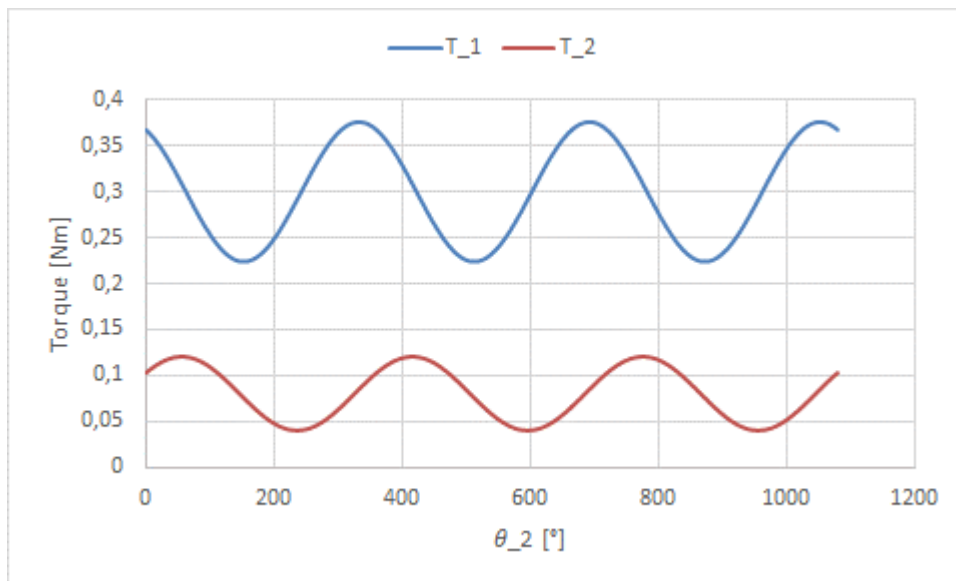


Figura 3.8 Curvas de Torque

As curvas de torque são fornecidas pelos fabricantes dos motores, portanto, com os torques obtidos, pode-se selecionar os motores adequados para o acionamento de cada junta.

3.3 Desenho e Processo de Fabricação

O manipulador foi projetado para ter um alcance de 280 mm no espaço das juntas rotacionais e 100 mm na junta prismática. A Tabela 3.5 mostra o tamanho, assim como o alcance de cada junta.

Tabela 3.5 Alcance do manipulador

Elo	Tipo de Junta	Comprimento [mm]	Alcance Rotacional [°]	Alcance Linear [mm]
1	Rotacional	168	0 - 180	-
2	Rotacional	112	0 - 180	-
3	Prismática	250	-	0 - 100

O desenho foi todo elaborado através da ferramenta computacional *FreeCad* para Linux. Figura 3.9 mostra a vista isométrica do robô retirada desse software.

Pensando sempre em facilitar o processo de fabricação, todas as peças foram desenvolvidas para que sejam confeccionadas com chapas de 1, 2 e 4 mm, tarugo de nylon e tubos de seção quadrada de 50x50 mm (além da base utilizada em MDF). A Tabela 3.6 mostra essas peças com a dimensão, a espessura e a quantidade total de cada grupo de componentes.

Tabela 3.6 Componentes do Manipulador

Especificação	Tipo	Dimensão [mm]	Espessura [mm]	Quantidade
Acoplamento Flexível	Mangueira e braçadeira	diâmetro 8	-	3
Anel Fixação	Latão	12x8	2	2
Base de Fixação	MDF	600x600	15	1
Cabine	Chapa alumínio	15x15x150	1	1
Camisa Engaste 1	Chapa alumínio	50x64x124	2	1
Corrediça Telescópica	-	250mm	-	1
Eixo Elo 1	Eixo de aço	105	diâmetro 8	1
Eixo Elo 2	Eixo de aço	100	diâmetro 8	1
Elo 1	Tubo seção quadrada	50x50x150	2	1
Elo 2	Tubo seção quadrada	50x50x150	2	1
Engaste Elo 1	Chapa alumínio	50x62x72	2	1
Espaçador	Chapa alumínio	20x50	4	2
Fixação Eixos 1 e 2	Tarugo de nylon	17x32	diâmetro 8	4
Fixação Haste	Chapa alumínio	30x50x50	2	4
Fuso Elo 3	Fuso de aço	220	diâmetro 5	1
Haste de Sustentação	Tubo seção quadrada	50x50x300	2	1
Mancal	Tarugo de nylon	46x26x9	diâmetro 8	4
Motor de Passo	-	-	-	3
Orelhas	Chapa alumínio	50x90	2	2
Rolamento	-	diâmetro8	-	4
Suporte Engaste 1	Chapa alumínio	32x50x72	2	1
Suporte Motor 1 e 2	Chapa alumínio	50x60x60	1	2
Suporte Motor 3	Chapa alumínio	42x52x150	1	1
Total de peças	-	-	-	43

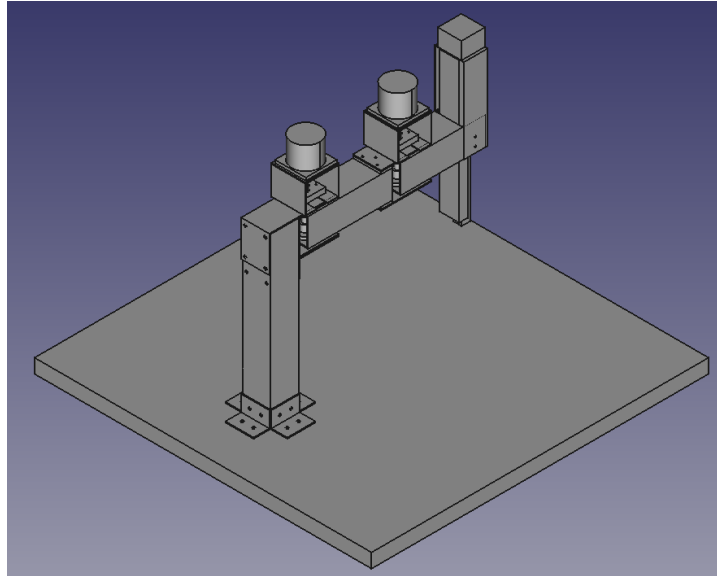


Figura 3.9 Vista isométrica do manipulador

Após a elaboração do desenho das peças, construiu-se uma maquete em tamanho real com papelão, cola quente e pregos (para representar os eixos), com o intuito de verificar possíveis melhorias. Essa maquete é apresentada na Figura 3.10.



Figura 3.10 Maquete do manipulador

Duas melhorias valem a pena destacar. Primeira foi a alteração da fixação do primeiro elo com a haste de sustentação, buscando uma maior rigidez na estrutura. A segunda foi a mudança da terceira junta (junta prismática): inicialmente, duas corredeiras telescópicas seriam utilizadas para essa junta, mas como a redução de peso é um fator crítico para um manipulador, optou-se por utilizar apenas uma, centrada no rasgo no final do segundo elo. A Figura 3.9 está representando o desenho final desenvolvido no projeto.

Decidido a versão final do desenho, assim como os materiais a serem utilizados, começou o processo de fabricação no Laboratório de Processos de Fabricação do Departamento de Engenharia Mecânica, UnB. A elaboração de todas as peças levou 20 dias para ser concluída. As peças foram solicitadas com base nos desenhos técnicos das mesmas (seção 8.1)

A Figura 3.11 mostra a estrutura mecânica do manipulador em uma base temporária (utilizada apenas para uma primeira montagem).

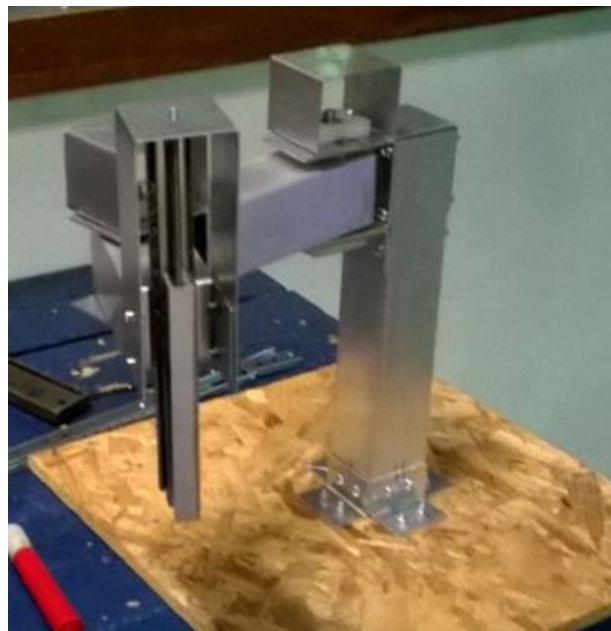


Figura 3.11 Estrutura mecânica do manipulador

3.3.1 Processo de Fabricação do Manipulador

Esse tópico irá apresenta o processo de fabricação das peças do manipulador, desenvolvido no Laboratório de Processos de Fabricação do Departamento de Engenharia Mecânica da Universidade de Brasília.

Iniciou-se esse processo cortando-se os tubos de seção quadrada de 50x50mm em 3 pedaços (dois com 150mm e um com 300mm), para formarem a haste de sustentação e os dois primeiros elos do manipulador. Em seguida, as chapas de alumínio, de diferentes espessuras, foram cortadas, dobradas e furadas. A Figura 3.12 apresenta algumas dessas peças no começo do seu processo de fabricação.

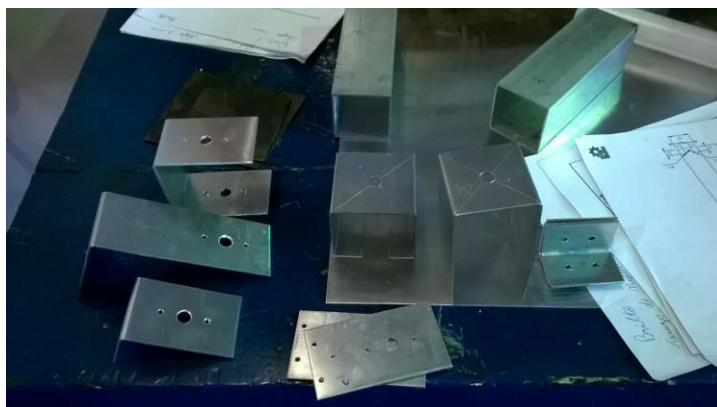


Figura 3.12 Início do processo de fabricação

Após as chapas e os tubos serem cortados e furados, iniciou-se a confecção das fixações do eixo com o tubo (peça necessária para que o movimento possa ser transmitido efetivamente para os elos do manipulador). Essas fixações foram feitas partindo-se de um tarugo de nylon de 32mm de diâmetro que foi torneado e furado para alcançar as especificações do desenho técnico. A Figura 3.13 apresenta essas fixações (e os eixos de aço) e a Figura 3.14 mostra o posicionamento dessas fixações em um determinado elo.



Figura 3.13 Fixações dos eixos

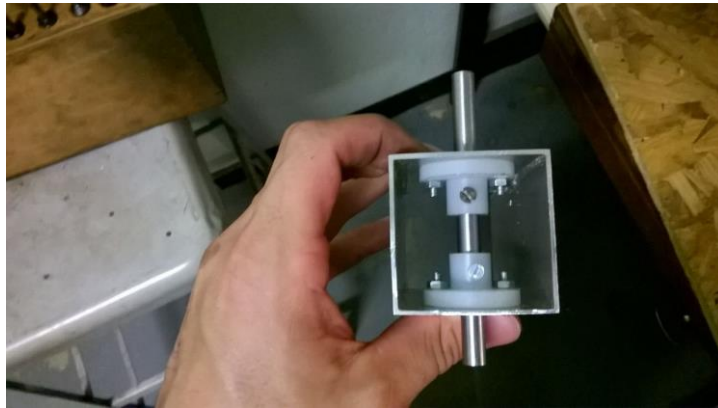


Figura 3.14 Posicionamento das fixações

Em seguida, elaboraram-se os mancais (um total de 4 peças) utilizando um tarugo de nylon de 50mm de diâmetro. O conjunto mancal/rolamento é responsável pelo alinhamento do eixo e diminuição do atrito na transmissão do movimento. A Figura 3.15 apresenta esse conjunto ainda sem os furos de fixação do mancal.



Figura 3.15 Conjunto mancal/rolamento

Para que o acoplamento não seja solicitado devido à carga do eixo, anéis de latão foram produzidos através do torneamento de um tarugo de latão. Com a peça

torneada e furada, utilizou-se um macho para a criação da rosca de 3mm de diâmetro (para que um parafuso pudesse ser utilizado na fixação).

O acoplamento dos motores relacionados às juntas rotacionais foi confeccionado utilizando uma mangueira de combustível com abraçadeiras de aço zincado, visando proporcionar um acoplamento flexível para que qualquer desalinhamento do eixo que possa ter ocorrido devido a imprecisões no processo de fabricação, não transmita momentos indesejáveis, ou seja, momentos perpendiculares ao sentido de rotação. A Figura 3.16 apresenta a primeira junta do manipulador destacando o acoplamento e o anel utilizados.

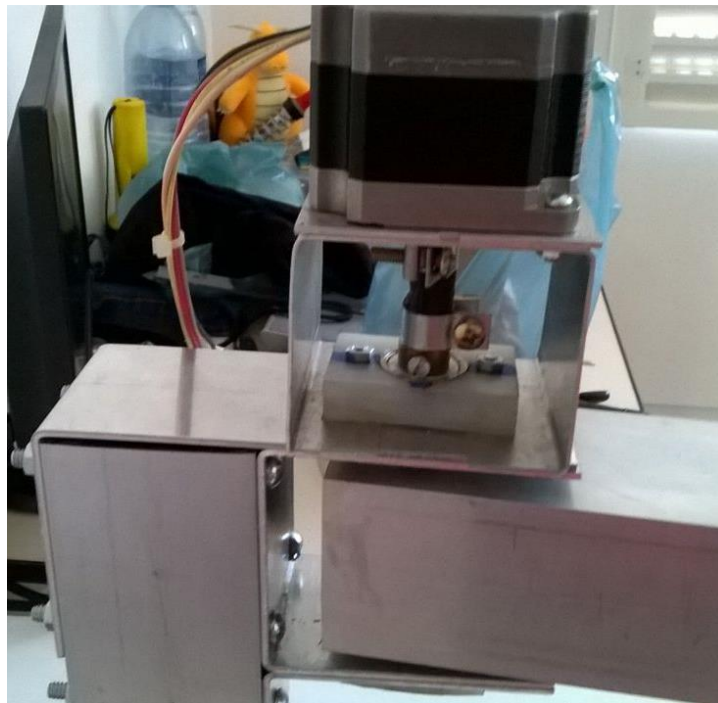


Figura 3.16 Acoplamento flexível e anel de sustentação

A junta prismática foi elaborada utilizando uma corrediça telescópica com uma barra roscada acoplada ao eixo do motor por um acoplamento flexível. Esse acoplamento foi feito utilizando uma mangueira flexível (a mesma utilizada para encapar um conjunto de fios) e abraçadeiras de nylon. Essa escolha se fez necessária devido ao espaço reduzido de trabalho que, conseqüentemente, proporcionou uma redução de peso nessa junta.

A corrediça telescópica funciona como um guia da “cabine” (junta prismática efetiva) impedindo que a mesma rotacione, fazendo com que o movimento rotacional

seja transformado em translacional. Essa corredeira foi cortada para alcançar um comprimento de 250mm, visando atender as especificações do projeto. A Figura 3.17 apresenta a junta prismática do manipulador.



Figura 3.17 Junta prismática do manipulador

Feitos alguns testes, constatou-se um desnivelamento do robô (ao se engastar uma caneta no órgão terminal do robô), fazendo com que os torques exigidos na movimentação variassem, mesmo com velocidades e acelerações dos motores constantes. Para resolver esse problema, um suporte para caneta foi desenvolvido, construído de tal forma a compensar esse desnivelamento.

O suporte foi confeccionado utilizando uma chapa de 2mm em que foi cortada e dobrada em forma de 'u', fazendo-se um furo em uma extremidade (para passar a caneta) e um rasgo na outra (para colocar duas molas: uma servindo como guia e outra como mola, efetivamente). Com esse suporte o desnivelamento pôde ser compensado, obtendo assim o resultado esperado. A Figura 3.18 mostra esse suporte para canetas.



Figura 3.18 Suporte para canetas

Com todas peças prontas, elaborou-se uma base de fixação definitiva para o manipulador. Essa base foi feita em MDF com as medidas de 600x600x15mm visando acomodar todo o volume de trabalho do manipulador. A Figura 3.19 apresenta o manipulador montado em sua base de fixação.

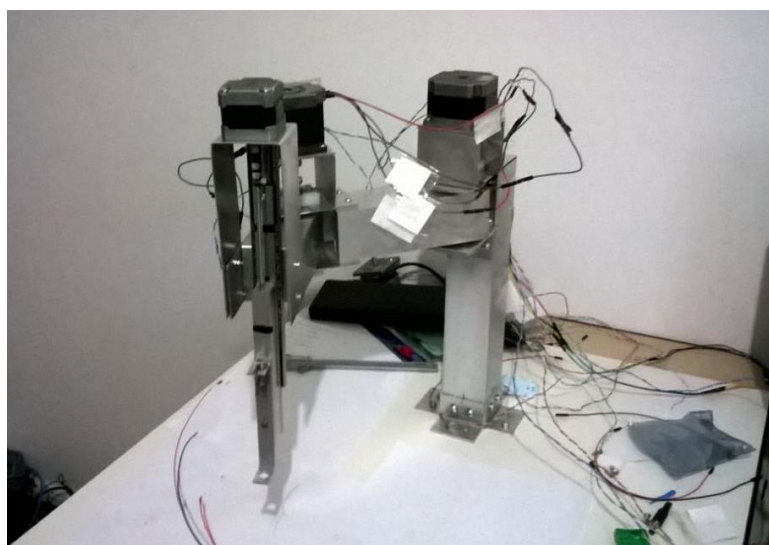


Figura 3.19 Manipulador construído

4 PROJETO ELETROELETRÔNICO

O projeto eletroeletrônico consistiu na seleção dos componentes de atuação e controle do manipulador.

Como o manipulador não possui nenhum sensor, para que sua posição pudesse ser controlada, motores de passos foram utilizados como atuadores, fazendo assim um controle em malha aberta. Utilizaram-se então motores de passo de 6 fios unipolares acionados no modo bipolar, buscando um maior acoplamento magnético das bobinas, aumentando assim o torque do motor.

Mesmo com o acionamento bipolar, o chaveamento das bobinas de forma a executar meio passo não gerou torque suficiente para garantir que não houvesse perda de passos. Optou-se, então, por diminuir a resolução do motor (de meio para passos completos por duas fases) para que um aumento de torque garantisse o funcionamento adequado do sistema, sem perdas de passos. Essa escolha foi feita pois, como não existem sensores no sistema, a contagem de passo é crucial para o posicionamento do robô possa ser obtido.

Esses motores foram acionados através de placas com o circuito integrado L298 da SGS-Thomson Microelectronics [11] que apresentam as pontes H necessárias para o controle do sentido dos motores de passo. Essa placa de acionamento é o Módulo Driver Motor com Dupla Ponte H – L298N (Figura 4.1).

Esse módulo apresenta, além das duas pontes H (detalhada na Figura 4.2), diodos de segurança, regulador de tensão, LEDs indicadores de direção, entre outros.

Os diodos de segurança formam um circuito de *free-wheeling* necessário pois, trabalha-se com impedâncias indutivas e, ao cortar um caminho de corrente, a corrente elétrica não pode variar instantaneamente, necessitando assim um caminho alternativo para que ela possa ser dissipada.

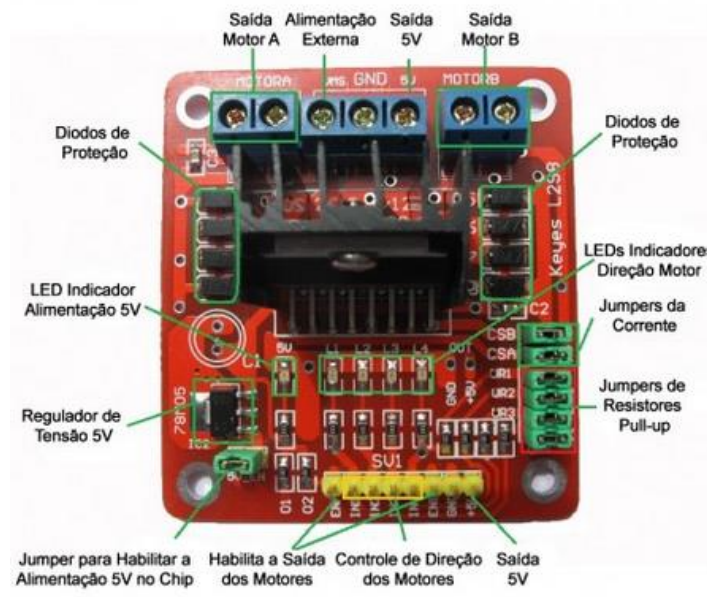


Figura 4.1 Módulo driver motor com dupla ponte H - L298N [12]

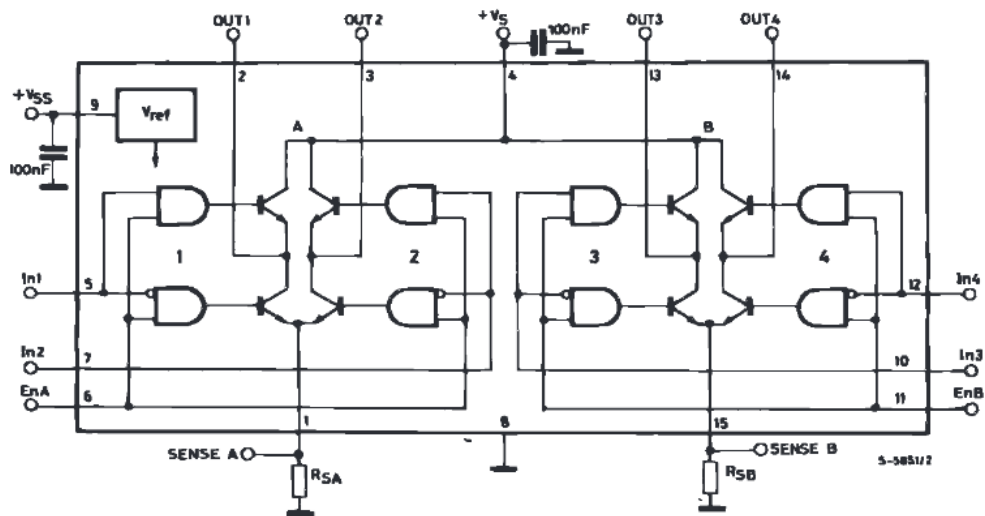


Figura 4.2 Circuito Integrado L298 [11]

Para o controle do manipulador, utilizou-se um computador conectado a dois microcontroladores: arduino UNO Genuíno [13]. Os arduinos possuem o papel de fazer o correto chaveamento das bobinas, passando o comando para o driver, para que os motores possam ser acionados. A Figura 4.3 apresenta um fluxograma demonstrando

como foi implementado o controle. Os detalhes de implementação dos softwares são apresentados na próxima seção (seção 5).

A Figura 4.3, destaca a utilização de 2 arduinos. Optou-se por essa configuração para aumentar o sincronismo das duas juntas rotacionais (motor 1 e 2), uma vez que, na definição do projeto, essas duas juntas devem começar e terminar o movimento ao mesmo tempo. O modo como foi feito o software de acionamento (por interrupções de software, controlando a velocidade de chaveamento das bobinas) também influenciou nessa decisão. A junta prismática (motor 3) possui uma movimentação muito lenta em relação as outras duas, inviabilizando o sincronismo das três juntas, pois o manipulador teria uma velocidade bastante reduzida, fazendo com que a utilização de 3 arduinos seja desnecessária.

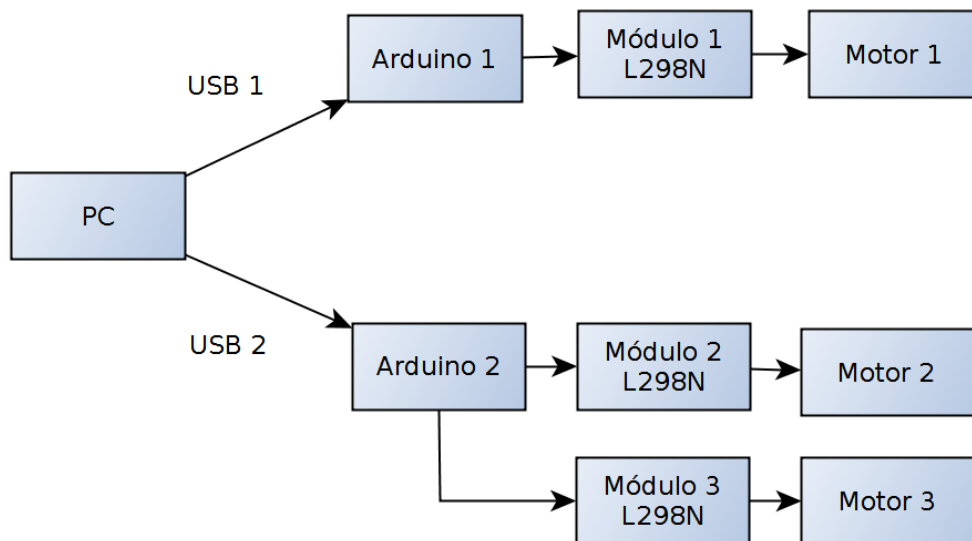


Figura 4.3 Fluxograma de Controle do Sistema

5 PROJETO DE SOFTWARE

O software desenvolvido teve como objetivo o controle de posicionamento do manipulador SCARA. Recebendo como entrada as posições destino e o tipo de interpolação a ser realizado: linear ou sem interpolação.

O software foi dividido em duas partes. Uma responsável pelos cálculos de movimentação como: cinemática direta e indireta e controle de velocidade, e o controle do sistema, elaborado na linguagem de programação C++. Outra responsável pelos acionamentos dos motores, implementada na linguagem nativa do arduino.

5.1 Software de Controle do Manipulador

O software de controle do manipulador foi elaborado na linguagem de programação C++, juntamente com o ambiente de desenvolvimento *Qt Creator* [14] e algumas bibliotecas proprietárias do *Qt*, para a comunicação serial com o arduino e a criação da interface homem-máquina. A interface de controle do manipulador é apresentada na Figura 5.1, com seu fluxograma (Figura 5.2).

Conforme apresentado na Figura 5.2, antes de exibir a interface, há uma verificação se os arduinos referentes a cada motor estão conectados às portas seriais para que os mesmos possam ser configurados. Caso algum dos arduinos não esteja conectado, uma janela é exibida indicando a consequência disso. A Figura 5.3 apresenta um exemplo que ilustra quando os 2 arduinos estão desconectados.



Figura 5.1 Interface do manipulador

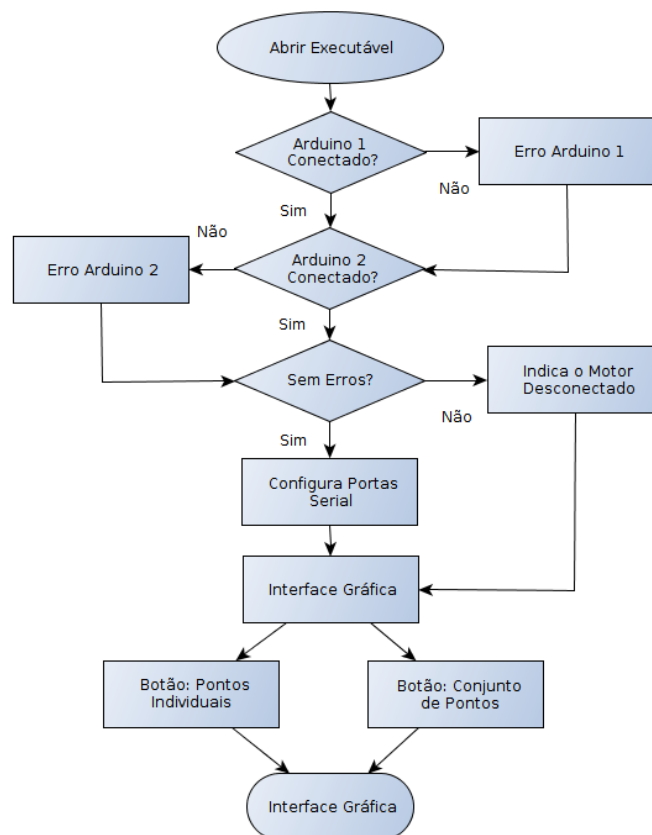


Figura 5.2 Fluxograma do programa de controle



Figura 5.3 Arduino 1 e 2 desconectados

Por essa interface, existem duas opções de entrada de dados. A primeira opção é através de pontos individuais inseridos na interface em que são escolhidos o tipo de interpolação. Após a movimentação, os valores de X atual, Y atual e Z atual são atualizados para que a posição do robô seja conhecida. E a segunda, através de um conjunto de pontos, inserido por um arquivo no formato .txt, em que as posições X, Y e Z no espaço cartesiano do robô são inseridos. Um exemplo de como são inseridos os pontos no arquivo é apresentado na Figura 5.4. Nesse arquivo, as posições X, Y e Z são separados por espaço, portando, na primeira linha, X é igual a 17, Y é igual a 17 e Z é igual 5. Ou seja, a primeira movimentação do manipulador seria: sair da posição inicial e ir para a posição (17,17,5) no espaço cartesiano.

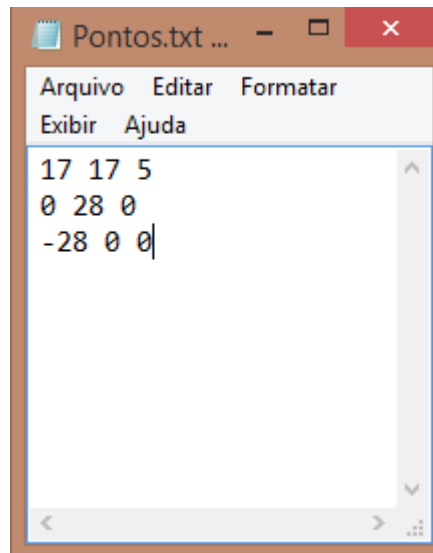


Figura 5.4 Formatação do arquivo de entrada da interface

O software foi dividido em módulos para que atualizações ou possíveis correções fossem facilmente implementadas. As figuras a seguir apresentam essa modulação do software tanto para o botão “Pontos Individuais” (Figura 5.5) quanto para “Conjunto de Pontos” (Figura 5.6). As funções constituintes desses botões serão detalhadas a seguir.

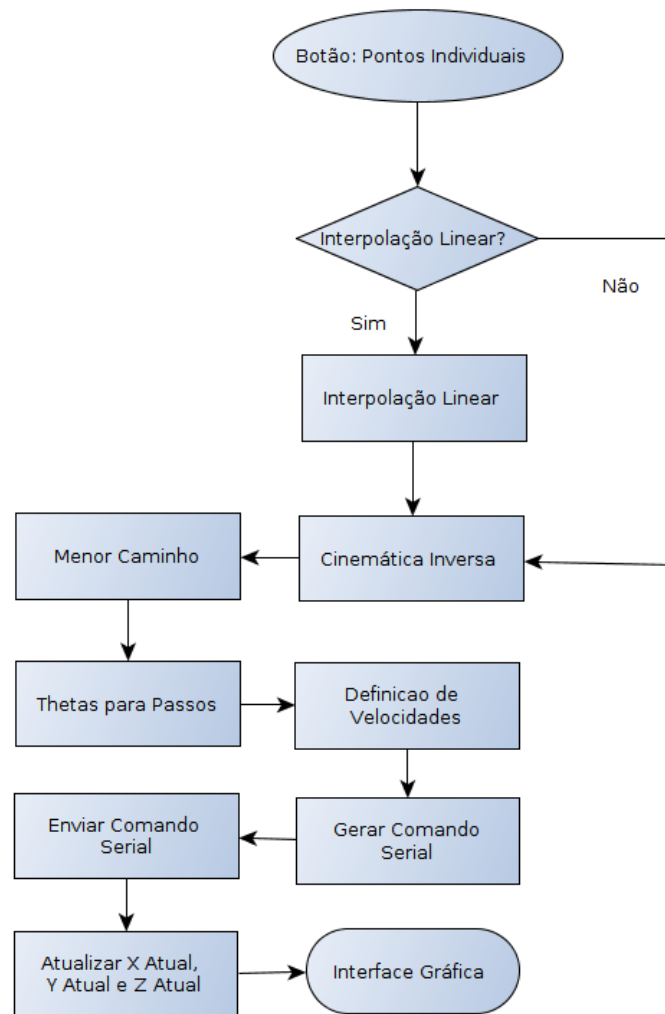


Figura 5.5 Fluxograma do botão: "Pontos Individuais"

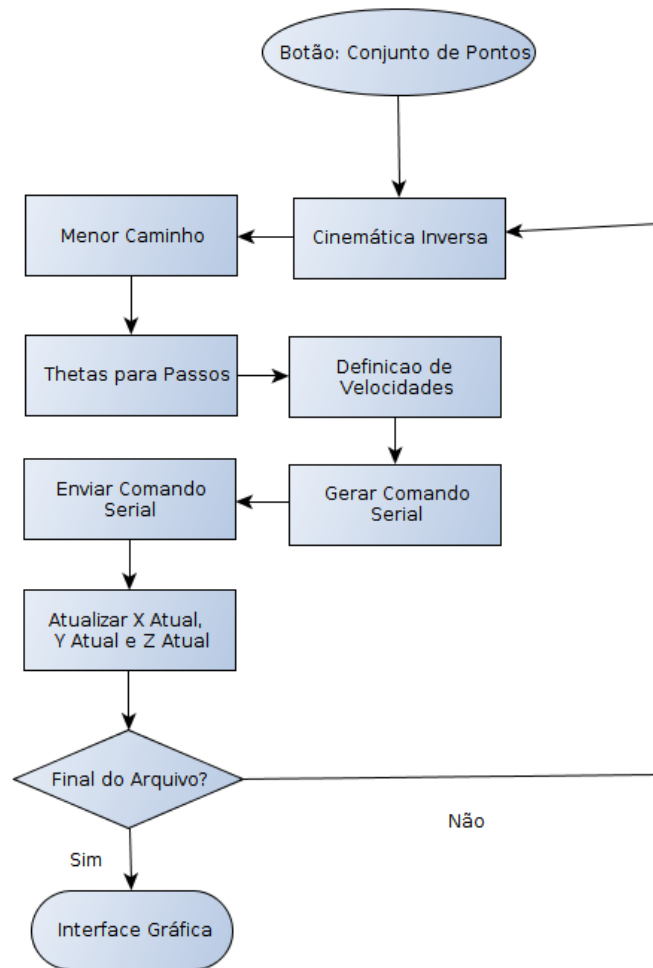


Figura 5.6 Fluxograma do botão: "Conjunto de Pontos"

5.1.1 Interpolação Linear

Essa função recebe como entrada as posições iniciais e finais no espaço cartesiano e retorna 256 pontos entre o ponto inicial e final, selecionados ao longo de uma linha reta, além dos pontos inicial e final (totalizando 258) armazenados em vetores. Os 256 pontos foram escolhidos de forma a ter um número relativamente grande de pontos entre os pontos inicial e final

5.1.2 Cinemática Inversa

A função Cinemática Inversa recebe os vetores das posições x e y e retorna e retorna os 4 ângulos possíveis das juntas rotacionais do manipulador para sua posição no espaço.

5.1.3 Menor Caminho

Recebe os 4 ângulos calculados na etapa anterior mais o ângulo de θ_1 atual e calcula a menor distância euclidiana dos dois θ_1 's em relação ao θ_1 atual. Caso os θ 's referentes a menor distância estejam no volume de trabalho, retorne esses θ 's. Caso não estejam no volume de trabalho, verifica se os outros pares de θ 's estão no volume de trabalho e retorna esses outros pares. Retorna uma janela indicando que está fora do volume de trabalho caso nenhum desses θ 's assim estejam. A Figura 5.7 mostra de forma graficamente a obtenção do menor caminho (representado por d_1^*) e a Figura 5.8 mostra a janela implementada.

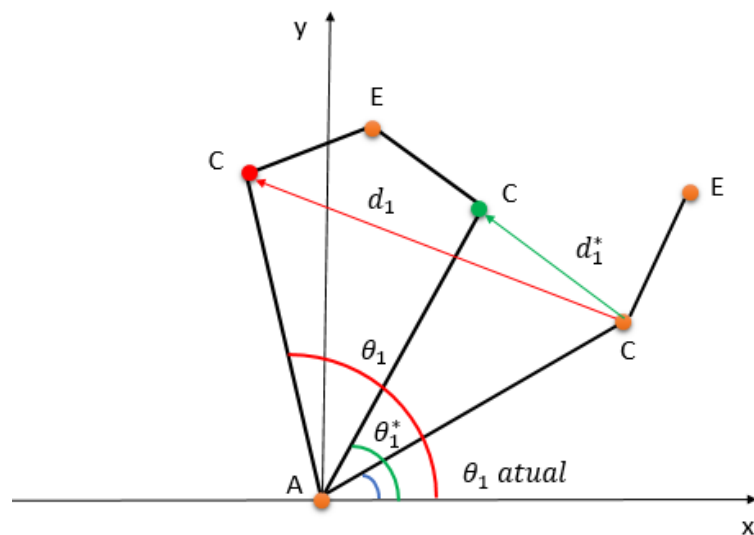


Figura 5.7 Determinação do menor caminho

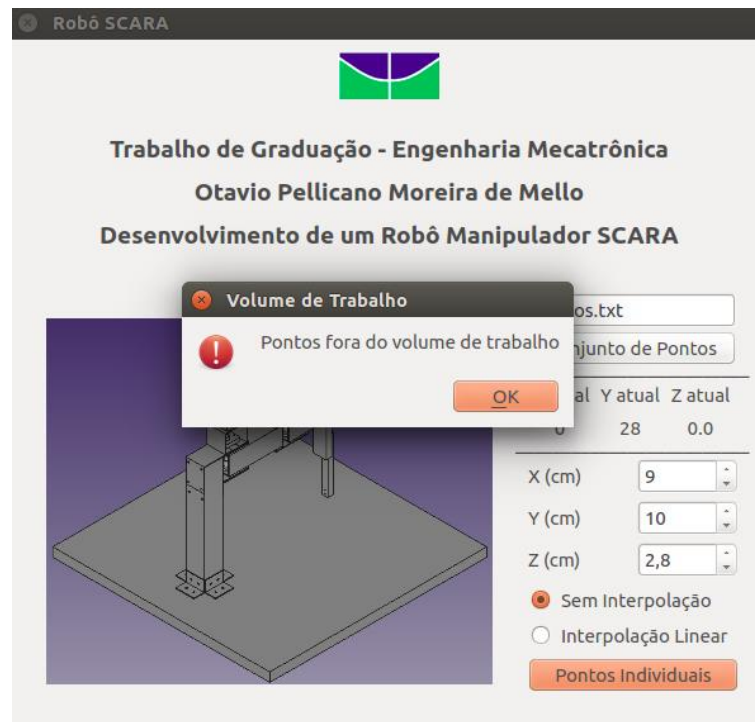


Figura 5.8 Janela de pontos fora do volume de trabalho

5.1.4 Thetas para Passos

Recebe com entrada os vetores das posições x e y. Verifica se a diferença entre cada posição há um deslocamento mínimo dos motores de passo ($1,8^\circ$). Caso esse deslocamento não seja alcançado, os θ 's relacionados são retirados dos vetores das posições x e y.

Feito esse filtro, os deslocamentos relativos são convertidos em passos e armazenados em vetores com os passos para os motores 1 e 2. O sinal do valor do passo, representa o sentido da rotação que será aplicada pelo motor.

Os passos para o motor 3 são calculados de forma direta, utilizando a diferença entre a posição final e a inicial do ponto Z. Essa simplificação foi aplicada, uma vez que a interpolação linear, assim como a movimentação simultânea dos motores, será executada apenas pelos motores das juntas rotacionais.

5.1.5 Definição de Velocidades

As entradas para essa função são os vetores com os passos do motor 1 e 2. Com esses passos são calculados os deslocamentos de cada motor das juntas rotacionais através da soma absoluta de cada vetor de passos. Para que os motores comecem e terminem ao mesmo tempo a seguinte equação deve ser satisfeita:

$$v_2 = \frac{\Delta\theta_2}{\Delta\theta_1} v_1 \quad (63)$$

em que $\Delta\theta_i$ é a variação do deslocamento angular do motor i e v_i é a velocidade ($i = 1, 2$).

No caso dessa função o deslocamento angular considerado foi justamente a soma absoluta de cada vetor de passo e, fixando a velocidade do motor 1 para a máxima permitida no projeto, calculou-se a velocidade do motor 2 da seguinte forma: se o $\Delta\theta_1$ é igual a zero, v_2 é igual a velocidade máxima do motor 2, v_1 é igual a zero e a função retorna v_1 e v_2 . Caso contrário, enquanto v_2 é maior que a velocidade máxima do motor 2, faça: equação acima e decremente v_1 em uma unidade. Se v_2 for menor ou igual a velocidade máxima permitida para esse motor, retorna v_1 e v_2 .

5.1.6 Gerar Comando Serial

Como está sendo utilizado 2 arduinos para o controle do manipulador, o Gerar Comando Serial gera 2 comandos distintos, mas similares, para cada arduino.

Para o primeiro, a velocidade e os passos do motor 1 são armazenadas em um vetor como números inteiros (fazendo arredondamentos necessários). No segundo, além da velocidade e passos do motor 2, a velocidade e passos do motor 3 também são armazenados em um vetor de inteiros. A saída dessa função é justamente esses 2 vetores com os comandos a serem enviados aos arduinos.

5.1.7 Enviar Comando Serial

As etapas Gerar Comando Serial e Enviar Comando Serial foram separadas com o intuito de diminuir o tempo de envio para cada arduino e, aumentando assim, o sincronismo dos motores. Essa função recebe os vetores gerados anteriormente e envia os comandos para os respectivos arduinos (utilizando a porta serial) que recebem esses valores para serem processados.

Se algum dos arduinos não receber o comando serial, uma janela aparece na tela avisando esse problema. A Figura 5.9 mostra a janela exibida quando o arduino 2 não consegue receber o comando serial.

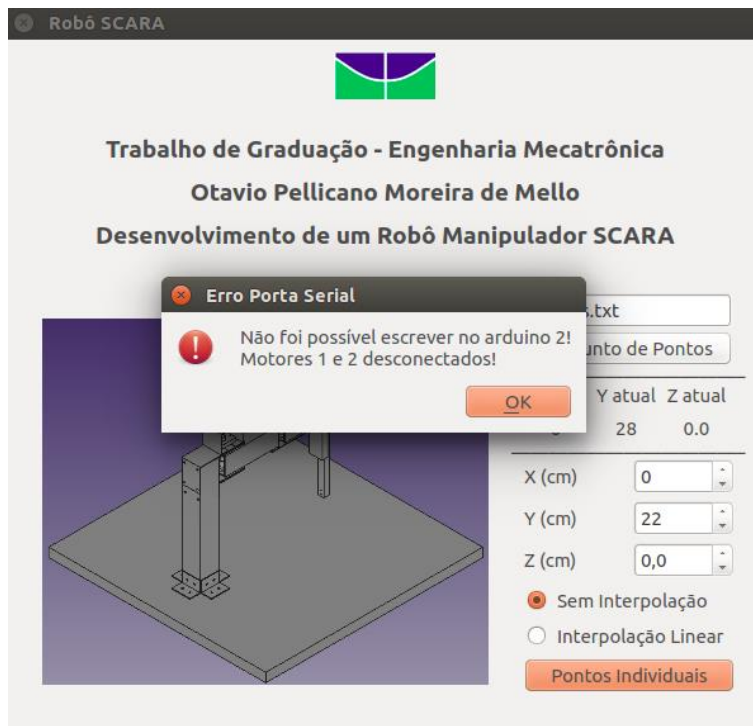


Figura 5.9 Erro ao escrever no arduino 2

5.1.8 Atualizar X Atual, Y Atual e Z Atual

Essa função apenas atualiza as posições atuais do manipulador para que o operador possa situar melhor o manipulador no espaço.

5.2 Software de Acionamento dos Motores

Esse software consiste na implementação do sentido, velocidade e do chaveamento das bobinas do motor de passo que serão utilizados para a movimentação do manipulador. Cada arduino apresenta seu próprio software com lógicas parecidas, mudando apenas a parte em que o arduino 2 utiliza para movimentar o motor da junta prismática. A Figura 5.10 mostra o fluxograma implementado em cada arduino.

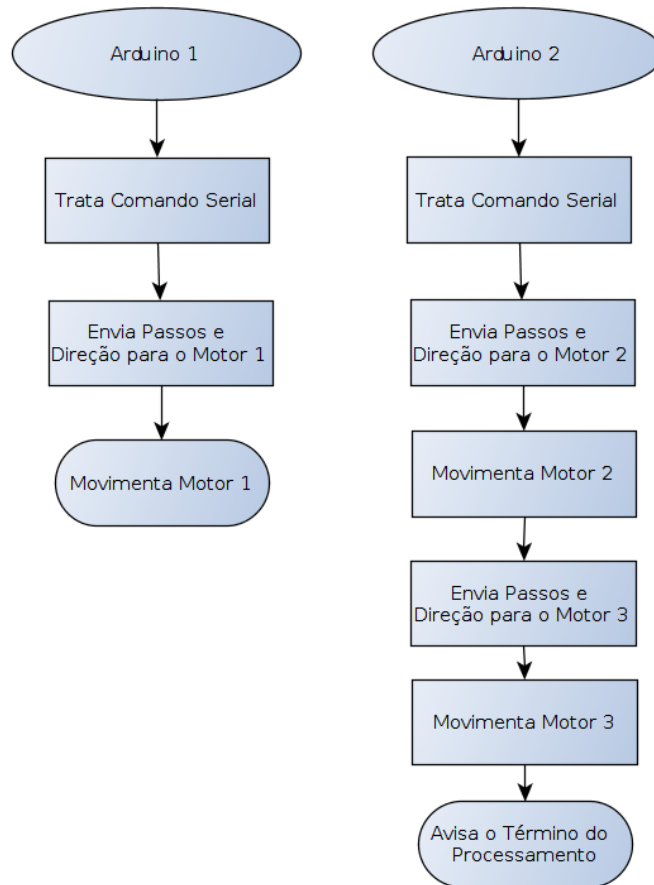


Figura 5.10 Fluxograma da implementação de cada arduino

Pensando em diminuir o tempo de resposta dos arduinos, os programas implementados foram feitos utilizando ponteiros e números de ponto fixo. O software de controle efetua os arredondamentos necessários (transformando para ponto fixo) antes de enviar os comandos seriais para os arduinos.

5.2.1 Tratar Comando Serial

Essa função recebe o vetor de números de ponto fixo que separa: a velocidade e os passos do motor 1 (arduino 1); as velocidades e passos do motor 2 e 3 (arduino 2). As velocidades recebidas pela porta serial são traduzidas para o período de chaveamento das bobinas da seguinte forma

$$periodoDeChav = \frac{deslocPorPasso}{velocidadeDeChav} \quad (64)$$

em que *periodoDeChav* é a período de chaveamento das bobinas, o tempo decorrido por passo; *deslocPorPasso* é o deslocamento rotacional devido a um passo (no caso do motor de 200 passos, usado no projeto, esse deslocamento é de $1,8^\circ$); e *velocidadeDeChav*, é a velocidade recebida pela porta serial.

5.2.2 Enviar Passos e Direção para o motor

Essa função recebe a quantidade de passos, o período de chaveamento e o sentido de rotação, calculado de acordo com o sinal no número de ponto fixo, para cada motor e efetua o chaveamento das bobinas para a movimentação do robô. As bobinas são chaveadas de maneira em que duas fases do motor são utilizadas a cada chaveamento, fornecendo assim um maior torque para o sistema.

5.2.3 Avisar o Término do Processamento

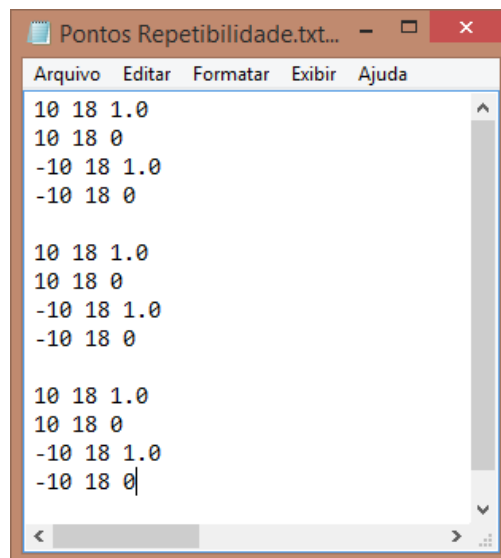
Após o término da movimentação do motor 3, um byte é enviado para o software de controle para que o mesmo possa identificar que a movimentação sessou. Para garantir que todos os motores realmente terminaram seus respectivos movimentos, uma pequena interrupção de software é feita no arduino 2. Essa interrupção se faz necessária, pois o motor 3 pode não ter efetuado nenhum movimento para um determinado conjunto de pontos.

6 TESTES DE FUNCIONAMENTO

Nesta seção serão demonstrados alguns testes de funcionamento com o intuito de validar a interpolação linear e a repetibilidade do manipulador. Para tanto, será comparado a trajetória em que o manipulador deveria percorrer (em forma de gráficos ou em formato .txt) com a trajetória realmente efetuada (uma caneta conectada ao órgão terminal do robô será utilizada nessa parte).

6.1 Repetibilidade do Manipulador

Para este teste, uma sequência de pontos foi escolhida e enviada através de um arquivo no formato .txt para o manipulador. Essa sequência passa pelos mesmos pontos 3 vezes, pois a repetibilidade que está sendo avaliada. A Figura 6.1 mostra os pontos enviados pelo manipulador, com a posição inicial (0, 28, 9.0), a Figura 6.2 os pontos alcançados e a Figura 6.3 apresenta em detalhe a marcação da caneta no papel.



```
Arquivo  Editar  Formatar  Exibir  Ajuda
10 18 1.0
10 18 0
-10 18 1.0
-10 18 0

10 18 1.0
10 18 0
-10 18 1.0
-10 18 0

10 18 1.0
10 18 0
-10 18 1.0
-10 18 0
```

Figura 6.1 Pontos para avaliar a repetibilidade

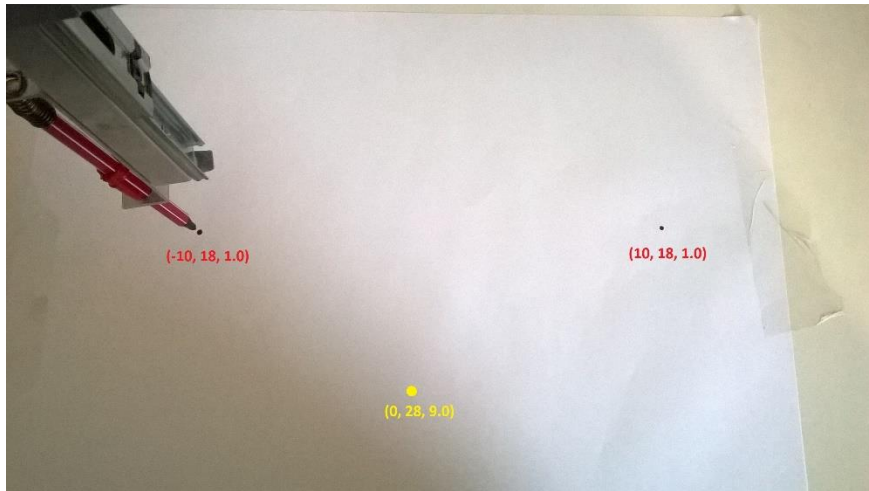


Figura 6.2 Repetibilidade do manipulador

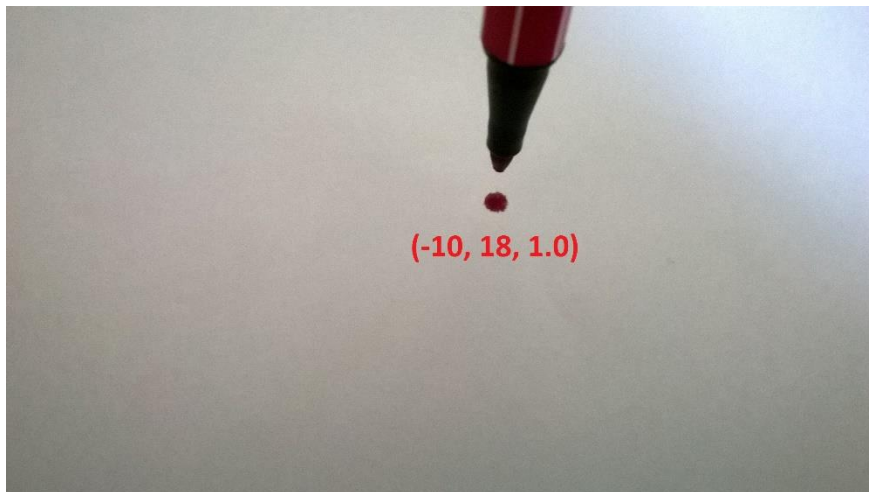


Figura 6.3 Detalhe da repetibilidade do manipulador

Como pode ser constatado, o manipulador desenvolvido apresenta uma excelente repetibilidade para os pontos simulados. Essa alta precisão é atribuída aos atuadores das juntas: motores de passo. Os motores de passo são incrementados discretamente à medida que as bobinas são chaveadas. Portanto, quando não há perda de passo, o robô sempre apresentará uma excelente repetibilidade.

6.2 Interpolação Linear

Para esta avaliação, testou a interpolação linear, visando verificar a precisão da trajetória gerada pelo software com a desenvolvida pelo manipulador. Como o intuito dessa parte é avaliar interpolação linear feita pelas duas juntas rotacionais, os valores da posição Z no plano serão omitidos.

A interpolação linear testada foi a da posição (0,22) a (0,28). A Figura 6.4 apresenta os pontos gerados pelo programa de controle. Conforme pode ser observado, existe uma diferença de mais de 1 cm entre o primeiro e o segundo ponto (para ser preciso, 1,4 cm). Essa diferença gerou um erro na interpolação do manipulador, ilustrado na Figura 6.5.

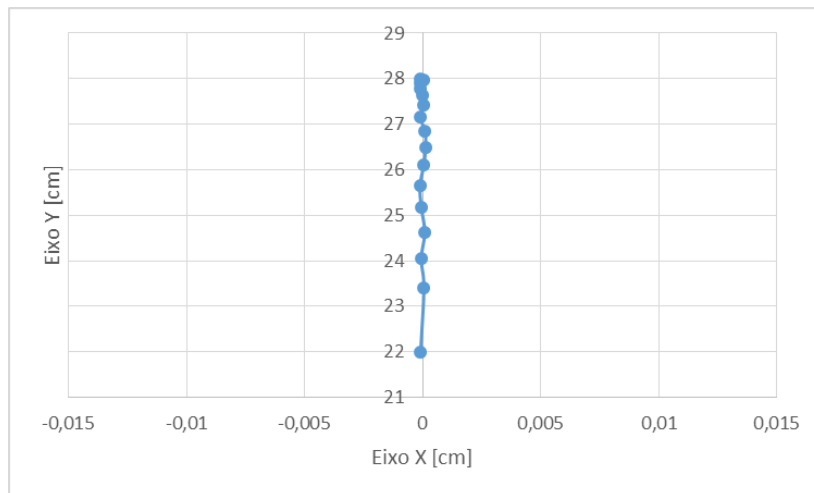


Figura 6.4 Trajetória, (0,22) a (0,28), gerada pelo software de controle

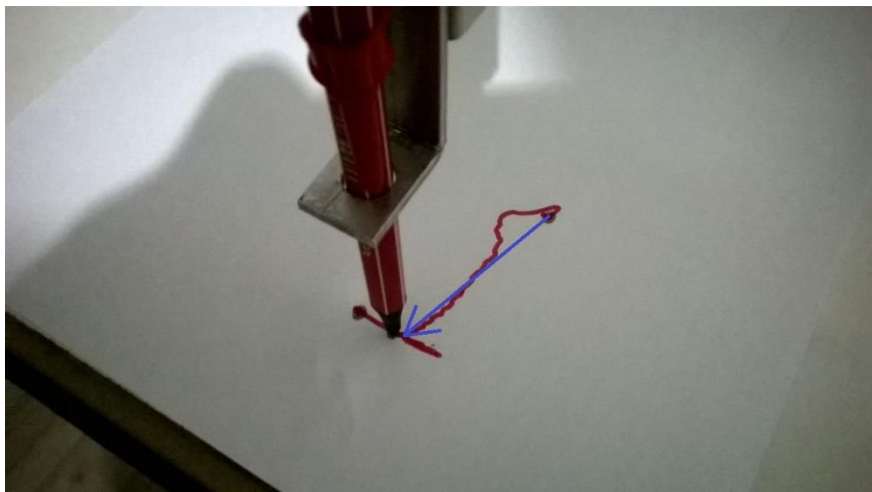


Figura 6.5 Trajetória, (0,22) a (0,28), gerada pelo manipulador

Embora os motores de passo apresentem uma excelente repetibilidade, a interpolação linear apresentou resultados longe do satisfatório. Isso ocorreu porque a resolução do motor de passo utilizado foi de $1,8^\circ$ por passo, dificultando esse tipo de interpolação. Uma alternativa para melhorar essa interpolação, seria o desenvolvimento de uma caixa de redução para cada junta rotacional. Isso aumentaria a resolução e o torque fornecidos a junta, possibilitando a escolha de mais pontos intermediários para a elaboração da interpolação linear.

7 CONCLUSÃO

O desenvolvimento do robô manipulador proporcionou a integração das três grandes áreas de engenharia mecatrônica: mecânica, elétrica e computacional. Os cálculos estáticos e dinâmicos; a seleção dos componentes eletroeletrônicos; o desenvolvimento dos softwares de controle e atuação; todas as etapas envolvendo o projeto mecatrônico puderam ser validadas com a construção de um protótipo. Os testes de funcionamento mostraram que apesar de a interpolação linear ter sido deficitária (devido à baixa resolução dos motores de passos utilizados, $1,8^\circ$ por passo), o manipulador apresentou uma boa repetibilidade no espaço de trabalho, pois a repetibilidade não é influenciada pela baixa resolução dos motores de passo. Uma alternativa sugerida para melhorar a interpolação linear, foi o desenvolvimento de uma caixa de redução para cada junta rotacional, pois, com isso, a resolução e o torque fornecidos a junta seriam aumentados, possibilitando a escolha de mais pontos intermediários na elaboração da interpolação linear.

REFERÊNCIAS BIBLIOGRÁFICAS

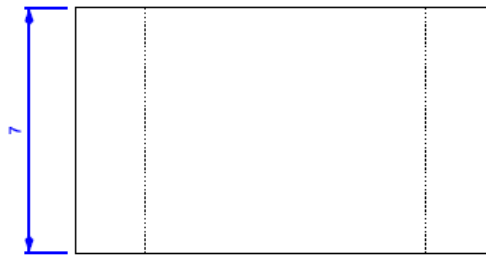
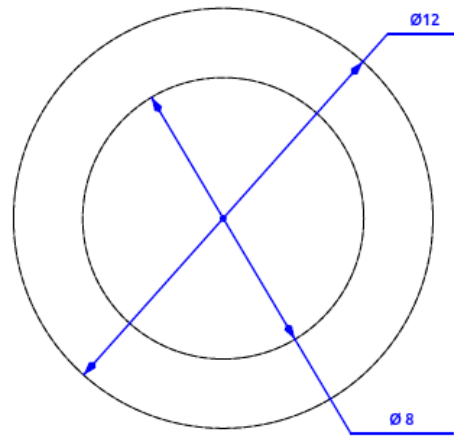
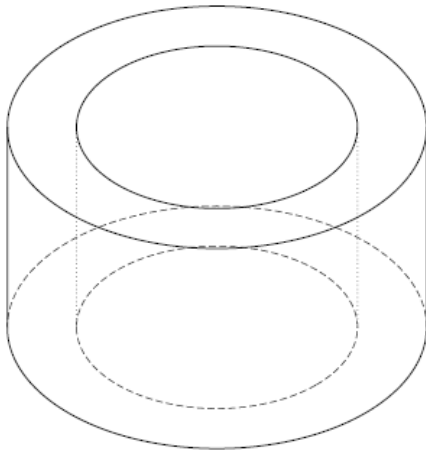
- [1] M. W. Spong e S. H. M. Vidyasagar, Robot Modeling and Control, United States of America: ed. John Wiley & Sons, Inc, 2006.
- [2] S. B. Niku, Introdução à Robótica, Rio de Janeiro: ed. LTC - Livros Técnicos e Científicos Editora Ltda, 2013.
- [3] R. M. Silva, “Introdução à Dinâmica e ao Controle de Manipuladores Robóticos”, Apostila, PUC-RS, Rio Grande do Sul.
- [4] V. B.-H. N. Araújo, J. Medeiros e A. Bitencourt, “Desenvolvimento de Robô Manipulador com Juntas Rotacionais e Prismáticas,” *VII CONNEPI: Congresso Norte Nordeste de Pesquisa e Inovação*, UFRN p. 8, 19-21 Outubro 2012.
- [5] L. Weihmann, “Descrição, Instalação, Programação e Funcionamento de um Robô Manipulador do tipo SCARA”, Dissertação (mestrado), UFSC, Santa Catarina, 1999.
- [6] M. P. Groover, M. Weiss, R. N. Nagel e N. G. Odrey, Robótica: tecnologia e programação, São Paulo: ed. McGraw-Hill Ltda, 1988.
- [7] F. P. Beer e E. R. Johnston, Resistência dos Materiais, São Paulo: ed. Pearson, 2010.
- [8] J. B. Neto, Mecânica Newtoniana, Lagrangiana e Hamiltoniana, São Paulo: ed. Livraria da Física, 2013.
- [9] I. N. da Silva, D. H. Spatti e R. A. Flauzino, Redes Neurais Artificiais: para engenharia e ciências aplicadas, São Paulo: ed. Artliber, 2010.
- [10] Catálogo de Produtos, “www.alumicopper.com.br,” AlumiCoper, [Online]. Available:
http://www.alumicopper.com.br/pdf/catalogo_produtos_alumicopper.pdf. [Acesso em 26 05 2016].

- [11] SGS-Thomson Microelectronics, “<http://www.st.com>,” [Online]. Available: http://www.st.com/content/st_com/en/search.html#q=L298-t=keywords-page=1. [Acesso em 12 06 2016].
- [12] HU Infinito, “www.huinfinito.com.br,” HU Infinito, [Online]. Available: http://www.huinfinito.com.br/controladores/583-modulo-driver-motor-com-dupla-ponteh-st-l298n.html?search_query=driver+motor+de+passo&results=3. [Acesso em 22 05 2016].
- [13] Arduino Genuino, “www.arduino.cc,” Arduino Genuino, [Online]. Available: <https://store.arduino.cc/product/GBX00066>. [Acesso em 12 05 2016].
- [14] Qt, “The IDE Qt Creator,” Qt, [Online]. Available: <https://www.qt.io/ide/>. [Acesso em 20 05 2016].

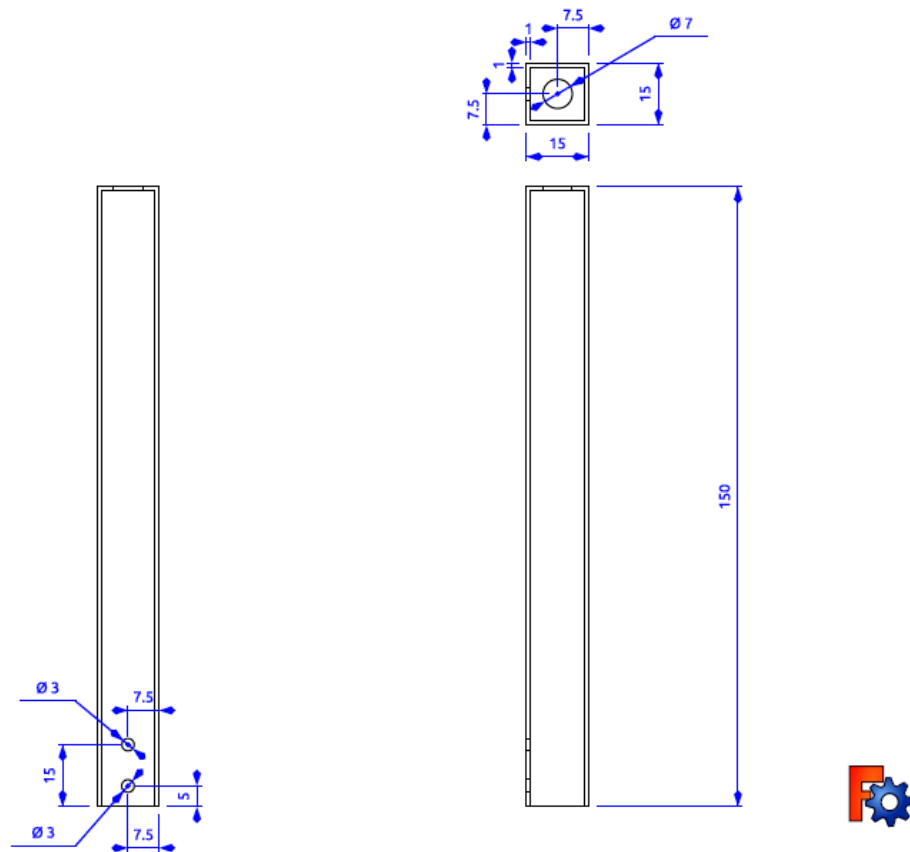
8 ANEXOS

8.1 Desenhos Técnicos do Manipulador

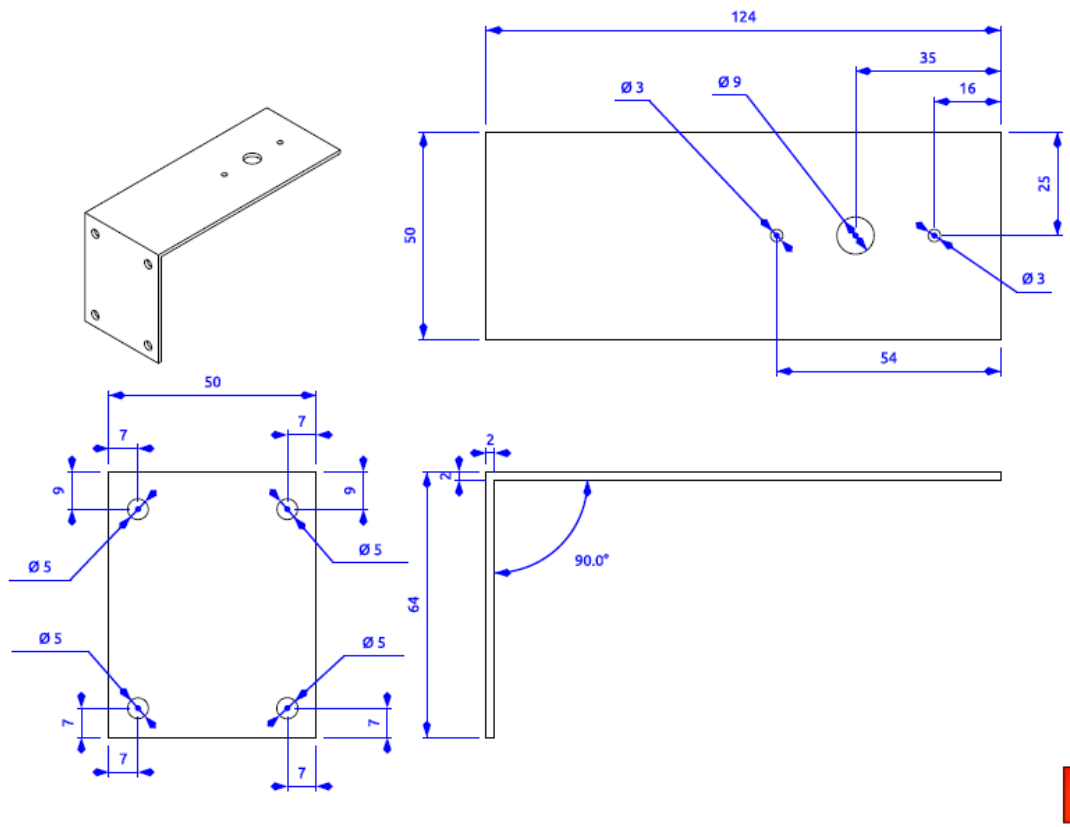
8.1.1 Anel Fixação



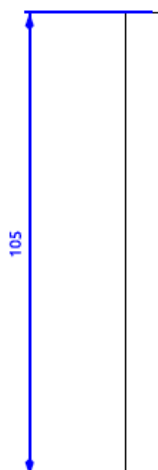
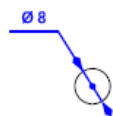
8.1.2 Cabine



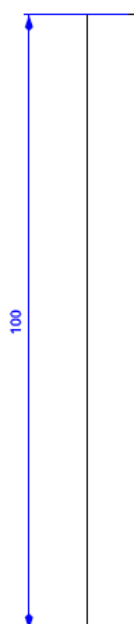
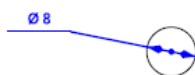
8.1.3 Camisa Engaste



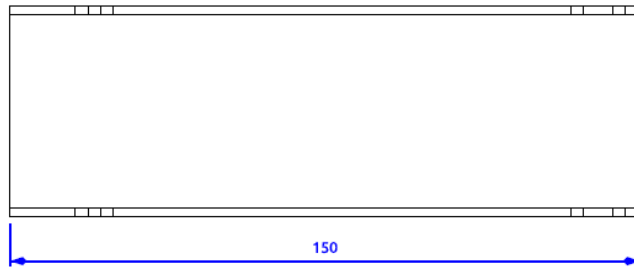
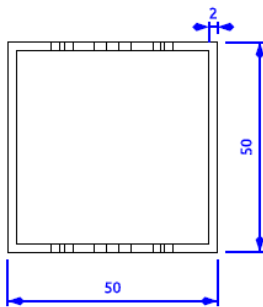
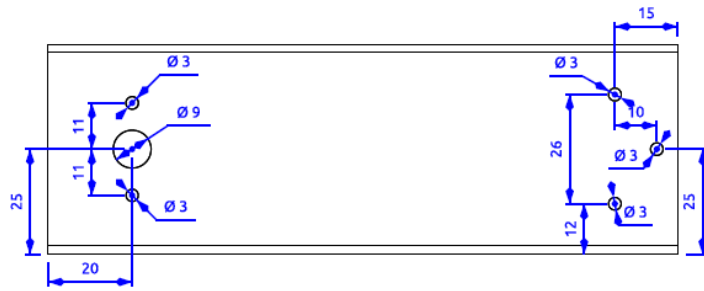
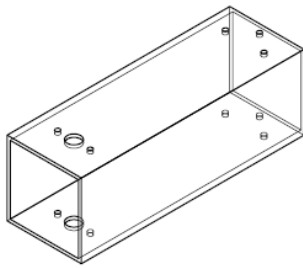
8.1.4 Eixo Elo 1



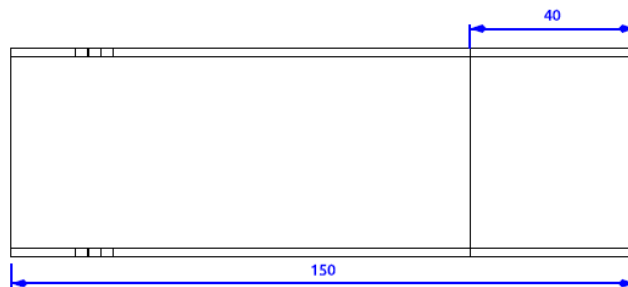
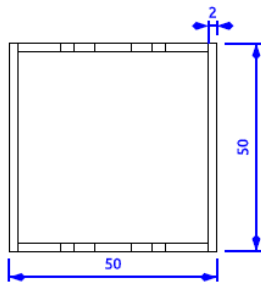
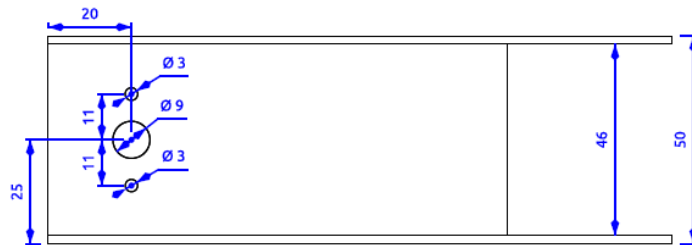
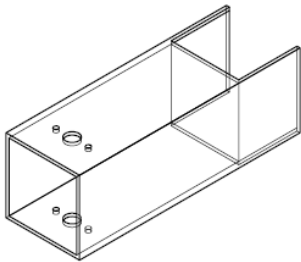
8.1.5 Eixo Elo 2



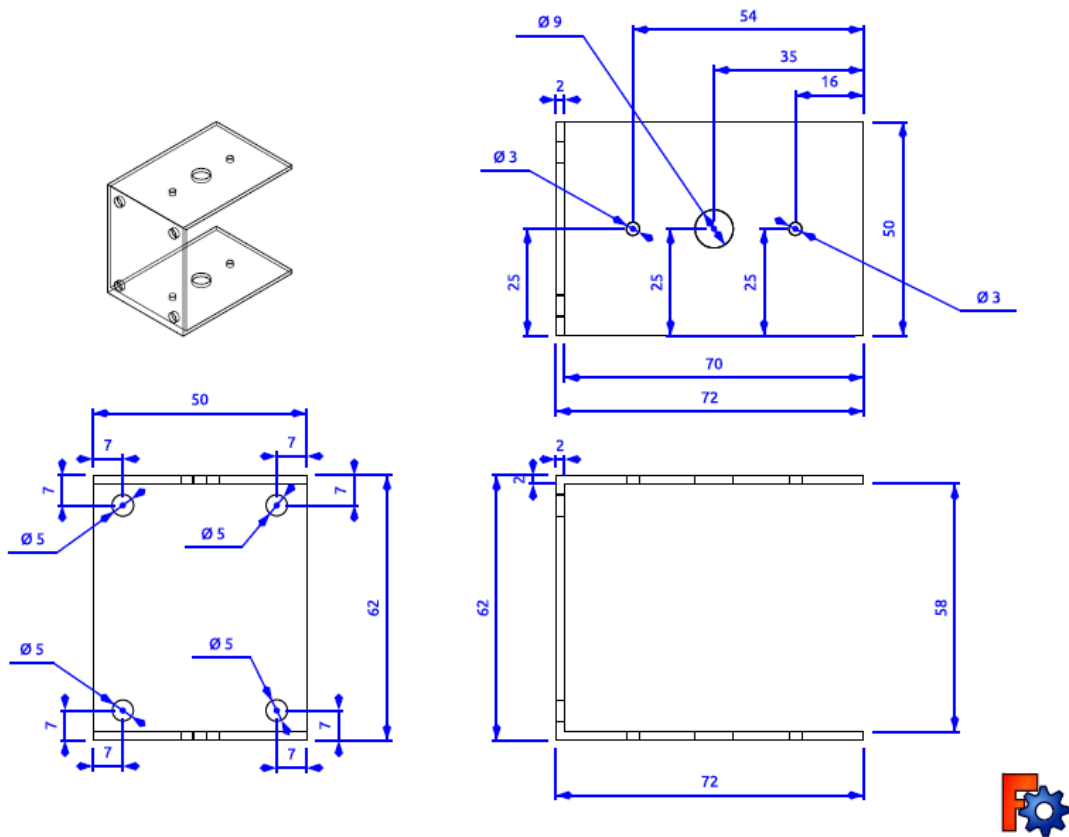
8.1.6 Elo 1



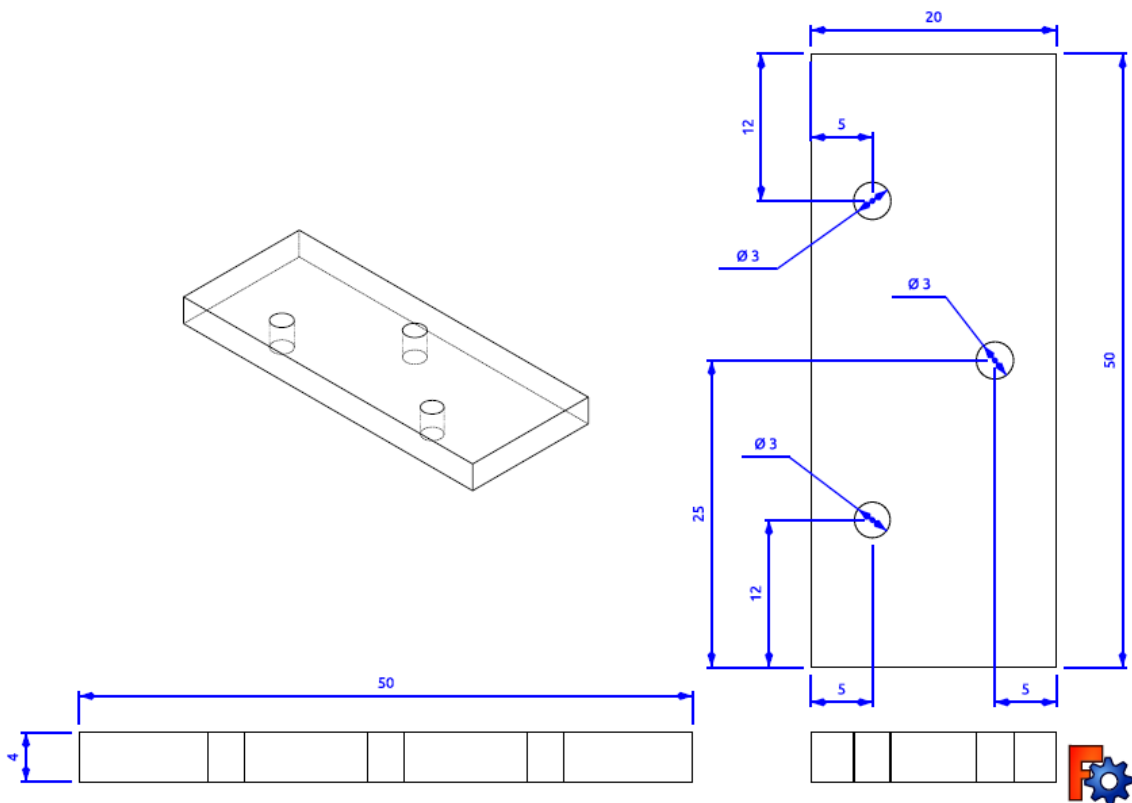
8.1.7 Elo 2



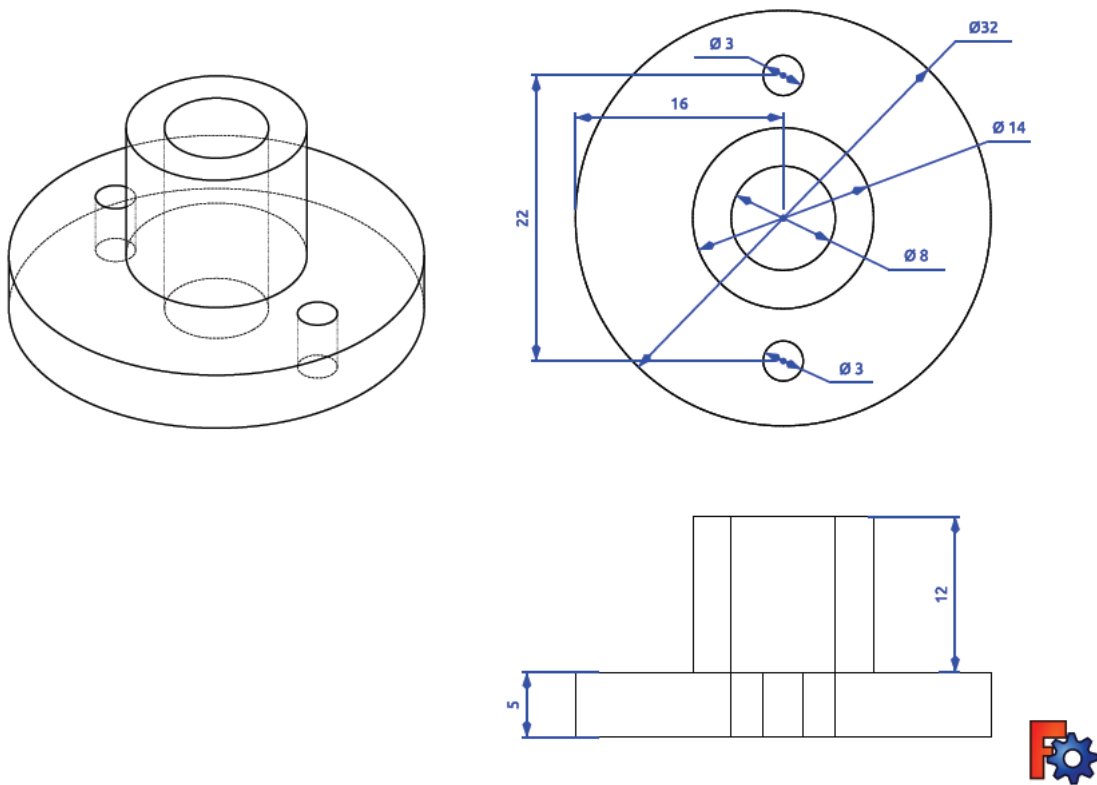
8.1.8 Engaste Elo 1



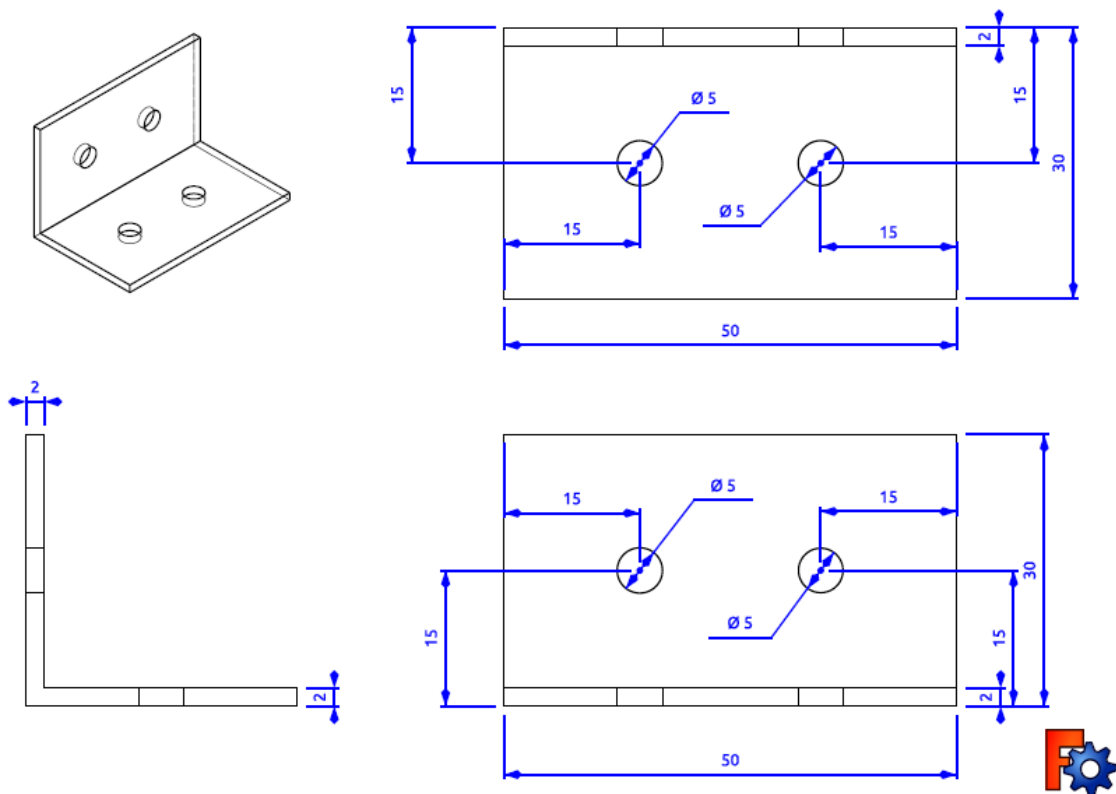
8.1.9 Espaçador



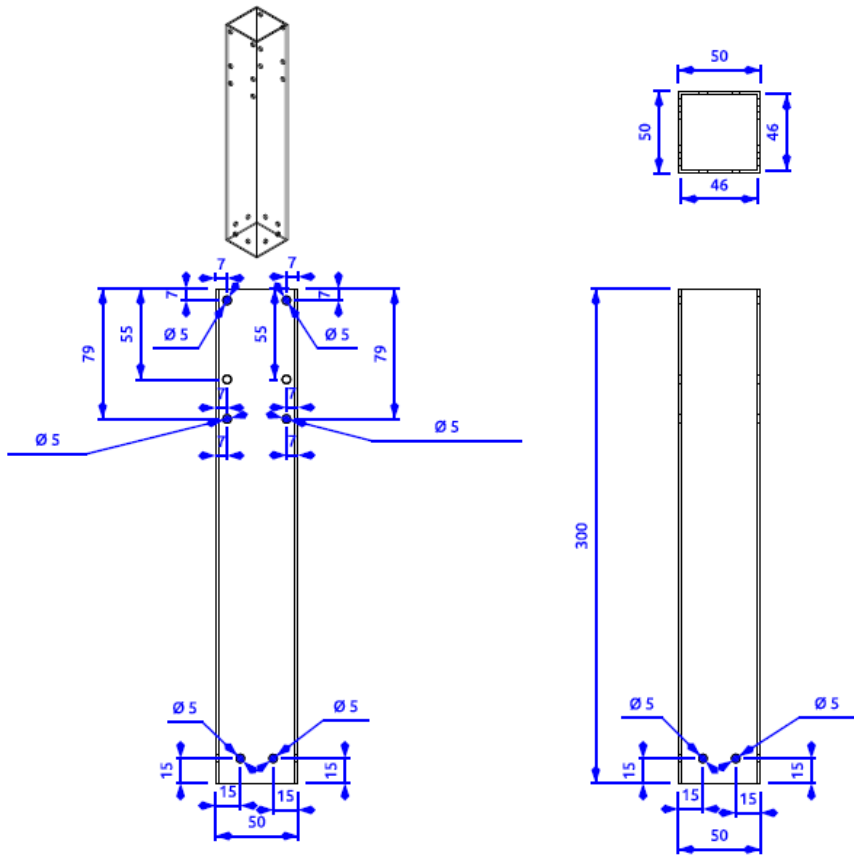
8.1.10 Fixação Eixo 1 e 2



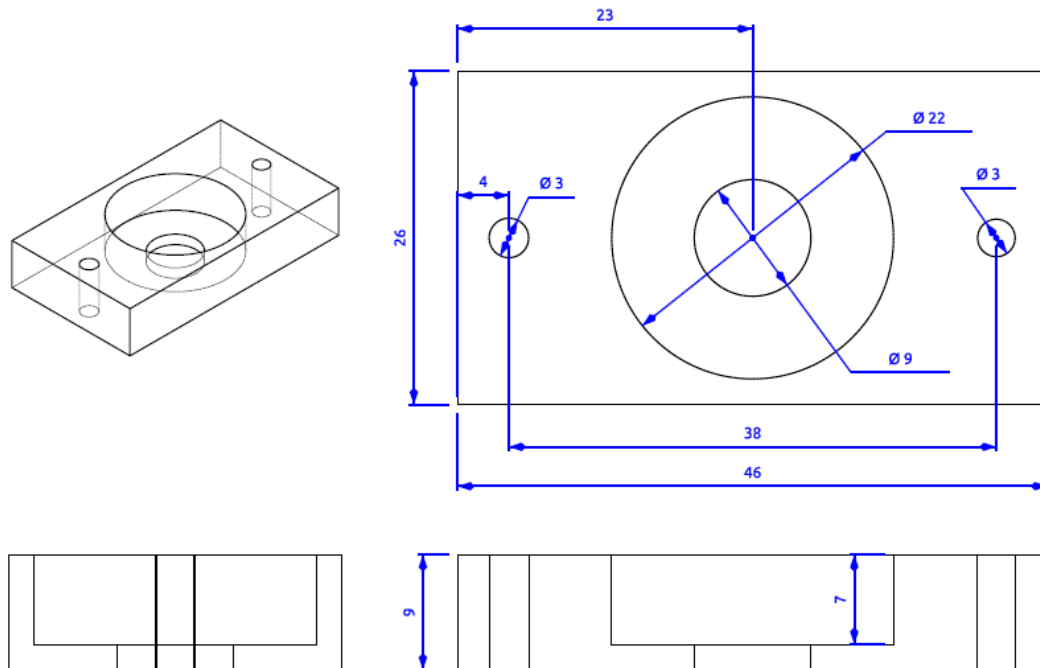
8.1.11 Fixação Haste



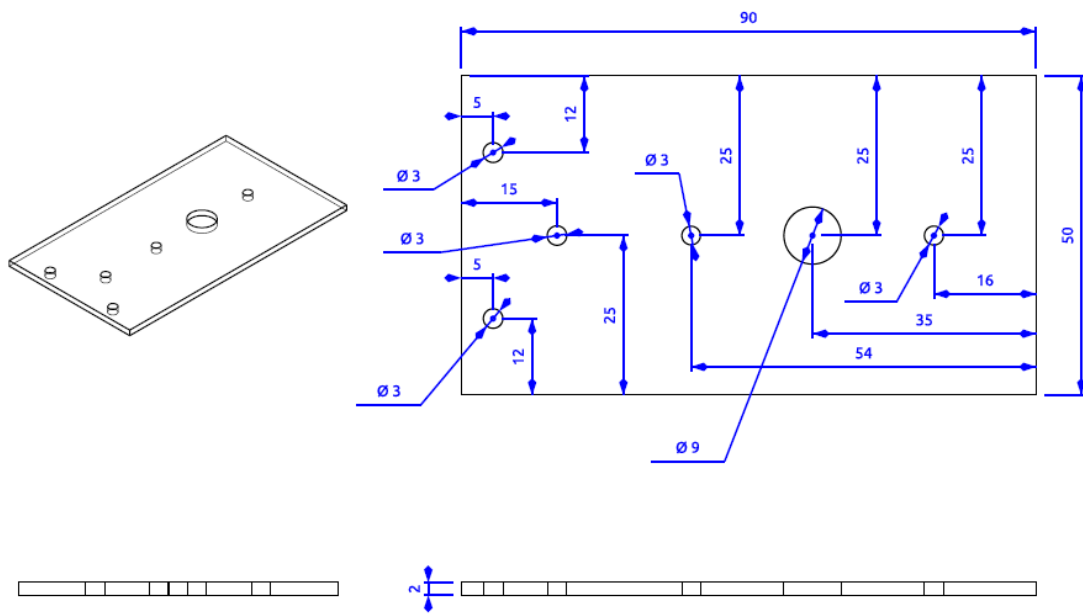
8.1.12 Haste de Sustentação



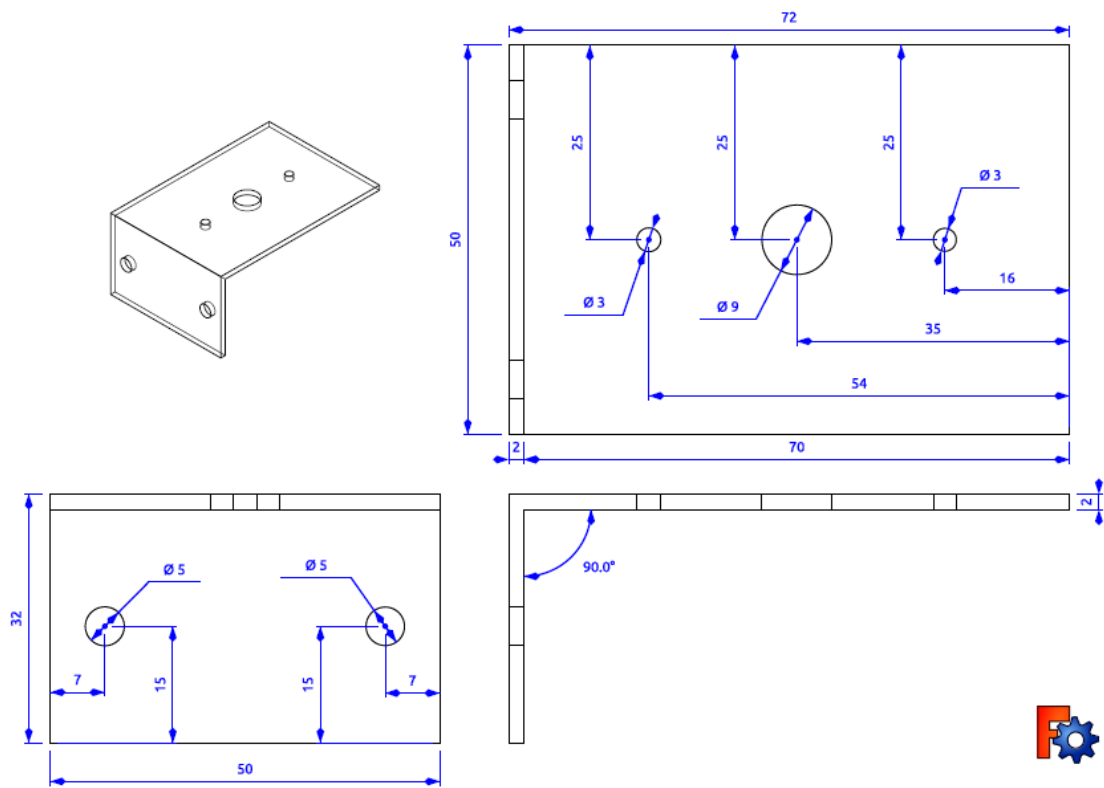
8.1.13 Mancal



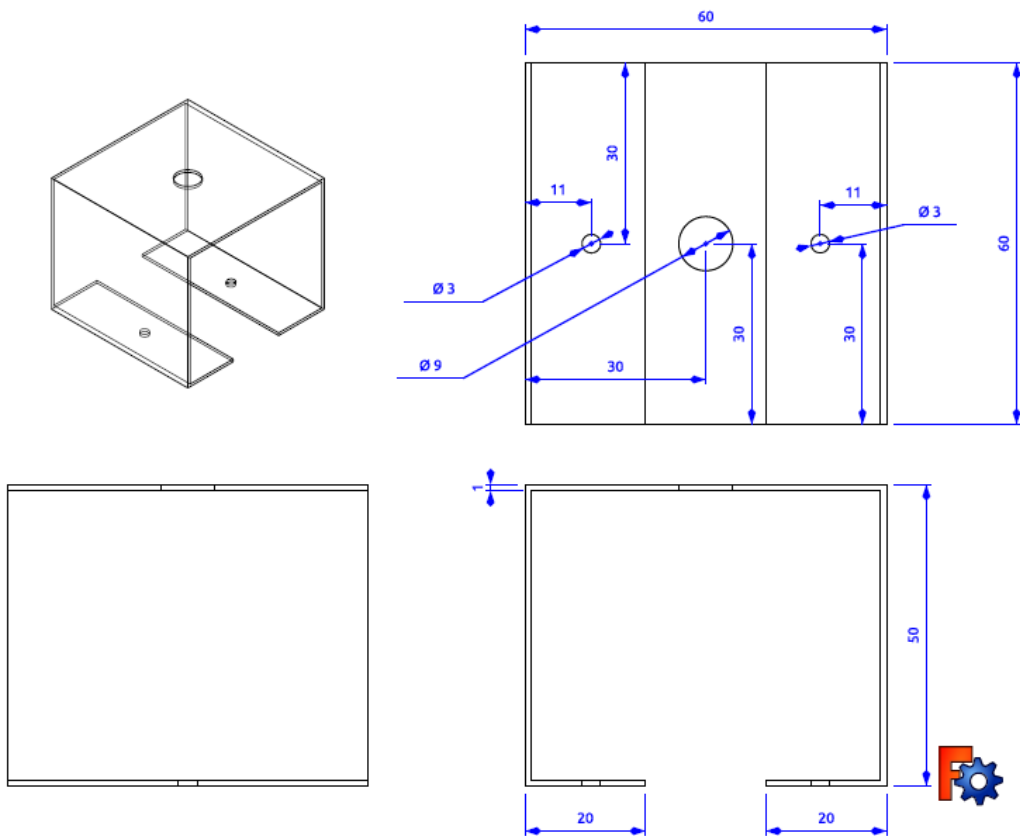
8.1.14 Orelhas



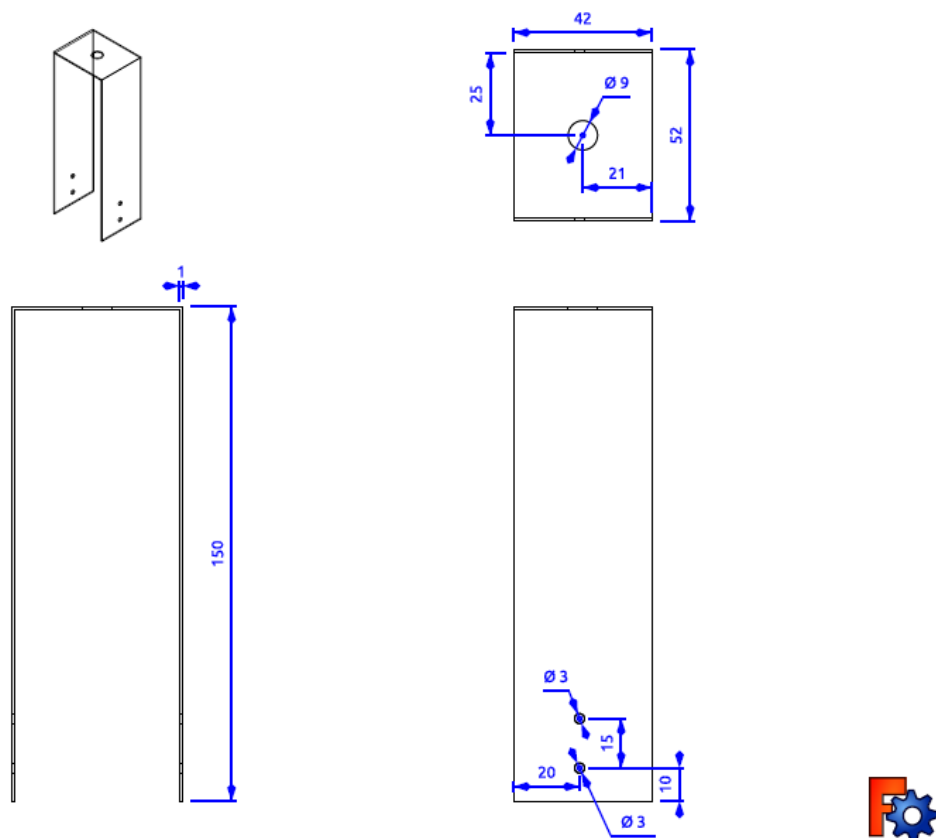
8.1.15 Suporte Engaste 1



8.1.16 Suporte Motor 1 e 2



8.1.17 Suporte Motor 3



8.2 Software de Acionamento Arduino 1

```
int IN1_1=9;
```

```
int IN2_1=10;
```

```
int IN3_1=11;
```

```
int IN4_1=12;
```

```
void setup()
```

```
{
```

```
  pinMode(IN1_1,OUTPUT);
```

```
  pinMode(IN2_1,OUTPUT);
```

```
  pinMode(IN3_1,OUTPUT);
```

```
  pinMode(IN4_1,OUTPUT);
```

```
  Serial.begin(9600);
```

```
}
```

```
void loop()
```

```
{
```

```
  if(Serial.available())
```

```
  {
```

```
    long linguica[300];
```

```
    int qnt = 0;
```

```
    while(Serial.available())
```



```

{
    linguica[qnt] = Serial.parseInt();
    qnt++;
}
qnt--;

long velocidade_1 = linguica[0];
int sentido_1;
long freqChav;

freqChav = (1.8/ velocidade_1)*10000;

int i;
long passo_1;
for(i = 1; i<= qnt; i++)
{
    if(linguica[i] < 0)
    {
        passo_1 = linguica[i]*(-1);
        sentido_1 = 1;
    }
    else
    {
        passo_1 = linguica[i];
        sentido_1 = 0;
    }
}

```

```

    movimentoPassoCompleto(&passo_1, &freqChav, &sentido_1);
}
}
}

void movimentoPassoCompleto(const long *qntPassos,const long *freqChav,const int
*sentido)
{
    int IN1, IN2, IN3, IN4;

    IN1 = IN1_1;
    IN2 = IN2_1;
    IN3 = IN3_1;
    IN4 = IN4_1;

    static int k = 0;

    for(int i = 0; i < *qntPassos; i++)
    {
        if(k == 0)
        {
            digitalWrite(IN1,HIGH);
            digitalWrite(IN2,LOW);
            digitalWrite(IN3,HIGH);
            digitalWrite(IN4,LOW);
            if(*sentido == 0)
            {

```

```
        k++;  
    }  
    else  
    {  
        k = 3;  
    }  
}  
else if(k == 1)  
{  
    digitalWrite(IN1,LOW);  
    digitalWrite(IN2,HIGH);  
    digitalWrite(IN3,HIGH);  
    digitalWrite(IN4,LOW);  
    if(*sentido == 0)  
    {  
        k++;  
    }  
    else  
    {  
        k--;  
    }  
}  
else if(k == 2)  
{  
    digitalWrite(IN1,LOW);  
    digitalWrite(IN2,HIGH);
```

```
digitalWrite(IN3,LOW);
digitalWrite(IN4,HIGH);
if(*sentido == 0)
{
    k++;
}
else
{
    k--;
}
}
else if(k == 3)
{
digitalWrite(IN1,HIGH);
digitalWrite(IN2,LOW);
digitalWrite(IN3,LOW);
digitalWrite(IN4,HIGH);
if(*sentido == 0)
{
    k = 0;
}
else
{
    k--;
}
}
```

```
        delay(*freqChav);        //tempo que determina a frequência de chaveamento.
    }

}
```

8.3 Software de Acionamento Arduino 2

```
int IN1_2=5;

int IN2_2=6;

int IN3_2=7;

int IN4_2=8;

int DIR_3 = 9;

int STEP_3 = 10;

void setup()
{

    pinMode(IN1_2,OUTPUT);
    pinMode(IN2_2,OUTPUT);
    pinMode(IN3_2,OUTPUT);
    pinMode(IN4_2,OUTPUT);
    pinMode(DIR_3,OUTPUT);
    pinMode(STEP_3,OUTPUT);

    Serial.begin(9600);

}
```

```

void loop()
{
  if(Serial.available())
  {

    long linguica[300];

    int qnt = 0;

    while(Serial.available())
    {
      linguica[qnt] = Serial.parseInt();

      qnt++;
    }
    qnt--;

    long velocidade_1 = linguica[0];

    long velocidade_2 = linguica[1];

    long passo_p = linguica[2];

    int sentido_1;

    long freqChav;

    freqChav = (1.8/ velocidade_1)*10000;

    int i;

```

```
long passo_1;
for(i = 3; i<= qnt; i++)
{
    if(linguica[i] < 0)
    {
        passo_1 = linguica[i]*(-1);
        sentido_1 = 0;
    }
    else
    {
        passo_1 = linguica[i];
        sentido_1 = 1;
    }
    movimentoPassoCompleto(&passo_1, &freqChav, &sentido_1);
}
```

```
if(passo_p < 0)
{
    sentido_1 = 1;
    passo_p = passo_p*(-1);
}
else
{
    sentido_1 = 0;
```

```

}

//Serial.println(passo_p);

//Serial.println(passo_p*15000);

movimentoPassoCompleto_3(&passo_p, &sentido_1);

delay(100);//para garantir que o motor 1 já terminou o seu deslocamento

Serial.print(1);//avisando que acabou o processamento

}

}

void movimentoPassoCompleto(const long *qntPassos,const long *freqChav,const int
*sentido)
{

int IN1, IN2, IN3, IN4;

IN1 = IN1_2;

IN2 = IN2_2;

IN3 = IN3_2;

IN4 = IN4_2;

```



```

static int k = 0;

for(int i = 0; i < *qntPassos; i++)
{
    if(k == 0)
    {
        digitalWrite(IN1,HIGH);
        digitalWrite(IN2,LOW);
        digitalWrite(IN3,HIGH);
        digitalWrite(IN4,LOW);
        if(*sentido == 0)
        {
            k++;
        }
        else
        {
            k = 3;
        }
    }
    else if(k == 1)
    {
        digitalWrite(IN1,LOW);
        digitalWrite(IN2,HIGH);
        digitalWrite(IN3,HIGH);
        digitalWrite(IN4,LOW);
    }
}

```

```
if(*sentido == 0)
{
    k++;
}
else
{
    k--;
}
}
else if(k == 2)
{
    digitalWrite(IN1,LOW);
    digitalWrite(IN2,HIGH);
    digitalWrite(IN3,LOW);
    digitalWrite(IN4,HIGH);
    if(*sentido == 0)
    {
        k++;
    }
    else
    {
        k--;
    }
}
else if(k == 3)
{
```

```

digitalWrite(IN1,HIGH);

digitalWrite(IN2,LOW);

digitalWrite(IN3,LOW);

digitalWrite(IN4,HIGH);

if(*sentido == 0)
{
    k = 0;
}
else
{
    k--;
}
}

delay(*freqChav);    //tempo que determina a frequência de chaveamento.
}

}

void movimentoPassoCompleto_3(long *qntPassos, int *sentido)
{
digitalWrite(DIR_3, *sentido);

for(long i = 0; i< *qntPassos; i++)
{
digitalWrite(STEP_3, HIGH);

delayMicroseconds(600);

digitalWrite(STEP_3, LOW);

}

}

```

8.4 Software de Controle

8.4.1 Interpolação Linear

```
#ifndef INTERPOLACAO_LINEAR
```

```
#define INTERPOLACAO_LINEAR
```

```
#include "funcoes.h"
```

```
vector<double> interpolacaoLinear(const double &x_0,const double &y_0, const double  
&x, const double &y, const char &x_ou_y)
```

```
{
```

```
    double dx = (x - x_0);
```

```
    double dy = (y - y_0);
```

```
    vecD x_vect = {};
```

```
    vecD y_vect = {};
```

```
    if(abs(dx) < 0.01)//para um dx infinito
```

```
    {
```

```
        double passoInterpolacao = abs(dy)/QNT_PONTOS_DE_INTERPOLACAO;
```

```
        if(dy > 0)
```

```
        {
```

```
            for(float i = 0; i <= abs(dy); i+=passoInterpolacao)
```

```
            {
```

```
                x_vect.push_back(x);
```

```
                y_vect.push_back(y_0 + i);
```

```
            }
```

```

}
else if(dy < 0)
{
    for(float i = 0; i <= abs(dy); i+=passoInterpolacao)
    {
        x_vect.push_back(x);
        y_vect.push_back(y_0 - i);
    }
}
else
{
    x_vect.push_back(x);
    y_vect.push_back(y);
}
}
else
{
    double m = dy/dx;
    double b = y_0 - m*x_0;

    double passoInterpolacao = abs(dx)/QNT_PONTOS_DE_INTERPOLACAO;

    for (float i = 0; i <= abs(dx); i += passoInterpolacao)
    {
        if(dx >= 0)
        {

```

```

        x_vect.push_back(x_0 + i);
        y_vect.push_back(m*(x_0 + i) + b);
    }else
    {
        x_vect.push_back(x_0 - i);
        y_vect.push_back(m*(x_0 - i) + b);
    }
}

bool retorno_x = false;
switch(x_ou_y)
{
    case 'x':
    case 'X':
        retorno_x = true;
        break;
    case 'y':
    case 'Y':
        retorno_x = false;
        break;
}

if(retorno_x)
{
    return x_vect;
}
}

```

```

    {
        return y_vect;
    }
}

```

```
#endif
```

8.4.2 Cinemática Inversa

```

#ifndef CINEMATICA_INVERSA
#define CINEMATICA_INVERSA

```

```
#include "funcoes.h"
```

```
mapIntDouble cinematicalInversa(const double &x,const double &y, const double &z)
```

```

{
    double theta_1, theta_2;

    mapIntDouble theta_1_2_z_5;

    double D;

    D = (pow(x, 2) + pow(y, 2) - pow(BRACO_1, 2) - pow(BRACO_2,
2))/(2*BRACO_1*BRACO_2);

    theta_2 = atan2(sqrt(1 - pow(D,2)),D);

```

```

if(pow(D,2) > 1)
{
    cerr << "pow(D,2) > 1 : " << pow(D,2) << endl;
}

    theta_1 = atan2(y,x) - atan2(BRACO_2*sin(theta_2), BRACO_1 +
BRACO_2*cos(theta_2));

theta_1_2_z_5.insert({1, theta_1});
theta_1_2_z_5.insert({2, theta_2});

    theta_2 = atan2(-sqrt(1 - pow(D,2)),D);
    theta_1 = atan2(y,x) - atan2(BRACO_2*sin(theta_2), BRACO_1 +
BRACO_2*cos(theta_2));

theta_1_2_z_5.insert({3, theta_1});
theta_1_2_z_5.insert({4, theta_2});

    theta_1_2_z_5.insert({5, z});

    return theta_1_2_z_5;

}

#endif

```


8.4.3 Menor Caminho

```
#ifndef MENOR_CAMINHO
```

```
#define MENOR_CAMINHO
```

```
#include "funcoes.h"
```

```
#include "volumeDeTrabalho.h"
```

```
mapIntDouble menorCaminho(const double &theta_1_0, mapIntDouble &  
theta_1_2_z_5)
```

```
{
```

```
    mapIntDouble theta_1_2_z;
```

```
    mapIntDouble theta_1 = {};
```

```
    mapIntDouble theta_2 = {};
```

```
    double z = theta_1_2_z_5[5];
```

```
    theta_1_2_z.insert({3,theta_1_2_z_5[5]});
```

```
    theta_1.insert({1, theta_1_2_z_5[1]});
```

```
    theta_1.insert({2, theta_1_2_z_5[3]});
```

```
    theta_2.insert({1, theta_1_2_z_5[2]});
```

```
    theta_2.insert({2, theta_1_2_z_5[4]});
```

```
double x_c_0 = BRACO_1*cos(theta_1_0);
```

```
double y_c_0 = BRACO_1*sin(theta_1_0);
```

```
double x_c_1 = BRACO_1*cos(theta_1[1]);
```

```
double y_c_1 = BRACO_1*sin(theta_1[1]);
```

```
double x_c_2 = BRACO_1*cos(theta_1[2]);
```

```
double y_c_2 = BRACO_1*sin(theta_1[2]);
```

```
double distancia_x_c_0_1 = sqrt(pow(x_c_1 - x_c_0, 2) + pow(y_c_1 - y_c_0, 2));
```

```
double distancia_x_c_0_2 = sqrt(pow(x_c_2 - x_c_0, 2) + pow(y_c_2 - y_c_0, 2));
```

```
if((distancia_x_c_0_1 < distancia_x_c_0_2) &&
volumeDeTrabalho(theta_1[1],theta_2[1],z))
{
    theta_1_2_z.insert({1, theta_1[1]});
    theta_1_2_z.insert({2, theta_2[1]});
}
else if(volumeDeTrabalho(theta_1[2],theta_2[2],z))
{
    theta_1_2_z.insert({1, theta_1[2]});
    theta_1_2_z.insert({2, theta_2[2]});
}
else if(volumeDeTrabalho(theta_1[1],theta_2[1],z))
```

```

{
    theta_1_2_z.insert({1, theta_1[1]});
    theta_1_2_z.insert({2, theta_2[1]});
}
else
{
    theta_1_2_z.insert({1, FORA_VOLUME_DE_TRABALHO});
    theta_1_2_z.insert({2, FORA_VOLUME_DE_TRABALHO});
    cerr << "\nFora do volume de trabalho!" << endl;
}

return theta_1_2_z;
}

```

```
#endif
```

8.4.4 Thetas para Passos

```
#ifndef THETASPARAPASSOS_H
```

```
#define THETASPARAPASSOS_H
```

```
#include "funcoes.h"
```

```
#include "conversoesDiversas.h"
```

```

vector<double> thetasParaPassos(const char &theta_1_ou_theta_2, vector<double>
&theta_1_vect, vector<double> &theta_2_vect)
{

    vector<double> theta_1_aux = theta_1_vect;
    vector<double> theta_2_aux = theta_2_vect;

    retirarThetasMenoresQueUmPasso(theta_1_aux, theta_2_aux);
    converterMovimentacaoEmDeltas(theta_1_aux, theta_2_aux);

    vector<double> passos_1 = {};
    vector<double> passos_2 = {};

    for(auto item : theta_1_aux)
    {

        passos_1.push_back(grausParaPassos(radianosParaGraus(item)));
    }
    for(auto item_2 : theta_2_aux)
    {

        passos_2.push_back(grausParaPassos(radianosParaGraus(item_2)));
    }
}

```

```

}

if(theta_1_ou_theta_2 == 1)
{
    return passos_1;
}
else
{
    return passos_2;
}
}

#endif // THETASPARAPASSOS_H

8.4.5 Definição de Velocidades

#ifndef DEFINICAODEVELOCIDADES_H
#define DEFINICAODEVELOCIDADES_H

#include "funcoes.h"

vector<double> definicaoDeVelocidades(const vecD &passos_1,const vecD
&passos_2)
{
    double desloc_1 = somaAbsolutaVetor(passos_1);

```

```

double desloc_2 = somaAbsolutaVetor(passos_2);

double velocidade_2, velocidade_1 = VELOCIDADE_1;

vecD velocidades = {};

if(desloc_1 != 0)
{
    do
    {
        velocidade_2 = (desloc_2/desloc_1)*velocidade_1;
        velocidade_1--;
    }while(velocidade_2 > VELOCIDADE_2_MAX);
    velocidade_1++;
}else
{
    velocidade_2 = VELOCIDADE_2;
}

velocidades.push_back(velocidade_1);
velocidades.push_back(velocidade_2);

return velocidades;
}

#endif // DEFINICAODEVELOCIDADES_H

```

8.4.6 Gerar Comando Serial

```
#ifndef COMANDOSERIAL_H
#define COMANDOSERIAL_H

#include "funcoes.h"

#include <QDebug>

QString gerarComandoSerialArduino_1(const double &velocidade_1, const
vector<double> &passos_1)
{
    QString linguica;

    int v_1 = round(velocidade_1);

    linguica = QString("%1 ").arg(v_1);

    for(vecD::const_iterator it = passos_1.begin(); it != passos_1.end();it++)
    {
        if(it + 1 != passos_1.end())
        {
            linguica += QString("%1 ").arg(*it);
        }
        else
        {
            linguica += QString("%1").arg(*it);
        }
    }
}
```

```
}
```

```
qDebug() << linguica;
```

```
return linguica;
```

```
}
```

```
QString gerarComandoSerialArduino_2(const double &velocidade_2, const  
vector<double> &passos_2, const double &velocidade_3, const double &passos_3)
```

```
{
```

```
    QString linguica;
```

```
    int v_2 = round(velocidade_2);
```

```
    int v_3 = round(velocidade_3);
```

```
    long passos_3_int = round(passos_3*FATOR_BRACO_3);
```

```
    linguica = QString("%1 %2 %3 ").arg(v_2).arg(v_3).arg(passos_3_int);
```

```
    for(vecD::const_iterator it = passos_2.begin(); it != passos_2.end();it++)
```

```
    {
```

```
        if(it + 1 != passos_2.end())
```

```
        {
```

```
            linguica += QString("%1 ").arg(*it);
```



```

    }
    else
    {
        linguica += QString("%1").arg(*it);
    }
}

```

```

qDebug() << linguica;

```

```

return linguica;

```

```

}

```

```

#endif // COMANDOSERIAL_H

```

8.4.7 Enviar Comando Serial

```

void Dialog::EnviarComandoArduino_1(const QString & comando)

```

```

{

```

```

    if(arduino_1->isWritable())

```

```

    {

```

```

        arduino_1->write(comando.toStdString().c_str());

```

```

    }else

```

```

    {

```

```

        QMessageBox::warning(this, "Erro Porta Serial", "Não foi possível escrever no
arduino 1!\nMotor 1 desconectado!");

```

```

    }
}

void Dialog::EnviarComandoArduino_2(const QString & comando)
{
    if(arduino_2->isWritable())
    {
        arduino_2->write(comando.toStdString().c_str());
    }else
    {
        QMessageBox::warning(this, "Erro Porta Serial", "Não foi possível escrever no
        arduino 2!\nMotores 1 e 2 desconectados!");
    }
}

```

8.4.8 Volume de Trabalho

```

#ifndef VOLUME_DE_TRABALHO
#define VOLUME_DE_TRABALHO

#include "funcoes.h"

bool volumeDeTrabalho(const double &theta_1, const double &theta_2, const double &z)
{
    if(theta_1 <= PI && theta_1 >= 0)
        if(theta_2 <= PI/2 && theta_2 >= -PI/2)

```

```
        if(z <= BRACO_3 && z >= -BRACO_3)
            return true;

        return false;
    }
```

```
#endif
```

8.4.9 Globais

```
#ifndef FUNCOES
```

```
#define FUNCOES
```

```
#include <cmath>
```

```
#include <vector>
```

```
#include <map>
```

```
#include <iostream>
```

```
#include <string>
```

```
#include <sstream>
```

```
#include <QtWidgets>
```

```
#include <QString>
```

```
#include <fstream>
```

```
#include <stdlib.h>
```

```
using namespace std;

typedef map< int, double, less<int> > mapIntDouble;
typedef vector<double> vecD;

//Constantes

double BRACO_1 = 16.8;
double BRACO_2 = 11.2;
double BRACO_3 = 9.0;
enum {FORA_VOLUME_DE_TRABALHO = 12345};

double PASSO = 1.8;//passo em graus

int FATOR_BRACO_3 = 15000;

double VELOCIDADE_1 = 80;
double VELOCIDADE_2 = 80;
double VELOCIDADE_2_MAX = 150;

double QNT_PONTOS_DE_INTERPOLACAO = pow(2,8);

double PI = acos(-1);

bool terminou_arduino_2 = false;
```

```
double Z_0 = 0;
double THETA_1_0 = PI/2 + 0.001;
double THETA_2_0 = 0;
```

```
#endif
```

8.4.10 Conversões Diversas

```
#ifndef CONVERSOES_DIVERSAS
#define CONVERSOES_DIVERSAS
```

```
#include "funcoes.h"
```

```
inline double radianosParaGraus(const double &theta)
```

```
{
    return 180*theta/PI;
}
```

```
inline double GrausParaRadianos(const double &theta)
```

```
{
    return PI*theta/180;
}
```

```
double arredondamentoMeioPasso(const double &thetaGraus)
```

```
{
    double entrada = thetaGraus/0.9;
    entrada = round(entrada);
```

```

double saida = entrada*0.9;

return saida;
}

double arredondamentoPassoCompleto(const double &thetaGraus)
{
double entrada = thetaGraus/1.8;

entrada = round(entrada);

double saida = entrada*1.8;

return saida;
}

inline double grausParaPassos(const double &thetaGraus)
{
return arredondamentoPassoCompleto(thetaGraus)/1.8;
}

void retirarThetasMenoresQueUmPasso( vecD &theta_1_vect, vecD &theta_2_vect)
{

//cout << "\nalive _ 1" << endl;

double passo = GrausParaRadianos(PASSO);

vecD theta_1_aux = {};

```

```

vecD theta_2_aux = {};

//double valor_teste_1, valor_teste_2;

theta_1_aux.push_back(theta_1_vect[0]);
theta_2_aux.push_back(theta_2_vect[0]);

vecD::iterator it_theta_1, it_theta_2;

for(it_theta_1 = theta_1_vect.begin(), it_theta_2 = theta_2_vect.begin()
    ; it_theta_1 != theta_1_vect.end(); it_theta_1++, it_theta_2++)
{

    if(it_theta_1 + 1 >= theta_1_vect.end())
    {
        break;
    }

    for(vecD::iterator it_1 = it_theta_1 + 1, it_2 = it_theta_2 + 1
        ; it_theta_1 != theta_1_vect.end(); it_1++, it_2++)
    {

        if((abs(*(it_1) - *(it_theta_1))) >= passo && (abs(*(it_2) - *(it_theta_2)) >= passo))
        {
            theta_1_aux.push_back(*(it_1));
            theta_2_aux.push_back(*(it_2));
            it_theta_1 = it_1 - 1;

```

```

        it_theta_2 = it_2 - 1;

        break;
    }
}

}

theta_1_aux.pop_back();

theta_2_aux.pop_back();

if( (abs(*(theta_1_vect.end() - 1) - *(theta_1_aux.end() - 1)) >= passo) &&
(abs(*(theta_2_vect.end() - 1) - *(theta_2_aux.end() - 1)) >= passo))
{
    theta_1_aux.push_back(*(theta_1_vect.end() - 1));
    theta_2_aux.push_back(*(theta_2_vect.end() - 1));
}
else if(theta_1_aux.size() == 1)
{
    theta_1_aux.push_back(*(theta_1_vect.end() - 1));
    theta_2_aux.push_back(*(theta_2_vect.end() - 1));
}
else
{
    theta_1_aux.pop_back();
    theta_2_aux.pop_back();
    theta_1_aux.push_back(*(theta_1_vect.end() - 1));
}

```



```
    theta_2_aux.push_back(*(theta_2_vect.end() - 1));  
}
```

```
theta_1_vect = theta_1_aux;
```

```
theta_2_vect = theta_2_aux;
```

```
}
```

```
void converterMovimentacaoEmDeltas(vecD &theta_1_vect,vecD &theta_2_vect)
```

```
{
```

```
    vecD deltas_1 = {};
```

```
    vecD deltas_2 = {};
```

```
    for(vecD::iterator it_1 = theta_1_vect.begin(), it_2 = theta_2_vect.begin();
```

```
        it_1 != theta_1_vect.end(); it_1++, it_2++)
```

```
    {
```

```
        if(it_1 + 1 != theta_1_vect.end())
```

```
        {
```

```
            deltas_1.push_back(*(it_1+1) - *it_1);
```

```
            deltas_2.push_back(*(it_2+1) - *it_2);
```

```
        }
```

```
}  
theta_1_vect = deltas_1;  
theta_2_vect = deltas_2;  
}  
  
int somaAbsolutaVetor(const vecD &vetorRef)  
{  
    int count = 0;  
    for(auto item:vetorRef)  
    {  
        count += abs(item);  
    }  
  
    return count;  
}  
  
#endif
```